MATH578 Wavelets

Gabriel Rémond-Tiedrez

Fall 2024

Contents

1 What can't Fourier do? 2

2 The language of signals and filters 3

3 Continuous Wavelet Transform 4
3.1 An Alternative to Fourier . . . 4
3.2 Visualisation . . . 6
3.3 Heisenberg Uncertainty for wavelets . . . 6

4 Discrete Wavelet Transform 8
4.1 Discretising Wavelets . . . 8
4.2 Scaling function . . . 9
4.3 Multi-Resolution Approximations . . . 10
4.4 Fast Wavelet Transform . . . 13
4.5 Compact support and vanishing moments . . . 14

5 Coding and applications 16
5.1 Implementations of the FWT . . . 16
5.2 Plotting wavelets . . . 19
5.3 Denoising . . . 19

Appendices 24

A Differentiating with the discretised continuous wavelet transform 24

B Numerically obtaining the filter from the scaling function 25

C Differentiation in wavelet bases 25

1 What can't Fourier do?

Jean Baptiste Joseph Fourier had a full life: the tenth child of his parents' marriage, by 10 he is orphaned, becomes professor by 16 after becoming obsessed with mathematics, takes part in the French Revolution, becomes prefect of the prefecture of Isère at Napoléon's request and is nominated as member of the prestigious Académie Française, as well as many science academies. But most people know him for the omnipresent Fourier transform and its associated Fourier series. And for good reason, one cannot go through a discussion on signal processing, partial differential equations, quantum mechanics or general functional analysis without a primer on the Fourier transform. Its power is not only in its ability to solve problems untouchable without it, but – as with most scientific revolutions – in the fundamentally different point of view it offers.

However, for all its beauty and functionality, the Fourier transform is limited from being a global operator, which becomes obvious once looking at the defining formula

$$\hat{f}(\xi) = \mathcal{F}(f)(\xi) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{+\infty} f(t) e^{-i\xi t} dt = \frac{1}{\sqrt{2\pi}} \langle f, e^{i\xi \cdot} \rangle, \tag{1.1}$$

where $f \in L^2(\mathbb{R})$. The Fourier transform takes inner products with basis functions $e^{i\xi t}$, which can be interpreted as how well $e^{i\xi t}$ describes the behaviour of f. One should think of t as living in the time domain and of ξ in the frequency domain, which is clearer from the inverse Fourier transform:

$$f(t) = \mathcal{F}^{-1}(f)(t) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{+\infty} \hat{f}(\xi) e^{i\xi t} d\xi, \tag{1.2}$$

where one can write f as a sum of the waves $e^{i\xi t}$, weighted by $\hat{f}(\xi)$. In the case where f is T-periodic, one has a discrete sum only involving frequencies that are integer multiples of $2\pi/T$. $\hat{f}(\xi)$ depends on all the values f takes, i.e. the operator $\mathcal{F}$ is global, with no regard for any local phenomena f could contain. So while the Fourier transform diagonalises linear operators, distinguishes frequencies and has perfect resolution in that world, its time resolution is infinitely bad.

As an example, consider the function $\varphi = \chi_{[-1/2,1/2]}(t)$, the indicator function of the interval $[-1/2, 1/2]$. φ is well localised in time, indeed it is 0 everywhere outside of an compact interval. One can check that its Fourier transform is $\hat{\varphi}(\xi) = \frac{\mathrm{sinc}(\xi/2)}{\sqrt{2\pi}}$, where $\mathrm{sinc}(t) = \frac{\sin(\pi t)}{\pi t}$ is the cardinal sin function. Our function φ was compactly supported, but its Fourier transform now has the whole real line as a support, so is not well localised in the frequency domain. In fact, we have a much more general statement as to the time and frequency spread of functions and their Fourier transforms.

Recall that for a function f with $\|f\|_{L^2} = 1$, its centre is defined as

$$\bar{f} = \int_{\mathbb{R}} t |f(t)|^2 dt \tag{1.3}$$

and its standard deviation σ_f to this centre is

$$\sigma_f^2 = \int_{\mathbb{R}} (t - \bar{f})^2 |f(t)|^2 dt. \tag{1.4}$$

Theorem 1. [Heisenberg Uncertainty Principle] For $f \in L^2(\mathbb{R})$, $\|f\|_{L^2} = 1$, the standard deviations $\sigma_f, \sigma_{\hat{f}}$ of f and its Fourier transform $\hat{f}$, respectively, satisfy

$$\sigma_f \sigma_{\hat{f}} \geq \frac{1}{2}. \tag{1.5}$$

Thus, any nonzero function f cannot be well localised in time will a Fourier transform $\hat{f}$ well localised in frequency. A proof of this standard result is in [15][pp. 30-33].

Knowing one pitfall of Fourier transforms, let us explore another option, which encapsulates both time and frequency information and which has been a major talking point all over science for the past few decades and a scientific revolution in its own right: wavelets.

2 The language of signals and filters

Before moving on to wavelets, here is a short presentation of terms coming from signal processing used throughout this discussion, as well as some intuition behind them. This will prove very useful as it is the most immediate application of Fourier analysis, and is a natural place to think about wavelets and their uses.

In signal processing, one deals with 2 objects: *signals* and *filters*. A *signal* is a given function – usually a discrete sampling – that can be an audio file, an image, etc. This signal can then be smoothed out by removing noise, or focused on its high frequency components, or compressed to be more manageable inside a computer. All of these operations use *filters*, which are operators on signals. Filters can get wild, but we will only use the simplest kind: linear time-invariant filters [13][Sec. 2.1].

For a signal $f \in L^2(\mathbb{R})$, a filter is simply an operator $\mathcal{L} : L^2(\mathbb{R}) \to L^2(\mathbb{R})$, taking $f \mapsto \mathcal{L}f$. In the case where $\mathcal{L}$ is linear and time-invariant – the latter meaning that a time delay in the input persists exactly in the output – filters are nothing but a convolution with their impulse response. More precisely if we consider a Dirac delta $\delta(t)$, the *impulse response* of the filter $\mathcal{L}$ is $h(t) = \mathcal{L}\delta(t)$, where δ is the Dirac distribution[1]. Then

$$\mathcal{L}f(t) = (f * h)(t) = \int_{-\infty}^{+\infty} f(u)h(t-u)du,$$

is a convolution. If one imposes the additional property that h is compactly supported, $\mathcal{L}$ is said to be a *Finite Impulse Response Filter* (FIR Filter). An computational advantage immediately becomes apparent: a finite integration is much easier than an integral over the whole real line.

Exactly the same can be said with linear time-invariant filters on discrete signals, expect that they are *discrete* convolutions with the impulse response $h[n]$, where the square brackets emphasise that h is now a discrete function. Similarly, if $h[n] = 0$ for all $|n| \geq N$, for some N, i.e. h has compact support, $\mathcal{L}$ is said to be a Finite Impulse Response Filter.

But to really get an intuition for what filters can do, one is compelled to go to frequency space. The reason for this is that the Fourier transform turns convolutions into multiplications, clarifying behaviour. Indeed, letting $\hat{f}, \hat{h}$ be the respective Fourier transforms of f and h, we have that

$$\mathcal{F}(f * h)(\xi) = \frac{1}{\sqrt{2\pi}}\hat{f}(\xi)\hat{h}(\xi),$$

so to understand what a filter $\mathcal{L}$ does to an $L^2(\mathbb{R})$ function, one can go to frequency space, multiply by $\hat{h}$ and come right back to the time-domain. The importance of this function gives it the name *transfer function*. We allow ourselves to shamelessly alternate between the time and frequency domain thanks to the Fast Fourier Transform (FFT) algorithm for discrete cases, a fast enough algorithm that going from one to the other is a reasonable computational ask.

Example 1. Consider $h(t) = \frac{\text{sinc}(\xi/2)}{\sqrt{2\pi}}$, which appeared previously in the Fourier transform of an indicator function. The same calculation shows $\hat{h}(\xi) = \chi_{[-1/2,1/2]}(t)$, the indicator function of the interval $[-1/2, 1/2]$. If we define a filter $\mathcal{L}$ to have impulse response h, we have

$$\mathcal{F}\left(\mathcal{L}(f)\right)(\xi) = \chi_{[-1/2,1/2]}(\xi)\hat{f}(\xi),$$

so $\mathcal{L}$ cuts off exactly any frequency greater than $1/2$ in absolute value. This is called a *low-pass filter*. General low-pass filters have $\hat{h}(0) \neq 0$ and $|\hat{h}(\xi)| \to 0$ as $|\xi| \to \infty$, meaning that the transfer function allows low frequencies to pass with a possible scaling, while getting rid of the higher ones.

Example 2. In the same vein, consider $g(t) = \frac{\text{sinc}(t/2)}{\sqrt{2\pi}}(e^{3it/2} + e^{-3it/2})$, which has Fourier transform $\hat{g} = \chi_{[-2,-1]} + \chi_{[1,2]}$. The corresponding filter therefore gets rid of any frequency outside of the 2 bands $[-2,-1], [1,2]$, giving the name *band-pass filter* to these kinds of operators. Band-pass filters have $\hat{g}(0) = 0$ as it is assumed the band does not include frequencies in a neighbourhood of 0, otherwise one might consider it a low-pass filter. Additionally, the transfer functions of band-pass filters also have $|\hat{g}(\xi)| \to 0$ as $|\xi| \to \infty$, to ensure they also have a high-frequency cut-off.

One can also consider other sorts of useful filters, as seen in [17][Sec. 1.3]. Note that both low-pass and band-pass filters can have finite energy (L^2-norm), whereas a high-pass filter (with transfer function $\chi_{[1,+\infty]}$ for example) would not.

[1] Technically an abuse of notation as $\delta \notin L^2(\mathbb{R})$.

3 Continuous Wavelet Transform

3.1 An Alternative to Fourier

In the literature, one can find 2 different approaches when introducing wavelets: those starting with the definition of the continuous wavelet transform [7, 13], and those starting with the definition of a multi-resolution approximation/analysis (MRA) [1, 15]. We will start with the former, which compares nicely with the Fourier transform, but we follow by a presentation of MRA's as they give the framework for discrete wavelets.

We will start by imposing loose, abstract requirements and retrieve the exact conditions needed for a function to be considered a wavelet. Consider a real function ψ, which we think of being well-localised in time and frequency, often called the "Mother wavelet". Now consider its "children wavelets"

$$\psi_{u,s}(t) = \frac{1}{\sqrt{s}}\psi\left(\frac{t-u}{s}\right) \tag{3.1}$$

for parameters $u \in \mathbb{R}, s > 0$. The translation parameter u moves $\psi_{u,s}$ to the right or left on the real line, and one should think of s as the scale parameter, controlling how concentrated $\psi_{u,s}$ is around its centre[2]. If ψ is centred around 0, then $\psi_{u,s}$ is centred around u. Note that $\|\psi_{u,s}\|_2 = \|\psi\|_2$ from the normalisation constant $1/\sqrt{s}$. The $\psi_{u,s}$ have Fourier transform

$$\hat{\psi}_{u,s}(\xi) = \sqrt{s}e^{-iu\xi}\hat{\psi}(s\xi), \tag{3.2}$$

where having a small parameter s yields large frequencies ξ and vice versa. The scale parameter should therefore be thought of as a kind of inverse to the frequency. Now consider the Continuous Wavelet Transform (CWT) of f at parameters u, s:

$$\mathcal{W}f(u,s) = \int_{-\infty}^{+\infty} f(t)\psi_{u,s}(t)dt = \langle f, \psi_{u,s}\rangle. \tag{3.3}$$

Just as in (1.1), this is interpreted as how well $\psi_{u,s}$ describes the behaviour of f. Since $\psi_{u,s}$ is localised in time around u, this will be larger if f correlates with this time localisation at u. Similarly, one can use the standard result from Parseval $\langle f, \psi_{u,s}\rangle = \langle \hat{f}, \hat{\psi}_{u,s}\rangle$ to see that $\mathcal{W}f(u,s)$ will be larger if $\hat{f}$ correlates with the frequency localisation corresponding to $\hat{\psi}_{u,s}$.

The Fourier transform is completely invertible: we can recover any function f from $\hat{f}$ using the inverse transform (1.2). If the wavelet transform is to make any sense, we should be able to recover f from some integration over our parameters u and s:

$$f(t) = \frac{1}{C_\psi}\int_0^{+\infty}\int_{-\infty}^{+\infty} \mathcal{W}f(u,s)\psi_{u,s}(t)du\frac{ds}{s^2}. \tag{3.4}$$

This is exactly theorem 4.3 in [13], and let's justify a few things, working from left to right.

Firstly, the $1/C_\psi$ factor is just a constant we have to work out to make sure our inverse transform is normalised, exactly like our $1/\sqrt{2\pi}$ factor in (1.1).

Next, the bounds of integration: u goes through the whole real line while s stays non-negative. This choice becomes obvious once we think back to what these parameters are: u translates our wavelet and s changes its scale, contracting and expanding it. We therefore want to consider all possible translations, while negative scales don't make much sense. Recall that $\psi_{u,s}$ has a $1/\sqrt{s}$ factor, so we also need to be careful with $s = 0$. In fact, we really are dealing with an improper integral in both the $+\infty$ boundary as well as the 0 boundary.

For the integrand, recall that the Fourier transform (1.1) takes inner products with the eigenfunctions $e^{i\xi t}$, which form a basis of the relevant function space. Then the inverse Fourier transform (1.2) recovers the function from an integral of each inner product multiplied by its corresponding basis element. If ψ is chosen appropriately so that the $\psi_{u,s}$'s form a basis for our function space, each wavelet coefficient $\mathcal{W}f(u,s) =$

[2] A small parameter s corresponds to a large scale, and a large s corresponds to a small scale.

$\langle f, \psi_{u,s} \rangle$ gives us the component of f for basis vector $\psi_{u,s}$. To recover f, we then need to integrate all inner products of f multiplied by the corresponding basis function. This is then weighted by $1/s^2$, which accounts for the scaling constants $1/\sqrt{s}$ and ensures our integral converges.

We will not go through the proof, but in the last step one requires

$$\hat{f}(\xi) = \frac{\hat{f}(\xi)}{C_\psi} \int_0^{+\infty} \frac{|\hat{\psi}(\omega)|^2}{\omega} d\omega,$$

so the constant obviously has to be

$$C_\psi = \int_0^{+\infty} \frac{|\hat{\psi}(\omega)|^2}{\omega} d\omega. \tag{3.5}$$

This comes with the caveat of ensuring this is finite. The condition $C_\psi < +\infty$ is called the *wavelet admissibility condition*. It implies $\hat{\psi}(0) = 0$, that is $\int_{-\infty}^{+\infty} \psi(t)dt = 0$: it has zero-average. In the language of filters, one could interpret $\hat{\psi}$ as the transfer function of a band-pass filter. Moreover, if we split the integral into

$$C_\psi = \int_0^1 \frac{|\hat{\psi}(\omega)|^2}{\omega} d\omega + \int_1^{+\infty} \frac{|\hat{\psi}(\omega)|^2}{\omega} d\omega,$$

we see that $|\hat{\psi}(\omega)|^2$ decreases rapidly as ω decreases to 0 since the first integral is finite, and $|\hat{\psi}(\omega)|^2$ decays faster than $1/\omega$ at infinity to ensure the second is finite. $\hat{\psi}$ is therefore localised not too far from 0, with $\hat{\psi}(0) = 0$.

Finally observe, using the perspective that ψ is the impulse response of a band-pass filter, that (3.3) can be written as a convolution

$$\mathcal{W}f(u, s) = f * \bar{\psi}_s(u),$$

where $\bar{\psi}_s(t) = \psi(-t/s)$. We can then interpret the wavelet transform as a linear filter evaluated at the translation parameter. The point is that we can then consider $\bar{\psi}_s$, a scaled and reversed version of ψ, as the impulse response, and all the intuition with filters transfers over.

One additional condition we impose on our wavelets is the normalisation $\|\psi\|_2 = 1$. We can now have a better mental picture of how ψ behaves: for it to have zero-average while having non-zero finite energy, it needs to oscillate around the time axis, with sufficient decay at infinity. If we think back on our goal to localise both time and frequency, the oscillation captures the necessary frequency component while imposing finite energy ensures some time localisation.

Let us see 2 simple wavelets. The first is the simplest one, called the Haar wavelet, after Hungarian mathematician Alfréd Haar:

$$\psi_{\text{Haar}}(t) = \begin{cases} 1, & 0 \le t < 1/2, \\ -1, & 1/2 \le t < 1, \\ 0, & \text{otherwise}, \end{cases} \tag{3.6}$$

plotted in Figure 1a. This has the clear advantage of being a sum of indicator functions, and all the calculations become trivial. For example, [18] explains the ideas with the Haar wavelet first, then moves on to the general case. One can easily compute the Fourier transform to be

$$\hat{\psi}_{\text{Haar}}(\xi) = \frac{1}{\sqrt{8\pi}} \operatorname{sinc}(\xi/4)(e^{i\xi/4} - e^{3i\xi/4}),$$

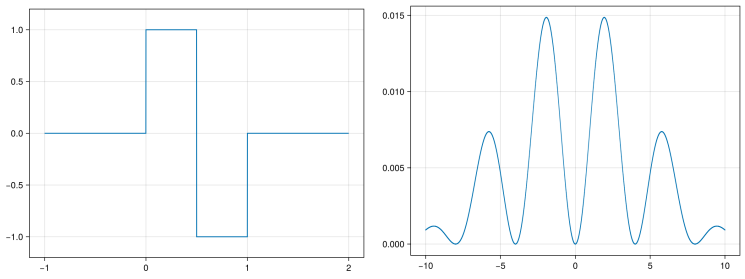
plotted in Figure 1b.

The other wavelet we will mention here is the second derivative of a Gaussian with standard deviation 1, called a Mexican hat wavelet – or sombrero – wavelet. We have

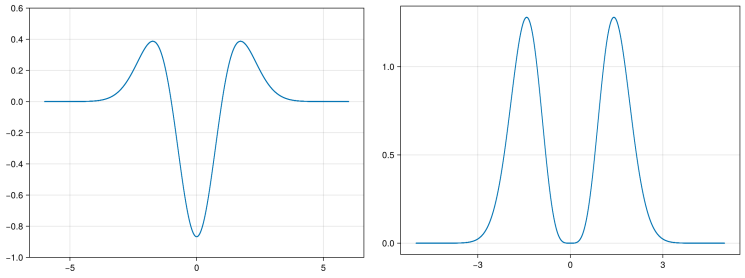
$$\psi_{\text{mex}}(t) = \frac{2}{\pi^{1/4}\sqrt{3}} \left(t^2 - 1\right) \exp\left(-t^2/2\right), \tag{3.7}$$

plotted in Figure 1c, with Fourier transform

$$\hat{\psi}_{\text{mex}}(\xi) = \frac{-2\pi^{1/4}}{\sqrt{3}} \xi^2 \exp\left(-\xi^2/2\right),$$



(a) Haar wavelet $\psi_{\text{Haar}}(t)$ (b) Fourier transform: $|\hat{\psi}_{\text{Haar}}(\xi)|^2$



(c) Mexican hat wavelet $\psi_{\text{mex}}(t)$ (d) Fourier transform: $|\hat{\psi}_{\text{mex}}(\xi)|^2$

Figure 1: Haar and Mexican hat wavelets with their Fourier transforms.

itself plotted in Figure 1d.

Another advantage, which could easily be taken for granted now, is that we have explicit forms for these wavelets. As we will see, wavelets are related to objects called conjugate mirror filters, and most of the analysis can be reduced to studying them. While these filters are freely and easily available online, they do not give a closed form for the wavelet function.

3.2 Visualisation

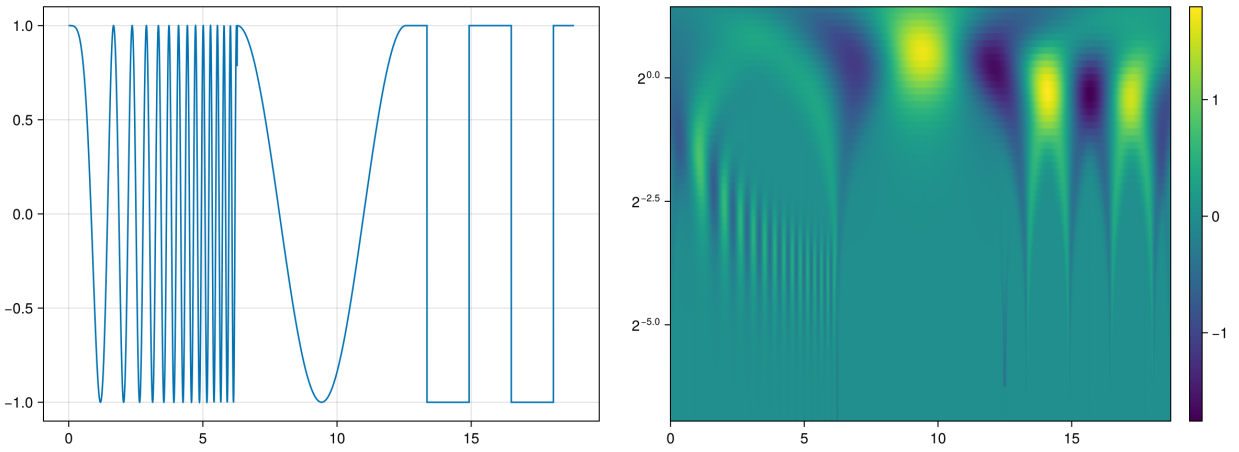
Visualising Fourier transforms is as easy as graphing the original function, since the Fourier transform outputs an object of the same dimension as the input. However, wavelet transforms of 1-dimensional functions give 2-dimensional families of coefficients, indexed by a pair (u, s), and k-dimensional functions have a $2k$-dimensional wavelet transform. This is one of the major downsides of wavelet analysis, as doubling the number of variables brings us closer to the curse of dimensionality. We plot the 2-dimensional CWT of a 1-dimensional function using a heat map and using the implementation in cwt.jl [16]. The code for plotting all the figures are available in figures.jl, if one wants to play around.

Figure 2 shows a function f with 3 different parts, as well as its CWT with the Mexican hat wavelet:

- for $0 \leq t < 2\pi$, $f(t) = \cos(2.25t^2)$ is a function with a linearly increasing frequency. This frequency change can be seen in the bottom left corner of Figure 2b;
- for $2\pi \leq t < 4\pi$, $f(t) = \cos(t)$ is a simple cosine wave, with non-negligible wavelet coefficients only for large scales/low frequencies;
- for $4\pi \leq t < 6\pi$, $f(t)$ is a blocky function, with a spike in the CWT corresponding to each edge. This spike phenomenon also appears at $t = 2\pi$, although somewhat toned down because f is not differentiable but is still continuous.

3.3 Heisenberg Uncertainty for wavelets

We have a transform that allows us to localise both in time and in frequency, but if you recall Heisenberg's Uncertainty Principle 1, this should be impossible.



(a) Test function f,

(b) CWT of f,

Figure 2: A test function and its Continuous Wavelet Transform, plotted as a heat map.

The problem with the above statement is our imprecise use of "localised": while wavelets do have a certain degree of both time and frequency localisation, this is still limited by the Uncertainty Principle. Interestingly, one can show that wavelets satisfy a worse inequality [2][Thm. 1.3][3]

$$\sigma_f \sigma_{\hat{f}} \geq \frac{3}{2}.$$

Having both time and frequency localisation comes at the cost of imperfect resolution in both worlds. A good way of understanding this property is graphing *Heisenberg boxes*. For a general L^2 function f, one can plot its centre c_f and standard deviation σ_f on the time axis, as well as the centre $c_{\hat{f}}$ and standard deviation $\sigma_{\hat{f}}$ of its Fourier transform on the frequency axis. The result is a Heisenberg box of width $2\sigma_f$ and height $2\sigma_{\hat{f}}$ centred at $(c_f, c_{\hat{f}})$ in the (t,ξ)-plane, which is understood as the localisation of f and $\hat{f}$, see Figure 3a.

Just as in [10][30:46], we first consider the 2 extremes in uncertainty: Dirac deltas, which have perfect time localisation but infinitely bad frequency localisation, and one-frequency trigonometric functions which have perfect frequency localisation but infinitely bad time localisation.

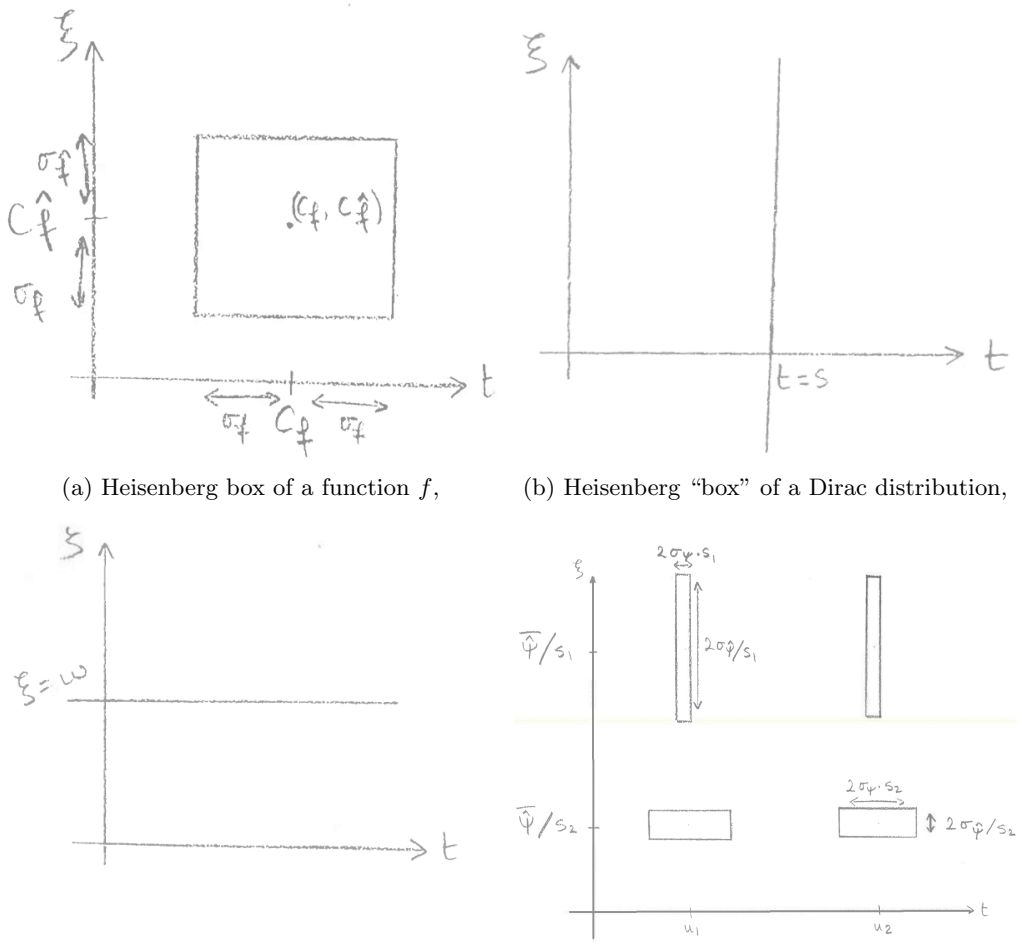
Define $\delta_s(t) = \delta(t-s)$ for some $s \in \mathbb{R}$. We have $c_{\delta_s} = s$ and $\sigma_{\delta_s} = 0$, as should be expected since δ_s is exactly localised at $t = s$. It has Fourier transform $\hat{\delta}_s(\xi) = \frac{1}{\sqrt{2\pi}} e^{-i\xi s}$, for which the centre $c_{\hat{\delta}_s}$ is undefined and for any arbitrary choice of centre, the standard deviation $\sigma_{\hat{\delta}_s}$ is infinite, again to be expected from previous observations. The corresponding Heisenberg 'box' is therefore the line $t = s$, as shown in Figure 3b. The other extreme is the trigonometric function $f_\omega(t) = e^{i\omega t}$, which has Fourier transform $\hat{f}_\omega(\xi) = \frac{1}{\sqrt{2\pi}}\delta_\omega(\xi)$. With the same reasoning as above, the corresponding Heisenberg 'box' here is the line $\xi = \omega$, as shown in Figure 3c. These pictures show how natural it is to think of these two kinds of functions as the extremes, and the functions we want to consider would ideally be exactly in the middle, attaining the lower bound of the Uncertainty Principle.

Let us now consider Heisenberg boxes for wavelets, more specifically how changing the parameters u, s change their location and shape. The child wavelet $\psi_{u,s}$ has centre $u + \bar{\psi}$ and standard deviation $\sigma(\psi_{u,s}) = s\sigma_\psi$. Recall that the $\psi_{u,s}$ have Fourier transform

$$\hat{\psi}_{u,s}(\xi) = \sqrt{s} e^{-iu\xi} \hat{\psi}(s\xi),$$

having centre $\overline{\hat{\psi}}/s$ and standard deviation $\sigma(\hat{\psi}_{u,s}) = \sigma_{\hat{\psi}}/s$. The corresponding Heisenberg boxes will be centred at $(u+\overline{\psi}, \overline{\hat{\psi}})$ with side length $s\sigma_\psi$ in the t-direction and side length $\sigma_{\hat{\psi}}/s$ in the ξ-direction, as shown

[3]This is a quantum mechanics paper, in which wavelets have very different meaning, but this is part of the fun in the history of the development of wavelets: it melded different communities together that would not usually interact, "people [...] found themselves outside of their usual universe" as Alex Grossman says in [5].



(a) Heisenberg box of a function f,

(b) Heisenberg "box" of a Dirac distribution,

(c) Heisenberg "box" of $e^{i\omega t}$,

(d) Heisenberg boxes of wavelets $\psi_{u,s}$ at $u_1 < u_2$ and $s_1 < s_2$,

Figure 3: Different Heisenberg boxes for different relevant functions.

in Figure 3d. While the boxes get squished and stretched, with changing time and frequency resolution by changing s, they maintain a constant area $\sigma_\psi \sigma_{\hat{\psi}}$: they all have the same overall uncertainty. The translation parameter u does not affect the shape, only the time location of the boxes. For small scales s – large frequencies – the standard deviation in frequency is large while the standard deviation in time is small. An equivalent formulation of this is that frequency resolution decreases as the frequency increases. Seeing uncertainty as lack of resolution becomes even more natural when working with applications to signal or image processing, where high uncertainty reduces the precision of numerical approximations.

4 Discrete Wavelet Transform

4.1 Discretising Wavelets

Just as the Fourier transform has a discrete counterpart that only considers a discrete set of frequencies, we can take children wavelets with a discrete set of (u, s) parameters. Just as in [7][p. 53], we naturally discretise the scale parameter taking $s = s_0^m$ for some fixed base scale $s_0 > 1$ and $m \in \mathbb{Z}$. For scale $s = 1$, $m = 0$, another natural choice is to consider translations by integer multiples of a base translation step u_0, that is $u = nu_0$ for $n \in \mathbb{Z}$. We would like to choose u_0 such that the translated wavelets' supports somehow fill the whole real line, we want to cover any function with our discrete collection of wavelets. We will need to adjust the translation step for different scales s_0^m to ensure that we have a similar spacing with the supports, so we

choose a dilated translation $u = nu_0 s_0^m$. This gives us discrete children wavelets[4]

$$\psi_{m,n}(t) = s_0^{-m/2}\psi\left(\frac{t - nu_0 s_0^m}{s_0^m}\right) = s_0^{-m/2}\psi(s_0^{-m}t - nu_0). \tag{4.1}$$

The discrete wavelet transform of a function f is then the collection of inner products $\langle f, \psi_{m,n}\rangle$, just as in (3.3). But just as we had the reconstruction formula (3.4) for the continuous transform, an immediate question is how do we recover f from the wavelet coefficients $\langle f, \psi_{m,n}\rangle$?

To fully answer this question, one is taken to the functional analytic theory of frames, a generalisation of bases. [7][Chap. 3] contains a primer in this theory to apply it to wavelets. In broad terms, a frame is a collection of functions behaving like a basis with potential redundancies. These redundancies are encoded in the so-called *frame bounds* $0 < A \leq B < \infty$, where

$$A\|f\|^2 \leq \sum_{m,n} |\langle f, \psi_{m,n}\rangle|^2 \leq B\|f\|^2,$$

and the frame is an orthonormal basis if and only if $A = B = 1$. The intuition is that we can reconstruct f from the wavelet coefficients if our wavelet collection $\psi_{m,n}$ forms a frame. The following result, Theorem 3.3.1 in [7], gives us a necessary condition for that to hold.

Theorem 2. If the $\psi_{m,n}(t) = s_0^{-m/2}\psi(s_0^{-m}t - nu_0)$, $m, n \in \mathbb{Z}$ form a frame for $L^2(\mathbb{R})$ with frame bounds A, B, then

$$\frac{u_0 \log s_0}{2\pi} A \leq \int_0^\infty \frac{|\hat{\psi}(\xi)|^2}{\xi} d\xi \leq \frac{u_0 \log s_0}{2\pi} B.$$

Comparing this discrete admissibility condition to its continuous version, one sees that $B < \infty$ implies the admissibility condition (3.5). While having redundancies can be useful in applications where precision is valued over efficiency, we will focus on orthonormal bases of wavelets, that is every cross inner product is 0, giving no redundancies. Taking $A = B = 1$ in the result above, we therefore require

$$\frac{u_0 \log s_0}{2\pi} = \int_0^\infty \frac{|\hat{\psi}(\xi)|^2}{\xi} d\xi,$$

a relation between our mother wavelet and the choices of base scale and translation step. We will consider $s_0 = 2$ in the following, a choice made natural by the dyadic structure of frequencies, e.g. doubling the frequency of a note gives a higher octave version of the same note. Then the base translation step is entirely determined by the choice of wavelet. Or looking at it the other way, one could ensure $u_0 = 1$ by modifying the initial choice of mother wavelet. The orthogonal wavelets we consider will therefore be of the form

$$\psi_{m,n}(t) = 2^{-m/2}\psi(2^{-m}t - n). \tag{4.2}$$

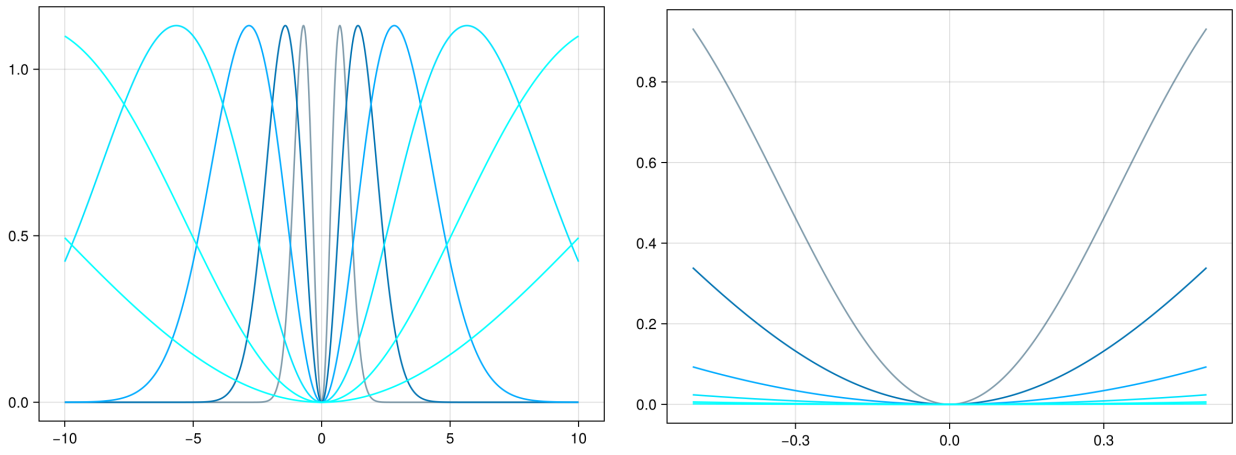
4.2 Scaling function

When numerically calculating the continuous wavelet transform, one quickly realises the discrete limits of a computer, and we used a slightly different discretisation to produce Figure 2, with a translation step independent of the scale parameter. But this was used as an aid to enhance the intuition we have about how the wavelet transform worked, not as a tool to process data. In particular we were not worrying with reconstructing the function f from the wavelet coefficients, see discussion in Appendix A.

We now set ourselves the following problem: given a sampled function f, can we numerically take a discrete wavelet transform, and exactly recover f with some inverse transform?

The first issue is that a wavelet transform requires integration, and varying the 2 parameters m, n gives us quite a few numerical integrals, a heavy task. This will be solved using *conjugate mirror filters*, to be introduced in the following section. These remove all integrals, as well as even removing the need to explicitly have the wavelet. With these, we can apply a wavelet transform without knowing what the wavelet even looks like!

[4]Note an slight inconsistency with notation here, in the continuous case the index of the child wavelet started with the translation parameter, followed by the scale parameter. This is reversed here to stay consistent with the literature.



(a) Multiple Fourier Transforms of $\hat{\psi}_{\text{mex}}$,

(b) Zooming into 0,

Figure 4: $-\hat{\psi}_{m,0}$ for the Mexican hat wavelet ψ_{mex} as in (3.7), for $-4 \leq m \leq 1$.

The second issue is limiting ourselves to finitely many scales. If the function is well-behaved and smooth enough, the small scales/high frequencies will not be too relevant and truncation to fine enough scales is sufficient, but large scales/low frequencies will be problematic. Or seen another way, taking a finite sum of wavelets, all with zero average, will return a function that also averages to zero, unlike some functions we want to approximate.

Recall the filter terminology, and that ψ can be interpreted as the impulse response of a band-pass filter, say with $\operatorname{supp}\hat{\psi} \subset [-\Omega_2, -\Omega_1] \cup [\Omega_1, \Omega_2]$, $0 < \Omega_1 < \Omega_2 < \infty$ for simplicity. Modifying the scale parameter m will squeeze and stretch the Fourier transform $\hat{\psi}_{m,n}$:

$$\operatorname{supp}\hat{\psi}_{m,n} = [-2^{-m}\Omega_2, -2^{-m}\Omega_1] \cup [2^{-m}\Omega_1, 2^{-m}\Omega_2].$$

Suppose that $\operatorname{supp}\hat{f} \subset [-K, K]$, then we only need to take m small enough that $K \leq 2^{-m}\Omega_2$ to ensure we can reach the highest frequencies of f. But we have no way of reaching 0 as $0 \notin \operatorname{supp}\hat{\psi}_{m,n}$ for all m. As $m \to \infty$, the bands get arbitrarily close to 0 without ever reaching it, as shown in Figure 4. This is where we need to introduce a scaling function φ to take care of these low frequencies/large scales.

Like in [13], let us define the *scaling function* by the modulus of its Fourier transform, as "an aggregation of wavelets at scales larger than 1":

$$|\hat{\varphi}(\omega)|^2 = \int_1^{+\infty} |\hat{\psi}(s\omega)|^2 \frac{ds}{s} = \int_\omega^{+\infty} \frac{|\hat{\psi}(\xi)|^2}{\xi} d\xi, \tag{4.3}$$

with an arbitrary choice of phase. Then $|\hat{\varphi}(0)|^2 = C_\psi \neq 0$, so φ can be interpreted as the impulse response of a low-pass filter, or equivalently $\hat{\varphi}$ can be interpreted as the transfer function of said low-pass filter. In particular it will be able to handle the low frequencies that the $\psi_{m,n}$ did not handle. This is sometimes called the 'father wavelet', but since the scaling function is entirely determined by the mother wavelet, and vice versa as we will see shortly, Strichartz [18] says "this shows a scandalous misunderstanding of human reproduction; in fact the generation of wavelets more closely resembles the reproductive life style of an amoeba". The function φ satisfies a so-called scaling or refinement equation, hence the name. An easier way of seeing this goes through multi-resolution approximations (MRA's), which we briefly present, but [14] gives much more detail.

4.3 Multi-Resolution Approximations

We told the story starting from wavelets, but we can also start from a scaling function and obtain the corresponding wavelets.

Start with a function φ, which we will think of as the impulse response of a low-pass filter, localised in space near the origin and with unit norm $\|\varphi\|_{L^2} = 1$. The canonical example of this is the Haar scaling function $\varphi(t) = \chi_{[0,1]}(t)$, which is good to keep in mind when going through this section. As mentioned previously, [18] contains a version of this explanation entirely using the Haar scaling function, since one can easily compute things in this case. We want to use this function to find approximations to $L^2(\mathbb{R})$ functions: for a given function $f \in L^2(\mathbb{R})$, the inner product $\langle f, \varphi\rangle = \langle \hat{f}, \hat{\varphi}\rangle$ will eliminate the large frequencies of f, essentially giving us a low-frequency approximation of f when considering

$$\langle f, \varphi\rangle\varphi.$$

Note that φ is localised around 0, so the above approximation is only valid in a neighbourhood of 0. To ensure we can approximate functions localised elsewhere, we should consider translations $\varphi_n(t) = \varphi(t-n)$ for $n \in \mathbb{Z}$ and the corresponding inner products:

$$\sum_{n\in\mathbb{Z}} \langle f, \varphi_n\rangle\varphi_n$$

can be thought of as an approximation of f. Recall that the inner product shown really is a convolution $f * \bar{\varphi}(n)$, where $\bar{\varphi}(t) = \varphi(-t)$, so the intuition for filters really does carry over.

But this gives us a very limited picture of what f is, indeed if the support of $\hat{\varphi}$ is $[-\Omega, \Omega]$, then any information that $\hat{f}$ contains outside of that region is lost. A solution to this is to consider re-scaling the functions φ_n by taking

$$\varphi_{m,n}(t) = 2^{-m/2}\varphi(2^{-m}t - n).$$

As seen previously for wavelets, this will modify the support of the Fourier transform to

$$\operatorname{supp}\hat{\varphi}_{m,n} = [-2^{-m}\Omega, 2^{-m}\Omega],$$

so we can keep decreasing m to ensure all of f's interesting high frequency behaviour is taken into account. The mental image you should have is that decreasing m compresses φ in the time domain, so we can get more precise understanding of the small details of f. We now have a sequence of functions with which we can approximate any $f \in L^2(\mathbb{R})$ for large negative m. Most functions f will require infinitely many $\varphi_{m,n}$ in their expansion, as usual in functional analysis.

We require one final important condition for φ to properly be called a scaling function, which is, of course, a scaling condition: we want $\varphi_{1,0}(t) = \frac{1}{\sqrt{2}}\varphi(t/2)$ to be exactly expressible in terms of the $\varphi_{0,n}(t) = \varphi(t-n)$, without needing any other scales. Mathematically this means

$$\frac{1}{\sqrt{2}}\varphi\left(\frac{t}{2}\right) = \sum_{n\in\mathbb{Z}} h[n]\varphi(t-n), \tag{4.4}$$

where the coefficients $h[n]$ are the inner products

$$h[n] = \left\langle \frac{1}{\sqrt{2}}\varphi\left(\frac{t}{2}\right), \varphi(t-n)\right\rangle, \tag{4.5}$$

and h is interpreted as a discrete filter. One can rewrite this in functional analysis language

$$\varphi_{1,0} \in \overline{\operatorname{span}\{\varphi_{0,n}\}_{n\in\mathbb{Z}}},$$

and in fact we have the much more general

$$\varphi_{m+j,0} \in \overline{\operatorname{span}\{\varphi_{m,n}\}_{n\in\mathbb{Z}}}$$

for any $j \geq 1$. The intuition is that increasing the scale parameter m makes our approximations coarser and coarser, so all the information that $\varphi_{m+j,0}$ can get must be retrievable only using the $\varphi_{m,n}, n \in \mathbb{Z}$. This can be shown quite easily for the Haar scaling function, e.g. finding $h[n]$ for Haar easily gives $h[0] = h[1] = 1/\sqrt{2}$.

An important result, Theorem 7.2 in [13], shows the equivalence of the scaling function φ and its associated filter h, given some technical conditions. The study of finding scaling functions therefore boils down to finding filters h satisfying certain properties.

Let us set $V_m = \operatorname{span}\{\varphi_{m,n}\}_{n\in\mathbb{Z}}$ to be the successive approximation spaces. We call the sequence $\{V_m\}_{m\in\mathbb{Z}}$ a *multi-resolution approximation (MRA)* if the following conditions hold [Definition 7.1 in [13]]:

1. $V_{m+1} \subset V_m$ for all $m \in \mathbb{Z}$,
2. $f(t) \in V_m \iff f(t/2) \in V_{m+1}$ for all $m \in \mathbb{Z}$,
3. $f(t) \in V_m \iff f(t - 2^m k) \in V_m$ for all $m, k \in \mathbb{Z}$,
4. $\bigcap_{-\infty}^{+\infty} V_m = \lim_{m \to +\infty} V_m = \{0\}$ and $\bigcup_{-\infty}^{+\infty} V_m = \lim_{m \to -\infty} V_m$ is dense in $L^2(\mathbb{R})$.

The first condition is exactly the discussion above, where coarser approximations spaces are included in the finer ones; condition 2 is a more general version of the scaling equation; condition 3 tells us about the translation invariance of the approximation spaces and what scaling is necessary for that translation; the last condition tells us that as the scale parameter goes to $+\infty$, the scale gets arbitrarily large so we retain no information and can only get the 0 function; while in the other direction, as the scale parameter goes to $-\infty$, the scale gets arbitrarily small so we perfectly retain all the information.

We want the $\varphi_{0,n}$ to all be linearly independent. The intuition is that φ is localised around 0, but the $\varphi_{0,n}$ are translations away from 0, so one can see it should be impossible to write φ as a finite linear combination of $\varphi_{0,n}$'s. $\varphi_{0,n}$ is therefore a basis of V_0, and in fact one can choose φ such that $\varphi_{0,n}$ is an orthonormal basis. The unit norm of each function easily comes from $\|\varphi\|_{L^2} = 1$, but the orthogonality does restrict the functions one can consider. Similarly, for a fixed scale parameter m, the $\{\varphi_{m,n}\}_{n \in \mathbb{Z}}$ form an orthonormal basis of the approximation space V_m, and the final condition for $\{V_m\}$ to be an MRA implies that the full collection $\{\varphi_{m,n}\}_{(m,n) \in \mathbb{Z}^2}$ spans the whole of L^2. It is important to note here that this collection will not be orthogonal: 2 scaling functions $\varphi_{m_1,n_1}, \varphi_{m_2,n_2}$ with different scale parameters $m_1 \neq m_2$ and overlapping supports will generally not be orthogonal[5].

A question should now arise, where are the wavelets? The above construction only mentions a scaling function, with no wavelet in sight. Let us think back on why we introduced scaling functions to begin with: we could not deal with arbitrarily low frequencies with finitely many wavelets, and the scaling function gave us those missing frequencies. We have a similar situation here: every time we decrease the scale parameter m to $m-1$, the transfer function $\hat{\varphi}_{m-1,n}$ of the associated low-pass filter doubles it support, so we see twice as many frequencies. But crucially, we also see the ones that $\hat{\varphi}_{m,n}$ already considered: in the interval $[-\Omega, \Omega]$, both $\hat{\varphi}_{m,n}$ and $\hat{\varphi}_{m-1,n}$ pick up the behaviour of $\hat{f}$. Wavelets are the functions that only see the interval $[-2\Omega, -\Omega] \cup [\Omega, 2\Omega]$, they encode the change one gets when going from one scale to the next.

In functional analytic language, note that $V_{m+1} \subsetneq V_m$ means that we can take the orthogonal complement W_{m+1} of V_{m+1} in V_m, the linear subspace $W_{m+1} \subset V_m$ such that $V_m = V_{m+1} \oplus W_{m+1}$. We can start at $m = 0$ and keep increasing it to see that

$$L^2(\mathbb{R}) = V_0 \oplus \bigoplus_{m=-\infty}^{m=0} W_m.$$

We can see this as V_0 being the lowest scale approximation space, and every W_m we add onto it increases how precise our approximation is. Note that by construction, all the W_m's are orthogonal, so the collection $\{W_m\}_{m \in \mathbb{Z}}$ forms an orthonormal basis of L^2, although only finitely many W_m's are available on a computer of course. If we find a function ψ that then generates W_0 by its integer translations, just like V_0 is generated by integer translates of φ, we will have found the corresponding wavelet!

In fact one can retrieve this wavelet using the filter h and an associated filter g, defined by

$$\hat{g}(\xi) = e^{i\xi} \hat{h}^*(\xi + \pi). \tag{4.6}$$

The filters h, g are called *conjugate mirror filters* because of the above relation, and one has

$$\hat{\psi}(\xi) = \frac{1}{\sqrt{2}} \hat{g}\left(\frac{\xi}{2}\right) \hat{\varphi}\left(\frac{\xi}{2}\right). \tag{4.7}$$

In particular, the wavelet ψ is entirely determined by the scaling function φ, so recalling that we obtained the scaling function from the wavelet in (4.3), ψ and φ are entirely equivalent.

[5] Try finding a counter-example with the Haar scaling function.

The equation (4.6) can be expressed in the (discrete) time domain to get

$$g[n] = (-1)^{1-n} h[1-n], \tag{4.8}$$

a much more computable formula if h is given. Another property of g – or depending on how one looks at it, its defining property – is

$$g[n] = \left\langle \frac{1}{\sqrt{2}} \psi\left(\frac{t}{2}\right), \varphi(t-n) \right\rangle, \tag{4.9}$$

so just as h relates a scaled version of φ to its integer translates, g relates a scaled version of ψ to integer translates of φ.

4.4 Fast Wavelet Transform

The wavelet and scaling function are entirely determined by 2 discrete filters h, g and we now describe how to use these filters in order to decompose the signal into wavelet coefficients and a coarser approximation. This algorithm is called a "pyramid scheme" in [8] and is the standard for computing discrete wavelet transforms. It is also known as the Fast Wavelet Transform (FWT), but a pyramid is a good mental image to have, as shown in Figure 5. Only $O(N)$ operations are required for a signal of length N, asymptotically faster than the $O(N \log N)$ complexity of the Fast Fourier Transform.

Consider the sequence of discrete signals $a_m[n] = \langle f, \varphi_{m,n} \rangle, d_m[n] = \langle f, \psi_{m,n} \rangle$: a_m is the approximation of the signal f at scale parameter m and d_m encodes the differences between successive scales, in particular the difference between scale parameters m and $m-1$. The d_m are referred to as the wavelet coefficients and the a_m as the scaling function or approximation coefficients. We introduce the notation $\overline{x}[n] = x[-n]$, flipping the sequence, and

$$\check{x}[n] = \begin{cases} x[p] & \text{if } n = 2p, \\ 0 & \text{if } n = 2p+1, \end{cases}$$

stretching the signal by a factor of 2 and adding 0's on the new indices, essentially creating holes in the signal. We can now write Theorem 7.7 in [13], which tells us how to go from a scale to the next coarsest one, and how to reconstruct the original approximation.

Theorem 3. One can decompose a_m into

$$a_{m+1}[n] = a_m \star \overline{h}[2n], \tag{4.10}$$

$$d_{m+1}[n] = a_m \star \overline{g}[2n]. \tag{4.11}$$

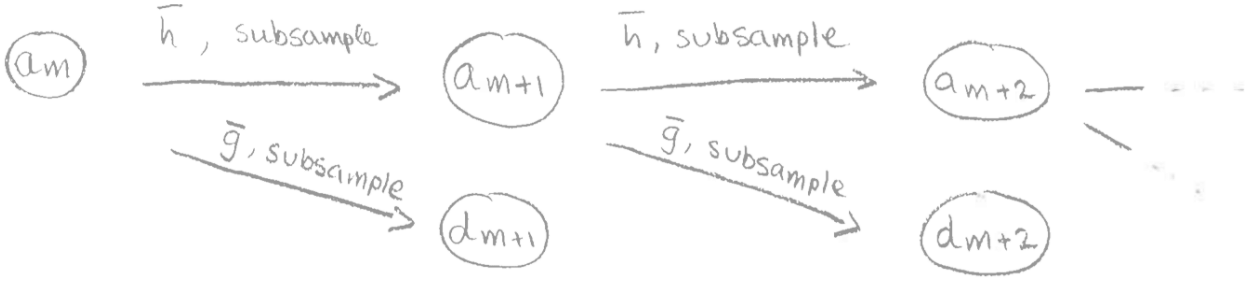
In the other direction, one can reconstruct a_m by

$$a_m[n] = \check{a}_{m+1} \star h[n] + \check{d}_{m+1} \star g[n]. \tag{4.12}$$

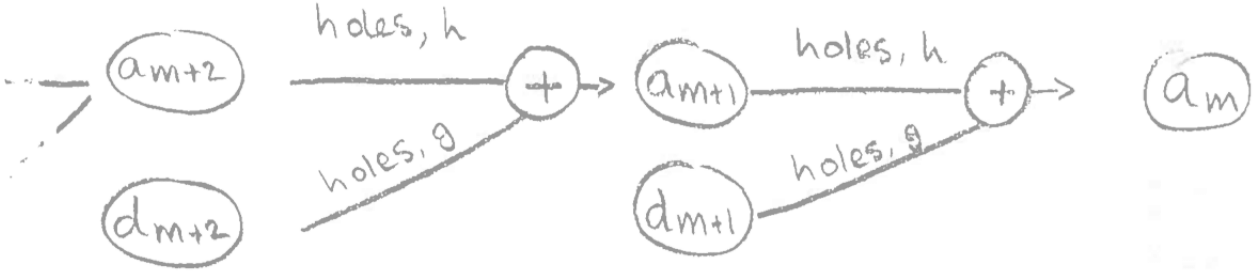
The proof of this result is quite simple as all of the heavy lifting has been done in establishing the relations between φ, ψ and their associated filters h, g. Once we have the next level a_{m+1} of approximations, we can decompose this again to get a_{m+2}, d_{m+2}, and so on and so forth. At each decomposition step, we obtain the next, coarser level of approximations, and the wavelet coefficients which tell us the change from one scale to the next. Applying the decomposition, say N times, gives you the wavelet coefficients $d_{m+1}, d_{m+2}, \ldots, d_{m+N}$ and the lowest scale approximation a_{m+N}. This algorithm is shown in Figure 5.

Observe that (4.10),(4.11) contain a subsampling by 2, which can be seen as an inverse to the check operator $x \mapsto \check{x}$. The intuition is that the approximation gets coarser at each increase of m, so the sample points spread out further. The subsampling exactly spreads the points by a factor of 2, just as expected with our dyadic scaling. Similarly at the reconstruction step, we need to create new sampling points in between the pre-existing ones, hence the creation of holes using the check operator.

This algorithm is $O(N)$ [8][p. 152], where the constant in front of N depends linearly on the length of the filters h, g, that is how many non-zero elements they have. As previously mentioned, this is asymptotically faster than the FFT that requires $O(N \log N)$ operations. The FFT's complexity contains a logarithm from its divide and conquer approach, that is the repeated splitting into 2 smaller similar calculations. But this



(a) Sketch of the 'pyramid', a filtering with $\overline{h}$ followed by a subsampling gives the next coarsest approximation coefficients, while a filtering with $\overline{g}$ followed by a subsampling gives the next coarsest wavelet coefficients;



(b) Sketch of the reconstruction, introducing holes into the coarser approximation and wavelet coefficients, applying the filter h, g, respectively and adding the results;

Figure 5: Structure of the FWT and iFWT algorithms, as in Figures 5 and 7 of [12].

splitting into 2 is exactly how the orthonormal wavelet transform is constructed[6], so we get rid of this factor of $\log N$ for free. Also note that this comparison between the FFT and the FWT doesn't mean wavelets are strictly better than the Fourier transform, indeed the FWT produces an array of twice the dimension we started with whereas the FFT gives the same dimension. Moreover, in this 1-dimensional case, the FFT outputs a simple vector array, while the data structure of the FWT's output is a collection of arrays of coefficients, a slightly stranger object to handle that calls for some adjustments in the treatment of data[7].

4.5 Compact support and vanishing moments

Now that we can obtain wavelet coefficients in reasonable time, let us discuss the properties on wavelets that make this algorithm customisable. As mentioned above, the total number of operations done in the FWT is proportional to the length of the filter h, and recalling the formula (4.5), one can find an upper bound for this using the length of φ's support. It is therefore natural to want our scaling functions, and in turn our wavelets, to be compactly supported with the smallest support possible.

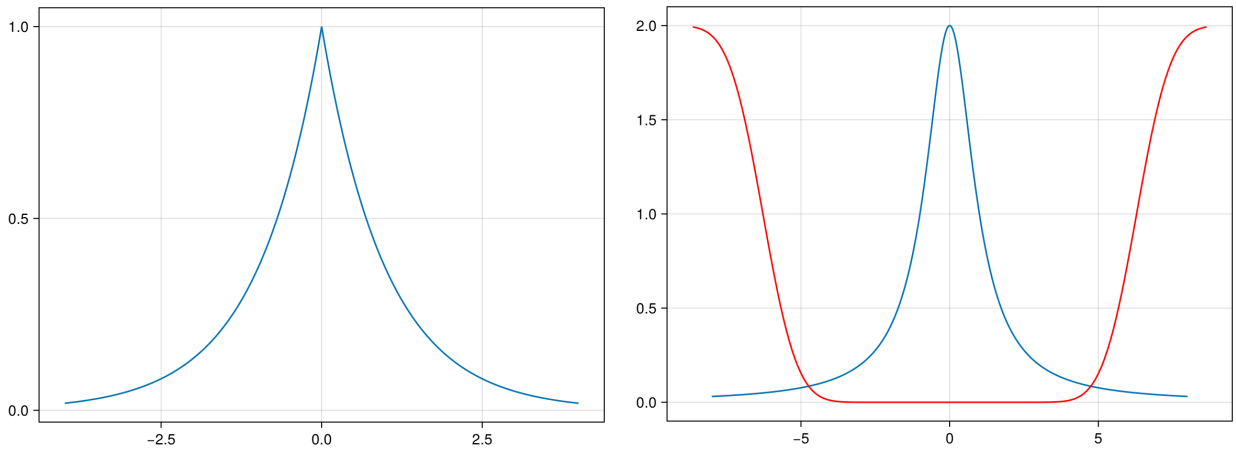
The most useful property of wavelet coefficients is that they provide a *sparse* representation: most of the wavelet coefficients will be small enough that one can simply consider them to be 0, saving lots of storage space. To this effect, a condition to impose on wavelets is for them to have many vanishing moments at 0, that is

$$\int_{-\infty}^{+\infty} t^k \psi(t) dt = 0, \text{for } 1 \leq k \leq p. \tag{4.13}$$

Note that $k = 0$ is simply the zero-mean condition we already imposed on wavelets. If the analysed signal is smooth, small scale wavelet coefficients will be negligibly small, hence the sparse representation. We present 2 explanations of why: one using the implications for the Fourier transform $\hat{\psi}$, and the other expanding the continuous wavelet transform as a Taylor series.

[6] Our choice of $s_0 = 2$

[7] One could store this more simply with a matrix, but as we will see later, we need to consider negative indices, which doesn't quite work for matrices in Julia. Moreover, same column entries generally do not correspond to the same sample point in time, a possible source of confusing that could justify keeping arrays separate.



(a) $f(t) = e^{-|t|}$,

(b) $\hat{f}(\xi)$ with $|\hat{\psi}_{u,s}|^2$ for Db10,

Figure 6: Inner products $\langle \hat{f}, \hat{\psi}_{m,n} \rangle$ are small for small scales.

First observe from Parseval's identity

$$\langle f, \psi_{m,n} \rangle = \langle \hat{f}, \hat{\psi}_{m,n} \rangle$$

that the wavelet coefficient $\langle f, \psi_{m,n} \rangle$ can be obtained entirely from the Fourier domain, so let us shift our perspective there. Recall Figures 1b,1d, motivating how one could interpret the wavelet ψ as the impulse response of a band-pass filter since $\hat{\psi}(0) = 0$. Taking the Fourier transform in (4.13), this is equivalent to $\hat{\psi}$ having the first p derivatives at the origin being 0, so the more vanishing moments ψ has, the flatter $\hat{\psi}$ is at 0. Also recall from (3.2) that increasing the scale parameter m contracts the Fourier Transform $\hat{\psi}_{m,n}$, while decreasing m stretches $\hat{\psi}_{m,n}$. For large negative m and for a wavelet with many vanishing moments, $\hat{\psi}_{m,n}$ will therefore be very close to 0 on a large neighbourhood of 0.

On the other hand, a function's smoothness translates to decay in the Fourier domain: if $f \in C^r$, then its Fourier transform $\hat{f}$ will decay like $1/|\xi|^r$ at infinity. In particular $\hat{f}$ will have an bounded effective support[8], and smoother f gives smaller effective support of $\hat{f}$.

Putting these 2 observations along with Parseval's identity shows that the inner product in the Fourier domain will be small for small scales, since $\hat{f}$ will take negligibly small values outside of a bounded interval, and with m negative and large, $\hat{\psi}_{m,n}$ will be essentially 0 on that interval, as shown in Figure 6b.

We now show an intuition for the use of vanishing moments using Taylor expansions, for which we will go back to the continuous wavelet transform for a bit. Consider some $f \in L^2(\mathbb{R})$, which we assume has at least p derivatives, its Taylor expansion about 0 is

$$f(t) = \sum_{k=0} f^{(k)}(0) \frac{t^k}{k!} + O(t^{p+1})$$

and substitute it into (3.3) with $u = 0$ to get

$$\mathcal{W}f(0,s) = \sum_{k=0}^{p} \frac{f^{(k)}(0)}{k!} \int_{-\infty}^{+\infty} t^k \psi\left(\frac{t}{s}\right) dt + \int_{-\infty}^{+\infty} O(t^{p+1}) \psi\left(\frac{t}{s}\right) dt.$$

The sum contains scaled k-th moments of ψ, so a wavelet with p vanishing moments will only have the error term, which can be rescaled to

$$\int_{-\infty}^{+\infty} O((ts)^p t) \psi(t) dt.$$

[8]The effective support of a function is the set where the function has absolute value greater than some threshold tolerance: $\{\xi \in \mathbb{R} : |\hat{f}(\xi)| > \varepsilon\}$ for some small $\varepsilon > 0$.

Note that ψ has finite L^2-norm and must have some decay at infinity and hence a bounded effective support; for small s, the integrand will be small on said effective support, implying the integral will be small, and hence and the whole wavelet coefficient will be small.

A proper treatment showing the wavelet coefficients decay given enough regularity of the signal f is in [9], and a version as written in [13][Thm. 6.3] is presented below:

Theorem 4. If $f \in L^2(\mathbb{R})$ is Lipschitz $\alpha \leq n$ at v[9], then there exists an A such that

$$|\mathcal{W}f(u,s)| \leq As^{\alpha+1/2}\left(1+\left|\frac{u-v}{s}\right|^{\alpha}\right).$$

The "ideal" wavelet therefore has infinitely many vanishing moments and an arbitrarily small support. But just as the Heisenberg Uncertainty Principle 1 put a relation between the time-spread and frequency-spread of functions, effectively bounding them, Daubechies [6] proved that a wavelet with p vanishing moments must have a support with width at least $2p-1$. This proof focuses on the corresponding filter h and the fact that a wavelet with p vanishing moments will have $\hat{h}$ with a zero of multiplicity p at $\xi = \pi$, and one has to work around this constraint to still obtain a valid filter h. Daubechies also found the filters and wavelets where these bounds are attained, now called Daubechies wavelets, and some are shown in Figure 7. The Daubechies wavelet with p vanishing moments is denoted Dbp. Note that Db1 is the discontinuous Haar wavelet, shown previously in Figure 1a.

One option with infinitely many vanishing moments, which must have unbounded support by Daubechies' result, is the Shannon wavelet. This is

$$\psi_{\text{Sha}}(t) = 2\,\text{sinc}(2t) - \text{sinc}(t), \tag{4.14}$$

with associated scaling function

$$\varphi_{\text{Sha}}(t) = \text{sinc}(t). \tag{4.15}$$

One can easily calculate the Fourier transform

$$\hat{\psi}_{\text{Sha}}(\xi) = \begin{cases} \sqrt{\pi/2}, & 1 \leq |\xi| \leq 2, \\ 0, & \text{otherwise}, \end{cases}$$

which is constant in a neighbourhood of 0, hence one can see that ψ has infinitely many vanishing moments. This comes at the cost of ψ having unbounded support and terrible decay, indeed $|\psi(t)| \sim 1/t$, making this a bad numerical choice.

5 Coding and applications

5.1 Implementations of the FWT

Note that we apply the FWT algorithm to infinite objects, indeed the sequences $a_m[n], d_m[n]$ really are defined for $n \in \mathbb{Z}$. Note that finite signals can be trivially extended to infinite signals by setting all other samples to 0: for a finite signal $f[n], N_1 \leq n \leq N_2$, define its infinite extension by

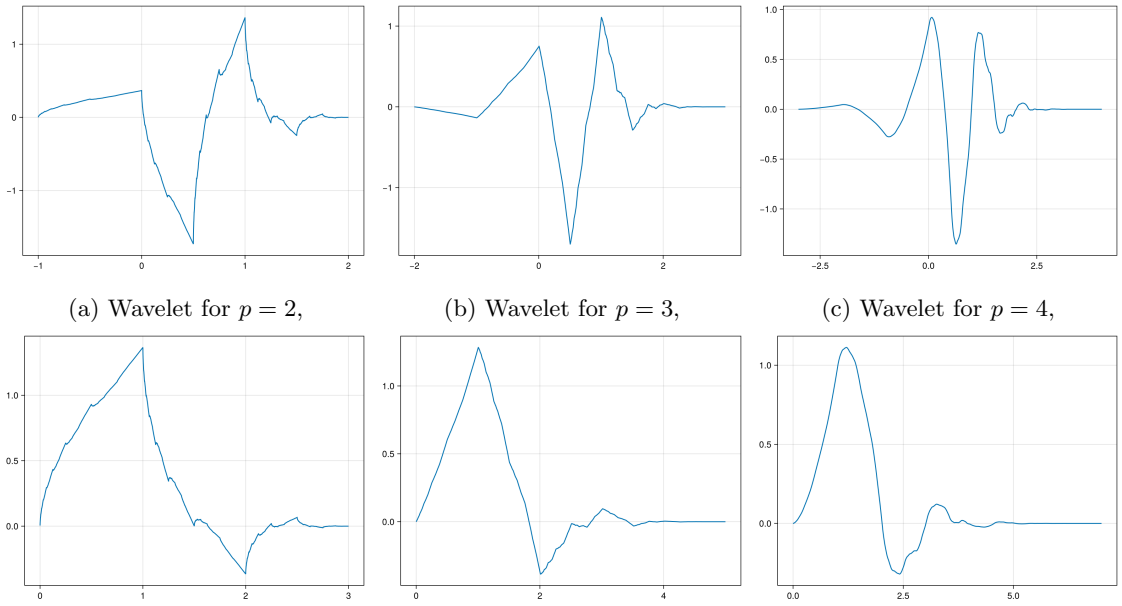
$$\bar{f}[n] = \begin{cases} f[n], \text{ if } N_1 \leq n \leq N_2, \\ 0, \text{ otherwise.} \end{cases}$$

We say the support of f or $\bar{f}$ is $[N_1, N_2]$[10]. This extension is a pedantic step, but it is needed to ensure mathematical correctness of the numerical algorithm.

All of the code, available on my GitHub [16], is written in julia, with the disadvantage of not allowing negative indices. Our consideration of general signals with support $[N_1, N_2]$, which could be anywhere on the number line means we have to consider sequences starting at any integer. Coding wise, we need

[9] A notion of regularity that the function is well estimated by polynomials of degree at most n near v.

[10] Really the support of these discrete functions is the interval's intersection with $\mathbb{Z}$.



(a) Wavelet for $p = 2$, (b) Wavelet for $p = 3$, (c) Wavelet for $p = 4$,

(d) Scaling function for $p = 2$, (e) Scaling function for $p = 3$, (f) Scaling function for $p = 4$,

Figure 7: Daubechies wavelets and their scaling function, for p vanishing moments.

to ensure we can have arrays with "indices" in all of $\mathbb{Z}$, including negative ones. This was hard-coded in signedArray.jl, where all of the operations required in the FWT were implemented. A signal f with support $[N_1, N_2]$ is represented by a `Tuple{Vector{Float64}, Int64}`, where the vector contains the entries $f[N_1], f[N_1 + 1], \ldots, f[N_2]$ and the integer is the start of the support N_1, understood as the integer needed to shift the indices to 0. More precisely, we represent f with `(array,N1)` for an array of length $N_2 - N_1 + 1$ such that

$$\texttt{array[n]} = f[n + N_1 - 1]. \tag{5.1}$$

We can now use fwt.jl to compute the Fast Wavelet Transform of signals. Note that the inputs $\texttt{a}_m\texttt{,d}_m\texttt{,h}$ are all signedArrays, and a collection of different filters taken from `https://wavelets.pybytes.com` is added in filters.jl. The output is an array of signedArrays, the first N being the wavelet coefficients $d_{m+1}, \ldots, d_{m+N}$'s and the last one being the approximation coefficients a_{m+N} at the lowest scale. The function `plotfwt` plots all of these signedArrays in that order.

A more pedantic note is that the starting point of the algorithm is $a_m = \langle f, \varphi_{m,n} \rangle$, but our samples might not be exactly in this form. Indeed we will start with samples $f(n \cdot \Delta t)$, Δt being the sampling distance, so implicitly we set

$$a_m[n] = f(n \cdot \Delta t),$$

but to properly apply our theory we should find a function $\tilde{f}$ such that

$$f(n \cdot \Delta t) = a_m[n] = \langle \tilde{f}, \varphi_{m,n} \rangle.$$

Let us think back on how scaling functions are interpreted in the context of Multi-Resolution Approximations, as approximation functions at a given scale. We also saw that these inner products can be rewritten as evaluating a convolution:

$$f(n \cdot \Delta t) = \tilde{f} * \bar{\varphi}_m(2^m n),$$

where $\bar{\varphi}_m(t) = 2^{-m/2}\varphi(-2^{-m}t)$ is a reversed scaling function at scale parameter m. If we see the convolution $\tilde{f} * \bar{\varphi}_m$ as an approximation to $\tilde{f}$ at that scale, we can interpret f as being an m-scale approximation of $\tilde{f}$, so setting $a_m[n] = f(n \cdot \Delta t)$ is not too artificial.

One major downside of this implementation is that it artificially creates array elements close to 0 at the extremities of our signedArrays, and the length of these sequences of 0's grow exponentially. We limit this

with a function `removeZeros` which truncates so that we only keep terms a tolerance away from 0, operating at each iteration of the FWT and iFWT.

Another way of circumventing this issue is to only consider periodic signals, or equivalently periodised wavelets, as in pfwt.jl. This is included in the GitHub repository for completeness, and [13][Sec. 7.5] discusses the theory and more elaborate solutions. Indeed periodising wavelets effectively gets rid of their vanishing moments on the boundary so the signal's smoothness there is not used. We will not use this implementation.

Figure 8 shows different wavelet transforms obtained with fwt.jl for a Gaussian

$$f_1(t) = e^{-(5t)^2/2}$$

and a box function

$$f_2(t) = \begin{cases} 1, & |t| < 1/2, \\ 0, & \text{otherwise}, \end{cases}$$

sampled with $\Delta t = 10^{-3}$ on the interval $[-1, 1]$.

The algorithm's input are samples of f_i, which we consider as an approximation of f at some scale parameter m, that is we start with the approximation coefficients ${a_m}$[11]. We then calculate the sequences of wavelet coefficients d_{m+1}, d_{m+2} as well as the coarsest approximation coefficients a_{m+2}. All of these are sequences in $\mathbb{Z}$, so the abscissa values are not between -1 and 1, but rather are integers from 0 to 1000 or 500. Figures 8a-8c in the first row are for f_1 with Db2, the second row 8d-8f for f_1 with Db9 and the last row 8g-8f is for f_2 with Db9.

We make the following observations:

- Comparing the first and second row, where the only difference is the number of vanishing moments in the wavelet, the visual difference is as expected: the wavelet coefficients are much smaller in the interior for Db9 than for Db2, while the coarsest approximation coefficients are essentially identical. However, both have comparatively large wavelet coefficients at the boundary, which comes from artificially setting the function to 0 outside of $[-1, 1]$. This renders vanishing moments useless at the boundary, but could be remedied by linearly modifying the samples to ensure they start and end at 0 [11], therefore removing discontinuities, although the function might still be non-differentiable at the boundary. This is a caveat of this implementation and is a similar issue to what one encounters when implementing a periodic FWT.

- A more subtle difference between the first 2 rows is that the abscissa slightly extends on both sides for Db9 compared to Db2. This is visually clearer in the approximation coefficients 8f with more indices below 0 than in 8c. This is because the filter associated to Db9 is longer than that associated to Db2, so the signedArray gets longer at every step of the FWT algorithm. Indeed the FWT involves a convolution with the filters h, g, which will extend the samples by its length at every iteration. This also occurs in the wavelet coefficients but it is harder to see in the figures.

- Both Figures 8a, 8b show a graph resembling the Mexican hat wavelet. In fact if one takes the wavelet transform of a smooth enough function f using a wavelet with p vanishing moments, the p-th derivative of f will show up in the wavelet coefficients. This is reminiscent of our discussion on vanishing moments using Taylor expansions.

- For f_2, the corresponding graphs 8g,8h,8i have different abscissa ranges $250 - 750$, $125 - 375$ and $125 - 375$ respectively. The function `removeZeros` is used at each iteration of `fwt`, and removes all the zeros of the original function. The function we're taking the FWT of here is really f_2, restricted to the $[-1/2, 1/2]$ interval, hence the truncated abscissa ranges. The only large wavelet coefficients are at the indices corresponding to values near $|t| = 1/2$, where the function is discontinuous.

- One possibly unexpected outcome is the fact that the coarsest approximation coefficients are a factor of 2 larger than the function samples. The key point here is that the sequences a_{m+2} are approximation *coefficients*, not approximations to the function. The factor of 2 is then easily explained: going 2 scales coarser means considering scaling functions and wavelets with a different pre-factor $2^{-(m+2)/2}$ instead

[11] In the code, we have to modify the samples' datatype to fit with the fwt function, inputting the signedArray `(f,0)`.

of $2^{-m/2}$, so the coefficients must compensate and increase by this factor of 2. This is a common observation to all figures here, but is again more visible on the approximation coefficients.

5.2 Plotting wavelets

Most wavelets we discuss here, and all those implemented in filters.jl, have no explicit expressions to describe them – apart from the trivial Haar wavelet or the Mexican hat wavelet. So how can we even plot them? As with most things related to wavelets, Mallat [13][p. 302, Wavelet Graphs] has an interesting remark on how to plot wavelets and scaling functions using the inverse Fast Wavelet Transform, using that their wavelet and approximation coefficients are mostly trivial. This is exactly the implementation in getWavelet.jl and is how Figure 7 was made.

Figure 9 shows the scaling function φ for Db2, more precisely it shows $\varphi_{-1,0}$ and $\varphi_{0,2}$, a scaled and translated φ, respectively. This is to show the interesting structure of φ, to emphasise how the scaling function really is a *scaling* function, defined by relations with scaled versions of itself. Note that $\varphi_{-1,0}$ and $\varphi_{0,2}$ will look similar for any choice of wavelet, but Db2 was chosen because of its non-differentiability, with which one can easily match one function's kinks with the other's. Indeed, the relation will be less visible in graphs with smoother wavelets.

Another graph of interest for wavelets and scaling functions is their Fourier transform, but numerically applying the FFT to the results above gives rather unaesthetic plots, so we will go with another approach. From the definition of the filter h, we can get a relation between the Fourier transforms $\hat{h}$ and $\hat{\varphi}$, (7.32) in [13]:

$$\hat{\varphi}(\xi) = \prod_{p=1}^{+\infty} \frac{\hat{h}(2^{-p}\xi)}{\sqrt{2}}. \tag{5.2}$$

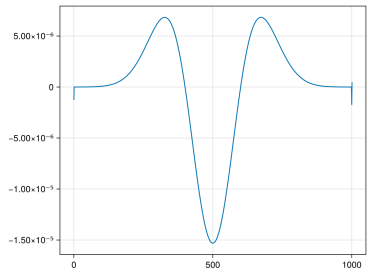
The Fourier transform $\hat{h}$ can be computed numerically using the FFT, and we can multiply a few factors in the above product to obtain an estimate of $\hat{\varphi}$. We can use (4.6) and (4.7) to obtain a similar estimate for $\hat{\psi}$. This is implemented in getFTfromh.jl, with 2 main inefficiencies: the first is that we compute more samples of $\hat{h}$ than we really need, but optimising this would require fiddling with the FFT to ensure we only calculate the necessary values; the second is an issue with finding indices – discussed in lines 30-37 of the julia code.

Figure 10 shows the Fourier transform of Daubechies wavelets and scaling functions with 2 and 10 vanishing moments. Notice that $\hat{\psi}$ is much flatter at 0 for $p = 10$ than for $p = 2$, exactly as our previous discussion on vanishing moments suggested. A very similar behaviour is visible for the scaling functions' Fourier transforms, which seem to get flatter at 0. Increasing the number of vanishing moments also makes the corresponding Fourier transforms have increasingly sharper transitions.

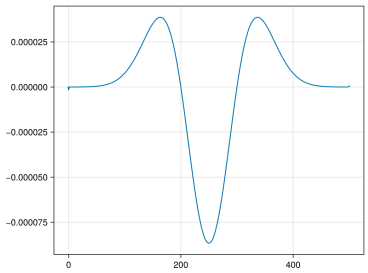
5.3 Denoising

Just as with the FFT, wavelets can be used to remove noise in signals, albeit the approach is simpler to implement with wavelets. Let us first briefly discuss the Fourier transform approach, for which the reader can find many resources online. The Fourier transform of a signal gives us how much of each frequency the signal has, so the Fourier transform of an exact sine wave would have a spike exactly at the corresponding frequency, and 0 elsewhere. Recall that the Fourier transform is highly non-local in time, so a disturbance anywhere in the signal will affect all frequencies. With this in mind, one can see that adding noise to the sine wave will introduce noise to all frequencies. But importantly, as long as the amplitude of the noise is reasonably small, the spike corresponding to the pure sine wave will still supercede the others. Removing noise in the Fourier domain therefore means setting all frequencies with amplitude below a certain threshold to 0.

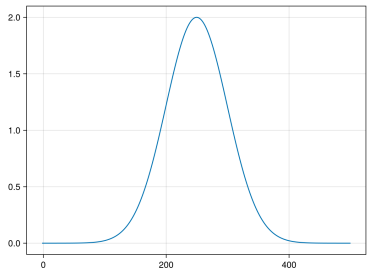
Note that noise can be considered as very high frequency signals, hopefully with small amplitude. This is not obvious if we consider the Fourier transform of a noisy signal, where noise spreads throughout the spectrum, because of the non-local property which motivated us to move to wavelets. Indeed, the wavelet transform of a noisy signal will only differ noticeably to that of the unperturbed signal in the high frequencies. With this in mind, we can simply cut off the high frequencies in the wavelet transform and part of the noise will be removed. This method also uses a threshold, but it is now a cut-off frequency rather than an amplitude, and is implemented in smoothWithfwt.jl.



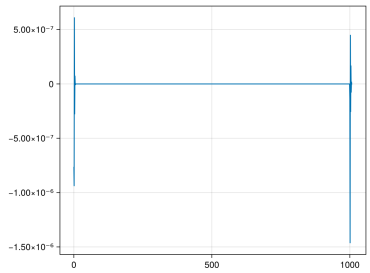
(a) Wavelet coefficients for f_1 with Db2 at scale parameter $m+1$,



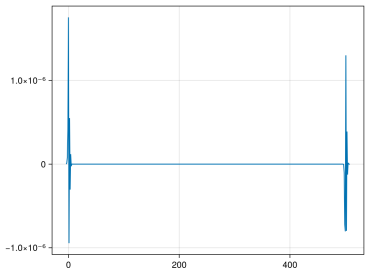
(b) Wavelet coefficients for f_1 with Db2 at scale parameter $m+2$,



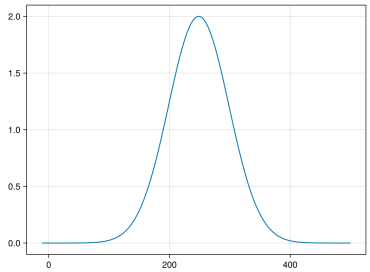
(c) Coarsest scaling function coefficients for f_1 with Db2, at scale parameter $m+2$,



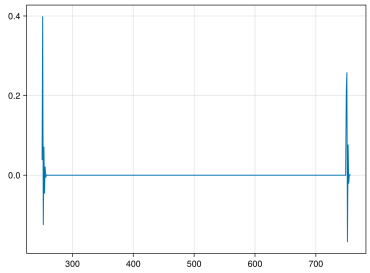
(d) Wavelet coefficients for f_1 with Db9 at scale parameter $m+1$,



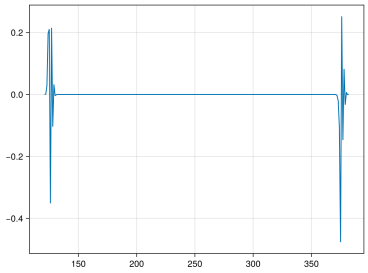
(e) Wavelet coefficients for f_1 with Db9 at scale parameter $m+2$,



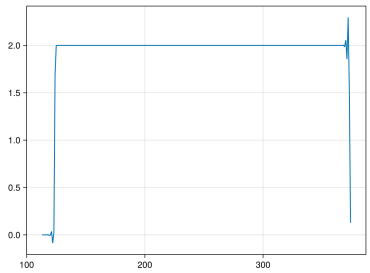
(f) Coarsest scaling function coefficients for f_1 with Db9, at scale parameter $m+2$,



(g) Wavelet coefficients for f_2 with Db9 at scale parameter $m+1$,

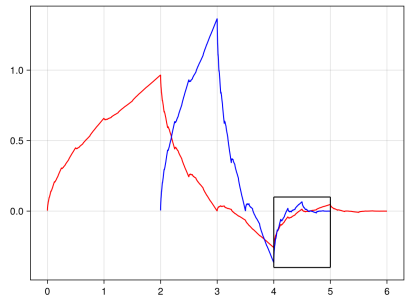


(h) Wavelet coefficients for f_2 with Db9 at scale parameter $m+2$,

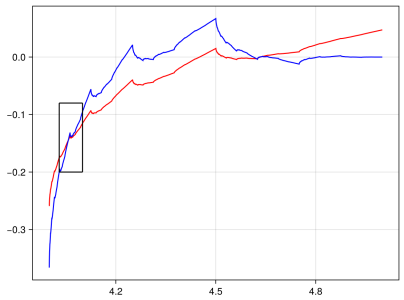


(i) Coarsest scaling function coefficients for f_2 with Db9, at scale parameter $m+2$,

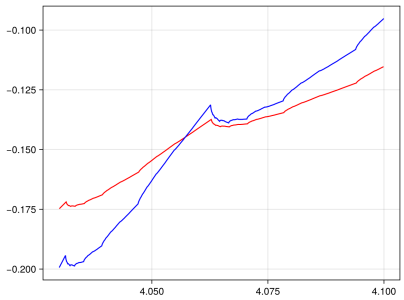
Figure 8: Different wavelet transforms, with 2 different functions and wavelets with different numbers of vanishing moments.



(a) From 0 to 6,

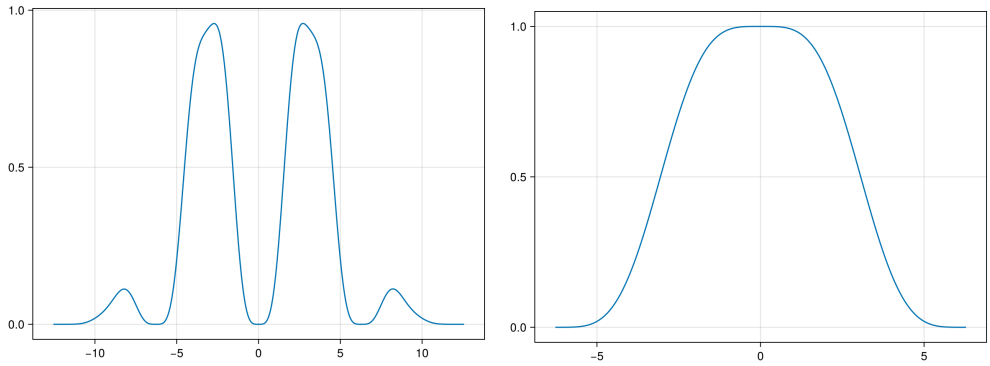


(b) From 4 to 5,

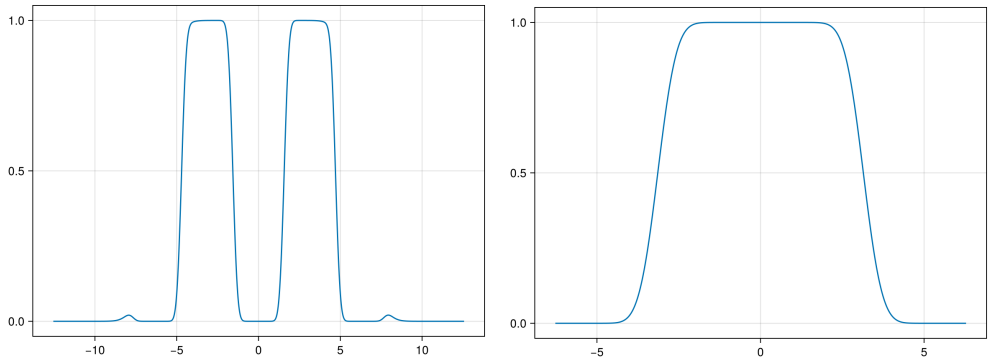


(c) From 4.03 to 4.1,

Figure 9: The translated scaling function $\varphi_{0,2}$ in blue and the scaled $\varphi_{-1,0}$ in red, zooming into the detail. The black box in 9a is the region shown in 9b, and similarly the one in 9b is the region shown in 9c.



(a) $|\hat{\psi}|^2$ for $p = 2$, (b) $|\hat{\varphi}|^2$ for $p = 2$,



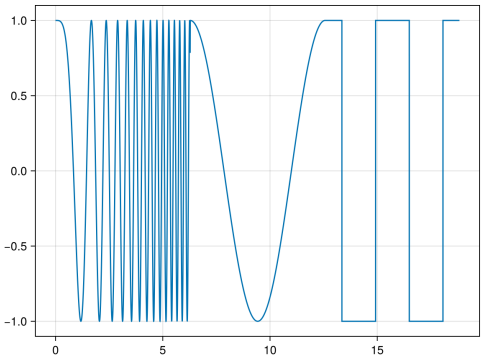
(c) $|\hat{\psi}|^2$ for $p = 10$, (d) $|\hat{\varphi}|^2$ for $p = 10$,

Figure 10: Fourier transform of Daubechies wavelets and their scaling functions for p vanishing moments.

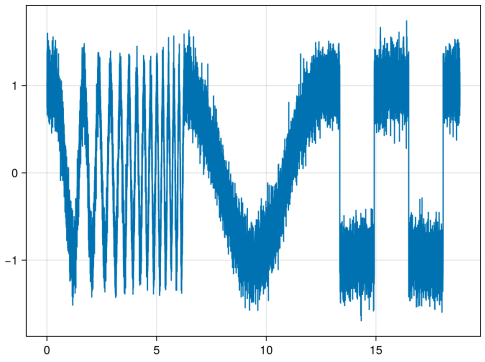
Figure 11 shows an example with an artificial choice of test signal, with Gaussian noise added using the pre-existing Noise.jl package. Figures 11c and 11d show the result of smoothing with 2 different choices of wavelet, the Daubechies wavelet with 2 and 10 vanishing moments respectively. Since the filter h for 10 vanishing moments has 5 times as many non-zero coefficients, the computation time is 5 times as long, and the visual difference in result is minimal. There is a very slight improvement in overall error however, Figure 11d showing less large spikes than 11c for $N = 5$.

Note that for $N = 9$ – Figure 11e – the "smoothed" signal strays quite far from the original. When applying `smoothWithfwt` with a large N, we apply `ifwt` to a very coarse approximation of the signal, with few samples. If one pushes this to the limit, one could even have a single-sample approximation of the signal, under which `ifwt` will return a graph with a similar profile to the wavelet, explaining the spikes appearing in Figure 11e hDb2. The Daubechies wavelets have regularity increasing with the number of vanishing moments, so while the improvement from 11c to 11d is minimal, the choice of wavelet can impact the regularity of the smooth signal by virtue of wavelets appearing: the non-differentiable Db2 wavelet will produce spikes where the smoother Db10 wavelet does not.

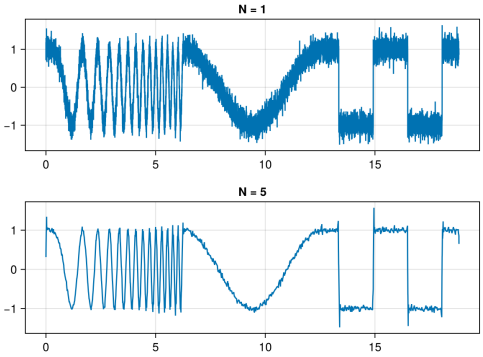
Just like with the FFT denoising method, we could modify our FWT using an amplitude threshold: setting all wavelet coefficients smaller than a certain value to 0. The purpose here is compression of information, indeed if we simply ignore all wavelet coefficients under a certain amplitude, the amount of storage needed drastically decreases. Moreover, if the signal is smooth and the wavelet has many vanishing moments, the wavelet coefficients are guaranteed to be small for small scales, so this compression has reasonable requirements. This algorithm is the norm for image compression and is exactly what JPEG2000 uses. Indeed, wavelets with many vanishing moments are very well adapted to deal with images for the following 2 reasons. Firstly, wavelets are good with detecting and reconstructing edges, unlike the Fourier transform. This is due to the time-localisation of wavelets, meaning an edge/discontinuity will be represented by large wavelet coefficients, recall Figures 8g-8i. The second aspect of images that make them suited for wavelet analysis is that everywhere other than the edges, they are usually quite smooth. Imagine the image of a black square on a white background: the image is essentially constant everywhere other than the sides of the square, so a wavelet with many vanishing moments will yield mostly small coefficients, to be discarded. The only information necessary for those large empty areas will be the coarsest approximation, e.g. what colors are the pixels.



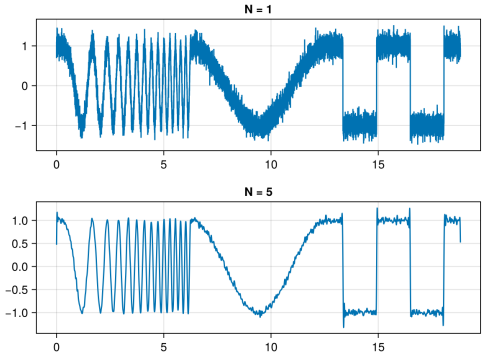
(a) Test signal,



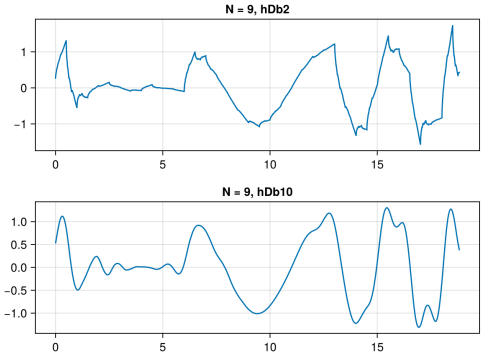
(b) Signal with Gaussian noise,



(c) Smoothed out signal with Db2,



(d) Smoothed out signal with Db10,



(e) N too large, "oversmoothing",

Figure 11: Example of noisy signal smoothed with smoothWithfwt.jl, where N is how many of the first wavelet coefficients we set to 0.

Appendices

The appendices contain further applications, which were not successful enough to include in the main text, but are still worthy of appearing here. No figures will be shown here either, but the julia files mentioned are worth checking out and playing around with.

A Differentiating with the discretised continuous wavelet transform

The theory for this is simple: if f is differentiable, differentiating f in the inverse CWT (3.4) is equivalent to integrating the wavelet coefficients against the derivative of the wavelet. If we can compute the continuous wavelet transform of f, then we should be able to reconstruct its derivative using a modified inverse CWT. The equation here is

$$\frac{d}{dt}f(t) = \frac{1}{C_\psi}\int_0^{+\infty}\int_{-\infty}^{+\infty} \mathcal{W}f(u,s)\cdot\frac{d}{dt}\left(\psi_{u,s}(t)\right)du\frac{ds}{s^2}.$$

The inverse CWT is implemented in icwt.jl. If one wants to simply use the iCWT to recover the original function, the input `fake`ψ should be the wavelet used for the CWT; while if one wants to calculate the derivative using the above formula, the input `fake`ψ should instead be the derivative of the wavelet. More generally, the iCWT as implemented in icwt.jl calculates a discretised version of

$$\frac{1}{C_\psi}\int_0^{+\infty}\int_{-\infty}^{+\infty} \mathcal{W}f(u,s)\cdot \texttt{fake}\psi_{u,s}(t)du\frac{ds}{s^2}. \tag{A.1}$$

While the idea is simple, the results are quite bad. The recovered function, or numerically differentiated function looks similar to the true picture in the interior, but it crosses the t-axis when it shouldn't, and is just plain wrong. Observe that we calculate a discrete, truncated integral, meaning a finite sum of wavelets. All these wavelets have zero average, hence so will their sum. The reconstruction of the function is therefore inherently flawed if the original function has a non-zero integral. This problem could be solved with a modified iCWT using a scaling function [13][Eqn. (4.35)], as the scaling function would take care of the non-zero integral part of the function. However one runs into a few issues when implementing this:

- Most wavelets are not expressed using closed forms, in most cases one has the filter h and calculations don't require anything else, and obtaining the wavelet can be done numerically at best. For these wavelets, (3.4) is irrelevant and cannot be used, so we are constrained to using wavelets for which we can find an explicit form, like the Haar or Mexican hat wavelet. In particular, all of the Daubechies wavelets are gone out the window.
- While the Haar wavelet and scaling function are very easy to implement and offer a perfect reconstruction, remember that for our objective of differentiating functions, we require taking derivatives of the wavelet, a silly idea for a discontinuous wavelet like Haar. We therefore add the constraint of using a differentiable wavelet.
- For the scaling function variant of the iCWT to be used, we need to find an explicit form of the scaling function. This turns out to be more complicated than one could think, and in fact seems to be a nightmare filled with Fourier transform integrals[12].
- One could use the smooth Shannon wavelet (4.14), with explicit and smooth scaling function (4.15). However its decay is painfully slow, meaning numerics would require taking a very large effective support, nipping any hope for efficiency in the bud due to the large intervals of integration.

A final note us that this discretisation is not exactly the same as our approach to discrete wavelets: while the scales are discretised in the same way $s = s_0^m$, the translation step is not scaled. Instead of being of the form $u = ns_0^m u_0$, we simply take $u = nu_0$ at all scales. This choice is made because the arrays all have the same dimension, making the stored information much easier to handle, but the other discretisation might yield better results.

[12] Try getting the scaling function for the Mexican hat wavelet, for example.

B Numerically obtaining the filter from the scaling function

Here is a method to numerically obtain the filter h from a scaling function. We still want to try differentiating functions with wavelets, but instead of calculating the integrals we want to use the FWT, for which the filter h is required. Note that most of the above problems still carry over, but this was coded before I realised all of that, and this is still an interesting code. We can compute h by using its definition (4.5), where the inner products are calculated through numerical integration.

If the above problems didn't occur, we would obtain the wavelet and approximation coefficients in $O(N)$ operations, and we then obtain an approximation to the derivative by taking

$$\frac{d}{dt}f(t) \approx \sum_{m=m_0}^{m_0+N}\sum_{n\in\mathbb{Z}} d_m[n]\frac{d}{dt}\psi_{m,n}(t) + \sum_{n\in\mathbb{Z}} a_{m_0+N}[n]\frac{d}{dt}\varphi_{m_0+N,n}, \tag{B.1}$$

where m_0, m_0+N are the finest and coarsest scales respectively. The error here would depend upon how accurate the filter h is – comparing its numerically obtained and exact values – as well as how small the wavelet coefficients are scale parameters $m < m_0$ would be, which depends upon how smooth the signal is and how many vanishing moments the wavelet has.

Note that the FWT is $O(N)$, but the constant in front of N depends linearly on the length of h. Moreover, for a wavelet with infinite support, h will have infinite length. In practice we use the same approach of considering an effective support of functions, and we discard entries of h smaller than some tolerance. The effective length of h therefore mostly depends on the decay of ψ.

This method is implemented in hFromScal.jl. I have only tested it on Daubechies scaling functions, ironically enough, although this could be used with any sampling of a scaling function. The process to test how good the algorithm was by getting a high-resolution sampling of the scaling function with `getScal` (in getWavelet.jl) then plugging it into hFromScal, thus giving a numerically obtained h_{approx}. One can then directly compare this to the original filter h, or one could plot the wavelet and/or scaling function one gets with h_{approx} and compare that to the one obtained with the original filter. The method is roughly correct, but not accurate enough to warrant any figures, or a section in the main text.

If $\frac{d}{dt}\psi, \frac{d}{dt}\varphi$ can be obtained explicitly and are written as functions in the code, the expression (B.1) actually defines a function. In particular one can evaluate the derivative at any sampling point. The downside of this is that one sample of the derivative requires the whole sum and repeatedly calling ψ', φ'.

C Differentiation in wavelet bases

The above method can be generalised for linear operators $T : L^2(\mathbb{R}) \to L^2(\mathbb{R})$, but we still require an explicit expression for $T\psi, T\varphi$, which we've seen can be problematic. Another approach is presented here, originally worked through by Beylkin in [4], as well as my implementation that doesn't work.

The theory is explained in more detail in Sections II and V of Beylkin's paper. We want to express the operator T in an orthogonal wavelet basis $\bigoplus_{m\in\mathbb{Z}} W_m$, but since the operator might not preserve the structure of this decomposition – $TW_m \subsetneq W_m$ in general – we must be more careful. In particular define the projection P_m onto the subspace $V_m = \overline{\text{span}}\{\varphi_{m,n}\}$ by

$$P_m f = \sum_{n\in\mathbb{Z}} \langle f, \varphi_{m,n}\rangle \varphi_{m,n}, \tag{C.1}$$

and similarly define the projection Q_m onto W_m by $Q_m = P_{m-1} - P_m$. Then one can write T in the following "telescopic" series

$$T = \sum_{m\in\mathbb{Z}} (Q_m T Q_m + Q_m T P_m + P_m T Q_m),$$

which can be truncated from the finest scale m_0 to the coarsest scale $m_0 + N$ to get an approximation T_{m_0} at scale m_0 of the form

$$T_{m_0} = \sum_{m=m_0}^{m_0+N} (Q_m T Q_m + Q_m T P_m + P_m T Q_m) + P_{m_0+N} T P_{m_0+N}. \tag{C.2}$$

Here $T_{m_0} = P_{m_0} T P_{m_0}$, and to simplify notation we write

$$A_m = Q_m T Q_m : W_m \to W_m, \tag{C.3}$$
$$B_m = Q_m T P_m : V_m \to W_m, \tag{C.4}$$
$$\Gamma_m = P_m T Q_m : W_m \to V_m, \tag{C.5}$$
$$T_m = P_m T P_m : V_m \to V_m. \tag{C.6}$$

We represent the operators A_m, B_m, Γ_m and T_{m_0+N} by matrices $\alpha^m_{n_1,n_2}, \beta^m_{n_1,n_2}, \gamma^m_{n_1,n_2}, r^{m_0+N}_{n_1,n_2}$, respectively. For the following we consider the operator $T = \frac{d}{dt}$, then the matrix elements for A_m, B_m, Γ_m can be explicitly computed to be

$$\alpha^m_{n_1,n_2} = 2^{-m} \int_{-\infty}^{+\infty} \psi(2^{-m}t - n_1) 2^{-m} \psi'(2^{-m}t - n_2) dt = 2^{-m} \alpha_{n_1 - n_2}, \tag{C.7}$$
$$\beta^m_{n_1,n_2} = 2^{-m} \int_{-\infty}^{+\infty} \psi(2^{-m}t - n_1) 2^{-m} \varphi'(2^{-m}t - n_2) dt = 2^{-m} \beta_{n_1 - n_2}, \tag{C.8}$$
$$\gamma^m_{n_1,n_2} = 2^{-m} \int_{-\infty}^{+\infty} \varphi(2^{-m}t - n_1) 2^{-m} \psi'(2^{-m}t - n_2) dt = 2^{-m} \gamma_{n_1 - n_2}, \tag{C.9}$$

respectively, where

$$\alpha_n = \int_{-\infty}^{+\infty} \psi(t) \psi'(t) dt, \tag{C.10}$$
$$\beta_n = \int_{-\infty}^{+\infty} \psi(t) \varphi'(t) dt, \tag{C.11}$$
$$\gamma_n = \int_{-\infty}^{+\infty} \varphi(t) \psi'(t) dt. \tag{C.12}$$

The first integral in (C.7) is obtained by writing a vector and its image in W_m in the natural basis $\{\psi_{m,n}\}_{n \in \mathbb{Z}}$, and similarly for (C.8) and (C.9). We then rescale the integrals, cancelling one 2^{-m} factor to get the second set of equations. Defining

$$r_n = \int_{-\infty}^{+\infty} \varphi(t) \varphi'(t) dt \tag{C.13}$$

and using the definitions of filters h and g (4.5), (4.9), we can rewrite

$$\alpha_n = 2 \sum_{\substack{-L+2 \le k_1 \le 1, \\ -L+2 \le k_2 \le 1}} g[k_1] g[k_2] r_{2n+k_1-k_2}, \tag{C.14}$$
$$\beta_n = 2 \sum_{\substack{-L+2 \le k_1 \le 1, \\ 0 \le k_2 \le L-1}} g[k_1] h[k_2] r_{2n+k_1-k_2}, \tag{C.15}$$
$$\gamma_n = 2 \sum_{\substack{0 \le k_1 \le L-1, \\ -L+2 \le k_2 \le 1}} h[k_1] g[k_2] r_{2n+k_1-k_2}, \tag{C.16}$$

where the ranges for k_1, k_2 are those for which h, g take non-zero values. As Beylkin says [8][p. 19], "the representation of $\frac{d}{dt}$ is completely determined by the coefficients r_n, or in other words, by the representation of $\frac{d}{dt}$ on the subspace V_0". These coefficients can then be obtained by solving a system of linear algebraic equations, for which Beylkin devises a method and gives the coefficients corresponding to the first few Daubechies wavelets.

This is the theory, but that always requires some re-shaping to fit on a computer. A noticeable difference here is that the decomposition (C.2) requires both the wavelet and scaling function coefficients of each scale, as opposed to our preferred way of storing them, forgetting all but the last scaling function coefficients. This is exactly a consequence of $TW_m \subsetneq W_m$ in general and will require a modified FWT algorithm storing

all wavelet coefficients, as well as scaling coefficients. The operator T_{m_0} is then split into operators with domain and image W_m or V_m for some m. One quickly notices that our iFWT algorithm is incompatible, and a more thoughtful modification is needed. In order to use our iFWT reconstruction algorithm as already implemented in fwt.jl, it is helpful to write

$$T_{m-1} = A_m + B_m + \Gamma_m + T_m, \tag{C.17}$$

and one can simply iterate this equation from $m = m_0$ all the way up to T_{m_0+N} to recover (C.2). The first two operators A_m, B_m on the right hand side of (C.17) have images in W_m, while the last two Γ_m, T_m have images in V_m, so at each step of the algorithm we start with some approximation coefficients $T_m a_m$, add $\Gamma_m d_m$, then use `ifwt` on this along with wavelet coefficients $A_m d_m + B_m a_m$ to obtain $T_{m-1}a_{m-1}$. We keep iterating this until we reach $m = m_0$, as implemented in beylkinDerivative.jl[13]. Note that m_0 might not be an integer, indeed $m_0 = \log_2(\Delta t)$ is chosen to ensure correctness of the scaling factors. Remember that in this discretisation of wavelets, the translation step – which one can understand as the sampling width – is 2^{-m}, hence the choice of finest scale parameter m_0. Beylkin starts at scale $m = 0$, where the notation is easier on the eyes, but the resulting derivative must be scaled appropriately.

Before moving onto how the results looked, note that the operator T_m is essentially a discretisation of the derivative operator on the scaling coefficients, which we've seen can be written as $\langle f, \varphi_{m,n}\rangle = f * \overline{\varphi}_m(n2^m)$. If the length of the filter h is finite, the operators mentioned here all have a finite width band of entries around the diagonal, and in particular the projected operator T_m is anti-symmetric and therefore essentially acts as a finite difference on $f * \overline{\varphi}_m$. This is done at scale parameter $m_0 + N$ and all subsequent steps in the algorithm bring this finite difference back to our original m_0 scale parameter.

The coefficients r are implemented directly from [3] in beylkinCoefficients.jl, for the first 6 Daubechies wavelets, and the function $\alpha\beta\gamma$ calculates the corresponding α, β, γ matrix elements. Note that all operators are banded about the diagonal with repeating entries, so we need only know one slice of it, hence the signedArray structure.

The implementation is in beylkinDerivative.jl, but it doesn't work. The issue seems to be that the coefficients $B_m a_m$ are too large. Indeed they are scaling coefficients, modified by an operator of order 1, which get mapped to the wavelet coefficients, hence the resulting wavelet coefficients are much larger than they ought to be, causing our result to oscillate. In fact one sees that the result is split into 2^N separate lines, where N is the number of steps taken in the FWT. I think the problem stems from the $B_m a_m$ coefficients, but it might be something else entirely[14].

[13] Beylkin does this differently, modifying the ifwt such that one can give both scaling and wavelet coefficients. Implementing this gives exactly the same result, so is not given in more detail here.

[14] Please let me know if you find what's wrong with it.

References

[1] G. Bachmann, L. Narici, and E. Beckenstein. *Fourier and Wavelet Analysis*. Universitext. Springer New York, 2002.

[2] G. Battle. Heisenberg Inequalities for Wavelet States. *Applied and Computational Harmonic Analysis*, 4(2):119–146, 1997.

[3] G. Beylkin. On the representation of operators in bases of compactly supported wavelets. *SIAM Journal on Numerical Analysis*, 29(6):1716–1740, 1992.

[4] G. Beylkin. Wavelets and fast numerical algorithms. *Proceedings of Synopsia in Applied Mathematics*, 1993.

[5] B. Burke Hubbard. *The World According to Wavelets: The Story of a Mathematical Technique in the Making, Second Edition*. CRC Press, 1998.

[6] I. Daubechies. Orthonormal bases of compactly supported wavelets. *Commun. Pure Appl. Math.*, 41(7):909–996, October 1988. Part II was published in *Siam J. Math. Anal.* **24**:2 (1993), as was Part III. MR:951745. Zbl:0644.42026.

[7] I. Daubechies. *Ten Lectures on Wavelets*. Society for Industrial and Applied Mathematics, 1992.

[8] V. R. G. Beylkin, R. Coifman. Fast wavelet transforms and numerical algorithms I. *Communications on Pure and Applied Mathematics*, 44(2):141–183, 1991.

[9] S. Jaffard. Pointwise smoothness, two-microlocalization and wavelet coefficients. *Publicacions Matemàtiques*, 35(1):155–168, 1991.

[10] A. Kirsanov. Wavelets: a mathematical microscope. `https://www.youtube.com/watch?v=jnxqHcObNK4`.

[11] J. Luo, B. Jing, and S. Jinhua. Application of the wavelet transforms on axial strain calculation in ultrasound elastography. *Progress in Natural Science*, 16:942–947, 09 2006.

[12] S. Mallat. A theory for multiresolution signal decomposition: the wavelet representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(7):674–693, 1989.

[13] S. Mallat. *A Wavelet Tour of Signal Processing, Third Edition: The Sparse Way*. Academic Press, Inc., USA, 3rd edition, 2008.

[14] Y. Meyer. *Ondelettes et opérateurs*. Number v. 1 in Actualités mathématiques. Hermann, 1990.

[15] I. Novikov, V. Protasov, and M. Skopina. *Wavelet Theory*. Translations of mathematical monographs. American Mathematical Society, 2011.

[16] G. Rémond-Tiedrez. Wavelets github repository. `https://github.com/gabrielr-t/Wavelets`.

[17] D. Stranneby and W. Walker. *Digital Signal Processing and Applications*. Newnes, Oxford, second edition edition, 2004.

[18] R. S. Strichartz. How to make wavelets. *The American Mathematical Monthly*, 100(6):539–556, 1993.