The Fast Wavelet Transform (FWT)

by

Keith G. Boyer

B.S.E.E.T., DeVry Institute of Technology, 1984

A thesis submitted to the

University of Colorado at Denver

in partial fulfillment

of the requirements for the degree of

Master of Science

Applied Mathematics

1995

This thesis for the Master of Science

degree by

Keith G. Boyer

has been approved by

_____

William L. Briggs

_____

William E. Cherowitzo

_____

Weldon A. Lodwick

_____

Date

Boyer, Keith G. (M.S., Applied Mathematics)
The Fast Wavelet Transform (FWT)
Thesis directed by Professor William L. Briggs

ABSTRACT

A mathematical basis for the construction of the fast wavelet transform (FWT), based on the wavelets of Daubechies, is given. A contrast is made between the continuous wavelet transform and the discrete wavelet transform that provides the fundamental structure for the fast wavelet transform algorithm. The wavelets considered here lead to orthonormal bases. To realize the orthonormality of these bases, the Fourier transform is used to construct equivalent realizations of the wavelet recursions in the spectral domain. These spectral representations lead to a complementary pair of filter transfer functions, from which the inverse Fourier transform can be used to generate equivalent Daubechies' convolution coefficients. The discrete wavelet transform, generated from the convolution filter operations, is incorporated into a recursive filter decimation algorithm that is the FWT. Finally, the FWT is applied to the image compression problem.

This abstract accurately represents the content of the candidate's thesis.

Signed _____
William L. Briggs

Contents

Chapter

Appendix

1. Introduction

1.1 Wavelets and Signal Analysis

A signal may exist in its original form as a continuous entity. This continuous signal may then be converted into a quantized form as a sequence of sampled values that describe the signal at discrete points in time. A signal may also be generated in its original form as a discrete sequence. Images and audio signals are examples of signals that originate in a continuous form, and are stored as discrete sequences on compact disks. These signals may also be generated directly as discrete sequences.

Signal analysis is born out of the need to understand the composition of both discrete and continuous signals. This analysis usually takes the form of a sub-space decomposition into a linear combination of basis functions. The classic decomposition tool is the Fourier transform, which decomposes a signal into a linear combination of complex exponential functions (equivalently, sines and cosines).

The concept of **signal stationarity** plays a key role in determining the effectiveness of the signal decomposition obtained from the Fourier transform. The stationarity of a signal is judged by its statistical invariance over time. If the probability of a signal being a certain value is constant over time, then the signal is said to be stationary. If, however, transient statistical events occur that cannot be predicted, the signal becomes nonstationary. Due to the periodic nature of the Fourier transform, it is ideal for the decomposition and analysis of stationary signals. On the other hand, transient nonstationary signals require a basis that is more localized in both time and frequency. To see how the Fourier transform is not localized in time and frequency, consider the transformation of a square pulse in the time domain. The Fourier transform of this

pulse is a **sinc** (sin(x)/x) function in the spectral domain, a function for which the energy is clearly not localized. As for the time domain, the periodic basis functions of the Fourier transform are clearly not localized [5].

Wavelets solve this problem by having a prescribed smoothness, with energy that is well-localized in both time and frequency. In general, not only do wavelets require fewer basis functions than trigonometric basis functions, but the well-localized nature of the wavelet basis ameliorates such reconstruction anomalies as the Gibbs' phenomenon [9]. To understand the wavelet basis, we will consider a continuous transformation in $L^2(\mathbb{R})$.

1.2 The Continuous Wavelet Transform

The wavelet basis is a family of functions based on a well-localized oscillating function $\psi(t)$ of the real variable t. $\psi(t)$ is called the "mother wavelet" because all other wavelet functions within the family are generated from translations and dilations of $\psi(t)$. Actually, the **mother wavelet** $\psi(t)$ is the function with zero translation and a dilation of 1. Equation (1.1) shows the family of functions generated from ψ by translation and dilation . In (1.1), *b* is the translation variable and *a* is the dilation variable.

$$\psi_{a,b}(t) = \frac{1}{\sqrt{a}}\psi\left(\frac{t-b}{a}\right), \quad a>0, \quad b\in\mathbb{R}. \tag{1.1}$$

A simple example of a wavelet might be the **Mexican hat** function

$$\psi(x) = (1-x^2)\, e^{-\frac{1}{2}x^2}. \tag{1.2}$$

The mother wavelet of (1.2) is the solid function is the center of figure (1.1), together with two translated dilations.

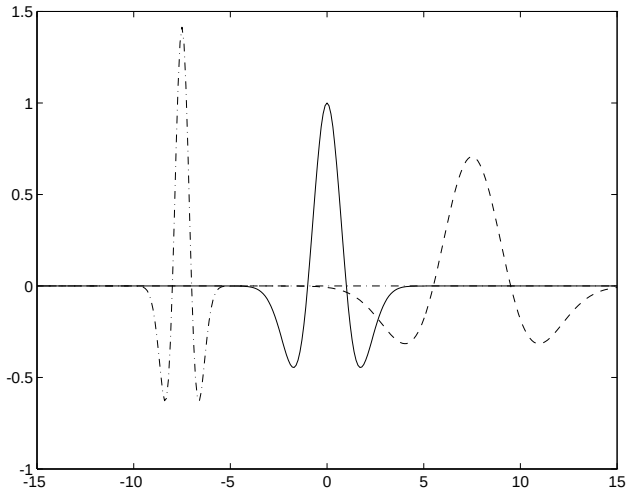

Figure 1.1: Translations and dilations of ψ(x).

A restriction on ψ(t) is that it have a zero integral. Actually, for reasons we will discover later, a further restriction on ψ(t) requires that the first m+1 moments vanish. This gives us condition (1.3), a series of integral moments equal to zero.

$$0 = \int_{-\infty}^{\infty} \psi(t)\,dt = \cdots = \int_{-\infty}^{\infty} t^m \psi(t)\,dt\,. \qquad (1.3)$$

Figure (1.2) shows plots of the integrands of (1.3) as an example of the vanishing moments of ψ(x) in (1.2). In this case, if we consider the integral of the functions in figure (1.2), moments 0 and 1 vanish, while moment 2 does not.

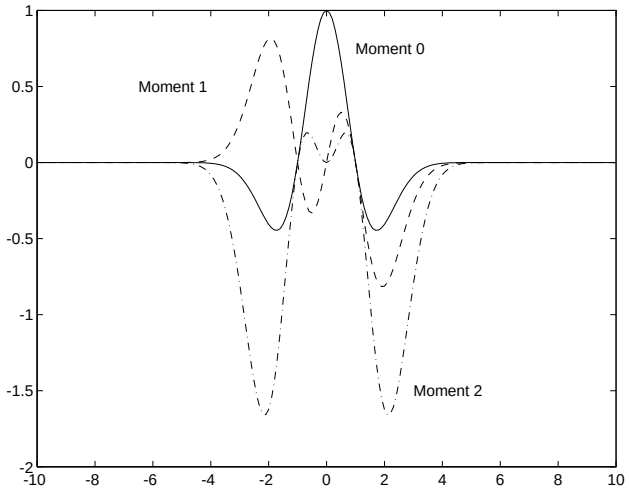

Figure 1.2: Moments of (1.2)

With a mother wavelet ψ meeting the restriction of (1.3), a corresponding set of functions $\{\psi_{ab}\}$ in the form of (1.1) can be considered. The translation and dilation parameters a,b will range over a continuous set S, such that the set of functions $\{\psi_{ab}\}$ has sufficient cardinality to allow any function f in $L^2(\mathbb{R})$ to be reconstructed from the coefficients of the wavelet transform defined by the inner product of f and ψ_{ab}, $\langle f,\psi_{ab}\rangle$. The inner product of the vector space $L^2(\mathbb{R})$ of square-integrable, measurable functions is defined by

$$\langle f(x), g(x)\rangle = \int_{-\infty}^{\infty} f(x)\,\overline{g(x)}\,dx \quad .$$

The inner product can be used to define the **continuous wavelet transform** F(a,b) of $f \in L^2(\mathbb{R})$ as

$$F(a,b) = \langle f(x), \psi_{a,b}(x)\rangle = \int_{-\infty}^{\infty} f(x)\,\overline{\psi_{a,b}(x)}\,dx \quad ,$$

for $f,\psi \in L^2(\mathbb{R})$.

This is the **analysis** of the function or signal that is the motivation for the construction of the wavelet family. The **synthesis** of $f(x)$ from F involves a linear combination of the original wavelets using the coefficients of F(a,b). The synthesis or inversion formula is given by

$$f(x) = \int_0^\infty \int_{-\infty}^\infty F(a,b)\psi_{a,b}(x)\frac{db\ da}{a^2} \quad .$$

These formulas, from a historical perspective, are interesting in that they show the basic analysis involved in the early development of wavelets. However, continuous analysis is usually impractical, which gave rise to discrete solutions of the wavelet analysis problem. For an extensive historical perspective, reference [5] is very complete.

1.3 Discretization of the Continuous Wavelet Transform

For the same reasons that we discretize differential equations, we will consider a discretization of the continuous wavelet transform in the (a,b) plane. This discretization will allow for numerical solutions based on a summation rather than a continuous integral.

Obviously, many possible discretizations of the continuous (a,b) plane exist. However, only certain restrictions on (a,b) will give the desired basis functions $\{\psi_{ab}\}$. One classic choice for restricting (a,b) is a binary discretization that leads to the wavelets of Daubechies [2]. This binary discretization sets $a=2^{-j}$ and $b=ak$, where $j,k \in \mathbf{Z}$. The discretization of $F(a,b)$ then becomes $F(2^{-j},2^{-j}k)$, $j,k \in \mathbf{Z}$, where the corresponding discrete wavelet functions are defined by

$$\psi_{jk}(x) = 2^{j/2}\psi(2^j x - k), \quad j,k \in \mathbb{Z}. \qquad (1.4)$$

Equation (1.4) is one of three formal definitions of a wavelet [5]. To introduce further wavelet definitions, we must first define the Fourier transform. The Fourier transform of the function f in $L^2(\mathbb{R})$ is

$$\hat{f}(\xi) = \int_{-\infty}^{\infty} f(t)\, e^{-i\xi t} dt, \quad \xi \in \mathbb{R} .$$

The inverse Fourier transform is the expression

$$f(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} \hat{f}(\xi)\, e^{i\xi t} d\xi, \quad t \in \mathbb{R} .$$

With the Fourier transform pair defined, we can consider the formal wavelet definitions.

(1) A wavelet is a function $\psi(t)$ in $L^2(\mathbb{R})$ whose Fourier transform $\hat{\psi}(\xi)$ satisfies the condition $\int_0^{\infty} |\hat{\psi}(t\xi)|^2 \frac{dt}{t} = 1$ almost everywhere, (continuous sense).

(2) A wavelet is a function $\psi(t)$ in $L^2(\mathbb{R})$ whose Fourier transform $\hat{\psi}(\xi)$ satisfies the condition $\sum_{-\infty}^{\infty} |\hat{\psi}(2^j\xi)|^2 = 1$ almost everywhere, (discrete sense).

(3) A wavelet is a function $\psi(t)$ in $L^2(\mathbb{R})$ such that $2^{j/2}\psi(2^j x - k)$, $j,k \in Z$ is an orthonormal basis for $L^2(\mathbb{R})$.

From definitions (2) and (3) above, we can see that each subspace is defined by an integer power of 2. These integer subspaces form a nested sequence of subspaces of $L^2(\mathbb{R})$ known as a **multiresolution analysis** (approximation), which is the subject of the next chapter.

2. Multiresolution Approximations

The construction of the fast wavelet transform (FWT) begins by splitting $L^2(\mathbb{R})$ into a sequence $(V_j), j \in Z$, of closed subspaces, each of which is spanned by an orthonormal basis of translates of a single function ϕ,

$$\phi_{j,k}(x) = 2^{j/2}\phi(2^j x - k), \quad k \in Z, \tag{2.1}$$

such that the following properties hold [14].

$$V_j \subset V_{j+1} \quad \forall\, j \in Z$$

$$\bigcup_{j=-\infty}^{+\infty} V_j \text{ is dense in } L^2(\mathbb{R}) \;\&\; \bigcap_{j=-\infty}^{+\infty} V_j = \{\varnothing\}$$

$$f(x) \in V_j \Leftrightarrow f(2x) \in V_{j+1} \quad \forall\, j \in Z$$

$$V_0 \text{ has an orthogonal basis of translates } \phi(x-k), \forall\, k \in Z$$

Given the properties for the basis of V_j above, we can define an approximation of a function $f(x) \in L^2(\mathbb{R})$, at the resolution 2^j, as the orthogonal projection of $f(x)$ onto V_j. This is an orthogonal projection because each basis function involved in the approximation of $f(x)$ is orthogonal to every other. The construction of the orthogonal projection above begins by finding a unique function $\phi(x) \in L^2(\mathbb{R})$.

A **wavelet basis**, related to $\phi(x)$, then evolves from the additional information of an approximation at a resolution 2^{j+1} compared with the resolution 2^j. This additional information is just the projection of the orthogonal complement of V_j in V_{j+1}. That is, $V_{j+1} = V_j \oplus W_j$, where W_j is the space defined by the projection of the orthogonal complement of V_j in V_{j+1}. The space W_j will be

spanned by the wavelet orthonormal basis defined by

$$\psi_{j,k}(x) = 2^{j/2}\psi(2^j x-k)\,, \quad k \in Z\,. \tag{2.2}$$

We can summarize the relationship of these subspaces at every level by

$$\ldots \subset V_{-2} \subset V_{-1} \subset V_0 \subset V_1 \subset V_2 \subset \ldots \quad \textit{and}$$
$$V_{j+1} = V_j \oplus W_j = V_0 \oplus W_0 \oplus W_1 \oplus \ldots \oplus W_j \ .$$

To understand the projection of f onto any of the particular subspaces, let's consider the simple Haar basis defined by

$$\phi(x) = \begin{cases} 1, & \textit{if } 0 \le x < 1, \\ 0, & \textit{otherwise} \end{cases} \ .$$

$\phi(x)$ is known as the **scaling function** or **father wave.** $\phi(x)$ is a solution to the **dilation equation**

$$\phi(x) = \sum_n c_n \phi(2x-n), \quad \sum_n c_n = 2 \ ,$$

for a finite number of non zero coefficients (For the Haar basis, $c_0 = c_1 = 1$). The corresponding **wavelet** or **mother wave** is defined by

$$\psi(x) = \sum_n (-1)^n c_{1-n} \phi(2x-n) \ .$$

For the same finite set of coefficients $\{c_n\}$, the corresponding Haar wavelet is $\psi(x)=\phi(2x)-\phi(2x-1)$, which is given explicitly by

$$\psi(x) = \begin{cases} 1, & \text{if } 0 \le x < \frac{1}{2}, \\ -1, & \text{if } \frac{1}{2} \le x < 1, \\ 0, & \text{otherwise} \end{cases} .$$

Figure (2.1) shows the Haar scaling function $\phi(x)$ and corresponding wavelet $\psi(x)$. This is one of the only wavelet functions that can be described explicitly. The wavelets that we will see throughout the rest of this work are recursively defined. As such, they cannot be viewed in the usual way. In later chapters we will see how to view these recursive functions.

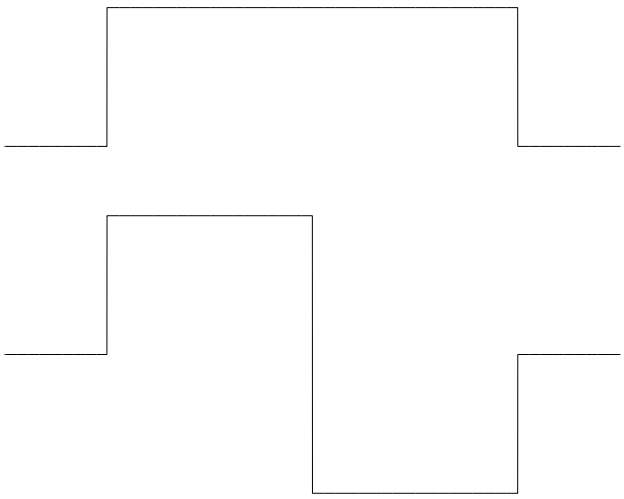

Figure 2.1: The Haar scaling function ϕ (top) and the Haar wavelet ψ (bottom).

If we consider the projection of a function f onto the Haar basis at two resolutions, we would get figure (2.2).

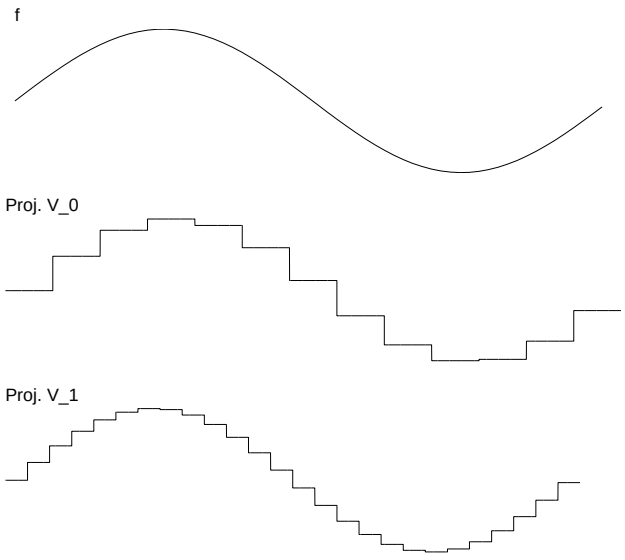

Figure 2.2: Some f(x) and its projection onto V_0 and V_1, spanned by two resolutions of the Haar basis.

In figure (2.2), f is projected by the scaling function $\phi(x)$ onto V_0. But we know that $V_0 \subset V_1$; that is, V_1 is a higher resolution than V_0. This explains why the projection of f onto V_1 is more accurate than the projection onto V_0. It is also important to realize that since $V_0 \subset V_1$, the projection of f onto V_0 can exist in V_1, but the projection of f onto V_1 cannot exist in V_0.

We briefly introduced the scaling function and restrictions on the coefficients $\{c_n\}$ in the above example. We will now develop restrictions on the wavelet and scaling function coefficients more formally.

2.1 Wavelet and Scaling Function Coefficients

With $V_j \subset V_{j+1}$ and $\phi(x) \in V_j \iff \phi(2x) \in V_{j+1}$, it follows that for some set of coefficients $\{c_n\}$, $\phi(x)$ can be written as a linear combination of $\phi(2x)$,

$$\phi(x) = \sum_{n=-\infty}^{\infty} c_n \, \phi(2x-n) , \quad x \in \mathbb{R} . \tag{2.3}$$

Alternatively, if we consider equation (2.3) written as a linear combination of $\phi(x)$ in the scaled form defined in equation (2.1), the recursion for $\phi(x)$ can be written in terms of a new set of coefficients $\{h_n\}$ as

$$\phi(x) = \sqrt{2} \sum_{n=-\infty}^{\infty} h_n \, \phi(2x-n) , \quad x \in \mathbb{R} , \tag{2.4}$$

so that

$$\sum_{n=-\infty}^{\infty} c_n = \sqrt{2} \sum_{n=-\infty}^{\infty} h_n \quad .$$

To normalize $\phi(x)$ we require $\int_{-\infty}^{\infty} \phi(x)\, dx = 1$, from which

$$1 = \int_{-\infty}^{\infty} \phi(x)\, dx = \sqrt{2} \sum_{n=-\infty}^{\infty} h_n \int_{-\infty}^{\infty} \phi(2x-n)\, dx$$

$$= \frac{1}{2} \sqrt{2} \sum_{n=-\infty}^{\infty} h_n \int_{-\infty}^{\infty} \phi(2x-n)\, d(2x-n)$$

$$= \frac{1}{\sqrt{2}} \sum_{n=-\infty}^{\infty} h_n \quad \rightarrow \quad \sum_{n=-\infty}^{\infty} h_n = \sqrt{2}.$$

This gives us a requirement on what will become the set of **filter coefficients** $\{h_n\}$. In a later derivation, this requirement will become an integral part of the normalization of the filter transfer functions that define the multiresolution analysis.

Now that we have defined $\phi(x)$, $\psi(x)$ can be expressed so that it is orthogonal to $\phi(x)$, namely

$$\psi(x) = \sqrt{2} \sum_{n=-\infty}^{\infty} g_n \, \phi(2x-n) , \quad x \in \mathbb{R} . \tag{2.5}$$

To make $\psi(x)$ orthogonal to $\phi(x)$, we require $g_n = (-1)^n \overline{h_{1-n}}$ so that the coefficient set $\{g_n\}$ is orthogonal to $\{h_n\}$, i.e., $h^t g = 0$.

From equation (1.3) of the introduction, we required that the zero moment or the mean of $\psi(x)$ equal zero (vanish). From this condition, we can derive the following requirement for the filter coefficients $\{g_n\}$.

$$0 = \int_{-\infty}^{\infty} \psi(x)\,dx = \sqrt{2} \sum_{n=-\infty}^{\infty} g_n \int_{-\infty}^{\infty} \phi(2x-n)\,dx$$

$$= \frac{1}{2}\sqrt{2} \sum_{n=-\infty}^{\infty} g_n \int_{-\infty}^{\infty} \phi(2x-n)\,d(2x-n)$$

$$= \frac{1}{\sqrt{2}} \sum_{n=-\infty}^{\infty} g_n \quad \rightarrow \quad \sum_{n=-\infty}^{\infty} g_n = 0.$$

2.2 Orthonormality of Compactly Supported Wavelets

At this point, we have defined two basis functions: $\phi(x)$ and $\psi(x)$. The problem now becomes how to pick the coefficients $\{h_n\}$ and $\{g_n\}$ to ensure that the functions $\phi(x)$ and $\psi(x)$ form orthonormal bases. To answer this question, we will turn to the Fourier transform. The spectral representation of these functions greatly simplifies the analysis of orthonormality.

To construct the spectral representation of these basis functions, we start with the multiresolution analysis defined by the filter coefficients $\{h_n\}$,

$$\phi(x) = \sqrt{2} \sum_{n=-\infty}^{\infty} h_n \, \phi(2x-n) .$$

Using the following properties of the Fourier transform,

$$f(at) \leftrightarrow \frac{1}{|a|} F\left(\frac{\xi}{a}\right) \quad (scaling)$$

$$f(t-t_0) \leftrightarrow e^{-i\xi t_0} F(\xi) \quad (delay) \quad ,$$

with a=2 and t_0=n, the Fourier Transform of $\phi(x)$ is

$$\hat{\phi}(\xi) = \frac{1}{\sqrt{2}} \sum_{n=-\infty}^{\infty} h_n e^{-i n \xi/2} \hat{\phi}(\xi/2) \quad .$$

As such, we can define a transfer function $m_0(\xi)$ so that

$$\hat{\phi}(\xi) = m_0(\xi/2) \hat{\phi}(\xi/2) \quad , \qquad (2.6)$$

where

$$m_0(\xi) = \frac{1}{\sqrt{2}} \sum_{n=-\infty}^{\infty} h_n e^{-in\xi} \quad . \qquad (2.7)$$

The transfer function $m_0(\xi)$ is 2π-periodic in $L^2([0,2\pi])$. As we will see later, the construction of $m_0(\xi)$ will yield a low pass filter corresponding to the scaling function $\phi(x)$, and a complementary high pass filter $m_1(\xi)$ corresponding to $\psi(x)$. This low pass/high pass relationship makes sense because $\psi(x)$ evolved from the additional information of an approximation at a resolution 2^{j+1} compared with the resolution 2^j. The additional information can be looked at as a high pass operation, namely the change between two resolutions. As a function of time, $\phi(x)$ and $\psi(x)$ will provide analysis through convolution operations. Because convolution in the time domain is equivalent to multiplication in the spectral domain, the convolution operations will be equivalent to multiplication by $m_0(\xi)$ and a complementary

filter $m_1(\xi)$, which will be developed later. Figure (2.3) shows the relationship of the complementary high pass/low pass 2π-periodic filters $m_0(\xi)$ and $m_1(\xi)$.

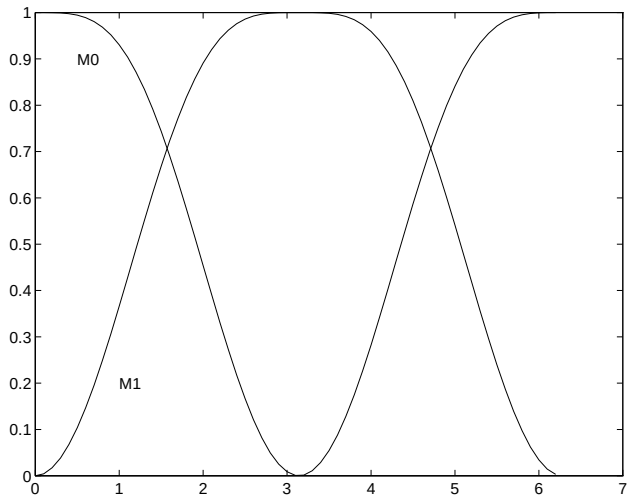

Figure 2.3: Low pass filter $m_0(\xi)$ and complementary high pass filter $m_1(\xi)$, in the frequency domain.

Finally, what are the necessary and sufficient conditions that will ensure that $\{\phi_k\}$ forms an orthonormal basis? To begin this construction, consider the definition of orthogonality in $L^2(\mathbb{R})$. If the scaling function $\phi(x)$ forms an orthonormal basis, then

$$\int_{-\infty}^{\infty} \phi(x)\overline{\phi(x-k)}\,dx = 0 \quad a.e.\ ,\ (k\neq 0)\ ,$$

or equivalently

$$\int_{-\infty}^{\infty} \phi(x)\overline{\phi(x-k)}\,dx = \delta_{k,0} \quad . \tag{2.8}$$

To take the Fourier transform of (2.8), we turn to Parseval's theorem.

Theorem 2.1 (Parseval):

$$\int_{-\infty}^{\infty} x(t)\,y^{*}(t)\,dt = \int_{-\infty}^{\infty} X(f)\,Y^{*}(f)\,df \quad ,$$

where X(f) and Y(f) are the Fourier transform of x(t) and y(t), respectively.

By this theorem, the integrand becomes $\hat{\phi}(\xi)\hat{\phi}^{*}(\xi) = |\hat{\phi}(\xi)|^{2}$. By the delay property of the Fourier transform, the time shift of (-k) results in a phase shift term $e^{ik\xi}$. The Fourier transform of the right side is 1, by the property that the Fourier transform of the delta function equals 1. Consequently, taking the Fourier transform of (2.8) yields

$$\int_{-\infty}^{\infty} |\hat{\phi}(\xi)|^{2} e^{ik\xi}\,d\xi = 1$$

$$or \quad \frac{1}{2\pi}\int_{0}^{2\pi} \sum_{l=-\infty}^{\infty} |\hat{\phi}(\xi+2\pi l)|^{2} e^{ik\xi}\,d\xi = 1 \quad .$$

It follows that

$$\sum_{l=-\infty}^{\infty} |\hat{\phi}(\xi+2\pi l)|^{2} \int_{0}^{2\pi} e^{ik\xi}\,d\xi = 2\pi \quad .$$

Evaluating the integral, we see that

$$\sum_{l=-\infty}^{\infty} |\hat{\phi}(\xi+2\pi l)|^{2} = 1 \quad .$$

Now, with reference to (2.6)

$$\hat{\phi}(2\xi) = m_0(\xi)\,\hat{\phi}(\xi) \quad ,$$

and

$$1 = \sum_{k=-\infty}^{\infty} \left|\hat{\phi}(2\xi+2\pi k)\right|^2 = \sum_{k=-\infty}^{\infty} \left|m_0(\xi+k\pi)\right|^2 \left|\hat{\phi}(\xi+k\pi)\right|^2 \quad .$$

Since m_0 is 2π-periodic, we can split the sum into odd and even k, which gives us

$$1 = \left|m_0(\xi)\right|^2 \sum_{k=-\infty}^{\infty} \left|\hat{\phi}(\xi+2k\pi)\right|^2 + \left|m_0(\xi+\pi)\right|^2 \sum_{k=-\infty}^{\infty} \left|\hat{\phi}(\xi+\pi+2k\pi)\right|^2 \quad .$$

Again, because

$$\sum_k \left|\hat{\phi}(\xi+2\pi k)\right|^2 = 1 \quad ,$$

we have

$$\left|m_0(\xi)\right|^2 + \left|m_0(\xi+\pi)\right|^2 = 1 \quad , \qquad (2.9)$$

a necessary condition on $m_0(\xi)$ given that $\phi_{o,n}$ is orthonormal.

2.3 Bi-Orthogonal Decomposition

As stated earlier, the space W_j spanned by ψ_j evolved from the additional information of the approximation at the resolution 2^{j+1} compared with the resolution 2^j. In terms of the spaces spanned by ϕ_j and ψ_j, this becomes $W_j \oplus V_j = V_{j+1}$ or $V_j = V_{j-1} \oplus W_{j-1}$. This implies that any function $f(x) \in V_j$ can be decomposed in terms of the bi-orthogonal spaces at resolution (j-1). With respect to the Fourier transform of f(x), this decomposition can be stated as follows.

Since ϕ_j forms a basis, we can express f(x) as a linear combination of ϕ,

$$f(x) = \sum_{k=-\infty}^{\infty} a_k \, \phi(x-k) \ .$$

The Fourier transform of f(x) is

$$\hat{f}(\xi) = a(\xi)\hat{\phi}(\xi) \ ,$$

where

$$a(\xi) = \sum_{k=-\infty}^{\infty} a_k \, e^{-ik\xi}$$

is a 2π-periodic function. For the purpose of this decomposition we define

$$H(\xi) = \frac{1}{\sqrt{2}} \sum_n h_n e^{-in\xi} \quad \textit{and}$$

$$G(\xi) = \frac{1}{\sqrt{2}} \sum_n g_n e^{-in\xi} \ ,$$

for the coefficients $\{h_n\}$ and $\{g_n\}$ defined previously (We use $H(\xi)$ instead of $m_0(\xi)$ here for ease of discussion). If we recall that $V_0 = V_{-1} \oplus W_{-1}$, we can decompose $a(\xi)\hat{\phi}(\xi)$ into the next lower resolution for some π-periodic functions $b(\xi)$ and $c(\xi)$ as

$$\hat{f}(\xi) = a(\xi)\hat{\phi}(\xi) = b(\xi)\hat{\phi}(2\,\xi) + c(\xi)\hat{\psi}(2\,\xi) \ .$$

Substituting

$$\hat{\phi}(2\,\xi) = H(\xi)\hat{\phi}(\xi) \quad \textit{and}$$

$$\hat{\psi}(2\,\xi) = G(\xi)\hat{\phi}(\xi) \ ,$$

we get

$$a(\xi) = b(\xi)H(\xi) + c(\xi)G(\xi) \quad .$$

From this decomposition, a paraunitary linear system can be constructed, through which a necessary condition for the operator of this linear system can be realized.

From $a(\xi) = b(\xi)H(\xi) + c(\xi)G(\xi)$ the following linear system can be constructed,

$$\begin{bmatrix} a(\xi) \\ a(\xi+\pi) \end{bmatrix} = \begin{bmatrix} H(\xi) & G(\xi) \\ H(\xi+\pi) & G(\xi+\pi) \end{bmatrix} \begin{bmatrix} b(\xi) \\ c(\xi) \end{bmatrix} .$$

Here we need to show that

$$U = \begin{bmatrix} H(\xi) & G(\xi) \\ H(\xi+\pi) & G(\xi+\pi) \end{bmatrix}$$

is unitary ($U^*U = UU^* = I$), so that

$$\begin{bmatrix} b(\xi) \\ c(\xi) \end{bmatrix} = \begin{bmatrix} H^*(\xi) & H^*(\xi+\pi) \\ G^*(\xi) & G^*(\xi+\pi) \end{bmatrix} \begin{bmatrix} a(\xi) \\ a(\xi+\pi) \end{bmatrix} .$$

This is a necessary condition so that if U is the transform operator for the fast wavelet transform, U^{-1} is just U^*. If U is unitary, then

$$U^*U = \begin{bmatrix} H^*(\xi)H(\xi) + H^*(\xi+\pi)H(\xi+\pi) & H^*(\xi)G(\xi) + H^*(\xi+\pi)G(\xi+\pi) \\ G^*(\xi)H(\xi) + G^*(\xi+\pi)H(\xi+\pi) & G^*(\xi)G(\xi) + G^*(\xi+\pi)G(\xi+\pi) \end{bmatrix} \text{ and}$$

$$UU^* = \begin{bmatrix} H(\xi)H^*(\xi) + G(\xi)G^*(\xi) & H(\xi)H^*(\xi+\pi) + G(\xi)G^*(\xi+\pi) \\ H(\xi+\pi)H^*(\xi) + G(\xi+\pi)G^*(\xi) & H(\xi+\pi)H^*(\xi+\pi) + G(\xi+\pi)G^*(\xi+\pi) \end{bmatrix} .$$

Consequently, the system is unitary if

$$H(\xi)H^*(\xi) + G(\xi)G^*(\xi) = |H(\xi)|^2 + |G(\xi)|^2 = 1$$

and

$$H^*(\xi)G(\xi) + H^*(\xi+\pi)G(\xi+\pi) = 0 \quad .$$

Actually, all the diagonal elements need to be 1 and the off diagonal elements need to be 0, so that U is unitary. However, only the elements listed are necessary due to the 2π-periodicity and symmetry of H(ξ) and G(ξ).

2.4 z-Transform and Filter Coefficients

At this point, from equation (2.7), we can define H(ξ) in terms of $m_0(\xi)$ as

$$H(\xi) = \sqrt{2}\, m_0(\xi) \quad .$$

That is, H(ξ) is the Fourier transform of the filter coefficients $\{h_n\}$. Similarly, G(ξ) is defined as the Fourier transform of the filter coefficients $\{g_n\}$. Therefore, we can look to the definition of the filter coefficients $\{g_n\}$ provided in (2.5), where, because ψ(x) is defined as the orthogonal to ϕ(x), $g_n = (-1)^n h^*_{1-n}$. From this definition, we can use the z-transform to define a suitable solution for G(ξ).

First consider that ψ(x) is, in reality, defined in terms of a finite sequence, so that equation (2.5) becomes

$$\psi(x) = \sqrt{2}\sum_{k=0}^{2N-1} g_k \phi(2x-k) \quad x \in \mathbb{R} \ . \qquad (2.10)$$

From this, we can re-define $\{g_k\}$, as a re-indexed finite sequence, as $g_k = (-1)^k h_{2N-1-k}$, $k=0,1,2,\ldots,2N-1$ (here $h^*=h$, since h is real). Now let $z = e^{i\xi}$, and define the following z-transform properties (here $\leftrightarrow$ denotes the transform, $Z\{f_k\} = F$).

$$(1) \quad f_{k\pm k_0} \leftrightarrow z^{\pm k_0} F(z)$$

$$(2) \quad a^k f_k \leftrightarrow F\left(\frac{z}{a}\right)$$

$$(3) \quad f_{-k} \leftrightarrow F(z^{-1}) \ .$$

From these properties, it follows that

$$g_k = (-1)^k h_{2N-1-k}$$

$$= (-1)^k h_{-(k-(2N-1))} \leftrightarrow z^{-(2N-1)} F(-z^{-1}) \ .$$

But, we know that $e^{i(\xi+\pi)} = e^{i\xi}e^{i\pi} = -z$, and $F(z^{-1})=F(z^*)$. From this, if we let $F(z)=m_0(z)\equiv m_0(\xi)$, we can define $m_1(\xi)$ in terms of $m_0(\xi)$:

$$m_1(z) = z^{-(2N-1)} m_0(-z^{-1}) \quad or$$

$$m_1(\xi) = e^{-i(2N-1)\xi} m_0^*(\xi+\pi) \ .$$

Restated, the paraunitary high pass/low pass filter pair are defined by

$$H(\xi) = \sqrt{2}\, m_0(\xi)$$

$$G(\xi) = \sqrt{2}\, m_1(\xi) \quad ,$$

where

$$m_0(\xi) = \frac{1}{\sqrt{2}} \sum_0^{2N-1} h_k\, e^{-ik\xi}$$

$$m_1(\xi) = \frac{1}{\sqrt{2}} \sum_0^{2N-1} g_k\, e^{-ik\xi} \quad .$$

Putting this all together, if we define $m_0(\xi)$ such that $H(\xi)$ and $G(\xi)$ form a paraunitary filter pair (i.e., such that the linear transformation matrix U, constructed of $H(\xi)$ and $G(\xi)$, is unitary), then the IFFT (Inverse Fast Fourier Transform) can be used to generate the sequence of filter coefficients $\{h_n\}$ and , therefore, $\{g_n\}$. The following construction and examples will hopefully clarify this entire process.

3. Daubechies' Construction

The construction of the Daubechies' orthonormal wavelet basis ([1][2]) begins with the concept of regularity and vanishing moments. Now that have defined $m_1(\xi)$ and its relationship to the Fourier transform of the filter coefficients $\{g_k\}$

$$m_1(\xi) = \frac{1}{\sqrt{2}} \sum_{k=0}^{2N-1} g_k e^{-ik\xi} \quad ,$$

from (2.6) the Fourier transform of (2.10) becomes

$$\hat{\psi}(\xi) = m_1(\xi/2)\, \hat{\phi}(\xi/2) \quad . \qquad (3.1)$$

Equation (1.3) puts the restriction on $\psi(t)$ that the first m+1 moments vanish. We must show that this restriction implies *m*th order differentiability of $\hat{\psi}(\xi)$. This differentiability will lead to a unique factorization of $m_0(\xi)$, which will be needed in the construction of the Daubechies' wavelet basis. First, consider the Fourier transform of $\psi(t)$,

$$\hat{\psi}(\xi) = \int_{-\infty}^{\infty} \psi(t)\, e^{-i\xi t} dt \quad .$$

The first derivative with respect to ξ yields

$$\frac{d\hat{\psi}}{d\xi} = \int_{-\infty}^{\infty} \psi(t)\,(-it)\, e^{-i\xi t} dt \quad .$$

Continued differentiation within the integral gives

$$\frac{d^m \hat{\psi}}{d\xi^m} = \int_{-\infty}^{\infty} (-it)^m \psi(t)\, e^{-i\xi t} dt \quad ,$$

so that

$$\frac{d^m \hat{\psi}}{d\xi^m}\Big|_{\xi=0} = (-i)^m \int_{-\infty}^{\infty} t^m \psi(t)\, dt \quad . \tag{3.2}$$

Now, from (1.3)

$$\int_{-\infty}^{\infty} t^j \psi(t)\, dt = 0 \quad j = 0,1,\ldots,m \ .$$

From this condition, (3.2) becomes

$$\frac{d^j \hat{\psi}}{d\xi^j}\Big|_{\xi=0} = 0 \ , \quad j = 0,1,\ldots,m \quad . \tag{3.3}$$

But we know that

$$\hat{\psi}(\xi) = e^{-i\xi/2} m_0^*(\xi/2+\pi)\, \hat{\phi}(\xi/2) \quad \text{and}$$

$$\hat{\phi}(0) \neq 0 \quad .$$

Therefore, with (3.3) showing mth order differentiability at $\xi=0$, we have that m_0 is m times differentiable at $\xi=\pi$:

$$\frac{d^j m_0}{d\xi^j}\Big|_{\xi=\pi} = 0 \ , \quad j = 0,1,\ldots,m \quad .$$

This implies that m_0 has a zero of order m+1 at $\xi=\pi$. At this point, we factor $m_0(\xi)$ into a polynomial $R(\xi)$ that

is the multiple zero at $\xi=\pi$ and another polynomial $Q(\xi)$, such that $m_0(\xi) = R(\xi)Q(\xi)$ (where R is order N=m+1 and Q is order m). To construct $R(\xi)$, we realize that when normalized $m_0(0)=(1)^{m+1}Q(0)$, and $m_0(\pi)=(0)^{m+1}Q(\pi)$. This gives the following factorization of $m_0(\xi)$,

$$m_0(\xi) = \left(\frac{1+e^{i\xi}}{2} \right)^{m+1} Q(\xi) \quad . \tag{3.4}$$

This factorization is important because it allows for the development of the polynomial Q in $z=e^{i\xi}$, from which the polynomial m_0 in z can be realized directly from the factorization of (3.4).

We now have everything necessary to develop a trigonometric polynomial for

$$m_0(\xi) = \frac{1}{\sqrt{2}} \sum_{k=0}^{2N-1} h_k e^{-ik\xi} \quad ,$$

such that

$$\left|m_0(\xi)\right|^2 + \left|m_0(\xi+\pi)\right|^2 = 1 \quad .$$

First, consider the factorization for $m_0(\xi)$:

$$m_0(\xi) = \left(\frac{1+e^{i\xi}}{2} \right)^{m+1} Q(\xi) \quad .$$

From this representation, it follows that

$$\left|m_0(\xi)\right|^2 = \left[\cos^2 \frac{1}{2}\xi\right]^N \left|Q(\xi)\right|^2 \quad .$$

With the coefficients of $|Q(\xi)|^2$ being real, it can be written as a real polynomial in $\cos(\xi)$, or equivalently as a polynomial P in $\sin^2(\xi/2)$. Letting $y = \sin^2(\xi/2) =$

(1-cos(ξ))/2, we can write

$$M_0(\xi) = \left|m_0(\xi)\right|^2 = \left[\cos^2\frac{1}{2}\xi\right]^N P(\sin^2\frac{\xi}{2})$$

as a polynomial is $\sin^2(\xi/2)$ such that from (2.9)

$$M_0(\xi) + M_0(\xi+\Pi) = 1 \quad .$$

As a polynomial in $y = \sin^2(\xi/2)$, P must satisfy

$$(1-y)^N P(y) + y^N P(1-y) = 1 \quad . \qquad \textbf{(3.5)}$$

To solve for P, I. Daubechies' [2,pp 169] introduces a rather obscure theorem by Bezout that succinctly states the unique existence and order of P given (3.5). The order of P is important since P will ultimately be written as a truncated Taylor expansion.

Theorem 3.1 (Bezout): *If p_1,p_2 are two polynomials of degree n_1,n_2 respectively, with no common zeros, then there exists unique polynomials q_1,q_2 of degree n_2-1,n_1-1, respectively, so that*

$$p_1(x)q_1(x) + p_2(x)q_2(x) = 1 \quad .$$

By this theorem, there exist unique polynomials q_1, q_2 of degree ≤ N-1 such that

$$(1-y)^N q_1(y) + y^N q_2(y) = 1 \quad .$$

Now substituting y for (1-y)

$$y^N q_1(1-y) + (1-y)^N q_2(1-y) = 1 \quad .$$

Then, by the uniqueness in Bezout's theorem (3.1), $q_2(y)=q_1(1-y)$, so that

$$(1-y)^N q_1(y) + y^N q_1(1-y) = 1 \quad .$$

Solving for $q_1(y)$ we get

$$q_1(y) = (1-y)^{-N} - y^N \frac{q_1(1-y)}{(1-y)^N} \quad .$$

Since q_1 is degree $\leq N-1$, $(1-y)^{-N}$ can be written as the first N terms of its Taylor expansion plus residual as

$$(1-y)^{-N} = \sum_{k=0}^{N-1} \binom{N+K-1}{k} y^k + O(y^N) \quad .$$

Substituting the Taylor expansion into $q_1(y)$ gives

$$q_1(y) = \sum_{k=0}^{N-1} \binom{N+K-1}{k} y^k + O(y^N) - y^N \frac{q_1(1-y)}{(1-y)^N}$$

$$= \sum_{k=0}^{N-1} \binom{N+K-1}{k} y^k + O(y^N) \quad .$$

Again, since $q_1(y)$ is degree $\leq N-1$, $q_1(y) \equiv P(y)$ becomes

$$P(y) = \sum_{k=0}^{N-1} \binom{N+K-1}{k} y^k \quad .$$

We can now return to the previous substitution $y=\sin^2(\xi/2)$, which give us P as a polynomial in $\sin^2(\xi/2)$ instead of y:

$$P(\sin^2\frac{\xi}{2}) = \sum_{k=0}^{N-1} \binom{N+K-1}{k} (\sin^2\frac{\xi}{2})^k = \left|\varrho(\xi)\right|^2 \quad .$$

With this representation, we can write the transfer function for $M_0(\xi)$:

$$M_0(\xi) = \left| m_0(\xi) \right|^2 = (\cos^2 \frac{\xi}{2})^N \sum_{k=0}^{N-1} \left(\begin{matrix} N+K-1 \\ k \end{matrix} \right) (\sin^2 \frac{\xi}{2})^k \quad .$$

What we are really interested in, however, is $Q(\xi)$, so that we have the necessary polynomial $m_0(\xi)$ and hence $H(\xi)$ in $z=e^{i\xi}$ to obtain the filter coefficients $\{h_n\}$ via the inverse Fourier transform.

To extract $Q(\xi)$ from $|Q(\xi)|^2$, we realize that $|Q(\xi)|^2 = P(\sin^2\xi/2)=Q(\xi)Q^*(\xi)$. This yields a spectral factorization for the roots of $|Q(\xi)|^2$, from which the "square root" of $|Q(\xi)|^2$ can be extracted [2, pp. 172-173].

3.1 Spectral Factorization of $|Q(\xi)|^2$

The process of extracting the square root from $|Q(\xi)|^2$ begins by considering the spectrum of the roots of the polynomial

$$\left| Q(\xi) \right|^2 = P(\sin^2 \frac{\xi}{2}) = \sum_{k=0}^{N-1} \left(\begin{matrix} N+K-1 \\ k \end{matrix} \right) (\sin^2 \frac{\xi}{2})^k \quad .$$

Realizing that this polynomial is a real product of conjugate factors ($|Q(\xi)|^2 = Q(\xi)Q^*(\xi)$), the resultant polynomial can be written as a real polynomial in $x=\cos(\xi)$. Stated as a polynomial in x, it can be factored as

$$\rho_A(x) = \alpha \prod_{j=1}^{N-1} (x-c_j) \quad ,$$

where the c_js appear as complex conjugate pairs or real singlets. Because these roots are a product of conjugate factor, we can extract a polynomial in $z=e^{i\xi}$ by writing a new polynomial as a product of averaged factors in z, with $|z|=1$,

$$P_A(z) = z^m \alpha \prod_{j=1}^{m} \left(\frac{z+z^{-1}}{2} - c_j\right) , \; m=N-1 \;\; ,$$

$$= \alpha \prod_{j=1}^{m} \left(\frac{1}{2} - c_j z + \frac{1}{2} z^2\right)$$

$$= \alpha \prod_{j=1}^{m} \left(z^2 - 2c_j z + 1\right)$$

$$= Q(\xi) Q^*(\xi) = Q(z) Q(z^{-1}) \quad .$$

That is, for each linear term of the polynomial $x=\cos(\xi)$ we will get a quadratic that averages the roots in z. The roots (r_j) of this new polynomial form the spectrum of $P_A(z)$. By the quadratic formula, these roots are of the form

$$r_j = c_j \pm \sqrt{c_j^2 - 1} \quad .$$

By spectral factorization, Q(z) can then be extracted by considering the roots of $P_A(z)$ outside the unit circle:

$$Q(z) = \prod_{j=1}^{m} (z - r_j) \quad .$$

We now have all the necessary roots to create the transfer functions $H(\xi)$ and $G(\xi)$, where

$$H(\xi) = \sqrt{2}\, m_0 \quad \text{and}$$

$$G(\xi) = \sqrt{2}\, m_1 \quad .$$

3.2 Daubechies' Examples for N=2 and N=3

The following examples use spectral factorization to construct the N=2 and N=3 Daubechies' polynomials $m_0(\xi)$. From these polynomials, $H(\xi)$ and $G(\xi)$ can be created, from which the corresponding filter coefficients $\{h_n\}$ and $\{g_n\}$ can be realized via the inverse Fourier transform. We will start the examples with the Daubechies N=2 case.

N=2: Let $x=\cos(\xi)$ and $z=e^{i\xi}$.

$$P(\sin^2\frac{\xi}{2}) = \sum_{k=0}^{N-1} \binom{N+K-1}{k} (\sin^2\frac{\xi}{2})^k$$

$$= 1 + 2\sin^2(\xi/2)$$

$$= 1 + 2\,(\frac{1-\cos\xi}{2}) = 2-\cos\xi$$

$$= (2-x) \rightarrow c_1 = 2 \quad .$$

Now let m=N-1, and place the root c_1 into $P_A(z)$.

$$P_A(z) = \alpha \prod_{j=1}^{m} (z^2 - 2c_j z + 1)$$

$$= z^2 - 4z + 1$$

$$= Q(z)Q(z^{-1}) \quad .$$

The roots of $P_A(z)$ are $r_1 = 3.7321$ and $r_2 = 0.2679$. By spectral factorization we take the root outside the unit circle. In this case, that is r_1. Consequently,

$$Q(z) = \frac{(z-r_1)}{(1-r_1)} = \frac{(z-3.7321)}{(1-3.7321)} \quad .$$

The $(1-r_1)$ factor simply normalizes Q(z). This factor can be realized simply by evaluating Q at $\xi=0$, that is Q(1). Now that we have Q(z), $m_0(\xi)$ becomes

$$m_0(z) = \left(\frac{1+z}{2}\right)^N \frac{(z-r_1)}{(1-r_1)} \quad \text{or}$$

$$m_0(\xi) = \left(\frac{1+e^{i\xi}}{2}\right)^N \frac{(e^{i\xi}-r_1)}{(1-r_1)} \quad .$$

Finally $m_0(\xi)$ must be normalized so that

$$\sum_k h_k = \sqrt{2} \quad .$$

This gives

$$H(\xi) = \left(\frac{1+e^{i\xi}}{2}\right)^2 (e^{i\xi}-r_1)\left(\frac{\sqrt{2}}{1-r_1}\right) \quad .$$

Now since

$$H(\xi) = \sum_{k=0}^{2N-1} h_k e^{-ik\xi} \quad ,$$

we can take the inverse discrete Fourier transform (using an IFFT) of the 2π-periodic function $H(\xi)$ to obtain the Daubechies' coefficients $\{h_n\}$. Using the Matlab (A.1) code in appendix A, the coefficients $\{h_n\}$ become

```
h = 0.48296291314453 - 0.000000000000000i
    0.83651630373781 - 0.000000000000000i
    0.22414386804201 - 0.000000000000000i
   -0.12940952255126 + 0.000000000000000i
```

After the coefficients are created, Matlab (A.1) test them to see that they do, in fact, equal the square root of 2. Matlab (A.1) further test that the transfer functions $H(\xi)$ and $G(\xi)$ do create a unitary operator, as discussed in section (2.3).

As a contrast for these examples, we will show a plot of $H_2(\xi)$ together with $H_3(\xi)$ after the next example. We now show the Daubechies' example for N=3.

N=3: Let $x=\cos(\xi)$ and $z=e^{i\xi}$.

$$P(\sin^2\frac{\xi}{2}) = \sum_{k=0}^{N-1}\binom{N+K-1}{k}(\sin^2\frac{\xi}{2})^k$$

$$= 1 + 3\sin^2(\xi/2) + 6(\sin^2(\xi/2))^2$$

$$= 1 + 3\left(\frac{1-\cos\xi}{2}\right) + 6\left(\frac{1-\cos\xi}{2}\right)^2$$

$$= x^2 - 3x + 8/3 \;\rightarrow\; c_1 = 1.5 \pm .6455j \;.$$

Now let m=N-1 and place the root c_1 into $P_A(z)$.

$$P_A(z) = \alpha\prod_{j=1}^{m}(z^2 - 2c_j z + 1)$$

$$= (z^2 - 2c_1 z + 1)(z^2 - 2c_1^{*} z + 1)$$

$$= z^4 - 2(c_1 + c_1^{*})z^3 + (2 + 4c_1 c_1^{*})z^2 - 2(c_1 + c_1^{*})z + 1$$

$$= Q(z)Q(z^{-1}) \;.$$

The roots of $P_A(z)$ are $r_1, r_1^{*} = 2.7127 \pm 1.4439j$ and $r_2, r_2^{*} = 0.2873 \pm 0.1529j$. Again, by spectral factorization we take the roots outside the unit circle. In this case, that is r_1 and r_1^{*}. Consequently,

$$Q(z) = \frac{(z - r_1)(z - r_1^{*})}{1 - (r_1 + r_1^{*}) + r_1 r_1^{*}} \;.$$

Again, the denominator simply normalizes Q(z) at $\xi=0$.

From this we get

$$H(\xi) = \left(\frac{1+e^{i\xi}}{2}\right)^3 Q(\xi) \left(\frac{\sqrt{2}}{1-(r_1+r_1^*)+r_1 r_1^*}\right) \quad .$$

Using Matlab (A.2), we again take the IFFT of the 2π-periodic function $H(\xi)$ to obtain the six Daubechies' coefficient set $\{h_n\}$.

```
h = 0.33267055295008 + 0.00000000000000i
    0.80689150931109 - 0.00000000000000i
    0.45987750211849 - 0.00000000000000i
   -0.13501102001025 + 0.00000000000000i
   -0.08544127388203 + 0.00000000000000i
    0.03522629188571 + 0.00000000000000i
```

The rest of Matlab (A.2) checks, again, that the sum of the coefficients is the square root of 2, and that the linear operator comprised of $H(\xi)$ and $G(\xi)$ is unitary.

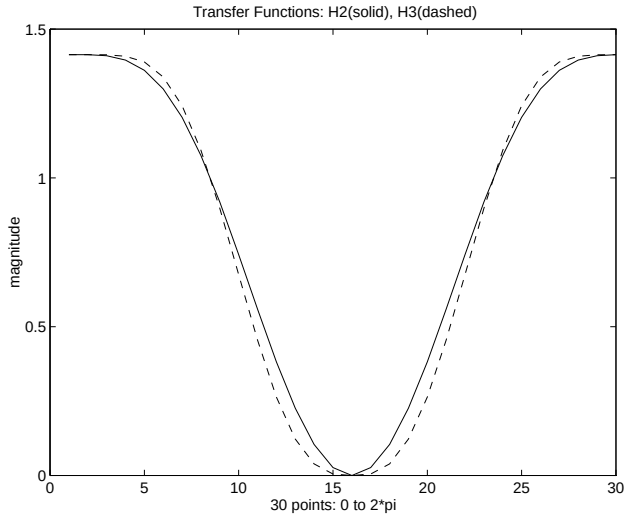

Figure 3.1: $H(\xi)$ for N=2 and N=3

Figure (3.1) shows $H(\xi)$ for N=2 and N=3 as a comparison of the higher degree of vanishing moments of H_3 verses H_2. Another way of looking at this is that H_3 has a steeper low-pass "roll-off" than H_2.

4. Fast Wavelet Transform

With the coefficient set $\{h_n\}$ defined, we can return to the multiresolution analysis defined previously by (2.4) and (2.5)

$$\phi(x) = \sqrt{2} \sum_{n=-\infty}^{\infty} h_n \phi(2x - n) \quad \text{and}$$

$$\psi(x) = \sqrt{2} \sum_{n=-\infty}^{\infty} g_n \phi(2x - n) \quad .$$

In terms of the spectral domain, we have written these recursions as

$$\hat{\phi}(\xi) = \frac{1}{\sqrt{2}} H(\xi/2) \hat{\phi}(\xi/2) \quad \text{and}$$

$$\hat{\psi}(\xi) = \frac{1}{\sqrt{2}} G(\xi/2) \hat{\phi}(\xi/2) \quad , \tag{4.1}$$

where, as a finite sequence

$$H(\xi) = \sum_{k=0}^{2N-1} h_k e^{-ik\xi} \quad \text{and}$$

$$G(\xi) = \sum_{k=0}^{2N-1} g_k e^{-ik\xi} \quad .$$

From this multiresolution analysis, a recursive decimation in frequency algorithm can be realized. In this algorithm developed by Mallat, H provides a low pass filter operator and G a high pass filter operator.

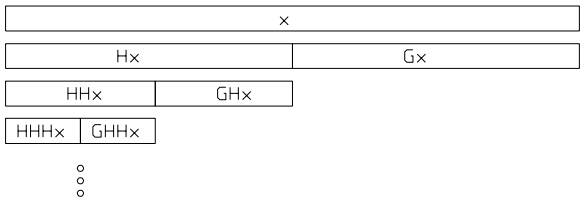

Figure 4.1: Recursive operations of the FWT algorithm, where H is the low pass operator and G the high pass operator.

Figure (4.1) shows the operation of the FWT algorithm with respect to the spectral operators H and G. The bar with an 'x' in it represents a vector in the spectral domain. After each operation, a vector of the same length is created. The left half of the vector is the result of the low pass operator H operating on 'x' (Hx), and the right half is the result of the high pass operator G operating on 'x' (Gx). Consequently, the right half of the new vector is the change between Hx and the original vector 'x'. This follows the multiresolution analysis relationship $V_0 = V_{-1} \oplus W_{-1}$ discussed in section (2.). That is, the vector x that lives in V_0 can be represented as a the sum of two vectors at the next lower resolution. At each step in figure (4.1), equation (4.1) shows us that there is a decimation in frequency of 1/2. Now, since this is a wavelet basis operation for ψ, the result needs to follow the operator G at each resolution. In other words, we need to leave what has been operated on by G and continue to the next resolution, which is a vector of half the size. It also follows that since G operates on the scaling function or father wave ϕ, Hx is what is operated on at the next resolution. Remember, H relates to $\phi(x)$ and G to $\psi(x)$. This algorithm continues until the vector at the current resolution can no longer be decimated.

Since multiplication in the spectral domain is equivalent to convolution in the time domain, we can

create a discrete convolution operator that is equivalent to the decimation in frequency operator derived from the recursions

$$\hat{\Phi}(2\xi) = \frac{1}{\sqrt{2}} H(\xi)\, \hat{\Phi}(\xi)$$

$$\hat{\Psi}(2\xi) = \frac{1}{\sqrt{2}} G(\xi)\, \hat{\Phi}(\xi) \quad ,$$

from which

$$\begin{bmatrix} \hat{\Phi}(2\xi) \\ \hat{\Psi}(2\xi) \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} H(\xi) \\ G(\xi) \end{bmatrix} \hat{\Phi}(\xi) \quad .$$

With a decimation in frequency of 2ξ, every other convolution is discarded yielding a discrete convolution operator of the form,

$$\mathbf{A} = \begin{bmatrix} h_0 & h_1 & h_2 & h_3 & \cdots & & & \\ & & h_0 & h_1 & h_2 & h_3 & \cdots & \\ & & & & h_0 & h_1 & h_2 & h_3 \ \cdots \\ & & & & & & \vdots & \\ h_n & -h_{n-1} & h_{n-2} & -h_{n-3} & \cdots & & & \\ & & h_n & -h_{n-1} & h_{n-2} & -h_{n-3} & \cdots & \\ & & & & h_n & -h_{n-1} & h_{n-2} & -h_{n-3} \ \cdots \\ & & & & & & \vdots & \end{bmatrix} = \begin{bmatrix} H \\ \\ \\ \\ G \\ \\ \\ \end{bmatrix} \quad .$$

For the case where N=2, the linear operation of Ax=b becomes,

$$\mathbf{X} = \mathbf{A}\, \mathbf{x} \quad \text{or}$$

$$\begin{bmatrix} Hx \\ \\ \\ \\ Gx \end{bmatrix} = \begin{bmatrix} h_0 & h_1 & h_2 & h_3 & & & & \\ & & h_0 & h_1 & h_2 & h_3 & & \\ & & & & h_0 & h_1 & h_2 & h_3 \\ h_2 & h_3 & & & & & h_0 & h_1 \\ h_3 & -h_2 & h_1 & -h_0 & & & & \\ & & h_3 & -h_2 & h_1 & -h_0 & & \\ & & & & h_3 & -h_2 & h_1 & -h_0 \\ h_1 & -h_0 & & & & & h_3 & -h_2 \end{bmatrix} \begin{bmatrix} x \\ \\ \\ \\ \\ \\ \\ \\ \end{bmatrix} .$$

This is the Fast Wavelet Transform (FWT). The Inverse Fast Wavelet Transform (IFWT) operator A^{-1} is just the Hermitian since A is unitary. Actually, since A has real entries in the Daubechies' case, A^{-1} is just the transpose of A, and

$$\begin{bmatrix} x \\ \\ \\ \\ \\ \\ \\ \\ \end{bmatrix} = \begin{bmatrix} h_0 & h_1 & h_2 & h_3 & & & & \\ & & h_0 & h_1 & h_2 & h_3 & & \\ & & & & h_0 & h_1 & h_2 & h_3 \\ h_2 & h_3 & & & & & h_0 & h_1 \\ h_3 & -h_2 & h_1 & -h_0 & & & & \\ & & h_3 & -h_2 & h_1 & -h_0 & & \\ & & & & h_3 & -h_2 & h_1 & -h_0 \\ h_1 & -h_0 & & & & & h_3 & -h_2 \end{bmatrix}^T \begin{bmatrix} X \\ \\ \\ \\ \\ \\ \\ \\ \end{bmatrix} .$$

With the FWT operator defined, an algorithm based on this operator can be developed. Also, we can take our first look at the time domain representation of the Daubechies' functions. Until this point, $\phi(x)$ has existed only as a recursive function which could not be displayed.

Suppose for a moment that we have the recursive FWT decomposition algorithm written. To view a basis function we look to another orthogonal basis decomposition algorithm, the FFT. Each element in the discrete spectral domain of the FFT corresponds to a discrete frequency in the time domain (a basis function). Similarly, if we take a single discrete non-zero element in the decomposition domain of the FWT and take the IFWT, a basis function should appear in the discrete time domain. For example, in the Daubechies case where N=2, the vector e(5) (which is a 1 in the fifth position - zero elsewhere) put through a 1024-point IFWT gives the basis function ψ_2 in figure (4.2).

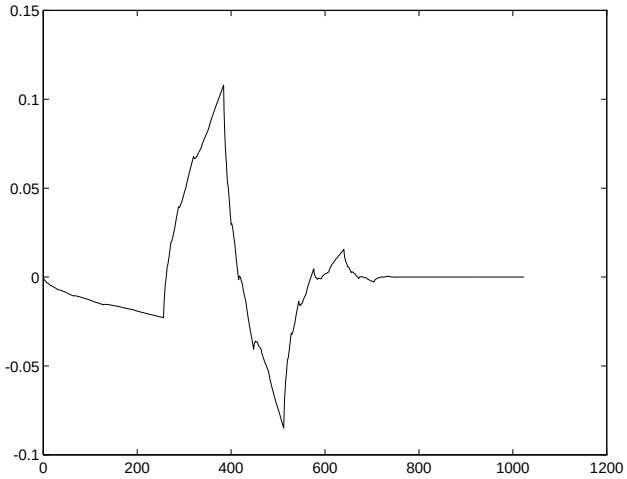

Figure 4.2: The mother wave $\psi(x)$ for Daubechies' N=2.

Similarly, we can view the more regular basis function ψ_3 (figure (4.3)):

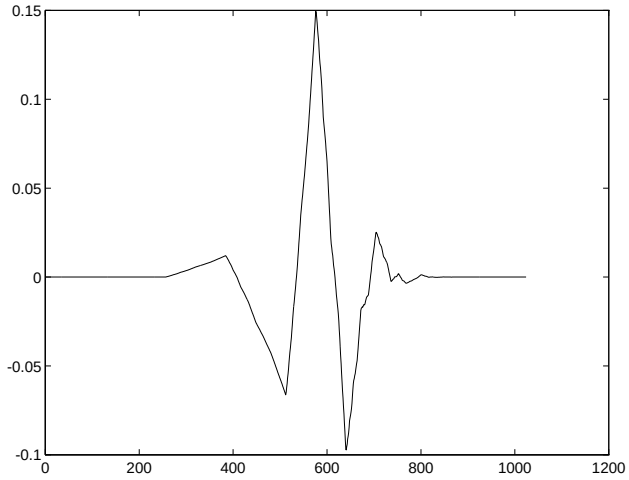

Figure 4.3: The mother wave $\psi(x)$ for Daubechies' N=3.

Notice how the higher number of coefficients for N=3 versus N=2 makes ψ_3 more regular than ψ_2. This progression continues, as the number of coefficients is increased. Figure (4.4) shows this clearly in the plot of ψ_{10}.

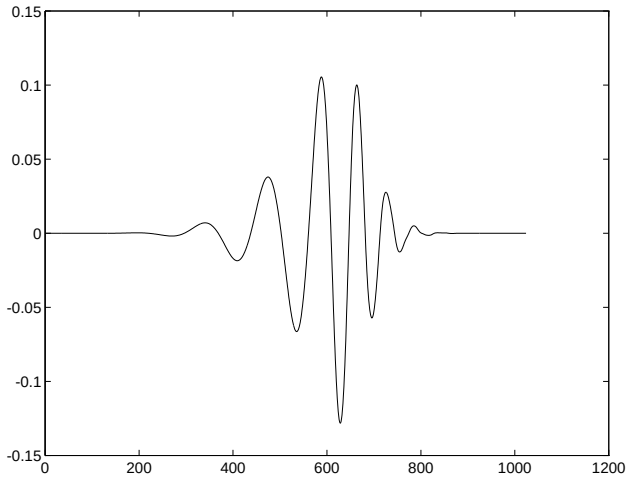

Figure 4.4: The mother wave $\psi(x)$ for Daubechies' N=10.

Not only can we now view the mother wave $\psi(x)$, but we can decompose some sample signals using the FWT. In viewing these examples, keep figure (4.1) and the discussion describing it in mind. The first examples step through the FWT one resolution at a time. Figure (4.5) shows the first level operation on a square pulse. Notice how the right side of the resultant vector is the change between the two resolutions.

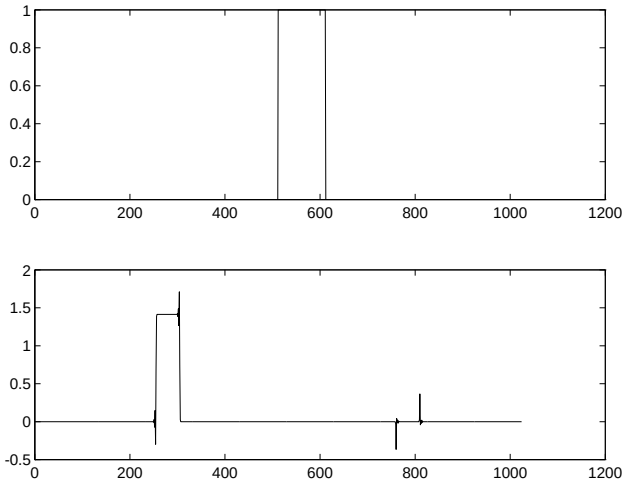

Figure 4.5: Top: original signal. Bottom: First level FWT operation.

Figure (4.6) shows the next level of the FWT decomposition. Again, if we follow figure (4.1), the next change between resolutions is represented by the middle spikes, and the signal that is left is the almost square pulse on the left.

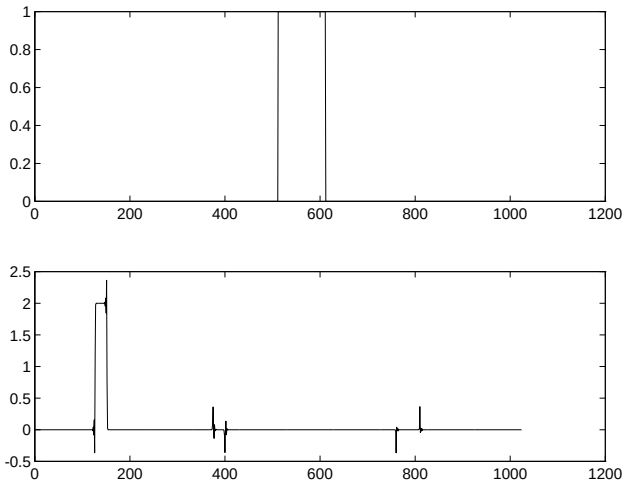

Figure 4.6: Top: original signal. Bottom: Second level FWT operation.

If these operations are continued to the lowest resolution, we get figure (4.7) - the full FWT decomposition of a square pulse. The important thing to realize hear is that most of the energy is well localized. **This is contrary to what a similar decomposition using the FFT would give, which is why wavelets were invented in the first place.** Additionally, Since each operation of the FWT is simply a sparse real matrix multiplication of a (NxM) matrix and a (Mx1) vector (where M is the size of the vector being transformed, and N is the size of the filter coefficient set), the complexity of the FWT algorithm is O(M). This compares to the FFT which is O(M*log(M)). This is another attractive attribute of the FWT over the FFT.

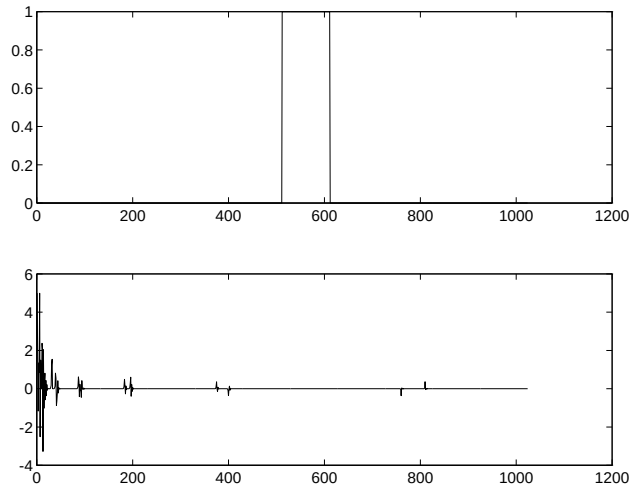

Figure 4.7: Top: original signal. Bottom: Full FWT decomposition.

One final signal decomposition that is rather interesting is that of sine wave (figure (4.8)). This is a signal that the Fourier transform is very good at decomposing (it is just a single frequency in the spectral domain). The FWT, however, doesn't do well at localizing the energy of this signal. Not to worry! This is NOT the type of signal being targeted by the FWT.

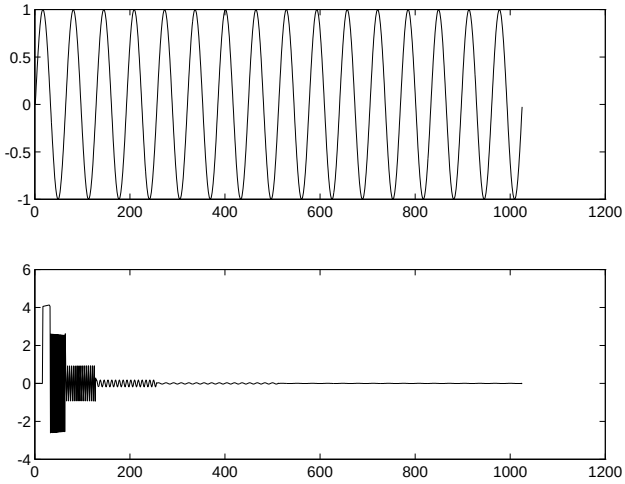

Figure 4.8: Top: original signal. Bottom: Full FWT decomposition.

Now that we have seen what the FWT algorithm can do, and how it is derived, we can detail the actual Matlab code that performs these operations.

4.1 The FWT Algorithm

Like the FFT, the FWT is a power-of-two algorithm. Each successive decomposition of the FWT operates on a vector half the size of the previous operation. This is due to the decimation in time of the multiresolution analysis defined by

$$\phi(x) = \sqrt{2} \sum_n h_n \phi(2x-n) \quad \textit{and}$$

$$\psi(x) = \sqrt{2} \sum_n g_n \phi(2x-n) \quad .$$

Consequently, this power of two algorithm and the FWT operator A, defined previously, define the structure of the FWT algorithm. The core of this algorithm is structured correspondingly, into two Matlab functions. The first function "waveletn.m" (Matlab A.3, appendix A) calls the core operator function "daubn.m" (Matlab A.4, appendix A) for each resolution of either the transform or inverse transform operation depending on whether the variable transform "xform" is greater than zero or not, respectively.

If "transform" is true, the operation continues until the length of the vector is no longer greater than or equal to four. The algorithm stops after four because no more decimation is possible. Each time through the "while" loop the vector is divided in half, corresponding to the multiresolution analysis decimation described above. If "transform" is false, then the inverse transform core operation is called recursively from the lowest resolution (nn=4) to the highest resolution defined by the original vector space size (n).

The function "daubn.m" is relatively straight forward. The transform operation follows the Row vector multiplies defined by the operator A. The main thing to realize here, and in the inverse transform operation, is that these are convolution operations (this is the same as saying that they are filter operation), and, as such,

wrap-around of the filter coefficients $\{h_n\}$ and $\{g_n\}$ must be accounted for. In this algorithm, the wrap-around is accounted for by the check and compensation of the pointer variable "jk" greater then "n".

If the transforming operation follows Row operations, then we would expect the inverse transform operation to follow column operations, and this is the case. The column operations continue for each Row element at column index "i" and "i+nh", where "nh" is simply half the vector length. Like the transform operation, the inner-loop has two multiplies. The loops differ mainly in how the two multiplies are contained in one sum for the inverse transform operation and two separate sums for the transform operation.

Probably the easiest way to understand this algorithm is to follow the Row/column operations of

$$\mathbf{X} = \mathbf{A}\,\mathbf{x} \quad \text{and} \quad \mathbf{x} = \mathbf{A}^{-1}\mathbf{X} \quad .$$

4.2 The FWT and Image Compression

The need to efficiently store and transmit digitized images is becoming increasingly important. The idea is to reduce the amount of data necessary to define an image, thereby reducing storage and transmission bandwidth requirements.

Compression of a digitized image takes two forms: lossy and lossless. Loseless data compression dates at least as far back as the 1970s when Lempel and Ziv, and independently Huffman, developed string matching algorithms that remove the redundancy from a string of characters. More recently, Langden and Rissanen of IBM Corp. developed a probability-based algorithm known as Arithmetic Coding [20]. These lossless algorithms operate recursively on the data set, removing the redundancy, until a certain level of entropy is reached. They are all lossless because the compressed sequence can be decompressed to obtain the exact original sequence.

Lossy data compression techniques also seek to reduce the size of the data set; however, they do not attempt to exactly reconstruct the original sequence when decompressed. This is where the science of lossy compression gets interesting. How much and what type of information can be lost, such that a reasonable facsimile of the original will exist upon decompression? Lossy techniques make sense only in situations such as image and audio compression because, as long as enough of the original content remains, the brain can discern what it is seeing or hearing. For example, a very detailed image of a house can be reduced to a stick drawing, and the brain can still realize the image content.

This stick drawing example really gets to the crux of the image compression problem. That is, what cannot be lost in a lossy image compression algorithm is the edge information. This is what has brought localized wavelet bases to the forefront in image compression algorithms. Because of the localized time and frequency nature of the wavelet basis, the edge information in the transform domain is also localized. This is contrary to the classic Fourier basis, where discontinuities are distributed throughout the spectral domain. This is not to say that the Fourier transform cannot be used in image compression. In fact, the classic JPEG image compression algorithm is based on the real Fourier transform (cosine transform). Very recently, however, wavelet based image compression algorithms have begun to overtake Fourier based algorithms in a big way with compression ratios on the order of 100:1, without loss of image quality. This is what the HARCC (Houston Advanced Research Centre) wavelet-based algorithm, released recently, boasts. HARCC is proprietary. As such, it will be a while before we know how these incredible ratios are obtained.

There are two applications of the FWT in image compression, one being lossless and the other lossy. In a lossless application the FWT is used to localize essential details, such that an image that would not compress due to its randomness will become compressible. In the lossy application, the FWT is performed on the

image. Less significant coefficients of the transformed image are discarded, leaving a highly compressed image. The decompression process then involves performing the IFWT with the reduced coefficient set. Since the larger coefficients retain the essential detail, an arbitrarily close approximation to the original image can be obtained as more coefficients are allowed.

4.3 The FWT Image Compression Algorithm

The FWT image compression algorithm is different than the complete FWT multiresolution decomposition discussed in section (4.). In this algorithm, two dimensions must be considered. Each resolution must operate on both rows and columns of the image matrix.

Accordingly, the algorithm performs just one decimation on each row vector. This separates the essential detail in the horizontal dimension. Next, each column vector is decimated in the same way, thus separating the essential detail in the vertical dimension. Matlab A.5 of Appendix A. shows an example that takes an original 512 X 512 gray-scale image (matrix "I", Matlab A.5) and performs the above two dimensional decomposition twice, creating two multiresolution decompositions. Each time the subspace divides its dimension by two. The wavelet basis used in this example is the Daubechies' N=3, six coefficient basis detailed in section (3.2). Figures A.1,A.2,A.3,A.4,A.5, and A.6 of Appendix A. follow the code in Matlab A.5. Figure A.1 is the original image of a mandrill (baboon), contained in the matrix "I" of MATLAB A.5. Figure A.2 shows the first two dimensional decomposition of mandrill image contained in matrix "I". Actually, the original image is 480 X 500, and as such, has been padded with zeros to a power of 2 (512 X 512). This can be seen as bars on the bottom and side of each quadrant of figure A.2, matrix "C" of Matlab A.5.

At this point the original image does not compress (lossless), but the first resolution decomposition (figure A.2) compresses lossless at approximately 3:1. This is what we expect from the wavelet transform, since it tends to localize the energy of the signal being transformed. As such, changes in the image that tend to be dispersed will become localized. The result of this will be more change in less area, leaving the remaining area with less change. The remaining area will, as a result of having less change, compress better (lossless).

Now we may ask: what would the upper left quadrant of figure (A.2) (which, using the notation of figure (4.1), becomes (HH) in two dimensions), 1/4 of the information, look liked if it were inverse transformed with all other quadrants zero. This image is in Figure A.3 (matrix "D" of Matlab A.5) . Since the quadrant (HH) is the low-pass filter region, the inverse transform will have an averaging effect. Notice how with only 1/4 the data of the original image a reasonable facsimile of the original image evolves from the inverse transform. Consequently, this is one possibility of lossy compression. That is, send only the low-pass quadrant, and take the inverse transform at the receiving end.

After one two dimensional operation, the dimension is now n/2 X n/2. If we perform the same decomposition on the quadrant upper-left quadrant (HH) of Figure A.2, we get the next multiresolution analysis shown in Figure A.4 (matrix "C" of Matlab A.5, again). Notice how the now twice averaged (Low-pass) image appears again in the upper left 1/16 of the matrix. Figure A.4 (Now Matrix "C" of Matlab A.5) now looks like

$$\begin{bmatrix} HHHH & HHHG & & \\ & & HG & \\ HHGH & HHGG & & \\ & & & \\ GH & & GG & \end{bmatrix},$$

where each H and G follow the low-pass and high-pass operations shown in figure (4.1), respectively. The operations follow, from right to left, row, column, row, column, etc ... .

If we now take the inverse transform of the upper left 1/16th of figure A.4 (HHHH), we will get an even more averaged image than Figure A.3. This inverse transformed image, using only 1/16 of the original, is shown in Figure A.5 (matrix "D1" of Matlab A.5). Notice how even though it is highly averaged the edge information is intact. Another thing to notice is the content of other portions of the decomposition of figure A.4 other than the upper left 1/16 (HHHH). These segments of the decomposition contain essential detail in the vertical, horizontal and diagonal dimensions.

The previous example was intended to show the spacial separation provided by the high and low pass multidimensional multiresolution decomposition. We threw away entire segments of the analysis. In a real compression algorithm, since every segment contains some essential detail we would keep some portion of each segment. The question becomes, what criteria should be used to determine what data should be retained and what should be discarded. As far as I can tell from the currently available literature, the criteria is probably the simplest one could imagine: keep the largest magnitude coefficients and discard the smallest. Matlab A.6 performs a two level transform of the original mandrill image (matrix "I" of Matlab A.6). It then

discards every coefficient less then 200 in magnitude. This leaves less then 1/20 of the coefficients. Figure A.6 shows the reconstruction of the original with only these few (1/20) coefficients. Again, notice how the edge information remains intact.

Obviously, the HARCC algorithm is more involved than this example, but it will be some time before we will know the details of this algorithm. All we do know is that it is based some wavelet basis.

Appendix A. Matlab Code and Compression Images

Matlab A.1

```
r=roots([1 -4 1])
```

Note: The discretization of the 2*pi interval includes only the end point at 2*pi, not the end point at zero. This is important so that the IFFT doesn't alias.

```
v=(2*pi/10)
x=2*pi:-v:v;

% Normalize H so that the sum of the hns is sqrt(2).
normalize factor=(sqrt(2)/(1-r(1)));

Q=(exp(i*x)-r(1));
H=(((1+exp(i*x))/2).^2).*Q*normalize factor;
h=ifft(H)
s=cumsum(H);
sqrt(2)-s(1)
```

```
output >> daub qn2

r =  3.73205080756888
     0.26794919243112

h = 0.48296291314453 - 0.00000000000000i
    0.83651630373781 - 0.00000000000000i
    0.22414386804201 - 0.00000000000000i
   -0.12940952255126 + 0.00000000000000i

sqrt(2) - sum{hn} =  0 + 2.195904039796593e-16i
```

This is the error between the calculated square root of 2 by Matlab and the sum of the hns. We can also look at the aforementioned unitary conditions in the spectral domain:

Note: Here c implies complement, and m implies x+pi.

```
v=(2*pi/30)
x=2*pi:-v:v;

a=1/(1-r(1));

H=a*(((1+exp(i*x))/2).^2).*(exp(i*x)-r(1));
Hc=a*(((1+exp(-i*x))/2).^2).*(exp(-i*x)-r(1));
Hm=a*(((1+exp(i*(x+pi)))/2).^2).*(exp(i*(x+pi))-r(1));
Hmc=a*(((1+exp(-i*(x+pi)))/2).^2).*(exp(-i*(x+pi))-r(1));
G=-a*exp(i*3*x).*(((1+exp(-i*(x+pi)))/2).^2).*(exp(-i*(x+pi))-r(1));
```

```
Gc=-a*exp(-i*3*x).*(((1+exp(i*(x+pi)))/2).^2).*(exp(i*(x+pi))-r(1));
Gm=-a*exp(i*3*(x+pi)).*(((1+exp(-i*(x)))/2).^2).*(exp(-i*(x))-r(1));
Gmc=-a*exp(-i*3*(x+pi)).*(((1+exp(i*(x)))/2).^2).*(exp(i*(x))-r(1));

Z0=Hc.*H+Hmc.*Hm; % should be 1
Z1=Gc.*H+Gmc.*Hm; % should be 0
Z2=H.*Hc+G.*Gc;   % should be 1
Z3=Hm.*Hc+Gm.*Gc; % should be 0
```

Matlab A.2

```
c=roots([1 -3 8/3])
r=roots([1 -2*(c(1)+c(2)) (2+4*c(1)*c(2)) -2*(c(1)+c(2)) 1])
```

Note: The discretization of the 2*pi interval includes only the end point at 2*pi, not the end point at zero. This is important so that the IFFT doesn't alias.

```
v=(2*pi/14);
x=2*pi:-v:v;
% Normalize H so that the sum of the hns is sqrt(2).
normalize factor= sqrt(2)/(1-(r(1)+r(2))+r(1)*r(2));

Q=exp(2*i*x)-(r(1)+r(2))*exp(i*x)+r(1)*r(2);
H=(((1+exp(i*x))/2).^3).*Q*normalize factor;
h=ifft(H)
s=cumsum(H);
sqrt(2)-s(1)

output >> daub qn3

c = 1.50000000000000 + 0.64549722436790i
    1.50000000000000 - 0.64549722436790i

r = 2.71274862195598 + 1.44388678261800i
    2.71274862195598 - 1.44388678261800i
    0.28725137804402 + 0.15289233388220i
    0.28725137804402 - 0.15289233388220i


h = 0.33267055295008 + 0.00000000000000i
    0.80689150931109 - 0.00000000000000i
    0.45987750211849 - 0.00000000000000i
   -0.13501102001025 + 0.00000000000000i
   -0.08544127388203 + 0.00000000000000i
    0.03522629188571 + 0.00000000000000i

sqrt(2) - sum{hn} =  0 + 2.831240458600424e-16i
```

Matlab A.3

```
% This function was derived with input from articles
% from William H. Press and Gilbert Strang.

% Prologue:  'a' defines the vector to be transformed
%            'n' defines the length of 'a'
%            'cl' are the low-pass filter coefficients {hn}
%            'ch' are the high-pass filter coefficients {gn}
%            'ncof' indicates the number of coefficients in cl or ch
%            'xform' true for transform false for inverse-transform

function a = waveletn(a,n,cl,ch,ncof,xform)
% Fast Wavelet Xform/invXform function
        if (xform > 0),
                nn=n;
                while (nn >= 4),
                      a=daubn(a,nn,cl,ch,ncof,xform);
                        nn=nn/2;
                end
        else
                nn=4;
                while (nn <= n),
                      a=daubn(a,nn,cl,ch,ncof,xform);
                        nn=nn*2;
                end
        end

return
```

Matlab A.4

```
% This function was derived with input from articles
% from William H. Press and Gilbert Strang.
% Prologue:  'a' defines the vector to be transformed
%            'n' defines the length of 'a'
%            'cl' are the low-pass filter coefficients {hn}
%            'ch' are the high-pass filter coefficients {gn}
%            'ncof' indicates the number of coefficients in cl or ch
%            'xform' true for transform false for inverse-transform

function a = daubn(a,n,cl,ch,ncof,xform)

nh=n/2;
% clear the work space
for i=1:n,
w(i)=0;
end
        if (xform > 0),
                i=1;
                for j=1:2:n,
                    jk=j;
                    for k=1:ncof,
                          if (jk>n)
                             jk=jk-n;
                          end
                        w(i)=w(i)+cl(k)*a(jk);
                        w(i+nh)=w(i+nh)+ch(k)*a(jk);
                        jk=jk+1;
                    end
                        i=i+1;
                end
        else
                i=1;
                for j=1:2:n,
                    jk=j;
                    for k=1:ncof,
                          if (jk>n)
                             jk=jk-n;
                          end
                        w(jk)=w(jk)+cl(k)*a(i)+ch(k)*a(i+nh);
                        jk=jk+1;
                    end
                        i=i+1;
                end
        end
%  save the work space result back into 'a'
        for i=1:n,
        a(i)=w(i);
        end
return
```

Matlab A.5

```
% Image Compression example using the FWT multiresolution
decomposition.

% Prologue:  'I' the original image contained in a 512x512 matrix of
%               gray scale values.
%            'C' the transformed gray scale matrix: I is decomposed
to
%               two levels in this example.
%            'ncof' the number of Daubechies' coefficients (filter
length).
%            'n' matrix dimention

n=512;
ncof=6;
% Transform the image I
xform=1;
% First along Rows
for i=1:n,
A(i,:)=daubn(I(i,:),n,cl,ch,ncof,xform);
end
% Then along Columns
for i=1:n,
C(:,i)=daubn(A(:,i),n,cl,ch,ncof,xform);
end

% Take the inverse transform of the HH portion of C.
% The other 3/4 of the matrix are zeroed.

xform=-1;
I=zeros(n,n);
I(1:n/2,1:n/2)=C(1:n/2,1:n/2);
for i=1:n,
A(1:n,i)=daubn(I(1:n,i),n,cl,ch,ncof,xform);
end
for i=1:n,
D(i,1:n)=daubn(A(i,1:n),n,cl,ch,ncof,xform);
end
% D contains the uncompressed image of the first resolution

% Take the transform to the next level of resolution;
% i.e., take the transform of the HH 1/4 portion of C.

xform=1;
for i=1:n/2,
A(1:n/2,i)=daubn(C(1:n/2,i),n/2,cl,ch,ncof,xform);
end
for i=1:n/2,
C(i,1:n/2)=daubn(A(i,1:n/2),n/2,cl,ch,ncof,xform);
end
```

```
% C now contains the second level of resolution

% Take the inverse transform of the HHHH portion of C.
% The other 15/16 of the matrix are zeroed.

xform=-1;
I=zeros(n/2,n/2);
I(1:n/4,1:n/4)=C(1:n/4,1:n/4);
for i=1:n/2,
A(1:n/2,i)=daubn(I(1:n/2,i),n/2,cl,ch,ncof,xform);
end
for i=1:n/2,
I1(i,1:n/2)=daubn(A(i,1:n/2),n/2,cl,ch,ncof,xform);
end
I=zeros(n,n);
I(1:n/2,1:n/2)=I1(1:n/2,1:n/2);
for i=1:n,
A(1:n,i)=daubn(I(1:n,i),n,cl,ch,ncof,xform);
end
for i=1:n,
D1(i,1:n)=daubn(A(i,1:n),n,cl,ch,ncof,xform);
end
% D1 contains the uncompressed image of the second resolution

% view the uncompressed image of the second resolution.
image(D1)
colormap(gray(200))
```

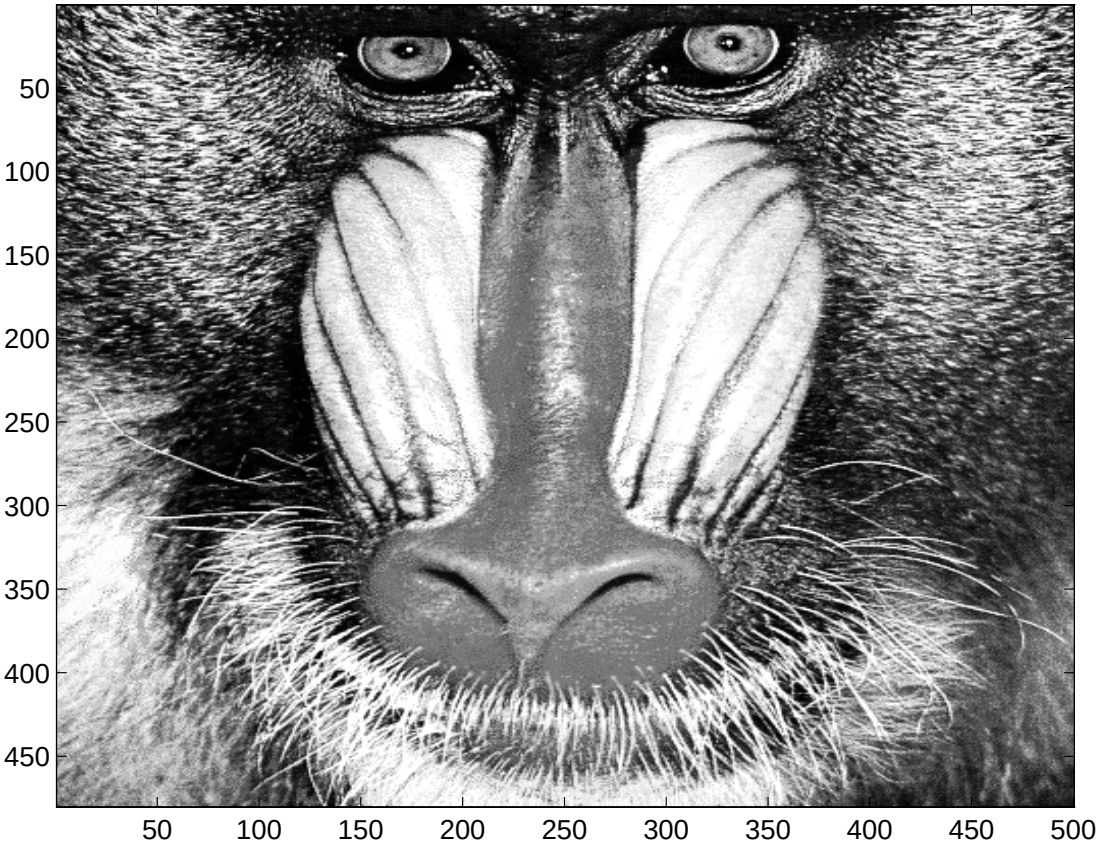

Figure A.1: This is the original image of a mandrill, contained in matrix "I" of Matlab A.5.

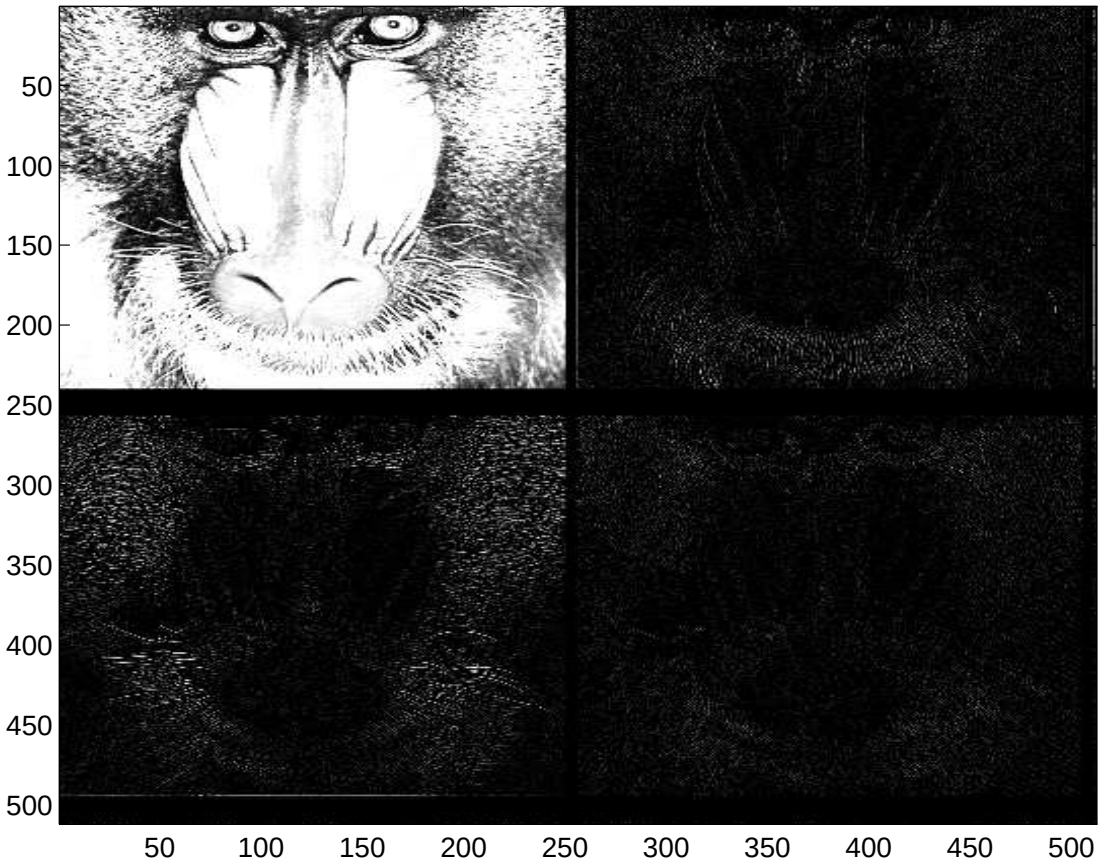

Figure A.2: This is the first level transform of the original image (nxn). This image is contained in matrix "C" of Matlab A.5.

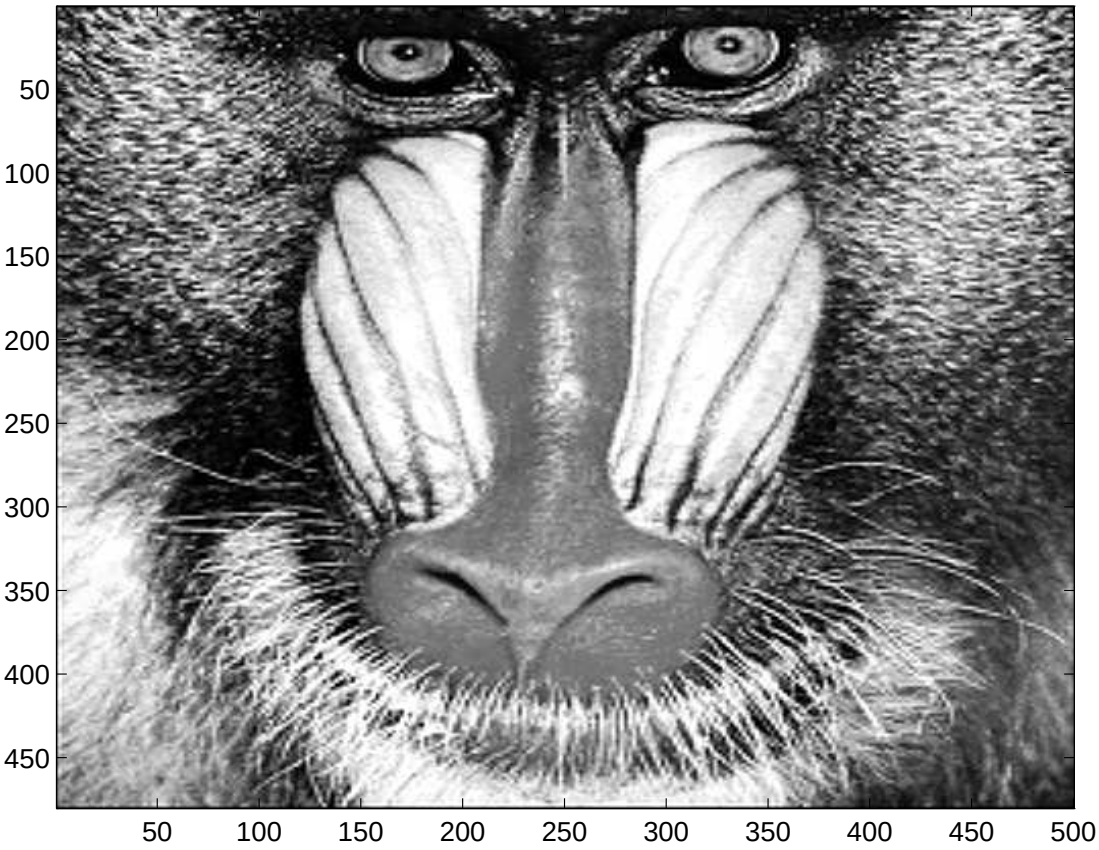

Figure A.3: This is the inverse transform of the (n/2 x n/2) upper left quadrant of figure A.2 (matrix "C" of Matlab A.5), with all other quadrants zero. This image is contained in matrix "D" of Matlab A.5.

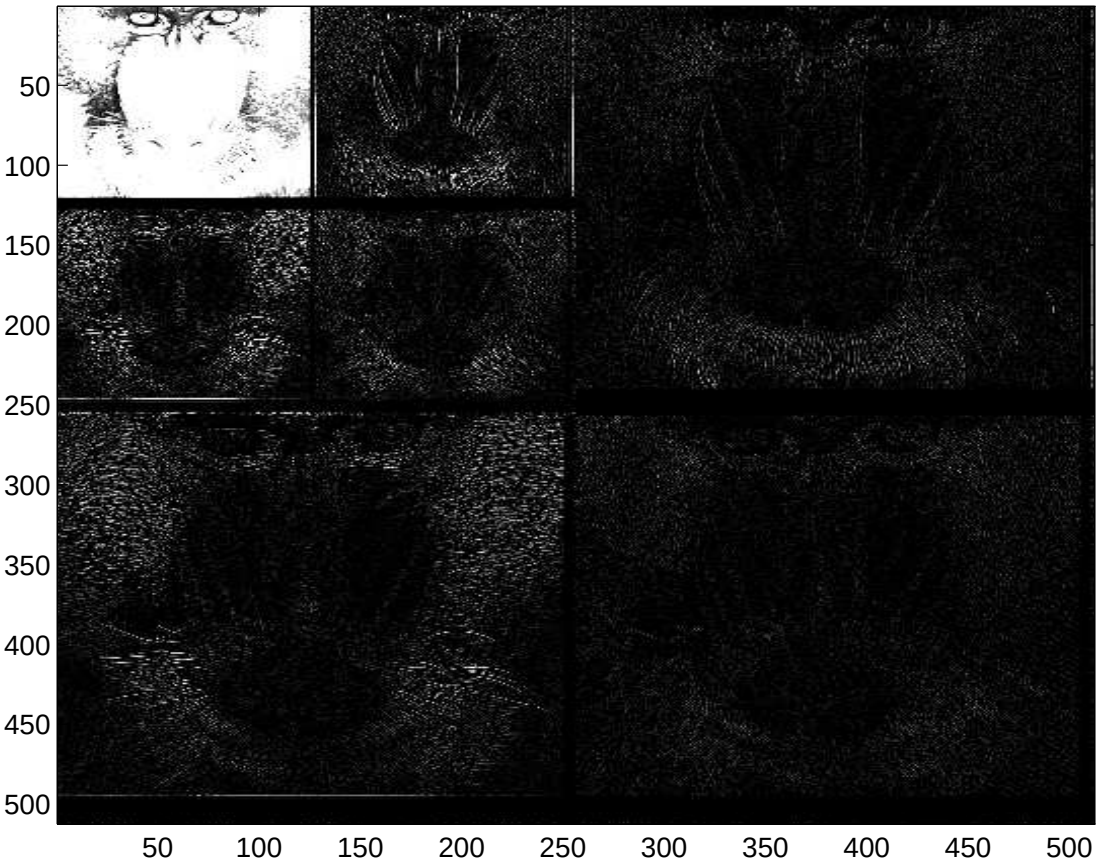

Figure A.4: This is the second level transform of the original image (nxn). The first operation was at the dimension (nxn), and the second operation was at the dimension (n/2 n/2). This image is contained in the second occurence of matrix "C" in Matlab A.5.

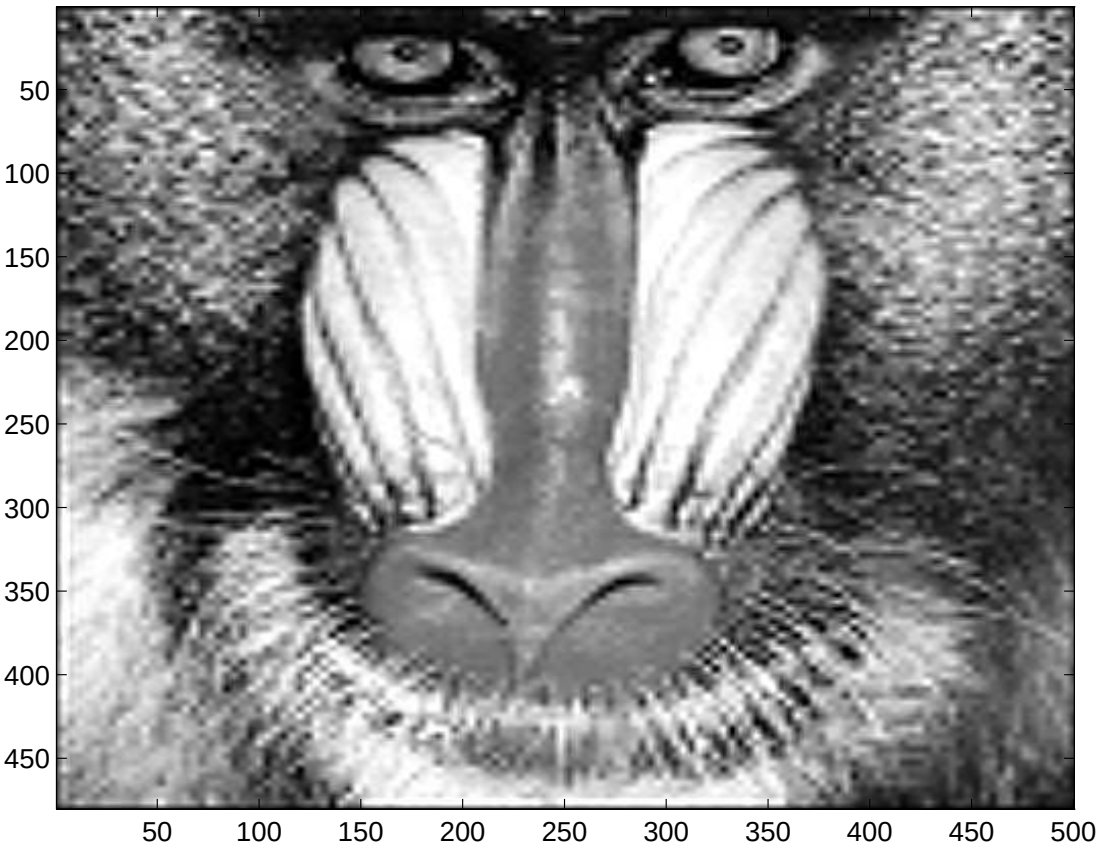

Figure A.5: This is the inverse transform of the (n/4 x n/4) upper left 1/16th of figure A.4 (matrix "C" of Matlab A.5), with the rest of the matrix zero. This image is contained in matrix "D1" of Matlab A.5.

Matlab A.6

```
% Image Compression example using the FWT multiresolution
decomposition.

% Prologue: 'I' the original image contained in a 512x512 matrix of
%               gray scale values.
%           'C' the transformed gray scale matrix: I is decomposed to
%               two levels in this example.
%     'ncof' the number of Daubechies' coefficients (filter  length).
%            'n' matrix dimention

n=512;
ncof=6;
% Transform the image I
xform=1;
% First along Rows
for i=1:n,
A(i,:)=daubn(I(i,:),n,cl,ch,ncof,xform);
end
% Then along Columns
for i=1:n,
C(:,i)=daubn(A(:,i),n,cl,ch,ncof,xform);
end

% Take the transform to the next level of resolution;
% i.e., take the transform of the HH 1/4 portion of C.

xform=1;
for i=1:n/2,
A(1:n/2,i)=daubn(C(1:n/2,i),n/2,cl,ch,ncof,xform);
end
for i=1:n/2,
C(i,1:n/2)=daubn(A(i,1:n/2),n/2,cl,ch,ncof,xform);
end
% C now contains the second level of resolution

% As an example, zero all coefficients less than 200.
for i=1:n,
for j=1:n,
if (abs(C(i,j)) < 200),
C(i,j)=0;
end
end
end
```

```
% Take the inverse transform of the second resolution
% transformed image with only the more significant
% coefficients retained.

xform=-1;

for i=1:n/2,
A(1:n/2,i)=daubn(C(1:n/2,i),n/2,cl,ch,ncof,xform);
end
for i=1:n/2,
C(i,1:n/2)=daubn(A(i,1:n/2),n/2,cl,ch,ncof,xform);
end

for i=1:n,
A(1:n,i)=daubn(C(1:n,i),n,cl,ch,ncof,xform);
end
for i=1:n,
C(i,1:n)=daubn(A(i,1:n),n,cl,ch,ncof,xform);
end
% C contains the uncompressed image of the second resolution

% view the uncompressed image of the second resolution.
image(C)
colormap(gray(200))
```

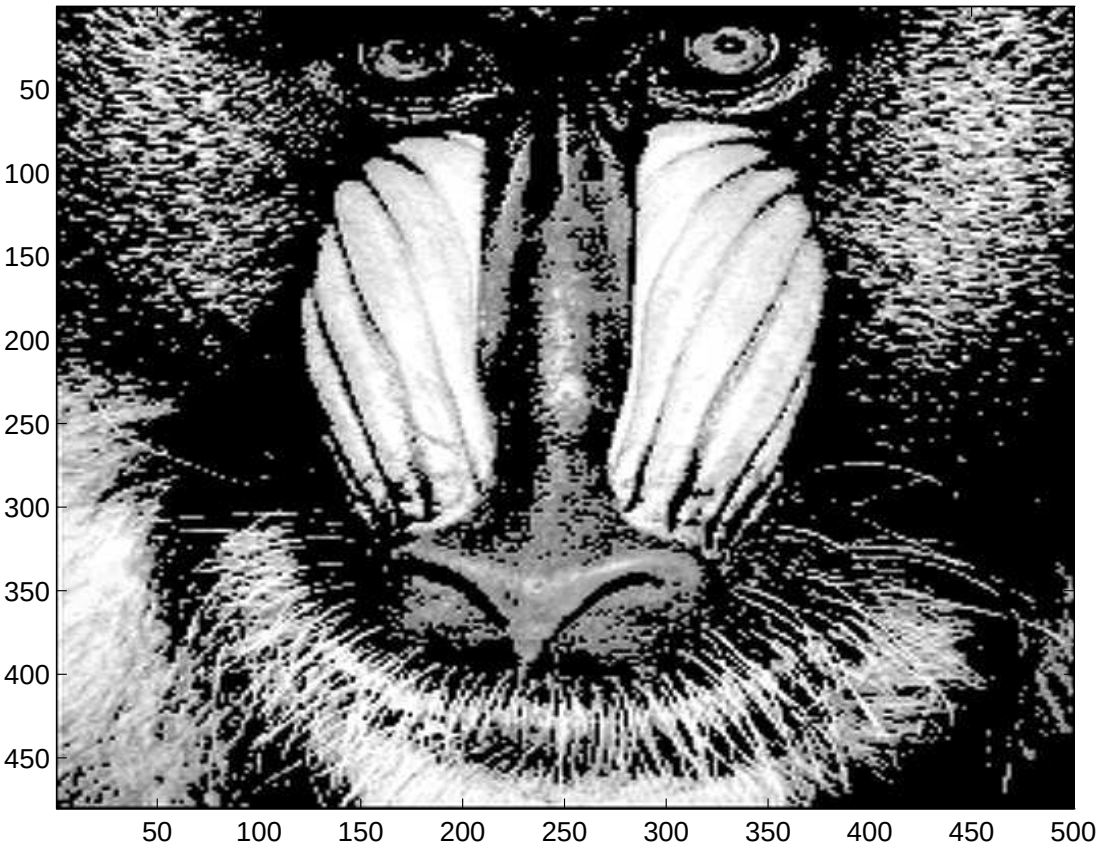

Figure A.6: Inverse transform of figure A.4 with only 1/20th of the coefficients retained.

References

[1] I. Daubechies, "Orthonormal Bases of Compactly Supported Wavelets," Communications on Pure and Applied Mathematics, vol. XLI, pp. 909-996, 1988.

[2] I. Daubechies, "Ten Lectures on Wavelets," SIAM, Philadelphia, PA, 1992.

[3] T.H. Koornwinder, "Wavelets: An Elementary Treatment of Theory and Applications," World Scientific Publishing Co., River Edge, NJ, 1993.

[4] G.G. Walter, "Wavelets and Other Orthogonal Systems With Applications," CRC Press, Boca Taton, FL, 1994.

[5] Y. Meyer, "Wavelets, Algorithms & Applications," SIAM, Philadelphia, PA, 1993.

[6] J.M. Combes and A. Grossmann, "Wavelets, Time-Frequency Methods and Phase Space," Springer-Verlag, Berlin Heidelberg, 1989.

[7] C.K. Chui, "Wavelets: A tutorial in Theory and Applications," Academic Press, San Diego, CA, 1992.

[8] L.L. Schumaker and G. Webb, "Recent Advances in Wavlet Analysis," Academic Press, San Diego, CA, 1994.

[9] M.B. Ruskai, "Wavelets and Their Applications," Jones and Bartlett, Boston, MA, 1992.

[10] P. Duhamel et al., "Special Issue on Wavelets and Signal Processing," IEEE Trans. on Signal Processing, vol. 41, 1993.

[11] O. Rioul, "A Remez Exchange Algorithm for Orthonormal Wavelets," IEEE Trans. on Circuits and Systems, vol. 41, pp. 550-560, Aug. 1994.

[12] S.G. Mallat, "Multiresolution Approximations and Wavelet Orthonormal Bases of $L^2(\mathbf{R})$," Trans. of the American Mathematical Society, vol. 315, pp. 69-86, Sept. 1989.

[13] S.G. Mallat, "A Theory for Multiresolution Signal Decomposition: The Wavelet Representation," IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. 11, pp. 674-693, Jul. 1989.

[14] G. Strang, "Wavelet Transforms Versus Fourier Transforms," American Mathematical Society, vol. 28, pp. 288-304, Apr. 1993.

[15] R.A. Gopinath, "Optimal Wavelet Representation of Signals and the Wavelet Sampling Theorem," IEEE Trans. on Circuits and Systems, vol. 41, pp. 262-277, Apr. 1994.

[16] I. Daubechies, "The Wavelet Transform, Time-Frequency Localization and Signal Analysis," IEEE Trans. on Information Theory, vol. 36, pp. 961-1004, Sept. 1990.

[17] P.J. Burt and E.H. Adelson, "The Laplacian Pyramid as a Compact Image Code," IEEE Trans. on Communications, vol. 31, pp. 532-540, 1983.

[18] J.J. Benedetto and M.W. Frazier, "Wavelets, Mathematics and Applications," CRC Press, Boca Raton, FL, 1994.

[19] M.V. Wickerhauser, "Adapted Wavelet Analysis, from Theory to Software," A.K Peters, Wellesley, MA, 1994.

[20] T.C. Bell et al., "Text Compression," Prentice Hall, Englewood Cliffs, NJ, 1990.