AUDIO COMPRESSION USING THE DISCRETE WAVELET TRANSFORM

NICHOLAS KLUGMAN*

Abstract. This paper introduces the reader to wavelet analysis, a means of localizing in both frequency and time. It then demonstrates a fast, general method to compute the discrete wavelet transform (DWT) of an input signal. Finally, the DWT is applied to audio compression and compared to the discrete cosine transform for the same purpose. The result is that DWT and DCT perform similarly in terms of compressed size for a given level of data loss, with the determining factor being which wavelet basis is chosen, as there are infinitely many.

Key words. wavelet, Daubechies, audio compression, multi-scale analysis

1. Introduction. Wavelet analysis, while a relatively young field, has grown in theory and application over the last forty years [3, 6, 7]. The ability to analyze both frequency and time simultaneously provides an advantage in some situations over strictly frequency-based methodologies, such as the Fourier transform. This paper endeavors to introduce the reader to wavelets, describe a general algorithm for computing the discrete wavelet transform, and apply this tool to lossy audio compression as an example of a field well-suited for wavelet analysis. In my proposal, I suggested that I would apply DWT to image compression (as in [4]) rather than audio compression, but upon reading that "our ear uses a wavelet transform when analyzing sound, at least in the very first stage," I found it more intriguing and appropriate to showcase wavelets in the context of audio[3].

The paper is organized as follows. A theoretical framework for understanding what wavelets are and why one might want to use them is given in Section 2. Section 3 describes the numerical process of calculating the discrete wavelet transform. We then apply the transform to audio compression in Section 4, and summarize in Section 5.

2. An Introduction to Wavelets. The simplest wavelet is the Haar wavelet, proposed by Alfred Haar in 1910 [3]. Based on the box function, the Haar wavelet is defined as

$$\psi(x) = \begin{cases} 1, x \in [0, \frac{1}{2}) \\ -1, x \in [\frac{1}{2}, 1) \\ 0, \text{ otherwise.} \end{cases} \tag{2.1}$$

Even so, wavelet analysis only took off in the 1980's, with Ingrid Daubechies and others making significant strides in the field. She identified a class of orthogonal wavelets with compact temporal support that exactly approximate polynomials of a given degree. [3] contains ten lectures Daubechies gave at a 1990 conference on wavelets. The first lecture gives some motivation for wavelet analysis, so we start there. Just the previous year, Gilbert Strang had published a review of wavelets, framing them as the result of analyzing dilation equations, and analyzing how their recursive nature lends itself to efficient algorithms for computing the Discrete Wavelet Transform (DWT) [7].

*Department of Mathematics, Massachusetts Institute of Technology, Cambridge, MA (nklugman@mit.edu).

2.1. Motivation for Wavelet Analysis. In the first lecture of [3], Daubechies explains the concept of time-frequency localization as the motivation for wavelets. The Fourier transform is the classic tool for localizing in frequency, but it doesn't localize in time. A standard modification to allow for localization in both frequency and time is the windowed Fourier transform,

$$(\mathcal{F}^{win} f)(\omega, t) = \int f(s) g(s-t) e^{-i\omega s} ds,$$

where g is a window function, often centered around 0 (so that the windowed transform will be centered around t), and decaying to 0 outside of some interval. The length of that interval determines the time-scale resolution. For that reason, we can only examine one time-scale resolution at each frequency, whereas with wavelets we can examine a different time-scale at each frequency.

The wavelet transform is given by

$$(T^{wav} f)(a, b) = \frac{1}{\sqrt{|a|}} \int f(t) \phi\left(\frac{t-b}{a}\right) dt, \tag{2.2}$$

where ϕ is called the "mother wavelet" function. This is sometimes discretized by $a = a_0^m$ and $b = n b_0 a_0^m$, yielding

$$T^{wav}_{m,n}(f) = a_0^{-\frac{m}{2}} \int f(t) \phi\left(a_0^{-m} t - n b_0\right) dt. \tag{2.3}$$

Here, larger values of $|a|$ correspond to lower frequencies and b is used to select location in time. The localization of wavelets in time and frequency renders some operators sparse (sometimes exactly and sometimes within some error bound) in the wavelet basis, allowing for faster computations and/or lower memory/storage overhead [6].

We can reconstruct the original function from its wavelet transform using the "resolution of identity formula,"

$$f = C_\psi^{-1} \int_{\mathbb{R}} \int_{\mathbb{R}} \frac{1}{a^2} \langle f, \psi^{a,b} \rangle \psi^{a,b} da \; db,$$

where $\psi^{a,b} = \frac{1}{\sqrt{a}} \psi\left(\frac{x-b}{a}\right)$ and

$$C_\psi = 2\pi \int_{\mathbb{R}} \frac{1}{|\xi|} |\hat{\psi}(\xi)|^2 d\xi,$$

where $\hat{\psi}$ represents the Fourier transform of ψ.

We are largely free in our choice of ψ at this point, requiring only that C_ϕ as defined above be finite (which itself requires $\hat{\psi}(0) = 0$). We utilize this freedom to choose a family of ψ's that provide an orthonormal basis for the functions we consider (specifically a basis for $L^2(\mathbb{R})$). Taking $a_0 = 2$ and $b_0 = 1$, we have

$$\psi_{m,n}(x) = 2^{-\frac{m}{2}} \psi\left(2^{-m} x - n\right).$$

One example ψ that provides an orthonormal basis is the Haar function, defined in Equation 2.1, though it provides low time-frequency localization.

In [7], Strang also discusses the relationship between wavelets and the Fourier transform. While the Fourier series is based on sines and cosines, which are perfectly

local in frequency and global in time, wavelets are meant to be local in frequency and time, without such heavy reliance upon cancellation to reconstruct time-frequency localized phenomena. Strang claims that the greatest advantage of $\{e^{inx}\}$ as a basis is that the differentiation operator is diagonal in this basis. Another way of saying this is that the basis functions are the eigenfunctions of the differentiation operator, or simply that differentiating one of these basis functions amounts to multiplying by a scalar. Wavelets do not share this property. Strang acknowledges our inability to diagonalize both differentiation with respect to time and with respect to frequency. They can, however, still form an orthogonal basis, though there's nothing inherent that requires orthogonality; we simply find it desirable.

2.2. Wavelets and Dilation Equations. In an effort to determine what functions ψ lend themselves to wavelet analysis, we now shift our focus slightly, to dilation equations (equations of the form $\psi(x) \to \psi(2x)$), which Strang uses to provide a foundation for understanding wavelets in [7].

Wavelets, transforming from a function of one variable to a function of two, provide a *multiresolution* representation of $f(x)$. That is to say, if $f(x) = \sum b_{j,k}\psi_{j,k}(x)$, $b_{j,k}$ carries information about the behavior of $f(x)$ around both $\xi = 2^j$ and $x = 2^{-j}k$.

Far less commonly known than difference equations (e.g. $f(x) = \sum c_k f(x-k)$) and differential equations (e.g $\frac{dx}{dt} = f(x,t)$) are dilation equations (e.g. $f(x) = \sum c_k f(ax-k)$, where a is the dilation parameter). Dilation equations are of particular interest in wavelets, because they provide a means of construction of wavelets. Assuming that we can normalize to $\int_{\mathbb{R}} \phi = 1$, then for the dilation equation

$$\phi(x) = \sum c_k \phi(2x-k), \tag{2.4}$$

we have the constraint $\sum c_k = 2$ by integrating both sides of the dilation equation.

One illustrative solution example is if $c_0 = 2$, then $\phi(x) = \delta(x)$. Another example is if $c_0 = c_1 = 1$, then

$$\phi_{box}(x) = \begin{cases} 1, 0 \le x < 1 \\ 0, \text{ otherwise.} \end{cases} \tag{2.5}$$

Also, if $c_0 = c_2 = \frac{1}{2}, c_1 = 1$, then

$$\phi_{hat}(x) = \begin{cases} x, 0 \le x < 1 \\ 2-x, 1 \le x < 2 \\ 0, \text{ otherwise.} \end{cases}$$

If we iterate

$$\phi_j(x) = \sum c_k \phi_{j-1}(2x-k), \tag{2.6}$$

with $\phi_0 = \phi_{box}$ as defined above (Equation 2.5), we can generate solutions to the dilation equation as $j \to \infty$. Taking $c_0 = 2$, ϕ_j approaches the delta function. Taking $c_0 = c_1 = 1$, ϕ_j remains the box function. Taking $c_0 = c_2 = \frac{1}{2}, c_1 = 1$ approaches the hat function.

Another means of constructing solutions is via the Fourier transform. Taking the Fourier transform of Equation 2.4 and applying the time scaling property of Fourier

transforms yields

$$\begin{aligned}\hat{\phi}(\xi) &= \sum c_k \int \phi(2x-k)e^{i\xi x}dx \\ &= \frac{1}{2}\left(\sum c_k e^{ik\xi/2}\right)\int \phi(y)e^{i\xi y/2}dy \\ &= P\left(\frac{\xi}{2}\right)\hat{\phi}\left(\frac{\xi}{2}\right),\end{aligned}$$

where

$$P(\xi) = \frac{1}{2}\sum c_k e^{ik\xi}. \tag{2.7}$$

Note, $P(0) = 1$. This yields a new dilation equation, but for the Fourier transform of ϕ. Reapplying this relation yields

$$\begin{aligned}\hat{\phi}(\xi) &= P\left(\frac{\xi}{2}\right)\hat{\phi}\left(\frac{\xi}{2}\right) \\ &= \prod_{j=1}^{N} P\left(\frac{\xi}{2^j}\right)\hat{\phi}\left(\frac{\xi}{2^N}\right) \\ &\to \prod_{j=1}^{\infty} P\left(\frac{\xi}{2^j}\right) \text{ as } n \to \infty.\end{aligned}$$

With an explicit means of calculating $\phi(x)$ in hand, we now analyze its properties. We begin by attempting to construct a $\phi(x)$ to approximate some polynomial $q(x)$ close to $x = 0$. Let the error in the approximation be $e(x) = \phi(x) - q(x)$. If, close to $x = 0$, $e(h) = O(h^p)$, the approximation is of order p, and ϕ approximates q exactly when q has degree $p-1$ or less. For this to be the case, $\hat{\phi}$ must have zeros of order p at all $\xi = 2\pi n$ (except for $\xi = 0$). By analyzing the infinite product, one can see that P has a zero of order p at $\xi = \pi$, i.e. $P^{(m)}(\pi) = 0$ for all $m \in \{0, 1, ..., p-1\}$. From Equation 2.7, we see that this is equivalent to

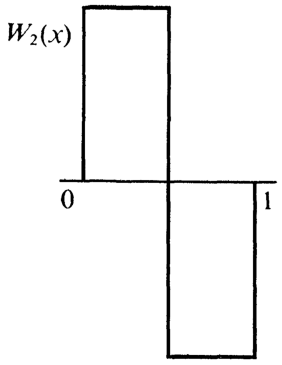
$$\sum (-1)^k k^m c_k = 0. \tag{2.8}$$

Daubechies uses this constraint, along with orthogonality, in producing her wavelet coefficients.

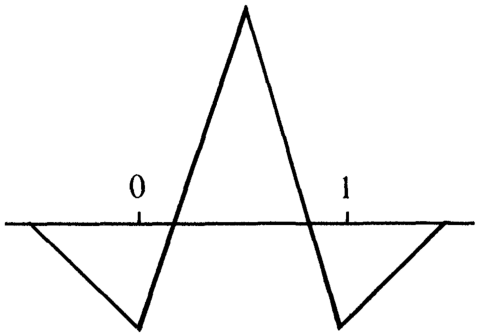
We now discuss the possibility of compact support of solutions to the dilation equation. Suppose $c_i = 0$ when $i \notin \{0, 1, ..., N\}$. Consider the first construction method, with iterative applications of the dilation relation. The support of $\phi_0 = \phi_{box}$ is $[0, 1]$. The support of ϕ_i is $[0, 2^{-i} + N\sum_{j=1}^{i} 2^{-j}]$. So, the support of ϕ is $[0, N]$, i.e. $\phi(x) = 0$ when $x \notin [0, N]$. For ϕ to be continuous, we have $\phi(0) = \phi(N) = 0$. Strang points out the importance that ϕ can have compact support, "homogeneous difference equations with zero boundary conditions lead to $\phi \equiv 0$, but not so for dilation equations" [7].

2.3. Orthogonal Wavelets. With this framework in place, we can give a definition of wavelets and analyze their role as an orthogonal basis [7]. Let

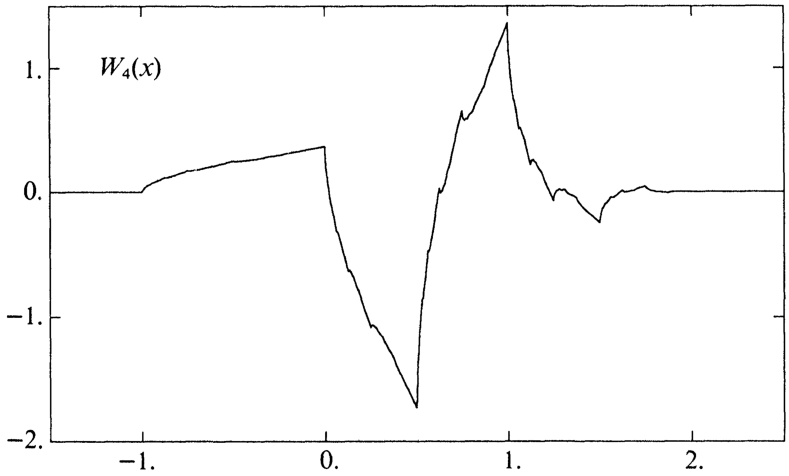
$$\psi(x) = \sum (-1)^k c_{1-k}\phi(2x-k). \tag{2.9}$$



(a) Haar wavelet from box function



(b) Wavelet from hat function



(c) Daubechies 4 ($DAUB4$) wavelet

Fig. 1: Examples wavelets. The Haar wavelet and Daubechies wavelet are both orthogonal to their translates, but the wavelet from the hat function is not. These graphs from [7]. Note that Strang uses W instead of the traditional ψ to denote a wavelet.

Some examples of wavelets generated with Equation 2.9 from the box function, the hat function, and utilizing Daubechies methodology are shown in Figure 1.

To inspect orthogonality of wavelets, let us suppose that the functions $\phi_0(2x-k)$ are orthogonal. Then, $\phi_1(x) = \sum c_k \phi_0(2x-k)$ shares the property, since

$$\int \phi_1(x)\phi_1(x-m)dx = \int \left(\sum_k c_k \phi_0(2x-k)\right)\left(\sum_l c_l \phi_0(2x-2m-l)\right) dx \tag{2.10}$$

$$= \sum_{k,l} c_k c_{k-2m} \int \phi_0^2(2x)dx. \tag{2.11}$$

Thus, if when $m \neq 0$

$$\sum_k c_k c_{k-2m} = 0,$$

orthogonality is preserved. Since the box function from which we constructed ϕ is orthogonal to its translates, we have that ϕ is as well.

The Daubechies wavelets (*DAUB(2p)*), are then constructed by requiring

$$\sum_k c_k c_{k-2m} = 2\delta_{0m} \ (m \in \mathbb{Z}) \tag{2.12}$$

$$\sum_k (-1)^k k^m c_k = 0 \ (m \in \{0, ..., p-1\}) \tag{2.13}$$

$$c_m = 0 \ (m \notin \{0, ..., 2p-1\}). \tag{2.14}$$

Equation 2.14 forces compact support of the wavelets. Equation 2.12 provides $2p-1$ nonzero equations, but $p-1$ are repeats(m and $-m$ provide the same information), so these only constrain p degrees of freedom. Equation 2.13 sets the approximation order to p and provides the last p equations for the $2p$ unknown c_m values.

3. The Discrete Wavelet Transform. Before developing a general framework for computing wavelet transforms, we examine the simple, but illustrative, case of *DAUB4*, as examined in [6].

3.1. *DAUB4*. Due to the recursive nature of wavelets, similar to FFT, the discrete wavelet transform (DWT) most naturally operates on vectors with a length that is a power of two, so we simplify our analysis by only considering vectors for which this holds. As previously established, transform is a linear operator that is invertible and, in particular, orthogonal. While FFT transforms from a basis of delta functions (the continuum limit of our familiar e_i basis) to a basis of sines and cosines, DWT transforms to a basis of wavelets (and a mother function).

Daubachies's simplest (by virtue of its being the most localized in time, though not smoothest) wavelet filter coefficients is dubbed *DAUB4*. Let $c_0 = \frac{1+\sqrt{3}}{4\sqrt{2}}$, $c_1 = \frac{3+\sqrt{3}}{4\sqrt{2}}$, $c_2 = \frac{3-\sqrt{3}}{4\sqrt{2}}$, and $c_3 = \frac{1-\sqrt{3}}{4\sqrt{2}}$. (For a description of how these coefficients came about apply $p = 2$ to the framework developed in Section 2 and apply the $\frac{1}{\sqrt{2}}$ normalization factor). Then,

$$D_4 = \begin{bmatrix} c_0 & c_1 & c_2 & c_3 & & & & \\ c_3 & -c_2 & c_1 & -c_0 & & & & \\ & & c_0 & c_1 & c_2 & c_3 & & \\ & & c_3 & -c_2 & c_1 & -c_0 & & \\ & & & \ddots & \ddots & \ddots & \ddots & \\ & & & & c_0 & c_1 & c_2 & c_3 \\ & & & & c_3 & -c_2 & c_1 & -c_0 \\ c_2 & c_3 & & & & & c_0 & c_1 \\ c_1 & -c_0 & & & & & c_3 & -c_2 \end{bmatrix} \tag{3.1}$$

is the *DAUB4* operator. Note that this matrix is orthogonal (because of the normalization factor), so its inverse is its transpose ($D_4^{-1} = D_4^T$).

To understand the behavior of this operator, we consider how each row operates on the input column vector. All of the odd rows consist of zeros and the row vector

$$\begin{bmatrix} c_0 & c_1 & c_2 & c_3 \end{bmatrix}$$

except for the second to last row, which consists of a wrapped version of this row vector. This row vector acts as a smoothing (or low-pass) filter for the input; in particular row $2i-1$ smooths entries $2i-1$ to $2i+2$ of the input (wrapping the index around at the end).

All of the even rows consist of zeros and the row vector

$$\begin{bmatrix} c_3 & -c_2 & c_1 & -c_0 \end{bmatrix}$$

except for the last row, which consists of a wrapped version of this row vector. This row vector acts as a detail (or high-pass) filter for the input; in particular row $2i-1$ gives the detail in entries $2i-1$ to $2i+2$ of the input (wrapping the index around at the end).

At this point, we can finally define the discrete wavelet transform (DWT). Suppose D is the filter coefficient matrix (D_4 in particular is shown above) and f is the input, with n components (for simplicity let n be a power of two). Applying D to f gives a vector that alternates between smoothed and detailed components. We can then permute Df to collect all the smoothed components in the first $\frac{n}{2}$ components and the detailed components in the last $\frac{n}{2}$ components. Applying D (note that this would actually be a $\frac{n}{2} \times \frac{n}{2}$ matrix, whereas we previously used the $n \times n$ matrix with the structure of D) to the first $\frac{n}{2}$ components of the permuted Df and so on recursively will eventually give two smoothed components (the first two components of the result), then a pair of detailed components, then a quartet of detailed components, then an octet, and so on. Actually, in the case of *DAUB4* we stop when we apply the 4×4 D_4 matrix and permute because we cannot shrink the D_4 matrix below 4×4. In the case of *DAUB(2p)*, we cannot shrink D_{2p} below $2p \times 2p$, so we stop with p smoothed components. We also would never explicitly form the D matrix as it turns an $O(pn)$ operation into a $O(n^2)$ matrix-vector multiply.

3.2. General DWT. We now discuss a more general algorithm to compute the discrete wavelet transform [1, 3, 5, 7]. Consider the vector f with length $n = 2^J$ and components sampling the input function, $f(x)$, on the interval $(0, 1]$ such that $f_i = f\left(\frac{i}{n}\right)$. Let L be a low-pass filter producing a vector with half as many entries such that

$$(Lf)_i = \frac{1}{2} \sum_j c_{2i-j} f_j, \ i \in \left\{1, 2, ..., \frac{n}{2}\right\}. \tag{3.2}$$

Note, $Lf = \frac{1}{2}(c * f) \downarrow 2$, where $*$ is the convolution operator and $\downarrow 2$ means down-sampling by a factor of two. Since $f \in \mathbb{R}^n$, j runs from 1 to n. Then, in the case of *DAUB(2p)*, we can shorten this, since $c_{2i-j} = 0$ when $(2i-j) \notin \{0, ..., 2p-1\}$. So, in that case, the index j runs from $2i-2p+1$ to $2i$ (with negative indices wrapping around f).

The decimating filter L gives also rise to a dual interpolating filter L^*, which will be used in reconstruction. We define the operation of L^* by

$$(L^* g)_j = \sum_i c_{2i-j} g_i, \ j \in \{1, 2, ..., n\}. \tag{3.3}$$

Note, $L^* g = c * (g \uparrow 2)$, where the upsample operator ($\uparrow$) places zeros in the odd components. Since $g \in \mathbb{R}^{\frac{n}{2}}$, the index of summation i runs from 1 to $\frac{n}{2}$. Once again, we can simplify the case of *DAUB(2p)*, so that i runs from $\left\lceil \frac{j}{2} \right\rceil$ to $p - 1 + \left\lceil \frac{j}{2} \right\rceil$.

Also, note that L has rank at most $\frac{n}{2}$ and L^*L is a projection operator that corresponds to blurring of or detail removal from f. (This removal comes from the data lost by downsampling and then upsampling.) In the example of the Haar wavelet (the wavelet induced by the box function, which also happens to be $DAUB2$), L^*L maps to the space of piecewise constant function samples. We can then capture only the detail by subtracting the blurred image from the original, i.e. by computing

$$f^{(J-1)} = f - L^*Lf.$$

We then repeat the process to recursively build up a final decomposition. Let $a^{(j-1)} = La^{(j)}$, $a^{(J)} = f$, and $f^{(j-1)} = (I - L^*L)\, a^{(j)} = a^{(j)} - L^*a^{(j-1)}$.

To recompute f from the details $a^{(0)}, f(j),\ j \in \{0, 1, .., J-1\}$, we reverse the process using the formula

$$a^{(j)} = f^{(j-1)} + L^*a^{(j-1)}.$$

Some (e.g. [3, 6]) also replace the $\frac{1}{2}$ in Equation 3.2 with a $\frac{1}{\sqrt{2}}$, adding this normalization factor also to equation 3.3 for symmetry (this also happens in FFT at times). With this change for symmetry, the basis becomes orthonormal.

Algorithm 3.1 Mallat's Pyramid Algorithm - Decomposition

Input: f, L, H
$a^J = f$
for j = J down to 1 **do**
  $a^{j-1} = La^j$
  $b^{j-1} = Ha^j$
end for
return $\left(a^0, b^0, b^1, ..., b^{J-1}\right)$

Algorithm 3.2 Mallat's Pyramid Algorithm - Reconstruction

Input: $a^0, b^0, b^1, ..., b^{J-1}$, L^*, H^*
for j = 1 to J **do**
  $a^j = L^*a^{j-1} + H^*b^{j-1}$
end for
return a^J

These algorithms for decomposition and construction were proposed by Stephane Mallat and are given in Algorithms 3.1 and 3.2 [5]. In these algorithms, $L_{ij} = c_{2i-j}$ as in Equation 3.2, and $H_{ij} = (-1)^{j+1}c_{j-2i+1}$, coming from Equation 2.9. The dimensions of L and H scale according to the vector they are operating on, but always have half as many rows as columns. The definition of the operation of H (analogous to Equation 3.2) is

$$(Hf)_i = \frac{1}{2}\sum_j (-1)^{j+1}c_{j-2i+1}f_j,\ i \in \left\{1, 2, ..., \frac{n}{2}\right\}, \tag{3.4}$$

and the corresponding interpolation operation is

$$(H^*g)_j = \sum_i (-1)^{j+1}c_{j-2i+1}g_i,\ j \in \{1, 2, ..., n\}. \tag{3.5}$$

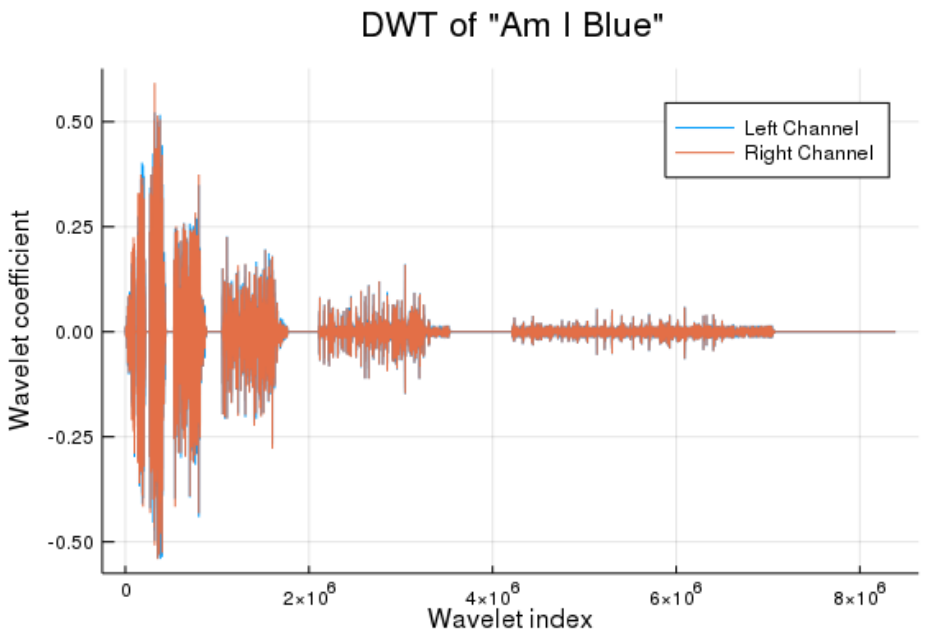


Fig. 2: Discrete wavelet transform ($DAUB4$) of the song "Am I Blue" [2].

Once again, in the case of $DAUB(2p)$ the summation need not span the entire range of i or j. When we operate with H, we only need to sum over j ranging from $2i-1$ to $2(p+i-1)$. When we operate with H^*, we only need to sum over i ranging from $\lceil \frac{j}{2} \rceil - p + 1$ to $\lceil \frac{j}{2} \rceil$.

These definitions lead to the conclusion that L and H are orthogonal, i.e.

$$HL^* = 0.$$

The orthogonality condition in Equation 2.12 becomes critical at this point, as it implies

- $LL^* = I$ and $HH^* = I$, and
- L^*L and H^*H are mutually orthogonal projection operators with

$$L^*L + H^*H = I.$$

This last property is what allows us to reconstruct a^j from $a^{j-1} = La^j$ and $b^{j-1} = Ha^j$; even though both L and H have a null space, their mutual orthogonality demonstrates that they leave enough information that together the original data can be recovered.

4. Audio Compression – Truncated Wavelet Approximations. Now that we understand the fundamentals of wavelets and can compute the wavelet transform of a signal, let us put them to use. The Fourier transform is often helpful since the complicated (one might even say convoluted) operation of convolution becomes simple multiplication in the Fourier domain, so with the aid of the FFT one can quickly compute convolutions. There are no analogous mathematical operations simplified by

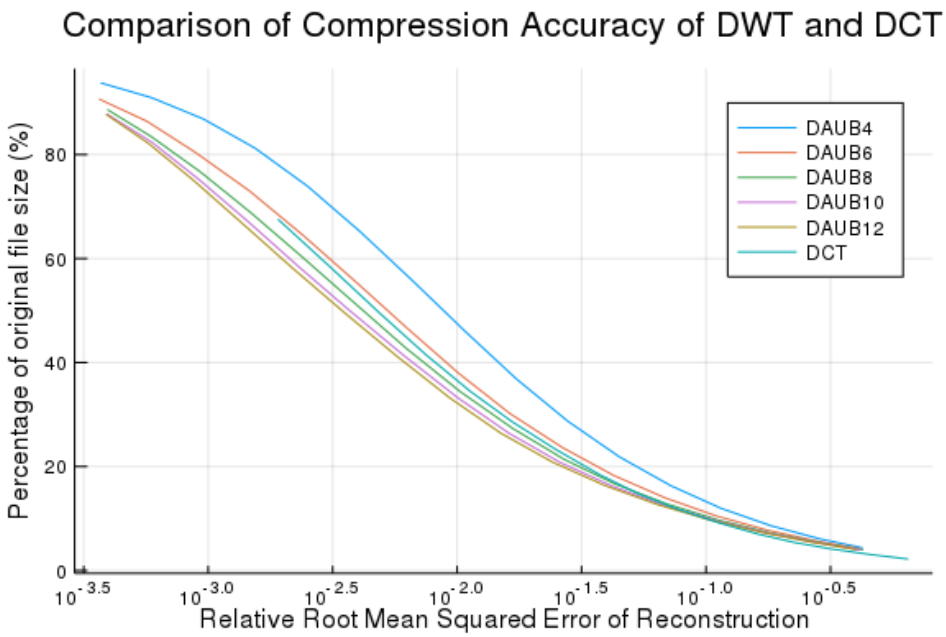


Fig. 3: Comparison of accuracy of reconstruction after compression using discrete cosine transform and discrete wavelet transform with Daubechies wavelets of varying smoothness.

transformation into the wavelet domain [6]. Therefore, most of the utility of wavelets come from their ability to render real-world signals sparse. Take, for example, the DWT of the audio for the song "Am I Blue," shown in Figure 2 [2]. It's clear that most of the signal comes from relatively few wavelet components.

This observation gives rise to a simple means of compressing audio, by transforming into the wavelet basis, throwing out any coefficients smaller than some threshold [6]. We can apply the same methodology as well to compressing after transforming into the Fourier domain using the discrete cosine transform (DCT). In figure 3, we plot relative compressed size against relative root mean squared error of reconstruction for both DWT (using *DAUB4* through *DAUB12*) and DCT.

In particular, we save the indices of the coefficients we keep and the compressed size is the size of that array and the array of coefficients. The relative size is the compressed size divided by the size of the original audio data array. The relative root mean squared error is the L^2 norm of the difference between the reconstruction and the original divided by the L^2 norm of the original.

As shown in Figure 3, with this naive compression algorithm, the discrete cosine transform outperforms *DAUB4* and *DAUB6*, in that for any given error in reconstruction, the wavelet transforms compress the audio almost as much, but not quite as much, as the discrete cosine transform. However, once we get up to *DAUB8*, *DAUB10*, and *DAUB12*, the wavelet transforms begin to outperform the cosine transform. We have thus demonstrated that, at least with this simplistic compression scheme and picking some of the simplest wavelet bases, the discrete wavelet transform performs just as well as, even better than the discrete cosine transform. One can easily imag-

ine more exotic wavelets that are even better suited for the task at hand, perhaps significantly outperforming the discrete cosine transform. This analysis also stresses the importance of picking an appropriate wavelet basis for the problem at hand.

5. Conclusions. Wavelet analysis is a quite young field, barely older than a century and not reaching any kind of maturity until the last forty years. Even so, we have seen in this paper the potential advantages of wavelet analysis, allowing us to simultaneously examine both the time and frequency domains. We have developed a theoretical framework for understanding wavelets in terms of the dilation equations that can be used to construct them. We have examined algorithms for both decomposition and reconstruction, both in the simple case of $DAUB4$ and the general case. Finally, we have applied these methodologies to audio compression and seen that with the right basis, DWT can outperform DCT in this application. These methodologies can be further applied to image compression [4]. Wavelets are also useful in some numerical applications, as they render some operators sparse [6]. An example would be the integration operator. Being localized in time (or space), and acting as a low-pass filter, the integration operator tends to be sparse in the wavelet domain. This property allows for fast solutions to integral equations. Despite its young age, the field of wavelet analysis is rich with both theory and applications.

REFERENCES

[1] G. Beylkin, R. Coifman, and V. Rokhlin, *Fast wavelet transforms and numerical algorithms i*, Communications on Pure and Applied Mathematics, 44 (1991), pp. 141 – 183, https://doi.org/10.1002/cpa.3160440202.

[2] K. Conroy, *Am I blue?*, 2004, realkevinconroy.com.

[3] I. Daubechies, *Ten Lectures on Wavelets*, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1992.

[4] S. Mallat and S. Zhong, *Characterization of signals from multiscale edges*, IEEE Trans. Pattern Anal. Mach. Intell., 14 (1992), pp. 710–732, https://doi.org/10.1109/34.142909, https://doi.org/10.1109/34.142909.

[5] S. G. Mallat, *A theory for multiresolution signal decomposition: the wavelet representation*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 11 (1989), pp. 674–693, https://doi.org/10.1109/34.192463.

[6] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in C (2Nd Ed.): The Art of Scientific Computing*, Cambridge University Press, New York, NY, USA, 1992.

[7] G. Strang, *Wavelets and dilation equations: A brief introduction*, SIAM Review, 31 (1989), pp. 614–627, http://www.jstor.org/stable/2031540.