Discrete Wavelet Transformations and Undergraduate Education

Catherine Bénéteau and Patrick J. Van Fleet

Wavelet theory was an immensely popular research area in the 1990s that synthesized ideas from mathematics, physics, electrical engineering, and computer science. In mathematics, the subject attracted researchers from areas such as real and harmonic analysis, statistics, and approximation theory, among others. Applications of wavelets abound today—perhaps the most significant contributions of wavelets can be found in signal processing and digital image compression. As the basic tenets of wavelet theory were established, they became part of graduate school courses and programs, but it is only in the last ten years that we have seen wavelets and their applications being introduced into the undergraduate curriculum.

The introduction of the topic to undergraduates is quite timely—most of the foundational questions posed by wavelet researchers have been answered, and several current applications of wavelets are firmly entrenched in areas of image processing. Several authors [7, 2, 8, 14] have written books in which they present the basic results of wavelet theory in a manner that is accessible to undergraduates. Others have authored books [15, 13, 1, 9] that include applications as part of their presentation.

Catherine Bénéteau is professor of mathematics at the University of South Florida. Her email address is cbenetea@usf.edu.

Patrick J. Van Fleet is professor of mathematics at the University of St. Thomas. His email address is pjvanfleet@stthomas.edu.

This material is based upon work supported by the National Science Foundation under grant DUE-0717662.

In much the same way that wavelet theory is the confluence of several mathematical disciplines, we have discovered that *the discrete wavelet transformation* is an ideal topic for a modern undergraduate class—the derivation of the discrete wavelet transformation draws largely from calculus and linear algebra, provides a natural conduit to Fourier series and discrete convolution, and allows near-immediate access to current applications. Students learn about signal denoising, edge detection in digital images, and image compression, and use computer software in both the derivation and implementation of the discrete wavelet transformation. In the process of investigating applications, students learn *how* the application often drives the development of mathematical tools. Finally, the design of more advanced *wavelet filters* allows the students to gain experience "working in the transform domain" and provides motivation for several important concepts from an undergraduate real analysis class.

In what follows, we outline the basic development of discrete wavelet transformations and discuss their connection with Fourier series, convolution, and filtering. In the process, we illustrate the application of discrete wavelet transformations to digital images. We next construct one pair of filters that are used by the JPEG2000 image compression standard. We conclude the paper by illustrating how wavelet filters are implemented by the JPEG2000 standard.

Let us begin by discussing the simplest discrete wavelet transformation.

The Discrete Haar Wavelet Transformation

The discrete Haar wavelet transformation is the most elementary of all discrete wavelet transforms and serves as an excellent pedagogical tool for the development of more sophisticated discrete wavelet transformations and their application to digital signals or images.

We motivate this transformation as follows: Suppose we wish to transmit a list (vector) of N numbers, N even, to a friend via the Internet. To reduce transfer time, we decide to send only a length $N/2$ approximation of the data. One way to form the approximation is to send pairwise averages of the numbers. For example, the vector $[6, 12, 15, 15, 14, 12, 120, 116]^T$ can be approximated by the vector $[9, 15, 13, 118]^T$. Of course, it is impossible to determine the original N numbers from this approximation, but if, in addition, we transmit the $N/2$ averaged *directed distances*, then our friend could completely recover the original data. For our example, the averaged directed distances are $[3, 0, -1, -2]^T$.

We define the one–dimensional *discrete Haar wavelet transformation* as the linear transformation

$$\mathbf{x} \to \left[\begin{array}{c} \mathbf{a} \\ \hline \mathbf{d} \end{array}\right],$$

where $\mathbf{x} \in \mathbf{R}^N$, N even, and the elements of $\mathbf{a}, \mathbf{d} \in \mathbf{R}^{N/2}$ are

$$a_k = \frac{x_{2k-1} + x_{2k}}{2},$$
$$d_k = \frac{-x_{2k-1} + x_{2k}}{2},$$

where $k = 1, \ldots, N/2$. Given these averages and differences, we can recover the original data.

In matrix form, the transformation can be expressed as

$$\tilde{W}_N = \left[\begin{array}{ccccccc} \frac{1}{2} & \frac{1}{2} & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \frac{1}{2} & \frac{1}{2} & & 0 & 0 \\ & & & & \ddots & & \\ 0 & 0 & 0 & 0 & \cdots & \frac{1}{2} & \frac{1}{2} \\ \hline -\frac{1}{2} & \frac{1}{2} & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & -\frac{1}{2} & \frac{1}{2} & & 0 & 0 \\ & & & & \ddots & & \\ 0 & 0 & 0 & 0 & \cdots & -\frac{1}{2} & \frac{1}{2} \end{array}\right]$$

$$= \left[\begin{array}{c} \tilde{H}_{N/2} \\ \hline \tilde{G}_{N/2} \end{array}\right] \tag{1}$$

with

$$\tilde{W}_N^{-1} = 2\tilde{W}_N^T = 2\left[\begin{array}{c|c} \tilde{H}_{N/2}^T & \tilde{G}_{N/2}^T \end{array}\right]. \tag{2}$$

What is the advantage of transforming $\mathbf{x}$ into $\mathbf{a}$ and $\mathbf{d}$? The vector $\mathbf{a}$ gives us an approximation of the original vector $\mathbf{x}$, while $\mathbf{d}$ allows us to identify large (or small) differences between the pairwise averages and the original data. Identifying large differences might be of interest if we were attempting to detect edges in a digital image, for example. On the other hand, we might be inclined to convert differences that are small in absolute value to zero—this action, assuming the original data were largely homogeneous, has the effect of producing a large number of zeros in the *quantized* differences vector $\tilde{\mathbf{d}}$. In this case, the modified transform is more amenable to data coders in image compression applications. Of course, we have no hope of recovering the original vector $\mathbf{x}$ from the modified transform, but depending on the application, the sacrifice of loss of resolution for storage size might be desirable.

The Two-Dimensional Discrete Haar Wavelet Transformation

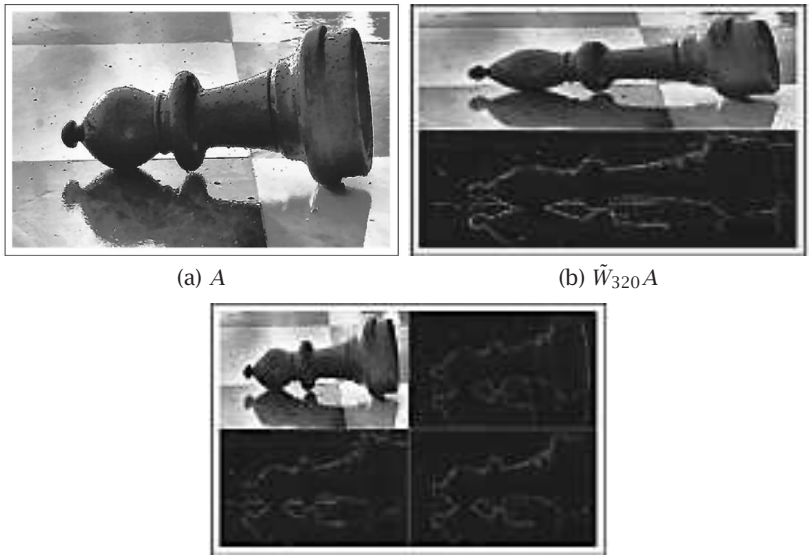
If we view a grayscale digital image as an $M \times N$ matrix A whose entries are integers between 0 and 255 inclusive, where 0 represents black and 255 is white, and integers in between indicate varying degrees of intensities of gray, then application of the one-dimensional Haar wavelet transform $\tilde{W}_M$ acts on the *columns* of A. Each column is transformed to a column whose top half gives an approximation, or *blur*, of the original column and whose bottom half holds the differences or *details* in the data.

We can process the rows of A as well by multiplying $\tilde{W}_M A$ by $\tilde{W}_N^T$. We define the *two-dimensional discrete Haar wavelet transformation* of the matrix A to be $\tilde{W}_M A \tilde{W}_N^T$. Of course, both M and N must be even integers. An image (stored in A), $\tilde{W}_M A$, and $\tilde{W}_M A \tilde{W}_N^T$ are plotted in Figure 1.

We can better understand the image displayed in Figure 1(c) if we express the two–dimensional transformation in block form. Indeed, using (1) and (2), we see that

$$\begin{aligned} \tilde{W}_M A \tilde{W}_N^T &= \left[\begin{array}{c} \tilde{H}_{M/2} \\ \hline \tilde{G}_{M/2} \end{array}\right] A \left[\begin{array}{c|c} \tilde{H}_{N/2}^T & \tilde{G}_{N/2}^T \end{array}\right] \\ &= \left[\begin{array}{c|c} \tilde{H}_{M/2} A \tilde{H}_{N/2}^T & \tilde{H}_{M/2} A \tilde{G}_{N/2}^T \\ \hline \tilde{G}_{M/2} A \tilde{H}_{N/2}^T & \tilde{G}_{M/2} A \tilde{G}_{N/2}^T \end{array}\right] \\ &= \left[\begin{array}{c|c} \mathcal{B} & \mathcal{V} \\ \hline \mathcal{H} & \mathcal{D} \end{array}\right]. \end{aligned}$$

It is a straightforward computation to see that if we partition A into 2×2 blocks $A_{j,k}$, $1 \le j \le M/2$, $1 \le k \le N/2$, the (j, k) elements of $\mathcal{B}$, $\mathcal{V}$, $\mathcal{H}$, $\mathcal{D}$ are



(a) A (b) $\tilde{W}_{320}A$

(c) $\tilde{W}_{320}A\tilde{W}_{512}^T$

Figure 1. A 320 x 512 digital grayscale image, the product $\tilde{W}_M A$, and the two-dimensional discrete Haar wavelet transformation $\tilde{W}_M A \tilde{W}_N^T$.

constructed using the four values in $A_{j,k}$. Indeed, if

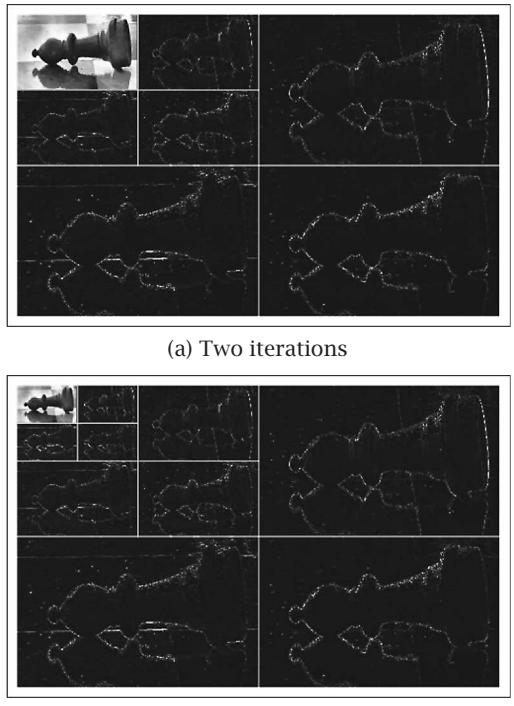
$$A_{j,k} = \begin{bmatrix} a & b \\ c & d \end{bmatrix},$$

then each (j,k) element of $\mathcal{B}$, $\mathcal{V}$, $\mathcal{H}$, $\mathcal{D}$ is

$$\frac{a+b+c+d}{4} \qquad \frac{(a+c)-(b+d)}{4}$$
$$\frac{(a+b)-(c+d)}{4} \qquad \frac{(a+d)-(b+c)}{4},$$

respectively. In this way we see that the elements of $\mathcal{B}$ are simply the averages of the elements in the blocks $A_{j,k}$, and the elements in $\mathcal{V}$, $\mathcal{H}$, and $\mathcal{D}$ are averaged differences in the vertical, horizontal, and diagonal directions, respectively.

If M and N are divisible by 2^p, then we can *iterate* the transformation a total of p times. Figure 1(c) shows one iteration of the discrete Haar wavelet transformation. We can perform a second iteration by applying the two-dimensional discrete Haar wavelet transformation to $\mathcal{B}$. Figure 2 shows two and three iterations of the discrete Haar wavelet transformation applied to A. In applications such as image compression, the iterated transformation produces more details portions, and in the case in which the original data are largely homogeneous, then we can expect a large number of near-zero values in the details portions of the iterated transform. These small values can be quantized to zero, and the data coder should be able to compress the modified transform in a highly efficient manner.



(a) Two iterations

(b) Three iterations

Figure 2. Multiple iterations of the discrete Haar wavelet transformation applied to the matrix A from Figure 1(a).

Alternatively, we can use the averages portion $\mathcal{B}$ in applications as well. Suppose we stored a four-iteration version of an image on a web server. Then

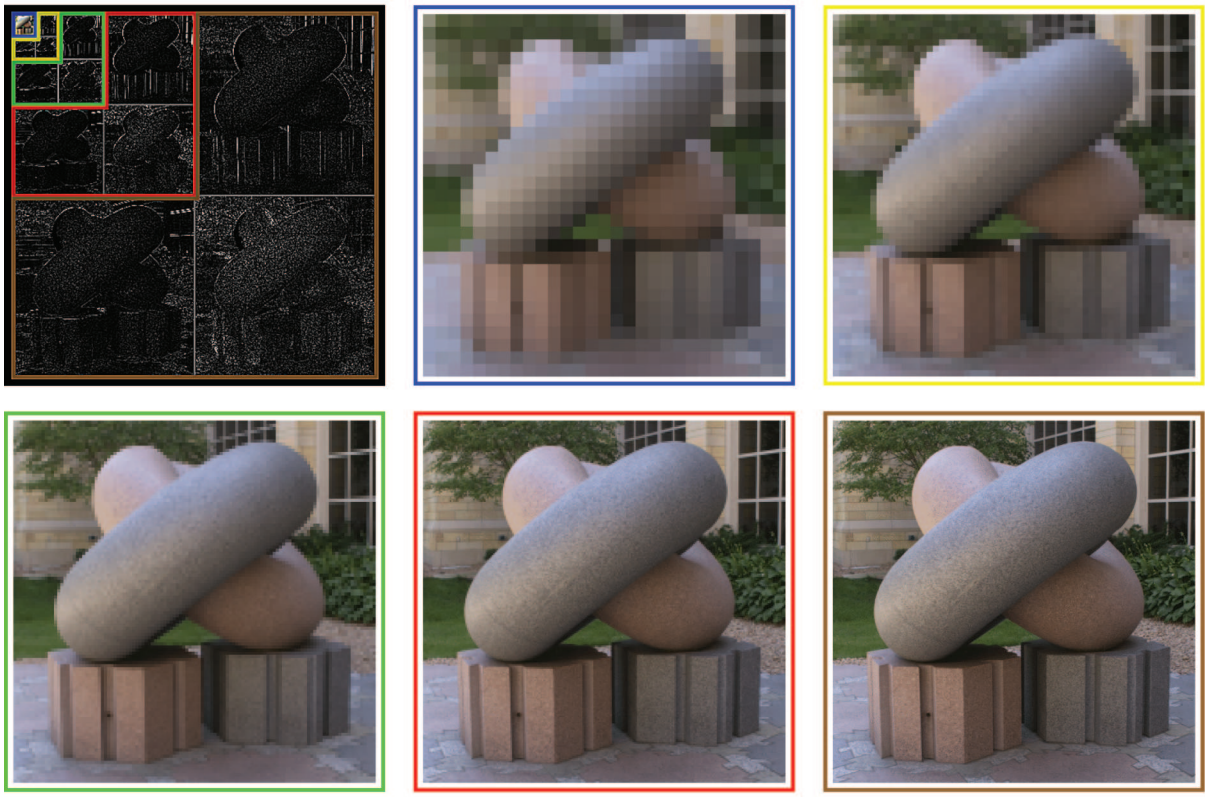


Figure 3. The wavelet transform of a digital image of Helaman Ferguson's *Four Canoes* is shown at the top left. The portion framed in blue is enlarged and transmitted. Next, the portion of the transform outlined in yellow is transmitted and, with $\mathcal{B}_4$, inverted and enlarged to create the image at top right. This process is repeated for the green, red, and brown portions of the transform to progressively create a sequence of images whose resolution increases by a factor of two at each step. The original image, outlined in brown, is at bottom right.

$\mathcal{B}_4$, the fourth iteration averages portion of the transform, could serve as a thumbnail image for the original. If a user requests the original image, we could first transmit $\mathcal{B}_4$ and then progressively send detail portions at each iterative level. The recipient can apply the inverse transform as detail portions are received to sequentially produce a higher-resolution version of the original image. Figure 3 illustrates the process.

Equation (2) tells us that the discrete Haar wavelet transformation is almost orthogonal. Indeed, if we define $W_N = \sqrt{2}\tilde{W}_N$, then $W_N^T = W_N^{-1}$, and our transformation is orthogonal. The inverse of the transformation now has a simple form, which allows us to easily reconstruct the signal from its transformed version.

Relation to Discrete Convolution and Filters

We have motivated the construction of W_N by insisting that W_N transform the original data $\mathbf{x}$ into an averages portion $\mathbf{a}$ and details portion $\mathbf{d}$. We could just as well develop W_N via discrete convolution.

Recall that if $\mathbf{h}$ and $\mathbf{x}$ are bi-infinite vectors, then the convolution of $\mathbf{h}$ with $\mathbf{x}$ is the bi-infinite vector $\mathbf{y}$, denoted by $\mathbf{y} = \mathbf{h} * \mathbf{x}$, where the nth component, $n \in \mathbb{Z}$, of the vector is

$$y_n = \sum_{k\in\mathbb{Z}} h_k x_{n-k}.$$

If we assume that $x_n = 0$ for $n < 1$ and $n > N$, where N is an even integer, then we can form $W_N\mathbf{x}$ by

- Computing $\mathbf{u} = \mathbf{h} * \mathbf{x}$, where the only nonzero values of $\mathbf{h}$ are $h_0 = h_1 = \sqrt{2}/2$.
- Computing $\mathbf{v} = \mathbf{g} * \mathbf{x}$, where the only nonzero values of $\mathbf{g}$ are $g_0 = \sqrt{2}/2$ and $g_1 = -\sqrt{2}/2$.
- *Downsampling* $\mathbf{u}$ and $\mathbf{v}$ by removing the odd components of each vector.
- Truncating the downsampled vectors appropriately to obtain $\mathbf{a}$ and $\mathbf{d}$.

The vectors $\mathbf{h}$ and $\mathbf{g}$ are often called *filters*. In fact, $\mathbf{h}$ and $\mathbf{g}$ are special kinds of filters. Let $\mathbf{x}$ be a vector whose elements $x_k = 1$, $k \in \mathbb{Z}$, and suppose

$\mathbf{y}$ is a vector whose elements are $y_k = (-1)^k$, $k \in \mathbb{Z}$. Then $\mathbf{h} * \mathbf{x} = \sqrt{2}\mathbf{x}$ and $\mathbf{h} * \mathbf{y}$ is the bi-infinite zero vector. Thus $\mathbf{h}$ is an example of a *lowpass filter*—it passes low-frequency signals through largely unchanged but attenuates the amplitudes of high-frequency data. In a similar manner we see that $\mathbf{g} * \mathbf{x}$ is the bi-infinite zero vector and $\mathbf{g} * \mathbf{y} = \sqrt{2}\mathbf{y}$. Here $\mathbf{g}$ is an example of a *highpass filter*—it allows high-frequency signals to pass largely unchanged but attenuates the amplitudes of low-frequency data. Thus in filtering terms, the discrete Haar wavelet transform is constructed by applying a lowpass filter and a highpass filter to the input data, downsampling both results, and then appropriately truncating the downsampled vectors. We will learn that all discrete wavelet transforms are built from a lowpass (*scaling*) filter $\mathbf{h}$ and a highpass (*wavelet*) filter. Fourier series are valuable tools for constructing these filters.

Fourier Series and Discrete Wavelet Transformations

A course on discrete wavelet transformations is an excellent venue for the introduction of Fourier series into the undergraduate curriculum. There are two main uses of Fourier series in such a course. We can determine whether a finite-length filter $\mathbf{f} = [f_0, \ldots, f_L]^T$ is lowpass or highpass by plotting the modulus of the Fourier series

$$F(\omega) = \sum_{k=0}^{L} f_k e^{ik\omega},$$

and, as we will see later, we can characterize a (bi)orthogonal discrete wavelet transformation by investigating an identity related to the Fourier series of the scaling filter(s).

For the discrete Haar wavelet transformation, the finite-length filters

$$\mathbf{h} = [h_0, h_1]^T = \left[\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}\right]^T$$

$$\mathbf{g} = [g_0, g_1]^T = \left[\frac{\sqrt{2}}{2}, -\frac{\sqrt{2}}{2}\right]^T$$

give rise to the Fourier series

$$H(\omega) = \frac{\sqrt{2}}{2} + \frac{\sqrt{2}}{2}e^{i\omega}$$

$$= \sqrt{2}e^{i\omega/2}\cos\left(\frac{\omega}{2}\right) \tag{3}$$

$$G(\omega) = \frac{\sqrt{2}}{2} - \frac{\sqrt{2}}{2}e^{i\omega}$$

$$= -\sqrt{2}ie^{i\omega/2}\sin\left(\frac{\omega}{2}\right) \tag{4}$$

for $\omega \in \mathbb{R}$. To determine the nature of the filter, we plot $|H(\omega)|$ and $|G(\omega)|$. Since $|H(\omega)|$ and $|G(\omega)|$ are both 2π-periodic, even functions, we plot them on the interval $[0, \pi]$. The functions are displayed in Figure 4.

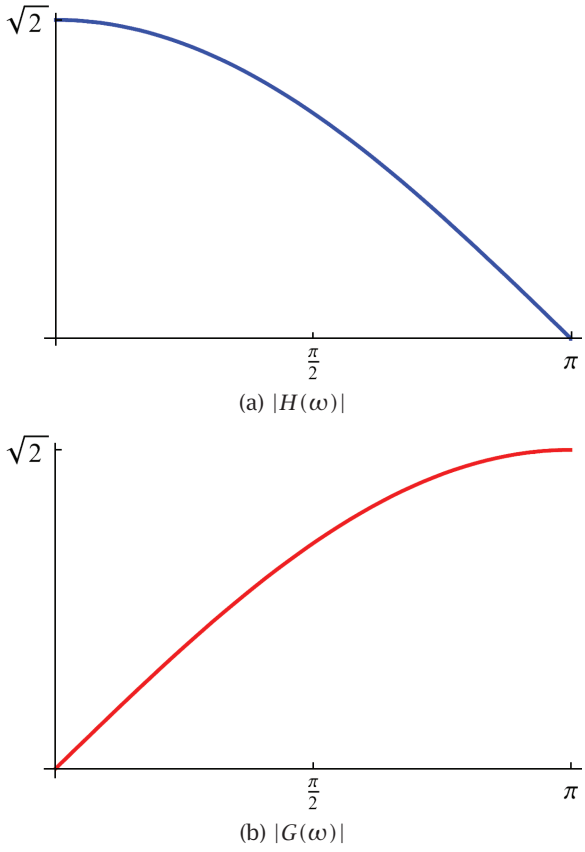


Figure 4. The functions $|H(\omega)|$ and $|G(\omega)|$.

At the lowest frequency, $\omega = 0$, $|H(0)| = \sqrt{2}$. At the highest frequency, $\omega = \pi$, $|H(\pi)| = 0$. In the frequency domain, these two conditions indicate that $\mathbf{h}$ is a lowpass filter. In a similar manner, $|G(0)| = 0$ and $|G(\pi)| = \sqrt{2}$ are conditions that imply that $\mathbf{g}$ is a highpass filter.

For $\omega \in \mathbb{R}$, we use (3) to write

$$|H(\omega)|^2 + |H(\omega + \pi)|^2 =$$

$$2\cos^2\left(\frac{\omega}{2}\right) + 2\cos^2\left(\frac{\omega + \pi}{2}\right) = 2. \tag{5}$$

In a similar manner, we use (3) and (4) to obtain

$$|G(\omega)|^2 + |G(\omega + \pi)|^2 = 2 \tag{6}$$

and

$$H(\omega)\overline{G(\omega)} + H(\omega + \pi)\overline{G(\omega + \pi)} = 0. \tag{7}$$

In the next section, we will see that the first step in a general strategy for finding longer, more sophisticated filters $\mathbf{h}$ and $\mathbf{g}$ is to find $\mathbf{h}$ so that (5) is satisfied and then make a judicious choice of $\mathbf{g}$ so that (6) and (7) hold as well. The additional lowpass conditions $|H(0)| = \sqrt{2}$ and $|H(\pi)| = 0$ with our choice of $\mathbf{g}$ ensure that the highpass conditions $|G(0)| = 0$, $|G(\pi)| = \sqrt{2}$ hold, and in this way, we see that all lowpass, highpass, and matrix orthogonality conditions placed on filters

h and **g** can be completely characterized using Fourier series.

Daubechies Orthogonal Scaling Filters

Suppose we apply the one-dimensional discrete Haar wavelet transformation to the vector $\mathbf{v} = [0, 0, 100, 100]^T$. We obtain the transformed vector $\sqrt{2}\,[0, 100|\, 0, 0]^T$. The differences portion of the transformed vector consists only of zeros, missing the large "edge" between 0 and 100 in **v**. The problem is that the Haar scaling/wavelet filter pair is too short. Let's consider longer filter pairs. These filters and the corresponding wavelets were discovered by Ingrid Daubechies (see, e. g., [4]).

We wish to construct an orthogonal matrix W_N that uses longer scaling and wavelet filters. It turns out that we need to consider even-length filters, so let's start with a scaling filter $\mathbf{h} = [h_0, h_1, h_2, h_3]^T$ and wavelet filter $\mathbf{g} = [g_0, g_1, g_2, g_3]^T$. We insist that

$$|H(0)| = |\,h_0 + h_1 + h_2 + h_3| = \sqrt{2} \tag{8}$$

$$H(\pi) = h_0 - h_1 + h_2 - h_3 = 0 \tag{9}$$

and

$$G(0) = g_0 + g_1 + g_2 + g_3 = 0 \tag{10}$$

$$|\,G(\pi)| = |\,g_0 - g_1 + g_2 - h_3| = \sqrt{2} \tag{11}$$

so that the filters **h**, **g** are lowpass and highpass, respectively.

We also wish to retain the structure of the transformation by suitably truncating two downsampled, convolution products. In this case, the longer lengths of our filters force us to "wrap" rows at the bottom of each block $H_{N/2}$, $G_{N/2}$ of the matrix W_N. For example, in the case in which $N = 8$, W_N takes the form

$$W_8 = \left[\begin{array}{cccccccc} h_3 & h_2 & h_1 & h_0 & 0 & 0 & 0 & 0 \\ 0 & 0 & h_3 & h_2 & h_1 & h_0 & 0 & 0 \\ 0 & 0 & 0 & 0 & h_3 & h_2 & h_1 & h_0 \\ h_1 & h_0 & 0 & 0 & 0 & 0 & h_3 & h_2 \\ \hline g_3 & g_2 & g_1 & g_0 & 0 & 0 & 0 & 0 \\ 0 & 0 & g_3 & g_2 & g_1 & g_0 & 0 & 0 \\ 0 & 0 & 0 & 0 & g_3 & g_2 & g_1 & g_0 \\ g_1 & g_0 & 0 & 0 & 0 & 0 & g_3 & g_2 \end{array}\right]. \tag{12}$$

The wrapping rows cause problems in applications since it inherently assumes that data are periodic, but on the other hand, it simplifies our final requirement that W_N be an orthogonal matrix. In this case, we must have

$$h_0^2 + h_1^2 + h_2^2 + h_3^2 = 1 \tag{13}$$

$$h_0 h_2 + h_1 h_3 = 0. \tag{14}$$

It turns out that if real numbers $h_0, \ldots, h_3$ satisfy (9), (13), and (14), then (8) holds. We will choose the positive square root so that

$$h_0 + h_1 + h_2 + h_3 = \sqrt{2}. \tag{15}$$

We now set $\mathbf{g} = [h_3, -h_2, h_1, -h_0]^T$. It is straightforward to show that if (9), (13), (14), and (15) hold, then the highpass conditions (10), (11) are satisfied and

$$\begin{aligned} g_0^2 + g_1^2 + g_2^2 + g_3^2 &= 1 \\ g_0 g_2 + g_1 g_3 &= 0. \end{aligned}$$

Moreover, we can show that

$$\begin{aligned} h_0 g_0 + h_1 g_1 + h_2 g_2 + h_3 g_3 &= 0 \\ h_1 g_3 + h_0 g_2 &= 0 \\ h_3 g_1 + h_2 g_0 &= 0 \end{aligned}$$

so that W_N is an orthogonal matrix. However, there are infinitely many solutions to the system

$$\boxed{\begin{array}{c} h_0^2 + h_1^2 + h_2^2 + h_3^2 = 1 \\ h_0 h_2 + h_1 h_3 = 0 \\ h_0 + h_1 + h_2 + h_3 = \sqrt{2} \\ h_0 - h_1 + h_2 - h_3 = 0 \end{array}}. \tag{16}$$

Here is where the Fourier series approach really makes an impact. Daubechies chose to impose the additional condition $H'(\pi) = 0$. This condition has the effect of "flattening" the graph of the modulus $|H(\omega)|$ even more at $\omega = \pi$; thus intuitively the filter **h** does an even better job of attenuating the amplitudes of the high-frequency portions of the input data. If we expand the left-hand side of $H'(\pi) = 0$, we obtain the following linear equation

$$h_1 - 2h_2 + 3h_3 = 0, \tag{17}$$

and we add (17) to our system (16).

It turns out that there are two real solutions to the system, one of which is a reflection of the other. The solution can be solved by hand (see, e. g., [12]) to obtain the *Daubechies four-term scaling filter*:

$$\begin{aligned} h_0 &= \frac{1+\sqrt{3}}{4\sqrt{2}}, \quad h_1 = \frac{3+\sqrt{3}}{4\sqrt{2}} \\ h_2 &= \frac{3-\sqrt{3}}{4\sqrt{2}}, \quad h_3 = \frac{1-\sqrt{3}}{4\sqrt{2}}. \end{aligned}$$

The modulus of $H(\omega)$ is plotted in Figure 5. Compare this graph with that of Figure 4.

In order to construct longer even-length filters $\mathbf{h} = [h_0, \ldots, h_L]^T$, L odd, we must write down general orthogonality conditions and lowpass conditions. We can ensure the orthogonality of W_N using the following theorem.

Theorem 1. *Suppose* $A(\omega) = \sum_{k\in\mathbb{Z}} a_k e^{ik\omega}$ *and* $B(\omega) = \sum_{k\in\mathbb{Z}} b_k e^{ik\omega}$ *are Fourier series. Then*

$$|\,A(\omega)|^2 + |\,A(\omega+\pi)|^2 = 2 \tag{18}$$

if and only if

$$\sum_{k\in\mathbb{Z}} a_k a_{k-2n} = \delta_{0,n} \tag{19}$$

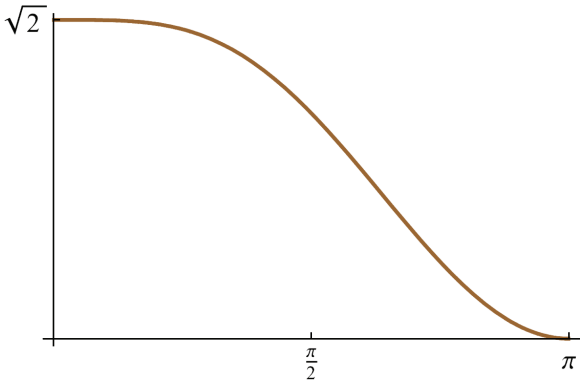


Figure 5. $|H(\omega)|$ for the length four Daubechies scaling filter.

for $n \in \mathbb{Z}$, and

$$A(\omega)\overline{B(\omega)} + A(\omega+\pi)\overline{B(\omega+\pi)} = 0 \tag{20}$$

if and only if

$$\sum_{k\in\mathbb{Z}} a_k b_{k-2n} = 0 \tag{21}$$

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

The proof of this theorem is straightforward. Note that equations (19) and (21) guarantee orthogonality of the matrix W_N.

Suppose $H(\omega)$ satisfies (18). If we take

$$G(\omega) = -e^{iL\omega}\overline{H(\omega+\pi)}, \tag{22}$$

then $G(\omega)$ satisfies (18), and $H(\omega)$, $G(\omega)$ satisfy (20). It is an easy exercise to show that the Fourier coefficients of $G(\omega)$ are $g_k = (-1)^k h_{1-k}$, $k \in \mathbb{Z}$.

The condition $H'(\pi) = 0$ leads naturally to the following generalization: If we want to produce longer scaling filters $\mathbf{h}$, each time we increase the filter length by two, we require an additional derivative condition at π. For example, the Daubechies scaling filter of length six is, in part, constructed by requiring the conditions that $H'(\pi) = 0$ and $H''(\pi) = 0$. Each time an additional derivative condition is imposed at π, the modulus of the corresponding Fourier series becomes flatter at the highest frequency $\omega = \pi$. So the general derivative conditions are

$$H^{(m)}(\pi) = 0, \quad m = 0, \ldots, \frac{L-1}{2}. \tag{23}$$

We also add the lowpass condition

$$H(0) = \sum_{k=0}^{L} h_k = \sqrt{2} \tag{24}$$

so that our general system is given by (19), (23), and (24). Daubechies proved that there are $M = 2^{\lfloor \frac{L+2}{4} \rfloor}$ real-valued scaling filters $\mathbf{h} = [h_0, h_1, \ldots, h_L]^T$, L is odd, that satisfy this system. Moreover, if we construct $G(\omega)$ using (22), then $G(0) = 0$ and $|G(\pi)| = \sqrt{2}$, as desired.

While this derivation of the system is easy to understand, solving it numerically for large filter lengths can be difficult. It should be noted that Daubechies's classical derivation of these filters is centered around the construction of a trigonometric polynomial that leads to the solution of (18). Particular roots of this polynomial are then used to find a Fourier series $H(\omega)$ that solves (23). This process (see [5] or [9]) works well for filters of long length, but the mathematics required to understand it may be better suited for advanced undergraduate or beginning graduate students.

Orthogonal Transforms in Applications

Now that we have the means for constructing orthogonal filters $\mathbf{h}$, $\mathbf{g}$ of any odd length, it is natural to consider the advantages and disadvantages of their implementation in applications.

Let's first consider filter length. Certainly, the time to compute $W_N\mathbf{v}$ is inversely proportional to the length of $\mathbf{h}$. On the other hand, it can be shown [4] that the derivative conditions (23) lead to filters $\mathbf{g}$ that annihilate (modulo wrapping rows) data obtained by uniformly sampling a polynomial of degree $(L-1)/2$.

In image compression we typically transform the image, then quantize the transform, and finally encode the result. It can be shown that if the transform is orthogonal, then the error incurred when quantizing the transform coefficients is exactly the same as the error between the original image and the compressed image.

There are disadvantages to using orthogonal transformations. In lossless image compression, the quantization step is omitted. In this case, we need to use a transformation that maps integers to integers. It is possible to modify orthogonal transforms (see [3]) and produce integer-to-integer mappings, but it is easier to modify the biorthogonal transformations described in the next section.

Unfortunately, the wrapping rows that appear in the W_N constructed from Daubechies scaling and wavelet filters pose a problem in many applications. In (12), there is one wrapping row in each block of the transform. Figure 6 illustrates the problem when the discrete Daubechies wavelet transformation is applied to $\mathbf{v} \in \mathbb{R}^{16}$, where $v_k = k$, $k = 1, \ldots, 16$. The wrapping rows cause the first two elements to be combined with the last two elements of $\mathbf{v}$ to form the last weighted average and last weighted difference. If the data are periodic, this makes perfect sense, but in many signal/image processing applications, the data are not periodic. The wrapping row problem, caused by truncating downsampled convolution products, is typical in applied mathematics.

Biorthogonal Scaling Filters

Fortunately, there is a way to deal with the wrapping row problem, and that is to develop filters that are *symmetric*.

An odd-length filter $\mathbf{h}$ is called symmetric if $h_k = h_{-k}$, while an even-length filter is called symmetric if $h_k = h_{1-k}$. Daubechies [5] proved that the only symmetric, finite length, orthogonal filter is the Haar filter, and we have already discussed some of its limitations. How can we produce symmetric filters while preserving some of the desirable properties of orthogonal filters? These desirable properties are finite length (computational speed), orthogonality (ease of inverse), and the ability to produce a good approximation of the original data with the scaling filter. Since the inverse need only be computed once, it was Daubechies's idea to relinquish orthogonality and to construct instead a discrete *biorthogonal* wavelet transformation.

The idea is to construct *two* sets of filters instead of one. In other words, we construct two wavelet transform matrices $\tilde{W}_N$ and W_N so that $\tilde{W}_N^{-1} = W_N^T$. In block form, we have

$$\begin{aligned}\tilde{W}_N W_N^T &= \left[\begin{array}{c} \tilde{H}_{N/2} \\ \hline \tilde{G}_{N/2} \end{array}\right] \cdot \left[\begin{array}{c|c} H_{N/2}^T & G_{N/2}^T \end{array}\right] \\ &= \left[\begin{array}{c|c} \tilde{H}_{N/2}H_{N/2}^T & \tilde{H}_{N/2}G_{N/2}^T \\ \hline \tilde{G}_{N/2}H_{N/2}^T & \tilde{G}_{N/2}G_{N/2}^T \end{array}\right] \\ &= \left[\begin{array}{c|c} I_{N/2} & 0_{N/2} \\ \hline 0_{N/2} & I_{N/2} \end{array}\right].\end{aligned}$$

Let $\tilde{\mathbf{h}}$ and $\mathbf{h}$ denote two filters and suppose $H(\omega)$, $\tilde{H}(\omega)$ are the associated Fourier series. We require both filters to be lowpass so we have $\tilde{H}(0) = H(0) = \sqrt{2}$ and $\tilde{H}(\pi) = H(\pi) = 0$. Instead of orthogonality, we insist that the filters satisfy the general biorthogonal condition

$$\tilde{H}(\omega)\overline{H(\omega)} + \tilde{H}(\omega+\pi)\overline{H(\omega+\pi)} = 2. \tag{25}$$

Although the filters are no longer orthogonal, they are close to orthogonal, and a proof very similar to that of Theorem 1 shows that (25) holds if and only if

$$\sum_{k\in\mathbb{Z}} \tilde{h}_k h_{k-2n} = \delta_{0,n}, \tag{26}$$

where $n \in \mathbb{Z}$. Equation (26) ensures us, *if* we wrap rows, that $\tilde{H}_{N/2}H_{N/2}^T = I_{N/2}$. If we take

$$\tilde{G}(\omega) = -e^{i\omega}\overline{H(\omega+\pi)} \tag{27}$$

$$G(\omega) = -e^{i\omega}\overline{\tilde{H}(\omega+\pi)}, \tag{28}$$

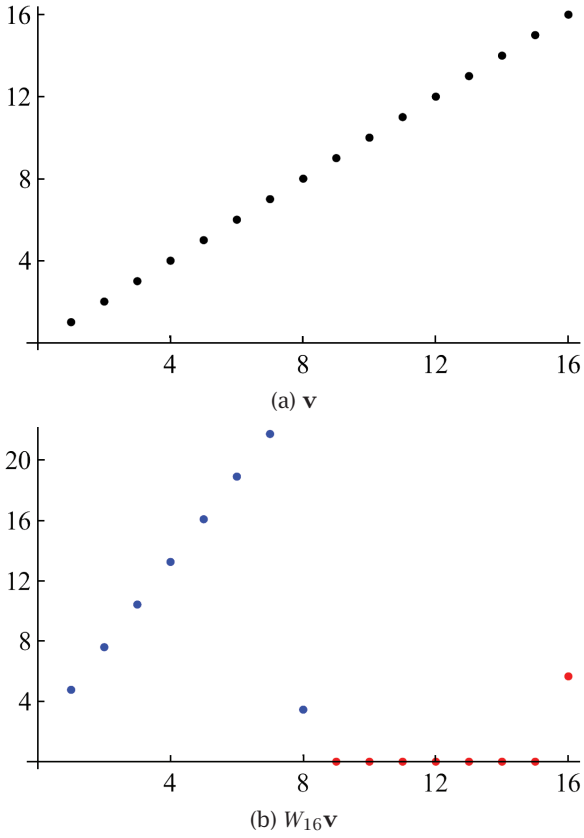


Figure 6. The discrete wavelet transform, constructed using the Daubechies length four scaling filter, applied to $\mathbf{v}$.

then we can easily verify that $\tilde{G}(0) = G(0) = 0$, $\tilde{G}(\pi) = G(\pi) = \sqrt{2}$, and

$$\tilde{G}(\omega)\overline{G(\omega)} + \tilde{G}(\omega+\pi)\overline{G(\omega+\pi)} = 2 \tag{29}$$

$$\tilde{H}(\omega)\overline{G(\omega)} + \tilde{H}(\omega+\pi)\overline{G(\omega+\pi)} = 0 \tag{30}$$

$$\tilde{G}(\omega)\overline{H(\omega)} + \tilde{G}(\omega+\pi)\overline{H(\omega+\pi)} = 0. \tag{31}$$

Equation (29) implies that $\tilde{G}_{N/2}G_{N/2}^T = I_{N/2}$, and (30), (31), in conjunction with (20) from Theorem 1, establishes $\tilde{H}_{N/2}G_{N/2}^T = \tilde{G}_{N/2}H_{N/2}^T = 0_{N/2}$. We can also expand the Fourier series in (27) and (28) to obtain the wavelet filter coefficients

$$\tilde{g}_k = (-1)^k h_{1-k} \quad \text{and} \quad g_k = (-1)^k \tilde{h}_{1-k}, \tag{32}$$

where $1-k$ ranges over the nonzero indices of $\tilde{\mathbf{h}}$, $\mathbf{h}$. Thus, if we solve (25), we can find all the filters we need to build $\tilde{W}_N$ and W_N. Of course (25) represents a quadratic system in terms of the scaling filter coefficients, and such systems are difficult to solve. Daubechies's solution to this problem was to simply pick the scaling filter $\tilde{\mathbf{h}}$ and then solve the *linear system* (25) for the scaling filter $\mathbf{h}$. Moreover, since we have given up orthogonality in this new construction, we can

insist that both scaling filters be symmetric. Let's look at an example.

Example 1. Suppose we want $\tilde{\mathbf{h}}$ to be a symmetric, length three, lowpass filter $\tilde{\mathbf{h}} = [\tilde{h}_{-1}, \tilde{h}_0, \tilde{h}_1]^T = [\tilde{h}_1, \tilde{h}_0, \tilde{h}_1]^T$. We seek two numbers $\tilde{h}_0$, $\tilde{h}_1$ so that $\tilde{H}(0) = \tilde{h}_1 + \tilde{h}_0 + \tilde{h}_1 = \sqrt{2}$ and $\tilde{H}(\pi) = -\tilde{h}_1 + \tilde{h}_0 - \tilde{h}_1 = 0$. The filter[1] $\tilde{\mathbf{h}}$ we seek is

$$[\tilde{h}_{-1}, \tilde{h}_0, \tilde{h}_1]^T = \frac{\sqrt{2}}{4}[1, 2, 1]^T.$$

To find a symmetric filter $\mathbf{h}$, we must satisfy the biorthogonality condition (25), together with the lowpass constraints $H(0) = \sqrt{2}$ and $H(\pi) = 0$. It turns out that $\mathbf{h}$ must have an odd length of $5, 9, 13, \ldots$. We choose symmetric $\mathbf{h} = [h_2, h_1, h_0, h_1, h_2]^T$ and use the lowpass condition $H(\pi) = 0$ in conjunction with (26) to obtain the system

$$\begin{aligned} h_0 - 2h_1 + 2h_2 &= 0 \\ h_0 + \ h_1 \qquad &= \sqrt{2} \\ h_1 + 2h_2 &= 0. \end{aligned}$$

Solving this system gives $h_2 = -\frac{\sqrt{2}}{8}$, $h_1 = \frac{\sqrt{2}}{4}$, and $h_0 = \frac{3\sqrt{2}}{4}$. We can next use (32) to find the coefficients of the wavelet filters $\mathbf{g}$ and $\tilde{\mathbf{g}}$. We have:

$\tilde{\mathbf{g}}$	$\mathbf{g}$
$\tilde{g}_{-1} = \frac{\sqrt{2}}{8}$	
$\tilde{g}_0 = \frac{\sqrt{2}}{4}$	$g_0 = \frac{\sqrt{2}}{4}$
$\tilde{g}_1 = -\frac{3\sqrt{2}}{4}$	$g_1 = -\frac{\sqrt{2}}{2}$
$\tilde{g}_2 = \frac{\sqrt{2}}{4}$	$g_2 = \frac{\sqrt{2}}{4}$
$\tilde{g}_3 = \frac{\sqrt{2}}{8}$	

Note that there is some symmetry in the highpass filters: $\tilde{g}_k = \tilde{g}_{2-k}$ and $g_k = g_{2-k}$. Here are examples of the biorthogonal transformation matrices with $N = 8$. We form the matrices by placing the 0th element in the upper left corner of each block with positive indexed elements placed consecutively to the right and negative indexed elements placed consecutively to the left with wrapping if necessary.

$$\tilde{W}_8 = \left[\begin{array}{cccccccc} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{4} & 0 & 0 & 0 & 0 & 0 & \frac{\sqrt{2}}{4} \\ 0 & \frac{\sqrt{2}}{4} & \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{4} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{\sqrt{2}}{4} & \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{4} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{\sqrt{2}}{4} & \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{4} \\ \hline \frac{\sqrt{2}}{4} & -\frac{3\sqrt{2}}{4} & \frac{\sqrt{2}}{4} & \frac{\sqrt{2}}{8} & 0 & 0 & 0 & \frac{\sqrt{2}}{8} \\ 0 & \frac{\sqrt{2}}{8} & \frac{\sqrt{2}}{4} & -\frac{3\sqrt{2}}{4} & \frac{\sqrt{2}}{4} & \frac{\sqrt{2}}{8} & 0 & 0 \\ 0 & 0 & 0 & \frac{\sqrt{2}}{8} & \frac{\sqrt{2}}{4} & -\frac{3\sqrt{2}}{4} & \frac{\sqrt{2}}{4} & \frac{\sqrt{2}}{8} \\ \frac{\sqrt{2}}{4} & \frac{\sqrt{2}}{8} & 0 & 0 & 0 & \frac{\sqrt{2}}{8} & \frac{\sqrt{2}}{4} & -\frac{3\sqrt{2}}{4} \end{array}\right]$$

[1] *Do you see how Pascal's triangle can be used to generate symmetric, lowpass filters?*

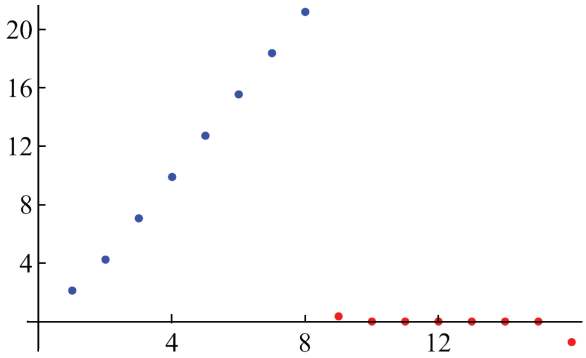


Figure 7. The modified biorthogonal wavelet transformation applied to the vector $\mathbf{v}$ from Figure 6(a).

and

$$W_8 = \left[\begin{array}{cccccccc} \frac{3\sqrt{2}}{4} & \frac{\sqrt{2}}{4} & -\frac{\sqrt{2}}{8} & 0 & 0 & 0 & -\frac{\sqrt{2}}{8} & \frac{\sqrt{2}}{4} \\ -\frac{\sqrt{2}}{8} & \frac{\sqrt{2}}{4} & \frac{3\sqrt{2}}{4} & \frac{\sqrt{2}}{4} & -\frac{\sqrt{2}}{8} & 0 & 0 & 0 \\ 0 & 0 & -\frac{\sqrt{2}}{8} & \frac{\sqrt{2}}{4} & \frac{3\sqrt{2}}{4} & \frac{\sqrt{2}}{4} & -\frac{\sqrt{2}}{8} & 0 \\ -\frac{\sqrt{2}}{8} & 0 & 0 & 0 & -\frac{\sqrt{2}}{8} & \frac{\sqrt{2}}{4} & \frac{3\sqrt{2}}{4} & \frac{\sqrt{2}}{4} \\ \hline \frac{\sqrt{2}}{4} & -\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{4} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{\sqrt{2}}{4} & -\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{4} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{\sqrt{2}}{4} & -\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{4} & 0 \\ \frac{\sqrt{2}}{4} & 0 & 0 & 0 & 0 & 0 & \frac{\sqrt{2}}{4} & -\frac{\sqrt{2}}{2} \end{array}\right]$$

It is straightforward to verify that $\tilde{W}_8^{-1} = W_8^T$. Note that $\tilde{W}_8$ and W_8 still have wrapping rows, but we can exploit the symmetry of the scaling filters to construct a transformation that diminishes the adverse effects of the wrapping rows. The idea is quite simple, and we illustrate it now with an example.

Example 2. Consider again the vector $\mathbf{v} \in \mathbb{R}^{16}$ where $v_k = k$. We form the vector $\mathbf{v}_p \in \mathbb{R}^{30}$ where

$$\mathbf{v}_p = [v_1, \ldots, v_{16} | v_{15}, v_{14}, \ldots, v_2]^T.$$

We next compute $\left[\begin{array}{c} \mathbf{a}_p \\ \hline \mathbf{d}_p \end{array}\right] = \tilde{W}_{30}\mathbf{v}_p$. The periodic nature of $\mathbf{v}_p$ means that the weighted averages and differences formed by the wrapping rows use consecutive elements from $\mathbf{v}$. For example, the first element of $\mathbf{a}_p$ is $\frac{\sqrt{2}}{4}v_2 + \frac{\sqrt{2}}{2}v_1 + \frac{\sqrt{2}}{4}v_2$ instead of the first element of $\tilde{W}_{16}$, which is $\frac{\sqrt{2}}{4}v_{16} + \frac{\sqrt{2}}{2}v_1 + \frac{\sqrt{2}}{4}v_2$. The elements of the transform that do not involve wrapping rows are the same as elements in $\tilde{W}_{16}\mathbf{v}$. We keep the first eight elements of $\mathbf{a}_p$ and $\mathbf{d}_p$ and concatenate them to form our transform. The result is plotted in Figure 7. Compare this result to that plotted in Figure 6(b).

We can certainly construct biorthogonal filter pairs of different lengths—we can either select $\tilde{\mathbf{h}}$, then set up and solve a linear system for $\mathbf{h}$, or we can use the closed formulas for $\tilde{H}(\omega)$ and $H(\omega)$

obtained by Daubechies in [6]. The biorthogonal filter pair from the above example is particularly interesting since it plays a significant role in the JPEG2000 lossless image compression standard. Next we present one more modification of the transformation from Example 2 to understand how the discrete biorthogonal wavelet transformation is used in the JPEG2000 image compression standard.

Discrete Wavelet Transformations and JPEG2000

In 1992 JPEG (Joint Photographic Image Experts Group) became an international standard for compressing digital images. In the late 1990s work began to improve the JPEG compression standard, and a new algorithm was developed. This algorithm uses the biorthogonal wavelet transform instead of the discrete cosine transform. Because of patent issues, JPEG2000 is not yet supported by popular web browsers, but it is likely that once the patent issues are resolved, JPEG2000 will replace JPEG as the most popular standard for image compression.

The basic JPEG2000 algorithm works as follows. Let us assume that we have a grayscale image A. We first subtract 128 from each entry of A. We apply as many iterations of the modified biorthogonal wavelet transformation (see Example 2) as we wish (and as possible, depending on the dimensions of the image). The filter pair used in the discrete biorthogonal wavelet transform depends on whether we choose to perform *lossy* or *lossless* compression. In the case of the former, the transformed image is quantized with many elements either reduced in magnitude or converted to zero. Of course, once we quantize the transformed image, we have no hope of recovering the original image, but this loss of resolution is not as important as the coder's ability to compress the quantized transform. In lossless compression, no quantization is performed. In both cases, the next step is to code the data. JPEG2000 uses an arithmetic coding algorithm known as Embedded Block Coding with Optimized Truncation (EBCOT) to actually encode the (quantized) transformed data. This last step is the actual compression step. After transmission of the data, the original image can be reconstructed by decoding, then applying the inverse transformation as many times as iterated. With lossless compression, we can recover the image exactly.

One of the many advantages of JPEG2000 over JPEG is its ability to allow the user to perform the lossless compression. The biorthogonal filter pair for this task is a modified version of the $\tilde{\mathbf{h}}$ and $\mathbf{h}$ from Example 1. The filter $\mathbf{h}$ is multiplied by $\sqrt{2}/2$ and $\tilde{\mathbf{h}}$ is multiplied by $\sqrt{2}$. Moreover, the filters are "reversed"—$\mathbf{h}$ is used to build $\tilde{W}_N$. For example, the modified transform for vectors of length 8 is:

$$\tilde{W}_8 = \left[\begin{array}{cccccccc} \frac{3}{4} & \frac{1}{4} & -\frac{1}{8} & 0 & 0 & 0 & -\frac{1}{8} & \frac{1}{4} \\ -\frac{1}{8} & \frac{1}{4} & \frac{3}{4} & \frac{1}{4} & -\frac{1}{8} & 0 & 0 & 0 \\ 0 & 0 & -\frac{1}{8} & \frac{1}{4} & \frac{3}{4} & \frac{1}{4} & -\frac{1}{8} & 0 \\ -\frac{1}{8} & 0 & 0 & 0 & -\frac{1}{8} & \frac{1}{4} & \frac{3}{4} & \frac{1}{4} \\ \hline -\frac{1}{2} & 1 & -\frac{1}{2} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -\frac{1}{2} & 1 & -\frac{1}{2} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -\frac{1}{2} & 1 & -\frac{1}{2} & 0 \\ -\frac{1}{2} & 0 & 0 & 0 & 0 & 0 & -\frac{1}{2} & 1 \end{array}\right].$$

We use the ideas of Example 2 to further modify the transformation so that it better handles output produced by wrapping rows. This modification is equivalent to computing $\tilde{W}_8^p$ to $\mathbf{v}$ where

$$\tilde{W}_8^p = \left[\begin{array}{cccccccc} \frac{3}{4} & \frac{1}{2} & -\frac{1}{4} & 0 & 0 & 0 & 0 & 0 \\ -\frac{1}{8} & \frac{1}{4} & \frac{3}{4} & \frac{1}{4} & -\frac{1}{8} & 0 & 0 & 0 \\ 0 & 0 & -\frac{1}{8} & \frac{1}{4} & \frac{3}{4} & \frac{1}{4} & -\frac{1}{8} & 0 \\ 0 & 0 & 0 & 0 & -\frac{1}{8} & \frac{1}{4} & \frac{5}{8} & \frac{1}{4} \\ \hline -\frac{1}{2} & 1 & -\frac{1}{2} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -\frac{1}{2} & 1 & -\frac{1}{2} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -\frac{1}{2} & 1 & -\frac{1}{2} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 \end{array}\right]. \tag{33}$$

This modified transform is still invertible, and it now does a good job of handling edge effects. However, the coding step of the JPEG2000 algorithm works best if the transformation maps integers to integers. How can we modify (33) for general size N so that it maps integers to integers? We could multiply the transformed data by 8, but that increases the range of the output, which is not desirable for coders. The algorithm used by JPEG2000 for lossless compression is called *lifting* and is due to Sweldens [11].

The Lifting Scheme

The idea behind lifting is quite simple. For a nice tutorial, see [10], and more details are provided in the books [12, 8].

To understand how lifting works, we apply (33) to $\mathbf{v} \in \mathbb{Z}^8$. The resulting product is $\tilde{W}_8^p \mathbf{v} =$

$$\left[\begin{array}{c} -\frac{1}{8}v_3+\frac{1}{4}v_2+\frac{3}{4}v_1+\frac{1}{4}v_2-\frac{1}{8}v_3 \\ -\frac{1}{8}v_1+\frac{1}{4}v_2+\frac{3}{4}v_3+\frac{1}{4}v_4-\frac{1}{8}v_5 \\ -\frac{1}{8}v_3+\frac{1}{4}v_4+\frac{3}{4}v_5+\frac{1}{4}v_6-\frac{1}{8}v_7 \\ -\frac{1}{8}v_5+\frac{1}{4}v_6+\frac{3}{4}v_7+\frac{1}{4}v_8-\frac{1}{8}v_7 \\ \hline -\frac{1}{2}v_1+v_2-\frac{1}{2}v_3 \\ -\frac{1}{2}v_3+v_4-\frac{1}{2}v_5 \\ -\frac{1}{2}v_5+v_6-\frac{1}{2}v_7 \\ -\frac{1}{2}v_7+v_8-\frac{1}{2}v_7 \end{array}\right] = \left[\begin{array}{c} \mathbf{a} \\ \hline \mathbf{d} \end{array}\right], \tag{34}$$

where we have rewritten a_1, a_4, and d_4 so that they display the same symmetry as the other elements in the product. Note that the center value of the

elements of $\mathbf{a}/(\mathbf{d})$ are built from the odd (even) elements of $\mathbf{v}$. If we form "even" and "odd" vectors

$$\mathbf{e} = [e_1, e_2, e_3, e_4]^T = [v_2, v_4, v_6, v_8]^T$$
$$\mathbf{o} = [o_1, o_2, o_3, o_4]^T = [v_1, v_3, v_5, v_7]^T,$$

then

$$d_k = e_k - \frac{1}{2}(o_{k+1} - o_k), \tag{35}$$

where $k = 1, 2, 3, 4$ and $o_5 := o_4$. The elements of the top half of the transformation can then be *lifted* from $\mathbf{d}$:

$$a_k = o_k + \frac{1}{4}(d_k + d_{k-1}), \tag{36}$$

where $k = 1, 2, 3, 4$ and $d_0 := d_1$.

This method is computationally more efficient than a fast matrix multiplication. Moreover, we can easily modify this transformation so that it maps integers to integers. Instead of computing the elements of $\mathbf{d}$ using (35), we compute $\mathbf{d}^*$, whose elements are

$$d_k^* = e_k - \left\lfloor \frac{1}{2}(o_k + o_{k+1}) \right\rfloor \tag{37}$$

and then compute $\mathbf{a}^*$, whose elements are

$$a_k^* = o_k + \left\lfloor \frac{1}{4}(d_k^* + d_{k-1}^*) + \frac{1}{2} \right\rfloor. \tag{38}$$

Adding $1/2$ to the term above eliminates bias. Certainly, the elements of $\mathbf{a}^*$ and $\mathbf{d}^*$ are integer-valued and only slightly different from those given by (35), (36), respectively. Moreover, the process is completely invertible. Given $\mathbf{d}^*$ and $\mathbf{a}^*$, we can certainly recover the values o_k using (38), and then given $\mathbf{o}$ and $\mathbf{d}^*$, we can use (37) to recover $\mathbf{e}$.

Concluding Remarks

In this article, we have provided a very brief introduction to discrete wavelet transformations and have shown how the topic effectively lends itself to an intriguing undergraduate applied mathematics course.

By introducing students to the discrete Haar wavelet transformation, we are able to not only construct a matrix representation of a tool used in several applications but also provide students with a concrete example of how convolution and Fourier series can lead to a more general mathematical model for the construction of more sophisticated scaling and wavelet filters.

The Daubechies filters are better suited for applications, but still suffer from the effects of truncating infinite convolution products. This problem leads to the development of biorthogonal scaling filters. Still more modification is needed to handle boundary effects, and students learn that these modifications are not only tractable but also lead to the lifting scheme. This computational method is not only faster than sparse matrix multiplication but can also be modified to produce the seemingly impossible invertible transformation that involves rounding! This final modification is an important tool used by the JPEG2000 image compression standard to perform lossless compression.

We have not discussed in this paper the classical approach to wavelet theory—the mathematics there is quite elegant (the books [2, 14, 9] are aimed at undergraduates). We believe, however, that an introduction to the discrete wavelet transformation, with immediate immersion in applications, is an effective way to introduce students to applications that are ubiquitous in today's digital age, to hone problem-solving skills, to promote and improve scientific computing, and to motivate concepts in upper-level undergraduate courses such as real and complex analysis.

References

[1] E. Aboufadel and S. Schlicker, *Discovering Wavelets*, Wiley–Interscience, Hoboken, NJ, 1999.

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

[3] A. Calderbank, I. Daubechies, W. Sweldens, and B.-L. Yeo, Wavelet transforms that map integers to integers, *Appl. Comp. Harm. Anal.* **5**, 3 (1998), 332–369.

[4] I. Daubechies, Orthogonal bases of compactly supported wavelets, *Comm. Pure Appl. Math.* **41** (1988), 909–996.

[5] ———, *Ten Lectures on Wavelets*, Society for Industrial and Applied Mathematicians, Philadelphia, PA, 1992.

[6] ———, Orthonormal bases of compactly supported wavelets II. Variations on a theme, *SIAM J. Math. Anal.* **24**, 2 (1993), 499–519.

[7] M. W. Frazier, *An Introduction to Wavelets Through Linear Algebra*, Undergraduate Texts in Mathematics, Springer, New York, NY, 1999.

[8] A. Jensen and A. la Cour-Harbo, *Ripples in Mathematics: The Discrete Wavelet Transform*, Springer, New York, NY, 2001.

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

[10] W. Sweldens, Wavelets and the lifting scheme: A 5 minute tour, *Z. Angew. Math. Mech.* **76** (Suppl. 2) (1996), 41–44.

[11] ———, The lifting scheme: A construction of second generation wavelets, *SIAM J. Math. Anal.* **29**, 2 (1997), 511–546.

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

[13] J. S. Walker, *A Primer on Wavelets and Their Scientific Applications*, 2nd ed., Chapman & Hall/CRC Press, Boca Raton, FL, 2008.

[14] D. F. Walnut, *An Introduction to Wavelet Analysis*, Birkhäuser, Cambridge, MA, 2002.

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