Hans-Georg Stark Wavelets and Signal Processing Hans-Georg Stark

# Wavelets and Signal Processing

An Application-Based Introduction

With 67 Figures and 7 Tables



Professor Dr. Hans-Georg Stark FH Aschaffenburg - University of Applied Sciences FB Ingenieurwissenschaften Würzburger Str. 45 63743 Aschaffenburg Germany hans-georg.stark@fh-aschaffenburg.de

Library of Congress Control Number: 2005921923

#### ISBN 3-540-23433-0 Springer Berlin Heidelberg New York

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilm or in any other way, and storage in data banks. Duplication of this publication or parts thereof is permitted only under the provisions of the German Copyright Law of September 9, 1965, in its current version, and permission for use must always be obtained from Springer. Violations are liable to prosecution under the German Copyright Law.

Springer is a part of Springer Science+Business Media

springeronline.com

© Springer-Verlag Berlin Heidelberg 2005 Printed in The Netherlands

The use of general descriptive names, registered names, trademarks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

Typesetting: By the author Production: LE-T<sub>E</sub>X Jelonek, Schmidt & Vöckler GbR, Leipzig Cover design: medionet AG, Berlin Printed on acid-free paper 7/3142/YL - 5 4 3 2 1 0

To Fabi and Paula

# Preface

"The idea of this book arose in a conversation with H. A. Bethe, who remarked that a little book about modern field theory which contained only memorable results would be a good thing"<sup>1</sup>.

As a graduate student working in theoretical physics, in particular desperately needing a readable and compact treatment of quantum field theory, I came across a beautifully written book: "PCT, Spin and Statistics and all that" [35].

Here we are dealing with signal processing instead of quantum field theory and the present book does not claim to reach a level of clarity and deepness comparable to the respective achievement in [35]. Nevertheless, the basic intention of this book perfectly is described by the above quotation when replacing "field theory" with "wavelet analyis": It should provide a quick and readable introduction to the wavelet topic for anyone interested in learning about wavelets and/or working with them and hopefully will contain only memorable results.

What does the profile of the "ideal reader" of this little book look like? The largest portion of the covered material should be readable for anyone with an elementary technical background and familiar with basic notions of calculus like integrals and complex numbers. Thus these parts should be accessible for readers with an engineering or science education on a bachelor level who are interested in a short and compact introduction into wavelet applications. The target group includes academia and (in particular) practitioners in industry.

Some parts of the book are of interest mainly for readers with a specialization on, e.g., electrical engineering or communications engineering. For these parts basic familiarity with notions from signal analysis like Fourier transforms and digital filtering will be helpful. Elementary information about these topics is collected in the appendix.

There are some excellent introductory books on wavelets and wavelet applications mainly focussing on the discrete wavelet transform (cf., e.g. [38]).

<sup>&</sup>lt;sup>1</sup> Quotation from the preface of [35]

#### VIII Preface

This book will contain a fairly large part dealing with the continuous transform, since in my opinion the continuous transform provides a very intuitive insight into the essence of wavelet techniques and since the continuous version is close to the Fourier transform, a standard tool in engineering.

As with all books, selection and presentation of the material reflect the author's view of the topic. My hope is that after reading the book the reader will have gained an intuitive insight into what wavelets are. This insight should then provide a basis on which to decide, whether wavelets are interesting for the particular needs of the reader and, if yes, how he or she wants to apply them. Therefore this book is intended to serve as an introductory guide to the topic.

Moreover, the wavelet story is a story about a fascinating and exciting scientific development. If this fascination can be felt during reading, this is among the best I can hope for.

It is a pleasure to thank Eva Hestermann-Beyerle and Monika Lempe from Springer-Verlag for their patience and professional support. Their comments had substantial influence on sharpening the profile of this book. The work reported on in section 4.2 would not have been possible without the collaboration and support of Gernod Laufkötter, Andreas Divivier and Thomas Goseberg, my friends and colleagues at tecmath AG. I am very grateful for the opportunity to work with them.

Dr. Hans-Jürgen Stahl checked the English of chap. 1, a copyeditor from Springer-Verlag did so with the complete manuscript. I am indebted to both for valuable hints which helped me to improve spelling and style.

Aschaffenburg, spring 2005

Hans-Georg Stark

# Contents

1	Introduction			
	1.1	Signals and Signal Processing	1	
	1.2	Local Analysis	2	
		1.2.1 Transforms	2	
		1.2.2 Fourier Transform	3	
		1.2.3 Short Time Fourier Transform (STFT)	4	
		1.2.4 Wavelet Transform	4	
		1.2.5 Visualization	7	
		1.2.6 Fourier vs. Wavelet Transform - A Comparison		
		Experiment	9	
	1.3	A Roadmap for the Book	11	
2	Cor	ontinuous Analysis		
	2.1	The Short Time Fourier Transform (STFT)	14	
		2.1.1 Definition, Computation and Reconstruction	14	
		2.1.2 Phase Space and Localization Parameters	18	
		2.1.3 Implementation with MATLAB and Visualization	19	
	2.2	The Continuous Wavelet Transform (CWT)	21	
		2.2.1 Definition, Computation and Reconstruction	21	
		2.2.2 Wavelet Examples	26	
		2.2.3 Implementation with MATLAB and Visualization	29	
		2.2.4 Application: Detection of Signal Changes	32	
	2.3	Case Studies	33	
		2.3.1 Analysis of Sensor Signals	33	
		2.3.2 Analysis and Classification of Audio Signals	36	
	2.4	Notes and Exercises	40	
3	The	e Discrete Wavelet Transform		
	3.1	Redundancy of the CWT and the STFT	43	
	3.2	The Haar-System	45	
		3.2.1 Continuous-Time Functions	46	

	3.2.2	Sequences			
3.	3.3 Generalization to Daubechies-Wavelets				
	3.3.1	From Filters to Functions			
	3.3.2	Transfer Properties			
3.	4 Multis	scale Analysis 60			
	3.4.1	One-Dimensional Signals 61			
	3.4.2	Two-Dimensional Signals (Images)			
	3.4.3	Implementations with the MATLAB Wavelet Toolbox 69			
	3.4.4	Generalization: Biorthogonal Filters			
3.	5 A Uni	ifying Viewpoint: Basis Systems			
	3.5.1	One-Dimensional Signals 76			
	3.5.2	Two-Dimensional Signals 79			
	3.5.3	Computation and Visualization with MATLAB 81			
3.	6 Case	Studies			
	3.6.1	Energy Compaction and Compression 81			
	3.6.2	Denoising a Sensor Signal / Real-Time Properties of			
		the Algorithm			
3.	7 Notes	and Exercises			
4 N	Ioro An	plications			
4 IV		Fransform Compression Scheme       95			
-1.	4.1.1	The General Procedure			
	4.1.2	Entropy Coders			
	4.1.3	Optimal Quantization and Examples			
	4.1.4	MATLAB Implementation			
4.		let-Based Similarity Retrieval in Image Archives			
4.		and Exercises			
5 A		s			
5.		er Transform and Uncertainty Relation			
5.		ete Fourier Transform (DFT) 128			
5.	0	al Filters			
5.		ons to Selected Problems			
	5.4.1	Problems from Sect. 2.4			
	5.4.2	Problems from Sect. 3.7			
	5.4.3	Problems from Sect. 4.3			
5.	5 Notat	ions and Symbols 146			
<b>References</b>					
Index					

# Introduction

## 1.1 Signals and Signal Processing

Wavelet analysis had its origins in the mid-eighties. From the very beginning it was driven by application needs: The desire to analyze seismic signals more sensitively than with Fourier techniques led to the first appearance of the continuous wavelet transform formula. In parallel it turned out that the new technique could be applied successfully to certain problems in theoretical physics as well as in pure mathematics. For one of the earliest collections of research and survey papers documenting the state of the art the reader is referred to [6].

It soon turned out that wavelet analysis successfully could be applied to many types of signal processing problems: In *signal analysis* the detection of discontinuities or irregularities was tackled with wavelets. The analysis of medical signals like electrocardiograms of the heart is one of the first reported examples of discontinuity detection (see [6]). For more applications, like the analysis of sensor signals in robotics, cf. sect. 2.3.1.

Signal compression is another impressive example of wavelet applications. JPEG 2000, the present version of the international standard on still image compression is based on wavelet techniques (see, e.g., [36]).

Wavelet applications both in signal analysis and signal compression shall be described in more detail in later sections. This chapter serves as a brief introduction to the main features of the wavelet transform by comparing wavelet transform with Fourier transform, the standard tool of signal analysis. For that purpose we shall work out the common aspects of wavelet and Fourier transforms and point out the main differences. For understanding the following section, the knowledge of the Fourier transform is not a necessary prerequisite. On the other hand, of course, it would be useful, if the reader already had some experience with applications of the Fourier transform. Basic facts about the Fourier transform are collected in the appendix, sections 5.2 and 5.3. Mathematical symbols, used throughout this book, are explained upon their first appearance. They are collected in sect. 5.1 of the appendix.

## 1.2 Local Analysis

In this section we will deal with signals which may be represented by a function f(t) depending on time t. We shall assume that t is a continuously varying parameter; thus f(t) is called a "continuous-time signal".

We shall try to transform f(t) into a representation, which encorporates the desired information about the signal as compactly as possible. The Fourier transform (cf. sections 1.2.2 and 1.2.3) supplies information about the contribution of certain frequencies to the signal, the wavelet transform (cf. sect. 1.2.4) indicates whether details of a certain size are present in a signal and quantifies their respective contribution. Both transforms are called "local" if they not only globally measure frequencies and detail sizes, respectively, but also indicate where they are located in the signal f(t).

There are many applications for the kind of signal information described above – we explicitly mention signal classification and data compression. These applications are described in more detail later, in the subsections below we indicate *how* frequencies and detail sizes may be measured. Furthermore, we will work out the aspects which are common to both transforms and illuminate the respective differences. The transform results shall be visualized and we will give an example which serves as an illustration for the above-mentioned compactness of the respective signal representations.

The purpose of this chapter is to introduce the ideas underlying Fourier and wavelet transforms, respectively. For more - in particular for mathematical - details the reader is referred to the following chapters.

#### 1.2.1 Transforms

All transforms of the signal f(t) described in this section share a common computation principle: The signal is multiplied with a certain "analysis function" and integrated about the time domain. In a symbolic notation the prescription for performing a transform reads

$$f(t) \xrightarrow{transform} \int_{-\infty}^{+\infty} f(u)\overline{g(u)} \, du \tag{1.1}$$

The "analysis function" g(u) characterizes the chosen transform. In general it may be a complex function, the overline denotes the complex conjugate entity. g(u) in a certain way depends on the parameters, i.e. frequencies or detail sizes, to be measured (see below). Thus, by the computation principle given above the transformed entity will depend on these parameters. In other

3

words: the transformed entity again will be a function. These functions we shall denote with "transform" or "transformed signal".

Another common aspect of all transforms discussed in this section is invertibility: From the transformed signal the original signal f(t) may be reconstructed. This is essential for understanding the comparison experiment carried out in sect. 1.2.6.

#### 1.2.2 Fourier Transform

The parameter relevant for the Fourier transform is the circular frequency  $\omega$ , the analysis function reads  $g_{\omega}(u) = e^{ju\omega}$ . Thus the transformed signal is a function depending on  $\omega$  and it is denoted with  $\hat{f}(\omega)$ :

$$\hat{f}(\omega) = \int_{-\infty}^{+\infty} f(u) \overline{g_{\omega}(u)} \, du \tag{1.2}$$

Figure 1.1 illustrates the above computation recipe by plotting both curves required for computing  $\hat{f}(\pi)$ . The signal is shown as a solid curve, the real part of the analysis function  $g_{\pi}(u) = e^{ju\pi}$  is dashed. Obviously, it is an harmonic oscillation with circular frequency  $\omega = \pi$ .

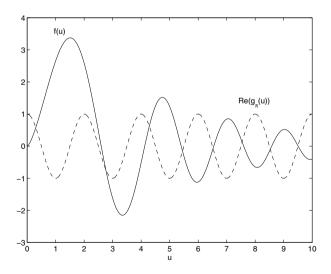


Fig. 1.1. Fourier transform: signal and analysis function

Why does  $\hat{f}(\pi)$  measure the appearance of  $\omega = \pi$  in the signal? The qualitative argument is as follows: If in some time interval the signal oscillates with circular frequency  $\omega = \pi$ , the signal and the analysis function have a

constant mutual phase shift in this interval and therefore provide a nonzero contribution to  $\hat{f}(\pi)$ .

Yet there is no possibility to *localize* the appearance of the circular frequency: If (the absolute value of)  $\hat{f}(\pi)$  is "large", we only know that the signal contains the circular frequency  $\pi$ , but we do not know *where* it appears, since the analysis function extends over the whole real axis. The only label parameterizing the analysis function is circular frequency.

#### 1.2.3 Short Time Fourier Transform (STFT)

This transform sometimes also is called "Windowed Fourier Transform" (WFT). The STFT looks for the appearance of the circular frequency  $\omega$  at a certain time t. The corresponding analysis function reads:  $g_{(\omega,t)}(u) = e^{ju\omega}w(u-t)$ . Here w(u) is a "window function", usually centered about the origin (for an example see below). In the expression w(u-t) this window is shifted to the desired time t.

Now the transformed signal depends on  $\omega$  and t! Since it also will depend on the shape of the window function, it is denoted with  $\hat{f}_w(\omega, t)$ :

$$\hat{f}_w(\omega, t) = \int_{-\infty}^{+\infty} f(u) \overline{g_{(\omega, t)}(u)} \, du$$
(1.3)

For a box window w of width 2, centered symmetrically about 0,  $\omega = \pi$  and t = 8, the computation principle is illustrated in Fig. 1.2. Again the dashed curve shows the real part of the analysis function  $g_{(\pi,8)}(u)$ ; it is obviously now localized at t = 8, since w(u - 8) denotes the box window, shifted by 8 units to the right.

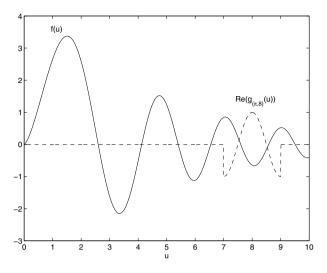
In general, the analysis function will be localized at the respective "analysis time" t . Therefore the transform provides not just global information about the appearance of a certain circular frequency, but in addition the time of this appearance.

The procedure described so far has a disadvantage: If in the above example one is interested in small details of the signal around t = 8, the corresponding frequency of the analysis function must be increased. As an example Fig. 1.2 is redrawn for  $\omega = 6\pi$  in Fig. 1.3.

Obviously the window width is constant and non-adaptive: If one is interested in very tiny signal details (high frequencies) in only a small neighborhood of t = 8, eventually signal parts, which actually are "not of interest", also will be analyzed. Zooming into small details - analogously to a microscope - is not supported.

#### 1.2.4 Wavelet Transform

The wavelet transform has such a zooming property. In contrast to the Fourier transform, the wavelet transform does not look for circular frequencies but



**Fig. 1.2.** STFT: signal and analysis function for  $\omega = \pi$ 

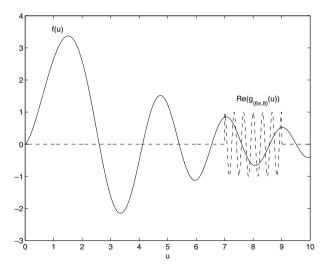


Fig. 1.3. STFT: signal and analysis function for  $\omega = 6\pi$ 

rather for detail sizes a at a certain time t. Instead of detail sizes we also will speak of "scale factors", both notions will be used equivalently. As mentioned already, high frequencies correspond to small details and vice versa, thus, when comparing wavelet with Fourier transforms we have to take into account that frequencies and detail sizes are inversely proportional to each other: There exists a constant  $\beta$  such that 6 1 Introduction

$$a = \frac{\beta}{\omega}.\tag{1.4}$$

We shall now briefly indicate, how the wavelet transform is computed.

Consider a (real or complex) analysis function g, oscillating around the u-axis (mathematically:  $\int_{-\infty}^{+\infty} g(u) du = 0$ ) and decreasing rapidly for  $u \to \pm \infty$ . Such a function is called a "wavelet". In eq. 1.4, relating scale factors with frequencies, the constant  $\beta$  depends on q.

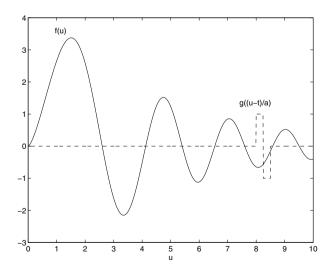
Starting from g consider the following family of functions:  $g_{(a,t)}(u) = \frac{1}{\sqrt{a}g\left(\frac{u-t}{a}\right)}$ . The members of this family are generated from g by shifting the function to t followed by shrinking (a < 1) or dilating (a > 1) the width of the function. The wavelet transform now reads:

$$L_g f(a,t) = \int_{-\infty}^{+\infty} f(u) \overline{g_{(a,t)}(u)} \, du$$
(1.5)

For the "Haar-wavelet"

$$g(u) = \begin{cases} 1 & 0 \le u < \frac{1}{2} \\ -1 & \frac{1}{2} \le u < 1 \\ 0 & \text{else} \end{cases}$$

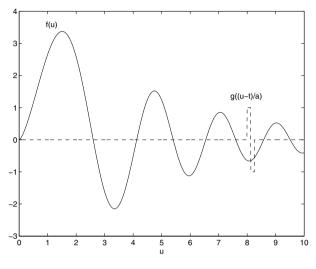
the computation of  $L_g f(a,t)$  with  $a = \frac{1}{2}$  and t = 8 is illustrated in Fig. 1.4.



**Fig. 1.4.** Wavelet transform: signal and analysis function for  $a = \frac{1}{2}$ 

The reader may note that the Haar-wavelet, originally situated in the interval [0,1) now has been shifted to the right by 8 units and its width has

shrunk by the factor  $\frac{1}{2}$ , corresponding to the chosen values of t and a. For  $a = \frac{1}{4}$  and t = 8 we obtain Fig. 1.5.



**Fig. 1.5.** Wavelet transform: signal and analysis function for  $a = \frac{1}{4}$ 

Compare Figs. 1.4 and 1.5 with Figs. 1.2 and 1.3 and note that the wavelet transform shows the desired zooming property in contrast to the STFT: When searching for smaller and smaller details (higher and higher frequencies) with the wavelet transform, the corresponding analysis function is oscillating faster and is contracted.

#### 1.2.5 Visualization

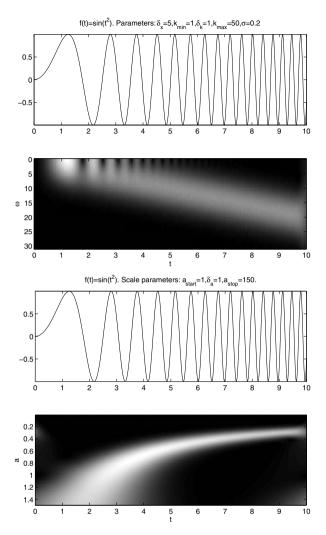
Both the STFT  $\hat{f}_w(\omega, t)$  and the wavelet transform  $L_g f(a, t)$  are functions depending on two variables. A suitable visualization of these functions is of essential importance in signal analysis. A wide-spread graphical representation of two-dimensional functions is the use of contour lines. In signal analysis one usually prefers the visualization of the absolute values of the respective transforms by gray values. High values are coded with bright, low values with dark gray values.

Figure 1.6 shows such a visualization for the STFT (above) and the wavelet transform (below). As a signal in both cases the "chirp"  $f(t) = \sin(t^2)$  has been used.

The chirp is an harmonic oscillation  $\sin(\omega t)$ , whose circular frequency increases with t:  $\omega = t$ . The linear increase of frequency is clearly visible with the STFT (see the upper part of Fig. 1.6).

Since (cf. eq. 1.4) detail size a and frequency  $\omega$  are inversely proportional with respect to each other, for the wavelet transform one would expect a

#### 8 1 Introduction



**Fig. 1.6.** Above: STFT of the chirp-signal  $f(t) = \sin(t^2)$ . Below: Wavelet transform of the chirp-signal  $f(t) = \sin(t^2)$ .

behavior corresponding to a hyperbola (i.e. proportional to  $\frac{1}{t}$ ). The lower part of Fig. 1.6 shows exactly this behavior.

Since the STFT depends on t and  $\omega$ , the gray value coding of the STFT has been performed on the t- $\omega$ -plane. This plane is also called "phase plane", the corresponding gray value coding "phase space representation" of the STFT. Analogously the t-a-plane is called "scale plane" and the corresponding gray value coding of the wavelet transform "scale space representation" of the wavelet transform.

9

#### 1.2.6 Fourier vs. Wavelet Transform - A Comparison Experiment

In a certain sense, the zooming property of the wavelet transform ensures that characteristic features of the analyzed signal on a certain scale are well represented by the transform values corresponding to this scale factor, i.e. not distributed among other scale factors. Moreover, these transform values will be localized at the respective signal parts, where the above-mentioned features are present. These concentration properties - both with respect to scale and time - may be formulated mathematically in a more rigorous way; the purpose of this section is, to give a plausibility argument for the above statement by performing a comparison experiment with the Fourier transform.

The signal displayed in Fig. 1.7 is a section from the beginning of an audio signal. Roughly in the middle, the sudden start of sound clearly can be recognized.

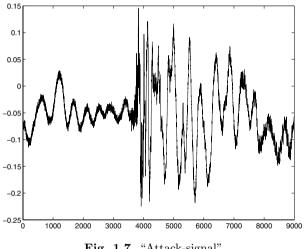


Fig. 1.7. "Attack-signal"

Such signals ("attack-signals") shall locally, i.e. in a small neighborhood of the point, where the sound starts, contain *high frequencies*, equivalently, there will be drastic changes on a *small scale*. In such a situation the zooming property of the wavelet transform should be advantageous when compared with the Fourier transform. To confirm this conjecture, the following experiment has been carried out:

1. Compute the Fourier transform of the signal, keep those 4% of the values of the transformed signal, having the largest absolute values. Put the remaining transform values equal to zero and reconstruct the signal from this modified transform (remember that, as stated in sect. 1.2.1, all transforms discussed here are invertible).

#### 10 1 Introduction

2. Perform the same procedure with the wavelet transform instead of the Fourier transform.

The results of the experiment are shown in Fig. 1.8. The dashed curve shows the result of the Fourier-reconstruction, the wavelet-reconstruction is displayed by +-symbols, the solid line represents the original signal. When comparing with Fig. 1.7, observe that the curves show an enlarged section of the signal from Fig. 1.7 in a neighborhood of the point where the sound starts.

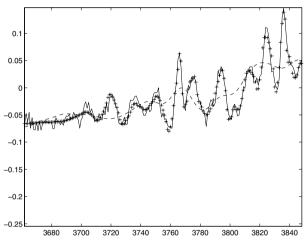


Fig. 1.8. "Attack-signal": reconstructions

Obviously, when using the Fourier transform, the suppression of 96% of the transform signal values - namely those transform signal values having lower absolute values than the retained ones - leads to a global smoothing (low pass filtering) and therefore the local peaks during the attack phase are not reproduced any more. In contrast, applying the same suppression procedure to the wavelet transform does *not* disturb the reproduction of these peaks. This is a clear indication for the above-mentioned concentration properties of the wavelet transform values.

As a final remark we should indicate that the original signal was not represented as a continuous-time signal, but it was discretely sampled. For such signals there exist variants of the continuous formulae 1.2 and 1.5, respectively, with which the above experiment has been carried out. These "discrete transforms" shall be described in later sections.

Moreover, as with any wavelet transform application, the result of the above experiment depends on the chosen wavelet g. The results displayed in Fig. 1.8 have been obtained using the db4-wavelet described in a later section.

## 1.3 A Roadmap for the Book

The topics sketched in this introduction will be described in more detail in chap. 2. Questions like "what is an optimal window function for the Short Time Fourier Transform?" or "how may visualizations as displayed in Fig. 1.6 be accomplished?" will be treated there. Moreover, we will indicate how the original signal may be reconstructed from the respective transforms.

This is of crucial importance for applications like signal compression: There, transforms of the original signal usually are computed in a first step. Subsequently these transforms are modified in such a way that the required storage space for the transformed signal is reduced considerably. Finally, in the "decompression" step the signals are reconstructed from the modified transforms. In chap. 2 also a brief survey is given of typical signal analysis applications of the Short Time Fourier Transform and the wavelet transform, respectively. Two industrial case studies are described in more detail. Section 2.4 contains some exercises. Partly these exercises will be of "paper-and-pen-type", but they also will consist in writing computer programs.

As a programming platform we use the tool MATLAB, which is widespread and a de-facto-standard in the engineering community. Of course it is possible to use the book just as a reference guide to wavelet techniques without performing any programming. For readers interested in using the software described in this book and in developing their own programs, however, the use of MATLAB will be a prerequisite. Readers having already some familiarity with programming languages and looking for a compactly written introduction to MATLAB are referred to [12]. This book provides a very nice and efficient description of MATLAB's main features. It is written in German; if this turns out to be an obstacle, reference [30] is highly recommended.

The MATLAB software which is discussed in this book and has been written by the author may be downloaded from

#### www.springeronline.com/de/3-540-23433-0

Most of it requires only the basis version of MATLAB. Some programs, however, make use of the MATLAB Wavelet Toolbox, a collection of waveletrelated signal processing algorithms. Its use is described in the user's guide [24], which is not just a software handbook. It is moreover a beautifully written introduction to wavelet techniques. Since in the author's opinion MATLAB together with the wavelet toolbox will turn out to be a standard software platform for wavelet-related signal processing, the present book also will provide a short introduction to the most important components of the MATLAB Wavelet Toolbox.

In chapters 1 and 2, respectively, signals are considered to be *continuous-time signals*, even though computerized versions of the algorithms described there necessarily involve a discretization. A different perspective is given in chap. 3: Here from the very beginning signals are *discrete sequences of numbers* and the wavelet transform described there is designed for such sequences.

The corresponding notions, wavelet constructions, signal transform and reconstruction formulae are given in this chapter. Most important for practical applications is the existence of fast algorithms both for transformation and reconstruction. They are also described in this chapter together with MATLAB implementations. Thereafter again applications and case studies are presented. In this context we shall also comment on real-time properties of fast wavelet algorithms. Section 3.7 provides exercises.

Chapter 4 is devoted to a more detailed description of some additional applications of the wavelet transform. First we shall focus on the most popular application, namely data compression. Subsequently, an application related to retrieving images from database management systems is described. Again at the end of this chapter some exercises are presented.

As far as mathematics is concerned, we assume some familiarity of the reader with basic calculus like integral calculus and complex numbers. We shall give no rigorous proofs of mathematical statements, rather we want to provide some intuitive insight into the essence of these statements and their practical meaning. More mathematical details related to background and applications of wavelets are collected in the appendix. It is intended to support the reader, if he or she feels that some additional information would be helpful to understand the main body of the text. Moreover, the appendix contains solutions to selected problems from the exercises and provides a list of frequently used symbols and notations.

# **Continuous Analysis**

In chap. 1 we introduced local transforms of continuous-time signals. These transforms now will be studied in more detail. In particular the concept of phase space localization will lead to the selection of a proper window function for the STFT.

Moreover, we will sketch fast algorithms and their implementation in MATLAB for computing both the STFT and the continuous wavelet transform. For both transforms reconstruction formulae will be provided.

Finally two industrial case studies are presented. The first case study deals with applications of the wavelet transform and the STFT, respectively, to the analysis of signals occurring at a (light) arc welding process. The second case study describes, how the above transforms may be used for classifying audio signals occurring at certain inspection procedures in the automotive industry. Readers primarily interested in applications may first study the definition parts in sections 2.1.1 and 2.2.1, respectively, then skip the rest of these sections and proceed directly to sections 2.2.2, 2.2.4 and 2.3.

Throughout this section the continuous-time signal to be analyzed shall be denoted with f(t). For completeness we also require f(t) to have finite energy (cf. sect. 5.1). As noted there, essentially all practically relevant signals fulfill this requirement.

When numerical algorithms and applications enter the picture, we will not deal with continuous-time signals any more. Instead, we shall consider sequences obtained by sampling the continuous-time signal f(t) for  $t = 0, T_S, 2T_S, \ldots, (N-1)T_S$ . Here  $T_S$  denotes the "sampling distance" and the sequence elements are denoted with  $f_k := f(kT_S)$  ( $k = 0, \ldots, N-1$ ). Note, that in the engineering community also f[k] is used instead of  $f_k$ . Putting everything together, we shall adhere to the following notation:

$$f = \{f_k\}_{k=0}^{N-1} = \{f(kT_S)\}_{k=0}^{N-1}$$
(2.1)

We shall use the symbol f both to represent either a continuous-time signal or a discretely sampled signal. The actual meaning always shall be

#### 14 2 Continuous Analysis

clear from the respective context. Our notations concerning discretely sampled signals are completed with the definitions for the sampling rate  $\nu_S$  and the corresponding circular frequency  $\omega_S$ :

$$\nu_S := \frac{1}{T_S} \tag{2.2}$$

$$\omega_S := \frac{2\pi}{T_S} \tag{2.3}$$

# 2.1 The Short Time Fourier Transform (STFT)

#### 2.1.1 Definition, Computation and Reconstruction

#### Definition

As described in sect. 1.2.3, the STFT performs a local frequency analysis by shifting a window function w to time t and subsequently computing the Fourier transform (cf. eq. 5.1) of the product of the signal and the window:

$$\hat{f}_w(\omega,t) = \int_{-\infty}^{+\infty} f(u) \overline{w(u-t)} e^{-j\omega u} \, du$$
(2.4)

As indicated, this formula is also valid when the window function is complex-valued. In this case, the overline denotes complex conjugation. The subscript w in the expression  $\hat{f}_w(\omega, t)$  indicates the dependency of the transformed signal on the chosen window function w.

Using the Fourier transform correspondence pairs introduced in sect. 5.1, eq. 2.4 also may be written as

$$f(t)\overline{w(t-\tau)} \circ - \bullet \hat{f}_w(\omega,\tau)$$
(2.5)

This equation is the key both for designing a fast algorithm for computing the STFT and for reconstructing the signal from the STFT.

For the rest of this section basic knowledge of Fourier transforms and the discrete Fourier transform as provided in the appendix, sections 5.1 and 5.2, respectively, is required. Readers primarily interested in applications may proceed from here to sect. 2.3.1 and then continue with sect. 2.2.1.

#### Computation

Given a sampled signal  $\{f(kT_S)\}_{k=0}^{N-1}$  (cf. eq. 2.1), the task is to compute a sampled version of the STFT, i.e. to compute the sequence  $\{\hat{f}_w(\omega, kT_S)\}_{k=0}^{N-1}$ .

It will turn out, that a fast algorithm can be given for suitably sampled  $\omega$ -values. We describe now both the algorithm and the restrictions under which it is valid.

Assume f and w to be band-limited and let  $\omega_{max}^{f}$  and  $\omega_{max}^{w}$  denote the maximum frequencies of f and w, respectively (cf. sect. 5.2). Analogously, let  $\omega_{max}$  denote the maximum frequency of  $f(t)w(t-\tau)$ . Then, as implied already by the notation,  $\omega_{max}$  indeed does not depend on  $\tau$  and, moreover,

$$\omega_{max} \le \omega_{max}^f + \omega_{max}^w. \tag{2.6}$$

Without going into details, we mention that this inequality essentially is a consequence from the fact that the spectra of f and w are convolved with each other.

Now define  $\omega_S$  corresponding to eq. 2.3 and assume that the sampling is such that the inequality

$$\omega_S > 2(\omega_{max}^f + \omega_{max}^w). \tag{2.7}$$

is fulfilled. Then eq. 2.6 implies that  $\omega_S > 2\omega_{max}$ . Thus, the Shannon condition 5.12 is fulfilled for  $f(t)\overline{w(t-\tau)}$ . Correspondence pair 2.5 together with theorem 5.1 then leads to the following

Fast algorithm for the computation of  $\hat{f}_w(\omega, t)$ :

1. Define the frequency sampling

$$\omega_k = \frac{k}{N} \omega_S \ (k = 0, \dots, m_N) \tag{2.8}$$

where

$$m_N = \begin{cases} \frac{N}{2} & (N \text{ even})\\ \frac{N-1}{2} & (N \text{ odd}) \end{cases}$$
(2.9)

2. For  $\tau = 0, T_S, 2T_S, \dots, (N-1)T_S$  perform the following steps: a) Define the sequence  $\{f(kT_S)w(kT_S-\tau)\}_{k=0}^{N-1}$  and compute its DFT via the FFT-algorithm. Let the DFT be denoted with  $\{G_k\}_{k=0}^{N-1}$ .

b) Then  

$$\hat{f}_w(\omega_k, \tau) = T_S G_k \ (k = 0, ..., m_N).$$
 (2.10)

Some comments are in place:

1. Usually the sampled signal – in particular the sampling rate  $\omega_S$  – is given. Thus, if the Shannon condition 5.12 is true for f, eq. 2.7 implies that  $\omega_{max}^w$  should be as small as possible in order to avoid distortions by aliasing effects (cf. sect. 5.2).

#### 16 2 Continuous Analysis

2. The remarks made after theorem 5.1 apply also here. Since finitely sampled signals never are band-limited in the strict sense, at least the spectra of f and w must be "sufficiently small" outside the respective intervals  $\left[-\omega_{max}^{f}, \omega_{max}^{f}\right]$  and  $\left[-\omega_{max}^{w}, \omega_{max}^{w}\right]$  such that eq. 2.10 is approximately true.

## Reconstruction

In this subsection we shall consider the question how to reconstruct the original continuous-time finite-energy signal f(t) from the transformed signal  $\hat{f}_w(\omega, \tau)$ . Again we start from correspondence pair 2.5.

Applying the Fourier inversion formula 5.2 to 2.5 we obtain

$$f(t)\overline{w(t-\tau)} = \frac{1}{2\pi} \int_{-\infty}^{+\infty} \hat{f}_w(\omega,\tau) e^{+j\omega t} d\omega$$

Multiplying this equation on both sides with  $w(t-\tau)$  and integrating with respect to  $\tau$  we obtain for the left-hand side:

$$\int_{-\infty}^{+\infty} f(t)|w(t-\tau)|^2 d\tau = f(t) \int_{-\infty}^{+\infty} |w(t-\tau)|^2 d\tau$$
$$= f(t) \int_{-\infty}^{+\infty} |w(u)|^2 du$$

In proceeding from the first to the second line the substitution  $u = t - \tau$  has been made.

For the right-hand side we obtain

$$\frac{1}{2\pi} \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} \hat{f}_w(\omega,\tau) w(t-\tau) e^{+j\omega t} \, d\omega d\tau$$

Equating both sides and resolving for f(t) we finally obtain the "continuous STFT reconstruction formula"

$$f(t) = \frac{\int\limits_{-\infty}^{+\infty} \int\limits_{-\infty}^{+\infty} \hat{f}_w(\omega, \tau) w(t - \tau) e^{+j\omega t} d\omega d\tau}{2\pi \int\limits_{-\infty}^{+\infty} |w(u)|^2 du}$$
(2.11)

Note that the denominator of this formula does only depend on the chosen window function w. Thus it may be precomputed and stored. But even then

for practical applications the usefulness of eq. 2.11 is limited. For sampled signals and the corresponding STFT the numerical approximation of the double integral in the numerator by some quadrature formula leads to an unacceptable computational effort.

Instead, assume that the sampling  ${f(kT_S)}_{k=0}^{N-1}$  (cf. eq. 2.1) of a continuoustime finite energy signal f(t) is given, which fulfills eq. 2.7. Assume, moreover, that the algorithm described in eqs. 2.8 – 2.10 has been performed, resulting in the computation of

$$\hat{f}_w(\omega,\tau) \ (\tau=0,T_S,\ldots,(N-1)T_S; \ \omega=0,\frac{1}{N}\omega_S,\ldots,\frac{m_N}{N}\omega_S)$$

with  $m_N$  given by eq. 2.9.

Then we use the correspondence pair eq. 2.5 again to reconstruct  $\{f(kT_S)\}_{k=0}^{N-1}$  as follows:

Fast STFT-reconstruction:

1. For a given  $\tau$  define with  $\omega_k = \frac{k}{N}\omega_S$   $(k = 0, ..., m_N)$  the sequence  $G_k := \frac{1}{T_S}\hat{f}_w(\omega_k, \tau)(k = 0, ..., N-1)$ , where the coefficients  $G_k(k = m_N + 1, ..., N-1)$  result from

$$G_{N-1} \stackrel{!}{=} \overline{G_1}, G_{N-2} \stackrel{!}{=} \overline{G_2}, \dots$$

(cf. sect. 5.2).

2. Then apply the inverse DFT (eq. 5.14) to this sequence, resulting in the sequence  $\{f(kT_S)\overline{w(kT_S-\tau)}\}_{k=0}^{N-1}$ . Schematically:

$$\{G_k\}_{k=0}^{N-1} \xrightarrow{IDFT} \{f(kT_S)\overline{w(kT_S-\tau)}\}_{k=0}^{N-1}$$
(2.12)

3. Perform the steps above for a suitable subset  $\{\tau_i\}_{i=1}^I$  of  $\tau$ -values, such that  $\tau_1 = 0$ ,  $\tau_I = (N-1)T_S$  and  $w(kT_S - \tau_i)$  is nonzero for any  $kT_S$  in the interval  $[\tau_i, \tau_{i+1}]$ . Then on these intervals  $f(kT_S)$ may be reconstructed by dividing the sequence obtained above through  $w(kT_S - \tau_i)$ .

We remark that the signal can be recovered by performing steps 1 and 2 above for *every*  $\tau$ -value. Step 3 aims at reducing the computational effort, since the IDFT-computation needs to be carried out only for a subset of  $\tau$ -values.

MATLAB-implementations of the STFT-algorithms presented in this section are discussed in sect. 2.1.3.

#### 2.1.2 Phase Space and Localization Parameters

In this section we adopt and deepen the viewpoint on the STFT described in sect. 1.2.3. The concept of localization parameters introduced in sect. 5.1 will lead us to the choice of a certain window function w.

Thus first we rewrite the STFT-formula 2.4 as

$$\hat{f}_w(\omega, t) = \int_{-\infty}^{+\infty} f(u) \overline{g_{(\omega, t)}(u)} \, du$$
(2.13)

where

$$g_{(\omega,t)}(u) = e^{ju\omega}w(u-t) \tag{2.14}$$

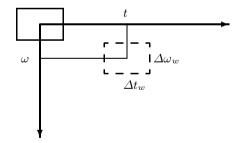
(cf. sect. 1.2.3). Assume that the window function has the following localization parameters (for the corresponding definitions cf. eqs. 5.6 - 5.9):

$$t_w = 0, \ \omega_w = 0, \ \Delta t_w, \ \Delta \omega_w.$$

Then it is an easy exercise to verify that the localization parameters of  $g_{(\omega,t)}$  read

$$t_{g_{(\omega,t)}}=t,\;\omega_{g_{(\omega,t)}}=\omega,\;\Delta t_{g_{(\omega,t)}}=\varDelta t_w,\;\varDelta\omega_{g_{(\omega,t)}}=\varDelta\omega_w.$$

The corresponding cells in the phase plane (cf. sect. 5.1) are drawn in Fig. 2.1.



**Fig. 2.1.** Cells in phase plane, occupied by w (solid) and  $g_{(\omega,t)}$  (dashed), respectively.

Thus, the transition  $w = g_{(0,0)} \rightarrow g_{(\omega,t)}$  corresponds to a pure shift of the cell in the phase plane. Shape and size remain unchanged!

Obviously, an optimal resolution both in time and frequency can be reached if  $\Delta t_w$  and  $\Delta \omega_w$  both are as small as possible. Having the fast STFTalgorithm 2.8 – 2.10 in mind, we remark that condition 2.6 is another strong argument for having an as small as possible  $\Delta \omega_w$ -value. Because of the Heisenberg uncertainty principle 5.10  $\Delta t_w$  and  $\Delta \omega_w$  cannot both get arbitrarily small. Therefore the optimal choice with respect to time and frequency resolution is the Gauss-function g given in 5.11. It minimizes the product  $\Delta t_w \Delta \omega_w$ .

The STFT with the corresponding window

$$w(t) = \frac{1}{\sqrt[4]{\pi}} e^{-\frac{t^2}{2}}.$$
(2.15)

is also called "Gabor transform".

#### 2.1.3 Implementation with MATLAB and Visualization

For this section we assume the reader to have basic familiarity with MAT-LAB. In particular, knowledge of matrix and vector manipulations and functions like fft(f) (computation of the DFT of the sequence f) and ifft(F) (computation of the IDFT of the sequence F) is required.

We turn now to a short description of a MATLAB-implementation of the STFT-algorithm 2.8 – 2.10. Subsequently, the implementation of the reconstruction algorithm leading to 2.12 is discussed.

The corresponding MATLAB-m-files have been tested with MATLAB 6.5, release 13 and may be downloaded from the URL given in sect. 1.3.

#### **Fast STFT-Computation**

The algorithm is implemented in the function mystft.

A prototype call of this function reads

```
[t,y,matrix]=mystft(signal,T,stept,ommin,stepom,ommax,text);
```

We first give a short explanation of the input parameters. The string text allows for a comment to be included in the title of the graphics generated by the function. The sampled signal is represented by the vector signal, T denotes the sampling distance, ommin, stepom, ommax are minimum index, stepwidth and maximum index in the frequency sampling 2.8, for which the desired  $\hat{f}_w(\omega, t)$ -values are computed. Thus, in particular, the user must take care of the restriction

#### $\operatorname{ommax} \leq m_N$

with  $m_N$  defined by 2.9. N denotes the signal length. In order to reduce the computing effort, stept was introduced. For, e.g., stept=3, steps 2a) and 2b) of the algorithm are performed only for every 3rd value of the time vector t=(0:(N-1))\*T. The  $\hat{f}_w(\omega, t)$ -values for intermediate times are obtained by replicating the last computed values.

As a result of the algorithm the computed  $\hat{f}_w(\omega_k, t_i)$ -values are stored in the matrix matrix, with k numbering rows and i numbering columns. This

20 2 Continuous Analysis

matrix is visualized as, e.g., in Fig. 1.6 using the imagesc-function. Together with the signal y and the time vector t it is returned by the function.

The basic computation and replication code is listed below:

```
%Loop for increasing time
for index=1:stept:N,
    y=feval(window,(t-t(index))/sigma).*signal;
    column=conj((fft(y))');
    column=column(ommin:stepom:ommax);
    ifill=min(stept,N-index+1);
    for i=1:ifill,
        matrix(:,index+i-1)=column;
    end
```

end

The window function must be provided as a function file, its name is stored in the string window, sigma denotes an additional user control of the window width.

## Fast STFT-Reconstruction

The corresponding function reads myistft.

A prototype call:

```
[t,reco,matrix]=myistft(matrix,T,stept);
```

Here matrix denotes the result of an STFT-computation with mystft. Since for the reconstruction algorithm the complete frequency sampling 2.8 is required, in the preceding mystft-call only the signal, the sampling distance and the time increment should be specified, since the complete frequency sampling then is computed by default. Moreover, for the STFT-computation and the reconstruction obviously the same window function and corresponding width must be used, mystft and myistft request these entities from the user.

stept denotes the length of the reconstruction intervals in multiples of the sampling time T (cf. step 3 of the reconstruction algorithm). It is recommended to choose it identical with the time step parameter in the preceding mystft-call, provided the shifted window-values do not vanish on this interval (cf. step 3 of the reconstruction algorithm).

Thus a typical dialogue is as follows:

```
>> [t,y,matrix]=mystft(signal,.017,10);
Enter window (in quotes): 'gauss'
Window width: 1
>> [t,recostft,matrix]=myistft(matrix,.017,10);
Enter window (in quotes): 'gauss'
Window width: 1
>>
```

For more implementation details the reader is referred to the source file.

# 2.2 The Continuous Wavelet Transform (CWT)

#### 2.2.1 Definition, Computation and Reconstruction

#### Definition

The (continuous) wavelet transform has been listed already in sect. 1.2.4, formula 1.5. We shall now give the precise formulation using the symbol  $\psi$  for the wavelet in accordance with the general use (cf., e.g., [7]).

Choose a finite energy function  $\psi(t)$  fulfilling the "admissibility condition"

$$c_{\psi} := 2\pi \int_{-\infty}^{\infty} \frac{|\hat{\psi}(\omega)|^2}{|\omega|} d\omega < \infty.$$
(2.16)

Any finite energy function satisfying 2.16 will be called a "wavelet". Then the "continuous wavelet transform" (CWT) of the signal f(t) is denoted with  $L_{\psi}f(a,t)$  and reads

$$L_{\psi}f(a,t) = \frac{1}{\sqrt{c_{\psi}}} \frac{1}{\sqrt{|a|}} \int_{-\infty}^{\infty} \overline{\psi}\left(\frac{u-t}{a}\right) f(u) \, du \ (a \neq 0), (t \in \mathcal{R}).$$
(2.17)

Again, the overline denotes complex conjugation if  $\psi(t)$  is complex valued. Some remarks might be appropriate here:

- 1. When comparing formula 2.17 with formula 1.5, note that in the latter formula for simplicity reasons the constant factor  $\frac{1}{\sqrt{c_{\psi}}}$ , related to the admissibility condition 2.16, has been omitted.
- 2. For practically relevant wavelets admissibility condition 2.16 is fulfilled, when

$$\int_{-\infty}^{+\infty} \psi(t) dt = 0.$$
(2.18)

Thus, as stated in sect. 1.2.4,  $\psi$  will oscillate around the *t*-axis, since the contributions of positive and negative function values to the total area, bounded by the function graph and the *t*-axis, must cancel each other. Since, moreover,  $\psi(t)$  is of finite energy, for  $t \to \pm \infty$  the function  $\psi(t)$  will decrease rapidly. Both facts taken together explain the term "wavelet" for the function  $\psi(t)$ . The "Haar-wavelet" has already been mentioned in sect. 1.2.4. In sect. 2.2.2 it once more will be treated together with further wavelet examples.

#### 22 2 Continuous Analysis

3. As explained in sect. 1.2.4, the CWT 2.17 is a kind of multiresolution analysis, since  $L_{\psi}f(a,t)$  provides information about signal details of size  $\approx a$ . Correspondingly, a will be called "detail size" or "scale factor". As noted in 1.2.4, scale factors and frequencies are inversely proportional to each other. Since the proportionality constant in eq. 1.4 depends on the wavelet, the correct relation reads

$$a = \frac{\beta_{\psi}}{\omega}.$$
 (2.19)

Readers mainly interested in applications may skip the rest of this section and proceed from here to sect. 2.2.2 and then to sect. 2.2.4.

The rest of this section and sect. 2.2.3 require basic knowledge of Fourier transforms and the discrete Fourier transform (cf. appendix, sections 5.1 and 5.2, respectively).

Equation 2.17 may be rewritten as a convolution: With

$$\psi_a(t) := \overline{\psi\left(\frac{-t}{a}\right)} \tag{2.20}$$

we obtain

$$L_{\psi}f(a,t) = \frac{1}{\sqrt{c_{\psi}}} \frac{1}{\sqrt{|a|}} \int_{-\infty}^{\infty} \psi_a(t-u)f(u) \, du.$$

Then from the convolution theorem (eq. 5.5) we may conclude that

$$L_{\psi}f(a,t) \circ - \bullet \frac{1}{\sqrt{c_{\psi}}} \frac{1}{\sqrt{|a|}} \hat{\psi}_a(\omega) \hat{f}(\omega).$$
(2.21)

Similarly to eq. 2.5 in the STFT-case, eq. 2.21 is the key both for a fast CWT-computation-algorithm and for reconstructing the signal from the CWT.

#### Computation

Given a sampled signal  $\{f(kT_S)\}_{k=0}^{N-1}$  (cf. eq. 2.1), we will describe now an algorithm to compute a sampled version of the CWT. As a result, we will compute the sequence  $\{L_{\psi}f(a,kT_S)\}_{k=0}^{N-1}$  for  $a \neq 0$ . We describe now both the algorithm and the restrictions under which it is valid.

Assume f and  $\psi$  to be band-limited and let  $\omega_{max}^f$ ,  $\omega_{max}^{\psi}$  and  $\omega_{max}^{\psi_a}$  denote the maximum frequencies of f,  $\psi$  and  $\psi_a$ , respectively (cf. sect. 5.2). Analogously, let  $\omega_{max}$  denote the maximum frequency of  $L_{\psi}f(a,t)$ . Then it is not hard to show that

$$\omega_{max}^{\psi_a} = \frac{1}{|a|} \omega_{max}^{\psi}.$$

Now assume that the sampling is such that the Shannon condition 5.12 is valid both for f(t) and  $\psi_a(t)$ , i.e.

$$\omega_S > 2 \max\left(\omega_{max}^f, \frac{1}{|a|} \omega_{max}^\psi\right). \tag{2.22}$$

Then the Shannon condition automatically is satisfied for  $L_{\psi}f(a,t)$ :

$$\omega_S > 2\omega_{max}.\tag{2.23}$$

This is true, since by 2.21 the spectrum of  $L_{\psi}f(a,t)$  will vanish if the spectrum of  $\psi_a(t)$  or the spectrum of f(t) vanishes. Therefore

$$\omega_{max} = \min\left(\omega_{max}^f, \frac{1}{|a|}\omega_{max}^\psi\right)$$

and 2.22 then implies 2.23.

Thus, if 2.22 is valid, the spectra of f(t),  $\psi_a(t)$  and  $L_{\psi}f(a,t)$  can be computed with the DFT as described in theorem 5.1, eq. 5.16. This leads to the following

Fast algorithm for the computation of  $L_{\psi}f(a,t)$   $(a \neq 0)$ :

- 1. Define the sequence  $\{\psi_a(kT_S)\}_{k=0}^{N-1}$ , where  $\psi_a$  is defined in eq. 2.20.
- 2.20.
   Compute the DFT of this sequence resulting in a sequence {A<sub>k</sub>}<sup>N-1</sup><sub>k=0</sub>. Moreover, compute the DFT {F<sub>k</sub>}<sup>N-1</sup><sub>k=0</sub> of {f(kT<sub>S</sub>)}<sup>N-1</sup><sub>k=0</sub>.
   Apply the IDFT to the product sequence 1/√(c<sub>ψ</sub>|a|) {F<sub>k</sub>A<sub>k</sub>}<sup>N-1</sup><sub>k=0</sub> to

obtain  $\{L_{\psi}f(a, kT_S)\}_{k=0}^{N-1}$ . Schematically:

$$\frac{1}{\sqrt{c_{\psi}|a|}} \{F_k A_k\}_{k=0}^{N-1} \xrightarrow{IDFT} \{L_{\psi} f(a, kT_S)\}_{k=0}^{N-1}.$$
 (2.24)

We conclude this section with some remarks:

- 1. The algorithm can be performed for any scale factor  $a \neq 0$ . Practically relevant are positive scale factors; for the sampling considered here, one usually takes  $a = T_S, 2T_S, \ldots, (N-1)T_S$ .
- 2. Again, usually the sampling distance  $T_S$  is given a priori. Thus for small scale factors one must be aware that eq. 2.22 is not valid any more. Therefor the algorithm inevitably will lead to distortions for  $a \to 0$ .
- 3. In analogy to the STFT-algorithm, the remarks made after theorem 5.1 apply also here. Since finitely sampled signals never are band-limited in the strict sense, again "maximum frequencies" must be understood such that the respective spectra are sufficiently small outside  $\left[-\omega_{max}^{f}, \omega_{max}^{f}\right]$ and  $\left[-\omega_{max}^{\psi}, \omega_{max}^{\psi}\right]$ , respectively.

#### Reconstruction

In this subsection we shall consider the question how to reconstruct the original continuous-time finite-energy signal f(t) from the CWT  $L_{\psi}f(a,t)$ . Similarly to the STFT-case we start from the correspondence pair eq. 2.21.

An easy calculation shows how the Fourier transform of  $\psi_a(t)$  may be expressed in terms of the Fourier transform of  $\psi$ . Inserting the result in the correspondence pair 2.21 one obtains

$$L_{\psi}f(a,t) \circ - \bullet \sqrt{\frac{|a|}{c_{\psi}}} \overline{\hat{\psi}(\omega a)} \hat{f}(\omega).$$
(2.25)

Recall (cf. sect. 5.1) that this relation is only a shorthand notation for

$$\int_{-\infty}^{+\infty} L_{\psi} f(a,t) e^{-j\omega t} dt = \sqrt{\frac{|a|}{c_{\psi}}} \overline{\hat{\psi}(\omega a)} \hat{f}(\omega).$$
(2.26)

The detailed calculations in the following procedure are left to the reader. We shall only write down the basic steps and the ideas behind them:

1. Multiply both sides of eq. 2.26 with

$$\frac{\hat{\psi}(\omega a)}{|a|^{\frac{3}{2}}\sqrt{c_{\psi}}}.$$

and integrate with respect to a. Recalling definition 2.16 of  $c_{\psi}$ , the purpose of this procedure is to eliminate all terms depending on  $\hat{\psi}$  on the right-hand side.

After the multiplication and integration procedure, the new right-hand side will read

$$\frac{1}{2\pi}\hat{f}(\omega).$$

Equating this with the corresponding new left-hand side, we obtain a formula which computes  $\hat{f}(\omega)$  from the CWT  $L_{\psi}f(t,a)$ .

2. Applying now the Fourier inversion formula 5.2 to both sides we end up with the desired reconstruction formula. It reads

$$f(t) = \frac{2\pi}{\sqrt{c_{\psi}}} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} L_{\psi} f(a, u) \frac{1}{\sqrt{|a|}} \psi\left(\frac{t-u}{a}\right) \frac{duda}{a^2}$$
(2.27)

With the same reasoning as for the STFT reconstruction formula 2.11 we note that the above reconstruction integral is only of limited practical relevance if sampled signals and the corresponding CWT are given.

Instead - in analogy to the STFT procedure - one may obtain a reconstruction algorithm essentially by reversing the steps of the computation algorithm given above.

Assume that the sampling  $\{f(kT_S)\}_{k=0}^{N-1}$  (cf. eq. 2.1) of a continuous-time finite energy signal f(t) is given such that eq. 2.22 is fulfilled. Assume, moreover, that the algorithm described above has been performed. Thus after evaluating eq. 2.24, the sequence

$$\{L_{\psi}f(a, kT_S)\}_{k=0}^{N-1}$$

is given.

Then we use the correspondence pair eq. 2.21 again to reconstruct  ${f(kT_S)}_{k=0}^{N-1}$  as follows:

Fast CWT-reconstruction:

- 1. Compute the sequence  $\{\psi_a(kT_S)\}_{k=0}^{N-1}$ , where  $\psi_a$  is defined in eq. 2.20 and compute the corresponding DFT denoted with  $\{A_k\}_{k=0}^{N-1}$ .
- 2. Compute the DFT of the sequence  $\{L_{\psi}f(a, kT_S)\}_{k=0}^{N-1}$ . Let us denote it with  $\{B_k\}_{k=0}^{N-1}$ . 3. Compute the sequence  $\{F_k\}_{k=0}^{N-1}$  where

$$F_k = \sqrt{c_\psi |a|} \frac{B_k}{A_k}.$$
(2.28)

4. Then apply the inverse DFT (eq. 5.14) to this sequence, resulting in the sequence  $\{f(kT_S)\}_{k=0}^{N-1}$ . Schematically:

$$\{F_k\}_{k=0}^{N-1} \xrightarrow{IDFT} \{f(kT_S)\}_{k=0}^{N-1}$$

$$(2.29)$$

Note that eq. 2.28 requires  $A_k$  to be nonzero for all k. If this is not the case, steps 1 – 3 must be repeated for suitable *a*-values, such that  $\{F_k\}_{k=0}^{N-1}$ completely may be recovered before computing 2.29.

Nevertheless, if the scale factor a is such that  $A_k \neq 0$  (k = 0, ..., N - 1), the algorithm described above leads to a recovery of the sampled signal from the corresponding CWT from a single a-value! This is an indication for the fact that the information stored in the  $L_{\psi}f(a,t)$ -coefficients is highly redundant. We will reduce this redundancy later (cf. chap. 3).

MATLAB-implementations of the CWT-algorithms presented in this section are discussed in sect. 2.2.3.

#### 2.2.2 Wavelet Examples

As mentioned in sect. 2.2.1, most practically relevant wavelets satisfy the admissibility condition 2.16 if relation 2.18 is fulfilled. Generalizing this relation we shall construct "wavelets  $\psi(t)$  with M vanishing moments" (cf. eq. 2.31).

The practical implications of condition 2.31 will be illustrated below.

#### **Construction** principle

Let  $\Phi(t)$  denote a piecewise smooth function being *M*-times differentiable and satisfying some additional slight technical restrictions. Define the wavelet  $\psi(t)$  by

$$\psi(t) := \frac{d^M \Phi}{dt^M}(t). \tag{2.30}$$

Then one may show that  $\psi(t)$  has M vanishing moments, i.e.

$$\int_{-\infty}^{+\infty} \psi(t)t^m \, dt = 0 \ (m = 0, 1, \dots, M - 1).$$
(2.31)

The interested reader is encouraged to prove this in the exercises.

#### M = 1: The Haar-wavelet

Consider

$$\Phi_H(t) = \begin{cases} t & 0 \le t < \frac{1}{2} \\ 1 - t & \frac{1}{2} \le t < 1 \\ 0 & \text{else} \end{cases}$$

The corresponding wavelet is called "Haar-wavelet" and reads

$$\psi_H(t) := \frac{d\Phi_H}{dt}(t) = \begin{cases} 1 & 0 \le t < \frac{1}{2} \\ -1 & \frac{1}{2} \le t < 1 \\ 0 & \text{else} \end{cases}$$
(2.32)

Condition 2.31 for the Haar-wavelet reads

$$\int_{-\infty}^{+\infty} \psi_H(t) \, dt = 0.$$

#### M = 2: The Mexican-hat-wavelet

Here

$$\Phi_{MH}(t) = -\frac{2}{\sqrt{3}}\pi^{-\frac{1}{4}}e^{-\frac{t^2}{2}}$$

The corresponding wavelet is called "Mexican-hat-wavelet" and reads

$$\psi_{MH}(t) := \frac{d^2 \Phi_{MH}}{dt^2}(t) = \frac{2}{\sqrt{3}} \pi^{-\frac{1}{4}} (1 - t^2) e^{-\frac{t^2}{2}}$$
(2.33)

Condition 2.31 for the Mexican-hat-wavelet reads

$$\int_{-\infty}^{+\infty} \psi_{MH}(t) dt = \int_{-\infty}^{+\infty} \psi_{MH}(t) t dt = 0.$$

#### Plots

Both wavelets treated so far are shown in Fig. 2.2. The wavelets are plotted as solid lines, the respective  $\Phi$ -functions are dashed.

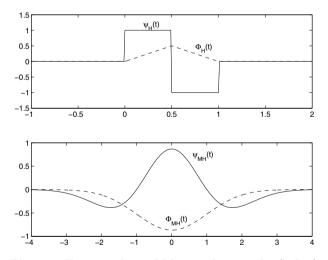


Fig. 2.2. Haar-wavelet and Mexican-hat-wavelet (below).

#### Practical implications of the vanishing moments condition

When for a given scale a and a given time t the CWT

28 2 Continuous Analysis

$$L_{\psi}f(a,t) = \frac{1}{\sqrt{c_{\psi}}} \frac{1}{\sqrt{|a|}} \int_{-\infty}^{\infty} \overline{\psi}\left(\frac{u-t}{a}\right) f(u) \, du$$

is computed, only those *u*-values will contribute, where  $\overline{\psi}\left(\frac{u-t}{a}\right)$  is nonzero. Assume that these values constitute an interval  $I_{(a,t)}$ . Now assume that  $\psi(t)$  has M vanishing moments and the signal f(u) on  $I_{(a,t)}$  may be modeled by a polynomial of degree k with k < M. Then one may show that  $L_{\psi}f(a,t) = 0$ . Again, the interested reader may work on the proof in the exercises.

This result can be interpreted as follows: Assume that every "smooth" part of the signal may be represented by a certain polynomial and signal changes are modeled by switching the polynomial to a new one. In other words: The signal is modelled by piecewise polynomial functions. Then signal changes will be well localized by the CWT provided the wavelet  $\psi(t)$  has enough vanishing moments. The reason is that in the smooth parts the CWT will vanish if the number of vanishing moments is large enough and the CWT will be concentrated around the time values, where signal changes (i.e. changes in the polynomials) occur.

This is illustrated in Fig. 2.3. Here the signal consists of two different straight lines (i.e. polynomials of degree k = 1) connected together at t = 5000. Recall (see above) that the Mexican-hat-wavelet has 2 vanishing moments, whereas the Haar-wavelet only has 1 vanishing moment. Thus, one expects that respective CWT should be better concentrated around the connection time for the Mexican-hat-wavelet than for the Haar-wavelet. This is clearly visible in Fig. 2.3. Note also that the Haar-wavelet leads to nonzero CWT values in the "smooth" signal parts in contrast to the Mexican-hat-wavelet.

#### The Morlet-wavelet

We close this section with the "Morlet-wavelet" which was among the first wavelets to be explored for signal analysis purposes [11].

It is complex valued and reads

$$\psi_M(t) = \pi^{-\frac{1}{4}} (e^{-j\alpha t} - e^{-\frac{\alpha^2}{2}}) e^{-\frac{t^2}{2}} (\alpha = \pi \sqrt{\frac{2}{\ln 2}}).$$
(2.34)

Since the corresponding wavelet transform  $L_{\psi_M} f(a, t)$  will be a complexvalued function, it may be decomposed into absolute value and phase corresponding to

$$L_{\psi_M} f(a,t) = |L_{\psi_M} f(a,t)| e^{j \Phi_M(a,t)}.$$
(2.35)

For possible applications of the phase function  $\Phi_M(a,t)$  the reader is referred to sect. 2.2.4.

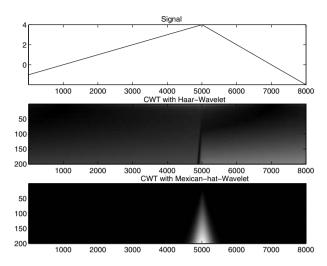


Fig. 2.3. Top: Signal. Middle: CWT with the Haar-wavelet. Bottom: CWT with the Mexican-hat-wavelet.

## 2.2.3 Implementation with MATLAB and Visualization

At the beginning of sect. 2.1.3 some preliminary statements concerning MAT-LAB have been made. They apply also to this section and will not be repeated here.

In this section we shall describe MATLAB-implementations both of the CWT-algorithm leading to 2.24 and of the corresponding reconstruction algorithm leading to 2.29. Again, the respective MATLAB-m-files have been tested with MATLAB 6.5, release 13 and may be downloaded from the URL given in sect. 1.3.

#### **Fast CWT-computation**

The corresponding function file reads mycwt. A sample function call is shown in the next line:

```
[t,y,matrix]=mycwt(signal,T,start,step,stop,comment);
```

Analogously to mystft the vector signal denotes the sampled signal, T the sampling distance, the string comment is included in the title of the graphics generated by the function. The parameters start, step and stop denote minimum index, increment and maximum index of the scale factors for which the CWT is computed, measured in multiples of T. Thus in MATLAB-notation the resulting scale values are given by the vector (start:step:stop)\*T.

As a result of the algorithm the computed  $L_{\psi}f(a_i, kT_S)$ -values are stored in the matrix matrix, with k numbering columns and i numbering rows. The  $a_i$ -values are the entries of the above vector containing the scale values. This matrix again is visualized as, e.g., in Fig. 1.6 using the imagesc-function. Since for complex-valued wavelets the phase plot of this matrix might be of interest (cf. sect. 2.2.4), the function allows both for a visualization of the absolute value of matrix and of its phase.

The basic computation code is listed below

```
%Initializations of scale factors and Fourier transform
a=(start:step:stop)*T;
rows=length(a);
yhat=fft(y);
%Matrix-Initialization
matrix=zeros(ze,n);
%Loop for increasing scale factors
for i=1:rows,
    psi_scale=conj(feval(wav,-t./a(i)));
    psi_scale_hat=fft(psi_scale);
    %Time translation such that minimal time=0
    trans=exp((-j*t(1)*(0:(N-1))*oms/N));
    %Fourier transform of wavelet transform;
    conv_hat=((yhat.*psi_scale_hat).*trans)/sqrt(a(i));
    matrix(i,:)=ifft(conv_hat);
end
```

end

The wavelet must be provided as a function file, whose name is stored in the string wav.

The code should be fairly self-explanatory for a reader used to MATLABnotation and having worked through sect. 2.2.1. A comment on the time translation vector **trans** should be added, however.

As mentioned in sect. 2.2.1 the CWT will be computed for the sampled time values  $0, T_S, 2T_S, \ldots, (N-1)T_S$ . Thus, in particular, all time values will be  $\geq 0$ . On the other hand, the computation of

$$\psi_a(t) := \overline{\psi\left(\frac{-t}{a}\right)}$$

is required. This implies that for wavelets, which are zero for  $t < 0 \psi_a(t)$  will vanish for t > 0, thus in particular for the above time sampling. The Haar-wavelet 2.32 is a simple example for such a wavelet. Therefore in mycwt the time vector t is designed such that t = 0 corresponds to the middle of the signal. Hence, the resulting CWT must be shifted with respect to the time domain such that after te shift the smallest t-value corresponds to t = 0. In the Fourier domain this time shift may be realized by a multiplication with

phase factors, which are collected in the vector trans. In the definition of this vector oms denotes the sampling circular frequency.

## Fast CWT-reconstruction

The reconstruction algorithm is implemented in the function file myicwt. A prototype call to this function reads

[t,reco,matrix]=myicwt(matrix,T,start,step,stop);

matrix is the result from a preceding call of mycwt, T again denotes the sampling distance. As with the mycwt-function from the respective input parameters the scale vector  $\mathbf{a}=(\mathtt{start:step:stop})*T$  is constructed. The entries of these vectors are scanned in order to select a component  $a_i$  such that the sequence  $\{A_k\}_{k=0}^{N-1}$  resulting from applying the DFT to the sequence  $\{\psi_{a_i}(kT_S)\}_{k=0}^{N-1}$  is nonzero for all k (cf. computation step 2.28 in the reconstruction algorithm). If such an  $a_i$  does not exist, the program will not work properly!

It is recommended to choose the scale-related input parameters start, step and stop identical to the respective parameters in the preceding mycwtcall. Thus, a typical dialogue reads:

```
>> [t,y,matrix]=mycwt(y,.01,1,1,200,'');
Enter wavelet (in quotes): 'mex_hat'
>> [t,reco,matrix]=myicwt(matrix,.01,1,1,200);
Enter wavelet (in quotes): 'mex_hat'
```

Note that obviously both in the mycwt- and the myicwt-call the same wavelet must be selected! In both functions the name of the wavelet function file is requested from the user. The reconstructed signal is stored in the vector reco.

## A note on the MATLAB Wavelet Toolbox

The MATLAB Wavelet Toolbox [24] provides the cwt-function in order to perform a CWT. Below a typical function call is shown:

```
matrix=cwt(y,1:1:200,'mexh');
```

The meaning of the respective parameters should be clear from the discussion above. Note that the start-, step- and stop-parameters from above must be merged to an input vector of the kind start:step:stop. Note also that the chosen wavelet must be specified by a corresponding input string, in the above example 'mexh' selects the Mexican-hat-wavelet 2.33.

As with all other features of the MATLAB Wavelet Toolbox the CWT also may be invoked from a graphical user interface which is launched by the wavemenu-command.

The other local transforms treated in this chapter (STFT, inverse STFT, inverse CWT) are not supported by the MATLAB Wavelet Toolbox.

## 2.2.4 Application: Detection of Signal Changes

In sect. 2.2.2 we modeled signals by piecewise polynomial functions. "Smooth" parts of the signal corresponded to time intervals, where the signal was represented by a fixed polynomial. The time values where two different polynomials are connected corresponded to "signal changes". In this section we will consider a more subtle example and discuss CWT-experiments with varying wavelets.

The model signal reads

$$f(t) = \begin{cases} \cos(t-\pi) & t < \pi\\ \cos(2t-2\pi) & t \ge \pi \end{cases}$$

Obviously, the first derivative  $\frac{df}{dt}(t)$  is continuous, whereas the second derivative is discontinuous at  $t = \pi$ . Thus, we have a kind of "hidden" signal change occurring not in the signal itself but in some higher derivative.

In the first experiment we computed the CWT of this signal with the complex Morlet-wavelet 2.34. Figure 2.4 visualizes the phase  $\Phi_M(a,t)$  of  $L_{\psi_M} f(a,t)$  (cf. eq. 2.35).

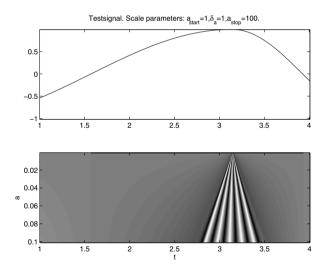
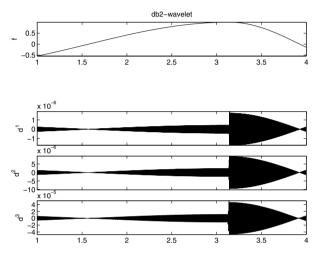


Fig. 2.4. CWT of the testsignal with the Morlet-wavelet 2.34, Phase plot.

The time where the signal change occurs is localized very clearly by straight lines converging to  $t = \pi$ . This qualitative behavior of the phase representation (convergence to points of signal changes) has been reported already very early (cf. the article of A. Grossmann, R. Kronland-Martinet and J. Morlet in [6]).

In chap. 3 we will introduce the famous family of Daubechies wavelets [7]. For a more detailed treatment we refer to this chapter. Here we only remark that the members of this family are numbered with dbn (n = 1, 2, 3, ...). An important aspect for signal analysis purposes is that dbn-wavelets have n vanishing moments (cf. eq. 2.31)!

Figure 2.5 shows the wavelet transform of the above signal with the db2wavelet for three increasing scale factors (cf. figure caption;  $T_S$  denotes the sampling distance).



**Fig. 2.5.** Top: Signal. Below: Wavelet transforms for  $a = T_S, a = 2T_S, a = 4T_S$  (from above). Wavelet: db2.

Figure 2.6 shows the same results using the db4-wavelet.

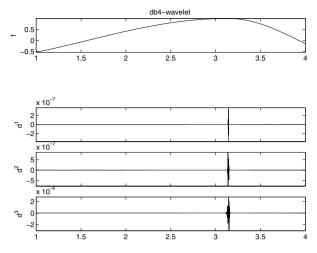
The figures confirm the qualitative behavior mentioned already in sect. 2.2.2: With increasing number of vanishing moments the respective wavelet transforms tend to be better localized around significant signal changes. Correspondingly, the vanishing in "smooth" signal regions is more pronounced for wavelets with a large number of vanishing moments.

Related to this is the fact that the dbn-wavelets tend to be "smoother" with increasing n. We will not stress these topics here; for an easy-to-read summary the reader is referred to [24].

## 2.3 Case Studies

## 2.3.1 Analysis of Sensor Signals

All signals discussed in this section were recorded in the sensor of a robot, carrying out a (light) arc welding process. The arc itself is used as a sensor measuring the distance from the welding tool to the workpiece.



**Fig. 2.6.** Top: Signal. Below: Wavelet transforms for  $a = T_S, a = 2T_S, a = 4T_S$  (from above). Wavelet: db4.

The results described in the following paragraphs have been obtained with the MATLAB-functions mystft and mycwt (cf. sections 2.1.3 and 2.2.3, respectively).

## STFT-application

The first example to be presented is the STFT analysis of such a signal. Here the welding robot performs an oscillatory motion which is transversal to the welding seam. Without going into details of the technology we mention that the frequency of this oscillation at  $t \approx 3$  sec should get doubled from  $\nu \approx 3$  Hz to  $\nu \approx 6$  Hz.

The sensor signal is shown in the upper part of Fig. 2.7.

It obviously is heavily corrupted with noise. The lower part of Fig. 2.7 shows the absolute values of the respective STFT signal. The STFT is coded with gray values as described in sect. 1.2.5. To simplify evaluation, the STFT was represented on the t- $\nu$ -plane in contrast to Fig. 1.6, where it was represented on the t- $\omega$ -plane.

Obviously, the frequency doubling mentioned above clearly is localized with the STFT. Moreover, the STFT shows how the corruption with noise leads to additional high-frequency components in the signal.

The window function used for the STFT-computation was the Gaussian window 2.15. Thus, actually the Gabor transform of the signal was computed.

## **CWT-applications**

Before discussing further examples of sensor signals occurring at an arc welding process, we remark that we will analyze signals stemming from three differ-

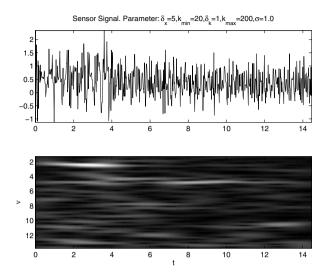


Fig. 2.7. STFT example: Signal (above) and STFT (below). Here the Gaborwindow (eq. 5.11) has been used.

ent technologies, the details of which cannot be explained here. We only note, that they are denoted with the abbreviations "MSG", "MAG" and "WIG", respectively. Apart from their differences they share a common property: usually the sensor performs an oscillatory motion transversally to the welding seam. For test purposes this oscillation may be switched off (third signal in Fig. 2.9). All time- and scale-values in the plots below are measured in seconds.

The result of the CWT analysis of the MAG- and MSG-signals, respectively, is depicted in Fig. 2.8. As expected, it shows bright stripes at scale factors associated to the wavelengths of the respective oscillations.

This remains true also for the three WIG-signals; the results of their respective CWT analysis are shown in Fig. 2.9. Note in particular the lack of the corresponding stripe for the third signal, where the oscillation was switched off.

Visual inspection alone does not reveal noticeable differences between the various technologies. Yet CWT analysis clearly indicates a difference: WIG-signals show bright peaks at scale  $a \approx 6$  sec. Indeed, a posteriori one might isolate corresponding "peak groups" when inspecting the signals from Fig. 2.9. This indicates that there exists a corresponding correlation in WIG-signals in contrast to MSG- and MAG-signals. Note that this correlation is present, even if the oscillation is switched off (cf. third signal in Fig. 2.9).

This example illustrates that the multiscale investigation of a signal, performed by the CWT, may offer valuable information concerning relevant parameters of the respective process behind the signal.

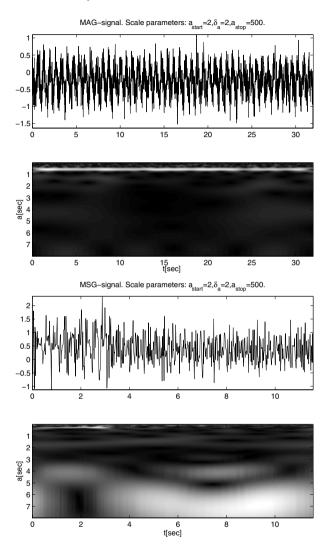


Fig. 2.8. Above: CWT of a MAG-signal. Below: CWT of a MSG-signal.

As a final remark we note that the wavelet used in the transforms visualized in Figs. 2.8 - 2.9 was the Morlet-wavelet 2.34.

## 2.3.2 Analysis and Classification of Audio Signals

In this section we describe the analysis of acoustic signals recorded during certain inspection procedures in the automotive industry<sup>1</sup>. In some applica-

<sup>&</sup>lt;sup>1</sup> The material presented in this subsection results from a cooperation with AUDI AG, a major German car manufacturer. It is published with kind permission.

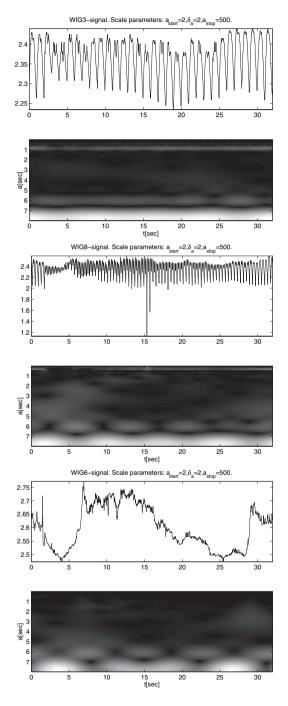


Fig. 2.9. CWT of three WIG-signals.

### 38 2 Continuous Analysis

tions such a signal may indicate the presence or absence of a defect; thus in this context it is used for diagnosis purposes. An example is described in [18]. In other cases the recorded signal simply should sound "pleasant" in order to fulfil comfort requirements. Both cases have in common that a human listener can discriminate "good" from "bad" signals. In the present section we describe the first steps towards a trainable classification scheme, based on a local analysis of such signals. This classification procedure should in principle be able to perform the above-mentioned discrimination of a human listener automatically. In order to compare the results of both techniques we used the CWT and the STFT.

We sketch now the classification procedure.

First the local transform is sampled on a grid of discrete values both with respect to time and scale (CWT) or time and frequency (STFT). For the CWT the sampling is illustrated in Fig. 2.10; in the upper half a typical acoustic signal is displayed.

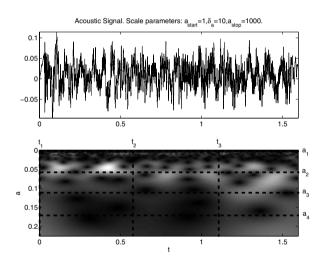


Fig. 2.10. CWT of an acoustic signal. Sampling for 3 time and 4 scale values.

In the next step from the sampled transform values a pattern vector is derived which serves as an input to the subsequent classification procedure. The classificator may be trained with "good" signals, leading to a geometrical description of the "good class" in the space of pattern vectors<sup>2</sup>. A given signal then is classified based on a certain notion of distance to that class.

Figure 2.11 shows some first classification results. Without going into details we mention that 15 signals were presented to the system. The first 5  $\,$ 

 $<sup>^2</sup>$  For an introduction to the basic principles of pattern recognition refer to, e.g., [37]

signals were "good"; in the figure they are denoted by io01.mat, ..., io05.mat. All other signals were "bad". For a signal to be qualified as "good" it must reach the maximum score of 9 in the bar graphs. The upper result was obtained with a CWT, the lower result with a STFT, the respective wavelets or window functions are indicated in the title line of the plots.

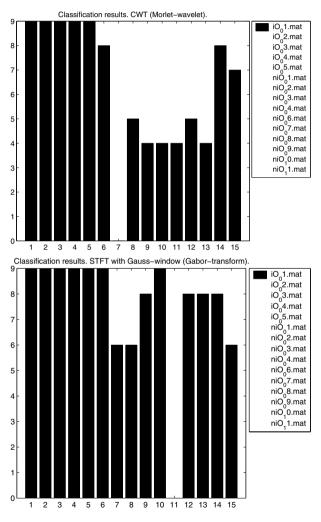


Fig. 2.11. Classification results of acoustic signals.

The results indicate that both transforms are useful in the context of our investigation, but the wavelet transform presumably leads to a clearer separation of "good" from "bad". Note that the Fourier-based procedure leads to some misclassifications (false-positives): The "bad" signals nio01.mat and nio06.mat, respectively, are classified as "good".

We finally remark that the results presented above were obtained with linearly sampled scale values (CWT) or frequency values (STFT), respectively. An option for future developments is the use of logarithmically sampled scales or frequencies. The resulting notions of "octaves" and "voices per octave" (for a short survey of these topics cf., e.g., [38]) proved to be successful when analyzing speech or music. Here we investigated signals resulting from a technical process, nevertheless the classification of these signals should perform similar to the human ear. In [4] a mathematical model of the cochlea, a spiral-like organ which is essential for hearing, is given, which relates the perception of the human ear to the CWT and incorporates a logarithmic response to frequencies!

## 2.4 Notes and Exercises

## Notes

For supplementary reading on the topics treated in this chapter the reader is referred to the bibliography. References [7] and [20] provide a thorough discussion of the mathematical foundations of the CWT. In particular, decay properties of the CWT, essential for quantifying its concentration property on the "important" signal regions are investigated. The STFT originally was introduced by D. Gabor [10]. The book [17] provides a nice overview on this topic. Short and comprehensive reviews of the CWT and STFT are also contained in [1] and [34].

## Exercises

1. a) Assume that the Fourier transform correspondence pair

$$y(t)\circ-\bullet\;\hat{y}(\omega)$$

is given. Show that then

$$y(t)t^{m} \circ - \bullet j^{m} \frac{d^{m}\hat{y}}{d\omega^{m}}(\omega).$$
(2.36)

Hint: Start from

$$\hat{y}(\omega) = \int_{-\infty}^{+\infty} y(t) e^{-j\omega t} dt$$

and differentiate both sides with respect to  $\omega$ .

b) Again assume that the Fourier transform correspondence pair

$$y(t) \circ - \bullet \hat{y}(\omega)$$

is given. Show that then

$$\frac{d^m y}{dt^m}(t) \circ - \bullet (j\omega)^m \hat{y}(\omega).$$
(2.37)

Hint: Start from

$$y(t) = \frac{1}{2\pi} \int_{-\infty}^{+\infty} \hat{y}(\omega) e^{+j\omega t} \, d\omega$$

and differentiate both sides with respect to t.

2. Given T with T > 0, the "box-window" of width T is defined as

$$w(t) = \begin{cases} 1 - \frac{T}{2} \le t \le \frac{T}{2} \\ 0 \text{ else} \end{cases}$$

Compute the respective localization parameters 5.6 - 5.9.

3. Consider a signal which may be represented by an integrable function f(t) and denote the first integral with F(t). Show that the CWT computed with the Haar-wavelet  $\psi_H$  defined in eq. 2.32 for a > 0 reads

$$L_{\psi_H} f(a,t) = \frac{1}{\sqrt{c_{\psi_H} a}} \left( 2F\left(t + \frac{a}{2}\right) - F(t) - F(t+a) \right).$$

This analytical result provides the possibility to check the fast CWTalgorithm described in sections 2.2.1 and 2.2.3:

Choose a sample function f(t) such that F(t) is known. Now choose the *a*-discretization and the *t*-discretization from sect. 2.2.1 and compute with MATLAB the matrix  $L_{\psi_H} f(a_i, t_k)$  using the above expression. Now visualize the absolute value with imagesc and compare the result with the visualization obtained from mycwt.

Note that, when gray value coding is used, **imagesc** automatically maps the lowest matrix value to black and the highest to white. Thus for visualization the factor  $c_{\psi_H}$ , which only plays the role of a global intensity factor, may be omitted when computing  $L_{\psi_H} f(a_i, t_k)$ .

- 4. Show that  $\psi(t) = \frac{d^M \Phi}{dt^M}(t)$  has M vanishing moments. Steps:
  - a) Show that  $\hat{\psi}(\omega) = (j\omega)^M \hat{\Phi}(\omega)$  using 2.37.
  - b) Show that  $\psi(t)t^m \circ \bullet j^m \frac{d^m \hat{\psi}}{d\omega^m}(\omega)$  using 2.36.
  - c) Conclude that  $\frac{d^m \hat{\psi}}{d\omega^m}(0) = 0$   $(m = 0, \dots, M 1)$  and relate this to the moment condition 2.31.

## 42 2 Continuous Analysis

- 5. Assume that the wavelet  $\psi$  has M vanishing moments. Assume, moreover, that f(t) is piecewise polynomial and the degree of the respective polynomials is below M. Show that then  $L_{\psi}f(a,t) = 0$  for "suitable" t and a. For the precise formulation refer to the section on practical implications of the vanishing moments condition (page 27 and following pages).
- 6. Try to implement your own MATLAB functions for computing the CWT and the STFT, respectively. For assistance refer to sections 2.1.1, 2.1.3, 2.2.1 and 2.2.3.

# The Discrete Wavelet Transform

## 3.1 Redundancy of the CWT and the STFT

The local transforms treated in chap. 2 map a continuous-time signal f(t) to functions of *two* variables: The STFT  $\hat{f}_w(\omega, t)$  and the CWT  $L_{\psi}f(a, t)$ , respectively. Thus the transformed signal in a certain sense is "larger" than the original signal.

This statement may be made more precise by restricting ourselves to the practically relevant case of a sampled signal and by applying the CWT to such a signal. The arguments in this introductory section correspondingly may be extended also to the STFT.

The sampled signal reads

$$f = \{f_k\}_{k=0}^{N-1} = \{f(kT_S)\}_{k=0}^{N-1}$$
(3.1)

(cf. eq. 2.1). It may be represented as a row vector of N real numbers. Applying the fast CWT-algorithm on page 23 for the scale factors  $a_1, \ldots, a_M$  to this vector we obtain the CWT-values

$$L_{\psi}f(a_i, kT_S) \ (k = 0, \dots, N-1; i = 1, \dots, M)$$

They may be arranged as an  $M \times N$ -matrix with *i* numbering rows and k numbering columns. In fact, all CWT-images discussed so far are visualizations of this matrix! Thus the original signal was represented by N real numbers whereas the transform is represented by MN numbers (which even may be complex, if the wavelet  $\psi$  is complex). Storing and transmitting the transformed signal on a computer network therefore requires roughly M times of the corresponding capacities for the original signal!

As we have seen in capter 2, this additional amount of data may be useful when *analyzing* the signal, i.e., looking for signal changes, dominant scales and/or frequencies etc. When performing such signal analysis tasks, the reconstruction formulae of sect. 2.2.1 usually are not needed. They only serve as an indication that no signal information is "lost", when applying the CWT. The application discussed in chap. 4 is signal *compression*. Here the wavelet transform can be taken as the first step in a compression procedure and this step has the following desirable properties:

- 1. If the original signal consists of N real values, the transformed signal also should consist of only N real values!
- 2. The transformed signal values after certain manipulations should allow for a representation with as few bits as possible. We shall not discuss the details here, they will be treated in chap. 4.
- 3. In the "decompression step" the signal should then be reconstructed from the N manipulated transform values!

Thus, for signal compression purposes we have a different intention; the task may be formulated as follows:

Is it possible to find wavelets  $\psi$  such that the CWT of f is computed on a discrete subgrid of the *t*-*a*-plane only, resulting in (roughly) Ntransform values  $L_{\psi}f(a_i, t_k)$  such that the original row vector f may be reconstructed from these values?

An inspection of the fast CWT-reconstruction algorithm described on page 25 shows the redundancy of the CWT  $L_{\psi}f(a,t)$ . If the scale factor a and the wavelet  $\psi$  are chosen such that in formula 2.28  $A_k \neq 0$  for all k, then the sampled signal  $\{f(kT_S)\}_{k=0}^{N-1}$  may be recovered by formula 2.29. The left part of Fig. 3.1 illustrates this situation: For recovering f not all values of the transform (i.e., the  $M \times N$ -matrix mentioned above) are needed; instead, one single row corresponding to the chosen a-value suffices. In the figure the matrix is visualized by the rectangular area, the selected row by the horizontal line. Similarly, for the STFT applied to f under proper conditions it suffices to pick out a single column from the corresponding  $N \times N$ -matrix of STFT values in order to reconstruct f. This can be concluded by inspecting the STFT-reconstruction algorithm – in particular eq. 2.12 – and is illustrated in the right part of Fig. 3.1.

In both cases only N transform values are needed to recover the original signal f, which is a strong indication for the fact that the information stored in the full CWT-matrix or STFT-matrix, respectively, is redundant.

In fact this is true: Mathematically two different CWT-transform values  $L_{\psi}f(a_1, t_1)$  and  $L_{\psi}f(a_2, t_2)$  are related with each other by the "reproducing kernel condition". An analogous statement holds for the STFT. For details refer to, e.g., [20].

The task formulated above, i.e., selecting a proper discrete subset of the t*a*-plane for the CWT (of the t- $\omega$ -plane for the STFT, respectively) such that f completely can be reconstructed from the respective transform values on these subsets only, can be solved. In the general case it leads to the notion

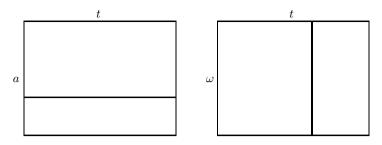


Fig. 3.1. Left: Under certain conditions the sampled signal f may be reconstructed from the corresponding CWT-matrix by selecting one suitable scale factor. Right: Under certain conditions the sampled signal f may be reconstructed from the corresponding STFT-matrix by selecting one suitable time value.

of "frames" [7], for wavelets the most popular version is the "discrete wavelet transform" (DWT) described in this chapter. In contrast to the situation illustrated in the left part of Fig. 3.1 the DWT will be a real "multiresolution analysis" of the signal, since the DWT-values will not be computed for a single scale factor only.

In the next section we shall discuss the simplest version of such a transform. We shall work out the aspects, which may be generalized and describe the general DWT-procedure in sect. 3.3. Section 3.4 is devoted to fast algorithms and related implementations with the MATLAB Wavelet Toolbox. In sect. 3.5 we will adopt a viewpoint on the developed techniques, which serves as a preparation for the compression procedures described in chap. 4. In sect. 3.6 some further applications are discussed, notes and exercises finally are collected in sect. 3.7.

Throughout this chapter we will consider sampled signals as defined in eq. 3.1. As indicated there, the k-th signal value for simplicity reasons mostly will be denoted with  $f_k$  instead of  $f(kT_S)$ . Thus, sampled signals are represented as finite sequences of numbers as indicated in table 5.2 (cf. sect. 5.5). They are always assumed to be sampled with a sampling distance  $T_S$ , even if this is not written down explicitly.

For simplicity in the next sections we shall assume that

$$T_S = 1. \tag{3.2}$$

The results obtained there may be extended to an arbitrary sampling distance  $T_S$  very easily and we shall indicate how.

## 3.2 The Haar-System

In this section we shall develop the DWT for the Haar-wavelet (cf. eq. 2.32). It will turn out that the resulting transform is linked to the corresponding CWT

as defined in eq. 2.17. Since the CWT can be applied only to a continuoustime signal f(t), whereas we start from a discretely sampled signal as defined in eq. 3.1, it will be necessary to construct a continuous-time signal f(t) from the discretely sampled signal  $\{f_k\}_{k=0}^{N-1}$ . We shall show in the first subsection how this can be done and how to develop the DWT for f(t). It will turn out that the notion of continuous-time signals may be omitted and the DWT may be formulated in a purely discrete manner, i.e., for sequences of numbers. The resulting facts and notions are collected in the second subsection.

As stated already after eq. 2.1, in order to keep notation simple we shall use the symbol f both to represent either a continuous-time signal or a discretely sampled signal. The actual meaning always shall be clear from the context. In sect. 3.2.1 we shall introduce some more continuous-time functions, which are uniquely connected with sequences. Here the same convention holds.

## 3.2.1 Continuous-Time Functions

Consider the following discretely sampled signal of length 8:

$$f = \{f_k\}_{k=0}^7 = \{8, 4, 6, 8, 9, 7, 2, 4\}$$
(3.3)

Moreover, we need the following function, which is called "scaling function":

$$\phi_H(t) := \begin{cases} 1 \ 0 \le t < 1\\ 0 \ \text{else} \end{cases}$$
(3.4)

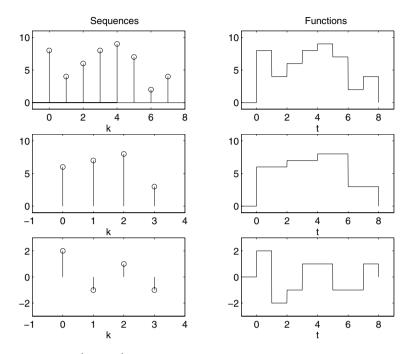
We shall see later that to each wavelet for which a DWT may be constructed, there belongs a unique scaling function. This explains the index Hin the above expression, since the scaling function defined by 3.4 belongs to the Haar-wavelet  $\psi_H$  (cf. eq. 2.32). We shall see later how this relation may be formulated mathematically.

The scaling function is used to build a continuous-time signal f(t) from the sequence  $\{f_k\}$ :

$$f(t) := \sum_{k=0}^{7} f_k \phi_H(t-k).$$
(3.5)

The top row of Fig. 3.2 shows the sequence  $\{f_k\}$  (left) and the corresponding function f(t) (right). It is a superposition of shifted versions of the scaling function 3.4, each multiplied with the corresponding sequence element. Thus, the continuous-time signal f(t) will be piecewise constant on intervals of length one, it is discontinuous at the sampling times.

In the next step we shall construct a "coarser version"  $f^1$  of the sequence f by computing the arithmetic mean of neighboring sequence elements as indicated below:



**Fig. 3.2.**  $f, f^1$  and  $d^1$  represented as sequences (left) and functions (right).

$$f^{1} = \{f_{k}^{1}\}_{k=0}^{3} = \left\{\frac{f_{0} + f_{1}}{2}, \frac{f_{2} + f_{3}}{2}, \frac{f_{4} + f_{5}}{2}, \frac{f_{6} + f_{7}}{2}\right\} = \{6, 7, 8, 3\}.$$
 (3.6)

Obviously this computation step leads to a length of  $f^1$ , which is half of the length of the original signal f. From this new sequence we shall construct a corresponding continuous-time signal  $f^1(t)$ , which again is piecewise constant, but now on intervals of length 2! Thus it covers the same time interval as f(t). Similarly to the procedure described for f(t) it is constructed from rescaled and shifted versions of the scaling function, multiplied with the corresponding sequence elements:

$$f^{1}(t) := \sum_{k=0}^{3} f_{k}^{1} \phi_{H} \left(\frac{t}{2} - k\right).$$
(3.7)

Both the sequence  $f_k^1$  and the corresponding continuous-time function  $f^1(t)$  are shown in the second row of Fig. 3.2. The coarsening procedure described above for sequences leads to an analogous procedure for the corresponding continuous-time signals:  $f^1(t)$  is obtained from f(t) by combining two neighboring steps of width 1 into one step of width 2. The new step height is obtained from the two former step heights by computing the arithmetic mean.

The right plot in the third row of Fig. 3.2 shows the difference signal of f(t) and  $f^{1}(t)$ , which we shall call  $d^{1}(t)$ :

$$d^{1}(t) := f(t) - f^{1}(t).$$
(3.8)

This signal also is called "detail signal". The reason is that, by definition, it represents the signal contributions which are needed to recover the original signal f(t) from the coarsened version  $f^1(t)$ .

An inspection of the plot shows that on the intervals [0, 2], [2, 4], [4, 6]and [6, 8] the detail signal is of mean zero: A step of a certain height and width 1 is followed by a step of the same magnitude and width, but with inverted sign. This is an immediate consequence of the coarsening procedure discussed above, leading from f(t) to  $f^1(t)$ . The Haar-wavelet (cf. eq. 2.32) was displayed in the upper part of Fig. 2.2. It consists of exactly two steps of width  $\frac{1}{2}$  and the same magnitude but with inverted sign. Therefore,  $d^1(t)$ can be written as a superposition of shifted Haar-wavelets, whose width is doubled and which are multiplied with a proper factor. These factors can be obtained directly by inspecting the graph of  $d^1(t)$ :

$$d^{1}(t) = 2\psi_{H}\left(\frac{t}{2}\right) - \psi_{H}\left(\frac{t}{2} - 1\right) + \psi_{H}\left(\frac{t}{2} - 2\right) - \psi_{H}\left(\frac{t}{2} - 3\right).$$

We shall formulate this relation in a slightly more general fashion. Define the sequence

$$d^{1} := \{d^{1}_{k}\}_{k=0}^{3} = \{2, -1, 1, -1\}.$$
(3.9)

Then  $d^1(t)$  may be written down similarly to  $f^1(t)$  (cf. eq. 3.7). But here we expand with respect to  $\psi_H$  instead of  $\phi_H$  and the expansion coefficients are given by the sequence  $\{d_k^1\}$  instead of  $\{f_k^1\}$ :

$$d^{1}(t) = \sum_{k=0}^{3} d_{k}^{1} \psi_{H} \left(\frac{t}{2} - k\right).$$
(3.10)

The sequence  $\{d_k^1\}_{k=0}^3$  is displayed in the left plot in the third row of Fig. 3.2. We obtained it by defining and investigating the continuous function  $d^1(t)$ . It is important to realize that this sequence also can be obtained directly from the original discrete signal  $\{f_k\}_{k=0}^7$ .

$$\{d_k^1\}_{k=0}^3 = \left\{\frac{f_0 - f_1}{2}, \frac{f_2 - f_3}{2}, \frac{f_4 - f_5}{2}, \frac{f_6 - f_7}{2}\right\} = \{2, -1, 1, -1\}.$$
 (3.11)

Note the similarity of this prescription to eq. 3.6!

We are now ready to formulate the connection of the coarsening and detailing procedure discussed so far to the CWT as defined in eq. 2.17:

$$L_{\psi_H} f(2,2k) = \sqrt{\frac{2}{c_{\psi_H}}} d_k^1 \ (k = 0,\dots,3).$$
(3.12)

The reader is invited to check this in the exercises.

It should be clear how the results and formulae of this section may be extended to a general discrete signal of length N. Note that via eq. 3.12 the CWT-values for the Haar-wavelet may be computed by a simple summation formula as given by eq. 3.11, i.e., without evaluating and approximating the integral in formula 2.17! But, as we have seen, in this way we obtain  $L_{\psi_H} f(a, t)$  only for a = 2 and t = 2k (k = 0, 1, ...).

We conclude this section with a collection of the results.

Fast algorithm for the computation of the CWT with the Haarwavelet:

- 1. Start from the discrete signal  $\{f_k\}$ , the Haar-wavelet  $\psi_H$  (cf. 2.32) and the corresponding scaling function  $\phi_H$  (cf. 3.4). Define the continuous-time signal f(t) as in 3.5.
- 2. Compute the sequences  $\{f_k^1\}$  and  $\{d_k^1\}$  as in eqs. 3.6 and 3.11, respectively.
- 3. Then the CWT with the Haar-wavelet, applied to f(t), for a = 2 and t = 2k (k = 0, 1, ...) may be evaluated via eq. 3.12.
- 4. f(t) can be reconstructed as follows: Define the coarsened signal  $f^{1}(t)$  and the detail signal  $d^{1}(t)$  as in eqs. 3.7 and 3.10, respectively. Then  $f(t) = f^{1}(t) + d^{1}(t)$ .

This procedure is a one-step-version of the "discrete wavelet transform" (DWT), carried out with the Haar-wavelet. We note already here that the computation may be iterated, starting with  $f^1(t)$  instead of f(t). This explains why we speak of a one-step-version. We shall describe the iteration procedure in some more detail later.

In Fig. 3.3 we visualize the subset of the *t*-*a*-plane, on which the CWTvalues are computed via the DWT. Note that f(t) may be reconstructed from the CWT-values on this subset together with the coarsened signal  $f^{1}(t)$ !

### 3.2.2 Sequences

The computation procedure given in eq. 3.6, leading from  $\{f_k\}$  to  $\{f_k^1\}$ , may be represented by the diagram shown in Fig. 3.4. The top row symbolizes the input sequence  $\{f_k\}$  (cf. 3.3), the bottom row the output sequence  $\{f_k\}$ .

Sequence  $f^1$  is obtained as follows: The elements of the input sequence pointing toward a certain element of the output sequence with an arrow are multiplied with the corresponding numbers and added at the node, where the arrows meet. Thus, e.g.  $f_0^1 = f_0 \frac{1}{2} + f_1 \frac{1}{2}$ . The diagram illustrates the 503 The Discrete Wavelet Transform

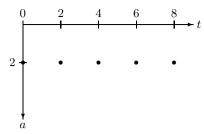


Fig. 3.3. One-step-version of the DWT: Subset of the *t-a*-plane.

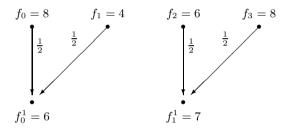


Fig. 3.4. Visualization of eq. 3.6.

computation of the first two elements of  $\{f_k^1\}$ . It is clear how to proceed with the remaining elements.

Comparing Fig. 3.4 with Fig. 5.6 we conclude that the computation of  $f^1$ may be written as

$$f^1 = Hf. aga{3.13}$$

As explained in sect. 5.3 (cf. in particular eqs. 5.17 and 5.20) the sequence  $f^1$  is obtained from f by a two-step procedure: First a digital filter with coefficients  $\{h_0 = \frac{1}{2}, h_1 = \frac{1}{2}\}$  is applied, subsequently the signal is downsampled, i.e., only every second sequence element is kept. This procedure is denoted with the symbol H, note in particular, that the number of sequence elements (the length of the signal) is divided by two.

The computation of sequence  $d^1$  was illustrated in eq. 3.11. It is now obvious that similarly to above we may write

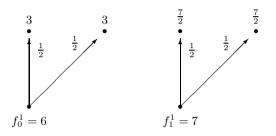
$$d^1 = Gf. \tag{3.14}$$

Again G denotes filtering followed by subsampling, but now the corresponding filter coefficients read  $\{g_0 = \frac{1}{2}, g_1 = -\frac{1}{2}\}.$ 

Equations. 3.13 and 3.14 act on sequences only. The corresponding continuoustime signals  $f^{1}(t)$  and  $d^{1}(t)$ , respectively, are not needed at all. But if we want

to reconstruct the discrete signal  $f = \{f_k\}$  from these sequences by the scheme described on page 49 (cf. in particular step 4 of this scheme), we must introduce the continuous-time functions  $f^{1}(t)$  and  $d^{1}(t)$  in order to recover the continuous-time signal f(t). From this signal the original discrete signal then may be recovered by properly evaluating the corresponding expansion formula 3.5. We will show now that this is not necessary. Instead, it will be possible to recover the original sequence by applying dual filters as described in sect. 5.3.

We start from sequence  $\{f_k^1\}_{k=0}^3$  (cf. eq. 3.6) and compute a new sequence  $\{F_k^1\}_{k=0}^7$  as illustrated in Fig. 3.5.



**Fig. 3.5.** Computation of the first four elements of  $H^*f^1$ .

The computation of  $F_0^1, \ldots, F_3^1$  is indicated in the diagram, the remaining elements are computed analogously. Thus finally we obtain

$$\{F_k^1\}_{k=0}^7 = \left\{3, 3, \frac{7}{2}, \frac{7}{2}, 4, 4, \frac{3}{2}, \frac{3}{2}\right\}.$$

The interpretation of diagram 3.5 is similar to diagram 3.4. The difference is that now the input sequence is symbolized by the bottom row and the output sequence by the top row. Obviously the length of the input sequence is doubled.

The indicated procedure is exactly the application of a dual filter as explained in sect. 5.3 (cf. Fig. 5.7). Thus we may write compactly

$$F^1 = H^* f^1, (3.15)$$

where the filter coefficients associated to the dual filter  $H^*$  read  $\begin{cases} h_0 = \frac{1}{2}, h_1 = \frac{1}{2} \\ \text{Analogously, starting from } \{d_k^1\}_{k=0}^3 \text{ (cf. eq. 3.11) we may compute } \end{cases}$ 

$$D^1 = G^* d^1, (3.16)$$

where the filter coefficients associated to the dual filter  $G^*$  read  $\left\{g_0 = \frac{1}{2}, g_1 = -\frac{1}{2}\right\}$ . The reader may verify that

52 3 The Discrete Wavelet Transform

$$\{D_k^1\}_{k=0}^7 = \left\{1, -1, -\frac{1}{2}, \frac{1}{2}, \frac{1}{2}, -\frac{1}{2}, -\frac{1}{2}, \frac{1}{2}\right\}.$$

Adding both sequences we obtain

$$\{(F^1 + D^1)_k\}_{k=0}^7 = \left\{4, 2, 3, 4, \frac{9}{2}, \frac{7}{2}, 1, 2\right\} = \frac{1}{2}\{f_k\}_{k=0}^7.$$

Thus, apart from the factor  $\frac{1}{2}$ , we have reconstructed the original discrete signal  $\{f_k\}$  by adding sequences  $F^1$  and  $D^1$ . Remembering that both sequences were obtained by applying dual filters we rewrite this equation as

$$f = 2(H^*f^1 + G^*d^1). (3.17)$$

This is the announced reconstruction formula: We obtain the original sequence directly from sequences  $f^1$  and  $d^1$ , respectively, by applying dual filters! Thus the discrete signal may be reconstructed without intermediately computing the corresponding continuous-time signal f(t). Remembering that  $f^1 = Hf$  and  $d^1 = Gf$  (cf. eqs. 3.13 and 3.14) we may rewrite 3.17 even more compactly:

$$2(H^*H + G^*G) = Id. (3.18)$$

The *Id*-symbol denotes the identity, leaving sequences unchanged. We finally summarize the results.

One-step-version of the DWT with the Haar-wavelet Let  $\{f_k\}$  denote the original discrete signal. H and G denote filters with subsampling and coefficients

$$\left\{h_0 = \frac{1}{2}, h_1 = \frac{1}{2}\right\}$$

and

$$\left\{g_0 = \frac{1}{2}, g_1 = -\frac{1}{2}\right\},\,$$

respectively.  $H^*$  and  $G^*$  denote the corresponding dual filters.

- 1. Decomposition
  - a) Compute the sequences  $f^1 = Hf$  and  $d^1 = Gf$ .
  - b)  $f^1$  then is a coarser version of the original signal,  $d^1$  stores the details needed to recover f.
- 2. Reconstruction
  - a) Compute and add the sequences  $H^*f^1$  and  $G^*d^1$ .
  - b) Multiply the resulting sequence with 2.

Let us close this section with some concluding remarks:

- The considerations of this section were carried out with the example signal 3.3 but may be generalized to arbitrary discrete signals of length N.
- 2. The procedure formulated above is purely discrete, i.e., does not rely on continuous-time signals. Thus it suffices for digital signal processing purposes.

The relation to the CWT may be established by introducing the continuoustime signal f(t) corresponding to eq. 3.5 and considering eq. 3.12.

3. In order to simplify terminology, we shall speak of  $H, H^*, G, G^*$  as "filters". From the respective symbol it will be clear that we actually mean filtering combined with down- or upsampling, respectively.

## 3.3 Generalization to Daubechies-Wavelets

Before generalizing the results of the preceding section to wavelets different from the Haar-wavelet we slightly modify the filters  $H, H^*, G, G^*$  introduced above. Let us assume that the corresponding coefficients read

$$\left\{h_0 = \frac{\sqrt{2}}{2}, h_1 = \frac{\sqrt{2}}{2}\right\}$$
(3.19)

and

$$\left\{g_0 = \frac{\sqrt{2}}{2}, g_1 = -\frac{\sqrt{2}}{2}\right\},\tag{3.20}$$

instead of the values used so far. This means that we multiply each of the filters with  $\sqrt{2}$  without changing the symbols. Then the reconstruction equation 3.18 gets even more simple:

$$H^*H + G^*G = Id. (3.21)$$

Correspondingly, the final multiplication with 2 in the reconstruction procedure (step 2 b in the scheme on page 52) can be omitted.

Decomposition and reconstruction usually are represented by the diagrams in eq. 3.22.

The left diagram represents decomposition, the right diagram reconstruction. The  $\oplus$ -symbol denotes the addition of the input sequences  $G^*d^1$  and  $H^*f^1$ .

#### 54 3 The Discrete Wavelet Transform

We summarize the results obtained so far:

1. Starting from coefficients 3.19 and 3.20, respectively, we constructed filter pairs  $H, H^*$  and  $G, G^*$  such that

$$\sum_{k} h_k = \sqrt{2} \tag{3.23}$$

$$\sum_{k} g_k = 0 \tag{3.24}$$

$$H^*H + G^*G = Id (3.25)$$

Note that, though never mentioned explicitly, eqs. 3.23 - 3.24 obviously are satisfied.

- 2. Given a discrete signal  $f = \{f_k\}$  a one-step-version of a DWT may be performed as indicated in eq. 3.22.
- 3. Defining the continuous-time signal  $f(t) = \sum_k f_k \phi_H(t-k)$ , the sequence  $d^1 = \{d_k^1\}$  is related to the the CWT of f(t), carried out for the Haarwavelet  $\psi_H$ , similarly to eq. 3.12. Note that with coefficients 3.20 this equation must be modified to

$$L_{\psi_H} f(2,2k) = \sqrt{\frac{1}{c_{\psi_H}}} d_k^1$$
 (3.26)

since, as explained above, the original g-coefficients were multiplied with  $\sqrt{2}$ .

For simplicity and future reference we shall denote this connection between a purely discrete viewpoint (signals are sequences) and a continuous viewpoint (signals are continuous-time functions) by

$$(H,G) \leftrightarrow (\phi_H,\psi_H).$$
 (3.27)

The dual filters  $(H^*, G^*)$  are omitted in this shorthand notation, since they are uniquely related to (H, G).

In sect. 2.2.2 we noted that the Haar-wavelet has only one vanishing moment. We also gave arguments there that for signal analysis purposes it is desirable to have as many vanishing moments as possible: For a signal which may be modeled by piecewise polynomial functions we noted that the CWT vanishes, if the wavelet has M vanishing moments and the highest degree k of the polynomials satisfies k < M. So the CWT is concentrated around signal regions which can *not* be modeled in this way. This is important for detecting and localizing signal changes. It moreover is useful for a property called "energy compaction" [38], a basic prerequisite for applying wavelets to signal compression. Later we shall discuss related examples <sup>1</sup>.

<sup>&</sup>lt;sup>1</sup> For the vanishing moments condition refer to eq. 2.31. The practical implications of this condition are discussed in more detail on pages 27 – 29. For an illustration refer to Fig. 2.3.

Therefore it is a natural question to ask if the procedure given above can be generalized:

Is it possible to develop a one-step DWT, i.e., to design filters satisfying eqs. 3.23 - 3.25, which are related to a scaling function  $\phi$  and a wavelet  $\psi$  as indicated in 3.27, such that  $\psi$  has more vanishing moments than  $\psi_H$ ?

This problem has been solved completely by Ingrid Daubechies [7]. She showed that equations 3.23 - 3.25 together with additional restrictions on the filter coefficients, reflecting in particular the vanishing moments condition, lead to a *family* of filters and corresponding scaling functions and wavelets. This family usually is called the "family of Daubechies-wavelets" and each member of this family is labelled by a natural number n = 1, 2, 3, ... It turns out that for each member of this family it is sufficient to specify

It turns out that for each member of this family it is sufficient to specify only the filter coefficients of H. The coefficients of G then can be computed according to eq. 3.28, in particular the number of g-coefficients will be identical with the number of h-coefficients. Analogously, it will be sufficient to specify the corresponding scaling function  $\phi$ , since then  $\psi$  also is determined (cf. eq. 3.29). For a given n the scaling functions and wavelets are denoted with dbn, the related filters will have 2n coefficients.

After these preliminaries concerning notation and basic properties of the family of Daubechies-wavelets we can give the answer to the question from above:

## dbn-wavelets have n vanishing moments!

In Table 3.1 we indicate the coefficients for the first three members of the family.

**Table 3.1.** Filter coefficients related to Daubechies-wavelets for n = 1, 2, 3.

$n \{h_k\}$
$1 \{h_k\}_{k=0}^1 = \left\{\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}\right\}$
$2 \{h_k\}_{k=0}^3 = \left\{\frac{1-\sqrt{3}}{4\sqrt{2}}, \frac{3-\sqrt{3}}{4\sqrt{2}}, \frac{3+\sqrt{3}}{4\sqrt{2}}, \frac{1+\sqrt{3}}{4\sqrt{2}}\right\}$
$3 \{h_k\}_{k=0}^5 = \{0.332671, 0.806892, 0.459878, -0.135011, -0.085441, 0.035226\}$

The corresponding g-coefficients may be computed according to

$$g_k = (-1)^k h_{1-k} (3.28)$$

(cf. [7]).

#### 3 The Discrete Wavelet Transform 56

The simplest member of the family of Daubechies-wavelets is db1. The corresponding scaling function and wavelet are given by the Haar-scaling function  $\phi_H$  and the Haar-wavelet  $\psi_H$ , respectively. Indeed, the related filters have 2 coefficients (cf. eqs 3.19 and eqs 3.20) and the *g*-coefficients satisfy the relations  $g_0 = \frac{\sqrt{2}}{2} = h_1$  and  $g_1 = -\frac{\sqrt{2}}{2} = -h_0$  in accordance with eq. 3.28. Likewise, for n = 2 eq. 3.28 implies

$$\{g_k\}_{k=-2}^1 = \left\{\frac{1+\sqrt{3}}{4\sqrt{2}}, -\frac{3+\sqrt{3}}{4\sqrt{2}}, \frac{3-\sqrt{3}}{4\sqrt{2}}, \frac{\sqrt{3}-1}{4\sqrt{2}}\right\},\$$

as easily may be verified. Relations 3.23 and 3.24 are also immediately checked for n = 2.

The wavelets may be computed from the respective scaling functions by the "scale-equation" (cf. [7]):

$$\psi(t) = \sqrt{2} \sum_{k} g_k \phi(2t - k).$$
 (3.29)

Again this relation is easily verified for the Haar-wavelet. With coefficients 3.20 the equation may be simplified as follows:

$$\psi_H(t) = \sqrt{2} \sum_{k=0}^{1} g_k \phi_H(2t-k)$$
  
=  $\sqrt{2} \left( \frac{\sqrt{2}}{2} \phi_H(2t) - \frac{\sqrt{2}}{2} \phi_H(2t-1) \right)$   
=  $\phi_H(2t) - \phi_H(2t-1).$ 

Indeed (cf. Fig. 2.2) the Haar-wavelet is composed from two shrunk versions of the scaling function on the intervals  $[0, \frac{1}{2})$  and  $[\frac{1}{2}, 1)$ , respectively, the latter one with inverted sign.

## 3.3.1 From Filters to Functions

In the preceding section we specified the filter coefficients for the first three members of the family of Daubechies-wavelets. But up to now only for db1, i.e., for the Haar-system, we know both the filter coefficients and the corresponding wavelet and scaling function.

In this section we describe an iterative procedure ("cascade algorithm"), which enables us to compute the scaling function  $\phi(t)$  for any member dbn of the family of Daubechies-wavelets, provided the h-coefficients are given [8]. The corresponding wavelet results from 3.29.

We shall give a detailed description of the procedure for the Haar-system, i.e., for db1. So the  $H^*$ -filter mentioned below belongs to the coefficients 3.19. The generalization to arbitrary dbn then should be obvious.

1. Define the sequence  $e = \{e_i\}$  with

$$e_i = \begin{cases} 1 \ i = 0\\ 0 \ \text{else} \end{cases}$$

Represent this sequence by a piecewise constant continuous-time function as follows:

- a) Subdivide the *t*-axis in intervals of length 1 such that t = 0 is the center of one of these intervals. Denote this interval with  $I_0$  and the neighboring intervals with  $I_i$  (i = 1, 2, 3, ...) and  $I_i$  (i = -1, -2, -3, ...), respectively.
- b) Represent the sequence  $e = \{e_i\}$  by a continuous-time function e(t), where

$$e(t) = \begin{cases} 1 \ t \in I_0 \\ 0 \ \text{else} \end{cases}$$

The top row of Fig. 3.6 shows the sequence  $e = \{e_i\}$  and the related function e(t).

2. Now compute the sequence  $e^1 = \sqrt{2}H^*e$ , i.e., apply a dual filter similarly as in Fig. 3.5, where the coefficient values are replaced by 1 instead of  $\frac{1}{2}$ and in the bottom row we insert the sequence values  $e_k$ . The resulting sequence is represented by the left graph in the second row of Fig. 3.6. In particular, we have  $e_0^1 = e_1^1 = 1$ .

Similarly to above, a continuous-time function  $e^1(t)$  is associated to this sequence with the following differences: The length of the intervals used for subdividing the *t*-axis is now  $\frac{1}{2}$  instead of 1. Moreover

$$e^{1}(t) = \begin{cases} e_{0}^{1} = 1 \ t \in I_{0} \\ e_{1}^{1} = 1 \ t \in I_{1} \\ 0 \quad \text{else} \end{cases}$$

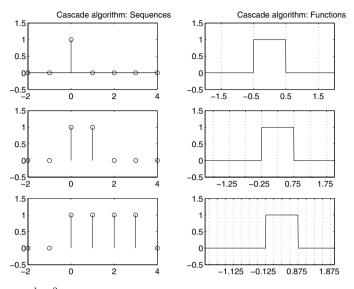
This function is represented by the right graph in the second row of Fig. 3.6.

3. Continue the steps described above, i.e., compute  $e^2 = \sqrt{2}H^*e^1$ ,  $e^3 = \sqrt{2}H^*e^2$ , .... Associate the corresponding continuous-time functions  $e^2(t)$ ,  $e^3(t)$ , ... where each function is piecewise constant on intervals, whose length is divided by two when proceeding from  $e^i(t)$  to  $e^{i+1}(t)$ . Sequence  $e^2 = \{e_k^2\}$  and the corresponding continuous-time function  $e^2(t)$ 

are shown in the last row of Fig. 3.6

Figure 3.6 shows how the "initial estimates"  $e(t), e^1(t), e^2(t), \ldots$  converge to the correct scaling function  $\phi_H(t)$  defined in eq. 3.4.

Likewise, for general n, the procedure described above leads to an approximative scheme for computing the dbn-scaling function. The quality of the approximation is controlled by the number of iterations. As mentioned above, the corresponding wavelet may be obtained from eq. 3.29.



**Fig. 3.6.**  $e, e^1, e^2$  represented as sequences (left) and functions (right). The grid lines in the function plots indicate the respective subdivision of the *t*-axis.

For the db2-case this is illustrated in Fig. 3.7. The approximation  $e^1(t)$  corresponding to the first iteration is displayed in the upper left diagram. The reader may verify that the values of the nonzero step heights are given by the corresponding coefficients from Table 3.1, multiplied with  $\sqrt{2}$ . The lower left diagram shows the wavelet approximation derived from  $e^1(t)$ ; the approximations to scaling function and wavelet after 7 iterations are displayed in the upper right and lower right diagrams.

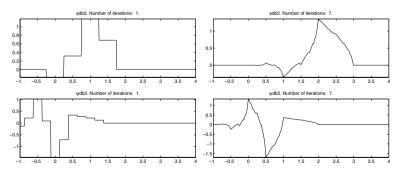


Fig. 3.7. Iterative construction of db2-scaling function and wavelet. One iteration (left), 7 iterations (right).

We note an "improvement in smoothness": The db1-wavelets and scaling functions are discontinuous, whereas in the db2-case both functions are continuous. Still these functions look rather rugged and in fact higher order dbn-wavelets and scaling functions get smoother. Mathematically this statement can be made precise using the notion of "regularity" [24], it is illustrated in Fig. 3.8 for db5.

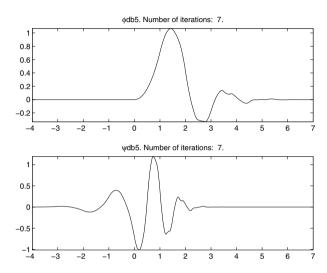


Fig. 3.8. db5: scaling function and wavelet after 7 iterations.

So, in fact, for increasing n we not only obtain wavelets with an increasing number of vanishing moments but also wavelets with increasing regularity. Both properties are desirable for signal analysis purposes<sup>2</sup>.

### 3.3.2 Transfer Properties

For any dbn-filter the *H*-symbol actually denotes the application of a digital filter  $\mathcal{H}$  with the coefficients  $\{h_k\}$ , followed by a subsampling procedure (cf. sect. 5.3, Fig. 5.5). For the *G*-symbol and the corresponding coefficients  $\{g_k\}$  analogous conventions hold.

From a digital signal processing viewpoint it is quite instructive to study the frequency response of  $\mathcal{H}$  and  $\mathcal{G}$ , respectively. In this section we shall consider only the amplitude response, the phase response will be investigated later. Frequency response, amplitude response and phase response are defined in eqs. 5.18 and 5.19, respectively. For a short summary of these topics the reader is referred to sect. 5.3.

 $<sup>^2</sup>$  We remind the reader to the comparison of db2 with db4 shown in Figs. 2.5 and 2.6, respectively.

#### 60 3 The Discrete Wavelet Transform

For db1 to db4 the amplitude response of the coresponding  $\mathcal{H}$  and  $\mathcal{G}$ filters are shown in Fig. 3.9. The graphs are arranged in clockwise order with db1 displayed in the upper left diagram. In each diagram both the amplitude responses of  $\mathcal{H}$  (solid) and  $\mathcal{G}$  (broken) are plotted.

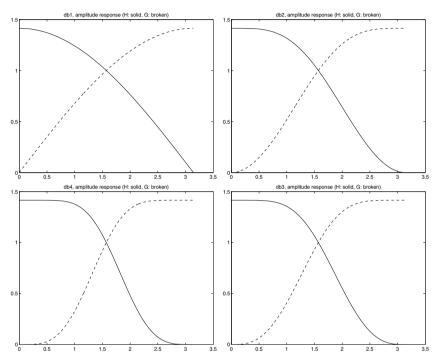


Fig. 3.9. Amplitude response of db1-db4 filters (clockwise from above left).

The plots indicate that the respective  $\mathcal{H}$ -filters exhibit a low-pass characteristic, whereas the  $\mathcal{G}$ -filters are high-pass filters. Moreover, obviously with increasing *n* the respective filter performance improves: The cutoff frequencies are more sharply localized and transmission as well as suppression of the respective frequency ranges works better and better.

This improvement in filter behavior corresponds to the discussion in the preceding section, where we demonstrated, how wavelets and scaling functions belonging to the filters "improve" when n is increased.

## 3.4 Multiscale Analysis

Let us assume that for a certain n the corresponding dbn-filter coefficients are given. So we have a pair of filters (H, G) whose coefficients satisfy eqs. 3.23 –

3.25. For a discretely sampled signal f a one-step-version of a DWT can be carried out as depicted in eq. 3.22.

In this section we shall show how this procedure may be iterated, leading to a *multiscale analysis* of f. It will turn out that the algorithm in a quite straightforward manner may be generalized to two-dimensional signals. So we will extend the DWT-techniques to image processing.

#### 3.4.1 One-Dimensional Signals

Let us first consider the decomposition component of eq. 3.22. Starting from f, the signals  $f^1$  and  $d^1$  are computed by applying the filter pair (H, G) to f. We may repeat this procedure beginning with  $f^1$  and obtaining  $f^2$  and  $d^2$ . If we proceed in this way, i.e., if we apply (H, G) successively to  $f^2, f^3, \ldots$ , after J steps the signal is decomposed as indicated by the left diagram of eq. 3.30.

Usually  $f^J$  is called "approximation signal", the remaining signals  $d^j$   $(j = 1, \ldots, J)$  "detail signals". Remember that  $f^J$  results from a repeated application of a low-pass filter, whereas for the detail signals the final computation step always is high-pass filtering. Below we shall discuss these signals in more detail and show some examples.

Note that in each filtering step the signal length is divided by 2, which may be represented as in Fig. 3.10.

From this diagram it is obvious that there exists an upper limit for the number of decomposition steps J: If the original signal is of length  $2^L$ , the maximum value of J will be L.

Since each decomposition step may be inverted as indicated in the right diagram of eq. 3.22, a *J*-step DWT-decomposition of f is inverted by the algorithm shown in the right diagram of eq. 3.30. Again, the  $\oplus$ -symbol represents addition of the respective input sequences.

Equation 3.26, establishing the link from sequences to continuous-time functions may be generalized to a J-step DWT as follows:

Defining the continuous-time signal  $f(t) = \sum_k f_k \phi(t-k)$ , the correspond-

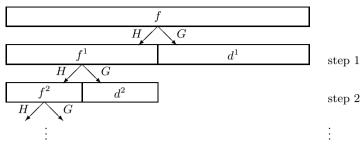


Fig. 3.10. J-step DWT-decomposition of a signal f

ing CWT-values for  $a = 2^j$  and  $t = 2^j k$  (with integer k) can be computed according to

$$L_{\psi}f(2^{j},2^{j}k) = \sqrt{\frac{1}{c_{\psi}}}d_{k}^{j} \ (j=1,\ldots,J).$$
(3.31)

Again,  $(\phi, \psi)$  denote scaling function and wavelet, associated to filters (H, G) as discussed above.

Similarly to Fig. 3.3 the above subset of the *t*-*a*-plane is visualized in Fig. 3.11 for J = 3. It is clear how to extend this subgrid for general J. This plot explains why the *J*-step-DWT decomposition also is called "multiscale analysis".

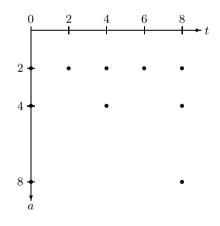


Fig. 3.11. 3-step-version of the DWT: Subset of the *t-a*-plane.

Note that from Fig. 3.10 it follows that a J-step DWT leaves the total number of data unchanged: Applying a J-step-DWT to the N sequence elements of

the original signal  $f = \{f_k\}_{k=0}^{N-1}$ , we obtain the sequences  $f^J, d^J, d^{J-1}, \ldots, d^1$ , where the total number of data resulting from adding the lengths of these sequences, again<sup>3</sup> will be N. Moreover, the decomposition part of eq. 3.30 provides a  $fast^4$  algorithm for computing wavelet transform values via the sequences  $d^j$ . Thus, when comparing with the situation visualized in Fig. 3.1, we conclude that the *J*-step-DWT leads to a simple and quick algorithm for computing wavelet transforms without redundancy on a subgrid of the *t*-*a*-plane, which takes into account *varying* scale factors in contrast to the discretized CWT.

Up to now (cf. eq. 3.2) we assumed  $T_S = 1$ . As announced on page 45 the extension to an arbitrary sampling distance  $T_S$  is very simple: Looking at Fig. 3.11 replace on the *t*-axis the labels  $0, 2, 4, 6, \ldots$  by  $0, 2T_S, 4T_S, 6T_S, \ldots$  and correspondingly on the *a*-axis the labels  $0, 2, 4, 8, \ldots$  by  $0, 2T_S, 4T_S, 8T_S, \ldots$ . So, e.g.,

$$L_{\psi}f(4T_S, 4T_S) = \sqrt{\frac{1}{c_{\psi}}}d_1^2.$$

## Examples

In Fig. 3.12 the result of a 2-step DWT with db1-filters is displayed. From above, the signal f, the detail signals  $d^1$  and  $d^2$  and the approximation signal  $f^2$  are shown.

Figure 3.13 shows the results of the same transform, but performed with db2-filters instead of db1.

The results are in accordance with the properties of dbn-filters collected in sections 3.3 and 3.3.1. Since the db2-wavelet is smoother and has more vanishing moments than the db1-wavelet, one expects the db2-detail signals to be better concentrated around significant signal changes than the db1-detailsignals. Since f has a regular appearance, i.e., significant signal changes are lacking, this simply means that detail signal values should be rather small. Obviously this is true for both filter types and indeed the db2-detail signals are closer to zero than the corresponding db1-detail signals.

The concentration property discussed above has another useful consequence. Remember that the signal can be reconstructed from approximation and detail signals by the reconstruction part of eq. 3.30. Therefore - due to the concentration property - the *J*-step-DWT is a method to concentrate the full information stored in the signal in few significant coefficients. In Figs. 3.12 and 3.13 these coefficients basically are the sequence elements of  $f^2$  and the concentration mechanism works more efficiently for db2 than for db1. This

 $<sup>^3</sup>$  To be precise, for arbitrary N the total number of transformed data can be slightly larger than N. Refer to sect. 3.4.3 for a short discussion.

<sup>&</sup>lt;sup>4</sup> In fact, since the algorithm reduces to discrete filtering, the number of operations needed for computing the  $d^{j}$ -sequences increases *linearly* with N.

#### 64 3 The Discrete Wavelet Transform

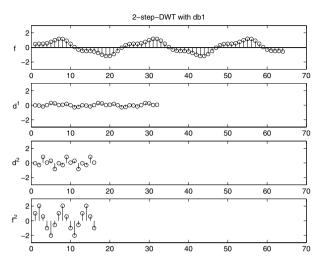


Fig. 3.12. 2-step DWT with db1.

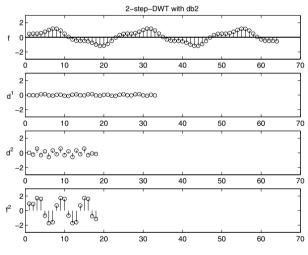


Fig. 3.13. 2-step DWT with db2.

concentration feature is called "energy compaction" [38] and a basic prerequisite for data compression. We will return to it in chap. 4.

In Fig. 3.14 it is illustrated with histograms. The top figure shows the histogram of f computed as follows: Take the smallest and the largest sequence element and subdivide the interval between these values in 20 equally spaced bins. Now for each bin count the number of sequence elements whose values are contained in it. For the transformed signal we proceeded as follows: Merge sequences  $f^2$ ,  $d^2$  and  $d^1$  into one sequence, then continue as described for f. In this way we obtained the db1-histogram (bottom left) and the db2-histogram (bottom right). First we note that both histograms are more sharply peaked (with the largest peak around zero) than the histogram of the original signal. This is in accordance with our discussion from above concerning the concentration features of the *J*-step-DWT in general. Moreover, the db2-histogram is more sharply peaked than the db1-histogram, which also matches our expectation.

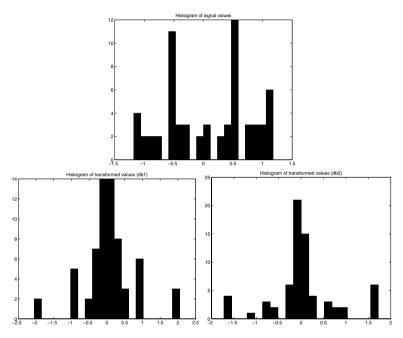


Fig. 3.14. Above: Histogram of the original signal. Below left: Histogram of the transformed signal (db1); below right: Histogram of the transformed signal (db2).

## 3.4.2 Two-Dimensional Signals (Images)

Until here, the signals considered were sequences, usually recorded as a time series. Digital images, in contrast, are represented as matrices

$$f = \{f_{mn}\} \ (m = 0, \dots, M - 1; \ n = 0, \dots, N - 1).$$
(3.32)

Similarly to the 1-d-case, f denotes the matrix,  $f_{mn}$  the element in row m and column n. M and N denote the total number of rows and columns, respectively. Since f represents a picture,  $f_{mn}$  will be called the "pixel" in row m and column n. The image is generated by representing each pixel by a gray value such that dark corresponds to small and bright to large pixel

values<sup>5</sup>. An example is shown in the upper part of Fig. 3.15, here M = 240 and N = 320.

The filters discussed so far transformed sequences into sequences. They can be extended to images by applying them to rows, or columns, respectively. The kind of application is indicated by a corresponding index. So, e.g., the filter  $H_r$  is defined as follows: Starting with the original digital image f a new image  $H_r f$  is generated by filtering every row of f separately with H. From the defining equation 5.20 it follows that

$$(H_r f)_{mn} = \sum_k h_{k-2n} f_{mk}.$$
 (3.33)

Note that, since the length of a sequence is divided by 2 when applying H, correspondingly the length of the rows, i.e., the number of columns, will be divided by two when applying  $H_r$ . As an example, the lower part of Fig. 3.15 shows  $H_r f$ , the original f is displayed in the upper part. Here we used the db2-H-filter with coefficients given in Table 3.1. Completely analogously, the symbol  $H_c$  is used if H is applied to columns. Correspondingly, in this case the number of rows will be divided by 2.

It should be clear now, how to apply the remaining filters, i.e., the *G*-filter and the dual filters  $H^*$  and  $G^*$ , to rows or columns, respectively, and how to denote this.

Given a digital image f, a one-step-DWT is defined by applying all four possible combinations resulting from filtering along rows and columns with Hand G, respectively:

$$f \mapsto \{H_r H_c f, H_r G_c f, G_r H_c f, G_r G_c f\}$$

$$(3.34)$$

Note that, for example,  $H_rH_cf$  will have a total number of pixels, which is divided by 4, when compared with the total number of pixels of f. The reason is that the number of columns and the number of rows is divided by 2. The same reasoning applies to the remaining images  $H_rG_cf$ ,  $G_rH_cf$  and  $G_rG_cf$ . Therefore, if we add the number of pixels of all generated images, we see that - as in the one-dimensional case - the total number of data remains unchanged, when a one-step-DWT is applied<sup>6</sup>.

The images  $H_rG_cf, G_rH_cf$  and  $G_rG_cf$ , are called "horizontal", "vertical" and "diagonal" detail signals, respectively. To motivate this terminology we consider  $H_rG_cf$ , the argument for the other images is completely analogous. Applying  $G_c$  to f, i.e., applying G to the columns of f, means that we compute the wavelet transform of each *column*. As we have seen, we compute a signal,

<sup>&</sup>lt;sup>5</sup> We shall restrict ourselves to this kind of images, also called "gray-level images". The techniques developed in this section easily can be adapted to other colormaps and to colored images. For a survey on image representations refer to, e.g., [14].

<sup>&</sup>lt;sup>6</sup> Analogously to the remarks made in the footnote on page 63, the total number of transformed data can be slightly increased. Refer to sect. 3.4.3 for a short discussion.



# H<sub>r</sub>-filtering with db2

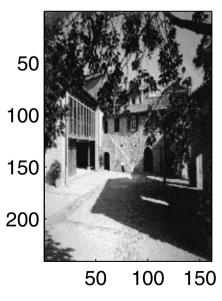


Fig. 3.15. Above: Original image. Below: Result of applying H (here: db2) to each row.

which basically "reacts" on signal changes. In other words: We look along the columns for signal changes. On the other hand, horizontal edges will give rise to signal changes when scanning columns, therefore the image  $H_rG_cf$  will react on horizontal edges. Moreover, since (cf. eq. 3.26) the wavelet transform is computed for scale  $2^1$ , the horizontal detail signal  $H_rG_cf$  also is denoted with  $d^{1h}$ .

Completely analogously, vertical and diagonal detail images react on correspondingly directed edges and therefore  $G_r H_c f$  is denoted with  $d^{1v}$ ,  $G_r G_c f$ with  $d^{1d}$ . We finally remark that - analogously to the 1-d-case treated above the low-pass filtered image  $H_r H_c f$  is called "approximation signal", denoted with  $f^1$ .

Usually the four subimages  $\{f^1, d^{1h}, d^{1v}, d^{1d}\}$  generated by a one-step-DWT are arranged like

$$\left(\frac{f^1}{d^{1v}}\frac{d^{1h}}{d^{1d}}\right) \tag{3.35}$$

Figure 3.16 shows a one-step-DWT carried out with Haar-filters. The resulting subimages are arranged as in eq. 3.35.

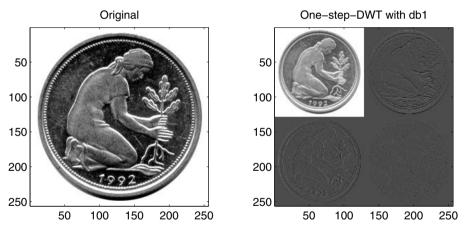


Fig. 3.16. Left: Original image. Right: Result of the one-step-DWT (here: db1).

In particular for horizontal edges, the above-mentioned directional sensitivity of the detail signal may be verified.

Again, analogously to the procedure described in sect. 3.4.1, a "multiscale analysis" of a digital image f can be performed by applying a one-step-DWT 3.34 successively to  $f^1, f^2 = H_r H_c f^1, f^3 = H_r H_c f^2, \ldots$  Thus the equivalent of eq. 3.30 reads

f

$$\begin{array}{c} \begin{array}{c} & & & & & & & & & \\ & & & & & & & \\ f & \frac{H_r G_c}{H_r G_c} d^{1h} & & & & & \\ & & & & & \\ & & & & & \\ & & & & \\ & & & & \\ H_r H_c \\ \\ & & & & \\ & & & \\ f^1 & \frac{H_r G_c}{G_r G_c} d^{2h} & & & \\ & & & & \\ f^1 & \frac{H_r G_c}{G_r G_c} d^{2h} & & & \\ & & & & \\ & & & \\ f^1 & \frac{H_r G_c}{G_r G_c} d^{2h} & & \\ & & & \\ & & & \\ & & & \\ & & & \\ H_r H_c \\ \\ \\ & & & \\ & & \\ & & \\ f^{J-1} & \frac{H_r G_c}{H_c} d^{Jv} & \\ \\ & & & \\ f^{J-1} & \frac{H_r G_c}{G_r G_c} d^{Jh} & \\ & & \\ & & \\ H_r H_c \\ \\ & & \\ & & \\ f^{J} & & \\ \end{array} \end{array}$$

$$\begin{array}{c} & & & & \\ & \\ & & \\ & & \\ & & \\ & & \\ & & \\ & & \\ & & \\ & & \\ &$$

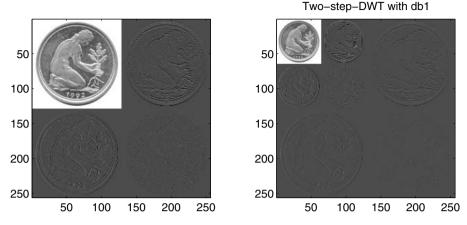
Again, the left diagram represents decomposition, the right diagram reconstruction. The symbol  $\oplus$  now denotes addition of matrices rather than sequences. Generalizing the representation from eq. 3.35, the result of a twostep-DWT, for example, can be represented as

$$\begin{pmatrix} \frac{f^2}{d^{2v}} \frac{d^{2h}}{d^{2d}} \\ \frac{d^{1v}}{d^{1v}} \frac{d^{1h}}{d^{1d}} \end{pmatrix}$$
(3.37)

So, if we continue to decompose the original of Fig. 3.16, we obtain a decomposition as shown in the right part of Fig. 3.17.

# 3.4.3 Implementations with the MATLAB Wavelet Toolbox

In this section we give a short description of functions contained in the MATLAB Wavelet Toolbox, which implement the *J*-step-DWT for 1-d and 2-d-signals, respectively. Related functions which make use of the toolbox-procedures have been tested with MATLAB 6.5, release 13 and may be down-



**Fig. 3.17.** Left: One-Step-DWT (cf. Fig. 3.16). Right: Two-Step-DWT, obtained by decomposing the upper left subimage of the one-step-DWT.

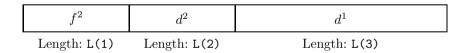
loaded from the URL given in sect. 1.3. Again, as mentioned earlier, a basic understanding of MATLAB-syntax is required.

## 1-d-functions

The decomposition part of the one-dimensional J-step-DWT (cf. eq. 3.30) is implemented in the toolbox-function wavedec. A prototype function call reads

[f\_trans,L] = wavedec(f,J,'dbn');

The original signal is stored in the vector **f**, the result of the transform in the vector **f\_trans**. The latter is organized as follows :



Here for illustration purposes J = 2 was chosen, refer also to Fig. 3.10. As indicated, the lengths of the individual subsequences are stored in the components of "book-keeping-vector" L. Thus, e.g., in MATLAB-notation  $d^2$  may be extracted from f\_trans as follows:

d\_2=f\_trans((L(1)+1):(L(1)+L(2)));

Note that, if N denotes the length of the original signal, then in our example

$$L(1)\approx \frac{\texttt{N}}{4},\; L(2)\approx \frac{\texttt{N}}{4},\; L(3)\approx \frac{\texttt{N}}{2}.$$

The reason for the approximate equality is that N not necessarily is a power of two; moreover, for dbn-filters with n > 1 the treatment of signal boundaries requires the subsequences to be slightly longer than the fractions computed above. As an illustration we include the following MATLAB-dialogue, demonstrating a 2-step-DWT with a signal of length 1367:

We see that f\_trans in general will be slightly longer than f and this "lengthening effect" increases with filter length.

So the book-keeping-vector is essential for the correct handling of the individual subsequences merged in the vector  $f\_trans$ . It is clear, therefore, that it is an important input into the toolbox function waverec implementing the reconstruction part of the one-dimensional *J*-step-DWT (cf. eq. 3.30). A prototype call reads

```
f = waverec(f_trans,L,'dbn');
```

Note that the decomposition depth J needs not to be specified, since it follows immediately from the length of the book-keeping-vector L, generated by the preceding decomposition.

# 2-d-functions

The decomposition part of the two-dimensional J-step-DWT (cf. eq. 3.36) is implemented in the toolbox-function wavedec2. A prototype function call reads

[f\_trans,L] = wavedec2(f,J,'dbn');

Here the original image is stored in the matrix **f** with M rows and N columns. The result of the transform is stored in the *vector* **f\_trans**, which is organized as follows :

$f^1$	$d^{1h}$	$d^{1v}$	$d^{1d}$
-------	----------	----------	----------

72 3 The Discrete Wavelet Transform

Here for illustration purposes J = 1 was chosen, refer also to eq. 3.35. Subsequence  $f^1$ , for example, contains all pixels belonging to the approximation image, but arranged as a *vector* obtained by scanning the approximation image row by row. Similarly the remaining subsequences are constructed.

Instead of a book-keeping-vector in te 2-d-case a "book-keeping-matrix" L is generated, which stores the matrix dimensions of the approximation signal and the detail signals as follows: L has 2 columns, the first row provides the number of rows and of columns, respectively, for  $f^1$ . The second row provides the dimensions of  $d^{1h}$ ,  $d^{1v}$  and  $d^{1d}$  (which are all equal). Similarly for J > 1 subsequent rows of L are generated.

The MATLAB-dialogue shown below illustrates the use of this function for an image with M = N = 16:

```
>> f=rand(16,16);
>> [f_trans,L]=wavedec2(f,2,'db1');
>> L(1:3,:)
ans =
     4
            4
     4
            4
     8
            8
>> length(f_trans)
ans =
   256
>> [f_trans,L]=wavedec2(f,2,'db2');
>> L(1:3,:)
ans =
     6
            6
     6
            6
     9
            9
>> length(f_trans)
ans =
   387
```

Note that – analogously to the 1-d-case – with increasing filter order the transformed signal will be larger than the original! Since the transformed signal is stored in a vector, where the boundaries of the individual subsequences may be determined from L, these subsequences can be extracted using L. To build the corresponding image, however, they must be rearranged to a matrix!

Let's continue the example from above with the task to extract the detail image  $d^{1d}$ :

As in the 1-d-case the signal can be reconstructed from  $f\_trans$ , provided the "building recipe" of  $f\_trans$  is known, which is stored in L. Thus the toolbox function waverec2 implementing the reconstruction part of the twodimensional *J*-step-DWT (cf. eq. 3.36) may be envoked as follows:

```
f = waverec2(f_trans,L,'dbn');
```

# dbn-coefficients

In sect. 3.3 we summarized the properties of dbn-filters without specifying how to compute the corresponding h-coefficients. Of course, there exists an algorithm for that purpose (cf., e.g., [20]).

In the MATLAB Wavelet Toolbox the function wfilters does the job. So, e.g.,

h=wfilters('dbn');

returns the *h*-coefficients belonging to dbn-filters in the vector  $\mathbf{h}$ . The maximum value of  $\mathbf{n}$  is 45.

# 3.4.4 Generalization: Biorthogonal Filters

As noted in the appendix, symmetric filters have a linear phase response, which reduces signal distortion. For more details the reader is referred to sect. 5.3. Phase response is defined in eqs. 5.18 - 5.19, examples for symmetric filters are shown on page 132.

From Table 3.1, we conclude that the *h*-coefficients belonging to db1 are symmetric, for n = 2 and n = 3, however, this is not true. In fact, all dbn-filters with n > 1 are not symmetric, therefore we cannot expect linear phase response for these filters. Figure 3.18 illustrates this fact by plotting phase responses for db1 and db2. Similarly to Fig. 3.9 the phase responses of  $\mathcal{H}$  and  $\mathcal{G}$  are plotted as solid and broken lines, respectively.

So the question arises, if filters maintaining the "good" properties of dbn with n > 1 and being symmetric can be constructed. This problem has been

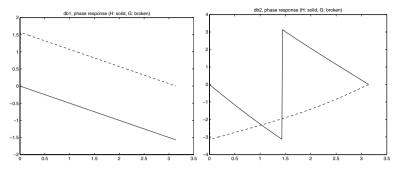


Fig. 3.18. Left: Phase response for db1. Right: Phase response for db2.

solved by A. Cohen [5]. For an easy-to-read summary see [24]. We shall briefly report the main results for the 1-*d*-case. The generalization to 2-*d*-signals then is obvious.

It turns out that eq. 3.27 representing the correlation between discrete filters and wavelets needs to be extended as follows:

 $(H, \tilde{H}, G, \tilde{G}) \leftrightarrow (\phi, \tilde{\phi}, \psi, \tilde{\psi}).$  (3.38)

The generalization of eq. 3.25 reads

$$\tilde{H}^*H + G^*\tilde{G} = Id. \tag{3.39}$$

In other words: We have a *pair* of wavelets and scaling functions, respectively, associated to a corresponding pair of high- and low-pass filters, such that a single decomposition step is performed by applying H and  $\tilde{G}$ . This step may be inverted by applying the dual filters  $\tilde{H}^*$  and  $G^*$ , respectively, and adding the results. So the *J*-step-transform 3.30 now may be represented as

The generalization of this equation to 2-d-signals is straightforward.

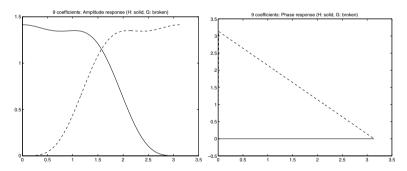
In image compression biorthogonal wavelet filters are of particular importance: The lossless mode of the JPEG 2000 still image compression standard [36] uses a biorthogonal filter pair  $H, \tilde{H}$  with 9 and 7 coefficients, respectively. The values are listed below:

$\{h_k\}$	$\{ ilde{h}_k\}$
$\begin{array}{l} h_{-4} = 0.026748757411\sqrt{2} \\ h_{-3} = -0.016864118443\sqrt{2} \\ h_{-2} = -0.078223266529\sqrt{2} \\ h_{-1} = 0.266864118443\sqrt{2} \\ h_{0} = 0.602949018236\sqrt{2} \\ h_{1} = 0.266864118443\sqrt{2} \\ h_{2} = -0.078223266529\sqrt{2} \\ h_{3} = -0.016864118443\sqrt{2} \\ h_{4} = 0.026748757411\sqrt{2} \end{array}$	$ \begin{split} \tilde{h}_{-3} &= -0.045635881557\sqrt{2} \\ \tilde{h}_{-2} &= -0.028771763114\sqrt{2} \\ \tilde{h}_{-1} &= 0.295635881557\sqrt{2} \\ \tilde{h}_{0} &= 0.557543526229\sqrt{2} \\ \tilde{h}_{1} &= 0.295635881557\sqrt{2} \\ \tilde{h}_{2} &= -0.028771763114\sqrt{2} \\ \tilde{h}_{3} &= -0.045635881557\sqrt{2} \end{split} $

Table 3.2. Filter coefficients of the biorthogonal 9-7-filter pair.

Note the symmetry of the filters! The corresponding G-filters are constructed similarly to eq. 3.28.

We end this section with a plot of amplitude and phase response of  $\mathcal{H}$  (solid) and  $\mathcal{G}$  (broken), respectively (both having 9 coefficients).



**Fig. 3.19.** Left: Amplitude response of  $\mathcal{H}$  (solid) and  $\mathcal{G}$  (broken); both filters have 9 coefficients. Right: Corresponding phase response.

# 3.5 A Unifying Viewpoint: Basis Systems

Earlier in this book (cf. sect. 1.2.6) we presented a qualitative comparision of the Fourier transform with the wavelet transform. In this section we shall consider the discrete versions, i.e., the DFT and the *J*-step-DWT, and discuss an aspect which both transforms have in common. It is shown that both transforms are expansions of the signal with respect to certain "basis systems" and these systems provide insight into the qualitative differences of both transforms.

Apart from sect. 3.6.1, the material covered in this section is not required for the rest of the book. So it may be skipped if the reader mainly is interested in a quick and application-oriented wavelet introduction.

## 3.5.1 One-Dimensional Signals

Let us start with a one-dimensional discretely sampled signal of length N:

$$f = \{f_k\}_{k=0}^{N-1}.$$
(3.41)

We introduce N sequences, called "natural basis system", as follows:

$$\begin{aligned}
\phi_{natural}^{0} &= \{1, 0, 0, \dots, 0\} \\
\phi_{natural}^{1} &= \{0, 1, 0, \dots, 0\} \\
\vdots &\vdots &\vdots \\
\phi_{natural}^{N-1} &= \{0, 0, 0, \dots, 1\}
\end{aligned}$$
(3.42)

Then f may be expanded as follows:

$$f = f_0 \phi_{natural}^0 + f_1 \phi_{natural}^1 + \dots$$
$$= \sum_{i=0}^{N-1} f_i \phi_{natural}^i$$
(3.43)

Note that in these formulae the summation of sequences is defined by elementwise addition! For N = 4 the natural basis system  $\phi_{natural}^0, \ldots, \phi_{natural}^3$  is plotted in the left column of Fig. 3.20.

Thus, when starting with the basis system 3.42, the signal values  $f_k$  may be viewed as expansion coefficients when expanding the signal with respect to this basis. It is a well known fact from linear algebra that basis systems are not unique: There exist other basis systems  $\phi^0, \ldots, \phi^{N-1}$  such that

$$f = \sum_{i=0}^{N-1} c_i \phi^i$$
 (3.44)

with proper expansion coefficients  $c_0, \ldots, c_{N-1}$ .

In fact, the IDFT 5.14 provides such a change of basis systems. Define the following sequences, called "Fourier basis":

$$\begin{split} \phi_{Fourier}^{0} &= \frac{1}{\sqrt{N}} \{1, 1, 1, \dots, 1\} \\ \phi_{Fourier}^{1} &= \frac{1}{\sqrt{N}} \{e^{\frac{j2\pi 1 \cdot 0}{N}}, e^{\frac{j2\pi 1 \cdot 1}{N}}, e^{\frac{j2\pi 1 \cdot 2}{N}}, \dots, e^{\frac{j2\pi 1 \cdot (N-1)}{N}} \} \\ \phi_{Fourier}^{2} &= \frac{1}{\sqrt{N}} \{e^{\frac{j2\pi 2 \cdot 0}{N}}, e^{\frac{j2\pi 2 \cdot 2}{N}}, e^{\frac{j2\pi 2 \cdot 2}{N}}, \dots, e^{\frac{j2\pi 2 \cdot (N-1)}{N}} \} \\ \vdots &\vdots \\ \phi_{Fourier}^{N-1} &= \frac{1}{\sqrt{N}} \{e^{\frac{j2\pi (N-1) \cdot 0}{N}}, e^{\frac{j2\pi (N-1) \cdot 1}{N}}, e^{\frac{j2\pi (N-1) \cdot 2}{N}}, \dots, e^{\frac{j2\pi (N-1) \cdot (N-1)}{N}} \} \end{split}$$

$$(3.45)$$

Then 5.14 may be rewritten as follows:

$$f = \frac{1}{\sqrt{N}} (F_0 \phi_{Fourier}^0 + F_1 \phi_{Fourier}^1 + \dots)$$
  
=  $\frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} F_i \phi_{Fourier}^i.$  (3.46)

In other words: When expanding f with respect to the Fourier basis system  $\phi_{Fourier}^0, \ldots, \phi_{Fourier}^{N-1}$ , the corresponding expansion coefficients are given by  $\frac{F_i}{\sqrt{N}}$ , where  $F_i$  denotes the *i*-th DFT-coefficient computed according to eq. 5.13.

For N = 4 the Fourier basis system is plotted in the middle column of Fig. 3.20. Note that the sequences belonging to the Fourier basis are complex! The plots show the real part of the respective basis sequences.

Now assume that a certain dbn-wavelet and a decomposition depth J has been chosen and perform a J-step-DWT corresponding to eq. 3.30. Assume that  $f^J, d^J, \ldots, d^1$  are arranged as indicated in Fig. 3.10 and merge these sequences into a new sequence called  $\tilde{f}$ . For J = 2 the procedure is indicated below:

$$\underbrace{(\begin{array}{c} f^2 \middle| d^2 \middle| d^1)}_{\tilde{f}} \tag{3.47}$$

Expanding  $\tilde{f}$  with respect to the corresponding natural basis system we may write

$$\tilde{f} = \sum_{i} \tilde{f}_{i} \tilde{\phi}^{i}_{natural}, \qquad (3.48)$$

where  $\tilde{f}_i$  denotes the *i*-the element of the sequence  $\tilde{f}$ . Now decompose each basis sequence  $\tilde{\phi}^i_{natural}$  into subsequences corresponding to the preceding *J*-step-DWT decomposition. Again, we indicate the procedure for J = 2:

$$\underbrace{\left(\begin{array}{c} \left(f_{\phi}^{2}\middle|d_{\phi}^{2}\middle|d_{\phi}^{1}\right)\\ \tilde{\phi}_{natural}^{i} \end{array}\right)}_{\tilde{\phi}_{natural}^{i}} \tag{3.49}$$

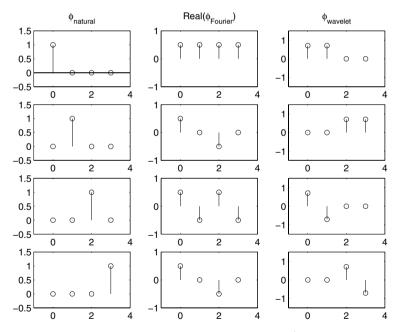
#### 78 3 The Discrete Wavelet Transform

Apply the reconstruction part of eq. 3.30 to  $f_{\phi}^{J}, d_{\phi}^{J}, \ldots, d_{\phi}^{1}$  and denote the reconstructed sequence with  $\phi_{wavelet}^{i}$ . Then from eq. 3.48 it follows that

$$f = \sum_{i} \tilde{f}_{i} \phi^{i}_{wavelet}.$$
(3.50)

So similarly to the DFT-case (eq. 3.46) a *J*-step-DWT may be viewed as a change of basis: Now the signal is expanded with respect to the wavelet basis instead of the Fourier basis and the expansion coefficients are given by the entries of the transformed sequences  $f^J, d^J, \ldots, d^1$ . This is in complete analogy to the DFT where the expansion coefficients were given by the DFTvalues. In contrast to the DFT-case, however, the wavelet basis is not unique: Obviously, the basis sequences  $\phi^i_{wavelet}$  ( $i = 0, 1, \ldots$ ) depend on both the chosen wavelet and the chosen decomposition depth J.

For N = 4 a wavelet basis system is plotted in the right column of Fig. 3.20. As mentioned above, this system depends on the chosen wavelet as well as on the chosen decomposition depth J. Here db1, i.e., the Haar-wavelet with the corresponding filters, and J = 1 were selected.



**Fig. 3.20.** Basis systems for N = 4. Left column:  $\phi_{natural}^k$ . Middle column:  $Re(\phi_{Fourier}^k)$ . Right column:  $\phi_{wavelet}^k$ . From above:  $k = 0, \ldots, 3$ .

From this figure, together with the above computations, we collect the following observations:

The signal  $f = \{f_k\}$  may be expanded with respect to 3 different basis systems:

The *natural* basis is "strictly local" in the sense that the corresponding basis sequences are nonzero at *exactly one* position, the expansion coefficients are given by the original signal values  $f_k$ .

The *Fourier* basis is "global" in the sense that the corresponding basis sequences may be nonzero at *all* positions, the expansion coefficients are proportional to the DFT-values  $F_k$ .

The wavelet basis is "local" in the sense that the corresponding basis sequences are concentrated around several positions, the expansion coefficients are given by the entries of the sequences  $f^J, d^J, \ldots, d^1$ . Thus, in a sense, the wavelet basis system interpolates between the strictly local natural basis and the global Fourier basis.

We will return to this qualitative difference in the behavior of the transform later (cf. sect. 3.6.1 and chap. 4).

### 3.5.2 Two-Dimensional Signals

What we have done for one-dimensional signals may be extended easily to the 2-d-case. We shall not bother the reader with detailed formulae but rather sketch the basic facts in eqs. 3.51 - 3.53 below.

As in sect. 3.4.2 the signal will be a matrix  $f = \{f_{mn}\}$  with *m* numbering rows and *n* numbering columns.

Expanding the signal with respect to the natural basis we obtain:

$$f = \sum_{mn} f_{mn} \phi_{natural}^{mn} \tag{3.51}$$

Here the matrices  $\phi_{natural}^{mn}$  are nonzero at *exactly one* pixel position. The expansion of f with respect to the Fourier basis reads:

$$f = \sum_{mn} A_{mn} \phi_{Fourier}^{mn} \tag{3.52}$$

Here the expansion coefficients  $A_{mn}$  are proportional to  $F_{mn}$ , i.e., the values of the two-dimensional DFT<sup>7</sup>. The matrices  $\phi_{Fourier}^{mn}$  may be nonzero at *all* pixel positions.

Finally, we may expand f with respect to a wavelet basis:

$$f = \sum_{mn} \tilde{f}_{mn} \phi_{wavelet}^{mn} \tag{3.53}$$

<sup>&</sup>lt;sup>7</sup> In sect. 5.2 the DFT was defined only for sequences. We remark that it may be extended to matrices by applying the DFT row- and columnwise. For details refer to, e.g. [33].

#### 80 3 The Discrete Wavelet Transform

Here the expansion coefficients  $\tilde{f}_{mn}$  result from merging the outcome of a two-dimensional *J*-step-DWT into one matrix analogously to eq. 3.37. The matrices  $\phi_{wavelet}^{mn}$  may be nonzero at *several* pixel position. Note that, as in the 1-d-case, the wavelet basis system will depend on the chosen dbn and decomposition step *J*.

We compare the qualitative differences of the basis systems in Fig. 3.21, where we used a gray value coding for the displayed matrices, with black representing the smallest value and white the largest. For the Fourier basis matrix the real part is displayed, for matrices belonging to natural and wavelet bases, respectively, absolute values are shown. Note that the respective features of the basis systems, recorded on page 79 for the one-dimensional case, obviously are valid also for 2-d-signals.

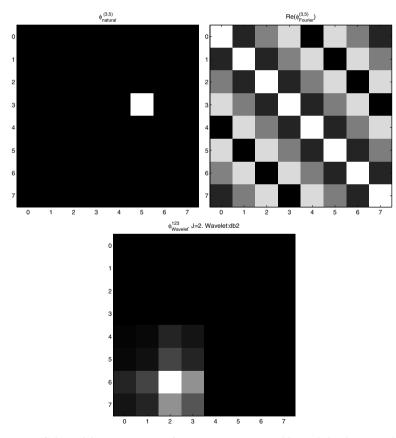


Fig. 3.21. Selected basis matrices for  $16 \times 16$ -images. Above left: A natural basis matrix, above right: Real part of a Fourier basis matrix, below: A wavelet basis matrix (db2, J = 2).

# 3.5.3 Computation and Visualization with MATLAB

Again, for this section basic familiarity with MATLAB is required. We shall discuss implementations generating the basis systems discussed above. The corresponding MATLAB-m-files have been tested with MATLAB 6.5, release 13 and may be downloaded from the URL given in sect. 1.3. Note that the m-files discussed in this section require the MATLAB Wavelet Toolbox!

As we have seen, for one-dimensional signals of length N each basis system is constructed by specifying N basis sequences  $\phi^0, \ldots, \phi^{N-1}$ . The function **bases1d** returns the three systems discussed in sect. 3.5.2 and visualizes the respective subsets  $\phi^0, \ldots, \phi^{k_{max}}$  by the sample call

# [phi\_natural,phi\_Fourier,phi\_wavelet]=bases1d(kmax,N);

Here  $0 \le k_{max} \le N - 1$  must be satisfied and phi\_natural, for example, is a  $N \times N$ -matrix with the sequences  $\phi_{natural}^0, \ldots, \phi_{natural}^{N-1}$  arranged rowwise. Likewise, the sequences constituting the Fourier and the wavelet basis system, respectively, are collected in the matrices phi\_Fourier and phi\_wavelet. Figure 3.20 was created with this function.

For quadratic images, i.e.,  $N \times N$ -images, representative basis matrices may be selected, plotted and returned with **bases2d**. A sample call reads

```
[phi_natural,phi_Fourier,phi_wavelet]=bases2d;
```

The selection of phi\_natural and phi\_Fourier, respectively, is done by specifying the index pair *mn* corresponding to formulae 3.51 and 3.52. The selection of phi\_wavelet is slightly more complicated since the toolbox functions wavedec2 and waverec2 are used, which reshape the outcome of a two-dimensional *J*-step-DWT as a vector (cf. sect. 3.4.3). Figure 3.21 was created with this function.

# 3.6 Case Studies

# 3.6.1 Energy Compaction and Compression

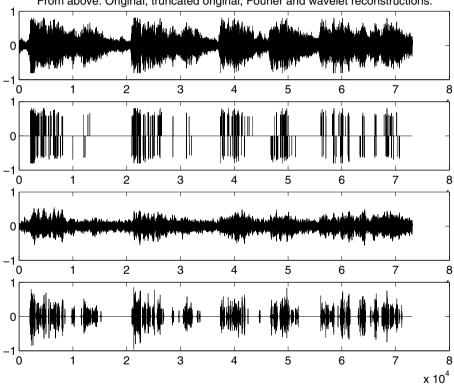
In this section more examples for the energy compaction property discussed above (cf. Figs. 3.12 - 3.14) are presented. Recall that "energy compaction" means that, after applying a *J*-step-DWT to the signal f, a large portion of the transformed signal values will be close to zero. In other words: The information stored in the signal will be concentrated in relatively few significant transform values. It is obvious that this property is essential for signal compression and we shall discuss a naïve compression model.

The DFT also shows energy compaction. Therefore, the sample signals we discuss will be transformed both with the DFT and the *J*-step-DWT and characteristic features of both transforms will be worked out.

#### **One-dimensional signals**

The top graph of Fig. 3.22 shows an acoustic signal. It was sampled with a sampling rate  $\nu_S = 8192 \,\text{Hz}$  (for the definition of the sampling rate refer to Table 5.2). The length of the signal f is N = 73113. For completeness we mention that f represents the first 9 seconds of G.F. Händel's famous "Hallelujah"-choir from the "Messiah"-oratory.

For the rest of this section we shall denote with F the sequence  $F = \{F_k\}$  of DFT-values (cf. eq. 5.13), the outcome  $f^J, d^J, d^{J-1}, \ldots, d^{\overline{1}}$  of a *J*-step-DWT will be merged into a sequence  $\tilde{f}$  as shown in eq. 3.47.



From above: Original, truncated original, Fourier and wavelet reconstructions.

**Fig. 3.22.** From above: Plot of f,  $f^p$ ,  $f^p_{Fourier}$  and  $f^p_{wavelet}$ . Parameters: p = 1%.  $J = 5, \, \mathrm{db20}.$ 

As before (cf. Fig. 3.14) we illustrate energy compaction with histograms. Similarly to the procedure described already, we compute the respective histograms of f, F and f as follows: Take the smallest and the largest absolute value of the corresponding sequence elements and subdivide the interval between these values in 100 equally spaced bins. Now for each bin count the number of sequence elements, whose absolute values are contained in it.

The result of this procedure is shown in Fig. 3.23.

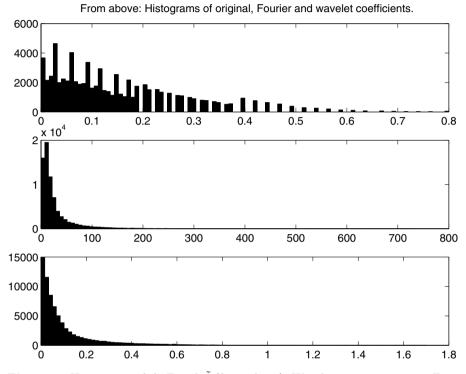


Fig. 3.23. Histograms of f, F and  $\tilde{f}$  (from above). Wavelet parameters as in Fig. 3.22.

We see that both transforms lead to histograms which are sharply peaked close to zero. Obviously, the "main information content" of the original signal is stored in relatively few significant values. For completeness we mention that the parameters belonging to the *J*-step-DWT were J = 5 (decomposition depth) and db20 (chosen wavelet). Observe that the histograms of both transforms are in sharp contrast to the histogram of the original, which is relatively uniform on its range of data!

We note already here (for more details refer to sect. 4.1.2) that for compression purposes the following properties are necessary/desirable: Transform the original signal such that

- 1. The transform is invertible (necessary condition) and
- 2. The histogram of the transform values is sharply peaked (desirable condition). In fact, the achievable compression rate of the procedure described in 4.1.2 will increase if the histogram is concentrated more and more on only few values.

Obviously the DFT and the J-step-DWT fulfill both requirements. In order to provide a feeling for the respective degree of energy compaction, when compared with the original, the following experiment was carried out (refer also to sect. 1.2.6):

- 1. Choose a percentage value  $p \ (0 .$
- 2. Starting from the histogram of the *J*-step-DWT  $\tilde{f}$  compute a truncated version  $\tilde{f}^p$  and a corresponding reconstruction  $f^p_{wavelet}$  as follows:
  - a) Sort the sequence elements  $\tilde{f}_k$  in descending order with respect to their absolute values. The resulting list we denote with  $l = \{l_k\}$ . Now choose a limit  $\tilde{f}^{limit}$  such that the absolute values of the first p% of *l*-entries are above or equal to  $\tilde{f}^{limit}$ , the rest below.
  - b) Then

$$\tilde{f}_k^p = \begin{cases} \tilde{f}_k \text{ if } |\tilde{f}_k| \ge \tilde{f}^{limit}\\ 0 \text{ else} \end{cases}$$

In other words: If the absolute value of a given transform coefficient belongs to the percentage p of transform coefficients having the *largest* absolute values, it is retained, otherwise it is "deleted", i.e., put equal to zero.

- c) Compute  $f_{wavelet}^p$  by applying the reconstruction part of eq. 3.30 to  $\tilde{f}^p = \{\tilde{f}_k^p\}.$
- 3. Proceed analogously with the DFT leading to the truncated transform sequence  $F^p = \{F_k^p\}$  and to the corresponding reconstruction  $f_{Fourier}^p$  by applying the IDFT (eq. 5.14) to  $F^p$ .
- 4. Proceed analogously with the original leading to the truncated version  $f^p$ .

The truncated signals  $f^p$ ,  $f^p_{Fourier}$  and  $f^p_{wavelet}$  will be such that their respective histograms are much more concentrated at zero than the histograms from Fig. 3.23: They are obtained from the latter by putting all coefficients with absolute values below the respective limits from above in the "zero-bin". Corresponding to the remarks made on page 84 therefore their compression performance will increase. We note already here that p provides a rough estimate for the achievable compression rate if  $f^p$ ,  $\tilde{f}^p$  or  $F^p$  are coded and stored as described in sect. 4.1.2 [1].

How are the signals  $f^p$ ,  $f^p_{Fourier}$  and  $f^p_{wavelet}$  distorted when compared with the original f? As we have seen in the section before, the signal may

be expanded with respect to the natural basis (eq. 3.43), the Fourier basis (eq. 3.46) or the wavelet basis (eq. 3.50) with expansion coefficients  $f_i$ ,  $\frac{F_i}{\sqrt{N}}$  and  $\tilde{f}_i$ , respectively. The truncation procedure from above means that in these expansions all contributions from basis sequences whose expansion coefficients are mapped to zero will vanish. Then by the *strict locality* of the natural basis system only p percent of  $f^p$  will be nonzero. On the other hand, this will not be true for the Fourier basis, due to its *global* nature, therefore  $f_{Fourier}^p$  might have nonzero entries everywhere. As stated before, the *locality*<sup>8</sup> of the wavelet basis system will lead to an "intermediate behavior": For small p-values one may expect that some parts of  $f_{wavelet}^p$  will be zero, but the "nonzero-portion" will be larger than for  $f^p$ .

These facts are illustrated in Fig. 3.22. Here from above f,  $f^p$ ,  $f^p_{Fourier}$  and  $f^p_{wavelet}$  are plotted with p-value and wavelet parameters indicated in the caption. It is clear that a sudden change from zero loudness to sound and vice versa will not be pleasing to the listener. Note that  $f^p$  will be zero over 99% of the playing time! Thus a "compression" resulting from the truncation procedure discussed above with p = 1% will lead to unacceptable results. Nevertheless, the basic mechanisms responsible for losses in signal quality when compressing it by concentrating histograms can be illustrated with these examples.

For compressing audio signals it turns out that the use of "Coiflets" (a variant of dbn-wavelets which ensures a close match of the approximation signal  $f^J$  with the original f) is promising [38]. Moreover, the desired concentration process of histograms should result from a "perceptually lossless quantization procedure" (see also sect. 4.3), which is more involved than the simple truncation procedure from above<sup>9</sup>.

# **Two-dimensional signals**

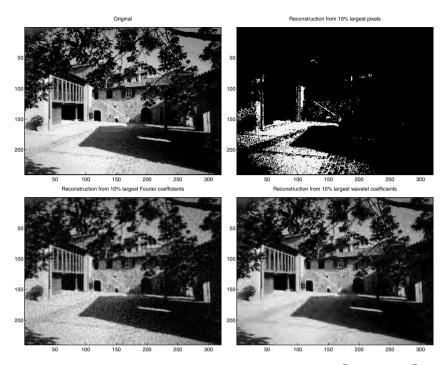
The experiment described above in complete analogy was carried out also for images. The terminology will be the same, i.e., the original image is denoted with f, F and  $\tilde{f}$  denote the corresponding two-dimensional DFT and J-step-DWT, respectively<sup>10</sup>. Again,  $f^p$ ,  $f^p_{Fourier}$  and  $f^p_{wavelet}$  denote the reconstructions resulting from transforms which were truncated in exactly the same way as described above. All images are subject to gray level coding, i.e.: zero is coded as black, the largest value as white.

The upper left subimage of Fig. 3.24 shows the original. The number of rows and columns of the corresponding pixel matrix were M = 240 and

<sup>&</sup>lt;sup>8</sup> For the notions of strictly local, global or local behavior, which are used here we refer the reader to page 79.

<sup>&</sup>lt;sup>9</sup> The successful MP3-compression scheme of audio signals combines the "discrete cosine transform", a variant of the DFT, with perceptually lossless quantization.

 $<sup>^{10}</sup>$  The matrices F and  $\tilde{f}$  were introduced in sect. 3.5.2, pages 79 ff.



**Fig. 3.24.** In clockwise order from above left: Plot of f,  $f^p$ ,  $f^p_{wavelet}$  and  $f^p_{Fourer}$ . Image dimensions: 240 × 320. Percentage: p = 10%. J = 4, db4.

N = 320, respectively. Figure 3.25 implies that the transforms lead to a concentration in the respective histograms, which is even more drastic than for 1-d-signals.

The remarks made before concerning locality of the respective basis systems apply also here. In particular, note that the chosen percentage of p = 10%implies that exactly 90% of  $f^p$  will be black, since the corresponding pixels are mapped to zero. The locality properties of the respective basis matrices (cf. Fig. 3.21) are reflected in the corresponding reconstructions: The global periodicity of Fourier-basis matrices leads to flickering artifacts in  $f_{Fourier}^p$ . The locality of wavelet-basis matrices leads to regions in  $f_{wavelet}^p$  which are of near constant intensity (they correspond to regions in the audio signal having constant loudness), whereas the zooming property of the wavelet transform (refer also to sect. 1.2.6) ensures that details like edges are relatively well reproduced. The visual reception here is different from listening: As explained before, changing from constant loudness to meaningful sound will be unacceptable to the listener, whereas changing from smoothened regions to detailed regions will not be so disturbing for the observer.

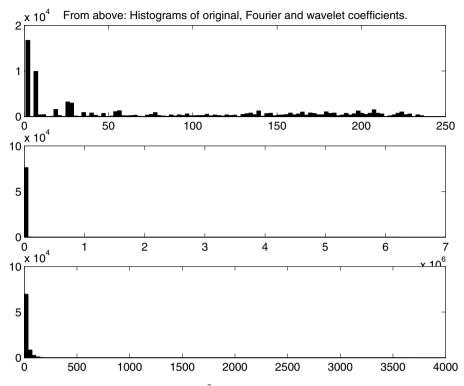


Fig. 3.25. Histograms of f, F and  $\tilde{f}$  (from above). Wavelet parameters as in Fig. 3.24.

# MATLAB Implementations

The truncation procedure described above is implemented in the functions CompAudioToolbox (one-dimensional signals) and CompImageToolbox (two-dimensional signals), respectively. These functions require the MATLAB Wavelet Toolbox and have been tested with MATLAB 6.5, release 13. They may be downloaded from the URL given in sect. 1.3.

Prototype function calls read

```
[ytrek,yfrek,ywtrek]=CompAudioToolbox(y);
```

and

```
[rek_pix,rek_ft,rek_wt]=CompImageToolbox(y);
```

respectively. The original signal is denoted with y, in the one-dimensional case the reconstructions  $f^p$ ,  $f^p_{Fourier}$  and  $f^p_{wavelet}$  are stored in ytrek, yfrek and ywtrek, analogously in the two-dimensional case these matrices are denoted with rek\_pix, rek\_ft and rek\_wt.

All required parameters (percentage p, requested dbn-wavelet, decomposition depth J) must be provided in a user dialogue.

# 3.6.2 Denoising a Sensor Signal / Real-Time Properties of the Algorithm

Removing noise is a standard topic in signal processing. Usually noise is attributed to high frequency components of the signal, which correspond to small detail sizes when performing a wavelet analysis (cf. eq. 1.4). Therefore the idea is to remove high frequency/small detail components such that underlying signal structures, which may be obscured by noise, are preserved and will emerge from the denoising process.

We have seen that a J-step-DWT 3.30 performs a multiscale analysis of a signal f in the sense that details up to  $\approx T_S 2^J$  are removed from  $f^J$ , whereas the detail signals  $d^J, d^{J-1}, \ldots, d^1$  store contributions from decreasing detail sizes in a way that makes it possible to recover f completely from  $f^J, d^J, d^{J-1}, \ldots, d^1$  by the reconstruction part from eq. 3.30. Therefore, a J-step-DWT seems to be a good candidate for denoising by the following three-step-procedure:

- 1. Perform a J-step-DWT.
- 2. Threshold detail signals corresponding to scales, which may be attributed to noise.
- 3. Reconstruct with manipulated detail signals.

Due to the pioneering work of D.L. Donoho and his collaborators (see, e.g. [9]) for certain noise models thresholding prescriptions for detail signal coefficients can be given, which remove the noise without disturbing important signal features. For a brief survey of these techniques refer to [1], the MATLAB Wavelet Toolbox contains related functions [24]. From the section before, where data compression was linked to thresholding, we see that denoising and compression are based on the same basic properties of the DWT, namely multiscale analysis and energy compaction.

In this section we shall illustrate denoising by a crude example. The purpose is to demonstrate the transfer properties of wavelet filters (see sect. 3.3.2) and to work out an aspect important for practical applications.

Let us again consider the sensor signal f from Fig. 2.7. Remember that this signal was recorded during an oscillatory process whose frequency changes from  $\nu \approx 3$  Hz to  $\nu \approx 6$  Hz at  $t \approx 3$  sec. Figure 2.7 shows that this transition may be recognized from the STFT, but f is obviously heavily corrupted with noise, which is reflected by nonzero high-frequency contributions.

In sect. 3.3.2 we saw that each *H*-filtering step in a *J*-step-DWT 3.30 corresponds to *low-pass filtering*. Since low-pass filtering removes high frequency components (small details) of f, one expects that  $f^J$ , which results from applying J low-pass filtering steps, will be a smoothened version of

f and the degree of smoothing will increase with J. Note, however, that the length of  $f^J$  will be equal to the length of f, divided by  $2^J$ , due to subsampling in every decomposition step! Therefore, a smoothened version  $f^{J,smooth}$  of f is obtained by applying the reconstruction part of eq. 3.30 to  $f^J, d^{J,smooth}, d^{J-1,smooth}, \ldots, d^{1,smooth}$ , where the entries of all  $d^{j,smooth}$ sequences are zero. Referring to the three-step-procedure from above, we apply a "brute-force-thresholding" by mapping all detail coefficients to zero, regardless of the respective level j, i.e., the respective scale sizes.

For J = 2, for example, the denoised signal  $f^{2,smooth}$  is computed as follows:

$$f \longrightarrow H \longrightarrow H^* \longrightarrow H^* \longrightarrow f^{2,smooth}.$$
 (3.54)

In Fig. 3.26 we show the resulting  $f^{2,smooth}$  together with its STFT (cf. sect. 2.1.3). Here db1-filters were chosen, i.e., the Haar-wavelet was used. Figure 3.27 shows the same results for db20-filters instead of db1-filters.

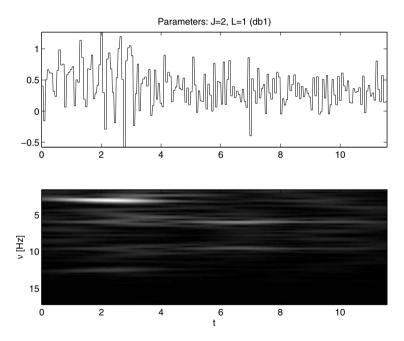
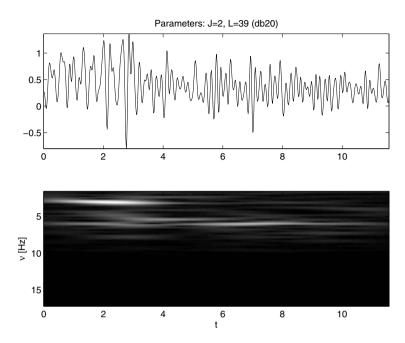


Fig. 3.26. Denoising the sensor signal from Fig. 2.7 with J=2 and L=1 (db1).

Comparing both figures with the original and its STFT (Fig. 2.7) we see that both procedures lead to very satisfying results: The frequency transition is now clearly visible in the STFT-plot and may even be localized by visual inspection of the denoised signals. Moreover, as is to be expected, the suppression of high frequencies works better with increasing order of dbn (see Fig. 3 The Discrete Wavelet Transform



**Fig. 3.27.** Denoising the sensor signal from Fig. 2.7 with J=2 and L=39 (db20).

3.9). Also the smoothness of the denoised signals improves with increasing order of dbn in perfect accordance with the corresponding wavelet properties (cf. sect. 3.3.1).

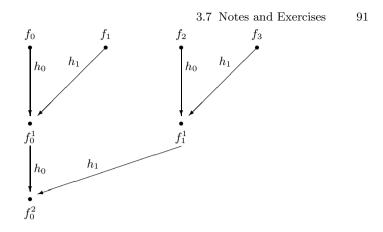
This simple example shows the potential of wavelet methods for denoising signals. Denoising a sensor signal can be important for obtaining reliable information from the sensor. Here, the signal stems from a light arc sensor of a welding robot and is needed for obtaining the actual distance of the welding tool from the welding seam. This distance information is of fundamental importance for controlling the position of the welding tool. In these kinds of applications a desirable feature of denoising is to compute the denoised signal "online": As soon as the k-th signal value  $f_k$  is recorded, the corresponding value  $f_k^{J,smooth}$  of the denoised signal should be available.

An online feature as defined above cannot be realized with wavelet filters. As an example we choose db1 and J = 2. The filter coefficients read  $\{h_0, h_1\}$ and we illustrate the two-fold H-filtering from eq. 3.54 by extending Fig. 5.6 to two decomposition steps (Fig. 3.28).

The diagram is plotted for  $T_S = 1$ . For a general sampling time  $T_S$  we will have a minimum delay time of  $3T_S$  in order to compute the  $f_k^2$ -values<sup>11</sup>. The reason for this delay is that the coefficient  $h_1$  of the db1-filter is "non-causal":

90

<sup>11</sup> Observe that this delay time will be a minimum value since addition and multiplication also consume processor resources.



**Fig. 3.28.** Two-fold *H*-application for a filter with coefficients  $\{h_0, h_1\}$ .

In order to compute , e.g.,  $f_0^1$ , the "future signal value"  $f_1$  is needed and this delay propagates from decomposition step to decomposition step.

In fact, it is not hard to obtain a general delay formula. With L denoting the number of non-causal filter coefficients (i.e., the number of  $h_k$ -coefficients with k > 0) the minimum delay  $t_{delay}$  reads

$$t_{delay} = T_S L(2^J - 1). \tag{3.55}$$

Indeed, for db1 with L = 1 and J = 2 we obtain the delay time read off from Fig. 3.28.

Note that there is a tradeoff between "denoising quality" and delay time! A dbn-*H*-filter has L = 2n - 1 non-causal coefficients. We saw in Figs. 3.26 and 3.27, respectively, that the features of the denoised signal improve with increasing *n*, but this improvement corresponds to a deterioration of onlinecapability. The corresponding *L*-vaues are given in the figure captions.

# 3.7 Notes and Exercises

#### Notes

There are numerous wavelet topics not treated at all in this book. We expliclitly mention wavelet packets and multiwavelets and encourage the reader to consult the references. The application-oriented reader might prefer [38], [24] or [1]. Readers interested in research are referred to [7], [20], [26], [3] or [23]. In reference [22] an exhaustive discussion of wavelet applications in signal processing is given. The two-dimensional *J*-step-DWT developed in sect. 3.4.2 is not the only way to define a discrete wavelet transform for images. "Isotropic wavelets", as constructed by P. Maaß, are an alternative approach. For a review refer to [21]. Another topic not covered in this book is the lifting scheme developed for the DWT. It addresses in particular implementations of the decomposition and reconstruction algorithm, which very efficiently reduce storage and computing time requirements. The lifting scheme is described in [16].

# Exercises

- 1. Verify eq. 3.12 in the following steps:
  - a) Sketch  $\frac{1}{\sqrt{|a|}}\psi_H\left(\frac{u-t}{a}\right)$  for a=2 and t=2k  $(k=0,\ldots,3)$ .
  - b) Decompose f(t) in the form  $f(t) = f^{1}(t) + d^{1}(t)$  and show that  $L_{\psi_{H}}f^{1}(2,2k) = 0 \ (k = 0, \dots, N-1).$
  - c) Show that  $L_{\psi_H} d^1(2, 2k) = \sqrt{\frac{2}{c_{\psi_H}}} d_k^1 \ (k = 0, \dots, N-1).$
- 2. Compute frequency response, amplitude response and phase response for the Haar-filter  $\mathcal{H}$  with coefficients

$$\left\{h_0 = \frac{\sqrt{2}}{2}, h_1 = \frac{\sqrt{2}}{2}\right\}.$$

These transfer functions are defined in eqs. 5.18 and 5.19, respectively. Compute these quantities also for the corresponding  $\mathcal{G}$ -filter with coefficients

$$\left\{g_0 = \frac{\sqrt{2}}{2}, g_1 = -\frac{\sqrt{2}}{2}\right\}.$$

3. For N = 4, db1 and J = 1 compute the one-dimensional wavelet basis

$$\phi^0_{wavelet}, \phi^1_{wavelet}, \phi^2_{wavelet}, \phi^3_{wavelet}$$

Steps:

a) Choose the natural basis

$$\tilde{\phi}^0_{natural}, \tilde{\phi}^1_{natural}, \tilde{\phi}^2_{natural}, \tilde{\phi}^3_{natural}$$

and decompose each of these sequences analogously to eq. 3.49, where the decomposition for J = 2 was sketched.

- b) For each basis sequence  $\tilde{\phi}^i_{natural}$  apply the reconstruction part of eq. 3.30 to the corresponding decomposition. For the coefficients belonging to the dual filters  $H^*$  and  $G^*$ , respectively, cf. 3.19 and 3.20!
- 4. Some MATLAB-exercises.
  - a) Develop a MATLAB-function with the following prototype call: transfers(coeff);

Here the row-vector coeff is constructed as follows:

coeff=[coefficients, lower, upper].

The vector coefficients stores the  $h_k$ -coefficients of a dbn-H-filter, lower denotes the lowest, upper the highest k-value. As an example, for db1 we have coeff=[0.7071, 0.7071, 0, 1.0000]. The vector coefficients can be obtained by the MATLAB Wavelet Toolbox function wfilters (cf. sect. 3.4.3), note that for a dbn-filter lower=0 and upper=2\*n-1.

Then transfers should compute and plot the corresponding amplitude and phase transfer functions (cf. eq. 5.18). Furthermore, the  $g_k$ coefficients should be determined according to eq. 3.28 and their transfer functions also should be computed and plotted.

b) Develop a MATLAB-function with the following prototype call:

out=imdec(original,J,'dbn');

Here the matrix **original** stores a gray level image, J denotes the decomposition depth of a J-step-DWT and **dbn** chooses the dbn-filters. Then **out** should store the output matrix arranged as indicated in eq. 3.37 (see also Fig. 3.17). The function should invoke the MATLAB Wavelet Toolbox function **wavedec2** with specification given in sect. 3.4.3.

c) (Involved!) Develop your own MATLAB-function realizing the iterative construction and plotting of scaling function  $\phi$  and wavelet  $\psi$ , when the filter coefficients  $\{h_k\}$  are given. The procedure ("cascade algorithm") is described in sect. 3.3.1.

Solutions to these problems can be found in folders Filters (a), TwoDim (b) and WaveIter (c). These folders may be downloaded the URL given in sect. 1.3.

5. (Involved!) In order to solve the problem formulated below we have to extend theorem 5.1 from sect. 5.2.

Let  $\{f(kT_S)\}$  be a sampling of a band-limited finite energy signal f(t) such that the Shannon condition 5.12 is fulfilled. Then not only the spectrum  $\hat{f}(\omega)$  and the DFT are related by eq. 5.16, moreover, the continuous-time signal f(t) may be recovered completely from the sample values  $\{f(kT_S)\}$ :

$$f(t) = \sum_{k} f(kT_S)\phi_{Sh}(t - kT_S).$$
 (3.56)

Here, the Shannon interpolation function is defined as follows:

$$\phi_{Sh}(t) = \frac{\sin\left(\pi\left(\frac{t}{T_S}\right)\right)}{\pi\left(\frac{t}{T_S}\right)}.$$
(3.57)

In what follows, denote with  $\phi(t)$  the Shannon interpolation function  $\phi_{Sh}(t)$  for  $T_S = 1$ , i.e.

$$\phi(t) = \frac{\sin(\pi t)}{\pi t}.$$
(3.58)

3 The Discrete Wavelet Transform

Show that

$$\phi(t) = \sqrt{2} \sum_{k} h_k \phi(2t - k) \text{ where } h_k = \frac{\phi(\frac{k}{2})}{\sqrt{2}}.$$
 (3.59)

Steps:

a) Show that

$$\hat{\phi}(\omega) = \begin{cases} 1 \ |\omega| \le \pi \\ 0 \ \text{else} \end{cases}$$

*Hint*: Use reconstruction formula 5.2!

b) Show that the maximum frequency  $\omega_{max}$  belonging to  $\phi(t)$  satisfies the Shannon condition 5.12 for  $T_S = \frac{1}{2}$ .

c) Put  $f(t) = \phi(t)$  and write down eq. 3.56 for  $T_S = \frac{1}{2}$ .

Supplementary information:

Remember that wavelet and scaling function are related by eq. 3.29, which is called a "scale-equation". It may be shown that the scaling function fulfills a similar scale equation:

$$\phi(t) = \sqrt{2} \sum_{k} h_k \phi(2t - k).$$

As an example, the reader may verify this relation for the Haar-wavelet and the corresponding filter coefficients  $\{h_k\}$ .

Then from eq. 3.59 follows that  $\phi$ , as defined in eq. 3.58, may be viewed as a scaling function related to the above filter coefficients. These filters in turn lead to a general J-step-DWT 3.30. In contrast to dbn-wavelets, however, the coefficient set

$$\{h_k\} = \left\{\frac{\phi(\frac{k}{2})}{\sqrt{2}}\right\}$$

will be infinite! Thus, any implementation on a computer necessarily will lead to truncation errors.

94

# More Applications

In this chapter we shall consider two DWT-applications: data compression and the use of wavelets for automatic indexing of images in image databases.

In the next section the general structure of the transform compression scheme is presented. We shall sketch the basic components and in particular will focus on the mechanism responsible for distortions when compressing data. In sect. 4.1.3 we describe a procedure for reducing these distortions. This procedure is implemented in MATLAB.

Section 4.2 is devoted to wavelet applications for image databases.

# 4.1 The Transform Compression Scheme

Any data compression scheme aims at reducing storage requirements. The data considered in this section are digital images<sup>1</sup>, which are represented as matrices

$$f = \{f_{mn}\} \ (m = 0, \dots, M - 1; \ n = 0, \dots, N - 1)$$

$$(4.1)$$

(cf. sect. 3.4.2). Here, M and N denote the number of rows and columns, respectively, the matrix entries  $f_{mn}$  are called "pixels". If a single pixel requires b bits of storage, the digital image f will require  $S_f$  bits of storage, where

$$S_f = NMb. \tag{4.2}$$

Instead of storage we shall also use the phrase "file size" from now on. So the file size of the original image f is computed according to eq. 4.2.

<sup>&</sup>lt;sup>1</sup> Note, however, that this restriction is made only for keeping notation consistent. The presented algorithms and techniques are also applicable to one-dimensional signals, i.e., sequences  $f = \{f_k\}_{k=0}^{N-1}$ .

#### 96 4 More Applications

To motivate the need of compression techniques, we start with a simple example. A wide-spread image format is the "SIF-format" with a resolution of M = 352 rows and N = 288 columns. It is used, for example, in low-cost web-cameras. A true-color image requires 24 bits for each pixel. So a true color image f recorded with SIF resolution will have a file size  $S_f$  computed as

$$S_f = 352 \times 288 \times 24 \ bits = 2433024 \ bits$$

Note that the SIF-resolution is rather poor! It is clear that with increasing resolution demands the volume of digitized images will increase drastically.

It is not only consumption of disk space which causes problems when processing digital images or video sequences. Downloading images from the internet, transmitting images over local or wide area networks will be extremely time consuming if the data volume is too large. Therefore, there is an urgent need for compression techniques leading to a reduced amount of data with only small or even insignificant distortions of the compressed images.

We close these introductory remarks by defining compression rate, formulating the goal of an effective compression procedure and introducing two commonly used distortion measures.

Every compression procedure transforms the original image f to a signal  $f^c$  with reduced file size. The "compression rate" measures this reduction and is defined as

$$C = \frac{S_f}{S_{f^c}}.$$
(4.3)

We shall not restrict ourselves to lossless compression procedures. So in general from  $f^c$  a signal will be recovered ("decompressed"), which we shall denote with  $f^{dec}$ . This signal should be "close" to the original f in a suitable sense (see below). Therefore, an efficient compression scheme is characterized by a large compression rate combined with only small signal distortions:

Compression goal: 
$$C \gg 1$$
 where  $f^{dec} \approx f$ . (4.4)

The following distortion measures give a precise meaning to  $f^{dec} \approx f$ : The "norm distance" of  $f^{dec}$  with respect to f is defined as follows:

$$\|f^{dec} - f\| = \sqrt{\sum_{mn} (f_{mn} - f^{dec}_{mn})^2}.$$
(4.5)

It is clear that a lossless compression procedure will lead to  $||f^{dec} - f|| = 0$ .

Another quantity, frequently used in particular in the engineering community [36], is the "peak-signal-to-noise-ratio" (PSNR), which is measured in Dezibel and reads 4.1 The Transform Compression Scheme 97

$$PSNR(f, f^{dec}) = 10 \log_{10} \left( \frac{MN(2^b - 1)^2}{\|f^{dec} - f\|^2} \right) [dB].$$
(4.6)

Here, as mentioned above, M and N denote the number of rows and columns, respectively, b is the number of bits required for storing a single pixel of the original f,  $||f^{dec} - f||$  denotes the norm distance introduced in eq. 4.5.

For readers familiar with the notion of signal energy we give a short explanation of the phrase "peak-signal-to-noise-ratio": When considering gray scale images with a resolution of b bits per pixel,  $2^b - 1$  will be the maximum intensity value of a pixel. Thus the above fraction relates the maximum energy, which may be stored in a  $M \times N$ -image to the energy stored in the error signal  $f^{dec} - f$ . This error signal may be attributed to noise, coming from the compression procedure.

Summarizing, an efficient compression scheme in the sense of eq. 4.4 will tend to minimize  $||f^{dec} - f||$ , or, equivalently, maximize  $PSNR(f, f^{dec})$ , at the same time keeping the compression rate C large.

A widespread compression procedure is the "transform compression scheme" and we shall describe its basic components in the next section.

# 4.1.1 The General Procedure

A "transform compression scheme" is a three-step-procedure as sketched below:

Compression: 
$$f \longrightarrow T \xrightarrow{f^T} Q \xrightarrow{f^Q} EC \longrightarrow f^c$$
 (4.7)

Decompression: 
$$f^{dec} \longleftarrow \boxed{T^{-1}} \stackrel{f^{Q^{-1}}}{\longleftarrow} \boxed{Q^{-1}} \stackrel{f^{EC^{-1}}}{\longleftarrow} \boxed{EC^{-1}} \longleftarrow f^c$$
(4.8)

The processing steps T, Q and EC must be chosen and adapted to each other such that requirement 4.4 for an efficient compression scheme is fulfilled. They are explained below:

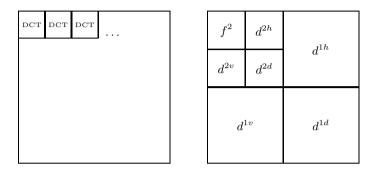
1. T denotes a linear invertible transform

$$f \leftrightarrow f^T$$
.

 $f^T$  again will be a matrix of the same<sup>2</sup> size  $M \times N$  as the original f. In this section we shall consider the *J*-step-DWT 3.36, since it is included in the actual JPEG 2000 standard for still image compression [36]. The remaining components of a transform compression scheme, however, work

 $<sup>^2</sup>$  For reasons explained in sect. 3.4.3 this only approximately will be true, if  $f^T$  is generated by a J-step-DWT.

with any linear and invertible transform. In the "old" version of the JPEG still image compression standard [39], for example,  $f^T$  resulted from a blockwise application of the "discrete cosine transform" (DCT) [39]. In Fig. 4.1 we display both  $f^T$ -variants. The result of a *J*-step-DWT again is arranged as indicated in eq. 3.37. Here J = 2 was chosen.



**Fig. 4.1.** Transformed images  $f^T$  for JPEG "old" (left) and JPEG 2000 (right).

As indicated already in sect. 3.4.4, JPEG 2000 recommends the use of a biorthogonal filter pair  $H, \tilde{H}$  with 9 and 7 coefficients, respectively.

2. Q denotes "quantization". Here *intervals* of the values of  $f^T$ -entries are represented by a *single* value (a so called "symbol")  $S_j$ . If the intervals have constant length, the quantization is called "uniform". An example is shown in Fig. 4.2. Every  $f^T$ -entry is subject to such a quantization leading to a matrix  $f^Q$  of size  $M \times N$ .

Since the map  $[q_j, q_{j+1}) \mapsto S_j$  is many-to-one, quantization is a *lossy* step. In fact, the "reconstruction error"  $f^{dec} - f$  actually is a quantization error, since quantization is the only lossy step at transform compression. Thus in eq. 4.8 the symbol  $Q^{-1}$  does not denote exact inversion of the quantization step Q but rather is a suitable prescription, to choose a value in the interval  $[q_i, q_{i+1})$ , when  $S_i$  is given.

Note that the truncation prescription discussed in sect. 3.6.1 (see page 84) actually is a quantization, representing the interval  $(-\tilde{f}^{limit}, \tilde{f}^{limit})$  by the symbol 0.

3. EC denotes an "entropy coder", which encodes the symbol matrix  $f^Q$  with as few bits as possible resulting in the compressed signal  $f^c$ . Note that, whichever way the entropy coder is implemented, entropy coding is invertible, therefore the transition from  $f^Q$  to  $f^c$  and vice versa is lossless:

$$f^Q \leftrightarrow f^c$$
.

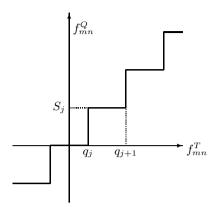


Fig. 4.2. A uniform quantization rule.

Moreover, the actual file size reduction is performed in this step and the purpose of the preceding steps is to generate a histogram of the symbols  $\{S_j\}$  such that entropy coding may be applied effectively. Basic notions of entropy coding are introduced below, for a review of entropy coding schemes see [36], [21].

Finally we note that the three-step-scheme depicted in equations 4.7 and 4.8, respectively, is oversimplified. Certain "modes" of JPEG 2000 do not show a strict separation of quantization and entropy coding. Rather, these steps are merged to an entangled procedure [36]. Moreover, compression efficiency can be increased by combining quantization with run length coding [1]. There exist many more variants of transform compression which will enable high compression rates at low distortion.

Nevertheless, in this section we shall stay with the three-step-scheme 4.7, since it allows for a systematic investigation of quantization errors (cf. sect. 4.1.3). The reader should keep in mind, however, that real implementations of compression schemes are more involved and we refer to the quoted references for more detailed information.

# 4.1.2 Entropy Coders

Let us assume that a quantization has been performed leading to n symbols. Therefore, after quantization, the quantized matrix  $f^Q$  will have entries with values taken from the set  $\{S_1, \ldots, S_n\}$ .

Assume that  $n_i$  denotes the number of  $S_i$ -occurrences in  $f^Q$  and denote with

$$p_i = \frac{n_i}{MN} \tag{4.9}$$

the corresponding relative frequency. Then the symbols  $S_i$  (i = 1, ..., n) will occur with relative frequencies  $p_i$  (i = 1, ..., n) satisfying

$$\sum_{i=1}^{n} p_i = 1. \tag{4.10}$$

The "entropy" of  $f^Q$  is defined as [36]

$$H(p_1, \dots, p_n) = -\sum_{i=1}^n p_i \log_2 p_i$$
(4.11)

H satisfies the inequalities

$$0 \le H(p_1, \dots, p_n) \le \log_2 n. \tag{4.12}$$

We explicitly note two important facts: The upper bound will be reached for

$$p_1 = p_2 = \ldots = p_n = \frac{1}{MN},$$
 (4.13)

the lower bound for

$$p_i = \begin{cases} 1 \ i = k\\ 0 \ i \neq k \end{cases} \tag{4.14}$$

Both equations illustrate an important aspect of entropy: If all symbols occur with equal frequency (eq. 4.13), H will reach its maximum; if actually only symbol  $S_k$  occurs in  $f^Q$  (eq. 4.14), H will reach its minimum. This may be generalized by the following observation:

H measures the concentration properties of the histogram counting the occurrences of the symbols  $S_i$  (i = 1, ..., n). If the histogram is flat, H will be large, if the histogram is sharply peaked, i.e., concentrated around relatively few symbols, H will be small.

For storing and processing  $f^Q$  on a digital computer, we need a "binary code", i.e., an invertible map of the symbols to sequences of two-valued bits. Usually the two possible values of a bit are represented as 0 and 1, respectively. A binary sequence representing a symbol will be called a "codeword". The length of the codeword representing symbol  $S_i$  we denote with  $l_i$ .

A binary coding of three symbols is shown in Table 4.1. As in this example, the length of the codewords in general may vary, a code with this property is called "variable length code".

We saw above that each symbol  $S_i$  will appear in  $f^Q$  with a relative frequency  $p_i$  resulting from eq. 4.9. So we can compute an "average codeword length"  $\overline{l}$  corresponding to

Table 4.1. A binary code for three symbols.

Symbol codeword				
$S_1 \\ S_2$	$\begin{array}{c} 1 \\ 01 \end{array}$			
$S_3$	00			

$$\overline{l} = \sum_{i=1}^{n} p_i l_i. \tag{4.15}$$

A fundamental theorem of information theory [29] states that for an arbitrary binary code the following inequality will be true:

$$\overline{l} \ge H(p_1, \dots, p_n). \tag{4.16}$$

Thus, the entropy computed according to eq. 4.11 is a lower bound for the average codeword length and a binary code, which has the property that  $\overline{l} \approx H(p_1, \ldots, p_n)$ , is called "entropy codec". Such codecs<sup>3</sup> also are called "optimal codecs", since it is not possible to represent the symbols (in the average) by shorter codewords.

We shall see below that algorithms for entropy coders can be given, so we may extract a "compression recipe" from what we have collected so far:

- 1. Try to reduce entropy by choosing a transform T such that the histogram of  $f_{mn}^T$ -values is concentrated about relatively few values (cf., e.g., Fig. 3.25).
- 2. Try to reduce entropy even more by applying a proper quantization rule such that the histogram of  $f_{mn}^Q$ -values is sharper peaked than the  $f_{mn}^T$ -histogram.
- 3. Apply an entropy codec to  $f^Q$  resulting in the compressed signal  $f^c$ .

As a consequence, the compression rate C defined in eq. 4.3 may be estimated as

$$C \approx \frac{b}{H(p_1, \dots, p_n)}.$$
(4.17)

The reason is that the file size of  $S_{f^c}$  of  $f^c$  may be estimated as  $S_{f^c} = MN\overline{l} \approx MNH(p_1, \ldots, p_n)$ , whereas the file size of the original reads  $S_f = MNb$  (cf. eq. 4.2, b denotes the number of bits needed to store a pixel of the original image).

 $<sup>^3</sup>$  The term "codec" is composed from coding and decoding stressing code invertibility.

## 102 4 More Applications

We shall now sketch two entropy codecs frequently used in image compression. Both algorithms are described in, e.g., [36]. The reader may consult [36] also for further related references.

# The Huffman-codec

Given the symbols  $\{S_1, \ldots, S_n\}$  and the corresponding relative frequencies  $\{p_1, \ldots, p_n\}$  the algorithm constructs a binary tree, whose leaves correspond to the symbols. The construction proceeds as follows:

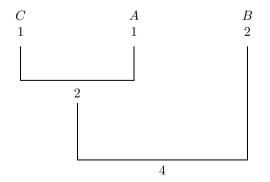
- 1. Define the "leaves" of the tree by putting all symbols in a row. Instead of leaf we will use the term "node" from now on. Each node will be labelled with the corresponding relative frequency  $p_i$ .
- 2. Merge the two nodes with the smallest probabilities ("children") into one node ("parent"), labelled with the sum of the two relative frequencies. Represent the merging process by a graph connecting both children with the parent node.
- 3. Repeat step 2 with the parent node(s) generated so far and the remaining leaves.
- 4. Stop, when the last node has been generated by the merging process. This node is called "root", obviously it will be labelled with 1, since it corresponds to the sum over all relative frequencies.
- 5. Code each symbol by the path from the root to the corresponding leaf, thereby coding each "left turn" with 0, each "right turn" with 1.

We illustrate the procedure by constructing the Huffman code for the symbol string ABBC.

The relative frequencies read

$$p_A = \frac{1}{4}, \ p_B = \frac{1}{2}, \ p_C = \frac{1}{4}$$

The tree-building process discussed above may be illustrated as follows:



Note that here the nodes are labelled with absolute rather than relative frequencies. This makes no difference, the root node for this example then necessarily is labelled with 4.

The resulting code reads:

Symbol codeword				
С	00			
Α	01			
В	1			

Table 4.2.	Huffman	code for	ABBC.
------------	---------	----------	-------

We finally check that for this example the average codeword length is determined by the entropy.

The entropy is computed as

$$H(p_A, p_B, p_C) = -\frac{1}{4} \log_2\left(\frac{1}{4}\right) - \frac{1}{2} \log_2\left(\frac{1}{2}\right) - \frac{1}{4} \log_2\left(\frac{1}{4}\right)$$
  
= 1.5

The coded string reads 011100. Therefore, the average codeword length results from

$$\overline{l}_{Huffman} = \frac{6}{4}$$
$$= 1.5$$

So indeed the Huffman code reaches the lower bound from eq. 4.16.

Obviously each Huffman code will lead a "codebook" associating to each symbol the corresponding codeword as in Table 4.2. So the minimum length of each codeword will be 1 and therefore, regardless of the symbol probabilities, the average codeword length of an arbitrary Huffman code will fulfill

$$\overline{l}_{Huffman} \ge 1. \tag{4.18}$$

On the other hand, the symbol probabilities  $\{p_1, \ldots, p_n\}$  may be such that entropy satisfies

$$H(p_1, \dots, p_n) < 1.$$
 (4.19)

In fact, if in a transform compression scheme the transform T is a J-step-DWT, or a two-dimensional DFT, after quantization the resulting histogram may be so sharply concentrated (cf. Fig. 3.25) such that eq. 4.19 will hold.

## 104 4 More Applications

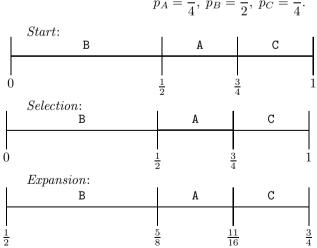
In this case obviously Huffman coding cannot be optimal, since it will not reach the lower bound from eq. 4.16. Any coding scheme reaching this bound, therefore necessarily will associate the symbol string stored in  $f^Q$  completely to a binary codeword. Below we shall describe such a coding procedure.

# Arithmetic Coding

Given the symbols  $\{S_1, \ldots, S_n\}$  and the corresponding relative frequencies  $\{p_1, \ldots, p_n\}$  the algorithm will code the given symbol string (in a transform compression scheme this string may result, e.g., from a rowwise scan of  $f^Q$ ) by a binary sequence, which we shall call "string codeword" below. We will describe now the procedure.

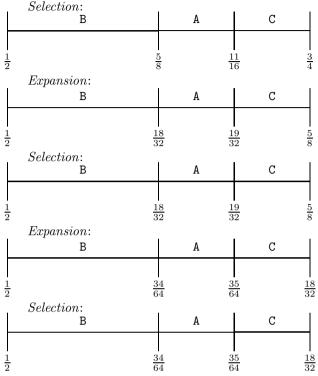
- 1. Start: Subdivide the interval [0, 1) into non-overlapping intervals of length  $p_i$  (i = 1, ..., n). Associate each symbol  $S_i$  with the corresponding subinterval.
- 2. *Selection*: Select the subinterval corresponding to the first symbol of the string.
- 3. *Expansion*: Decompose the selected interval into subintervals associated to the individual symbols completely analogously to step 1.
- 4. Repeat selection and expansion until the last symbol of the string has been processed. In this way we will obtain a final interval [u, o). Now choose a number in this interval such that its binary expansion<sup>4</sup> is as short as possible. This binary expansion will be the string codeword.

Again we shall illustrate the procedure by determining the arithmetic code of ABBC with relative frequencies



 $p_A = \frac{1}{4}, \ p_B = \frac{1}{2}, \ p_C = \frac{1}{4}.$ 

<sup>&</sup>lt;sup>4</sup> Readers not familiar with computing the binary expansion of a decimal number q with  $0 \le q < 1$  are referred to sect. 4.3, eq. 4.35, for an example.



We obtain the final interval

$$[u, o) = \left[\frac{35}{64}, \frac{18}{32}\right).$$

The binary expansion of  $u=\frac{35}{64}$  reads .100011. Analogously, the binary expansion of  $o=\frac{18}{32}$  is given by .1001.

So the string codeword representing  $\tt ABBC$  reads 100011. The average codeword length is computed as

$$\overline{l}_{AC} = \frac{6}{4} = 1.5$$

Here, obviously arithmetic coding and Huffman coding both are optimal, since their respective average codeword lengths are given by the entropy (see above).

We shall now consider the following string:

$$A \underbrace{B...B}_{n-1 \text{ times}}$$
(4.20)

#### 106 4 More Applications

The string will have a total number of n symbols and therefore the relative frequencies read

$$p_A = \frac{1}{n}, \qquad \qquad p_B = \frac{n-1}{n}.$$

If n is large enough, entropy will fulfill  $H(p_A, p_B) < 1$  and we shall investigate how arithmetic coding performs.

To do so, we need the following result: Arithmetic coding of the symbol string 4.20 will lead to the final interval [u, o), where

$$u = \frac{1}{n} \left( 1 - \left( 1 - \frac{1}{n} \right)^{n-1} \right), \qquad o = \frac{1}{n}.$$
(4.21)

To prove this, we consider Table 4.3, where we start with u = 0 and o = 1. Each selection will update u, o and the width of the actual interval; in the table these updates are tracked.

Table 4.3. Arithmetic coding of string 4.20.

Symb	pol $u$	0	Interval width
A	0	$\frac{1}{n}$	$\frac{1}{n}$
В	$\frac{1}{n^2}$	$\frac{n}{1}$	$\frac{n-1}{n^2}$
В	$\frac{1}{n^2} + \frac{n-1}{n^3}$	$\frac{1}{n}$	$(n-1)^{-2}$
В	$\frac{\frac{1}{n^2} + \frac{n-1}{n^3}}{\frac{1}{n^2} + \frac{n-1}{n^3} + \frac{(n-1)^2}{n^4}}$	$\frac{1}{n}$	$\frac{\binom{n^3}{(n-1)^3}}{n^4}$
÷	:	÷	:

Thus finally

$$u = \frac{1}{n^2} \left( 1 + \frac{n-1}{n} + \frac{(n-1)^2}{n^2} + \dots + \frac{(n-1)^{n-2}}{n^{n-2}} \right)$$
  
=  $\frac{1}{n^2} \left( 1 + \left( 1 - \frac{1}{n} \right) + \left( 1 - \frac{1}{n} \right)^2 + \dots + \left( 1 - \frac{1}{n} \right)^{n-2} \right)$   
=  $\frac{1}{n^2} \frac{1 - \left( 1 - \frac{1}{n} \right)^{n-1}}{\frac{1}{n}}$   
=  $\frac{1}{n} \left( 1 - \left( 1 - \frac{1}{n} \right)^{n-1} \right)$ 

From the second to the third line we used the geometrical expansion formula

4.1 The Transform Compression Scheme 107

$$\sum_{k=0}^{n} q^{k} = \frac{1-q^{n+1}}{1-q} \ (q \neq 1).$$

As an example we study the symbol string ABBBBBBB with relative frequencies

$$p_A = \frac{1}{8}, \qquad \qquad p_B = \frac{7}{8}$$

The entropy reads

$$H(p_A, p_B) = -\frac{1}{8}\log_2\left(\frac{1}{8}\right) - \frac{7}{8}\log_2\left(\frac{7}{8}\right) = 0.5436$$

From eq. 4.21 the boundaries of the final interval are computed as:

$$[u, o) = \left[\frac{185}{2437}, \frac{1}{8}\right)$$

The binary expansion of  $u = \frac{185}{2437}$  is given by .000100110110111100001001. Correspondingly,  $o = \frac{1}{8}$  is expanded as .001. From both expansions we obtain for the string codeword the binary se-

quence 00011.

Therefore the average codeword length reads

$$\overline{l}_{AC} = \frac{5}{8} = 0.6250$$

In contrast, since the Huffman code encodes A with 1 and B with 0 (or vice versa), we have

$$\overline{l}_{Huffman} = 1.$$

So arithmetic coding will be much closer to the lower bound from eq. 4.16 than Huffman coding.

Since the intervals generated throughout the coding procedure are nested, decoding works as follows: Subdivide the interval [0,1) as described above and determine the subinterval containing the number associated with the string codeword. The corresponding symbol will be the first element of the decoded symbol string. Then take this interval, subdivide it again, and again determine the subinterval containing the above number. This will produce the second element of the decoded string. Proceed in this way and produce the next symbols. Since this process will not terminate, the total number of symbols must be transmitted in order to stop the decoding process after the last symbol.

#### 4.1.3 Optimal Quantization and Examples

We return now to the transform compression scheme described in eqs. 4.7 and 4.8. In sect. 4.1.1 we saw that signal distortions exclusively can be attributed to quantization and in this section we shall investigate how quantization distortion may be reduced.

Given the original f, we are interested in the distortion of the decompressed signal  $f^{dec}$ . We saw above that entropy coding is reversible, therefore we may combine equations 4.7 and 4.8 such that we obtain  $f^{dec}$  from f as follows:

Compression: 
$$f \longrightarrow T \longrightarrow Q \longrightarrow Q^{-1} \longrightarrow T^{-1} \longrightarrow f^{dec}$$
 (4.22)

The "norm distance"

$$D := \|f^{dec} - f\| \tag{4.23}$$

of  $f^{dec}$  with respect to f, which was introduced in eq. 4.5, measures distortion. A high *D*-value corresponds to large, a low *D*-value to small signal distortion. What are the parameters, *D* depends on?

In Fig. 4.2 we saw that a quantization rule corresponds to a piecewise constant function, which uniquely is determined when the quantization parameters  $q_j$  and the levels  $S_j$  on the respective intervals  $[q_j, q_{j+1})$  are known. This function will be called "quantization function". Then we observe the following:

Assume that there are n + 1 quantization parameters  $q_1, \ldots, q_{n+1}$ . Assume moreover that there exists a prescription, how the levels  $S_i$  are computed from the quantization parameters (for an example see below). Furthermore, assume that a certain choice of the transform T has been made. Then distortion as determined in eqs. 4.22 and 4.23 will depend only on the quantization parameters:

$$D = D(q_1, \dots, q_{n+1}) \tag{4.24}$$

In this section we shall use the following prescription to compute the levels ("symbols")  $S_j$ :

$$S_j = \frac{q_j + q_{j+1}}{2} \ (j = 1, \dots, n) \tag{4.25}$$

The quantization function is constructed as follows: On the intervals  $[q_j, q_{j+1})$  (j = 1, ..., n) the function value will be  $S_j$  (j = 1, ..., n), on  $[q_{n+1}, \infty)$  the function will remain on  $S_n$ , on  $(-\infty, q_1)$  the function value

is  $S_1$ . So after quantization the matrix elements of  $f^Q$  will have values<sup>5</sup> taken from the symbol set  $S_1, \ldots, S_n$ .

As mentioned in the preceding section, rowwise scanning of  $f^Q$  will produce a symbol string, where each  $S_i$  will appear with a relative frequency  $p_i$ . Since the symbols uniquely are computed from the quantization parameters as in eq. 4.25, the resulting entropy H (cf. eq. 4.11) actually will depend on these parameters:

$$H = H(q_1, \dots, q_{n+1}) \tag{4.26}$$

Distortion is minimized, if  $q_1, \ldots, q_{n+1}$  are chosen such that  $D(q_1, \ldots, q_{n+1})$  is minimized. On the other hand, eq. 4.17 tells us that at the same time  $H(q_1, \ldots, q_{n+1})$  should be kept as small as possible in order to obtain a large compression rate C. So, the overall optimization goal may be formulated as follows:

Determine  $q_1, \ldots, q_{n+1}$  such that the target function  $t(q_1, \ldots, q_{n+1})$  is minimized, where

$$t(q_1, \dots, q_{n+1}) = \lambda D(q_1, \dots, q_{n+1}) + (1 - \lambda) H(q_1, \dots, q_{n+1}) \quad (4.27)$$

The parameter  $\lambda$  is called "distortion weight",  $(1-\lambda)$  correspondingly "entropy weight".

So, the target function is a weighted average of distortion and entropy and it depends on the choice of  $\lambda$ , which of the conflicting "sub-goals" (distortion reduction vs. compression efficiency) is favored.

It is clear that there is no simple analytical expression for t. The optimization goal formulated above, therefore, must be solved numerically. A numeric procedure always will start from an initial set  $(q_1^0, \ldots, q_{n+1}^0)$  and iteratively will produce  $(q_1^k, \ldots, q_{n+1}^k)$   $(k = 1, 2, \ldots)$  converging to the desired values  $(q_1, \ldots, q_{n+1})$ , where t is minimal. In sect. 4.1.4 we briefly indicate which procedure has been used.

Before discussing Figs. 4.3 - 4.5, where results of numeric optimization are displayed, we make two remarks:

1. The initial set is computed according to

$$q_1^0 = \min_{mn}(f_{mn}^T) < q_2^0 < \ldots < q_{n+1}^0 = \max_{mn}(f_{mn}^T)$$
(4.28)

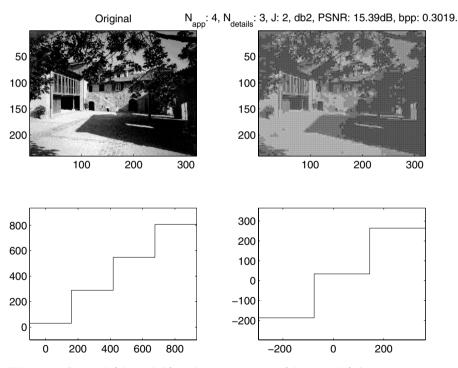
<sup>&</sup>lt;sup>5</sup> For completeness we mention that in eq. 4.22 we shall omit dequantization  $Q^{-1}$ . Thus, the inverse transform  $T^{-1}$  directly will be applied to  $f^Q$ . Since  $S_j$  is computed according to eq. 4.25, actually we perform a dequantization by selecting the midpoints of each quantization interval  $[q_j, q_{j+1})$ .

#### 110 4 More Applications

with  $q_{j+1}^0 - q_j^0 = \frac{q_{n+1}^0 - q_1^0}{n}$  (j = 1, ..., n). So we take the smallest and the largest element of the transformed image  $f^T$  (cf. eq. 4.7) and subdivide the resulting interval in n subintervals of equal length. Therefore the initial quantization will be uniform (cf. sect. 4.1.1).

2. For T we choose a J-step-DWT. Thus, the transformed image  $f^T$  is constructed as depicted in the right diagram in Fig. 4.1 for J = 2. We shall allow for separate quantization rules for the approximation signal  $f^J$  and the detail signals. So, in general the numbers of quantization levels for the approximation image  $f^J$  and the detail signals, respectively, will differ.

The upper row of Fig. 4.3 shows the original f (left) and the decompressed image  $f^{dec}$  (right). The lower left diagram displays the quantization function for the approximation signal  $f^J$ , the lower right diagram the quantization function for the detail signals. Here four quantization levels were chosen for the approximation signal, three quantization levels for the detail signals, respectively. The figure shows the results for the uniform start quantization.

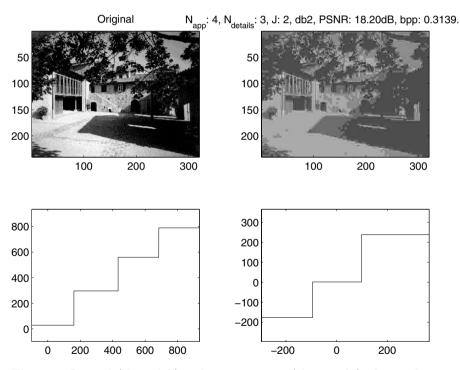


**Fig. 4.3.** Original (above left) and reconstruction (above right) for start quantization. Below left: Uniform initial quantization of approximation signal, Below right: Uniform initial quantization of detail signals.

It is clear that due to the low number of quantization levels distortion will be large. Distortion was measured by the PSNR (cf. eq. 4.6) (remember that large distortion corresponds to a low PSNR-value). Since entropy estimates the average codeword length of a symbol, H also is displayed. It is specified as the number of "bits per pixel" (bpp). The respective values in Fig. 4.3 are PSNR $(f, f^{dec}) = 15.39$  dB and H = 0.3019 bpp. The original was represented with b = 8 bits per pixel. So, we have an estimated compression rate of  $C = \frac{b}{H} = 26.5$ .

Before discussing Figs. 4.4 and 4.5, we note that the respective subimages are arranged as in Fig. 4.3. Moreover, PSNR- and bpp-values are determined analogously.

Figure 4.4 shows the results obtained when starting from the initial quantization from Fig. 4.3 and stopping after 60 iterations. In the target function 4.27  $\lambda = 1$  was chosen, i.e., the entropy weight was zero. So we only aimed at minimizing distortion without taking entropy into account.

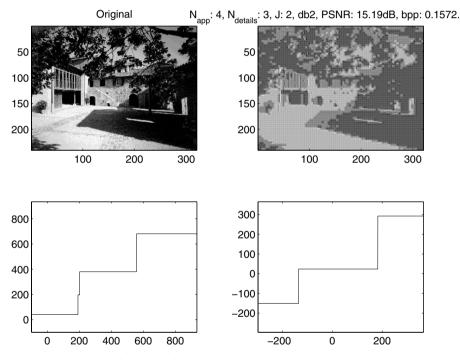


**Fig. 4.4.** Original (above left) and reconstruction (above right) after 60 function evaluations, entropy weight: 0. Below left: Resulting non-uniform quantization of approximation signal, Below right: Resulting non-uniform quantization of detail signals.

#### 112 4 More Applications

Obviously, the visual impression of  $f^{dec}$  is much better now than in Fig. 4.3. This is reflected also by the PSNR: Now we have  $PSNR(f, f^{dec}) = 18.20 \text{ dB}$ . As expected, the bpp-value is slightly larger than in Fig. 4.3: It reads H = 0.3139 bpp corresponding to an estimated compression rate  $C = \frac{b}{H} = 25.5$ . Note that the resulting quantization functions deviate from uniform quantization both for the approximation image and the detail signals!

In the final example we again started from the initial quantization from Fig. 4.3 and stopped after 60 iterations. But now in the target function 4.27  $\lambda = 0$  was chosen, i.e., the distortion weight was zero. So we only aimed at minimizing entropy (maximizing compression rate) without taking distortion into account. The results are displayed in Fig. 4.5.



**Fig. 4.5.** Original (above left) and reconstruction (above right) after 60 function evaluations, entropy weight: 1. Below left: Resulting non-uniform quantization of approximation signal, Below right: Resulting non-uniform quantization of detail signals.

Indeed now a low bit-per-pixel-value is obtained: We have H = 0.1572 bpp corresponding to an estimated compression rate  $C = \frac{b}{H} = 50.9!$  But for the PSNR-value we now obtain PSNR $(f, f^{dec}) = 15.19$  dB. So it decreased with respect to the initial quantization. As expected, the decompressed image  $f^{dec}$  deteriorates when compared with the initial quantization. It is instructive to

study the quantization curves. Above we saw that low entropy values correspond to sharply concentrated histograms. Such histograms can be reached by generating wide quantization intervals. The corresponding symbols then will appear frequently leading to concentrated histograms. Compare the quantization curves for the detail signals in Fig. 4.3 and Fig. 4.5, respectively! Here this broadening effect is clearly visible. Note also that the original 4-levelquantization of the approximation signal has turned basically to a 3-level quantization, i.e., essentially only three symbols remained! This again is in accordance with the above remarks about histogram concentration.

### 4.1.4 MATLAB Implementation

In this section we briefly describe a MATLAB implementation of the quantization optimization outlined in the preceding section. The related functions require at minimum the MATLAB Wavelet Toolbox and have been developed and tested with MATLAB 6.5, release 13. They may be downloaded from the URL given in sect. 1.3. Again, as mentioned earlier, a basic understanding of MATLAB-syntax is required.

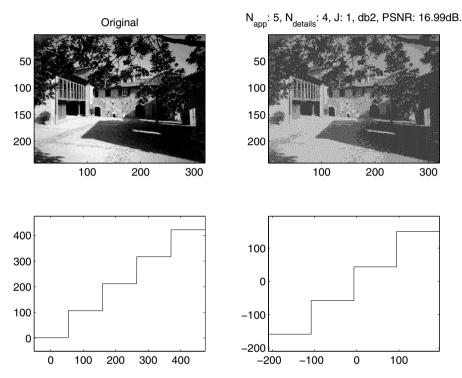
Quantization optimization is invoked by the following prototype function call:

#### [q\_approx\_signal,q\_detail\_signal]=optshell(f);

The original image matrix is stored in f. It must be provided as a  $M \times N$ matrix with double-entries. The parameters characterizing the transform T are selected interactively. These are decomposition depth J and selected wavelet string 'dbn', where n = 1, 2, ... (refer to sect. 3.4.3). Moreover, for the approximation signal and the detail signals the respective number of quantization levels ("symbols") must be specified. The respective initial quantization parameters are then computed corresponding to eq. 4.28. In this formula  $f^T$  must be replaced by the approximation image  $f^J$  in order to obtain the start quantization of the approximation signal; for the start quantization of the detail signals a corresponding replacement has to be done. The quantization parameters  $q_i$  resulting from optimization are returned in vectors  $q_approx_signal$  and  $q_detail_signal$  for approximation signal and detail signals, respectively.

As mentioned in the preceding section, beginning from the start quantization iteratively new quantization parameters are computed such that the resulting values of the target function 4.27 are essentially descending towards a minimal value (if it exists). The function optshell allows for choosing from two iteration methods. One method is the "Nelder-Mead" algorithm [19]. It is implemented in the MATLAB-built-in function fminsearch, which is invoked if Nelder-Mead is selected. The other method which can be selected, is the "Quasi-Newton" algorithm [2]. The selection invokes the function fminunc, which implements this algorithm. This function requires the installation of the MATLAB Optimization Toolbox. It therefore cannot be used if the MATLAB Optimization Toolbox is not available.

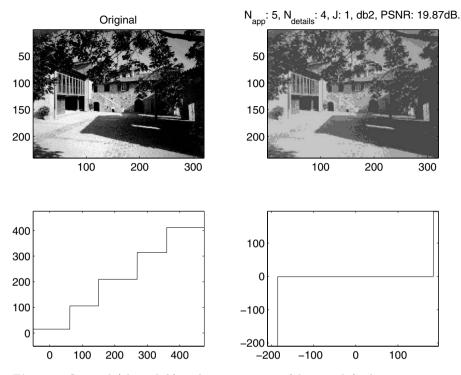
Figures 4.3 - 4.5 were obtained by invoking fminsearch. Below we show a result invoking fminunc. Here we chose five quantization levels for the approximation signal, four quantization levels for the detail signals. Figure 4.6 shows the results for the uniform start quantization.



**Fig. 4.6.** Original (above left) and reconstruction (above right) for start quantization. Below left: Uniform initial quantization of approximation signal, Below right: Uniform initial quantization of detail signals.

Figure 4.7 shows the results after a certain number of iterations. As in Fig. 4.4, entropy weight was set to zero. Referring to the target function 4.27, therefore, we aimed at reducing distortion without taking entropy into account.

The visual improvement corresponds to a clear improvement in PSNRvalues: In Fig. 4.6 we have PSNR $(f, f^{dec}) = 16.99$  dB, in Fig. 4.7 PSNR $(f, f^{dec}) =$ 19.87 dB was obtained. The bpp-values were as follows: For Fig. 4.6 H =1.6637 bpp was measured, corresponding to an estimated compression rate  $C = \frac{b}{H} = 4.81$ . In Fig. 4.7 we have H = 0.7182 bpp corresponding to a compression rate  $C = \frac{b}{H} = 11.14$ .



**Fig. 4.7.** Original (above left) and reconstruction (above right) after optimization run, entropy weight: 0. Below left: Resulting non-uniform quantization of approximation signal, Below right: Resulting non-uniform quantization of detail signals.

Thus, in this experiment not only distortion was reduced, optimization even lead to an improved compression performance!

We close this section with some general remarks concerning the optimization implementation. As stated above, the numerical procedures discussed in this section iteratively will produce quantization parameters such that the corresponding target function values will be descending, hopefully towards a local minimum situated in a neighborhood of the start quantization. Since no a-priori-information about the "shape" of the target function  $t(q_1, \ldots, q_{n+1})$ (cf. eq. 4.27) is given, it is not guaranteed that such a minimum exists. If it exists, it is not guaranteed that it will be reached within reasonable computing time. So, one usually limits the maximum number of iterations and will be satisfied if, within this number of iterations, the target function value will decrease reasonably. In sect. 4.3 we shall make some final comments on the optimization subject.

# 4.2 Wavelet-Based Similarity Retrieval in Image Archives

In this section we report on an industrial project performed by the author at tecmath AG. The work has been published in [31, 32]. In the following we summarize the main results.

Conventional digital image database management systems usually are structured as follows: Each stored image file is described by a text document, which contains administrative data like date and time of image capture. Moreover, this document includes well-defined terms and phrases describing image contents. When a user of the database queries for images related to a certain topic, he or she might enter corresponding keywords. These keywords can then be used for query mechanisms like, e.g., full text retrieval. So the result of the research will lead to a list of documents, whose content description parts contain the keywords. The retrieved documents are linked to the corresponding images, which then may be inspected.

It is well known that a verbal description of images often is inadequate, since the description stored in the database management system might not match the expectations of the user. Moreover, it is not possible to "present" an example image to the system and to look for all images "similar" to the presented image when conventional image archives as described above are used. Image database management systems allowing for such retrieval mechanisms are called "content-based access systems" (CBA-systems). It is clear that in view of multimedia technology CBA-systems are of growing importance. In order to build systems "looking for similar images", it is necessary to give a precise meaning to the notion of "similarity" and we shall describe a simple model based on a J-step-DWT.

As a preparation, we shall explore in some more detail, how a *J*-step DWT of an image may be used for measuring the appearance of edges.

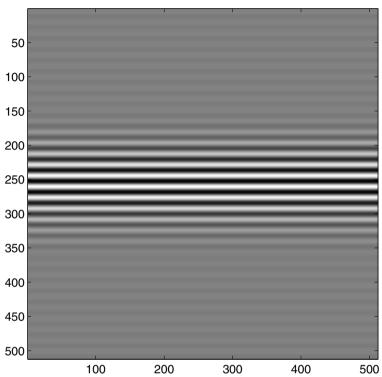
Remember that the result of a J-step DWT of an image (cf. eq. 3.36) usually is arranged as indicated in eq. 3.37. As an example consider the original f shown in Fig. 4.8.

A 7-step-DWT, performed with db2 is visualized in Fig. 4.9. The detail signals  $d^{jh}, d^{jv}, d^{jd}$  (j = 1, ..., 7) clearly confirm the directional sensitivity mentioned in sect. 3.4.2: Since the original shows a damped periodic pattern of *horizontal* stripes, vertical and diagonal detail signals are close to zero.

Moreover, we note that the detail signals will "react" in particular for j-values such that the associated scale factors  $2^{j}$  (measured in pixels) approximately correspond to stripe width<sup>6</sup>. The reaction may be measured by computing the detail signal variance<sup>7</sup>. This number estimates the average deviation of the detail signal from its mean value and therefore will be large for

<sup>&</sup>lt;sup>6</sup> As an example consider  $d^{3h}$  in Fig. 4.9

<sup>&</sup>lt;sup>7</sup> We remind the reader that the variance  $\sigma^2$  of a data set  $\{y_1, \ldots, y_n\}$  is computed as follows:  $\sigma^2 = \frac{1}{n-1} \sum_{i=1}^n (y_i - \overline{y})^2$ , where  $\overline{y} = \frac{1}{n} \sum_{i=1}^n y_i$ .



Period: 16 Pixels

Fig. 4.8.  $512 \times 512$ -Original. Stripe width: 16 pixels.

detail signals with large oscillations about their mean. For a  $J\-$  step-DWT we shall denote variances with

$$\sigma_{1h}^2, \sigma_{1v}^2, \sigma_{1d}^2, \dots, \sigma_{Jh}^2, \sigma_{Jv}^2, \sigma_{Jd}^2.$$
(4.29)

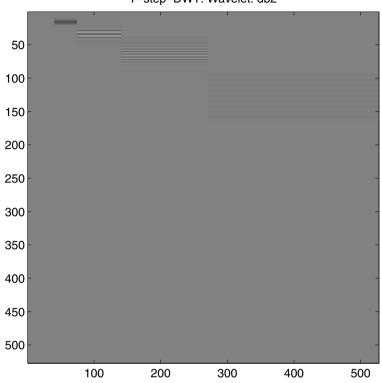
For three test images with increasing stripe width a 7-step-DWT with the db2-wavelet has been performed. The corresponding  $\sigma_{jh}^2$ -values are plotted in Fig. 4.10.

The reader may confirm that indeed the maximum of these values belongs to such a *j*-value that the corresponding scale factor  $2^j$  is approximately equal to the respective stripe width.

In addition, scale-dependent anisotropy-measures are defined as follows:

$$a_j = \frac{\sigma_{1v}^2}{\sigma_{1h}^2} \ (j = 1, \dots, J) \tag{4.30}$$

They indicate, if structures of size  $\approx 2^j$  are preferably vertically  $(a_j > 1)$  or horizontally  $(a_j < 1)$  oriented.



7-step-DWT. Wavelet: db2

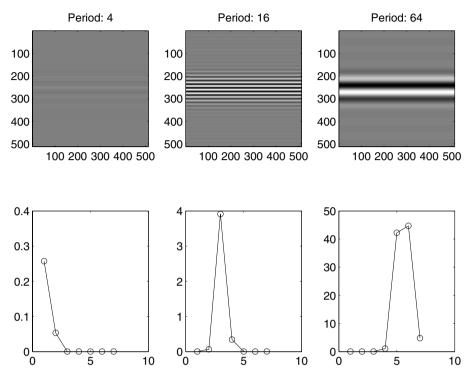
Fig. 4.9. 7-step-DWT of the image displayed in 4.8. Wavelet: db2.

Summarizing, the following procedure is suggested:

Given an image 
$$f$$
, define the "pattern vector"  
 $x^{f} = (\sigma_{1h}^{2}, \sigma_{1v}^{2}, \sigma_{1d}^{2}, a_{1}, \dots, \sigma_{Jh}^{2}, \sigma_{Jv}^{2}, \sigma_{Jd}^{2}, a_{J}).$  (4.31)  
The entries  $\sigma_{jh}^{2}$  indicate whether preferably horizontally oriented  
structures of size  $\approx 2^{j}$  (in pixels) are present in  $f$ . Analogously, the  
remaining variances are interpreted. Moreover, the entries  $a_{j}$  indicate  
anisotropy as stated above.

Two images  $f_1$  and  $f_2$  are considered to be "similar", if  $x^{f_1}$  is close to  $x^{f_2}$  in a suitable sense (see eq. 4.32).

The "distance" of two pattern vectors  $x^{f_1}$  and  $x^{f_1}$ , extracted from the original images  $f_1$  and  $f_2$ , respectively, is measured by the norm  $||x^{f_1} - x^{f_2}||$ , which is computed as follows (see also eq. 4.5):



**Fig. 4.10.** Top row: Original images with (from left) a stripe width of 4, 16 and 64 pixels, respectively. Below: corresponding plots of  $\sigma_{1h}^2, \ldots, \sigma_{7h}^2$  resulting from a 7-step-DWT with the db2-wavelet.

$$\|x^{f_1} - x^{f_2}\| = \sqrt{\sum_{i=1}^{4J} (x_i^{f_1} - x_i^{f_2})^2}.$$
(4.32)

Here  $x_i^{f_1}$  (i = 1, ..., 4J) denotes the *i*-th entry of the pattern vector  $x^{f_1}$ , correspondingly  $x_i^{f_2}$  is defined.

Based on this similarity measure, in [31], [32] a digital image archive is proposed that allows for retrieving images "similar" to a presented image. In the rest of this section we describe archiving and retrieval procedures and discuss examples<sup>8</sup>.

When *entering* an image f into the database management system, in addition to manual input of descriptive and/or administrative data the pattern vector  $x^{f}$  is computed, stored and linked to the image.

When querying the database for images similar to a presented image g (*similarity retrieval*), the following steps are performed:

<sup>8</sup> Figures 4.11 - 4.13 published with kind permission of tecmath AG.

#### 120 4 More Applications

- 1. Compute the pattern vector  $x^g$ .
- 2. Generate an "answer list" by sorting the stored images  $f_i$  such that

$$\|x^{g} - x^{f_{1}}\| \le \|x^{g} - x^{f_{2}}\| \le \|x^{g} - x^{f_{3}}\| \le \dots$$
(4.33)

3. Present the answer list  $f_1, f_2, f_3, \ldots$  to the user.

An example is shown in Fig. 4.11



Fig. 4.11. Similarity retrieval.

Here, the top row shows (from left) g and  $f_i$ , analogously the bottom row shows from left  $f_{i+1}$  and  $f_{i+2}$ . The user may scroll through this list starting with  $f_1$ . This is the situation displayed here: Rowwise from left to right images  $g, f_1, f_2$  and  $f_3$  are shown.

The test archive underlying Fig. 4.11 consists of  $\approx 300$  images. The archive exhibits a large variety concerning image contents: The images show natural scenes, have been recorded in a television-studio, result from medical applications (magnetic resonance imaging) and so on. A result of a query for magnetic resonance images similar to a given image g is shown in Fig. 4.12. It again shows  $g, f_1, f_2$  and  $f_3$  which are arranged as in Fig. 4.11.

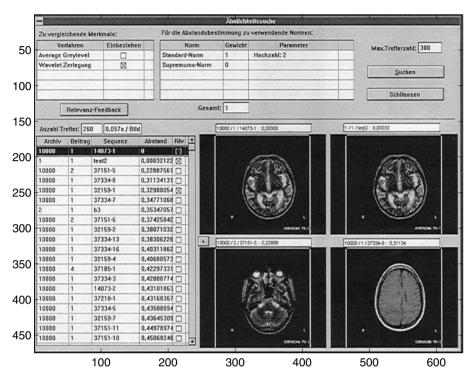


Fig. 4.12. Another retrieval result.

It is clear that there might be top-ranked images among the first positions of the answer set, which are not relevant for the user. In this case the retrieval iteratively can be improved by "relevance feedback" [28]. We describe now the procedure, again the given image will be denoted with g.

- 1. Let the user qualify the first positions of the answer list as "relevant" or "irrelevant". Denote with  $f_{mis}$  the "most misleading" irrelevant image, i.e., the irrelevant image having the topmost position in the answer list. Furthermore, denote with  $\mathcal{A}^{rel}$  the set of relevant images.
- 2. Start a new similarity retrieval run by replacing pattern vector  $x^g$  with  $x^{new}$  computed according to

$$x^{new} = \frac{x^g}{\|x^g\|} - \frac{x^{f_{mis}}}{\|x^{f_{mis}}\|} + \sum_{f^i \in \mathcal{A}^{rel}} \frac{x^{f_i}}{\|x^{f_i}\|}.$$
(4.34)

3. Inspect the newly generated answer list and proceed with step 1, if necessary.

In formula 4.34, for example,  $||x^g||$  is computed according to

$$\|x^g\| = \sqrt{\sum_{i=1}^{4J} (x_i^g)^2}$$

(cf. eq. 4.32). Analogously the other norm quantities are determined.

This procedure has been implemented in the system. Based on the retrieval result from Fig. 4.12 relevance feedback has been carried out. To generate the new retrieval run images  $f^1$  and  $f^4$  (not shown in Fig. 4.12) have been marked as relevant. This can be verified by checking the corresponding tick boxes in the answer list. So in this example  $f_{mis} = f^2$ .

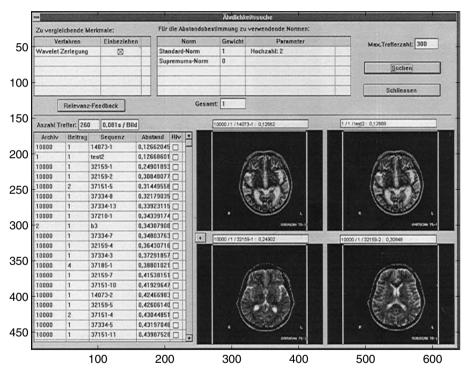


Fig. 4.13. Improvement by relevance feedback.

The newly generated answer list is displayed in Fig. 4.13 with  $g, f^1, f^2$ and  $f^3$  as arranged before. Obviously the result indeed improves retrieval performance with respect to similarity.

We close this section with some final comments. Of course, given two images f and g, it would be possible to measure their similarity by putting the raw data, i.e., the matrix elements  $f_{mn}$  and  $g_{mn}$ , directly into formula 4.5. The reason for using pattern vectors  $x^f$  and  $x^g$  instead, is reduction of computing time. If the images, for example, are stored in SIF-format (i.e.,  $352 \times 288$ ) the sum in 4.5 has to be taken over 101376 summands. In contrast, when choosing, e.g., J = 4, the pattern vectors will have 16 components. So the sum in formula 4.32 extends over only 16 summands. Obviously, it is a necessary prerequisite that the pattern vectors capture image features essential for similarity. Moreover, the computing effort to obtain the pattern vector from the original must be low. This is true for the two-dimensional *J*-step-DWT 3.36. The effort for computing this transform will increase only *linearly* with the number of matrix elements of the original. On a PC-platform with a Pentium II processor a similarity retrieval run on the test archive described above (consisting of  $\approx 300$  images) required only a few seconds.

# 4.3 Notes and Exercises

## Notes

The Huffman-codec first was published in [13]. It is the entropy coding scheme supported by the "baseline version" of the "old" JPEG still image compression standard. JPEG 2000 uses a special variant of the arithmetic coding scheme. This variant in particular allows for encoding symbol strings "online": It is not necessary to read the total string before the string codeword is produced. Moreover, a progressive transmission mode, i.e., image transmission with better and better resolution, is supported. In [40] a realization of an arithmetic coder on the basis of a field programmable gate array (FPGA) is described. The coder is adapted to a *J*-step-DWT.

"Rate-distortion-theory" [15] states that for a given distortion any compression scheme will produce a bpp-value bounded below by a function called "rate-distortion-function", which may be computed theoretically. Thus, methods like the optimization procedure from sect. 4.1.3 implement a practical approach to this theoretical limit. The reader should keep in mind, however, that the optimization algorithm described above primarily is intended as an easily accessible tool for illustrating basic facts of coding like the tradeoff of distortion reduction vs. compression efficiency. So it mainly serves didactic purposes.

In sect. 3.6.1 we already mentioned the need for "perceptually lossless quantization". This means that one looks for quantization functions such that all elements of a quantization interval  $[q_i, q_{i+1})$  are visually indistiguishable (this applies accordingly also to audio compression). It is still an open issue to define a quantitative measure for "perceptual similarity" of images. Distortion measures like the PSNR are widely used, but in order to decide on quantization errors on a *physiological* basis, it still is common practice to present images (or audio signals) to human "judges". Thus, automatic optimizers like the one described in sect. 4.1.3 are only one step in a complex testing and adaptation scenario.

For additional reading about similarity retrieval and content-based access to multimedia archives the reader is referred to [25],[27].

## 124 4 More Applications

# Exercises

- 1. Construct a Huffman code for the symbol string ABCCDDEE and compare the resulting average codeword length  $\overline{l}_{Huffman}$  with the entropy  $H(p_A, p_B, p_C, p_D, p_E)$ .
- 2. For generating a codeword with the arithmetic coder it will be necessary to convert a decimal number q with 0 < q < 1 to its binary representation. For q = 0.7 the algorithm is demonstrated below.

So the binary expansion of 0.7 reads  $.1\overline{0110}$  where the overline denotes the periodic pattern to be repeated infinitely.

Write a MATLAB-function (sample call: DecBinConv(q,n)), which reads a decimal number q with 0 < q < 1 and successively displays the single bits of its binary expansion on the screen! Use the algorithm sketched in eq. 4.35 and limit the maximum number of displayed digits to n.

3. Determine the arithmetic code for the symbol string ABCCDDEE and compare the resulting average codeword length  $\overline{l}_{AC}$  with the entropy  $H(p_A, p_B, p_C, p_D, p_E)$ .

*Hint*: To determine the selected intervals MATLAB's number format format rat is useful. It displays numbers as fractions of small integers.

 Write a MATLAB-function with sample call ArithBound(str,symb,freq);

String str is the input string. It consists of symbols which are stored in the string symb. Each element of symb appears in str with a frequency, which is a corresponding element of array freq.

Then the function should display the binary expansions of boundaries of the final interval produced by arithmetic coding of str on the screen! For computing the binary expansion use function DecBinConv sketched above. As an example, the coding of ABBC could be invoked by

ArithBound('ABBC', 'BAC', [2 1 1]); (cf. page 104).

# Appendix

# 5.1 Fourier Transform and Uncertainty Relation

In this section basic definitions and facts concerning Fourier transform are recalled. The main purpose is to provide the Fourier transform notions required for understanding the material covered in chap. 2. For more detailed information and for a deeper introduction to the topic the reader is referred, e.g., to [34].

Let f(t) denote a continuous-time signal. Moreover, assume that f(t) is a finite-energy-signal. This means that the following relation holds:

$$\int_{-\infty}^{+\infty} |f(t)|^2 \, dt < \infty.$$

For any practically relevant signal the finite energy assumption is fulfilled.

Then the "Fourier transform" is defined as follows:

$$\hat{f}(\omega) = \int_{-\infty}^{+\infty} f(t)e^{-j\omega t} dt$$
(5.1)

Thus, the Fourier transform is a function depending on the real variable  $\omega$ , called the "frequency".  $\hat{f}(\omega)$  again is a finite energy function on the frequency domain. It sometimes also is called "spectrum" of f(t).

The signal may be reconstructed from its spectrum:

$$f(t) = \frac{1}{2\pi} \int_{-\infty}^{+\infty} \hat{f}(\omega) e^{+j\omega t} d\omega$$
(5.2)

Equations 5.1 and 5.2 establish a one-to-one correspondence between a signal and its Fourier transform. A useful notation for this fact is to connect both entites to a "correspondence pair":

126 5 Appendix

$$f(t) \circ - \bullet \hat{f}(\omega) \tag{5.3}$$

Now let f(t) and g(t) denote two finite energy signals, such that the "convolution" of f with g, defined by

$$(f * g)(t) := \int_{-\infty}^{+\infty} f(u)g(t-u) \, du$$
 (5.4)

leads to a function  $f\ast g,$  which also has finite energy. Then the "convolution theorem" states that

$$(f * g)(t) \circ - \bullet \hat{f}(\omega)\hat{g}(\omega).$$
(5.5)

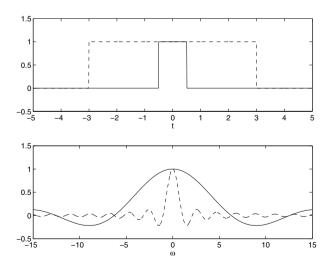
Thus, convolution in the time domain corresponds to the product of the respective spectra in the frequency domain! The convolution theorem is used in sections 2.2.1 and 2.2.3, respectively, for developing a reconstruction formula for the CWT and a fast algorithm for the computation of the CWT.

As a simple example illustrating Fourier transforms consider the function

$$\chi_T(t) = \begin{cases} 1 - \frac{T}{2} \le t \le \frac{T}{2} \\ 0 \text{ else} \end{cases}$$

It easily may be verified that its Fourier transform reads  $\hat{\chi}_T(\omega) = \frac{\sin(\frac{\omega T}{2})}{\frac{\omega}{2}}$ .

Figure 5.1 shows plots of  $\chi_T(t)$  and  $\frac{\hat{\chi}_T(\omega)}{T}$ , respectively, for T = 1 and T = 6.



**Fig. 5.1.** Above:  $\chi_1(t)$  (solid) and  $\chi_6(t)$  (dashed). Below: Corresponding Fourier transforms  $\hat{\chi}_1(\omega)$  (solid) and  $\frac{\hat{\chi}_6(\omega)}{6}$  (dashed).

Obviously the behavior of the function and its Fourier transform is in a certain sense complementary: To a function, which is "wide spread" in the time domain, belongs a Fourier transform, which is "narrow" in the frequency domain and vice versa. This rather qualitative statement shall now be sharpened.

Given a finite energy signal f(t), denote with E its total energy, i.e.,  $E := \int_{-\infty}^{+\infty} |f(t)|^2 dt$ . Then define the following "localization parameters":

$$t_f := \frac{1}{E} \int_{-\infty}^{+\infty} t |f(t)|^2 dt$$
 (5.6)

$$\Delta t_f := \sqrt{\frac{1}{E} \int_{-\infty}^{+\infty} (t - t_f)^2 |f(t)|^2} dt$$
(5.7)

$$\omega_f := \frac{1}{2\pi E} \int_{-\infty}^{+\infty} \omega |\hat{f}(\omega)|^2 \, d\omega \tag{5.8}$$

$$\Delta\omega_f := \sqrt{\frac{1}{2\pi E} \int_{-\infty}^{+\infty} (\omega - \omega_f)^2 |\hat{f}(\omega)|^2 \, d\omega}$$
(5.9)

 $t_f$  indicates where on the time domain f(t) is located,  $\Delta t_f$  provides information about how f(t) is localized around  $t_f$ : For large  $\Delta t_f$ -values f(t)will be wide spread around  $t_f$ , for small  $\Delta t_f$ -values f(t) will be narrowly concentrated around  $t_f$ . The same arguments apply to  $\hat{f}(\omega)$ ,  $\omega_f$  and  $\Delta \omega_f$ .

Both aspects may be unified by representing f in the t- $\omega$ -plane ("phase plane", cf. sect. 1.2.5). Here f(t) essentially "lives" in a rectangular region with area  $\Delta t_f \Delta \omega_f$  as indicated in Fig. 5.2.

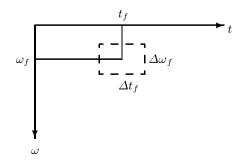


Fig. 5.2. Cell in phase plane, occupied by f.

The "Heisenberg uncertainty relation" then tells us that this area cannot be arbitrarily small:

$$\Delta t_f \Delta \omega_f \ge \frac{1}{2} \tag{5.10}$$

This relation is of fundamental importance. It's consequences reach from quantum mechanics to signal analysis. Indeed, Werner Heisenberg discovered that this relation led to the famous uncertainty principle, stating that position and momentum of an elementary particle never together can be measured with arbitrary precision. In signal analysis, relation 5.10 implies that no window function can be chosen, which is arbitrarily sharply concentrated both in time and frequency domain.

Fortunately, there exists a function whose localization parameters yield the lower bound in the inequality 5.10: Let g(t) be defined as in eq. 5.11

$$g(t) := \frac{1}{\sqrt[4]{\pi}} e^{-\frac{t^2}{2}}.$$
(5.11)

Then

$$\Delta t_g \Delta \omega_g = \frac{1}{2}$$

# 5.2 Discrete Fourier Transform (DFT)

Starting from a correspondence pair as given in eq. 5.3 we shall now consider the following task:

Choose a "sampling distance"  $T_S$  and denote the corresponding circular frequency with  $\omega_S := \frac{2\pi}{T_S}$ . Assume now that f is known only for  $t = \ldots, -2T_S, T_S, 0, T_S, 2T_S, \ldots$ , i.e., the data form a sequence  $\{f(kT_S)\}$ . For simplicity we also will use the symbol  $f_k$  instead of  $f(kT_S)$ . An example could be a sampled acoustic signal on your computer or your CD-player. Is it possible to gain some information about the spectrum  $\hat{f}(\omega)$  from that sequence? Provided certain restrictions are met, which will be given below, the answer will be "yes".

We shall assume that the finite energy signal f(t) is "band-limited", i.e.,  $\hat{f}(\omega) = 0$  outside some finite frequency interval. Let the "maximum frequency"  $\omega_{max} > 0$  be defined such that  $[-\omega_{max}, \omega_{max}]$  is the *smallest* possible interval outside of which  $\hat{f}(\omega)$  vanishes. A band-limited function will be said to satisfy the "Shannon condition" if

$$\omega_S > 2\omega_{max}.\tag{5.12}$$

Now consider a finitely sampled signal of length N, i.e., a sequence  $\{f_k\}_{k=0}^{N-1}$ . Then the "discrete Fourier Transform" ("DFT") reads

5.2 Discrete Fourier Transform (DFT) 129

$$F_k = \sum_{l=0}^{N-1} f_l e^{-\frac{j2\pi lk}{N}} \ (k = 0, 1, \dots, N-1)$$
(5.13)

The original data  $\{f_k\}_{k=0}^{N-1}$  may be recovered from the DFT-sequence  ${F_k}_{k=0}^{N-1}$  by the "inverse discrete Fourier Transform" ("IDFT"):

$$f_k = \frac{1}{N} \sum_{l=0}^{N-1} F_l e^{\frac{j2\pi lk}{N}} \ (k = 0, 1, \dots, N-1)$$
(5.14)

Before explaining how the DFT is related with  $\hat{f}(\omega)$ , we make two remarks:

- 1. There exists a fast algorithm for computing the DFT, namely the famous "Fast Fourier Transform" ("FFT"). 2. If the sampled signal  $\{f_l\}_{l=0}^{N-1}$  is real-valued, the following relation holds:

$$F_{N-k} = \overline{F_k} \ (k = 1, \dots, N-1)$$

As mentioned already, the overline denotes complex conjugation. From this relation follows that the DFT-sequence  $\{F_k\}_{k=0}^{N-1}$  is completely determined if roughly the first half of the sequence is known. To be precise:  $\{F_k\}_{k=0}^{N-1}$  is completely determined, provided the coefficients  $F_k$  are known for  $k = 0, \ldots, m_N$  with

$$m_N = \begin{cases} \frac{N}{2} & (N \text{ even})\\ \frac{N-1}{2} & (N \text{ odd}) \end{cases}$$
(5.15)

The following theorem establishes the announced relation between  $\{F_k\}_{k=0}^{N-1}$ and  $\hat{f}(\omega)$ :

**Theorem 5.1.** Let f(t) be a band-limited finite energy signal with f(t) = 0for t < 0 and  $t > (N-1)T_S$ . Let  $\{f(kT_S)\}_{k=0}^{N-1}$  be a finite sampling of the signal such that

$$\omega_S = \frac{2\pi}{T_S}$$

satisfies the Shannon condition 5.12. Define the frequency sampling

$$\omega_k = \frac{k}{N} \omega_S \ (k = 0, \dots, m_N).$$

Now compute the DFT-sequence corresponding to eq. 5.13 with  $f_l$  =  $f(lT_S)$ . Then we have

$$\hat{f}(\omega_k) \approx T_S F_k \ (k = 0, \dots, m_N).$$
(5.16)

We close this section with some remarks

- 1. Since the DFT may be computed very quickly with the FFT-algorithm, relation 5.16 provides a fast algorithm for computing the *continuous* spectrum  $\hat{f}(\omega)$  at the  $\omega_k$ -values given above.
- 2. If f(t) is nonzero only on the interval  $[0, (N-1)T_S]$ , band limitation strictly cannot be fulfilled [34]. This means that the Shannon condition must be understood such that  $\hat{f}(\omega)$  is "sufficiently small" outside the interval  $[-\omega_{max}, \omega_{max}]$ . If this is not the case, one has to be aware that, in particular for high

If this is not the case, one has to be aware that, in particular for high  $\omega_k$ -values, the estimation of  $\hat{f}(\omega_k)$  via eq. 5.16 leads to distortions by "aliasing". This explains, why in eq. 5.16 the  $\approx$ -symbol was used instead of strict equality.

3. Readers interested in more details of the topics covered in this section are referred to textbooks on digital signal processing and its mathematical foundations, respectively. Examples are included in the bibliography ([33], [34]).

# 5.3 Digital Filters

# Filtering

In this section we shall introduce basic notions of digital filtering. This section is intended only for providing first principles and, particularly, for fixing the notation. For more details on the topic the reader is referred to, e.g., [33]. We shall consider only discretely sampled signals using the same conventions as in sect. 5.2.

A digital filter generates from a discretely sampled signal  $f = \{f_k\}$  a new signal  $\mathcal{H}f = \{(\mathcal{H}f)_k\}$ . Usually this process is visualized as shown below:

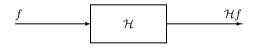
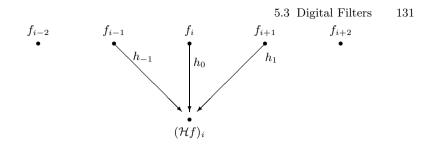


Fig. 5.3. A digital filter.

The filter is characterized by a finite set of "filter coefficients"  $\{h_i\}$  and the new discrete signal  $\mathcal{H}f$  is computed as follows:

$$(\mathcal{H}f)_k = \sum_k h_{i-k} f_i.$$
(5.17)

It is useful to visualize eq. 5.17 as in Fig. 5.4. Here the filter is specified by three filter coefficients  $\{h_{-1}, h_0, h_1\}$ .



**Fig. 5.4.** Visualization of eq. 5.17 for a filter with coefficients  $\{h_{-1}, h_0, h_1\}$ .

The top row symbolizes the input sequence  $\{f_k\}$ , the bottom row the output sequence  $\{(\mathcal{H}f)_k\}$ . The computation of the *i*-th element of the output sequence is illustrated as follows: The elements of the input sequence, which point toward the *i*-th element of the output sequence with an arrow, are multiplied with the corresponding filter coefficients and added at the node, where the arrows meet. Thus in our case  $(\mathcal{H}f)_i = f_{i-1}h_{-1} + f_ih_0 + f_{i+1}h_1$ .

For completeness we note that a filter of this kind is called FIR-filter [33], provided  $h_1 = h_2 = h_3 = \ldots = 0$ . Since we allow also for nonzero values of these coefficients, we shall consider the general case of "non-causal filters"<sup>1</sup>: If, e.g.,  $h_1 \neq 0$  the output signal value corresponding to time  $iT_S$  may not be computed before time  $(i + 1)T_S$  since we need also the "future value"  $f_{i+1}$ . Moreover, we also note that the indexing of the filter coefficients is slightly different from engineering standard [33], but this does not affect the main aspects of the transfer functions discussed below.

The transfer properties of a filter are determined by the corresponding "frequency response":

$$\mathcal{H}(\Omega) = \sum_{k} h_k e^{-jk\Omega} \ (0 \le \Omega < \pi)$$
(5.18)

The spectrum of  $\mathcal{H}f$ , as measured, e.g., by the DFT (cf. sect. 5.2), is obtained from the spectrum of f by a multiplication with the frequency response. Details shall be omitted here, again we refer the interested reader to [33]. The frequency response is a complex quantity. We may decompose it with respect to absolute value and phase, thus obtaining "amplitude response"  $A(\Omega)$  and "phase response"  $\Phi(\Omega)$ :

$$\mathcal{H}(\Omega) = A(\Omega) e^{j\Phi(\Omega)}.$$
(5.19)

Again, without going into details we note here that the amplitude response  $A(\Omega)$  is responsible for transmitted or suppressed frequencies. Moreover, it is desirable to have a phase response  $\Phi(\Omega)$ , which is *linear* with respect to  $\Omega$ , since in this case the filtering process leads to a reduced signal distortion.

<sup>&</sup>lt;sup>1</sup> This feature is of relevance for online-applications, see sect. 3.6.2.

Without proof we remark that "symmetric filters" lead to linear phase response. Here symmetry is defined as follows: If the number of coefficients is odd, then the coefficient must be symmetric with respect to the center coefficient. If the number of coefficients is even, then the coefficients must be symmetric with respect to the "center line" separating the first half from the second half. For example, filters with coefficients  $\{h_{-1}, h_0, h_1\} = \{-1, 2, -1\}$ or  $\{h_0, h_1, h_2, h_3\} = \{1, 2, 2, 1\}$ , respectively, both are symmetric.

As stated above, the main purpose of this section is to introduce the basic notions and to fix the notation. Examples related to amplitude and frequency response, repectively, are given in sect. 3.3.

## Filtering with subsampling

"Subsampling" a sequence means that only every second sequence element is kept from the original sequence. This process usually is represented by a box containing a  $\downarrow$  2-symbol (cf. Fig. 5.5 below). Obviously at this step the length of the original sequence is divided by two.

The graphical representation of a filtering process and a subsequent downsampling procedure is shown in Fig. 5.5.

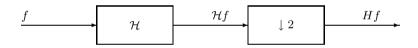


Fig. 5.5. A digital filter followed by downsampling.

Thus, the new sequence obtained after filtering and downsampling is denoted with  $Hf = \{(Hf)_k\}$ . Mathematically it is computed as follows:

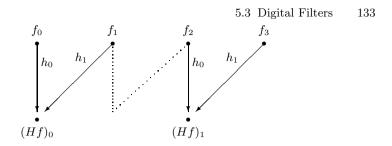
$$(Hf)_k = \sum_k h_{i-2k} f_i.$$
 (5.20)

For a sequence  $\{f_k\}_{k=0}^3$  of length 4 and a filter with coefficients  $\{h_0, h_1\}$  the process is visualized in Fig. 5.6.

The procedure discussed so far leads to sequences with half as many entries as the original sequence, where each new sequence element is computed as a linear combination of the original sequence elements.

## Filtering with upsampling, dual filters

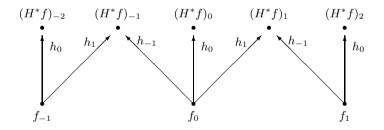
We will need also the complementary procedure: From a given sequence a new sequence will be generated with twice as may elements as the original,



**Fig. 5.6.** Filtering with downsampling: Visualization of eq. 5.20 for a filter with coefficients  $\{h_0, h_1\}$ . The element of  $\mathcal{H}f$  omitted by the downsampling procedure is indicated with dotted lines.

the new sequence elements also will be linear combinations of the old sequence elements.

We again start with a set of filter coefficients  $\{h_i\}$ . Remember that we may associate a filtering and subsampling procedure, denoted with H, as described above. The "dual filter"  $H^*$ , applied to a sequence, will produce a sequence, whose length will be doubled, as compared with the length of the original sequence. The simplest way to introduce  $H^*$  is a proper visualization. For the case of three filter coefficients  $\{h_{-1}, h_0, h_1\}$  it is given in Fig. 5.7.



**Fig. 5.7.** Visualization of a dual filter  $H^*$  with coefficients  $\{h_{-1}, h_0, h_1\}$ .

The interpretation is very simple: Imagine the diagram of Fig. 5.6 written down for three filter coefficients  $\{h_{-1}, h_0, h_1\}$  and invert it. This means that now the top row represents the new sequence, the bottom row the original. Again, if arrows meet at an element of the new sequence, the respective old sequence elements must be multiplied with the corresponding filter coefficients and added. Thus, e.g.,  $(H^*f)_{-1} = f_{-1}h_1 + f_0h_{-1}$ . It is clear from Fig. 5.7 that the signal length is doubled at this process.

The summation formula associated with dual filters looks as follows:

134 5 Appendix

$$(H^*f)_k = \sum_k h_{k-2i} f_i.$$
 (5.21)

# 5.4 Solutions to Selected Problems

## 5.4.1 Problems from Sect. 2.4

1. a) Given  $y(t) \circ - \bullet \hat{y}(\omega)$  we have

$$\hat{y}(\omega) = \int_{-\infty}^{+\infty} y(t) e^{-j\omega t} dt.$$

Differentiating this equation on both sides with respect to  $\omega$  we obtain

$$\frac{d\hat{y}}{d\omega}(\omega) = \frac{d}{d\omega} \left( \int_{-\infty}^{+\infty} y(t)e^{-j\omega t} dt \right)$$
$$= \int_{-\infty}^{+\infty} y(t) \frac{de^{-j\omega t}}{d\omega} dt$$
$$= \int_{-\infty}^{+\infty} y(t)(-jt)e^{-j\omega t} dt$$

Here on the right-hand side we interchanged differentiation and integration, which may be justified. Iterating this process we obtain

$$\frac{d^m \hat{y}}{d\omega^m}(\omega) = \int_{-\infty}^{+\infty} y(t)(-jt)^m e^{-j\omega t} dt.$$

Recalling definition 5.1 of the Fourier transform, we conclude that

$$y(t)(-jt)^m \circ - \bullet \frac{d^m \hat{y}}{d\omega^m}(\omega).$$

This may be rewritten as

$$y(t)t^m \circ - \bullet j^m \frac{d^m \hat{y}}{d\omega^m}(\omega),$$

which is the desired result.

b) Proceed analogously as above starting with

$$y(t) = \frac{1}{2\pi} \int_{-\infty}^{+\infty} \hat{y}(\omega) e^{+j\omega t} \, d\omega.$$

2. In order to compute all localization parameters  $\hat{w}(\omega)$  is needed. Therefore we insert

$$w(t) = \begin{cases} 1 - \frac{T}{2} \le t \le \frac{T}{2} \\ 0 \text{ else} \end{cases}$$

in eq. 5.1 and obtain

$$\hat{w}(\omega) = \int_{-\infty}^{+\infty} w(t)e^{-j\omega t} dt = \int_{-\frac{T}{2}}^{+\frac{T}{2}} e^{-j\omega t} dt$$
$$= \int_{-\frac{T}{2}}^{+\frac{T}{2}} (\cos(\omega t) - j\sin(\omega t)) dt = 2 \int_{0}^{+\frac{T}{2}} \cos(\omega t) dt$$
$$= \frac{2\sin\left(\frac{\omega T}{2}\right)}{\omega}$$

Here we applied the fact that  $\cos(\omega t)$  is an even and  $\sin(\omega t)$  an odd function. Moreover,  $E = \int_{-\infty}^{+\infty} |w(t)|^2 dt = T$ . Then from eq. 5.6 follows that

$$t_w = \frac{1}{E} \int_{-\infty}^{+\infty} t |w(t)|^2 dt = \frac{1}{T} \int_{-\frac{T}{2}}^{+\frac{T}{2}} t dt$$
$$= 0$$

Similarly, eq. 5.8 implies that

$$\omega_w = \frac{1}{2\pi E} \int_{-\infty}^{+\infty} \omega |\hat{w}(\omega)|^2 \, d\omega = \frac{1}{2\pi T} \int_{-\infty}^{+\infty} \frac{4\sin^2\left(\frac{\omega T}{2}\right)}{\omega} \, d\omega$$
$$= 0$$

#### 136 5 Appendix

Here we made use of the fact that  $\frac{4\sin^2\left(\frac{\omega T}{2}\right)}{\omega}$  is an odd function. From eq. 5.7 we obtain

$$\Delta t_w = \sqrt{\frac{1}{E} \int_{-\infty}^{+\infty} (t - t_w)^2 |w(t)|^2 dt}$$
$$= \sqrt{\frac{1}{T} \int_{-\frac{T}{2}}^{+\frac{T}{2}} t^2 dt} = \sqrt{\frac{2}{T} \int_{0}^{+\frac{T}{2}} t^2 dt}$$
$$= \frac{T}{2\sqrt{3}}$$

Finally, eq. 5.9 implies

$$\Delta\omega_w = \sqrt{\frac{1}{2\pi E} \int_{-\infty}^{+\infty} (\omega - \omega_w)^2 |\hat{w}(\omega)|^2 \, d\omega}$$
$$= \sqrt{\frac{1}{2\pi T} \int_{-\infty}^{+\infty} 4\sin^2\left(\frac{\omega T}{2}\right) \, d\omega}$$
$$= \infty$$

3. We start with  $\psi_H$  as defined in eq. 2.32. For computing the CWT 2.17 with the Haar-wavelet, we first evaluate the expression  $\overline{\psi_H}\left(\frac{u-t}{a}\right)$ , considered as a function of u. Since the Haar-wavelet is real-valued and we assume a > 0 this leads to

$$\overline{\psi_H}\left(\frac{u-t}{a}\right) = \psi_H\left(\frac{u-t}{a}\right)$$
$$= \begin{cases} 1 & 0 \le \frac{u-t}{a} < \frac{1}{2} \\ -1 & \frac{1}{2} \le \frac{u-t}{a} < 1 \\ 0 & \text{else} \end{cases}$$
$$= \begin{cases} 1 & t \le u < t + \frac{a}{2} \\ -1 & t + \frac{a}{2} \le u < t + a \\ 0 & \text{else} \end{cases}$$

Inserting this result in formula 2.17 we obtain

$$\int_{-\infty}^{\infty} \overline{\psi_H}\left(\frac{u-t}{a}\right) f(u) \, du = \int_{t}^{t+\frac{a}{2}} f(u) \, du - \int_{t+\frac{a}{2}}^{t+a} f(u) \, du$$
$$= F\left(t+\frac{a}{2}\right) - F(t) - F(t+a) + F\left(t+\frac{a}{2}\right)$$

Thus

$$L_{\psi_H}f(a,t) = \frac{1}{\sqrt{c_{\psi}a}} \left(2F\left(t+\frac{a}{2}\right) - F(t) - F(t+a)\right).$$

4. For any wavelet  $\psi$  the correspondence pair  $\psi(t)t^m \circ - \bullet j^m \frac{d^m \hat{\psi}}{d\omega^m}(\omega)$  (cf. problem 1) will be true.

When, moreover,  $\psi$  is such that  $\psi(t) = \frac{d^M \Phi}{dt^M}(t)$  for some suitable function  $\Phi(t)$ , in addition

$$\psi(t) = \frac{d^M \Phi}{dt^M}(t) \circ - \bullet (j\omega)^M \hat{\Phi}(\omega) = \hat{\psi}(\omega)$$

(cf. problem 1). Combining both relations we conclude that

$$\psi(t)t^m \circ - \bullet j^m \frac{d^m}{d\omega^m} \left( (j\omega)^M \hat{\varPhi}(\omega) \right) \ (m = 0, \dots, M - 1).$$

A moment integral (cf. eq. 2.31) reads  $\int_{-\infty}^{+\infty} \psi(t) t^m dt$ . Rewriting it as

$$\int_{-\infty}^{+\infty} \psi(t)t^m \, dt = \int_{-\infty}^{+\infty} \psi(t)t^m e^{-j0t} \, dt$$

and recalling definition 5.1 of the Fourier transform we conclude that it is identical to the Fourier transform of  $\psi(t)t^m$ , evaluated for  $\omega = 0$ . Therefore

$$\int_{-\infty}^{+\infty} \psi(t) t^m dt = j^m \frac{d^m}{d\omega^m} \left( (j\omega)^M \hat{\Phi}(\omega) \right) \Big|_{\omega=0} \quad (m=0,\ldots,M-1).$$

If  $\hat{\varPhi}(0)$  and derivatives  $\hat{\varPhi}'(0), \hat{\varPhi}''(0), \dots, \hat{\varPhi}^{M-1}(0)$  exist and are finite,

$$\frac{d^m}{d\omega^m} \left( (j\omega)^M \hat{\varPhi}(\omega) \right) \Big|_{\omega=0} = 0 \ (m = 0, \dots, M-1).$$
  
Therefore  $\int_{-\infty}^{+\infty} \psi(t) t^m \, dt = 0 \ (m = 0, \dots, M-1).$ 

#### 138 5 Appendix

5. Let f(t) satisfy the assumption made on page 27 and following pages: On  $I_{(a,t)}$  it may be written as a polynomial of degree k with k < M:

$$f(u) = \sum_{m=0}^{k} a_m u^m \ (u \in I_{(a,t)}).$$

Then

$$\int_{-\infty}^{\infty} \overline{\psi}\left(\frac{u-t}{a}\right) f(u) \, du = \sum_{m=0}^{k} a_m \int_{-\infty}^{\infty} u^m \overline{\psi}\left(\frac{u-t}{a}\right) \, du$$
$$= \sum_{m=0}^{k} a_m \int_{-\infty}^{\infty} (t+au)^m \overline{\psi}(u) \, du \frac{1}{a}$$

Proceeding from the first to the second line we substituted  $u' = \frac{u-t}{a}$  as a new variable of integration. The vanishing of the integrals follows from expanding  $(t + au)^m$  with the binomial expansion formula and from the vanishing moments condition.

So finally we obtain  $L_{\psi}f(a,t) = 0$ .

### 5.4.2 Problems from Sect. 3.7

1. We consider  $\frac{1}{\sqrt{2}}\psi_H\left(\frac{u-2k}{2}\right)$ . From eq. 2.32 follows that for  $k=0,\ldots,3$ 

$$\psi_H\left(\frac{u-2k}{2}\right) = \begin{cases} 1 & 0 \le \frac{u-2k}{2} < \frac{1}{2} \\ -1 & \frac{1}{2} \le \frac{u-2k}{2} < 1 \\ 0 & \text{else} \end{cases}$$
$$= \begin{cases} 1 & 2k \le u < 2k+1 \\ -1 & 2k+1 \le u < 2k+2 \\ 0 & \text{else} \end{cases}$$

Thus, this function will be nonzero only on the interval [2k, 2k + 2). On the other hand, on [2k, 2k + 2) the function  $f^1(t)$  will be constant with value  $\frac{f_{2k}+f_{2k+1}}{2}$ . Therefore

$$L_{\psi_H} f^1(2, 2k) = \frac{1}{\sqrt{c_{\psi_H} 2}} \int_{-\infty}^{\infty} \psi_H \left(\frac{u - 2k}{2}\right) f^1(u) \, du$$
$$= \frac{1}{\sqrt{c_{\psi_H} 2}} \frac{f_{2k} + f_{2k+1}}{2} \int_{2k}^{2k+2} \psi_H \left(\frac{u - 2k}{2}\right) \, du = 0$$

On [2k, 2k+2) the function  $d^1(t)$ , however, will take the following values:

$$d^{1}(t) = \begin{cases} \frac{f_{2k} - f_{2k+1}}{2} & 2k \le u < 2k+1\\ -\frac{f_{2k} - f_{2k+1}}{2} & 2k+1 \le u < 2k+2\\ 0 & \text{else} \end{cases}$$
$$= \begin{cases} d_{k}^{1} & 2k \le u < 2k+1\\ -d_{k}^{1} & 2k+1 \le u < 2k+2\\ 0 & \text{else} \end{cases}$$

Therefore

$$\begin{split} L_{\psi_H} d^1(2, 2k) &= \frac{1}{\sqrt{c_{\psi_H} 2}} \int_{-\infty}^{\infty} \psi_H \left(\frac{u - 2k}{2}\right) d^1(u) \, du \\ &= \frac{1}{\sqrt{c_{\psi_H} 2}} \left( d_k^1 \underbrace{\int_{2k}^{2k+1} \psi_H \left(\frac{u - 2k}{2}\right) \, du}_{1} - d_k^1 \underbrace{\int_{2k+1}^{2k+2} \psi_H \left(\frac{u - 2k}{2}\right) \, du}_{-1} \right) \\ &= \frac{1}{\sqrt{c_{\psi_H} 2}} 2 d_k^1 \\ &= \frac{2}{\sqrt{c_{\psi_H}}} d_k^1 \end{split}$$

2. Inserting  $\left\{h_0 = \frac{\sqrt{2}}{2}, h_1 = \frac{\sqrt{2}}{2}\right\}$  in eq. 5.18 we obtain

$$\mathcal{H}(\Omega) = \sum_{k} h_k e^{-jk\Omega} = \frac{\sqrt{2}}{2} (1 + e^{-j\Omega}) \ (0 \le \Omega < \pi).$$

Observe that

$$(1 + e^{-j\Omega}) = (e^{j\frac{\Omega}{2}} + e^{-j\frac{\Omega}{2}})e^{-j\frac{\Omega}{2}} = 2\cos\left(\frac{\Omega}{2}\right)e^{-j\frac{\Omega}{2}}.$$

So,  $\mathcal{H}(\Omega)$  may be written as follows:

$$\mathcal{H}(\Omega) = \sqrt{2}\cos\left(\frac{\Omega}{2}\right)e^{-j\frac{\Omega}{2}}.$$

Comparing with eq. 5.19 we may directly read off amplitude and phase response:

140 5 Appendix

$$A(\Omega) = \sqrt{2}\cos\left(\frac{\Omega}{2}\right), \qquad \Phi(\Omega) = -\frac{\Omega}{2}.$$

Note that amplitude and phase response associated with the Haar-filter have been depicted in Figs. 3.9 and 3.18, respectively!

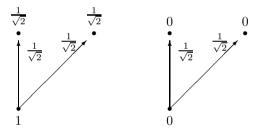
The transfer functions of the g-filter are computed analogously.

3. As an example we consider the computation of  $\phi_{wavelet}^0$ . We start with the natural basis sequence  $\tilde{\phi}_{natural}^0 = \{1, 0, 0, 0\}$ . In analogy to eq. 3.49 where the decomposition of such a natural basis sequence was sketched for J = 2, we decompose this sequence into two subsequences according to

$$\tilde{\phi}_{natural}^{0} = \{\underbrace{\{1,0\}}_{f_{\phi}^{1}}, \underbrace{\{0,0\}}_{d_{\phi}^{1}}\}$$

We now apply the reconstruction part of eq. 3.30 for J = 1 to these sequences. The coefficients belonging to the dual filters  $H^*$  and  $G^*$ , respectively, are given by 3.19 and 3.20.

As an example, the computation of  $H^* f_{\phi}^1$  may be visualized as follows (see also Fig. 3.5):



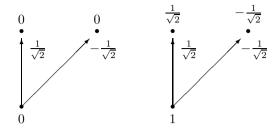
Similarly we obtain  $G^*d^1_{\phi} = \{0, 0, 0, 0\}$ . Adding both sequences leads to

$$\phi_{wavelet}^0 = \left\{ \frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}, 0, 0 \right\}.$$

As a final example we illustrate the computation of  $\phi^4_{wavelet}.$  Here we start with

$$\tilde{\phi}^4_{natural} = \{\underbrace{\{0,0\}}_{f^1_{\phi}}, \underbrace{\{0,1\}}_{d^1_{\phi}}\}$$

and obtain  $G^*d^1_\phi$  via



Here  $H^* f_{\phi}^1 = \{0, 0, 0, 0\}$ . Adding both sequences results in

$$\phi_{wavelet}^4 = \left\{0, 0, \frac{1}{\sqrt{2}}, -\frac{1}{\sqrt{2}}\right\}.$$

Correspondingly,  $\phi_{wavelet}^2$  and  $\phi_{wavelet}^3$  are computed. 4. For solutions to problems 4a)-c) we refer the reader to the folders mentioned on page 93.

As an example we include the solution to problem 4a):

```
function []=transfers(a)
close all;
if (size(a,1)>size(a,2)),
    a=a';
end
%Extraction of h-coefficients
uh=a(size(a,2)-1); oh=a(size(a,2));
h=a(1:(size(a,2)-2));
%Computation of g-coefficients
ug=1-oh; og=1-uh; index=length(a)-2;
sign=(-1)^(ug); g=sign*h(index);
for i=(ug+1):og,
    index=index-1;
    sign=-sign;
    g=[g,sign*h(index)];
end
%Plots
do=input('Step size: ');
om=0:do:pi;
H=h*exp(-j*(uh:oh)'*om);
G=g*exp(-j*(ug:og)'*om);
```

figure(1); plot(om,abs(H),'k-',om,abs(G),'k--'); title(...); figure(2); plot(om,angle(H),'k-',om,angle(G),'k--'); title(...);

5. Applying reconstruction formula 5.2 to

$$\hat{\phi}(\omega) = \begin{cases} 1 \ |\omega| \le \pi \\ 0 \ \text{else} \end{cases}$$

we obtain

$$\frac{1}{2\pi} \int_{-\infty}^{\infty} \hat{\phi}(\omega) e^{j\omega t} d\omega = \frac{1}{2\pi} \int_{-\pi}^{\pi} e^{j\omega t} d\omega$$
$$= \frac{1}{\pi} \int_{0}^{\pi} \cos(\omega t) d\omega = \frac{\sin(\pi t)}{\pi t}$$
$$= \phi(t)$$

with  $\phi(t)$  as defined in eq. 3.58. Thus, obviously the maximum frequency  $\omega_{max}$  belonging to  $\phi(t)$  is given by  $\omega_{max} = \pi$ .

Choosing the sampling distance  $T_S = \frac{1}{2}$  we obtain for the corresponding sampling circular frequency  $\omega_S = \frac{2\pi}{T_S} = 4\pi$ . Thus, obviously

$$\omega_S > 2\omega_{max}$$

Comparing with eq. 5.12 we conclude that the Shannon condition is fulfilled when sampling  $\phi(t)$  with  $T_S = \frac{1}{2}$ . The corresponding sample values read  $\{\phi(\frac{k}{2})\}$  and the reconstruction equation 3.56 implies that

$$\phi(t) = \sum_{k} \phi\left(\frac{k}{2}\right) \phi_{Sh}\left(t - \frac{k}{2}\right) = \sum_{k} \phi\left(\frac{k}{2}\right) \frac{\sin(\pi(2t - k))}{\pi(2t - k)}$$
$$= \sqrt{2} \sum_{k} \frac{\phi\left(\frac{k}{2}\right)}{\sqrt{2}} \phi(2t - k)$$

## 5.4.3 Problems from Sect. 4.3

1. Given the string ABCCDDEE we compute relative frequencies, entropy, tree and code as follows:

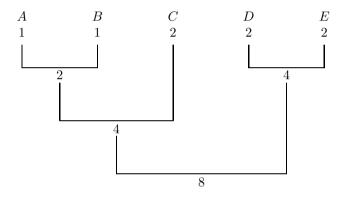
Relative frequencies:

$$p_A = p_B = \frac{1}{8}, \ p_C = p_D = p_E = \frac{1}{4}$$

Entropy:

$$H(p_A, p_B, p_C, p_D, p_E) = -\frac{2}{8}\log_2\left(\frac{1}{8}\right) - \frac{3}{4}\log_2\left(\frac{1}{4}\right)$$
  
= 2.25

Tree:



Code:

Table 5.1. Huffman code for ABCCDDEE.

Symbol codeword		
A	000	
В	001	
С	01	
D	10	
С	11	

Average codeword length:

$$\bar{l}_{Huffman} = \frac{3+3+2+2+2+2+2+2+2}{8}$$
$$= \frac{18}{8}$$
$$= 2.25$$

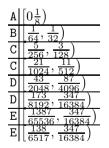
So in this example the Huffman code reaches the lower bound from eq. 4.16.

2. We show here a solution which is slightly more general than the solution to the original problem: The produced digits are stored in a string which is returned by the function.

```
function out=DecBinConv(q,n)
%Preparations
q=2*q;
out=num2str(floor(q));
q=q-floor(q);
k=1;
%Conversion loop
while((q~=0)&(k<n)),
    q=2*q;
    out=strcat(out,num2str(floor(q)));
    q=q-floor(q);
    k=k+1;
end</pre>
```

3. To generate the string codeword we start with the following subdivision of the unit interval [0, 1):

Selections and subsequent expansions lead to the following list of selected intervals:



So the final interval reads

$$[u, o) = \left[\frac{138}{6517}, \frac{347}{16384}\right).$$

Computing the respective binary expansions we obtain

So, finally, the string codeword reads 000001010101011111. From this the average codeword length may be computed:

$$\overline{l}_{AC} = \frac{19}{8}$$
$$= 2.375$$

4. Again we show here a solution, which is slightly more general than the solution to the original problem: The binary expansions of the interval boundaries are returned by the function.

```
function[lowerbin,upperbin]=ArithBound(str,symb,freq)
```

```
%Preliminaries
format rat:
format compact;
freq=[0,cumsum(freq)/sum(freq)]; disp('Start: '), disp(freq),
n=length(str);
lower=0; upper=1;
%String-loop
k=1;
while(k<=n),
    %Expansion
    expanded=lower+freq*(upper-lower);
    m=1;
    while(str(k)~=symb(m)),
        m=m+1;
    end
    %Selection
    lower=expanded(m); upper=expanded(m+1);
    disp(symb(m)), disp([lower upper]),
    k=k+1;
end
lowerbin=DecBinConv(lower,100);
upperbin=DecBinConv(upper,100);
```

## 5.5 Notations and Symbols

In this section frequently used symbols and notations are collected. The first column describes the meaning, the corresponding symbol used throughout this book is given in the second column. The third column displays alternative symbols used particularly in the engineering community.

Explanation	Symbol in this tex	At Alternative symbol
Set of real numbers	R	
Set of complex numbers	$\mathbf{C}$	
Continuous-time signal	f(t)	
Wavelet	$\psi(t)$	
Imaginary unit	j	i
Complex conjugate of a		
complex number $\alpha$	$\overline{\alpha}$	$lpha^*$
Fourier transform of $f(t)$	$\hat{f}(\omega)$	$\mathcal{F}f(\omega)$
Fourier transform pair	$f(t) \circ - \bullet \hat{f}(\omega)$	
Sampling distance	$T_S$	
Sampling rate	$\nu_S = \frac{1}{T_S}$	
Sampling rate Sampling circular frequence	$\psi \omega_S = \frac{2\pi}{T_z}$	
Sequence of numbers	$f = \{f_k\}$	$f = \{f[k]\}$

Table 5.2. Symbols and Notations

# References

- 1. W. Bäni (2002) Wavelets, eine Einführung für Ingenieure. Oldenbourg, München
- C.G. Broyden (1970) The Convergence of a Class of Double-Rank Minimization Algorithms. Journal Inst. Math. Applic. 6: 76–90
- 3. C.S. Burrus, R.A. Gopinath, H. Guo (1998) Introduction to Wavelets and Wavelet Transforms, A Primer. Prentice-Hall, Upper Saddle River (NJ)
- 4. C.K. Chui (1997) Wavelets: A Mathematical Tool for Signal Analysis. SIAM, Philadelphia
- A. Cohen (1992) Ondelettes, analyses multirésolution et traitement numérique du signal. Ph. D. Thesis, University of Paris IX, Dauphine
- J.M. Combes et al (eds) (1989) Wavelets, Time-Frequency Methods and Phase Space. Springer, Berlin Heidelberg New York
- 7. I. Daubechies (1992) Ten Lectures on Wavelets. SIAM, Philadelphia
- I. Daubechies (1989) Orthonormal Bases of Wavelets with Finite Support -Connection with Discrete Filters. In: J.M. Combes et al (eds) Wavelets, Time-Frequency Methods and Phase Space. Springer, Berlin Heidelberg New York
- D.L. Donoho (1995) De-Noising by Soft-Thresholding. IEEE Trans. on Inf. Theory 41: 613–627
- D. Gabor (1946) Theory of Communication. J. Inst. Electr. Engrg. 93(III): 429–457
- P. Goupillaud, J. Morlet, A. Grossmann, (1984/1985) Cycle-Octave and Related Transforms in Seismic Signal Analysis. Geoexploration 23: 85–102
- 12. F. Grupp, F. Grupp (2002) MATLAB 6 für Ingenieure. Oldenbourg, München
- D. Huffman (1952) A Method for the Construction of Minimum Redundancy Codes. Proceedings of the IRE 40: 1098–1101
- 14. B. Jaehne (1995) *Digital Image Processing*. Springer, Berlin Heidelberg New York
- N.S. Jayant, P. Noll (1984) Digital Coding of Waveforms. Prentice-Hall, Upper Saddle River (NJ)
- 16. A. Jensen, A. laCour-Harbo (2001) Ripples in Mathematics The Discrete Wavelet Transform. Springer, Berlin Heidelberg New York
- 17. G. Kaiser (1993) A Friendly Guide to Wavelets. Birkhäuser, Boston Basel Berlin
- M. Kobayashi (1996) Listening for Defects: Wavelet-Based Acoustical Signal Processing in Japan. SIAM News 29, No. 2

- J.C. Lagarias, J. A. Reeds, M. H. Wright, P. E. Wright (1998) Convergence Properties of the Nelder-Mead Simplex Method in Low Dimensions. SIAM Journal of Optimization 9:112–147
- A.K. Louis, P. Maaß, A. Rieder (1997) Wavelets, Theory and Applications. Wiley, New York
- P. Maaß, H.-G. Stark (1994) Wavelets and Digital Image Processing. Surveys on Mathematics for Industry 4: 195–235
- S. Mallat (1998) A Wavelet Tour of Signal Processing. Academic Press, New York (NY)
- 23. Y. Meyer (1993) Wavelets, Algorithms and Applications. SIAM, Philadelphia
- 24. M. Misiti Y. Misiti, G. Oppenheim, J.-M. Poggi (2000) Wavelet Toolbox for Use with MATLAB - User's Guide. The MATH WORKS Inc.
- A.H.H. Ngu, Q.Z. Sheng, D.Q. Huynh, R. Lei (2000) Combining Multi-visual Features for Efficient Indexing in a Large Image Database. The VLDB Journal 9:1–15
- 26. Y. Nievergelt (2001) Wavelets Made Easy. Birkhäuser, Boston Basel Berlin
- S. Santini, R. Jain (1997) Similarity is a geometer. Multimedia Tools Appl. 5:277-306
- G. Salton, C. Buckley (1990) Improving Retrieval Performance by Relevance Feedback. Journal of the ASIS 41:288–297
- C.E. Shannon (1948) A Mathematical Theory of Communication. Bell System Technical Journal 27
- K. Sigmon (1994) MATLAB Primer. Chapman & Hall/CRC, Boca Raton London New York Washington D.C.
- H.-G. Stark (1996) On Image Retrieval with Wavelets. International Journal of Imaging Systems and Technology 7:200–210
- 32. H.-G. Stark (1997) Wavelets und Bildarchive. at Automatisierungstechnik 45:577–584
- 33. S.D. Stearns, R.A. David (1996) Signal Processing Algorithms in MATLAB. Prentice Hall, Upper Saddle River (NJ)
- W. Strampp, E.V. Vorozhtsov (2004) Mathematische Methoden der Signalverarbeitung. Oldenbourg, München
- 35. R.F. Streater, A.S. Wightman (1980) *PCT, Spin and Statistics and All That.* Benjamin/Cummings, Reading (MA), London, Amsterdam
- 36. T. Strutz (2000) Bilddatenkompression, Grundlagen, Codierung, MPEG, JPEG. Vieweg, Braunschweig
- J.T. Tou, R.C. Gonzalez, (1974) Pattern Recognition Principles. Addison-Wesley Publishing Company, Reading (MA)
- J.S. Walker (1999) A Primer on Wavelets and Their Scientific Applications. Chapman & Hall/CRC, Boca Raton London New York Washington D.C.
- G.K. Wallace (1991) The JPEG Still Picture Compression Standard, Comm. of the ACM 34: 31–44
- 40. D. Ziener (2002) FPGA-Implementierung eines arithmetischen Codierers. Diploma Thesis, Aschaffenburg University of Applied Sciences

# Index

Admissibility condition, 21 Approximation signal, 61 Arithmetic coding, 104 Bits per pixel, 111 bpp, 111 Cascade algorithm, 56 Cochlea, 40 Coiflets, 85 Compression bit per pixel (bpp), 111 compression rate, 96 naïve scheme, 84 peaked histograms, 84 transform compression scheme, 95 CWT, 4, 21 fast algorithm, 23 reconstruction, 24 transform, 21 visualization, 7 Daubechies-wavelets, 53 Denoising, 88 Detail signal, 61 diagonal details, 66 horizontal details, 66 vertical details, 66 DFT, 128 Digital filtering, 130 Discrete Fourier transform, 128 Distortion measures norm distance, 96 PSNR, 96

DWT, 43 one-dimensional signals, 61 real-time properties, 88 two-dimensional signals, 68 Energy compaction, 64, 81 Entropy, 100 Entropy coding, 99 Fourier transform continuous transform, 3, 125 discrete Fourier transform, 128 uncertainty relation, 128 Gabor transform, 19 Huffman coding, 102 Iterative optimization Nelder-Mead, 114 Quasi-Newton, 114 **JPEG**, 98 JPEG 2000, 75, 98 progressive mode, 123 Local analysis, 2 Localization parameters, 127 MATLAB, 11 Optimization Toolbox, 114 STFT functions myistft, 20 mystft, 19

wavelet functions CompAudioToolbox, 87 CompImageToolbox, 87 bases1d. 81 bases2d, 81 mycwt, 29 myicwt, 31 optshell, 113 Wavelet Toolbox, 11 cwt. 31 wavedec, 70 wavedec2. 71 waverec, 71 waverec2.73 wfilters, 73 Multiscale analysis, 60 Norm distance, 96 Pattern recognition, 38 Pattern vector, 38, 118 Peak-signal-to-noise-ratio, 96 Phase plane, 8, 127 localization parameters, 127 Phase space representation, 8 PSNR, 96 Quantization non-uniform, 111 optimal, 108 perceptually lossless, 85, 123 uniform, 99 Rate-distortion-function, 123 Reconstruction from sampled signals, 93 Relevance feedback, 121 Sampling circular frequency, 146 Sampling distance, 13, 146 Sampling rate, 146 Scale plane, 8 Scale space representation, 8 Shannon sampling condition, 128 reconstruction from sampled signals, 93

spectrum estimation, 129 Short time Fourier transform (STFT), 4.14 fast algorithm, 15 Signal continuous-time, 2, 125, 146 discrete (sampled), 13, 43, 128, 146 finite-energy, 125 Signal analysis, 1 Signal classification, 36 Signal compression, 1 Similarity retrieval, 116 Spectrum estimation, 129 STFT, 4, 14 Transfer function amplitude response, 131 biorthogonal filters, 75 dbn-filters, 60, 74 frequency response, 131 phase response, 131 Uncertainty relation, 128 Vanishing moments condition, 26 practical implications, 27 Variance, 116 Wavelet transform continuous transform (CWT), 4, 21 discrete transform (DWT), 43 Wavelets admissibility condition, 21 biorthogonal wavelets, 73 Coiflets, 85 Daubechies-wavelets, 53 dbn-wavelets, 55 generation by derivation, 26 Haar-wavelet, 26 mexican-hat-wavelet. 27 Morlet-wavelet, 28 vanishing moments, 26 WFT, 4, 14 Windowed Fourier transform (WFT), 4, 14