Wavelets

1 Introduction

Signal representation and analysis is a fundamental component of modern technology, with a range of applications in fields such as data analysis, audio/video processing, communications, and data compression. Traditionally, the Fourier Transform has offered a means by which such data may be represented and analysed by decomposing them into global oscillations with fixed amplitude and frequency. However, even with modifications that provide temporal information, this global representation is inefficient at capturing local information and as such, Fourier-based transforms struggle to interpret sudden changes and temporal discontinuities in data, which are not only common in real-world applications, but also tend to be the most significant parts of the data, such as edges in images or syllables in speech. Wavelets are a class of functions that offer a solution to the issues of Fourier analysis by utilising a vector basis under which signals may be represented that provides a dynamic balance between localised information in time and frequency. This essay introduces the concept of wavelets and the motivations for their use before examining their mathematical construction through dilation equations and function vector spaces, as well as exploring how they compare to alternative representations and analysis methods, and presenting their applications in data analysis, encoding, and signal processing.

2 Motivation and the Wavelet Concept

2.1 Uncertainty and Resolution

With the analysis of any signal, we are interested in the concept of *resolution*: the accuracy to which we can differentiate between similar frequency components and the time windows in which they occur. There is an intuitive trade-off between accuracy in frequency and time: the longer we observe a frequency, the better we can measure it; and if we observe a signal at an instant in time, we cannot say anything about its frequency. This trade-off is quantified in a variation of the Gabor Uncertainty Principle:

As an informal example[1], consider trying to measure the frequency of a signal over some duration Δt. We will assume we measure the frequency by counting the zero crossings of the signal and consider the simple case where we seek to differentiate two sine waves with frequencies f_0 and $f_0 + \Delta f$ such that the signals are of the form $\sin(2\pi f_0 t)$ and $\sin[2\pi(f_0 + \Delta f)t]$. In general, each period is composed of two zero crossings and a period occurs over a time difference of $T = 1/f$ so we may count, in Δt time, $2f\Delta t$ crossings. In order to differentiate the frequencies, we require that the difference in zero crossings is at least one. That is, assuming $\Delta f > 0$:

$$\begin{aligned} 2(f_0 + \Delta f)\Delta t - 2f_0\Delta t &\geq 1, \\ \text{so} \qquad \Delta f \Delta t &\geq \frac{1}{2}. \end{aligned}$$

[1] Derivation adapted from the ideas of [1]. Depending on the interpretation of Δf and Δt, the inequality may take different constants. However, the key is that there is an inherent trade-off between frequency and time resolution.

It follows that the limiting accuracies with which we can differentiate frequencies and times are inversely proportional, so in order to achieve a certain resolution in measuring frequency, we must sacrifice some resolution in measuring time. The problem of optimising this time-frequency trade-off is a key motivator for the development of wavelets as an alternative to Fourier Transform-based methods.

In the context of the Fourier Transform, the time and frequency domains are opposite ends of this spectrum, offering either arbitrarily perfect resolution in time or in frequency. The resolution in the time and frequency domains may be represented in a time-frequency plot as a series of vertical and horizontal 'boxes' (bounded by the lines) which can be differentiated by the analysis. As a metaphor, we can think of this like the pixels on a screen: the more pixels we have, the more accurately we can produce an image. In this sense, the vertical pixels on our screen represent possible changes in frequency, and similar for horizontal pixels and time. Unfortunately, the area of each pixel we have is bounded below by the uncertainty principle described earlier, so we need to decide how to shape them to produce the best image. Then, the solution these two domains provide is to offer perfect resolution but only in one dimension each:

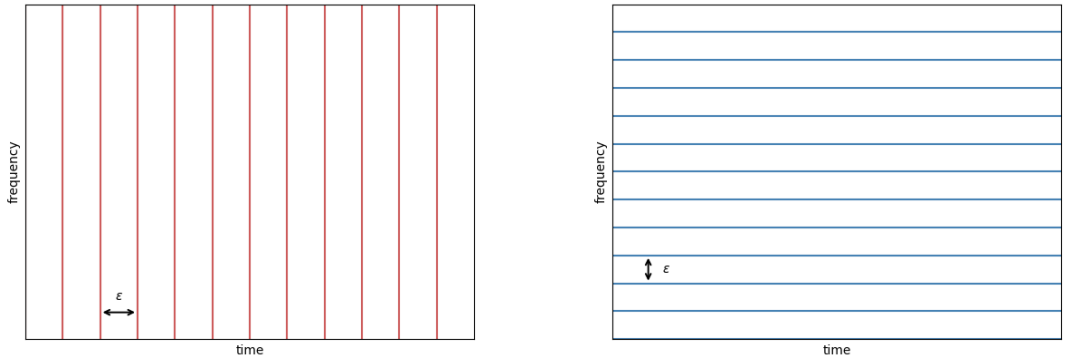

Figure 1: Visual representation of the resolutions of time (left) and frequency (right) domains[2].

In practice, however, analysis and representation often require simultaneous insight into both time *and* frequency (returning to the metaphor, any reasonable screen should divide its pixels between both axes). This is the solution provided by the Short-Time Fourier Trasnform, which takes a signal divided into multiple 'windows' of time and performs frequency analysis on each window. The STFT may be used to achieve a predetermined time resolution through repeated Fourier Transforms over each time window, leading to a grid of boxes which may be distinguished by STFT analysis. Recall that the time-frequency uncertainty requires that for a finer Δt, we lose precision on f, so the selection of time window size leads to a trade-off in resolutions (if we want to make our pixels thinner in one dimension, they must become wider in the other), which can be visualised in the figure below, where the minimum area of each box is bounded.

[2]Note that the lines may be arbitrarily close. Figures 1 through 3 were produced using Matplotlib.

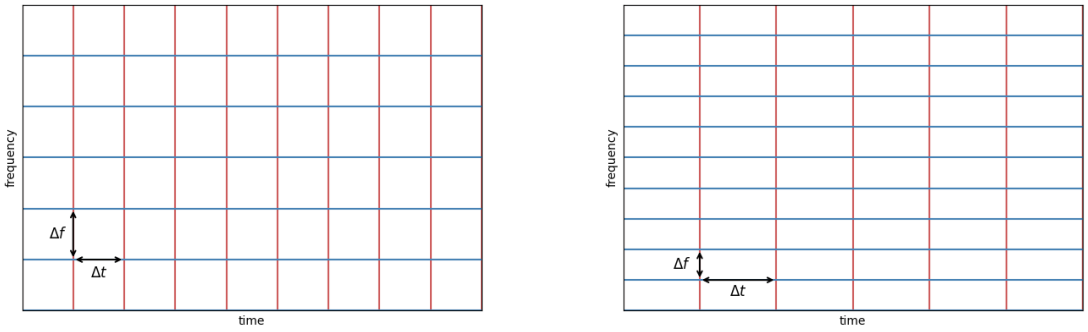

Figure 2: Visual representation of the resolution of STFT with varying time window sizes.

The STFT method has some key drawbacks, namely that it requires us to determine the particular degree to which we sacrifice time and frequency resolution in advance, and applies this sacrifice to all frequencies equally, which in some signals may fail to capture sufficient time or frequency resolution. The STFT output may also vary depending on the particular time-frequency division selected and can experience information loss at its boundaries, leading to inconsistent analysis [2]. (As a side note, the last issue can be rectified using overlapping time windows as well as 'windowing' functions that smooth transitions between overlaps, similar to the MP3 algorithm).

2.2 Establishing the Concept of Wavelets

In practical data, low frequency signals tend to represent long term trends, while high frequency signals tend to represent quick changes. Take a series of temperature data for example: low frequency seasonal changes occur over long periods of time, while a weather event causing a sudden temperature change is best captured in high frequency data.

This then motivates a new construction in which the resolution at which we analyse a signal may be determined dynamically, applying a time resolution proportional to the frequency of each component of a signal. An intuitive way to build this up is in a binary tree-like structure: For the lowest frequencies we will assign a single slice. Then, a slightly higher frequency should demand more time accuracy, so we divide the single slice into two (again, recall this comes at the cost of frequency precision). Then, for an even higher frequency, we might divide the two slices into four, and so on. In terms of our time-frequency graph, this may be visualised as such:

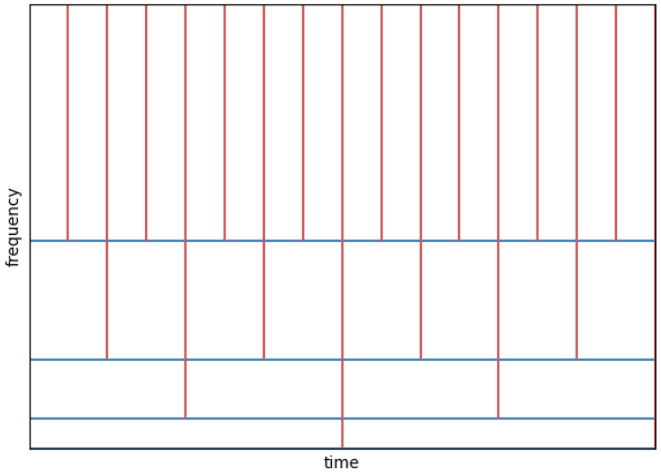

Figure 3: Visual representation of the resolution of a wavelet-based decomposition.

This division provides an elegant solution to the problem established in section 2.1: instead of assigning precisions uniformly over all components as in the case of STFT (or not having any information about one dimension in the case of FT), we can achieve better time accuracy where it matters (with high frequency changes) and better frequency accuracy when the time accuracy is less important (with low frequency changes). The problem, then, is how to efficiently process signals into this form. Here, we provide the high-level idea behind such a construction, which will then be mathematically refined in the following sections.

The idea is to, paralleling the sine and cosine Fourier basis, construct an set of basis vectors (functions) in which a signal may be represented. Where the Fourier basis vectors oscillate for infinite time, our wavelet basis vectors should be localised to a particular time and frequency range - if we can construct this basis such that these ranges correspond with the resolution 'boxes' in the graph above, then we can get one step closer to representing a signal as wavelets. Furthermore, if the resulting basis is orthonormal, then we may represent a given signal function in terms of this basis using vector projection.

3 Wavelet Construction

Note: this section uses examples and ideas combined and adapted from [3], [5], [11], and [12]. In particular, all practice exercises are sourced from section 11.12 of BHK.

3.1 Mathematical Foundations

We begin with some mathematical background information on concepts that are central to the construction of wavelet systems and analysing signals using wavelets: dilation equations and function vector spaces.

3.1.1 Dilation Equations

Translation and dilation are familiar function transformations, respectively shifting or scaling all points on the function relative to an origin. A *dilation equation* defines a function as a linear combination of its images under these transformations:

$$f(x) = \sum_k c_k f(\alpha_k x - k).$$

For our purposes, we are particularly interested in dilation equations where a function is related to its dilations by powers of 2 (in order to satisfy the binary tree-like structure). Additionally, we will assume that we are only interested in the behaviour of our function for non-negative x, so the dilation equation may take the form:

$$f(x) = \sum_{k=0}^{d-1} c_k f(2x - k).$$

Such dilations are called *factor of two reductions* or *dyadic dilates*. Generally, many such dilation equations have no closed form solutions and their solutions must be approximated for use. Solutions often take the form of fractals or splines [5][6]. We will explore several methods by which solutions or approximations may be found.

Approach using Successive Approximation

This computational approach iteratively improves an approximation for the solution to a dilation equation. The general idea is to make a guess for $f(x)$ and then evaluate each $f(2x-k)$ using averages from this guess. Then an improved guess for $f(x)$ may be obtained directly from the function definition.

Algorithm 1: IterativeDilation(c_k, d, l, h):
Calculate an approximate solution to $f(x) = \sum_{k=0}^{d-1} c_k f(2x-k)$ over $[l, h)$.

$\mathbf{f} \leftarrow$ vector of $f(x)$ values corresponding to evenly spread x values, containing an initial guess
while not converged **do**
  for $i \in \{0, 2, \ldots\}$ **do**
    $\mathbf{g}_0[i/2] \leftarrow \dfrac{\mathbf{f}[i] + \mathbf{f}[i+1]}{2}$ `// vector` $\mathbf{g}_i$ `represents values for` $f(2x-i)$`, regions without values are filled with zeroes`
  for $k \in \{1, \ldots, d-1\}$ **do**
    $\mathbf{g}_k[i + shift] = \mathbf{g}_0[i]$ `// shift calculated based on precision and k - see implementation in appendix for more details`
  $\mathbf{f} \leftarrow \sum_{k=0}^{d-1} c_k \mathbf{g}_k$
return f

The resulting vector then contains an approximation for $f(x)$. As an example, we apply this algorithm to a problem from the exercises of [3]. The algorithm was implemented in Python using Matplotlib to graph the output. The full source code is available in the appendix, only output will be shown here.

(BHK Exercise 11.9) Compute an approximation to the scaling function that comes from the dilation

equation $\phi(x) = \frac{1+\sqrt{3}}{4}\phi(2x) + \frac{3+\sqrt{3}}{4}\phi(2x-1) + \frac{3-\sqrt{3}}{4}\phi(2x-2) + \frac{1-\sqrt{3}}{4}\phi(2x-3)$.

We use IterativeDilation to compute the result. The program was run with a low bound of 0 and a high bound of 8 with 800 slices for 10 iterations and with vector $\mathbf{c} \approx (0.683, 1.183, 0.317, -0.183)^T$ as input[3]. The resulting plot is shown:

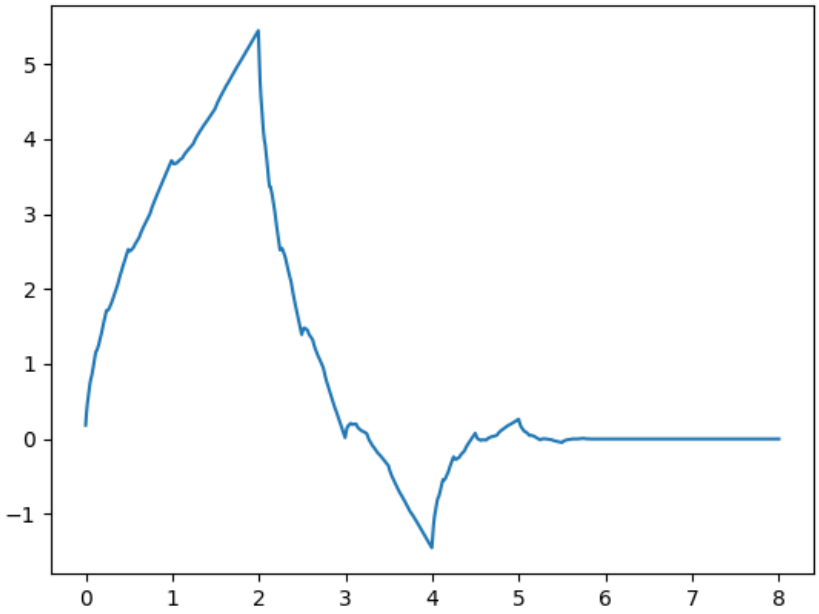

Figure 4: Algorithm 1 output for D_4 scale function dilation equation.

As a side note, this equation is the scaling function for the Daubechies wavelet, a well-known wavelet which has been used in a range of applications (See section 3.3). In reality, the Daubechies scale function is fractal in nature and consists of sharp, undifferentiable edges which are not well captured by the approximation [5].

Approach using Approximations with Matrix Eigenvectors

Consider the dilation equation $f(x) = \sum_{k=0}^{d-1} c_k f(2x-k)$. In the interest of simplicity, define $n = d-1$. We will assume our solution is zero outside of $[0, n]$ and consider the following:

$$\begin{aligned}
f(0) &= c_0 f(0) + c_1 f(-1) + c_2 f(-2) + c_3 f(-3) \ldots &&= c_0 f(0) \\
f(1) &= c_0 f(2) + c_1 f(1) + c_2 f(0) + c_3 f(-1) + \ldots &&= c_2 f(0) + c_1 f(1) + c_0 f(2) \\
f(2) &= c_0 f(4) + c_1 f(3) + c_2 f(2) + c_3 f(1) + \ldots &&= c_4 f(0) + c_3 f(1) + \ldots + c_0 f(4) \\
\vdots &= \vdots &&= \vdots \\
f(n-1) &= c_0 f(2n-2) + \ldots + c_n f(2n-2-n) &&= c_n f(n-2) + \ldots + c_{n-2} f(n) \\
f(n) &= c_0 f(2n) + c_1 f(2n-1) + \ldots + c_n f(n) &&= c_n f(n)
\end{aligned}$$

[3]BHK actually gives $\frac{1\sqrt{3}}{4}$ for the last coefficient, which I assume is a mistake as they refer to the resulting function as D_4, the Daubechies 4-tap scaling function which takes $\frac{1-\sqrt{3}}{4}$.

In general, we can say that

$$f(k) = \sum_{i=0}^{n} C_{k,i} f(i), \text{ where } C_{k,i} = \begin{cases} c_{2k-i} & \text{if } \max(0, 2k-n) \leq i \leq \min(n, 2k) \\ 0 & \text{otherwise.} \end{cases}$$

Let $\mathbf{M}$ be the matrix such that the element in its i^{th} row and j^{th} column is $C_{i,j}$. Then we may observe that if $\mathbf{v} = (f(0), f(1), \ldots, f(n))^T$ then

$$\mathbf{v} = \mathbf{M}\mathbf{v}.$$

That is, $\mathbf{v}$ is an eigenvector of $\mathbf{M}$ corresponding to eigenvalue 1. Supposing that we have such a vector $\mathbf{v}$, we obtain values for f for all integer inputs. Consider then

$$f\Big(\frac{x}{2}\Big) = \sum_{k=0}^{n} c_k f(x-k),$$

for which the inputs to f inside the sum are all integers. Similarly,

$$f\Big(\frac{x}{2^r}\Big) = \sum_{k=0}^{n} c_k f\Big(\frac{x}{2^{r-1}} - k\Big).$$

That is, the values of f at precisions of 2^{-r} depend entirely on the values at precisions of $2^{-(r-1)}$, so we may calculate each level of precision iteratively until we reach a satisfactory precision. Pseudocode for such an algorithm is provided below:

Algorithm 2: MatrixDilation(c_k, n, P):
Calculate an approximate solution to $f(x) = \sum_{k=0}^{n} f(2x-k)$ to precision 2^{-P}

$\mathbf{M} \leftarrow$ coefficient matrix
$\mathbf{v} \leftarrow$ eigenvector of $\mathbf{M}$ corresponding to eigenvalue 1
$(f(0), f(1), \ldots, f(n)) \leftarrow \mathbf{v}$
$p \leftarrow 1$
while $p \leq P$ **do**
  for $i \in \{1, 3, \ldots, n2^p - 1\}$ **do**
    $f\Big(\frac{i}{2^p}\Big) \leftarrow \sum_{k=0}^{n} c_k f\Big(\frac{i}{2^{p-1}} - k\Big)$
  $p \leftarrow p + 1$
return $f(x)$

As an example, we may apply this algorithm to determine solutions to some dilation equations from BHK. I implemented the algorithm in Python, using NumPy's linear algebra library to find eigenvectors and Matplotlib to graph the output. The full source code is available in the appendix, only output will be shown here.

(BHK Exercise 11.4) Find a solution to the dilation equation $f(x) = f(2x) + f(2x-1) + f(2x-2) + f(2x-3)$.

We provide the input $\mathbf{c} = (1, 1, 1, 1)^T$, $n = 3$ and solve using MatrixDilation to a precision of 2^{-8}. The coefficient matrix, selected eigenvector, and resulting plot are shown.

$$\mathbf{M} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\mathbf{v} \approx (0, -0.707, 0, 0.707)^T$$

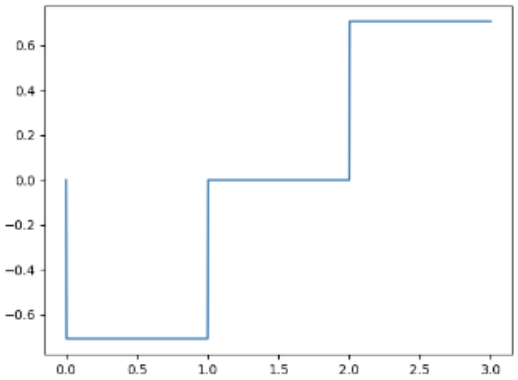

Figure 5: Algorithm 2 output for $f(x) = f(2x) + f(2x-1) + f(2x-2) + f(2x-3)$.

Inspecting the resulting graph and noting from the equation that if $f(x)$ is a solution then so is $cf(x)$ for $c \in \mathbb{R}$, we obtain a solution:

$$f(x) = \begin{cases} -c & \text{if } 0 \leq x \leq 1 \\ c & \text{if } 2 \leq x \leq 3 \\ 0 & \text{otherwise.} \end{cases}$$

(BHK Exercise 11.5) Find a solution to the dilation equation $f(x) = f(2x) + 2f(2x-1) + 2f(2x-2) + 2f(2x-3) + f(2x-4)$. What is the value of $\int_{-\infty}^{\infty} f(x)dx$?

Similarly, we analyse the graph produced by $\mathbf{c} = (1, 2, 2, 2, 1)^T$ with $n = 4$ to a precision of 2^{-8}. The output is:

$$\mathbf{M} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 2 & 2 & 1 & 0 & 0 \\ 1 & 2 & 2 & 2 & 1 \\ 0 & 0 & 1 & 2 & 2 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\mathbf{v} \approx (0, 0.5, -0.5, -0.5, 0.5)^T$$

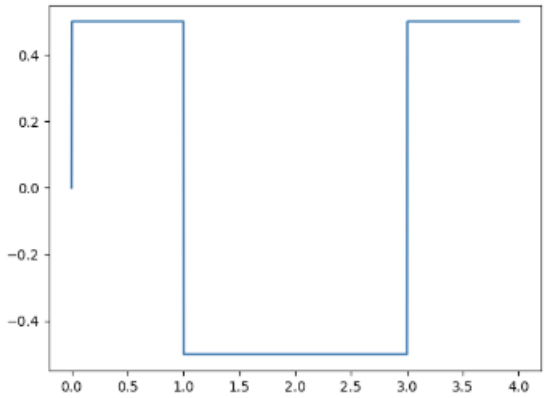

Figure 6: Algorithm 2 output for $f(x) = f(2x) + 2f(2x-1) + 2f(2x-2) + 2f(2x-3) + f(2x-4)$.

Inspecting the resulting graph we obtain a solution for $c \in \mathbb{R}$:

$$f(x) = \begin{cases} c & \text{if } 0 \leq x \leq 1 \text{ or } 3 \leq x \leq 4 \\ -c & \text{if } 1 \leq x \leq 3 \\ 0 & \text{otherwise.} \end{cases}$$

In order to determine the value of the integral independent of the solution we found, we observe the following, letting $c_k = \mathbf{c} \cdot \mathbf{e}_{k+1}$ where $\mathbf{e}_j$ is the j^{th} standard basis vector, and assuming the integral is

convergent:

$$\begin{aligned}\int_{-\infty}^{\infty} f(x)dx &= \int_{-\infty}^{\infty} \sum_{k=0}^{4} c_k f(2x-k)dx \\ &= \sum_{k=0}^{4}\Big(c_k \int_{-\infty}^{\infty} f(2x-k)dx\Big) \\ &= \Big(\sum_{k=0}^{4} c_k\Big)\Big(\frac{1}{2}\int_{-\infty}^{\infty} f(u)du\Big) \\ &= 4\int_{-\infty}^{\infty} f(u)du.\end{aligned}$$

It follows then that the integral is zero.

Approach using Fourier Transform

Consider the dilation equation $f(x) = \sum c_k f(2x-k)$. Taking the Fourier transform of both sides provides:

$$\hat{f}(\omega) = \sum c_k \int_{-\infty}^{\infty} f(2x-k)e^{-i\pi\omega x}dx,$$

after which the substitution of $u = 2x-k$ gives:

$$\begin{aligned}\hat{f}(\omega) &= \sum \frac{c_k}{2}\int_{-\infty}^{\infty} f(u)e^{-i\pi\omega(u+k)/2}du \\ &= \sum \frac{c_k}{2}e^{-i\pi\omega k/2}\int_{-\infty}^{\infty} f(u)e^{-i\pi\omega u/2}du \\ &= \Big(\sum \frac{c_k}{2}e^{-i\pi\omega k/2}\Big)\hat{f}\Big(\frac{\omega}{2}\Big).\end{aligned}$$

Let $P(\omega) = \frac{1}{2}\sum \frac{c_k}{2}e^{-i\pi\omega k}$ and observe that the function $\hat{f}$ satisfies

$$\begin{aligned}\hat{f}(\omega) &= P\Big(\frac{\omega}{2}\Big)\hat{f}\Big(\frac{\omega}{2}\Big) \\ &= P\Big(\frac{\omega}{2}\Big)P\Big(\frac{\omega}{2^2}\Big)\hat{f}\Big(\frac{\omega}{2^2}\Big) \\ &= \Big(\prod_{k=1}^{n} P\Big(\frac{\omega}{2^k}\Big)\Big)\hat{f}\Big(\frac{\omega}{2^n}\Big) \\ &\to \Big(\prod_{k=1}^{\infty} P\Big(\frac{\omega}{2^k}\Big)\Big)\hat{f}(0).\end{aligned}$$

This allows for the Fourier transform of f to either be approximated or evaluated exactly using series. The solution f can then be recovered either by recognising the resulting Fourier transform or via the inverse Fourier transform.

3.1.2 Function Vector Spaces

Note: the theorems in this subsection are sourced from [4], which also provides rigorous proofs for each statement, omitted here for brevity. In particular, a justification for the extension of the projection theorem to an infinite basis is omitted, as the focus is an intuition to support the application of the

concepts to wavelets.

Much like the familiar finite-dimensional vector spaces such as $\mathbb{R}^n$, we may define vector spaces over functions. This section provides a short explanation of how the concepts of vector bases, inner products, and projection may be extended to a vector space consisting of functions.

Firstly, we define a set $\mathcal{L}_2$ consisting of *finite energy* functions, which each satisfy the property that $\int_{-\infty}^{\infty} |f(t)|^2 dt$ is not divergent. If we take this set over the field $\mathbb{R}$ and equip it with the zero function $f(t) = 0$ as well as the standard scalar multiplication and addition of functions, it is possible to show that the 10 vector space axioms are satisfied[4] [4]. Furthermore, the space may be equipped with the inner product

$$\langle f, g \rangle = \int_{-\infty}^{\infty} f(t)g(t)dt$$

which induces the L^2 norm

$$||f||_2 = \int_{-\infty}^{\infty} |f(t)|^2 dt.$$

Recall that a basis $\mathcal{B}$ of $\mathbb{R}^n$ is a linearly independent subset of $\mathbb{R}^n$ that spans the space. In particular, if for all pairs $b_i, b_j \in \mathcal{B}$ we have that $\langle b_i, b_j \rangle$ is 1 when $i = j$ and 0 otherwise, then the vectors in the basis are unit and pairwise orthogonal, and by extension $\mathcal{B}$ is an orthonormal basis for $\mathbb{R}^n$. Given such an orthonormal basis, the projection theorem states that we may represent any element in $\mathbb{R}^n$ as a linear combination of the basis vectors, each scaled by their inner product with the element.

We extend this idea to our finite-energy functions vector space. Suppose $\mathcal{C} = \{h_1, h_2, \ldots\}$ is an orthonormal basis for $\mathcal{L}_2$. Then for any element $f \in \mathcal{L}_2$:

$$\begin{aligned} f(t) &= \sum_{i=1}^{\infty} \langle f, h_i \rangle h_i \\ &= \sum_{i=1}^{\infty} \left[\left(\int_{-\infty}^{\infty} f(t)h_i(t)dt \right) h_i \right]. \end{aligned}$$

We recall as well the concept of an orthogonal complement. Let X be a subspace of vector space V. If Y is the set of all $\mathbf{y} \in V$ such that,

$$\langle \mathbf{y}, \mathbf{x} \rangle = 0 \text{ for all } \mathbf{x} \in X$$

then Y is the orthogonal complement of X, denoted $X^{\perp}$. We observe the property that $X \oplus X^{\perp} = V$, where $A \oplus B$ denotes the direct sum, the set of vector sums of elements of A, B with the added condition that $A \cap B$ contains only the zero vector. The most relevant consequence of this is that if we have an orthonormal basis $\mathcal{B}$ for X and an orthonormal basis $\mathcal{C}$ for $X^{\perp}$, then their union forms an orthonormal basis for V.

Finally, we introduce the concept of *support*. We define the support of a function $f : X \to \mathbb{R}$ as the set $\{x \in X \text{ for } f(x) \neq 0\}$. That is, the subset of its domain that it maps to non-zero outputs. We will say a function has *finite support* if its support is a subset of some bounded interval.

[4]We assume the functions are real valued for simplicity. Wavelets can be complex valued, particularly those used for time-frequency analysis, but the reasoning can be easily extended.

3.2 Constructing a Wavelet Basis

To reiterate, we are interested in constructing a basis of functions that satisfy two conditions: the basis should be orthonormal[5], and the functions in the basis should be translated dyadic dilates of one another. Additionally, the functions should have finite support to capture local information, unlike the Fourier basis. For each step, we will apply it to a simple example, the Haar Wavelet, so that we may see the ideas in action.

Construction begins with a *scale* function ϕ that satisfies a particular dilation equation $\phi(x) = \sum_{k=0}^{d-1} \phi(2x-k)$. The choice of this dilation equation determines the resulting wavelet. For example, we have already seen the dilation equation $\phi(x) = \frac{1+\sqrt{3}}{4}\phi(2x) + \frac{3+\sqrt{3}}{4}\phi(2x-1) + \frac{3-\sqrt{3}}{4}\phi(2x-2) + \frac{1-\sqrt{3}}{4}\phi(2x-3)$ for the Daubechies 4-tap (D_4) wavelet. In our example, the Haar Wavelet has a scale function satisfying $\phi(x) = \phi(2x) + \phi(2x-1)$. We can use one of the methods from section 3.1 to show that a valid solution is

$$\phi(x) = \begin{cases} 1 & \text{if } 0 \leq x \leq 1 \\ 0 & \text{otherwise.} \end{cases}$$

Now, we introduce the notation $\phi_{jk}(x) = 2^{j/2}\phi(2^j x - k)$ such that each $j, k \in \mathbb{N}$ produces a particular dilation and translation. The purpose of the constant multiple $2^{j/2}$ is to normalise the function under the L^2 norm (assuming $\phi(x)$ is normalised). Each 'level' of j is essentially a finer dilation that produces a more accurate representation in the time domain (as we sought in figure 3) with a narrower range of support. In the context of the Haar Wavelet, these appear as finer and finer rectangles with different translations, which perhaps lends some insight as to how a linear combination of these would approximate a function. The following plot contains some examples of such ϕ_{jk} for the Haar, produced using Matplotlib:

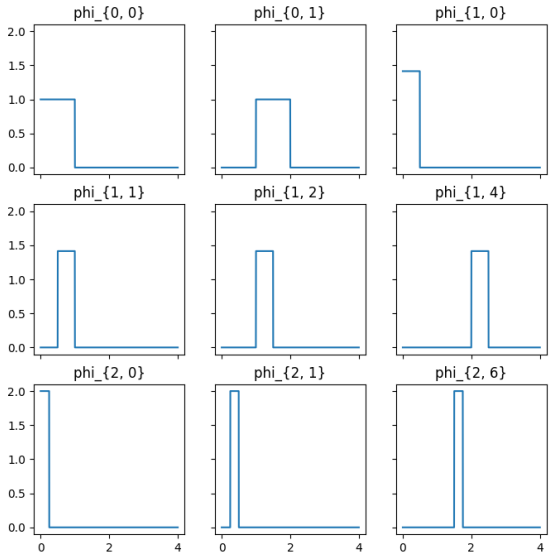

Figure 7: Selected dilation-translations of Haar scale function.

[5]In practice there are variations of wavelet bases that are not orthogonal but we will focus on constructing an orthogonal basis.

Observe that as each daughter wavelet is dilated inwards, the frequency it represents increases alongside the precision in time where it may be positioned. This vector basis thus satisfies the conditions we wanted from our wavelet frequency-time graph.

Let us denote with V_j the space (of functions) spanned by ϕ_{jk} for $k \in \mathbb{N}$. We observe that every ϕ_{jk} is an element of the span of $\{\phi_{j+1,r} \text{ for } r \in \mathbb{N}\}$ and therefore each V_j is a subset of V_{j+1}.

Proof. Consider from the dilation equation that

$$\begin{aligned}
\phi(2^j x - k) &= \sum_{r=0}^{d-1} c_k \phi\left(2^{j+1}x - 2k - r\right) \\
2^{-j/2}\phi_{jk}(x) &= \sum_{r=0}^{d-1} c_k 2^{-(j+1)/2}\phi_{j+1,2k+r}(x) \\
\phi_{jk}(x) &= \sum_{r=0}^{d-1} \frac{c_k}{\sqrt{2}}\phi_{j+1,2k+r}(x).
\end{aligned}$$

Then $\phi_{jk} \in \text{span}\{\phi_{j+1,\ell} \text{ for } \ell \in \mathbb{N}\}$ for all $k \in \mathbb{N}$, we can conclude that $\text{span}\{\phi_{jk} \text{ for } k \in \mathbb{N}\} \subseteq \text{span}\{\phi_{j+1,k} \text{ for } k \in \mathbb{N}\}$. Therefore, $V_j \subseteq V_{j+1}$. □

Now, let $W_j = V_j^{\perp}$ with respect to V_{j+1}. That is to say, every element of W_j is orthogonal to V_j and their sum is V_{j+1}. The idea is, as discussed earlier, every level of V_j covers a narrower time region than the previous, allowing for more detail, so the elements of V_j can be interpreted to represent lower-frequency components (approximations), while the elements of W_j represent higher-frequency details. Let $W_j = \text{span}\{\psi_{jk} \text{ for } k \in \mathbb{N}\}$ and define the 'mother wavelet' $\psi(x)$ to be given by

$$\psi(x) = \sum_{k=0}^{d-1} (-1)^k c_{d-1-k}\phi(2x - k).$$

It can be shown that this satisfies the orthogonal complement requirement under certain conditions on c_k and that integer translations of ψ are normal, as well as orthogonal to each other and to the scaling function $\phi(x)$ [3][11]. As an example, we provide a proof that each ψ_{jk} is orthogonal to the scale function[6]. We will assume that the the scale function is normalised and all of its integer translations are pairwise orthogonal (1) and that our scale function has an even number of terms (2). Let δ_{xy} denote the Kronecker Delta.

Proof. From the definition we have that

$$\begin{aligned}
\langle \phi(x), \psi(x-k) \rangle &= \int_{-\infty}^{\infty} \sum_{s=0}^{d-1} c_s \phi(2x - s) \sum_{t=0}^{d-1} (-1)^t c_{d-1-t}\phi(2x - 2k - t)dx \\
&= \sum_{s=0}^{d-1}\sum_{t=0}^{d-1} (-1)^t c_s c_{d-1-t} \int_{-\infty}^{\infty} \phi(2x - s)\phi(2x - 2k - t)dx \\
&= \sum_{s=0}^{d-1}\sum_{t=0}^{d-1} (-1)^t c_s c_{d-1-t}\delta_{s,2k+t} \text{ using (1)}
\end{aligned}$$

[6] This proof is adapted from Lemma 11.7 of [3]. I have tried to add as much extra detail as possible to make the proof clearer.

$$= \sum_{s=0}^{d-1} \sum_{\substack{t=0 \\ s=2k+t}}^{d-1} (-1)^t c_s c_{d-1-t} \delta_{s,2k+t} + \sum_{s=0}^{d-1} \sum_{\substack{t=0 \\ s\neq 2k+t}}^{d-1} (-1)^t c_s c_{d-1-t} \delta_{s,2k+t}$$
$$= \sum_{t=0}^{d-1} (-1)^t c_{2k+t} c_{d-1-t}.$$

Recalling that $c_\ell = 0$ for $\ell < 0$ and $\ell > d-1$, we obtain the condition $0 \leq 2k+t \leq d-1$ and $0 \leq d-1-t \leq d-1$. This requires that $-2k \leq t \leq d-1-2k$ so the sum is equivalent to

$$\sum_{t=0}^{d-1-2k} (-1)^t c_{2k+t} c_{d-1-t} = \sum_{s=0}^{d-1-2k} (-1)^{d-1-2k-s} c_{2k+d-1-2k-s} c_{d-1-d+1+2k+s}$$
$$= - \sum_{s=0}^{d-1-2k} (-1)^s c_{d-1-s} c_{2k+s} \text{ using (2).}$$

It follows that the sum is 0. Therefore, the inner product of ϕ_{00} and $\psi(x-k)$ is 0. We then observe that

$$\begin{aligned}
\langle \phi, \psi_{jk} \rangle &= \int_{-\infty}^{\infty} \phi(x) 2^{j/2} \psi(2^j x - k) dx \\
&= 2^j \int_{-\infty}^{\infty} \sum d_\ell \phi(2^j x - \ell) \psi(2^j x - k) dx \\
&= 2^j \sum d_\ell \int_{-\infty}^{\infty} \phi(2^j x - \ell) \psi(2^j x - k) dx \\
&= \sum d_\ell \int_{-\infty}^{\infty} \phi(u) \psi(u + \ell - k) du, \text{ where } u = 2^j x - \ell \\
&= 0, \text{ using the previous part of the proof.}
\end{aligned}$$

It is always possible to find such a sequence d_ℓ as $V_0 \subseteq V_j$ implies that $\phi(x) \in \text{span}\{\phi_{j\ell} \text{ for } \ell \in \mathbb{N}\}$. It follows then that $\langle \phi_{00}, \psi_{jk} \rangle = 0$ for all j, k. □

A more intuitive overview for why this function satisfies the role of a basis for W_j is is that if we consider some ϕ_{0k} with support $[a, b]$, then the $\phi_{1\ell}$ functions gain an extra degree of flexibility, allowing for $[a, (a+b)/2)$ and $[(a+b)/2, b]$ to be changed independently. Then the definition of ψ provides a basis vector which can express these details when combined with ϕ (A concrete example of this is shown later). For the Haar Wavelet, we obtain $\psi(x) = \phi(2x) - \phi(2x-1)$, the piecewise expression and graph (produced using Matplotlib) for which are shown below:

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

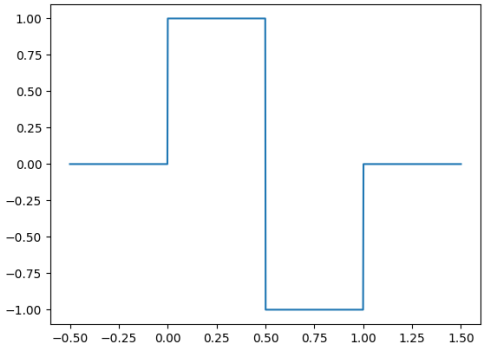

Figure 8: Haar mother wavelet expression and graph.

We now seek to unify the set of vectors $\{\phi_{jk}, \psi_{jk} \text{ for } j,k \in \mathbb{N}\}$ into a single basis. In particular, though the linear independence properties we require are satisfied on each level j (using the pairwise orthogonality between each ϕ_{jk} and $\psi_{j\ell}$ for fixed j), it is quite obvious that crossing between levels of j leads to linear dependencies, as each V_j is a subset of V_{j+1}. We recall that each lower-frequency component ϕ_{jk} is composed of the sum of higher-frequency components on level $j+1$ which explains this dependence. An intuitive solution then, is instead of taking all the components from V_{j+1}, we will take only what we are missing on level V_j, that is, the details in $W_j = V_j^{\perp}$. We can verify this is valid for the Haar functions, for which $\text{span}\{\phi_{jk}, \phi_{j+1,2k}, \phi_{j+1,2k+1}\} = \text{span}\{\phi_{jk}, \psi_{jk}\}$.

Proof. We observe that $\phi(x) = \phi(2x) + \phi(2x-1)$ and $\psi(x) = \phi(2x) - \phi(2x-1)$ by definition. It follows that

$$\begin{aligned}
\phi_{jk}(x) &= 2^{j/2}\big[\phi\big(2^{j+1}x - 2k\big) + \phi\big(2^{j+1}x - 2k - 1\big)\big] \\
&= 2^{j/2}\phi_{j+1,2k}(x) + 2^{j/2}\phi_{j+1,2k+1}. \\
\psi_{jk}(x) &= 2^{j/2}\big[\phi\big(2^{j+1}x - 2k\big) - \phi\big(2^{j+1}x - 2k - 1\big)\big] \\
&= 2^{j/2}\phi_{j+1,2k}(x) - 2^{j/2}\phi_{j+1,2k+1}(x).
\end{aligned}$$

Then we may write

$$\begin{aligned}
\phi_{j+1,2k}(x) &= 2^{-(j/2+1)}\phi_{jk}(x) + 2^{-(j/2+1)}\psi_{jk}(x), \\
\phi_{j+1,2k+1}(x) &= 2^{-(j/2+1)}\phi_{jk}(x) - 2^{-(j/2+1)}\psi_{jk}(x),
\end{aligned}$$

from which the equivalence of the spans is clear. □

This implies that for the Haar wavelet, we may obtain an orthonormal basis $\{\phi_{0k}, \psi_{j,2\ell} \text{ for } j,k,\ell \in \mathbb{N}\}$. The values for which k and ℓ are bounded depend on the support of the function we are seeking to represent in this basis, and in practice we may limit j to a certain value depending on the resolution to which we wish to represent the function. Some plots of selected basis vectors produced using Matplotlib are shown in the figure below:

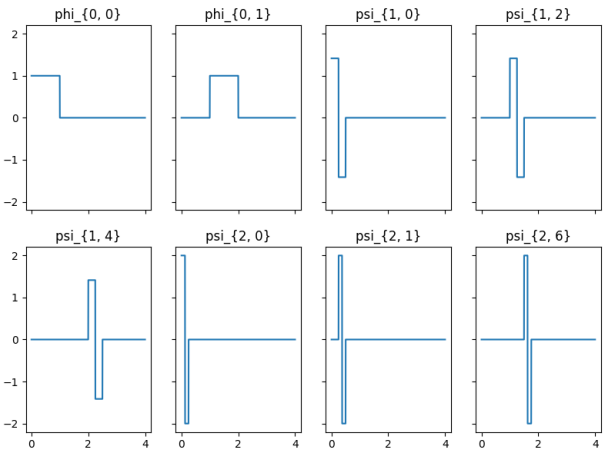

Figure 9: Selected basis vectors from the Haar wavelet family.

To summarise, the scale function ϕ solving a dilation equation is translated and dilated to form an increasingly detailed sequence of subspaces $V_0, V_1, V_2, \ldots$. We then introduce the orthogonal complement $W_0, W_1, W_2, \ldots$ spanned by wavelet functions ψ_{jk}, which can be used to express the V subspaces under an orthonormal basis. We showed how this can be applied to a simple wavelet example, but the same steps may be extended to any dilation equation satisfying certain constraints[7].

3.3 Wavelet Transform

Given a wavelet basis, we now seek to express a function $f : [0, \infty) \to \mathbb{R}$ over this basis[8]. We will focus on the discrete wavelet transform. Supposing we have an orthonormal basis $\mathcal{W} = \{\phi_{0k}, \psi_{j\ell}\}$ for suitable j, k, ℓ such that $f \in \text{span}(\mathcal{W})$, the projection theorem gives:

$$f(x) = \sum_{k=0}^{\infty} \langle f, \phi_{0k} \rangle \phi_{0k}(x) + \sum_{j=0}^{\infty} \sum_{\ell=0}^{\infty} \langle f, \psi_{j\ell} \rangle \psi_{j\ell}(x).$$

However, if we note the way in which the wavelet basis was constructed, we can achieve a lot more flexibility in representing a function. In particular, we recall that the sets V_j for $j \in \mathbb{N}$ capture increasingly fine approximations for a function using only the scale functions ϕ_{jk}. Supposing we were to stop at V_j, the lost details between V_j and V_{j+1} can be represented in W_j such that $V_j \oplus W_j = V_{j+1}$. We can continue this to get a more and more detailed orthonormal basis: $V_j \oplus W_j \oplus W_{j+1} \oplus W_{j+2} \oplus \ldots$ up to some decided bound W_{J-1}, which can be concluded from the sampling rate, such that f is approximately in V_J (in the sense that inner products of f onto elements of W_J are near zero) [12]. Then,

$$f(x) = \sum_{k=0}^{\infty} \langle f, \phi_{ik} \rangle \phi_{ik}(x) + \sum_{j=i}^{J-1} \sum_{k=0}^{\infty} \langle f, \psi_{jk} \rangle \psi_{jk}(x).$$

In the case that f has finite support $[0, M]$ and the wavelets have finite support $[0, L]$ then we only need to take the sum up to $k = 2^j M - L$ - that is, the true length of the infinite sum varies with the support of f and is not actually infinite in practice. This gives us the flexibility of choosing a particular level of approximation versus detail. As an example, we apply this to the Haar basis we derived earlier (note that the exact basis from earlier was specifically for $i = 0$ but is easily extended). I implemented the program using Numpy and Matplotlib, with source code available in the appendix. We find an approximation for the function $f(x)$ which is $x \sin(\exp(x))$ with support $[0, 4]$ and provide several examples of different i, J values. The function $f(x)$ is plotted in blue while the approximation under the Haar basis is red. The transparent orange plot is the lower frequency ϕ approximation while the green plot is the higher frequency ψ details. As an example we find that for $i = 0, J = 1$, we may write $f \approx 0.41\phi_{00} - 0.47\phi_{01} + 0.09\phi_{02} + 0.07\phi_{03} - 0.17\psi_{00} + 0.00\psi_{01} - 0.18\psi_{02} + 0.06\psi_{03}$. Notably, as we increase i, J we get a finer approximation for the function, with particular importance for the higher frequencies for the larger x-values. We can see that without sufficient high-frequency ψ details, the basis struggles to represent the curve properly. Additionally, we see one of the most significant drawbacks of

[7] These constraints are more relevant to the construction of wavelet systems and less so to their use. More information on these constraints can be found in [3] and [5].

[8] If we take $f : \mathbb{R} \to \mathbb{R}$ then we should allow for negative k as well.

the Haar wavelet, which is that it is not continuous and therefore produces sharp, jagged approximations regardless of the precision. Note as well the distinction between the approximation and detail levels and how this might be used in particular applications.

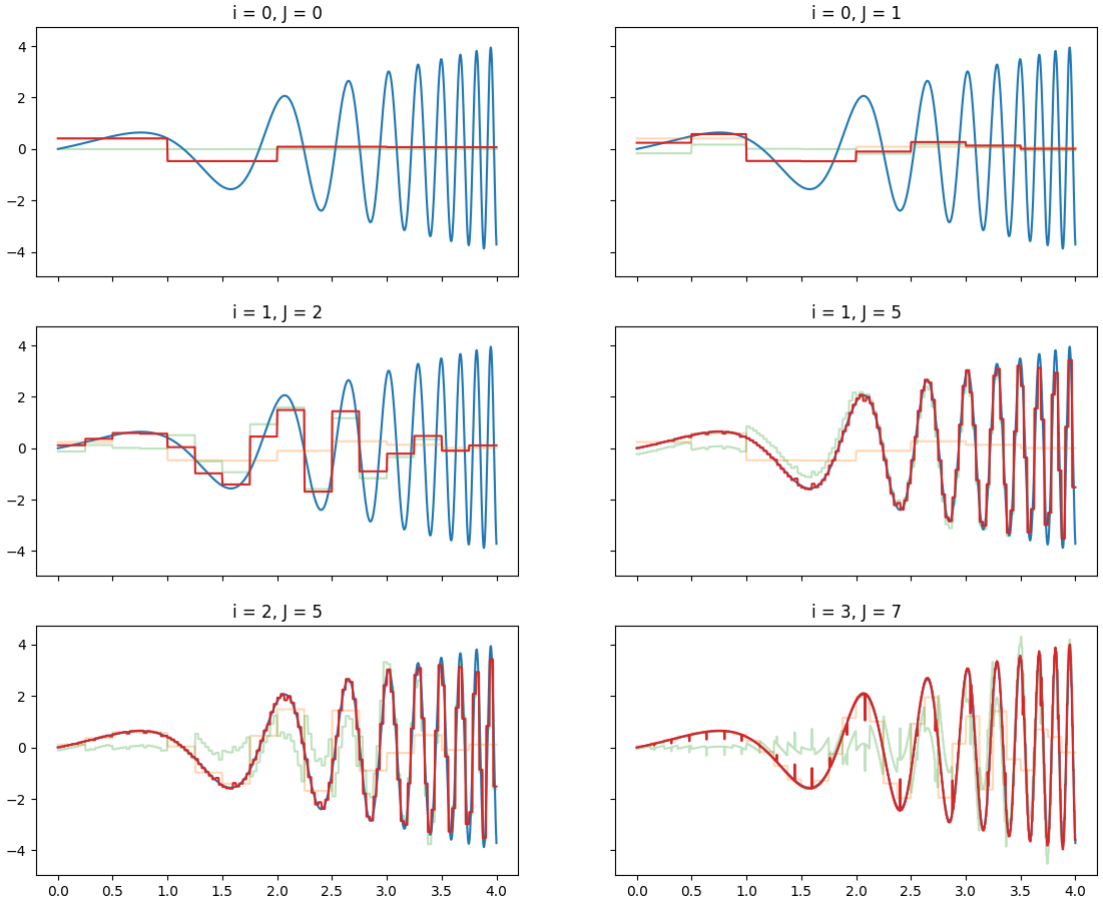

Figure 10: Program output for projection onto Haar basis for various i, J values.

As another example, we project $\sin(\pi x) + 2\sin(4\pi x)$ onto the Haar basis with $i = 1, J = 5$:

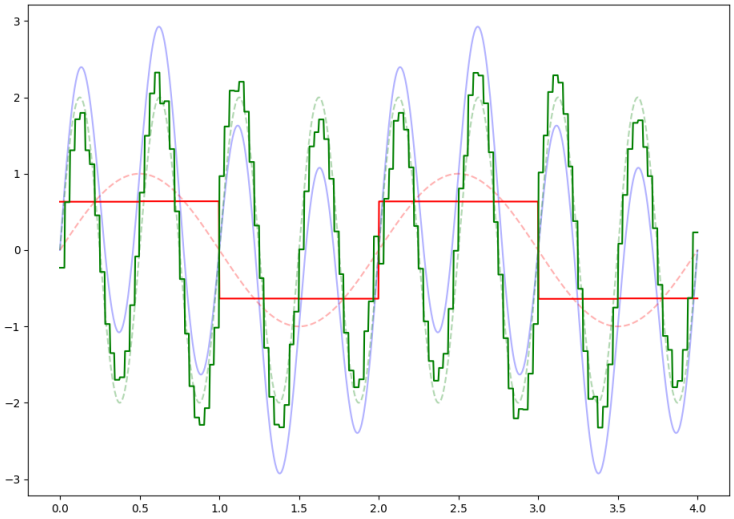

Figure 11: Program output for projection onto Haar basis of $\sin(\pi x) + 2\sin(4\pi x)$.

In this plot, the function is plotted in blue and its components $\sin(\pi x)$ and $2\sin(4\pi x)$ are plotted as dotted lines in red and green, respectively. The solid red and green lines are the projection onto the approximation and detail bases. The key aspect of this that I wanted to note was that the projection is able to separate the two frequencies in a similar manner to the Fourier transform, highlighting how wavelets may be used in frequency analysis. Of course, there are many other considerations for a time-frequency scalogram using wavelets, but this is a rudimentary view into how one might work. More examples in section 4.

While this implementation does satisfy what we seek to do, it is clear that it is not fast enough for practical use. Fortunately, there exists a faster variant, the fast wavelet transform, which we will sketch the high-level ideas of. We consider a vector representation of our function $\mathbf{f}$ which is just $f(x)$ sampled at N points. Supposing that we want to compute a wavelet transform from level i up to level $J-1$, we will start at level $J-1$ in subspace V_J and use a pair of convolution-based low and high pass filters to produce a pair of vectors $\mathbf{f}_A$ and $\mathbf{f}_D$ which respectively contain an averaged approximation of the function which are analogous to the lower frequency $\phi_{J-1,k}$ vectors, as well as the details that make up the difference to the approximation, analogous to the $\psi_{J-1,k}$ details. These vectors are then downsampled by a factor of 2 to produce two vectors of length approximately $N/2$. We may then calculate the $\psi_{J-1,k}$ coefficients from the high frequency $\mathbf{f}_D$ vector. Then, the $\mathbf{f}_A$ approximation vector (analogous to the $\phi_{J-1,k}$ basis vectors) is passed down to the next level $J-2$ as $\mathbf{f}$ and we recurse, analysing in subspace V_{J-1} using the facts that $V_{J-1} \oplus W_{J-1} = V_J$ and that we have extracted the $\psi_{J-1,k}$ coefficients. This is repeated at each level until we reach i, such that we are left with one set of ϕ_{ik} coefficients and the ψ_{jk} coefficients at each level [12]. The resulting tree-like structure is visible in the diagram below:

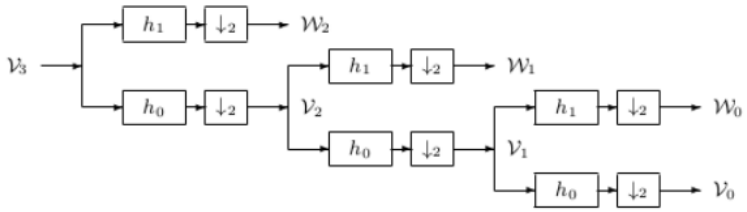

Figure 12: Discrete Wavelet Transform Tree (from Burrus et. al [12]).

Each convolution is $O(N)$ as the filters are of constant length, so the time complexity satisfies $T(N) = T(N/2) + cN$. Case 3 of the Master Theorem with regularity condition satisfied gives $T(N) \in \Theta(N)$, so this fast algorithm runs linear with the size of the input vector.

3.4 Example Wavelets

Part of the power of wavelets is their generality: different wavelet families may be applied to different problems that they are better suited to. In this section, we show some example wavelets that are used in practical applications for the purpose of helping to visualise the appearance of such functions particularly in comparison to the Fourier basis. In particular, note the support of each wavelet and how differing wavelet shapes may cater to particular uses. *All examples in this section are sourced from MathWorks' Wavelet Toolbox Documentation [8].*

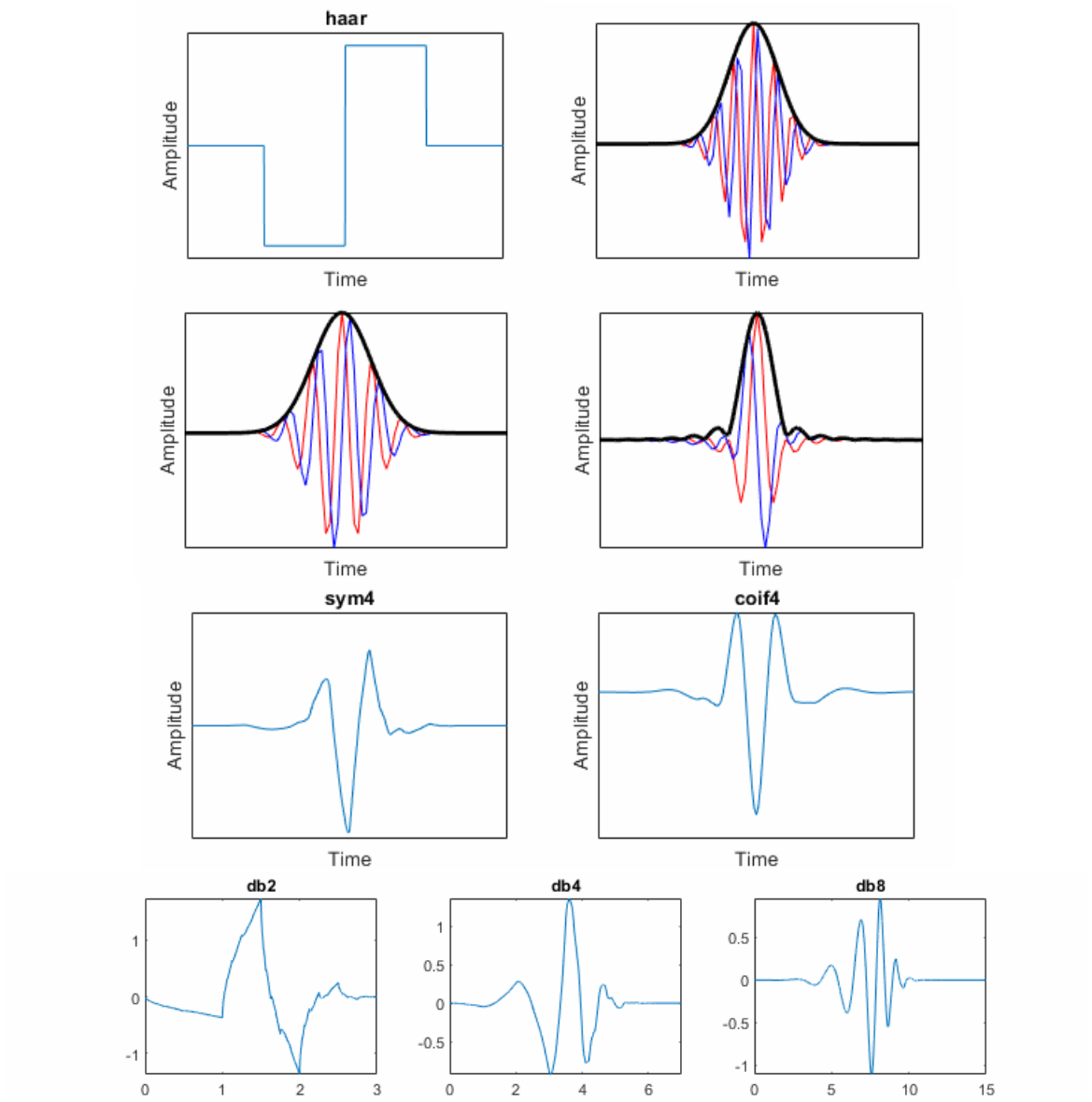

Figure 13: Selected Wavelets (from MathWorks [8]).

The simplest wavelet is the Haar Wavelet (top left), which we have already visited. Its shape is well suited to edge detection. The next three wavelets are the Morse, Analytic Morlet, and Bump Wavelets, respectively, which are complex-valued and used for time-frequency analysis with the continuous wavelet transform. Note the difference provided by each wavelet in terms of variance in the time dimension (which we recall is inversely proportional to variance in frequency measurement). The next five wavelets (from the Symlets, Coiflets, and Daubechies families) are used for signal representation and processing with the discrete transform and are orthogonal. For example, they have applications in denoising and image processing. We now explore these applications in more detail.

4 Applications of Wavelets

Wavelets have found use in a range of applications with sets of wavelet families developed for use in particular applications. We will discuss some practical applications of wavelets in the areas of time-

frequency analysis, denoising, and compression as well as compare their performance to alternatives.

4.1 Time-Frequency Analysis

Much like the Fourier Transform and Short-Time Fourier Transform, wavelets can be used to perform time-frequency analysis of a signal. In this case, the frequency is represented based on the powers of two dilation of the wavelets in the form of a scalogram which can be obtained using a continuous wavelet transform (note the logarithmic axis in the scalogram of figure 14). The limitations of the FT and STFT were a key aspect motivating the construction of wavelets in section 2, as can be seen in this example from MathWork's Wavelet Toolbox Documentation, which shows the Fourier Transform and spectrograms obtained from the analysis of a hyperbolic chirp signal.

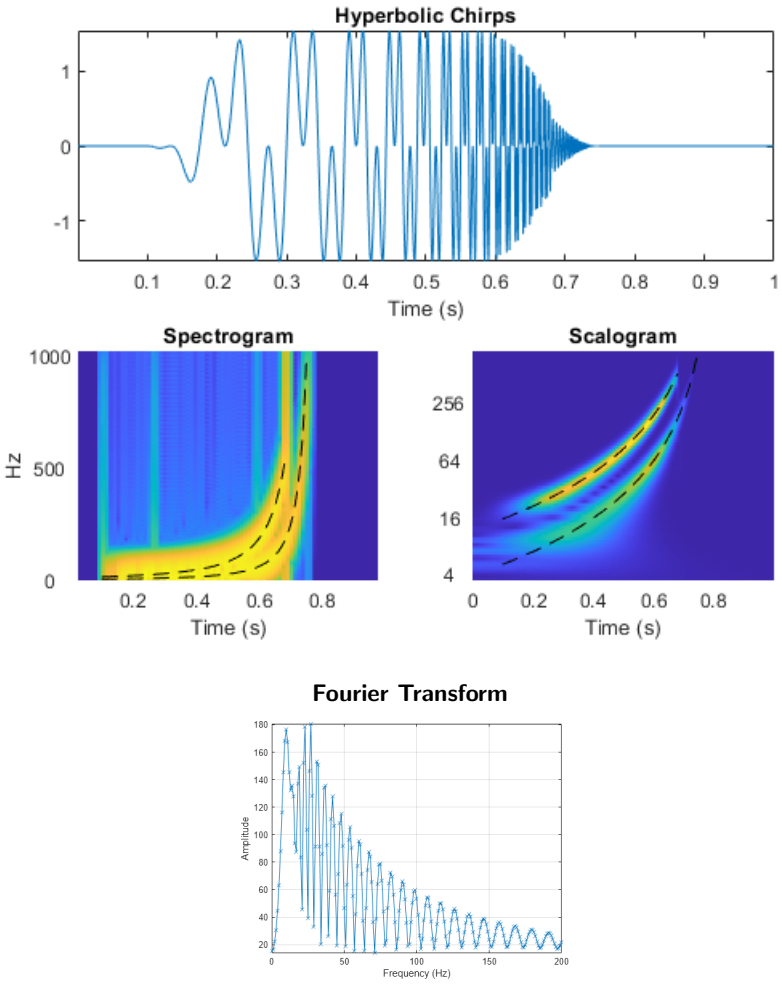

Figure 14: Hyperbolic chirp signal and corresponding STFT spectrogram, CWT scalogram, and FT frequency domain (from MathWorks [7]).

We can see that the Fourier Transform at the bottom of the figure accurately captures all frequencies in the signal, but does not provide any temporal information. Specifically, the information that the chirp is increasing in frequency over time is not presented. Furthermore, as discussed, the STFT struggles with representing quick, large changes in frequency and therefore does not capture the frequencies well in a localised manner in comparison to the scalogram. The exact frequencies can be recovered from the

scalogram using information about the particular wavelet used, which in this case is the Generalised Morse Wavelet (see section 3.3).

4.2 Denoising

As we have touched upon, wavelet analysis can be used to remove noise from a signal or image by first performing a transform and then performing 'thresholding'. The general idea is that upon representation in the wavelet basis, small details (wavelet components) with low amplitude and high frequency can be isolated and either removed (hard thresholding) or the whole signal can be compressed towards zero amplitude to remove these small details (soft thresholding). Again, wavelet methods exhibit better performance with signals with sudden changes in frequency when compared to STFT, which makes them particularly ideal for denoising as meaningful audio tends to be composed of such sudden changes. This can be seen in an example from MATLAB of a generated signal augmented with Gaussian white noise which is then denoised using a variation of soft threshold wavelet denoising. The wavelet used is the symlet sym8.

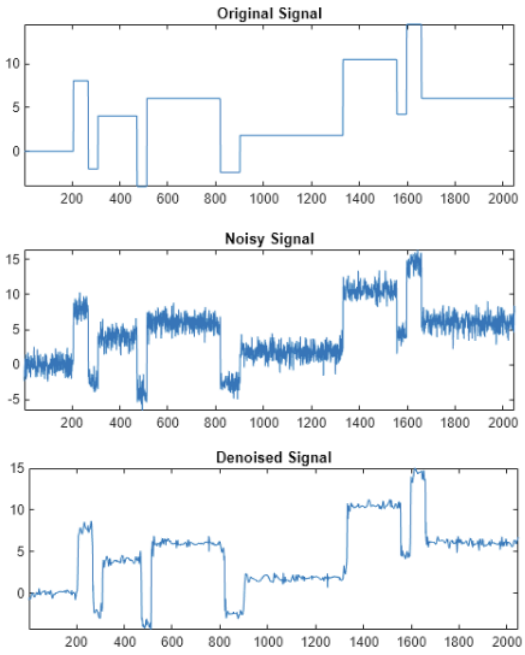

Figure 15: Generated signal (top) augmented with white noise (centre) and denoised using wavelet denoising (bottom) (from MathWorks [9]).

I was interested in how my Python Haar projection implementation would handle noise removal, so I implemented hard thresholding in the projection and I ran it again with the curve from section 3 augmented by Gaussian noise such that it is of the form $x\sin(\exp(x)) + \eta$ where $\eta \sim N(0, 0.2^2)$ and different i, J values. We plot the results of two different types of noise removal that I considered: one in which we take only the low-frequency ϕ vectors after projection and one in which we utilise a hard threshold to remove detail vectors whose coefficients are below a certain value.

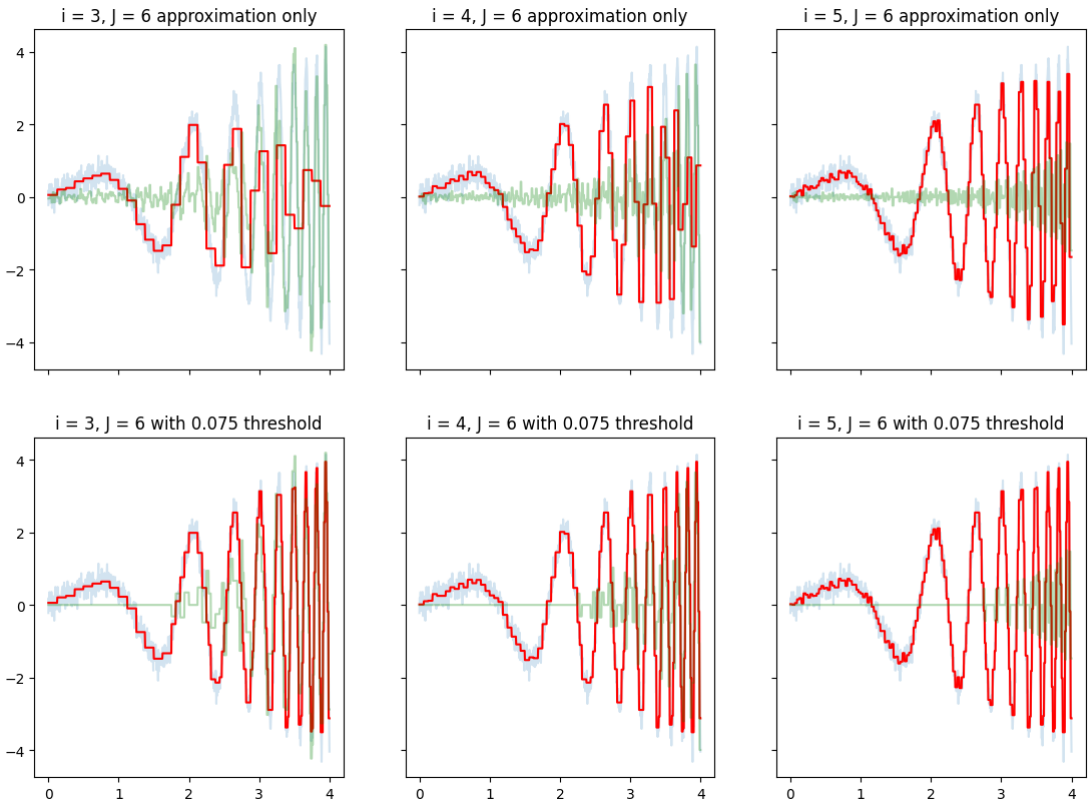

Figure 16: Program output for projection onto Haar basis after addition of Gaussian noise.

I repeated this for $\eta \sim N(0, 0.5^2)$.

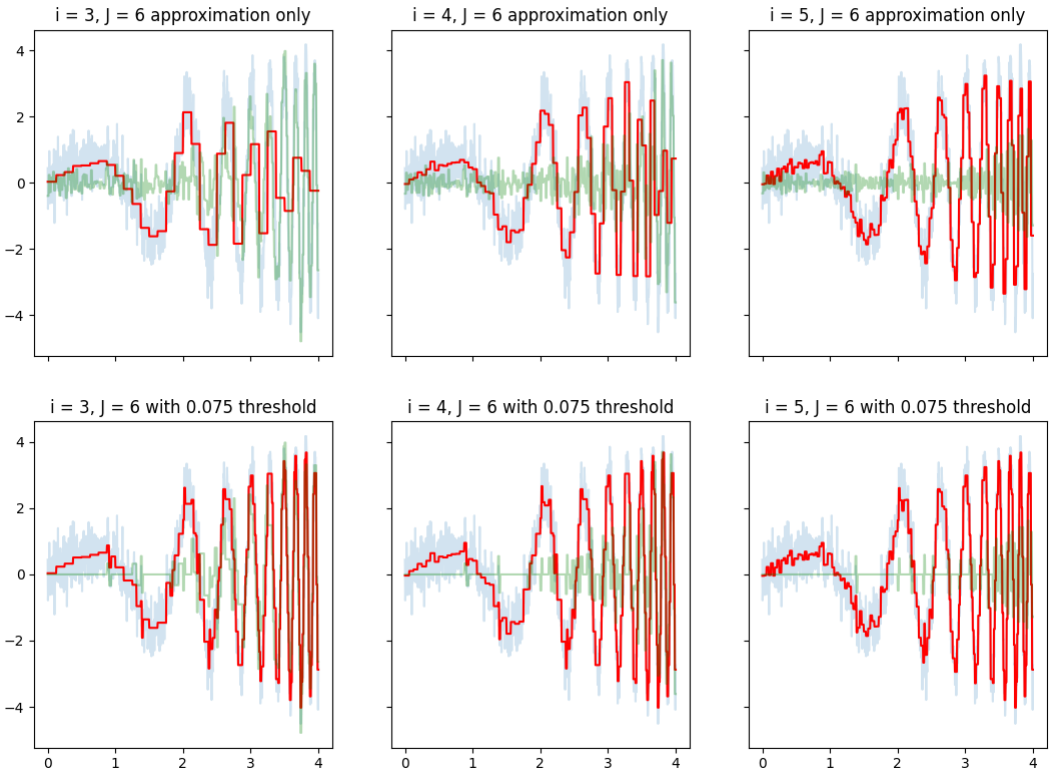

Figure 17: Program output for projection onto Haar basis after addition of Gaussian noise.

In the top row of plots in each figure, the red is the ϕ approximation only, while the blue and green remain the function and wavelet details, respectively. In the bottom line of plots, the red curve is the whole wavelet projection, but ψ details have been thresholded to only details exceeding 0.075 (value found empirically). In both examples, we can see that the low frequency ϕ approximations struggle to represent the curve once it reaches higher frequencies (as we should expect). We can see that for higher i values, the ϕ projection appears to overfit the noise and begins to incorporate some high frequency elements in the noise which are clearly not part of the original curve. Therefore, if we only use the ϕ values, we either are lacking the details to display higher frequency (eg. for $i = 3, 4$), or we are overfitting (eg. $i = 5$). With thresholding however, we can see that the lower i values are still able to approximate the higher frequency values in the curve while not overfitting, as the details that represent those high frequencies have not been cut as they are above the threshold. For the higher variance example, we can see that the thresholding is slightly less efficient, as the noise is large enough that some spikes caused by noise are being recognised as details in the curve (perhaps a higher threshold would be more effective). We can see again the limitations of the Haar wavelet in the jagged approximation of all curves, but the program still does a relatively good job of filtering out noise, hinting at the possible efficacy of such a transform with a more well chosen basis.

4.3 Data Compression

As we may expect by now, another application of Fourier Transforms that can be extended to the more general wavelets is data compression. This may be done through a similar approach to denoising, where thresholding is used to remove small details which have little effect on the signal, removing unnecessary information. In particular, for purposes such as image and audio compression, the particular criteria by which components are thresholded may be fine-tuned to exploit human visual or auditory perception.

Perhaps the most notable application of wavelets to data compression is JPEG2000, an image compression algorithm intended to replace JPEG (which used a Fourier-related transform) using discrete wavelet transform. Despite JPEG2000 offerring higher compression efficiency, better resilience to noise, higher quality, and being faster than JPEG, a reluctance to adopt it and its increased memory demands led to its decline [10].

4.4 Other Applications and Wavelet Adoption

Wavelets are used in a range of other unexpected applications, which are beyond the scope of this essay, including in numerical analysis, convolutional neural networks, financial/economic trend analysis, and in steganography[9]. In general, there are many applications where wavelet methods can be observed to outperform existing approaches, but a recurring trend is that outside of specialised use their mathematical complexity limits their adoption.

[9]Some examples of literature covering these topics:
Vasilyev, Yuen, Paolucci 1997 - *Solving PDEs Using Wavelets* [link]
Fujieda, Takayama, Hachisuka 2018 - *Wavelet Convolutional Neural Networks* [link]
Crowley 2006 - *Analysing Productivity Cycles in the Euro area, US and UK Using Wavelet Analysis* [link]
Hemalatha, Acharya, Renuka 2014 - *Wavelet Transform Based Steganography Technique to Hide Audio Signals in Image* [link].

5 Conclusion

Wavelets are a powerful tool that present a solution to limitations encountered by traditional methods like the Fourier and Short-Term Fourier Transforms. We have demonstrated their versatile mathematical construction, which allows for the creation of a range of wavelet families whose properties can be tailored to various tasks. These diverse families have found use in a wide range of contemporary applications such as in signal analysis and processing, and compression algorithms. Despite their inconsistent adoption across some applications, they are a comparatively new concept whose capabilities may be further expanded and have demonstrable advantages in a diverse range of use cases.

6 References

1. MacLennan B. 1994, *Gabor Representations of Spatiotemporal Visual Images*, University of Tennessee Department of Computer Science (p. 2). [link]

2. Sun P. 2014, *Comparison of STFT and Wavelet Transform in Time-frequency Analysis*, (pp. 5-8, 10-12). [link]

3. Blum A., Hopcroft J., Kannan R. 2020, *Foundations of Data Science*, Cambridge University Press (pp. 341-359).

4. Gallager R. 2006, *Course materials for 6.450 Principles of Digital Communications I, Fall 2006.*, MIT OpenCourseWare (pp. 141-152). [link]

5. Strang G. 1989, *Wavelets and Dilation Equations: A Brief Introduction*, Society for Industrial and Applied Mathematics Review (SIREV), 31(4) (pp. 614-627). [link]

6. Graps A. 1995, *An Introduction to Wavelets*, IEEE Computational Science and Engineering, 2. [link]

7. MathWorks 2024, *Time-Frequency Analysis and Continuous Wavelet Transform*, mathworks.com [link]

8. MathWorks 2024, *Choose a Wavelet*, mathworks.com [link]

9. MathWorks 2024, *Wavelet Denoising and Nonparametric Function Estimation*, mathworks.com [link]

10. Kelecheva B. 2018, *Why JPEG2000 Never Took Off*, ANSI [link]

11. Vidakovic B., Mueller P. 1994, *Wavelets For Kids*, Duke University Institute of Statistics and Decision Scieences, [link]

12. Sidney Burrus C., Gopinath R., Guo H. 1998, *Wavelets and Wavelet Transforms*, La Recherche 67, [link]

7 Appendix

Algorithm 1 Implementation in Python

```
# Python implementation of IterativeDilation function to approximate the solution to a dilation equation
# The values in lines 8 through 13 may be changed to satisfy different dilation equations, currently used to
#   compute the Daubechies scaling function for BHK exercise 11.9

import numpy as np
import matplotlib.pyplot as plt

low = 0
precision = 1000
high = 8
d = 4
c = [0.683012701892, 1.18301270189, 0.316987298108, -0.183012701892]
runs = 100

slices = (high - low) * precision

def improve(c, phi):
    phis = np.zeros((d, slices))
    # average consecutive values for phi(2x)
    for i in range(0, slices - 1, 2):
        phis[0][i // 2] = (phi[i] + phi[i + 1]) / 2
    # shift phi(2x) for phi(2x-k)
    for i in range(slices // 2 - 1):
        for j in range(1, d):
            if i + j * precision < slices:
                phis[j][i + j * precision] = phis[0][i]
    # evaluate improved phi(x)
    phi_new = np.zeros(slices)
    for i in range(slices):
        for j in range(d):
            phi_new[i] += c[j] * phis[j][i]
    return phi_new

x = np.linspace(low, high, slices)
phi = np.array([1 for _ in x])
for _ in range(runs):
    phi = improve(c, phi)
plt.plot(x, phi)
plt.show()
```

Algorithm 2 Implementation in Python

```
# Python implementation of MatrixDilation function to approximate the solution to a dilation equation
#   which is zero outside of [0, n]
```

```
# The coefficients on line 9 may be changed to satisfy different dilation equations, currently used to
#   compute a solution for BHK exercise 11.5

import numpy as np
import matplotlib.pyplot as plt

coefficients = np.array([1, 2, 2, 2, 1])

def C(k, i, n, c):
    return c[2 * k - i] if max(0, 2 * k - n) <= i and i <= min(n, 2 * k) else 0

def coefficient_matrix(c, n):
    M = np.zeros((n + 1, n + 1))
    for i in range(n + 1):
        for j in range(n + 1):
            M[i][j] = C(i, j, n, c)
    return M

def MatrixDilation(c, n, P = 8):
    # generate the coefficient matrix
    M = coefficient_matrix(c, n)
    print(M)
    # find an eigenvector corresponding to eigenvalue 1
    eigenvalues, eigenvectors = np.linalg.eig(M)
    index = -1
    for i in range(len(eigenvalues)):
        if abs(eigenvalues[i] - 1) < 10e-5:
            index = i
    if index == -1:
        print("no eigenvector found")
        return [], []
    # extract eigenvector
    v = eigenvectors[:, index]
    print(v)
    # r stores pairs containing the points
    r = [(i, v[i]) for i in range(n + 1)]
    # insert each depth
    for p in range(1, P + 1):
        twoexp = 2 ** p
        for i in range(1, twoexp * n + 1, 2):
            yval = 0
            for k in range(n + 1):
                if 2 * i / twoexp - k >= 0 and 2 * i / twoexp - k <= n:
                    yval += c[k] * r[i - k * 2 ** (p - 1)][1]
            r.append((i / twoexp, yval))
        r.sort()
    return [t[0] for t in r], [t[1] for t in r]

n = len(coefficients) - 1
```

```
x, y = MatrixDilation(coefficients, n)
plt.plot(np.array(x), np.array(y))
plt.show()
```

Haar Wavelet Projection/Denoising in Python

```
# Python program for graphing Haar wavelet approximations to a function func with
    support in
# [0, xmax].

import numpy as np
import matplotlib.pyplot as plt

slices = 1000
xmax = 4
def func(t):
    return t * np.sin(np.exp(t)) if t >= 0 and t <= 4 else 0
def augmented(t):
    return func(t) + np.random.normal(0, 0.2)

x = np.linspace(0, xmax, slices)
# Haar basis definitions
def haar(t):
    return 1 if t >= 0 and t <= 1 else 0
def wavelet(t):
    return 1 if t >= 0 and t < 0.5 else -1 if t >= 0.5 and t <= 1 else 0
def phi(t, j, k):
    return 2 ** (j / 2) * haar(2 ** j * t - k)
def psi(t, j, k):
    return 2 ** (j / 2) * wavelet(2 ** j * t - k)

# Approximate inner product
def innerproduct(f, g):
    return np.inner(f, np.vectorize(g)(x)) * xmax / slices

# Approximates the projection onto haar basis, set threshold for denoise
def approximate(f, start_depth, end_depth, threshold = 0):
    approx = np.zeros(slices)
    # evaluate phis at level i
    kmax = 2 ** start_depth * xmax
    for k in range(kmax):
        ip = innerproduct(f, lambda t : phi(t, start_depth, k))
        print("phi @", start_depth, k, "=", ip)
        approx += ip * np.vectorize(lambda t : phi(t, start_depth, k))(x)
    details = np.zeros(slices)
    # evaluate psi at each level upwards
    for j in range(start_depth, end_depth):
        kmax = 2 ** j * xmax
        for k in range(kmax):
            ip = innerproduct(f, lambda t : psi(t, j, k))
            print("psi @", j, k, "=", ip)
```

```
            if abs(ip) > threshold:
                details += ip * np.vectorize(lambda t : psi(t, j, k))(x)
    return approx, details

# Example usage (used in figures 10, 11, 16, 17)
y = np.vectorize(func)(x)

fig, axes = plt.subplots(3, 2, sharex=True, sharey=True)
lvals = [0, 0, 1, 1, 2, 3]
hvals = [0, 1, 2, 5, 5, 7]
for i in range(3):
    for j in range(2):
        l = lvals[i * 2 + j]
        h = hvals[i * 2 + j]
        axes[i][j].plot(x, y)
        a, d = approximate(y, l, h)
        axes[i][j].plot(x, a, alpha = 0.3)
        axes[i][j].plot(x, d, alpha = 0.3)
        axes[i][j].plot(x, a + d)
        axes[i][j].set_title(f"i = {l}, J = {h}")
plt.show()

y = np.vectorize(augmented)(x)
fig, axes = plt.subplots(2, 3, sharex=True, sharey=True)
lvals = [3, 4, 5]
hvals = [6, 6, 6]
for j in range(3):
    l = lvals[j]
    h = hvals[j]
    axes[0][j].plot(x, y, alpha = 0.2)
    axes[1][j].plot(x, y, alpha = 0.2)
    a, d = approximate(y, l, h)
    axes[0][j].plot(x, a, color = "red")
    axes[0][j].plot(x, d, alpha = 0.3, color = "green")
    axes[0][j].set_title(f"i = {l}, J = {h} approximation only")
    ta, td = approximate(y, l, h, threshold = 0.075)
    axes[1][j].plot(x, ta + td, color = "red")
    axes[1][j].plot(x, td, alpha = 0.3, color = "green")
    axes[1][j].set_title(f"i = {l}, J = {h} with 0.075 threshold")
plt.show()
```