Optimally matched wavelets

von Henning Thielemann

Dissertation

zur Erlangung des Grades eines Doktors der Naturwissenschaften
– Dr. rer.nat. –

Vorgelegt im Fachbereich 3 (Mathematik und Informatik)
der Universität Bremen
im Dezember 2005

Datum des Promotionskolloquiums: 2006-02-09

Gutachter:	Prof. Dr. Peter Maaß	(Universität Bremen)
	Dr. Gerd Teschke	(Konrad-Zuse-Zentrum für Informationstechnik Berlin)

Contents

Zusammenfassung

Diese Arbeit behandelt diskrete Waveletfunktionen, welche so konstruiert werden, dass sie vorgegebene Muster annähern. Für den Entwurf von biorthogonalen Waveletbasen stellen wir einen Ansatz vor, welcher das sogenannte Lifting Scheme verwendet. Dieses Schema stellt eine Parametrisierung aller biorthogonalen Waveletbasen dar. Mithilfe dieses Schemas kann man die Waveletkonstruktion auf ein lineares Ausgleichsproblem zurückführen. Die Berechnung der optimalen Lösung wird durch die spezielle Struktur des Problems sehr effizient.

Jede verfeinerbare Funktion, welche perfekte Rekonstruierbarkeit ermöglicht, kann als duale Generatorfunktion eingesetzt werden. Durch die Wahl der Generatorfunktion wird gleichzeitig die Glattheit der zugehörigen Waveletfunktion festgelegt. Weiter wird der Frage nachgegangen, wie man auch die Glattheit der primalen Waveletfunktionen sicherstellen kann. Eine vielversprechende Möglichkeit besteht darin, die diskrete Wavelettransformation leicht abzuwandeln. Man erhält so einen Spezialfall der Wavelettransformation mit doppelter Koeffizientendichte. Waveletbasenpaare, welche bei dieser Transformation verwendet werden können, erreichen sowohl hohe Glattheit als auch eine gute Annäherung der dualen Waveletfunktionen an das vorgegebene Muster. Da die Anzahl der Abtastwerte durch die Transformation verdoppelt wird, ist diese Transformation redundant. Die Waveletkonstruktion und die entsprechende Transformation werden an MEG-Daten und an der Prozessüberwachung von Linearführungen getestet.

Neben diesem Hauptergebnis wird die Übertragungsmatrix von verfeinerbaren Funktionen untersucht. Es werden interessante Eigenschaften und effiziente Berechnungen des Spektralradius, der Summe der Eigenwertpotenzen und der Determinante hergeleitet. Außerdem wird eine allgemeine Lifting-Zerlegung für alle Filterbänke der CDF-Familie vorgestellt.

Die mathematische Notation weicht in mancher Hinsicht von der üblichen Notation ab und orientiert sich an der funktionalen Programmierung. Die FOURIER-Transformation wird vermieden, soweit dies sinnvoll ist, so dass die Berechnungen einfacher in Computerprogramme übersetzt werden können. Es werden Symbole für die Skalierung und Verschiebung von Funktionen eingeführt und als Rechenoperationen behandelt, welche bislang nur zur Veranschaulichung eingesetzt wurden.

Abstract

This thesis addresses the problem of constructing a discrete wavelet approximating the shape of a given pattern. For the design of a biorthogonal wavelet basis we present an approach, which is based on the lifting scheme. The lifting scheme is a parametrisation of all biorthogonal wavelets. It reduces our problem to a linear least squares problem. The special structure of the problem allows for an efficient optimisation algorithm.

Every refinable function can be used as a dual generator, if it respects the perfect reconstruction constraints. The smoothness of the generator also implies the smoothness of the dual wavelet. Strategies for obtaining also a smooth primal wavelet function are discussed. The most promising way includes a slight modification of the discrete wavelet transform, leading to a special case of the so called double density transform. With this modification we can achieve both good matching and high smoothness of the wavelets. It doubles the amount of data and is thus redundant. The method is applied to the analysis of MEG data and to the condition monitoring on linear guideways.

Furthermore the transfer matrix of refinable functions is explored in various ways. Interesting properties and efficient computations of the spectral radius, the sum of eigenvalue powers, and the determinant are investigated. An explicit lifting decomposition of CDF filter banks is shown.

The mathematical details are presented in a notation inspired by functional programming. Since the FOURIER transform is avoided where this is sensible, the results are easily accessible for implementation in computer programs. Symbols for function scaling and translation that were only used for illustrative purposes in former wavelet related papers are now integrated into a strict formalism.

Chapter 1

Introduction

1.1 Problem

The wavelet transform is a tool for decomposing signals into bricks. Signals can be time series like audio signals, two dimensional data like images and so on. This is similar to the FOURIER transform where a signal is decomposed into waves of different frequencies. It allows to determine cycles in the signal. But the waves cover the whole considered interval and thus local features of a signal cannot be retrieved from the transformed data. In contrast to that, the wavelet transform uses bricks, which are narrow in the spatial domain, but which are also able to resolve frequencies or scales. This is similar to the wave-particle dualism in quantum theory. Indeed realizations like HEISENBERG's uncertainty principle are also essential for wavelet analysis.

We do now try to characterise the bricks of the wavelet transform, namely the wavelets.

> A *wavelet* is a function that is concentrated both in the spatial domain and in the frequency domain.

"Spatial domain" also includes "time". The concentration in the frequency domain causes oscillations in the function, hence the name "wavelet" as synonym for "small wave". Each type of wavelet transform needs wavelet functions with specific properties. More specific definitions can be given when considering transform instances.

All bricks of the wavelet transform are derived from a prototype by translation and dilation. The transforms detect the same pattern at different sizes, i.e. scales. A signal which has similar features of sizes over many different magnitudes is called to have a multi-scale structure. We know this structure from fractals and many natural phenomena. Last but not least this document exhibits a multi-scale organisation constructed of chapters, section, subsections and so on.

Over the last decades a lot of research was done in wavelet analysis. There are two main kinds of wavelet transforms which are widely applied:

The continuous wavelet transform is an integral transform just for mathematical purposes. Translation and dilation of the input signal simply translates the result. It can be discretised but the discretised transform increases the amount of data considerably and the result data is redundant, yet not suitable for numerically stable inversion of the transform. Consequently the continuous wavelet transform is mainly applied for analysis and visualisation, e.g. of medical and seismic data [DHK+03].

The discrete counterpart of the wavelet transform is not just a discretisation of the continuous transform. This is rather different from the situation of the FOURIER transform as we will see in more detail in Chapter 2. The discrete transform is designed for perfect reconstruction, high computation speed and no output redundancy. The disadvantages are that it reacts sensible on translations and dilations of the input signal and that the choice of wavelets is quite restricted. The discrete wavelet transform is successfully applied in audio [VK95, Mal99] and image compression [SP96, TM02, Str02], even suitable for hardware accelerated compression [Rit02], as well as for the solution of partial differential equations as alternative to finite element methods [Dah97, DDHS97, BBC+01]. In contrast to that, the de-noising of signals, which is also a popular application, suffers from artifacts in the reconstructed signal or too much smoothing.

There is an application which was not payed so much attention to in the past: The detection of certain patterns in a signal. The continuous wavelet transform is quite good at finding occurrences of certain patterns in a signal but it is computationally expensive and unable to extract or remove these patterns cleanly. The discrete wavelet transform suffers from the restricted choice of admissible wavelet functions. In Chapter 3 we will derive a method for designing wavelets which match a given pattern. The resulting filter banks will not be satisfying due to their insufficient numerical stability. This problem is treated in Chapter 4. Finally, Chapter 5 completes this work with implementation details, potential applications and experiments.

Acknowledgements I thank Professor Maaß for his motivation and consultation about this thesis. There was a fruitful discussion with and assistance by many people who gave me new ideas and helped me in areas of mathematics I am not so familiar with. First of all my colleague KRISTIAN BREDIES helped me a lot with functional analysis, especially function spaces, and pseudo inverses. GÜNTHER ROTE gave me hints for tackling the determinant factorisation, FABIAN WIRTH pointed me to resultants, HANS TRIEBEL assisted with embedding of smoothness spaces, ROBERT STRICH helped me expressing some of my results in terms of group theory, THORSTEN RAASCH introduced me how to use wavelets for solving partial differential equations, and this list is certainly not complete. Last but not least I am grateful to the careful proof readers, especially KRISTIAN BREDIES.

1.2 Notations

In this section we will introduce and discuss several notations that are used throughout the document.

1.2.1 Some notes about notations

Much like natural languages the mathematical notation evolves over the time, new symbols appear, the meaning of some symbols changes, or a notation becomes uncommon. While natural languages are well studied there seems to be only low effort in studying mathematical notation. Finding good mathematical notations is like modelling real-world problems with mathematics. Like problem modelling, mathematical notation could be a mathematical area of its own. [Caj93]

There is a lot of common abuse of notation around and even notations that are commonly accepted are not always satisfying. But what is abuse if nobody is authorised to tell what the right use is? Without a broadly accepted guide of the right notation it seems to be nonsensical to discuss abuse. Nevertheless this discussion is worthwhile because there are notation requirements that probably everyone accepts in general – as long as there are no conflicts with a special notation one is familiar with. It is not easy to formulate these demands in a precise, orthogonal (i.e. axiomatic) but comprehensible style, so we are content with some examples illustrating why particular notations should be favoured over others.

Since wavelet analysis is part of *Functional Analysis* a central matter of this area are functions, functions of functions (so-called operators) and so on. In this area it is often necessary to define functions. A clean way to do this is to define a function pointwise, just like $f(x) = x + 2$. But it is often inconvenient to introduce a new identifier for a function that is used only once. So there are several notations to construct a function without a name, the most prominent ones are certainly $x \mapsto x + 2$ and $\cdot + 2$.

The dot can be used for showing that some arguments of a function call are omitted, i.e. we have still a function in the omitted arguments. This notation is useful for readability since one can denote the absolute value function by $|\cdot|$ whereas $||$ could be mixed up with the sign for parallelism or with norm bars. But the extension of the dot notation to more complex expressions such as $f(\cdot - t)$ has various disadvantages:

1. Using the dot you cannot easily express a constant function, e.g. $x \mapsto 2$.
2. The dot notation does not allow for multiple argument functions like $(x, y) \mapsto x^2 + y^2$.
3. To say what belongs to the function you need to declare the scope of the dot expression. There is no such generally accepted notation for the scope and thus the expressions are ambiguous. E.g. it is not known whether $g(f(\cdot))$ means

- $x \mapsto g(f(x))$, i.e. $g \circ f$,
- $g(x \mapsto f(x))$, i.e. $g(f)$ or even
- $g(f(x \mapsto x))$, i.e. $g(f(\mathrm{id}))$

That is why in this document we avoid the dot notation and stick to the arrow notation (also known as ALONZO CHURCH's *lambda* notation). Indeed the latter one is used much less in the area this work belongs to, but we share the negative attitude of Dijkstra with respect to justifications like "The paper is addressed to the XYZ community, which habitually expresses itself in that style, deviation from which would result in failure to reach his intended audience." [Dij86a].

Sometimes it is argued that abuse in notation makes expressions shorter and thus easier to read. In general it is true that disregarding some writing rules extends the set of possible character strings to express a formula and thus allows to choose a quite short one. But there are also lots of complications by misunderstood formalisms, just like the not so uncommon expression $f(\cdot)$ which is actually equivalent to f.

We do not only want to get rid of some unfavourable notations, we also want to introduce some new notations. These are largely inspired by *functional programming* [Hug84, Dau04], which focuses on the notion of a *function*.

A function is some black box which maps certain input values (arguments) to output values (function values). The output only depends on the input, the function cannot maintain a *state*. The result of the function may be undefined for some arguments. In other words, functions need not to be *total*, they are in general *partial*. Many works on functional analysis are based on equivalence classes of functions with respect to the "equal almost everywhere" relation $\overset{\text{a.e.}}{=}$. They identify a function with the corresponding equivalence class. But assigning an argument to a unique function value is such an essential property of functions that we do not want to lose it. So, in this respect we follow books like [EG92]. Strictly spoken, due to this interpretation we do not have function norms but only semi-norms because $\|f\| = 0$ merely means that f is zero almost everywhere.

Functions can have functions both as arguments and as results. These so-called *higher order function*s are known as *operator*s or *functional*s to the functional analysis community. It seems comfortable not to use the restrictive scheme $f : A \to B$ to tell that a function maps from set A to set B, but to interpret $A \to B$ as the set of all functions mapping from A to B. Thus we can write $f \in A \to B$. Using this interpretation we can easily formulate, e.g. , that the differentiation operator D for real functions is a (obviously not total) function with $D \in (\mathbb{R} \to \mathbb{R}) \to (\mathbb{R} \to \mathbb{R})$.

We want to provide a *declarative style* to make clear what we actually mean rather than letting the meaning result from what we say. The functional notation helps to achieve this goal. E.g. , if you are familiar with the arrow notation, you will probably agree that the expression $(\psi \uparrow b) \to a$ makes unambiguously clear that we want to dilate the function ψ by a factor of b and then move it by a distance of a to the right. It is not the intuitive use of arrows that makes things clear, but it is the fact that the arrow operators process functions rather than function values. Dilation and translation operators D^a and T^b like in [LMR97] share this spirit.

In contrast to that, the usual notation $\psi\left(\frac{t-a}{b}\right)$ denotes a function value. If one wants to interpret it as an expression describing a function it is not sure which of the variables a, b, t shall vary and which are constant. The reader needs some time to analyse whether the expression describes a dilation or a shrinking (in time direction), a translation to the right or to the left. He also needs some time to detect the order of translation and dilation. By converting each operation inside the argument into the corresponding operation of the function we can show the equivalence of both expressions.

$$\begin{aligned}\psi\left(\frac{t-a}{b}\right) &= (\psi \uparrow b)\ (t-a) \\ &= \big((\psi \uparrow b) \to a\big)\ (t)\end{aligned}$$

Writing an expression with functions rather than function values is known as the *point-free style*.

A similar example relates to filter masks (cf. Definition 1.2.1). It is common to consider a filter mask as a polynomial, thus writing $h(z)$. If one wants to shift the filter mask by one step, this is commonly denoted by $z \cdot h(z)$. But strictly spoken $z \cdot h(z)$ is a complex number. You can apply a lot of operations to complex

numbers, but not each of them can be expressed in terms of operations to filter masks. For instance you can write $\left|h(z)\right|^2$ but then you leave it open whether that still represents a filter, i.e. whether it is always possible to find a filter g for which $\forall z\ \left|h(z)\right|^2 = g(z)$ holds. Actually for $\left|h(z)\right|$ this is not possible in general.

Another argument for the point-free style is that some common filter operations cannot be expressed as functions of evaluated polynomials. Up-sampling is an example but it can be expressed with a modified argument to the polynomial function. E.g. h is g up-sampled if $\forall z\ h(z) = g(z^2)$. The down-sampling operation is more difficult. Down-sampling means removing all odd-indexed elements of a sequence. In the traditional style we had to say that the filter h is the down-sampled version of g if $2 \cdot h(z^2) = g(z) + g(-z)$. For a down-sampling by factor 3 this would be even more complicated: $3 \cdot h(z^3) = g(z) + g(z \cdot e^{\mathrm{i}\cdot 2\cdot\pi/3}) + g(z \cdot e^{\mathrm{i}\cdot 4\cdot\pi/3})$. Alternatively with trigonometric polynomial functions we would describe it with $2 \cdot \widehat{h}(2\cdot\omega) = \widehat{g}(\omega) + \widehat{g}(\omega+\pi)$. This way down-sampling can only be written implicitly: "h is g down-sampled, if a specific relation between h and g holds." It is not even obvious whether the relation is unique, that is whether there is only one down-sampling for every g. We want to make that explicit, we want a symbolic analogon to "g down-sampled". Therefore we will simply write $h = g \downarrow 2$. The sign $\downarrow$ denotes an infix operator just like $+$. A very similar notation was already used in [Str96, DS98].

The overall consequence is that we will define and use operations on filter masks, rather than describing their effect on polynomial function values. The properties in Lemma 1.2.2 will help to manipulate expressions containing this operation. Admittedly some of them are trivial when treating filters like evaluated polynomial functions, but a few properties are really novel.

Let us summarise the pros and cons of extensive use of operations in the spatial domain instead of operations in the frequency space for real functions.

Advantages:

- The FOURIER transform is not defined for all functions from $\mathbb{R} \to \mathbb{R}$ and for all sequences from $\mathbb{Z} \to \mathbb{R}$, and it is defined depending on the subsets. That is functions from $\mathcal{L}_2(\mathbb{R})$ need a FOURIER transform definition different from that for functions from $\mathcal{L}_1(\mathbb{R})$. It is even harder to design a universal transform and thus it is reasonable to avoid the transform where not really necessary.
- The convolution of two functions is defined in the spatial domain. It can be simplified when considering the frequency spectrum of the functions: Convolving two functions means multiplying their frequency spectra pointwise. But this equivalence can only be used if the FOURIER transform can be applied to both operands. For instance the identity function from $\mathbb{R} \to \mathbb{R}$ is not contained in common spaces compatible with the FOURIER transform like $\mathcal{L}_1(\mathbb{R})$ and $\mathcal{L}_2(\mathbb{R})$. A convolution with a finitely supported function, for instance the convolution $\chi_{[-1,1]} * \mathrm{id}$, poses no problem in the spatial domain. The result is obviously $2 \cdot \mathrm{id}$.

Disadvantages:

- There is no function from $\mathbb{R} \to \mathbb{R}$ which is neutral with respect to convolution. Approaches like *distribution*s or *Non-Standard Analysis* [LR94] can resolve this problem. But they are quite complicated. In the frequency space there is no problem since the function which is constantly 1 (i.e. $\xi \mapsto 1$) is neutral with respect to pointwise multiplication. Thus it is hard to find a setting with a neutral element for convolution but it is easy to tell the frequency spectrum of that element.
- For many proofs we need associativity of the convolution. In the FOURIER domain convolution becomes multiplication. Multiplication is associative, so for $\mathcal{L}_2(\mathbb{R})$ functions convolution is associative. But a simple criterion for associativity is not obvious.

Consequently, conforming to a basic rule of good mathematical style [Dij86b] "Only introduce identifiers you need" we want to write $g(z)$ and $\widehat{\varphi}(\xi)$ only if we really want to state something about the function values. Extending this wish for simplicity we want to apply the FOURIER transform only if needed. That is we write $\varphi * \psi$ instead of $\widehat{\varphi} \cdot \widehat{\psi}$ wherever possible. Since the dot like in $\varphi \cdot \psi$ is commonly associated with the point-wise multiplication of functions we avoid using it for convolution, also for discrete signals. Because of the usage of $*$ instead of $\cdot$ it is probably better to use adapted notations for multiplication related operations like power, product, matrix multiplication, and determinant.

In functional programming a feature named *currying* is quite popular. It is in honour of the logician HASKELL CURRY although SCHÖNFINKEL was the one who first described this idea [Tho99]. It means

that a function with more than one argument is defined by a higher order function. We can define

$$\begin{aligned} f &\in \mathbb{R} \times \mathbb{R} \to \mathbb{R} \\ f(x, y) &= x + y \end{aligned}$$

which is the un-curried form. But the curried form

$$\begin{aligned} f &\in \mathbb{R} \to \mathbb{R} \to \mathbb{R} \\ f(x)(y) &= x + y \end{aligned}$$

is an equivalent yet different definition. The curried definition allows us to omit the second argument. This is certainly not worth a new term in mathematics but in the area of computer languages this method is restricted to languages providing higher order functions. So, in computer science the term *partial application* was introduced.

Operators (i.e. functions which map functions to functions) are naturally in curried form. That is we write $Df(x)$ (meaning $(Df)(x)$) instead of $D(f, x)$ for the derivative of f evaluated at x. The partial applied form is Df and it denotes the derivative of f.

It is convenient to use partial application for infix operators, too, where it is called *section*. We allow notations like $(x+)$ and $(+x)$ for the functions $y \mapsto x + y$ and $y \mapsto y + x$, respectively. For example, the composition $(\cdot y) \circ (x+)$ means $z \mapsto (x + z) \cdot y$.

There are very different notations for function application. Sometimes arguments are written without parentheses, e.g. for standard functions and operators like in $\sin x$ and Df, respectively, sometimes arguments are enclosed in parentheses, e.g. for custom functions, say $f(x)$ instead of $f\ x$, sometimes arguments are written as subscript as for tuples, sequences, parametrised functions or families of functions, e.g. x_i, f_α. We stick to the traditional notations here in many cases, but we want to interpret these objects uniformly as functions. For sequences this means that we treat them as functions of integer variables. The notation A^n is equivalent to $\{1, \ldots, n\} \to A$ and $A^{n \times m}$ is equivalent to $(\{1, \ldots, n\} \times \{1, \ldots, m\}) \to A$. This allows us to write $f \circ x$ if we want to apply the function f to all elements of the sequence x.

Function application is left associative.

$$f(x)(y) = (f(x))(y)$$

This implies that ABx is different from $A(Bx)$, more precisely

$$\begin{aligned} ABx &= (AB)x \\ A(Bx) &= (A \circ B)x \end{aligned}$$

In contrast to that, function set construction notation (the "$\to$" operator) shall be right associative. That is $A \to B \to C$ means $A \to (B \to C)$. This complements the left associativity of function application since if $f \in A \to B \to C$ and $x \in A$ then $f(x) \in B \to C$.

1.2.2 Operators and basic properties

We will deal with several types of objects: Two basic types are discrete signals (sequences) and continuous signals (real functions). We will define several operations for them in the following paragraphs.

Discrete signals

1.2.1 Definition (Discrete signal, discrete filter, sequence). A function h from $\mathbb{Z} \to \mathbb{R}$ is called a *discrete real signal*. Analogously a function h from $\mathbb{Z} \to \mathbb{C}$ is called a *discrete complex signal*. A function or sequence h from $\ell_0(\mathbb{Z})$ is called a *finitely supported discrete signal*, that is, only a finite number of coefficients are not zero. We will define a special frequently used signal and some operations that can be applied to discrete signals.

Neutral element		$\delta = (\dots, 0, 0, \mathbf{1}, 0, 0, \dots)$
of convolution	$\forall k \in \mathbb{Z}$	$\delta_k = \begin{cases} 1 & : k = 0 \\ 0 & : k \neq 0 \end{cases}$
Component selection		h_k
Negation	$\forall k \in \mathbb{Z}$	$(-h)_k = -h_k$
Alternating negation	$\forall k \in \mathbb{Z}$	$(h_-)_k = (-1)^k \cdot h_k$
Adjoint	$\forall k \in \mathbb{Z}$	$h^*_k = \overline{h_{-k}}$
Translation	$\forall \{k, c\} \subset \mathbb{Z}$	$(h \to c)_k = h_{k-c}$
	$\forall \{k, c\} \subset \mathbb{Z}$	$(h \leftarrow c)_k = h_{k+c}$
Down-sampling	$\forall \{k, c\} \subset \mathbb{Z}$	$(h \downarrow c)_k = h_{c \cdot k}$
Up-sampling	$\forall \{k, c\} \subset \mathbb{Z}$	$(h \uparrow c)_k = \sum_{j : k = c \cdot j} h_j$
		$= \begin{cases} h_{k/c} & : k \equiv 0 \mod c \\ 0 & : k \not\equiv 0 \mod c \end{cases}$
Addition	$\forall k \in \mathbb{Z}$	$(h + g)_k = h_k + g_k$
Index interval		$\operatorname{ix} h = \{\min A, \dots, \max A\}$
(Convex hull of support)		with $A = \{k \ : \ h_k \neq 0\}$
Set sum		$A + B = \{x + y \ : \ (x, y) \in A \times B\}$
Sum		$\sum h = \sum_{j \in \mathbb{Z}} h_j$
Scalar product		$\langle h, g \rangle = \sum_{j \in \mathbb{Z}} h_j \cdot \overline{g_j}$
Norm		$\|h\|_p = \sqrt[p]{\sum_{j \in \mathbb{Z}} \lvert h_j \rvert^p}$
		$\|h\|_\infty = \sup_{j \in \mathbb{Z}} \lvert h_j \rvert$
		$\|h\|_2 = \sqrt{\langle h, h \rangle}$
FOURIER transform $\widehat{h} \in \mathbb{R} \to \mathbb{C}$	$\forall \omega \in \mathbb{R}$	$\widehat{h}(\omega) = \sum_{j \in \mathbb{Z}} h_j \cdot e^{-\mathrm{i} \cdot \omega \cdot j}$
Convolution	$\forall k \in \mathbb{Z}$	$(h * g)_k = \sum_{j \in \mathbb{Z}} h_j \cdot g_{k-j}$
		$= \langle h, g^* \to k \rangle$
Deconvolution		$h = (h \not* g) * g$
Convolution power		$h^{*0} = \delta$
	$\forall k \in \mathbb{Z}$	$h^{*k+1} = h * h^{*k}$
Matrix-vector-convolution $A \in \ell_0(\mathbb{Z})^{n \times m} \wedge$ $b \in \ell_0(\mathbb{Z})^m$	$\forall k \in \{1, \dots, n\}$	$(A \circledast b)_k = \sum_{l \in \{1, \dots, m\}} A_{k,l} * b_l$
Matrix-matrix-convolution $A \in \ell_0(\mathbb{Z})^{n \times m} \wedge$ $B \in \ell_0(\mathbb{Z})^{m \times r}$	$\forall k \in \{1, \dots, n\}$ $\forall j \in \{1, \dots, r\}$	$(A \circledast B)_{k,j} = \sum_{l \in \{1, \dots, m\}} A_{k,l} * B_{l,j}$

- A signal will be written with the usual tuple notation, where the value at index zero is highlighted

with bold typeset, like this:

$$h = \big(\dots, h_{-2}, h_{-1}, \boldsymbol{h_0}, h_1, h_2, \dots\big)$$

Values not written are zero.

- $\widehat{h}$ is called the *symbol* of h or the *trigonometric polynomial function* associated with h.
- We use a special arrow for down-sampling ($h \downarrow c$ instead of $h \downarrow c$) to make clear that this operation cannot be reversed in general.

There are some basic properties that we will use throughout the document without always emphasising when we do it.

1.2.2 Lemma. Let h, g, and m be discrete signals, and let j and k be integers. Then the following connections hold.

$\delta * h = h$	neutral element of convolution	
$g * h = h * g$	commutativity of convolution	
$(g * h) * m = g * (h * m)$	associativity of convolution	
$(g * h) \to k = g * (h \to k)$	translation is a kind of convolution	
$(h \uparrow k) \downarrow k = h$	invertibility of up-sampling	(1.2.1)
$(h \uparrow k) \uparrow j = h \uparrow (k \cdot j)$	associativity of up-sampling	
$(h \downarrow k) \downarrow j = h \downarrow (k \cdot j)$	associativity of down-sampling	
$(g + h) \uparrow k = (g \uparrow k) + (h \uparrow k)$	distributivity of up-sampling with respect to addition	
$(g + h) \downarrow k = (g \downarrow k) + (h \downarrow k)$	distributivity of down-sampling with respect to addition	
$(g * h) \uparrow k = (g \uparrow k) * (h \uparrow k)$	distributivity of up-sampling with respect to convolution	
$(g \uparrow k * h) \downarrow k = g * (h \downarrow k)$	kind of distributivity for down-sampling	(1.2.2)
$\big((h \uparrow k) \to j\big) \downarrow k = \begin{cases} h \to \left(\frac{j}{k}\right) & : k \mid j \\ 0 & : \text{else} \end{cases}$	commutation of translation and down-sampling	(1.2.3)
$\operatorname{ix}(g * h) = \operatorname{ix} g + \operatorname{ix} h$	index interval of convoluted signals	
$\big(h_-\big)_- = h$	inversion of sign alternation	
$(g * h)_- = g_- * h_-$	alternation is distributive with respect to convolution	
$\big(h^{*n}\big)_- = \big(h_-\big)^{*n}$	alternation commutes with powers	
$(h \to k)_- = (-1)^k \cdot h_- \to k$	alternation of translated signals	
$\langle g \uparrow k, h \uparrow k\rangle = \langle g, h\rangle$	scalar product is invariant under up-sampling	
$\|h \uparrow k\|_p = \|h\|_p$	p-norms are invariant under up-sampling	
$\widehat{\delta}\ (\omega) = 1$	FOURIER transform of the unit impulse	
$\widehat{h^*} = \overline{\widehat{h}}$	FOURIER transform of the adjoint	
$\widehat{h \uparrow k} = \widehat{h} \downarrow k$	FOURIER transform of an upsampled signal	
$\widehat{h \to k}\ (\omega) = \widehat{h}\ (\omega) \cdot e^{-\mathrm{i} \cdot k \cdot \omega}$	FOURIER transform of a translated signal	
$\widehat{h * g} = \widehat{h} \cdot \widehat{g}$	FOURIER transform is a homomorphism mapping convolution to multiplication	

1.2.3 Remark. The identity (1.2.2) is an exception due to its asymmetry. The problem is that the distributivity with respect to down-sampling, that is $(g * h) \downarrow k = (g \downarrow k) * (h \downarrow k)$, does not hold in general.

1.2.4 Definition (Polynomial, Series). A *polynomial* is a tuple from $\mathbb{N}_0 \to R$, a LAURENT *polynomial* is a tuple from $\mathbb{Z} \to R$ (i.e. functions with a finite number of non-zero values) where $(R, 0, 1, +, \cdot)$ is a ring. A *series* is also a sequence from $\mathbb{N}_0 \to R$ or $\mathbb{Z} \to R$, respectively, but with no restriction to the number of non-zero values.

Both $(\mathbb{N}_0 \to R, (0), (1), +, *)$ and $(\mathbb{Z} \to R, (\mathbf{0}), \delta, +, *)$ are rings. With this definition a polynomial is a function which maps indices to coefficients. Actually this algebraic view on polynomials cancels the difference between polynomials and signals.

It is popular to compute weighted sums of powers using polynomials, actually many people refer to these power sums as polynomials. For both kinds of polynomials we introduce the "evaluator" E from $(J \to R) \to (R \to R)$ with the index set J which is either $\mathbb{N}_0$ or $\mathbb{Z}$

$$Ep(z) = \sum_{j \in J} p_j \cdot z^j \qquad \text{[Str98, Definition 1.34].}$$

The evaluator maps a polynomial to a *polynomial function* (also *entire rational function* or *integral rational function*).

Although we will mostly use the point-free style, some properties will convince the reader easier if evaluated polynomial functions are used instead of polynomials. The following properties hold. The first two justify that for every z the function $h \mapsto Eh(z)$ is called the *evaluation homomorphism* (German: *Einsetzungshomomorphismus*).

1.2.5 Lemma.

$$\begin{aligned}
E(h+g)(z) &= Eh(z) + Eg(z) && \text{homomorphism with respect to polynomial } + \text{ and scalar } + \\
E(h*g)(z) &= Eh(z) \cdot Eg(z) && \text{homomorphism with respect to } * \text{ and } \cdot \\
E(h_-)(z) &= Eh(-z) && \text{alternating signs of the coefficients} \\
E(h \uparrow k)(z) &= Eh\left(z^k\right) && \text{up-sampling of signals} \\
E(h \rightarrow k)(z) &= Eh(z) \cdot z^k && \text{translation of signals} \\
\widehat{h}(\omega) &= Eh\left(e^{-\mathrm{i}\cdot\omega}\right) && \text{FOURIER transform}
\end{aligned} \tag{1.2.4}$$

Given the pointwise sum and product of functions the evaluator is itself a homomorphism because $E(h+g) = Eh + Eg$ and $E(h*g) = Eh \cdot Eg$.

1.2.6 Definition (Degree of a Polynomial). The *degree of a polynomial* is a measure for the number of non-zero coefficients of a polynomial.

$$\deg \in (J \to R) \to (\mathbb{N}_0 \cup \{-\infty\})$$

For a polynomial p from $\mathbb{N}_0 \to R$ the degree is the highest index of non-zero coefficients. If all coefficients are zero the degree is negative infinity. (Given an appropriate extension of the set of integers.)

$$\deg p = \max\left\{j \;:\; p_j \neq 0\right\}$$

For a LAURENT polynomial p from $\mathbb{Z} \to R$ the degree is the difference between maximum and minimum index of non-zero coefficients. If all coefficients are zero the degree is negative infinity, again.

$$\deg p = \max\left\{j \;:\; p_j \neq 0\right\} - \min\left\{j \;:\; p_j \neq 0\right\}$$

1.2.7 Lemma. For polynomials p and q it holds

$$\deg(p * q) = \deg p + \deg q \quad .$$

Continuous signals

1.2.8 Definition (Continuous signal, real function). A function φ from $\mathbb{R} \to \mathbb{R}$ is called a *continuous real signal*. Following operations can be applied:

Evaluation		$\varphi(t)$
Negation	$\forall t \in \mathbb{R}$	$(-\varphi)(t) = -\varphi(t)$
Addition	$\forall t \in \mathbb{R}$	$(\varphi + \psi)(t) = \varphi(t) + \psi(t)$
Scalar multiplication	$\forall \{t, c\} \subset \mathbb{R}$	$(c \cdot \varphi)(t) = c \cdot \varphi(t)$
Multiplication	$\forall t \in \mathbb{R}$	$(\varphi \cdot \psi)(t) = \varphi(t) \cdot \psi(t)$
Absolute value	$\forall t \in \mathbb{R}$	$\lvert\varphi\rvert(t) = \lvert\varphi(t)\rvert$
Power	$\forall t \in \mathbb{R}$	$\varphi^2(t) = \varphi(t)^2$
Adjoint	$\forall t \in \mathbb{R}$	$\varphi^*(t) = \overline{\varphi(-t)}$
		$\varphi^* = \overline{\varphi \uparrow (-1)}$
Translation	$\forall \{t, c\} \subset \mathbb{R}$	$(\varphi \rightarrow c)(t) = \varphi(t - c)$
	$\forall \{t, c\} \subset \mathbb{R}$	$(\varphi \leftarrow c)(t) = \varphi(t + c)$
Dilation	$\forall \{t, c\} \subset \mathbb{R}$	$(\varphi \uparrow c)(t) = \varphi\left(\frac{t}{c}\right)$
	$\forall \{t, c\} \subset \mathbb{R}$	$(\varphi \downarrow c)(t) = \varphi(c \cdot t)$
	$\forall c \in \mathbb{R}$	$\varphi \downarrow c = \varphi \circ (c\cdot)$
Tensor product	$\forall (t, s) \in \mathbb{R}^2$	$(\varphi \otimes \psi)(t, s) = \varphi(t) \cdot \psi(s)$
Integral $\int \in \mathcal{P}(\mathbb{R}) \rightarrow (\mathbb{R} \rightarrow A) \rightarrow A$		$\varphi \overset{\text{a.e.}}{=} \left(t \mapsto \int_{(0,t)} \varphi\right)'$
Scalar product		$\langle \varphi, \psi \rangle = \int_{\mathbb{R}} \left(\varphi \cdot \overline{\psi}\right)$
Norm		$\lVert\varphi\rVert_p = \sqrt[p]{\int_{\mathbb{R}} \left(\tau \mapsto \lvert\varphi(\tau)\rvert^p\right)}$
		$\lVert\varphi\rVert_\infty = \operatorname{esssup}_{\tau \in \mathbb{R}} \lvert\varphi(\tau)\rvert$
		$\lVert\varphi\rVert_2 = \sqrt{\langle \varphi, \varphi \rangle}$
Convolution	$\forall t \in \mathbb{R}$	$(\varphi * \psi)(t) = \int_{\mathbb{R}} \left(\tau \mapsto \varphi(\tau) \cdot \psi(t - \tau)\right)$
	$\forall t \in \mathbb{R}$	$(\varphi * \psi)(t) = \langle \varphi, \psi^* \rightarrow t \rangle$
FOURIER transform $\widehat{\varphi} \in \mathbb{R} \rightarrow \mathbb{C}$	$\forall \omega \in \mathbb{R}$	$\widehat{\varphi}(\omega) = \frac{1}{\sqrt{2 \cdot \pi}} \cdot \left\langle \varphi, t \mapsto e^{\mathrm{i} \cdot \omega \cdot t} \right\rangle$
		$\mathcal{F}\varphi = \widehat{\varphi}$

Similar to the operations on discrete signals we want to state corresponding properties for continuous signals.

1.2.9 Lemma. Let φ and ψ be continuous signals, and let c and d be non-zero real numbers, and let p be a positive real number. Then the following connections hold.

$\varphi \downarrow c = \varphi \uparrow \left(\frac{1}{c}\right)$	shrinking expressed in terms of dilation
$(\varphi \uparrow c) \downarrow c = \varphi$	invertibility of dilation
$(\varphi \uparrow c) \uparrow d = \varphi \uparrow (c \cdot d)$	associativity of dilation
$(\varphi \downarrow c) \downarrow d = \varphi \downarrow (c \cdot d)$	associativity of shrinking
$\langle \psi \uparrow c, \varphi \uparrow c \rangle = \lvert c\rvert \cdot \langle \psi, \varphi \rangle$	distributivity of dilation with respect to the inner product
$\lVert\varphi \uparrow c\rVert_p = \sqrt[p]{\lvert c\rvert} \cdot \lVert\varphi\rVert_p$	commutation of dilation and norm
$\lvert c\rvert \cdot (\psi * \varphi) \uparrow c = (\psi \uparrow c) * (\varphi \uparrow c)$	distributivity of dilation with respect to convolution
$(\psi * \varphi) \downarrow c = \lvert c\rvert \cdot (\psi \downarrow c) * (\varphi \downarrow c)$	distributivity of shrinking with respect to convolution (1.2.5)

$\|\widehat{\varphi}\|_2 = \|\varphi\|_2$	the FOURIER transform is an *isometry*
$\left\langle \widehat{\varphi}, \widehat{\psi} \right\rangle = \langle \varphi, \psi \rangle$	the FOURIER transform is *unitary*
$\sqrt{2 \cdot \pi} \cdot \varphi \overset{\text{a.e.}}{=} t \mapsto \left\langle \widehat{\varphi}, \omega \mapsto e^{-\mathrm{i} \cdot t \cdot \omega} \right\rangle$	inverse of the FOURIER transform
$\widehat{\widehat{\varphi}} \overset{\text{a.e.}}{=} \varphi \uparrow -1$	duplicate FOURIER transform flips the signal
$\widehat{\varphi^*} = \overline{\widehat{\varphi}}$	FOURIER transform of the adjoint
$\widehat{\varphi \uparrow c} = \lvert c \rvert \cdot \widehat{\varphi} \downarrow c$	FOURIER transform of dilated function
$\widehat{\varphi \rightarrow k}\ (\omega) = \widehat{\varphi}\ (\omega) \cdot e^{-\mathrm{i} \cdot k \cdot \omega}$	FOURIER transform of translated function
$\widehat{\varphi * \psi} = \sqrt{2 \cdot \pi} \cdot \widehat{\varphi} \cdot \widehat{\psi}$	FOURIER transform is almost a homomorphism mapping convolution to multiplication
$\widehat{\varphi'}(\omega) = \mathrm{i} \cdot \omega \cdot \widehat{\varphi}(\omega)$	FOURIER transform of the derivative
$\widehat{\varphi'} = \mathcal{F}(\omega \mapsto -\mathrm{i} \cdot \omega \cdot \varphi(\omega))$	FOURIER transform of the function with linear multiplier
$\mathcal{F}\left(t \mapsto e^{-t^2/2}\right) = \omega \mapsto e^{-\omega^2/2}$	The GAUSSian is an eigenfunction of the FOURIER transform

Discrete and Continuous signals mixed

We also need operations that connect discrete and continuous signals.

1.2.10 Definition.

Mixed convolution

$$h * \varphi = \sum_{j \in \mathbb{Z}} h_j \cdot (\varphi \rightarrow j)$$

$$\forall t \in \mathbb{R} \quad (h * \varphi)(t) = \sum_{j \in \mathbb{Z}} h_j \cdot \varphi(t - j)$$

discretisation operator Q

$Q \in (\mathbb{R} \to A) \to (\mathbb{Z} \to A)$

$$\forall j \in \mathbb{Z} \qquad (Qf)_j = f(j)$$

1.2.11 Lemma. For each sequence h and each function φ holds

$$Q(h * \varphi) = h * Q\varphi \quad .$$

- We allow convolution of discrete signals with continuous signals. In this case a discrete signal h behaves just like a linear combination of translated DIRAC impulses (δ).

$$\sum_{k \in \mathbb{Z}} h_k \cdot (\delta \rightarrow k)$$

- The term $h * \varphi$, where h is a discrete signal and φ is a continuous one with small support, can also be considered as composing the shape of h with shifted versions of φ. More precisely, if φ is interpolating ($Q\varphi = \delta$) then $h * \varphi$ interpolates h. This can be considered as an inversion of the discretisation in the sense

$$Q(h * \varphi) = h \quad .$$

  This connection is a simple consequence of Lemma 1.2.11 above.

Precedences

The following tables give the precedences of the used infix operators in the order of decreasing precedence.

- Arithmetic

x^y, x^{*n}	right associative
$x \uparrow 2, x \downarrow 2, \varphi \uparrow 2, \varphi \downarrow 2, f \circ g$	left associative
$\varphi/\psi, \varphi \cdot \psi$	left associative
$\varphi * \psi, \varphi \leftarrow k, \varphi \rightarrow j$	left associative
$\varphi + \psi, \varphi - \psi$	left associative
$x \mapsto f(x)$	right associative

- Set operations

$\times$	left associative
$\cup, \cap$	left associative
$\rightarrow$	right associative
$\in, \subset, \supset, \subseteq, \supseteq$	

Adjoint operators

1.2.12 Definition (Adjoint). Given two HILBERT spaces A and B and a linear operator T from $A \rightarrow B$, then the *adjoint* operator T^* from $B \rightarrow A$ is the one for which holds

$$\forall x \in A \, \forall y \in B \quad \langle Tx, y \rangle_B = \left\langle x, T^* y \right\rangle_A \qquad .$$

The adjoint is a generalisation of the transposition of a matrix to complex numbers and linear operators in arbitrary vector spaces.

1.2.13 Lemma. Some laws for computations involving the adjoint in HILBERT spaces such as $\mathcal{L}_2(\mathbb{R})$ and $\ell_2(\mathbb{Z})$.

$f^{**} = f$	adjoining twice is the identity
$f^{-1^*} = f^{*-1}$	adjoining and inversion commute
$(g \circ f)^* = f^* \circ g^*$	adjoint of composition of linear functions
$(*h)^* = \left(*h^*\right)$	adjoint of the discrete convolution
$(*\varphi)^* = \left(*\varphi^*\right)$	adjoint of the continuous convolution
$(*\varphi)^* = Q \circ \left(*\varphi^*\right)$	adjoint of the mixed convolution
$(\uparrow k)^* = (\downarrow k)$	adjoint of up-sampling
$(\uparrow k)^* = (\downarrow k) \circ (\cdot k)$	adjoint of dilation
$(\cdot\varphi)^* = \langle, \varphi \rangle$	adjoint of weighting, adjoint of inner product

Because $(*h^*)$ is the adjoint operator of $(*h)$ we also call h^* the adjoint filter of h.

Discrete mathematics

1.2.14 Definition (Residue class). Let $(R, 0, 1, +, \cdot)$ be a ring and $\{j, k\} \subseteq R$. With $[j]_k$ we denote the *residue class* with respect to the divisor k which contains the representative j.

$$[j]_k = \left\{ i \ : \ k \mid (i - j) \right\}$$

Roughly spoken $[j]_k$ is the set of all ring elements that share the remainder of the division of j by k. But the definition above is more general because the ring need not to provide a division with unique remainder.

Chapter 2

From continuous to discrete wavelets

This chapter introduces the basics of continuous and discrete wavelet transforms which can be found in standard wavelet literature such as [Mal99, Dau92, LMR97]. We do not present new results here, but we want to give an overview over important facts and their proofs. It may also help to become familiar with the notation. The chapter is finished by a survey of some variants of the standard wavelet transforms.

2.1 Continuous wavelet transform

2.1.1 What we are looking for

We want to design a transformation W^* with $W^* \in (\mathbb{R} \to \mathbb{C}) \to (\mathbb{R} \to \mathbb{R} \to \mathbb{C})$ which reveals some structures of a signal that cannot be immediately seen when looking at it. Intuitively we have some expectations to such a transformation.

1. If a signal is amplified by a factor c, then we expect that the transformation of this signal also becomes amplified by c.
2. If two signals are superposed, then the transformation of the superposition shall be the superposition of the separate transformations of these signals.
3. If the signal is delayed for some time, then its transformation is delayed for the same amount of time.

The first two claims are summarised with the term *linearity*, whereas the latter one is called *time-invariance*. Their mathematical descriptions can be derived from the above naive description in a straightforward way (cf. Figure 2.1).

Linearity: amplification	$\forall c \forall f$	$W^*(c \cdot f) = c \cdot W^* f$
Linearity: superposition	$\forall f \forall g$	$W^*(f + g) = W^* f + W^* g$
Time invariance	$\forall t \forall f$	$W^*(f \to t) = W^* f \to t$

From these claims it already follows that W^* must be some sort of *convolution*. We have even more claims for the transformation W^* concerning the composition and decomposition of a signal f.

Composition We want to compose a signal from wavelet functions, which can vary in their position and in their scale. That is, we have a wavelet function ψ and want to compose our signal by superposing functions from $\{\psi \uparrow a \to b : a \in \mathbb{R}_{>0} \wedge b \in \mathbb{R}\}$. The weighting factors for all of these functions constitute a two-dimensional representation, in contrast to the signal which is only one-dimensional.

Let W_ψ be the transform ($W_\psi \in (\mathbb{R} \to \mathbb{R} \to \mathbb{C}) \to (\mathbb{R} \to \mathbb{C})$) which synthesises a one-dimensional signal from a two-dimensional wavelet representation, say g. If g has the value c at the time b and the scale a (i.e. $g(a)(b) = c$), then we expect that the represented signal $W_\psi g$ contains a wavelet at this time and this scale with amplitude c (see Figure 2.2).

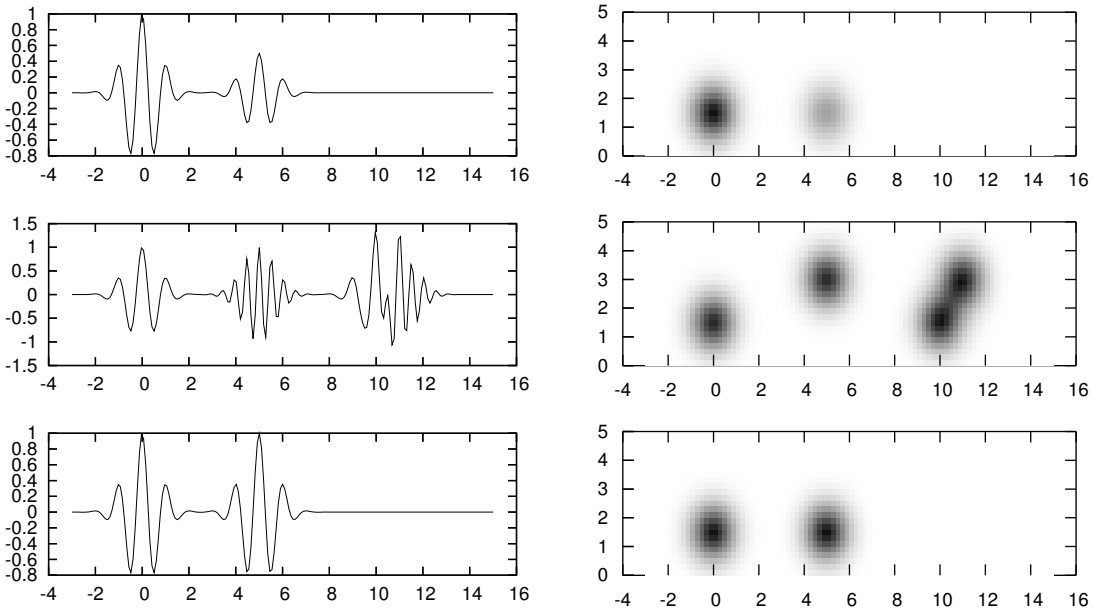


Figure 2.1: Three axioms of linear translation invariant operators covering: commutation of the operator with amplification, distribution of the operator over sums, commutation of the operator with translation. The left column shows the signal f, the right column shows the corresponding wavelet transform $W^* f$.

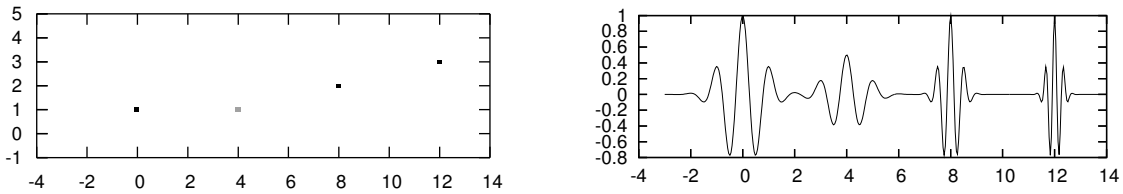


Figure 2.2: The meaning of wavelet coefficients. The left plot visualises the wavelet domain and the right plot shows the spatial domain. Both plots show different representations of the same signal. A wavelet transformed signal is a function of two arguments. The box of the plot is the function domain, the grey colours represent the function values: White is 0 and black is 1. The horizontal position is the time point of the occurrence of a wavelet, the vertical position is the scale of the wavelet, with large scales at the bottom, and the darkness of a dot represents the amplitude of the corresponding wavelet.

Because of the required linearity W_ψ is already determined.

$$(W_\psi g)(t) = \int_{\mathbb{R}_{>0}} \left(a \mapsto \int_{\mathbb{R}} \left(b \mapsto g(a)(b) \cdot \left(\frac{\psi \uparrow a}{\sqrt{a}} \to b \right)(t) \right) \right)$$

In the above formula we collect all dilated and translated versions of the wavelet ψ. Each version of the wavelet is amplified with the corresponding coefficient $g(a)(b)$. We have normalised $\psi \uparrow a$ by $\sqrt{a}$ in order to assert the same $\mathcal{L}_2(\mathbb{R})$ norm for each dilation of ψ, that is $\left\| \frac{\psi \uparrow a}{\sqrt{a}} \right\|_2 = \|\psi\|_2$. This normalisation ensures symmetry with the decomposition transform below.

We compute the composed signal point-wise, that is we compute $(W_\psi g)(t)$ rather than $W_\psi g$. This is a must since we do not want to worry about integrals of functional values. Consequently we take the value at time t of each version of the wavelet and integrate over the values that all versions of the wavelet contribute to this point.

Decomposition A reasonable design of a decomposition transformation is not as obvious as the composition transformation. But the following observation leads to a good decomposition strategy.

2.1.1 Lemma (CAUCHY-SCHWARZ *inequality*). The value of the scalar product is always bounded by the norms of the operands.

$$\forall \{f, g\} \subset \mathcal{L}_2(\mathbb{R}) \quad \left|\langle f, g\rangle\right| \leq \|f\|_2 \cdot \|g\|_2$$

The magnitude of the scalar product meets the product of the norms of its operands if and only if the operands are *collinear*, that is one operand is the weighted version of the other one.

$$\left|\langle f, g\rangle\right| = \|f\|_2 \cdot \|g\|_2 \quad \Leftrightarrow \quad \exists \lambda \quad f \overset{\text{a.e.}}{=} \lambda \cdot g \vee g \overset{\text{a.e.}}{=} (t \mapsto 0)$$

If one has a fixed g and considers a set of normalised functions f ($\|f\|_2 = 1$), then the scalar product $\langle f, g\rangle$ is maximal if f has the same shape as g. If $\|g\|_2 = 1$ then $\langle f, g\rangle \cdot g$ is the projection of f into the linear space spanned by g, that is the scalar product is the weight of g with which it is contained in f.

Given these properties of the scalar product it is certainly a good idea to define the wavelet analysis transform by scalar products of the signal with the dilated and translated versions of the wavelet ψ.

$$({W_\psi}^* f)(a)(b) = \left\langle f, \frac{\psi \uparrow a}{\sqrt{a}} \to b \right\rangle$$

For general patterns ψ the decomposed signal does not look as smooth and clear as Figure 2.1 suggests. However for a special wavelet, namely the MORLET wavelet (Figure 2.4), which looks like a spring (more precisely: a *helix*), the effect applies, that rotation of a spring can be hardly distinguished from translation. This is the reason why shifting the MORLET wavelet along the signal changes mainly the complex phase of the correlation coefficient but influences much less its absolute value. This leads to the smooth shape of the absolute values shown in Figure 2.3.

Note that the use of the adjoint notation ${W_\psi}^*$ is intended. Indeed the decomposition transform ${W_\psi}^*$ is formally the adjoint of W_ψ with respect to $\mathcal{L}_2(\mathbb{R})$ and $\mathcal{L}_2(\mathbb{R}^2)$. We still have to check how ${W_\psi}^*$ is related to the inverse of W_ψ.

2.1.2 Inversion formula and admissibility

Now we want to see what conditions must be fulfilled in order to use W_ψ for inverting ${W_\psi}^*$. According to the definition of the convolution we can rewrite both transformations more concisely.

$$\begin{aligned}
({W_\psi}^* f)(a) &= f * \left(\frac{\psi \uparrow a}{\sqrt{a}}\right)^* \\
&= f * \frac{\psi^* \uparrow a}{\sqrt{a}} \qquad (2.1.1)\\
(W_\psi g)(t) &= \int_{\mathbb{R}_{>0}} \left(a \mapsto \int_{\mathbb{R}} \left(b \mapsto g(a)(b) \cdot \left(\frac{\psi \uparrow a}{\sqrt{a}} \to b\right)(t)\right)\right) \\
&= \int_{\mathbb{R}_{>0}} \left(a \mapsto \int_{\mathbb{R}} \left(b \mapsto g(a)(b) \cdot \left(\frac{\psi \uparrow a}{\sqrt{a}}\right)(t-b)\right)\right) \\
&= \int_{\mathbb{R}_{>0}} \left(a \mapsto \left(g(a) * \frac{\psi \uparrow a}{\sqrt{a}}\right)(t)\right) \qquad (2.1.2)
\end{aligned}$$

We clearly see that both transformations are essentially based on convolutions. Now we plug them together by setting $g(a) = w(a) \cdot ({W_\psi}^* f)(a)$, where $w(a)$ is a scale dependent weighting. This weighting can also be interpreted as a weighting of the measure in the wavelet space. That is if we do not consider $\mathcal{L}_2\left(\mathbb{R}^2\right)$ but a space where the measure is weighted depending on the scale then the adjoint of the wavelet decomposition transform with respect to this inhomogeneous space is actually the composition transform.

We will determine $w(a)$ such that the transformation inverts the analysis transformation ${W_\psi}^*$.

$$\begin{aligned}
\left(W_\psi(w \cdot {W_\psi}^* f)\right)(t) &= \int_{\mathbb{R}_{>0}} \left(a \mapsto \left(w(a) \cdot f * \frac{\psi^* \uparrow a}{\sqrt{a}} * \frac{\psi \uparrow a}{\sqrt{a}}\right)(t)\right) \\
&= \int_{\mathbb{R}_{>0}} \left(a \mapsto w(a) \cdot \left(f * \left(\psi^* * \psi\right) \uparrow a\right)(t)\right) \qquad (2.1.3)
\end{aligned}$$

Structurally, the convolution commutes with the integration. However integrability issues let this connection fail sometimes.

$$W_\psi\left(w \cdot {W_\psi}^* f\right) = f * \left(t \mapsto \int_{\mathbb{R}_{>0}} \left(a \mapsto w(a) \cdot \left(\left(\psi^* * \psi\right) \uparrow a\right)(t)\right)\right)$$

Thus going through wavelet analysis and subsequent synthesis is in total a convolution. To get perfect reconstruction (i.e. $\forall f \; W_\psi\left(w \cdot {W_\psi}^* f\right) \overset{\text{a.e.}}{=} f$) we need to convolve f with the neutral element of the convolution. Unfortunately there is no function which is neutral with respect to convolution. One can only imagine a strange function called the DIRAC impulse which is zero everywhere except for argument zero. For argument zero it is infinite and the kind of infinity is adjusted such that the integral of the DIRAC impulse is one. Since the DIRAC impulse is not a function from $\mathbb{R} \to \mathbb{R}$, we cannot use an approach where the right operand of the convolution is made equivalent to the DIRAC impulse. We could show that the superposition of all $\left(\psi^* * \psi\right) \uparrow a$ vanishes everywhere except at point zero, if we guess the weighting properly and ψ has a vanishing moment. Maybe Non-Standard Analysis is a way out here ([LR94]) since it provides infinitesimal small and large numbers.

It is certainly better to go a different way. The trick to derive the conditions on w and ψ is to switch to the FOURIER domain where the convolution turns into a multiplication. The neutral element of the multiplication of functions is obviously the function which is constant one.

$$\begin{aligned}
\mathcal{F}\big(W_\psi(w \cdot {W_\psi}^* f)\big) &= \mathcal{F}\left(t \mapsto \int_{\mathbb{R}_{>0}} \Big(a \mapsto w(a) \cdot \big(f * (\psi^* * \psi) \uparrow a\big)(t)\Big)\right) \\
&= \omega \mapsto \int_{\mathbb{R}_{>0}} \left(a \mapsto \mathcal{F}\Big(w(a) \cdot f * \big(\psi^* * \psi\big) \uparrow a\Big)(\omega)\right) \\
&= \omega \mapsto \int_{\mathbb{R}_{>0}} \left(a \mapsto w(a) \cdot a \cdot \left(2 \cdot \pi \cdot \widehat{f} \cdot \left(\widehat{\psi^*} \cdot \widehat{\psi}\right) \downarrow a\right)(\omega)\right) \\
&= 2 \cdot \pi \cdot \widehat{f} \cdot \left(\omega \mapsto \int_{\mathbb{R}_{>0}} \left(a \mapsto w(a) \cdot a \cdot \left(\left|\widehat{\psi}\right|^2 \downarrow a\right)(\omega)\right)\right)
\end{aligned}$$

This means, in order to get $\mathcal{F}\big(W_\psi(w \cdot {W_\psi}^* f)\big) = \widehat{f}$ we have to make the right operand of the multiplication a function that is constant $\frac{1}{2 \cdot \pi}$.

$$\begin{aligned}
&\omega \mapsto \int_{\mathbb{R}_{>0}} \left(a \mapsto w(a) \cdot a \cdot \left|\left(\widehat{\psi} \downarrow a\right)(\omega)\right|^2\right) \\
=\ &\omega \mapsto \int_{\mathbb{R}_{>0}} \left(a \mapsto w(a) \cdot a \cdot \left|\widehat{\psi}(a \cdot \omega)\right|^2\right)
\end{aligned}$$

Substitute $\alpha = a \cdot \omega$.

$$= \omega \mapsto \int_{\mathbb{R}_{>0}} \left(\alpha \mapsto \frac{1}{\omega} \cdot w\left(\frac{\alpha}{\omega}\right) \cdot \frac{\alpha}{\omega} \cdot \left|\widehat{\psi}(\alpha)\right|^2\right)$$

To make the overall function (with respect to ω) constant, it must be $w(a) = \frac{1}{c_\psi \cdot a^2}$ for some constant c_ψ.

$$= \omega \mapsto \int_{\mathbb{R}_{>0}} \left(\alpha \mapsto \frac{1}{c_\psi \cdot \alpha} \cdot \left|\widehat{\psi}(\alpha)\right|^2\right)$$

Since we want to obtain $\omega \mapsto \frac{1}{2 \cdot \pi}$ we have to choose

$$c_\psi = 2 \cdot \pi \cdot \int_{\mathbb{R}_{>0}} \left(\alpha \mapsto \frac{\left|\widehat{\psi}(\alpha)\right|^2}{\alpha}\right) \quad . \tag{2.1.4}$$

2.1.3 Graduation of scales

So far we have used a linear *graduation* of scales. That is if we increase the parameter a by an amount of Δa the scale increases by Δa. One can argue that it is more natural to grade the scales exponentially, because e.g. the human auditory system perceives equal ratios of frequencies as equivalent. In an exponential graduation the scale is multiplied by a certain factor when we increase the scale parameter by a specific difference. The discrete wavelet transform as introduced in Section 2.2.3 will naturally use the exponential graduation.

If we switch to different graduation, we have to adapt some weighting. This can be seen also intuitively, because the more area a part of the wavelet transform covers the less it must be weighted.

Let γ be a graduation, that is a differentiable function which maps an interval A surjectively to $\mathbb{R}_{>0}$. Then we can substitute a by $\gamma(\alpha)$ in the formula (2.1.3), where α is our new scale parameter.

$$\begin{aligned}
\left(W_\psi(w \cdot {W_\psi}^* f)\right)(t) &= \int_A \left(\alpha \mapsto \gamma'(\alpha) \cdot w(\gamma(\alpha)) \cdot \left(f * \left(\psi^* * \psi\right) \uparrow \gamma(\alpha)\right)(t)\right) \\
&= \int_A \left(\alpha \mapsto \frac{\gamma'(\alpha)}{c_\psi \cdot \gamma(\alpha)^2} \cdot \left(f * \left(\psi^* * \psi\right) \uparrow \gamma(\alpha)\right)(t)\right) \\
&= \int_A \left(\alpha \mapsto \frac{\gamma'(\alpha)}{c_\psi \cdot \gamma(\alpha)^2} \cdot \left(f * \frac{\psi^* \uparrow \gamma(\alpha)}{\sqrt{\gamma(\alpha)}} * \frac{\psi \uparrow \gamma(\alpha)}{\sqrt{\gamma(\alpha)}}\right)(t)\right) \\
\left(V_{\psi,\gamma} f\right)(\alpha) &= f * \frac{\psi^* \uparrow \gamma(\alpha)}{\sqrt{\gamma(\alpha)}} \\
\left(V^{-1}_{\psi,\gamma} g\right)(t) &= \int_A \left(\alpha \mapsto \frac{\gamma'(\alpha)}{c_\psi \cdot \gamma(\alpha)^2} \cdot \left(g(\alpha) * \frac{\psi \uparrow \gamma(\alpha)}{\sqrt{\gamma(\alpha)}}\right)(t)\right)
\end{aligned}$$

If we choose $\gamma(\alpha) = e^\alpha$ and $A = \mathbb{R}$, this simplifies to

$$\begin{aligned}
\left(V_{\psi,\exp} f\right)(\alpha) &= f * \frac{\psi^* \uparrow e^\alpha}{\sqrt{e^\alpha}} \\
\left(V^{-1}_{\psi,\exp} g\right)(t) &= \int_{\mathbb{R}} \left(\alpha \mapsto \frac{1}{c_\psi \cdot e^\alpha} \cdot \left(g(\alpha) * \frac{\psi \uparrow e^\alpha}{\sqrt{e^\alpha}}\right)(t)\right) \quad .
\end{aligned}$$

We become aware that when using the exponential graduation we would not need a scale-dependent weighting if we normalised the scaled versions of the wavelet ψ to a constant $\|\cdot\|_1$ norm.

As a spin-off we obtain an alternative description of c_ψ if we substitute α by e^α in (2.1.4).

$$c_\psi = 2 \cdot \pi \cdot \int_{\mathbb{R}} \left(\alpha \mapsto \left|\widehat{\psi}(e^\alpha)\right|^2\right)$$

2.1.4 Uncertainty principle and time frequency atoms

Ideally the wavelet analysis transform should turn a single occurrence of the wavelet in the signal into a single dot in the wavelet space and the synthesis transform should map a single wavelet coefficient into a single wavelet. However, the analysis transform maps a function with a one-dimensional domain to a function with a two-dimensional domain. It is intuitively clear that there must be some redundancy in the generated data, that is some points depend on others. This prohibits that a wavelet is mapped to a single dot.

For our application of finding patterns in a signal this raises the problem that it is not possible to obtain sharp peaks at the locations and scales of occurrences of the pattern. Instead even if a pattern appears in a signal without distortion the wavelet transform generates a blurred dot.

The problem is quantitatively described by uncertainty principles. They are treated in great detail e.g. in [Tes01]. We want to demonstrate the problem here only for the simple case of the FOURIER transform.

2.1.2 Theorem (HEISENBERG's uncertainty principle).
Prerequisite. Let S be the so called *position operator* and D be the so called *momentum operator* with

$$\begin{aligned}
Df &= f' \\
Sf(t) &= t \cdot f(t) \quad .
\end{aligned}$$

D must be restricted to a domain where it is somehow "skew-selfadjoint" or "skew-HERMITian", namely $D^* = -D$. That is for the integration by parts must hold $\langle Df, g\rangle + \langle f, Dg\rangle = 0$. This is fulfilled if

$$\begin{aligned}
\lim_{t\to\infty} f(t) \cdot g(t) &= 0 \\
\lim_{t\to-\infty} f(t) \cdot g(t) &= 0 \quad .
\end{aligned}$$

So in the following let $f \in \mathcal{L}_2(\mathbb{R})$. Further on both $D(Sf)$ and $S(Df)$ must be defined and

$$\lim_{t\to\infty} t \cdot f(t)^2 = 0$$
$$\lim_{t\to-\infty} t \cdot f(t)^2 = 0 \quad .$$

Claim. HEISENBERG's *uncertainty principle* states, roughly spoken, that the variance of a function and the variance of its FOURIER transform cannot be minimised simultaneously [Ham89, LMR97]. The variance with respect to zero is a functional defined as $\operatorname{var} f = \|Sf\|_2^2$. The exact statement is

$$\operatorname{var} f \cdot \operatorname{var} \widehat{f} \geq \frac{1}{4} \cdot \|f\|_2^2 \cdot \left\|\widehat{f}\right\|_2^2$$

or equivalently

$$\|Sf\|_2 \cdot \|Df\|_2 \geq \frac{1}{2} \cdot \|f\|_2^2 \quad .$$

The functions with least "waste" of variance are dilated GAUSSians.

$$\operatorname{var}\left(t \mapsto \sqrt{\lambda} \cdot e^{-(\lambda\cdot t)^2/2}\right) \cdot \operatorname{var}\left(t \mapsto \frac{1}{\sqrt{\lambda}} \cdot e^{-(t/\lambda)^2/2}\right) = \frac{1}{2} \cdot \sqrt{\pi}$$

Proof. The elegant classic proof from quantum mechanics can be found in [Tri80, Theorem 34.2]. It begins with the observation that the *commutator* of D and S, namely $D \circ S - S \circ D$, is the identity operator.

$$\begin{aligned} D(Sf)(t) - S(Df)(t) &= f(t) + t \cdot f'(t) - t \cdot f'(t) \\ &= f(t) \end{aligned}$$

The further proof consists essentially of the application of the CAUCHY-SCHWARZ inequality (Lemma 2.1.1).

$$\begin{aligned} \|f\|_2^2 &= \left|\langle f, f\rangle\right| \\ &= \left|\left\langle D(Sf) - S(Df), f\right\rangle\right| \\ &= \left|\left\langle D(Sf), f\right\rangle - \left\langle S(Df), f\right\rangle\right| \end{aligned}$$

$D^* = -D \ \wedge \ S^* = S$

$$\begin{aligned} &= \left|-\langle Sf, Df\rangle - \langle Df, Sf\rangle\right| \\ &= \left|\langle Sf, Df\rangle + \langle Df, Sf\rangle\right| \\ &= \left|\langle Sf, Df\rangle + \overline{\langle Sf, Df\rangle}\right| \\ &= \left|2 \cdot \Re\langle Sf, Df\rangle\right| \\ &\leq 2 \cdot \left|\langle Sf, Df\rangle\right| \\ &\leq 2 \cdot \|Sf\|_2 \cdot \|Df\|_2 \end{aligned}$$

The CAUCHY-SCHWARZ inequality for Sf and Df is an equation if both functions are collinear. The inequality $\left|\Re\langle Sf, Df\rangle\right| \leq \left|\langle Sf, Df\rangle\right|$ becomes an equality if and only if $\langle Sf, Df\rangle$ is real.

- Case $Sf = (t \mapsto 0)$
  This means that f is constant zero almost everywhere. (The only exception can be at zero.)
- Case $\exists\lambda\ Df = \lambda \cdot Sf$

$$\begin{aligned} \langle Sf, Df\rangle &= \overline{\lambda} \cdot \langle Sf, Sf\rangle \\ &= \overline{\lambda} \cdot \|Sf\|_2^2 \end{aligned}$$

Since $\langle Sf, Df\rangle$ shall be real and $\|Sf\|_2^2$ is always real, λ must be, too.
The equation $Df = \lambda \cdot Sf$ is an explicit differential equation that we are going to solve now. We shall note before that if f is zero somewhere it is zero everywhere, because $f = t \mapsto 0$ is a solution and the solution of the differential equation is unique. However the case $f = t \mapsto 0$ was already considered above, so we can assume the f is everywhere distinct from zero. This is the reason why we can divide by f without restrictions.

$$\begin{aligned} f' &= t \mapsto \lambda \cdot t \cdot f(t) \\ \frac{f'}{f} &= t \mapsto \lambda \cdot t \qquad \Big| \qquad g \mapsto t \mapsto \int_{(0,t)} g \\ t \mapsto \ln(f(t)) - \ln(f(0)) &= t \mapsto \lambda \cdot \frac{t^2}{2} \\ t \mapsto \frac{f(t)}{f(0)} &= t \mapsto e^{\lambda \cdot \frac{t^2}{2}} \\ \forall t \quad f(t) &= f(0) \cdot e^{\lambda \cdot \frac{t^2}{2}} \end{aligned}$$

For $\lambda < 0$ f is square integrable, but for $\lambda \geq 0$ it diverges.

□

The variance according to the definition above is measured with respect to zero. It could also be measured with respect to the centre of gravity. Then it would be invariant under translation.

A natural kind of analysis is to combine the FOURIER transform with the spatial representation of a signal. Because of the minimal variance of a GAUSSian we could correlate a signal with a GAUSSian which is translated both in time and in frequency (i.e. modulated). Such shifted and modulated GAUSSians are called *time frequency atoms*. The resulting transform is known as GABOR transform.

2.1.3 Definition (GABOR transform). The GABOR *transform* G or *windowed* FOURIER *transform* from $(\mathbb{R} \to \mathbb{R}) \to (\mathbb{R} \to \mathbb{C}) \to (\mathbb{R} \to \mathbb{R} \to \mathbb{C})$ of a function f with respect to a *window* (or *envelope*) g is defined as

$$G_g f(\omega) = f * \left(t \mapsto g(t) \cdot e^{-\mathrm{i}\cdot\omega\cdot t}\right) \quad .$$

Due to the minimal uncertainty of the GAUSSian it is popular to choose $g(t) = e^{-(\lambda\cdot t)^2/2}$ where λ controls the width of the window.

We observe that the more narrow the GAUSSian the better the resolution in time and the worse the resolution of frequencies. When the GAUSSian approaches zero variance we obtain the original signal for all frequencies. (The limit function of the GAUSSian had to have a finite positive area but no extent, which is certainly impossible.) When the GAUSSian approaches infinite variance we obtain the frequency spectrum in each vertical slice. (Here the limit function of the GAUSSian would be a constant function with finite positive area, which is also not possible.) What remains constant for all dilations of the GAUSSian is the overall resolution. Therefore the GABOR transform is ideally suited for demonstrating the effect of the uncertainty principle and why it is not possible to increase the overall resolution arbitrarily, see Figure 2.3. It is also the foundation for pictures of HEISENBERG boxes, such as Figure 2.7.

The GABOR transform cannot be expressed as a wavelet transform. The closest analogon is the transform with a MORLET *wavelet*. Unfortunately this wavelet is not suitable for the basic continuous wavelet transform since the integral in (2.1.4) diverges. Nonetheless the MORLET wavelet is a very important wavelet for the continuous wavelet transform. Modifications of the transform or the wavelet can fix this problem.

The main difference between the GABOR transform and the MORLET wavelet transform is that in the wavelet transform *frequency* and *scale* are coupled by inverse proportionality whereas in the GABOR transform the scale (i.e. the width of the envelope) is constant and only the frequency varies. (Figure 2.4) The wavelet transform matches the property of many natural signals that high frequent signal components vary faster than low frequent ones.

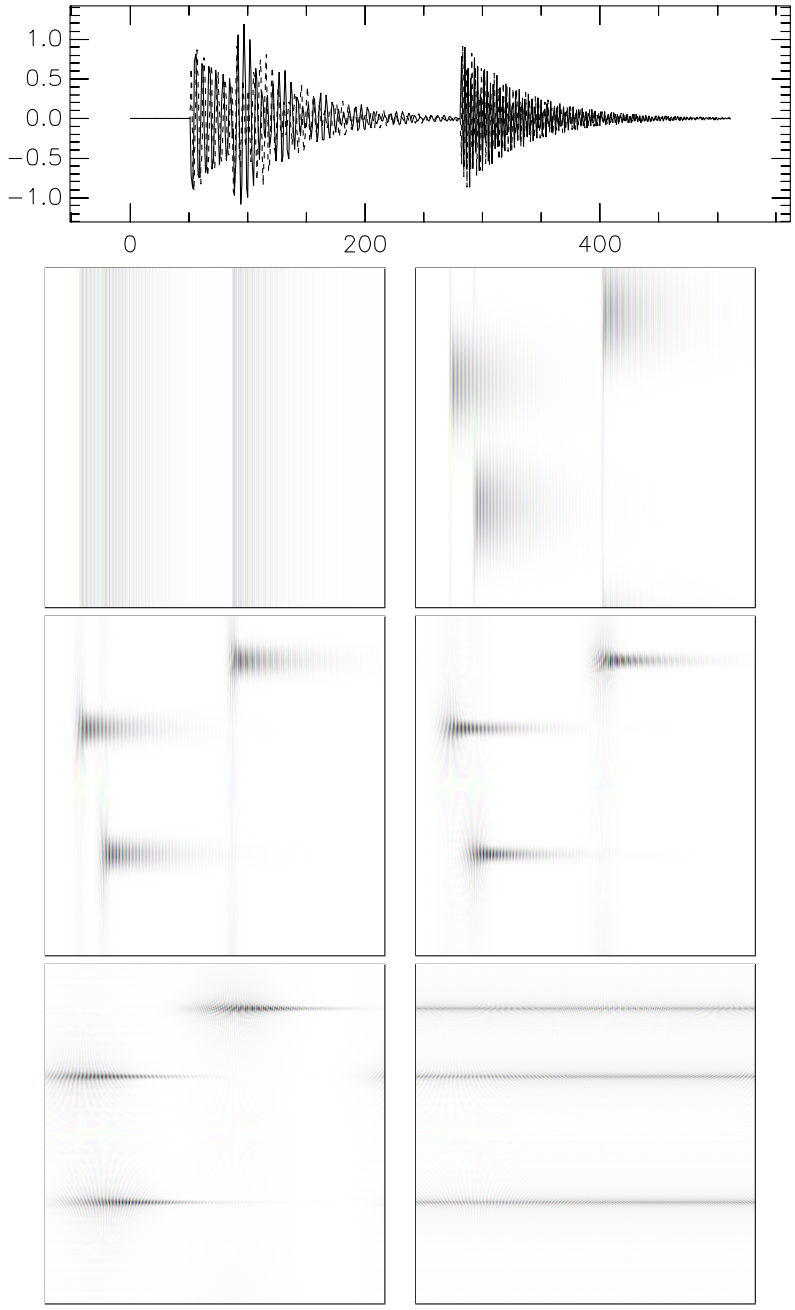
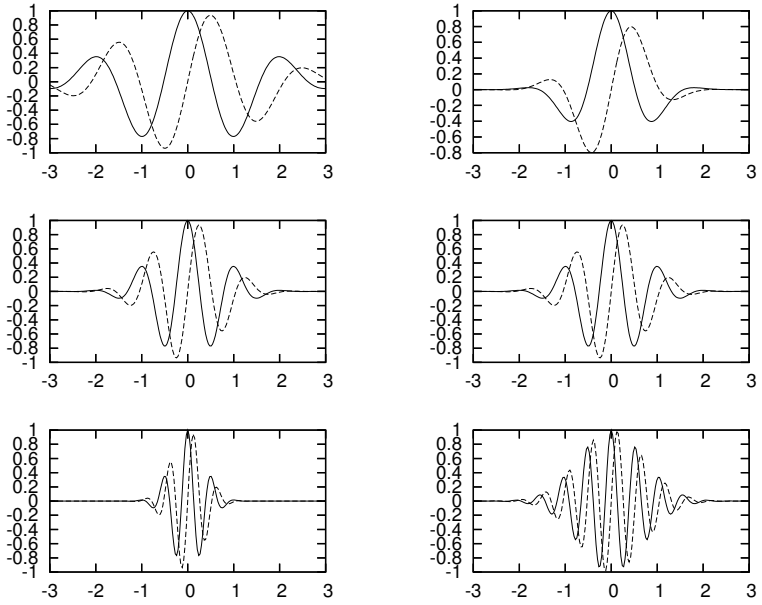


Figure 2.3: GABOR transforms with different window sizes of the complex signal displayed at the top. The result of the GABOR transform contains complex values which are displayed as follows: The darkness of a dot corresponds to the absolute value of the GABOR coefficient and the colour corresponds to the complex argument (the angle or the phase shift). The frequency zero is in the vertical centre. The sign of the frequency corresponds to the orientation of the complex helix. The first image was generated with a discrete GABOR window of size one, the last image with a window which is constant. The first one merely shows the time samples of the signal whereas the last image shows the FOURIER transform when viewed from the side. It can be seen that the more sharp the image is in vertical direction the more blurred is it in the horizontal direction, and vice versa. That is, roughly spoken, the uncertainty principle.



Wavelet transform: MORLET wavelet GABOR transform

Figure 2.4: Different building blocks for the MORLET wavelet transform and the GABOR transform. For easy separation of phase and amplitude these functions are complex valued. The real part is plotted with a solid line and the imaginary part is plotted with a dashed line.

2.1.5 Different filters for analysis and synthesis

We have seen that the condition for a function being a wavelet is rather weak. This is due to the redundancy of the transform. It gives us some freedom in choosing the wavelet. For instance it is possible to invoke different wavelets for the analysis transform (2.1.1) and the synthesis transform (2.1.2). The derivation in Section 2.1.2 needs only small modification and leads to an adaption of (2.1.4):

The transform W_φ inverts ${W_\psi}^*$ (more precisely $\forall f\ W_\varphi \left(w \cdot {W_\psi}^* f\right) \overset{\text{a.e.}}{=} f$ with $w(a) = \frac{1}{c_{\psi,\varphi} \cdot a^2}$) if and only if the constant $c_{\psi,\varphi}$ with $c_{\psi,\varphi} \neq 0$ and

$$c_{\psi,\varphi} = 2 \cdot \pi \cdot \int_{\mathbb{R}_{>0}} \left(\alpha \mapsto \frac{\overline{\widehat{\psi}(\alpha)} \cdot \widehat{\varphi}(\alpha)}{\alpha} \right)$$

exists. Using different wavelets for analysis and synthesis is also quite popular with the discrete wavelet transform. There such wavelet bases are called *biorthogonal*.

2.2 Discrete wavelet transform

In the previous section we have considered the continuous wavelet transform and we have seen how it fits our problem. To be able to transform measured (i.e. sampled) data we must be able to adapt the transformation to discrete data. In general this is not easy for any function transformation and sometimes it is necessary to start from scratch when it comes to a discrete version of a continuous transform.

2.2.1 From CWT to discretised CWT

When switching from the continuous wavelet transform to the discrete one we have to limit and discretise both the time and the scale. While the time discretisation and the time bounds are naturally given by the discrete input signal, the discretisation of the scale is not obvious.

The discrete convolution is a good approximation to the continuous convolution, so the discretised wavelet transform will also split a signal into several bands where each band is obtained by a convolution of the signal with some wavelet function. The most general approach uses a sequence $\boldsymbol{g}$ of n analysis filters and a sequence $\widetilde{\boldsymbol{g}}$ of n corresponding synthesis filters. Then the synthesis and analysis transforms are given by

$$\begin{aligned}(W_{\boldsymbol{g}}{}^* x)_j &= x * \boldsymbol{g}_j^* \\ W_{\widetilde{\boldsymbol{g}}} y &= \sum_{j=1}^{n} y_j * \widetilde{\boldsymbol{g}}_j \qquad .\end{aligned}$$

Note that the sum adds signals rather than scalar values. Since multiple filters are employed this scheme is called a *filter bank*. The condition for *perfect reconstruction* can be derived quite similarly to that of the continuous transform.

$$\begin{aligned}W_{\widetilde{\boldsymbol{g}}}(W_{\boldsymbol{g}}{}^* x) &= \sum_{j=1}^{n} x * \boldsymbol{g}_j^* * \widetilde{\boldsymbol{g}}_j \\ &= x * \sum_{j=1}^{n} \boldsymbol{g}_j^* * \widetilde{\boldsymbol{g}}_j\end{aligned}$$

This means that the superposition of the convolutions of the analysis and synthesis filters must result in the unit impulse, that is

$$\sum_{j=1}^{n} \boldsymbol{g}_j^* * \widetilde{\boldsymbol{g}}_j = \delta \qquad . \tag{2.2.1}$$

This is the same situation as for the continuous transformation but here we are lucky that the neutral element of the discrete convolution does not need a special treatment.

Please note that in the further presentation of the theory we want to substitute $\boldsymbol{g}_j^*$ by $\boldsymbol{g}_j$. We used the adjoint in order to match the idea of correlating a signal with a pattern, but the following derivations are simplified when we incorporate the adjoint into the analysing filters.

2.2.2 From discretised CWT to shift invariant DWT

In the above approach it remains vague how the filters $\boldsymbol{g}_j, \widetilde{\boldsymbol{g}}_j$ may actually be chosen. If we simply sample continuous wavelet functions we will not match the reconstruction property (2.2.1). It is to be assumed that there is no simple or natural way to achieve this. Instead of following this direction we want to introduce a technique that elegantly preserves the invertibility, provides a uniform shape of the wavelet throughout all scales and also addresses efficiency issues.

The trick is to divide the multi-scale transformation into a cascade of two-scale transformations. The two-scale transformation splits the signal into a low-frequency (smooth) part and a high-frequency part by the filters $\boldsymbol{h}_j$ and $\boldsymbol{g}_j$, respectively. The low-frequency output is fed to the next two-scale transform. That is, if the input signal is $\boldsymbol{x}_0$, we compute for n scales ($j \in \{0, \ldots, n-1\}$) iteratively

$$\begin{aligned}\boldsymbol{x}_{j+1} &= \boldsymbol{x}_j * \boldsymbol{h}_j \\ \boldsymbol{y}_{j+1} &= \boldsymbol{x}_j * \boldsymbol{g}_j\end{aligned} \tag{2.2.2}$$

and the result of the transformation are the $n+1$ signals $\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_n, \boldsymbol{x}_n$. If we define

$$\begin{aligned}H_0 = \delta \qquad H_{j+1} &= H_j * \boldsymbol{h}_j \\ G_{j+1} &= H_j * \boldsymbol{g}_j\end{aligned}$$

which can be unrolled to

$$\begin{aligned}H_{j+1} &= \boldsymbol{h}_0 * \boldsymbol{h}_1 * \cdots * \boldsymbol{h}_{j-1} * \boldsymbol{h}_j \\ G_{j+1} &= \boldsymbol{h}_0 * \boldsymbol{h}_1 * \cdots * \boldsymbol{h}_{j-1} * \boldsymbol{g}_j\end{aligned}$$

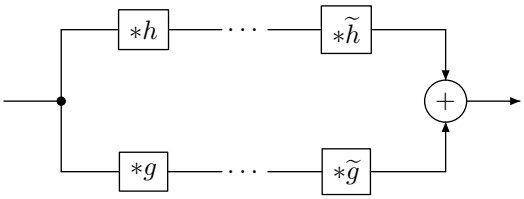


Figure 2.5: One level of a generalised discrete translation invariant wavelet transform: Analysis, interim processing (the dots) and synthesis. The interim processing may consist of more levels of subband coding, de-noising, compression and others.

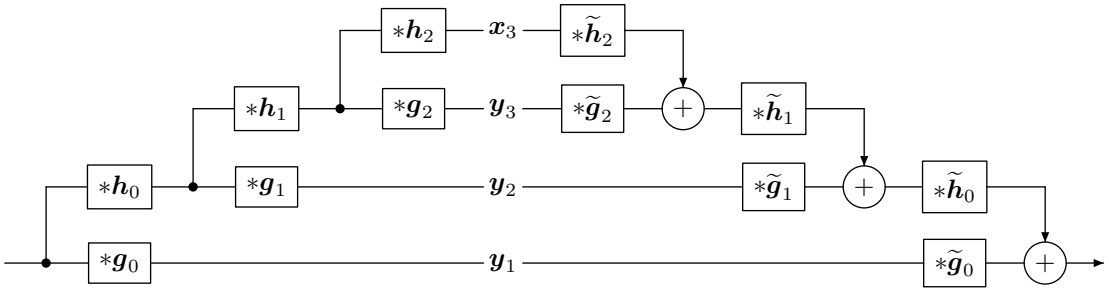


Figure 2.6: A complete generalised translation invariant wavelet transform build from three levels. The large scales (because of longer filters) are at the top.

then we obtain explicit representations for the resulting signals.

$$\begin{aligned}\boldsymbol{x}_j &= \boldsymbol{x}_0 * H_j\\ \boldsymbol{y}_j &= \boldsymbol{x}_0 * G_j\end{aligned}$$

Thus the cascaded transform can be considered as a discretised continuous transform with respect to scale dependent filters $G_1, G_2, \ldots, G_n, H_n$. Usually the signal $\boldsymbol{x}_n$ is a strongly low-pass filtered $\boldsymbol{x}_0$, whereas $\boldsymbol{y}_1$ is $\boldsymbol{x}_0$ filtered through a high-pass, and the other signals are band pass filtered with respect to consecutive frequency bands.

Reconstruction It is convenient to split the whole synthesis transform up into small pieces analogously to the analysis transform. If we can invert each step of the analysis transform we can invert the total transform. Also the synthesis transform shall be linear and translation invariant thus there is no other choice than

$$\boldsymbol{x}_j = \boldsymbol{x}_{j+1} * \widetilde{\boldsymbol{h}}_j + \boldsymbol{y}_{j+1} * \widetilde{\boldsymbol{g}}_j \tag{2.2.3}$$

with still unknown filter sequences $\widetilde{\boldsymbol{h}}$ and $\widetilde{\boldsymbol{g}}$.

Figure 2.5 shows the analysis and synthesis at a single level. Figure 2.6 shows the cascade of such levels establishing the total wavelet transform.

Let (h, g) be the analysis filter pair. We want to find a condition for the synthesis filter pair $(\widetilde{h}, \widetilde{g})$ which reconstructs every input signal x. To this end we fuse (2.2.2) and (2.2.3):

$$\begin{aligned}\forall x \quad & x * h * \widetilde{h} + x * g * \widetilde{g} = x\\ \Leftrightarrow \quad & h * \widetilde{h} + g * \widetilde{g} = \delta\end{aligned}$$

which is in fact a BEZOUT *equation* [Wei05, BEZOUT's Theorem], [Str95, Theorem 25.20]. From Algebra we know that we can find appropriate filters $\widetilde{h}$ and $\widetilde{g}$ if and only if h and g are relatively prime with respect to convolution.

Solutions with minimal filter sizes can be computed efficiently using the EUCLID*ean algorithm* (Section 3.2.1). If $\widetilde{h}$ and $\widetilde{g}$ are particular solutions, the general solution is of the form

$$\begin{aligned}\widetilde{\boldsymbol{h}}_s &= \widetilde{h} + g * s\\ \widetilde{\boldsymbol{g}}_s &= \widetilde{g} - h * s\end{aligned}$$

with $s \in \ell_0(\mathbb{Z})$.

Usually the two-scale transforms are designed in a way that each transform operates on a doubled scale (i.e. halved frequency) relatively to the previous one. This is achieved by up-sampling the filter by a factor of two. The invertibility of the transform is not affected by the up-sampling.

$$\begin{aligned} h * \widetilde{h} + g * \widetilde{g} &= \delta \qquad | \quad \uparrow 2\\ (h * \widetilde{h} + g * \widetilde{g}) \uparrow 2 &= \delta \uparrow 2\\ (h \uparrow 2) * (\widetilde{h} \uparrow 2) + (g \uparrow 2) * (\widetilde{g} \uparrow 2) &= \delta\end{aligned}$$

Since the frequency is halved again and again we obtain an exponential graduation of frequencies.

$$\begin{aligned}\boldsymbol{h}_j &= h \uparrow 2^j\\ \boldsymbol{g}_j &= g \uparrow 2^j\end{aligned}$$

The shift invariant wavelet transform can then be written explicitly as

$$\begin{aligned}\boldsymbol{x}_j &= \boldsymbol{x}_0 * H_j \qquad & H_{j+1} &= h * h \uparrow 2 * h \uparrow 4 * \cdots * h \uparrow 2^{j-1} * h \uparrow 2^j\\ \boldsymbol{y}_j &= \boldsymbol{x}_0 * G_j \qquad & G_{j+1} &= h * h \uparrow 2 * h \uparrow 4 * \cdots * h \uparrow 2^{j-1} * g \uparrow 2^j \quad .\end{aligned}$$

The advantage of this choice is clearly that we only need to design one filter pair, namely h, g. The disadvantage is that the choice is more difficult. Choosing h and g such that H_j and G_j have a smooth shape or even that G_j matches a specific pattern (the main goal of this work) is a challenging problem, which we consider in more detail in Section 4.2 and Section 3.3, respectively.

Because we will need the structure of H_j and G_j frequently in this document, we introduce an operator for it.

2.2.1 Definition (Refinement operator). For a mask h the *refinement operator* $\mathcal{R}_h$ from $\ell_0(\mathbb{Z}) \to \ell_0(\mathbb{Z})$ performs the following:

$$\mathcal{R}_h g = h * (g \uparrow 2) \tag{2.2.4}$$

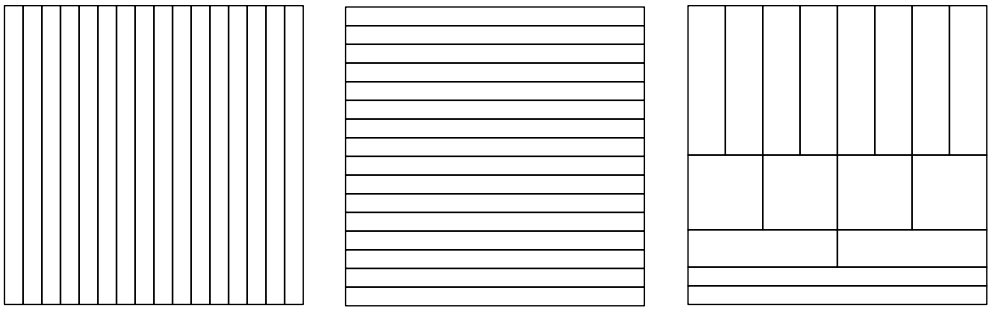
If h has index interval $\{\nu, \ldots, \kappa\}$ then $\mathcal{R}_h^j \delta$ has index interval $\Big\{(2^j - 1) \cdot \nu, \ldots, (2^j - 1) \cdot \kappa\Big\}$.

Using this notation we can write

$$\begin{aligned}H_{j+1} &= \mathcal{R}_h^j h\\ G_{j+1} &= \mathcal{R}_h^j g \qquad .\end{aligned}$$

Note that the kind of transformation described in this section maps a discrete translation of the input signal to a translation of the output sequences. Thus it fills the gap between the better known continuous and discrete transforms (see Section 2.2.3). It was invented several times [Fow05] and got a lot of different names, like *quasi-continuous wavelet transform* [Mae97], *stationary wavelet transform* [NS95, MMOP04], *translation invariant wavelet transform* [CL01, CD95], *shift invariant wavelet transform* [LGO+95b], *algorithmé à trous* [HKMMT89, Dut89, Mal99], *cycle spinning* [CD95], *maximal overlap wavelet transform* [PW00], *redundant wavelet transform* [LB98], *undecimated wavelet transform* [LGO+95a]. Additionally there are variations of that generalisation such as the *over-complete discrete wavelet transform* [Bra03] which employs down-sampling in some transform levels but not all.

These terms reflect the contexts in which the transform was developed or applied. The term *quasi-continuous* refers to the similarity with the continuous wavelet transform. The terms *translation invariant* and *shift invariant* are a bit misleading because they originally mean that the result does not depend on



Time sampled signal Frequency transformed signal Wavelet coefficients

Figure 2.7: HEISENBERG boxes of different signal representations

the translation of the input signal. This applies e.g. to the absolute frequency spectrum. In fact the result of the translation invariant transform depends on the translation of input but in a very natural way - it is also translated. In other words: The translation invariant transformation commutes with the shift operation. The name *algorithmé à trous* describes the up-sampling of the filters, i.e. holes (= French "trous") are added to the filters for higher scales. An alternative implementation is addressed with *cycle spinning*. It means that instead of modifying the wavelet transform the signal is shifted and each shifted version is transformed. Actually this method computes the same wavelet coefficients as the *algorithmé à trous*, but some of them multiple times. The fact that the wavelets of the represented frame overlap is captured by the notion *maximal overlap wavelet transform*. Since "decimation" is an synonym for down-sampling here, *undecimated* means that in contrast to the discrete wavelet transform the down-sampling is left out. Last but not least the terms *redundant* and *over-complete* point to the increased size of transformed data compared with the discrete wavelet transform.

Furthermore a note on a generalisation: Instead of two-scale transforms one can use any larger number m of scales in one transform level. This is useful to insert interim scales if the doubled-scale grid is too coarse. Again you have one low-pass filter $\boldsymbol{h}_j$ for the j-th scale, the low-pass filters are cascaded. Additionally you have $m-1$ high-pass filters $\boldsymbol{g}_{0,j}, \ldots, \boldsymbol{g}_{m-2,j}$ that are specific to the interim frequency bands. The perfect reconstruction condition is quite the same: The greatest common divisor of $\boldsymbol{h}_j, \boldsymbol{g}_{0,j}, \ldots, \boldsymbol{g}_{m-2,j}$ that is $\gcd\left(\ldots \gcd(\boldsymbol{h}_j, \boldsymbol{g}_{0,j}), \ldots \boldsymbol{g}_{m-2,j}\right)$ must be a unit. The EUCLIDean algorithm can be used to determine the greatest common divisor and to solve the BEZOUT equation when applied to the nested applications of $\gcd$.

2.2.3 From shift invariant DWT to DWT

Here we reach the destination of our travel from the continuous to the discrete wavelet transform. For some applications like compression it is not satisfying that the size of data is increased by the transform. It is true that the research on compression based on redundant transforms lead to considerable success with so called *matching pursuit*s, but with matching pursuits perfect reconstruction is hard to achieve. (See Section 2.2.5.) Also for other applications it is of interest to reduce the amount of processed data, rather than to increase it. Even more, the uncertainty principle (Section 2.1.4) gives reasons why an increased amount of data does not necessarily increases the amount of information. That is the shift invariant transform does not increase resolution.

We observe that in the translation invariant DWT independent from the scale there is the same distance between two wavelets that are neighbouring with respect to time. This is what makes this transform translation invariant. It means that the wavelets overlap relatively less in the small scales and much in the large scales. It seems to be natural to use a sampling grid that becomes coarser as the scale increases. Figure 2.7 illustrates the favoured division of the time-frequency plane.

This division together with invertibility can be achieved with the same trick we used to verify invertibility of the translation invariant DWT (Section 2.2.2): We use a cascade of two-scale transformations and

if each two-scale transformation is invertible and preserves the size of the signal, then the whole transformation is invertible and generates data of the size of the input signal.

We achieve the coarser grid at the larger scales by dropping each odd indexed value of the result signals in each step of the transformation. That is, compared with (2.2.2) we add down-sampling by a factor of 2.

$$\begin{aligned} \boldsymbol{x}_{j+1} &= \left(\boldsymbol{x}_j * \boldsymbol{h}_j\right) \downarrow 2 \\ \boldsymbol{y}_{j+1} &= \left(\boldsymbol{x}_j * \boldsymbol{g}_j\right) \downarrow 2 \end{aligned} \tag{2.2.5}$$

This scheme was known as *subband coder* in signal processing even before the theory of the discrete wavelet transform. Because there is no redundancy this transform is also called *critically sampled*.

It is worth looking for explicit formulations for the resulting signals.

$$\begin{aligned} \boldsymbol{x}_j &= ((\ldots((\boldsymbol{x}_0 * \boldsymbol{h}_0) \downarrow 2 * \boldsymbol{h}_1) \downarrow 2 \ldots * \boldsymbol{h}_{j-2}) \downarrow 2 * \boldsymbol{h}_{j-1}) \downarrow 2 \\ &= ((\ldots((\boldsymbol{x}_0 * \boldsymbol{h}_0 * \boldsymbol{h}_1 \uparrow 2) \downarrow 2) \downarrow 2 \ldots * \boldsymbol{h}_{j-2}) \downarrow 2 * \boldsymbol{h}_{j-1}) \downarrow 2 \\ &= ((\ldots(\boldsymbol{x}_0 * \boldsymbol{h}_0 * \boldsymbol{h}_1 \uparrow 2) \downarrow 4 \ldots * \boldsymbol{h}_{j-2}) \downarrow 2 * \boldsymbol{h}_{j-1}) \downarrow 2 \\ &= \left(\boldsymbol{x}_0 * \boldsymbol{h}_0 * \boldsymbol{h}_1 \uparrow 2 * \ldots * \boldsymbol{h}_{j-2} \uparrow 2^{j-2} * \boldsymbol{h}_{j-1} \uparrow 2^{j-1}\right) \downarrow 2^j \\ \boldsymbol{y}_j &= \left(\boldsymbol{x}_0 * \boldsymbol{h}_0 * \boldsymbol{h}_1 \uparrow 2 * \ldots * \boldsymbol{h}_{j-2} \uparrow 2^{j-2} * \boldsymbol{g}_{j-1} \uparrow 2^{j-1}\right) \downarrow 2^j \end{aligned}$$

Again, we want to supply filters which accumulate the operations applied to an input signal over all scales.

$$H_0 = \delta \qquad \begin{aligned} H_{j+1} &= H_j * \boldsymbol{h}_j \uparrow 2^j \\ G_{j+1} &= H_j * \boldsymbol{g}_j \uparrow 2^j \end{aligned}$$

$$\begin{aligned} H_{j+1} &= \boldsymbol{h}_0 * \boldsymbol{h}_1 \uparrow 2 * \cdots * \boldsymbol{h}_{j-1} \uparrow 2^{j-1} * \boldsymbol{h}_j \uparrow 2^j \\ G_{j+1} &= \boldsymbol{h}_0 * \boldsymbol{h}_1 \uparrow 2 * \cdots * \boldsymbol{h}_{j-1} \uparrow 2^{j-1} * \boldsymbol{g}_j \uparrow 2^j \end{aligned}$$

These can be used to describe the transformation shortly.

$$\begin{aligned} \boldsymbol{x}_j &= \left(\boldsymbol{x}_0 * H_j\right) \downarrow 2^j \\ \boldsymbol{y}_j &= \left(\boldsymbol{x}_0 * G_j\right) \downarrow 2^j \end{aligned}$$

In order to provide a uniform shape of all G_j and to apply the theory of *refinable function*s usually the same filter h is used for all $\boldsymbol{h}_j$, and another filter g is used for all $\boldsymbol{g}_j$. In anticipation of our later use of the low-pass filter for refinable functions and because we can avoid $\sqrt{2}$ factors in some filter masks, we want to separate the factor $\sqrt{2}$ from the filters h and g.

$$\begin{aligned} \boldsymbol{h}_j &= \sqrt{2} \cdot h \\ \boldsymbol{g}_j &= \sqrt{2} \cdot g \end{aligned}$$

We end up with a flowchart as in Figure 2.8.

Note that non-uniform filters are used rarely, but nevertheless they can be useful e.g. for generating exponentials. If for some c holds

$$\boldsymbol{h}_j = \left(\mathbf{1}, e^{2^j \cdot c}\right)$$

then we obtain a discretised exponential for H_j.

$$H_j = \left(e^{k \cdot c} : k \in \left\{0, \ldots, 2^j - 1\right\}\right)$$

This is particularly interesting for complex c where the H_j are helixes (waves) of the same frequency with exponential envelopes of different widths.

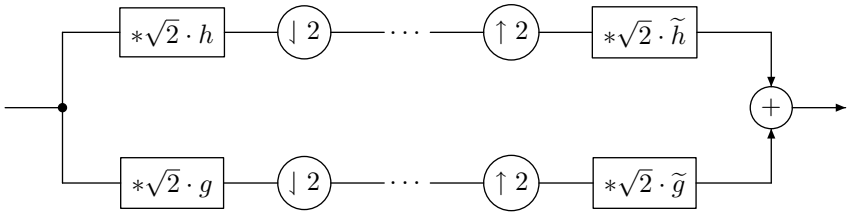


Figure 2.8: One level of a discrete wavelet transform

Discretised continuous wavelet transform	$n \cdot s$
Translation invariant wavelet transform	$n \cdot \log_2 n$
Discrete wavelet transform	n

Table 2.1: Time and space requirements for several kinds of wavelet transform with s scales applied to a signal of size n. The dependency on the filter length is neglected.

Reconstruction

We want to explore how we can restore the signal from the result we obtain from the discrete transform including down-sampling. Further on we want to check when this is possible. In contrast to the translation invariant transform which is not surjective the level-wise reconstruction is obligatory for the critically sampled discrete transform.

2.2.2 Lemma.
Prerequisite. A function sequence f from $\{0, \ldots, n-1\} \to A \to A$ of surjective maps. A is an arbitrary set here but it will be a set of signals in our application.
Claim. The composed map $f_0 \circ f_1 \circ \cdots \circ f_{n-1}$ is injective if and only if each f_j is injective.

Proof.

1. If each f_j is invertible then the inverse of the composition is the reversed composition of the inverted maps: $f_{n-1}^{-1} \circ f_{n-2}^{-1} \circ \cdots \circ f_0^{-1}$.
2. Since each f_j is surjective each value of the range of the composition has an inverse image with respect to f_{n-1}, which in turn has an inverse image with respect to f_{n-2}, and so on. Analogously each composition of surjective maps is surjective.
3. If one of the maps is not invertible consider the one with the least index, say j. Since it is not invertible there must exist two distinct arguments x_0' and x_1' which have the same value y' with respect to f_j ($f_j(x_0') = y'$ and $f_j(x_1') = y'$). Since all maps are surjective x_0' and x_1' must have origins x_0 and x_1 with respect to $f_0 \circ f_1 \circ \cdots \circ f_{j-1}$. Because these functions map uniquely $x_0' \neq x_1'$ implies $x_0 \neq x_1$. Since the values of the maps with indices above j depend exclusively on y' the total composition yield the same value y. That is the composition computes the value y independent from whether the input is x_0 or x_1. Thus the composition is not invertible in this case.

□

For f being the sequence of transformation levels of the analysis transform we conclude that the whole multi-scale transformation is invertible if and only if each of the two-scale transformations is invertible. The lemma does not apply to the translation invariant transform because the steps of this transform produce redundant data and thus are not surjective.

Now we want to find out how to invert a single level of the analysis transform. The down-sampling operation in (2.2.5) is somewhat difficult to handle thus we want to get rid of it. To this end we need the following property of down-sampling.

2.2.3 Lemma.
Prerequisite. $\{x, h\} \subset \ell_0(\mathbb{Z})$

Claim.

$$(x * h) \downarrow k = \sum_{j=0}^{k-1} (x \rightarrow j \downarrow k) * (h \leftarrow j \downarrow k)$$

Proof. In $(x \leftarrow j) \downarrow k$ the coefficients with indices from the residue class $[j]_k$ are preserved. This means that for all j from $\{0, \ldots, k-1\}$ the sequence of translated and down-sampled variants of x contains all coefficients of x. Thus you can reconstruct x from them. The expression $x \downarrow k \uparrow k$ in fact means that the coefficients with multiples of k as indexes are kept and the others are cleared.

$$\begin{aligned} x * h &= \left(\sum_{j=0}^{k-1} x \rightarrow j \downarrow k \uparrow k \leftarrow j \right) * \left(\sum_{l=0}^{k-1} h \leftarrow l \downarrow k \uparrow k \rightarrow l \right) \\ &= \sum_{j=0}^{k-1} \sum_{l=0}^{k-1} ((x \rightarrow j \downarrow k) * (h \leftarrow l \downarrow k)) \uparrow k \rightarrow (l - j) \end{aligned}$$

On down-sampling by a factor of k all filters vanish which have only zeros at the indices which are multiples of k (see (1.2.3)). Since $l - j \in \{1 - k, \ldots, k - 1\}$ this is true for all translations with $l - j \neq 0$.

$$\begin{aligned} (x * h) \downarrow k &= \sum_{j=0}^{k-1} \sum_{l=0}^{k-1} ((x \rightarrow j \downarrow k) * (h \leftarrow l \downarrow k)) \uparrow k \rightarrow (l - j) \downarrow k \\ &= \sum_{j=0}^{k-1} ((x \rightarrow j \downarrow k) * (h \leftarrow j \downarrow k)) \uparrow k \downarrow k \\ &= \sum_{j=0}^{k-1} (x \rightarrow j \downarrow k) * (h \leftarrow j \downarrow k) \end{aligned}$$

□

This connection allows for a different interpretation of the expression $(x * h) \downarrow 2$. Instead of convolving x with h and then down-sample, we can split x and h into their even-indexed and odd-indexed subsequences, convolve corresponding subsequences and add them. This allows to represent one transformation step with a matrix-vector convolution, that is a matrix-vector multiplication where the convolution plays the role of the multiplication.

$$\begin{pmatrix} \boldsymbol{x}_{j+1} \\ \boldsymbol{y}_{j+1} \end{pmatrix} = \sqrt{2} \cdot \begin{pmatrix} h \downarrow 2 & (h \leftarrow 1) \downarrow 2 \\ g \downarrow 2 & (g \leftarrow 1) \downarrow 2 \end{pmatrix} \circledast \begin{pmatrix} \boldsymbol{x}_j \downarrow 2 \\ (\boldsymbol{x}_j \rightarrow 1) \downarrow 2 \end{pmatrix}$$

According to [DS98] we introduce the notions of polyphase and modulation matrices. They lead straightforwardly to the reconstruction conditions as known from [Dau92].

2.2.4 Definition (Polyphase matrix). For discrete signals h, g we use the abbreviations

$$\begin{aligned} h_{\mathsf{e}} &= h \downarrow 2 & h_{\mathsf{o}} &= (h \leftarrow 1) \downarrow 2 \\ g_{\mathsf{e}} &= g \downarrow 2 & g_{\mathsf{o}} &= (g \leftarrow 1) \downarrow 2 \end{aligned}$$

which let us define the *polyphase matrix* P as follows:

$$P = \sqrt{2} \cdot \begin{pmatrix} h_{\mathsf{e}} & h_{\mathsf{o}} \\ g_{\mathsf{e}} & g_{\mathsf{o}} \end{pmatrix} \quad .$$

The Figure 2.9 shows how the discrete wavelet transform can be implemented in terms of the polyphase matrix.

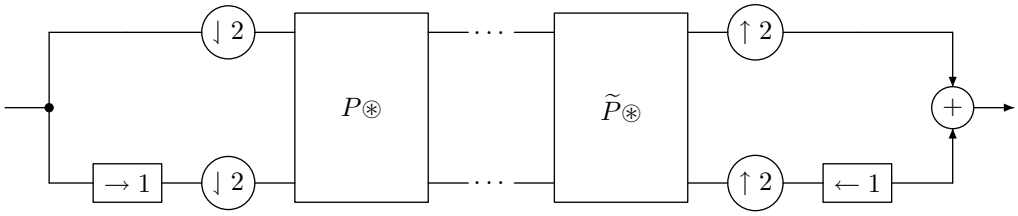


Figure 2.9: Discrete wavelet transform with a polyphase matrix

2.2.5 Definition (Modulation matrix). For a polyphase matrix as given above the corresponding *modulation matrix* is

$$\begin{aligned} h_- &= h_{\mathrm{e}} \uparrow 2 - (h_{\mathrm{o}} \uparrow 2) \to 1 \\ g_- &= g_{\mathrm{e}} \uparrow 2 - (g_{\mathrm{o}} \uparrow 2) \to 1 \\ M &= \begin{pmatrix} h & h_- \\ g & g_- \end{pmatrix} \qquad . \end{aligned}$$

The signal h_- differs from h only in the signs of the odd indexed coefficients.

Obviously both polyphase and modulation matrix carry the same information. Very similar statements can be formulated for either representation. None of them is really superior to the other one, we will use both of them as it is convenient.

Since we converted a transformation step into a matrix-vector multiplication we get the idea that the inverse transform is just the solution of a system of two linear equations. However the problem is that a convolutional division (which is in fact the solution of linear difference equation with constant coefficients) has the risk of numerical instabilities. Thus we have to explore in which cases we can invert without division.

2.2.6 Lemma (Invertibility of matrices over rings).
Prerequisite. $(R, 0, 1, +, \cdot)$ is a ring and M a matrix, $M \in R^{n\times n}$.
Claim. An inverse matrix $M^{-1} \in R^{n\times n}$ exists ($M \cdot M^{-1} = I$) if and only if the determinant of M is invertible (a so called *unit*).

Proof.

1. M is invertible "⇒" $\det M$ is a unit

$$\begin{aligned} 1 &= \det I \\ &= \det\left(M \cdot M^{-1}\right) \qquad \text{product of determinants} \\ &= \det M \cdot \det M^{-1} \end{aligned}$$

   The unit element 1 from R can only be factorised into units of that ring, thus $\det M$ must be a unit.
2. $\det M$ is a unit "⇒" M is invertible
   If M is considered as row vector of column vectors $(m_1, \ldots, m_n)$ each element of M^{-1} can be calculated by CRAMER*'s rule* [Str95, Theorem 18.8]:

$$(M^{-1})_{i,j} = (\det M)^{-1} \cdot \det(m_1, \ldots, m_{j-1}, e_i, m_{j+1}, \ldots, m_n)$$

   where e_i is the unit vector where all components are zero except the ith one which is equal to 1. The inversion of $\det M$ is possible because $\det M$ is a unit according to the assumption.

□

2.2.7 Corollary. Consider the set of discrete signals of finite length: Since one-length discrete signals are the only ones that are invertible with respect to convolution, a polyphase matrix P is invertible with finite length signals if and only if the convolutional determinant of P is a monomial.

$$\det P = c \cdot \delta \rightarrow k$$

where

$$\det P = \frac{1}{2} \cdot (h_{\mathsf{e}} * g_{\mathsf{o}} - h_{\mathsf{o}} * g_{\mathsf{e}})$$

2.2.8 Corollary. If the polyphase matrix is invertible then h_{e} and h_{o} must be relatively prime. If h_{e} and h_{o} are not relatively prime, you can extract a common divisor which is not a monomial. This divisor can also be extracted from the determinant.

Let s be such a common divisor

$$h_{\mathsf{e}} = s * h'_{\mathsf{e}}$$
$$h_{\mathsf{o}} = s * h'_{\mathsf{o}}$$

then we obtain

$$\begin{aligned}\det \begin{pmatrix} h_{\mathsf{e}} & h_{\mathsf{o}} \\ g_{\mathsf{e}} & g_{\mathsf{o}} \end{pmatrix} &= \det \begin{pmatrix} s * h'_{\mathsf{e}} & s * h'_{\mathsf{o}} \\ g_{\mathsf{e}} & g_{\mathsf{o}} \end{pmatrix} \\ &= s * \det \begin{pmatrix} h'_{\mathsf{e}} & h'_{\mathsf{o}} \\ g_{\mathsf{e}} & g_{\mathsf{o}} \end{pmatrix} \quad .\end{aligned}$$

Since s is not a monomial, the determinant is not, as well and thus the matrix cannot be inverted.

2.2.9 Definition (Complementary filter pair). A pair of filters (h, g) is called *complementary* [DS98] if and only if

$$\det \begin{pmatrix} h_{\mathsf{e}} & h_{\mathsf{o}} \\ g_{\mathsf{e}} & g_{\mathsf{o}} \end{pmatrix} = \frac{1}{2} \cdot \delta \quad .$$

2.2.10 Corollary. If a pair of filters (h, g) is complementary, the polyphase matrix can be inverted. By CRAMER's rule we obtain

$$P^{-1} = \sqrt{2} \cdot \begin{pmatrix} g_{\mathsf{o}} & -h_{\mathsf{o}} \\ -g_{\mathsf{e}} & h_{\mathsf{e}} \end{pmatrix} \quad .$$

2.2.11 Corollary. If a pair of filters (h, g) is complementary, a single level of the discrete wavelet transformation can be inverted by

$$\begin{aligned}\begin{pmatrix} \boldsymbol{x}_j \downarrow 2 \\ (\boldsymbol{x}_j \rightarrow 1) \downarrow 2 \end{pmatrix} &= \sqrt{2} \cdot \begin{pmatrix} g_{\mathsf{o}} & -h_{\mathsf{o}} \\ -g_{\mathsf{e}} & h_{\mathsf{e}} \end{pmatrix} \circledast \begin{pmatrix} \boldsymbol{x}_{j+1} \\ \boldsymbol{y}_{j+1} \end{pmatrix} \\ &= \sqrt{2} \cdot \begin{pmatrix} (g \leftarrow 1) \downarrow 2 & -(h \leftarrow 1) \downarrow 2 \\ -g \downarrow 2 & h \downarrow 2 \end{pmatrix} \circledast \begin{pmatrix} \boldsymbol{x}_{j+1} \\ \boldsymbol{y}_{j+1} \end{pmatrix} \quad .\end{aligned}$$

Because we can reconstruct a signal from its even and odd indexed coefficients by

$$x = (x \downarrow 2) \uparrow 2 + ((x \rightarrow 1) \downarrow 2) \uparrow 2 \leftarrow 1$$

we derive a compact formula from the up-sampled expression

$$\begin{pmatrix} \boldsymbol{x}_j \downarrow 2 \uparrow 2 \\ (\boldsymbol{x}_j \rightarrow 1) \downarrow 2 \uparrow 2 \end{pmatrix} = \sqrt{2} \cdot \begin{pmatrix} (g \leftarrow 1) \downarrow 2 \uparrow 2 & -(h \leftarrow 1) \downarrow 2 \uparrow 2 \\ -g \downarrow 2 \uparrow 2 & h \downarrow 2 \uparrow 2 \end{pmatrix} \circledast \begin{pmatrix} \boldsymbol{x}_{j+1} \uparrow 2 \\ \boldsymbol{y}_{j+1} \uparrow 2 \end{pmatrix}$$

by vectorially convolving both sides with $(\delta, \delta \leftarrow 1)$ from left.

$$\begin{aligned}\boldsymbol{x}_j &= \sqrt{2}\cdot\Big((g \leftarrow 1)_{\_}, h_{\_} \leftarrow 1\Big) \circledast \begin{pmatrix} \boldsymbol{x}_{j+1} \uparrow 2 \\ \boldsymbol{y}_{j+1} \uparrow 2 \end{pmatrix} \\ &= \sqrt{2}\cdot\Big((g \leftarrow 1)_{\_} * (\boldsymbol{x}_{j+1} \uparrow 2) + (h_{\_} \leftarrow 1) * (\boldsymbol{y}_{j+1} \uparrow 2)\Big)\end{aligned}$$

2.2.12 Notation. If a pair of filters is (h, g) complementary, then the pair of filters for reconstruction (the one for which $\boldsymbol{x}_j = \sqrt{2}\cdot(\widetilde{h} * (\boldsymbol{x}_{j+1} \uparrow 2) + \widetilde{g} * (\boldsymbol{y}_{j+1} \uparrow 2))$ hold) is called the *dual filter pair* $(\widetilde{h}, \widetilde{g})$.

$$\begin{aligned}\widetilde{h} &= (g \leftarrow 1)_{\_} \\ \widetilde{g} &= h_{\_} \leftarrow 1 \\ P^{-1} &= \sqrt{2}\cdot\begin{pmatrix} \widetilde{h}_{\mathrm{e}} & \widetilde{g}_{\mathrm{e}} \\ \widetilde{h}_{\mathrm{o}} \rightarrow 1 & \widetilde{g}_{\mathrm{o}} \rightarrow 1 \end{pmatrix}\end{aligned}$$

2.2.13 Theorem.
Claim. The modulation matrix (Definition 2.2.5) is regular if and only if the polyphase matrix is regular. It holds

$$M^{-1} = \begin{pmatrix} \widetilde{h} & \widetilde{g} \\ \widetilde{h}_{\_} & \widetilde{g}_{\_} \end{pmatrix}$$

Proof. You can transform between a polyphase matrix and the corresponding modulation matrix by an orthogonal transformation. We expand the slightly modified primal and dual polyphase matrices into representations using the modulation matrices.

$$\begin{pmatrix} h_{\mathrm{e}} \uparrow 2 & h_{\mathrm{o}} \uparrow 2 \rightarrow 1 \\ g_{\mathrm{e}} \uparrow 2 & g_{\mathrm{o}} \uparrow 2 \rightarrow 1 \end{pmatrix} = \begin{pmatrix} h & h_{\_} \\ g & g_{\_} \end{pmatrix}\cdot\frac{1}{2}\cdot\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

$$\begin{pmatrix} \widetilde{h}_{\mathrm{e}} \uparrow 2 & \widetilde{g}_{\mathrm{e}} \uparrow 2 \\ \widetilde{h}_{\mathrm{o}} \uparrow 2 \rightarrow 1 & \widetilde{g}_{\mathrm{o}} \uparrow 2 \rightarrow 1 \end{pmatrix} = \frac{1}{2}\cdot\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}\cdot\begin{pmatrix} \widetilde{h} & \widetilde{g} \\ \widetilde{h}_{\_} & \widetilde{g}_{\_} \end{pmatrix}$$

We know that the dual polyphase matrix is the inverse of the primal polyphase matrix, thus if we replace them with their modulation matrix representations we obtain that the convolutional product of the primal modulation matrix and its proposed inverse is indeed the identity matrix.

$$\begin{aligned}\begin{pmatrix} \delta & 0 \\ 0 & \delta \end{pmatrix} &= \sqrt{2}\cdot\begin{pmatrix} h_{\mathrm{e}} \uparrow 2 & h_{\mathrm{o}} \uparrow 2 \rightarrow 1 \\ g_{\mathrm{e}} \uparrow 2 & g_{\mathrm{o}} \uparrow 2 \rightarrow 1 \end{pmatrix} \circledast \sqrt{2}\cdot\begin{pmatrix} \widetilde{h}_{\mathrm{e}} \uparrow 2 & \widetilde{g}_{\mathrm{e}} \uparrow 2 \\ \widetilde{h}_{\mathrm{o}} \uparrow 2 \rightarrow 1 & \widetilde{g}_{\mathrm{o}} \uparrow 2 \rightarrow 1 \end{pmatrix} \qquad \text{Definition 2.2.12} \\ &= \frac{1}{2}\cdot\begin{pmatrix} h & h_{\_} \\ g & g_{\_} \end{pmatrix}\cdot\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \circledast \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}\cdot\begin{pmatrix} \widetilde{h} & \widetilde{g} \\ \widetilde{h}_{\_} & \widetilde{g}_{\_} \end{pmatrix} \\ &= \begin{pmatrix} h & h_{\_} \\ g & g_{\_} \end{pmatrix} \circledast \begin{pmatrix} \widetilde{h} & \widetilde{g} \\ \widetilde{h}_{\_} & \widetilde{g}_{\_} \end{pmatrix}\end{aligned}$$

□

2.2.14 Corollary. The filters g and h form a *perfect reconstruction* filter bank if and only if the determinant of the modulation matrix $\begin{pmatrix} h & h_{\_} \\ g & g_{\_} \end{pmatrix}$ is a certain monomial, more precisely

$$h * g_{\_} - h_{\_} * g = \delta \rightarrow 1 \quad .$$

This is a conclusion of the Definition 2.2.5 of the modulation matrix, the polyphase matrix conversion used in Theorem 2.2.13 and the fact that the determinant of a polyphase matrix of a complementary filter pair is $\frac{1}{2}\cdot\delta$ (Definition 2.2.9).

2.2.15 Remark. From expanding $M^{-1} \circledast M = I$, the condition for perfect reconstruction in the presence of down-sampling can be given in a style closer to the condition given for translation invariant transformation in Section 2.2.2.

$$\begin{aligned} h * \widetilde{h} + g * \widetilde{g} &= \delta \\ h * \widetilde{h}_{-} + g * \widetilde{g}_{-} &= 0 \end{aligned}$$

That is, a filter pair (h, g) suitable for the discrete wavelet transform with down-sampling can also be used for the translation invariant transform. The second equation is additional for the down-sampled transform. Roughly spoken it means that no information is lost by down-sampling.

2.2.16 Lemma.
Prerequisite. The primal filter pair (h, g) is complementary, and $(\widetilde{h}, \widetilde{g})$ is the dual filter pair.
Claim.

$$\left(h * \widetilde{h}\right) \downarrow 2 = \frac{1}{2} \cdot \delta \tag{2.2.6}$$

$$(g * \widetilde{g}) \downarrow 2 = \frac{1}{2} \cdot \delta \tag{2.2.7}$$

$$(h * \widetilde{g}) \downarrow 2 = 0 \tag{2.2.8}$$

$$\left(g * \widetilde{h}\right) \downarrow 2 = 0$$

Proof.

$$P = \sqrt{2} \cdot \begin{pmatrix} h_{\mathrm{e}} & h_{\mathrm{o}} \\ g_{\mathrm{e}} & g_{\mathrm{o}} \end{pmatrix}$$

Because of Definition 2.2.12

$$\begin{aligned} P^{-1} &= \sqrt{2} \cdot \begin{pmatrix} \widetilde{h}_{\mathrm{e}} & \widetilde{g}_{\mathrm{e}} \\ \widetilde{h}_{\mathrm{o}} \to 1 & \widetilde{g}_{\mathrm{o}} \to 1 \end{pmatrix} \\ I &= P \circledast P^{-1} \\ \frac{1}{2} \cdot \begin{pmatrix} \delta & 0 \\ 0 & \delta \end{pmatrix} &= \begin{pmatrix} h_{\mathrm{e}} & h_{\mathrm{o}} \\ g_{\mathrm{e}} & g_{\mathrm{o}} \end{pmatrix} \circledast \begin{pmatrix} \widetilde{h}_{\mathrm{e}} & \widetilde{g}_{\mathrm{e}} \\ \widetilde{h}_{\mathrm{o}} \to 1 & \widetilde{g}_{\mathrm{o}} \to 1 \end{pmatrix} \\ &= \begin{pmatrix} h_{\mathrm{e}} * \widetilde{h}_{\mathrm{e}} + h_{\mathrm{o}} * \widetilde{h}_{\mathrm{o}} \to 1 & h_{\mathrm{e}} * \widetilde{g}_{\mathrm{e}} + h_{\mathrm{o}} * \widetilde{g}_{\mathrm{o}} \to 1 \\ g_{\mathrm{e}} * \widetilde{h}_{\mathrm{e}} + g_{\mathrm{o}} * \widetilde{h}_{\mathrm{o}} \to 1 & g_{\mathrm{e}} * \widetilde{g}_{\mathrm{e}} + g_{\mathrm{o}} * \widetilde{g}_{\mathrm{o}} \to 1 \end{pmatrix} \\ &= \begin{pmatrix} \left(h * \widetilde{h}\right) \downarrow 2 & (h * \widetilde{g}) \downarrow 2 \\ \left(g * \widetilde{h}\right) \downarrow 2 & (g * \widetilde{g}) \downarrow 2 \end{pmatrix} \end{aligned}$$

□

2.2.17 Remark (Orthogonal filter banks). If it holds $\widetilde{h} = h^*$ then we have an *orthogonal filter bank*. In this case from Lemma 2.2.16 follows orthogonality between h and its even translates in the sense that

$$\begin{aligned} \frac{1}{2} \cdot \delta &= \left(h * \widetilde{h}\right) \downarrow 2 \\ &= \left(h * h^*\right) \downarrow 2 \\ \langle h, h \to 2 \cdot k \rangle &= \begin{cases} \frac{1}{2} & : k = 0 \\ 0 & : \text{else} \end{cases} . \end{aligned}$$

For the high-pass filter of an orthogonal filter bank holds $g = {h^*}_{-} \to 1$.

Transformations between perfect reconstruction filter banks

In this section we want to explore operations on filter banks that do not interfere with the perfect reconstruction property.

2.2.18 Theorem.
Prerequisite. Let (h, g) be a complementary filter pair. The filter h contains the factor s, i.e. there is some h' with $h = h' * s$ and the filter g' is defined by $g' = g * s\_$.
Claim. The filter pair (h', g') is complementary as well. Intuitively spoken, factors can be moved from one filter to the other including alternating the filter coefficients' signs, while preserving the perfect reconstruction property.

Proof. We start with Corollary 2.2.14.

$$\begin{aligned}\delta \to 1 &= h\_ * g - h * g\_ \\ &= h'\_ * s\_ * g - h' * s * g\_ \\ &= h'\_ * g' - h' * g'\_\end{aligned}$$

□

We will need this property when considering vanishing moments in Section 4.3.2.

2.2.19 Remark. We can also compute the dual filter pair $(\widetilde{h'}, \widetilde{g'})$ for the filters of the previous theorem. Due to Definition 2.2.12 it is given by

$$\begin{aligned}\widetilde{h'} &= (g' \leftarrow 1)\_ \\ &= (g * s\_ \leftarrow 1)\_ \\ &= (g \leftarrow 1)\_ * (s\_)\_ \\ &= \widetilde{h} * s \\ \widetilde{g'} &= h'\_ \leftarrow 1 \\ \widetilde{g'} * s\_ &= h'\_ * s\_ \leftarrow 1 \\ &= h\_ \leftarrow 1 \\ &= \widetilde{g} \qquad .\end{aligned}$$

This means if the factor s is moved from h to g with alternated signs then $s\_$ is moved from $\widetilde{g}$ to $\widetilde{h}$.

Another transformation which converts a perfect reconstruction filter bank into another one is the lifting scheme which is described in Section 3.2. The lifting operation fixes one of the filters h and g while it changes the other one.

2.2.4 Multi-scale analysis

We have considered discrete versions of the wavelet transform as discrete convolutions, so far. We want to bridge from the discrete wavelet transform to the continuous one in a more direct way. Is it possible to replace the discrete input signal x by a real function f and is there a continuous wavelet function ψ such that the discrete wavelet transform with filters h and g is simply a continuous wavelet transform sampled at several discrete points? Actually, this is possible.

We want to interpret the coefficients of the interim signals of the discrete wavelet transform $\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots$ as coefficients of appropriately translated and scaled versions of some continuous function φ. The coefficient vector $\boldsymbol{x}_j$ shall represent the function $\boldsymbol{x}_j * \varphi \uparrow 2^j$. The functions representable by $\boldsymbol{x}_j$ shall be denoted with V_j. We expect that all functions that can be represented at a coarse scale can also be represent at a finer scale, i.e. $V_{j+1} \subset V_j$. If at some scale the high-pass coefficients vanishes ($\boldsymbol{y}_j = 0$), then a function at this scale can also be represented at the next coarser scale. These claims are subsumed in the term multi-scale analysis. [Mal99]

In order to define the multi-scale analysis we need the notion of a RIESZ basis, especially a RIESZ basis of integer translates of a function.

2.2.20 Definition (RIESZ basis). A sequence f from $\mathbb{Z} \to \mathbb{H}$ of functions f_k from a HILBERT space $\mathbb{H}$ is called a RIESZ *basis* of $\mathbb{H}$ if the set of linear combinations of f_k is dense in $\mathbb{H}$ and the norm in $\mathbb{H}$ is equivalent to the ℓ_2 norm of expansion coefficient sequences, that is

$$\exists \{A, B\} \subset \mathbb{R}_{>0} \quad \forall c \in \ell_2(\mathbb{R}) \quad A \cdot \|c\|_2^2 \leq \left\| \sum_{k \in \mathbb{Z}} c_k \cdot f_k \right\|_{\mathbb{H}}^2 \leq B \cdot \|c\|_2^2 \quad .$$

2.2.21 Definition. If the sequence of translates $(\varphi \to k : k \in \mathbb{Z})$ from a HILBERT space $\mathbb{H}$ forms a RIESZ basis of the closure of its linear span, we say that φ has the RIESZ basis property $\mathcal{B}(\varphi)$, that is

$$\mathcal{B}(\varphi) \quad \Leftrightarrow \quad \exists \{A, B\} \subset \mathbb{R}_{>0} \quad \forall c \in \ell_2(\mathbb{R}) \quad A \cdot \|c\|_2^2 \leq \|c * \varphi\|_{\mathbb{H}}^2 \leq B \cdot \|c\|_2^2 \quad .$$

2.2.22 Definition (multi-scale analysis, multi-resolution analysis). A *multi-scale analysis* or *multi-resolution analysis* of $\mathcal{L}_2(\mathbb{R})$ is a sequence $\left(V_j : j \in \mathbb{Z}\right)$ of spaces V_j with respect to a function φ if the following holds.

	$\overline{\bigcup_{j \in \mathbb{Z}} V_j} = \mathcal{L}_2(\mathbb{R})$	upper limit
	$\bigcap_{j \in \mathbb{Z}} V_j = \{0\}$	lower limit
$\forall j$	$V_{j+1} \subset V_j$	nesting of spaces
$\forall j$	$f \in V_{j+1} \Leftrightarrow f \downarrow 2 \in V_j$	scales of spaces
	$V_0 = \overline{\left\{c * \varphi \ : \ c \in \ell_0(\mathbb{Z})\right\}}$	
	$\mathcal{B}(\varphi)$	The integral translates of φ must form a RIESZ basis.

2.2.23 Remark.

1. The spaces V_j form a chain of inclusion.

$$\{0\} \subset \cdots \subset V_2 \subset V_1 \subset V_0 \subset V_{-1} \subset V_{-2} \subset \cdots \subset \mathcal{L}_2(\mathbb{R})$$

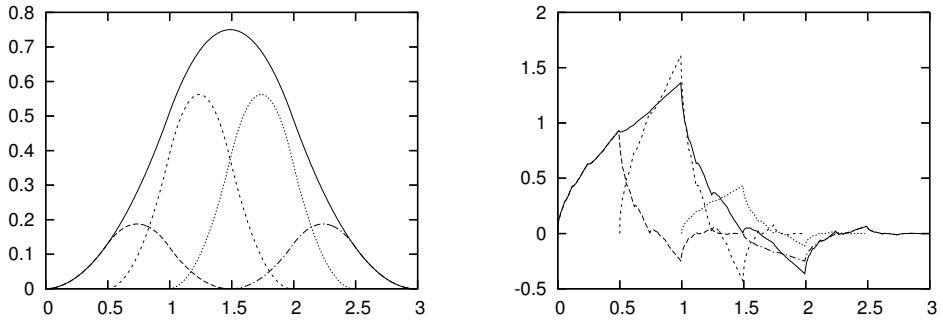
2. From the scaling relation between the spaces it follows:

$$V_j = \overline{\left\{(c * \varphi) \uparrow 2^j \ : \ c \in \ell_0(\mathbb{Z})\right\}} \quad .$$

3. Because of the nesting of the spaces V_j and because of the finer scales which V_j has in addition to V_{j+1}, it must be possible to represent φ in terms of small dilated versions of itself.

$$\begin{aligned} & V_1 \supset \left\{(c * \varphi) \uparrow 2 \ : \ c \in \ell_0(\mathbb{Z})\right\} \\ \Rightarrow \quad & \varphi \uparrow 2 \in V_1 \\ & V_1 \subset V_0 \\ \Rightarrow \quad & \varphi \uparrow 2 \in V_0 \\ & V_0 = \overline{\left\{c * \varphi \ : \ c \in \ell_0(\mathbb{Z})\right\}} \\ \Rightarrow \exists c \in \ell_2(\mathbb{Z}) \quad & \varphi \uparrow 2 \overset{\text{a.e.}}{=} c * \varphi \end{aligned}$$

   The last line is known as the *two-scale equation* or the *refinement relation*.
   We only know, that $\varphi \uparrow 2 \in V_0$, but we cannot assert that $\varphi \uparrow 2 \in \left\{c * \varphi \ : \ c \in \ell_0(\mathbb{Z})\right\}$. Thus in order to represent $\varphi \uparrow 2$ in terms of φ we may need limit processes. Consequently we need the RIESZ basis property in the last implication: Since $\varphi \in \mathcal{L}_2(\mathbb{R})$ we can bound the norm of c and thus the series implied by the convolution with the signal c of potentially infinite size can be evaluated in the $\mathcal{L}_2(\mathbb{R})$ sense.
4. The operator $f \mapsto (c * f) \downarrow 2$ is linear, thus φ is an eigenfunction of this operator with respect to the eigenvalue 1.



CDF-3,m DAUBECHIES-2

Figure 2.10: Refinement of a quadratic B-spline and the orthogonal DAUBECHIES-2 generator

5. Methods for approximating the shape of a refinable function are discussed in Section 3.1.1.

2.2.24 Definition. Let h be a finitely supported signal, $h \in \ell_0(\mathbb{Z})$. A function φ with $\varphi \in \mathbb{R} \to \mathbb{R}$ which satisfies a certain self-similarity condition

$$\varphi = 2 \cdot (h * \varphi) \downarrow 2 \tag{2.2.9}$$

$$\text{alternatively} \qquad \forall t \in \mathbb{R} \quad \varphi(t) = 2 \cdot \sum_{k \in \mathbb{Z}} h_k \cdot \varphi(2 \cdot t - k)$$

is called a *refinable function* with respect to the *refinement mask* h. Within a multi-scale analysis it is called the *scaling function* or the *generator*.

If h is used for the analysis transform then φ is called the *primal generator*. Analogously $\widetilde{\varphi}$, which is the refinable function with respect to the synthesis generator mask $\widetilde{h}$ is the *dual generator*.

The Figure 2.10 illustrates how a refinable function can be assembled by shrunken versions of itself.

2.2.25 Remark. The term "refinable function" defines only a relation, not a construction. Until here it is not clear what refinable functions for a mask exist (none, one, two, more) and whether there is only one refinement mask for a refinable function. Section 3.1.1 shows ways of construction a refinable function from a mask.

In contrast to most other wavelet related works we do not require that the mask sum is 1. This allows gives us a little more freedom but is certainly more dangerous.

2.2.26 Definition. If φ is refinable with respect to h and (h, g) is a complementary filter pair, then the function ψ with $\psi \in \mathbb{R} \to \mathbb{R}$ which is a linear combination of integral translates of φ, i.e.

$$\psi = 2 \cdot (g * \varphi) \downarrow 2 \tag{2.2.10}$$

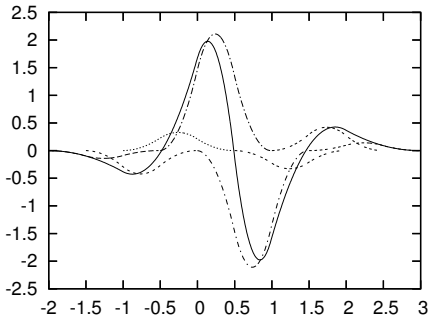
$$\text{alternatively} \qquad \forall t \in \mathbb{R} \quad \psi(t) = 2 \cdot \sum_{k \in \mathbb{Z}} g_k \cdot \varphi(2 \cdot t - k)$$

is called a *wavelet function* of a multi-scale analysis.

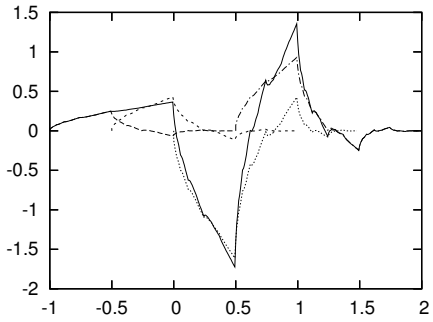
If g is used for the analysis transform then ψ is called the *primal wavelet*. Analogously the wavelet with respect to the dual wavelet mask $\widetilde{\psi}$ is called the *dual wavelet function.*

Figure 2.11 shows how a discrete wavelet is assembled of translated versions of the generator function.

2.2.27 Lemma.
Prerequisite. The function φ is refinable with respect to h and it is integrable.
Claim. If the integral of φ is different from zero then its refinement mask sums up to 1.



CDF-3,3



DAUBECHIES-2

Figure 2.11: Build a wavelet from a generator: Wavelets of the biorthogonal wavelet basis CDF-3,3 and of the orthogonal DAUBECHIES-2 basis

Proof.

$$\begin{aligned}
\varphi &= 2 \cdot (h * \varphi) \downarrow 2 \\
\int_{\mathbb{R}} \varphi &= 2 \cdot \int_{\mathbb{R}} \big((h * \varphi) \downarrow 2\big) \\
&= \int_{\mathbb{R}} (h * \varphi) \\
&= \sum h \cdot \int_{\mathbb{R}} \varphi \\
1 &= \sum h \quad \vee \quad \int_{\mathbb{R}} \varphi = 0
\end{aligned}$$

□

2.2.28 Lemma.
Claim. If a function φ is differentiable and refinable with respect to h then its derivative is refinable with respect to $2 \cdot h$.

Proof.

$$\begin{aligned}
\varphi &= 2 \cdot (h * \varphi) \downarrow 2 \\
\varphi' &= 2 \cdot ((h * \varphi) \downarrow 2)' \\
&= 2 \cdot 2 \cdot ((h * \varphi)' \downarrow 2) \\
&= 2 \cdot (2 \cdot h * \varphi') \downarrow 2
\end{aligned}$$

□

Section 2.2.5 gives an example of a refinable function which both has integral zero and is the derivative of another refinable function.

2.2.29 Remark. According to a suitably generalised notion of refinement to distributions the DIRAC impulse is refinable with respect to δ. Then its j-th derivative must be refinable with respect to $2^j \cdot \delta$. The power function $t \mapsto t^j$ is refinable with respect to $2^{-j-1} \cdot \delta$. The truncated power function $t \mapsto t_+^j$ is refinable with respect to the same mask. This is related because the truncated power functions are intuitively antiderivatives of the DIRAC impulse.

2.2.30 Theorem.
Prerequisite. Let φ_0 and φ_1 be refinable functions with respect to the masks h_0 and h_1, respectively. The convolution of both functions $\varphi_0 * \varphi_1$ must exist.
Claim. The function $\varphi_0 * \varphi_1$ is refinable with respect to $h_0 * h_1$.

Proof. We show that $\varphi_0 * \varphi_1$ fulfils the refinement condition with respect to $h_0 * h_1$:

$$\begin{aligned}
\varphi_0 * \varphi_1 &= \big(2 \cdot (h_0 * \varphi_0) \downarrow 2\big) * \big(2 \cdot (h_1 * \varphi_1) \downarrow 2\big) \\
&= 4 \cdot \big((h_0 * \varphi_0) \downarrow 2\big) * \big((h_1 * \varphi_1) \downarrow 2\big) \\
&\overset{(1.2.5)}{=} 2 \cdot \big((h_0 * \varphi_0) * (h_1 * \varphi_1)\big) \downarrow 2 \\
&= 2 \cdot (h_0 * h_1 * \varphi_0 * \varphi_1) \downarrow 2
\end{aligned}$$

□

2.2.31 Lemma.
Claim. The refinement relation remains valid after discretisation.

Proof.

$$\begin{aligned}
\varphi &= 2 \cdot (h * \varphi) \downarrow 2 \\
Q\varphi &= Q(2 \cdot (h * \varphi) \downarrow 2) \\
&= 2 \cdot (Q(h * \varphi)) \downarrow 2 \\
&= 2 \cdot (h * Q\varphi) \downarrow 2
\end{aligned}$$

□

It will be called the *discrete refinement relation.*

Since we need the term $(h * x) \downarrow 2$ frequently including nested forms we will introduce an operator for it.

2.2.32 Definition. For a given mask h from $\ell_0(\mathbb{Z})$ we define the linear operator $\mathcal{T}_h$ as follows

$$\mathcal{T}_h x = (h * x) \downarrow 2$$

2.2.33 Remark. The operators $\mathcal{R}_h$ and $\mathcal{T}_h$ are similar. Intuitively spoken $\mathcal{R}_h$ let a filter mask grow from inside, that is the mask is stretched and then convolved with h. By way of contrast $\mathcal{T}_h$ applies the filter h to the outside of the input filter and then the result is shrunk. Both operators are designed in such a way that simple iteration (operator power) makes sense. For instance if we let $\mathcal{R}_h$ grow from the outside we would need a parameter specifying the current level of refinement.

2.2.34 Remark. $\mathcal{T}_h$ is a linear operator which can be represented by an infinite matrix. We will later consider eigenvalues and eigenvectors. But the length of an eigenvector must not be changed by $\mathcal{T}_h$. So what lengths can a finite eigenvector have? Let x be an eigenvector.

$$\begin{aligned}
\lambda \cdot x &= \mathcal{T}_h x \\
&= (h * x) \downarrow 2 \\
\operatorname{ix} x &= \operatorname{ix}((h * x) \downarrow 2) \\
\min(\operatorname{ix} x) &= \min(\operatorname{ix}((h * x) \downarrow 2)) \\
&= \left\lceil \frac{\min(\operatorname{ix} h) + \min(\operatorname{ix} x)}{2} \right\rceil \\
2 \cdot \min(\operatorname{ix} x) &= \min(\operatorname{ix} h) + \min(\operatorname{ix} x) + (-\min(\operatorname{ix} h) - \min(\operatorname{ix} x)) \bmod 2 \\
\min(\operatorname{ix} x) &= \min(\operatorname{ix} h) + (-\min(\operatorname{ix} h) - \min(\operatorname{ix} x)) \bmod 2 \\
\max(\operatorname{ix} x) &= \max(\operatorname{ix} h) - (\max(\operatorname{ix} h) + \max(\operatorname{ix} x)) \bmod 2
\end{aligned}$$

This means that the index interval of an eigenvector x starts at $\min(\operatorname{ix} h)$ or $\min(\operatorname{ix} h) + 1$ and ends at $\max(\operatorname{ix} h)$ or $\max(\operatorname{ix} h) - 1$. Therefore $\operatorname{ix} x \subseteq \operatorname{ix} h$ and for considerations of eigenvectors we can restrict the infinite matrix to a finite square matrix of size $1 + \deg h$.

2.2.35 Definition. For a given mask h from $\ell_0(\mathbb{Z})$ where ν is the smallest index of any non-zero entry ($\nu = \min(\mathrm{ix}\, h)$) and κ the biggest one ($\kappa = \max(\mathrm{ix}\, h)$), we will call the matrix T_h with $T_h \in \mathbb{R}^{\{\nu,\ldots,\kappa\}^2}$ and

$$(T_h)_{j,k} = h_{2\cdot j-k}$$

$$T_h = \begin{pmatrix} h_\nu & & & & & \\ h_{\nu+2} & h_{\nu+1} & h_\nu & & & \\ h_{\nu+4} & h_{\nu+3} & h_{\nu+2} & h_{\nu+1} & h_\nu & \\ \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \\ & h_\kappa & h_{\kappa-1} & h_{\kappa-2} & h_{\kappa-3} & h_{\kappa-4} \\ & & & h_\kappa & h_{\kappa-1} & h_{\kappa-2} \\ & & & & & h_\kappa \end{pmatrix}$$

the *dyadic band matrix* of h.

The special matrix $2 \cdot T_{h*h^*}$ is called the *transition matrix* of h [SN97].

2.2.36 Theorem.
Prerequisite. The filters $h, \widetilde{h}$ are the low-passes of a perfect reconstruction filter bank, that is they fulfil $2 \cdot (h * \widetilde{h}) \downarrow 2 = \delta$. The functions φ and $\widetilde{\varphi}$ are refinable with respect to h and $\widetilde{h}$, respectively. They are also normalised to $\langle \varphi, \widetilde{\varphi}^* \rangle = 1$. The matrix $2 \cdot T_{h*\widetilde{h}}$ has the single eigenvalue 1.
Claim. The basis for V_0 consisting of the translated primal generators φ is orthogonal to the basis of adjoint dual generators $\widetilde{\varphi}$ in the sense

$$\forall k \in \mathbb{Z} \qquad \langle \varphi, \widetilde{\varphi}^* \to k \rangle = \begin{cases} 1 & : k = 0 \\ 0 & : \text{otherwise} \end{cases} .$$

Proof. We must show that

$$Q(\varphi * \widetilde{\varphi}) = \delta \tag{2.2.11}$$

in other words: $\varphi * \widetilde{\varphi}$ must be an *interpolating function*.

Due to Theorem 2.2.30 the function $\varphi * \widetilde{\varphi}$ is refinable with respect to $h * \widetilde{h}$. Because of Lemma 2.2.31 for the discretised function Φ ($\Phi = Q(\varphi * \widetilde{\varphi})$) the discrete refinement relation

$$\Phi = 2 \cdot (h * \widetilde{h} * \Phi) \downarrow 2$$

holds.

We verify that $\Phi = \delta$ is a solution of this refinement equation.

$$2 \cdot (h * \widetilde{h} * \delta) \downarrow 2 \quad = \quad 2 \cdot (h * \widetilde{h}) \downarrow 2 \overset{(2.2.6)}{=} \delta$$

The matrix $T_{h*\widetilde{h}}$ is defined such that $2 \cdot T_{h*\widetilde{h}} \cdot \Phi = 2 \cdot (h * \widetilde{h} * \Phi) \downarrow 2$. Φ must be an eigenvector of $2 \cdot T_{h*\widetilde{h}}$ with respect to the eigenvalue 1. This eigenvalue has multiplicity 1 and this means that there are no other solutions to the refinement equation.

□

2.2.37 Remark. The previous theorem can be considered as a light-weight version of the theorem of COHEN, DAUBECHIES, FEAUVEAU [Dau92] which also tells when φ and $\widetilde{\varphi}$ are functions of $\mathcal{L}_2(\mathbb{R})$.

2.2.38 Theorem.
Prerequisite. The primal filter pair (h, g) is complementary, and $(\widetilde{h}, \widetilde{g})$ is the dual filter pair. The functions $\varphi, \psi, \widetilde{\varphi}, \widetilde{\psi}$ are the corresponding primal and dual generators and wavelets. The generators are normalised

to $\left\langle \varphi, \widetilde{\varphi}^* \right\rangle = 1$. The matrix $T_{h*\widetilde{h}}$ has the single eigenvalue 1.

Claim. The basis of the scaled primal functions is orthogonal to the basis of scaled adjoint dual functions. That is

$$\forall \left\{ j,k,\widetilde{j},\widetilde{k} \right\} \subset \mathbb{Z} \qquad \left\langle \frac{(\psi \to k) \uparrow 2^j}{\sqrt{2^j}}, \frac{\left(\widetilde{\psi}^* \to \widetilde{k}\right) \uparrow 2^{\widetilde{j}}}{\sqrt{2^{\widetilde{j}}}} \right\rangle = \begin{cases} 1 & : j = \widetilde{j} \wedge k = \widetilde{k} \\ 0 & : \text{otherwise} \end{cases}$$

$$\forall \left\{ j,k,\widetilde{j},\widetilde{k} \right\} \subset \mathbb{Z} \wedge j \geq \widetilde{j} \qquad \left\langle \frac{(\varphi \to k) \uparrow 2^j}{\sqrt{2^j}}, \frac{\left(\widetilde{\psi}^* \to \widetilde{k}\right) \uparrow 2^{\widetilde{j}}}{\sqrt{2^{\widetilde{j}}}} \right\rangle = 0$$

$$\forall \left\{ j,k,\widetilde{j},\widetilde{k} \right\} \subset \mathbb{Z} \wedge j \leq \widetilde{j} \qquad \left\langle \frac{(\psi \to k) \uparrow 2^j}{\sqrt{2^j}}, \frac{\left(\widetilde{\varphi}^* \to \widetilde{k}\right) \uparrow 2^{\widetilde{j}}}{\sqrt{2^{\widetilde{j}}}} \right\rangle = 0$$

Proof.

1. Without loss of generality let $j \geq \widetilde{j}$ and set $n = j - \widetilde{j}$.

$$\begin{aligned} \left\langle \frac{(\psi \to k) \uparrow 2^j}{\sqrt{2^j}}, \frac{\left(\widetilde{\psi}^* \to \widetilde{k}\right) \uparrow 2^{\widetilde{j}}}{\sqrt{2^{\widetilde{j}}}} \right\rangle &= \left\langle \frac{(\psi \to k) \uparrow 2^n}{\sqrt{2^n}}, \widetilde{\psi}^* \to \widetilde{k} \right\rangle \\ &= \left\langle \frac{\psi \uparrow 2^n}{\sqrt{2^n}} \to (2^n \cdot k), \widetilde{\psi}^* \to \widetilde{k} \right\rangle \end{aligned}$$

   (a) Case: $n = 0$

$$\begin{aligned} \psi * \widetilde{\psi} &= 2 \cdot (g * \widetilde{g} * \varphi * \widetilde{\varphi}) \downarrow 2 \\ Q(\psi * \widetilde{\psi}) &= 2 \cdot Q\big(g * \widetilde{g} * (\varphi * \widetilde{\varphi})\big) \downarrow 2 \\ &= 2 \cdot \big(g * \widetilde{g} * Q(\varphi * \widetilde{\varphi})\big) \downarrow 2 \\ &\overset{(2.2.11)}{=} 2 \cdot (g * \widetilde{g}) \downarrow 2 \\ &\overset{(2.2.7)}{=} \delta \end{aligned}$$

   (b) Case: $n > 0$

$$\begin{aligned} \frac{\psi \uparrow 2^n}{\sqrt{2^n}} * \widetilde{\psi} &= 2^{-n/2} \cdot \left(\mathcal{R}^n_{2\cdot h} g * \widetilde{g} * \varphi * \widetilde{\varphi}\right) \downarrow 2 \\ &= 2^{n/2} \cdot \left(\mathcal{R}^n_h g * \widetilde{g} * \varphi * \widetilde{\varphi}\right) \downarrow 2 \\ Q\left(\frac{\psi \uparrow 2^n}{\sqrt{2^n}} * \widetilde{\psi}\right) &= 2^{n/2} \cdot Q\left(\mathcal{R}^n_h g * \widetilde{g} * \varphi * \widetilde{\varphi}\right) \downarrow 2 \\ &= 2^{n/2} \cdot \Big(\mathcal{R}^n_h g * \widetilde{g} * \underbrace{Q(\varphi * \widetilde{\varphi})}_{\delta}\Big) \downarrow 2 \\ &= 2^{n/2} \cdot \left(\widetilde{g} * h * \left(\mathcal{R}^{n-1}_h g\right) \uparrow 2\right) \downarrow 2 \\ &\overset{(1.2.2)}{=} 2^{n/2} \cdot \underbrace{(\widetilde{g} * h) \downarrow 2}_{0} * \mathcal{R}^{n-1}_h g \\ &\overset{(2.2.8)}{=} 0 \end{aligned}$$

2. Let $n = j - \widetilde{j}$.

$$\begin{aligned} Q\left(\frac{\varphi \uparrow 2^n}{\sqrt{2^n}} * \widetilde{\psi}\right) &= 2^{n/2} \cdot (\widetilde{g} * h) \downarrow 2 * \mathcal{R}^n_h \delta \\ &\overset{(2.2.8)}{=} 0 \end{aligned}$$

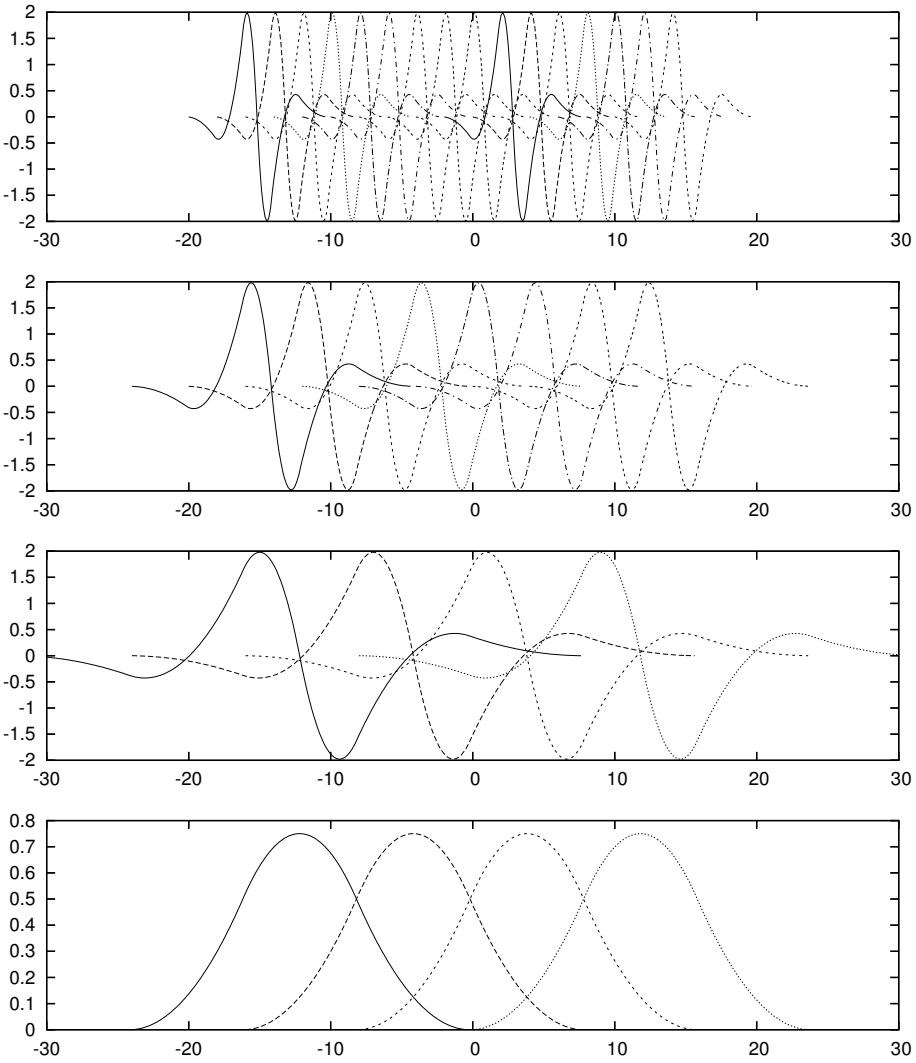


Figure 2.12: Dual basis functions of the CDF-3,3 wavelet

3. This is analogous to the previous case.

□

Our goal was to find the connection from the discrete wavelet transform back to the continuous one. The previous steps have shown how the discrete filters are related to continuous functions. But what about the signal? If we turn the discrete signal x into a function where each coefficient of x is the amplification of small scaled translated generators $\widetilde{\varphi}$, i.e. $x * \widetilde{\varphi}$, then the continuous wavelet transform with respect to ψ with subsequent sampling is the discrete wavelet transform.

If we want to restrict the sampling to a finite number of scales we have to complement the largest scale of wavelets with generators of this scale. Figure 2.12 shows some basis functions for a discrete transform interpreted in terms of continuous basis functions.

2.2.39 Theorem.
Prerequisite. Let x be a discrete signal, $x \in \ell_0(\mathbb{Z})$. Additionally the preconditions of Theorem 2.2.38 hold.
Claim. The discretised continuous wavelet transform with respect to a discrete wavelet basis can be

obtained by discrete convolution with the refined wavelet mask $\mathcal{R}^j_{2\cdot h} g$ (see Definition 2.2.1).

$$\forall j \in \mathbb{N} \quad \forall k \in \mathbb{Z} \qquad {W_\psi}^* \left(x * \widetilde{\varphi}\right)(2^j)(2^j \cdot k) = 2^{j/2} \cdot \left(x * \mathcal{R}^j_h g\right)_{2^j \cdot k}$$

Proof. We verify that the claim is even true if down-sampling is omitted, i.e. if we sample at each integral k instead of $2^j \cdot k$.

$$\begin{aligned}
{W_\psi}^* \left(x * \widetilde{\varphi}\right)\left(2^j\right) &= \frac{\psi \uparrow 2^j}{\sqrt{2^j}} * x * \widetilde{\varphi} \\
&= 2^{j/2} \cdot \mathcal{R}^j_h g * \varphi * x * \widetilde{\varphi} \\
Q\left({W_\psi}^* \left(x * \widetilde{\varphi}\right)\left(2^j\right)\right) &= 2^{j/2} \cdot x * \mathcal{R}^j_h g * \underbrace{Q(\varphi * \widetilde{\varphi})}_{\delta} \\
Q\left({W_\psi}^* \left(x * \widetilde{\varphi}\right)\left(2^j\right) \downarrow 2^j\right) &= 2^{j/2} \cdot \left(x * \mathcal{R}^j_h g\right) \downarrow 2^j
\end{aligned}$$

The last equation is equivalent to the claim. □

2.2.5 Generalisations

The main features of the discrete wavelet transform are the bijection (i.e. no redundancy and unique representation) and fast computation (linear time if the size of the filter is neglected). The discrete wavelet transform works on exponentially graded scales and a resolution which decreases with increasing scale. This basic method can be extended in many ways. Probably every obvious generalisation of this scheme is already proposed and investigated. We want to acknowledge some of them here.

Translation invariant transform

In this work we started with the translation invariant wavelet transform and came to the wavelet transform with down-sampling. But usually the translation invariant transform is considered as an extension of the critically sampled transform. We already got to know the transform where the sampling rate is the same for all scales, but in Section 4.3.2 of this work we will also motivate a transform where the sampling rate is increased by a constant factor for all scales (in our setting 2).

The first variant can also be considered as a discrete wavelet transform applied to shifted versions of the signal. Wavelet coefficients from different transforms can then be merged to a total translation invariant coefficient set but several coefficients in fine scales are present multiple times. [CD95, CL01]

Matching pursuits

Translation invariant transforms tend to produce a high amount of output. That is why the question arose how to reduce the output data size while retaining the translation invariance. The idea is to replace the basis of wavelets by a *frame*, intuitively spoken an "over-complete RIESZ basis". In contrast to a basis the functions of a frame need not to be linearly independent and thus the expansion of a signal into a frame is not necessarily unique. This gives us more freedom of choosing appropriate frame members for representing a signal.

We consider the frame of all wavelets shifted at the fine scale but select wavelets from it depending on the signal content. A simple idea is to transform a signal translationally invariant and keep only the portion of the largest coefficients. The problem is that we do not work with a basis but with a frame. That is the wavelets correlate with each other and thus close to each large coefficient some other large coefficients may exist.

An improved algorithm respects this problem: Perform a translation invariant transform and keep only a small portion of the highest coefficients. Transform the signal back but amplify it in a way that each coefficient is turned into a wavelet with its normal amplitude. This differs from the normal translation

invariant reconstruction because there it is assumed that many coefficients can contribute energy to one wavelet. The reconstructed signal should be an approximation to the original signal. Then subtract the reconstructed signal from the original one and repeat the approximation for the residue signal.

This algorithm is called *matching pursuit* [MZ93, Mal99] and the related theory explains how many large coefficients should be chosen in one go and when to abort the iteration. Redundancy in the wavelet frame can also be achieved by different kinds of wavelets. By providing a wavelet dictionary which consists of both tonal and transient audio atoms you can split a signal into these components.

Multichannel transform

Another approach addresses the graduation of scales. For instance if we down-sample the signal by a factor of 3 at each level we obtain two high-band signals per level and the scales are powers of 3. In an n-channel transform smooth generators can be produced by masks very similar to those popular for the original 2-channel transform (we anticipate Section 4.2 here), namely $\underbrace{(1, 1, \ldots, 1)}_{n \text{ components}}$.

The n-channel transform is determined by an $n \times n$ polyphase matrix. The transform can be considered as splitting the signal into n interleaved slices, then apply a matrix-vector-convolution of the polyphase matrix and the slices vector. The perfect reconstructability condition remains the same: The polyphase determinant must be a monomial. This implies that the polynomials of each column must be relatively prime.

A lifting decomposition (an anticipation, again, see Section 3.2) is possible but even more ambiguous. We can use the EUCLIDean algorithm to find row additions that cancel, say, the left bottom element of the polyphase matrix. Then we proceed eliminating the element above and so on. To make sure that the top left element is non-zero you can either swap some rows before the elimination (the pivot strategy of the GAUSS elimination) or swap the rows on every lifting step (the lifting strategy). This procedure can be continued to other columns until we obtain an upper triangular polyphase matrix. Because of the determinant being a monomial all elements on the diagonal are monomials thus the elements in the last column can be eliminated by $n-1$ scaled additions of the bottom row to the rows above. Then we proceed with the columns to the left until we obtain a diagonal matrix. The lifting decomposition is rather much like the GAUSS elimination. The main difference is that due to the missing division we need more than one step for eliminating one matrix element.

If a square polyphase matrix poses in principle no problem – what about a rectangular one? A rectangular polyphase matrix is of interest for a redundant transform like the one presented in Section 4.3.2. In general we can neither compute a determinant of a rectangular matrix nor an inverse. But we can explore conditions to reconstruct atomic signals, say a unit vector containing one monomial. If the polyphase matrix is $\left(h_{i,j} : (i,j) \in \{0, \ldots, n-1\} \times \{0, \ldots, m-1\}\right)$ with $n \geq m$ (equal or more rows than columns) and the signal vector is $\underbrace{(\delta, (\mathbf{0}), \ldots, (\mathbf{0}))}_{m \text{ components}}$ then the transformed signal is $(h_{0,0}, \ldots, h_{n-1,0})$. Let the reconstruction polyphase matrix be $\left(\widetilde{h}_{j,i} : (j,i) \in \{0, \ldots, m-1\} \times \{0, \ldots, n-1\}\right)$ then the reconstructed signal is $\widetilde{h}_{0,0} * h_{0,0} + \cdots + \widetilde{h}_{0,n-1} * h_{n-1,0}$ which shall be δ, again. That is both sets $\left\{h_{0,0}, \ldots, h_{n-1,0}\right\}$ and $\left\{\widetilde{h}_{0,0}, \ldots, \widetilde{h}_{0,n-1}\right\}$ must fulfil a BEZOUT equation. Since we consider univariate polynomials, this is equivalent to the condition that a monomial is a greatest common divisor of the filters. This condition is necessary but certainly not sufficient.

A necessary and sufficient condition for a (not necessarily square) polyphase matrix P being invertible is that if there is some $m \times n$ transformation matrix T such that the convolutional matrix product $T \circledast P$ has a unit determinant. The condition is sufficient since with

$$\widetilde{P} = (T \circledast P)^{-1} \circledast T$$

$\widetilde{P}$ is a left inverse of P because

$$\begin{aligned} \widetilde{P} \circledast P &= (T \circledast P)^{-1} \circledast T \circledast P \\ &= I \quad . \end{aligned}$$

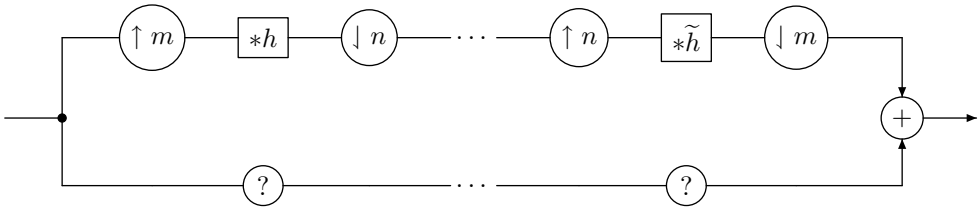


Figure 2.13: One level of a discrete wavelet transform based on fractional refinement

Conversely, the condition is necessary since if there is a left inverse $\widetilde{P}$ of P then there is a transformation T, namely $T = \widetilde{P}$, which satisfies $\widetilde{P} = (T \circledast P)^{-1} \circledast T$.

A kind of a permutation matrix is a very simple choice for T. More precisely, $T \circledast P$ is a sub-matrix of P generated by selecting some rows from it and $(T \circledast P)^{-1} \circledast T$ puts the rows of the inverted sub-matrix back to where they were taken from P. A transformation of this form is not possible for every invertible P. Even more it would not make any use of the redundant data generated by the application of P.

A more interesting choice is $T = P^T$ because then $\widetilde{P}$ is a pseudo inverse (or MOORE-PENROSE inverse). [Sto99, Theorem 4.8.5.1] Consequently a monomial determinant of $P^T \circledast P$ is a sufficient condition for the existence of a MOORE-PENROSE inverse in the ring of LAURENT polynomials. This is also a necessary condition according to the following reasoning. Since determinants are defined only for square matrices the proof of Lemma 2.2.6 cannot be adapted immediately. We will need two properties of pseudo inverses in order to derive the necessary condition.

$$\begin{aligned} \widetilde{P} \circledast P &= I \quad \Big| \quad \text{pseudo inverse law: } P = P \circledast \widetilde{P} \circledast P \\ \widetilde{P} \circledast P \circledast \widetilde{P} \circledast P &= I \quad \Big| \quad \text{pseudo inverse law: } P \circledast \widetilde{P} = (P \circledast \widetilde{P})^T \\ \widetilde{P} \circledast \widetilde{P}^T \circledast P^T \circledast P &= I \\ \det(\widetilde{P} \circledast \widetilde{P}^T) * \det(P^T \circledast P) &= \delta \quad . \end{aligned}$$

Consequently both $\det(\widetilde{P} \circledast \widetilde{P}^T)$ and $\det(P^T \circledast P)$ must be monomials.

But the existence of a pseudo inverse is not necessary for perfect reconstruction. E.g. the polyphase matrix P with $P = \begin{pmatrix} (\mathbf{1}, -1) \\ (1, 1) \end{pmatrix}$ has a left inverse $\frac{1}{2} \cdot \begin{pmatrix} (\mathbf{1}) & (\mathbf{1}) \end{pmatrix}$ but $\det \left(P^T \circledast P \right) = (\mathbf{2}, 0, 2)$ is not a unit.

As described above a left invertible polyphase matrix can be decomposed into lifting steps with the modified GAUSS elimination. The upper triangular matrix generated by the GAUSS elimination (in a redundant transform, i.e. $n > m$, the lower $n - m$ rows are zero) can be inverted from the left only by a (non-square) upper triangular matrix. We conclude that the elements on the diagonal must be monomials. Thus the second phase of the elimination works which turns the triangular matrix into one of diagonal shape.

Multichannel transforms are more popular for multi-dimensional filters. (See below)

Fractional refinement

An integer factor between scales is often too large. How can one achieve fractional scale factors?

If we want a scale factor of $\frac{n}{m}$ then the computation of the low-pass band becomes

$$y = (h * (x \uparrow m)) \downarrow n$$

(cf. Figure 2.13).

For complementing the low-pass band with a high-pass band we can use the multichannel approach of the previous section with square polyphase matrices. In contrast to the multichannel approach we do not

choose a single channel as low-pass channel which is cascaded but we use m channels for the low-pass cascade from a total of n channels. From the computation of y follows

$$\begin{aligned} y \rightarrow k \downarrow m &= (h * (x \uparrow m)) \downarrow n \rightarrow k \downarrow m \\ &= (h * (x \uparrow m) \rightarrow (n \cdot k)) \downarrow m \downarrow n && \mid \quad (1.2.2) \\ &= (h \rightarrow (n \cdot k) \downarrow m * x) \downarrow n && \mid \quad \text{Lemma 2.2.3} \\ &= \sum_{j=0}^{n-1} (h \rightarrow (n \cdot k - m \cdot j) \downarrow m \downarrow n) * (x \rightarrow j \downarrow n) \end{aligned}$$

which let us choose the low-pass portion of the polyphase matrix P and the sliced input signal x according to

$$\begin{aligned} \forall k \in \{0, \ldots, m-1\} \wedge j \in \{0, \ldots, n-1\} \qquad & P_{k,j} = h \rightarrow (n \cdot k - m \cdot j) \downarrow (n \cdot m) \\ \forall j \in \{0, \ldots, n-1\} \qquad & \boldsymbol{x}_j = x \rightarrow j \downarrow n \quad . \end{aligned}$$

E.g. for a ratio of 3/2 we use a 3×3 polyphase matrix. The input signal x and the output signal sequence $\boldsymbol{y}$ are connected by

$$\begin{pmatrix} \boldsymbol{y}_0 \\ \boldsymbol{y}_1 \\ \boldsymbol{y}_2 \end{pmatrix} = \begin{pmatrix} h \downarrow 6 & h \rightarrow -2 \downarrow 6 & h \rightarrow -4 \downarrow 6 \\ h \rightarrow 3 \downarrow 6 & h \rightarrow 1 \downarrow 6 & h \rightarrow -1 \downarrow 6 \\ ? & ? & ? \end{pmatrix} \circledast \begin{pmatrix} x \downarrow 3 \\ x \rightarrow 1 \downarrow 3 \\ x \rightarrow 2 \downarrow 3 \end{pmatrix}$$

Now, $\boldsymbol{y}_0$ and $\boldsymbol{y}_1$ are merged to a total low-pass band, precisely $\boldsymbol{y}_0 \uparrow 2 + \boldsymbol{y}_1 \uparrow 2 \leftarrow 1$. This signal is then fed to the next level of the transform. With a square matrix this transform is not redundant. However it is easy to extend this method to redundancy by switching to non-square polyphase matrices. Perfect reconstruction is treated exactly as in the previous section about a multichannel transform.

It remains the question how this kind of subband coding can be interpreted in terms of wavelets. An essential difference to the transformations mentioned above is that the wavelet and generator functions do not have uniform shapes. The question arises how different the shapes are and how to make them approximately uniform. The computation of the low-pass band suggests that the refinement equation (2.2.9) becomes

$$\varphi = \frac{n}{m} \cdot (h * (\varphi \uparrow m)) \downarrow n$$

for fractional refinement. But this cannot be true since the shapes of the shifted generators differ [CD93]. Instead we can state a refinement for the sequence φ of all generators.

$$\varphi_k = \frac{n}{m} \cdot \left(\sum_{j \in \mathbb{Z}} h_{m \cdot j - n \cdot k} \cdot \varphi_j \uparrow m \right) \downarrow n$$

Another central question here is how to assert smooth wavelets. (We anticipate the treatise in Section 4.2.) In [RB97] it is shown that similar to the case of integral refinement factors masks of the form $\underbrace{(1, 1, \ldots, 1)}_{n \text{ components}}$ lead to smooth generators. Here n is the numerator of the (cancelled) scale factor $\frac{n}{m}$. Experience shows that the shape of smooth wavelets and generators depend only slightly on their position.

Multi-dimensional wavelets

The next generalisation presented here is the increasing of dimensions of the signal. So far we considered only functions from $\mathbb{R} \rightarrow \mathbb{R}$ and discrete signals from $\ell_0(\mathbb{Z})$. This can be extended to d dimensions, namely $\mathbb{R}^d \rightarrow \mathbb{R}$ and $\ell_0\left(\mathbb{Z}^d\right)$. If we want to treat images it is $d = 2$, for video streams or true 3D images it is $d = 3$.

The most simple extension is a wavelet transform which is applied separately to some dimensions. For an image this would mean to apply a one-dimensional wavelet transform to each row and then to each column. If the rows are transformed with generator φ_0 and wavelet ψ_0 and the columns are transformed with generator φ_1 and wavelet ψ_1 then the composed transform can be considered as a transform with respect to the two-dimensional generator $\varphi_0 \otimes \varphi_1$ and the wavelets $\psi_0 \otimes \varphi_1$, $\varphi_0 \otimes \psi_1$, $\psi_0 \otimes \psi_1$.

The next generalisation is to leave tensor product functions and to turn to general higher dimensional functions, namely from $\mathbb{R}^d \to \mathbb{R}$. With a refinement by the factor k for all dimensions we have a $k^d \times k^d$ polyphase matrix containing d-dimensional filters, i.e. polynomials with respect to d variables. For a square matrix the necessary and sufficient condition for invertibility is again that the determinant must be a unit.

For multivariate polynomials the connection between the greatest common divisor and the solution of the BEZOUT equation does no longer hold. More precisely, for multivariate polynomials x, y, p, q, r the equation

$$x * p + y * q = r$$

preserves that a greatest common divisor of p and q must divide r, too, because any common divisor of p and q divides r. But it is no longer true that a greatest common divisor of p and q can be expressed by a linear combination of them. For instance the two-dimensional masks (i.e. the two-variate polynomials) $\begin{pmatrix} \mathbf{1} & 0 \\ 0 & 1 \end{pmatrix}$ and $\begin{pmatrix} \mathbf{0} & 1 \\ 1 & 0 \end{pmatrix}$ are relatively prime, but their greatest common divisor $\begin{pmatrix} \mathbf{1} \end{pmatrix}$ cannot be represented by a linear combination of them.

This consideration shows that a necessary condition for perfect reconstruction is still that the elements in a column (or a row) of the polyphase matrix must be relatively prime. By expanding a determinant with respect to a column it becomes clear that it must be possible to represent a monomial by a linear combination of the elements of this column. But here this condition is stronger than the request for relative primes. An alternative formulation is that the ideal spanned by all elements of a column must contain a monomial (and thus all polynomials). A formulation with GROEBNER bases like "the reduced GROEBNER basis of the column elements must be $\{1\}$" is not so obvious because we have LAURENT polynomials rather than ordinary ones. Approaches for not only biorthogonal but even orthogonal filter banks are also more involved for higher dimensions [Maa96].

The next generalisation concerns the shrink operation. Beyond uniform shrinking in each dimension in higher dimensions it is possible to transform affinely. We will redefine the shrink operation for matrices as factors.

$$\forall t \in \mathbb{R}^d \wedge M \in \mathbb{R}^{d \times d} \quad (f \downarrow M)(t) = f(M \cdot t)$$

Analogous to Definition 2.2.24 this let us define refinable functions with respect to a general (but integer) dilation matrix M, $M \in \mathbb{Z}^{d \times d}$ [BW92, Kla01] and a higher dimensional mask h, $h \in \ell_0\left(\mathbb{Z}^d\right)$.

$$\varphi = \det M \cdot (h * \varphi) \downarrow M$$

$$\text{alternatively} \qquad \forall t \in \mathbb{R}^d \quad \varphi(t) = \det M \cdot \sum_{k \in \mathbb{Z}^d} h_k \cdot \varphi(M \cdot t - k)$$

The multiplication with M maps the grid $\mathbb{Z}^d$ to an affinely distorted grid $M \cdot \mathbb{Z}^d$. The columns of M contain the vectors of the *parallelepiped* the half-opened box $[0,1)^d$ is mapped to. Thus the shrinking operation reduces a parallelepiped to this unit box. Each box $k + [0,1)^d$ with $k \in \mathbb{Z}^d$ contains exact one integral point ($\#\left((k + [0,1)^d) \cap \mathbb{Z}^d\right) = 1$) whereas the transformed box $M \cdot (k + [0,1)^d)$ contains $|\det M|$ integral points ($\#\left(M \cdot (k + [0,1)^d) \cap \mathbb{Z}^d\right) = |\det M|$). This reduction factor means that the factor space $\mathbb{Z}^d/(M \cdot \mathbb{Z}^d)$ contains $|\det M|$ distinct moved variants of $M \cdot \mathbb{Z}^d$. This is also the number of input and output channels and the size of the polyphase matrix (cf. Figure 2.14).

Multiwavelets

The *multiwavelet* transform processes a certain number of signals and generates the same number of wavelet transforms. But it is not only the parallel application of some wavelet transforms to some sig-

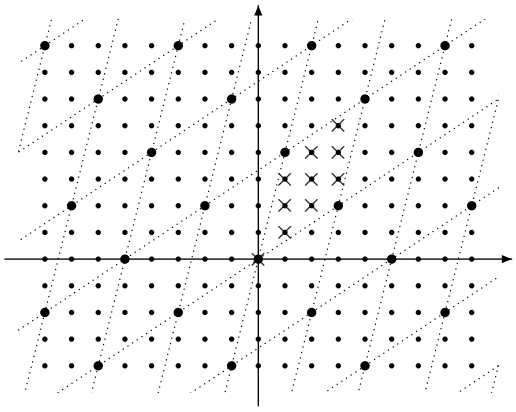


Figure 2.14: Refinement with a general dilation matrix: The matrix $\begin{pmatrix} 3 & 1 \\ 2 & 4 \end{pmatrix}$ represents the basis vectors $(3, 2)$ and $(1, 4)$ which span the dotted lattice. All crosses are images with respect to the dilation of points within the unit box. The absolute value of the determinant is 10, thus there are 10 crosses in the half-open parallelogram. (One cross coincides with the $(0, 0)$ dot.)

nals but the wavelet transforms are interleaved. It is useful to put the filters for the generators into a matrix H from $\ell_0(\mathbb{Z})^{n\times n}$ and setup a matrix refinement equation for the vector Φ of generators ($\Phi \in (\mathbb{R} \to \mathbb{R})^n$). Be careful to not mix this up with the polyphase or modulation matrix. Analogously there is a matrix G describing the wavelets in a vector Ψ.

$$\Phi = (\downarrow 2) \circ (H \circledast \Phi)$$
$$\Psi = (\downarrow 2) \circ (G \circledast \Phi)$$

Note that we use the function composition here to apply a function to all elements of a vector as described in the introduction (Section 1.2.1).

Each level of the transform can be described by a polyphase matrix which is organised in blocks. If X is the vector of input signals, Y_0 the vector of low-pass outputs and Y_1 the vector of high-pass outputs then one transformation step can be expressed by

$$\begin{pmatrix} Y_0 \\ Y_1 \end{pmatrix} = \begin{pmatrix} (\downarrow 2) \circ H & (\downarrow 2) \circ (\leftarrow 1) \circ H \\ (\downarrow 2) \circ G & (\downarrow 2) \circ (\leftarrow 1) \circ G \end{pmatrix} \circledast \begin{pmatrix} (\downarrow 2) \circ X \\ (\downarrow 2) \circ (\rightarrow 1) \circ X \end{pmatrix} .$$

This representation clearly shows the connection to the multichannel wavelet transform (see above). An n-wavelet transform with m channels from which k low-pass channels are chosen for cascading can be considered as a scalar wavelet transform with $n \cdot m$ channels from which $n \cdot k$ are cascaded. [RN96]

The interpretation of the discrete input signals in terms of multiwavelets is that each signal contains coefficients of another generator function. If you interpret a single signal as a sequence of coefficients for a cycling list of certain generators then the multiwavelet transform becomes useful also for single signals. If the generators are equal (or at least do not differ too much) then it is reasonable to simply split a single signal into n slices before applying the multiwavelet transform.

By dropping the restriction of uniform shapes of all versions of generators and wavelets in a basis some combinations of features like finite support, orthogonality, symmetry, and interpolation are possible which are impossible for the plain DWT. [GHM94, Sel99] The deviations in shape may be even reduced to a negligible amount.

Recursive filters

There is also an extension which is not related to the structure of the transform but to the nature of the wavelets. A criterion for perfect reconstruction is that the polyphase matrix has a monomial as determinant.

This is necessary because in the ring of LAURENT polynomials only monomials are units. This restriction can be lifted by allowing fractions of filters. Division of filters means polynomial division, in general this results in a series. Because the filters are LAURENT polynomials the division is ambiguous. Polynomial division is known as *recursive filter* [Ham89] or *infinite impulse response filter* (short *IIR filter*) [Zöl02] in the signal processing community or as the solution of a linear *difference equation* with constant coefficients to the math community [KP01].

Fractions of filters provide more degrees of freedom. For example with plain LAURENT polynomials only the generators of a discrete wavelet transform are refinable functions, but a wavelet becomes refinable when fractions of filters are allowed for refinement. [SSZ99]

If (h, g) is the filter pair of a discrete wavelet transform where φ is refinable with respect to h and ψ is the wavelet associated with g then $\psi \uparrow 2$ is refinable with respect to $(h * (g \uparrow 2)) \not* g$. The form of this expression reveals how to derive it: We expand $\psi \uparrow 2$ into generators according to the wavelet definition, then we lift the generator one scale higher, then we go back to the wavelet, which is now also one scale larger, namely $\psi \uparrow 4$.

$$\begin{aligned}
\psi &= 2 \cdot (g * \varphi) \downarrow 2 \\
\psi \uparrow 2 &= 2 \cdot g * \varphi \\
(g \uparrow 2) * h * (\psi \uparrow 2) &= 2 \cdot (g \uparrow 2 * g * h * \varphi) \\
(g \uparrow 2) * h * (\psi \uparrow 2) &= g * (g \uparrow 2) * (\varphi \uparrow 2)
\end{aligned}$$

because $\psi \uparrow 4 = 2 \cdot (g \uparrow 2) * (\varphi \uparrow 2)$

$$\begin{aligned}
2 \cdot (g \uparrow 2) * h * (\psi \uparrow 2) &= g * (\psi \uparrow 4) \\
2 \cdot (g \uparrow 2 * h) \not* g * (\psi \uparrow 2) &= \psi \uparrow 4
\end{aligned}$$

It is interesting in which case a wavelet is also refinable with respect to a finite (non-recursive) filter, that is in which cases can $g \uparrow 2 * h$ be divided by g. If the filter bank allows perfect reconstruction then h and g are relatively prime, that is g must divide $g \uparrow 2$. Thus also g_- must divide $(g \uparrow 2)_-$ which is equal to $g \uparrow 2$. Thus $\operatorname{lcm}(g, g_-)$ must divide $g \uparrow 2$. Because of perfect reconstruction also g and g_- must be relatively prime which implies $\operatorname{lcm}(g, g_-) = g * g_-$. The polynomials $g * g_-$ and $g \uparrow 2$ have the same degree. It follows that they are equal apart from a weighting. We can split g into linear factors which leads to an 1:1 correspondence between linear factors of g and g_- and quadratic factors of $g \uparrow 2$. If $x * x_- = c \cdot x \uparrow 2$ and $y * y_- = d \cdot y \uparrow 2$ then

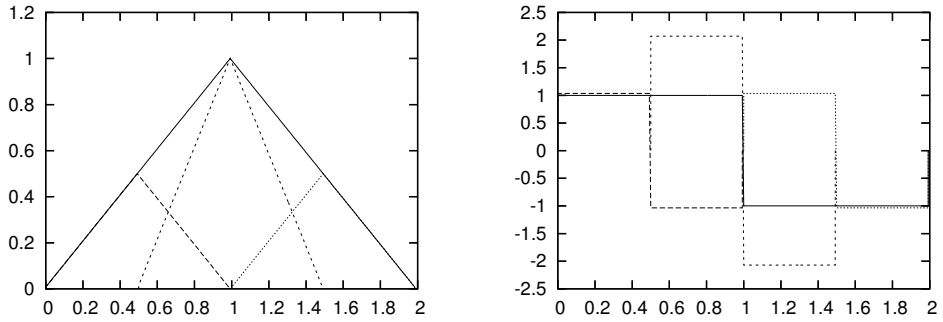
$$\begin{aligned}
(x * y) * (x * y)_- &= x * x_- * y * y_- \\
&= c \cdot d \cdot x \uparrow 2 * y \uparrow 2 \\
&= c \cdot d \cdot (x * y) \uparrow 2 \quad .
\end{aligned}$$

For which linear factors does the condition hold?

$$\begin{aligned}
(\mathbf{1}, \alpha) * (\mathbf{1}, -\alpha) &\stackrel{!}{=} c \cdot (\mathbf{1}, 0, \alpha) \\
\left(\mathbf{1}, 0, -\alpha^2\right) &= c \cdot (\mathbf{1}, 0, \alpha) \\
c &= 1 \\
\alpha &\in \{0, -1\} \quad .
\end{aligned}$$

For $\alpha = 0$ the linear factor is a constant factor. This means that masks of refinable wavelets must be convolutional powers of the mask $(\mathbf{1}, -1)$.

An example for this observation is that the HAAR wavelet (mask $(\mathbf{1}, -1)$) is refinable with respect to the doubled mask $\frac{1}{2} \cdot (\mathbf{1}, 2, 1)$ of the hat generator function (cf. Figure 2.15). We verify that $\left(\frac{1}{2} \cdot (\mathbf{1}, 1) * ((\mathbf{1}, -1) \uparrow 2)\right) \not* (\mathbf{1}, -1) = \frac{1}{2} \cdot (\mathbf{1}, 2, 1)$. Because of Lemma 2.2.28 this is no surprise since the HAAR wavelet is in some sense the derivative of the hat function. (To be precise the HAAR wavelet is the *weak derivative* of the hat function.)



linear B-spline HAAR wavelet

Figure 2.15: Refinement of a wavelet: The linear B-spline (hat function) and the HAAR wavelet are refinable with respect to essentially the same mask. The mask for the HAAR wavelet is amplified by a factor of 2 compared to that of the hat function. The HAAR functions are slightly distorted in order to reveal overlapping lines.

Chapter 3

Matching wavelets

In this chapter we will derive a main result of this work: A method for designing a wavelet that matches a given pattern. This is somehow similar to previous work of the author [Thi01]. Again, the lifting scheme plays a central role. The application in [Thi01] is image compression and the approach is to generate image specific wavelet filters which minimise the energy on the high-pass bands before quantisation. The wavelet filter is adapted separately for each level of the transformation.

Our current goal is different: We want to compute one wavelet filter bank for a wavelet that matches a pattern. The designed wavelet is the same for all levels disregarding dilations. This constraint can be easily dropped while retaining perfect reconstructability but it causes problems when it comes to the numerical analysis of the reconstruction.

At the end of the chapter we will know how to design matched wavelets efficiently. This allows us a discrete wavelet transform with respect to a matched wavelet with perfect reconstruction and the typical efficiency of the discrete transform.

3.1 Refinable functions

Refinable functions as introduced in Definition 2.2.24 originate from considering the discrete wavelet transform as discretised continuous wavelet transform. Discrete wavelets are linear combinations of integral translates of refinable functions. Thus it is worth exploring properties of refinable functions first.

3.1.1 Construction of refinable functions

Until now we have only considered a lose connection of a refinable function with a refinement mask. We will now construct refinable functions from refinement masks.

The cascade algorithm

The *cascade algorithm* uses the refinement relation for the approximation of the shape of a refinable function. The refinement relation

$$\varphi = 2 \cdot (h * \varphi) \downarrow 2$$

is interpreted as recursively defined function sequence with

$$\varphi_{j+1} = 2 \cdot (h * \varphi_j) \downarrow 2 \quad .$$

According to [Str96, Theorem 3 and 4] the iteration converges with respect to the $\mathcal{L}_2(\mathbb{R})$ norm if $\forall t \in \mathbb{R}\ \sum_{n\in\mathbb{Z}} \varphi_0(t-n) = 1$ and if $2 \cdot T_h$ has the eigenvalue 1 and all other eigenvalues have a smaller absolute value.

The cascade algorithm can be used to approximate the shape of refinable functions. We look for a function which is refinable with respect to h. For simplification we let $m = 2 \cdot h$. We start with the characteristic function $\chi_{[0,1)}$ which is clearly an admissible starting function. After j iterations we obtain

$$\begin{aligned}\varphi_j &= (m * \ldots (m * (m * \varphi_0) \downarrow 2) \ldots) \downarrow 2 \\ &= (\mathcal{R}_m^j \delta * \varphi_0) \downarrow 2^j \quad .\end{aligned}$$

This can be verified with the Definition 2.2.1 of $\mathcal{R}_m$:

$$\begin{aligned}\text{Base case:} \quad & \varphi_0 = \mathcal{R}_m^0 \delta * \varphi_0 \\ \text{Recurrence:} \quad & \varphi_{j+1} = \left(\mathcal{R}_m^j \delta * \varphi_1\right) \downarrow 2^j \\ &= \left(\mathcal{R}_m^j \delta * (m * \varphi_0) \downarrow 2\right) \downarrow 2^j \\ &= \left(\mathcal{R}_m^j \delta \uparrow 2 * m * \varphi_0\right) \downarrow 2^{j+1} \\ &= \left(\mathcal{R}_m \left(\mathcal{R}_m^j \delta\right) * \varphi_0\right) \downarrow 2^{j+1} \\ &= \left(\mathcal{R}_m^{j+1} \delta * \varphi_0\right) \downarrow 2^{j+1}\end{aligned}$$

This means that φ_j is a linear combination of translates in a 2^{-j}-lattice of $\varphi_0 \downarrow 2^j$ which are rather narrow functions. Thus the shape of $\left(\mathcal{R}_m^j \delta * \varphi_0\right) \downarrow 2^j$ is dominated by $\mathcal{R}_m^j \delta$ and you can argue that the shape of $\mathcal{R}_m^j \delta$ approaches that of $\varphi \uparrow 2^j$ for increasing j.

The cascade algorithm can be implemented as recursion

$$\begin{aligned}\boldsymbol{x}_0 &= \delta \\ \boldsymbol{x}_{j+1} &= \boldsymbol{x}_j * (m \uparrow 2^j) \quad .\end{aligned}$$

The shape of the corresponding wavelet can be approximated by $\boldsymbol{x}_j * (2 \cdot g \uparrow 2^j)$.

There is an alternative approach for the approximation which leads to a slightly different algorithm. A single coefficient in the wavelet transform represents a generator or wavelet function. You can reconstruct a generator or wavelet approximation by reconstructing (Corollary 2.2.11) a wavelet transform that is entirely zero except for one coefficient.

$$\begin{aligned}\boldsymbol{x}_0 &= \delta \\ \boldsymbol{x}_{j+1} &= (\boldsymbol{x}_j \uparrow 2) * m\end{aligned}$$

This method is equal to applying the refinement operator multiple times yielding $\boldsymbol{x}_{j+1} = 2 \cdot \mathcal{R}_m^j h$ for the generator and $2 \cdot \mathcal{R}_m^j g$ for the wavelet.

The difference between these two algorithms is the same as the one mentioned in Remark 2.2.33.

All figures (including of course Figure 3.1) of refinable functions in this thesis were prepared with this "inner" cascade algorithm.

The infinite product

The "inner" cascade algorithm can be performed in the frequency domain, too. This is especially useful for exploring the smoothness of refinable functions. In the FOURIER domain the convolution becomes a multiplication and convergence can be considered pointwise.

$$\begin{aligned}\varphi &= 2 \cdot (h * \varphi) \downarrow 2 \\ \widehat{\varphi} &= \widehat{(h * \varphi)} \uparrow 2 \\ &= (\widehat{h} \cdot \widehat{\varphi}) \uparrow 2 \\ \widehat{\varphi}(\xi) &= \widehat{h}\left(\frac{\xi}{2}\right) \cdot \widehat{\varphi}\left(\frac{\xi}{2}\right)\end{aligned}$$

Insert the refinement relation for $\widehat{\varphi}$ repeatedly. Assume that $\widehat{\varphi}$ is continuous at 0.

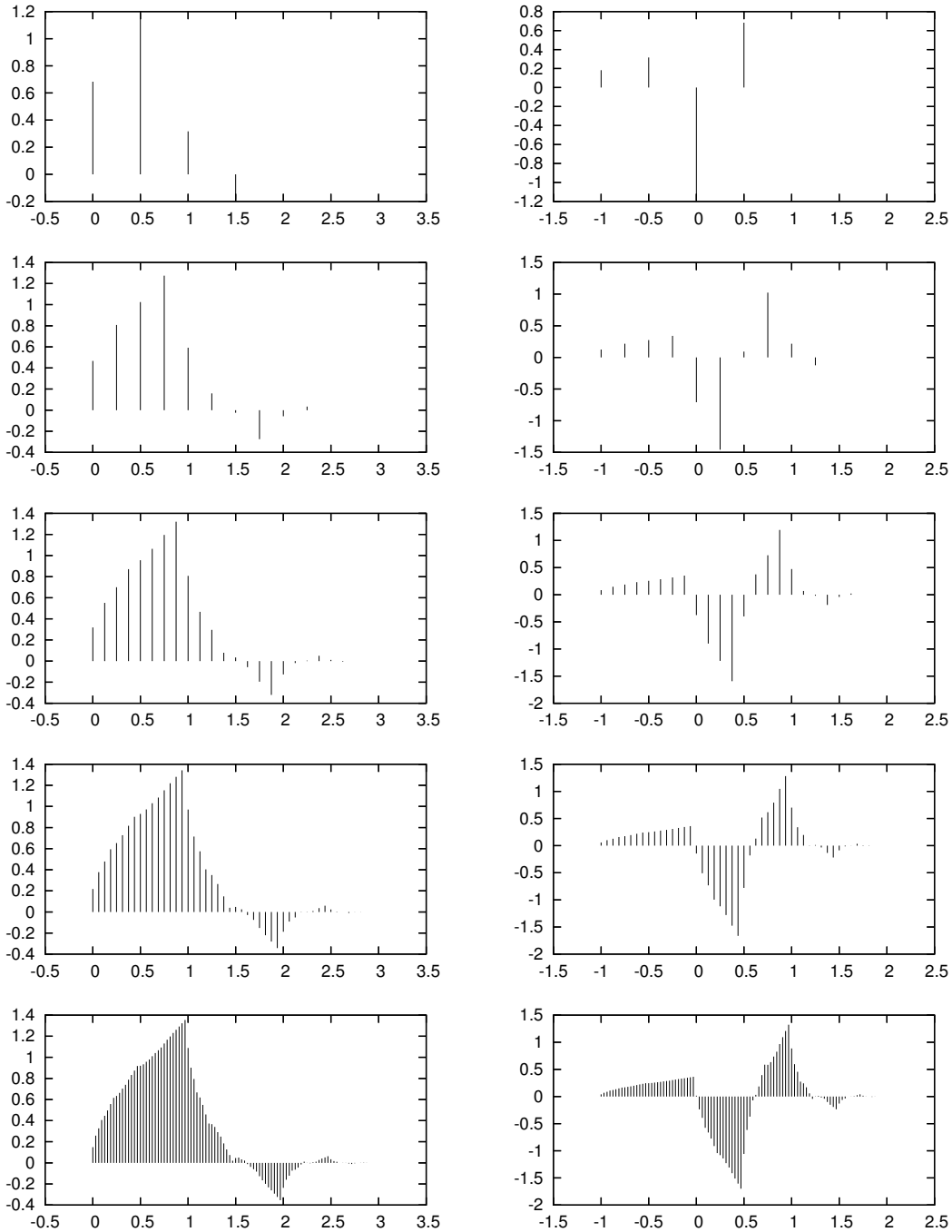


Figure 3.1: The "inner" cascade algorithm for the Daubechies-2 wavelet basis: The left column shows approximations of the generator, that is $2 \cdot \mathcal{R}_m^j h$, and the right column shows the wavelet approximation $2 \cdot \mathcal{R}_m^j g$ both at 2^{-j-1} rasters for $j \in \{0, \ldots, 4\}$

$$\widehat{\varphi}(\xi) = \widehat{\varphi}(0) \cdot \prod_{j=1}^{\infty} \widehat{h}\left(2^{-j} \cdot \xi\right)$$

Note that the last step yielding the infinite product is informal. We cannot conclude from an arbitrary number of substitutions to a limit. We have not derived that the frequency spectrum of every refinable function can be represented by this product, we only know that every function representable by the infinite product is refinable. Indeed from Section 2.2.5 we already know a counterexample: The HAAR wavelet is refinable with respect to $\frac{1}{2} \cdot (\mathbf{1}, 2, 1)$ but the sum of the mask coefficients is 2 thus the product diverges for $\xi = 0$.

3.1.2 Transfer operator

A refinable function is an eigenfunction of the transfer operator as defined in Definition 2.2.24. Thus the filter mask and the dyadic band matrix (Definition 2.2.35) are the objects we must consider in order to find out details about a refinable function.

Evaluating refinable functions for integer arguments

If you discretise the refinement equation on both sides as in Lemma 2.2.31 you obtain an equation system whose solution yields the values of the refinable function at integral positions. Let φ be refinable with respect to h, and let ν be the lower and κ the upper index of h. Then we obtain

$$\begin{aligned} Q\varphi &= 2 \cdot (h * Q\varphi) \downarrow 2 \\ \begin{pmatrix} \varphi(\nu) \\ \varphi(\nu+1) \\ \vdots \\ \varphi(\kappa) \end{pmatrix} &= 2 \cdot T_h \cdot \begin{pmatrix} \varphi(\nu) \\ \varphi(\nu+1) \\ \vdots \\ \varphi(\kappa) \end{pmatrix} \\ 0 &= (2 \cdot T_h - I) \cdot \begin{pmatrix} \varphi(\nu) \\ \varphi(\nu+1) \\ \vdots \\ \varphi(\kappa) \end{pmatrix} \quad . \end{aligned}$$

To obtain a solution different from zero the matrix $2 \cdot T_h - I$ must be singular, that is T_h must have the eigenvalue $\frac{1}{2}$.

Once we have the values for all integral arguments we can obtain the values of φ for all dyadic arguments, i.e. all arguments of the form $k \cdot 2^{-j}$ for $k \in \mathbb{Z}$ and $j \in \mathbb{N}_0$. For simplicity we use the substitution $m = 2 \cdot h$, again.

$$\begin{aligned} \varphi \uparrow 2 &= m * \varphi \\ Q(\varphi \uparrow 2) &= Q(m * \varphi) \\ &= m * Q\varphi \end{aligned}$$

This computation can be iterated, where the jth iteration step is

$$\begin{aligned} \varphi \uparrow 2^{j+1} &= (m * \varphi) \uparrow 2^j \\ Q\left(\varphi \uparrow 2^{j+1}\right) &= Q\left((m * \varphi) \uparrow 2^j\right) \\ &= m \uparrow 2^j * Q\left(\varphi \uparrow 2^j\right) \quad . \end{aligned}$$

By unrolling the iteration we obtain the explicit formula

$$Q\left(\varphi \uparrow 2^j\right) = \mathcal{R}_m^j \delta * Q\varphi \quad .$$

Evaluating scalar products of refinable functions

The algorithm of the previous section can be re-used for the computation of scalar products between translates of two functions φ_0 and φ_1 which are refinable with respect to h_0 and h_1, respectively. [DM93] It holds

$$\begin{aligned}\langle \varphi_0 \to t_0, \varphi_1 \to t_1 \rangle &= \langle \varphi_0, \varphi_1 \to (t_1 - t_0) \rangle \\ &= (\varphi_0 * {\varphi_1}^*)(t_1 - t_0) \qquad .\end{aligned}$$

Due to Theorem 2.2.30 the function $\varphi_0 * {\varphi_1}^*$ is refinable with respect to $h_0 * {h_1}^*$. With the algorithm in Section 3.1.2 the scalar products of all dyadic shifts can be computed irrespective of a constant factor.

Properties of the dyadic band matrix

3.1.1 Theorem.
Claim.

$$\operatorname{trace} T_g \cdot \operatorname{trace} T_h = \operatorname{trace} T_{g*h}$$

Proof.

$$\begin{aligned}\operatorname{trace} T_g &= \sum_{j \in \mathbb{Z}} g_j \\ &= Eg\,(1) \\ Eg\,(1) \cdot Eh\,(1) &\overset{(1.2.4)}{=} E\,(g * h)\,(1)\end{aligned}$$

□

3.1.2 Lemma.
Claim. For each filter h the filter $h * h_-$ has zero coefficients at each odd index.

$$h * h_- = (h * h_-) \downarrow 2 \uparrow 2$$

Proof.

$$\begin{aligned}(h * h_-)_- &= h_- * (h_-)_- \\ &= h_- * h \\ &= h * h_-\end{aligned}$$

Because $x_- = x \Rightarrow x = x \downarrow 2 \uparrow 2$ it follows that $h * h_- = (h * h_-) \downarrow 2 \uparrow 2$. □

The determinant $\det T_g$ is a homogenous polynomial term with degree $1 + \deg g$ with respect to the variables g_i. Please note the difference in viewing g as a polynomial here! The polynomial term $\det T_{g*h}$ consequently is also homogenous with respect to the coefficients of g and h and is of degree $2 \cdot (1 + \deg g + \deg h)$.

3.1.3 Theorem.
Claim. The term $\det T_{g*h}$ contains the factors $\det T_g$ and $\det T_h$, but with possibly less multiplicity.

$$\exists n \in \mathbb{N} \qquad (\det T_g \cdot \det T_h) \mid \left(\det T_{g*h}\right)^n$$

Proof. Due to HILBERT's Nullstellensatz [Wei05, HILBERT's Nullstellensatz] it is enough to show, that $\det T_{g*h}$ is zero whenever $\det T_g$ or $\det T_h$ is zero. This way we cannot prove that $(\det T_g \cdot \det T_h) \mid \det T_{g*h}$ because $\det T_g$ may contain a squared factor where $\det T_{g*h}$ contains the same factor only once.

Without loss of generality we prove $\det T_g = 0 \Rightarrow \det T_{g*h} = 0$. The condition $\det T_g = 0$ means, that T_g is singular which in turn means that there is a non-zero vector p with $T_g \cdot p = 0$. This is equivalent to $(g * p) \downarrow 2 = 0$ where $\operatorname{ix} p \subseteq \operatorname{ix} g$. We verify that a vector from the kernel of T_{g*h} is $h_- * p$. Indeed it has the proper index interval, because

$$\operatorname{ix}(h_- * p) = \operatorname{ix} h_- + \operatorname{ix} p \subseteq \operatorname{ix} h + \operatorname{ix} g = \operatorname{ix}(g * h) \quad .$$

$$\begin{aligned}
(g * h * (h_- * p)) \downarrow 2 &= ((h * h_-) \downarrow 2 \uparrow 2 * g * p) \downarrow 2 && \mid \quad \text{Lemma 3.1.2} \\
&= (h * h_-) \downarrow 2 * (g * p) \downarrow 2 && \mid \quad (1.2.2) \\
&= (h * h_-) \downarrow 2 * 0 \\
&= 0
\end{aligned}$$

□

3.1.4 Remark. Assume a p with $\operatorname{ix} p \subseteq \operatorname{ix} g$ and $(g * p) \downarrow 2 = 0$ then $\min(\operatorname{ix} p) > \min(\operatorname{ix} g)$ and $\max(\operatorname{ix} p) < \max(\operatorname{ix} g)$ because the first and the last coefficient of $g * p$ are products of the first and last coefficients of g and p, respectively.

Due to Lemma 2.2.3 the equation $(g * p) \downarrow 2 = 0$ is equivalent to

$$g \downarrow 2 * p \downarrow 2 + (g \leftarrow 1) \downarrow 2 * (p \rightarrow 1) \downarrow 2 = 0 \quad .$$

This is a BEZOUT equation, again. It holds

$$\begin{aligned}
\deg(p \downarrow 2) &< \deg((g \leftarrow 1) \downarrow 2) \\
\deg((p \rightarrow 1) \downarrow 2) &< \deg(g \downarrow 2)
\end{aligned}$$

because $g \leftarrow 1$ has at least two non-zero coefficients more than p at the left end and the same is valid for g and $p \rightarrow 1$. This means that $g \downarrow 2$ and $(g \leftarrow 1) \downarrow 2$ have a non-trivial common divisor.

Thus $\det T_g = 0$ if and only if g disallows perfect reconstruction (cf. Corollary 2.2.8), regardless whether it is used as low-pass or as high-pass filter.

This result is already nice but we can improve it considerably. We will now derive a factorisation of $\det T_{g*h}$ and besides this will yield an efficient computation of $\det T_g$.

3.1.5 Notation. Until the end of this section we want to consider polynomials with zero as least index where the leading coefficient need not to be zero. That is $(1, 2, 1)$ and $(1, 2, 1, 0)$ shall be different polynomials. To this end we attach a non-negative integer to each polynomial which tells the highest index of coefficients to be considered. It induces the domain of the mask, called $\operatorname{dom} h$. The domain $\operatorname{dom} h$ is a contiguous set of the form $\{0, \ldots, n\}$ with $n \geq 0$. The domain must address at least all non-zero coefficients of h, i.e. $h_j \neq 0 \Rightarrow j \in \operatorname{dom} h$. For any $h \in (\mathbb{Z} \to R) \times \mathbb{N}_0$ we introduce the notations

$$\begin{aligned}
\#h &= \max(\operatorname{dom} h) \\
\operatorname{first} h &= h_{\min(\operatorname{dom} h)} = h_0 \\
\operatorname{last} h &= h_{\max(\operatorname{dom} h)} = h_{\#h} \\
\forall j \in \mathbb{N}_0 \qquad (\operatorname{keepeven} h)_{2 \cdot j} &= h_{2 \cdot j} \\
\forall j \in \mathbb{N}_0 \qquad (\operatorname{keepeven} h)_{2 \cdot j + 1} &= 0 \\
\forall j \in \mathbb{N}_0 \qquad (\operatorname{keepodd} h)_{2 \cdot j} &= 0 \\
\forall j \in \mathbb{N}_0 \qquad (\operatorname{keepodd} h)_{2 \cdot j + 1} &= h_{2 \cdot j + 1} \quad .
\end{aligned}$$

On down-sampling the domain shrinks accordingly.

$$\#(h \downarrow 2) = \left\lfloor \frac{\#h}{2} \right\rfloor$$

The function roots computes the multiset of roots of a polynomial. That is with $\prod$ as convolutional product it holds

$$h = \operatorname{last} h \cdot \prod_{\alpha \in \operatorname{roots} h} (-\alpha, 1) \quad .$$

We continue with an observation about the structure of T_g: Reordering of the columns of T_g reveals the similarity to a SYLVESTER matrix.

3.1.6 Definition (SYLVESTER matrix). Given the polynomials g and h from $(\mathbb{Z} \to R) \times \mathbb{N}_0$ with $n = \#g$ and $m = \#h$. Now the SYLVESTER *matrix* $S_{g,h}$ from $R^{\{0,\ldots,n+m-1\} \times (\{0\} \times \{0,\ldots,m-1\} \cup \{1\} \times \{0,\ldots,n-1\})}$ is defined by

$$\begin{aligned} (S_{g,h})_{j,(0,k)} &= g_{j-k} \\ (S_{g,h})_{j,(1,k)} &= h_{j-k} \end{aligned} \quad \text{[Str98, Definition 15.10].}$$

Considering the polynomials as columns we could also write informally

$$S_{g,h} = \begin{pmatrix} g & g \to 1 & \ldots & g \to (m-1) & h & h \to 1 & \ldots & h \to (n-1) \end{pmatrix} \quad .$$

For instance for $\#g = 4$ and $\#h = 3$ the SYLVESTER matrix is

$$S_{g,h} = \begin{pmatrix} g_0 & 0 & 0 & h_0 & 0 & 0 & 0 \\ g_1 & g_0 & 0 & h_1 & h_0 & 0 & 0 \\ g_2 & g_1 & g_0 & h_2 & h_1 & h_0 & 0 \\ g_3 & g_2 & g_1 & h_3 & h_2 & h_1 & h_0 \\ g_4 & g_3 & g_2 & 0 & h_3 & h_2 & h_1 \\ 0 & g_4 & g_3 & 0 & 0 & h_3 & h_2 \\ 0 & 0 & g_4 & 0 & 0 & 0 & h_3 \end{pmatrix} \quad .$$

This matrix was introduced to describe the BEZOUT equation as simultaneous linear equations. For polynomials g, h, x, y with $\#x = \#h - 1$ and $\#y = \#g - 1$ it holds for the concatenation of x and y

$$S_{g,h} \cdot \begin{pmatrix} x \\ y \end{pmatrix} = g * x + h * y \quad .$$

This means for given g, h and p we can determine x and y with $g*x+h*y = p$ by solving the simultaneous linear equations $S_{g,h} \cdot \begin{pmatrix} x \\ y \end{pmatrix} = p$. (With the size of $S_{g,h}$ as defined above $\#p$ is limited to $\#g + \#h - 1$.) We know that the BEZOUT equation can be solved if and only if $\gcd(g, h)$ divides p. The question whether g and h have a non-trivial common divisor can be answered with the SYLVESTER matrix, too. Without constraints the equation $g * x + h * y = 0$ is e.g. fulfilled by $x = h$, $y = -g$. But if we restrict the degrees of x and y by $\#x < \deg h$ and $\#y < \deg g$ then g and h must have a common divisor in order to allow non-trivial solutions. Non-trivial solutions of the homogenous equation $S_{g,h} \cdot \begin{pmatrix} x \\ y \end{pmatrix} = 0$ are possible if and only if $S_{g,h}$ is singular, or equivalently $\det S_{g,h} = 0$. This determinant got the name resultant.

3.1.7 Definition (Resultant). Given the polynomials g and h the *resultant* is defined as

$$\operatorname{res}(g, h) = \det S_{g,h}$$

[Str98, Definition 15.12].

The resultant can be used to eliminate variables in a multivariate polynomial equation system. Let $g(y)$ and $h(y)$ be univariate polynomials parametrised by a tuple y of some variables. So, g and h actually are multivariate. If we want to find x and y such that both $E\left(g(y)\right)(x) = 0$ and $E\left(h(y)\right)(x) = 0$, then this

is only possible if $g(y)$ and $h(y)$ share some roots. This is equivalent to the condition $\operatorname{res}(g(y), h(y)) = 0$. If $(y, x) \mapsto E\big(g(y)\big)(x)$ and $(y, x) \mapsto E\big(h(y)\big)(x)$ are multivariate polynomial functions then $y \mapsto \operatorname{res}(g(y), h(y))$ is also a multivariate polynomial function with one variable less. If more polynomial equations are available then this method can be used to successively eliminate variables.

We do not need this application but we want to use other properties of the resultant to show our factor theorem.

3.1.8 Lemma (Properties of the resultant).
Claim.

$$\operatorname{res}(g,h) = 0 \Leftrightarrow g \text{ and } h \text{ have a non-trivial common divisor} \tag{3.1.1}$$
$$\operatorname{res}(g,h) = (-1)^{\#g \cdot \#h} \cdot \operatorname{res}(h,g) \tag{3.1.2}$$
$$\operatorname{res}(g_-,h) = \operatorname{res}(h_-,g) \tag{3.1.3}$$
$$\#g = 0 \Rightarrow \operatorname{res}(g,h) = (\operatorname{first} g)^{\#h} \tag{3.1.4}$$
$$\#g = \#h + \#s \Rightarrow \operatorname{res}(g,h) = \operatorname{res}(g + s * h, h) \tag{3.1.5}$$
$$\operatorname{res}(g,(0,h_0,h_1,\ldots,h_n)) = \operatorname{first} g \cdot \operatorname{res}(g,(h_0,h_1,\ldots,h_n)) \tag{3.1.6}$$
$$\operatorname{res}(g,(h_0,h_1,\ldots,h_n,0)) = \operatorname{last} g \cdot \operatorname{res}(g,(h_0,h_1,\ldots,h_n)) \tag{3.1.7}$$

$$\forall A \in \mathbb{R}^{2\times 2} \wedge \#g = \#h \wedge \begin{pmatrix} x \\ y \end{pmatrix} = A \cdot \begin{pmatrix} g \\ h \end{pmatrix} \Rightarrow$$

$$\operatorname{res}(x,y) = (\det A)^{\#g} \cdot \operatorname{res}(g,h) \tag{3.1.8}$$
$$\operatorname{res}(\operatorname{keepeven} g, \operatorname{keepodd} g) = \operatorname{first} g \cdot \operatorname{last} g \cdot \operatorname{res}(g \downarrow 2, g \leftarrow 1 \downarrow 2)^2 \tag{3.1.9}$$
$$\operatorname{res}(g,g_-) = (-2)^{\#g} \cdot \operatorname{first} g \cdot \operatorname{last} g \cdot \operatorname{res}(g \downarrow 2, g \leftarrow 1 \downarrow 2)^2 \tag{3.1.10}$$
$$\operatorname{last} g \neq 0 \wedge \operatorname{last} h \neq 0 \Rightarrow \operatorname{res}(g,h) = \operatorname{last} h^{\#g} \cdot \operatorname{last} g^{\#h} \cdot \prod_{\alpha \in \operatorname{roots} g} \prod_{\beta \in \operatorname{roots} h} (\beta - \alpha) \tag{3.1.11}$$
$$\operatorname{res}(g * h, p) = \operatorname{res}(g,p) \cdot \operatorname{res}(h,p) \tag{3.1.12}$$
$$\det T_g = (-1)^{\left\lfloor \frac{\#g+1}{4} \right\rfloor} \cdot \operatorname{first} g \cdot \operatorname{last} g \cdot \operatorname{res}(g \downarrow 2, g \leftarrow 1 \downarrow 2) \tag{3.1.13}$$

Proof.

- (3.1.1) was already shown above.
- (3.1.2) is because of swapping columns in a matrix changes the sign of the determinant.
- (3.1.3) results from (3.1.2) and the fact that changing the sign of a column or a row changes the sign of the determinant.
- (3.1.4) describes the determinant of a scaled identity matrix.
- (3.1.5) is a consequence of the fact that adding linear combinations of some columns to a different column does not change the determinant.
- (3.1.6) can be verified by expanding the determinant with respect to the first row.
- (3.1.7) can be verified by expanding the determinant with respect to the last row.
- (3.1.8) is due to multiple application of the linear transformation to all pairs of corresponding columns and the determinant product theorem.
- (3.1.9) describes the effect of reordering rows and columns in the SYLVESTER matrix.
- (3.1.10) is a result from (3.1.8) and (3.1.9) because $\begin{pmatrix} g \\ g_- \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \cdot \begin{pmatrix} \operatorname{keepeven} g \\ \operatorname{keepodd} g \end{pmatrix}$ and $\det \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} = -2$.
- (3.1.11) is proven in [Str98, Definition 15.14]. The product is obviously zero if and only if g and h share roots. Although it may contain factors which are not in the ring of coefficients of g and h the product belongs to this ring. This is because the product is invariant with respect to arbitrary permutations of the roots. [Str98, Definition 17.2]

- (3.1.12) is justified by (3.1.11) and the fact that for all polynomials g and h the roots of $g * h$ are those of g and of h: $\operatorname{roots}(g * h) = \operatorname{roots} g \cup \operatorname{roots} h$.
- (3.1.13) follows from expanding $\det T_g$ with respect to the first and the last row yielding the factors $\operatorname{first} g$ and $\operatorname{last} g$, then reordering the columns.

□

3.1.9 Remark (Fast computation of the resultant). Together, (3.1.4), (3.1.5) and (3.1.7) allow us to use the EUCLID*ean algorithm* for fast computation of the resultant. [Str98, Definition 15.12] Assumed that g is the larger polynomial we can cancel some of the coefficients of g according to (3.1.5). With (3.1.7) we can reduce the number of coefficients g. As soon as the shortened g becomes shorter than h we exchange g and h. The cancellation is repeated until one polynomial reaches degree 0, then we apply (3.1.4) and stop.

With (3.1.13) we can compute $\det T_g$ efficiently.

Now we have all prerequisites for improving Theorem 3.1.3.

3.1.10 Theorem.
Claim.

$$\det T_g \cdot \det T_h \cdot \operatorname{res}(g_-, h) = \det T_{g*h}$$

Proof.

$$\begin{aligned}
\operatorname{res}(g*h, (g*h)_-) &= \operatorname{res}(g, (g*h)_-) \cdot \operatorname{res}(h, (g*h)_-) && \text{(3.1.12)}\\
&= \operatorname{res}(g, g_-) \cdot \operatorname{res}(g, h_-) \cdot \operatorname{res}(h, g_-) \cdot \operatorname{res}(h, h_-) && \text{(3.1.12)}\\
&= \operatorname{res}(g, g_-) \cdot \operatorname{res}(h, h_-) \cdot \operatorname{res}(g, h_-)^2 && \text{(3.1.3)}\\
(-2)^{\#(g*h)} \cdot \operatorname{first}(g*h) \cdot \operatorname{last}(g*h) \cdot \operatorname{res}((g*h) \lfloor 2, (g*h) \leftarrow 1 \lfloor 2)^2 &&&\\
&= (-2)^{\#g} \cdot \operatorname{first} g \cdot \operatorname{last} g \cdot \operatorname{res}(g \lfloor 2, g \leftarrow 1 \lfloor 2)^2 \cdot &&\\
&\quad (-2)^{\#h} \cdot \operatorname{first} h \cdot \operatorname{last} h \cdot \operatorname{res}(h \lfloor 2, h \leftarrow 1 \lfloor 2)^2 \cdot \operatorname{res}(g, h_-)^2 && \text{(3.1.10)}\\
\operatorname{first}(g*h)^2 \cdot \operatorname{last}(g*h)^2 \cdot \operatorname{res}((g*h) \lfloor 2, (g*h) \leftarrow 1 \lfloor 2)^2 &&&\\
&= \operatorname{first} g^2 \cdot \operatorname{last} g^2 \cdot \operatorname{res}(g \lfloor 2, g \leftarrow 1 \lfloor 2)^2 \cdot &&\\
&\quad \operatorname{first} h^2 \cdot \operatorname{last} h^2 \cdot \operatorname{res}(h \lfloor 2, h \leftarrow 1 \lfloor 2)^2 \cdot \operatorname{res}(g, h_-)^2 &&\\
\left|\operatorname{first}(g*h) \cdot \operatorname{last}(g*h) \cdot \operatorname{res}((g*h) \lfloor 2, (g*h) \leftarrow 1 \lfloor 2)\right| &&&\\
&= \left|\operatorname{first} g \cdot \operatorname{last} g \cdot \operatorname{res}(g \lfloor 2, g \leftarrow 1 \lfloor 2)\right| \cdot &&\\
&\quad \left|\operatorname{first} h \cdot \operatorname{last} h \cdot \operatorname{res}(h \lfloor 2, h \leftarrow 1 \lfloor 2) \cdot \operatorname{res}(g, h_-)\right| &&\\
\det T_{g*h} &= \det T_g \cdot \det T_h \cdot \operatorname{res}(g_-, h) && \text{(3.1.13)}
\end{aligned}$$

□

3.2 Lifting scheme

In this section we want to derive the *lifting scheme* [DS98]. The lifting scheme is a parametrisation of perfect reconstruction filter banks. This means, every filter bank generated by lifting is a perfect reconstruction filter bank, and in turn every perfect reconstruction filter bank can be represented by lifting. We need this parametrisation in order to match the shape of a function with a discrete wavelet while asserting perfect reconstruction.

Lifting is tightly connected to the EUCLIDean algorithm so we will shortly introduce it.

3.2.1 EUCLIDean algorithm

3.2.1 Definition (Greatest common divisor). Given two elements h and g of an EUCLIDean ring, the element u is called a *greatest common divisor* if all common common divisors of h and g divide u.

All greatest common divisors of two elements differ only by a unit factor.

The EUCLID*ean algorithm* computes a greatest common divisor u of two elements h and g. Additionally it finds elements x and y that solve the BEZOUT equation

$$h \cdot y + g \cdot x = u$$

and it generates a list of computation steps that we will call *lifting step*s later.

3.2.2 Definition (EUCLIDean algorithm). The two elements h and g from an EUCLIDean ring are the inputs. We compute

$$\begin{aligned} p_0 &= h \\ p_1 &= g \\ p_{j+2} &= p_j \bmod p_{j+1} \\ &= p_j - p_{j+1} \cdot s_j \quad . \end{aligned}$$

This is repeated until $p_{n+1} = 0$. The result of the algorithm is p_n. [Str98, Example 1.26c]

3.2.3 Lemma.
Claim. The result u of the EUCLIDean algorithm applied to g and h is a greatest common divisor of g and h.

Proof. We show the invariant that each pair (p_j, p_{j+1}) has the same common divisors as (g, h). This is shown by induction. The base case $p_0 = h, p_1 = g$ is trivial. The induction step consists of the observation that the pair (p_{j+1}, p_{j+2}) has the same common divisors as (p_j, p_{j+1}). This is true because every common divisor of (p_{j+1}, p_{j+2}) is a common divisor of (p_j, p_{j+1}) and vice versa. This in turn is provided by the fact that p_j and p_{j+2} differ by a multiple of p_{j+1}, namely $p_{j+2} - p_j = p_{j+1} \cdot s_j$.

Because we have an EUCLIDean ring, there is a size function f such that the remainder has always a size smaller than the divisor. This implies $f(p_{j+2}) < f(p_{j+1})$ which let the algorithm terminate with $f(p_{n+1}) = 0$, this means $p_{n+1} = 0$. Thus the set of common divisors of g and h is identical to the set of common divisors of p_n and 0. All ring elements are divisors of 0. Consequently the set consists entirely of divisors of p_n. A divisor of p_n which is divided by all divisors of p_n is clearly p_n itself, which implies that p_n is the greatest common divisor of p_n and p_{n+1}. □

The elements s_j can be used to re-assemble the pair (h, g) from scratch.

$$\begin{pmatrix} p_j \\ p_{j+1} \end{pmatrix} = \begin{pmatrix} s_j & 1 \\ 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} p_{j+1} \\ p_{j+2} \end{pmatrix}$$

$$\begin{pmatrix} h \\ g \end{pmatrix} = \prod_{j=0}^{n-1} \begin{pmatrix} s_j & 1 \\ 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} u \\ 0 \end{pmatrix}$$

This is the idea which underlies the lifting decomposition.

3.2.2 Translation invariant lifting

At the beginning we want to derive a lifting scheme for the translation invariant discrete wavelet transform as introduced in Section 2.2.2. That is we have the primal filters h and g and search for dual filters $\widetilde{h}$ and $\widetilde{g}$ with

$$h * \widetilde{h} + g * \widetilde{g} = \delta \qquad .$$

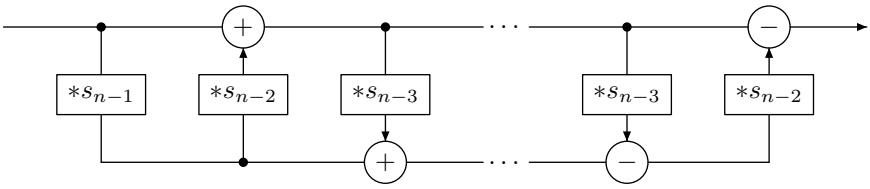


Figure 3.2: A translation invariant wavelet transform implemented according to the lifting scheme. The matrices of the form $\begin{pmatrix} s_j & \delta \\ \delta & 0 \end{pmatrix}$ would have to be represented by crossings. We resolve these crossings by swapping the channels after each lifting step.

Since δ is a convolutional unit we know that δ is a greatest common divisor of h and g and thus every common divisor of h and g is a unit.

Because of that we already know that the EUCLIDean algorithm will return a unit u given a perfect reconstructible filter pair (h, g). We are more interested in the dual filters $\widetilde{h}$ and $\widetilde{g}$ and the representation of the wavelet transform implied by the algorithm. This representation consists entirely of a kind of addition matrices, which can be considered as convolve-and-accumulate steps.

$$\begin{aligned}
\begin{pmatrix} h*x \\ g*x \end{pmatrix} &= \prod_{j=0}^{n-1} \begin{pmatrix} s_j & \delta \\ \delta & 0 \end{pmatrix} \circledast \begin{pmatrix} u*x \\ 0 \end{pmatrix} \\
y_{n+1} &= 0 \\
y_n &= u*x \\
y_j &= y_{j+1} * s_j + y_{j+2} \\
h*x &= y_0 \\
g*x &= y_1
\end{aligned}$$

This representation can be inverted easily step by step.

$$\begin{aligned}
\begin{pmatrix} u*x \\ 0 \end{pmatrix} &= \prod_{j=n-1}^{0} \begin{pmatrix} 0 & \delta \\ \delta & -s_j \end{pmatrix} \circledast \begin{pmatrix} h*x \\ g*x \end{pmatrix} \\
y_0 &= h*x \\
y_1 &= g*x \\
y_{j+2} &= y_j - y_{j+1} * s_j \\
y_n &= u*x
\end{aligned}$$

We see that the last step (i.e. $j = n-1$) can be omitted because it only yields $y_{n+1} = 0$. This means that reconstruction filters $\widetilde{h}$ and $\widetilde{g}$ constructed by this method tend to be shorter than the original filters h and g.

The matrices of the form $\begin{pmatrix} s_j & \delta \\ \delta & 0 \end{pmatrix}$ somehow mean that we swap the channels in each lifting step. We can avoid these swaps by changing the direction of two subsequent matrices.

$$\begin{aligned}
\begin{pmatrix} s_j & \delta \\ \delta & 0 \end{pmatrix} \circledast \begin{pmatrix} s_{j+1} & \delta \\ \delta & 0 \end{pmatrix} &= \begin{pmatrix} s_j & \delta \\ \delta & 0 \end{pmatrix} \circledast \begin{pmatrix} 0 & \delta \\ \delta & 0 \end{pmatrix} \circledast \begin{pmatrix} 0 & \delta \\ \delta & 0 \end{pmatrix} \circledast \begin{pmatrix} s_{j+1} & \delta \\ \delta & 0 \end{pmatrix} \\
&= \begin{pmatrix} \delta & s_j \\ 0 & \delta \end{pmatrix} \circledast \begin{pmatrix} \delta & 0 \\ s_{j+1} & \delta \end{pmatrix}
\end{aligned}$$

This modification is also used for our illustrations of the lifting scheme. The analysis and synthesis using lifting is illustrated in Figure 3.2 this way.

Eventually we obtain the dual filters by omitting the application to h and g.

$$u = \begin{pmatrix} 0 & 1 \end{pmatrix} \cdot \prod_{j=n-1}^{1} \begin{pmatrix} 0 & \delta \\ \delta & -s_j \end{pmatrix} \circledast \begin{pmatrix} h \\ g \end{pmatrix}$$

$$\begin{pmatrix} \widetilde{h} & \widetilde{g} \end{pmatrix} = \begin{pmatrix} 0 & u^{*-1} \end{pmatrix} \circledast \prod_{j=n-1}^{1} \begin{pmatrix} 0 & \delta \\ \delta & -s_j \end{pmatrix}$$

An interesting fact is that the reconstruction by the lifting scheme allows more freedom. The analysis transform consists of a sequence of operations and the synthesis transform consists of the reversed sequence of inverted operations. The lifting step $y_j = y_{j+2} + y_{j+1} * s_j$ and its inverse $y_{j+2} = y_j - y_{j+1} * s_j$ can be generalised to $y_j = y_{j+2} + f(y_{j+1}, s_j)$ and $y_{j+2} = y_j - f(y_{j+1}, s_j)$ where $f \in (\mathbb{Z} \to \mathbb{R})^2 \to (\mathbb{Z} \to \mathbb{R})$. There are no restrictions on f (only that it is really a function, that is its values depend exclusively on the arguments). Of course the choice of f as well as the computational realization of "$+$" and "$-$" have an influence on numeric stability. A useful application of this observation is to choose f as a convolution with rounding. This way bit-exact reconstructions are possible with integer arithmetic. [UVWJ+98]

So far we did not explain what the mod operation means for our filters, which have both negative and positive indices. They are not ordinary polynomials but LAURENT polynomials so there is no natural choice for a division. However if the polynomials are symmetric it is natural to choose symmetric quotients and remainders. This choice is unique. We are not aware of a reasonable criterion for the general case. [DS98]

3.2.3 Lifting in the presence of down-sampling

The lifting scheme for the critically sampled DWT is rather similar to that for the translation invariant transform as shown in the previous section. To be historically correct, the lifting scheme for the normal DWT is the original one. [DS98] But there is a bit more we must pay attention to, thus we consider it now.

Now we do not only have a restriction for $h * \widetilde{h} + g * \widetilde{g}$ but also the condition $h * \widetilde{h}_- + g * \widetilde{g}_- = 0$ (see Remark 2.2.15). We will see that the lifting scheme with down-sampling can be developed nicely if we decompose the down-sampled parts of h and g, namely h_e and g_e instead of h and g (see Definition 2.2.4).

Given a complementary filter pair (h, g) we apply the EUCLIDean algorithm to h_e and g_e. This yields a sequence of filters s_j with

$$\sqrt{2} \cdot \begin{pmatrix} h_\mathsf{e} \\ g_\mathsf{e} \end{pmatrix} = \prod_{j=0}^{n-1} \begin{pmatrix} s_j & \delta \\ \delta & 0 \end{pmatrix} \circledast \begin{pmatrix} u \\ 0 \end{pmatrix} \quad .$$

The wavelet transform is completely determined by the poly-phase matrix, thus we extend the columnar matrices on both sides. On the right side we introduce some still unknown filters q_0 and q_1.

$$\sqrt{2} \cdot \begin{pmatrix} h_\mathsf{e} & h_\mathsf{o} \\ g_\mathsf{e} & g_\mathsf{o} \end{pmatrix} = \prod_{j=0}^{n-1} \begin{pmatrix} s_j & \delta \\ \delta & 0 \end{pmatrix} \circledast \begin{pmatrix} u & q_0 \\ 0 & q_1 \end{pmatrix}$$

Using the determinant product theorem and the definition of filter complementarity (Definition 2.2.9) we compute the determinants of both sides of the equation.

$$\delta = (-1)^n \cdot u * q_1$$
$$q_1 = (-1)^n \cdot u^{*-1}$$

With $s_n = (-1)^n \cdot u * q_0$ we can almost complete the lifting decomposition.

$$\sqrt{2} \cdot \begin{pmatrix} h_\mathsf{e} & h_\mathsf{o} \\ g_\mathsf{e} & g_\mathsf{o} \end{pmatrix} = \prod_{j=0}^{n} \begin{pmatrix} s_j & \delta \\ \delta & 0 \end{pmatrix} \circledast \begin{pmatrix} 0 & (-1)^n \cdot u^{*-1} \\ u & 0 \end{pmatrix}$$

If u is not exactly δ but a scalar multiple $\frac{1}{\alpha} \cdot \delta$ then we can completely decompose the flipped diagonal matrix into four final lifting steps.

$$\begin{pmatrix} 0 & (-1)^n \cdot \alpha \\ \frac{1}{\alpha} & 0 \end{pmatrix} = \begin{pmatrix} d & 1 \\ 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} c & 1 \\ 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} b & 1 \\ 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} a & 1 \\ 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} 0 & (-1)^n \\ 1 & 0 \end{pmatrix}$$

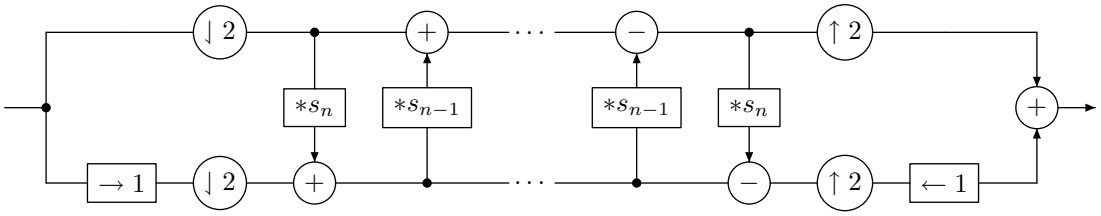


Figure 3.3: A subband coder implemented according to the lifting scheme. The crossings in the flowchart are suppressed as in Figure 3.2

According to [DS98] we can choose

$$\begin{aligned} a &= 1 \\ b &= \alpha - 1 \\ c &= -\frac{1}{\alpha} \\ d &= \alpha - \alpha^2 \quad . \end{aligned}$$

But there is one degree of freedom which can be used to reduce the magnitude of values in the lifting filters, reducing the danger of numerical cancellations. In [Thi01, Section 3.2.1, Paragraph "Weighting by lifting"] a variant is derived where for $\alpha \leq 1$ we can choose

$$\begin{aligned} a &= -\sqrt{(2+\alpha)\cdot(1-\alpha)\cdot\alpha} \\ b &= \sqrt{\frac{1-\alpha}{(2+\alpha)\cdot\alpha}} \\ c &= \sqrt{\frac{(2+\alpha)\cdot(1-\alpha)}{\alpha}} \\ d &= -\sqrt{\frac{(1-\alpha)\cdot\alpha}{2+\alpha}} \quad . \end{aligned}$$

For $\alpha > 1$ we work with $\frac{1}{\alpha}$ and apply the reverse lifting sequence. These four additional lifting steps can be appended to the lifting sequence s with

$$s_{n+1} = (\boldsymbol{d}) \quad s_{n+2} = (\boldsymbol{c}) \quad s_{n+3} = (\boldsymbol{b}) \quad s_{n+4} = (\boldsymbol{a})$$

leading to a total number of $n + 5$ lifting steps.

Summarised the transformation using lifting consists of the following steps.

1. Split the input x into the even indexed part $x \downarrow 2$ and the odd indexed part $(x \to 1) \downarrow 2$. This alone can be considered as one level of the discrete wavelet transform with respect to the *lazy wavelet* (or better the *lazy filter bank*). The lazy wavelet basis consists entirely of DIRAC impulses and is not considered as true wavelet basis, because DIRAC impulses are not functions.
2. After the splitting the lifting steps are applied as in the previous section. We obtain the signal filtered by both h and g in the down-sampled forms.

This interpretation of lifting is illustrated in Figure 3.3.

3.2.4 Remark. Instead of applying the lifting filters to down-sampled signals you can also up-sample the lifting filters and apply them to the original input signal. You have to down-sample the two last signals after this lifting procedure in order to get the low and the high band.

$$\sqrt{2}\cdot\begin{pmatrix} h_{\mathrm{e}}\uparrow 2 & h_{\mathrm{o}}\uparrow 2\\ g_{\mathrm{e}}\uparrow 2 & g_{\mathrm{o}}\uparrow 2\end{pmatrix}=\prod_{j=0}^{n+4}\begin{pmatrix} s_j\uparrow 2 & \delta\\ \delta & 0\end{pmatrix}\circledast\begin{pmatrix} 0 & (-1)^n\cdot\delta\\ \delta & 0\end{pmatrix}$$

$$\begin{aligned}\sqrt{2}\cdot\begin{pmatrix} h*x\\ g*x\end{pmatrix}&=\sqrt{2}\cdot\begin{pmatrix} h_{\mathrm{e}}\uparrow 2 & h_{\mathrm{o}}\uparrow 2\\ g_{\mathrm{e}}\uparrow 2 & g_{\mathrm{o}}\uparrow 2\end{pmatrix}\circledast\begin{pmatrix}\delta\\ \delta\to 1\end{pmatrix}*x\\ &=\prod_{j=0}^{n+4}\begin{pmatrix} s_j\uparrow 2 & \delta\\ \delta & 0\end{pmatrix}\circledast\begin{pmatrix} 0 & (-1)^n\cdot\delta\\ \delta & 0\end{pmatrix}\circledast\begin{pmatrix} x\\ x\to 1\end{pmatrix}\\ &=\prod_{j=0}^{n+4}\begin{pmatrix} s_j\uparrow 2 & \delta\\ \delta & 0\end{pmatrix}\circledast\begin{pmatrix} x\to 1\\ (-1)^n\cdot x\end{pmatrix}\end{aligned}$$

This allows to write the lifting composition as

$$\begin{aligned} y_{n+1}&=(-1)^n\cdot x\\ y_n&=x\to 1\\ y_j&=y_{j+1}*s_j\uparrow 2+y_{j+2}\\ \sqrt{2}\cdot h*x&=y_0\\ \sqrt{2}\cdot g*x&=y_1\quad .\end{aligned}$$

3.2.5 Remark. It is also possible to apply the EUCLIDean algorithm to the filters h_{e} and h_{o}, instead of h_{e} and g_{e}. Then you get the lifting decomposition in the order from lazy wavelet to the wanted wavelet. This implies that most of the information necessary for lifting is already contained in the low-pass filter.

3.2.6 Remark. Lifting allows conversion between any two complementary filter banks. The easiest way to achieve this is to decompose both filter banks down to the lazy filter bank. Then one of the decomposition step lists is appended in reversed order and with altered signs to the other list.

A more direct way even allows to convert between two non-complementary filter banks. But the premise is that their polyphase matrices have the same determinant which must be different from zero. If we have two polyphase matrices P_0 and P_1 and we search lifting filters s_j with

$$P_0=\prod_{j=0}^{n-1}\begin{pmatrix} s_j & \delta\\ \delta & 0\end{pmatrix}\circledast P_1$$

then we can compute $P_0\circledast {P_1}^{-1}$ e.g. with CRAMER's rule because the determinants can be divided. Consequently the convolutional determinant of $P_0\circledast {P_1}^{-1}$ is δ. Therefore the lifting decomposition can be successfully applied to this matrix quotient. We obtain the lifting filters as solution of

$$P_0\circledast {P_1}^{-1}=\prod_{j=0}^{n-1}\begin{pmatrix} s_j & \delta\\ \delta & 0\end{pmatrix}\quad .$$

Let us summarise some features of the lifting scheme:

- Lifting always allows for perfect reconstruction. Conversely, every perfect reconstruction filter bank can be represented by lifting. That is lifting is a parametrisation of perfect reconstruction filter banks.
- It is enough to have an invertible addition in order to assert perfect reconstruction down to the bit representation even for numerical instable filter banks.
- We have more freedom for the design of lifting steps. Certain non-linearities are possible while retaining perfect reconstructability.
- It speeds up computation by a factor of at least 2. This factor can only be achieved because lifting excludes filter banks that do not allow perfect reconstruction.
- It can be performed in-place. That is the signal can be transformed without requiring additional memory for temporary results. (Strictly spoken: Only constant amount of additional memory is needed).

The perfect reconstruction property is essential for our matching problem.

3.2.4 Lifting decomposition of CDF wavelets

In the previous sections we have shown using the EUCLIDean algorithm that every perfect reconstruction filter bank can be decomposed into lifting steps. It is an interesting question whether we are able to find explicit lifting decompositions for a whole class of (well-known) filter banks. There is already a table of lifting decompositions of some CDF filter banks [URB97]. Fortunately there is even a general formula for the lifting decomposition of the CDF wavelets.

The family of *CDF wavelet*s (COHEN-DAUBECHIES-FEAUVEAU) was developed in order to get a filter bank where the synthesis low-pass is a pure power of $\frac{1}{2}\cdot(\mathbf{1},1)$ and the analysis low-pass contains a specific power of $\frac{1}{2}\cdot(\mathbf{1},1)$ as convolutional factor [Dau92, Section 8.3.4]. Let n be the exponent of $\frac{1}{2}\cdot(\mathbf{1},1)$ in the low-pass filter h and $\widetilde{n}$ be the exponent of $\frac{1}{2}\cdot(\mathbf{1},1)$ in the low-pass filter $\widetilde{h}$ (which is also the exponent of $\frac{1}{2}\cdot(\mathbf{1},-1)$ in g). We will translate h in order to simplify subsequent calculations. Especially we want to renounce symmetry in order to have a simple power with no case distinctions for translations and the translation of h shall be independent of $\widetilde{n}$. For the CDF filter bank $n+\widetilde{n}$ must be even. (In general this is not necessary, as the perfect reconstruction filter bank consisting of $\frac{1}{2}\cdot(\mathbf{1},1)$ and $(\mathbf{0},1)$ shows.) Thus N defined by $N=\frac{n+\widetilde{n}}{2}$ is an integer. We want to abbreviate $\frac{1}{2}\cdot(\mathbf{1},1)$ with p which implies $\frac{1}{2}\cdot(\mathbf{1},-1)=p_-$.

$$\begin{aligned} h &= p^{*n} \\ g &= p_-^{*\widetilde{n}} * q \leftarrow N \end{aligned}$$

The problem is to determine q. We use the substitution $w=\frac{1}{4}\cdot(1,\mathbf{2},1)$ ($w=p^{*2}\leftarrow 1$) and start our derivation with the condition for perfect reconstruction (Corollary 2.2.14).

$$\begin{aligned} h*g_- - h_-*g &= \delta \rightarrow 1 \\ h*g_- &= (-1)^N\cdot p^{*n+\widetilde{n}} * q_- \leftarrow N \\ &= (-w)^{*N} * q_- \\ h_-*g &= \big(h*g_-\big)_- \\ &= (-w_-)^{*N} * q \qquad \big| \qquad w_- = \delta - w \\ &= (w-\delta)^{*N} * q \\ (-w)^{*N} * q_- - (w-\delta)^{*N} * q &= \delta \rightarrow 1 \\ (w-\delta)^{*N} * q &= -\delta \rightarrow 1 + (-w)^{*N} * q_- \\ (\delta-w)^{*N} * q &= (-1)^{N+1}\cdot\delta \rightarrow 1 + w^{*N} * q_- \end{aligned}$$

The next step is a kind of division by $(\delta-w)^{*N}$. We know about the series for the power function [KRG95]

$$\begin{aligned} (1+x)^\alpha &= \sum_{k=0}^{\infty}\binom{\alpha}{k}\cdot x^k \\ \binom{\alpha}{k} &= \prod_{j=1}^{k}\frac{\alpha-k+j}{j} \end{aligned}$$

which becomes the binomial formula for non-negative integral α. Because q is the solution of a BEZOUT equation we know that q must have finite degree. This means that the convolutional division $\Big((-1)^{N+1}\cdot\delta\rightarrow 1+w^{*N}*q_-\Big) \not* (\delta-w)^{*N}$ is defined. From the BEZOUT equation we know even more that q must have a degree smaller than N. It follows that we can truncate the series after the N-th

term. The term $w^{*N} * q_-$ must complement the remainder in the division after N steps. The solution

$$\begin{aligned} q &= (-1)^{N+1} \cdot \sum_{k=0}^{N-1} \binom{-N}{k} \cdot (-w)^{*k} \rightarrow 1 \\ &= (-1)^{N+1} \cdot \sum_{k=0}^{N-1} \binom{N+k-1}{k} \cdot w^{*k} \rightarrow 1 \\ g &= (-1)^{N+1} \cdot p_-^{*\widetilde{n}} * \sum_{k=0}^{N-1} \binom{-N}{k} \cdot (-w)^{*k} \leftarrow (N-1) \\ &= (-1)^{N+1} \cdot p_-^{*\widetilde{n}} * \sum_{k=0}^{N-1} \binom{N+k-1}{k} \cdot w^{*k} \leftarrow (N-1) \end{aligned} \tag{3.2.1}$$

can be verified by insertion into the condition we started on.

Note that there is also a generalisation of CDF wavelets to B-Splines of fractional order. That is the low-pass is a certain power $p^{*\alpha}$ with a fractional α. If α is not an integer these masks are infinite and not necessarily everywhere non-negative [UB99, UB00, BU00]. Other contributions on spline wavelets address orthogonal wavelet bases (BATTLE-LEMARIÉ wavelets) [Bat87, Lem88], semi-orthogonal bases [CW92, UAE92, UAE93] and shift orthogonal bases [UTA98].

In the next three theorems we want to state the lifting decomposition of wavelets of the CDF family. From Remark 3.2.5 we know that we need only one of the filters in order to compute all lifting steps except the last one. We develop the lifting sequence for the binomial mask $\left(\frac{1}{2} \cdot (\mathbf{1}, 1)\right)^{*n}$ because of its simple structure. The lifting composition will produce a high-pass filter which is shorter than the low-pass. But the BEZOUT equation for determining the shortest counterpart for the low-pass has a unique solution. We conclude that the lifting sequence will describe just the CDF-$n, (n \bmod 2)$ filter bank.

In the next two theorems we will present the structure of the intermediate filters of the lifting composition. In principle we could determine them by posing a BEZOUT problem on the high-pass, compute the un-lifted low-pass and iterate that procedure. Unfortunately the trick with the aborted power function series does no longer work in subsequent steps. So we have to guess the structure and prove it afterwards.

In the structure of the intermediate filters we will recognise how a factor $\frac{1}{4} \cdot (1, \mathbf{2}, 1)$ is latently added in each lifting step. In the last step all of these binomial factors are revealed. These first $\left\lceil \frac{n}{2} \right\rceil$ lifting steps which construct the binomial low-pass filter will generate a high-pass with only $n \bmod 2$ vanishing moments. The last step will add the remaining ones.

Note that for the two following theorems we will translate the filters more symmetrical in order to lift between the lazy wavelet and the CDF wavelet.

3.2.7 Theorem (Lifting decomposition of CDF-$n, 0$ for even n).
Prerequisite. Let a be the sequence which is recursively defined by

$$\begin{aligned} a_0 &= \frac{1}{n} \\ a_m &= \frac{1}{(n - 2 \cdot m) \cdot (n + 2 \cdot m) \cdot a_{m-1}} \\ &= \frac{1}{(n^2 - 4 \cdot m^2) \cdot a_{m-1}} \quad . \end{aligned}$$

It can also be written explicitly

$$\begin{aligned} a_0 &= \frac{1}{n} \\ a_1 &= \frac{n}{(n-2)\cdot(n+2)} \\ a_2 &= \frac{(n-2)\cdot(n+2)}{(n-4)\cdot n\cdot(n+4)} \\ &\vdots \\ a_m &= \prod_{j=-m}^{m} (n+2\cdot j)^{(-1)^{m+j+1}} \quad . \end{aligned}$$

Claim. With the lifting steps

$$\begin{aligned} x_{-1} &= (1, \mathbf{0}) \\ x_0 &= (\mathbf{1}) \\ x_{m+1} &= x_{m-1} + a_m \cdot (2\cdot m+1, \mathbf{0}, 2\cdot m+1) \leftarrow (-1)^m * x_m \end{aligned}$$

$x_{n/2}$ is essentially a binomial filter, concrete

$$x_{n/2} = \left(\frac{1}{2}, \mathbf{1}, \frac{1}{2}\right)^{*n/2} \leftarrow \left(\frac{n}{2} \bmod 2\right) \quad .$$

The filter pair $(2^{-n/2}\cdot x_{n/2}, 2^{n/2}\cdot x_{n/2-1})$ is that of the CDF-$n,0$ wavelet.

Proof. The problem becomes simpler if we translate the filters such that they are symmetric with respect to their origin. This means

$$y_m = x_m \rightarrow (m \bmod 2)$$

which fits into a modified lifting scheme

$$\begin{aligned} y_{-1} &= (\mathbf{1}) \\ y_0 &= (\mathbf{1}) \\ y_{m+1} &= y_{m-1} + a_m \cdot (2\cdot m+1, \mathbf{0}, 2\cdot m+1) * y_m \quad . \end{aligned}$$

and consequently $y_{n/2} = \left(\frac{1}{2}, \mathbf{1}, \frac{1}{2}\right)^{*n/2}$. We have simplified expressions but no longer real lifting steps because a lifting filter must be of the form $s \uparrow 2$.

In the next step we get rid of the fractions by multiplying each y_m by its denominator. For negative m the product must be generalised by suitable divisions.

$$z_{m+1} = \left(\prod_{j=0}^{m} (n - 2\cdot m + 4\cdot j)\right) \cdot y_{m+1} \tag{3.2.2}$$

Now multiply the lifting step with the product.

$$\begin{aligned} z_{-1} &= \frac{1}{n} \cdot \delta \\ z_0 &= \delta \\ z_{m+1} &= (n-2\cdot m)\cdot(n+2\cdot m)\cdot z_{m-1} + (2\cdot m+1)\cdot(1,\mathbf{0},1) * z_m \end{aligned} \tag{3.2.3}$$

By considering the expressions for some z_m we derive an assumption about a general formula for z_m. Due to the claim the sequence item $z_{n/2}$ must be a power of $(1, \mathbf{2}, 1)$. Actually, a structure in z_m becomes

visible only if it is expanded with respect to powers of $(1, \mathbf{2}, 1)$. Our new claim is

$$\begin{aligned}
w &= (1, \mathbf{2}, 1) \\
z_0 &= \delta \\
z_1 &= (n-2) \cdot \delta + w \\
z_2 &= (n-2) \cdot (n-4) \cdot \delta + 3 \cdot (n-4) \cdot w + 3 \cdot w^{*2} \\
z_3 &= (n-2) \cdot (n-4) \cdot (n-6) \cdot \delta + 6 \cdot (n-4) \cdot (n-6) \cdot w + 15 \cdot (n-6) \cdot w^{*2} + 15 \cdot w^{*3} \\
&\vdots \\
z_m &= \sum_{j=0}^{m} b_{m,j} \cdot w^{*j} \cdot \prod_{k=j+1}^{m} (n - 2 \cdot k) \quad . \qquad (3.2.4)
\end{aligned}$$

It occurs b the family of BESSEL polynomials (A1498 in [Slo03]), which is defined by

$$\begin{aligned}
\forall j \geq 0 \qquad & b_{m,j} = \frac{1}{j! \cdot 2^j} \cdot \prod_{k=1-j}^{j} (m + k) \\
\forall j < 0 \qquad & b_{m,j} = 0 \quad .
\end{aligned}$$

For $j > m$ it is $b_{m,j} = 0$ because the product contains a zero factor. For integral m and $j \in \{0, \ldots, m\}$ it holds

$$b_{m,j} = \frac{(m+j)!}{(m-j)! \cdot j! \cdot 2^j}$$

and with the relation $b_{-m,j} = b_{m-1,j}$ we can convert between negative and positive m.

The linear factors with respect to n cause that z_m is a convolutional multiple of the power $w^{*\mu}$ with $\mu = \begin{cases} \frac{n}{2} & : \frac{n}{2} \leq m \\ 0 & : \text{else} \end{cases}$. In particular $z_{n/2} = b_{n/2,n/2} \cdot w^{*n/2}$.

$$\begin{aligned}
z_{n/2} &= b_{n/2,n/2} \cdot w^{*n/2} \\
&= \frac{n!}{(n/2)! \cdot 2^{n/2}} \cdot w^{*n/2} \\
&= \left(\prod_{j=0}^{n/2-1} (1 + 2 \cdot j) \right) \cdot w^{*n/2} \\
&= \left(\prod_{j=0}^{n/2-1} (2 + 4 \cdot j) \right) \cdot \left(\frac{1}{2}, \mathbf{1}, \frac{1}{2} \right)^{*n/2}
\end{aligned}$$

The last equation is conform to (3.2.2). Thus if the explicit formula for z_m is confirmed then the theorem is proven.

In order to simplify writing we call the coefficients of the powers of w in (3.2.4) $c_{m,j}$ and rewrite the recursion (3.2.3) using these numbers.

$$\begin{aligned}
p_{m,j} &= \prod_{k=j+1}^{m} (n - 2 \cdot k) \\
c_{m,j} &= b_{m,j} \cdot p_{m,j} \\
z_m &= \sum_{j=0}^{m} c_{m,j} \cdot w^{*j} = \sum_{j \in \mathbb{Z}} c_{m,j} \cdot w^{*j} \\
z_{m+1} &= (n - 2 \cdot m) \cdot (n + 2 \cdot m) \cdot z_{m-1} + (2 \cdot m + 1) \cdot (w - (\mathbf{2})) * z_m \\
c_{m+1,j} &= (n - 2 \cdot m) \cdot (n + 2 \cdot m) \cdot c_{m-1,j} + (2 \cdot m + 1) \cdot (c_{m,j-1} - 2 \cdot c_{m,j})
\end{aligned}$$

We are now going to check whether the explicit representation of the intermediate steps fits into the recursion formula.

The base case consists of z_0 and z_1.

$$\begin{aligned} y_0 &= (\mathbf{1}) \\ y_1 &= y_{-1} + a_0 \cdot (1, \mathbf{0}, 1) * y_0 \\ &= \left(\frac{1}{n}, \mathbf{1}, \frac{1}{n}\right) \\ z_0 &= (\mathbf{1}) \\ z_1 &= n^2 \cdot z_{-1} + (1, \mathbf{0}, 1) * z_0 \\ &= (1, \boldsymbol{n}, 1) \\ &= (\boldsymbol{n - 2}) + (1, \mathbf{2}, 1) \end{aligned}$$

For $m \geq 2$ we check whether the recursion for z_m holds. For $j \leq 0$ and $j > m$ the coefficients $c_{m,j}$ are zero which allowed us to express z_m by the sum over all indices j. The recursion properties on $b_{m,j}$ are true for all integral j as well. So we do not need to distinguish between the regular case and $j \leq 0$ and $j > m$.

$$(m+1) \cdot (m+j) \cdot b_{m,j-1} = m \cdot (m-j+1) \cdot b_{m,j-1} + j \cdot (2 \cdot m + 1) \cdot b_{m,j-1}$$

because $b_{m,j} - b_{m-1,j} = (m-j+1) \cdot b_{m,j-1} = (m+j-1) \cdot b_{m-1,j-1}$

$$(m+1) \cdot (b_{m+1,j} - b_{m,j}) = m \cdot (b_{m,j} - b_{m-1,j}) + j \cdot (2 \cdot m + 1) \cdot b_{m,j-1}$$

$$-2 \cdot (m+1) \cdot b_{m+1,j} = 2 \cdot m \cdot b_{m-1,j} - 2 \cdot (2 \cdot m + 1) \cdot (b_{m,j} + j \cdot b_{m,j-1})$$

because $b_{m+1,j} = b_{m-1,j} + (2 \cdot m + 1) \cdot b_{m,j-1}$

$$b_{m+1,j} \cdot (n - 2 \cdot (m+1)) = (n + 2 \cdot m) \cdot b_{m-1,j} + (2 \cdot m + 1) \cdot (b_{m,j-1} \cdot (n - 2 \cdot j) - 2 \cdot b_{m,j})$$

multiply with $p_{m,j}$, i.e. $\prod_{k=j+1}^{m}(n - 2 \cdot k)$

$$\begin{aligned} b_{m+1,j} \cdot p_{m+1,j} = &(n - 2 \cdot m) \cdot (n + 2 \cdot m) \cdot b_{m-1,j} \cdot p_{m-1,j} \\ &+ (2 \cdot m + 1) \cdot \left(b_{m,j-1} \cdot p_{m,j-1} - 2 \cdot b_{m,j} \cdot p_{m,j}\right) \end{aligned}$$

□

3.2.8 Theorem (Lifting decomposition of CDF-$n, 1$ for odd n).
Prerequisite. Let a be the sequence which is recursively defined by

$$\begin{aligned} a_0 &= \frac{1}{n} \\ a_m &= \frac{1}{(n - 2 \cdot m + 1) \cdot (n + 2 \cdot m - 1) \cdot a_{m-1}} \\ &= \frac{1}{\left(n^2 - (2 \cdot m - 1)^2\right) \cdot a_{m-1}} \quad . \end{aligned}$$

It can also be written explicitly

$$\begin{aligned} a_0 &= \frac{1}{n} \\ a_1 &= \frac{n}{(n-1) \cdot (n+1)} \\ a_2 &= \frac{(n-1) \cdot (n+1)}{(n-3) \cdot n \cdot (n+3)} \\ &\vdots \\ a_m &= n^{-(-1)^m} \cdot \prod_{j=0}^{m} \left(n^2 - (2 \cdot j + 1)^2\right)^{(-1)^{m+j}} \quad . \end{aligned}$$

Claim. With the lifting steps

$$\begin{aligned} x_{-1} &= (1, \mathbf{0}) \\ x_0 &= (\mathbf{1}) \\ x_1 &= x_{-1} + x_0 * a_0 \cdot (\mathbf{1}) \\ x_{m+1} &= x_{m-1} + x_m * a_m \cdot (2 \cdot m - 1, \mathbf{0}, 2 \cdot m + 1) \leftarrow (-1)^m \end{aligned}$$

1. For $\frac{n+1}{2}$ even, let $m = \frac{n+1}{2}$ and $k = 0$.
2. For $\frac{n+1}{2}$ odd, let $m = \frac{n-1}{2}$ and $k = 1$.

$$x_{(n+1)/2} = 2^k \cdot \frac{\binom{2\cdot m}{m}}{\binom{m}{m/2}} \cdot \left(\frac{1}{4} \cdot (1, \mathbf{2}, 1)\right)^{*(n-1)/2} * \frac{1}{2} \cdot (\mathbf{1}, 1) \leftarrow k$$

Proof. As in the case of even order CDF wavelets we want to translate the filters in order to make them somehow symmetric. More precisely

$$y_m = x_m \rightarrow (m \bmod 2)$$

which leads to the modified lifting scheme

$$\begin{aligned} y_{-1} &= (\mathbf{1}) \\ y_0 &= (\mathbf{1}) \\ y_1 &= y_{-1} + y_0 * a_0 \cdot (\mathbf{0}, 1) \\ y_{m+1} &= y_{m-1} + y_m * a_m \cdot (2 \cdot m - 1, \mathbf{0}, 2 \cdot m + 1) \quad . \end{aligned}$$

Our new claim is that

- for $\frac{n+1}{2}$ even with $m = \frac{n+1}{2}$ and $k = 0$, and
- for $\frac{n+1}{2}$ odd with $m = \frac{n-1}{2}$ and $k = 1$

we obtain

$$y_{(n+1)/2} = 2^k \cdot \frac{\binom{2\cdot m}{m}}{\binom{m}{m/2}} \cdot \left(\frac{1}{4} \cdot (1, \mathbf{2}, 1)\right)^{*(n-1)/2} * \frac{1}{2} \cdot (\mathbf{1}, 1) \quad .$$

Multiplying y_m with the denominator leads to

$$\begin{aligned} z_{2\cdot m} &= y_{2\cdot m} \cdot 2 \cdot \prod_{j=1}^{m} \left(n^2 - (4 \cdot j - 3)^2\right) \\ z_{2\cdot m+1} &= y_{2\cdot m+1} \cdot 2 \cdot n \cdot \prod_{j=1}^{m} \left(n^2 - (4 \cdot j - 1)^2\right) \end{aligned} \tag{3.2.5}$$

and the recursion relation

$$\begin{aligned} z_{-1} &= \frac{2 \cdot n}{n^2 - 1} \cdot (\mathbf{1}) \\ z_0 &= (\mathbf{2}) \\ z_1 &= z_{-1} \cdot \left(n^2 - 1\right) + z_0 * (\mathbf{0}, 1) \\ z_{m+1} &= z_{m-1} \cdot \left(n^2 - (2 \cdot m - 1)^2\right) + z_m \cdot (2 \cdot m - 1, \mathbf{0}, 2 \cdot m + 1) \quad . \end{aligned}$$

With

$$\begin{aligned} w &= (1, \mathbf{2}, 1) \\ v &= (-1, \mathbf{0}, 1) \end{aligned}$$

our new claim is

$$
\begin{aligned}
z_0 &= 2 \cdot \delta \\
z_1 &= 2 \cdot (n-1) \cdot \delta + (w+v) \\
z_2 &= 2 \cdot (n-1) \cdot (n-3) \cdot \delta + 2 \cdot (n-3) \cdot (2 \cdot w + v) + 3 \cdot (w+v) * w \\
z_3 &= 2 \cdot (n-1) \cdot (n-3) \cdot (n-5) \cdot \delta + 3 \cdot (n-3) \cdot (n-5) \cdot (3 \cdot w + v) \\
&\quad +6 \cdot (n-5) \cdot (3 \cdot w + 2 \cdot v) * w + 15 \cdot (w+v) * w^{*2} \\
&\vdots \\
\forall m > 0 \quad z_m &= \sum_{j=0}^{m} \frac{(m+j-1)!}{(m-1)! \cdot 2^{j-1}} \cdot \left(\binom{m}{j} \cdot w^{*j} + \binom{m-1}{j-1} \cdot v * w^{*j-1} \right) \cdot \prod_{k=j+1}^{m} (n - 2 \cdot k + 1) \\
&= \sum_{j=0}^{m} \frac{2 \cdot b_{m,j}}{m+j} \cdot \left(m \cdot w^{*j} + j \cdot v * w^{*j-1} \right) \cdot \prod_{k=j+1}^{m} (n - 2 \cdot k + 1) \quad .
\end{aligned}
$$

If we can prove this we are done since

$$
\begin{aligned}
z_{(n+1)/2} &= \frac{2 \cdot b_{(n+1)/2,(n+1)/2}}{n+1} \cdot \left(\frac{n+1}{2} \cdot w^{*(n+1)/2} + \frac{n+1}{2} \cdot v * w^{*(n-1)/2} \right) \\
&= \frac{(n+1)!}{\frac{n+1}{2}! \cdot 2^{(n+1)/2}} \cdot (w+v) * w^{*(n-1)/2}
\end{aligned}
$$

1. for $\frac{n+1}{2}$ even let $m = \frac{n+1}{2}$, rewrite the coefficient of $(w+v) * w^{*m-1}$

$$
\begin{aligned}
&\frac{(2 \cdot m)!}{m! \cdot 2^m} \\
&\quad = \frac{\binom{2 \cdot m}{m}}{\binom{m}{m/2} \cdot 2^m} \cdot \frac{m!^2}{(m/2)!^2} \\
&\quad = \frac{\binom{2 \cdot m}{m}}{\binom{m}{m/2}} \cdot 2^{1-m} \cdot \left(\prod_{j=0}^{m/2-1} (2 \cdot (1 + 2 \cdot j) \cdot (m/2 + 1 + j)) \right) \\
&\quad = \frac{\binom{2 \cdot m}{m}}{\binom{m}{m/2}} \cdot 2^{1-2 \cdot m} \cdot \left(\prod_{j=0}^{m/2-1} (2 \cdot (m - 1 - 2 \cdot j)) \right) \cdot \left(\prod_{j=0}^{m/2-1} (2 \cdot (m + 2 + 2 \cdot j)) \right) \\
&\quad = \frac{\binom{2 \cdot m}{m}}{\binom{m}{m/2}} \cdot 2^{1-2 \cdot m} \cdot \left(\prod_{j=0}^{m/2-1} ((n - 1 - 4 \cdot j) \cdot (n + 1 + 4 \cdot j)) \right)
\end{aligned}
$$

2. for $\frac{n+1}{2}$ odd let $m = \frac{n-1}{2}$, rewrite the coefficient of $(w+v) * w^{*m}$

$$
\begin{aligned}
&\frac{(2\cdot(m+1))!}{(m+1)!\cdot 2^{m+1}}\\
&\quad=\frac{\binom{2\cdot m}{m}\cdot(2\cdot m+1)}{\binom{m}{m/2}\cdot 2^{m+1}}\cdot\frac{m!^2}{(m/2)!^2}\\
&\quad=\frac{\binom{2\cdot m}{m}}{\binom{m}{m/2}}\cdot 2^{-m}\cdot n\cdot\left(\prod_{j=0}^{m/2-1}(2\cdot(1+2\cdot j)\cdot(m/2+1+j))\right)\\
&\quad=\frac{\binom{2\cdot m}{m}}{\binom{m}{m/2}}\cdot 2^{-2\cdot m}\cdot n\cdot\left(\prod_{j=0}^{m/2-1}(2\cdot(m-1-2\cdot j))\right)\cdot\left(\prod_{j=0}^{m/2-1}(2\cdot(m+2+2\cdot j))\right)\\
&\quad=\frac{\binom{2\cdot m}{m}}{\binom{m}{m/2}}\cdot 2^{-2\cdot m}\cdot n\cdot\left(\prod_{j=0}^{m/2-1}((n-3-4\cdot j)\cdot(n+3+4\cdot j))\right)\quad .
\end{aligned}
$$

In both cases we end up with (3.2.5).

What remains to show is the correctness of the explicit formula for z_m. As in the case of even order n we will express the proof in terms of the coefficients $c_{m,j}$ of w^{*m} and of the coefficients $d_{m,j}$ of $v*w^{*m-1}$ in z_m. The recursions on $c_{m,j}$ and $d_{m,j}$ are mutually depending.

$$
z_m=\left(\boldsymbol{c_{m,0}}\right)+\sum_{j=1}^{m}\left(c_{m,j}\cdot w^{*j}+d_{m,j}\cdot v*w^{*j-1}\right)
$$

We introduce an abbreviation for the product.

$$
\begin{aligned}
p_{m,j}&=\prod_{k=j+1}^{m}(n-2\cdot k+1)\\
c'_{m,j}&=\begin{cases}\frac{2^{1-j}}{j!}\cdot\prod_{k=0}^{j-1}(m^2-k^2) & :j\geq 0\\ 0 & :j<0\end{cases}\\
d'_{m,j}&=\begin{cases}\frac{2^{1-j}}{(j-1)!}\cdot m\cdot\prod_{k=1}^{j-1}(m^2-k^2) & :j>0\\ 0 & :j\leq 0\end{cases}
\end{aligned}
$$

For $m+j\neq 0$ this is equivalent to

$$
\begin{aligned}
c'_{m,j}&=\frac{2\cdot b_{m,j}\cdot m}{m+j}\\
d'_{m,j}&=\frac{2\cdot b_{m,j}\cdot j}{m+j}\quad .\\
c_{m,j}&=c'_{m,j}\cdot p_{m,j}\\
d_{m,j}&=d'_{m,j}\cdot p_{m,j}
\end{aligned}
$$

$$
\begin{aligned}
v^{*2}&=w*(w-(\mathbf{4}))\\
z_m*w&=\sum_{j=0}^{m}\left(c_{m,j}\cdot w^{*j+1}+d_{m,j}\cdot v*w^{*j}\right)\\
z_m*v&=\sum_{j=0}^{m}\left(c_{m,j}\cdot v*w^{*j}+d_{m,j}\cdot(w-(\mathbf{4}))*w^{*j}\right)\\
z_{m+1}&=z_{m-1}\cdot\left(n^2-(2\cdot m-1)^2\right)+z_m\cdot(2\cdot m\cdot(w-(\mathbf{2}))+v)\\
c_{m+1,j}&=c_{m-1,j}\cdot\left(n^2-(2\cdot m-1)^2\right)+(2\cdot m\cdot(c_{m,j-1}-2\cdot c_{m,j})+d_{m,j-1}-4\cdot d_{m,j})\\
d_{m+1,j}&=d_{m-1,j}\cdot\left(n^2-(2\cdot m-1)^2\right)+(2\cdot m\cdot(d_{m,j-1}-2\cdot d_{m,j})+c_{m,j-1})
\end{aligned}
$$

We start the proof of correctness of the explicit representation of z_m with base cases z_0 and z_1.

$$\begin{aligned}
y_0 &= (\mathbf{1}) \\
y_1 &= y_{-1} + a_0 \cdot (\mathbf{0}, 1) * y_0 \\
&= \left(\mathbf{1}, \frac{1}{n}\right) \\
z_0 &= (\mathbf{2}) \\
z_1 &= (n^2 - 1) \cdot z_{-1} + (\mathbf{0}, 1) * z_0 \\
&= 2 \cdot (\boldsymbol{n}, 1) \\
&= 2 \cdot (\boldsymbol{n - 1}) + (1, \mathbf{2}, 1) + (-1, \mathbf{0}, 1)
\end{aligned}$$

For $m > 1$ we will track down the recursion on c and d simultaneously. The steps can be read in the order given here but the proof direction is reversed. This means that divisions are actually multiplications which is important in cases where a division by zero might occur.

Divide the recursion by $p_{m,j}$, i.e. $\prod_{k=j+1}^{m}(n - 2 \cdot k + 1)$.

$$\begin{aligned}
&c'_{m+1,j} \cdot (n - 2 \cdot m - 1) = \\
&\qquad c'_{m-1,j} \cdot (n + 2 \cdot m - 1) + (2 \cdot m \cdot c'_{m,j-1} + d'_{m,j-1}) \cdot (n - 2 \cdot j + 1) - 4 \cdot (m \cdot c'_{m,j} + d'_{m,j})
\end{aligned}$$

$$\begin{aligned}
&d'_{m+1,j} \cdot (n - 2 \cdot m - 1) = \\
&\qquad\qquad d'_{m-1,j} \cdot (n + 2 \cdot m - 1) + (2 \cdot m \cdot d'_{m,j-1} + c'_{m,j-1}) \cdot (n - 2 \cdot j + 1) - 4 \cdot m \cdot d'_{m,j}
\end{aligned}$$

Because $c'_{m+1,j} - c'_{m-1,j} = 2 \cdot m \cdot c'_{m,j-1} + d'_{m,j-1}$ and $d'_{m+1,j} - d'_{m-1,j} = 2 \cdot m \cdot d'_{m,j-1} + c'_{m,j-1}$ we can eliminate the coefficients of $n - 1$.

$$\begin{aligned}
-c'_{m+1,j} \cdot 2 \cdot m &= c'_{m-1,j} \cdot 2 \cdot m + (2 \cdot m \cdot c'_{m,j-1} + d'_{m,j-1}) \cdot 2 \cdot (1 - j) - 4 \cdot (m \cdot c'_{m,j} + d'_{m,j}) \\
-d'_{m+1,j} \cdot 2 \cdot m &= d'_{m-1,j} \cdot 2 \cdot m + (2 \cdot m \cdot d'_{m,j-1} + c'_{m,j-1}) \cdot 2 \cdot (1 - j) - 4 \cdot m \cdot d'_{m,j}
\end{aligned}$$

$$\begin{aligned}
0 &= (c'_{m+1,j} + c'_{m-1,j}) \cdot m + (2 \cdot m \cdot c'_{m,j-1} + d'_{m,j-1}) \cdot (1 - j) - 2 \cdot (m \cdot c'_{m,j} + d'_{m,j}) \\
0 &= (d'_{m+1,j} + d'_{m-1,j}) \cdot m + (2 \cdot m \cdot d'_{m,j-1} + c'_{m,j-1}) \cdot (1 - j) - 2 \cdot m \cdot d'_{m,j}
\end{aligned}$$

Divide by $\begin{cases} \frac{1}{j! \cdot 2^{j-1}} \cdot \prod_{k=2-j}^{j-2}(m + k) & : j \geq 0 \\ 0 & : j < 0 \end{cases}$
and $\begin{cases} \frac{1}{(j-1)! \cdot 2^{j-1}} \cdot \prod_{k=2-j}^{j-2}(m + k) & : j > 0 \\ 0 & : j \leq 0 \end{cases}$, respectively.

$$\begin{aligned}
0 &= (m + 1) \cdot m \cdot (m + j - 1) \cdot (m + j) + (m - 1) \cdot m \cdot (m - j + 1) \cdot (m - j) \\
&\quad + (2 \cdot m^2 + j - 1) \cdot (1 - j) \cdot 2 \cdot j - 2 \cdot (m^2 + j) \cdot (m + j - 1) \cdot (m - j + 1) \\
&= (m^2 + (j + 1) \cdot m + j) \cdot m \cdot (m + j - 1) + (m^2 - (j + 1) \cdot m + j) \cdot m \cdot (m - j + 1) \\
&\quad + (2 \cdot m^2 + j - 1) \cdot (1 - j) \cdot 2 \cdot j - 2 \cdot (m^2 + j) \cdot (m + j - 1) \cdot (m - j + 1) \\
&= (2 \cdot j \cdot m + j) \cdot m \cdot (m + j - 1) + (-2 \cdot j \cdot m + j) \cdot m \cdot (m - j + 1) \\
&\quad + (2 \cdot m^2 + j - 1) \cdot (1 - j) \cdot 2 \cdot j - 2 \cdot j \cdot (m + j - 1) \cdot (m - j + 1) \\
&= (2 \cdot m + 1) \cdot m \cdot (m + j - 1) + (-2 \cdot m + 1) \cdot m \cdot (m - j + 1) \\
&\quad + 2 \cdot (2 \cdot m^2 + j - 1) \cdot (1 - j) - 2 \cdot (m + j - 1) \cdot (m - j + 1) \\
&= (2 \cdot m^2 + j - 1) \cdot (m + j - 1) - (2 \cdot m^2 + j - 1) \cdot (m - j + 1) + 2 \cdot (2 \cdot m^2 + j - 1) \cdot (1 - j)
\end{aligned}$$

$$\begin{aligned}0 &= (m+j-1)\cdot(m+j)\cdot m + (m-j+1)\cdot(m-j)\cdot m\\ &\quad -2\cdot m\cdot(2\cdot j-1)\cdot(j-1) - 2\cdot m\cdot(m+j-1)\cdot(m-j+1)\\ &= (m+j-1)\cdot(m+j) + (m-j+1)\cdot(m-j)\\ &\quad -2\cdot(2\cdot j-1)\cdot(j-1) - 2\cdot(m+j-1)\cdot(m-j+1)\\ &= (m+j-1)\cdot(2\cdot j-1) + (m-j+1)\cdot(1-2\cdot j) - 2\cdot(2\cdot j-1)\cdot(j-1)\end{aligned}$$

□

3.2.9 Remark. Because of

$$\begin{aligned}\lim_{m\to\infty}\frac{\binom{2\cdot(m+1)}{m+1}}{\binom{2\cdot m}{m}} &= \lim_{m\to\infty}\frac{(2\cdot m+2)\cdot(2\cdot m+1)}{(m+1)^2}\\ &= 4\end{aligned}$$

the value of $\binom{2\cdot m}{m}$ grows asymptotically like 4^m. More precisely according to [Mol98] it holds

$$\frac{4^m}{\sqrt{4\cdot m+1}} \le \binom{2\cdot m}{m} \le \frac{4^m}{\sqrt{2\cdot m+1}} \quad .$$

Thus $\frac{\binom{2\cdot m}{m}}{\binom{m}{m/2}}$ is approximately $2^{m-1/2}$.

The previous theorems showed what steps are necessary to lift from a lazy filter bank to the one of the CDF-$n, (n \bmod 2)$ wavelet. How to get from there to CDF-$n, \widetilde{n}$?

We only need one lifting step and we can calculate it immediately by resolving the equation for the last lifting step to the lifting filter. Let (h, g') be the filter pair of CDF-$n, (n \bmod 2)$, and g be the high-pass of CDF-$n, \widetilde{n}$ there must be a filter s with

$$\begin{aligned}g &= g' + h * (s \uparrow 2)\\ s &= ((g-g') \not* h) \downarrow 2 \quad .\end{aligned}$$

We need a polynomial division but in Theorem 3.3.1 we will see that h is a divisor of $g-g'$ and that the result is a filter with zeros at odd indices. We prefer the notation $g \not* h$ to $g * h^{*-1}$ because the division of g by h is unique if there is no remainder, whereas $g * h^{*-1}$ suggests that there is a unique inverse of h which is in general not true.

From (3.2.1) we know both g and g' explicitly. Let $N = \frac{n+\widetilde{n}}{2}$, $M = \frac{n+\widetilde{n} \bmod 2}{2}$, $w = \frac{1}{4}\cdot(1, \mathbf{2}, 1)$ ($w = p^{*2} \leftarrow 1$).

$$\begin{aligned}&g - g'\\ &\quad = (-1)^{N+1}\cdot p_{-}^{*\widetilde{n}} * \sum_{j=0}^{N-1}\binom{-N}{j}\cdot(-w)^{*j} \leftarrow (N-1)\\ &\qquad\quad -(-1)^{M+1}\cdot p_{-}^{*\widetilde{n} \bmod 2} * \sum_{j=0}^{M-1}\binom{-M}{j}\cdot(-w)^{*j} \leftarrow (M-1)\\ &\quad = (-1)^{M+1}\cdot p_{-}^{*\widetilde{n} \bmod 2} \leftarrow (M-1)\\ &\qquad\quad * \left((-1)^{N-M}\cdot p_{-}^{*N-M} \leftarrow (N-M) * \sum_{j=0}^{N-1}\binom{-N}{j}\cdot(-w)^{*j} - \sum_{j=0}^{M-1}\binom{-M}{j}\cdot(-w)^{*j}\right)\\ &\quad = (-1)^{M+1}\cdot p_{-}^{*\widetilde{n} \bmod 2} \leftarrow (M-1) * \left(w_{-}^{*N-M} * \sum_{j=0}^{N-1}\binom{-N}{j}\cdot(-w)^{*j} - \sum_{j=0}^{M-1}\binom{-M}{j}\cdot(-w)^{*j}\right)\end{aligned}$$

$$
\begin{aligned}
(-1)^{M+1}\cdot(g-g') \not* p_{-}^{*\widetilde{n} \bmod 2} &\rightarrow (M-1)\\
&= (\delta-w)^{*N-M} * \sum_{j=0}^{N-1}\binom{-N}{j}\cdot(-w)^{*j} - \sum_{j=0}^{M-1}\binom{-M}{j}\cdot(-w)^{*j}\\
&= (\delta-w)^{*N-M} * \left(\sum_{j=0}^{\infty}\binom{-N}{j}\cdot(-w)^{*j} - \sum_{j=N}^{\infty}\binom{-N}{j}\cdot(-w)^{*j}\right)\\
&\quad - \left(\sum_{j=0}^{\infty}\binom{-M}{j}\cdot(-w)^{*j} - \sum_{j=M}^{\infty}\binom{-M}{j}\cdot(-w)^{*j}\right)\\
&= (\delta-w)^{*N-M} * \left((\delta-w)^{*-N} - \sum_{j=N}^{\infty}\binom{-N}{j}\cdot(-w)^{*j}\right)\\
&\quad -(\delta-w)^{*-M} + \sum_{j=M}^{\infty}\binom{-M}{j}\cdot(-w)^{*j}\\
&= \sum_{j=M}^{\infty}\binom{-M}{j}\cdot(-w)^{*j} - (\delta-w)^{*N-M} * \sum_{j=N}^{\infty}\binom{-N}{j}\cdot(-w)^{*j}
\end{aligned}
$$

The last expression does not seem to be much simpler than the first one. But now we can convince ourselves that w^{*M} is a factor of $g-g'$ and we can remove it.

$$
\begin{aligned}
(g-g') \not* w^{*M} &= -p_{-}^{*\widetilde{n} \bmod 2} *\\
&\left(\sum_{j=M}^{\infty}\binom{-M}{j}\cdot(-w)^{*(j-M)} - (\delta-w)^{*N-M} * \sum_{j=N}^{\infty}\binom{-N}{j}\cdot(-w)^{*(j-M)}\right) \leftarrow (M-1)
\end{aligned}
$$

Because of $h = p^{*n}$ or equivalently $w^{*M} = h * p^{*\widetilde{n} \bmod 2} \leftarrow M$ it holds

$$
\begin{aligned}
(g-g') \not* h &= -\left(\frac{1}{4}\cdot(\mathbf{1},0,-1)\right)^{*\widetilde{n} \bmod 2} \leftarrow (n+\widetilde{n} \bmod 2 - 1)\\
&\quad * \left(\sum_{j=M}^{\infty}\binom{-M}{j}\cdot(-w)^{*(j-M)} - (\delta-w)^{*N-M} * \sum_{j=N}^{\infty}\binom{-N}{j}\cdot(-w)^{*(j-M)}\right)\\
&= -\left(\frac{1}{4}\cdot(1,\mathbf{0},-1)\right)^{*\widetilde{n} \bmod 2} \leftarrow (n-1)\\
&\quad * \left(\sum_{j=M}^{\infty}\binom{-M}{j}\cdot(-w)^{*(j-M)} - (\delta-w)^{*N-M} * \sum_{j=N}^{\infty}\binom{-N}{j}\cdot(-w)^{*(j-M)}\right) \quad .
\end{aligned}
$$

Unfortunately even the last expression is not suitable for checking if the described filter is up-sampled (i.e. has the form $s \uparrow 2$). However this form is probably the most appropriate for computation. First compute the series with respect to w. Keep only the first $2\cdot(N-M)$ terms (or $\widetilde{n}-\widetilde{n} \bmod 2$) because the remaining ones vanish. Then evaluate the truncated series with respect to the signal representation $\frac{1}{4}\cdot(1,\mathbf{2},1)$ of w.

3.3 Optimal matches

3.3.1 State of the Art

The idea of creating wavelets matched to a pattern is not new. The topic is sporadically explored for at least ten years now, but there was no breakthrough for the application of wavelets for pattern detection so far.

Since a discrete wavelet function is composed by $\psi = 2 \cdot (g * \varphi) \downarrow 2$ (Definition 2.2.26) some authors argue that the shape of the wavelet ψ is dominated by the shape of the high-pass filter g while the low-pass filter h must preserve a smooth shape of φ. With this simplification GREINER [Gre96] constructs wavelets for classifications of textures. As matching criteria he employs the EUCLIDean norm of the difference between the high-pass filter and a sampled pattern and alternatively some criterion derived from a *principal component analysis*. These criteria together with the requirement of filter bank orthogonality yield a quadratic minimisation problem with non-linear constraints, which he solves with the NEWTON *method* applied to the LAGRANGE *multiplier* formulation. Additionally he shows that when dropping the orthogonality constraint one can design a filter bank with more than two channels which allows the separation of multiple textures in one transform.

ZHANG, DAVIDSON and WONG [ZDW04] consider the projection of a pattern f into a wavelet subspace of low resolution. They design an orthogonal wavelet transform by optimising the filters in the sense that the pattern can be represented essentially by large scale wavelets. Ideally if a match with respect to a scale is perfect and the pattern is wavelet transformed with the matched filter bank then the wavelet coefficients of the finer scales are zero. The authors of that article estimate the error made by substituting φ by h and formulate the optimisation criterion in terms of $h * h^*$ and $Qf * Qf^*$. They end up with a semi-definite programming problem which can be solved with interior point methods.

Also CHAPA and RAGHUVEER consider orthogonal wavelet bases [RC00]. Their approach can be interpreted as a generalisation of the frequency domain construction of MEYER wavelets. They separate the matching procedure into matching the absolute FOURIER spectra and matching the phase spectra of pattern and wavelet. The wavelet is band-limited. This ensures that the wavelet transform of the pattern with respect to the matched wavelet has significant coefficients in one scale only, independent from the translation of the input data. The band-limit constraint turns out to be rather restrictive such that $\left|\widehat{\psi}\right|$ is 1 in the interval $[\pi + \alpha, 2 \cdot (\pi - \alpha)]$ and 0 in the interval $[0, \pi - \alpha]$ and above $2 \cdot (\pi + \alpha)$ for some α from $[0, \frac{\pi}{3}]$. Because of symmetry this is also true for the respective negative intervals. For $\alpha = 0$ one obtains SHANNON's wavelet and for $\alpha = \frac{\pi}{3}$ one has the maximum flexibility. The match of the pattern to the absolute wavelet spectrum turns into a linear least squares problem with linear constraints. The phase is found via matching the group delay. This turns into a linear least squares problem, too.

In [GJP02] GUPTA, JOSHI, and PRASAD use a filter h as low-pass and $h^*\_ \rightarrow 1$ as highpass, just like in the orthogonal case (see Remark 2.2.17). The optimisation target is the energy of the difference between the lowest scale signal $\boldsymbol{x}_0 * \varphi$ and the first band signal $(\boldsymbol{y}_1 * \psi) \uparrow 2$, which is equal to $(\boldsymbol{x}_1 * \varphi) \uparrow 2$. The optimisation criterion is to find the filter h for a given pattern $\boldsymbol{x}_0$ which minimises the energy of $(\boldsymbol{x}_1 * \varphi) \uparrow 2$. This means that the pattern can be represented mostly by the wavelet coefficients at the first level and the scaling coefficients are of minor significance. The optimisation is considerably simplified to a linear equation system by neglecting orthogonality and smoothness constraints. The disadvantage is clearly that the method yields in general no perfect reconstruction filter banks.

An algebraical approach is introduced by DE VILLIERS, MICCHELLI, and SAUER [DVMS00]. They construct refinable functions which match a pattern exactly at integral positions. If x is a sampled pattern, then they ask for a mask h such that for the corresponding generator $Q\varphi = x$ holds. Due to Lemma 2.2.31 it holds

$$\begin{aligned} Q\varphi &= 2 \cdot (h * Q\varphi) \downarrow 2 \\ &= 2 \cdot (h \downarrow 2 * (Q\varphi) \downarrow 2 + (h \leftarrow 1) \downarrow 2 * (Q\varphi \rightarrow 1) \downarrow 2) \quad . \end{aligned}$$

This is a BEZOUT equation which can be solved with the EUCLIDean algorithm (Section 3.2.1) if and only if the greatest common divisor of $(Q\varphi) \downarrow 2$ and $(Q\varphi \rightarrow 1) \downarrow 2$ divides $Q\varphi$. The refinable function can also be reconstructed if not the values at integral arguments are given but the integrals over integer intervals, i.e. $x_i = \int_{(i,i+1)} \varphi$. It seems to be an open question how to modify a pattern minimally such that it matches the values of a refinable function at integral arguments.

3.3.2 Deriving a least squares problem

We will follow our approach of a matching algorithm based on lifting. It will roughly work this way: First we choose a generator and then we ask for a complementary wavelet which matches a pattern. The

constructed wavelet bases are in general not orthogonal. In Chapter 4 we will refine this method for preserving smoothness of the primal functions.

We start with the observation, that for a specific filter h there is only a restricted choice of complementary filters g. More precisely, the lifting steps as in Remark 3.2.4 are the only possibility to convert between filter banks which have one filter in common.

3.3.1 Theorem.
Prerequisite. The filters h and g are complementary as well as h and g'.
Claim. The filter g' can be derived from g by adding a multiple of h, where the factor polynomial contains only even powers.

$$\exists s \qquad g' - g = h * (s \uparrow 2)$$

Proof. First we will show that h divides $g' - g$.

From Corollary 2.2.14 it can be concluded that

$$\begin{aligned} h * g_{-} - h_{-} * g &= \delta \rightarrow 1 \\ h * g'_{-} - h_{-} * g' &= \delta \rightarrow 1 \end{aligned}$$

and thus both differences are equal

$$\begin{aligned} h * g'_{-} - h_{-} * g' &= h * g_{-} - h_{-} * g \\ h * \left(g'_{-} - g_{-}\right) &= h_{-} * \left(g' - g\right) \qquad . \end{aligned} \tag{3.3.1}$$

Because h divides the right hand side, it must divide left hand side, too. According to Corollary 2.2.8, h and h_{-} must be relatively prime and thus h must divide $g' - g$.

Now we will verify that the quotient s' of $(g' - g) \not\equiv h$ has only even-indexed coefficients. We re-write (3.3.1) using s':

$$\begin{aligned} h * h_{-} * s'_{-} &= h_{-} * h * s' \\ s'_{-} &= s' \end{aligned}$$

This means that s' has only even indexed coefficients, and thus we obtain $s = s' \downarrow 2$. □

Our goal is to construct wavelet functions that are close to a given target function f from $\mathbb{R} \rightarrow \mathbb{R}$. The first approach will be to fix the scaling function φ and to find a complementary wavelet function ψ as approximation for f. To this end we translate the above theorem to real functions. It means that all possible wavelet functions ψ' complementary to φ and only these are linear combinations of ψ and integral translations of φ.

3.3.2 Theorem.
Prerequisite. The function φ is refinable with respect to h, ψ and ψ' are wavelet functions with respect to g and g', respectively. The filter pair (h, g) is complementary.
Claim. ψ' is complementary to φ if and only if ψ' is a linear combination of ψ and translates of φ, i.e.

$$\psi' = \psi + s * \varphi \quad .$$

Proof. The functions ψ and ψ' are defined by (2.2.10)

$$\begin{aligned} \psi &= 2 \cdot (g * \varphi) \downarrow 2 \\ \psi' &= 2 \cdot (g' * \varphi) \downarrow 2 \end{aligned}$$

and thus

$$\psi' - \psi = 2 \cdot ((g' - g) * \varphi) \downarrow 2$$

due to Theorem 3.3.1 there is a filter s which satisfies

$$\begin{aligned}\psi' - \psi &= 2 \cdot ((s \uparrow 2) * h * \varphi) \downarrow 2 \\ &= s * 2 \cdot (h * \varphi) \downarrow 2 \\ &= s * \varphi \quad .\end{aligned}$$

The converse follows from Remark 3.2.4. □

Given a target function f we want to find an s such that the shape of $\psi + s * \varphi$ is similar to that of f. That is we want to solve the linear least squares problem:

$$\operatorname*{argmin}_s \|\psi + s * \varphi - f\|_2 \quad .$$

Since we have a linear problem the solution does not depend on the choice of ψ out of the affine space of functions complementary to φ.

But we do not know whether the amplitude of f is chosen properly. Thus we also allow a weighting of ψ, that is

$$\operatorname*{argmin}_{c,s} \|c \cdot \psi + s * \varphi - f\|_2$$

or more verbose

$$\operatorname*{argmin}_{c,s} \left\| c \cdot \psi + \sum_{k \in \mathbb{Z}} s_k \cdot (\varphi \to k) - f \right\|_2 \quad .$$

The solution is interpreted as: Using the lifting filter $\frac{s}{c}$ we obtain $\psi + \frac{s}{c} * \varphi$ which is close to $\frac{f}{c}$. The solution of the least squares problem is the projection of the pattern f into the vector space spanned by ψ and $\varphi \to k$. Examples for such basis are plotted in Figure 3.4. The matching process is demonstrated in Figure 3.5.

In a multi-resolution analysis the amplitude of the wavelet function ψ depends on the function φ. If the coefficient c becomes very small compared to the coefficients of s by the optimisation then the result has to be considered as bad approximation.

3.3.3 Solving the least squares problem

Now we want to dig into the details of the solution of the least squares problem. We want to discuss two essential approaches.

Standard linear least squares solvers

An easy way to solve the least squares problem is to fully discretise it and let it solve by some general least squares solver like LAPACK's GELS routine.

This preserves a good numerical behaviour but it is quite inefficient. Let f be the discretised pattern, n the refinement level, let H and G be the refined filter masks (Definition 2.2.1) and s the lifting filter

$$\begin{aligned}H &= \mathcal{R}_h^{n-1} h = h * h \uparrow 2 * \cdots * h \uparrow 2^{n-2} * h \uparrow 2^{n-1} \\ G &= \mathcal{R}_h^{n-1} g = h * h \uparrow 2 * \cdots * h \uparrow 2^{n-2} * g \uparrow 2^{n-1}\end{aligned}$$

denote their boundary indices with

$$\begin{aligned}k_0 &= \min\{i \ : \ H_i \neq 0\} & k_1 &= \max\{i \ : \ H_i \neq 0\} \\ l_0 &= \min\{i \ : \ G_i \neq 0\} & l_1 &= \max\{i \ : \ G_i \neq 0\} \\ j_0 &= \min\{i \ : \ s_i \neq 0\} & j_1 &= \max\{i \ : \ s_i \neq 0\}\end{aligned}$$

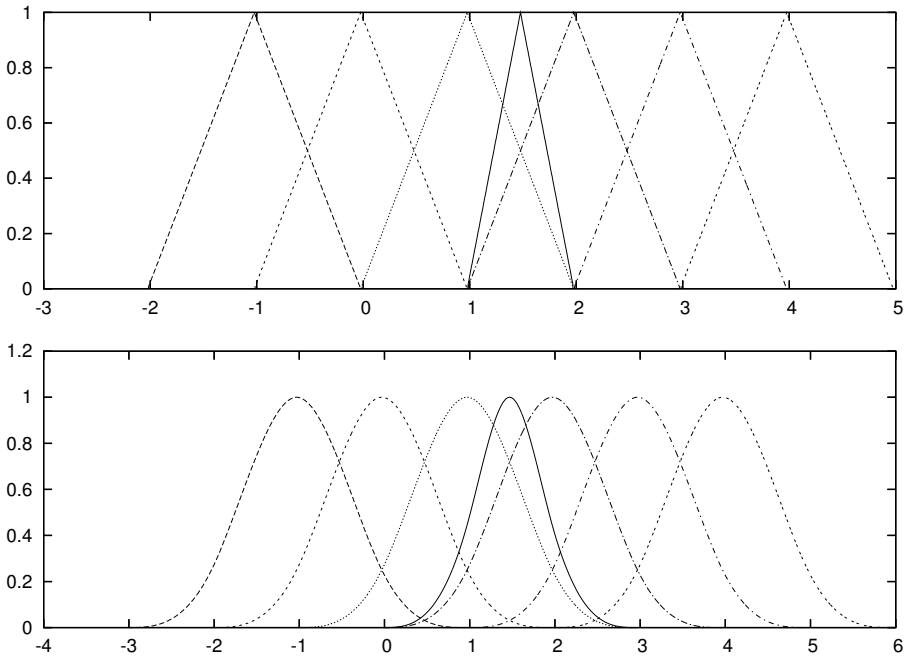


Figure 3.4: A basis for matching patterns: Matching with a wavelet via lifting means projection into a sub-space. Bases are shown for matching when the generator is a B-spline of second order (hat function) or of fourth order (cubic spline). Each basis consists of one generator (solid) line and several wavelets build from the CDF-2,0 and CDF-4,0 filter banks.

and declare the matrix A with

$$A = \begin{pmatrix}
H_{k_0} & 0 & \ddots & 0 & 0 & 0 \\
H_{k_0+1} & 0 & \ddots & 0 & 0 & 0 \\
H_{k_0+2} & 0 & \ddots & 0 & 0 & 0 \\
\vdots & \vdots & & \vdots & \vdots & \vdots \\
H_{k_0+2^n} & H_{k_0} & \ddots & 0 & 0 & 0 \\
H_{k_0+2^n+1} & H_{k_0+1} & \ddots & 0 & 0 & G_{l_0} \\
H_{k_0+2^n+2} & H_{k_0+2} & \ddots & 0 & 0 & G_{l_0+1} \\
\ddots & \ddots & & \ddots & \ddots & \vdots \\
\ddots & H_{2^{n-1}-1} & & H_{-2^{n-1}-1} & \ddots & G_{-1} \\
\ddots & H_{2^{n-1}} & & H_{-2^{n-1}} & \ddots & G_0 \\
\ddots & H_{2^{n-1}+1} & & H_{-2^{n-1}+1} & \ddots & G_1 \\
\ddots & \ddots & & \ddots & \ddots & \vdots \\
0 & 0 & \ddots & H_{k_1-2} & H_{k_1-2^n-2} & G_{l_1-1} \\
0 & 0 & \ddots & H_{k_1-1} & H_{k_1-2^n-1} & G_{l_1} \\
0 & 0 & \ddots & H_{k_1} & H_{k_1-2^n} & 0 \\
\vdots & \vdots & & \vdots & \vdots & \vdots \\
0 & 0 & \ddots & 0 & H_{k_1-2} & 0 \\
0 & 0 & \ddots & 0 & H_{k_1-1} & 0 \\
0 & 0 & \ddots & 0 & H_{k_1} & 0
\end{pmatrix} .$$

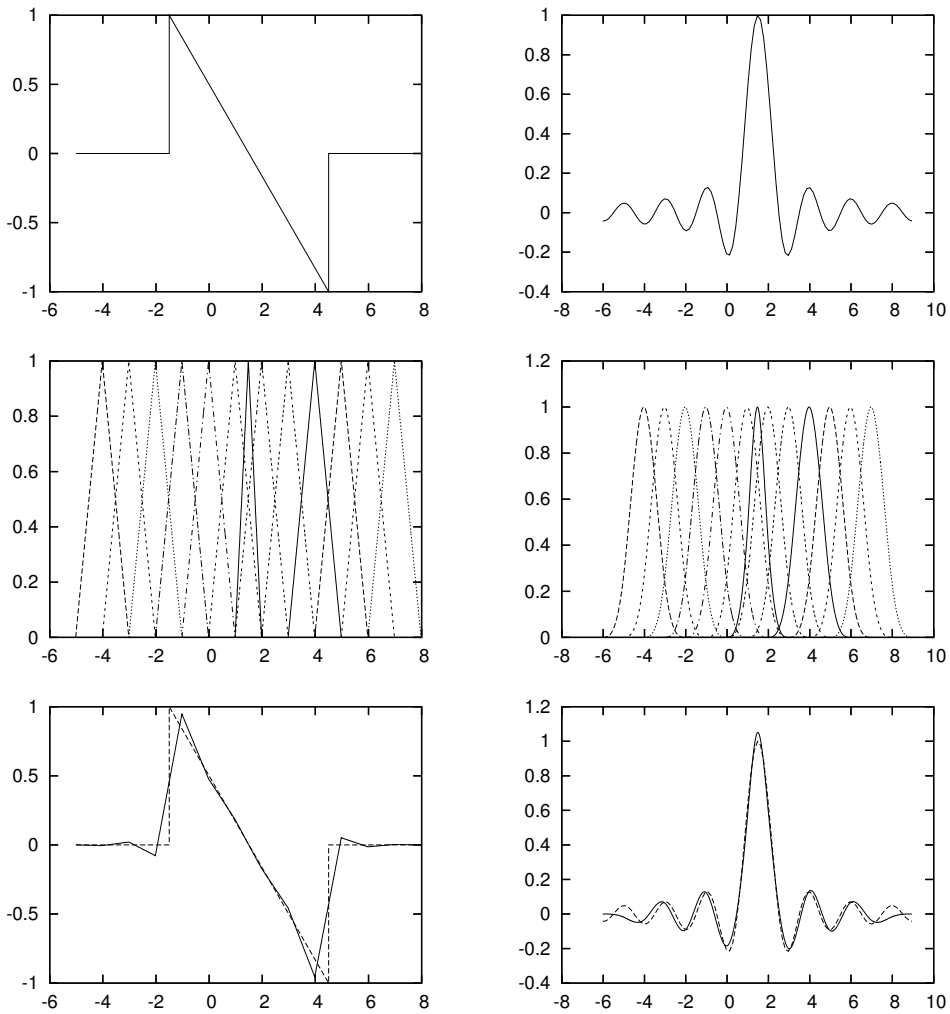


Figure 3.5: Match of wavelets with patterns: A clipped ramp function is matched with a wavelet associated with the hat generator (CDF-2) and an appropriately translated and dilated sinc function ($\mathrm{sinc}\, x = \begin{cases} 1 & : x = 0 \\ \frac{\sin x}{x} & : \text{else} \end{cases}$) is matched with a wavelet complementary to the CDF-4 generator. The lifting filter s is restricted to 12 coefficients thus the basis for the linear least squares problem (both figures in the vertical centre) contains 12 copies of the generator. The ramp is also an unlucky example where the coefficient c of ψ becomes zero thus perfect reconstruction is not possible. The reason is that the wavelet of the CDF-2,0 filter bank is an even function and the pattern is an odd function, and both are symmetric with respect to the same centre.

Then we solve the least squares problem $\operatorname{argmin}_x \|A \cdot x - f\|_2$ and obtain s and c in terms of a vector of the following form

$$\begin{pmatrix} s_{j_0} \\ s_{j_0+1} \\ \vdots \\ s_{j_1-1} \\ s_{j_1} \\ c \end{pmatrix} \quad .$$

If f is not completely sampled equidistantly but some values are missing then we can simply remove the corresponding rows from A and solve the remaining equation system.

For an irregularly sampled pattern we need to approximate the generator and the wavelet function at the same nodes like the pattern. To this end we can use the refinement method described in Section 3.1.2.

Custom solution with normal equations

Let f be a discrete or continuous pattern, that is for some index set I it is $f \in I \to \mathbb{R}$, $x \in \ell_0(\mathbb{Z})$ and A be a linear operator from $\ell_0(\mathbb{Z}) \to (I \to \mathbb{R})$. The vector x minimizes $\|Ax - f\|_2$, if and only if $A^*(Ax) = A^*f$. [Sto99, Section 4.8.1]

$$\forall y \ \|Ay - f\|_2 \geq \|Ax - f\|_2 \quad \Leftrightarrow \quad A^*(Ax) = A^*f$$

This optimization problem is *convex* and thus there is always a minimizing vector x. If $A^* \circ A$ is invertible, then the solution is unique and it holds

$$\operatorname*{argmin}_x \|Ax - f\|_2 = \left(A^* \circ A\right)^{-1} (A^*f) \quad .$$

The vector x is discrete, independently of I. Consequently $A^* \circ A$ can be represented by a matrix. Thus we can reduce the optimisation problem to the solution of a system of linear equations, called the *normal equations*. It allows for several optimisations but the condition of $A^* \circ A$ may be large, which means that the solution of the equations system raises numerical problems. A rule of thumb is: The larger the size of the filter h, the more the translates of h overlap, the worse is the condition number with respect to inversion.

Efficient solution of the normal equations

Let A be the linear operator from the previous section. The matrix representation of $A^* \circ A$ has a block structure. It can be represented by

$$A^* \circ A \sim \begin{pmatrix} \mathcal{H} & \mathcal{K} \\ \mathcal{K}^* & (\mathcal{G}) \end{pmatrix}$$

with

$$\begin{aligned} \mathcal{H} &\in \mathbb{R}^{\{j_0,\ldots,j_1\} \times \{j_0,\ldots,j_1\}} \\ \mathcal{K} &\in \mathbb{R}^{\{j_0,\ldots,j_1\} \times 1} \\ \mathcal{G} &\in \mathbb{R} \quad . \end{aligned}$$

According to [HJ85] the inverse is also of block form, more precisely

$$\begin{aligned} \begin{pmatrix} \mathcal{H} & \mathcal{K} \\ \mathcal{K}^* & (\mathcal{G}) \end{pmatrix}^{-1} &= \frac{1}{q} \cdot \begin{pmatrix} q \cdot \mathcal{H}^{-1} + \mathcal{J} \cdot \mathcal{J}^* & -\mathcal{J} \\ -\mathcal{J}^* & (1) \end{pmatrix} \\ \mathcal{J} &= \mathcal{H}^{-1} \cdot \mathcal{K} \\ q &= \mathcal{G} - \langle \mathcal{K}, \mathcal{J} \rangle \quad . \end{aligned}$$

Because of the symmetric TOEPLITZ structure of $\mathcal{H}$, every matrix multiplication involving $\mathcal{H}^{-1}$ can be computed efficiently [Amm96].

3.3.4 Efficient computation of normal equations

Continuous pattern

If the pattern f is given as continuous function and the scalar products $\langle \varphi \to k, f \rangle$ and $\langle \psi, f \rangle$ can be computed, then f can be matched immediately with φ and ψ.

We have

$$A(s,c) = s * \varphi + c \cdot \psi$$

and consequently

$$\begin{aligned} A^* f &= (Q(f * \varphi^*), \langle f, \psi \rangle) \\ A^*(A(s,c)) &= \big(Q((s * \varphi + c \cdot \psi) * \varphi^*), \langle s * \varphi + c \cdot \psi, \psi \rangle\big) \\ &= \Big(s * Q(\varphi * \varphi^*) + c \cdot Q(\psi * \varphi^*), \big\langle s, Q(\varphi^* * \psi) \big\rangle + c \cdot \|\psi\|_2^2\Big) \quad . \end{aligned}$$

The matrix for $A^* \circ A$ consists merely of discretised convolutions of φ and ψ with φ^* and ψ^*. They can be computed efficiently by the algorithm in Section 3.1.2.

Discretised pattern

If the pattern is given in discretised form $f \in \ell_0(\mathbb{Z})$ we should match it with the refined filters since they are implicitly applied by the wavelet transform. This means we do not interpret the pattern as a linear combination $f * \varphi$ of small generators, but we interpret it just as the plain sequence f.

We obtain

$$A(s,c) = s \uparrow 2^n * H + c \cdot G$$

and with Lemma 1.2.13

$$\begin{aligned} A^* f &= ((f * H^*) \downarrow 2^n, \langle f, G \rangle) \\ A^*(A(s,c)) &= \big(((s \uparrow 2^n * H + c \cdot G) * H^*) \downarrow 2^n, \langle s \uparrow 2^n * H + c \cdot G, G \rangle\big) \\ &= \Big(s * (H * H^*) \downarrow 2^n + c \cdot (G * H^*) \downarrow 2^n, \big\langle s, (H^* * G) \downarrow 2^n \big\rangle + c \cdot \|G\|_2^2\Big) \quad . \end{aligned}$$

It holds $\|G\|_2^2 = (G * G^*)_0$ and of course $\|G\|_2^2 = \big((G * G^*) \downarrow 2^n\big)_0$. Analogously it holds $\langle f, G \rangle = \big((f * G^*) \downarrow 2^n\big)_0$. Further we can compute $(H * H^*) \downarrow 2^n$, $(G * H^*) \downarrow 2^n$, and $(G * G^*) \downarrow 2^n$ rather efficiently by a repeated convolution with immediate down-sampling.

One thing we need is the connection

$$\mathcal{R}_h^n x * \mathcal{R}_g^n y = \mathcal{R}_{h*g}^n (x * y) \quad .$$

It remains to find an efficient computation which benefits from the down-sampling. The following lemma is essentially the consideration for the cascade algorithm in Section 3.1.1 ported to discrete signals.

3.3.3 Lemma.
Claim.

$$\mathcal{T}_h^n x = \Big(h \uparrow 2^{n-1} * \ldots * h \uparrow 2 * h * x\Big) \downarrow 2^n$$

Proof. The claim is equivalent to

$$\mathcal{T}_h^n x = \big(\mathcal{R}_h^n \delta * x\big) \downarrow 2^n \quad .$$

We use induction over n. First we verify that

$$\mathcal{T}_h^0 x = x = \mathcal{R}_h^0 \delta * x$$

For the induction step we need (1.2.2) of Lemma 1.2.2:

$$\begin{aligned}
\mathcal{T}_h^n x &= \left(\mathcal{R}_h^n \delta * x\right) \downarrow 2^n \\
\mathcal{T}_h^{n+1} x &= \mathcal{T}_h^n \left(\mathcal{T}_h x\right) \\
&= \left(\mathcal{R}_h^n \delta * \mathcal{T}_h x\right) \downarrow 2^n \\
&= \left(\mathcal{R}_h^n \delta * (h * x) \downarrow 2\right) \downarrow 2^n \\
&\overset{(1.2.2)}{=} \left(\mathcal{R}_h^n \delta \uparrow 2 * h * x\right) \downarrow 2^{n+1} \\
&\overset{(2.2.4)}{=} \left(\mathcal{R}_h \left(\mathcal{R}_h^n \delta\right) * x\right) \downarrow 2^{n+1} \\
&= \left(\mathcal{R}_h^{n+1} \delta * x\right) \downarrow 2^{n+1} \qquad .
\end{aligned}$$

□

The iterated application of $\mathcal{T}_h$ is more efficient than computing $\mathcal{R}_h^n \delta$, because the first one has shorter filters as intermediate results. The computations are summarised

$$\begin{aligned}
(f * H^*) \downarrow 2^n &= \mathcal{T}_{h^*}^n f \\
(f * G^*) \downarrow 2^n &= \mathcal{T}_{g^*}^n f \\
(H * H^*) \downarrow 2^n &= \mathcal{T}_{h*h^*}^n \delta \\
(G * H^*) \downarrow 2^n &= \mathcal{T}_{g*h^*} \left(\mathcal{T}_{h*h^*}^{n-1} \delta\right) \\
(G * G^*) \downarrow 2^n &= \mathcal{T}_{g*g^*} \left(\mathcal{T}_{h*h^*}^{n-1} \delta\right) \qquad .
\end{aligned}$$

3.3.5 Conclusion

Now we want to summarise the advantages and disadvantages of the presented method.

Advantages

- The smoothness of the matched wavelet can be chosen arbitrarily.
- The found filter pairs are automatically complementary. If this is not necessary there are certainly much simpler methods which perform fast pattern detection.
- The resulting least squares problem is easy to understand. The user can intuitively decide which kinds of patterns can be matched well and which cannot. He can intuitively decide at which scale the pattern should be matched and how long the filters should be.
- The optimisation is fast.

Disadvantages

- The found filter pair is not orthogonal i.e. analysis and synthesis wavelet differ.
- Smoothness of the dual basis functions is hard to achieve. Figure 3.6 shows that for our previous examples. Refer to Section 4.3 for strategies to fix this problem.
- Patterns in the signal can only be retrieved if they are at the scales and positions of the dyadic grid.

The last point can be weakened in the following way: Several wavelets can be matched to shifted and rescaled variants of the pattern. Then one wavelet transform per pattern variant can be applied. The result is redundant, of course. It can also be unsatisfying since the matching quality will not be homogeneous over all shifts and scales of the pattern.

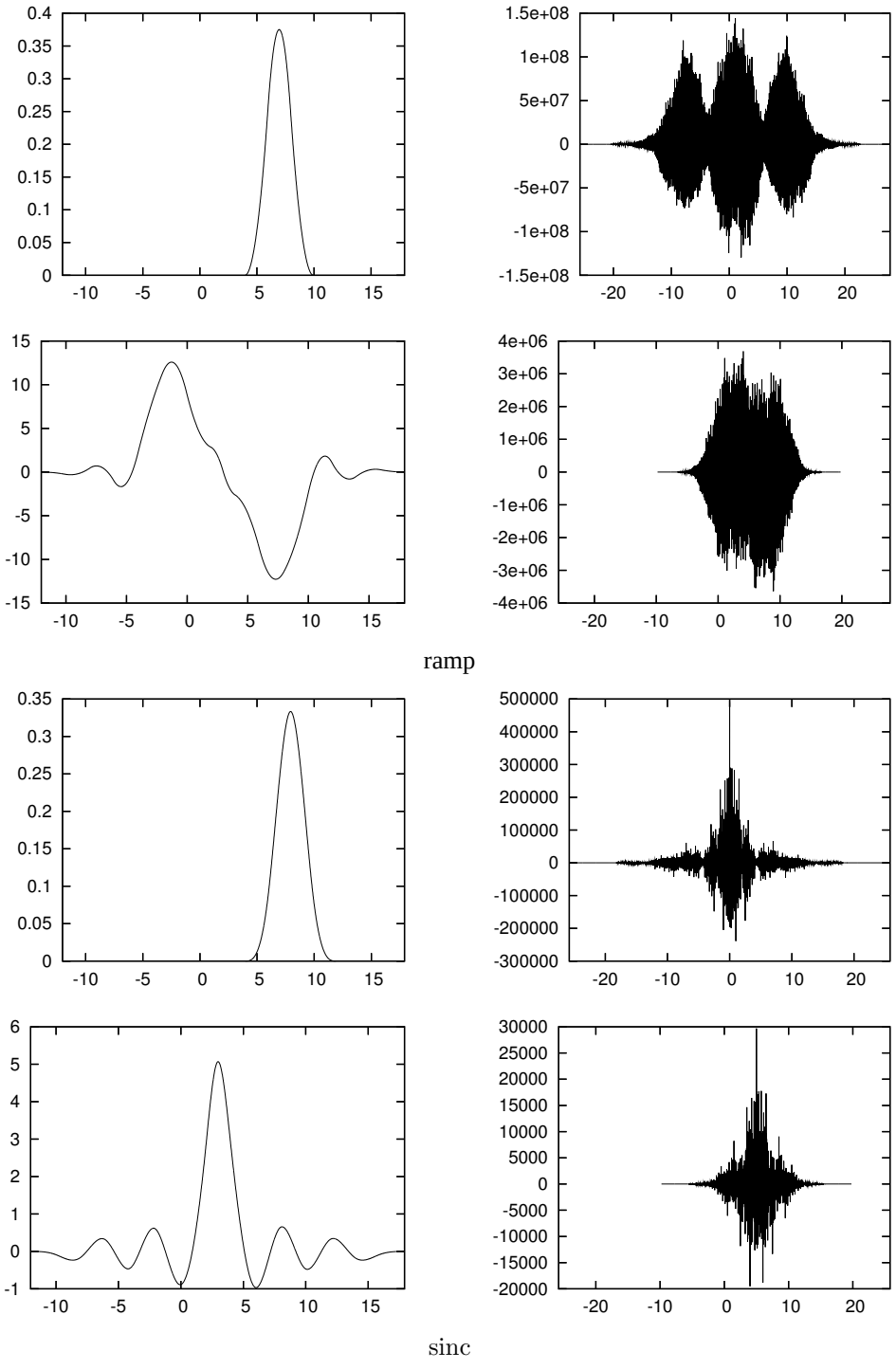


Figure 3.6: Primal and dual generators and wavelets for matched wavelets: We show primal and dual functions of the match examples from Figure 3.5. We match the ramp with respect to CDF-3,1 (that is the matched wavelet is forced to have one vanishing moment, see Section 4.3.2 for details) because the match with CDF-2,0 does not allow for perfect reconstruction. The sinc function is matched to CDF-4,0. The fractal shape of the dual functions implies bad numerical properties of the wavelet transform with respect to these wavelets.

Chapter 4

Smoothness of matched wavelets

The approximation of a given function f by a discrete wavelet function ψ is one part of our matching problem. The second part is that the wavelet transformation with respect to the filter masks h and g and its inverse transformation should have good numerical properties. An important criterion for a numerical stable wavelet transformation is the smoothness of the dual functions $\widetilde{\psi}$ and $\widetilde{\varphi}$. The wavelet transform consists of correlating the dual functions $\widetilde{\psi}$, $\widetilde{\varphi}$ and their dilates and translates with the signal x, that is $\langle \widetilde{\varphi} \to k, x \rangle$ and $\left\langle \left(\widetilde{\psi} \to k \right) \downarrow 2^j, x \right\rangle$. You can imagine that correlating a signal with non-smooth functions causes strong dependencies from the shift of the signal x. Using smooth dual functions reduces the sensitivity against shift of the signal, but due to the recursive structure of $\widetilde{\varphi}$ creating smooth dual generator functions is more difficult as one may think at first.

4.1 How to measure smoothness

Smoothness of functions is usually described by function spaces. This section gives a short introduction to function spaces as far as necessary for our consideration of discrete wavelets. For a more detailed introduction refer to e.g. [Tri92].

4.1.1 Smoothness and differentiability

What is the mathematical meaning of the smoothness of a function? We have the notion of *continuity* which means, intuitively spoken, that a function has no bumps. But it may have sharp bends. A continuously differentiable function cannot have sudden bends thus "smooth" is often used as synonym for *continuously differentiable*. We can state that differentiating makes a function less smooth and that a function which is differentiable of high order is quite smooth.

However it must be noted that this kind of smoothness includes arbitrary sharp bends or even almost bumps. The problem is certainly that our intuition of smoothness depends on the scale. Something that is smooth on a small scale may no longer be considered smooth at a large scale. Because differentiability does not depend on scales it cannot fully reflect an intuitive notion of smoothness.

4.1.1 Definition (Spaces of differentiable functions). For $k \in \mathbb{N}_0$ the space denoted by $C^k(\mathbb{R})$ is the space of k times continuously differentiable bounded functions. That is $C^0(\mathbb{R})$ contains all bounded and uniformly continuous functions and if f' is bounded and $f' \in C^k(\mathbb{R})$ then $f \in C^{k+1}(\mathbb{R})$.

The $C^k(\mathbb{R})$ spaces are related to the supremum norm, they contain only bounded functions. Given a function f from $C^k(\mathbb{R})$ you can measure the weight and the roughness of the function by a term like $\|f\|_\infty + \left\|f'\right\|_\infty + \cdots + \left\|f^{(k)}\right\|_\infty$.

An alternative space of functions of increasing smoothness is the SOBOLEV space where the $\mathcal{L}_p(\mathbb{R})$ norm is used for measurement of the weight and the roughness of a function, like in $\|f\|_p + \left\|f'\right\|_p + \cdots +$

$\left\| f^{(k)} \right\|_p$.

4.1.2 Definition (SOBOLEV spaces). For $k \in \mathbb{N}_0$ the SOBOLEV *space* denoted by $W_p^k(\mathbb{R})$ is the space of functions f for which each of the first k weak derivatives is in $\mathcal{L}_p(\mathbb{R})$. This is equivalent to the criterion whether $\|f\|_p + \|f'\|_p + \cdots + \left\| f^{(k)} \right\|_p$ exists.

4.1.2 Smoothness and the frequency domain

Our considerations of smoothness shall lead to an algorithm which makes a wavelet smoother which results from the pattern match algorithm. Using the differentiation order as measure of smoothness would lead to a discrete optimisation problem. This seems to be more complicated than generalising the smoothness measurement to finer (i.e. fractional) grades of smoothness. We just need a definition of fractional derivatives.

Here the frequency spectrum (that is the FOURIER transform) of a function comes to our help. If we have decomposed a function into complex exponential functions the differentiation is easy since the derivative of $x \mapsto e^{\mathrm{i}\cdot\xi\cdot x}$ is $x \mapsto \mathrm{i}\cdot\xi\cdot e^{\mathrm{i}\cdot\xi\cdot x}$. That is the differentiation only changes the weighting of the function. This leads to the statement

$$\begin{aligned} \widehat{f'}(\xi) &= \mathrm{i}\cdot\xi\cdot\widehat{f}(\xi) \\ \widehat{f^{(k)}}(\xi) &= (\mathrm{i}\cdot\xi)^k\cdot\widehat{f}(\xi) \end{aligned}$$

which can be easily generalised to any real s

$$\widehat{f^s}(\xi) = e^{\mathrm{i}\cdot s\cdot\pi/2}\cdot\xi^s\cdot\widehat{f}(\xi) \qquad .$$

We see that differentiation amplifies high frequencies, thus differentiation is a high-pass filter. If a norm of a FOURIER transformed function remains finite after some fractional differentiation this indicates a smooth function. If the FOURIER transform of a function decays fast enough for high frequencies it fulfils this condition.

We are now going to define generalisations of the spaces of differentiable functions and the SOBOLEV spaces to fractional differentiations. They are also more general in the sense that they do not only contain functions but also distributions, namely objects like the DIRAC impulse. For this purpose we need the SCHWARZ space $S(\mathbb{R})$ consisting of fast decaying arbitrarily often differentiable functions and its dual space $S'(\mathbb{R})$ which is the set of all complex-valued tempered distributions on $\mathbb{R}$.

HOELDER **continuity**

The HOELDER-ZYGMUND function space $\mathcal{C}^s(\mathbb{R})$ with $s \in \mathbb{R}$ and $s \geq 0$ contains all functions that are up to $\lceil s \rceil$ times differentiable and some more functions. It is a generalisation of the spaces of differentiable functions. A characterisation can be given using a smooth dyadic resolution of the unity of the FOURIER domain and according band pass filtering [Tri92, pages 14-17].

4.1.3 Definition (Dyadic resolution of the unity). The sequence ψ of functions from $\mathbb{N}_0 \to \mathbb{R} \to \mathbb{R}$ is called *dyadic resolution of the unity* if

$$\begin{aligned} & & \operatorname{supp}\psi_0 &\subseteq [-2,2] \\ \forall j \in \mathbb{N} & & \operatorname{supp}\psi_j &\subseteq \left\{ \xi \,:\, |\xi| \in [2^{j-1}, 2^{j+1}] \right\} \\ \exists C > 0 \quad \forall j \in \mathbb{N}_0 & & \left\|\psi_j\right\|_\infty &< C \\ \forall \xi \in \mathbb{R} & & \sum_{j=0}^{\infty} \psi_j(\xi) &= 1 \quad . \end{aligned}$$

A dyadic resolution ψ is called smooth if the functions of the sequence ψ are arbitrarily often differentiable.

4.1.4 Definition (HOELDER-ZYGMUND space). For a smooth dyadic resolution of the unity ψ the HOELDER-ZYGMUND *function space* $\mathcal{C}^s(\mathbb{R})$ is defined as

$$\mathcal{C}^s(\mathbb{R}) = \left\{ f \;:\; f \in S'(\mathbb{R}) \wedge \exists C \quad \forall j \in \mathbb{N}_0 \quad 2^{j\cdot s} \cdot \left\| \widehat{\psi_j} * f \right\|_\infty < C \right\} \quad .$$

The so defined space is equal for all smooth dyadic resolutions of the unity.

SOBOLEV **smoothness**

The fractional SOBOLEV space $H_p^s(\mathbb{R})$ is the generalisation of the SOBOLEV space $W_p^s(\mathbb{R})$. We restrict ourselves to $H_2^s(\mathbb{R})$ which allows for a characterisation that was used by VILLEMOES to explore the smoothness of refinable functions.

4.1.5 Definition. The SOBOLEV function space $H_2^s(\mathbb{R})$ is defined as

$$H_2^s(\mathbb{R}) = \left\{ f \;:\; f \in S'(\mathbb{R}) \wedge \left(\xi \mapsto \left(1 + |\xi|^2\right)^s \cdot \widehat{f}(\xi)^2 \right) \in \mathcal{L}_1(\mathbb{R}) \right\} \quad .$$

4.2 Computing the smoothness of refinable functions

4.2.1 Convolutional decomposition

From the differentiation property of the FOURIER transform we can conclude that a function is smooth if its FOURIER transform decays fast. However it should be noted that this is not necessary. A function with a FOURIER transform consisting of peaks that become narrow fast enough at high frequencies, can also be smooth.

If we have two functions f and g whose FOURIER transforms are majorised by $\xi \mapsto |\xi|^s$ and $\xi \mapsto |\xi|^t$, respectively, then the FOURIER transform of their convolution $\widehat{f * g}$ is bounded by $\xi \mapsto \sqrt{2\cdot\pi} \cdot |\xi|^{s+t}$. This means if $\widehat{f}$ really decays ($s < 0$) then $f * g$ will be smoother than g by the decay (this means: the smoothness order) of f.

Theorem 2.2.30 allows us to consider a refinable function as convolution of other refinable functions. If we factorise a refinement mask then we obtain associated refinable functions which are convolutional factors of the original refinable function. There are factors that are of a special interest because they help smoothing a refinable function.

The *Box-Spline* (also *B-Spline*) of jth degree B_j is a function which consists piecewise of polynomial functions of degree j. It is a convolutional power of the characteristic function $\chi_{[0,1)}$, the function which is one at the interval $[0,1)$ and zero elsewhere.

$$B_j = \chi_{[0,1)}^{*(j+1)}$$

Further on the characteristic function $\chi_{[0,1)}$ is refinable with respect to the mask $\frac{1}{2}\cdot(\mathbf{1},1)$. Theorem 2.2.30 implies that B_j is refinable with respect to $\left(\frac{1}{2}\cdot(\mathbf{1},1)\right)^{*(j+1)}$.

Every real polynomial can be decomposed into polynomials of degree 2. [Str95, Theorem 25.21, Lemma 25.24] Among the possible linear factors (polynomials of degree 1) the factors related to B-Splines ($\frac{1}{2}\cdot(\mathbf{1},1)$) are exceptionally smooth. The function $\chi_{[0,1)}$ which is associated with the mask $\frac{1}{2}\cdot(\mathbf{1},1)$ belongs to all $H_2^s(\mathbb{R})$ for $s < \frac{1}{2}$. Whereas a refinable function of a mask $\left(\frac{\mathbf{1}}{2} + \varepsilon, \frac{1}{2} - \varepsilon\right)$ with $\varepsilon \neq 0$ does belong at most to $H_2^s(\mathbb{R})$ with $s < 0$.

The next two theorems quantify the smoothness advance that is achieved by convolving with the characteristic function.

4.2.1 Theorem.
Prerequisite. Let $\varphi \in C^\alpha(\mathbb{R})$.
Claim. Then $\varphi * \chi_{[0,1)} \in C^{\alpha+1}(\mathbb{R})$.

Proof. We will show that $\varphi * \chi_{[0,1)}$ is as smooth as the integral of φ.

The following identity holds:

$$\begin{aligned}(\varphi * \chi_{[0,1)})(x) &= \int_{\mathbb{R}} \Big(t \mapsto \varphi(t) \cdot \chi_{[0,1)}(x-t)\Big) \\ &= \int_{\mathbb{R}} \Big(t \mapsto \varphi(t) \cdot \chi_{(-1,0]}(t-x)\Big) \\ &= \int_{\mathbb{R}} \Big(t \mapsto \varphi(t) \cdot \chi_{(x-1,x]}(t)\Big) \\ &= \int_{(x-1,x)} \varphi \\ &= \int_{(0,x)} \varphi - \int_{(0,x-1)} \varphi\end{aligned}$$

given that the integral $\int_{(0,x)} \varphi$ exists.

Since $\varphi \in C^{\alpha}(\mathbb{R})$ it is $\left(x \mapsto \int_{(0,x)} \varphi\right) \in C^{\alpha+1}(\mathbb{R})$ and thus the difference of such functions is in $C^{\alpha+1}(\mathbb{R})$, too. □

Also for the fractional SOBOLEV smoothness we obtain one degree of smoothness more for every convolution with $\chi_{[0,1)}$.

4.2.2 Theorem.
Prerequisite. $\varphi \in H_2^s(\mathbb{R})$
Claim. $\varphi * \chi_{[0,1)} \in H_2^{s+1}(\mathbb{R})$

Proof.

$$\begin{aligned}\widehat{\chi_{[-1,1)}}(\xi) &= \frac{2}{\sqrt{2\cdot\pi}} \cdot \frac{\sin\xi}{\xi} \\ \chi_{[0,1)} &= (\chi_{[-1,1)} \to 1) \downarrow 2 \\ \widehat{\chi_{[0,1)}}(\xi) &= \frac{2}{\sqrt{2\cdot\pi}} \cdot \frac{\sin\frac{\xi}{2}}{\xi} \cdot e^{-\mathrm{i}\cdot\xi/2}\end{aligned}$$

$$\begin{aligned}\forall\,|\xi| \le \frac{\pi}{2} \qquad & 1+|\xi|^2 \le 1+\frac{\pi^2}{4} \quad \wedge \quad \left|\frac{\sin\frac{\xi}{2}}{\xi/2}\right|^2 \le 1 \\ \forall\,|\xi| > \frac{\pi}{2} \qquad & \frac{1+|\xi|^2}{|\xi|^2} \le \frac{4}{\pi^2}+1 \quad \wedge \quad \left|2\cdot\sin\frac{\xi}{2}\right|^2 \le 4\end{aligned}$$

$$\forall\xi \qquad \left(1+|\xi|^2\right)\cdot\left|2\cdot\frac{\sin\frac{\xi}{2}}{\xi}\right|^2 \le 6$$

$$\begin{aligned}\left\|\xi \mapsto \left(1+|\xi|^2\right)^{s+1} \cdot \varphi * \widehat{\chi_{[0,1)}}(\xi)^2\right\|_1 &= \left\|\xi \mapsto \left(1+|\xi|^2\right)^{s+1} \cdot 2\cdot\pi\cdot\widehat{\varphi}(\xi)^2 \cdot \widehat{\chi_{[0,1)}}(\xi)^2\right\|_1 \\ &\le 6 \cdot \left\|\xi \mapsto \left(1+|\xi|^2\right)^{s} \cdot \widehat{\varphi}(\xi)^2\right\|_1\end{aligned}$$

and consequently if $\varphi \in H_2^s(\mathbb{R})$ then $\varphi * \chi_{[0,1)} \in H_2^{s+1}(\mathbb{R})$. □

What we got is that if h is a mask for a refinable function then $h * \frac{1}{2}\cdot(\mathbf{1},1)$ is the mask for a smoother function with usually similar shape. We will thus call $\frac{1}{2}\cdot(\mathbf{1},1)$ the *smoothness factor*.

Thus it is worth separating B-spline factors from a refinable function before considering its smoothness. In this sense each refinable function can be considered as a convolution of a B-spline and a very fractal

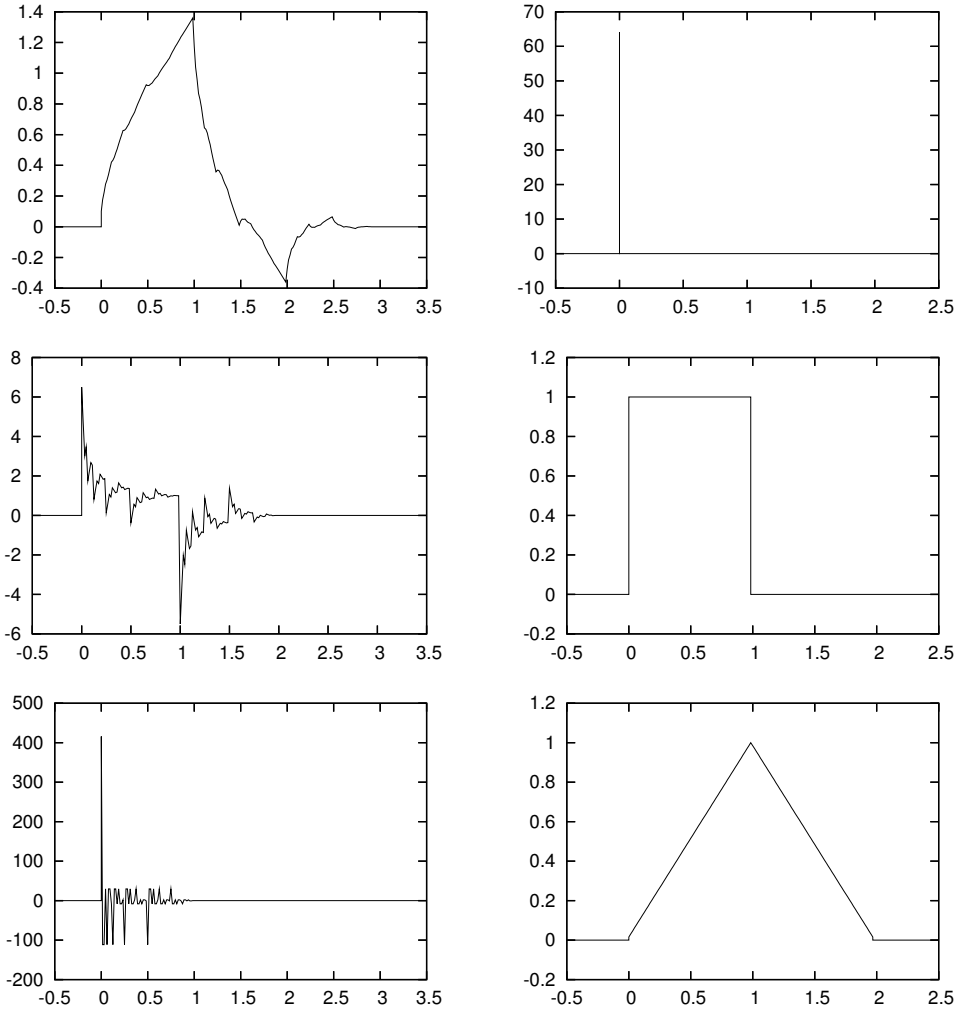


Figure 4.1: The generator of the orthogonal DAUBECHIES-2 wavelet basis decomposed into a B-spline and a fractal part. The convolution of both functions in each row results in the DAUBECHIES-2 generator. The function graphs are generated with the cascade algorithm (Section 3.1.1) over 6 levels of refinement. There is certainly no function which is refinable with respect to the DAUBECHIES-2 generator mask without any B-Spline factor, as well as there is no function (but the DIRAC impulse distribution) which is refinable with respect to δ. Thus their function graphs (especially the heights) are arbitrary.

object, that is a non-regular distribution (Figure 4.1). Roughly spoken the B-spline is responsible for the smoothness and the fractal portion defines the shape.

The factor $\frac{1}{2} \cdot (\mathbf{1}, 1)$ in a mask can also be characterised by a zero of the FOURIER symbol. Since $w = \frac{1}{2} \cdot (\mathbf{1}, 1)$ is associated with $\widehat{w}(\xi) = \frac{1}{2} \cdot \left(1 + e^{\mathrm{i} \cdot \xi}\right)$ it is $\widehat{w}(\pi) = 0$, thus a mask h contains the factor w exactly n times if and only if $\widehat{h}$ has a n times zero at π.

The consideration of smoothness of refinable functions leads us to the eigenvalue spectrum of the refinement operator. There is a tight connection between the smoothness factors and some special eigenvalues of the refinement operator. [Str96]

4.2.3 Theorem.
Prerequisite. Let $h \in \ell_0(\mathbb{Z})$ with $\sum h = 1$ and φ be refinable with respect to $\frac{1}{2} \cdot (\mathbf{1}, 1) * h$. The operator $\mathcal{T}_h$ is a convolution with subsequent down-sampling ($\mathcal{T}_h x = (h * x) \downarrow 2$) as in Definition 2.2.32.
Claim.

- $2 \cdot \mathcal{T}_{\frac{1}{2} \cdot (\mathbf{1}, 1) * h}$ has all eigenvalues of $2 \cdot \mathcal{T}_h$ multiplied with $\frac{1}{2}$ (that is the eigenvalues of $\mathcal{T}_h$) and additionally the eigenvalue 1.
- If x is a right eigenvector of $2 \cdot \mathcal{T}_h$ then $(\mathbf{1}, -1) * x$ is a right eigenvector of $2 \cdot \mathcal{T}_{\frac{1}{2} \cdot (\mathbf{1}, 1) * h}$. That is the neighbouring elements of x are subtracted. The extra eigenvalue of $2 \cdot \mathcal{T}_{\frac{1}{2} \cdot (\mathbf{1}, 1) * h}$ has the discretised refinable function $Q\varphi$ as right eigenvector.
- If y is a left eigenvector of $2 \cdot \mathcal{T}_h$ then $y \not* (-1, \mathbf{1})$ is a left eigenvector of $2 \cdot \mathcal{T}_{\frac{1}{2} \cdot (\mathbf{1}, 1) * h}$. The eigenvector y will almost always have infinite support thus there is no satisfying notion of divisibility. The division is not unique. One example is the discrete integration (cumulative sum) of the sequence y. The extra eigenvalue of $2 \cdot \mathcal{T}_{\frac{1}{2} \cdot (\mathbf{1}, 1) * h}$ has the left eigenvector $(\ldots, 1, \mathbf{1}, 1, \ldots)$.

Proof. First we consider how eigenvalues and eigenvectors evolve from $\mathcal{T}_h$ to $\mathcal{T}_{\frac{1}{2} \cdot (\mathbf{1}, 1) * h}$. For the right eigenvectors we find

$$\begin{aligned}
\lambda \cdot x &= \mathcal{T}_h x \\
&= (h * x) \downarrow 2 \\
\lambda \cdot (\mathbf{1}, -1) * x &= (\mathbf{1}, -1) * (h * x) \downarrow 2 \\
&= (h * (\mathbf{1}, 0, -1) * x) \downarrow 2 \\
\frac{\lambda}{2} \cdot (\mathbf{1}, -1) * x &= \left(\frac{1}{2} \cdot (\mathbf{1}, 1) * h * (\mathbf{1}, -1) * x\right) \downarrow 2 \\
&= \mathcal{T}_{\frac{1}{2} \cdot (\mathbf{1}, 1) * h}((\mathbf{1}, -1) * x) \quad .
\end{aligned}$$

Left eigenvectors of $\mathcal{T}_h$ are right eigenvectors of ${\mathcal{T}_h}^*$.

$$\begin{aligned}
\mathcal{T}_h &= (\downarrow 2) \circ (h*) \\
{\mathcal{T}_h}^* &= (h*)^* \circ (\downarrow 2)^* \qquad | \qquad \text{Lemma 1.2.13} \\
&= (h^**) \circ (\uparrow 2)
\end{aligned}$$

$$\begin{aligned}
{\mathcal{T}_h}^* y &= h^* * (y \uparrow 2) \\
\overline{\lambda} \cdot y &= h^* * (y \uparrow 2) \\
\overline{\lambda} \cdot y \not* (-1, \mathbf{1}) &= h^* * (y \uparrow 2) \not* (-1, \mathbf{1}) \\
&= (1, \mathbf{1}) * h^* * (y \uparrow 2) \not* (-1, 0, \mathbf{1}) \\
&= (1, \mathbf{1}) * h^* * ((y \not* (-1, \mathbf{1})) \uparrow 2) \\
\frac{\overline{\lambda}}{2} \cdot y \not* (-1, \mathbf{1}) &= \left(\frac{1}{2} \cdot (\mathbf{1}, 1) * h\right)^* * ((y \not* (-1, \mathbf{1})) \uparrow 2) \\
&= {\mathcal{T}_{\frac{1}{2} \cdot (\mathbf{1}, 1) * h}}^*((y \not* (-1, \mathbf{1})) \uparrow 2)
\end{aligned}$$

What about the eigenvectors of the new eigenvalue 1? The statement for the right eigenvector was already confirmed in Section 3.1.2. For the left eigenvector we obtain

$$\begin{aligned} 2 \cdot \mathcal{T}_{\frac{1}{2}\cdot(\mathbf{1},1)*h}{}^{*}\,(\dots,1,\mathbf{1},1,\dots) &= (1,\mathbf{1}) * h^{*} * (\dots,1,\mathbf{1},1,\dots) \uparrow 2 \\ &= h^{*} * (\dots,1,\mathbf{1},1,\dots) \\ &= \overline{\sum h} \cdot (\dots,1,\mathbf{1},1,\dots) \\ &= (\dots,1,\mathbf{1},1,\dots) \quad . \end{aligned}$$

□

4.2.4 Remark. In Remark 2.2.34 it was shown that the sets of finitely supported right eigenvectors of T_h and of $\mathcal{T}_h$ are equal. In contrast to that for each finite left eigenvector x we derive

$$\begin{aligned} \deg x &= \deg(h^{*} * (x \uparrow 2)) \\ \deg x &= \deg h + 2 \cdot \deg x \\ 0 &= \deg h + \deg x \quad . \end{aligned}$$

This can be fulfilled only for zero degree h and x. This implies that for $\deg h > 0$ the left eigenvectors of $\mathcal{T}_h$ have infinite support.

The above theorem shows that a power $\left(\frac{1}{2}\cdot(\mathbf{1},1)\right)^{*K}$ of smoothness factors in a mask h corresponds to the eigenvalues $\frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \dots, 2^{-K}$ of T_h. The converse is not true: The mask $\left(\frac{1}{2}, \mathbf{0}, \frac{1}{2}\right)$ induces the eigenvalues $\frac{1}{2}, \frac{1}{2}, 0$ but contains no smoothness factor. Every convolution with a smoothness factor halves all eigenvalues. The absolute value of the largest eigenvalue which is not associated to smoothness factors determines the smoothness of the refinable function. The smaller this eigenvalue the smoother the function. For some smoothness measurements we will have to consider $T_{h*h^{*}}$ instead of T_h.

4.2.2 VILLEMOES machine

In Section 4.1.2 we got to know how smoothness of functions can be expressed by their memberships in spaces of functions of certain degrees of smoothness. The family of SOBOLEV spaces and the family of HOELDER spaces are certainly the most popular ones. In the previous section we have seen how the factor $\frac{1}{2}\cdot(\mathbf{1},1)$ in a refinement mask contributes to the smoothness of the refinable function. We will now complement that with smoothness estimates of the remaining mask.

The key component of all smoothness measures for refinable functions is the following operation defined for a mask m:

1. Extract the convolutional factor $\frac{1}{4}\cdot(1,\mathbf{2},1)$ as often as possible from m. That is choose K such that

$$m = \left(\frac{1}{4}\cdot(1,\mathbf{2},1)\right)^{*K} * h$$

   where h contains no further smoothness factor $\frac{1}{4}\cdot(1,\mathbf{2},1)$. (Equivalent: $\widehat{h}$ has no double zero at π, i.e. $\widehat{h}(\pi) \neq 0$ or $\widehat{h}'(\pi) \neq 0$, or alternatively: Eh has no double zero at -1). Note that the mask $\frac{1}{4}\cdot(1,\mathbf{2},1)$ corresponds to the symbol $\xi \mapsto \left(\cos\frac{\xi}{2}\right)^2$.
2. Set up the matrix T_h and compute the absolute value of its largest eigenvalue. This is denoted by the *spectral radius* $\varrho(T_h)$.
3. The result of the operation is

$$\begin{aligned} M_m &= 2 \cdot K - \log_2 \varrho(2 \cdot T_h) \\ &= 2 \cdot K - 1 - \log_2 \varrho(T_h) \quad . \end{aligned}$$

HOELDER **continuity**

The definition Definition 4.1.4 of HOELDER-ZYGMUND spaces is not suitable for application on refinable functions. We use the following embedding instead.

4.2.5 Lemma.

$$\left\{ f \;:\; f \in S'(\mathbb{R}) \wedge \left(\xi \mapsto \left(1+|\xi|\right)^s \cdot \widehat{f}(\xi) \right) \in \mathcal{L}_1(\mathbb{R}) \right\} \subset \mathcal{C}^s(\mathbb{R})$$

Proof. Consider a more special resolution of the unity where ψ_0 has values between 0 and 1 everywhere and which is 1 in the interval $[-1,1]$. All other functions are defined by $\psi_j = \psi_0 \uparrow 2^j - \psi_0 \uparrow 2^{j-1}$ (see [Tri92], page 15).

For all $j \in \mathbb{N}_0$ it holds that

$$\begin{aligned} 2^{j\cdot s} \cdot \left\| \widehat{\psi_j} * f \right\|_\infty &\leq 2^{j\cdot s} \cdot \left\| \psi_j \cdot \widehat{f} \right\|_1 \\ &\leq \sum_{k=0}^{\infty} 2^{k\cdot s} \cdot \left\| \psi_k \cdot \widehat{f} \right\|_1 \\ &\leq 2^s \cdot \int_{\mathbb{R}} \left(\xi \mapsto (1+|\xi|)^s \cdot \left| \widehat{f}(\xi) \right| \right) \quad . \end{aligned}$$

□

An estimate of the HOELDER continuity for refinable functions in terms of their refinement mask was derived from this embedding by CONZE and RAUGI [CR90, Con90]. For a summary see [Dau92]. The price to be paid for using an embedding is that we cannot find the maximum s ("the true smoothness") for which the considered refinable function is in $\mathcal{C}^s(\mathbb{R})$.

The estimate can be made more simple in the case that $\widehat{m}$ is a positive function, that is $\forall \xi \in \mathbb{R} \quad \widehat{m}(\xi) \geq 0$.

4.2.6 Theorem (HOELDER continuity of a refinable function). Given the mask m decide:

1. If $\widehat{m}$ is positive, set $s_0 = M_m$.
2. If $\widehat{m}$ is not positive, set $s_0 = \frac{1}{2} \cdot (M_{m*m^*} - 1)$.

Let φ be the refinable function associated with the mask m. Then it holds

$$\forall s \in \mathbb{R} \quad s < s_0 \Rightarrow \varphi \in \mathcal{C}^s(\mathbb{R}) \quad .$$

SOBOLEV **smoothness**

The SOBOLEV smoothness of a refinable function can be characterised similarly to Theorem 4.2.6 [Vil93, Theorem 2.3].

4.2.7 Theorem (SOBOLEV smoothness of a refinable function).
Prerequisite. Given the mask m let $s_0 = \frac{1}{2} \cdot M_{m*m^*}$. We need the condition $\mathcal{B}(\varphi)$ from Definition 2.2.21 saying that the integral translates of φ form a RIESZ basis.
Claim.

1. $\forall s \in \mathbb{R} \quad s < s_0 \Rightarrow \varphi \in H_2^s(\mathbb{R})$
2. $\forall s \in \mathbb{R} \quad \mathcal{B}(\varphi) \wedge \varphi \in H_2^s(\mathbb{R}) \Rightarrow s < s_0$

That means s_0 can be regarded as an accurate measurement of the smoothness of φ.

This smoothness measurement method can be generalised to non-separable multi-dimensional wavelets. [RS96, DGM99]

4.2.3 Simple estimates

The estimates given in this section are already presented in [Thi04], but especially the last and most precise estimation could be simplified a lot by the notion of the circular convolution. The computation of this estimate can be accelerated with the discrete FOURIER transform, but it is still only competitive for very long masks.

Theorem 4.2.6 and Theorem 4.2.7 states that the smoothness of a refinable function depends on the number of factors $\frac{1}{2} \cdot (\mathbf{1}, 1)$ in m and on the remaining factor h. More precisely the spectral radius of either T_h or T_{h*h^*} is the critical quantity. The number of factors $\frac{1}{2} \cdot (\mathbf{1}, 1)$ is easy to handle normally, but the largest eigenvalue of T_h is not. Thus we will focus on the remaining mask h and $\varrho(T_h)$.

4.2.8 Remark. According to Lemma 2.2.27 it is sensible to restrict our considerations to masks h with sum 1 $\left(\widehat{h}(0) = 1\right)$. Hence the sum of the coefficients of $h * h^*$ also equals 1 $\left(\widehat{h * h^*}(0) = \left|\widehat{h}(0)\right|^2 = 1\right)$. According to Theorem 4.2.6 and Theorem 4.2.7 we will consider only matrices T of positive filter polynomials and their filter coefficients will always sum up to 1.

4.2.9 Notation. For brevity we want to use the variables ν and κ for the start and the end index of the filter h.

$$\nu = \min(\operatorname{ix} h)$$
$$\kappa = \max(\operatorname{ix} h)$$

Similar to Definition 3.1.5 we want to denote the size of the mask h with $\#h$.

4.2.10 Lemma. The first and the last non-zero mask coefficient, h_ν and h_κ respectively, are eigenvalues of the matrix T_h.

Proof. Expand the determinant $\det(T_h - \lambda \cdot I)$ for the top and the bottom row. □

There are some simple general ways of estimating the spectral radius of a matrix. E.g. $\varrho(T_h) \leq \|T_h\|$ holds for any matrix norm. We will show that such estimates are too weak in some cases. This should motivate the search for stronger estimates as presented at the end of this section.

The following statements show that the column and row sum matrix norms are bounded from below. Thus estimates based on these norms cannot benefit from the fact that longer filters allow smaller spectral radii.

4.2.11 Lemma.

1. If $\kappa - \nu$ is even, then the row sum norm of the matrix T_h is at least 1.
2. If $\kappa - \nu$ is odd, then the row sum norm of the matrix T_h is at least $\frac{1}{2}$.

Proof.

- Case $2 \mid (\kappa - \nu)$:
  The $\frac{\nu+\kappa}{2}$th row of T_h which is the centre row consists of all mask coefficients $h_\nu, \ldots, h_\kappa$ thus

$$\begin{aligned} \|T_h\|_\infty &= \max_{j \in \operatorname{ix} h} \sum_{k \in \operatorname{ix} h} \left|(T_h)_{j,k}\right| \\ &\geq \sum_{k \in \operatorname{ix} h} \left|(T_h)_{\frac{\nu+\kappa}{2},k}\right| \quad = \sum_{k \in \operatorname{ix} h} |h_k| \\ &\geq \left|\sum_{k \in \operatorname{ix} h} h_k\right| \qquad\qquad = 1 \end{aligned}$$

- Case $2 \nmid (\kappa - \nu)$:
  The $\frac{\nu+\kappa-1}{2}$th row of T_h consists of all mask coefficients except h_κ and the $\frac{\nu+\kappa+1}{2}$th row of T_h

consists of all mask coefficients except h_ν and thus

$$\begin{aligned}
\|T_h\|_\infty &\geq \max_{j\in\left\{\frac{\nu+\kappa-1}{2},\frac{\nu+\kappa+1}{2}\right\}} \sum_{k\in \operatorname{ix} h} \left|(T_h)_{j,k}\right| \\
&= \max\left\{|h_\nu|, |h_\kappa|\right\} + \sum_{k\in \operatorname{ix} h\setminus\{\nu,\kappa\}} |h_k| \\
&\geq \frac{1}{2}\cdot(|h_\nu| + |h_\kappa|) + \sum_{k\in \operatorname{ix} h\setminus\{\nu,\kappa\}} |h_k| \\
&\geq \frac{1}{2}\cdot\left(\sum_{k\in \operatorname{ix} h} |h_k| + \sum_{k\in \operatorname{ix} h\setminus\{\nu,\kappa\}} |h_k|\right) \\
&\geq \frac{1}{2}\cdot\left(1 + \sum_{k\in \operatorname{ix} h\setminus\{\nu,\kappa\}} |h_k|\right) \\
&\geq \frac{1}{2}
\end{aligned}$$

□

The column sum norm might be better suited.

4.2.12 Lemma. The column sum norm of the matrix T_h is at least $\frac{1}{2}$.

Proof. For $\nu = \kappa$ it must be $h_\nu = 1$ (Definition 2.2.24) and thus $\|T_h\|_1 = 1$. For $\nu < \kappa$ the matrix T_h has at least two columns. We consider the first two:

$$\begin{aligned}
\|T_h\|_1 &= \max_{k\in \operatorname{ix} h} \sum_{j\in \operatorname{ix} h} \left|(T_h)_{j,k}\right| \\
&= \max_{k\in\{\nu,\nu+1\}} \sum_{j\in \operatorname{ix} h} \left|(T_h)_{j,k}\right| = \max_{k\in\{0,1\}} \sum_{j\in[k]_2} |h_j| \\
&\geq \frac{1}{2}\cdot \sum_{k\in\{0,1\}} \sum_{j\in[k]_2} |h_j| \\
&\geq \frac{1}{2}\cdot \sum_{j\in \operatorname{ix} h} |h_j| \\
&\geq \frac{1}{2}\cdot \left|\sum_{j\in \operatorname{ix} h} h_j\right| = \frac{1}{2}
\end{aligned}$$

□

4.2.13 Lemma.

1. If $\#h$ is even then the FROBENIUS norm of the matrix T_h is at least $\frac{1}{\sqrt{2}}$.
2. If $\#h$ is odd then the FROBENIUS norm of the matrix T_h is at least $\sqrt{\frac{\#h-1}{2\cdot\#h}}$.

Proof. By counting the number of occurrences of each coefficient of h in T_h we obtain the following relations.

- Case $2 \mid \#h$:

$$\|T_h\|_F^2 = \frac{\#h}{2}\cdot\|h\|_2^2$$

- Case $2 \nmid \#h$:

$$\|T_h\|_F^2 = \frac{\#h-1}{2} \cdot \|h\|_2^2 + \sum_{k \in [\nu]_2} |h_k|^2$$

The EUCLIDean norm of h can be bounded by

$$\begin{array}{lll} \|h\|_2 & \Big| & \text{inequality of quadratic and arithmetic mean} \\ \quad \geq \frac{1}{\sqrt{\#h}} \cdot \|h\|_1 & \Big| & \sum h = 1 \\ \quad \geq \frac{1}{\sqrt{\#h}} & & \end{array}$$

and we obtain for both cases

1.
$$\begin{aligned} \|T_h\|_F &= \sqrt{\frac{\#h}{2}} \cdot \|h\|_2 \\ &\geq \sqrt{\frac{1}{2}} \end{aligned}$$

2.
$$\begin{aligned} \|T_h\|_F &\geq \sqrt{\frac{\#h-1}{2}} \cdot \|h\|_2 \\ &\geq \sqrt{\frac{\#h-1}{2 \cdot \#h}} \quad . \end{aligned}$$

□

So the lower bounds for the FROBENIUS norm are between the row sum norm and the column sum norm.

It is clear that long filters allow for at least the smoothness of short filters simply because long filters have additional degrees of freedom compared with short filters. The next statement quantifies this observation and gives a theoretical limit of the smoothness for a refinable function depending on the length of the mask.

4.2.14 Lemma. The spectral radius of the matrix T_h is always at least $\frac{1}{\#h}$.

$$\varrho\left(T_h\right) \geq \frac{1}{\#h}$$

Proof. We make use of the fact that the diagonal of T_h consist of all coefficients of the mask. We use the index set $\operatorname{ix} h$ for the eigenvalues λ_j, too, although the eigenvalues do not correspond one-to-one to the mask coefficients.

$$\begin{aligned} \#h \cdot \max_{j \in \operatorname{ix} h} \left|\lambda_j\right| &\geq \sum_{j \in \operatorname{ix} h} \left|\lambda_j\right| \\ &\geq \left| \sum_{j \in \operatorname{ix} h} \lambda_j \right| = \left|\operatorname{trace}(T_h)\right| \\ &= \left| \sum_{j \in \operatorname{ix} h} h_j \right| = 1 \end{aligned}$$

□

However the estimate of the smoothness depending on the mask can be refined using $\operatorname{trace}\left(T_h^2\right)$ instead of $\operatorname{trace} T_h$. More generally we observe that if T_h has eigenvalues $\lambda_\nu, \lambda_{\nu+1}, \ldots, \lambda_\kappa$ then T_h^n has eigenvalues $\lambda_\nu^n, \lambda_{\nu+1}^n, \ldots, \lambda_\kappa^n$. Thus $\operatorname{trace}(T_h^n) = \sum_{j \in \operatorname{ix} h} \lambda_j^n$.

It is $T_h \cdot x = (h * x) \downarrow 2$. With the help of Definition 2.2.1 and Lemma 3.3.3 we can express the matrix power as convolution with subsequent down-sampling, as well.

$$T_h^n \cdot x = \left(\mathcal{R}_h^n \delta * x\right) \downarrow 2^n$$

For brevity we substitute

$$\mathcal{R}_h^n \delta = H^n \qquad .$$

We derive the matrix representation

$$T_h^n = \left(\left(H^n\right)_{2^n \cdot j - k} : (j,k) \in (\operatorname{ix} h)^2 \right) \qquad .$$

We realize that the trace of T_h^n is essentially a sum of selected coefficients of H^n, namely

$$\begin{aligned} \operatorname{trace}(T_h^n) &= \sum_{j \in \operatorname{ix} h} \left(H^n\right)_{(2^n-1)\cdot j} \\ &= \sum \left(H^n \downarrow (2^n - 1)\right) \qquad . \end{aligned}$$

It is not necessary to explicitly perform all convolutions which are involved in H^n. We can save a lot of computations if we compute cyclic convolutions with respect to the period $2^n - 1$. Then the trace is given by the zeroth coefficient of the cyclic convolution product.

4.2.15 Definition (Transformation to a periodic filter). We define the operator C_n from $(\mathbb{Z} \to R) \to (\mathbb{Z}_n \to R)$ which turns a straight filter into a circular one. The set $\mathbb{Z}_n$ is the ring of residue classes of $\mathbb{Z}$ with respect to n, that is $\mathbb{Z}_n = \mathbb{Z}/(n \cdot \mathbb{Z})$. This means that the indices of the resulting periodic filter are residue classes. The residue class denoted by $[i]_n$ is defined as $\{j \ : \ i \equiv j \mod n\}$. [Str95, Example 6.21]

$$\left(C_n h\right)_r = \sum_{j \in r} h_j$$

4.2.16 Remark. The usage of residue classes as indices allows us to easily adapt the definitions for convolution and up-sampling for periodic filters. The filters h and g must be of the same period, i.e. $\{h, g\} \subset \mathbb{Z}_n \to R$.

$$\begin{array}{lllll} \text{Down-sampling} & \forall c \in \mathbb{Z} & \forall k \in \mathbb{Z}_n & (h \downarrow c)_k = h_{[c]_n \cdot k} & \\ \text{Up-sampling} & \forall c \in \mathbb{Z} & \forall k \in \mathbb{Z}_n & (h \uparrow c)_k = \displaystyle\sum_{j : k = [c]_n \cdot j} h_j & (4.2.1) \\ \text{Convolution} & & \forall k \in \mathbb{Z}_n & (h * g)_k = \displaystyle\sum_{j \in \mathbb{Z}_n} h_j \cdot g_{k-j} & \end{array}$$

In contrast to straight signals a cyclic signal will not become shorter by down-sampling. Instead the signal is padded periodically. A new property of the cyclic up-sampling is that it can lead to overlapping. These are resolved by summing all overlapping values. Using this definition the following properties hold, namely the transformation to the periodic filter commutes both with up-sampling and with convolution.

$$\begin{array}{ll} \text{Up-sampling} & C_n h \uparrow c = C_n(h \uparrow c) \\ \text{Convolution} & C_n h * C_n g = C_n(h * g) \end{array} \qquad (4.2.2)$$

However the equation $C_n h \downarrow c = C_n(h \downarrow c)$ does not hold in general. For example:

$$\begin{aligned} C_2(\mathbf{1}, 0, -1) \downarrow 2 &= (\mathbf{0}, 0) \downarrow 2 = (\mathbf{0}, 0) \\ C_2((\mathbf{1}, 0, -1) \downarrow 2) &= C_2(\mathbf{1}, -1) = (\mathbf{1}, -1) \end{aligned}$$

Now we can formulate the computation of the trace as

$$\begin{aligned}\operatorname{trace}(T_h^n) &= \left(C_{2^n-1}(H^n)\right)_{[0]} \\ &= \left(C_{2^n-1}\left(h \uparrow 2^{n-1}\right) * \ldots * C_{2^n-1}(h \uparrow 2) * C_{2^n-1}h\right)_{[0]} \quad .\end{aligned} \tag{4.2.3}$$

The last convolution need not to be performed completely since we are only interested in the $[0]_{2^n-1}$ indexed coefficient of the result. But we have still to perform $n-2$ cyclic convolutions. This can be drastically reduced to about $2 \cdot \log_2 n$ convolutions by using the following recursion scheme. It expresses the $2 \cdot k$ and the $2 \cdot k+1$ times refinement in terms of the k times refinement. Thus in each recursion step the number of refinements is halved. This scheme is also known for integer powers with respect to associative operations such as the matrix multiplication.

$$\begin{aligned}C_{2^n-1}\left(H^{2\cdot k}\right) &= C_{2^n-1}\left(H^k \uparrow 2^k\right) * C_{2^n-1}\left(H^k\right) \\ &= C_{2^n-1}\left(H^k\right) \uparrow 2^k * C_{2^n-1}\left(H^k\right) \\ C_{2^n-1}\left(H^{2\cdot k+1}\right) &= C_{2^n-1}h \uparrow 2^{2\cdot k} * C_{2^n-1}\left(H^k\right) \uparrow 2^k * C_{2^n-1}\left(H^k\right)\end{aligned}$$

A 2^k times refinement is the relative best case where we need k convolutions. A $2^{k+1}-1$ times refinement is the relative worst case where we need $2 \cdot k$ convolutions.

It is worth noting that the up-sampling is especially simple here because overlaps cannot occur. That is up-sampling is just a rearrangement of coefficients.

4.2.17 Lemma. Let $h \in \mathbb{Z}_q \to R$ and let c and q be relatively prime integers, then the up-sampling $h \uparrow c$ does not cause overlaps.

Proof. According to the definition of the cyclic up-sampling (4.2.1) the claim is equivalent to the statement that for each k from $\mathbb{Z}_q$ there is only one j with $k = [c]_q \cdot j$. Since c is relatively prime to q there is an inverse $[c]_q^{-1}$ of $[c]_q$ in the ring $\mathbb{Z}_q$. Thus for each k there is exactly one associated j, namely $[c]_q^{-1} \cdot k$. □

In our situation it is $q = 2^n - 1$ and $c = 2^k$. They are relatively prime because the only prime factor of c is 2 whereas q is odd. We can explicitly express the inverse as $[c]_q^{-1} = [2^{n-k}]_q$.

Now we know how to reduce the number of convolutions but have not considered the convolutions themselves. We know that the discrete FOURIER transform turns cyclic convolutions into multiplications which dramatically speeds up the computation. We will also see that with the FOURIER transform we even need no prior reduction of the number of convolutions.

4.2.18 Definition (Discrete FOURIER Transform). The *Discrete FOURIER Transform* is a function from $(\mathbb{Z}_q \to \mathbb{C}) \to (\mathbb{Z}_q \to \mathbb{C})$. For a cyclic signal h from $\mathbb{Z}_q \to \mathbb{C}$ it is defined as:

$$\begin{aligned}z &= e^{-\frac{2\cdot\pi\cdot\mathrm{i}}{q}} \\ \operatorname{DFT}(h)_k &= \sum_{j\in\mathbb{Z}_q} h_j \cdot z^{k\cdot j}\end{aligned}$$

Where the exponentiation with a set of numbers (resulting from the residue class $k \cdot j$) is meant as exponentiation with an arbitrary element of the set. This is unambiguous since all elements lead to the same power.

4.2.19 Lemma (Inverse Discrete FOURIER Transform). The Discrete FOURIER Transform can be inverted by

$$\begin{aligned}z &= e^{\frac{2\cdot\pi\cdot\mathrm{i}}{q}} \\ \operatorname{DFT}^{-1}(h)_k &= \frac{1}{q} \cdot \sum_{j\in\mathbb{Z}_q} h_j \cdot z^{k\cdot j} \quad .\end{aligned} \qquad \text{[VK95]}$$

It follows that

$$\begin{aligned} h_{[0]} &= \mathrm{DFT}^{-1}(\mathrm{DFT}(h))_{[0]} \\ &= \frac{1}{q} \cdot \sum (\,\mathrm{DFT}(h)) \quad . \end{aligned} \tag{4.2.4}$$

4.2.20 Lemma. The Discrete FOURIER Transform turns convolution into element-wise multiplication.

$$\mathrm{DFT}(h * g)_k = \mathrm{DFT}(h)_k \cdot \mathrm{DFT}(g)_k$$

Thus using this transform we can simplify the convolution a lot.

$$\begin{aligned} \mathrm{DFT}\left(C_{2^n-1}(H^n)\right) &= \mathrm{DFT}\left(C_{2^n-1}h \uparrow 2^{n-1} * \ldots * C_{2^n-1}h \uparrow 2 * C_{2^n-1}h\right) \\ &= \mathrm{DFT}(C_{2^n-1}h \uparrow 2^{n-1}) \cdot \ldots \cdot \mathrm{DFT}(C_{2^n-1}h \uparrow 2) \cdot \mathrm{DFT}(C_{2^n-1}h) \end{aligned}$$

But we also want to reduce the number transforms to a minimum. Can we compute $\mathrm{DFT}(h \uparrow c)$ easily from $\mathrm{DFT}(h)$? For the continuous FOURIER transform and for the FOURIER series expansion we know that dilation is turned into shrinkage. Analogously it is $\mathrm{DFT}(h \uparrow c) = \mathrm{DFT}(h) \downarrow c$. If c is not relatively prime to the length of h then $h \uparrow c$ involves summing of overlapping values, but on the other side $\mathrm{DFT}(h) \downarrow c$ omits some FOURIER coefficients and periodically pads the remaining ones.

$$\mathrm{DFT}\left(C_{2^n-1}(H^n)\right) = \mathrm{DFT}(C_{2^n-1}h) \downarrow 2^{n-1} \cdot \ldots \cdot \mathrm{DFT}(C_{2^n-1}h) \downarrow 2 \cdot \mathrm{DFT}(C_{2^n-1}h)$$

In this representation we need to compute the FOURIER transform only once. Since 2^k and $2^n - 1$ are relatively prime the down-sampling is a plain rearrangement. To compute the total product we have to perform $(n-1) \cdot (2^n - 1)$ multiplications. This can be reduced to at most $(2^n - 1)$ multiplications by a more detailed analysis of the rearrangements. The kth coefficient of the product of the FOURIER transform is given by

$$\mathrm{DFT}\left(C_{2^n-1}(H^n)\right)_k = \prod_{j=0}^{n-1} \left(\mathrm{DFT}(C_{2^n-1}h)\right)_{k \cdot [2^j]_{2^n-1}} \quad .$$

Since $1 \equiv 2^n \mod (2^n - 1)$ the indices of $\mathrm{DFT}(C_{2^n-1}h)$ in the product for the coefficients $\mathrm{DFT}(C_{2^n-1}(H^n))_{k \cdot [2^m]_{2^n-1}}$ for specific k are only cycled depending on m. That is for fixed k and varying m the coefficients $\mathrm{DFT}(C_{2^n-1}(H^n))_{k \cdot [2^m]_{2^n-1}}$ are equal.

The sequence $k \cdot [1]_{2^n-1}, k \cdot [2]_{2^n-1}, \ldots, k \cdot [2^{n-1}]_{2^n-1}$ may contain equal elements. If two elements are equal then their successors are equal, too. Thus equal elements lead to cycles.

This situation can be described by group theory as follows: There is a group $(G, [1]_{2^n-1}, \cdot)$ with elements from $\mathbb{Z}_{2^n-1}$, namely $G = \left\{[2^j]_{2^n-1} \;:\; j \in \{0, \ldots, n-1\}\right\}$. The set $G \cdot x$ of all possible products between elements of G and a certain element x from $\mathbb{Z}_{2^n-1}$ is called the *orbit* of x, $G \cdot x = \{g \cdot x \;:\; g \in G\}$. The *stabiliser* (also *group of isotropy*, [Str98, Theorem 17.14], [Wei05, Stabilizer]) of x is the subgroup from G with the elements which let x unchanged, i.e. $\mathrm{stab}_G(x) = \{g \;:\; g \in G \wedge g \cdot x = x\}$. For example, for $n = 6$ it is $G = \left\{[1], [2], [4], [8], [16], [32]\right\}$ and we obtain for instance

$$\begin{aligned} G \cdot [5] &= \left\{[5], [10], [20], [40], [17], [34]\right\} & \mathrm{stab}_G([5]) &= \left\{[1]\right\} \\ G \cdot [9] &= \left\{[9], [18], [36]\right\} & \mathrm{stab}_G([9]) &= \left\{[1], [8]\right\} \\ G \cdot [21] &= \left\{[21], [42]\right\} & \mathrm{stab}_G([21]) &= \left\{[1], [4], [16]\right\} \quad . \end{aligned}$$

According to [Bre05] there is a nice connection to the binary representation of the numbers. The multiplication $2 \cdot x$ in $\mathbb{Z}_{2^n-1}$ means rotating the n digit binary representation of x by one to the left. Thus if the bit pattern of x contains an m-periodic pattern then we obtain m distinct bit patterns by bit rotation, i.e. $\#(G \cdot x) = m$. Indeed 9 has the binary representation 001001 and 21 corresponds to 010101.

In general it holds the following connection between the cardinalities of G, of the orbit and of the stabiliser:

$$\#G = \#(G \cdot x) \cdot \#\,\mathrm{stab}_G(x) \quad .$$

If $\#G$ is prime then the sequence $[k]_{2^n-1}, [k \cdot 2]_{2^n-1}, \ldots, [k \cdot 2^{n-1}]_{2^n-1}$ has prime length and there cannot be true sub-cycles. In this case the set of indices $\{1, 2, \ldots, 2^n - 2\}$ can be partitioned into $\frac{2^n-2}{n}$ orbits of size n. That $\frac{2^n-2}{n}$ is an integer can also be verified by FERMAT's Little Theorem. The formula gives an idea of how fast the number of orbits grows for increasing n, though we cannot give such a simple expression for general n.

We partition the set of all indices into orbits with respect to the powers of two, that is

$$\mathcal{K} = \left\{ G \cdot k \ : \ k \in \mathbb{Z}_{2^n-1} \right\} \quad .$$

Orbits which overlap are equal. Since $\mathcal{K}$ is an ordinary set, each orbit occurs only once. For each orbit we have to evaluate a product, but since we do this only once per orbit this leads to a sophisticated computation of the trace of the dyadic band matrix power. Starting from (4.2.3) and (4.2.4) we obtain

$$\begin{aligned} \operatorname{trace}(T_h^n) &= \frac{1}{2^n - 1} \cdot \sum_{k \in \mathbb{Z}_{2^n-1}} \operatorname{DFT}\left(C_{2^n-1}(H^n)\right)_k \\ &= \frac{1}{2^n - 1} \cdot \sum_{\mathcal{J} \in \mathcal{K}} \#\mathcal{J} \cdot \left(\prod_{j \in \mathcal{J}} \operatorname{DFT}(C_{2^n-1} h)_j \right)^{n/\#\mathcal{J}} \quad . \end{aligned}$$

In this formula each coefficient of the FOURIER transformed signal is invoked only once. If we neglect the organisation effort for the partition into orbits we achieve roughly linear time consumption with respect to $2^n - 1$. The slowest part of the computation is now the FOURIER transform which needs time proportional to $n \cdot (2^n - 1)$.

We have now derived an algorithm which needs l computation steps in order to periodically sum a filter h of length l. For computing the trace of the n-th power of the dyadic band matrix of h we need computation time proportional to $n \cdot 2^n$. That is the computation power increases exponentially with respect to n. In contrast to that with a slightly optimised implementation of the matrix power we can compute the trace in about $2 \cdot \log_2 n \cdot l^2$ steps. We realise that the algorithm based on cyclic convolution can only compete for very small n. Especially for $\operatorname{trace}(T_h^2)$ it turns out to be very handy. We will concentrate on this case for the rest of this section. It is

$$\begin{aligned} \operatorname{trace}(T_h^2) &= \sum (C_3 h \uparrow 2 * C_3 h) \\ \text{because } 2 \equiv -1 \mod 3 \quad & \\ &= \sum (C_3 h \uparrow (-1) * C_3 h) \\ &= (C_3 h)_{[0]}^2 + (C_3 h)_{[1]}^2 + (C_3 h)_{[2]}^2 \quad . \end{aligned}$$

4.2.21 Theorem. For a given mask h with finite support let $y = C_3 h$ and $B_h = \sqrt{y_{[0]}^2 + y_{[1]}^2 + y_{[2]}^2}$. Then a lower bound for the spectral radius is given by

$$\frac{1}{\sqrt{\#h}} \cdot B_h \leq \varrho\left(T_h\right) \quad .$$

If the eigenvalues of T_h are all real then there is a simple upper bound:

$$\varrho\left(T_h\right) \leq B_h \quad .$$

Proof.

1.

$$\begin{aligned} \#h \cdot \max_{j \in \operatorname{ix} h} \left|\lambda_j\right|^2 &\geq \sum_{j \in \operatorname{ix} h} \left|\lambda_j\right|^2 \\ &\geq \left| \sum_{j \in \operatorname{ix} h} \lambda_j^2 \right| \\ &= \left| \operatorname{trace}(T_h^2) \right| = B_h^2 \end{aligned}$$

2.

$$\begin{aligned}\max_{j\in \mathrm{ix}\, h} \left|\lambda_j\right|^2 &\leq \sum_{j\in \mathrm{ix}\, h} \left|\lambda_j\right|^2 \\ &= \sum_{j\in \mathrm{ix}\, h} {\lambda_j}^2 \quad = B_h^2\end{aligned}$$

□

4.2.22 Remark. One might hope that the eigenvalues of matrices of the form T_{h*h^*} are always real. The example $h = (2, 0, 0, -1)$ disproves this assumption. It is $h * h^* = (-2, 0, 0, 5, 0, 0, -2)$ and T_{h*h^*} has the eigenvalues $\pm 1 \pm 3\mathrm{i}, -2, -2, 5$.

Indeed there is a family of filters h which lead to a constant value of B_{h*h^*} according to Theorem 4.2.21 while the spectral radius of T_{h*h^*} is not bounded. Such a family is $\{(1 + x, 0, 0, -x) \ : \ x \in \mathbb{R}\}$.

4.2.23 Remark. One might also assume that the existence of a complementary filter g (i.e. a filter g such that h and g allow for perfect reconstruction, see Definition 2.2.9) already implies that all eigenvalues of T_{h*h^*} are real. This is also not true since for $h = (2, 0, 0, -1), g = (0, 0, 1, 0)$ the filter g is complementary to h.

Whether the spectral radius is closer to the upper bound or closer to the lower bound depends on the distribution of the eigenvalues of the matrix T_h. In the case that the eigenvalues have similar magnitude the spectral radius will be close to the lower bound. If there are only a few large eigenvalues and many small ones then the spectral radius will be close to the upper bound.

A simple lower estimate for the spectral radius that does not depend on the filter coefficients is given by

4.2.24 Lemma.

$$\varrho\left(T_h\right) \geq \frac{1}{\sqrt{3 \cdot \#h}} \quad .$$

Proof. We derive this from Theorem 4.2.21 using the inequality of quadratic and arithmetic mean

$$\begin{aligned}\sqrt{\frac{1}{3} \cdot \left(y_{[0]}^2 + y_{[1]}^2 + y_{[2]}^2\right)} &\geq \frac{1}{3} \cdot \left(y_{[0]} + y_{[1]} + y_{[2]}\right) \\ \frac{1}{\sqrt{3}} \cdot B_h &\geq \frac{1}{3}\end{aligned}$$

and the last holds because

$$y_{[0]} + y_{[1]} + y_{[2]} = \sum h = 1$$

due to Remark 4.2.8. □

4.2.25 Corollary. For a refinable function or distribution φ with respect to h (containing no smoothness factor $\frac{1}{2} \cdot (\mathbf{1}, 1)!$) the computation of the SOBOLEV smoothness according to Theorem 4.2.7 yields the estimate

$$\begin{aligned}s_0 &= \frac{1}{2} \cdot M_{h*h^*} \\ &= \frac{1}{2} \cdot \left(-1 - \log_2 \varrho\left(T_{h*h^*}\right)\right) \\ &\leq \frac{1}{2} \cdot \left(-1 - \log_2 \frac{1}{\sqrt{3 \cdot \#(h * h^*)}}\right) \\ &= \frac{1}{2} \cdot \left(-1 + \frac{1}{2} \cdot \log_2(3 \cdot (2 \cdot \#h - 1))\right) \\ &\leq \frac{1}{4} \cdot \log_2 \left(\frac{3}{2} \cdot \#h\right) \quad .\end{aligned}$$

This means that apart from adding smoothness factors the bound for the SOBOLEV smoothness can be increased by 1 only if the mask length grows by a factor of 16! The mask (free of smoothness factors) of a refinable function from $H_2^1(\mathbb{R})$ must have at least length 11 ($\frac{32}{3}$).

We will now consider an optimisation for estimating the SOBOLEV smoothness of φ. According to Theorem 4.2.7 we have to process $h * h^*$ instead of the pure filter mask h to that end. Then $B_{h*h^*} = \sqrt{\sum_{j\in\mathbb{Z}_3} \left(C_3(h*h^*)\right)_j^2}$. Because the periodic summation and the cyclic convolution commute ((4.2.2) from Remark 4.2.16) we can avoid the need for an explicit convolution $h * h^*$. With y as defined in Theorem 4.2.21 and

$$\begin{aligned} p_1 &= y_{[0]} + y_{[1]} + y_{[2]} = 1 \\ p_2 &= y_{[0]}^2 + y_{[1]}^2 + y_{[2]}^2 \end{aligned}$$

we obtain

$$\begin{aligned} \left(C_3(h*h^*)\right)_{[0]} &= y_{[0]}\cdot y_{[0]} + y_{[1]}\cdot y_{[1]} + y_{[2]}\cdot y_{[2]} = p_2 \\ \left(C_3(h*h^*)\right)_{[1]} &= y_{[0]}\cdot y_{[1]} + y_{[1]}\cdot y_{[2]} + y_{[2]}\cdot y_{[0]} = \frac{p_1^2 - p_2}{2} \\ \left(C_3(h*h^*)\right)_{[2]} &= y_{[0]}\cdot y_{[2]} + y_{[1]}\cdot y_{[0]} + y_{[2]}\cdot y_{[1]} = \frac{p_1^2 - p_2}{2} \end{aligned}$$

and thus

$$\begin{aligned} B_{h*h^*} &= \sqrt{p_2^2 + 2\cdot\left(\frac{1-p_2}{2}\right)^2} \\ &= \sqrt{\frac{3}{2}\cdot\left(p_2 - \frac{1}{3}\right)^2 + \frac{1}{3}} \quad . \end{aligned}$$

4.2.4 Examples

We will now compare our simple estimates with the exact regularities provided by Theorem 4.2.7 for two standard families of wavelet bases. The considered wavelet bases have filter polynomials that are not positive in general thus the HOELDER smoothness estimate according to Theorem 4.2.6 is derived from the SOBOLEV smoothness. Hence we only consider estimates of the SOBOLEV smoothness. The orthogonal DAUBECHIES wavelets as well as the biorthogonal COHEN-DAUBECHIES-FEAUVEAU wavelets (CDF) are chosen because they can be automatically constructed up to high orders (see [Dau92, Sections 6.1 and 8.3.4]). The considered filter masks lead to transition matrices with real eigenvalues and thus both estimates of Theorem 4.2.21 can be applied.

The complete algorithm for estimating the SOBOLEV smoothness is

1. Let m be the filter mask.
2. Divide m by the highest possible power $\left(\frac{1}{2}\cdot(\mathbf{1},1)\right)^K$ (alternatively divide $\widehat{m}(\xi)$ by $\left(1+e^{-\mathrm{i}\cdot\xi}\right)^K$), the quotient is h.
3. Compute the sums $y_k = \sum_{j\in\mathbb{Z}} h_{k+3\cdot j}$ for $k \in \{0,1,2\}$.
4. Compute the square sum $p_2 = y_0^2 + y_1^2 + y_2^2$.
5. Compute $B_{h*h^*} = \sqrt{\frac{3}{2}\cdot\left(p_2 - \frac{1}{3}\right)^2 + \frac{1}{3}}$.
6. Eventually the SOBOLEV smoothness limit s_0 is bounded by

$$s_0 \le K - \log_4\left(2\cdot B_{h*h^*}\right) + \frac{1}{2}\cdot\log_4\left(2\cdot\#h - 1\right)$$

and further if one knows that the eigenvalues are all real then

$$K - \log_4\left(2\cdot B_{h*h^*}\right) \le s_0 \quad .$$

4.2.26 Remark. Step 2 is numerical critical because the resulting filter has coefficients that vary heavily in magnitude. Thus even the simple criterion of the sum of the coefficients being 1 is violated!

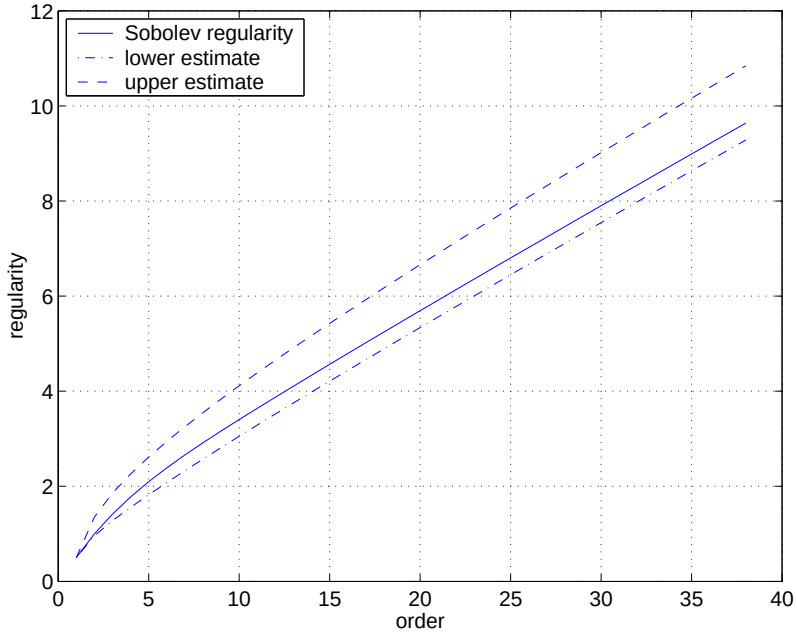


Figure 4.2: SOBOLEV smoothness of DAUBECHIES wavelets (${}_N\phi$ as in [Dau92], ${}_1\phi$ is the HAAR generator) depending on the order of the wavelets.

Orthogonal DAUBECHIES wavelets

For a given power of the factor $\frac{1}{2}\cdot(\mathbf{1},1)$ in m (this is considered as the *order*) the DAUBECHIES wavelet filter is the shortest one that leads to an orthogonal wavelet basis. Actually there are several filters possible for one order but they all share the same filter $m * m^*$ and thus the same SOBOLEV smoothness. Figure 4.2 shows that the upper estimate of the smoothness is at most 1.5 too high and the lower estimate at most 0.5 too low.

Biorthogonal spline wavelets (CDF)

In contrast to orthogonal bases the CDF wavelet basis consists of two different generator functions, that are a primal and a dual generator. The dual generator ${}_{\widetilde{N}}\widetilde{\phi}$ is a $\widetilde{N}$th order B-spline, its Sobolev smoothness is $s_0 = \widetilde{N} - \frac{1}{2}$ and this is also the result of our estimate due to Theorem 4.2.21 since the filter consists only of a power of $\frac{1}{2}\cdot(\mathbf{1},1)$ and the eigenspectrum of the transition matrix of the remaining filter of length 1 will be estimated exactly.

That is why the more interesting function is the primal generator ${}_{\widetilde{N},N}\phi$ whose filter contains the Nth power of $\frac{1}{2}\cdot(\mathbf{1},1)$ and the remaining filter depends only on $\frac{N+\widetilde{N}}{2}$. The dependency on N is clear thus we content ourselves with the analysis of $\left({}_{N,N}\phi : N\in\mathbb{N}\right)$ which is a sequence of functions of decreasing smoothness as can be seen in Figure 4.3.

The maximum deviation from the lower bound is 0.4 and the deviation from the upper bound is at most 1.5.

4.3 Enhancing the smoothness of primal generators

In the previous section we have learnt how to compute the smoothness of refinable functions. We are now going to add the smoothness of the primal generator to the optimisation derived in Section 3.3.2. The smoothness is determined by the number of smoothness factors $\frac{1}{2}\cdot(\mathbf{1},1)$ in the mask and the spectral radius of the transition matrix of the remaining mask. Thus there are two approaches to make the primal

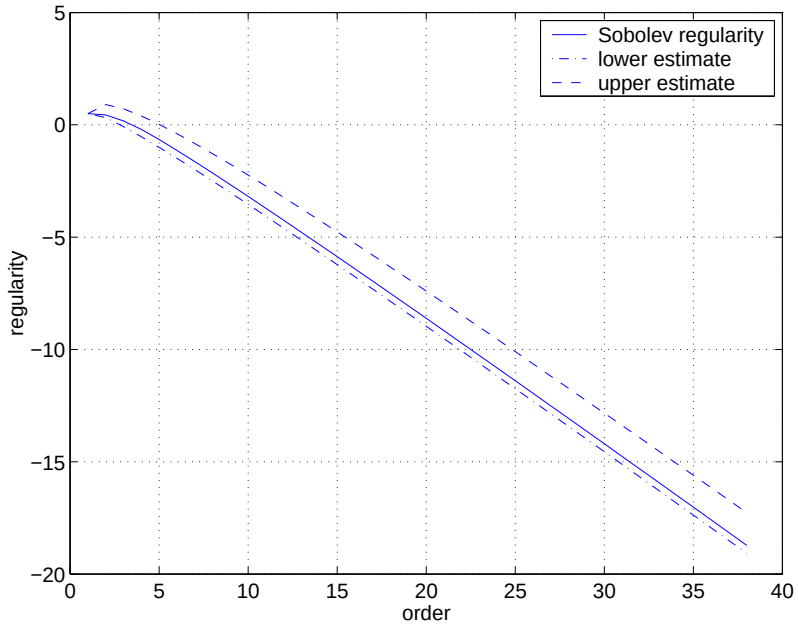


Figure 4.3: SOBOLEV smoothness of the CDF primal generator ${}_{N,N}\phi$ depending on the order N.

generator smoother. On the one hand we can increase the number of smoothness factors and on the other hand we can decrease the spectral radius connected to the remaining mask. The first approach is targeted at a discrete parameter with a predictable influence and the second one deals with continuous parameters which requires non-linear optimisation techniques.

4.3.1 Reducing the spectral radius

Due to Corollary 4.2.25 the smoothing effect of spectral radius reduction is not big. Nonetheless sometimes one can get a bit more smoothness when giving up some smoothness factors [Oja98] and replacing them by general factors which minimise the spectral radius.

Let σ from $\mathbb{R} \times \ell_0(\mathbb{Z}) \to \mathbb{R}$ be a smoothness (or better: roughness) estimate. We add it as a penalty term to our optimisation criterion.

$$\underset{(c,s)}{\operatorname{argmin}} \left(\|c \cdot \psi + s * \varphi - f\|_2 + \sigma(c,s) \right)$$

Note that the power of the term for the difference of wavelet and pattern is irrelevant for the optimisation result of pure matching. Since we add a penalty term it is now worth considering the power in more detail. It holds

$$\underset{(c,s)}{\operatorname{argmin}} \left(\|c \cdot \psi + s * \varphi - f\|_2 \right) = \underset{(c,s)}{\operatorname{argmin}} \left(\|c \cdot \psi + s * \varphi - f\|_2^2 \right) \quad .$$

In contrast to the function $(c,s) \mapsto \|c \cdot \psi + s * \varphi - f\|_2$ without a power the function $(c,s) \mapsto \|c \cdot \psi + s * \varphi - f\|_2^2$ is differentiable (also at the minimum) and the derivative is a linear operator. So we change the minimisation criterion to

$$\underset{(c,s)}{\operatorname{argmin}} \left(\|c \cdot \psi + s * \varphi - f\|_2^2 + \sigma(c,s) \right) \quad .$$

There are several sensible choices for σ. All of them are based on the SOBOLEV smoothness which is determined by the eigenvalue spectrum of the transition matrix. Since we have (still) no constraint on

the number of smoothness factors in the low-pass smoothness factors will occur only accidentally in the low-pass and we skip an extra handling of them.

Given the low-pass h (corresponding to φ) and the high-pass g (corresponding to ψ) the mask g' of the matched high-pass is $g' = g + \frac{1}{c} \cdot (s \uparrow 2) * h$. According to Definition 2.2.12 the dual low-pass is $\widetilde{h}' = (g' \leftarrow 1)\_$. Eventually we obtain the transition matrix M of the dual low-pass ($M = T_{\widetilde{h}' * \widetilde{h}'^*}$) whose spectral radius depends decreasingly on the smoothness of the generator. That is, the smaller the spectral radius the smoother the generator.

With the dependency of M on s and c in mind we can describe some choices for σ.

- The spectral radius of M is the exact choice, i.e. $\sigma(c, s) = \varrho(M)$. But it needs more computation than the alternatives below and it is numerically not as stable as other ones.
- The sum of the squares of the eigenvalues of M, i.e. $\sigma(c, s) = \operatorname{trace} M^2$, can be computed very efficiently (Theorem 4.2.21). But it is only an upper bound if all eigenvalues are real and since we cannot assert that we cannot use this estimate here, unfortunately.
- The sum norm of M, i.e. $\sigma(c, s) = \|M\|_1$
- The FROBENIUS norm of M, i.e. $\sigma(c, s) = \|M\|_F$. This is similar to $\left\|\widetilde{h}' * \widetilde{h}'^*\right\|_2$, which is equal to $\left\|\xi \mapsto \left|\widehat{\widetilde{h}}'(\xi)\right|^2\right\|_2$. Because the mapping from s to $\widetilde{h}'$ is affine, the mapping to $\widehat{\widetilde{h}}'$ is linear and the norm is convex, the mapping from s to $\left\|\xi \mapsto \left|\widehat{\widetilde{h}}'(\xi)\right|^2\right\|_2$ is convex. A mapping with respect to c is certainly not convex. In practice both the FROBENIUS norm and the autocorrelation norm have exhibited the best numerical stability.

Because of the division by c in the lifting step the magnitude of c is essential. Small values of c lead to big lifting coefficients and big coefficients in g' and $\widetilde{h}'$. Big coefficients in $\widetilde{h}'$ lead to large estimates of the spectral radius and tend to produce large eigenvalues. This is the reason why the optimisation with a penalty term mainly increases c. Eventually the practice shows that without support by smoothing factors in the opposite basis it is hardly possible to achieve considerable smoothness or at least regular functions as generators. Figure 4.4 shows an example.

4.3.2 Adding smoothness factors

In the optimisation approach we derived in Section 3.3.2 we could choose the low-pass filter h freely. The choice of h limits the choices of complementary high-pass filters g. The dual low-pass filter $\widetilde{h}$ depends only on g by $\widetilde{h} = (g \leftarrow 1)\_$ (Definition 2.2.12). How can we assert that $\widetilde{h}$ contains, say, m times $\frac{1}{2} \cdot (\mathbf{1}, 1)$, i.e.

$$\exists \widetilde{h}' \quad \widetilde{h} = \widetilde{h}' * \left(\frac{1}{2} \cdot (\mathbf{1}, 1)\right)^{*m} \qquad ?$$

This question was already answered in Theorem 2.2.18. With the connection $\widetilde{h}' = (g' \leftarrow 1)\_$ this property is equivalent to

$$\exists g' \quad g = g' * \left(\frac{1}{2} \cdot (\mathbf{1}, 1)\right)^{*m}\_$$

and

$$\exists g' \quad g = g' * \left(\frac{1}{2} \cdot (\mathbf{1}, -1)\right)^{*m} \qquad . \tag{4.3.1}$$

We realize that m factors $\frac{1}{2} \cdot (\mathbf{1}, 1)$ in the dual low-pass filter $\widetilde{h}$ are equivalent to m factors $\frac{1}{2} \cdot (\mathbf{1}, -1)$ in the primal high-pass filter g. Figure 4.5 shows how these factors affect generator and wavelet functions. Convolving with the mask $\frac{1}{2} \cdot (\mathbf{1}, -1)$ means computing differences. This mask factor is related to vanishing moments which are introduced in the following section.

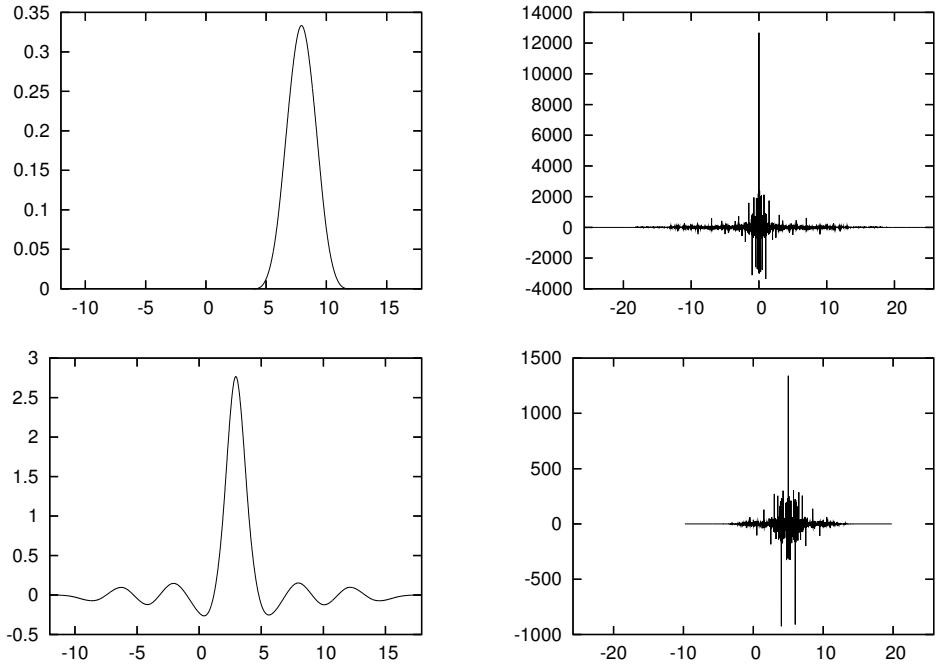


Figure 4.4: Match of wavelets with reduced spectral radius of the transition matrix. For high weighting of the smoothness term the optimisation process becomes numerically unstable. We show here the result of highest possible weighting using the FROBENIUS criterion.

Vanishing moments

4.3.1 Definition (Moment). The sequence M of *moment*s is a sequence of functionals from $\mathbb{N}_0 \to (\mathbb{R} \to \mathbb{R}) \to \mathbb{R}$. If $\left(t \mapsto f(t) \cdot t^j\right) \in \mathcal{L}_1(\mathbb{R})$ then the jth moment of a function f is defined by

$$M_j f = \int_{\mathbb{R}} \left(t \mapsto f(t) \cdot t^j\right) \quad .$$

4.3.2 Remark.

- The zeroth moment is the ordinary integral. If it is limited to an interval of length l then $\frac{M_0 f}{l}$ is often referred to as the *direct current component*.
- The first moment is the torsional moment if the function is considered as the mass distribution of a stick. If the function is translated until the torsional moment becomes zero ($M_1(f \leftarrow x) = 0$) you find the centre x of gravity. It can be computed by $x = \frac{M_1 f}{M_0 f}$.

4.3.3 Definition (Vanishing moments). A *vanishing moment* is a moment of a function that is zero. A function f has m vanishing moments ($m > 0$) if

$$\forall j \in \{0, \ldots, m-1\} \qquad M_j f = 0 \quad .$$

4.3.4 Remark. There are other characterisations for vanishing moments. Here are four alternative characterisations.

1. Scalar product with polynomial functions
   The functional M_j computes a scalar product with a power function with exponent j. The power functions for exponents from 0 to $m-1$ are a basis for all polynomials of degree until $m-1$. Thus we can state:
   The function f has m vanishing moments if and only if the scalar product of f with any polynomial of degree up to $m-1$ is zero.

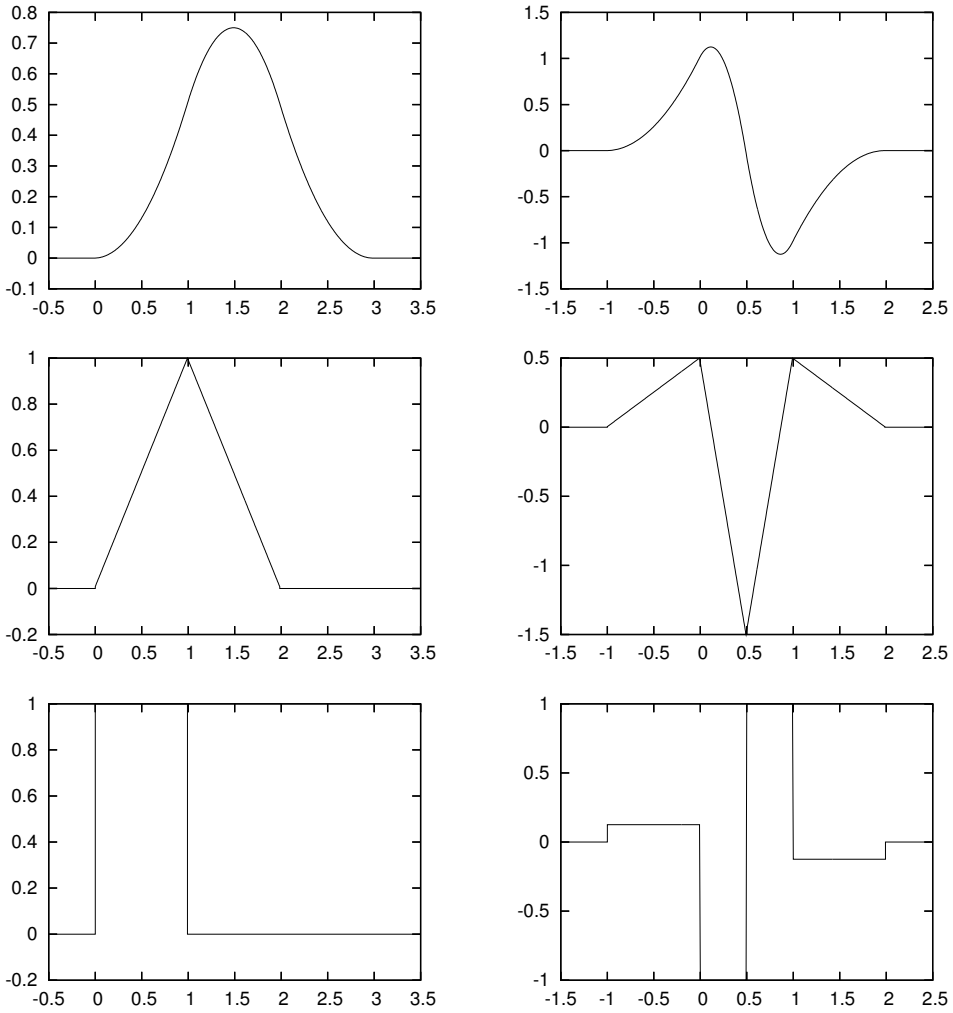


Figure 4.5: Shifting the factor $\frac{1}{2} \cdot (\mathbf{1}, 1)$ from the low-pass to the high-pass let the pair of generator and wavelet of CDF-3,1 basis turn to that of CDF-2,2, and CDF-1,3. In each step the smoothness decreases and the number of vanishing moments of the wavelet increases.

2. Convolution with polynomial functions
   The space of polynomial functions up to a certain degree is invariant with respect to translation of the functions. Therefore if f has m vanishing moments, then each translate of the function f has m vanishing moments, too. Further if p is a polynomial function with a degree below m then not only $\langle f, p^* \rangle = 0$, but even $f * p = 0$.
   Conversely m distinct translates of a polynomial function p of degree $m-1$, say all neighbouring integral translates of p, constitute a basis of the space of all polynomial functions of degree below m. Thus if for a polynomial function of degree $m-1$ holds $f * p = 0$ then f has m vanishing moments.
   Summarised: Let p be a polynomial function of degree $m-1$. The function f has m vanishing moments if and only if $f * p = 0$.
3. Zeros of the FOURIER symbol
   It is possible to characterise the number of vanishing moments by multiplicity of zeros in the frequency domain.

$$\begin{aligned}
\widehat{f}(0) &= \int_{\mathbb{R}} f \\
\widehat{f'} &= \mathcal{F}(t \mapsto -\mathrm{i} \cdot t \cdot f(t)) \\
\widehat{f'}(0) &= -\mathrm{i} \cdot \int_{\mathbb{R}} \big(t \mapsto t \cdot f(t)\big) \\
\widehat{f^{(n)}}(0) &= (-\mathrm{i})^n \cdot \int_{\mathbb{R}} \big(t \mapsto t^n \cdot f(t)\big)
\end{aligned}$$

   Consequently if $\widehat{f} \in C^m(\mathbb{R})$, then f has m vanishing moments if and only if $\widehat{f}$ has m zeros at 0, i.e.

$$\forall j \in \{0, \ldots, m-1\} \quad \widehat{f}^{(j)}(0) = 0 \quad .$$

   This allows for a generalisation: An mth order pole of $\widehat{f}$ at 0 can be considered as $-m$ vanishing moments.
4. Derivatives of well behaved functions
   If $\widehat{f}$ is continuous in 0 then $\omega \mapsto \omega^m \cdot \widehat{f}(\omega)$ has an mth order zero at 0. Thus if $\widehat{f}$ is continuous in 0, f is m times differentiable and, say, $\forall j \in \{0, \ldots, m\}$ $\widehat{f^{(j)}} \in \mathcal{L}_2(\mathbb{R})$ (i.e. $f \in W_2^m(\mathbb{R})$) then according to the previous item $f^{(m)}$ has m vanishing moments. Consequently the m times integral of f has $-m$ vanishing moments.

We will now see that the number of factors $\frac{1}{2} \cdot (\mathbf{1}, -1)$ in the high-pass filter is equal to the number of vanishing moments in the corresponding wavelet.

4.3.5 Lemma.
Prerequisite. Let φ be a function from $\mathcal{L}_1(\mathbb{R})$ with no vanishing moment.
Claim. The function $2 \cdot (g * \varphi) \downarrow 2$ has m vanishing moments if and only if g has the factor $(\mathbf{1}, -1)^{*m}$.

Proof. It is easier to perform the proof in the frequency domain.

$$\mathcal{F}(2 \cdot (g * \varphi) \downarrow 2) = (\widehat{g} \cdot \widehat{\varphi}) \uparrow 2$$

- $(\mathbf{1}, -1)^{*m} \mid g$ implies that $2 \cdot (g * \varphi) \downarrow 2$ has m vanishing moments
  Let $g = (\mathbf{1}, -1)^{*m} * g'$.

$$\begin{aligned}
\widehat{(\mathbf{1}, -1)}(\omega) &= 1 - e^{-\mathrm{i} \cdot \omega} \\
\mathcal{F}(g * \varphi)(\omega) &= \mathcal{F}\Big((\mathbf{1}, -1)^{*m} * g' * \varphi\Big)(\omega) \\
&= \Big(1 - e^{-\mathrm{i} \cdot \omega}\Big)^m \cdot \widehat{g' * \varphi}(\omega)
\end{aligned}$$

  That is the FOURIER transform of $g * \varphi$ has a zero of order m at 0. From the frequency domain characterisation given above it follows that this function has m vanishing moments.

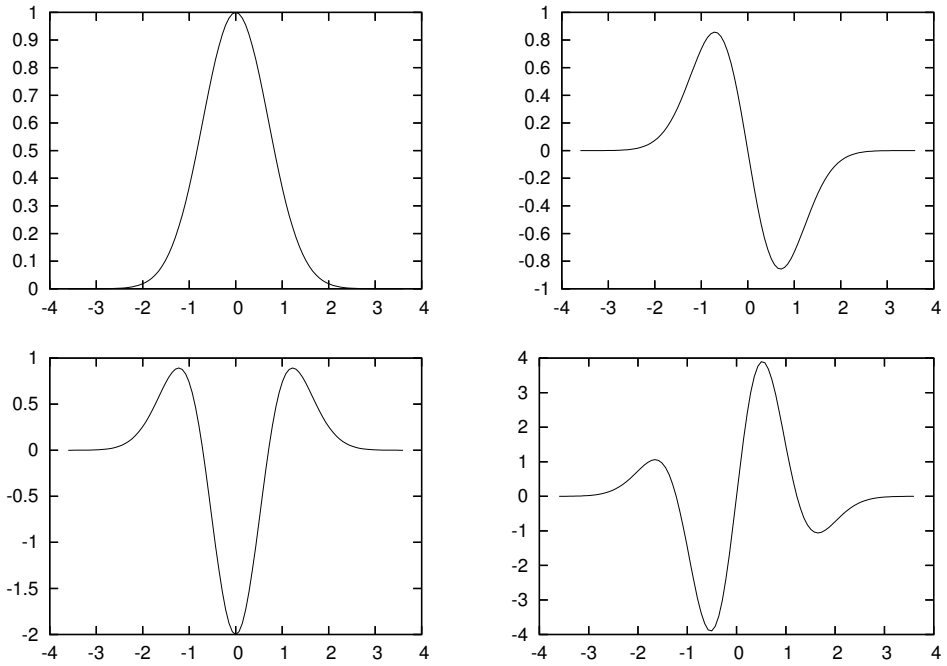


Figure 4.6: Functions with increasing number of vanishing moments: Derivatives of a GAUSSian ($t \mapsto e^{-t^2}$). The GAUSSian has zero vanishing moments and each differentiation adds one vanishing moment. We observe that a higher number of vanishing moments tend to let a function oscillate.

- $2 \cdot (g * \varphi) \downarrow 2$ has m vanishing moments implies that $(\mathbf{1}, -1)^{*m} \mid g$
  Because $\widehat{\varphi}(0) \neq 0$ the function $(\widehat{g} \cdot \widehat{\varphi}) \uparrow 2$ can have an mth order zero at 0 only if $\widehat{g}$ has an mth order zero at 0. Since g is a polynomial an mth order zero can only be caused by the convolutional factor $(\mathbf{1}, -1)^{*m}$.

□

If the analysing wavelet has m vanishing moments then the correlations of the analysing wavelet with polynomials up to degree $m - 1$ are zero. Thus these polynomials are represented by the low-pass band only.

If the discrete input signal x of the wavelet transform is not considered as representation for the continuous function $x * \varphi$ but as a discrete sequence then m vanishing moments guarantee that the low-pass band coefficients represent the discrete polynomial functions up to degree m.

We recall that if h and g are complementary filters then all high-pass filters complementary to h can be represented by the lifting step

$$g_s = g + (s \uparrow 2) * h \quad .$$

All lifted filters constitute an affine space. We will see that the high-pass filters with m vanishing moments form an affine subspace of this space.

Because s can be 0 the un-lifted high-pass filter g belongs to this subspace. Thus it must have m vanishing moments. We are now interested in lifting steps that preserve the vanishing moments of g. In the optimisation target we add the convolution of h with the up-sampled lifting filter s to g. How are the vanishing moment factors distributed over h and $s \uparrow 2$? The low-pass filter h must not have any vanishing moment, otherwise g and h share a common divisor and are not complementary, cf. Corollary 2.2.8. Thus $s \uparrow 2$ must contain all vanishing moment factors.

But the lifting filter must be in the up-sampled form, that is all odd indexed coefficients must be zero. If $(\mathbf{1}, -1)$ is a factor of $s \uparrow 2$ then $(\mathbf{1}, -1)\_$, that is $(\mathbf{1}, 1)$, is a factor of $(s \uparrow 2)\_$. Since $(s \uparrow 2)\_ = s \uparrow 2$,

also $(\mathbf{1}, 1)$ is a factor of $s \uparrow 2$. Thus $(\mathbf{1}, -1) * (\mathbf{1}, 1)$ which equals $(\mathbf{1}, 0, -1)$ is a factor of $s \uparrow 2$ and $(\mathbf{1}, -1)$ is a factor of s. The same applies to powers of $(\mathbf{1}, -1)$.

To get all high-pass filters g_s with m vanishing moments we need an initial high-pass filter g and the lifting step

$$g_s = g + (s \uparrow 2) * (\mathbf{1}, 0, -1)^{*m} * h$$

and the optimisation problem is modified to

$$\underset{c,s}{\operatorname{argmin}} \left\| c \cdot \psi + s * (\mathbf{1}, -1)^{*m} * \varphi - f \right\|_2 \quad .$$

Figure 4.7 shows examples of matching wavelets with vanishing moment constraints.

Uncouple vanishing moments and smoothness

In equation (4.3.1) we have seen that smoothness factors $\frac{1}{2} \cdot (\mathbf{1}, 1)$ of the dual low-pass $\widetilde{h}$ are tightly connected to the vanishing moment factors $\frac{1}{2} \cdot (\mathbf{1}, -1)$ of the primal high-pass g, namely m dual smoothness factors are equivalent to m primal vanishing moments.

But vanishing moments have a strong influence on the shape of a wavelet. If the wavelet is forced to have m_ψ vanishing moments and the pattern has more vanishing moments, say m_f ($m_f > m_\psi$), we have no problems since then the wavelet can get additional moments by the lifting. But if $m_f < m_\psi$ then the match will not be very good, as the Figure 4.7 shows.

It seems that we need a way for decoupling the vanishing moments of the primal wavelet from the smoothness factors of the dual generator. We see three methods to attack this problem: Preprocess the pattern before matching, preprocess the input signal before each wavelet transform, modify the transform itself.

1. Preprocess the pattern in order to increase the number of vanishing moments. Preprocess the input signal in the same way in order to increase the number of vanishing moments of the contained patterns. Let P be a preprocessing which increases the number of vanishing moments by $m_\psi - m_f$ Then we have to choose a wavelet ψ with m_ψ vanishing moments and solve

   $$\underset{c,s}{\operatorname{argmin}} \left(\left\| c \cdot \psi + s * (1, -1)^{*m_\psi} * \varphi - Pf \right\|_2 \right) \quad .$$

   If we want to preprocess the input only once this global preprocessing P must have a comparable effect on each scale, i.e.

   $$\forall k \in \mathbb{Z} \quad \exists c \in \mathbb{R} \quad P(f \uparrow 2^k) = c \cdot Pf \uparrow 2^k \quad .$$

   Thus discrete differences cannot be used. Differentiation fulfils this property and increases the number of vanishing moments but it cannot be applied to discrete data. Even more by matching a differentiated pattern with a wavelet we change the norm used in the optimisation criterion. Differentiation amplifies high frequencies thus the optimisation will result in a matched wavelet where the high frequencies match much better than the low ones. I.e. details or even high frequent noise are more respected than the overall shape of the pattern.
2. Match the pattern with a wavelet where $m_\psi - m_f$ vanishing moments are omitted. That is we choose a complementary filter pair (h, g) and separate some vanishing moment factors from g, i.e. $g = (\mathbf{1}, -1)^{*(m_\psi - m_f)} * g'$. The wavelet ψ' corresponding to g' has only m_f vanishing moments. We use it instead of ψ for the matching.

   $$\psi' = (g' * \varphi) \downarrow 2$$

   $$\underset{c,s}{\operatorname{argmin}} \left(\left\| c \cdot \psi' + s * (1, -1)^{*m_f} * \varphi - f \right\|_2 \right)$$

   When using the matched wavelet in a DWT the omitted vanishing moments are involved in order to achieve perfect reconstructability. The input signal must be preprocessed for obtain matching qualities. This means that the input must be convolved with an inverse w of $(\mathbf{1}, -1)$ such

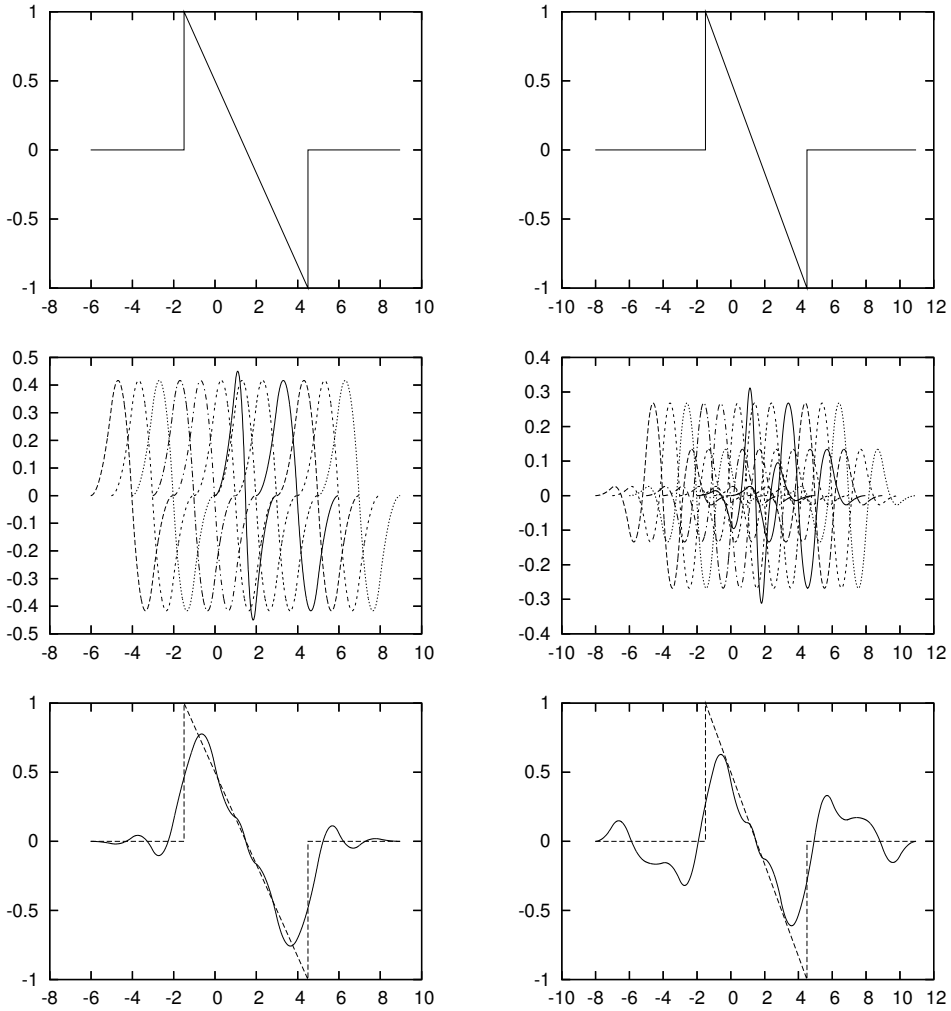


Figure 4.7: Match of wavelets with patterns with vanishing moments: A clipped ramp function is matched with a wavelet associated with the quadratic spline generator (CDF-3). The pattern has one vanishing moment. If the wavelet is constraint with one vanishing moment (left column) the match is quite good. If too much vanishing moments are forced (right column: 5 vanishing moments) the match is not satisfying. (Even more in this example the coefficient c of ψ is rather small, which is numerically bad for reconstruction.)

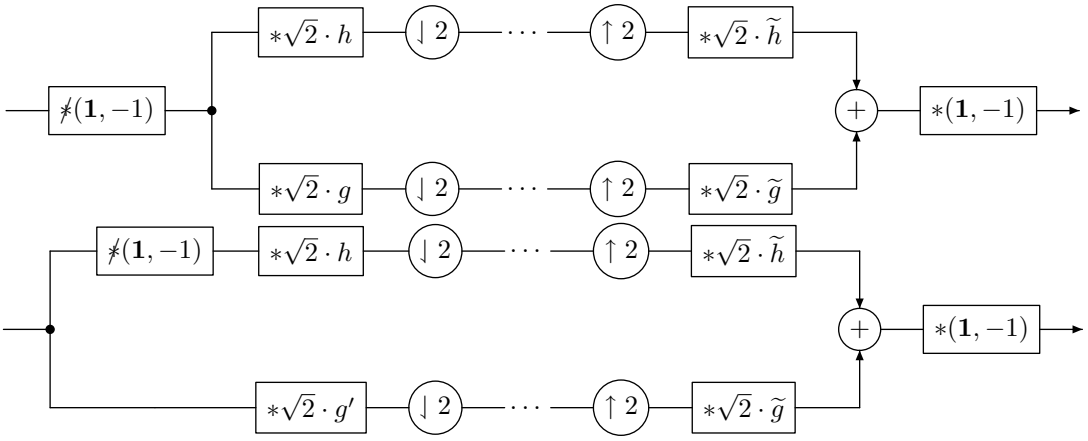


Figure 4.8: Subband coder including a integration pre- and difference post-processing. The first variant uses explicit integration and differentiation whereas in the second variant matching integration and vanishing moment factors are cancelled.

as $(\dots,0,0,\mathbf{1},1,1,\dots)$ or $(\dots,-1,-1,\mathbf{0},0,0,\dots)$. The signal must be preprocessed at each transformation level within the analysis transform and the signal has to be post-processed on the synthesis transform. (Figure 4.8)
Instead of the original DWT

$$\begin{aligned}
\text{Analysis} \qquad \boldsymbol{x}_{j+1} &= \sqrt{2}\cdot\left(\boldsymbol{x}_j * h\right)\downarrow 2\\
\boldsymbol{y}_{j+1} &= \sqrt{2}\cdot\left(\boldsymbol{x}_j * g\right)\downarrow 2\\
\text{Synthesis} \qquad \boldsymbol{x}_j &= \sqrt{2}\cdot\left(\left(\boldsymbol{x}_{j+1}\uparrow 2\right)*\widetilde{h}+\left(\boldsymbol{y}_{j+1}\uparrow 2\right)*\widetilde{g}\right)
\end{aligned}$$

we compute

$$\begin{aligned}
\text{Analysis} \qquad \boldsymbol{x}'_j &= w^{*(m_\psi-m_f)} * \boldsymbol{x}_j\\
\boldsymbol{x}_{j+1} &= \sqrt{2}\cdot\left(\boldsymbol{x}'_j * h\right)\downarrow 2\\
\boldsymbol{y}_{j+1} &= \sqrt{2}\cdot\left(\boldsymbol{x}'_j * g\right)\downarrow 2\\
\text{Synthesis} \qquad \boldsymbol{x}_j &= \sqrt{2}\cdot\left(\left(\boldsymbol{x}_{j+1}\uparrow 2\right)*\widetilde{h}+\left(\boldsymbol{y}_{j+1}\uparrow 2\right)*\widetilde{g}\right)*(\mathbf{1},-1)^{*(m_\psi-m_f)} \quad .
\end{aligned}$$

The integration of $\boldsymbol{x}_j$ and the vanishing moments of g can be merged. This is just the reduction from g to g' which yields $\boldsymbol{x}'_j * g = \boldsymbol{x}_j * g'$. We can formulate the transform by

$$\begin{aligned}
\text{Analysis} \qquad \boldsymbol{x}_{j+1} &= \sqrt{2}\cdot\left(w^{*(m_\psi-m_f)} * \boldsymbol{x}_j * h\right)\downarrow 2\\
\boldsymbol{y}_{j+1} &= \sqrt{2}\cdot\left(\boldsymbol{x}_j * g'\right)\downarrow 2\\
\text{Synthesis} \qquad \boldsymbol{x}_j &= \sqrt{2}\cdot\left(\left(\boldsymbol{x}_{j+1}\uparrow 2\right)*\widetilde{h}+\left(\boldsymbol{y}_{j+1}\uparrow 2\right)*\widetilde{g}\right)*(\mathbf{1},-1)^{*(m_\psi-m_f)} \quad .
\end{aligned}$$

The problem is obviously that the signal must be integrated for the low-pass channel which leads to potentially unlimited signal values. Even more the integration is repeated in each level of the transform.

3. The problem of the previous approach is that the low-pass band contains the integrated data and in each transformation level it is integrated $m_\psi - m_f$ times, again. Can we simply omit the integration? Certainly not since this would prohibit perfect reconstruction. But we notice that the high-pass analysis-synthesis path of the transform discussed in the previous item is quite that of

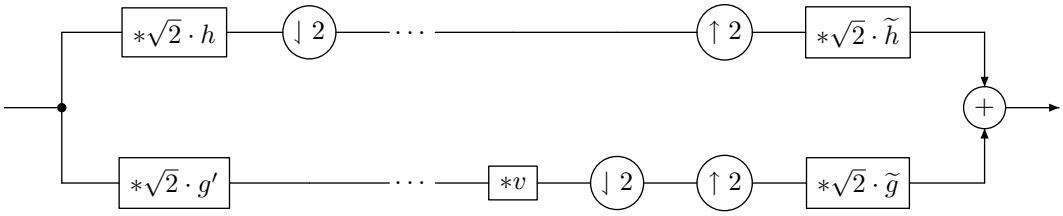


Figure 4.9: A double density subband coder which defers some vanishing moment factors v from the analysis high-pass to the synthesis part. It allows to match patterns with few vanishing moments while the connected generator function has more smoothness factors.

the original DWT, only the convolution with some vanishing moment factors is deferred to the synthesis. We can achieve the same effect by modifying the DWT such that the application of the vanishing moment factors is deferred to the synthesis. Since the down-sampling is applied after the analysis convolution it must be deferred, too. This results in a synthesis step where all odd-indexed signal values are cleared. (Figure 4.9)

$$\begin{aligned}
&\text{Analysis} && \boldsymbol{x}_{j+1} = \sqrt{2}\cdot\left(\boldsymbol{x}_j * h\right)\downarrow 2\\
&&& \boldsymbol{y}_{j+1} = \sqrt{2}\cdot \boldsymbol{x}_j * g'\\
&\text{Synthesis} && \boldsymbol{x}_j = \sqrt{2}\cdot\left(\left(\boldsymbol{x}_{j+1}\uparrow 2\right) * \widetilde{h} + \left(\left(\boldsymbol{y}_{j+1} * (\mathbf{1},-1)^{*(m_\psi - m_f)}\right)\downarrow 2\uparrow 2\right) * \widetilde{g}\right) \quad .
\end{aligned}$$

The optimisation problem for matching the wavelet is the same as in the previous item.

The last of the three suggestions is certainly most promising. It is known to the literature by the term *double-density DWT* [Sel01]. Let us elaborate on it.

4.3.3 Double density transform

The transform differs from the critically sampled DWT in the point that the down-sampling is omitted in the analysis high-pass path, i.e. in the path which is not cascaded. That is why the amount of data of the wavelet transform (only) doubles. The wavelet basis turns into a wavelet frame. Although we chose the discrete wavelet transform in order to avoid redundancy the slight redundancy we get by our modification leads to a slightly more shift-invariant transform.

According to the multichannel transform presented in Section 2.2.5 we can create polyphase matrices for the analysis and the synthesis transform. The analysis transform can be described by the matrix convolution

$$\begin{pmatrix} \boldsymbol{x}_{j+1} \\ \boldsymbol{y}_{j+1}\downarrow 2 \\ \boldsymbol{y}_{j+1}\rightarrow 1\downarrow 2 \end{pmatrix} = \sqrt{2}\cdot \begin{pmatrix} h\downarrow 2 & h\leftarrow 1\downarrow 2 \\ g'\downarrow 2 & g'\leftarrow 1\downarrow 2 \\ g'\rightarrow 1\downarrow 2 & g'\downarrow 2 \end{pmatrix} \circledast \begin{pmatrix} \boldsymbol{x}_j\downarrow 2 \\ \boldsymbol{x}_j\rightarrow 1\downarrow 2 \end{pmatrix} \quad .$$

The polyphase matrix for the synthesis needs some prior reflection.

$$
\begin{aligned}
v &= (\mathbf{1},-1)^{*(m_\psi - m_f)} \\
\frac{1}{\sqrt{2}} \cdot \boldsymbol{x}_j \downarrow 2 &= \Big((\boldsymbol{x}_{j+1} \uparrow 2) * \widetilde{h} + \Big((\boldsymbol{y}_{j+1} * v) \downarrow 2 \uparrow 2 \Big) * \widetilde{g} \Big) \downarrow 2 \\
&= \boldsymbol{x}_{j+1} * (\widetilde{h} \downarrow 2) + \Big((\boldsymbol{y}_{j+1} * v) \downarrow 2 \Big) * (\widetilde{g} \downarrow 2) \qquad | \qquad \text{Lemma 2.2.3} \\
&= \boldsymbol{x}_{j+1} * (\widetilde{h} \downarrow 2) + \Big((\boldsymbol{y}_{j+1} \downarrow 2) * (v \downarrow 2) + (\boldsymbol{y}_{j+1} \rightarrow 1 \downarrow 2) * (v \leftarrow 1 \downarrow 2) \Big) * (\widetilde{g} \downarrow 2) \\
\frac{1}{\sqrt{2}} \cdot \boldsymbol{x}_j \rightarrow 1 \downarrow 2 &= \boldsymbol{x}_{j+1} * (\widetilde{h} \rightarrow 1 \downarrow 2) + \\
&\qquad \Big((\boldsymbol{y}_{j+1} \downarrow 2) * (v \downarrow 2) + (\boldsymbol{y}_{j+1} \rightarrow 1 \downarrow 2) * (v \leftarrow 1 \downarrow 2) \Big) * (\widetilde{g} \rightarrow 1 \downarrow 2)
\end{aligned}
$$

$$
\begin{pmatrix} \boldsymbol{x}_j \downarrow 2 \\ \boldsymbol{x}_j \rightarrow 1 \downarrow 2 \end{pmatrix} = \sqrt{2} \cdot \begin{pmatrix} \widetilde{h}_{\mathsf{e}} & v_{\mathsf{e}} * \widetilde{g}_{\mathsf{e}} & v_{\mathsf{o}} * \widetilde{g}_{\mathsf{e}} \\ \widetilde{h}_{\mathsf{o}} \rightarrow 1 & v_{\mathsf{e}} * \widetilde{g}_{\mathsf{o}} \rightarrow 1 & v_{\mathsf{o}} * \widetilde{g}_{\mathsf{o}} \rightarrow 1 \end{pmatrix} \circledast \begin{pmatrix} \boldsymbol{x}_{j+1} \\ \boldsymbol{y}_{j+1} \downarrow 2 \\ \boldsymbol{y}_{j+1} \rightarrow 1 \downarrow 2 \end{pmatrix}
$$

It is left as an exercise for the reader to check whether the dual polyphase matrix is a left inverse of the primal polyphase matrix (multiply and check for identity matrix). :-)

After having discussed the discrete aspect of the modified transform we want to find out what it means for the continuous interpretation of the transform. A critically sampled transform over n levels is equal to computing the scalar products of the signal with the primal wavelet and generator functions from the set

$$
\left\{ (\varphi^* \rightarrow k) \uparrow 2^n \;:\; k \in \mathbb{Z} \right\} \cup \left\{ (\psi^* \rightarrow k) \uparrow 2^j \;:\; k \in \mathbb{Z} \wedge j \in \{1,\ldots,n\} \right\},
$$

which forms a basis (compare with Theorem 2.2.39). In contrast to that the modified transform computes scalar products with respect to the frame

$$
\left\{ (\varphi^* \rightarrow k) \uparrow 2^n \;:\; k \in \mathbb{Z} \right\} \cup \left\{ (\psi^* \rightarrow k) \uparrow 2^j \;:\; k \in \frac{1}{2} \cdot \mathbb{Z} \wedge j \in \{1,\ldots,n\} \right\} \quad .
$$

That is the lattice of the wavelets is twice as narrow as in the critically sampled transform.

It is a bit more difficult to derive what happens on the dual basis. Let us repeat that the continuous reconstruction of the original DWT at scale j is computed by $\left(\boldsymbol{y}_j \uparrow 2 * \widetilde{g} * \widetilde{\varphi} \right) \uparrow 2^{j-1}$ or shortly $\left(\boldsymbol{y}_j * \widetilde{\psi} \right) \uparrow 2^j$. We obtain the dual basis functions by choosing unit vectors for $\boldsymbol{y}_j$, i.e. wavelet representations where only one coefficient is one while all others are zero. This yields the dual basis

$$
\left\{ (\widetilde{\varphi} \rightarrow k) \uparrow 2^n \;:\; k \in \mathbb{Z} \right\} \cup \left\{ (\widetilde{\psi} \rightarrow k) \uparrow 2^j \;:\; k \in \mathbb{Z} \wedge j \in \{1,\ldots,n\} \right\} \quad .
$$

For the modified DWT the continuous reconstruction is computed by

$$
\left(\left(\boldsymbol{y}_j * (\mathbf{1},-1)^{*(m_\psi - m_f)} \right) \downarrow 2 \uparrow 2 * \widetilde{g} * \widetilde{\varphi} \right) \uparrow 2^{j-1}
$$

or shortly

$$
\left(\left(\boldsymbol{y}_j * (\mathbf{1},-1)^{*(m_\psi - m_f)} \right) \downarrow 2 * \widetilde{\psi} \right) \uparrow 2^j \quad ,
$$

that is the dual frame is

$$
\begin{aligned}
&\left\{ (\widetilde{\varphi} \rightarrow k) \uparrow 2^n \;:\; k \in \mathbb{Z} \right\} \\
&\qquad \cup \left\{ \left((\mathbf{1},-1)^{*(m_\psi - m_f)} \rightarrow k \downarrow 2 * \widetilde{\psi} \right) \uparrow 2^j \;:\; k \in \mathbb{Z} \wedge j \in \{1,\ldots,n\} \right\} \quad .
\end{aligned}
$$

Because of the translation inside the down-sampling we obtain two kinds of wavelets. In one wavelet the even indexed coefficients of the power of the vanishing moment mask are included, in the other one the odd indexed coefficients are. This leads to the frame representation

$$\begin{aligned}\left\{(\widetilde{\varphi} \to k) \uparrow 2^n \;:\; k \in \mathbb{Z}\right\} \\ \cup \left\{\left((\mathbf{1},-1)^{*(m_\psi - m_f)} \downarrow 2 * \widetilde{\psi} \to k\right) \uparrow 2^j \;:\; k \in \mathbb{Z} \wedge j \in \{1,\ldots,n\}\right\} \\ \cup \left\{\left(\left((\mathbf{1},-1)^{*(m_\psi - m_f)} \to 1\right) \downarrow 2 * \widetilde{\psi} \to k\right) \uparrow 2^j \;:\; k \in \mathbb{Z} \wedge j \in \{1,\ldots,n\}\right\} \quad .\end{aligned}$$

Figure 4.10 shows these functions for wavelets matching the sinc function for different orders of smoothness.

Let us recapitulate how we arrived here. We started with the discrete wavelet transform, introduced tools to create wavelets matching patterns while satisfying the requirements of the transform. Now we end up with a modified transform. Are the requirements still the same? They are certainly sufficient but do we have more degrees of freedom now? Should we switch to a lifting scheme which is adapted to the non-square polyphase matrix? These are still open questions.

4.3.4 Conclusion

In this chapter we complemented the matching algorithm from Chapter 3 with smoothness constraints for both primal and dual wavelets. We have seen that the smoothness of a refinable function depends on the number of smoothness factors in the refinement mask and the roughness of the refinable function (or distribution) associated with the remaining mask. The roughness corresponds to the spectral radius of the transition matrix. This in turn can be efficiently estimated by various matrix norms and the sum of powers of the transition eigenvalues. Using these estimates we can setup a regularised optimisation problem which can be solved by some numerical minimiser. We found that the roughness is bounded to below depending on the refinement mask size.

In contrast to the reduction of the spectral radius of the transition matrix, adding smoothness factors to the refinement mask has the stronger effect. But it also increases the number of vanishing moments which is bad for matching patterns with a small number of vanishing moments. A double density transform is a possible way to decouple smoothness factors and vanishing moments at the expense of a redundant transform.

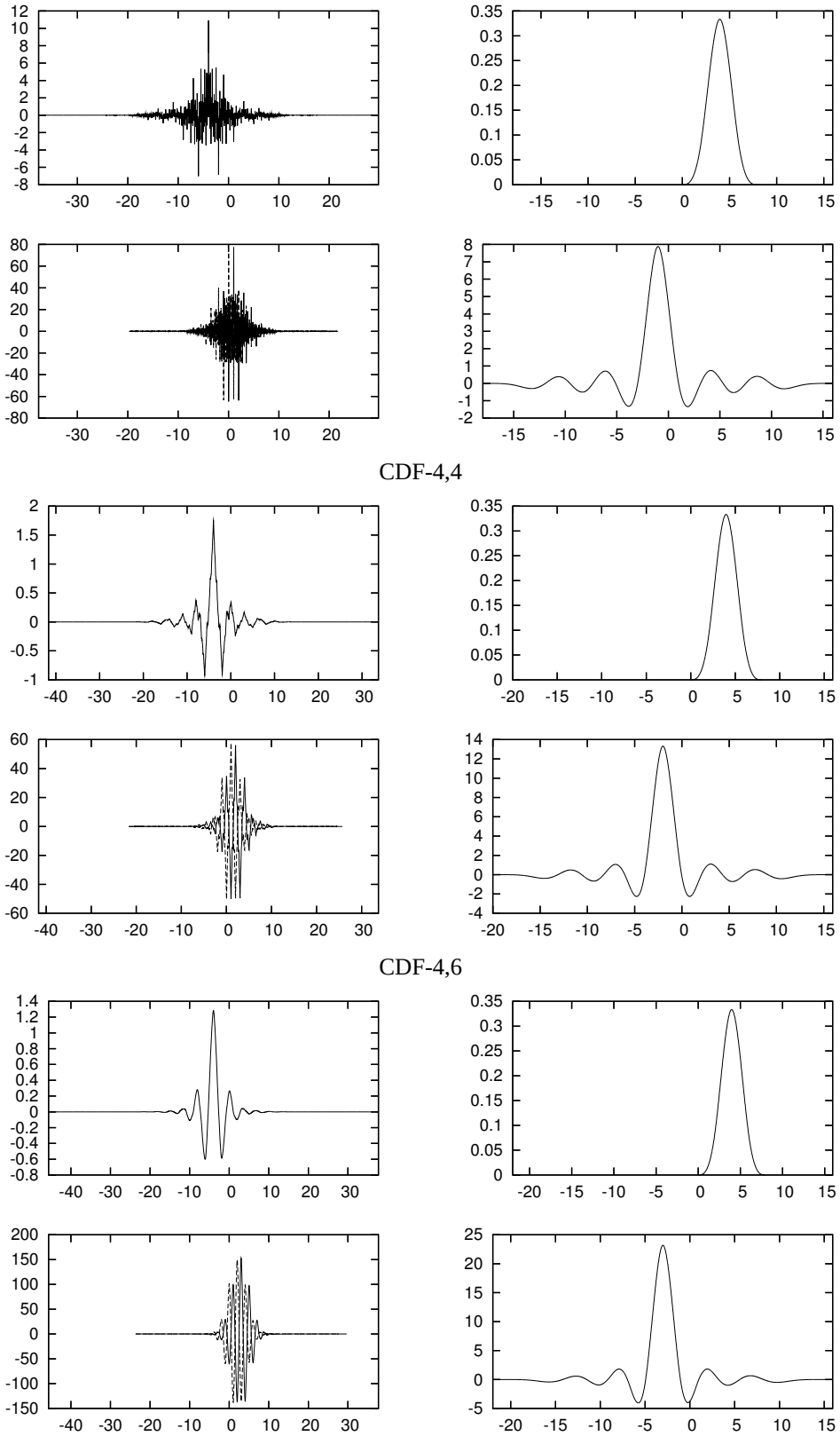


Figure 4.10: Match the sinc function with a CDF-$4, m$ wavelet. The sinc function has no vanishing moment, so the wavelet has none, too. The CDF label denotes the number of smoothness factors of the respecting generator.

Chapter 5

Application of matched wavelets

In this chapter we want to give an overview over an implementation of the methods derived in Chapter 3 and Chapter 4. Furthermore we will perform some tests based on real world data.

5.1 Software library in Modula-3

All methods we developed in this work are implemented in a Modula-3 library. Modula-3 is a successor of Modula-2 and Pascal and somehow a sibling of Oberon [DO]. It was chosen because – as the name suggests – it allows clean and safe modularisation, it is very expressive due to exceptions and a sophisticated range of types, it is very safe because of static type checking and a garbage collector, but it is still a system programming language and allows efficient machine oriented programming. The library can be downloaded from the author's homepage `http://research.henning-thielemann.de/`.

We give an overview over the modules of the library according to the used directory structure. Almost all modules are generic modules which can be instantiated for different numeric types such as floating point numbers of various precisions and complex numbers.

- `signal`
  - `Signal`
    Implementation of a discrete signal from $\ell_0(\mathbb{Z})$, a signal consists of an array containing the sampled data and the index, the leading element of the array is associated with, i.e. in terms of Definition 3.1.5 the minimal element of the domain
  - `SignalFmtLex`
    Conversion between a discrete signal and its text representation
  - `ScaledSignal`
    A discrete signal with a sample rate, i.e. a sequence from $\ell_0(c \cdot \mathbb{Z})$ with $c \in \mathbb{R}$
  - `Convolution`
    The convolution of discrete signals, implemented both naively and using the FFT [FJ]
- `wavelet`
  - `continuous`
    - `transform`
      - `ContinuousWaveletTransform`
        Data type for the result of the discretised continuous wavelet transform (see Section 2.2.1)
      - `ContinuousWaveletAnalysis`
        Compute wavelet coefficients for a signal
      - `ContinuousWaveletSynthesis`
        Restore signal from wavelet coefficients, the reconstruction will not be perfect even in the absence of rounding errors
  - `discrete`
    - `basis`

- `FilterBank`
  Conversion between filter banks in the presence of down-sampling and polyphase filter banks
- `DyadicFilterBank`
  Conversion between primal and dual filters, low-pass and high-pass, both for biorthogonal and orthogonal filter banks
- `BSplineWavelet`
  Generate filter banks for the biorthogonal CDF family
- `DaubechiesWavelet`
  Generate filter banks for the orthogonal DAUBECHIES family
- `WaveletPlot`
  Plot wavelet functions associated with a filter bank [LF05]

* `refinable`
  - `RefinableFunction`
    Generation of transition matrices, the cascade algorithm
  - `RefinableSmooth`
    Estimates of the smoothness of a refinable function
* `transform`
  - `DiscreteWaveletTransform`
    Discrete wavelet analysis and synthesis for any number of channels
  - `DyadicDiscreteWaveletTransform`
    The classic discrete wavelet transform for two channels, including translation invariant versions, see Section 2.2.3 and Section 2.2.2
  - `DWTPlot`
    Plot wavelet coefficients of multiple levels [LF05]
* `match`
  - `WaveletMatch`
    Match a wavelet with a pattern without smoothness constraints, construction of matrices for normal equations, use of LAPACK for solving the simultaneous linear equations [ABB+99]
  - `WaveletMatchBasis`
    Basis type for matching with smoothness constraints
  - `WaveletMatchGradientLift`
    Compute the derivatives of both parts of the target functional (namely the $\mathcal{L}_2(\mathbb{R})$ distance of pattern and wavelet, and the smoothness penalty term) with respect to the lifting filter
  - `WaveletMatchGradient`
    Put together derivatives of both parts of the target functional to a total derivative. A coefficient for ψ can be computed but it can also be fixed to a value by the caller.
  - `WaveletMatchSmooth`
    The main matching procedure supporting vanishing moment and smoothness constraints, visualisation of the optimisation procedure

Let us now go into details of the implementation of the regularised optimisation. The simple optimisation problem

$$\operatorname*{argmin}_{c,s} \|c \cdot \psi + s * \varphi - f\|_2$$

is solved in `WaveletMatch` with LAPACK's `GELS` routine. This can either be done in the direct way or using the optimised construction of the normal equations from Section 3.3.4. An optimal solution respecting smoothness constraints, i.e.

$$\operatorname*{argmin}_{c,s} \left(\left\| c \cdot \psi' + s * (1,-1)^{*m_f} * \varphi - f \right\|_2^2 + \lambda \cdot \sigma(c,s) \right)$$

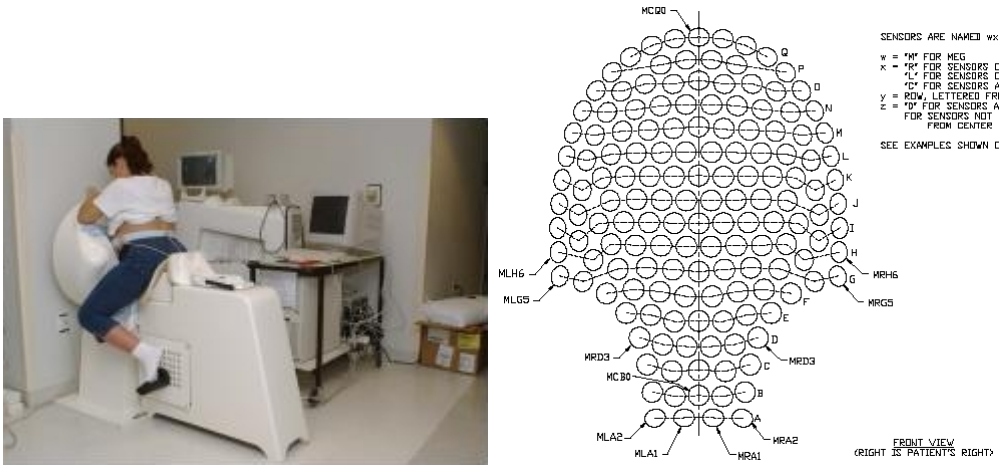


Figure 5.1: Superconducting quantum interference device: The device for measuring fetal magneto-encephalograms (fMEG) and the geometry of the array of 151 sensors.

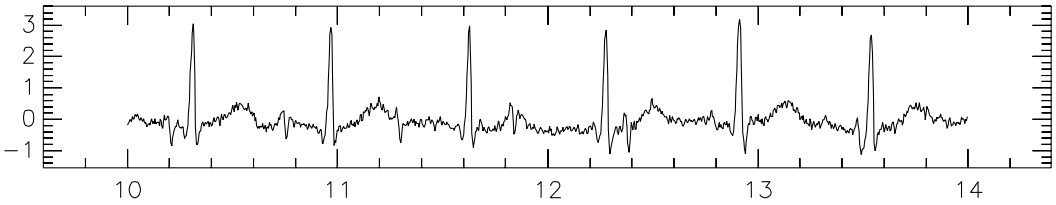


Figure 5.2: A signal measured by one of the sensors. It is a superposition of mMCG, fMCG, fMEG, uterine smooth muscle signal and noise

is approximated iteratively with NEWTON's method in the module `WaveletMatchSmooth`. That is, we search for s and c where the gradient of the functional is zero (or at least small). The parameter λ controls the regularisation. If $\lambda = 0$ we fall back to a linear least squares problem where one iteration step solves the equation exactly. If $\lambda \neq 0$ then the iteration becomes useful. Unfortunately the computation of $\sigma(c, s)$ contains a division by c which causes instabilities. Therefore the regularisation parameter λ is increased successively, where for each λ some NEWTON iterations are performed. The algorithm starts with $\lambda = 0$ (or close to zero) which let the wavelet match optimally but ignores smoothness. Then λ is increased to the wanted value. Too large values for λ lead to heavy numerical difficulties.

5.2 Analysis of SQUID data

Now we want to test our method with real world data. It is difficult to find a problem our method can be applied to without major modifications. Natural data usually does not have a strict dyadic multi-scale structure that the discrete wavelet transform requires. However we will use a pattern cancellation problem in order to demonstrate the steps of the application of our method, the difficulties and the perspectives.

Our cooperation partner HUBERT PREISSL from the Universität Tübingen deals with prenatal diagnostics [VRM+04]. He makes use of a *superconducting quantum interference device* (short *SQUID*, see Figure 5.1) which is able to detect biological activities of the fetus. This device has 151 sensors, each recording a signal at 250 Hz.

A signal of each sensor as in Figure 5.2 is a superposition of

- the fetal magneto-encephalogram (fMEG),
- the fetal magneto-cardiogram (fMCG),
- the maternal magneto-cardiogram (mMCG),

- the uterine smooth muscle signal (magneto-myogram),
- motion artifacts, and noise.

The magneto-cardiograms are quite periodic signals with rather constant wave shapes. However the periods of both components differ. The uterine smooth muscle has a low frequency (at most 5 Hz). We are interested in the signal of the uterine smooth muscle signal and the fetal magneto-encephalogram, both are dominated by the magneto-cardiograms.

The uterine smooth muscle signal can be separated easily by a low-pass. We subtract that signal from the composite signal. In other words we apply a high-pass.

We can now try to remove the magneto-cardiograms by identifying their waveforms, match a wavelet with them, transform the signal with respect to the matched wavelet, clear the coefficients associated with the occurrences of heart beats and transform the signal back.

In [VRM+04] the data of all channels is used to filter out the magneto-cardiograms. Of course this leads to more reliable results but for simplicity we want to cope only with single signals here.

To be able to match the wavelet we need a prototype pattern. But our data contains two cardiograms and noise. We could try to match the wavelet to many appearances of the maternal heart beat. It is reasonable to match a wavelet with an average of multiple waves because this leads to the same result like matching simultaneously with several patterns. We can verify this by comparing the zeros of the derivatives of a general linear least squares problem. Here x stands for the lifting parameters, b_i for the ith occurrence of the pattern, and A for the matrix of the operator mapping lifting parameters to lifted wavelets.

$$\begin{aligned} D\left(x \mapsto \sum_{i=1}^{n} \|A \cdot x - b_i\|_2^2\right) &= x \mapsto \sum_{i=1}^{n} 2 \cdot A^* \cdot (A \cdot x - b_i) \\ &= x \mapsto 2 \cdot A^* \cdot \left(n \cdot A \cdot x - \sum_{i=1}^{n} b_i\right) \\ &= x \mapsto n \cdot 2 \cdot A^* \cdot \left(A \cdot x - \frac{1}{n} \cdot \sum_{i=1}^{n} b_i\right) \\ &= n \cdot D\left(x \mapsto \left\| A \cdot x - \frac{1}{n} \cdot \sum_{i=1}^{n} b_i \right\|_2^2\right) \end{aligned}$$

The next step is to retrieve several patterns from a signal x. This is done by tracking the period of the cardiogram. An initial value of the period is retrieved from the autocorrelation $x * x^*$ of a clip at the beginning of the signal. We choose the position of the first peak after 0 as the initial period t. Then for each peak we look ahead t samples and search for the maximum value in an environment of this position. This is considered as the next peak and the period is adapted accordingly. This procedure works quite well for many signals. It could even be improved using more advanced methods which are known as *pitch detection* and *pitch tracking* in the area of audio signal processing [Zöl02].

All found instances of waves can now simply be averaged. This leads to very smooth noise-free shapes as shown in Figure 5.4. This raises the question whether the signal can be de-noised by permanent averaging. Indeed there is such a method called *comb filter*. The name is due to the comb structure of its transfer function. [Zöl02] It is a recursive filter and thus it is computed by a recursion formula which is equivalent to a power series division. If x is the input signal and y the comb filtered signal with a delay of t and a gain of c then it holds

$$\begin{aligned} y = (1-c) \cdot x + c \cdot y \to t \\ y - c \cdot y \to t = (1-c) \cdot x \\ y = (1-c) \cdot x \not\leftarrow (\delta - c \cdot \delta \to t) \end{aligned}$$

or written element-wise

$$\forall j \in \mathbb{Z} \qquad y_j = (1-c) \cdot x_j + c \cdot y_{j-t} \quad .$$

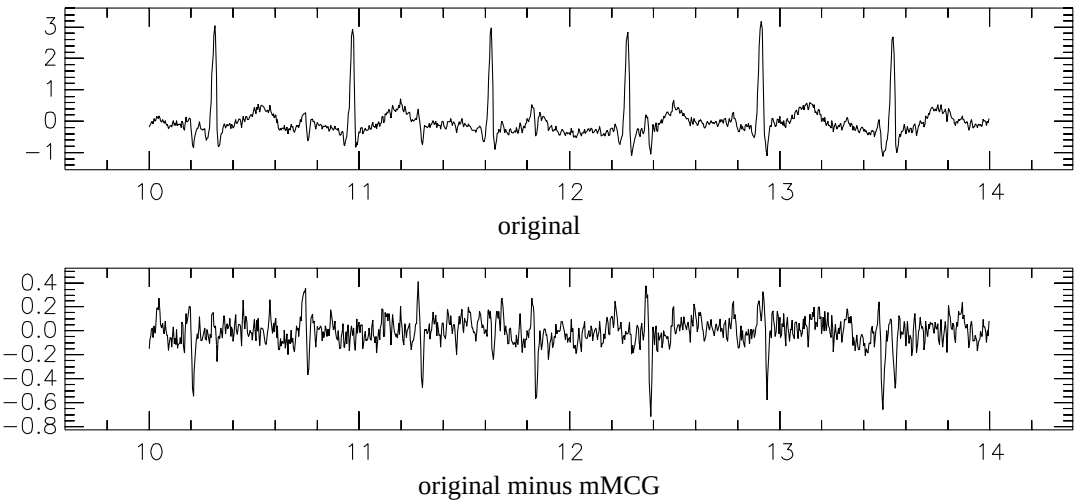


Figure 5.3: SQUID signal: mMCG cancelled with comb filter

The gain should be from the interval $(0, 1)$. If it is close to 1 then the smoothing effect dominates but the filter can react only slowly to changes in the shape.

A modulated comb filter can additionally handle varying periods. Instead of a fixed delay t we use a sequence t of the current periods which must be interpolated from the period information we collected at the pitch tracking stage. If two subsequent maxima are at j and k then it should be $t_k = k - j$. The modulated comb filter is computed by

$$\forall j \in \mathbb{Z} \qquad y_j = (1 - c) \cdot x_j + c \cdot y_{j - \lfloor t_j \rfloor} \quad .$$

To reduce resampling noise we interpolate the feedback data. Linear interpolation leads to

$$\begin{aligned} &\operatorname{convexcomb}(a, b, \lambda) = (1 - \lambda) \cdot a + \lambda \cdot b \\ &\forall j \in \mathbb{Z} \quad y_j = (1 - c) \cdot x_j + c \cdot \operatorname{convexcomb}\left(y_{j - \lfloor t_j \rfloor}, y_{j - \lfloor t_j \rfloor - 1}, t_j - \lfloor t_j \rfloor\right) \quad . \end{aligned}$$

The modulated comb filter results in a cleaned cardiogram which can be subtracted from the composite signal. The result can be seen in Figure 5.3. The comb filter needs some time to adapt to the signal. Therefore we show the result after 10 seconds. Luckily, the mMCG peaks have quite a constant shape so the method works well. It can even discover small peaks superposed by large peaks. This can be checked when assuming that both cardiograms are almost periodic. Indeed what remains after cancellation of the mMCG is the magneto-cardiogram of the fetus (fMCG), the fetal magneto-encephalogram (if present) and noise. With the same technique the remaining signal could be freed from the fMCG. We assume that the results would be even better if all channels of the SQUID are processed.

We will continue with trying to achieve the same with a wavelet transform. The question arises whether a matched wavelet performs better than a standard wavelet. So there is certainly more to explore but for now we want to content with treating problems and chances of matched wavelets. We match the averaged cardiogram wave with a wavelet complementary to a cubic B-Spline as generator. (Figure 5.4) This ensures a smooth wavelet. The shape of the wave consists of a steep part at the beginning and a flat peak at the end. The steep part forces us to match a wavelet at a small scale. In our example we have only one level of refinement. According to the counting in Section 2.2.2 we will call that scale number 2. The flat peak at the right forces us to reserve 20 coefficients for the lifting filter. This is an unfortunate situation because the main advantage of the DWT is its speed. Long filters undermine that.

If we do not defer vanishing moments (see Section 4.3.2) the transform can be executed and inverted but the wavelet coefficients grow considerably on every level. Small modifications like clearing single coefficients destroy the signal on reconstruction. The reduction of the spectral radius of the transition

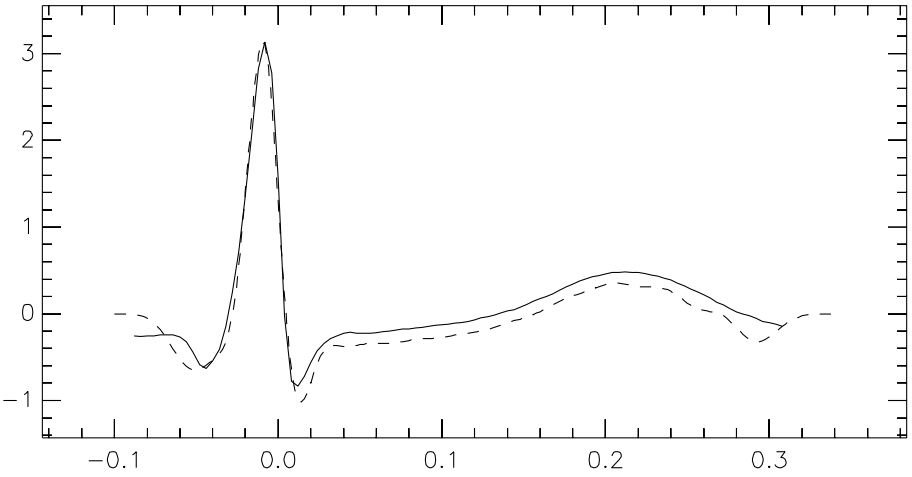


Figure 5.4: Matched wavelet with maternal cardiogram: The continuous line is the averaged waveform from the SQUID data. The dashed line is a wavelet matched with respect to the CDF-4,8 basis where 6 vanishing moments are deferred. That is, the wavelet has 2 vanishing moments but the opposite generator has 8 smoothness factors. So this is literally a *mother wavelet*.

matrix cannot remedy that problem. That is, we must apply our modified transform and we must match the wavelet with the optimisation algorithm that is adapted accordingly.

We have two alternatives: We can use the pattern matched wavelet for analysis or synthesis. In the first case we expect peaks at every position of a wave of the maternal magneto-cardiogram. That is, we should be able to detect these peaks. But the waves of the mMCG are so dominant that we do not need a transformation at all in order to detect them. In the second case the pattern matched wavelet occurs in the reconstructed signal – or not if the corresponding coefficient is cancelled. This variant seems to be the appropriate for our application.

In a first test we want to check the quality of pattern detection of our method. To this end we transform the extracted peak that was already used for matching. We execute the transform by simple convolution, storing all non-zero values, i.e. we do not clip the wavelet coefficients sequences to the length of the input signal. This is the easiest way to assert perfect reconstruction. This method means essentially zero padding. It implies that we should ignore coefficients close to the start and the end of the coefficient sequences because these coefficients depend on the padded zeros rather than measured data.

The low-pass filters are normalised to an EUCLIDean norm of $\sqrt{2}$. With this normalisation all generators have the same $\mathcal{L}_2(\mathbb{R})$ norm which is then true for the wavelets, too. Only with this normalisation wavelet coefficients of different scales can be compared.

According to the labelling introduced in Section 2.2.2 we denote the input signal with $\boldsymbol{x}_0$, the generator coefficients sequence of the largest scale $\boldsymbol{x}_n$, and the wavelet coefficients sequences with $\boldsymbol{y}_j$, where $j \in \{1, \ldots, n\}$. Because we matched the pattern in scale 2 at position 0 we expect that the wavelet transform has an exceptionally big coefficient $\boldsymbol{y}_2$.

In Figure 5.5 we have decomposed our peak prototype into five levels using correlation with the matched wavelet. This corresponds to the first case. Since the transform is based on convolution rather than correlation we have to flip all filters before the transformation. We observe that the sequence $\boldsymbol{y}_2$ is quite symmetric. The reason is that apart from down-sampling this signal is a convolution of the pattern and the flipped matched wavelet, so almost an autocorrelation function. Indeed the coefficient at position 0 on scale 2 is significantly larger than others of that scale. It is also the overall largest wavelet coefficient but not so significant. On the one hand this will make the detection of the particular scale of a pattern more difficult. On the other hand it weakens the influence of the strict dyadic scale graduation. For our application the scale dependence is only important in so far as we want to detect the pattern at a specific scale and nowhere else. Further tests show that the detection is also robust with respect to translations of the signal.

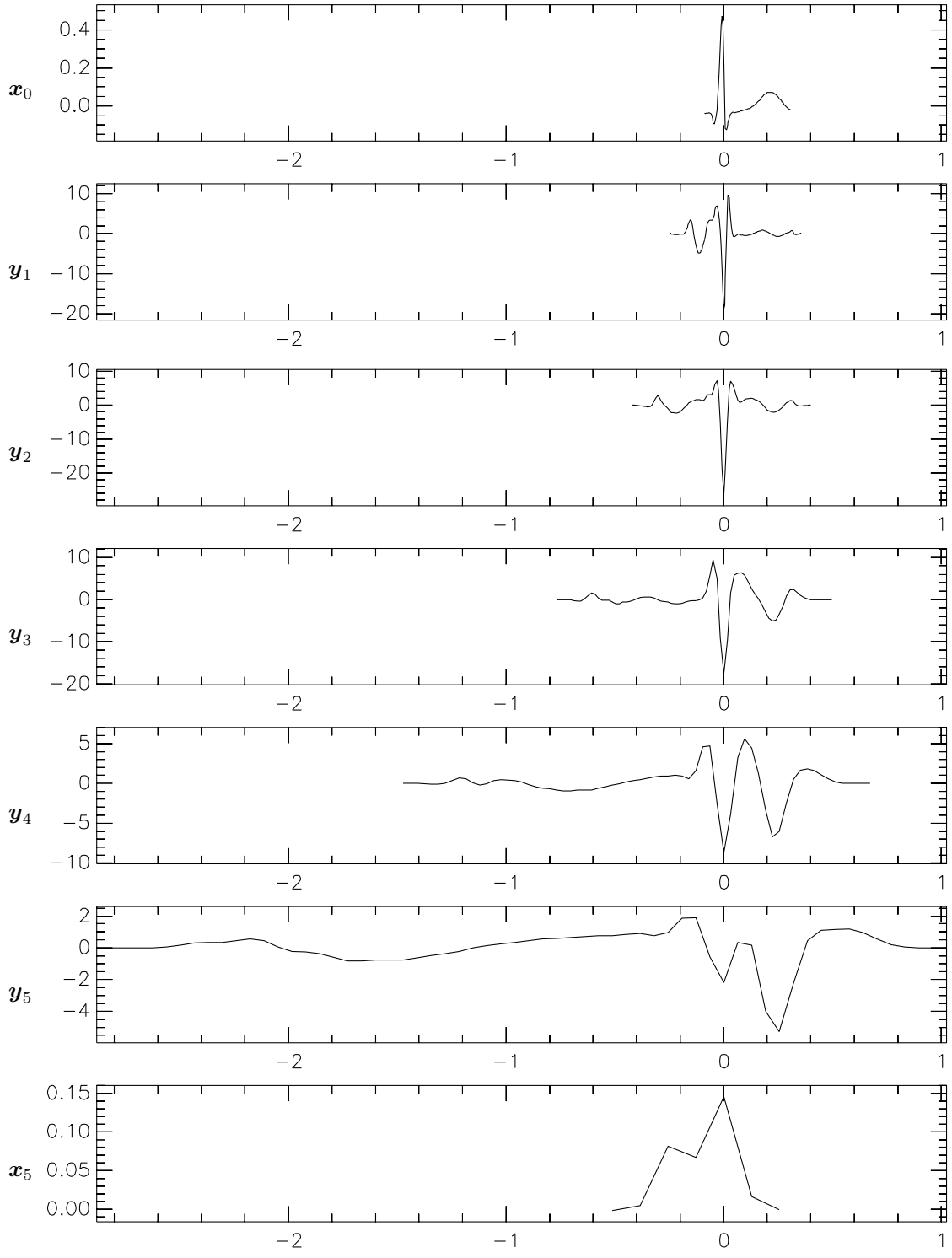


Figure 5.5: Single peak: Discrete wavelet analysis with matched wavelet and deferred vanishing moments. Only wavelet coefficients are shown. The coefficients are plot with respect to a dyadic grid. The geometry would be better reflected if the positions are interleaved but this makes reading abscissa values more complicated. x_0 is the analysed signal. The transform is based on simple convolutions without clipping, i.e. we apply zero padding. Clipping is omitted for the sake of perfect reconstruction.

In Figure 5.6 we have decomposed our peak pattern with the discrete wavelet analysis including clearing of odd indexed coefficients and subsequent application of the vanishing moments. This data can be re-synthesised with the matched wavelet, that is, each wavelet coefficient is mapped to the approximated pattern. The effect of filtering with the vanishing moment mask is obvious: Oscillations as fast as the sampling allows. The wavelet coefficient at position 0 on scale 2 is no longer the largest. At least the largest coefficients are concentrated around position 0 on each scale.

The roughness of the wavelet coefficients sequences raises the question how sensitive the transform is with respect to translations. For the test of shift sensitivity we transform the peak shifted by one to the right. If the signal is shifted by multiples of 2 then scale 2 is shifted by the half distance. This implies that only odd translations make essential differences. In Figure 5.7 the transform is depicted. The shift has visible influence on the result. However the (weak) concentration of coefficients in time remains.

The next task we want to tackle is the transform of a real SQUID signal. Figure 5.8 and Figure 5.9 show the results. For cancellation of the mMCG signal it should suffice to transform until scale 2 because we would not touch higher scales. Nonetheless we observe that the wavelet coefficients do not contain low frequent components. This is of course because the wavelet coefficients are results of high-pass filtering. We can consider the low-pass band $\boldsymbol{x}_5$ as an approximation to the uterine muscle signal. The question remains whether that signal could be better extracted and cancelled by a simple (shift insensitive) low-pass filter.

When correlating the signal with the matched wavelet (Figure 5.8) we clearly see the occurrence of each peak of the mMCG at the scales from 1 to 3. But as already said, the detection of the pattern is not the problem here since we can find it even without any transformation. In contrast to this plot in Figure 5.9 there is no consistent correspondence between the pattern in the data and significant coefficients in scale 2. This correspondence is better in scale 3 but still inhomogeneous.

The next question is: How to process the wavelet coefficients in order to extract or cancel the mMCG? Our method would work perfectly if $\boldsymbol{y}_2$ represents the mMCG. This should apply at least to the transform which uses the matched wavelet for the reconstruction. Figure 5.10 and Figure 5.11 show what each level of the transform represents. These signals are obtained when all but one levels are cleared and the remaining coefficients are transformed back to the spatial domain. We clearly see that our hope does not come true.

The problems are certainly also due to the redundancy of the transform. Redundancy means that even if it is possible to represent the mMCG only with $\boldsymbol{y}_2$ there is no need to do so. It is also possible to represent the mMCG with other coefficients. We do not know how to force the transform to use only $\boldsymbol{y}_2$ for representing the mMCG.

If our assumptions about the result of the transform were true we could simply clear $\boldsymbol{y}_2$ in order to eliminate the mMCG. But now we can only apply some heuristics. For de-noising thresholding algorithms are quite popular. These algorithms apply a FOURIER transform or a wavelet transform. It is assumed that noise is present in all coefficients homogeneously but weak whereas important information is contained in large coefficients. Because of this all coefficients are modified by a shrinkage function with respect to a shrinking parameter λ. Popular instances are *soft shrinkage* and *hard shrinkage*. But interim types are also possible. [Lor05] We want to continue with the soft shrinkage. Roughly spoken the absolute value of each coefficient c is reduced by λ and coefficients with an absolute value smaller than λ vanish.

$$S_\lambda(c) = \begin{cases} c - \lambda & : c > \lambda \\ 0 & : |c| \leq \lambda \\ c + \lambda & : c < -\lambda \end{cases}$$

In our application the significant coefficients shall be caused by the pattern to be removed. Ideally, with soft shrinkage we could extract the mMCG which can then be eliminated by subtraction. We obtain

$$\begin{aligned} c - S_\lambda(c) &= \min\left\{\lambda, \max\left\{-\lambda, c\right\}\right\} \\ &= \begin{cases} \lambda & : c > \lambda \\ c & : |c| \leq \lambda \\ -\lambda & : c < -\lambda \end{cases} . \end{aligned}$$

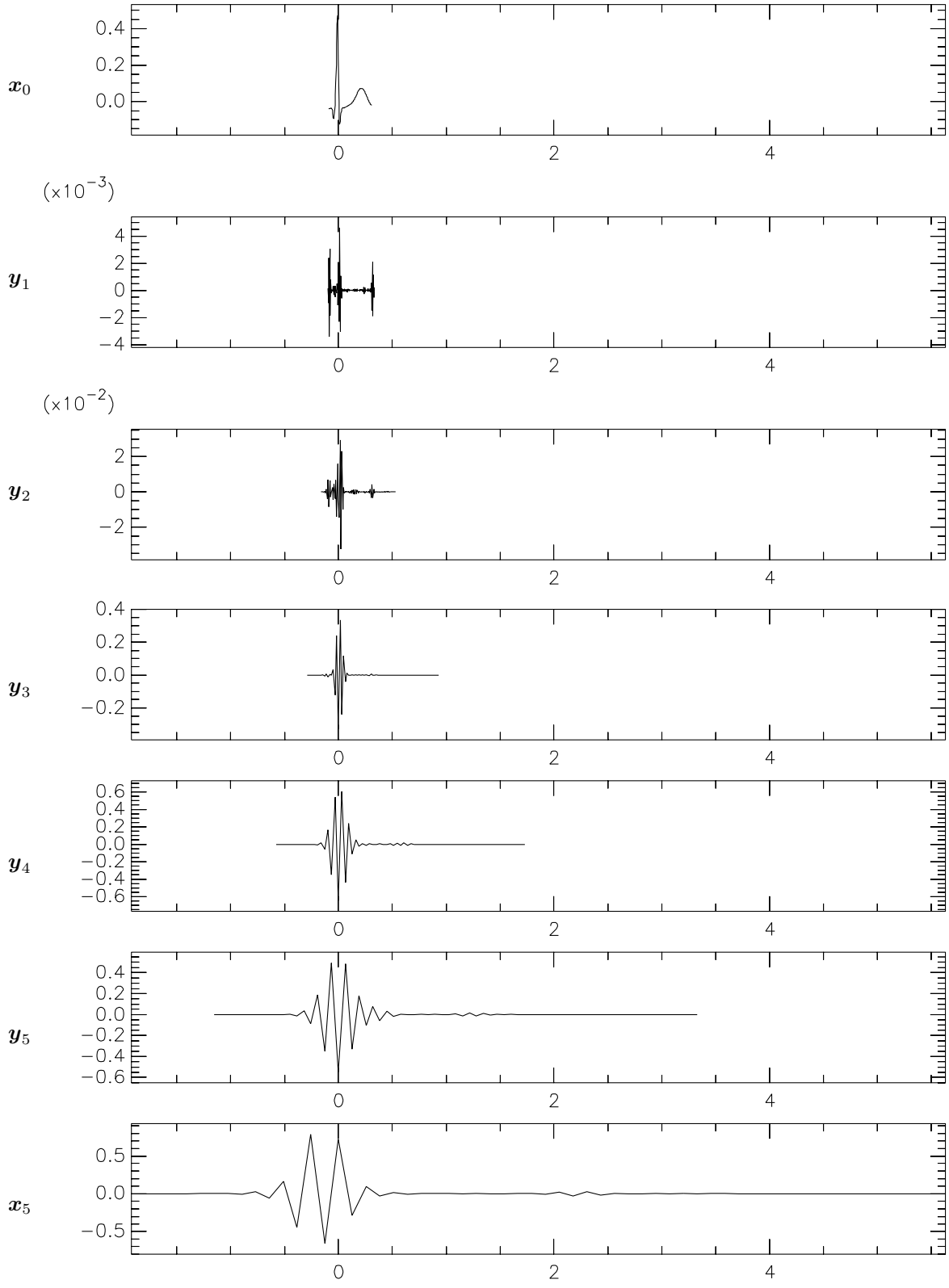


Figure 5.6: Single peak: Discrete wavelet analysis with prefetched vanishing moments. The corresponding synthesis invokes the matched wavelet. Only wavelet coefficients are shown. $\boldsymbol{x}_0$ is the analysed signal.

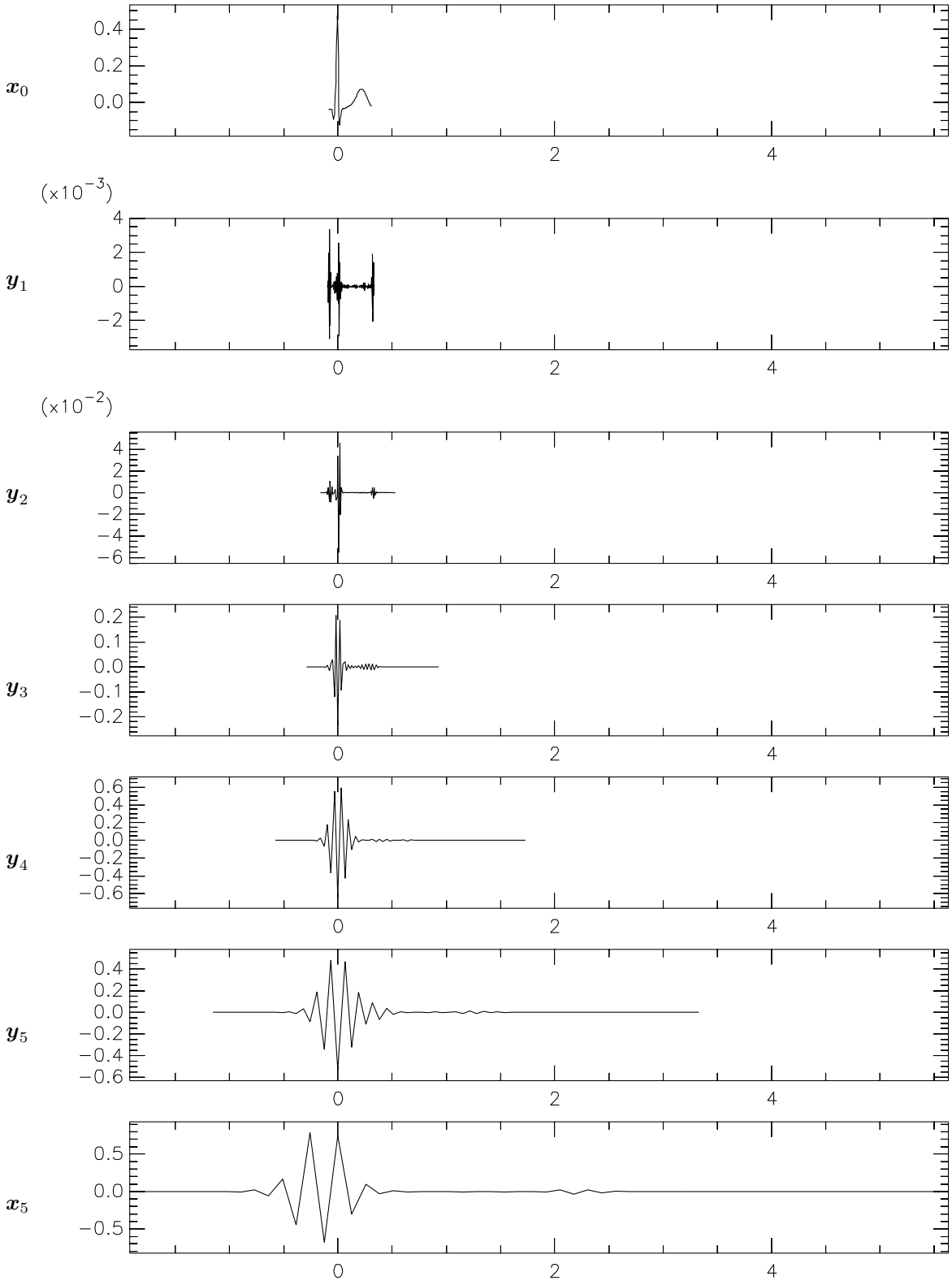


Figure 5.7: Single shifted peak: Discrete wavelet analysis with prefetched vanishing moments.

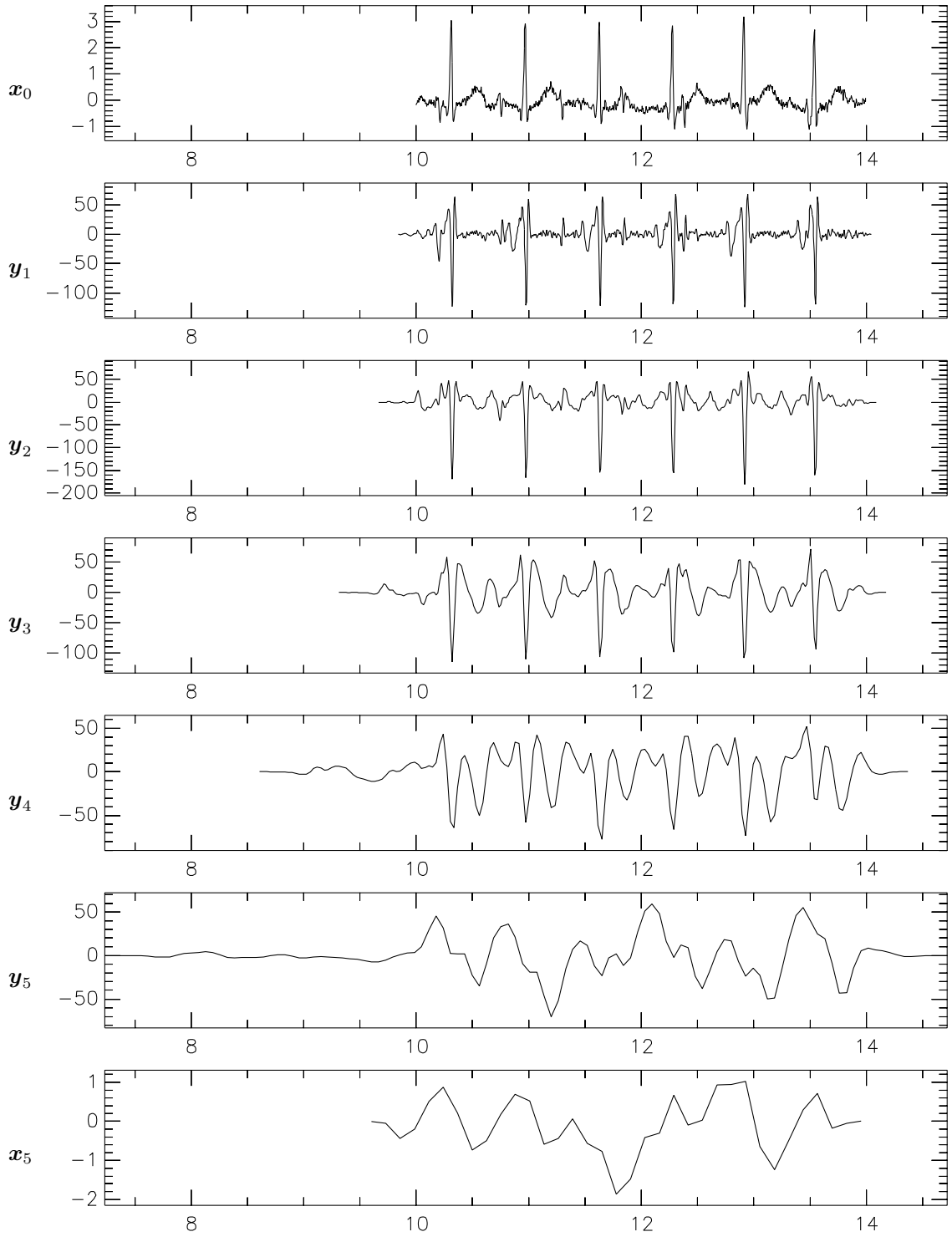


Figure 5.8: SQUID signal: Discrete wavelet analysis with matched wavelet and deferred vanishing moments. Only wavelet coefficients are shown. x_0 is the analysed signal.

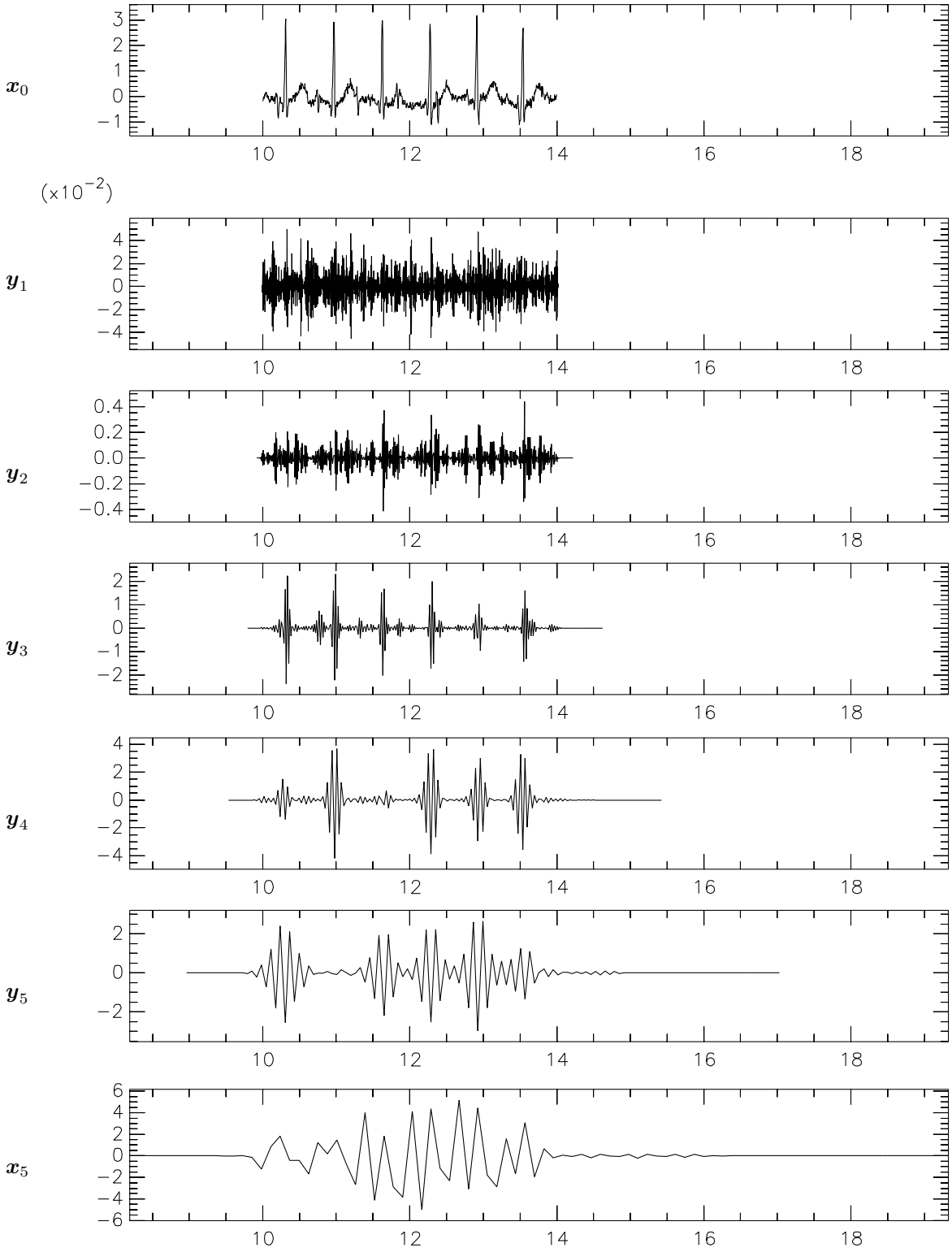


Figure 5.9: SQUID signal: Discrete wavelet analysis with prefetched vanishing moments. The corresponding synthesis invokes the matched wavelet. Only wavelet coefficients are shown. $\boldsymbol{x}_0$ is the analysed signal.

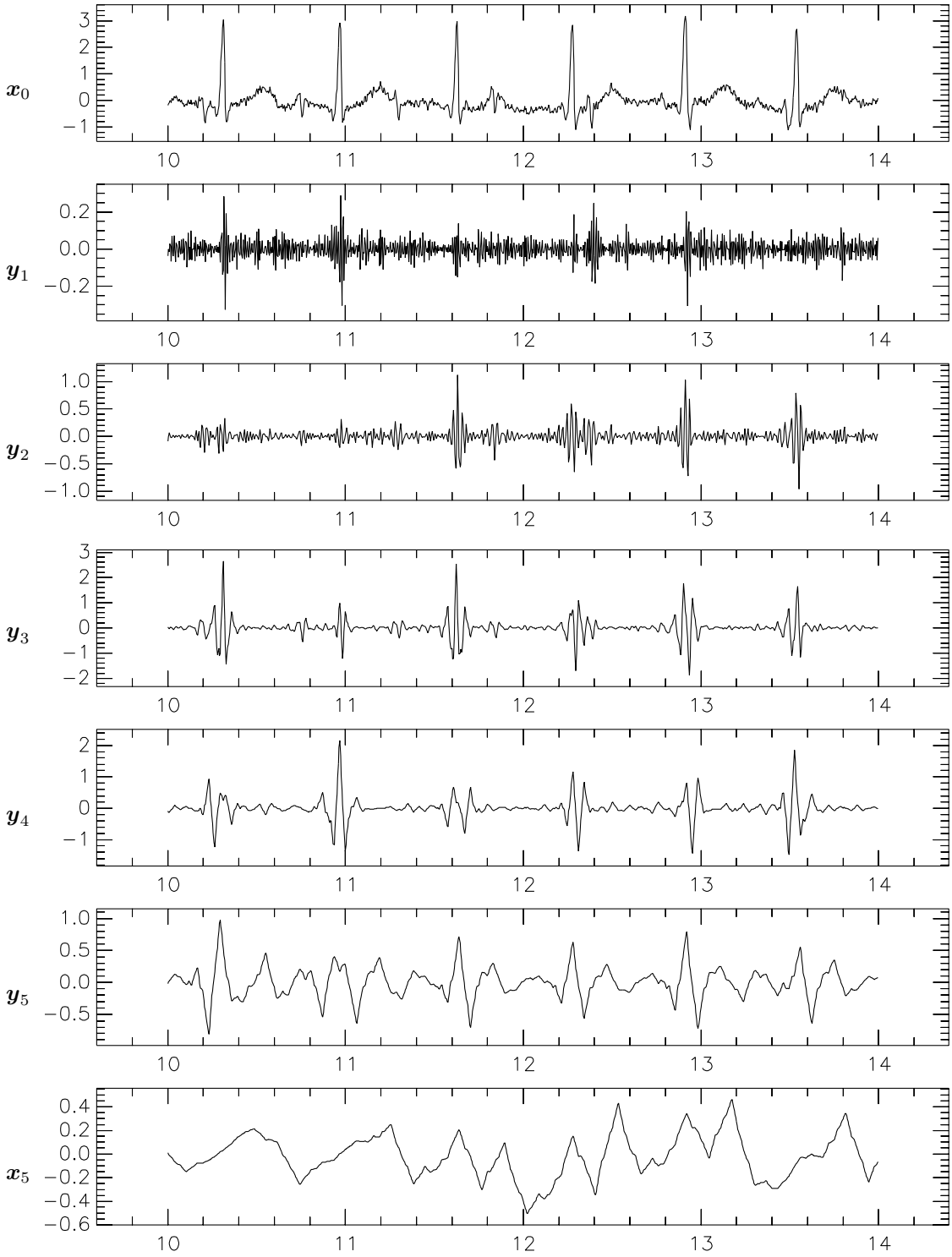


Figure 5.10: SQUID signal: Discrete wavelet analysis with matched wavelet and deferred vanishing moments. The signal components at each scale are shown. That is, for $\boldsymbol{x}_j$ we plot $\left(\boldsymbol{x}_j * \widetilde{\varphi}\right) \uparrow 2^j$ and for $\boldsymbol{y}_j$ we plot $\left(\left(\boldsymbol{y}_j * (\mathbf{1}, -1)^{*(m_\psi - m_f)}\right) \downarrow 2 * \widetilde{\psi}\right) \uparrow 2^j$. Cf. Section 4.3.3. $\boldsymbol{x}_0$ is the analysed signal.

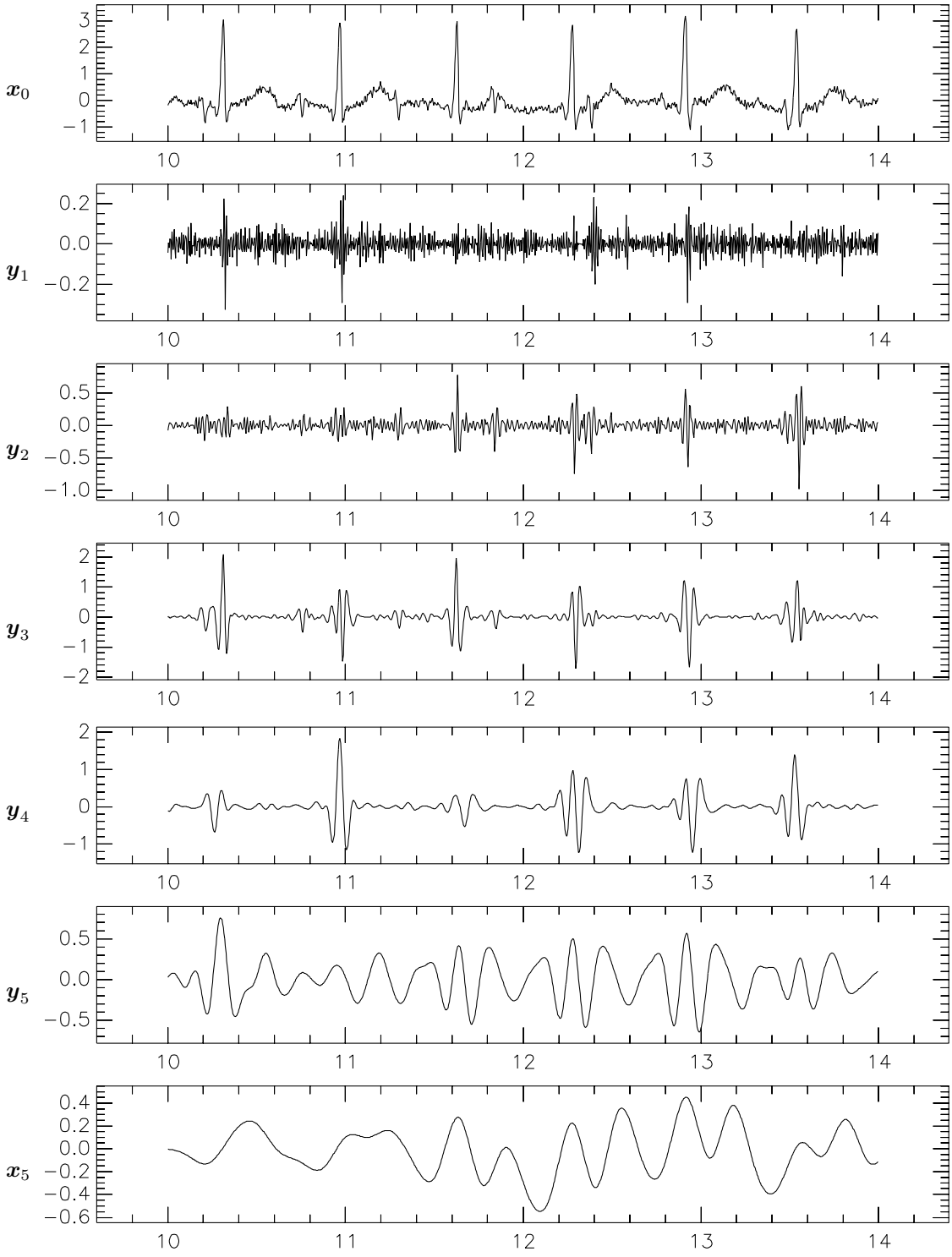


Figure 5.11: SQUID signal: Discrete wavelet analysis with prefetched vanishing moments. The corresponding synthesis invokes the matched wavelet. The signal components at each scale are shown. That is, for $\boldsymbol{x}_j$ we plot $\left(\boldsymbol{x}_j * \widetilde{\varphi}\right) \uparrow 2^j$ and for $\boldsymbol{y}_j$ we plot $\left(\boldsymbol{y}_j * \widetilde{\psi}\right) \uparrow 2^j$. $\boldsymbol{x}_0$ is the analysed signal.

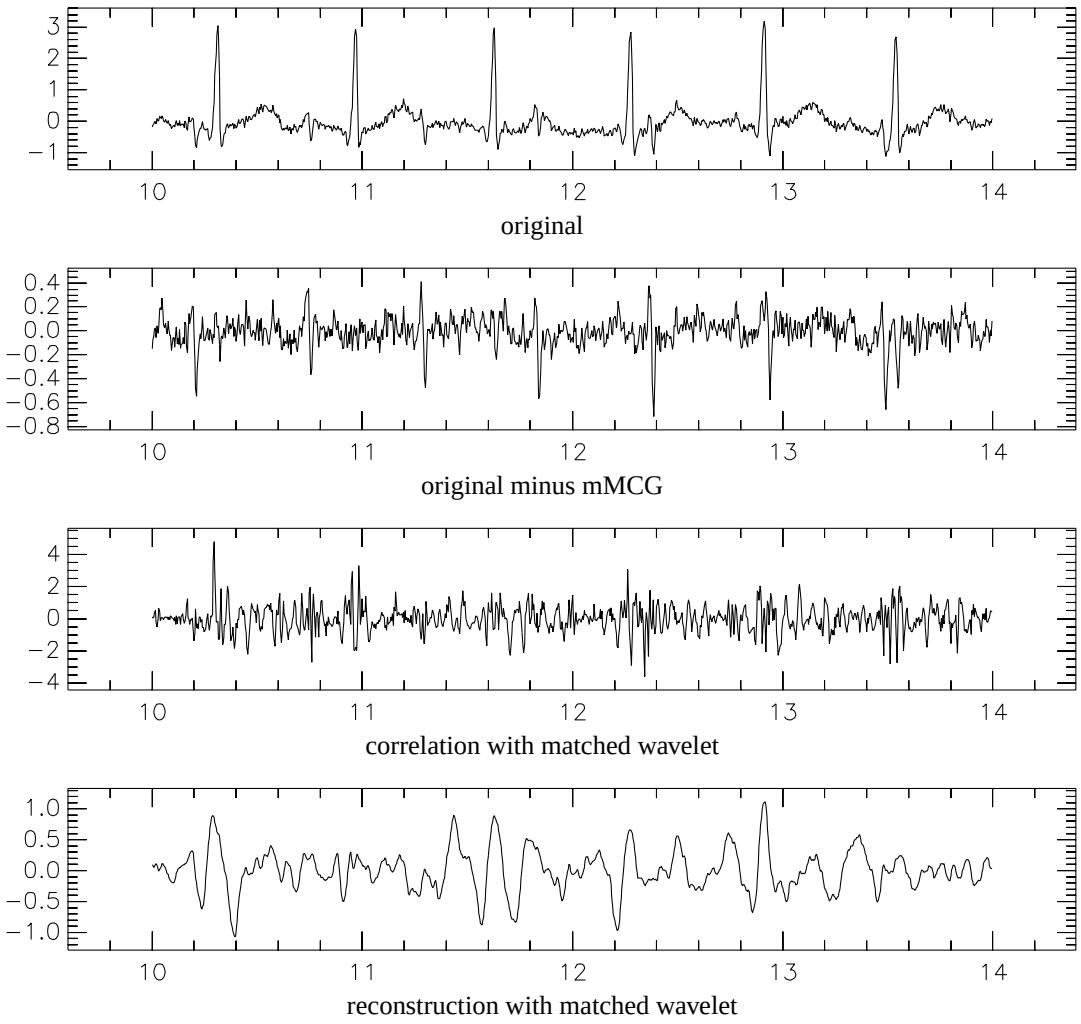


Figure 5.12: SQUID signal: cancelled pattern. The first signal is the original signal, again. For comparison the second plot is the result from the comb filter cancellation, again. The third plot is the result from the transform using correlation with the matched wavelet. The fourth plot was created by reconstruction with a matched wavelet.

That is, we clip all values to the range $[-\lambda, \lambda]$.

We will close this section with an exemplary application of the shrinkage algorithm. Because the lack of a good strategy we clip all coefficients to a tenth of the quadratic mean. This way we stay independent from the energy of each level. The result can be seen in Figure 5.12.

5.3 Condition monitoring on linear guideways

In [P+05] PRÜNTE employs the wavelet transform with matched wavelets for the supervision of linear guideways. Machinery must be monitored in order to detect defects early. Defects can be abrasion of guideways or of balls in ball bearings, impurities inside of a carriage, blocking of the balls, sealing wears, and local defects (pittings) of guideways. Sensor data shall be processed in order to encounter defects. Ideally it is possible to classify the defects.

The cited paper analyses data from an encapsulated piezo-ultrasonic microphone in order to discover pittings in a guideway. Depending on the speed of the machine the acoustic response of the pitting is dilated.

Because of this the wavelet transform is an appropriate tool for this task. Former work on this topic was based on the continuous wavelet transform with promising results. Defects were detected by looking for wavelet coefficients that are above a certain threshold. With a discrete wavelet transform with respect to matched wavelets the threshold method remains the same. The gain is of course the acceleration of the computation. In order to reduce dependencies on the dyadic grid of scales and positions multiple wavelet transforms are performed with respect to wavelets that are matched to slightly rescaled and translated patterns. This can be considered as a generalisation of *phaselets*.

Since the task is the plain detection of patterns no smoothness constraints for the reconstruction are necessary. This simplifies the application but raises the question why there must be a perfect reconstruction filter bank if no reconstruction takes place. Indeed the constraint of perfect reconstruction is justified by the property that a perfect reconstruction transform keeps all information of the input data and nothing is lost.

5.4 Summary and outlook

The experiences from the applications presented here are the following: Matching discrete wavelets with patterns by lifting has proven to work. In order to be numerically stable the discrete wavelet transform must be modified. This causes a double amount of transformed data. On the one hand this is not so bad because of a reduced sensitivity to translations of the signal. On the other hand redundancy makes it difficult to clearly extract or cancel certain signal components. The sensitivity to dyadic scales is less than expected initially.

A real world application where the method works well (or even where it is superior to traditional approaches) is still missing. Because the matching problem can be solved analytically it is possible to match a wavelet with an analytic function. That is, mathematically motivated patterns are also good candidates for matching and maybe there are such patterns and related problems where our method can prove its strengths.

Another way out is to modify the optimisation target. Instead of simply matching a wavelet with a pattern we can try to solve a problem like: “Each dilated and translated version of the pattern shall induce significant wavelet coefficients only in a specific concentrated area.” This is obviously much more complicated.

We have seen that the coupling of vanishing moments of the primal wavelet and the smoothness of the dual generator is an unfortunate connection. Our modified transform cannot suppress vanishing moments and the oscillations they cause. It can only shift them to where their effect is not so serious. Future efforts may concentrate on some of the generalisations listed in Section 2.2.5 such as multi-channel or multi-wavelet transforms where no coupling between vanishing moments and smoothness exists.

Appendix A

Miscellaneous

A.1 Tools

Both for the research and the preparation of this document a lot of tools was used. Their authors put much effort into them, but they gave away the tools free of charge with full documentation and with open source code. They shall be appreciated here. Without them this document would not be possible, at least it would have made less fun.

- LaTeX
  TeX is the well-known type-setting system by DONALD E. KNUTH. LaTeX by LESLIE LAMPORT is build on TeX and provides commands for structuring documents. `BibTeX` was used for creating a references list from the citations in the document and a literature data base. The `IEEEtrantools` package was used because of its `IEEEeqnarray*` environment with flexible column definitions and specific placement of equation labels with `\yeslabel`. In contrast to WYSIWYG systems the TeX based type-setting allows programming and more abstraction, that is a better separation between layout and intention.
- `MetaPost`
  `MetaPost` is a programming language for graphics. It is similar to `MetaFont` which is the font generation part of the TeX system. It certainly suffers from the problem that programming requires more effort at the first time but simplifies repetitive tasks as well as later modification and corrections.
- Haskell [PJ98]
  Haskell is a functional programming language, that is, everything must be expressed in terms of functions. This means that all input and output of a piece of an algorithm is explicit. Haskell has non-strict semantics which allow for potentially infinite data structures. The syntax is close to mathematical notation but prefers strictness and unambiguity to conciseness. That is, it can also help understanding mathematical notation. Haskell supports higher order functions and is thus close to ideas of functional analysis. The `QuickCheck` package was used to verify some of the statements of this work with random input. Both `GHC`, the Glasgow Haskell Compiler, and the Haskell interpreter `Hugs` were used.
- `lhs2TeX`
  This is a TeX preprocessor which was originally developed in order to typeset Haskell programs in an appealing way. In this work it was used for inserting generated graphics and typesetting Haskell expressions in mathematical formulas. That is, some of the formulas in the document can be evaluated in a Haskell environment.
- functional MetaPost
  This package is a Haskell wrapper to `MetaPost`. That is, graphics can be programmed in Haskell without directly addressing `MetaPost`. All flowcharts in this document are generated with this tool.
- GNUPlot
  GNUPlot is a popular plotting program. It can be invoked by a Haskell wrapper which allows programming of plots like those of refinable functions.
- `Modula-3` [DO]
  `Modula-3` is a safe strongly typed systems programming language. It is a joint development of Digital Equipment Corporation and Olivetti. The Polytechnique Montreal Distribution of `Modula-3` is based on DEC SRC `Modula-3` release 3.6 with enhancements by LOUIS DUBEAU, JEROME COLLIN and MICHEL DAGENAIS. The Critical Mass `Modula-3` compiler is another further development of the SRC compiler. The numerical analysis library `m3na` was used as a starting point for advanced mathematical operations.
- PLPlot [LF05]
  This is a plotting library with a binary interface which can be used from `Modula-3`.
- FFTW [FJ]
  The "Fastest FOURIER Transform in the West" is a library which performs several tests to achieve optimal performance an any machine. It provides algorithms with run-time proportional to $n \cdot \log n$ for data sets of any size n, not only powers of 2. It can be called from `Modula-3`.
- LAPACK [ABB+99]
  Is the standard package for doing numerical linear algebra. We invoke it from `Modula-3`.

- SWIG
  The “Simplified Wrapper and Interface Generator” assists with generating wrappers to C and C++ libraries. It was used to generate wrappers from `Modula-3` to PLPlot and FFTW.
- CVS
  The “Concurrent Version System” manage several versions of files. Additionally it allows developers to merge changes that were made independently to the same file.
- Darcs
  “David’s Advanced Revision Control System” is also a version management system. It is written in Haskell. It manages patches which can be shared between several developers. A central repository like for CVS is not necessary. With some restrictions the order of patches can be changed. It allows renaming of files, directories and tokens.
- Integer sequence lookup [Slo03]
  This is a WWW site developed and maintained by NEIL J. A. SLOANE which looks for (well-studied) integer sequences that contain a given subsequence. This way you find an assumption that is worth a trial of a proof. It let you easily enter into areas of mathematics you have never cared of.
- Inverse symbolic calculator, now PLOUFFE’s inverter [Plo06]
  This is a WWW project, too. The inverse symbolic calculator finds expressions containing standard algebraic and transcendent functions which approximate a given fraction. This is an invaluable tool for finding assumptions about the structure of an expression for a value that was found numerically.

List of Figures

List of Tables

Bibliography

[ABB+99] Ed Anderson, Zhaojun Bai, C. Bischof, Susan Blackford, James Demmel, Jack Dongarra, J. Du Croz, A. Greenbaum, Sven Hammarling, A. McKenney, and D. Sorensen. *LAPACK User's Guide*, 1999.

[Amm96] Gregory S. Ammar. Classical foundations of algorithms for solving positive definite toeplitz equations. *Calcolo*, 33:99–113, 1996.

[Bat87] G. Battle. A block spin construction of ondelettes. Part I: Lemarié functions. *Communication on mathematics and physics*, 110:601–615, 1987.

[BBC+01] Arne Barinka, Titus Barsch, Philippe Charton, Albert Cohen, Stephan Dahlke, Wolfgang Dahmen, and Karsten Urban. Adaptive wavelet schemes for elliptic problems: Implementation and numerical experiments. *SIAM J. Sci. Comput.*, 23(3):910–939, 2001.

[Bra03] Andrew P. Bradley. Shift-invariance in the discrete wavelet transform. In C. Sun, H. Talbot, S. Ourselin, and T. Adriaansen, editors, *Proceedings of VIIth Digital Image Computing: Techniques and Applications*. Sydney, December, 10th-12th 2003.

[Bre05] Kristian Bredies. Remark on binary representation of elements of $\mathbb{Z}_{2^n-1}$. private communication, December 2005.

[BU00] Thierry Blu and Michael Unser. The fractional spline wavelet transform: Definition and implementation. In *Proceedings of the Twenty-Fifth IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP'00)*, volume I, pages 512–515, Istanbul, Turkey, June 5th–9th, 2000.

[BW92] Marc A. Berger and Yang Wang. Multidimensional two-scale dilation equations. In Charles K. Chui, editor, *Wavelets: A Tutorial in Theory and Applications*, volume 2 of *Wavelet Analysis and its Applications*, chapter IV, pages 295–323. Academic Press, Inc., 1992.

[Caj93] Florian Cajori. *A History of Mathematical Notation*. Dover, New York, 1993.

[CD93] Albert Cohen and Ingrid Daubechies. Orthonormal bases of compactly supported wavelets. III. Better frequency resolution. *SIAM Journal on Mathematical Analysis*, 24(2):520–527, 1993.

[CD95] Ronald R. Coifman and David L. Donoho. Translation-invariant de-noising. In A. Antoniadis and G. Oppenheim, editors, *Wavelets and Statistics*, volume 103 of *Lecture Notes in Statistics*, pages 125–150. Springer-Verlag, New York, 1995.

[CL01] Antonin Chambolle and Bradley J. Lucier. Interpreting translation-invariant wavelet shrinkage as a new image smoothing scale space. *IEEE Transactions on Image Processing*, 10:993–1000, 2001.

[Con90] Jean-Pierre Conze. Sur le calcul de la norme de sobolev des fonctions d'échelles. Technical report, Dept. de Math. Université de Rennes (France), 1990.

[CR90] Jean-Pierre Conze and Albert Raugi. Fonctions harmoniques pour un opérateur de transition et applications. (harmonic functions for a transition operator and applications). *Bulletin de la Société Mathématique de France*, 118(3):273–310, 1990.

[CW92] Charles K. Chui and Jian-Zhong Wang. On compactly supported spline wavelets and a duality principle. *Transactions of the American Mathematical Society*, 330:903–916, 1992.

[Dah97] Wolfgang Dahmen. Wavelet and multiscale methods for operator equations. *Acta Numerica*, 6:55–228, 1997.

[Dau92] Ingrid Daubechies. *Ten Lectures on Wavelets*, volume 61 of *Cbms-Nsf Regional Conference Series in Applied Mathematics*. SIAM (Society for Industrial and Applied Mathematics), 1992.

[Dau04] Hal Daume. Yet another haskell tutorial. `http://www.isi.edu/~hdaume/htut/`, 2004.

[DDHS97] Stephan Dahlke, Wolfgang Dahmen, Reinhard Hochmuth, and Reinhold Schneider. Stable multiscale bases and local error estimation for elliptic problems. *Appl. Numer. Math.*, 23:21–48, 1997.

[DGM99] Stephan Dahlke, Karlheinz Gröchenig, and Peter Maass. A new approach to interpolating scaling functions. *Applicable Analysis. An International Journal*, 72(3-4):485–500, 1999.

[DHK+03] Mamadou Sanou Diallo, Matthias Holschneider, Michail Kulesh, Frank Scherbaum, and Frank Adler. Characterization of seismic waves polarization attributes using continuous wavelet transforms. Technical report, Preprint series of the DFG priority program 1114 "Mathematical methods for time series analysis and digital image processing", July 2003.

[Dij86a] Edsger W. Dijkstra. Address to my students, April 1986.

[Dij86b] Edsger W. Dijkstra. The nature of my research and why I do it, November 1986.

[DM93] Wolfgang Dahmen and Charles A. Micchelli. Using the refinement equation for evaluating integrals of wavelets. *SIAM Journal Numerical Analysis*, 30:507–537, 1993.

[DO] Digital Equipment Corporation (DEC) and Olivetti. Modula-3 homepage. `http://www.m3.org/`.

[DS98] Ingrid Daubechies and Wim Sweldens. Factoring wavelet transforms into lifting steps. *J. Fourier Anal. Appl.*, 4(3):245–267, 1998.

[Dut89] Pierre Dutilleux. An implementation of the algorithme à trous to compute the wavelet transform. In Jean-Michel Combes, Alexander Grossman, and Philippe Tchamitchian, editors, *Wavelets: Time-Frequency Methods and Phase Space*, pages 298–304. Berlin, Germany: Springer-Verlag, 1989.

[DVMS00] Johan M. De Villiers, Charles A. Micchelli, and Tomas Sauer. Building refinable functions from their values at integers. *Calcolo*, 37:139–158, 2000.

[EG92] Lawrence Evans and Ronald F. Gariepy. *Measure theory and fine properties of functions*. CRC Press, 1992.

[FJ] Matteo Frigo and Steven G. Johnson. FFTW: The Fastest FOURIER Transform in the West, version 3. `http://www.fftw.org/`.

[Fow05] James E. Fowler. The redundant discrete wavelet transform and additive noise. *IEEE Signal Processing Letters*, 12(9):629–632, September 2005.

[GHM94] Jeffrey S. Geronimo, Douglas P. Hardin, and Peter R. Massopust. Fractal functions and wavelet expansions based on several scaling functions. *Journal of Approximation Theory*, 78(3):373–401, September 1994.

[GJP02] Anubha Gupta, S.D. Joshi, and Surendra Prasad. On a new approach for estimating wavelet matched to signal. In *Proceedings of the Eighth National Conference on Communications, Bombay*, January 2002.

[Gre96] Thomas Greiner. Orthogonal and biorthogonal texture matched wavelet filterbanks for hierarchical texture analysis. *Signal Processing*, 54:1–22, April 1996.

[Ham89] Richard W. Hamming. *Digital Filters*. Signal Processing Series. Prentice Hall, January 1989.

[HJ85] Roger Horn and Charles Johnson. *Matrix Analysis*. Cambridge University Press, 1985.

[HKMMT89] Matthias Holschneider, Richard Kronland-Martinet, Jean Morlet, and Philippe Tchamitchian. A real-time algorithm for signal analysis with the help of the wavelet transform. In Jean-Michel Combes, Alexander Grossman, and Philippe Tchamitchian, editors, *Wavelets: Time-Frequency Methods and Phase Space*, pages 286–297. Berlin, Germany: Springer-Verlag, 1989.

[Hug84] John Hughes. Why functional programming matters. `http://www.md.chalmers.se/~rjmh/Papers/whyfp.pdf`, 1984.

[Kla01] Esther Klann. Verfeinerbare Funktionen, Verfeinerungsschemata und Kaskadenalgorithmus. Master's thesis, Universität Hamburg, 2001.

[KP01] Walter G. Kelley and Allan C. Peterson. *Difference Equations. An Introduction with Applications*. Harcourt Academic Press, 2001.

[KRG95] Herbert Kästner, Helmut Rudolph, and Siegfried Gottwald, editors. *Kleine Enzyklopädie Mathematik*. Bibliographisches Institut, Mannheim, 1995.

[LB98] James M. Lewis and C. Sidney Burrus. Approximate continuous wavelet transform with an application to noise reduction. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Seattle, WA, May 1998.

[Lem88] Pierre-Gilles Lemarié. Ondelettes à localisation exponentielle. *J. Math. pures et appl.*, 67(3):227–236, 1988.

[LF05] Maurice J. LeBrun and Geoffrey Furnish. PLPlot: A library for scientific function plots. `http://www.plplot.org/`, 2005.

[LGO+95a] Markus Lang, Haitao Guo, Jan E. Odegard, C. Sidney Burrus, and Raymond O. Wells. Noise Reduction Using an Undecimated Discrete Wavelet Transform. In *IEEE Signal Processing Letters*, 1995.

[LGO+95b] Markus Lang, Haitao Guo, Jan E. Odegard, C. Sidney Burrus, and Raymond O. Wells. Nonlinear Processing of a Shift Invariant DWT for Noise Reduction. In *SPIE Symp. OE/Aerospace Sensing and Dual Use Photonics, Algorithm for Synthetic Aperture Radar Image*, Orlando, FL, April 1995.

[LMR97] Alfred Karl Louis, Peter Maaß, and Andreas Rieder. *Wavelets: Theory and Application*. Wiley, Chichester, 1997.

[Lor05] Dirk A. Lorenz. *Wavelet Shrinkage in Signal and Image Processing - An Investigation of Relations and Equivalences*. PhD thesis, Universität Bremen, February 2005.

[LR94] Dieter Landers and Lothar Rogge. *Nichtstandard-Analysis: mit 204 Übungsaufgaben*. Springer-Lehrbuch. Springer, 1994.

[Maa96] Peter Maaß. Families of orthogonal 2d wavelets. *SIAM Journal of Mathematical Analysis*, 27(5):1454–1481, September 1996.

[Mae97] Stéphane H. Maes. Fast quasi-continuous wavelet algorithms for analysis and synthesis of one-dimensional signals. *SIAM Journal on Applied Mathematics*, 57(6):1763–1801, December 1997.

[Mal99] Stéphane Mallat. *A Wavelet Tour of Signal Processing*. Academic Press, San Diego, CA, USA, 1999.

[MMOP04] Michel Misiti, Yves Misiti, Georges Oppenheim, and Jean-Michel Poggi. *Wavelet Toolbox User's Guide*. The MathWorks, Inc., third edition, 2004.

[Mol98] Wolfgang Moldenhauer. Über Abschätzungen von $2n$ über n (I). *Wurzel*, 8:172–176, August 1998.

[MZ93] Stéphane G. Mallat and Zhifeng Zhang. Matching pursuits with time-frequency dictionaries. *IEEE Transactions on Signal Processing*, 41(12):3397–3415, December 1993.

[NS95] Guy P. Nason and Bernard W. Silverman. The stationary wavelet transform and some statistical applications. In A. Antoniadis and G. Oppenheim, editors, *Wavelets and Statistics*, volume 103 of *Lecture Notes in Statistics*, pages 281–299. Springer-Verlag, New York, 1995.

[Oja98] Harri Ojanen. Orthonormal compactly supported wavelets with optimal Sobolev regularity. *arXiv*, July 1998.

[P+05] Ludger Prünte et al. Condition monitoring on linear guideways by discrete selflet transformation. to be published, September 2005.

[PJ98] Simon Peyton-Jones. Haskell 98 language and libraries, the revised report. `http://www.haskell.org/definition/`, 1998.

[Plo06] Simon Plouffe. Plouffe's inverter. `http://pi.lacim.uqam.ca/`, 2006.

[PW00] Donald B. Percival and Andrew T. Walden. *Wavelet Methods for Time Series Analysis*. Number 4 in Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, 2000.

[RB97] Olivier Rioul and Thierry Blu. Simple regularity criteria for subdivision schemes. II. The rational case. *preprint*, 1997.

[RC00] M. Rao Raghuveer and Joseph O. Chapa. Algorithms for designing wavelets to match a specified signal. *IEEE Transactions on Signal Processing*, 48(12):3395–3406, 2000.

[Rit02] Jörg Ritter. *Wavelet based image compression using FPGAs*. PhD thesis, Martin-Luther-Universität Halle-Wittenberg, December 2002.

[RN96] Peter Rieder and Josef A. Nossek. Smooth multiwavelets based on two scaling functions. In *Proceedings of the IEEE-SP International Symposium on Time-Frequency and Time-Scale Analysis*, pages 309–312, June 18–21 1996.

[RS96] Sherman D. Riemenschneider and Zuowei Shen. Multidimensional interpolatory subdivision schemes. *SIAM Journal on Numerical Analysis*, 34(6):2357–2381, 1996.

[Sel99] Ivan W. Selesnick. Interpolating multiwavelet bases and the sampling theorem. *IEEE Transactions on Signal Processing*, 47(6):1615–1621, 1999.

[Sel01] Ivan W. Selesnick. The double density DWT. In Arthur A. Petrosian and Francois G. Meyer, editors, *Wavelets in Signal and Image Analysis: From Theory to Practice*. Kluwer, 2001.

[Slo03] Neil James Alexander Sloane. The on-line encyclopedia of integer sequences. `http://www.research.att.com/~njas/sequences/`, 2003.

[SN97] Gilbert Strang and Truong Nguyen. *Wavelets and Filter Banks*. Wellesley-Cambridge Press, 1997.

[SP96] Amir Said and William A. Pearlman. A new fast and efficient image codec based on set partitioning in hierarchical trees. *IEEE Transactions on Circuits and Systems for Video Technology*, 6, June 1996.

[SSZ99] Gilbert Strang, Vasily Strela, and Ding-Xuan Zhou. Compactly supported refinable functions with infinite masks. In L. Baggett and D. Larson, editors, *The Functional and Harmonic Analysis of Wavelets and Frames*, volume 247 of *Contemporary Mathematics*, pages 285–296. American Mathematical Society, 1999.

[Sto99] Josef Stoer. *Numerische Mathematik 1*. Springer-Lehrbuch. Springer, eighth edition, 1999.

[Str95] Gernot Stroth. *Lineare Algebra*. Heldermann Verlag, 1995.

[Str96] Gilbert Strang. Eigenvalues of $(\downarrow 2)H$ and convergence of the cascade algorithm. *IEEE Transactions on Signal Processing*, 44:233–238, 1996.

[Str98] Gernot Stroth. *Algebra*. deGruyter, 1998.

[Str02] Tilo Strutz. *Bilddatenkompression: Grundlagen, Codierung, MPEG, JPEG*. Praxiswissen. Vieweg, second edition, 2002.

[Tes01] Gerd Teschke. *Waveletkonstruktion ueber Unschärferelationen und Anwendungen in der Signalanalyse*. PhD thesis, University of Bremen, 2001.

[Thi01] Henning Thielemann. Adaptive construction of wavelets for image compression. Master's thesis, Martin-Luther-Universität Halle-Wittenberg, Fachbereich Mathematik/Informatik, 2001.

[Thi04] Henning Thielemann. Bounds for smoothness of refinable functions. *Linear Algebra and its Applications*, 391:245–260, November 2004.

[Tho99] Simon Thompson. *Haskell: The Craft of Functional Programming*. Pearson, Addison Wesley, second edition edition, 1999.

[TM02] David S. Taubman and Michael W. Marcellin. *JPEG 2000: Image Compression Fundamentals, Standards, and Practice*. Kluwer Academic Publishers, January 2002.

[Tri80] Hans Triebel. *Höhere Analysis*. Harri Deutsch, 1980.

[Tri92] Hans Triebel. *Theory of Function Spaces II*. Monographs in Mathematics. Birkhäuser, 1992.

[UAE92] Michael Unser, Akram Aldroubi, and Murray Eden. On the asymptotic convergence of B-spline wavelets to Gabor functions. *IEEE Transactions on Information Theory*, 38(2):864–872, March 1992. Part II.

[UAE93] Michael Unser, Akram Aldroubi, and Murray Eden. A family of polynomial spline wavelet transforms. *Signal Processing*, 30(2):141–162, January 1993.

[UB99] Michael Unser and Thierry Blu. Construction of fractional spline wavelet bases. In *Proceedings of the SPIE Conference on Mathematical Imaging: Wavelet Applications in Signal and Image Processing VII*, volume 3813, pages 422–431, Denver CO, USA, July 19-23, 1999.

[UB00] Michael Unser and Thierry Blu. Fractional splines and wavelets. *SIAM Review*, 42(1):43–67, March 2000.

[URB97] Geert Uytterhoeven, Dirk Roose, and Adhemar Bultheel. Wavelet transforms using the lifting scheme. Technical report, Katholieke Universiteit Leuven, Belgium, 1997.

[UTA98] Michael Unser, Philippe Thevenaz, and Akram Aldroubi. Shift-orthogonal wavelet bases. *IEEE Transactions on Signal Processing*, 46(7):1827–1836, 1998.

[UVWJ+98] Geert Uytterhoeven, Henk Van Wulpen, Maarten Jansen, Dirk Roose, and Adhemar Bultheel Bultheel. WAILI: A software library for image processing using integer wavelet transforms. In K. M. Hanson, editor, *Medical Imaging 1998: Image Processing*, volume 3338, pages 1490–1501. International Society for Optical Engineering, 1998.

[Vil93] Lars F. Villemoes. Sobolev regularity of wavelets and stability of iterated filter banks. *Progress in wavelet analysis and applications*, pages 243–251, 1993.

[VK95] Martin Vetterli and Jelena Kovačević. *Wavelets and subband coding*. Signal Processing Series. Prentice-Hall, 1995.

[VRM+04] Jiri Vrba, Stephen E. Robinson, Jack McCubbin, Curtis L. Lowery, Han Eswaran, James D. Wilson, Pamela Murphy, and Hubert Preißl. Fetal MEG redistribution by projection operators. *IEEE Transactions on Biomedical Engineering*, 51(7):1207–1218, July 2004.

[Wei05] Eric W. Weisstein. Mathworld. `http://mathworld.wolfram.com/`, 2005.

[ZDW04] Jian-Kang Zhang, Timothy N. Davidson, and K. Max Wong. Efficient design of orthonormal wavelet bases for signal representation. *IEEE Transactions on Signal Processing*, 52(7):1983–1996, July 2004.

[Zöl02] Udo Zölzer, editor. *DAFx: Digital Audio Effects*. John Wiley and Sons Ltd., February 2002.

Index