2012-01-01

Lifting and Wavelets: A Factorization Method

Adrian Delgado
University of Texas at El Paso, adelgado8@miners.utep.edu

LIFTING AND WAVELETS: A FACTORIZATION METHOD

ADRIAN DELGADO

Department of Mathematical Sciences

APPROVED:

Helmut Knaust, Ph.D., Chair

Emil Schwab, Ph.D.

Ricardo Freitas von Borries, Ph.D.

Benjamin C. Flores, Ph.D.
Interim Dean of Graduate School

LIFTING AND WAVELETS: A FACTORIZATION METHOD

by

ADRIAN DELGADO

THESIS

Presented to the Faculty of the Graduate School of

The University of Texas at El Paso

in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE

Department of Mathematical Sciences

UNIVERSITY OF TEXAS AT EL PASO

May 2012

Acknowledgements

I would like to express my appreciation to my advisory committee: Dr. Helmut Knaust, Dr. Emil Schwab, and Dr. Ricardo Freitas von Borries. Thanks to my graduate advisor Dr. Knaust for his time, patience, and understanding. I have worked with Dr. Knaust and Dr. Schwab from the Mathematics department since I was an undergraduate. I have learned a lot through coursework, employment, and direction under them. I thank you Dr. von Borries for taking the time to be in my committee from your department in Electrical Engineering. I also would like to thank my family and friends for the support and help throughout my studies. I am here not only because of my hard work, but because of everyone mentioned above and their contribution. I have been blessed to receive different forms of help and support from everyone around me and I thank everyone through this acknowledgement.

Abstract

We begin with the concept of a discrete wavelet transformation. We begin with a scaling function satisfying a certain number of properties to be able to implement the wavelet transformation. We will then require translates of the scaling function to obey the same set of properties and this set will form what is called a *Multiresolution Analysis*. We then switch the ideas to the Fourier transform domain were we get equivalent results. However, instead of choosing a scaling function with coefficients to satisfy a set of properties, we will work backwards and construct a scaling function based on having a set of coefficients satisfy the necessary properties. This method of construction will lead us to some classic wavelet transformations known as the *Haar* and *Daubechies* wavelets. We then introduce the lifting method for wavelet transformations. Unlike the discrete wavelet transform, the lifting scheme divides the signal into even and odd components and performs both a prediction and update step. We will then show through implementation how this method is computationally more efficient. Furthermore, as shown in Daubechies and Sweldens [2], we will prove that any discrete wavelet transformation can be decomposed into lifting steps. This decomposition corresponds to a factorization of the associated polyphase matrix of a wavelet transform. This factorization is implemented with the help of the Euclidean algorithm, with the focus on the Laurent polynomials to represent our filters. This paper will then conclude with some classical wavelet transformations and compare the results to using the lifting method through implementation on *Mathematica*.

Table of Contents

Page

1 Introduction

Fourier analysis has been around since the nineteenth century with applications aimed at both the mathematics and engineering community. The theory of Fourier series is concerned with the idea of being able to write a periodic function as a sum of *simple waves*. A *simple wave* can be thought of as a linear combination of the trigonometric functions:

$$\sin(nt) \qquad \text{or} \qquad \cos(nt).$$

Expressing a function in this form will allow us to identify various frequency components of the function. Wanting to identify these relative frequencies is a common problem in signal analysis. In signal analysis, the idea of filtering out unwanted noise is tied to identifying these unwanted (high-frequency) components. Being able to identify these terms in our function, will allow us to eliminate these unwanted components.

Another reason for using Fourier analysis is in signal compression. Signal compression can be thought of as transmitting information to a receiver while minimizing the storage requirement. One approach we can take is expressing the function $f(t)$ as its Fourier series:

$$f(t) = \sum_n a_n \cos(nt) + b_n \sin(nt).$$

We then want to discard the coefficients a_n and b_n that are smaller than some tolerance for error. Only these coefficients above this tolerance will be used to recover the information sent. Therefore, since we are discarding certain components of the signal while maintaining the quality, we have compressed the size of the original information sent.

We may deal with functions that have more localized features, or sharp jumps. Thus, the sine and cosine waves may not model these certain functions well. The function can have *sharp* components that the are not modeled well by these trigonometric functions locally, so instead we will need a different set of building blocks which we will call *wavelets*. A wavelet can be thought of as a wave that travels for a certain period, but is only nonzero over a finite interval. Once we have a wavelet function constructed, we are able to translate or scale it accordingly to filter or compress a signal.

When we decompose a signal (through either Fourier or wavelet methods), we want the building blocks (sines, cosines or wavelets) to satisfy properties for efficient implementations. A desired property for our set of functions is *orthogonality*. An example of orthogonality is illustrated as follows through

sine functions:

$$\frac{1}{\pi}\int_0^{2\pi} \sin(nt)\sin(mt)dt = \begin{cases} 0, & n \neq m \\ 1, & n = m. \end{cases}$$

Here the orthogonality follows from trigonometric identities which we will use when we derive the Fourier coefficients, a_n and b_n, in the next section. When we construct the wavelet function, a goal will be to have translates and rescalings of the wavelet to also satisfy a certain group of properties, including orthogonality. As defined by Stephane Mallat, once these sets of functions are formed, we can define what we call a *Multiresolution Analysis*.

The Fourier and Wavelet methods follow different algorithms for decomposing a signal. Wim Sweldens, an applied mathematician, introduced an idea to implement the same wavelet transformations to a signal through what we will call a *lifting method*. Given a signal, we first decompose the signal into its polyphase components: the even samples (x_e) and the odd samples (x_o). For example, if we have a vector $x \in \mathbb{R}^8$, splitting into its polyphase components we obtain

$$x = (x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8) \qquad \Longrightarrow \qquad x = (x_2, x_4, x_6, x_8, x_1, x_3, x_5, x_7).$$

When we work with a signal (function) we will be interested in the decomposition and reconstruction of the signal. A signal usually exhibits some correlation between its entries. Consider the signal

$$v = (51, 51, 19, 18, 42, 45, 37, 40).$$

Since we have some correlation between the entries of our signal v, we will be able to make what we will call a *prediction* based on the differences between two successive samples:

$$d = x_e - P(x_o).$$

If the difference is small, we have a good prediction. This is a simpler computation for a prediction step. As we will see in this paper, we can compute more complex prediction steps or more than one prediction step for certain transformations.

We will also compute the mean of two samples. This computation will help us preserve some detail of the original signal. We will use the term *update* for this step:

$$s = x_e + U(d).$$

As we will see in further examples, we can perform more than one update step and it can be more

elaborate than just means between samples. One important feature of this lifting implementation is that the steps are invertible. Invertibility, in terms of signal decomposition, will allow a perfect reconstruction of the given signal.

The filters we will be working with require us to use Laurent polynomials to represent them. A filter h associated with its wavelet transformation will be represented by a Laurent polynomial as follows

$$h(z) = \sum_{k=k_b}^{k_e} h_k z^k.$$

We will then recall the Euclidean algorithm, which was originally developed to find the greatest common divisor of two natural numbers, and extend the idea to find the greatest common divisor of two Laurent polynomials. A key difference we will see is that the division will not be unique. We will also see that the greatest common divisor of two Laurent polynomials is defined up to a monomial.

Through the use of the Euclidean Algorithm, we will show that any wavelet transformation can be factored into lifting steps. The decomposition will consist of products of elementary matrices from the ring $SL(2; \mathbf{R}[z, z^{-1}])$, invertible 2×2 matrices with determinant 1. Last, we will discuss the computational efficiency of computing wavelet transformations through the lifting method. We will compute the implementation on *Mathematica* and see that computing through the lifting method reduces the computational cost considerably when compared to the standard algorithm. When we refer to the standard algorithm of computing wavelet transformations we will refer to applying the polyphase matrix.

2 Fourier Analysis

2.1 Fourier Series

In mathematics, the study of Fourier series, named after the French mathematician Joseph Fourier $(1768 - 1830)$, allows us to represent functions using sums of trigonometric functions. In Fourier's attempt to solve problems in heat conduction, he expressed functions as series of trigonometric functions. This expansion of trigonometric functions led to the Fourier series of a function f, which decomposes a given periodic function and writes it as a sum of simple periodic functions. Specifically, a function is approximated by a sum of sines and cosines or complex exponentials. As a result of this, we are able to write a function f by its Fourier series representation

$$f(x) = a_0 + \sum_{n=1}^{\infty}(a_n \cos nx + b_n \sin nx) \tag{2.1}$$

where a_n and b_n are the Fourier coefficients of f. We calculate a_0, a_n, and b_n as follows

$$a_0 = \frac{1}{2\pi}\int_{-\pi}^{\pi} f(x)dx$$

$$a_n = \frac{1}{\pi}\int_{-\pi}^{\pi} f(x)\cos(nx)dx$$

$$b_n = \frac{1}{\pi}\int_{-\pi}^{\pi} f(x)\sin(nx)dx$$

for $n = 1, 2, 3, \ldots$.

The derivation of each coefficient has to do with having the following integral relations:

$$\frac{1}{\pi}\int_{-\pi}^{\pi} \cos(nx)\cos(kx)dx = \begin{cases} 1 & n = k \geq 1 \\ 2 & n = k = 0 \\ 0 & \text{otherwise} \end{cases} \tag{2.2}$$

$$\frac{1}{\pi}\int_{-\pi}^{\pi} \sin(nx)\sin(kx)dx = \begin{cases} 1 & n = k \geq 1 \\ 0 & \text{otherwise} \end{cases} \tag{2.3}$$

$$\frac{1}{\pi}\int_{-\pi}^{\pi} \cos(nx)\sin(kx)dx = 0 \text{ for all integers } n, k. \tag{2.4}$$

We prove equations (2.2)-(2.4) using the following sum to product trigonometric identities:

$$\cos((n+k)x) = \cos nx \cos kx - \sin nx \sin kx \tag{2.5}$$

$$\cos((n-k)x) = \cos nx \cos kx + \sin nx \sin kx. \tag{2.6}$$

Adding equations (2.5) and (2.6) and integrating over the interval $[-\pi, \pi]$ gives us

$$\int_{-\pi}^{\pi} \cos nx \cos kx dx = \frac{1}{2} \int_{-\pi}^{\pi} (\cos((n+k)x + \cos(n-k)x)) dx.$$

We integrate the right side for the 3 cases.

If $n \neq k$, then

$$\int_{-\pi}^{\pi} \cos nx \cos kx dx = \frac{1}{2} \left[\frac{\sin(n+k)x}{n+k} + \frac{\sin(n-k)x}{n-k} \right] \Bigg|_{-\pi}^{\pi} = 0.$$

If $n = k \geq 1$, then

$$\int_{-\pi}^{\pi} \cos^2 nx dx = \int_{-\pi}^{\pi} \frac{1}{2}(1 + \cos 2nx) dx = \pi.$$

If $n = k = 0$, then

$$\int_{-\pi}^{\pi} 1 dx = 2\pi.$$

Thus, in either case we get (2.2). We can obtain the identity (2.3) by subtracting the equations (2.5) and (2.6) and then integrating as in the previous example. Equation (2.4) follows from $\cos(nx)\sin(nx)$ being an odd function and integrating over $[-\pi, \pi]$ we get 0 for all integers n and k.

Now we use the orthogonality conditions (2.2)-(2.4) to obtain the Fourier coefficients: a_n, b_n, and a_0. We begin with equation (2.1):

$$f(x) = a_0 + \sum_{k=1}^{\infty} (a_k \cos kx + b_k \sin kx).$$

To find a_n, we multiply both sides by $\cos(nx)/\pi$ and integrate over $[-\pi, \pi]$ to yield

$$\frac{1}{\pi} \int_{-\pi}^{\pi} f(x) \cos(nx) dx = \frac{1}{\pi} \int_{-\pi}^{\pi} \left(a_0 + \sum_{k=1}^{\infty} (a_k \cos kx + b_k \sin kx \right) \cos(nx) dx.$$

From the orthogonality conditions (2.2)-(2.4), only the cosine terms with $n = k$ remain on the right side, and we obtain

$$\frac{1}{\pi} \int_{-\pi}^{\pi} f(x) \cos(nx) dx = a_n \quad n \geq 1.$$

Similarly, we can multiply equation (2.1) by $\sin(nx)$ and integrating over $[-\pi, \pi]$ to result in b_n as follows

$$\frac{1}{\pi}\int_{-\pi}^{\pi} f(x)\sin(nx)dx = b_n \quad n \geq 1.$$

The last coefficient a_0 follows from integrating (2.1) in the following way

$$\frac{1}{2\pi}\int_{-\pi}^{\pi} f(x) = \frac{1}{2\pi}\int_{-\pi}^{\pi}\left(a_0 + \sum_{k=1}^{\infty}(a_k\cos kx + b_k \sin kx)\right)dx.$$

Each sine and cosine term integrates to zero and therefore we have

$$\frac{1}{2\pi}\int_{-\pi}^{\pi} f(x) = \frac{1}{2\pi}\int_{-\pi}^{\pi} a_0 dx = a_0.$$

Hence, we have defined the Fourier series of a function f and its Fourier coefficients a_n, b_n, and a_0. We can restate the Fourier series of a function f by recalling Euler's formula

$$e^{it} = \cos t + i \sin t.$$

Therefore, we can write the complex form of the Fourier series of f as follows

$$f(x) = \sum_{n=-\infty}^{\infty} c_n e^{inx} \tag{2.7}$$

where c_n is

$$c_n = \frac{1}{2\pi}\int_{-\pi}^{\pi} f(x)e^{-inx}\,dx$$

for $n \in \mathbb{Z}$.

For a periodic function f on an interval $[-L, L]$, we can write the Fourier series of f by a simple change of variable to integrate on $[-L, L]$ rather than $[-\pi, \pi]$. So now we are able to write the Fourier series of f that is periodic on the interval $[-L, L]$ as

$$f(x) = a_0 + \sum_{n=1}^{\infty}\left(a_n \cos\left(\frac{n\pi x}{L}\right) + b_n \sin\left(\frac{n\pi x}{L}\right)\right)$$

where the Fourier coefficients are defined as

$$a_0 = \frac{1}{2L}\int_{-L}^{L} f(x)dx$$

$$a_n = \frac{1}{L}\int_{-L}^{L} f(x)\cos\left(\frac{n\pi x}{L}\right)dx$$

$$b_n = \frac{1}{L}\int_{-L}^{L} f(x)\sin\left(\frac{n\pi x}{L}\right)dx$$

for $n = 1, 2, 3, \ldots$.

Similarly, we are able to write the complex form of the Fourier series of f that is periodic on the interval $[-L, L]$ as

$$f(x) = \sum_{n=-\infty}^{\infty} c_n e^{\frac{inx}{L}}$$

where c_n is

$$c_n = \frac{1}{2L}\int_{-L}^{L} f(x)e^{\frac{-inx}{L}}\,dx$$

for $n \in \mathbb{Z}$.

The next figure shows an example of a function f and its Fourier series for five and ten partial sums. Notice that the larger the number of coefficients we have the better approximation we get on average.

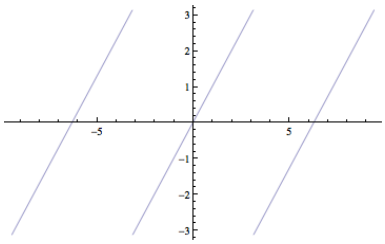

Figure 1: Periodic function f

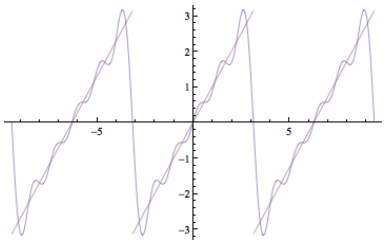

Figure 2: Partial sum (S_5)

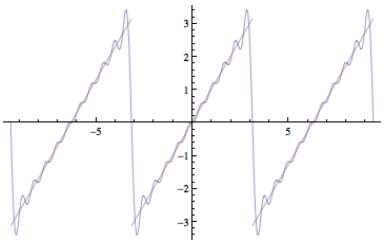

Figure 3: Partial sum (S_{10})

We now introduce the idea of convergence for Fourier series in the L^2-Norm. The space of square-integrable functions will be denoted by: $L^2(\mathbb{R})$. The $L^2(\mathbb{R})$ space has the following inner product:

$$< f, g >= \int_{\mathbb{R}} f(x)\overline{g(x)}dx \quad \text{for } f, g \in L^2.$$

The norm $\|f\|_2$ is therefore defined as

$$\|f\|_2 = \int_{\mathbb{R}} f(x)\overline{f(x)}dx = \int_{\mathbb{R}} |f(x)|^2 dx.$$

We say that a sequence of functions (f_n) converges in the L^2-norm to f if and only if

$$\lim_{n\to\infty} \|f - f_n\|_2 = 0$$

Since $L^2 \subset L^1$, then we can also obtain the Fourier coefficients for a function $f \in L^2$. We follow the approach of Deitmar [3] and introduce the following theorem, known as Parseval's theorem, which states

that the Fourier series of a function f converges to f in the L^2-Norm.

Theorem 1. *Let $f : \mathbb{R} \to \mathbb{C}$ be periodic and Lebesgue integrable on [0,1]. Then the Fourier series of f converges to f in L^2-norm. If c_k denotes the Fourier coefficients of f, then*

$$\sum_{-\infty}^{\infty} |c_k|^2 = \int_0^1 |f(x)|^2 \, dx.$$

Proof. We let $f = u + iv$ so that we break up f into its real and imaginary parts. The partial sums of the Fourier series of f satisfy $S_n(f) = S_n(u) + iS_n(v)$. So for the Fourier series of f to converge in the L^2-norm, it suffices to show that the Fourier series of u and v converge to u and v respectively in the L^2-norm. Thus, if we show the case of a real valued function then the claim will hold true in the complex case as well. Let $\epsilon > 0$. Since f is Lebesgue integrable, there are simple functions ψ, φ defined on $[0, 1]$ such that

$$-1 \leq \varphi \leq f \leq \psi \leq 1 \tag{2.8}$$

and

$$\int_0^1 (\psi(x) - \varphi(x))dx \leq \frac{\epsilon^2}{8}$$

We define a function $g = f - \varphi$. From equation (2.3), by subtracting φ we obatin $g \geq 0$ and

$$|g|^2 \leq |\psi - \varphi|^2 \leq 2(\psi - \varphi).$$

Next, we then have

$$\int_0^1 |g(x)|^2 dx \leq 2 \int_0^1 (\psi(x) - \varphi(x))dx \leq \frac{\epsilon^2}{4}. \tag{2.9}$$

From the definition of g, if we solve for f and write the partial sums $S_n(f)$, we get

$$S_n(f) = S_n(\varphi) + S_n(g).$$

Since φ is a simple function defined on $[0, 1]$, then its Fourier series converges to φ in the L^2-norm. Therefore, there is an $n_0 \geq 0$ such that,

$$\|\varphi - S_n(\varphi)\|_2 \leq \frac{\epsilon}{2}.$$

Since g is the difference of two integrable functions it is also integrable. Therefore, we have

$$\|g - S_n(g)\|_2{}^2 \leq \|g\|^2 \leq \frac{\epsilon^2}{4}.$$

where the last inequality is from equation (2.4). Therefore,

$$\|g - S_n(g)\|_2 \leq \frac{\epsilon}{2}$$

Hence, for $n \geq n_0$,

$$\|f - S_n(f)\|_2 \leq \|\varphi - S_n(\varphi)\|_2 + \|g - S_n(g)\|_2 \leq \frac{\epsilon}{2} + \frac{\epsilon}{2} = \epsilon.$$

□

2.2 Convolution

The concept of convolution is central to Fourier Analysis. Convolution can be thought of as the integration (area) of overlap between one function as it is shifted over another function.

Definition 2.1 We define the convolution of two functions, or convolution product, $f, g \in L^2(\mathbb{R})$ as

$$(f * g)(x) = \int_{\mathbb{R}} f(y)g(x-y)dy.$$

For $f, g, h \in L^2(\mathbb{R})$ the following properties of convolution hold:

- $(f * g)(x) = (g * f)(x)$
- $(f * g) * h = f * (g * h)$
- $f * (g + h) = f * g + f * h$.

Proof. The proof of the first property follows from a substitution of $y \to x - y$ yielding

$$(f * g)(x) = \int_{\mathbb{R}} f(y)g(x-y)dy = \int_{\mathbb{R}} f(x-y)g(y)dy = (g * f)(x).$$

The second property follows from the fact that all the integrals converge absolutely Rudin [7]. Thus,

we are allowed to change the order of integration in

$$\begin{aligned} f * (g * h)(x) &= \int_{\mathbb{R}} f(y) \int_{\mathbb{R}} g(z)h(x-y-z)dzdy \\ &= \int_{\mathbb{R}} g(z) \int_{\mathbb{R}} f(y)h(x-y-z)dydz \\ &= \int_{\mathbb{R}} \int_{\mathbb{R}} f(y)g(z-y)h(x-z)dzdy \\ &= (f * g) * h(x). \end{aligned}$$

The last property is immediate from the linearity of integrals. □

2.3 Fourier Transform

In the previous section, we defined the Fourier series of a function $f \in L^2([-\pi, \pi])$ as

$$f(x) = \sum_{n=-\infty}^{\infty} c_n e^{inx}$$

where c_n is

$$c_n = \frac{1}{2\pi} \int_{-\pi}^{\pi} f(x)e^{-inx}\, dx \text{ for } n \in \mathbb{Z}.$$

The motivation for the Fourier transform is whether we can formulate a similar representation for functions f defined on the entire real line and not necessarily periodic.

Definition 2.2 We then define the Fourier transform of $f \in L^2(\mathbb{R})$ by

$$\hat{f}(\xi) = \frac{1}{\sqrt{2\pi}} \int_{\mathbb{R}} f(x)e^{-ix\xi}dx.$$

The advantage of the Fourier transform is moves us from the time-domain to the representation in the frequency-domain. In mathematics, physics, and engineering this tool is very useful. For example, for a signal that is sampled over some duration in the time-domain, running a Fourier transform will show us the frequencies that make up the signal in the frequency-domain.

For $f \in L^2(\mathbb{R})$ and $a \in \mathbb{R}$, we list some properties and consequences of the Fourier transform:

- If $g(x) = f(x-a)$, then $\hat{g}(\xi) = \hat{f}(\xi)e^{-ia\xi}$
- If $g(x) = f(x)e^{-iax}$, then $\hat{g}(\xi) = \hat{f}(\xi - a)$
- If $g(x) = f(x\lambda)$ for $\lambda > 0$, then $\hat{g}(\xi) = \frac{1}{\lambda}\hat{f}(\frac{\xi}{\lambda})$
- If $g \in L^2(\mathbb{R})$ and $h = f * g$, then $\hat{h}(\xi) = \hat{f}(\xi)\hat{g}(\xi)$.

Proof. The first and second properties can be viewed as a translation rule for the Fourier Transform. We prove the first by letting $g(x) = f(x-a)$ and applying the definition of the transform we have

$$\begin{aligned}
\hat{g}(\xi) &= \frac{1}{\sqrt{2\pi}} \int_{\mathbb{R}} g(x) e^{-ix\xi} dx \\
&= \frac{1}{\sqrt{2\pi}} \int_{\mathbb{R}} f(x-a) e^{-ix\xi} dx \\
&= \frac{1}{\sqrt{2\pi}} \int_{\mathbb{R}} f(t) e^{-i(t+a)\xi} dt \\
&= e^{-ia\xi} \hat{f}(\xi).
\end{aligned}$$

Notice we used the substitution $x - a \to t$ and pulled out the constant $e^{-ia\xi}$ to prove this first property. The second property is proved in a similar manner.

The third property is called the dilation rule for the Fourier Transform. Similarly, by the definition and having $g(x) = f(x\lambda)$ we yield

$$\begin{aligned}
\hat{g}(\xi) &= \frac{1}{\sqrt{2\pi}} \int_{\mathbb{R}} g(x) e^{-ix\xi} dx \\
&= \frac{1}{\sqrt{2\pi}} \int_{\mathbb{R}} f(x\lambda) e^{-ix\xi} dx \\
&= \frac{1}{\sqrt{2\pi}} \frac{1}{\lambda} \int_{\mathbb{R}} f(t) e^{-it/\lambda\xi} dt \\
&= \frac{1}{\lambda} \hat{f}(\frac{\xi}{\lambda}).
\end{aligned}$$

The last property also follows from the fact that we are able to change the order of integration by Fubini's Theorem Rudin [7] and use a substitution of $x - z \to y$.

$$\begin{aligned}
\hat{h}(\xi) &= \frac{1}{\sqrt{2\pi}} \int_{\mathbb{R}} (f * g)(x) e^{-ix\xi} dx \\
&= \frac{1}{\sqrt{2\pi}} \int_{\mathbb{R}} \int_{\mathbb{R}} f(x-z) g(z) dz e^{-ix\xi} dx \\
&= \frac{1}{\sqrt{2\pi}} \int_{\mathbb{R}} \int_{\mathbb{R}} f(x-z) e^{-ix\xi} dx g(z) dz \\
&= \frac{1}{\sqrt{2\pi}} \int_{\mathbb{R}} \int_{\mathbb{R}} f(y) e^{-it\xi} dy g(z) dz \\
&= \hat{f}^(\xi) \hat{g}(\xi)
\end{aligned}$$

□

When we apply the Fourier transform one might be interested in recovering the original function f. The next theorem illustrates that the Fourier Transform is inverse to itself by a sign switch. We follow the approach of Rudin [7] and state the following propositions first.

We need what we call an *auxiliary* function $h_\lambda(x)$. For $\lambda > 0$ and $x \in \mathbb{R}$ we define the function as follows:

$$h_\lambda(x) = \int_{-\infty}^{\infty} e^{-\lambda|t|} e^{2\pi i t x} dt.$$

Proposition 1. *For our auxiliary function we have*

$$h_\lambda(x) = \frac{2\lambda}{4\pi^2 x^2 + \lambda^2} \text{ and } \int_{-\infty}^{\infty} h_\lambda(x) = 1.$$

Proof. We split the integral of our auxiliary function as follows

$$\begin{aligned} h_\lambda(x) &= \int_{-\infty}^{0} e^{2\pi i t x + \lambda t} dt + \int_{0}^{\infty} e^{2\pi i t x - \lambda t} dt \\ &= \frac{e^{2\pi i t x + \lambda t}}{2\pi i x + \lambda}\Big|_{-\infty}^{0} + \frac{e^{2\pi i t x - \lambda t}}{2\pi i x - \lambda}\Big|_{0}^{\infty} \\ &= \frac{1}{\lambda + 2\pi i x} + \frac{1}{\lambda - 2\pi i x} \\ &= \frac{2\lambda}{\lambda^2 + 4\pi^2 x^2}. \end{aligned}$$

To show that the auxiliary function has integral 1, we compute

$$\begin{aligned} \int_{-\infty}^{\infty} h_\lambda(x) dx &= \frac{2}{\lambda} \int_{-\infty}^{\infty} \frac{1}{1 + (2\pi x/\lambda)^2} dx \\ &= \frac{1}{\pi} \int_{-\infty}^{\infty} \frac{1}{1 + x^2} dx \\ &= 1. \end{aligned}$$

□

Proposition 2. *If $f \in L^1(\mathbb{R})$, then for $\lambda > 0$,*

$$f * h_\lambda(x) = \int_{-\infty}^{\infty} e^{-\lambda|t|} \hat{f}(t) e^{ixt} dt.$$

Proof. We compute

$$\begin{aligned} f * h_\lambda(x) &= \int_{-\infty}^{\infty} f(y) h_\lambda(x-y) dy \\ &= \int_{-\infty}^{\infty} f(y) \int_{-\infty}^{\infty} e^{-\lambda|t|} e^{it(x-y)} dt dy \\ &= \int_{-\infty}^{\infty} e^{-\lambda|t|} e^{ixt} \int_{-\infty}^{\infty} f(y) e^{-iyt} dy dt \\ &= \int_{-\infty}^{\infty} e^{-\lambda|t|} \hat{f}(t) e^{ixt} dt. \end{aligned}$$

□

Proposition 3. *For $f \in L^\infty(\mathbb{R})$ and $x \in \mathbb{R}$ we have*

$$\lim_{\lambda \to 0} f * h_\lambda(x) = f(x).$$

Proof. Since the auxiliary function has integral 1, we calculate

$$\begin{aligned} f * h_\lambda(x) - f(x) &= \int_{-\infty}^{\infty} f(y) h_\lambda(x-y) dy - \int_{-\infty}^{\infty} f(x) h_\lambda(y) dy \\ &= \int_{-\infty}^{\infty} (f(x-y) - f(x)) h_\lambda(y) dy \\ &= \int_{-\infty}^{\infty} (f(x-y) - f(x)) \frac{1}{\lambda} h_1(y/\lambda) dy \\ &= \int_{-\infty}^{\infty} (f(x-\lambda y) - f(x)) h_1(y) dy. \end{aligned}$$

Notice the last integrand is dominated by $2\|f\|_\infty h_1(y)$ and converges pointwise for every y, as $\lambda \to 0$. Hence, by the dominated convergence theorem Royden [5] the proposition follows. □

Theorem 2. *(Inversion Formula) Let $f \in L^2(\mathbb{R})$ and assume $\hat{f}$ also in $L^2(\mathbb{R})$, then for every $x \in \mathbb{R}$,*

$$\widehat{\hat{f}}(x) = f(-x)$$

or equivalently

$$f(x) = \frac{1}{\sqrt{2\pi}} \int_{\mathbb{R}} \hat{f}(\xi) e^{i\xi x} d\xi.$$

Proof. By Proposition 2, we have for $\lambda > 0$

$$f * h_\lambda(x) = \int_{-\infty}^{\infty} e^{-\lambda|t|} \hat{f}(t) e^{ixt} dt.$$

We notice that the integrand on the right hand side is bounded by $|\hat{f}(t)|$. Since $h_\lambda(y) \to 1$ as $\lambda \to 0$, the right side then converges to $f(x)$, for every $x \in \mathbb{R}$, by the dominated convergence theorem Royden [5]. □

We conclude this section with Plancherel's Theorem. The theorem states that the norm of $f \in L^2(\mathbb{R})$ is the same as the norm of $\hat{f} \in L^2(\mathbb{R})$. We follow the approach of Ruch and Van Fleet [6].

Theorem 3. *(Plancherel's Theorem) For $f \in L^2(\mathbb{R})$ we have that*

$$\|f\|_2 = \|\hat{f}\|_2.$$

Proof. We have the following inner product

$$\begin{aligned}
\|\hat{f}\|_2 &= \int_{\mathbb{R}} \hat{f}(\xi)\overline{\hat{f}(\xi)} d\xi \\
&= \frac{1}{\sqrt{2\pi}} \int_{\xi\in\mathbb{R}} \hat{f}(\xi) \left(\overline{\int_{t\in\mathbb{R}} f(t) e^{-i\xi t} dt} \right) d\xi \\
&= \frac{1}{\sqrt{2\pi}} \int_{\xi\in\mathbb{R}} \hat{f}(\xi) \left(\int_{t\in\mathbb{R}} \overline{f(t) e^{-i\xi t}} dt \right) d\xi \\
&= \frac{1}{\sqrt{2\pi}} \int_{\xi\in\mathbb{R}} \hat{f}(\xi) \left(\int_{t\in\mathbb{R}} \overline{f(t) e^{i\xi t} dt} \right) d\xi \\
&= \int_{t\in\mathbb{R}} \overline{f(t)} \left(\frac{1}{\sqrt{2\pi}} \int_{\xi\in\mathbb{R}} \hat{f}(\xi) e^{i\xi t} d\xi \right) dt \\
&= \int_{t\in\mathbb{R}} \overline{f(t)} f(t) dt \\
&= \|f\|_2
\end{aligned}$$

Note that we have used Fubini's theorem from Rudin[7] to exchange the order of integration. We have also used the inversion formula to get back the original function in the second to last identity. □

3 Wavelet Analysis

In wavelet analysis, we require two important functions: the scaling function ϕ and the wavelet function ψ. Using these two functions we are able to generate a family of functions that can be used to approximate a function (signal). We then start with some definitions and properties needed to establish these sets of functions. Throughout this section, we follow the ideas of Boggess and Narcowich [1].

3.1 The Multiresolution Framework

Definition 3.1 Let $\{V_j\}, j = ..., -2, -1, 0, 1, 2, ...$ be a sequence of subspaces of functions in $L^2(\mathbb{R})$. The collection of spaces $\{V_j\}_{j\in\mathbb{Z}}$ is called a multi-resolution analysis (MRA) with scaling function ϕ if the following conditions hold:

1. $V_j \subset V_{j+1}$
2. $\overline{\cup V_j} = L^2(\mathbb{R})$
3. $\cap V_j = \{0\}$
4. A function $f(x)$ belongs to V_j if and only if the function $f(2^{-j}x)$ belongs to V_0.
5. The function ϕ belongs to V_0 and the set $\{\phi(x-k)\}_{k\in\mathbb{Z}}$ is an orthonormal basis (in the L^2 sense) for V_0.

Depending on the giving scaling function ϕ we can result with different $\{V_j\}$ spaces. We would want to have scaling functions with compact support and continuity to better analyze decompositions and reconstructions of functions (signals). Given a multiresolution analysis, we have the property that translates of the scaling function are orthonormal. In the following theorem we show that we have a similar result.

Theorem 4. *If $\{V_j\}_{j\in\mathbb{Z}}$ is an MRA with scaling function ϕ. Then for any $j \in \mathbb{Z}$, the set of functions*

$$S = \{\phi_{jk}(x) = 2^{j/2}\phi(2^j x - k)\}_{k\in\mathbb{Z}}$$

forms an orthonormal basis for V_j.

Proof. We must show the set of functions S span V_j. If $f(x) \in V_j$, then by the scaling property of an MRA, we have $f(2^{-j}) \in V_0$. Since the set $\{\phi(x-k)\}_{k\in\mathbb{Z}}$ forms an orthonormal basis for V_0, then $f(2^{-j})$ is a linear combination of $\{\phi(x-k)\}_{k\in\mathbb{Z}}$. Therefore,

$$f(2^{-j}) = \sum_{k\in\mathbb{Z}} p_k\{\phi(x-k)\}_{k\in\mathbb{Z}} \text{ for some } p_k \in \mathbb{R}$$

We let x be $2^j x$ and multiply and divide by $2^{j/2}$ to obtain

$$f(x) = \sum_{k\in\mathbb{Z}} \frac{p_k}{2^{j/2}} 2^{j/2}\phi(2^{j/2}x-k) = \sum_{k\in\mathbb{Z}} \frac{p_k}{2^{j/2}} 2^{j/2}\phi_{jk}(t)$$

So our $p'_k = \dfrac{p_k}{2^{j/2}}$ and we have $f(x) \in V_j$.

Last, we must show that the set S is orthonormal or that

$$< \phi_{jk}, \phi_{jl} >= \delta_{jk} = \begin{cases} 0 & j \neq k \\ & \\ 1 & j = k \end{cases}.$$

This follows from the following calculations.

$$\begin{aligned} < \phi_{jk}, \phi_{jl} > &= \int_{-\infty}^{\infty} 2^{j/2}\phi(2^j x-k)\overline{2^{j/2}\phi(2^j x-l)}dx \\ &= 2^j \int_{-\infty}^{\infty} \phi(2^j x-k)\overline{\phi(2^j x-l)}dx \\ &= \int_{-\infty}^{\infty} \phi(y-k)\overline{\phi(y-l)}dy \\ &= \delta_{kl}. \end{aligned}$$

Here we used a substitution of $y = 2^j x$ in the second to last equation and the last equation is obtained since the set $\{\phi(x-k)\}_{k\in\mathbb{Z}}$ is an orthonormal basis. □

The central equation needed in multiresolution analysis is called the *scaling relation.* It is presented in the following theorem which relates $\phi(x)$ and translates of $\phi(2x)$. One observation we will see later is that if our scaling function ϕ has compact support, then only a finite number of the p_k coefficients of our scaling equation will be nonzero.

Theorem 5. *If $\{V_j\}_{j\in\mathbb{Z}}$ is an MRA with scaling function ϕ, then we have the following scaling relation:*

$$\phi(x) = \sum_{k\in\mathbb{Z}} p_k\phi(2x-k)$$

where

$$p_k = \sqrt{2}\int_{-\infty}^{\infty} \phi(x)\overline{\phi(2x-k)}dx.$$

Proof. Since the spaces $\{V_J\}$ are nested, then for $\phi(x) \in V_0$ we have $\phi(x) \in V_1$. From the previous theorem we have that $\phi(x)$ can be expressed as a linear combination of elements of the set $\{\phi_{1k}(x)\}_{k\in\mathbb{Z}}$.

Therefore, there exists p_k such that

$$\begin{aligned}\phi(x) &= \sum_{k\in\mathbb{Z}} p_k \phi_{1k}(x) \\ &= \sqrt{2}\sum_{k\in\mathbb{Z}} p_k \phi(2x-k)\end{aligned}$$

where from letting

$$p_k =< \phi(x), \phi_{1k}(x) >= \sqrt{2}\int_{-\infty}^{\infty} \phi(x)\overline{\phi(2x-k)}dx$$

we have established the scaling equation. □

Now that we have established the scaling equation we have the following property of the coefficients p_k. The following theorem shows the important result which is necessary of our coefficients p_k to satisfy.

Theorem 6. *If $\{V_j\}_{j\in\mathbb{Z}}$ is an MRA with scaling function ϕ, then the following holds:*

1. $\sum_{k\in\mathbb{Z}} p_k = \sqrt{2}$
2. $\sum_{k\in\mathbb{Z}} p_k p_{k-2l} = \delta_{0l}$
3. $\sum_{k\in\mathbb{Z}} p_k^2 = 1$

Proof. We using the scaling equation from the previous theorem

$$\phi(x) = \sum_{k\in\mathbb{Z}} p_k \phi(2x-k).$$

By integrating both sides of the scaling equation over $\mathbb{R}$ we obtain

$$\begin{aligned}\int_{-\infty}^{\infty}\phi(x)dx &= \int_{-\infty}^{\infty}\sqrt{2}\sum_{k\in\mathbb{Z}} p_k\phi(2x-k)dx \\ &= \sqrt{2}\sum_{k\in\mathbb{Z}} p_k \int_{-\infty}^{\infty}\phi(2x-k)dx \\ &= \frac{\sqrt{2}}{2}\sum_{k\in\mathbb{Z}} p_k \int_{-\infty}^{\infty}\phi(y)dy \\ &= \frac{\sqrt{2}}{2}\sum_{k\in\mathbb{Z}} p_k\end{aligned}$$

Where the last two equations come from letting $(y = 2x-k)$ and $\int\phi = 1$. Hence, we have

$$1 = \int_{-\infty}^{\infty}\phi(x)dx = \frac{\sqrt{2}}{2}\sum_{k\in\mathbb{Z}} p_k$$

which is the desired equation

$$\sum_{k\in\mathbb{Z}} p_k = \sqrt{2}.$$

For the second equation, we begin with our scaling equation again

$$\phi(x) = \sum_{k\in\mathbb{Z}} p_k \phi(2x-k).$$

If we multiple each side of the scaling equation by $\phi_{0,l}(t)$ and integrate both sides over $\mathbb{R}$ we obtain

$$\sum_{k\in\mathbb{Z}} p_k p_{k-2l} = \delta_{0,l}$$

Where we see that the right side is due to our orthonormal set.

For the third equation, we let $l = 0$ in equation (2.) and obtain

$$\sum_{k\in\mathbb{Z}} p_k^2 = 1.$$

which is equation (3.). This concludes the proof. □

3.2 Orthogonality via the Fourier Transform

We recall the definition of the Fourier transform for a function $f \in L^2(\mathbb{R})$ is given by

$$\hat{f}(\xi) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} f(x) e^{-ix\xi} dx.$$

An observation to make is that since we have $\hat{\phi}(0) = \frac{1}{\sqrt{2\pi}} \int \phi(x)dx$, then the normalization condition ($\int \phi = 1$) becomes

$$\hat{\phi}(0) = \frac{1}{\sqrt{2\pi}}.$$

We are then led to translate the orthonormality conditions of ϕ and ψ into equivalent forms of the Fourier transform.

Theorem 7. *A function ϕ satisfies the orthonormality condition if and only if*

$$2\pi \sum_{k\in\mathbb{Z}} |\hat{\phi}(\xi + 2\pi k)|^2 = 1 \text{ for all } \xi \in \mathbb{R}$$

Furthermore, a function $\psi(x)$ is orthogonal to $\phi(x-l)$ for all $l \in \mathbb{Z}$ if and only if

$$\sum_{k\in\mathbb{Z}} \hat{\phi}(\xi+2\pi k)\overline{\hat{\psi}(\xi+2\pi k)} = 0 \text{ for all } \xi \in \mathbb{R}.$$

Proof. We can start by stating the orthonormality condition as

$$\int_{-\infty}^{\infty} \phi(x-k)\overline{\phi(x-l)}dx = \delta_{kl}$$

We substitute $y = x-k$ and let $n = l-k$ to obtain

$$\int_{-\infty}^{\infty} \phi(y)\overline{\phi(y-n)}dy = \delta_{0n}.$$

Using Plancherel's identity(Theorem 3) and the translation property for the Fourier transform we obtain

$$\begin{aligned}
\delta_{0n} &= \int_{-\infty}^{\infty} \phi(y)\overline{\phi(y-n)}dy \\
&= \int_{-\infty}^{\infty} \hat{\phi}(\xi)\overline{\widehat{\phi(y-n)}(\xi)}d\xi \\
&= \int_{-\infty}^{\infty} \hat{\phi}(\xi)\overline{\hat{\phi}(\xi)e^{-in\xi}}d\xi \\
&= \int_{-\infty}^{\infty} |\hat{\phi}(\xi)|^2 e^{in\xi}d\xi.
\end{aligned}$$

So we have now the equation

$$\int_{-\infty}^{\infty} |\hat{\phi}(\xi)|^2 e^{in\xi}d\xi = \delta_{0n}$$

which we can split into intervals of period 2π and obtain

$$\int_{-\infty}^{\infty} |\hat{\phi}(\xi)|^2 e^{in\xi}d\xi = \sum_{j\in\mathbb{Z}} \int_{2\pi j}^{2\pi(j+1)} |\hat{\phi}(\xi)|^2 e^{in\xi}d\xi = \delta_{0n}.$$

Now replacing ξ by $\xi + 2\pi j$. Therefore, the limits of integration change from 0 to 2π to obtain

$$\begin{aligned}
\sum_{j\in\mathbb{Z}} \int_{2\pi j}^{2\pi(j+1)} |\hat{\phi}(\xi)|^2 e^{in\xi}d\xi &= \int_0^{2\pi} \sum_{j\in\mathbb{Z}} |\hat{\phi}(\xi+2\pi j)|^2 e^{in(\xi+2\pi j)}d\xi \\
&= \int_0^{2\pi} \sum_{j\in\mathbb{Z}} |\hat{\phi}(\xi+2\pi j)|^2 e^{in\xi}(e^{2\pi ij})^n d\xi \\
&= \int_0^{2\pi} \sum_{j\in\mathbb{Z}} |\hat{\phi}(\xi+2\pi j)|^2 e^{in\xi}d\xi.
\end{aligned}$$

So our equation now becomes

$$\int_0^{2\pi} \sum_{j\in\mathbb{Z}} |\hat{\phi}(\xi + 2\pi j)|^2 e^{in\xi} d\xi = \delta_{0n}.$$

If we let

$$F(\xi) = 2\pi \sum_{j\in\mathbb{Z}} |\hat{\phi}(\xi + 2\pi j)|^2.$$

we notice F is 2π-periodic since

$$\begin{aligned} F(\xi + 2\pi) &= 2\pi \sum_{j\in\mathbb{Z}} |\hat{\phi}(\xi + 2\pi(j+1))|^2 \\ &= 2\pi \sum_{m\in\mathbb{Z}} |\hat{\phi}(\xi + 2\pi m)|^2 \\ &= F(\xi). \end{aligned}$$

Since F is 2π-periodic it has a Fourier series

$$F(x) = \sum_{n=-\infty}^{\infty} c_n e^{inx}$$

where the Fourier coefficients are given by

$$c_n = \frac{1}{2\pi} \int_0^{2\pi} F(\xi) e^{-in\xi} d\xi.$$

So the orthonormality condition is now equivalent to $c_{-n} = \delta_{0n}$ In other words if $n = 0$ we obtain

$$c_0 = \frac{1}{2\pi} \int_0^{2\pi} F(\xi) d\xi = 1.$$

Therefore, $F(\xi) = 1$ or equivalently

$$2\pi \sum_{k\in\mathbb{Z}} |\hat{\phi}(\xi + 2\pi k)|^2 = 1.$$

□

Next, we translate the scaling condition $\phi(x) = \sum_{k\in\mathbb{Z}} p_k \phi(2x - k)$ in terms of the Fourier transform.

Theorem 8. *The scaling condition* $\phi(x) = \sum_{k\in\mathbb{Z}} p_k \phi(2x - k)$ *is equivalent to*

$$\hat{\phi}(\xi) = \hat{\phi}(\xi/2) P(e^{-i\xi/2})$$

where the polynomial P *is defined as* $P(z) = \frac{1}{2} \sum_{k\in\mathbb{Z}} p_k z^k$.

Proof. We begin with the scaling equation

$$\phi(x) = \sum_{k\in\mathbb{Z}} p_k \phi(2x-k)$$

and take the transform of both sides

$$\begin{aligned}
\hat{\phi}(x) &= \sum_{k\in\mathbb{Z}} p_k \widehat{\phi(2x-k)} \\
&= \frac{1}{2}\sum_{k\in\mathbb{Z}} p_k e^{-ik(\xi/2)}\hat{\phi}(\xi/2) \\
&= \hat{\phi}(\xi/2)P(e^{-i\xi/2})
\end{aligned}$$

where we have used the translation property of the Fourier transform and thus conclude the proof. □

Now having seen the results of Theorem 7 and 8, we can combine the necessary conditions on the polynomial $P(z)$ to result in a multiresolution analysis.

Theorem 9. *Given a function ϕ that satisfies the orthonormality condition and the scaling condition, then the polynomial $P(z) = \sum_{k\in\mathbb{Z}} p_k z^k$ satisfies the following equation:*

$$|P(z)|^2 + |P(-z)|^2 = 1 \text{ for } z \in \mathbb{C} \text{ with } |z| = 1.$$

Proof. Since ϕ satisfies the orthonormality condition, we have

$$\sum_{k\in\mathbb{Z}} |\hat{\phi}(\xi + 2\pi k)|^2 = \frac{1}{2\pi} \text{ for all } \xi \in \mathbb{R}$$

and we also have ϕ satisfying the scaling condition

$$\hat{\phi}(\xi) = \hat{\phi}(\xi/2)P(e^{-i\xi/2}).$$

We begin by splitting the sum into even and odd indices and use the scaling equation:

$$\begin{aligned}
\frac{1}{2\pi} &= \sum_{k\in\mathbb{Z}} |\hat{\phi}(\xi+2\pi k)|^2 \\
&= \sum_{l\in\mathbb{Z}} |\hat{\phi}(\xi+(2l)2\pi)|^2 + \sum_{l\in\mathbb{Z}} |\hat{\phi}(\xi+(2l+1)2\pi)|^2 \\
&= \sum_{l\in\mathbb{Z}} (|P(e^{-i(\xi/2+2l\pi)}|^2 |\hat{\phi}(\xi/2+(2l)\pi)|^2) + \sum_{l\in\mathbb{Z}} (|P(e^{-i(\xi/2+(2l+1)\pi)}|^2 |\hat{\phi}(\xi/2+(2l+1)\pi)|^2) \\
&= |P(e^{-i\xi/2})|^2 \sum_{l\in\mathbb{Z}} |\hat{\phi}(\xi/2+2\pi l)|^2 + |P(-e^{-i\xi/2})|^2 \sum_{l\in\mathbb{Z}} |\hat{\phi}((\xi/2+\pi)+2\pi l)|^2 \\
&= |P(e^{-i\xi/2})|^2 \frac{1}{2\pi} + |P(-e^{-i\xi/2})|^2 \frac{1}{2\pi}
\end{aligned}$$

where we used the fact that $e^{-2\pi li} = 1$ and $e^{-\pi i} = -1$. Therefore our equation now becomes

$$1 = |P(e^{-i\xi/2})|^2 + |P(-e^{-i\xi/2})|^2.$$

Since this equation holds for all $\xi \in \mathbb{R}$, we conclude

$$|P(z)|^2 + |P(-z)|^2 = 1 \text{ for all } z \in \mathbb{C} \text{ with } |z| = 1.$$

□

Given a multiresolution analysis with scaling function ϕ and satisfying the scaling condition, we have the associated wavelet function ψ satisfying the following equation:

$$\psi(x) = \sum_{k\in\mathbb{Z}} (-1)^k p_{1-k} \phi(2x - k).$$

We then let $Q(z) = -zP(z)$ and get a similar result for the scaling condition as in Theorem 8 and obtain

$$\hat{\psi}(\xi) = \hat{\phi}(\xi/2) Q(e^{-i\xi/2}).$$

Therefore, we state the following theorem, which is analogous to Theorem 9, which associates the function ψ and its associated polynomial $Q(z)$.

Theorem 10. *Let ϕ satisfy the orthonormality and scaling conditions with $\psi(x) = \sum_{k\in\mathbb{Z}} q_k \phi(2x - k)$. Having $Q(z) = \sum_{k\in\mathbb{Z}} q_k z^k$, then the following is satisfied:*

$$P(z)\overline{Q(z)} + P(-z)\overline{Q(-z)} = 0$$

for $|z| = 1$.

The proof is similar to the previous theorem or you can see the result in Bogges and Narcowich [1]. We are able to combine the results of the previous two theorems and state the following result using the matrix

$$M = \begin{bmatrix} P(z) & P(-z) \\ Q(z) & Q(-z) \end{bmatrix}.$$

The result is that the matrix M associated with an orthornormal scaling function and orthonormal wavelet must be unitary, i.e. $M \cdot M^* = I$, so that we have the results from the from the previous theorems

stated by the identity. So our matrix M^* will be of the form

$$M^* = \begin{bmatrix} \overline{P(z)} & \overline{Q(z)} \\ \overline{P(-z)} & \overline{Q(-z)} \end{bmatrix}.$$

Note that in Theorem 9, having a scaling function ϕ, its polynomial $P(z)$ must satisfy the equation $|P(z)|^2 + |P(-z)|^2 = 1$ for $z \in \mathbb{C}$ with $|z| = 1$. If we wanted to construct the scaling function ϕ, we can define the polynomial $P(z)$ that satisfies the equation $|P(z)|^2 + |P(-z)|^2 = 1$ and make our construction based on the coefficients.

Theorem 11. *Let* $P(z) = \frac{1}{2}\sum_{k\in\mathbb{Z}} p_k z^k$ *that satisfies:*

- $P(1) = 1$
- $|P(z)|^2 + |P(-z)|^2 = 1$ *for* $|z| = 1$
- $|P(e^{it})| > 0$ *for* $|t| \leq \pi/2$

with $\phi_0(x)$ *the Haar scaling function and* $\phi_n(x) = \sum_{k\in\mathbb{Z}} p_k \phi_{n-1}(2x-k)$ *for* $n \geq 1$. *Then the sequence* ϕ_n *converges pointwise and in* L^2 *to a function* ϕ *which satisfies the orthonormality condition and scaling equation.*

Proof. The proof of convergence is proven by showing that $\hat{\phi}_n$ converges uniformly on compact subsets of $\mathbb{R}$. Then, you show that ϕ_n converges in the L^2 norm. Last, since each of the ϕ_n and its translates are also orthonormal, then the same must be true of its L^2 limit. The proof of convergence is explained in detail by Boggess and Narcowich [1]. Once convergence has been established, we need ϕ to satisfy the scaling equation and we then show it satisfies the the orthonormality and normalization conditions as well. We start with ϕ_1 and by the scaling equation we have

$$\phi_1(x) = \sum_{k\in\mathbb{Z}} p_k \phi_0(2x-k).$$

This is equivalent to the equation in the transform domain:

$$\hat{\phi}_1(\xi) = P(e^{-\xi/2})\hat{\phi}_0(\xi/2).$$

We have

$$\hat{\phi}_0(0) = \frac{1}{\sqrt{2\pi}} \int_0^1 \phi_0(x) e^0 dx = \frac{1}{\sqrt{2\pi}}$$

and $P(1) = 1$. Therefore, for ϕ_1 we also obtain

$$\hat{\phi}_1(0) = P(e^0)\hat{\phi}_0(0) = \frac{1}{\sqrt{2\pi}}$$

so that ϕ_1 also satisfies the normalization condition as well. Using an inductive process, we are able to establish that if ϕ_{n-1} satisfies the normalization condition, then ϕ_n satisfies the same condition.

Notice by the scaling equation and the properties of the transform we have

$$\sum_{k\in\mathbb{Z}} |\hat{\phi}_1(\xi + 2\pi k)|^2 = \sum_{k\in\mathbb{Z}} |P(e^{-i\xi/2+\pi ki})|^2 |\hat{\phi}_0(\xi/2 + \pi k)|^2.$$

For the normalization condition, we assume ϕ_0 satisfies the orthonormality condition and begin by splitting the sum of ϕ_1 into even and odd indices

$$\begin{aligned}
\sum_{k\in\mathbb{Z}} |\hat{\phi}_1(\xi + 2\pi k)|^2 &= \sum_{l\in\mathbb{Z}} \left(|P(e^{-i(\xi/2+2l\pi)}|^2 |\hat{\phi}_0(\xi/2 + (2l)\pi)|^2 \right) + \sum_{l\in\mathbb{Z}} \left(|P(e^{-i(\xi/2+(2l+1)\pi)}|^2 |\hat{\phi}_0(\xi/2 + (2l+1)\pi)|^2 \right) \\
&= |P(e^{-i\xi/2})|^2 \sum_{l\in\mathbb{Z}} |\hat{\phi}_0(\xi/2 + 2\pi l)|^2 + |P(-e^{-i\xi/2})|^2 \sum_{l\in\mathbb{Z}} |\hat{\phi}_0((\xi/2 + \pi) + 2\pi l)|^2 \\
&= |P(e^{-i\xi/2})|^2 \frac{1}{2\pi} + |P(-e^{-i\xi/2})|^2 \frac{1}{2\pi} \\
&= \frac{1}{2\pi} \left(|P(e^{-i\xi/2})|^2 + |P(-e^{-i\xi/2})|^2 \right) \\
&= \frac{1}{2\pi}.
\end{aligned}$$

Thus, ϕ_1 satisfies the orthonormality condition. Similarly, we can establish when ϕ_{n-1} satisfies the orthonormality condition, then ϕ_n will also satisfy the orthonormality condition. By an inductive process, we can conclude that ϕ_n will satisfy the conditions of the orthonormality and scaling equation for all n. If we take the limit as $n \to \infty$ we will get what we call the limiting function, ϕ, that follows the conditions of orthonormality and scaling as we wanted. □

3.3 Examples

Haar(D2)

We first present the example of the Haar wavelet. The coefficients are $p_0 = p_1 = 1$ and all other $p_k = 0$. Then the polynomial is $P(z) = \frac{1+z}{2}$. We check that it satisfies the conditions:

- $P(1) = \frac{1+1}{2} = 1$
- $|P(z)|^2 + |P(-z)|^2 = 1$ for z $\in \mathbb{C}$ with $|z| = 1$.

$$\begin{aligned} |P(z)|^2 + |P(-z)|^2 &= \frac{|1+z|^2}{4} + \frac{|1-z|^2}{4} \\ &= \frac{1 + 2Re\{z\} + |z|^2}{4} + \frac{1 - 2Re\{z\} + |z|^2}{4} \\ &= 1. \end{aligned}$$

Using the iterative algorithm in the previous theorem starting with the Haar scaling function ϕ_0 we have

$$\begin{aligned} \phi_1(x) &= p_0\phi_0(2x) + p_1\phi_0(2x-1) \\ &= \phi_0(2x) + \phi_0(2x-1) \\ &= \phi_0(x). \end{aligned}$$

Likewise $\phi_2 = \phi_1 = \phi_0$. Thus using the Haar scaling function ϕ_0 in the scaling equation, the iterative method will keep reproducing ϕ_0.

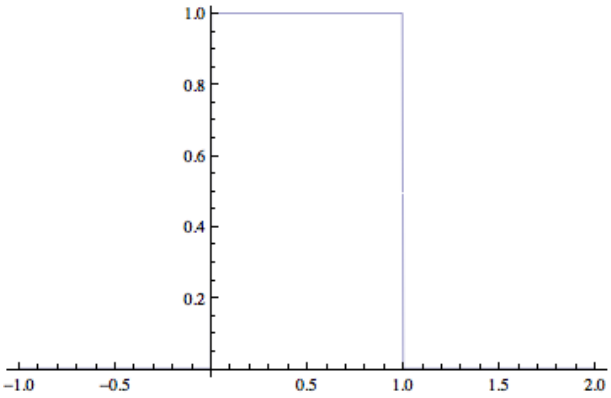

Figure 4: Plot of ϕ_0

Daubechies-4(D4)

A second examples is the D4 wavelet with $P(z) = \frac{1}{2}\sum_{k=0}^{3} p_k z^k$ where the coefficients are given by

$$p_0 = \frac{1+\sqrt{3}}{4}, \qquad p_1 = \frac{3+\sqrt{3}}{4}, \qquad p_2 = \frac{3-\sqrt{3}}{4}, \qquad p_3 = \frac{1-\sqrt{3}}{4}.$$

We begin by checking the conditions:

$$P(1) = \frac{1}{2}\left(\frac{1+\sqrt{3}}{4} + \frac{3+\sqrt{3}}{4} + \frac{3-\sqrt{3}}{4} + \frac{1-\sqrt{3}}{4}\right) = 1.$$

We then use an inductive process to obtain ϕ_1 from the coefficients p_k and the scaling function ϕ_0. Computing we get

$$\begin{aligned}
\phi_1(x) &= \sum_{k=0}^{3} p_k \phi_0(2x-k) \\
&= p_0\phi_0(2x) + p_1\phi_0(2x-1) + p_2\phi_0(2x-2) + p_3\phi_0(2x-3) \\
&= \frac{1+\sqrt{3}}{4}\phi_0(2x) + \frac{3+\sqrt{3}}{4}\phi_0(2x-1) + \frac{3-\sqrt{3}}{4}\phi_0(2x-2) + \frac{1-\sqrt{3}}{4}\phi_0(2x-3).
\end{aligned}$$

We could then compute $\phi_2, \phi_3, \phi_4, ...$ in a recursive matter. We plot the first few iterations of $\phi_1, \phi_2, \phi_4, \phi_6$ below. The more times we apply the inductive algorithm, the closer we get to the limiting function, ϕ, which is the D4 scaling function.

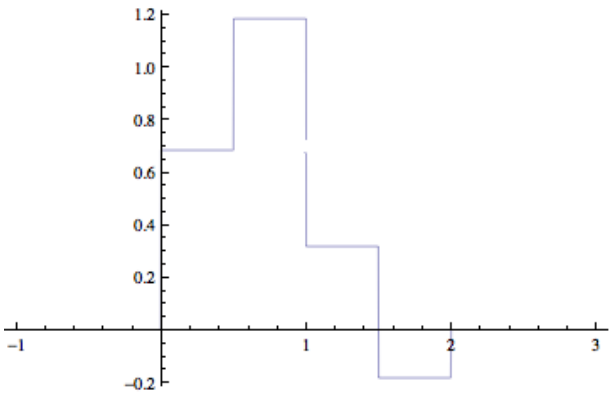

Figure 5: Plot of ϕ_1

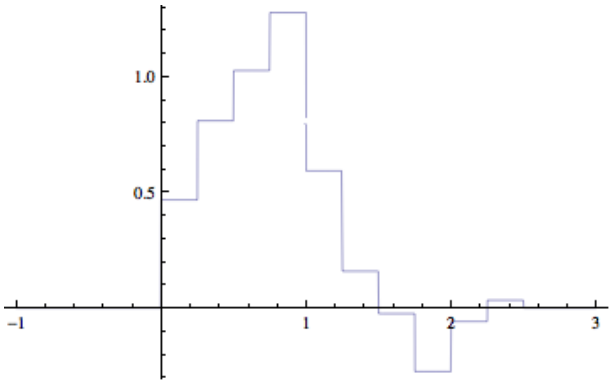

Figure 6: Plot of ϕ_2

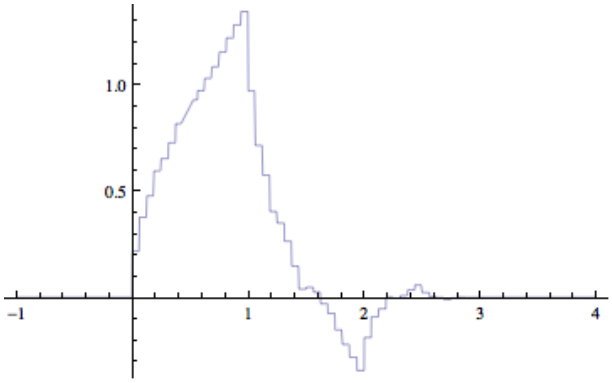

Figure 7: Plot of ϕ_4

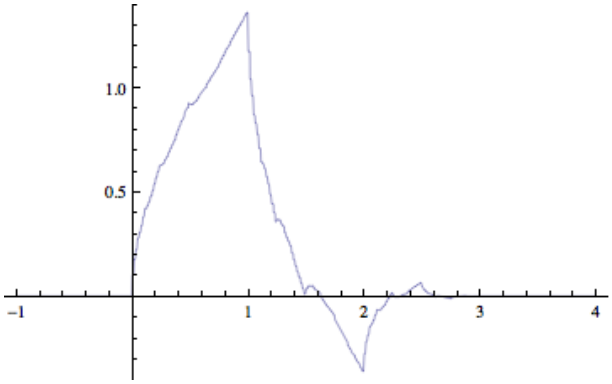

Figure 8: Plot of ϕ_6

4 Factoring Wavelet Transformations

4.1 Lifting

We denote a vector x of length n by

$$\mathbf{x} = (x[0], x[1], x[2], \ldots, x[N-1]).$$

We change the notation to $x[n]$ instead of x_n to emphasize that a signal can be thought of as a function of the parameter n. We recall the condition for an infinite signal to have finite energy being

$$\sum_{n=-\infty}^{\infty} [x[n]]^2 < \infty.$$

Notice that all finite signals will satisfy the above condition and will have finite energy. Before defining the Lifting Method, we first consider an example. Let us consider the length 8 discrete signal

$$\mathbf{x} = (56, 40, 8, 24, 48, 48, 40, 16).$$

We take the first two entries of $\mathbf{x}$, denote them by a and b respectively, and perform the following computations

$$s = \frac{a+b}{2}$$

$$d = a - s$$

i.e. s will be the mean of the two and d the difference of the first entry and s. We continue this process with the next pairs of entries and store their means and differences by order of the means followed by the differences. We implement this with our example signal $\mathbf{x}$.

56	40	8	24	48	48	40	16
48	16	48	28	**8**	**−8**	**0**	**12**
32	38	**16**	**10**	**8**	**−8**	**0**	**12**
35	**−3**	**16**	**10**	**8**	**−8**	**0**	**12**

Figure 9: Mean and difference (**bold**) computations for $\mathbf{x}$.

In the figure above, we have 3 iterations of the lifting method since our signal is of length $\mathbf{x} = 2^3$. The first row represents the original signal and the following row represent the 3 iterations of the lifting method.

It is important to observe that no information has been lost in this transformation of the signal. In fact, we are able to retrieve the signal by the following inverse computations

$$a = s + d$$

$$b = s - d.$$

In Figure 9, for the vector $\mathbf{x}$ the first two entries of the third row can be obtained by the last row as $32 = 35 - (-3)$ and $38 = 35 - (-3)$. Similarly, the first four entries of the second row using the third row we have $48 = 32 + 16$, $16 = 32 - 16$, $48 = 38 + 10$, and $28 = 38 - 10$. The first row is obtained by the same procedure and thus we have retrieved the original signal $\mathbf{x}$. An important observation to make is that once we have calculated s, we no longer need the information for b. Also once we calculate d, we no longer need a. Thus as far as memory is concerned, we can replace the storage of b with s, and a with d.

This transform that uses means and differences brings us to the lifting definition. We can consider the operations, mean and difference, as specific cases of more general operations. If there is some structure to the signal, then when taking two samples their difference will most likely be small. In other words, we can say the first sample is a *prediction* of the second sample. However, we can use other another *prediction* than the one used in the previous example. The other operation we performed with two samples is computing their mean. It is observed that the pairwise mean values contain the overall structure of the signal with only half the number of samples. We will refer to this operation as the *update* process. Similar to the prediction operation, we can also generalize the update operation to more than just calculating pairwise means. As shown in the figure, we outline the steps performed in the previous example.

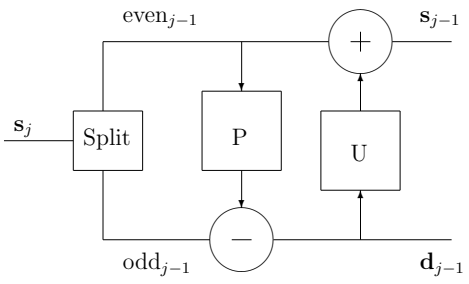

Figure 10: One step of lifting

Given that we start with a signal $\mathbf{s}_j$ of length 2^j, we first transform the signal to two sequences, each of length 2^{j-1}, denoted $\mathbf{s}_{j-1}$ and $\mathbf{d}_{j-1}$. We explain the procedure of the figure by the following three steps:

- **Split** – The entries of the signal are separated into their even and odd components.
- **Prediction** – If our signal contains some structure, we are able to have a prediction of the odd sample based on knowing some information of the even samples. We then replace the odd entry by the correction to the prediction which we noted as the difference in our previous example. Therefore, our prediction step is forming the vector $\mathbf{d}$ by the following computation

$$\mathbf{d}_{j-1} = odd_{j-1} - P(even_{j-1}).$$

  Thus, our transformed signal is composed of an odd sample minus a prediction based on the number of even samples.

- **Update** – We next update our even entry to reflect our knowledge of the signal after performing the prediction step. In our example, we had a the pairwise average as our update step. In general, our update step is based on an even entry plus an update based on the know difference detail. We form the vector $\mathbf{s}$ as follows

$$\mathbf{s}_{j-1} = even_{j-1} + U(d_{j-1}).$$

Here we have performed one iteration of a lifting step. As noted above, if we start with a signal of length 2^j and repeat the lifting step we are allowed to compute up to a j number of times. Our result would be the stored differences from each lifting step $\mathbf{d_j}, \mathbf{d_{j-1}}, ..., \mathbf{d_0}$ and a single number $\mathbf{s}_0[0]$, which is the mean value of all the entries from our original signal. The figure below illustrates two steps of lifting.

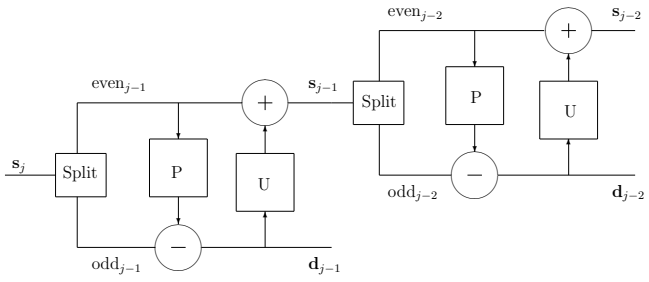

Figure 11: Two lifting steps

We next introduce the Z-transform of a signal. We can think of this similar to the Fourier transform, but for a discrete signal. The Z-transform takes a signal from the time domain into the frequency domain. Since we will be working with *Finite Impulse Response* (FIR) filters, we say a filter h is a (FIR) filter if only a finite number of the filter coefficients are non-zero.

Definition 3.1 The *Z-transform* of a sequence $(x_k) \in \ell^2$ is given by

$$x(z) = \sum_k x_k z^{-k}.$$

We can split the signal $x(z)$ to the components even(x_e) and odd(x_o).

$$\begin{aligned} x(z) &= \sum_k x_{2k} z^{-2k} + \sum_k x_{2k+1} z^{-(2k+1)} \\ &= \sum_k x_{2k} (z^2)^{-k} + z^{-1} \sum_k x_{2k+1} (z^2)^{-k} \\ &= x_e(z^2) + z^{-1} x_o(z^2). \end{aligned}$$

More generally, we can then write the even and odd components as:

$$x_e(z) = \sum_{k \in \mathbb{Z}} x_{2k} z^{-k} \qquad x_o(z) = \sum_{k \in \mathbb{Z}} x_{2k+1} z^{-k}.$$

Similarly, we are able to get the even and odd components from the *Z-transform* as follows:

$$x_e(z^2) = \frac{x(z) + x(-z)}{2} \qquad x_o(z^2) = \frac{x(z) - x(-z)}{2z^{-1}}.$$

4.2 Laurent Polynomials

A *Laurent polynomial* is an extension of the regular polynomials over a certain field with exponents allowed to be any integer. If h is a (FIR) filter, then it follows that the *Z-transform* of h is a Laurent polynomial of the form

$$h(z) = \sum_{k=a}^{b} h_k z^{-k}, \qquad a \leq b \qquad a, b \in \mathbb{Z}$$

where the *degree* of the Laurent polynomial is defined by

$$\deg\ h = |h| = b - a.$$

Note a key difference. As a regular polynomial, z^n, has degree n, while as a Laurent polynomial it

has degree 0. The Laurent polynomials form a commutative ring denoted by $\mathbf{R}[z, z^{-1}]$. We have that the sum or difference of two Laurent polynomials is a Laurent polynomial. The product of two Laurent polynomials with degree l and l' respectively is also a Laurent polynomial with degree $l + l'$.

We apply the Division Algorithm for our (FIR) filters. Thus, if we have two Laurent polynomials $a(z)$ and $b(z) \neq 0$, then there exists Laurent polynomials $q(z)$(the quotient) and $r(z)$(the remainder) with $|r(z)| < |b(z)|$ such that

$$a(z) = b(z)q(z) + r(z).$$

Since the multiplicative identity in $\mathbf{R}[z, z^{-1}]$ is 1, we say $h \not\equiv 0$ is a *unit* if and only if h has a multiplicative inverse. Equivalently, if h has a multiplicative identity then it has degree 0, which are the monomials for our ring $\mathbf{R}[z, z^{-1}]$. In other words, we have $h^{-1}(z) = \frac{1}{C}z^{-k}$, where $C \in \mathbb{R}$.

Since a Laurent polynomial is invertible if and only if it is a monomial, then in the ring $\mathbf{R}[z, z^{-1}]$ we have non-uniqueness for the division of two Laurent polynomials. We illustrate this idea with the following example.

Consider that we want to divide the Laurent polynomial $a(z) = -z^{-1} + 5 - z$ by $b(z) = 2z^{-1} + 2$. Thus, we need a Laurent polynomial $q(z)$ of degree 1 so that a Laurent polynomial $r(z)$ given by

$$r(z) = a(z) - b(z)q(z)$$

is of degree zero. Here is were we have non-uniqueness. Our first choice of $q(z) = -\frac{1}{2} + 3z$ gives us

$$r(z) = (-z^{-1} + 5 + z) - (2z^{-1} + 2)(-\frac{1}{2} + 3z) = -5z$$

where the remainder is of degree zero. The second choice of $q(z) = -\frac{1}{2} + \frac{1}{2}z$ yields

$$r(z) = (-z^{-1} + 5 + z) - (2z^{-1} + 2)(-\frac{1}{2} + \frac{1}{2}z) = 5$$

where we again result in reminder of degree zero. The last choice of $q(z) = 2 + 3z$ results in

$$r(z) = (-z^{-1} + 5 + z) - (2z^{-1} + 2)(2 + \frac{1}{2}z) = -5z^{-1}$$

with the remainder of degree zero.

The result of the Division Algorithm not being unique will be useful for our factorization. In fact, we will always be able to compute the Division Algorithm and choose the remainder of degree zero to

be the constant term.

We will work with the ring of matrices $\mathbf{Mat}(2 \times 2, \mathbf{R}[z, z^{-1}])$ whose elements are of the form

$$\begin{bmatrix} a(z) & b(z) \\ c(z) & d(z) \end{bmatrix}, \qquad a, b, c, d \in \mathbf{R}[z, z^{-1}]$$

We state the perfect reconstruction conditions for filters $h, g, \tilde{h} = \bar{g}$, and $\tilde{g} = \bar{h}$ in terms of their *Z-transforms*

$$h(z)\tilde{h}(z^{-1}) + g(z)\tilde{g}(z^{-1}) = 2$$

$$h(z)\tilde{h}(-z^{-1}) + g(z)\tilde{g}(-z^{-1}) = 0$$

We define the *modulation matrix* $M(z)$ as

$$M(z) = \begin{bmatrix} h(z) & h(-z) \\ g(z) & g(-z) \end{bmatrix}$$

Similarly, we define the dual modulation matrix $\tilde{M}(z)$ as

$$\tilde{M}(z) = \begin{bmatrix} \tilde{h}(z) & \tilde{h}(-z) \\ \tilde{g}(z) & \tilde{g}(-z) \end{bmatrix}.$$

The perfect reconstruction conditions can now be stated as

$$\tilde{M}(z^{-1})^T M(z) = 2\mathbf{I}$$

where $\mathbf{I}$ is the 2×2 identity matrix. If all our entries are FIR filters, then the matrices $M(z)$ and $\tilde{M}(z)$ belong to $GL(2 \times 2, \mathbf{R}[z, z^{-1}])$, the set of invertible 2×2 matrices.

4.3 Polyphase Representation

The *polyphase matrix*, $P(z)$, of a FIR filter pair (h, g) and their conjugates $(\tilde{h}, \tilde{g})$ is

$$P(z) = \begin{bmatrix} h_e(z) & g_e(z) \\ h_o(z) & g_o(z) \end{bmatrix}$$

where $\tilde{P}(z)$ is similarly defined as

$$\tilde{P}(z) = \begin{bmatrix} \tilde{h}_e(z) & \tilde{g}_e(z) \\ \tilde{h}_o(z) & \tilde{g}_o(z) \end{bmatrix}.$$

Note we can rewrite our polyphase matrix using the following identity: $P(z^2)^T = \frac{1}{2}M(z)\begin{bmatrix} 1 & z \\ 1 & -z \end{bmatrix}$. We prove the following identity below

$$\begin{aligned} \frac{1}{2}M(z)\begin{bmatrix} 1 & z \\ 1 & -z \end{bmatrix} &= \frac{1}{2}\begin{bmatrix} h(z) & h(-z) \\ g(z) & g(-z) \end{bmatrix}\begin{bmatrix} 1 & z \\ 1 & -z \end{bmatrix} \\ &= \begin{bmatrix} \frac{h(z)+h(-z)}{2} & \frac{zh(z)-zh(-z)}{2} \\ \frac{g(z)+g(-z)}{2} & \frac{zg(z)-zg(-z)}{2} \end{bmatrix} \\ &= \begin{bmatrix} \frac{h(z)+h(-z)}{2} & \frac{h(z)-h(-z)}{2z^{-1}} \\ \frac{g(z)+g(-z)}{2} & \frac{g(z)-g(-z)}{2z^{-1}} \end{bmatrix} \\ &= P(z^2)^T \end{aligned}$$

Using the previous identity, we are now able to restate our perfect reconstruction condition as shown in the next theorem due to the work of Daubechies [2].

Theorem 12. *We now use the identity,* $P(z^2)^T = \frac{1}{2}M(z)\begin{bmatrix} 1 & z \\ 1 & -z \end{bmatrix}$*, to restate our perfect reconstruction condition as*

$$P(z)\tilde{P}(z^{-1})^T = I.$$

Proof. Using the identity $P(z^2)^T = \frac{1}{2}M(z)\begin{bmatrix} 1 & z \\ 1 & -z \end{bmatrix}$, we have

$$\begin{aligned} P(z)\tilde{P}(z^{-1})^T &= \left(\frac{1}{2}M(z)\begin{bmatrix} 1 & z \\ 1 & -z \end{bmatrix}\right)^T \left(\frac{1}{2}\tilde{M}(z^{-1})\begin{bmatrix} 1 & z^{-1} \\ 1 & -z^{-1} \end{bmatrix}\right) \\ &= \left(\frac{1}{2}\begin{bmatrix} 1 & z \\ 1 & -z \end{bmatrix}^T M(z)^T\right)\left(\frac{1}{2}\tilde{M}(z^{-1})\begin{bmatrix} 1 & z^{-1} \\ 1 & -z^{-1} \end{bmatrix}\right) \\ &= \left(\frac{1}{2}\begin{bmatrix} 1 & z \\ 1 & -z \end{bmatrix}^T\right)\left(\frac{1}{2}M(z)^T\tilde{M}(z^{-1})\right)\left(\begin{bmatrix} 1 & z^{-1} \\ 1 & -z^{-1} \end{bmatrix}\right) \end{aligned}$$

$$= \left(\frac{1}{2} \begin{bmatrix} 1 & z \\ 1 & -z \end{bmatrix}^T \right) (I) \left(\begin{bmatrix} 1 & z^{-1} \\ 1 & -z^{-1} \end{bmatrix} \right)$$

$$\begin{aligned} &= \frac{1}{2} \begin{bmatrix} 1 & 1 \\ z & -z \end{bmatrix} \begin{bmatrix} 1 & z^{-1} \\ 1 & -z^{-1} \end{bmatrix} \\ &= \frac{1}{2} \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} \\ &= I \end{aligned}$$

□

Since $P(z)$ has Laurent polynomial entries, from our perfect reconstruction condition we now conclude that

$$\begin{aligned} \det\left(P(z^2)\tilde{P}(z^{-1})^T\right) &= \det\left(P(z^2)\right) \det\left(\tilde{P}(z^{-1})^T\right) \\ &= 1 \cdot 1 \\ &= 1 \end{aligned}$$

Therefore, $\det P(z^2)$ is a monomial and we are able to scale to obtain $\det P(z^2) = 1$, i.e. P(z) is in $SL(2; \mathbf{R}[z, z^{-1}])$. Hence, for our wavelet transformation we only need to construct a polyphase matrix $P(z^2)$ with determinant 1. Since our polyphase matrix is invertible, we can then obtain $\tilde{P}(z^2) = \left[P((z^2)^{-1})^T\right]^{-1}$. Therefore, we solve and obtain the following filter pair $(\tilde{h}, \tilde{g})$ from our original pair (h, g):

$$\tilde{h}_e(z) = g_o(z^{-1}), \qquad \tilde{h}_o(z) = -g_e(z^{-1}), \qquad \tilde{g}_e(z) = -h_o(z^{-1}), \qquad \tilde{g}_o(z) = h_e(z^{-1}). \tag{4.1}$$

Which implies $\tilde{g}(z) = z^{-1}h(-z^{-1})$ and $\tilde{h}(z) = -z^{-1}g(-z^{-1})$.

4.4 Lifting Scheme

Definition A filter pair (h, g) is complementary if its corresponding polyphase matrix $P(z)$ has determinant 1. Note: If (h, g) is complementary, then so is $(\tilde{h}, \tilde{g})$ since we are able to rewrite the filter pair

$(\tilde{h}, \tilde{g})$ using (4.1).

Now that we have stated the definition of a complementary filter, in the next two theorems we follow the approach of Daubechies [2] and obtain the *prediction* and *update* steps for our filters in terms of the polyphase matrix.

Theorem 13. *(Primal Lifting) Let (h, g) be complementary, then any other filter g^* complementary to h is of the form:*

$$g^*(z) = g(z) + h(z)s(z^2)$$

where $s(z^2)$ is a Laurent polynomial.

Proof. Since we can decompose $g^*(z)$ into its even and odd components, we obtain

$$\begin{aligned} g^*(z) &= g(z) + h(z)s(z^2) \\ &= \left(g_e(z^2) + z^{-1}g_o(z^2)\right) + \left(h_e(z^2) + z^{-1}h_o(z^2)\right) s(z^2) \\ &= \left(g_e(z^2) + h_e(z^2)s(z^2)\right) + z^{-1}\left(g_o(z^2) + h_o(z^2)s(z^2)\right) \\ &= g_e^*(z^2) + z^{-1}g_o^*(z^2). \end{aligned}$$

So for our filter pair (h, g^*), the polyphase matrix $P^*(z)$ can be expressed as

$$\begin{aligned} P^*(z) &= \begin{bmatrix} h_e(z) & g_e^*(z) \\ h_o(z) & g_o^*(z) \end{bmatrix} \\ &= \begin{bmatrix} h_e(z) & g_e(z^2) + h_e(z^2)s(z^2) \\ h_o(z) & g_o(z^2) + h_o(z^2)s(z^2) \end{bmatrix} \\ &= \begin{bmatrix} h_e(z) & g_e(z) \\ h_o(z) & g_o(z) \end{bmatrix} \begin{bmatrix} 1 & s(z) \\ 0 & 1 \end{bmatrix} \\ &= P(z) \begin{bmatrix} 1 & s(z) \\ 0 & 1 \end{bmatrix}. \end{aligned}$$

The determinant of $P^*(z)$ is 1 since

$$\det P^*(z) = \det P(z) \cdot \det \begin{bmatrix} 1 & s(z) \\ 0 & 1 \end{bmatrix} = 1 \cdot 1 = 1.$$

□

The dual polyphase matrix $\tilde{P}^*(z)$ is then

$$\begin{aligned}\tilde{P}^*(z) &= \left[P^*(z^{-1})^T\right]^{-1} \\ &= \left(\begin{bmatrix}1 & s(z^{-1}) \\ 0 & 1\end{bmatrix}^T P(z^{-1})^T\right)^{-1}\end{aligned}$$

$$\begin{aligned}&= \left[P(z^{-1})^T\right]^{-1}\begin{bmatrix}1 & 0 \\ s(z^{-1}) & 1\end{bmatrix}^{-1} \\ &= \tilde{P}(z)\begin{bmatrix}1 & 0 \\ -s(z^{-1}) & 1\end{bmatrix}^{-1}.\end{aligned}$$

So we obtain a new filter pair $(\tilde{h}^*, \tilde{g})$ where

$$\tilde{h}^*(z) = \tilde{h}(z) - \tilde{g}(z)s((z^2)^{-1}.$$

Theorem 14. *(Dual Lifting)*

Let (h, g) be complementary. Then any other filter h^* complementary to g is of the form:

$$h^*(z) = h(z) + g(z)t(z^2)$$

where $t(z^2)$ is a Laurent polynomial.

Proof. Similarly to Theorem 1, after dual lifting we get the new polyphase matrix

$$P^*(z) = P(z)\begin{bmatrix}1 & 0 \\ t(z) & 1\end{bmatrix}.$$

Dual lifting also creates a new filter $\tilde{g}^*(z)$ given by

$$\tilde{g}^*(z) = \tilde{g}(z) - \tilde{h}(z)t((z^2)^{-1}).$$

□

4.5 The Euclidean Algorithm

Recall, that the Laurent polynomials form a commutative ring denoted by $\mathbf{R}[z, z^{-1}]$. More than a ring, the commutative ring of Laurent polynomials, $\mathbf{R}[z, z^{-1}]$, is a *Euclidean domain*. Consequently, $\mathbf{R}[z, z^{-1}]$ being a Euclidean domain makes it also a *unique factorization domain*, meaning that every non-zero element can be written in an essentially unique way as a product of a unit and prime elements of the ring. A Euclidean domain is an integral domain endowed with a Euclidean function. In $\mathbf{R}[z, z^{-1}]$, we define the Euclidean function as the *degree* of a Laurent polynomial h, by

$$\deg h = |h| = b - a.$$

The Euclidean algorithm was originated to find the greatest common divisor (GCD) between two natural numbers. Here it is used to find the GCD between two Laurent polynomials. The main difference in finding the GCD for Laurent polynomials is that is defined up to units. The units in the ring of Laurent polynomials are the monomials. This is similar for the case of regular polynomials where the GCD is defined up to a constant. Similar to natural numbers who are relatively prime if their gcd is one, two Laurent polynomials are relatively prime if their GCD has degree zero (monomial).

Theorem 15. *(Euclidean Algorithm for Laurent Polynomials)*

Let $a(z)$ and $b(z) \neq 0$ be two Laurent polynomials with deg $a(z) \geq$ deg $b(z)$. We let $a_0(z) = a(z)$ and $b_0(z) = b(z)$. There exists a Laurent polynomial $q_i(z)$ (quotient) with deg $q_i(z) =$ deg $a(z) -$ deg $b(z)$ and starting at $i = 0$ we compute:

- $a_{i+1}(z) = b_i(z)$
- $b_{i+1}(z) = a_i(z) - b_i(z)q_{i+1}(z)$.

Then $a_n(z) = gcd(a(z), b(z))$ where n is the smallest number for which $b_n(z) = 0$.

Since deg $b_{i+1}(z) <$ deg $b_i(z)$, then there is an m for which deg $b_m(z) = 0$. Thus, the algorithm has a total of $n = m + 1$ number of iterations. Therefore, the number of steps is bounded by $n \leq |b(z)| + 1$. For $q_{i+1}(z) = a_i(z)/b_i(z)$, we have that

$$\begin{bmatrix} a_n(z) \\ 0 \end{bmatrix} = \prod_{i=n}^{1} \begin{bmatrix} 0 & 1 \\ 1 & -q_i(z) \end{bmatrix} \begin{bmatrix} a(z) \\ b(z) \end{bmatrix}.$$

Therefore, by inverting we obtain

$$\begin{bmatrix} a(z) \\ b(z) \end{bmatrix} = \prod_{i=1}^{n} \begin{bmatrix} q_i(z) & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} a_n(z) \\ 0 \end{bmatrix}.$$

Thus, $a_n(z)$ divides both $a(z)$ and $b(z)$. If $a_n(z)$ results in a monomial, then $a(z)$ and $b(z)$ are relatively prime.

If we take the same example of Laurent polynomials $a(z) = a_0(z) = -z^{-1} + 5 - z$ and $b(z) = b_0(z) = 2z^{-1} + 2$. Then applying the Euclidean algorithm we get

- $a_1(z) = b_0(z) = 2z^{-1} + 2$
- $b_1(z) = a_0(z) - b_0(z)q_1(z) = (-z^{-1} + 5 - z) - (2z^{-1} + 2)\left(-\frac{1}{2} + \frac{1}{2}z\right) = 5$

A second iteration yields

- $a_2(z) = b_1(z) = 5$
- $b_2(z) = a_1(z) - b_1(z)q_2(z) = (2z^{-1} + 2) - (5)\left(\frac{2z^{-1}}{5} + \frac{2}{5}\right) = 0$

Thus, the $gcd(a(z), b(z)) = 5$. Hence, $a(z)$ and $b(z)$ are relatively prime and we have

$$\begin{bmatrix} -z^{-1} + 5 - z \\ 2z^{-1} + 2 \end{bmatrix} = \begin{bmatrix} -\frac{1}{2} + \frac{1}{2}z & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} \frac{2z^{-1}}{5} + \frac{2}{5} & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 5 \\ 0 \end{bmatrix}.$$

In this example, the number of iterations was $n = 2 = |b(z)| + 1$.

4.6 Factoring Algorithm

We consider a complementary filter pair (h, g) that can be factored into lifting steps. First, note that the polyphase components $h_e(z)$ and $h_0(z)$ must be relatively prime, if not we would have a common factor in our $\det P(z)$ which must be 1. We can then run the Euclidean algorithm on $h_e(z)$ and $h_0(z)$ and obtain a gcd of a monomial. Since the division is not unique, we can choose to have the GCD to be a monomial which is a constant. We will call this constant K and follow the ideas of Daubechies [2] to obtain

$$\begin{bmatrix} h_e(z) \\ h_0(z) \end{bmatrix} = \prod_{i=1}^{n} \begin{bmatrix} q_i(z) & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} K \\ 0 \end{bmatrix}.$$

A special case if $|h_e(z)| > |h_0(z)|$, then we can simply choose $q_1(z) = 0$. Otherwise we can assume the n to always be even, because if n is odd we multiply $h(z)$ by z and $g(z)$ by z^{-1}. The $\det P(z)$ is still

1 by a simple calculation. Given a filter h we can find a complementary filter g^* by letting

$$P^*(z) = \begin{bmatrix} h_e(z) & g_e^*(z) \\ h_0(z) & g_0^*(z) \end{bmatrix} = \prod_{i=1}^{n} \begin{bmatrix} q_i(z) & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} K & 0 \\ 0 & 1/K \end{bmatrix}.$$

where $\det P^*(z) = 1$ since n is even it follows

$$\det P^*(z) = \det \left(\prod_{i=1}^{n} \begin{bmatrix} q_i(z) & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} K & 0 \\ 0 & 1/K \end{bmatrix} \right) = (-1)^n(1) = 1.$$

We make the observation that the first matrix in our product can be rewritten as follows

$$\begin{bmatrix} q_i(z) & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & q_i(z) \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ q_i(z) & 1 \end{bmatrix}. \tag{4.2}$$

Using the first form of (4.2) when i is odd and the second form when i is even we can rewrite $P^*(z)$ as follows

$$\begin{aligned} P^*(z) &= \prod_{i=1}^{n/2} \begin{bmatrix} 1 & q_{2i-1}(z) \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ q_{2i}(z) & 1 \end{bmatrix} \begin{bmatrix} K & 0 \\ 0 & 1/K \end{bmatrix} \\ &= \prod_{i=1}^{n/2} \begin{bmatrix} 1 & q_{2i-1}(z) \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ q_{2i}(z) & 1 \end{bmatrix} \begin{bmatrix} K & 0 \\ 0 & 1/K \end{bmatrix} \end{aligned} \tag{4.3}$$

Now we are able to recover the filter g from g^* by the lifting scheme below

$$P(z) = P^*(z) \begin{bmatrix} 1 & s(z) \\ 0 & 1 \end{bmatrix} \tag{4.4}$$

Theorem 16. *Given a complementary filter pair (h, g), then there exists Laurent polynomials $s_i(z)$ and $t_i(z)$ for $1 \leq i \leq m$ and a nonzero constant K so that*

$$P(z) = \prod_{i=1}^{m} \begin{bmatrix} 1 & s_i(z) \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ t_i(z) & 1 \end{bmatrix} \begin{bmatrix} K & 0 \\ 0 & 1/K \end{bmatrix}.$$

We can prove the theorem by combining the previous results in equation (4.3) and (4.4). Last, define $m = n/2 + 1$ for $t_m(z) = 0$ and $s_(z) = K^2 s(z)$. In other words, we have that every finite filter wavelet transform can be factored into m lifting steps followed by a scaling.

The dual polyphase matrix $\tilde{P}(z)$ is then given as

$$\tilde{P}(z) = \prod_{i=1}^{m} \begin{bmatrix} 1 & 0 \\ -s(z^{-1}) & 1 \end{bmatrix} \begin{bmatrix} 1 & -t(z^{-1}) \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1/K & 0 \\ 0 & K \end{bmatrix}.$$

4.7 Examples

From Daubechies [2], we confirm the factorizations of the following wavelet transformations.

(Haar Wavelet)

We consider the Haar filter pair (normalized) given by $h(z) = 1 + z^{-1}$, $g(z) = -1/2 + 1/2z^{-1}$, $\tilde{h}(z) = 1/2 + 1/2z^{-1}$, and $\tilde{g}(z) = -1 + z^{-1}$. Therefore, our polyphase matrix $P(z)$ is the following

$$\begin{aligned} P(z) &= \begin{bmatrix} 1 & -1/2 \\ 1 & 1/2 \end{bmatrix} \\ &= P^*(z) \begin{bmatrix} 1 & s(z) \\ 0 & 1 \end{bmatrix} \\ &= \begin{bmatrix} h_e(z) = 1 & g_e^*(z) \\ h_o(z) = 1 & g_o^*(z) \end{bmatrix} \begin{bmatrix} 1 & s(z) \\ 0 & 1 \end{bmatrix} \\ &= \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} 1 & -1/2 \\ 0 & 1 \end{bmatrix} \end{aligned}$$

where the last equation comes from solving for $g_e^*(z)$ and $g_o^*(z)$ by the following system

$$\begin{cases} -1/2 = s(z) + g_e^*(z) \\ 1/2 = s(z) + g_o^*(z). \end{cases}$$

Letting $s(z) = 1/2$ we solve and get $g_e^*(z) = 0$ and $g_o^*(z) = 1$. Thus, our polyphase matrix $P(z)$ for the *Haar* can be factored into the following form

$$P(z) = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} 1 & -1/2 \\ 0 & 1 \end{bmatrix}.$$

This factorization corresponds to the following forward transform using lifting steps:

$$\begin{aligned} s_l^{(0)} &= x_{2l} \\ d_l^{(0)} &= x_{2l+1} \\ d_l &= d_l^{(0)} - s_l^{(0)} \\ s_l &= s_l^{(0)} + 1/2 d_l. \end{aligned}$$

Using our perfect reconstruction condition (), we solve for $\tilde{P}(z)$ and get

$$\begin{aligned} \tilde{P}(z) &= \tilde{P}(z^{-1}) \\ &= P(z)^{-1} \\ &= \left(\begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} 1 & -1/2 \\ 0 & 1 \end{bmatrix} \right)^{-1} \\ &= \begin{bmatrix} 1 & 1/2 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix} \end{aligned}$$

where we used the fact that $\tilde{P}(z) = \tilde{P}(z^{-1})$ in the first equation. This leads to the following lifting steps for the inverse transform:

$$\begin{aligned} s_l^{(0)} &= s_l - 1/2 d_l \\ d_l^{(0)} &= d_l + s_l^{(0)} \\ x_{2l+1} &= d_l^{(0)} \\ x_{2l} &= s_l^{(0)}. \end{aligned}$$

Daubechies-4 (D4)

For the $(D4)$ our filters of h and g are given by:

$$\begin{aligned} h(z) &= h_0 + h_1 z^{-1} + h_2 z^{-2} + h_3 z^{-3} \\ g(z) &= -h_3 z^2 + h_2 z - h_1 + h_0 z^{-1} \end{aligned}$$

where the coefficients are given by

$$h_0 = \frac{1+\sqrt{3}}{4\sqrt{2}}, \qquad h_1 = \frac{3+\sqrt{3}}{4\sqrt{2}}, \qquad h_2 = \frac{3-\sqrt{3}}{4\sqrt{2}}, \qquad h_3 = \frac{1-\sqrt{3}}{4\sqrt{2}}.$$

We notice that for the $D4$ we have $P(z) = \tilde{P}(z)$, thus the polyphase matrix is given by

$$\begin{aligned} P(z) &= \begin{bmatrix} h_0 + h_2 z^{-1} & -h_3 z - h_1 \\ h_1 + h_3 z^{-1} & h_2 z + h_0 \end{bmatrix} \\ &= \begin{bmatrix} \frac{1+\sqrt{3}}{4\sqrt{2}} + \frac{3-\sqrt{3}}{4\sqrt{2}} z^{-1} & -\frac{1-\sqrt{3}}{4\sqrt{2}} - \frac{3+\sqrt{3}}{4\sqrt{2}} \\ \frac{3+\sqrt{3}}{4\sqrt{2}} + \frac{1-\sqrt{3}}{4\sqrt{2}} z^{-1} & \frac{3-\sqrt{3}}{4\sqrt{2}} z + \frac{1+\sqrt{3}}{4\sqrt{2}} \end{bmatrix}. \end{aligned}$$

Applying the Euclidean algorithm, we get

$$\begin{aligned} a_1 &= b_0 = \tfrac{3+\sqrt{3}}{4\sqrt{2}} + \tfrac{1-\sqrt{3}}{4\sqrt{2}} z^{-1} \\ b_1 &= a_0 - b_0 q_1 \\ &= \tfrac{1+\sqrt{3}}{4\sqrt{2}} \tfrac{3-\sqrt{3}}{4\sqrt{2}} z^{-1} - \left(\tfrac{3+\sqrt{3}}{4\sqrt{2}} + \tfrac{1-\sqrt{3}}{4\sqrt{2}} z^{-1} \right) (-\sqrt{3}) \\ &= \tfrac{1+\sqrt{3}+3\sqrt{3}+3}{4\sqrt{2}} \\ &= \tfrac{1+\sqrt{3}}{\sqrt{2}} \end{aligned}$$

$$\begin{aligned} a_2 &= b_1 = \tfrac{1+\sqrt{3}}{2} \\ b_2 &= a_1 - b_1 q_2 \\ &= \tfrac{3+\sqrt{3}}{4\sqrt{2}} + \tfrac{1-\sqrt{3}}{4\sqrt{2}} z^{-1} - \left(\tfrac{1+\sqrt{3}}{\sqrt{2}} \right) \left(\tfrac{\sqrt{3}}{4} + \tfrac{\sqrt{3}-2}{4} z^{-1} \right) \\ &= \tfrac{3+\sqrt{3}}{4\sqrt{2}} - \tfrac{3+\sqrt{3}}{4\sqrt{2}} + \tfrac{1-\sqrt{3}}{4\sqrt{2}} z^{-1} - \tfrac{3-2+3-2\sqrt{3}}{4\sqrt{2}} z^{-1} \\ &= 0. \end{aligned}$$

Therefore, our $gcd = b_1 = \frac{1+\sqrt{3}}{2}$ and our factorization is the following

$$P(z) = \begin{bmatrix} 1 & -\sqrt{3} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ \frac{\sqrt{3}}{4} + \frac{\sqrt{3}-2}{4} z^{-1} & 1 \end{bmatrix} \begin{bmatrix} 1 & z \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{\sqrt{3}+1}{\sqrt{2}} & 0 \\ 0 & \frac{\sqrt{3}-1}{\sqrt{2}} \end{bmatrix}.$$

Since our polyphase matrix is unitary, we use the fact that $\tilde{P}(z) = P(z)$ and obtain one form of factorization as

$$\tilde{P}(z^{-1})^T = \begin{bmatrix} \frac{\sqrt{3}+1}{\sqrt{2}} & 0 \\ 0 & \frac{\sqrt{3}-1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} 1 & 0 \\ z^{-1} & 1 \end{bmatrix} \begin{bmatrix} 1 & \frac{\sqrt{3}}{4} + \frac{\sqrt{3}-2}{4} z \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ -\sqrt{3} & 1 \end{bmatrix}.$$

Thus in terms of lifting steps we get the following for the forward transform:

$$\begin{aligned} d_l^{(1)} &= x_{2l+1} - \sqrt{3}x_{2l} \\ s_l^{(1)} &= x_{2l} + \sqrt{3}/4 d_l^{(1)} + (\sqrt{3}-2)/4 d_{l+1}^{(1)} \\ d_l^2 &= d_l^{(1)} + s_{l-1}^{(1)} \\ s_l &= (\sqrt{3}+1)/\sqrt{2} s_l^{(1)} \\ d_l &= (\sqrt{3}-1)/\sqrt{2} d_l^{(2)}. \end{aligned}$$

Where the inverse transform is obtained from inverting our lifting steps:

$$\begin{aligned} d_l^{(2)} &= (\sqrt{3}+1)/\sqrt{2} d_l \\ s_l^{(1)} &= (\sqrt{3}-1)/\sqrt{2} s_l \\ d_l^1 &= d_l^{(2)} - s_{l-1}^{(1)} \\ x_{2l} &= s_l^{(1)} - \sqrt{3}/4 d_l^{(1)} - (\sqrt{3}-2)/4 d_{l+1}^{(1)} \\ x_{2l+1} &= d_l^{(1)} + \sqrt{3} x_{2l}. \end{aligned}$$

Our second form of factorization comes from the fact that $P(z)^{-1} = \tilde{P}(z^{-1})^T$ and use equation () to get the factorization as

$$P(z)^{-1} = \begin{bmatrix} \frac{\sqrt{3}-1}{\sqrt{2}} & 0 \\ 0 & \frac{\sqrt{3}+1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} 1 & -z \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ -\frac{\sqrt{3}}{4} - \frac{\sqrt{3}-2}{4} z^{-1} & 1 \end{bmatrix} \begin{bmatrix} 1 & \sqrt{3} \\ 0 & 1 \end{bmatrix}.$$

which leads to the forward transform lifting steps:

$$\begin{aligned} s_l^{(1)} &= x_{2l} + \sqrt{3} x_{2l+1} \\ d_l^{(1)} &= x_{2l+1} - \sqrt{3}/4 s_l^{(1)} - (\sqrt{3}-2)/4 s_{l-1}^{(1)} \\ s_l^2 &= s_l^{(1)} - d_{l+1}^{(1)} \\ s_l &= (\sqrt{3}-1)/\sqrt{2} s_l^{(1)} \\ d_l &= (\sqrt{3}+1)/\sqrt{2} d_l^{(2)} \end{aligned}$$

Where the inverse transform is obtained from inverting our lifting steps:

$$\begin{aligned} d_l^{(2)} &= (\sqrt{3}-1)/\sqrt{2} d_l \\ s_l^{(1)} &= (\sqrt{3}+1)/\sqrt{2} s_l \\ d_l^1 &= d_l^{(2)} + s_{l-1}^{(1)} \\ x_{2l} &= s_l^{(1)} + \sqrt{3}/4 d_l^{(1)} + (\sqrt{3}-2)/4 d_{l+1}^{(1)} \\ x_{2l+1} &= d_l^{(1)} - \sqrt{3} x_{2l}. \end{aligned}$$

Cohen-Daubechies-Feauveau 5-3 CDF(5,3)

The next filter we introduce is know as the *LeGall* filter pair which is used for the JPEG-2000 lossless compression algorithm Van Fleet [8]. For the CDF$(5,3)$, the filter h is symmetric and given by:

$$h(z) = h_2 z^{-2} + h_1 z^{-1} + h_0 + h_1 z + h_2 z^2$$

where the coefficients are given by

$$h_0 = \frac{3}{4}, \qquad h_1 = \frac{1}{4}, \qquad h_2 = -\frac{1}{8}.$$

Applying the Euclidean algorithm and letting $a_0 = -\frac{1}{8}z^{-2} + \frac{3}{4} - \frac{1}{8}z^2$ and $b_0 = \frac{1}{4}z^{-1} + \frac{1}{4}z$, we obtain

$$\begin{aligned} a_1 &= b_0 = \tfrac{1}{4}z^{-1} + \tfrac{1}{4}z \\ b_1 &= a_0 - b_0 q_1 \\ &= \left(-\tfrac{1}{8}z^{-2} + \tfrac{3}{4} - \tfrac{1}{8}z^2\right) - \left(\tfrac{1}{4}z^{-1} + \tfrac{1}{4}z\right)\left(-\tfrac{1}{2}(z^{-1} + z)\right) \\ &= 1 \end{aligned}$$

$$\begin{aligned} a_2 &= b_1 = 1 \\ b_2 &= a_1 - b_1 q_2 \\ &= \left(\tfrac{1}{4}z^{-1} + \tfrac{1}{4}z\right) - (1)\left(\tfrac{1}{4}(z^{-1} + z)\right) \\ &= 0. \end{aligned}$$

Therefore, our $gcd = b_1 = 1$ and our factorization is the following

$$P(z) = \begin{bmatrix} 1 & -\frac{1}{2}(z^{-1}+z) \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ \frac{1}{4}(z^{-1}+z) & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1/1 \end{bmatrix} = \begin{bmatrix} 1 & -\frac{1}{2}(z^{-1}+z) \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ \frac{1}{4}(z^{-1}+z) & 1 \end{bmatrix}.$$

Hence, in terms of lifting steps we get the following for the forward transform:

$$\begin{aligned} s_l^{(0)} &= x_{2l} \\ d_l^{(0)} &= x_{2l+1} \\ d_l^{(1)} &= s_l^{(0)} - \tfrac{1}{2}(d_l^{(0)} + d_{l-1}^{(0)}) \\ s_l^{(1)} &= d_l^{(0)} + \tfrac{1}{4}(s_l^{(0)} + s_{l-1}^{(0)}) \\ s_l &= (1)s_l^{(1)} \\ d_l &= (1/1)d_l^{(2)} \end{aligned}$$

Note we will be able to omit the last two lifting steps since our $gcd = 1$ for this filter.

Daubechies-6 (D6)

For our next example we have the Daubechies-6 which is of length 6 where the low pass filter is given by the Laurent polynomial

$$h(z) = \sum_{k=-2}^{3} h_k z^{-k}$$

where the coefficients are

$$h_{-2} = \frac{\sqrt{2}(1+\sqrt{10}+\sqrt{5+2\sqrt{10}})}{32} \qquad h_{-1} = \frac{\sqrt{2}(5+\sqrt{10}+3\sqrt{5+2\sqrt{10}})}{32}$$

$$h_0 = \frac{\sqrt{2}(10-2\sqrt{10}+2\sqrt{5+2\sqrt{10}})}{32} \qquad h_1 = \frac{\sqrt{2}(10-2\sqrt{10}-2\sqrt{5+2\sqrt{10}})}{32}$$

$$h_2 = \frac{\sqrt{2}(5+\sqrt{10}-3\sqrt{5+2\sqrt{10}})}{32} \qquad h_3 = \frac{\sqrt{2}(1+\sqrt{10}-5\sqrt{5+2\sqrt{10}})}{32}.$$

The polyphase components for the (D6) are

$$h_e(z) = h_{-2}z + h_0 + h_2 z^{-1} \qquad g_e(z) = -h_3 z - h_1 - h_{-1}z^{-1}$$

$$h_0(z) = h_{-1}z + h_1 + h_3 z^{-1} \qquad g_0(z) = h_2 z + h_0 + h_{-2}z^{-1}.$$

The factorization is then given by:

$$P(z) = \begin{bmatrix} 1 & 0 \\ \alpha & 1 \end{bmatrix} \begin{bmatrix} 1 & \beta z^{-1} + \beta' \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ \gamma + \gamma' z & 1 \end{bmatrix} \begin{bmatrix} 1 & \delta \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \xi & 0 \\ 0 & \frac{1}{\xi} \end{bmatrix}$$

where the coefficients are

$$\begin{aligned} \alpha &= -0.4122865950 \\ \beta &= -1.5651362796 \\ \beta' &= 0.3523876576 \\ \gamma &= 0.0284590896 \\ \gamma' &= 0.4921518449 \\ \delta &= -0.3896203900 \\ \xi &= 1.9182029462. \end{aligned}$$

This leads to the forward transform lifting steps:

$$\begin{aligned}
s_l^{(1)} &= x_{2l} + \alpha x_{2l+1} \\
d_l^{(1)} &= x_{2l+1} + \beta s_{l-1}^{(1)} + \beta' s_l^{(1)} \\
s_l^{(2)} &= s_l^{(1)} + \gamma d_l^{(1)} + \gamma' d_{l+1}^{(1)} \\
d_l^{(2)} &= d_l^{(1)} + \delta s_l^2 \\
s_l &= \xi s_l^{(2)} \\
d_l &= d_l^{(2)}/\xi
\end{aligned}$$

Where the inverse transform is obtained from inverting our lifting steps:

$$\begin{aligned}
d_l^{(2)} &= d_l \xi \\
s_l^{(2)} &= s_l/\xi \\
d_l^{(1)} &= d_l^{(2)} - \delta s_l^{(2)} \\
s_l^{(1)} &= s_l^{(2)} - \gamma d_l^{(1)} - \gamma' d_{l+1}^{(1)} \\
x_{2l+1} &= d_l^{(1)} - \beta s_{l-1}^{(1)} - \beta' s_l^{(1)} \\
x_{2l} &= s_l^{(1)} - \alpha x_{2l+1}.
\end{aligned}$$

Cohen-Daubechies-Feauveau 9-7 (CDF(9,7))

Here we consider the popular filter pair used in the JPEG 2000 algorithm. Here we have a biorthogonal symmetric filter pairs. The lowpass filters $\tilde{h}$ and h each have 9 and 7 coefficients respectively and similarly for the high pass filter pair g and $\tilde{g}$. We consider the following polyphase components for the forward transform:

$$\tilde{h}_e(z) = h_4(z^2 + z^{-2}) + h_2(z + z^{-1}) + h_0, \qquad \tilde{h}_o(z) = h_3(z^2 + z^{-1}) + h_1(z + 1).$$

The factorization is then given by:

$$\tilde{P}(z) = \begin{bmatrix} 1 & \alpha(1+z^{-1}) \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ \beta(1+z) & 1 \end{bmatrix} \begin{bmatrix} 1 & \gamma(1+z^{-1}) \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ \delta(1+z) & 1 \end{bmatrix} \begin{bmatrix} \xi & 0 \\ 0 & \frac{1}{\xi} \end{bmatrix}$$

where the coefficients are given as follows

$$\begin{aligned}
\alpha &= -1.586134342 \\
\beta &= -0.05298011854 \\
\gamma &= 0.8829110762 \\
\delta &= 0.4435068522 \\
\xi &= 1.149604398.
\end{aligned}$$

Thus, in terms of lifting steps we obtain

$$\begin{aligned}
s_l^{(0)} &= x_{2l} \\
d_l^{(0)} &= x_{2l+1} \\
d_l^{(1)} &= d_l^{(0)} + \alpha(s_l^{(0)} + s_{l+1}^{(0)}) \\
s_l^{(1)} &= s_l^{(0)} + \beta(d_l^{(1)} + d_{l-1}^{(1)}) \\
d_l^{(2)} &= d_l^{(1)} + \gamma(s_l^{(1)} + s_{l+1}^{(1)}) \\
s_l^{(2)} &= s_l^{(1)} + \delta(d_l^{(2)} + d_{l-1}^{(2)}) \\
s_l &= \xi s_l^{(2)} \\
d_l &= d_l^{(2)}/\xi.
\end{aligned}$$

Inverting the previous steps we get the inverse transform.

5 Conclusion and Remarks

We conclude with the comparison of the lifting method resulting in fewer computations than the standard wavelet transform as is done in Daubechies [2]. The standard wavelet transform will be referred to as applying the polyphase matrix $P(z)$ to the filter h as shown below

$$P(z) \begin{bmatrix} h_e(z) \\ z^{-1}h_o(z) \end{bmatrix}.$$

Notice that in either method we will have subsampled and we can compare the computations. The unit we will use to compare the two algorithms is *cost*. Cost of computing either algorithm is based on the number of multiplications and additions. For a filter h, the cost of applying the standard algorithm to a filter h is $|h| + 1$ multiplications and $|h|$ additions. The cost of the standard algorithm is then $2(|h| + |g|) + 2$. We see this for our example of the D4. The filters h and g for our D4 are given as

$$h(z) = h_0 + h_1 z^{-1} + h_2 z^{-2} + h_3 z^{-3}$$
$$g(z) = -h_3 z^2 + h_2 z - h_1 + h_0 z^{-1}.$$

Therefore, for h we have $|h| = 3$ and for g we have $|g| = 3$. Hence, the cost of implementing the standard algorithm for our D4 is

$$2(|h| + |g|) + 2 = 2(3 + 3) + 2 = 14.$$

If the filter h is symmetric and $|h|$ is even, then the cost of applying the standard algorithm $3|h|/2+1$. As we saw in our examples, the CDF(9,7) was a symmetric filter with $|h| = 8$ and $|g| = 6$. Thus, the cost of implementing the CDF(9,7) through the standard algorithm is

$$\left(\frac{3|h|}{2} + 1\right) + \left(\frac{3|g|}{2} + 1\right) = \frac{3(8)}{2} + 1 + \frac{3(6)}{2} + 1 = 23.$$

We consider a more general case with non-symmetric filters. Let $|h| = 2N$ and $|g| = 2M$. The cost using the standard algorithm is then

$$2(|h| + |g|) + 2 = 2(2N + 2M) + 2 = 4(N + M) + 2.$$

For the cost using the lifting method, we have that the polyphase components will have $|h_e| = N, |h_o| = N - 1, |g_e| = M$, and $|g_o| = M - 1$. When we compute the Euclidean algorithm for the a filter

pair (h_e, h_o) it requires N steps since each step has $|q_i| = 1$. From (.) we require one extra lifting step for (g_e, g_o) with $|s| = M - N$. Thus, the total cost of the lifting algorithm for this filter pair is:

Scaling :	2
N lifting steps :	$4N$
Final lifting step :	$2(M - N + 1)$
Total :	$2(N + M + 2)$

In the next figure we summarize our examples and their costs. We compute the speed up in percent between the lifting and standard algorithm for a filter pair (h, g).

Wavelet	Standard	Lifting	Speedup (%)
Haar	3	3	0
Daubechies-4	14	9	56
Daubechies-6	22	14	57
LeGall	11	6	83
CDF(9,7)	23	14	64
$\|h\| = 2N, \|g\| = 2M$	$4(N + M) + 2$	$2(N + M + 2)$	≈ 100

Figure 12: Cost of Lifting vs. Standard algorithms

In this paper, we began with an introduction to Fourier series and constructions of scaling functions and their associated wavelets. We then defined the concept of a multiresoultion analysis and how these sets of functions allow us to construct a scaling function by having the coefficients p_k satisfy a set of properties. We next introduced the idea of the lifting method which gives us another way of implementing a wavelet transformation. To represent our filters we represent them by Laurent polynomials and discuss the non-uniqueness of dividing two Laurent polynomials. We then showed that every wavelet transformation can be decomposed into lifting steps. Our decomposition is implemented by using the Euclidean algorithm on Laurent polynomials. Since our filters have a corresponding polyphase matrix in $SL(2; \mathbf{R}[z, z^{-1}])$, we are able to write the polyphase matrix as a product of elementary matrices also in $SL(2; \mathbf{R}[z, z^{-1}])$. We then concluded with some classic examples such as the Haar, $D4$, $D6$, and $CDF(9, 7)$. Our comparison between the standard algorithm and the lifting method shows that the speed up times are more than doubled for the longer length filters.

References

[1] A. Boggess and F.J. Narcowich. *A First Course in Wavelets with Fourier Analysis.* Prentice Hall, Upper Saddle River, NJ, 2001.

[2] I. Daubechies and W. Sweldens. *"Factoring Wavelet Transformations into Lifting Steps".* Fourier Analysis and Applications. 4:3 (1998), 247-269.

[3] A. Deitmar. *A First Course in Harmonic Analysis.* Springer-Verlag, New York, NY, 2005.

[4] A. Jensen and A. la Cour-Harbo. *Ripples in Mathematics: The Discrete Wavelet Transform.* Springer-Verlag, Berlin 2001.

[5] H.L. Royden and P.M. Fitzpatrick. *Real Analysis.* Prentice Hall, Boston, MA, 2010.

[6] D.K. Ruch and P.J. Van Fleet. *Wavelet Theory: An Elementary Approach with Applications.* Wiley-Interscience, Hoboken, NJ, 2009.

[7] W. Rudin. *Real and Complex Analysis.* McGraw-Hill, 1987.

[8] P.J. Van Fleet. *Discrete Wavelet Transformations: An Elementary Approach with Applications.* Wiley-Interscience, Hoboken, NJ, 2008.

Appendix

The appendix contains the code for the factorization of the wavelet transformation through both the standard and lifting method. We used the software *Mathematica* for all the implementations. The first we display is the Haar lowpass, h, and the high pass, g, filters with the coefficients multiplied by a normalizing factor of $\frac{1}{\sqrt{2}}$. We then display the 8×8 matrix used for implementing the Haar wavelet transform.

Haar (D2)

In[573]:=
```
h0 = 1 / Sqrt[2]
h1 = 1 / Sqrt[2]
g0 = 1 / Sqrt[2]
g1 = -1 / Sqrt[2]
```

Out[573]= $\frac{1}{\sqrt{2}}$

Out[574]= $\frac{1}{\sqrt{2}}$

Out[575]= $\frac{1}{\sqrt{2}}$

Out[576]= $-\frac{1}{\sqrt{2}}$

In[577]:=
```
H1[n_] := Module[{h0 = 1 / Sqrt[2], h1 = 1 / Sqrt[2],
    h2 = 1, g0 = 1 / Sqrt[2], g1 = -1 / Sqrt[2]},     Join[
    SparseArray[{{i_, j_} /; j == 2 i -> h0, {i_, j_} /; j == 2 i - 1 -> h1}, {n / 2, n}],
     SparseArray[{{i_, j_} /; j == 2 i → g1, {i_, j_} /; j == 2 i - 1 → g0}, {n / 2, n}]]]
MatrixForm[H1[8]]
```

Out[578]//MatrixForm=

$$\begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix}$$

Here we check that the reconstruction conditions hold for our Haar wavelet transformation matrix and its transpose.

In[579]:=
```
MatrixForm[H1[8].Transpose[H1[8]]]
```

Out[579]//MatrixForm=

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

We have the implementation of 3 transforms of the Haar to a vector **w**.

Example of vector w (3 DWT-Haar)

```
In[596]:= a1 = (1 / Sqrt[2]) * H1[8].w;
          s1 = Take[a1, 4];
          d1 = Take[a1, -4];
          a2 = (1 / Sqrt[2]) * H1[4].s1;
          s2 = Take[a2, 2];
          d2 = Take[a2, -2];
          a3 = (1 / Sqrt[2]) * H1[2].s2;
          s3 = Take[a3, 1];
          d3 = Take[a3, -1];
          Join[s3, d3, d2, d1]
Out[605]= {35, -3, 16, 10, 8, -8, 0, 12}
```

We check our results with the lifting method and we see we get the same through either implementation. The first row represents the same original vector **w** as used in the example above. The last three rows represent three lifting steps and the last row is our transformed vector.

Example of vector w (3 lifting steps)

```
In[580]:= w = {56, 40, 8, 24, 48, 48, 40, 16}
          odd1 = Partition[w, 2][[All, 1]];
          even1 = Partition[w, 2][[All, 2]];
          s1 = (odd1 + even1) / 2;
          d1 = odd1 - s1;
          w1 = Join[s1, d1]
          odd2 = Partition[s1, 2][[All, 1]];
          even2 = Partition[s1, 2][[All, 2]];
          s2 = (odd2 + even2) / 2;
          d2 = odd2 - s2;
          w2 = Join[s2, d2, d1]
          odd3 = Partition[s2, 2][[All, 1]];
          even3 = Partition[s2, 2][[All, 2]];
          s3 = (odd3 + even3) / 2;
          d3 = odd3 - s3;
          w3 = Join[s3, d3, d2, d1]
Out[580]= {56, 40, 8, 24, 48, 48, 40, 16}
Out[585]= {48, 16, 48, 28, 8, -8, 0, 12}
Out[590]= {32, 38, 16, 10, 8, -8, 0, 12}
Out[595]= {35, -3, 16, 10, 8, -8, 0, 12}
```

Similarly, we have the code for the D4. We display the coefficients and the wavelet transform matrix. We also include the lifting implementation to demonstrate the transformed vector for both methods.

Daubechies-4 (D4)

In[606]:=
```
h0 = (1 + Sqrt[3]) / (4 * Sqrt[2])
h1 = (3 + Sqrt[3]) / (4 * Sqrt[2])
h2 = (3 - Sqrt[3]) / (4 * Sqrt[2])
h3 = (1 - Sqrt[3]) / (4 * Sqrt[2])
g0 = h3
g1 = -h2
g2 = h1
g3 = -h0
```

Out[606]= $\frac{1+\sqrt{3}}{4\sqrt{2}}$

Out[607]= $\frac{3+\sqrt{3}}{4\sqrt{2}}$

Out[608]= $\frac{3-\sqrt{3}}{4\sqrt{2}}$

Out[609]= $\frac{1-\sqrt{3}}{4\sqrt{2}}$

Out[610]= $\frac{1-\sqrt{3}}{4\sqrt{2}}$

Out[611]= $-\frac{3-\sqrt{3}}{4\sqrt{2}}$

Out[612]= $\frac{3+\sqrt{3}}{4\sqrt{2}}$

Out[613]= $-\frac{1+\sqrt{3}}{4\sqrt{2}}$

In[614]:=
```
D4[n_] := Module[{h0 = (1 + Sqrt[3]) / (4 * Sqrt[2]),
   h1 = (3 + Sqrt[3]) / (4 * Sqrt[2]), h2 = (3 - Sqrt[3]) / (4 * Sqrt[2]),
   h3 = (1 - Sqrt[3]) / (4 * Sqrt[2]), g0 = h3, g1 = -h2, g2 = h1, g3 = -h0},
  Join[SparseArray[{{i_, j_} /; j == 2 i - 1 -> h0, {i_, j_} /; j == 2 i -> h1,
     {i_, j_} /; j == 2 i + 1 -> h2, {i_, j_} /; j == 2 i + 2 -> h3,
     {n / 2, 1} -> h2, {n / 2, 2} -> h3}, {n / 2, n}],
   SparseArray[{{i_, j_} /; j == 2 i - 1 → g0, {i_, j_} /; j == 2 i → g1,
     {i_, j_} /; j == 2 i + 1 → g2, {i_, j_} /; j == 2 i + 2 → g3,
     {n / 2, 1} → g2, {n / 2, 2} → g3}, {n / 2, n}]]]
MatrixForm[
 D4[
  8]]
```

Out[615]//MatrixForm=

$$\begin{pmatrix}
\frac{1+\sqrt{3}}{4\sqrt{2}} & \frac{3+\sqrt{3}}{4\sqrt{2}} & \frac{3-\sqrt{3}}{4\sqrt{2}} & \frac{1-\sqrt{3}}{4\sqrt{2}} & 0 & 0 & 0 & 0 \\
0 & 0 & \frac{1+\sqrt{3}}{4\sqrt{2}} & \frac{3+\sqrt{3}}{4\sqrt{2}} & \frac{3-\sqrt{3}}{4\sqrt{2}} & \frac{1-\sqrt{3}}{4\sqrt{2}} & 0 & 0 \\
0 & 0 & 0 & 0 & \frac{1+\sqrt{3}}{4\sqrt{2}} & \frac{3+\sqrt{3}}{4\sqrt{2}} & \frac{3-\sqrt{3}}{4\sqrt{2}} & \frac{1-\sqrt{3}}{4\sqrt{2}} \\
\frac{3-\sqrt{3}}{4\sqrt{2}} & \frac{1-\sqrt{3}}{4\sqrt{2}} & 0 & 0 & 0 & 0 & \frac{1+\sqrt{3}}{4\sqrt{2}} & \frac{3+\sqrt{3}}{4\sqrt{2}} \\
\frac{1-\sqrt{3}}{4\sqrt{2}} & -\frac{3-\sqrt{3}}{4\sqrt{2}} & \frac{3+\sqrt{3}}{4\sqrt{2}} & -\frac{1+\sqrt{3}}{4\sqrt{2}} & 0 & 0 & 0 & 0 \\
0 & 0 & \frac{1-\sqrt{3}}{4\sqrt{2}} & -\frac{3-\sqrt{3}}{4\sqrt{2}} & \frac{3+\sqrt{3}}{4\sqrt{2}} & -\frac{1+\sqrt{3}}{4\sqrt{2}} & 0 & 0 \\
0 & 0 & 0 & 0 & \frac{1-\sqrt{3}}{4\sqrt{2}} & -\frac{3-\sqrt{3}}{4\sqrt{2}} & \frac{3+\sqrt{3}}{4\sqrt{2}} & -\frac{1+\sqrt{3}}{4\sqrt{2}} \\
\frac{3+\sqrt{3}}{4\sqrt{2}} & -\frac{1+\sqrt{3}}{4\sqrt{2}} & 0 & 0 & 0 & 0 & \frac{1-\sqrt{3}}{4\sqrt{2}} & -\frac{3-\sqrt{3}}{4\sqrt{2}}
\end{pmatrix}$$

```
In[616]:= N[Chop[MatrixForm[D4[8].Transpose[D4[8]]]]]
```

Out[616]//MatrixForm=

$$\begin{pmatrix} 1. & 0. & 0. & 0. & 0. & 0. & 0. & 0. \\ 0. & 1. & 0. & 0. & 0. & 0. & 0. & 0. \\ 0. & 0. & 1. & 0. & 0. & 0. & 0. & 0. \\ 0. & 0. & 0. & 1. & 0. & 0. & 0. & 0. \\ 0. & 0. & 0. & 0. & 1. & 0. & 0. & 0. \\ 0. & 0. & 0. & 0. & 0. & 1. & 0. & 0. \\ 0. & 0. & 0. & 0. & 0. & 0. & 1. & 0. \\ 0. & 0. & 0. & 0. & 0. & 0. & 0. & 1. \end{pmatrix}$$

Example Via Lifting

```
In[617]:= w = RandomInteger[{-128, 127}, {16}]
          odd1 = Partition[w, 2][[All, 2]];
          even1 = Partition[w, 2][[All, 1]];
          d1 = odd1 - Sqrt[3] * even1;
          s1 = even1 + (Sqrt[3] / 4) * d1 + ((Sqrt[3] - 2) / 4) * RotateLeft[d1];
          d2 = d1 + RotateRight[s1];
          s = ((Sqrt[3] + 1) / Sqrt[2]) * s1;
          d = RotateLeft[((1 - Sqrt[3]) / Sqrt[2]) * d2];
          Chop[N[Join[s, d]]]
```

Out[617]= {80, 88, 90, 36, -119, -82, -49, -17, -121, -55, -82, -69, -21, 110, 118, -21}

Out[625]= {127.765, 57.5197, -134.85, -57.8898, -113.897, -116.265, 111.041, 45.9663, 27.8224, -79.6585, 1.0006, -64.504, -7.28343, -44.6153, 86.9129, 13.8573}

Example Via D4 Wavelet Transform

```
In[626]:= Chop[N[D4[16].w]]
```

Out[626]= {127.765, 57.5197, -134.85, -57.8898, -113.897, -116.265, 111.041, 45.9663, 27.8224, -79.6585, 1.0006, -64.504, -7.28343, -44.6153, 86.9129, 13.8573}

The next set of code we have has the CDF$(5,3)$ wavelet transformation matrices. Notice we have a biorthogonal pair here. We also demonstrate the example through both the standard and lifting method.

CDF (5,3)

```
In[68]:= LG1[n_] := Module[{h0 = 3 / 4, h1 = 1 / 4, h2 = -1 / 8, g0 = 1, g1 = -1 / 2},
           Join[SparseArray[{{i_, j_} /; j == 2 i - 3 -> h2, {i_, j_} /; j == 2 i - 2 -> h1,
               {i_, j_} /; j == 2 i - 1 -> h0, {i_, j_} /; j == 2 i -> h1, {i_, j_} /; j == 2 i + 1 ->
                h2, {n / 2, 1} -> h2, {1, n - 1} -> h2, {1, n} → h1}, {n / 2, n}],
             SparseArray[{{i_, j_} /; j == 2 i - 1 → g1, {i_, j_} /; j == 2 i → g0,
               {i_, j_} /; j == 2 i + 1 → g1, {n / 2, 1} → g1}, {n / 2, n}]]]
         MatrixForm[
          LG1[
           16]]
```

Out[69]//MatrixForm=

$$\begin{pmatrix} \frac{3}{4} & \frac{1}{4} & -\frac{1}{8} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\frac{1}{8} & \frac{1}{4} \\ -\frac{1}{8} & \frac{1}{4} & \frac{3}{4} & \frac{1}{4} & -\frac{1}{8} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -\frac{1}{8} & \frac{1}{4} & \frac{3}{4} & \frac{1}{4} & -\frac{1}{8} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -\frac{1}{8} & \frac{1}{4} & \frac{3}{4} & \frac{1}{4} & -\frac{1}{8} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -\frac{1}{8} & \frac{1}{4} & \frac{3}{4} & \frac{1}{4} & -\frac{1}{8} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\frac{1}{8} & \frac{1}{4} & \frac{3}{4} & \frac{1}{4} & -\frac{1}{8} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\frac{1}{8} & \frac{1}{4} & \frac{3}{4} & \frac{1}{4} & -\frac{1}{8} \\ -\frac{1}{8} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\frac{1}{8} & \frac{1}{4} & \frac{3}{4} & \frac{1}{4} \\ -\frac{1}{2} & 1 & -\frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -\frac{1}{2} & 1 & -\frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -\frac{1}{2} & 1 & -\frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -\frac{1}{2} & 1 & -\frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\frac{1}{2} & 1 & -\frac{1}{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\frac{1}{2} & 1 & -\frac{1}{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\frac{1}{2} & 1 & -\frac{1}{2} \\ -\frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

In[70]:=
```
LG2[n_] := Module[{h0 = 1, h1 = 1 / 2, g0 = 3 / 4, g1 = -1 / 4, g2 = -1 / 8},
    Join[SparseArray[{{i_, j_} /; j == 2 i - 2 -> h1,
      {i_, j_} /; j == 2 i - 1 -> h0, {i_, j_} /; j == 2 i -> h1, {1, n} → h1}, {n / 2, n}],
    SparseArray[{{i_, j_} /; j == 2 i - 2 → g2, {i_, j_} /; j == 2 i - 1 → g1,
      {i_, j_} /; j == 2 i → g0, {i_, j_} /; j == 2 i + 1 → g1, {i_, j_} /; j == 2 i + 2 → g2,
      {n / 2, 1} → g1, {n / 2, 2} → g2, {1, n} → g2}, {n / 2, n}]]]
MatrixForm[
 LG2[
  16]]
```

Out[71]//MatrixForm=

$$\begin{pmatrix}
1 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} \\
0 & \frac{1}{2} & 1 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & \frac{1}{2} & 1 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 1 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 1 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 1 & \frac{1}{2} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 1 & \frac{1}{2} & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 1 & \frac{1}{2} \\
-\frac{1}{4} & \frac{3}{4} & -\frac{1}{4} & -\frac{1}{8} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\frac{1}{8} \\
0 & -\frac{1}{8} & -\frac{1}{4} & \frac{3}{4} & -\frac{1}{4} & -\frac{1}{8} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -\frac{1}{8} & -\frac{1}{4} & \frac{3}{4} & -\frac{1}{4} & -\frac{1}{8} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & -\frac{1}{8} & -\frac{1}{4} & \frac{3}{4} & -\frac{1}{4} & -\frac{1}{8} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & -\frac{1}{8} & -\frac{1}{4} & \frac{3}{4} & -\frac{1}{4} & -\frac{1}{8} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\frac{1}{8} & -\frac{1}{4} & \frac{3}{4} & -\frac{1}{4} & -\frac{1}{8} & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\frac{1}{8} & -\frac{1}{4} & \frac{3}{4} & -\frac{1}{4} & -\frac{1}{8} \\
-\frac{1}{4} & -\frac{1}{8} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\frac{1}{8} & -\frac{1}{4} & \frac{3}{4}
\end{pmatrix}$$

In[72]:=
```
MatrixForm[LG1[12].Transpose[LG2[12]]]
MatrixForm[LG2[12].Transpose[LG1[12]]]
```

Out[72]//MatrixForm=

$$\begin{pmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{pmatrix}$$

Out[73]//MatrixForm=

$$\begin{pmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{pmatrix}$$

Example CDF(5,3) via Lifting

```
In[74]:= v = RandomInteger[{-128, 127}, {16}]
         odd = Partition[v, 2][[All, 1]];
         even = Partition[v, 2][[All, 2]];
         d = even - 1 / 2 (odd + RotateLeft[odd]);
         s = odd + 1 / 4 (d + RotateRight[d]);
         N[Join[s, d]]
Out[74]= {-24, -46, -82, 30, -115, -114, -125, 66, -7, -55, 61, 56, 113, -69, 30, 53}

Out[79]= {-9.75, -48.125, -81.375, -90.5, 5.5, 32.75,
          70.125, 7.375, 7., 128.5, 6., 132., -82., -31., -140.5, 50.}
```

Example CDF(5,3) via Wavelet Transform

```
In[80]:= N[LG1[16].v]
Out[80]= {-9.75, -48.125, -81.375, -90.5, 5.5, 32.75,
          70.125, 7.375, 7., 128.5, 6., 132., -82., -31., -140.5, 50.}
```

The next set of code is for the D6. We again state the wavelet transformation matrices and the implementation through both the standard and lifting method.

Daubechies-6 (D6)

```
In[15]:= N[h0 = (Sqrt[2] (1 + Sqrt[10] + Sqrt[5 + 2 Sqrt[10]])) / 32]
         N[h1 = (Sqrt[2] (5 + Sqrt[10] + 3 Sqrt[5 + 2 Sqrt[10]])) / 32]
         N[h2 = (Sqrt[2] (10 - 2 Sqrt[10] + 2 Sqrt[5 + 2 Sqrt[10]])) / 32]
         N[h3 = (Sqrt[2] (10 - 2 Sqrt[10] - 2 Sqrt[5 + 2 Sqrt[10]])) / 32]
         N[h4 = (Sqrt[2] (5 + Sqrt[10] - 3 Sqrt[5 + 2 Sqrt[10]])) / 32]
         N[h5 = (Sqrt[2] (1 + Sqrt[10] - Sqrt[5 + 2 Sqrt[10]])) / 32]
         N[g0 = h5]
         N[g1 = -h4]
         N[g2 = h3]
         N[g3 = -h2]
         N[g4 = h1]
         N[g5 = -h0]
Out[15]= 0.332671

Out[16]= 0.806892

Out[17]= 0.459878

Out[18]= -0.135011

Out[19]= -0.0854413

Out[20]= 0.0352263

Out[21]= 0.0352263

Out[22]= 0.0854413

Out[23]= -0.135011

Out[24]= -0.459878

Out[25]= 0.806892

Out[26]= -0.332671
```

```
In[37]:= D6[n_] := Module[{h0 = N[(Sqrt[2] (1 + Sqrt[10] + Sqrt[5 + 2 Sqrt[10]])) / 32],
           h1 = N[(Sqrt[2] (5 + Sqrt[10] + 3 Sqrt[5 + 2 Sqrt[10]])) / 32],
           h2 = N[(Sqrt[2] (10 - 2 Sqrt[10] + 2 Sqrt[5 + 2 Sqrt[10]])) / 32],
           h3 = N[(Sqrt[2] (10 - 2 Sqrt[10] - 2 Sqrt[5 + 2 Sqrt[10]])) / 32],
           h4 = N[(Sqrt[2] (5 + Sqrt[10] - 3 Sqrt[5 + 2 Sqrt[10]])) / 32],
           h5 = N[(Sqrt[2] (1 + Sqrt[10] - Sqrt[5 + 2 Sqrt[10]])) / 32],
           g0 = N[h5],
           g1 = N[-h4],
           g2 = N[h3],
           g3 = N[-h2],
           g4 = N[h1],
           g5 = N[-h0]},
          Join[SparseArray[
            {{i_, j_} /; j == 2 i - 1 -> h0, {i_, j_} /; j == 2 i -> h1, {i_, j_} /; j == 2 i + 1 -> h2,
             {i_, j_} /; j == 2 i + 2 -> h3, {i_, j_} /; j == 2 i + 3 -> h4,
             {i_, j_} /; j == 2 i + 4 -> h5, {n / 2 - 1, 1} -> h4, {n / 2 - 1, 2} -> h5,
             {n / 2, 1} -> h2, {n / 2, 2} -> h3, {n / 2, 3} -> h4, {n / 2, 4} -> h5}, {n / 2, n}],
           SparseArray[{{i_, j_} /; j == 2 i - 1 → g0, {i_, j_} /; j == 2 i → g1,
             {i_, j_} /; j == 2 i + 1 → g2, {i_, j_} /; j == 2 i + 2 → g3, {i_, j_} /; j == 2 i + 3 →
              g4, {i_, j_} /; j == 2 i + 4 → g5, {n / 2 - 1, 1} → g4, {n / 2 - 1, 2} → g5,
             {n / 2, 1} → g2, {n / 2, 2} → g3, {n / 2, 3} → g4, {n / 2, 4} → g5}, {n / 2, n}]]]
      MatrixForm[
       D6[
        12]]
```

Out[38]//MatrixForm=

$$\begin{pmatrix}
0.332671 & 0.806892 & 0.459878 & -0.135011 & -0.0854413 & 0.0352263 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0.332671 & 0.806892 & 0.459878 & -0.135011 & -0.0854413 & 0.0352263 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0.332671 & 0.806892 & 0.459878 & -0.135011 & -0.0854413 & 0.0352263 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0.332671 & 0.806892 & 0.459878 & -0.135011 & -0.0854413 & 0.0352263 \\
-0.0854413 & 0.0352263 & 0 & 0 & 0 & 0 & 0 & 0 & 0.332671 & 0.806892 & 0.459878 & -0.135011 \\
0.459878 & -0.135011 & -0.0854413 & 0.0352263 & 0 & 0 & 0 & 0 & 0 & 0 & 0.332671 & 0.806892 \\
0.0352263 & 0.0854413 & -0.135011 & -0.459878 & 0.806892 & -0.332671 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0.0352263 & 0.0854413 & -0.135011 & -0.459878 & 0.806892 & -0.332671 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0.0352263 & 0.0854413 & -0.135011 & -0.459878 & 0.806892 & -0.332671 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0.0352263 & 0.0854413 & -0.135011 & -0.459878 & 0.806892 & -0.332671 \\
0.806892 & -0.332671 & 0 & 0 & 0 & 0 & 0 & 0 & 0.0352263 & 0.0854413 & -0.135011 & -0.459878 \\
-0.135011 & -0.459878 & 0.806892 & -0.332671 & 0 & 0 & 0 & 0 & 0 & 0 & 0.0352263 & 0.0854413
\end{pmatrix}$$

```
In[39]:= a1 = -.4122865950;
         a2 = -1.5651362796;
         a3 = .3523876576;
         a4 = .0284590896;
         a5 = .4921518449;
         a6 = -.3896203900;
         a7 = 1.9182029462;
```

Example D6 via Lifting

```
In[57]:= w = RandomInteger[{-128, 127}, {16}]
         odd = Partition[w, 2][[All, 1]];
         even = Partition[w, 2][[All, 2]];
         s1 = odd + a1 * even;
         d1 = even + (a2 * RotateLeft[s1] + a3 * s1);
         s2 = s1 + (a4 * d1 + a5 * RotateRight[d1]);
         d2 = d1 + a6 * s2;
         s = RotateLeft[a7 * s2];
         d = RotateLeft[(-1 / a7) * d2];
         Chop[N[Join[s, d]]]
```

```
Out[57]= {121, 59, -45, 90, 39, -82, -112, -10, -1, -20, -110, -89, -93, -118, -99, -46}
```

```
Out[66]= {48.7935, 95.8733, -103.966, -36.8243, -51.2517, -128.406, -173.729, -15.3566,
           32.7377, -48.4961, 19.9342, -54.6176, 18.2505, -9.23701, 99.1687, -117.137}
```

Example D6 via Wavelet Transform

```
In[67]:= Chop[N[D6[16].w]]
```

```
Out[67]= {48.7935, 95.8733, -103.966, -36.8243, -51.2517, -128.406, -173.729, -15.3566,
           32.7377, -48.4961, 19.9342, -54.6176, 18.2505, -9.23701, 99.1687, -117.137}
```

The last code is for the CDF(9,7). We again state the wavelet transformation matrices and the implementation through both the standard and lifting method.

CDF(9,7)

```
In[83]:= CDF97a[n_] := Module[{h3 = -0.0645389,
           h2 = -0.0406894,
           h1 = 0.418092,
           h0 = 0.788486,
           g0 = 0.852699,
           g1 = -0.377403,
           g2 = -0.110624,
           g3 = 0.0238495,
           g4 = 0.0378285},
          Join[SparseArray[{{i_, j_} /; j == 2 i - 1 -> h3, {i_, j_} /; j == 2 i -> h2,
              {i_, j_} /; j == 2 i + 1 -> h1, {i_, j_} /; j == 2 i + 2 -> h0,
              {i_, j_} /; j == 2 i + 3 -> h1, {i_, j_} /; j == 2 i + 4 -> h2,
              {i_, j_} /; j == 2 i + 5 -> h3, {n / 2 - 2, 1} -> h3, {n / 2 - 1, 1} -> h1,
              {n / 2 - 1, 2} -> h2, {n / 2 - 1, 3} -> h3, {n / 2, 1} -> h1, {n / 2, 2} -> h0,
              {n / 2, 3} -> h1, {n / 2, 4} -> h2, {n / 2, 5} -> h3}, {n / 2, n}],
            SparseArray[{{i_, j_} /; j == 2 i - 1 → g4, {i_, j_} /; j == 2 i → g3,
              {i_, j_} /; j == 2 i + 1 → g2, {i_, j_} /; j == 2 i + 2 → g1,
              {i_, j_} /; j == 2 i + 3 → g0, {i_, j_} /; j == 2 i + 4 → g1,
              {i_, j_} /; j == 2 i + 5 → g2, {i_, j_} /; j == 2 i + 6 → g3,
              {i_, j_} /; j == 2 i + 7 → g4, {n / 2 - 3, 1} → g4, {n / 2 - 2, 1} → g2, {n / 2 - 2, 2} → g3,
              {n / 2 - 2, 3} → g4, {n / 2 - 1, 1} → g0, {n / 2 - 1, 2} → g1, {n / 2 - 1, 3} → g2,
              {n / 2 - 1, 4} → g3, {n / 2 - 1, 5} → g4, {n / 2, 1} → g2, {n / 2, 2} → g1, {n / 2, 3} → g0,
              {n / 2, 4} → g1, {n / 2, 5} → g2, {n / 2, 6} → g3, {n / 2, 7} → g4}, {n / 2, n}]]]
      MatrixForm[
       CDF97a[
        16]]

In[85]:= CDF97b[n_] := Module[{h3 = -0.0645389,
           h2 = -0.0406894,
           h1 = 0.418092,
           h0 = 0.788486,
           g0 = 0.852699,
           g1 = -0.377403,
           g2 = -0.110624,
           g3 = 0.0238495,
           g4 = 0.0378285},
          Join[SparseArray[{{i_, j_} /; j == 2 i - 2 → g4, {i_, j_} /; j == 2 i - 1 → -g3,
              {i_, j_} /; j == 2 i → g2, {i_, j_} /; j == 2 i + 1 → -g1, {i_, j_} /; j == 2 i + 2 → g0,
              {i_, j_} /; j == 2 i + 3 → -g1, {i_, j_} /; j == 2 i + 4 → g2,
              {i_, j_} /; j == 2 i + 5 → -g3, {i_, j_} /; j == 2 i + 6 → g4, {n / 2 - 2, 1} → -g3,
              {n / 2 - 2, 2} → g4, {n / 2 - 1, 1} → -g1, {n / 2 - 1, 2} → g2, {n / 2 - 1, 3} → -g3,
              {n / 2 - 1, 4} → g4, {n / 2, 1} → -g1, {n / 2, 2} → g0, {n / 2, 3} → -g1,
              {n / 2, 4} → g2, {n / 2, 5} → -g3, {n / 2, 6} → g4}, {n / 2, n}],
            SparseArray[{{i_, j_} /; j == 2 i → -h3, {i_, j_} /; j == 2 i + 1 → h2,
              {i_, j_} /; j == 2 i + 2 → -h1, {i_, j_} /; j == 2 i + 3 → h0,
              {i_, j_} /; j == 2 i + 4 → -h1, {i_, j_} /; j == 2 i + 5 → h2,
              {i_, j_} /; j == 2 i + 6 → -h3, {n / 2 - 2, 1} → h2, {n / 2 - 2, 2} → -h3,
              {n / 2 - 1, 1} → h0, {n / 2 - 1, 2} → -h1, {n / 2 - 1, 3} → h2,
              {n / 2 - 1, 4} → -h3, {n / 2, 1} → h2, {n / 2, 2} → -h1, {n / 2, 3} → h0,
              {n / 2, 4} → -h1, {n / 2, 5} → h2, {n / 2, 6} → -h3}, {n / 2, n}]]]
      MatrixForm[
       CDF97b[
        16]]

In[89]:= a1 = -1.586134342;
      a2 = -0.05298011854;
      a3 = 0.8829110762;
      a4 = 0.4435068522;
      a5 = 1.149604398;
```

In[209]:=
```
v = RandomInteger[{-128, 127}, {16}]
odd = Partition[v, 2][[All, 1]];
even = Partition[v, 2][[All, 2]];
d1 = odd + a1 * (even + RotateRight[even]);
s1 = even + a2 * (d1 + RotateLeft[d1]);
d2 = d1 + a3 * (s1 + RotateRight[s1]);
s2 = s1 + a4 * (d2 + RotateLeft[d2]);
s = RotateLeft[a5 * s2];
d = RotateLeft[d2 / a5, 2];
N[Join[s, d]]
```

Out[209]= {26, -49, -125, -110, 41, 58, -106, -107, -97, 67, 28, -66, -55, 24, 23, 14}

Out[218]= {-128.104, 54.9063, -189.983, 57.1697, -78.288, 13.5334, 30.8312, -66.9496, 53.3999, -63.5895, -57.102, 22.4876, -22.6543, -3.99432, 33.7338, -30.1634}

In[219]:=
```
CDF97b[16].v
```

Out[219]= {-128.634, 54.9063, -189.983, 57.1697, -78.288, 13.5334, 30.8312, -66.9497, 53.3999, -63.5896, -57.102, 22.4876, -22.6544, -3.9943, 33.7338, -30.1635}

List of Figures

Curriculum Vita

Adrian Delgado was born in El Paso, Texas. The second son of Mario and Miriam Delgado, he graduated J.M. Hanks High School, El Paso, Texas in the spring of 2006. He enrolled at University of Texas at El Paso in the fall of 2007. While pursuing a bachelor's degree in mathematics, he worked as a tutor and peer leader for freshmen mathematics courses. After completing a student internship at Montwood HIgh School, El Paso, Texas, he received his bachelor's of science degree from University of Texas at El Paso in the spring of 2010. Eager to further his studies in mathematics, in the fall of 2010 he enrolled in Graduate School at the University of Texas at El Paso in the Department of Mathematics.