Adaptive construction of wavelets for image compression

Diploma thesis in computer science

Henning Thielemann

Tutor: Dipl.-Inform. Jörg Ritter

Martin-Luther-University Halle-Wittenberg
Institute of Computer Science

July 31, 2001

I affirm, that I have created this diploma thesis on my own, with the exclusive usage of the referenced literature.

Henning Thielemann

Contents

Introduction

The discrete wavelet transformation (see section 1.1 for details) has become a very popular tool to preprocess images to improve the performance of many tasks in the field of analysis and compression of signals. In this work we focus on image data and the enhancement of the compression results.

Before the JPEG-2000 standard the block-wise FOURIER transformation was used for image preprocessing in the JPEG File Interchange Format (JFIF) standard of the Joint Photographic Expert Group (JPEG) for a long time. The purpose of both wavelet and FOURIER transformation is to emphasize the important details of an image and suppress those which could be disregarded. The preprocessing itself is reversible in both cases, but using a FOURIER transformation algorithm it is harder to avoid rounding errors than in case of wavelets. That is because one dimensional wavelets can generally be computed with the lifting scheme which prevents from rounding errors. [4]

Once the transformation on an image is completed the compression is performed. It can be either lossless, which means that it converts to a more compact data representation only, or it can be lossy, in which case details are canceled up to a given threshold.

One can verify intuitively, that the wavelet decomposition is a more natural description of images than block-wise FOURIER transform: Imagine you get a picture A and the same picture B digitized at the double resolution. When lossily compressing A and B to the same file size, the reconstructed images will differ quite much if packed with JFIF because the blocks used for the FOURIER transform have different relative sizes compared to the sizes of A and B. Thus the blocks are relatively smaller in B and the scope for detecting structures is smaller.

The advantage of wavelet transformation is that since images are decomposed into informations about structures on each scale, one can easily obtain an image from B similar to A by ignoring information about small scales. That is one possibility to achieve compression in a lossy way.

In this work a further attempt is started to improve the image transformation for better compression results. The properties whose optimization is considered here are

1. High correlation between the relevances of details in the image and its transformed counterpart. Since the compression effect is achieved by neglecting details in the transformed image, it is important that details in the transformed image correspond to details in the original image and that high values in the transformed image correspond to more relevant features of the original image. This correlation can be mathematically expressed by close bounds of the wavelet transformation operator.

2. Small values for the most pixels in the transformed images which ensure a concentration of high values on a few pixels

This document is structured accordingly: After introducing in the mathematical background of the wavelet transformation and declaring some notations used in this document in chapter 1, the second chapter covers the determination and optimization of the EUCLIDean norm estimations between signal and wavelet coefficients vector. A family of wavelets called CHEBYSHEV wavelets is introduced, which has symmetrical filters as well as close norm bounds. We explore in detail later how weighting the filters of a wavelet filter pair influences the norm bounds. In the third chapter an image dependent construction of lifting steps with least mean square linear prediction is developed. Some variants of the basic scheme are presented. All wavelets which are discussed in this thesis are tested as preprocessing for a compression with an Embedded Zero Tree (EZT) coder. From all of these transformations (CHEBYSHEV, standard wavelet with weighting, linear prediction) the weighting method leads to the most improvement of the compression rate.

I want to thank Prof. Dr. Paul Molitor for being open-minded for student's problems always, Dipl.-Inform. Jörg Ritter for his extensive support for this work, my siblings for proof-reading, Helmut Podhaisky, Clemens Ladisch and Andreas Beckmann for their exhaustless pool of advises for LaTeX and V. Marino and L. Schmiedtchen for kindly supporting me with needed articles.

This document and the C++ source code of the complete transformation and compression software package are available from the included CD. This and possibly revised future versions may be also downloaded from

`http://www.henning-thielemann.de/`

Chapter 1

Mathematical background

1.1 What are wavelets?

Wavelets (little waves) are functions that fulfill certain self-similarity conditions. When talking about wavelets, we mostly mean a pair of functions: the scaling function ϕ and the wavelet function ψ. [16] Several extensions to this basic scheme exist, but for the introduction we will concentrate on this case. The self similarity (*refinement condition*) of the scaling function ϕ is bounded to a filter h and is defined by

$$\phi(t) = 2\sum_{k\in\mathbb{Z}} h_k \phi(2t-k) \qquad h_k \in \mathbb{R} \tag{1.1.1}$$

which means that ϕ remains unchanged if you compress it in t direction by a factor of 2, filter it with h and amplify the values by 2, successively (figure 1.1). One could also say, that ϕ is the eigenfunction with eigenvalue 1 of the linear operator that is described by the refinement. Since eigenfunctions are unique only if the amplitude is given, the scaling function is additionally normalized to

$$\sum_{k\in\mathbb{Z}} \phi(k) = 1$$

to make it unique.

The wavelet function ψ is built on ϕ with help of the filter g (figure 1.2):

$$\psi(t) = 2\sum_{k\in\mathbb{Z}} g_k \phi(2t-k) \qquad g_k \in \mathbb{R} \tag{1.1.2}$$

ϕ and ψ are uniquely determined by the filters h and g.

Variants of these functions are defined, which are translated by an integer, compressed by a power of two and usually amplified by a power of $\sqrt{2}$:

$$\begin{aligned} \psi_{j,l}(t) &= 2^{j/2}\psi(2^j t - l) \\ \phi_{j,l}(t) &= 2^{j/2}\phi(2^j t - l) \\ &\text{with } \{j,l\} \subset \mathbb{Z}, t \in \mathbb{R} \end{aligned} \tag{1.1.3}$$

- j denotes the scale – the bigger j the higher the frequency and the thinner the wavelet peak
- l denotes the translation – the bigger l the more shift to the right, and the bigger j the smaller the steps

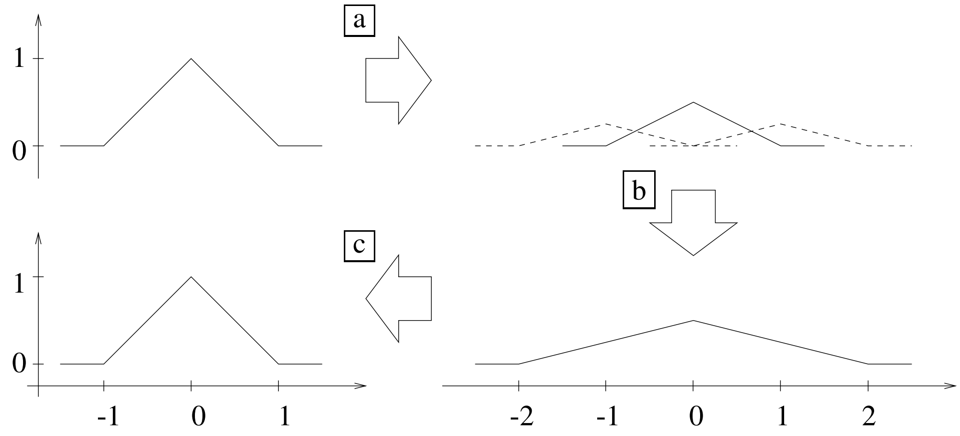


Figure 1.1: Refinement condition of the scaling function – In step $\boxed{a}$ the scaling function is duplicated, translated and amplified, the used filter coefficients $h_{-1} = \frac{1}{4}, h_0 = \frac{1}{2}, h_1 = \frac{1}{4}$ correspond to the scaling function filter of the CDF-2,2 wavelet that will be quoted frequently in this document. In step $\boxed{b}$ the translated duplicates are added (step $\boxed{a}$ and $\boxed{b}$ form the filtering). Step $\boxed{c}$ scales the function in abscissa and ordinate direction. A scaling function is characterized by being invariant under the sequence of the steps $\boxed{a}$, $\boxed{b}$, $\boxed{c}$.

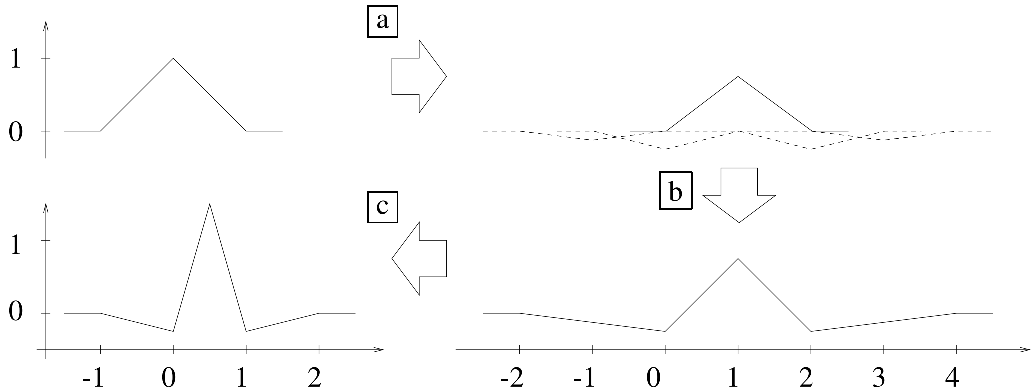


Figure 1.2: Building the wavelet function from scaling functions – The steps are analogous to figure 1.1. The filter g is borrowed from the CDF-2,2 wavelet, again, and is determined by the coefficients $g_{-1} = -\frac{1}{8}, g_0 = -\frac{1}{4}, g_1 = \frac{3}{4}, g_2 = -\frac{1}{4}, g_3 = -\frac{1}{8}$

The goal is to represent signals as linear combinations of wavelet functions at several scales and of scaling functions of the widest required scale. The choice of wavelet functions as primitives promises to be good, because natural signals like audio streams or images consist of the same structures at different scales and different positions.

$$x = \sum_{l\in\mathbb{Z}} c_l \phi_{-J,l} + \sum_{j=-J}^{0} \sum_{l\in\mathbb{Z}} d_{j,l}\psi_{j,l}$$

$c_l, d_{j,l}$ are the wavelet coefficients. They form the transformed signal we want to feed into a compression routine. J corresponds to the number of different scales we can represent, which is equal to the number of transformation levels that will be considered later in detail. The bigger J the more coarse structures can be described. A possible set of scaling and wavelet functions is shown in figure 1.3.

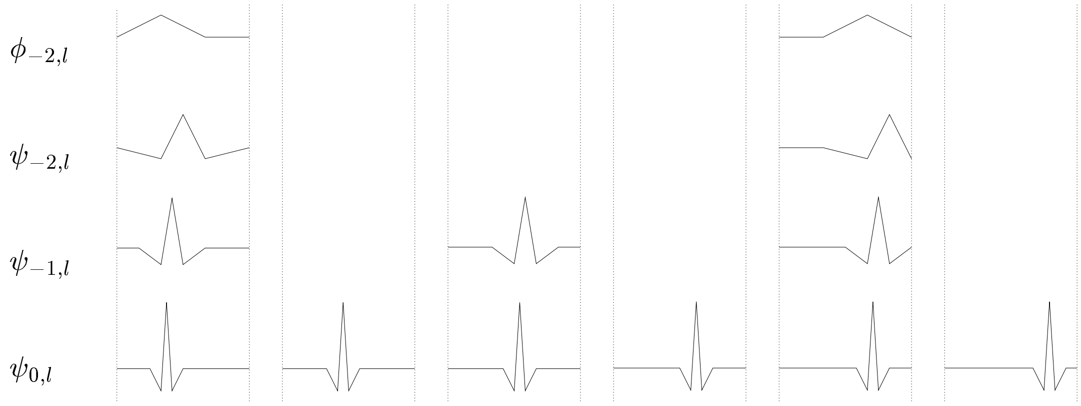


Figure 1.3: A basis consisting of scaling and wavelet functions of the CDF-2,2 wavelet – This example basis covers three levels of wavelet functions. Only a finite clip of translates is displayed. To visualize the translation of the functions, the abscissa is clipped to a finite interval.

What we usually start on, are discrete functions, known as sampled audio or image data. For simplicity we consider only one dimensional data. In the case of the two dimensional image data we process rows and columns separately. Its values $\ldots, x_{-1}, x_0, x_1, x_2, \ldots$ represent the amplitudes of pulses. If we want to integrate such signals into the wavelet theory we have to read the x_l as amplitudes of small scaling functions.

$$x = \sum_{l\in\mathbb{Z}} x_l \phi_{0,l}$$

Figure 1.4 gives an example for a signal that is approximated with a linear combination of scaling functions.

How can we convert between both signal representations? Retrieving the signal from the wavelet decomposition is the direction which follows from the refinement relation (1.1.2) immediately. In the wavelet decomposition of a signal we will replace all functions of the coarsest structure at level J by their refinements. This way we come to the wavelet decomposition of level $J-1$, what can be iterated

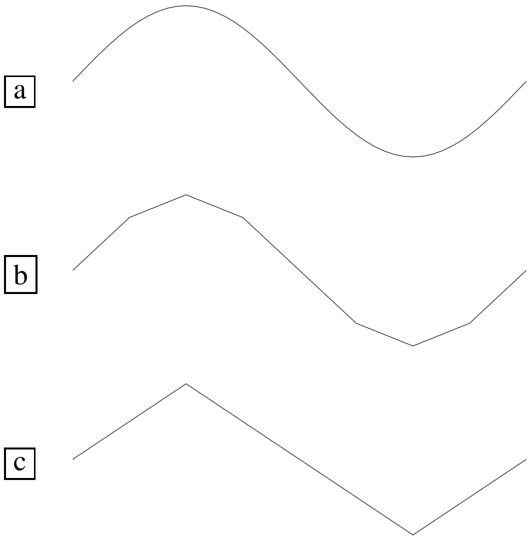


Figure 1.4: A sine signal approximated by a linear combination of scaling functions. $\boxed{\text{a}}$ is the original sine, $\boxed{\text{b}}$ is an approximation with scaling functions of level 0, $\boxed{\text{c}}$ is an approximation with scaling functions of level -1. If you have decomposed a signal into its wavelet representation with at least one level (which holds the information of $\boxed{\text{b}}$), you can obtain the approximation $\boxed{\text{c}}$ by erasing the wavelet coefficients of the wavelet functions at level 0 in the approximation $\boxed{\text{b}}$.

until reaching level 0 which is equal to the searched signal representation.

$$x(t) = \sum_{l\in\mathbb{Z}} c_l \phi_{-J,l}(t) + \sum_{j=-J}^{0} \sum_{l\in\mathbb{Z}} d_{j,l}\psi_{j,l}(t)$$

According to the definition of $\phi_{j,l}, \psi_{j,l}$ (1.1.3)

$$= 2^{-J/2}\left(\sum_{l\in\mathbb{Z}} c_l\phi(2^{-J}t-l) + \sum_{l\in\mathbb{Z}} d_{-J,l}\psi(2^{-J}t-l)\right) + \sum_{j=1-J}^{0}\sum_{l\in\mathbb{Z}} d_{j,l}\psi_{j,l}(t)$$

Apply refinement conditions (1.1.1),(1.1.2)

$$= 2^{1-J/2}\left(\sum_{l\in\mathbb{Z}} c_l \sum_{k\in\mathbb{Z}} h_k\phi(2\cdot(2^{-J}t-l)-k) + \sum_{l\in\mathbb{Z}} d_{-J,l}\sum_{k\in\mathbb{Z}} g_k\phi(2\cdot(2^{-J}t-l)-k)\right) + \sum_{j=1-J}^{0}\sum_{l\in\mathbb{Z}} d_{j,l}\psi_{j,l}(t)$$

Substitute $L := l$

$$= 2^{1-J/2}\sum_{k\in\mathbb{Z}}\left(\sum_{L\in\mathbb{Z}}(c_L h_k + d_{-J,L}g_k)\,\phi(2\cdot(2^{-J}t-L)-k)\right) + \sum_{j=1-J}^{0}\sum_{l\in\mathbb{Z}} d_{j,l}\psi_{j,l}(t)$$

Substitute back $l := 2L + k$

$$= 2^{1-J/2}\sum_{l\in\mathbb{Z}}\underbrace{\left(\sum_{L\in\mathbb{Z}} c_L h_{l-2L} + \sum_{L\in\mathbb{Z}} d_{-J,L}g_{l-2L}\right)}_{c'_l :=}\phi(2^{1-J}t-l) + \sum_{j=1-J}^{0}\sum_{l\in\mathbb{Z}} d_{j,l}\psi_{j,l}(t)$$

$$= \sum_{l\in\mathbb{Z}}\sqrt{2}c'_l\phi_{1-J,l}(t) + \sum_{j=1-J}^{0}\sum_{l\in\mathbb{Z}} d_{j,l}\psi_{j,l}(t)$$

Indeed, this is the signal representation as $J-1$ level wavelet decomposition. We see that the new coefficients c'_l are derived from c_l and $d_{-J,l}$ by a kind of filtering. The difference to traditional filtering is, that for even l, c'_l depends only on h_k and g_k with even k, and for odd l, c'_l depends only on h_k and g_k with odd k. This is the reason why we will split both g and h in its even and odd indexed coefficients for most of our investigations.

It is easy to see that the conversion from wavelet coefficients to signal values is possible without knowing ϕ or ψ, the only information needed, are the filters which belong to them. Under certain conditions, the same is true for the reverse conversion. This will become clearer in section 2.1.2. It allows us to limit our view to the filters g and h and hide the functions ϕ and ψ – they will not appear any longer in this document.

The Discrete Wavelet Transform (*DWT*) which is used to analyse signals for finding out specific properties or for further processes like compression consists of applying the filters g and h to the signal and obtaining a high frequency band H and a low frequency band L, both with the half resolution in t of the input signal. As in figure 1.5 the DWT is usually recursively applied on each L band to explore bigger structures of the original signal.

It is possible to extend this scheme in a way, that the H bands are filtered and split, too.

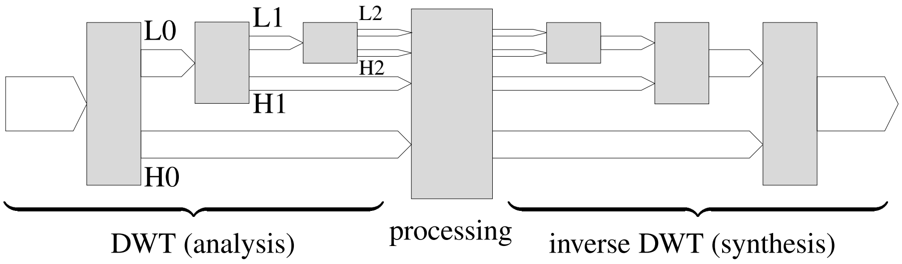


Figure 1.5: The tree of levels as produced by the wavelet transformation. – The input is split into the bands L0 and H0 with the half sample rate by applying the filters h and g, respectively. Then in turn, L0 is split into L1 and H1 with the same filter procedure and so on.

1.2 Notations

Let us introduce some advantageous notations that shall be used throughout this document.

Since we work with discrete signals here, signals and filters can be presented by vectors. In many cases there is no need to distinguish between signals and filters. In these cases we will call them vectors in general. A signal x starting at sampling time $m \in \mathbb{Z}$ ending at $n \in \mathbb{Z}, m \leq n$ is written as

$$x = (x_m, x_{m+1}, \ldots, x_{n-1}, x_n)^T .$$

Analogous a filter f with coefficients indexed from m to n is declared with

$$f = (f_m, f_{m+1}, \ldots, f_{n-1}, f_n)^T$$

Indeed m, n may be negative, which is uncommon for the usual vector notation. Because of that, the range of indices is not obvious in general. For that purpose we will emphasize the index 0:

$$(f_m, f_{m+1}, \ldots, f_{-1}, \boldsymbol{f_0}, f_1, \ldots, f_{n-1}, f_n) \qquad (m \leq 0, \;\; 0 \leq n)$$

Usually, we will choose the indices m and n so that the leading and trailing coefficient f_m and f_n are non-zero. The length of a vector x is defined as

$$|x| = n - m$$

This means that a vector consisting of one component has the length 0! This is wanted because it simplifies length calculations for filters applied to signals or for cascaded filters. $n - m + 1$ is called the *number of taps*.

The *convolution* $y = f * x$ of two vectors f and x is defined by the calculation of the components of y:

$$y_j = \sum_{k \in \mathbb{Z}} f_k x_{j-k}$$

One can easily verify that it holds:

$$|y| = |f| + |x|$$

Unless stated otherwise, filter coefficients and signal values are complex numbers.

Sometimes it is an advantage to imagine signals and filters as LAURENT polynomials.

$$x(z) = x_m z^m + x_{m+1} z^{m+1} + \cdots + x_{n-1} z^{n-1} + x_n z^n$$

We will use the notations x and $x(z)$ simultaneously. x means the vector as well as the associated polynomial. Vectors x and y can be convoluted $x * y$ whereas the polynomials x and y are multiplied $x \cdot y$ – the operation is the same. In opposition to x, $x(z)$ is a complex number. It is the value of the polynomial at the argument z. E.g. it is not useful to differentiate $x(z)$, but differentiating of x is. '$\circ$' should be used as general place holder for the argument of a polynomial. E.g. we can write $x(\circ^2)$, insinuating that $x(\circ^2)$ is a polynomial again.

With this notation we can write some other things quite easy.

$x(e^{-i\omega})$	the complex amplitude of the frequency $\omega \in \mathbb{R}$ in the signal x, thus $x(e^{-i\omega})$ provides the FOURIER transform of the signal x when varying ω
$x(-\circ)$	every even component of x is negated
$x(\circ^2)$	a zero coefficient is inserted between every component; the polynomial gets even-indexed coefficients exclusively
$x(\circ^{-1})$	the components are in reversed order
$f(z) \cdot x(z) = (f * x)(z)$	the product of polynomials is equal to the convolution of their coefficient vectors
$f(\circ) = f(\circ^{-1})$	identifies a symmetrical filter f, if all coefficients are real

The conjugated filter $\overline{f}$ of a filter f will be defined as follows:

$$\begin{aligned} f &= (f_m, f_{m+1}, \dots, f_{n-1}, f_n) \\ \overline{f} &= \left(\overline{f_n}, \overline{f_{n-1}}, \dots, \overline{f_{m+1}}, \overline{f_m}\right) \end{aligned}$$

Different from what you might expect, it does not hold, that the so defined conjugated filter assumes the conjugated values for equal arguments (i.e. $\overline{f}(z) = \overline{f(z)}$) in general. This is only true for arguments on the complex unit circle, i.e. $|z| = 1$, since

$$\begin{aligned} \overline{f(z)} &= \overline{\sum_{k\in\mathbb{Z}} f_k z^k} \\ &= \sum_{k\in\mathbb{Z}} \overline{f_k}\,\overline{z}^k \end{aligned}$$

because $|z| = 1$

$$\begin{aligned} &= \sum_{k\in\mathbb{Z}} \overline{f_k} z^{-k} \\ &= \sum_{k\in\mathbb{Z}} \overline{f_{-k}} z^k \\ &= \overline{f}(z) \end{aligned} \tag{1.2.1}$$

The definition of conjugated filters helps us to explain symmetrical filters. Since one expects of a symmetrical filter f with real coefficients that it does not change the phases of the frequencies of the signal it is applied to, we will use symmetry and linear phase behaviour as synonyms in the complex coefficients case, too. That means a symmetrical filter has to fulfill

$$\forall\, |z| = 1 : \quad f(z) \in \mathbb{R}$$

or equivalently

$$\forall\, |z| = 1 : \quad f(z) = \overline{f(z)} \stackrel{(1.2.1)}{=} \overline{f}(z)$$

Since the FOURIER transformation is injective it follows $f = \overline{f}$. That means that $f = \overline{f}$ is an equivalent formulation for linear phase filters.

Chapter 2

Transformation norm bounds

For lossy compression it is important to know how much a modification of the transformed signal distorts the restored signal. You expect that small modification of the transformed signal causes small distortions in the restored data. But there is an uncertainty in general.

Consider a vector x of the input signal (e.g. an image or audio data) and the operation W which is performed by a complete wavelet transformation. The transformed signal Wx is now modified by a lossy compression. The resulting signal after decompression shall be denoted by Wy. This is permitted, because every wavelet transformation usable for compression must be invertible, which means that for any modified transformed data there is an restored input signal y.

We are looking for an accurate estimation of how much the signal changes if a modification occurs on the transformed signal. For measuring the difference of two signals the peak signal-to-noise ratio (PSNR, see [5]) is widely used. It is a compromise between visual perception and easiness of calculation. The PSNR is a logarithmical scaled form of the EUCLIDean metric where the possible value range of the sampled data has an influence, too.

Let x, y be signals, each consisting of n values with a possible range of $[0, x_{\max}]$ (e.g. $[0, 255]$ for 8 bit images), then the peak signal-to-noise ratio is defined by

$$\begin{aligned}
\mathrm{PSNR}(x,y) &:= 20\log_{10}\frac{x_{\max}\cdot\sqrt{n}}{\|x-y\|_2}\,\mathrm{dB} \\
&= 10\,\mathrm{dB}\cdot\left(\log_{10}(x_{\max}^2\cdot n) - 2\log_{10}\|x-y\|_2\right)
\end{aligned}$$

Since the logarithm function is monotonous and the value range is constant, any optimization of the PSNR value can be done through optimizing the EUCLIDean norm, which is much easier.

Translated to this terms, we want estimations for $\|x-y\|_2$ depending on $\|Wx - Wy\|_2$ of the form

$$\forall x, y: \quad A\cdot\|x-y\|_2 \le \|Wx - Wy\|_2 \le B\cdot\|x-y\|_2$$

where the constants A and B (*bounds*) has to be determined. Good constants A and B are those, that lead to equality for some (not necessarily equal) pairs of x and y:

$$\begin{aligned}
\exists x, y: &\quad A\cdot\|x-y\|_2 = \|Wx - Wy\|_2 \\
\exists x, y: &\quad \|Wx - Wy\|_2 = B\cdot\|x-y\|_2
\end{aligned}$$

In the context of wavelet functions, constants A and B similar to these are called *frame bounds*. Estimations are given in [3], chapter 3.3. Since we are working with discrete sequences of coefficients instead of continuous functions and wavelet functions, this is not what we are looking for.

Since the wavelet transformation is a linear operator we get

$$Wx - Wy = W(x - y)$$

and we can replace $x - y$ by z.

$$\begin{aligned} \|x - y\|_2 &= \|z\|_2 \\ \|Wx - Wy\|_2 &= \|Wz\|_2 \end{aligned}$$

Thus we can limit the consideration of bounds to single signals rather than pairs of signals.

$$\forall z: \quad A \cdot \|z\|_2 \leq \|Wz\|_2 \leq B \cdot \|z\|_2$$

The discrete wavelet transformation is a linear operation on a finite vector space and thus associated with a matrix. Matrix operations are always bounded and its smallest bound is called *matrix norm*. So, on the one hand we can write as upper bound estimation

$$\|Wz\|_2 \quad \leq \quad \|W\|_2 \cdot \|z\|_2$$

on the other hand we get a lower bound with help of the identity $z = W^{-1}Wz$, provided that W is reversible

$$\begin{aligned} \|z\|_2 &= \left\|W^{-1}Wz\right\|_2 \\ &\leq \left\|W^{-1}\right\|_2 \cdot \|Wz\|_2 \end{aligned} \tag{2.0.1}$$

and finally

$$\|W\|_2^{-1} \cdot \|Wz\|_2 \leq \quad \|z\|_2 \quad \leq \left\|W^{-1}\right\|_2 \cdot \|Wz\|_2$$

or equivalently

$$\left\|W^{-1}\right\|_2^{-1} \cdot \|z\|_2 \leq \|Wz\|_2 \leq \|W\|_2 \cdot \|z\|_2$$

We note that $\left\|W^{-1}\right\|_2^{-1}$ is the lower bound for an operator W. Missing a symbol for lower bounds, it shall be used even in the case that W is not invertible. If W is not invertible, $\left\|W^{-1}\right\|_2$ could be interpreted as limit ∞ because there are non-zero vectors z which will be mapped to $Wz = 0$, and because (2.0.1) it is $\left\|W^{-1}\right\|_2 \geq \frac{\|z\|_2}{\|Wz\|_2} = \infty$ and this means $\left\|W^{-1}\right\|_2^{-1} = 0$.

Calculating the norm bounds for the whole transformation W is too complex, thus we will content with the bounds for one transformation step. How does the bounds for the whole transformation correlate with the bounds for the single transformation steps?

Let T_j, T_k be some transformation steps. We know from Linear Algebra that the EUCLIDean matrix norm is sub-multiplicative:

$$\|T_j \cdot T_k\|_2 \leq \|T_j\|_2 \cdot \|T_k\|_2 \tag{2.0.2}$$

We have to take into account, that later transformation steps work only on a part of the data that the former transformation steps have produced. But this does not change the bounds of the single transformation steps because there are always signals x that are transformed to $y = T_{j-1} \cdot \dots \cdot T_0 \cdot x$ where all coefficients of y that will not be processed by T_j are zero already. In other words: When searching for bounds for transformation steps, we need not to distinguish between the levels where the transformation is applied.

Thus for J transformation steps it follows:

$$\|W\|_2 = \left\| \prod_{j=0}^{J-1} T_j \right\|_2 \qquad \le \prod_{j=0}^{J-1} \|T_j\|_2$$

Analogous for the inverse transformation it follows:

$$\begin{aligned} \left\|W^{-1}\right\|_2 &= \left\| \prod_{j=J-1}^{0} T_j^{-1} \right\|_2 &\le \prod_{j=0}^{J-1} \left\|T_j^{-1}\right\|_2 \\ \left\|W^{-1}\right\|_2^{-1} &= \left\| \prod_{j=J-1}^{0} T_j^{-1} \right\|_2^{-1} &\ge \prod_{j=0}^{J-1} \left\|T_j^{-1}\right\|_2^{-1} \end{aligned}$$

For identical steps this leads to:

$$\begin{aligned} \|W\|_2 &= \left\| \prod_{j=0}^{J-1} T \right\|_2 &\le \|T\|_2^J \\ \left\|W^{-1}\right\|_2^{-1} &= \left\| \prod_{j=J-1}^{0} T^{-1} \right\|_2^{-1} &\ge \left\|T^{-1}\right\|_2^{-J} \end{aligned}$$

Remark. We realize, that the consideration of single transformation steps instead of the whole transformation results in coarser bounds.

2.1 Determining the EUCLIDean norm bounds for one wavelet transform step

2.1.1 Simple filters

The wavelet transformation consists mainly of signal filtering. Thus we will start on calculating the norm bounds when applying a plain linear filter f to a signal x:

$$y := f * x$$

The convolution performed by f can be written as a matrix

$$F = \begin{pmatrix} f_n & f_{n-1} & f_{n-2} & \cdots & f_{m+1} & f_m & 0 & \cdots & 0 & 0 & 0 \\ 0 & f_n & f_{n-1} & \cdots & f_{m+2} & f_{m+1} & f_m & \cdots & 0 & 0 & 0 \\ 0 & 0 & f_n & \cdots & f_{m+3} & f_{m+2} & f_{m+1} & \cdots & 0 & 0 & 0 \\ \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \\ 0 & 0 & 0 & \cdots & f_{n-1} & f_{n-2} & f_{n-3} & \cdots & f_m & 0 & 0 \\ 0 & 0 & 0 & \cdots & f_n & f_{n-1} & f_{n-2} & \cdots & f_{m+1} & f_m & 0 \\ 0 & 0 & 0 & \cdots & 0 & f_n & f_{n-1} & \cdots & f_{m+2} & f_{m+1} & f_m \end{pmatrix}$$

In general, a matrix norm $\|\cdot\|$ which is associated with a vector norm must fulfill for vectors x

$$\forall x : \quad \|Fx\| \leq \|F\| \cdot \|x\|$$

and the matrix norm is the smallest possible constant in that unequation. Thus it is defined as

$$\begin{aligned} \|F\| &= \sup_{x \neq 0} \frac{\|Fx\|}{\|x\|} \\ &= \sup_{x \neq 0} \left\| F \cdot \frac{x}{\|x\|} \right\| \end{aligned}$$

let $y := \frac{x}{\|x\|}$

$$= \max_{\|y\|=1} \|Fy\|$$

In case of the EUCLIDean norm an identity is known, which requires the calculation of the eigenvalues of $\overline{F}^T F$.

$$\begin{aligned} \|F\|_2 &= \sqrt{\lambda_{\max}\left(\overline{F}^T F\right)} \\ \left\|F^{-1}\right\|_2^{-1} &= \sqrt{\lambda_{\min}\left(\overline{F}^T F\right)} \end{aligned}$$

It is also well known, that the eigenvectors $x_{\max}$ and $x_{\min}$ associated with the eigenvalues $\lambda_{\max}$ and $\lambda_{\min}$ respectively, are the vectors where the bound estimations become equalities:

$$\begin{aligned} \|F \cdot x_{\max}\|_2 &= \|F\|_2 \cdot \|x_{\max}\|_2 \\ \|F \cdot x_{\min}\|_2 &= \left\|F^{-1}\right\|_2^{-1} \cdot \|x_{\min}\|_2 \end{aligned}$$

But determining the bounds with the help of the filter operation matrix is costly and the result depends on the signal length. But we remember that the FOURIER transform turns filtering into multiplying and that the EUCLIDean norms are left unchanged due to PARSEVAL's equation.

We have not considered values outside the known signal so far. We could fill them with zeros, or we could mirror the signal at its time boundaries to obtain a continuous extension (this is widely used for the computation of the DWT), but for easy filtering using the frequency spectrum the best choice is the assumption of periodic signals.

Let n be the length of x and assume this signal is periodic by n, then $\zeta^k := e^{-2\pi ik/n}$ identifies the frequency which has k cycles within the n values of the signal. With it you can describe the signal and the filtered signal in the frequency space by

$$\begin{aligned}\hat{x} &= \left(x(\zeta^0), x(\zeta^1), \ldots, x(\zeta^{n-1})\right)^T \\ \widehat{f*x} &= \left((f*x)(\zeta^0), (f*x)(\zeta^1), \ldots, (f*x)(\zeta^{n-1})\right)^T \\ &= \left(f(\zeta^0)\cdot x(\zeta^0), f(\zeta^1)\cdot x(\zeta^1), \ldots, f(\zeta^{n-1})\cdot x(\zeta^{n-1})\right)^T\end{aligned}$$

PARSEVAL now tells us that

$$\begin{aligned}\|x\|_2 &= \|\hat{x}\|_2 \\ \|f*x\|_2 &= \left\|\widehat{f*x}\right\|_2\end{aligned}$$

In other words: In the frequency space the filter operation is associated with a diagonal matrix $\hat{F}$

$$\begin{aligned}\widehat{f*x} &= \hat{F}\cdot\hat{x} \\ \hat{F} &= \begin{pmatrix} f(\zeta^0) & 0 & \cdots & 0 \\ 0 & f(\zeta^1) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & f(\zeta^{n-1}) \end{pmatrix}\end{aligned}$$

Since the norms of the vectors are equal, the matrix norms are equal, too, because the EUCLIDean matrix norm is defined via the vector norm.

$$\|F\|_2 = \left\|\hat{F}\right\|_2 \tag{2.1.1}$$

We know that $\overline{\hat{F}}^T\hat{F}$ is also a diagonal matrix then, and in this case the eigenvalues are just the values on the diagonal and the associated eigenvectors are unit vectors. We obtain

$$\begin{aligned}\left\|\hat{F}\right\|_2 &= \sqrt{\max\left\{\overline{f(\zeta^k)}f(\zeta^k) : k = 0,\ldots,n-1\right\}} \\ &= \max\left\{\sqrt{\overline{f(\zeta^k)}f(\zeta^k)} : k = 0,\ldots,n-1\right\} \\ &= \max\left\{\left|f(\zeta^k)\right| : k = 0,\ldots,n-1\right\}\end{aligned}$$

with $\hat{x} = \vec{e}_k$ the k. unit vector as the one where $\left\|\widehat{f*x}\right\|_2$ reaches the upper bound.

For better comprehensibility, we will also derive this conclusion immediately without usage of eigenvalues.

First, we show that the largest absolute frequency coefficient of the filter vector is a bound for the filter operation performed by f:

$$\begin{aligned}
\|f * x\|_2 &= \left\|\widehat{f * x}\right\|_2 \\
&= \left\|\left(f(\zeta^0) \cdot x(\zeta^0), \dots, f(\zeta^{n-1}) \cdot x(\zeta^{n-1})\right)^T\right\|_2 \\
&= \sqrt{\sum_{j=0}^{n-1} \overline{f(\zeta^j)} f(\zeta^j) \cdot \overline{x(\zeta^j)} x(\zeta^j)} \\
&\quad \text{let } k \text{ be an index with maximum } \left|f(\zeta^k)\right| \\
&\leq \sqrt{\sum_{j=0}^{n-1} \overline{f(\zeta^k)} f(\zeta^k) \cdot \overline{x(\zeta^j)} x(\zeta^j)} \\
&= \left|f(\zeta^k)\right| \cdot \sqrt{\sum_{j=0}^{n-1} \overline{x(\zeta^j)} x(\zeta^j)} \\
&= \left|f(\zeta^k)\right| \cdot \|x\|_2
\end{aligned}$$

You easily check that the bound is reached for $\hat{x} = \vec{e}_k$

$$\begin{aligned}
&\left\|\left(f(\zeta^0) \cdot x(\zeta^0), \dots, f(\zeta^{n-1}) \cdot x(\zeta^{n-1})\right)^T\right\|_2 \\
&= \left\|f(\zeta^k) \cdot \vec{e}_k\right\|_2 \\
&= \left|f(\zeta^k)\right| \cdot \|\vec{e}_k\|_2 \\
&= \left|f(\zeta^k)\right| \cdot \|\hat{x}\|_2
\end{aligned}$$

We see that both ways lead to the same result.

The k. unit vector in the frequency space is associated with a harmonic oscillation with the frequency $2\pi k/n$.

$$x_{\max} = \left(\zeta^0, \zeta^k, \zeta^{2k}, \dots, \zeta^{(n-1)k}\right)^T$$

That means that the norm bounds can always be reached with a harmonic oscillation of a specific frequency.

What we have derived as norm for the filter operation performed by f still depends on the signal length. To overcome this problem we have to extend the search for a maximum to different signal lengths of n. We slightly expand the meaning of F to the signal length independent filter operation performed by convoluting with f:

Because of (2.1.1) it is

$$\begin{aligned}
\|F\|_2 &= \left\|\hat{F}\right\|_2 \\
&= \sup\left\{\max\left\{\left|f(e^{-2\pi i k/n})\right| : k = 0, \dots, n-1\right\} : n \in \mathbb{N}\right\}
\end{aligned}$$

Substitute $r := k/n$

$$= \sup \left\{ \left| f(e^{-2\pi i r}) \right| : r \in \mathbb{Q} \cap [0,1) \right\}$$

$|x|$, f, e^x are continuous in $\mathbb{C}$

$$= \sup \left\{ \left| f(e^{-2\pi i r}) \right| : r \in \mathbb{R} \cap [0,1) \right\}$$
$$= \sup \left\{ |f(z)| : z \in \mathbb{C} \;\; \wedge \;\; |z| = 1 \right\}$$

f is a polynomial and thus continuous in $\mathbb{C}$

$$= \max \left\{ |f(z)| : z \in \mathbb{C} \;\; \wedge \;\; |z| = 1 \right\}$$

As shown in figure 2.1 we can interpret this as the search for a maximum of $|f(z)|$ on the complex unit circle.

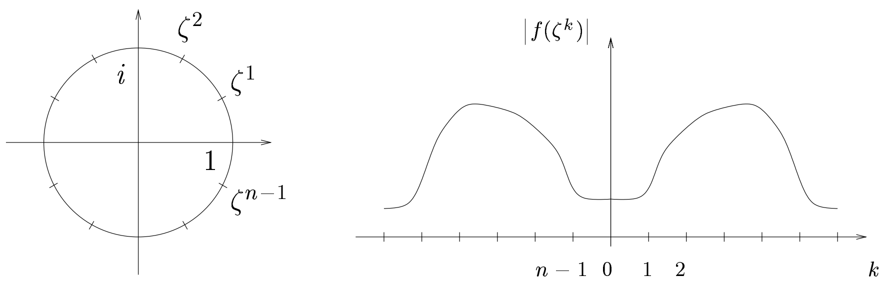


Figure 2.1: Search for a maximum on the complex unit circle – For a signal of the length n the circle is divided into n partitions. We have to determine $|f(z)|$ for each of the n points on the circle and choose the maximum value out of them. As the signal becomes longer, the grid will become finer. In the limit process we have to consider the whole continuous circle.

2.1.2 Filter matrices

Working with multiple bands: Now, consider filters which are used for wavelet transformations. More precise: We consider filters that are used in each transformation step. Here we process the signal with d filters rather than one filter. Usually, two filters are used for the wavelet transformation in one dimension, i.e. $d = 2$. If working in two dimensions, the number of channels (associated with the sub pictures) increases to $d = 4$. Eventually, every other number of filters d is possible for more general wavelet schemes, independent from the number of dimensions [7].

As we have seen in (2.0.2), we can get more accurate norm estimations, if we merge some consecutive transformation steps. Merging J transformation steps each consisting of d_j wavelet filters can be expressed by a single filter matrix operation with $\prod_{j=0}^{J-1} d_j$ output channels. In case of a four-filter wavelet transformation of an image over J levels the merging would result into a 4^J-filter wavelet.

Given a number of filters $f_0, f_1, \ldots, f_{d-1}$, the input signal x, the d output bands y_j, you can describe one wavelet transformation step by

$$\begin{aligned}
y_{0,0} &= \sum_{k=0}^{n-1} f_{0,k} x_{-k} \\
y_{1,0} &= \sum_{k=0}^{n-1} f_{1,k} x_{1-k} \\
&\vdots \quad \vdots \\
y_{d-1,0} &= \sum_{k=0}^{n-1} f_{d-1,k} x_{d-1-k} \\
y_{0,1} &= \sum_{k=0}^{n-1} f_{0,k} x_{d-k} \\
y_{1,1} &= \sum_{k=0}^{n-1} f_{1,k} x_{d+1-k} \\
&\vdots \quad \vdots
\end{aligned} \tag{2.1.2}$$

which is illustrated in figure 2.2, too.

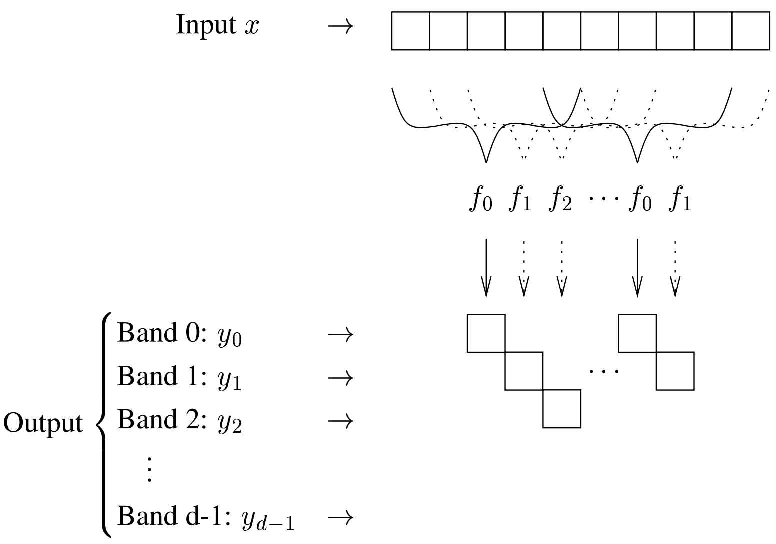


Figure 2.2: Wavelet transformation step with d filters

This representation has the disadvantage that it differs from the original filter scheme. Only every d. output value of a specific filter f_j is really used here. We can reduce the new scheme to the plain filter scheme if we split all filters f_j and the input signal x into d sub-filters $f_{j,k}$ and input bands x_k, respectively, with $0 \le k < d$. The coefficients of the sub-filters and input bands are:

$$\begin{aligned}
f_{j,k,l} &:= f_{j,l\cdot d+k} \\
x_{k,l} &:= x_{l\cdot d+k}
\end{aligned}$$

We will not consider every coefficient explicitly, but we will work with the denotations $f_{j,k}$ and x_k for the sub-filters and input bands. Unfortunately the denotation x_k for the k. input band coincides with the symbol for the k. value of the whole input signal x. Since we will no longer consider the input signal as one object, we mean the k. input band when writing x_k from now on.

With this convention we can rewrite the indexed equations (2.1.2) into one matrix multiplication with the convolution as "scalar" multiplication operation.

$$\underbrace{\begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_{d-1} \end{pmatrix}}_{\vec{y}:=} = \underbrace{\begin{pmatrix} f_{0,0} & f_{0,1} & \cdots & f_{0,d-1} \\ f_{1,0} & f_{1,1} & \cdots & f_{1,d-1} \\ \vdots & \vdots & \ddots & \vdots \\ f_{d-1,0} & f_{d-1,1} & \cdots & f_{d-1,d-1} \end{pmatrix}}_{\mathcal{F}:=} * \underbrace{\begin{pmatrix} x_0 \\ x_1 \\ \vdots \\ x_{d-1} \end{pmatrix}}_{\vec{x}:=} \tag{2.1.3}$$

$\mathcal{F}$ is the so called *polyphase matrix*.

Let us find an expression in the frequency space equivalent to (2.1.3). We introduced $\vec{x}$ which contains all input bands, in each component one of the d bands. Because of this one could imagine $\vec{x}$ as a matrix, but this is disadvantageous, because one would not have an explanation for $\mathcal{F} * \vec{x}$ then. We better imagine $\vec{x}$ as a vector consisting of polynomials. Then $\vec{x}(z)$ contains the amplitudes of the frequency associated with z for each band. We can also interpret this, as if each component of $\vec{x}$ stores the spectrum of one band. Similar interpretations can be found for $\vec{y}$ and $\mathcal{F}$.

$$\begin{aligned} \vec{x}(z) &:= (x_0(z), x_1(z), \ldots, x_{d-1}(z))^T \\ \vec{y}(z) &:= (y_0(z), y_1(z), \ldots, y_{d-1}(z))^T \\ \mathcal{F}(z) &:= \begin{pmatrix} f_{0,0}(z) & f_{0,1}(z) & \cdots & f_{0,d-1}(z) \\ f_{1,0}(z) & f_{1,1}(z) & \cdots & f_{1,d-1}(z) \\ \vdots & \vdots & \ddots & \vdots \\ f_{d-1,0}(z) & f_{d-1,1}(z) & \cdots & f_{d-1,d-1}(z) \end{pmatrix} \end{aligned}$$

Since the discrete FOURIER transformation is bijective, we can consider (2.1.3) at each of the frequencies $\omega = 2\pi k/n$ and get an equivalent formulation with $\zeta := e^{-2\pi i/n}$:

$$\forall k \in \{0, \ldots, n-1\} : \quad \vec{y}(\zeta^k) = \mathcal{F}(\zeta^k) \cdot \vec{x}(\zeta^k)$$

We put everything into one big matrix.

$$\begin{pmatrix} \vec{y}(\zeta^0) \\ \vec{y}(\zeta^1) \\ \vdots \\ \vec{y}(\zeta^{n-1}) \end{pmatrix} = \begin{pmatrix} \mathcal{F}(\zeta^0) & 0 & \cdots & 0 \\ 0 & \mathcal{F}(\zeta^1) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \mathcal{F}(\zeta^{n-1}) \end{pmatrix} \cdot \begin{pmatrix} \vec{x}(\zeta^0) \\ \vec{x}(\zeta^1) \\ \vdots \\ \vec{x}(\zeta^{n-1}) \end{pmatrix} \tag{2.1.4}$$

Determining the norm bounds: Finally, we have to find the norm of the big block diagonal $\mathcal{F}(\zeta^k)$-matrix. It is not difficult to see, that the EUCLIDean norm of the big matrix is equal to the biggest norm among the F blocks.

$$\left\| \operatorname{diag}\left(\mathcal{F}(\zeta^0), \mathcal{F}(\zeta^1), \ldots, \mathcal{F}(\zeta^{n-1})\right) \right\|_2 = \max\left\{ \left\| \mathcal{F}(e^{-2\pi i k/n}) \right\|_2 : k = 0, \ldots, n-1 \right\}$$

We also see that the vector, where the upper bound is reached, has the form $\vec{x}(\zeta^k) \cdot \vec{e}_k$.

That means that the bound is only reached, if all input bands contain harmonic oscillations with the same frequency but with specific phase differences and amplitude ratios.

We can formulate an identity analogous to PARSEVAL's theorem that proves, that the operator norm of the operation performed in (2.1.4) is equal to that of (2.1.3).

$$\begin{aligned}
\|x\|_2^2 &= \|x_0\|_2^2 + \|x_1\|_2^2 + \cdots + \|x_{d-1}\|_2^2 \\
&= \|\widehat{x_0}\|_2^2 + \|\widehat{x_1}\|_2^2 + \cdots + \left\|\widehat{x_{d-1}}\right\|_2^2 \\
&= \sum_{j=0}^{d-1}\sum_{k=0}^{n-1} \overline{x_j(\zeta^k)} x_j(\zeta^k)
\end{aligned}$$

Sort by frequencies instead of bands first

$$\begin{aligned}
&= \sum_{k=0}^{n-1}\sum_{j=0}^{d-1} \overline{x_j(\zeta^k)} x_j(\zeta^k) \\
&= \left\|\vec{x}(\zeta^0)\right\|_2^2 + \left\|\vec{x}(\zeta^1)\right\|_2^2 + \cdots + \left\|\vec{x}(\zeta^{n-1})\right\|_2^2 \\
&= \left\|\left(\vec{x}(\zeta^0), \vec{x}(\zeta^1), \ldots, \vec{x}(\zeta^{n-1})\right)^T\right\|_2^2
\end{aligned}$$

Thus, the vector norm over all bands (of either the input or the output) is equal to the norm over the spectra of all bands sorted by frequencies. This in turn means, that the matrix norm in the frequency space is equal to the operator norm of our filter matrix in the time space.

We want to achieve independency from the signal length again. This is very similar to the one-dimensional case with one filter and one input band. We want to reuse F as denotation for the signal length independent filter matrix operation, again.

$$\begin{aligned}
\|F\|_2 &= \sup\left\{\max\left\{\left\|\mathcal{F}(e^{-2\pi ik/n})\right\|_2 : k = 0, \ldots, n-1\right\} : n \in \mathbb{N}\right\} \\
&= \sup\left\{\left\|\mathcal{F}(e^{-2\pi ir})\right\|_2 : r \in \mathbb{Q} \cap [0,1)\right\} \\
&= \sup\left\{\left\|\mathcal{F}(e^{-2\pi ir})\right\|_2 : r \in \mathbb{R} \cap [0,1)\right\} \\
&= \sup\left\{\|\mathcal{F}(z)\|_2 : z \in \mathbb{C} \ \wedge\ |z| = 1\right\} \\
&= \max\left\{\|\mathcal{F}(z)\|_2 : z \in \mathbb{C} \ \wedge\ |z| = 1\right\} \\
&= \max\left\{\sqrt{\lambda_{\max}\left(\overline{\mathcal{F}(z)}^T \mathcal{F}(z)\right)} : z \in \mathbb{C} \ \wedge\ |z| = 1\right\}
\end{aligned}$$

In the same way you find the lower norm bound of the filter matrix operation

$$\begin{aligned}
\left\|F^{-1}\right\|_2 &= \max\left\{\sqrt{\lambda_{\max}\left(\overline{\mathcal{F}^{-1}(z)}^T \mathcal{F}^{-1}(z)\right)} : z \in \mathbb{C} \ \wedge\ |z| = 1\right\} \\
&= \max\left\{\sqrt{\lambda_{\min}\left(\overline{\mathcal{F}(z)}^T \mathcal{F}(z)\right)^{-1}} : z \in \mathbb{C} \ \wedge\ |z| = 1\right\} \\
&= \min\left\{\sqrt{\lambda_{\min}\left(\overline{\mathcal{F}(z)}^T \mathcal{F}(z)\right)} : z \in \mathbb{C} \ \wedge\ |z| = 1\right\}^{-1} \\
\left\|F^{-1}\right\|_2^{-1} &= \min\left\{\sqrt{\lambda_{\min}\left(\overline{\mathcal{F}(z)}^T \mathcal{F}(z)\right)} : z \in \mathbb{C} \ \wedge\ |z| = 1\right\}
\end{aligned}$$

Inversion of filter matrices: Let us observe some details of the invertibility of $\mathcal{F}$. We have seen that one wavelet transformation step can be written as

$$\vec{y} = \mathcal{F} * \vec{x}$$

The transformation for image preprocessing should be lossless, so that a lossless compressor can be implemented with it. In mathematical terms: The transformation operator should be invertible, i.e.

$$\vec{x} = \mathcal{F}^{-1} * \vec{y}$$

It simplifies theory and implementation, if one can use the same structure for the analysis and synthesis transformation. In other words, we are interested in those $\mathcal{F}$'s where $\mathcal{F}^{-1}$ is a filter matrix, too, and it holds

$$\mathcal{F}^{-1} * \mathcal{F} = I$$

With a determinant that uses the convolution of filters as multiplication, the determinant product theorem holds.

$$\begin{aligned} \det(I) &= \det\left(\mathcal{F}^{-1} * \mathcal{F}\right) \\ 1 &= \det\left(\mathcal{F}^{-1}\right) * \det\left(\mathcal{F}\right) \end{aligned}$$

Both $\det \mathcal{F}^{-1}$ and $\det \mathcal{F}$ are polynomials. Since we work with filters of finite length (*finite impulse response filters, FIR*), factorization of 1 into polynomials is only possible if the factors are monomials. (This is analogous to the case of 2×2 matrices presented in [4].) Let be $\det \mathcal{F} = c \cdot z^k$ with $c \in \mathbb{C} \setminus \{0\}$ we can eliminate the scaling c and the shifting z^k without loss of generality:

The shifting z^k can be avoided if the signal is rotated by k positions. This is no serious modification, because the signal was assumed to be periodic. In return the rows of $\mathcal{F}$ are rotated $k \bmod d$ rows upwardly, the last $k \bmod d$ rows are divided by $z^{\lfloor \frac{k}{d} \rfloor + 1}$ and the others are divided by $z^{\lfloor \frac{k}{d} \rfloor}$. With this shifting it can be always assured that $\det \mathcal{F} = c$.

The scaling c can be removed if every filter of $\mathcal{F}$ is divided by the same k. complex root of c. This is equivalent to the uniform amplification of the output bands by $\sqrt[k]{c}$. Thus $\mathcal{F}$ can always be normalized to $\det \mathcal{F} = 1$. This does not change the characteristics of the output signal and may help to balance the numerical properties of the analysis and the synthesis transformation.

With the same structure for a filter matrix operation and its reverse operation, there is no need to distinguish between them in further explorations. The difference within the whole wavelet transform is, that the decomposition transformation (*analysis*) requires the splitting of an input signal into d bands before applying the filter matrix, whereas the matrix operation in the composition transformation (*synthesis*) is post-processed with the joining (interleaving) of the d output bands.

In the following sections we will consider the decomposition direction exclusively and we will always assert, that the decomposition is invertible.

2.1.3 Filter 2×2 matrices

We want to deduce the case of 2 filters from the previous results. Our filter matrix is now

$$\mathcal{F} = \begin{pmatrix} f_{0,0} & f_{0,1} \\ f_{1,0} & f_{1,1} \end{pmatrix}$$

Since we can assume $\det \mathcal{F} = 1$ according to section 2.1.2, the inverse matrix can be easily given.

$$\mathcal{F}^{-1} = \begin{pmatrix} f_{1,1} & -f_{0,1} \\ -f_{1,0} & f_{0,0} \end{pmatrix} \tag{2.1.5}$$

We will switch to a shorter notation for $\mathcal{F}$ as used in [4]. The names h and g for the filters were already used in the introduction in section 1.1.

$$P = \begin{pmatrix} h_e & h_o \\ g_e & g_o \end{pmatrix} \tag{2.1.6}$$

Note that we talk about the decomposition transformation. Because of that we had to use the tilde denotation $\widetilde{P}, \widetilde{g_e}, \widetilde{h_e}, \ldots$ if strictly following [4], but we have omitted it for simplicity. (2.1.5) shows that the structural difference between P and $\widetilde{P}$ is even smaller than in the case of arbitrary number of bands.

Determining the norm bounds: For determining the EUCLIDean operator norm of P we have to know the eigenvalues of $\overline{P}^T P$.

$$\begin{aligned} \overline{P}^T P &= \begin{pmatrix} \overline{h_e}h_e + \overline{g_e}g_e & \overline{h_e}h_o + \overline{g_e}g_o \\ \overline{h_o}h_e + \overline{g_o}g_e & \overline{h_o}h_o + \overline{g_o}g_o \end{pmatrix} \\ &=: \begin{pmatrix} a & \overline{b} \\ b & c \end{pmatrix} \end{aligned} \tag{2.1.7}$$

The eigenvalues of $\overline{P}^T P$ can be obtained as zeros of the characteristic polynomial of $\overline{P}^T P$:

$$\begin{aligned} 0 &= \det\left(\overline{P}^T P - \lambda I\right) \\ &= \det\begin{pmatrix} a-\lambda & \overline{b} \\ b & c-\lambda \end{pmatrix} \\ &= (a-\lambda)(c-\lambda) - \overline{b}b \\ &= \lambda^2 - (a+c)\lambda + ac - \overline{b}b \end{aligned} \tag{2.1.8}$$

$$\lambda\left(\overline{P}^T P\right) = \frac{1}{2}\left(a + c \pm \sqrt{(a+c)^2 - 4(ac - \overline{b}b)}\right) \tag{2.1.9}$$

$$= \frac{1}{2}\left(a + c \pm \sqrt{(a-c)^2 + 4\overline{b}b}\right) \tag{2.1.10}$$

We can see in (2.1.10) that the radicand is non-negative and that is why λ is always real. Additionally the formulation in (2.1.9) makes clear that $\lambda \geq 0$. You can also derive both facts from the general theory of matrices, however.

For simplification we want to introduce two variables which will be used frequently in this work:

$$p := \tfrac{1}{2}(a+c) \tag{2.1.11}$$

$$= \tfrac{1}{2}\left(\overline{h_e}h_e + \overline{g_e}g_e + \overline{h_o}h_o + \overline{g_o}g_o\right) \tag{2.1.12}$$

$$= \frac{1}{2}\|P\|_F^2 \qquad \text{(FROBENIUS' norm of } P\text{)}$$

$$q := \det(P)$$

$$= h_e g_o - h_o g_e \tag{2.1.13}$$

$\Rightarrow$

$$\begin{aligned} ac - \overline{b}b &= \det(\overline{P}^T P) \\ &= \overline{\det(P)}\det(P) \\ &= \overline{q}q \end{aligned}$$

$\Rightarrow$

$$\begin{aligned} 0 &= \lambda^2 - 2p\lambda + \overline{q}q \\ \lambda &= p \pm \sqrt{p^2 - \overline{q}q} \qquad (2.1.14) \\ \lambda_+ &:= p + \sqrt{p^2 - \overline{q}q} \\ \lambda_- &:= p - \sqrt{p^2 - \overline{q}q} \\ \|F\|_2 &= \sqrt{p + \sqrt{p^2 - \overline{q}q}} \\ \left\|F^{-1}\right\|_2^{-1} &= \sqrt{p - \sqrt{p^2 - \overline{q}q}} \qquad (2.1.15) \end{aligned}$$

Remember that p and q are functions of z, a value on the complex unit circle which marks the considered frequency. To find the norm of the filter matrix operation we have to search for the maximum λ regarding z. Since the square root expression in (2.1.14) is non-negative, it is clear that the maximum $\lambda_{\max}$ among all λ's is a λ_+ and the minimum $\lambda_{\min}$ is one of the possible λ_-'s. As explained in section 2.1.2, $q = 1$ can be adjusted for every invertible wavelet filter pair. That is why $\lambda_+ \cdot \lambda_- = \overline{q}q = 1$ and that means that λ_+ is maximal if and only if λ_- is minimal. Since λ_+ depends monotonely on p, λ_+ is maximal if and only if p is maximal. One consequence is that the lower and the upper norm bounds of a wavelet transform step are reached when a certain frequency occurs on both input bands. They may differ in phase and amplitude, though.

An example is given in section 2.1.6.

Signals where the bounds are reached: We can calculate a signal where the norm bounds are reached by calculating the eigenvectors of $\overline{P}^T P$ for that z where $\lambda_+(z) = \lambda_{\max}$. Two cases may occur:

1. $\lambda_{\min} = \lambda_{\max}$
   Because $\lambda_{\min}\lambda_{\max} = 1$ and $\lambda \geq 0$ it must be $\lambda_{\max} = 1$ and the characteristic polynomial is $(\lambda - 1)^2$. According to (2.1.8) it follows $b = 0$ and $\overline{P}^T P = I$. This means that any vector is eigenvector of the eigenvalue 1. This denotes the case of unitary wavelets (*orthogonal* when working in $\mathbb{R}$).

2. $\lambda_{\min} < \lambda_{\max}$
   That means that the multiplicity of each of the eigenvalues $\lambda_{\min}$ and $\lambda_{\max}$ is below 2. Since any eigenvalue is associated with at least one eigenvector, both $\lambda_{\min}$ and $\lambda_{\max}$ must have exactly one eigenvector. You can easily guess it:

   $$(d, b)^T := (\lambda - c, b)^T = \left(\frac{1}{2}\left(a - c \pm \sqrt{(a-c)^2 + 4\overline{b}b}\right), b\right)^T$$

   and this is the only one, when ignoring any scaling.

$$\left(\ldots, d\zeta^{-1}, b\zeta^{-1}, \boldsymbol{d\zeta^0}, b\zeta^0, d\zeta^1, b\zeta^1, \ldots\right)$$

is the signal where the upper bound is reached if $d = \lambda_{\max} - c$ and the lower bound if $d = \lambda_{\min} - c$.

2.1.4 Conclusions

Balance of norms: An interesting conclusion is that if the polyphase determinant is normalized to 1 the geometrical average of $\lambda_{\min}$ and $\lambda_{\max}$ ($\sqrt{\lambda_{\min}\lambda_{\max}}$) is 1. That means that the lower and upper bounds are balanced or in other words that the transformation does not change the norm of the signal on average.

Invertibility: You can also verify invertibility conditions with (2.1.14). According to (2.1.15) it holds $\lambda_{\min} = 0 \Leftrightarrow \left\|F^{-1}\right\|_2^{-1} = 0$. We will show that $\lambda_{\min} = 0$ is equivalent to singularity of F.

1. "$\Rightarrow$"

$$\begin{aligned}
\lambda_{\min} = 0 \quad &\Rightarrow \quad \|F\|_2 = 0 \\
&\Rightarrow \quad \exists x : \|x\|_2 = 1 \quad \wedge \quad \|Fx\|_2 = 0 \\
&\Rightarrow \quad \exists x : x \neq 0 \quad \wedge \quad Fx = 0 \\
&\Rightarrow \quad F \text{ is not invertible}
\end{aligned}$$

2. "$\Leftarrow$"
   Let F be non-invertible, which means that for some different x, y it is

$$\begin{aligned}
Fx &= Fy \\
\Rightarrow 0 &= Fx - Fy \qquad | \qquad F \text{ is linear} \\
&= F \cdot (x - y) \\
0 &= \|F \cdot (x - y)\|_2 \\
&\geq \sqrt{\lambda_{\min}} \cdot \|x - y\|_2
\end{aligned}$$

   Because $x \neq y$ it is $x - y \neq 0$ and $\|x - y\|_2 > 0$, thus $\lambda_{\min} = 0$.

Indeed, singularity of F is equivalent to $\lambda_{\min} = 0$ which is equivalent to $q = 0$ for some z because of (2.1.14). This coincides with our observation in section 2.1.2, that the determinant of an invertible polyphase matrix can only be a monomial $c \cdot z^k$ with $c \neq 0, |z| = 1$.

Unitarity: You can also derive the condition for unitarity (orthogonality if working in $\mathbb{R}$) of $\mathcal{F}$, which is already known from [3], theorem 10.1.6, that means $\overline{\mathcal{F}}^T \mathcal{F} = I$. In case of two wavelet filters you get

$$\begin{aligned} \overline{h_e}h_e + \overline{g_e}g_e &= 1 \\ \overline{h_o}h_o + \overline{g_o}g_o &= 1 \\ \overline{h_o}h_e + \overline{g_o}g_e &= 0 \end{aligned} \tag{2.1.16}$$

Because of (2.1.11) and (2.1.14) this is equivalent to

$$\overline{\mathcal{F}}^T \mathcal{F} = I \quad \Leftrightarrow \quad p = 1 \ \wedge \ \overline{q}q = 1$$

Because of our conventions, $q = 1$ is always ensured, so that $p = 1$ is the important criterion.

Unitarity and symmetry: Additionally, we can derive from (2.1.16) the fact, that the lazy wavelet (which is associated with the identity matrix as polyphase matrix, $P = I$) is the only symmetric unitary two-channel wavelet with an odd number of filter taps. Note that this situation is different from the one considered in [3], theorem 8.1.4, where filters with even numbers of taps are considered. Therefore the result found there is the HAAR wavelet.

We can prove our statement in two ways:

One possibility is to write down the coefficients of $\overline{h_e}h_e + \overline{g_e}g_e$, compare them with $(\dots, 0, \mathbf{1}, 0, \dots)$ and induce from the outer to the inner coefficients, that all coefficients of both h_e and g_e have to be zero, except of $h_{e,0}$ which is 1.

The other way can be written more formally:

Let h_e, h_o, g_e, g_o be symmetrical in the sense that

$$\begin{aligned} \overline{h_e}(z) &= h_e(z) & \overline{g_e}(z) &= g_e(z) \cdot z^{-1} \\ \overline{h_o}(z) &= h_o(z) \cdot z & \overline{g_o}(z) &= g_o(z) \end{aligned} \tag{2.1.17}$$

$$\begin{aligned} \overline{h_e}h_e(z) + \overline{g_e}g_e(z) &= 1 \qquad \mid \quad \text{symmetry of } h_e, g_e \\ h_e^2(z) + z g_e^2(z) &= 1 \\ h_e^2(z^2) + z^2 g_e^2(z^2) &= 1 \\ \left(h_e(z^2) + iz g_e(z^2)\right)\left(h_e(z^2) - iz g_e(z^2)\right) &= 1 \end{aligned} \tag{2.1.18}$$

We know that 1 can be factored into monomials only, that means $h_e(z^2) + izg_e(z^2) = cz^k$. $h_e(z^2)$ has only even exponents for z, whereas $izg_e(z^2)$ has only odd exponents. This means one of h_e and g_e must be zero and the other a monomial. This leads to two cases:

1. $h_e(z^2) = 0$
   (2.1.18) $\Rightarrow z^2 g_e^2(z^2) = 1 \Rightarrow zg_e(z^2) = \pm 1$ which is obviously impossible, because $zg_e(z^2)$ contains only odd exponents of z.

2. $izg_e(z^2) = 0$
   (2.1.18) $\Rightarrow h_e^2(z^2) = 1 \Rightarrow h_e(z^2) = \pm 1 \Rightarrow h_e(z) = \pm 1$ which is what we have claimed.

The same strategy can be used to conclude that $h_o(z) = 0 \ \wedge \ g_o(z) = \pm 1$.

2.1.5 Approximation of bounds for 2-filter wavelets

Transform z to real axis: In section 2.1.3 we found out that we have to maximize $p(z)$ on the complex unit circle in order to maximize $\lambda_+(z)$ and to minimize $\lambda_-(z)$. Since p is a polynomial we can do this by locating the zeros of the derivative of p. A problem that remains is that we have to differentiate along the circle.

One possibility to fulfill this restriction, is to consider z as a function of the angle ω:

$$z = e^{i\omega}$$

The derivative of p regarding ω is a polynomial in z again:

$$\begin{aligned}
\frac{\mathrm{d}}{\mathrm{d}\,\omega} p(z) &= \frac{\mathrm{d}}{\mathrm{d}\,\omega} \sum_{k\in\mathbb{Z}} p_k z^k \\
&= \sum_{k\in\mathbb{Z}} p_k \frac{\mathrm{d}}{\mathrm{d}\,\omega} e^{ik\omega} \\
&= \sum_{k\in\mathbb{Z}} p_k ik e^{ik\omega} \\
&= i \sum_{k\in\mathbb{Z}} k p_k \cdot z^k
\end{aligned}$$

We have to find all complex zeros z_j of $\frac{\mathrm{d}}{\mathrm{d}\,\omega} p(z)$. From all z_j that are on the complex unit circle ($|z_j| = 1$), the one with maximal $p(z_j)$ is the one we need.

Calculation may be simplified, if we reduce our problem to a function on the real domain. We can do this by projecting $p(z)$ on the plane spanned by the real axis and the value axis. The symmetry of p assures that corresponding values on both half circles are equal. We simply use the real part of z and rewrite the polynomials p and q to depend from it. As sketched in figure 2.3 the values for $z = e^{i\omega}$ for $\omega \in [-\pi, 0]$ are mapped to $r \in [-1, 1]$.

$$\begin{aligned}
r &:= \Re(z) \\
&= \tfrac{1}{2}(z + \overline{z}) \qquad \Big| \qquad |z| = 1 \Rightarrow \overline{z} = z^{-1} \\
&= \tfrac{1}{2}(z + z^{-1}) \\
\widetilde{p}(r) &:= p(z) \\
\widetilde{\overline{q}q}(r) &:= \overline{q}q(z)
\end{aligned} \tag{2.1.19}$$

Since p and $\overline{q}q$ are symmetrical polynomials it is possible to determine the coefficients for the polynomials in r if we start on the outermost coefficients of p and $\overline{q}q$, then subtract the found r-powers expressed in z, then iterate on the remaining polynomial. The `maple` procedure in figure 2.4 shows the details.

Find extrema: Now we have to find the global extrema of the real polynomial $\widetilde{p}$ in the range $[-1, 1]$. We have to search for the real zeros $\{r_0, r_1, \ldots, r_{m-1}\}$ of $\frac{\mathrm{d}}{\mathrm{d}\,r}\widetilde{p}(r)$, where m denotes the number of real zeros. For simplification we introduce

$$s(r) := \frac{\mathrm{d}}{\mathrm{d}\,r} \widetilde{p}(r)$$

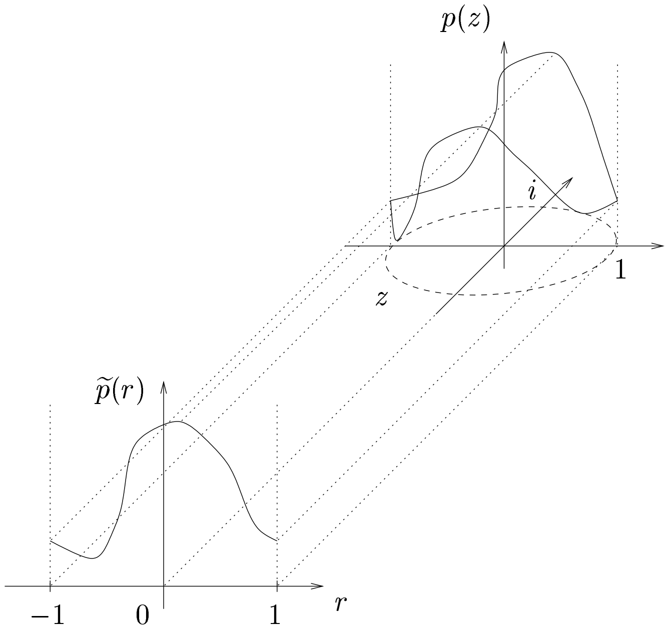


Figure 2.3: The transformation from z to r corresponds to the projection of the complex unit circle onto the real axis.

```
transformZ2R := proc(pp, z, r)
  local p, q, c, k;
    p := pp ;
    q := 0 ;
    while 0 < degree(p) do
      if degree(p, z) + ldegree(p, z) ≠ 0 then
        ERROR("Non-symmetric exponents")
      fi;
      if lcoeff(p, z) ≠ tcoeff(p, z) then
        ERROR("Non-symmetric coefficients")
      fi;
      c := lcoeff(p, z) ;
      k := degree(p, z) ;
      p := p − expand(c × (z + 1/z)^k) ;
      q := q + c × (2 × r)^k
    od;
    q := q + p ;
    RETURN(q)
  end
```

Figure 2.4: `maple` procedure for converting a polynomial of z on the complex unit circle into a polynomial of $r \in [-1, 1]$

Further on we have to take into account, that the global maximum might be at the borders of $[-1,1]$.

Thus it holds that

$$\begin{aligned}\max_{z\in\mathbb{C}\ \wedge\ |z|=1} p(z) &= \max_{r\in\mathbb{R}\cap[-1,1]} \widetilde{p}(r) \\ &= \max\{\widetilde{p}(r) : r\in\{-1,1,r_0,r_1,\ldots,r_{m-1}\}\}\end{aligned}$$

In our software package the search for zeros is performed by a NEWTON-MAEHLY iteration [19] which calculates all zeros simultaneously. Let n be the degree of s $(m \le n)$ and $\{r_0^*, r_1^*, \ldots, r_{n-1}^*\}$ the set of approximations of the zeros of $s(r)$. We choose one zero of them, say r_{n-1}^* and divide s by a polynomial s^* which consists of all other approximated zeros and run a NEWTON iteration on this quotient function $t := \frac{s}{s^*}$. If the approximations of the other zeros are good, t will be almost linear, otherwise t is a function that has the same zeros as s and additionally some singularities at the approximated zeros.

$$s^*(r) \;:=\; \sum_{k=0}^{n-2}(r - r_k^*)$$

Using the shorthand $r = r_{n-1}^*$ the NEWTON iteration step for refining the approximation r is:

$$r \;:=\; r - \frac{t(r)}{t'(r)}$$

where $\frac{t}{t'}$ can be simplified as follows:

$$\begin{aligned}\frac{t}{t'} &= \frac{(s/s^*)}{(s/s^*)'} \\ &= \frac{s}{s^*}\cdot\frac{s^{*2}}{s's^* - ss^{*\prime}} \\ &= \frac{ss^*}{s's^* - ss^{*\prime}} \\ &= \frac{s}{s' - s\cdot\frac{s^{*\prime}}{s^*}}\end{aligned}$$

Because of the known factorization representation of s^* and the differentiation rule for products, $\frac{s^{*\prime}}{s^*}$ can also be simplified:

$$\frac{s^{*\prime}(r)}{s^*(r)} \;=\; \sum_{k=0}^{n-2}\frac{1}{r - r_k^*}$$

The complete NEWTON iteration step is then:

$$r \;:=\; r - \underbrace{\frac{s(r)}{s'(r) - s(r)\cdot\sum_{k=0}^{n-2}\frac{1}{r-r_k^*}}}_{\Delta r :=}$$

$s(r)$ and $s'(r)$ can be calculated with HORNER's scheme. To ensure that exactly n zeros can be found, the algorithm is implemented with complex numbers. The iteration will be started with quite random values, since only a pair of equal values is problematically. When we choose one approximation r^* out of a pair of equal approximations $r_j^* = r_k^*$, it is $r^* - r_k^* = 0$, $\frac{1}{r^*-r_k^*} = \infty$ and thus $\Delta r = 0$. This means that the iteration will not change r, independent from whether r is a good approximation or not. Furthermore not all of the initial values will be chosen to be real, otherwise the iteration will produce real values exclusively and thus can not find non-real zeros.

To find all zeros of a polynomial, the zeros are selected successively from $\{r_0^*, r_1^*, \dots, r_{n-1}^*\}$ to do an iteration for improving the approximation. This is repeated again and again until the approximated zeros do not change noticeable anymore or $|s(r)|$ falls below a given limit.

2.1.6 Example: Bounds for CDF-2,2 wavelet

The CDF-2,2 should serve as an example wavelet for which we want to determine the norm bounds. Remember, because we are talking about analysis transformation but omitted the tilde for filter notation, the filters h and g given here, are not the ones used in the introduction in section 1.1.

$$\begin{array}{rcllllll} h & := & \frac{1}{8}(& -1 & 2 & \mathbf{6} & 2 & -1 \quad) \\ g & := & \frac{1}{2}(& & & \mathbf{-1} & 2 & -1 \quad) \end{array}$$

$$\begin{array}{rclllll} h_e & := & \frac{1}{8}(& -1 & \mathbf{6} & -1 &) \\ g_e & := & \frac{1}{2}(& & \mathbf{-1} & -1 &) \end{array} \qquad \begin{array}{rcllll} h_o & := & \frac{1}{4}(& 1 & \mathbf{1} &) \\ g_o & := & (& & \mathbf{1} &) \end{array}$$

Remember, that the the multiplication of polynomials is equal to the convolution of their coefficients (section 1.2). The required polynomial products evaluate to:

$$\begin{array}{rrcllllll} & \overline{h_e}h_e & = & \frac{1}{64}(& 1 & -12 & \mathbf{38} & -12 & 1 \quad) \\ & \overline{g_e}g_e & = & \frac{1}{4}(& & 1 & \mathbf{2} & 1 & \quad) \\ & \overline{h_o}h_o & = & \frac{1}{16}(& & 1 & \mathbf{2} & 1 & \quad) \\ & \overline{g_o}g_o & = & (& & & \mathbf{1} & & \quad) \\ (2.1.12) \Rightarrow & p & = & \frac{1}{128}(& 1 & 8 & \mathbf{142} & 8 & 1 \quad) \\ & \widetilde{p} & = & & & \frac{1}{32}(& \mathbf{35} & 4 & 1 \quad) \end{array}$$

$\widetilde{p}$ is derived from p according to figure 2.4:

$$\begin{aligned} r^2 &= \frac{1}{4}(z^{-2} + 2 + z^2) & p(z) &= \frac{1}{128}(z^{-2} + 8z^{-1} + 142 + 8z^1 + z^{-2}) \\ r &= \frac{1}{2}(z^{-1} + z^1) & p(z) &= \frac{1}{32}r^2 + \frac{1}{128}(8z^{-1} + 140 + 8z^1) \\ & & p(z) &= \frac{1}{32}(r^2 + 4r) + \frac{140}{128} \\ & & \widetilde{p}(r) &= \frac{1}{32}(r^2 + 4r + 35) \end{aligned}$$

We see that the coefficient of the square term of $\widetilde{p}(r) = \frac{1}{32}(35 + 4r + r^2)$ is positive, that is why p is convex, which means that its maximum is at the borders of $[-1, 1]$.

$$\widetilde{p}_{\max} = \max\{p(-1), p(1)\} = \max\left\{1, \frac{5}{4}\right\} = \frac{5}{4}$$

$$\widetilde{p}_{\max} = p_{\max} \Rightarrow$$

$$(2.1.14) \Rightarrow \qquad \lambda_{\text{max/min}} = \frac{5}{4} \pm \sqrt{\left(\frac{5}{4}\right)^2 - 1} = \frac{5}{4} \pm \frac{3}{4}$$

$$\lambda_{\max} = 2 \qquad \|F\|_2 = \sqrt{2}$$
$$\lambda_{\min} = \frac{1}{2} \qquad \left\|F^{-1}\right\|_2^{-1} = \frac{1}{\sqrt{2}}$$

Which means that
$$\forall x: \quad \frac{1}{\sqrt{2}} \|x\|_2 \leq \|Fx\|_2 \leq \sqrt{2}\,\|x\|_2$$

We are also interested in the signals, where these bounds are reached. We follow the solution sketched in section 2.1.3.

p is maximal for $r = 1 \Leftrightarrow z = 1$ so we calculate b, c, d for this z according to (2.1.7):

$$\begin{aligned} b &= \overline{h_o(1)}h_e(1) + \overline{g_o(1)}g_e(1) = -\frac{3}{4} \\ c &= \overline{h_o(1)}h_o(1) + \overline{g_o(1)}g_o(1) = \frac{5}{4} \\ d &\in \left\{\frac{1}{2}, 2\right\} - \frac{5}{4} = \left\{-\frac{3}{4}, \frac{3}{4}\right\} \end{aligned}$$

We scale this by $-\frac{4}{3}$ for simplification and obtain the two signals:

$$\begin{array}{lcccccccl} (\ldots, & -1, & 1, & \mathbf{-1}, & 1, & -1, & 1, & \ldots) & \text{hits the upper bound} \\ (\ldots, & 1, & 1, & \mathbf{1}, & 1, & 1, & 1, & \ldots) & \text{hits the lower bound} \end{array}$$

2.2 Symmetrical wavelets with close norm bounds

We will continue on concentrating on wavelets with two filters.

In section 2 we have discussed, why it is important to have close lower and upper norm bounds. In the case of unitary wavelets the bounds are equal, which is obviously the optimum. Another common restriction is the request for symmetrical wavelets, because they have good visual qualities. Unfortunately it is not possible to find symmetrical unitary wavelets other than the lazy wavelet, as shown in section 2.1.4. Nevertheless, we will try to construct symmetrical wavelets that have bounds that are as close as possible in a certain sense.

Similar approaches were already explored:

1. In section 8.1.1 of [3] orthogonal wavelets are constructed which are almost symmetrical. Since symmetry is equivalent to linear phase behaviour, it is tried there to minimize the deviation from linear phase behaviour of the filters.
2. In section 8.3.5 of [3] (also in section 6.C.1 of [2]) an example of a symmetric wavelet is given, that is almost orthogonal. It is based on the Laplacian filter.

We will follow another approach, that utilizes the knowledge of how to determine the norm bounds for a single wavelet transformation step.

2.2.1 Optimization using CHEBYSHEV polynomials

To prevent an optimization approach from resulting in the lazy wavelet we have to set restrictions additional to the symmetry. A possible choice is to fix the filter length. If we work with the filter

$$f = (f_m, f_{m+1}, \ldots, f_{n-1}, f_n)$$

the number of taps of f might be less than $n-m$ although f has $n-m+1$ coefficients. $|f| < n-m$ means $f_m = 0$ or $f_n = 0$. Thus we must assure $f_m \neq 0$ and $f_n \neq 0$. But since the maximum norm of p, which is the measurement for close norm bounds, depends continuously on the filter coefficients, it makes no sense to exclude the set of points with $f_m = 0$ or $f_n = 0$ which has no interior. That is the reason why it seems to be better to fix f_m and f_n on some suitable values.

Fixing the outermost coefficients, then searching for wavelets with some inner coefficients that minimize the norm bound difference will safely exclude the lazy wavelet. But when constructing new wavelets, the desired characteristics like interpolating behaviour (see below) additional to close norm bounds may not be bounded to specific values for the outermost coefficients. In other words, we are trying to construct wavelets that have naturally close norm bounds (like unitary wavelets), but it is not sure that wavelets chosen from this family under certain further aspects A have optimal norm bounds in the sense, that from all wavelets that fulfill A the ones we are about to construct here will have closest possible norm bounds!

We see that the outermost coefficients of p depend on the outermost coefficients of the longer filter out of h and g, or on both of them, if the filters have equal size. Thus fixing the outermost coefficients of h and g leads to fixing the outermost coefficient of p. Now, we ask for values of the other coefficients of p that lead to a minimal maximum norm of p. Polynomials that have a minimal global maximum for a fixed leading coefficient are known as CHEBYSHEV polynomials [1, 6, 19], which are introduced in the context of real numbers, usually. Fortunately, CHEBYSHEV polynomials have a much simpler representation in z, that can be obtained if we do the reverse argument transformation (i.e. from r to z) to the one derived in section 2.1.5. The polynomial in z corresponding to the n. CHEBYSHEV polynomial is

$$t_n(z) = \tfrac{1}{2}(z^{-n} + z^n)$$

As you can easily see it has exactly n maxima each with a value of 1.

The proof that CHEBYSHEV polynomials are those with minimal global maximum depending on the leading coefficient is usually done in two steps: First it is shown that all local minima and maxima have the same magnitude. The second step is to show, that any other polynomial of the same degree and leading coefficient must have at least the global maximum as the corresponding CHEBYSHEV polynomial. Refer to [6, 19] for the detailed proof.

Now we would like to know, if the requirements for p (e.g. symmetry and $p^2 \geq \overline{q}q$) allow us to set p to a function basing on a CHEBYSHEV polynomial.

We have to preserve that $p^2 \geq \overline{q}q$ because it assures that λ is always real. We can set p to t_n scaled by a real coefficient and add a sufficient big offset. We can also allow t_n to be rotated on the complex unit circle. This can be described with a generalized form

$$t_n(\mu, z) := \tfrac{1}{2}(\mu z^{-n} + \overline{\mu} z^n)$$

$|\mu|$ is the scaling factor and $\arg(\mu)$ the rotation angle. Now let

$$\begin{aligned} p(z) \quad &:= \quad t_n(\mu, z) + |\mu| + 1 \qquad \Big| \qquad |t_n(\mu, z)| \leq |\mu| \\ &\geq \quad 1 = q(z) \end{aligned} \tag{2.2.1}$$

The balance of the maxima remains, of course.

It is still not clear if any filter pair exists, that leads to polynomials p of this form, but in the next section we will derive an example set. Maybe such wavelets got another name in the wavelet literature already, but here wavelets with $p(z) = \frac{1}{2}(\mu z^{-n} + \overline{\mu} z^n) + |\mu| + 1$ will be called CHEBYSHEV *wavelets*.

Without knowing any filter pair that is associated with a p of the required form, we can already determine the bounds of CHEBYSHEV wavelets:

It is $p(z) \leq 2|\mu| + 1$ with equality e.g. for $|\mu| \cdot z^n = \mu$. $\Rightarrow$

$$
\begin{aligned}
(2.1.15) \Rightarrow \qquad \|F\|_2^2 = \lambda_{\max} &= p_{\max} + \sqrt{p_{\max}^2 - \overline{q}q} \\
&= 2|\mu| + 1 + \sqrt{(2|\mu| + 1)^2 - 1} \\
&= 2|\mu| + 1 + \sqrt{4|\mu|^2 + 4|\mu|} \\
&= 2|\mu| + 1 + 2\sqrt{|\mu|(|\mu| + 1)} \\
&= \left(\sqrt{|\mu| + 1} + \sqrt{|\mu|}\right)^2 \\
\|F\|_2 &= \sqrt{|\mu| + 1} + \sqrt{|\mu|} \\
\left\|F^{-1}\right\|_2^{-2} = \lambda_{\min} &= p_{\max} - \sqrt{p_{\max}^2 - \overline{q}q} \\
&= 2|\mu| + 1 - 2\sqrt{|\mu|(|\mu| + 1)} \\
\left\|F^{-1}\right\|_2^{-1} &= \sqrt{|\mu| + 1} - \sqrt{|\mu|}
\end{aligned}
$$

2.2.2 Special case: Filter length 5 and 3

As an exercise we will construct CHEBYSHEV wavelets for a pair of filters with lengths 5 and 3 like the CDF-2,2 wavelet has. For more generality we will start with complex filter coefficients. Note that h_0 and g_0 are real nevertheless, because of the symmetry of h and g.

The coefficient identifiers are chosen in a way that emphasizes the symmetry, but it does not reflect the indices of the vector components. The vector component indices are given by the stressed component which has index zero, as supplied before.

$$
\begin{array}{rcccccccc}
h & := & (& h_2 & h_1 & \boldsymbol{h_0} & \overline{h_1} & \overline{h_2} &) \\
g & := & (& & & \boldsymbol{g_1} & g_0 & \overline{g_1} &)
\end{array}
$$

$$
\begin{array}{rccccccrcccc}
h_e & := & (& h_2 & \boldsymbol{h_0} & \overline{h_2} &) & h_o & := & (& h_1 & \boldsymbol{\overline{h_1}} &) \\
g_e & := & (& & \boldsymbol{g_1} & \overline{g_1} &) & g_o & := & (& & \boldsymbol{g_0} &)
\end{array}
$$

$$
\begin{array}{rcccccccc}
\overline{h_e}h_e & = & (& h_2^2 & 2h_2h_0 & \boldsymbol{h_0^2 + 2\overline{h_2}h_2} & 2\overline{h_2}h_0 & \overline{h_2}^2 &) \\
\overline{g_e}g_e & = & (& & g_1^2 & \boldsymbol{2\overline{g_1}g_1} & \overline{g_1}^2 & &) \\
\overline{h_o}h_o & = & (& & h_1^2 & \boldsymbol{2\overline{h_1}h_1} & \overline{h_1}^2 & &) \\
\overline{g_o}g_o & = & (& & & \boldsymbol{g_0^2} & & &) \\
h_e g_o & = & (& & h_2 g_0 & \boldsymbol{h_0 g_0} & \overline{h_2} g_0 & &) \\
h_o g_e & = & (& & h_1 g_1 & \boldsymbol{\overline{h_1}g_1 + h_1\overline{g_1}} & \overline{h_1 g_1} & &)
\end{array}
$$

p and q are computed as defined in (2.1.12) and (2.1.13):

$$
2p = (h_2^2, 2h_2h_0 + h_1^2 + g_1^2, \boldsymbol{h_0^2 + 2\overline{h_1}h_1 + 2\overline{h_2}h_2 + g_0^2 + 2\overline{g_1}g_1}, 2\overline{h_2}h_0 + \overline{h_1}^2 + \overline{g_1}^2, \overline{h_2}^2) \tag{2.2.2}
$$

$$
q = (h_2g_0 - h_1g_1, \boldsymbol{h_0g_0 - (\overline{h_1}g_1 + h_1\overline{g_1})}, \overline{h_2}g_0 - \overline{h_1g_1}) \tag{2.2.3}
$$

Determine filter coefficients: Now we identify p with the appropriate CHEBYSHEV polynomial as defined in (2.2.1). We abbreviate the CHEBYSHEV polynomial $t_2(\mu, \circ)$ as $t_2(\mu)$:

$$\begin{aligned} 2p &= 2 \cdot t_2(\mu) + 2 \cdot (0, 0, |\boldsymbol{\mu}| + \mathbf{1}, 0, 0) \\ &= (\mu, 0, \mathbf{2}\,|\boldsymbol{\mu}| + \mathbf{2}, 0, \overline{\mu}) \end{aligned} \tag{2.2.4}$$

Comparison of the coefficients of p in (2.2.2) and (2.2.4):

$$\begin{aligned} p_{-2}: &\quad \mu = h_2^2 \\ &\quad |\mu| = \overline{h_2} h_2 \end{aligned} \tag{2.2.5}$$

$$\begin{aligned} p_{-1}: &\quad 0 = 2h_2 h_0 + h_1^2 + g_1^2 \\ p_0: &\quad 2\,|\mu| + 2 = h_0^2 + 2\overline{h_1} h_1 + 2\overline{h_2} h_2 + g_0^2 + 2\overline{g_1} g_1 \\ &\quad 2 = h_0^2 + 2\overline{h_1} h_1 + g_0^2 + 2\overline{g_1} g_1 \end{aligned} \tag{2.2.6}$$

p_1 and p_2 need not to be considered, since all polynomials involved here are symmetrical.

Remember that q is normalized to 1:

$$q = (0, \mathbf{1}, 0) \tag{2.2.7}$$

Comparison of the coefficients of q in (2.2.3) and (2.2.7):

$$\begin{aligned} q_{-1}: &\quad 0 = h_2 g_0 - h_1 g_1 \\ q_0: &\quad 1 = h_0 g_0 - (\overline{h_1} g_1 + h_1 \overline{g_1}) \end{aligned} \tag{2.2.8}$$

Multiply (2.2.8) with 2 and subtract it from (2.2.6). We respect that $\overline{h_e} = h_e \Rightarrow h_0 \in \mathbb{R} \ \wedge \ \overline{g_o} = g_o \Rightarrow g_0 \in \mathbb{R}$ and obtain:

$$\begin{aligned} 0 &= p_0 - 2\,|\mu| - 2q_0 \\ &= h_0^2 + 2\overline{h_1} h_1 + g_0^2 + 2\overline{g_1} g_1 - 2h_0 g_0 + 2\overline{h_1} g_1 + 2h_1 \overline{g_1} \\ &= \underbrace{(h_0 - g_0)^2}_{\in\mathbb{R},\geq 0} + 2\underbrace{\overline{(h_1 + g_1)}(h_1 + g_1)}_{\in\mathbb{R},\geq 0} \end{aligned}$$

$\Rightarrow$

$$\begin{aligned} 0 &= h_0 - g_0 \\ 0 &= h_1 + g_1 \\ h_0 &= g_0 && (2.2.9) \\ h_1 &= -g_1 && (2.2.10) \end{aligned}$$

We establish that CHEBYSHEV wavelet filters of the lengths (5,3) have an obvious structure: Except of the outermost coefficient, the corresponding coefficients of the filters h and g have the same or an alternating value.

Using this, the comparison of q's coefficients can be simplified to:

$$0 = h_0 h_2 + h_1^2 \tag{2.2.11}$$

$$1 = h_0^2 + 2\overline{h_1} h_1 \tag{2.2.12}$$

We note that the same equations could be obtained by simplifying the comparison of p's coefficients.

Further simplifications can be made if we restrict $\{h_1, h_2, \mu\} \subset \mathbb{R}$. Without it there may be more or even an infinite number of complex solutions.

$$\begin{aligned}
(2.2.11) \Rightarrow h_1^2 &= -h_0 h_2 \\
(2.2.12) \Rightarrow 1 &= h_0^2 + 2h_1^2 \\
&= h_0^2 - 2h_0 h_2 \\
0 &= h_0^2 - 2h_0 h_2 - 1 \\
h_0 &= \underline{\underline{h_2 \pm \sqrt{h_2^2 + 1}}} \\
h_{0+} &:= h_2 + \sqrt{h_2^2 + 1} > 0 \\
h_{0-} &:= h_2 - \sqrt{h_2^2 + 1} < 0
\end{aligned} \tag{2.2.13}$$

$$\begin{aligned}
(2.2.11) \Rightarrow h_1^2 &= -h_0 h_2 \\
h_1 &= \pm\sqrt{-h_0 h_2} \\
h_1 &= \underline{\underline{\begin{cases} \pm\sqrt{-h_{0-} \cdot h_2} & : h_2 \geq 0 \\ \pm\sqrt{-h_{0+} \cdot h_2} & : h_2 < 0 \end{cases}}}
\end{aligned}$$

Now every coefficient is determined. We see that for any given real h_2 four CHEBYSHEV-(5,3) wavelets with real coefficients exist. Thus we constructed a whole family of wavelets which depends mainly on one parameter.

For $h_2 = 0$ we obtain the lazy wavelet, which is the only orthogonal wavelet in this family.

$$\begin{aligned}
\|F\|_2 &= \sqrt{\mu + 1} + \sqrt{\mu} \\
&= \sqrt{h_2^2 + 1} + |h_2| \\
&= \begin{cases} h_{0+} & : h_2 \geq 0 \\ -h_{0-} & : h_2 < 0 \end{cases} \\
&= \max\{h_{0+}, -h_{0-}\} \\
\left\|F^{-1}\right\|_2^{-1} &= \sqrt{\mu + 1} - \sqrt{\mu} \\
&= \sqrt{h_2^2 + 1} - |h_2| \\
&= \begin{cases} -h_{0-} & : h_2 \geq 0 \\ h_{0+} & : h_2 < 0 \end{cases} \\
&= \min\{h_{0+}, -h_{0-}\}
\end{aligned}$$

The signal where the norm bounds are touched: What does the signal look like, where norm bounds are reached? We calculate the eigenvector of $\overline{P}^T P$ as explained in section 2.1.3 using the variables a, b, c, defined in (2.1.7).

$$\begin{aligned}
a &= (h_2^2, 2h_0h_2 + g_1^2, \boldsymbol{h_0^2 + 2\overline{h_2}h_2 + 2\overline{g_1}g_1}, 2h_0\overline{h_2} + \overline{g_1}^2, \overline{h_2}^2) \\
b &= (\boldsymbol{h_1}, \overline{h_1}) * (h_2, \boldsymbol{h_0}, \overline{h_2}) + (\boldsymbol{g_0}) * (\boldsymbol{g_1}, \overline{g_1}) \\
&= (h_1h_2, \boldsymbol{\overline{h_1}h_2 + h_1h_0}, \overline{h_1}h_0 + h_1\overline{h_2}, \overline{h_1}\overline{h_2}) + (\boldsymbol{g_0g_1}, g_0\overline{g_1}) \\
&= (h_1h_2, \boldsymbol{h_0h_1 + \overline{h_1}h_2 + g_0g_1}, h_0\overline{h_1} + h_1\overline{h_2} + g_0\overline{g_1}, \overline{h_1h_2}) \\
c &= (h_1^2, \boldsymbol{2\overline{h_1}h_1 + g_0^2}, \overline{h_1}^2)
\end{aligned}$$

$h_0 \overset{(2.2.9)}{=} g_0 \ \wedge \ h_1 \overset{(2.2.10)}{=} -g_1 \Rightarrow$

$$\begin{aligned}
a &= (h_2^2, 2h_0h_2 + h_1^2, \boldsymbol{h_0^2 + 2\overline{h_2}h_2 + 2\overline{h_1}h_1}, 2h_0\overline{h_2} + \overline{h_1}^2, \overline{h_2}^2) \\
b &= (h_1h_2, \boldsymbol{\overline{h_1}h_2}, h_1\overline{h_2}, \overline{h_1h_2}) \\
c &= (h_1^2, \boldsymbol{h_0^2 + 2\overline{h_1}h_1}, \overline{h_1}^2)
\end{aligned}$$

(2.2.11) $\wedge$ (2.2.12) $\Rightarrow$

$$\begin{aligned}
a &= (h_2^2, -h_1^2, \boldsymbol{1 + 2\overline{h_2}h_2}, -\overline{h_1}^2, \overline{h_2}^2) \\
c &= (h_1^2, \boldsymbol{1}, \overline{h_1}^2) \\
a - c &= (h_2^2, -2h_1^2, \boldsymbol{2\overline{h_2}h_2}, -2\overline{h_1}^2, \overline{h_2}^2)
\end{aligned}$$

For those z where p is maximal (if $\mu \neq 0$ then it is maximal for $z^{-2} = \frac{\mu}{|\mu|} \overset{(2.2.5)}{=} \frac{h_2}{\overline{h_2}}$) this results in

$$\begin{aligned}
b &= 2h_1\overline{h_2} + 2\overline{h_1h_2}z^{-1} \qquad (\overline{h_2} = h_2z^2) \\
&= 2\overline{h_2}(h_1 + \overline{h_1}z^{-1}) \\
\bar{b}b &= 4\overline{h_2}h_2(h_1^2z + \overline{h_1}^2z^{-1} + 2\overline{h_1}h_1) \\
a - c &= 4\overline{h_2}h_2 - 2(h_1^2z + \overline{h_1}^2z^{-1}) \\
(a-c)^2 + 4\bar{b}b &= 4\left((2\overline{h_2}h_2 - (h_1^2z + \overline{h_1}^2z^{-1}))^2 + 4\overline{h_2}h_2(h_1^2z + \overline{h_1}^2z^{-1} + 2\overline{h_1}h_1)\right) \\
&= 4\left((2\overline{h_2}h_2)^2 + (h_1^2z + \overline{h_1}^2z^{-1})^2 + 8\overline{h_2}h_2\overline{h_1}h_1\right)
\end{aligned}$$

Further simplifications can be done, if we switch back to real h_1, h_2 again and let $z = 1$:

$$\begin{aligned}
(a-c)^2 + 4b^2 &= 16(h_2^2 - h_1^2)^2 + 64h_1^2h_2^2 \\
&= 16(h_2^2 + h_1^2)^2 \\
\lambda - c = \tfrac{1}{2}\left(a - c \pm \sqrt{(a-c)^2 + 4b^2}\right) &= 2h_2^2 - 2h_1^2 \pm 2(h_2^2 + h_1^2)
\end{aligned}$$

We obtain two eigenvectors

$\left(4h_2^2, 4h_1h_2\right)^T \sim (h_2, h_1)^T$ for the upper bound

$\left(-4h_1^2, 4h_1h_2\right)^T \sim (-h_1, h_2)^T$ for the lower bound

A lifting step factorization can be made in a general form, too. Please refer to section 2.3 for a generic factorization of symmetric wavelet filter pairs of the length (5,3).

2.2.3 CDF-2,2 counterpart

Since the linear interpolation as prediction of the H band seems to be a good choice for many images and we want to combine this with close norm bounds, we will try to construct a symmetric linear interpolating CHEBYSHEV wavelet.

Linear interpolation is expressed by $g = c \cdot (\mathbf{-1}, 2, -1)$ or equally $g_0 = -2g_1$ (figure 2.5). This in turn is equivalent to $h_0 = 2h_1$ because of the simplified structure we observed in (2.2.9) and (2.2.10).

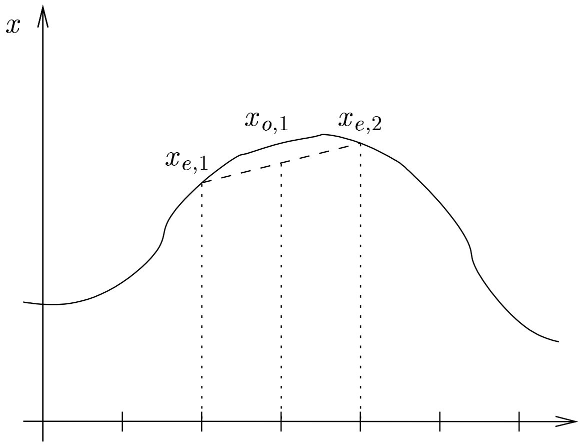


Figure 2.5: How one can predict the values on the H band (odd indexed signal values) by linear interpolation in the L band (even indexed signal values) – Linear interpolation between $x_{e,1}$ and $x_{e,2}$ for the position $o, 1$ means calculating $\frac{1}{2}(x_{e,1} + x_{e,2})$. If the prediction is good it holds $x_{o,1} \approx \frac{1}{2}(x_{e,1} + x_{e,2})$ and $x_{o,1} - \frac{1}{2}(x_{e,1} + x_{e,2})$ will be quite small. The filter corresponding to this term is $(-\frac{1}{2}, 1, -\frac{1}{2})$.

$$
\begin{aligned}
(2.2.13) \Rightarrow \qquad 6h_1^2 &= 1 \\
h_1 &= \sqrt{\frac{1}{6}} \\
h_0 &= \sqrt{\frac{2}{3}} \\
(2.2.11) \Rightarrow \qquad h_2 &= -\sqrt{\frac{1}{24}}
\end{aligned}
$$

$$
\begin{array}{rclccccccc}
h & := & \sqrt{\frac{1}{24}} & (& -1 & 2 & \mathbf{4} & 2 & -1 &) \\
g & := & \sqrt{\frac{1}{24}} & (& & & \mathbf{-2} & 4 & -2 &)
\end{array}
$$

We immediately get the signals

$$\begin{array}{llllllll l l} (\ldots & -1, & 2, & \mathbf{-1}, & 2, & -1, & 2, & \ldots) & \text{for the upper bound} & \sqrt{\frac{3}{2}} \\ (\ldots & 2, & 1, & \mathbf{2}, & 1, & 2, & 1, & \ldots) & \text{for the lower bound} & \sqrt{\frac{2}{3}} \end{array}$$

Precaution: With this construction we obtained wavelet filter pairs that have as close as possible norm bounds dependent on the outermost coefficient of h. This was implicitly done by selecting CHEBYSHEV polynomials for p. Then we have chosen this coefficient to get linear interpolation by g. The more natural question is to ask for all filters g with linear interpolation (i.e. $c \cdot (\mathbf{-1}, 2, -1)$ for any $c \neq 0$) and to ask for the filter pair of this type which has closest norm bounds. Both approaches are different, the answers are different as well. Later in section 2.3 we will construct wavelet filter pairs with the same lengths (5,3) which give the answer to the second question.

Practical results achieved with the CHEBYSHEV wavelet constructed here are presented in section 3.2.1, where it is compared to weighted variants of the CDF-2,2 wavelet.

2.2.4 Generalization to other filter lengths

Let us explore what CHEBYSHEV wavelets look like if extended to other lengths. When deriving the connections between the coefficients of CHEBYSHEV wavelet filters, we found the interesting dependencies (2.2.9), (2.2.10) between the coefficients of h and g by using the fact, that 0 is the only complex number with magnitude 0. Is it possible to extend this trick to other filter lengths?

First we have to make some observations about the filter lengths of reversible (bi-orthogonal) symmetric wavelets as in (2.1.17).

$$\begin{array}{rcl} 1 & = & \det P \\ & = & h_e g_o - h_o g_e \\ h_e g_o - 1 & = & h_o g_e \\ |h_e g_o - 1| & = & |h_o g_e| \\ h_e g_o \text{ is symmetric} \Rightarrow |h_e g_o| & = & |h_e g_o - 1| \ \vee \ h_e g_o = 1 \\ & & h_e g_o = 1 \text{ is not very interesting,} \\ & & \text{we will continue to consider } h_e g_o \neq 1 \\ |h_e g_o| & = & |h_o g_e| \\ |h_e| + |g_o| & = & |h_o| + |g_e| \end{array}$$

Because of the symmetry it is

$$2 \mid |h_e| \ \wedge \ 2 \mid |g_o| \ \wedge \ 2 \nmid |h_o| \ \wedge \ 2 \nmid |g_e| \tag{2.2.14}$$

We note that there is always one of the filters h_e, h_o, g_e, g_o that is strictly longer than the others, in other words: There can not be two longest filters.

Proof. The proof is done indirectly:

Assumed that two filters have the same length and no other filter is longer, due to the parity (2.2.14) it can only be $|h_e| = |g_o|$ or $|h_o| = |g_e|$. Without loss of generality we assume $|h_e| = |g_o|$. Because of the parity of the filter lengths it is $|h_e| \neq |h_o| \;\wedge\; |h_e| \neq |g_e|$ and according to our assumption that h_e is the longest filter, this means $|h_e| > |h_o| \;\wedge\; |h_e| > |g_e|$. This leads to $|h_e| + |g_o| > |h_o| + |g_e|$ and is a contradiction to the equality which we have established at the beginning. □

Let f be the longest filter out of $\{h_e, h_o, \overline{g_e}, g_o\}$. Note that in case g_e is the longest filter, we will set f to the mirrored g_e in order to better fit into the scheme. Let n be the highest index and m the lowest index of f. Since any other filter must be shorter at least by one coefficient, n and m do also bound the indices of all the other filters. The following four cases are possible, the figure shows all filters at their maximum possible length compared to the longest filter:

$$
\begin{array}{ll ccccccc ll}
 & & m & & & 0 & & & n & & \\
 & & & m' & & & & n' & & & \\
 & & m'' & & & & & n'' & & & \\
h_e & (& * & * & * & \star & * & * & * &) & \widehat{=} f \\
h_o & (& * & * & * & \star & * & * & &) & \\
g_e & (& & * & * & \star & * & * & * &) & \\
g_o & (& & * & * & \star & * & * & &) &
\end{array}
\qquad
\begin{array}{ll cccccc ll}
 & & m & & & 0 & & n & & \\
 & & & m' & & & & n' & & \\
 & & & m'' & & & n'' & & & \\
h_e & (& & * & * & \star & * & * &) & \\
h_o & (& * & * & * & \star & * & * &) & \widehat{=} f \\
g_e & (& & & * & \star & * & * &) & \\
g_o & (& & * & * & \star & * & * &) &
\end{array}
$$

$$
\begin{array}{ll ccccccc ll}
 & & m & & & 0 & & n & & & \\
 & & & m' & & & & n' & & & \\
 & & & m'' & & & n'' & & & & \\
h_e & (& & * & * & \star & * & * & &) & \\
h_o & (& & * & * & \star & * & & &) & \\
g_e & (& & * & * & \star & * & * & * &) & \widehat{=} f \\
g_o & (& & * & * & \star & * & * & &) &
\end{array}
\qquad
\begin{array}{ll ccccccc ll}
 & & m & & & 0 & & & n & & \\
 & & & m' & & & & n' & & & \\
 & & m'' & & & & & n'' & & & \\
h_e & (& & * & * & \star & * & * & &) & \\
h_o & (& * & * & * & \star & * & * & &) & \\
g_e & (& & * & * & \star & * & * & * &) & \\
g_o & (& * & * & * & \star & * & * & * &) & \widehat{=} f
\end{array}
$$

We observe that it is always $n' = -m'$ and $n'' = 1 - m''$.

We repeat the strategy that was successful in section 2.2.2 and look at the absolute coefficient of p first:

$$2p_0 = \sum_j \overline{h_e}_{-j} h_{ej} + \overline{h_o}_{-j} h_{oj} + \overline{g_e}_{-j} g_{ej} + \overline{g_o}_{-j} g_{oj}$$

Apply the definition of conjugated filters (1.2.1)

$$
\begin{aligned}
&= \sum_j \overline{h_{e,j}} h_{e,j} + \overline{h_{o,j}} h_{o,j} + \overline{g_{e,j}} g_{e,j} + \overline{g_{o,j}} g_{o,j} \\
2\,|\mu| &\stackrel{!}{=} \overline{f_n} f_n + \overline{f_m} f_m = 2\overline{f_n} f_n \qquad\qquad \text{because of symmetry } f_n = \overline{f_m} \\
2 + 2 = 2p_0\,|\mu| &= 2\,|\mu| + \sum_{j=m'}^{n'} \overline{h_{e,j}} h_{e,j} + \sum_{j=-m'}^{-n'} \overline{g_{o,j}} g_{o,j} + \sum_{j=m''}^{n''} \overline{h_{o,j}} h_{o,j} + \sum_{j=-n''}^{-m''} \overline{g_{e,j}} g_{e,j} \\
&= 2\,|\mu| + \sum_{j=m'}^{n'} \left(\overline{h_{e,j}} h_{e,j} + \overline{g_{o,-j}} g_{o,-j}\right) + \sum_{j=m''}^{n''} \left(\overline{h_{o,j}} h_{o,j} + \overline{g_{e,-j}} g_{e,-j}\right)
\end{aligned}
\tag{2.2.15}
$$

$$\begin{aligned}1 = q_0 &= \sum_j h_{e,j} g_{o,-j} - h_{o,j} g_{e,-j} \\ &= \sum_{j=m'}^{n'} h_{e,j} g_{o,-j} - \sum_{j=m''}^{n''} h_{o,j} g_{e,-j} \end{aligned} \tag{2.2.16}$$

Now we can build a complete square from p_0 and q_0:

$$\begin{aligned} 0 &= 2p_0 - 2|\mu| - (\overline{q_0} + q_0) \\ &= \sum_{j=m'}^{n'} \underbrace{(\overline{h_{e,j}} - g_{o,-j})(h_{e,j} - \overline{g_{o,-j}})}_{\in\mathbb{R},\geq 0} + \sum_{j=m''}^{n''} \underbrace{(\overline{h_{o,j}} + g_{e,-j})(h_{o,j} + \overline{g_{e,-j}})}_{\in\mathbb{R},\geq 0} \end{aligned}$$

$\Rightarrow$

$$\begin{aligned} \forall j \in \{m', \dots, n'\} : 0 &= (\overline{h_{e,j}} - g_{o,-j})(h_{e,j} - \overline{g_{o,-j}}) \\ h_{e,j} &= \overline{g_{o,-j}} = \overline{g_o}_j \\ \forall j \in \{m'', \dots, n''\} : 0 &= (\overline{h_{o,j}} + g_{e,-j})(h_{o,j} + \overline{g_{e,-j}}) \\ h_{o,j} &= -\overline{g_{e,-j}} = -\overline{g_e}_j \end{aligned}$$

E.g. in the first case where the longest filter is $f = h_e$ that means that

$$h_e(z) = \overline{g_o(z)} + h_{e,m} \cdot z^m + h_{e,n} \cdot z^n \tag{2.2.17}$$

$$h_o(z) = -\overline{g_e(z)} \tag{2.2.18}$$

Although it is $m = -n$ here, the different variable names were left for easier carrying the results to the remaining cases, which are similar.

We see that (2.2.17) and (2.2.18) form a necessary condition for a wavelet to be a CHEBYSHEV wavelet.

If we carefully think over the insertion and replacement steps that were made in section 2.2.2, we note that after finding out the relations (2.2.9), (2.2.10), the coefficient comparisons of p and q coincided. The equations (2.2.9), (2.2.10) could be carried to other lengths – is this true for this observation, too?

The answer is yes, and for the case $f = h_e$ we can show exemplarily, that if (2.2.17) and (2.2.18) are true then the conditions

1. $p(z) = t(\mu, z) + |\mu| + 1$ with $\mu = 2p_{n-m} = h_{e,n}^2$

2. $q(z) = 1$

are equivalent.

$$
\begin{aligned}
& 2 \cdot (t(\mu, z) + |\mu| + 1) = \\
2p(z) \quad &= \quad \overline{h_e(z)}h_e(z) + \overline{h_o(z)}h_o(z) + \overline{g_e(z)}g_e(z) + \overline{g_o(z)}g_o(z) \\
&\overset{(2.2.18)}{=} \quad \overline{h_e(z)}h_e(z) + \overline{g_o(z)}g_o(z) - 2h_o(z)g_e(z) \\
&\overset{(2.2.17)}{=} \quad \overline{h_{e,n}}h_{e,n} + \overline{h_{e,m}}h_{e,m} + \overline{h_{e,n}}h_{e,m} \cdot z^{m-n} + \overline{h_{e,m}}h_{e,n} \cdot z^{n-m} + \\
& \qquad g_o(z)(h_{e,m} \cdot z^m + h_{e,n} \cdot z^n) + \\
& \qquad \overline{g_o(z)}(\overline{h_{e,m}} \cdot z^{-m} + \overline{h_{e,n}} \cdot z^{-n}) + \\
& \qquad 2\overline{g_o(z)}g_o(z) - 2h_o(z)g_e(z) \qquad\qquad | - 2 \cdot (t(\mu, z) + |\mu|)
\end{aligned}
$$

Note that because of the symmetry of h_e it is

1. $h_{e,n} = \overline{h_{e,m}}$
2. $\mu = h_{e,n}^2 = \overline{h_{e,m}}^2 = \overline{h_{e,m}}h_{e,n}$
3. $|\mu| = \overline{h_{e,n}}h_{e,n} = \overline{h_{e,m}}h_{e,m}$

$$
\begin{aligned}
2 = \; & g_o(z)(h_{e,m} \cdot z^m + h_{e,n} \cdot z^n) + \\
& \overline{g_o(z)}(\overline{h_{e,m}} \cdot z^{-m} + \overline{h_{e,n}} \cdot z^{-n}) + \\
& 2\overline{g_o(z)}g_o(z) - 2h_o(z)g_e(z)
\end{aligned}
$$

apply $g_o(z) = \overline{g_o(z)} \;\wedge\; \overline{h_{e,m}} \cdot z^{-m} = h_{e,n} \cdot z^n \;\wedge\; \overline{h_{e,n}} \cdot z^{-n} = h_{e,m} \cdot z^m$

$$
\begin{aligned}
&= 2g_o(z)(h_{e,m} \cdot z^m + h_{e,n} \cdot z^n) + 2g_o(z)^2 - 2h_o(z)g_e(z) \\
&= 2(h_e(z)g_o(z) - h_o(z)g_e(z)) \\
&= 2q(z)
\end{aligned}
$$

Finally we obtain, that if the wavelet filters h and g share the same coefficients according to (2.2.17) and (2.2.18), we have to preserve only that $h_e g_o - h_o g_e = q = 1$ and it follows immediately, that p has our special form of a CHEBYSHEV polynomial $t(\mu, z) + |\mu| + 1$.

2.3 Linear interpolation

In section 2.2.3 we have tried to find (5,3)-wavelets with both linear interpolation and close norm bounds. The CHEBYSHEV wavelets are a general approach to combine filter symmetry with close norm bounds. So, was the selection of linear interpolating wavelets out of the class of CHEBYSHEV wavelets optimal with respect to the bounds? We will investigate what the closest bounds with linear interpolating (5,3)-wavelets with real coefficients possibly are.

To be sure that we construct invertible wavelets, we will construct the lifting factorization of the requested wavelet in the next paragraph. We will see that the lifting steps depend only on two parameters if we restrict to a linear interpolating filter g. In the subsequent paragraph we will determine values for these parameters that make the norm bounds of the wavelet as close as possible.

General lifting factorization: Fortunately the lifting factorization (see [4] and figure 3.3) of any invertible symmetrical (5,3)-wavelet can be made with the same steps, differing only in the choice of two parameters.

From the invertibility of the wavelet follows:

$$\begin{aligned} 1 &= q \\ (0, \mathbf{1}, 0) &= h_e g_o - h_o g_e \\ &= (h_2 g_0 - h_1 g_1, \boldsymbol{h_0 g_0 - (\overline{h_1} g_1 + h_1 \overline{g_1})}, \overline{h_2} g_0 - \overline{h_1 g_1}) \end{aligned}$$

comparison of the components results in

$$0 = h_2 g_0 - h_1 g_1 \tag{2.3.1}$$

$$1 = h_0 g_0 - (\overline{h_1} g_1 + h_1 \overline{g_1}) \tag{2.3.2}$$

The factorization starts with the last lifting step. Its lifting filter t is chosen in a way that the outermost coefficients of h vanish if we reverse the lifting step. Let $t = (\frac{h_2}{g_1}, \boldsymbol{\frac{\overline{h_2}}{\overline{g_1}}})$ then we can reduce h by g with help of $t(\circ^2)$ and obtain the filters

$$\begin{aligned} h' &:= h - \left(\frac{h_2}{g_1}, 0, \boldsymbol{\frac{\overline{h_2}}{\overline{g_1}}}\right) g \\ &= (h_2, h_1, \boldsymbol{h_0}, \overline{h_1}, \overline{h_2}) - \left(h_2, \frac{h_2 g_0}{g_1}, \boldsymbol{\frac{h_2 \overline{g_1}}{g_1} + \frac{\overline{h_2} g_1}{\overline{g_1}}}, \frac{\overline{h_2} g_0}{\overline{g_1}}, \overline{h_2}\right) \\ &= \Bigg(0, \frac{1}{g_1}(h_1 g_1 - h_2 g_0), \\ &\quad \boldsymbol{\frac{h_0 g_0 - (\overline{h_1} g_1 + h_1 \overline{g_1})}{g_0} + g_1 \cdot \left(\frac{\overline{h_1}}{g_0} - \frac{\overline{h_2}}{\overline{g_1}}\right) + \overline{g_1} \cdot \left(\frac{h_1}{g_0} - \frac{h_2}{g_1}\right)}, \\ &\quad \frac{1}{\overline{g_1}}(\overline{h_1 g_1} - \overline{h_2} g_0), 0\Bigg) \\ &\overset{(2.3.1)}{=} \left(0, 0, \boldsymbol{\frac{h_0 g_0 - (\overline{h_1} g_1 + h_1 \overline{g_1})}{g_0}}, 0, 0\right) \\ &\overset{(2.3.2)}{=} \left(0, 0, \boldsymbol{\frac{1}{g_0}}, 0, 0\right) \\ g' &:= g \end{aligned}$$

Now h' and g' are the filters that are present before the lifting step t is applied. Since g' is still not a constant in general, we need yet another lifting step with the filter $s = (\boldsymbol{g_0 g_1}, g_0 \overline{g_1})$:

$$\begin{aligned} h'' &:= h' \\ g'' &:= g' - (\boldsymbol{g_0 g_1}, 0, g_0 \overline{g_1}) h' \\ &= (\boldsymbol{g_1}, g_0, \overline{g_1}) - (\boldsymbol{g_1}, 0, \overline{g_1}) \\ &= (\mathbf{0}, g_0, 0) \end{aligned}$$

We obtained h'' and g'' which are constants that are reciprocal to each other. With four additional lifting steps they could be factorized into the lazy wavelet ($h''' = 1, g''' = 1$) according to [4].

The full lifting sequence is:

1. Weighting the L band with $\frac{1}{g_0}$ and the H band with g_0
2. Lifting from L band to H band with lifting filter $s = (\boldsymbol{g_0 g_1}, g_0 \overline{g_1})$
3. Lifting from H band to L band with lifting filter $t = (\frac{h_2}{g_1}, \overline{\boldsymbol{\frac{h_2}{g_1}}})$

Determine the parameters: The previous thought has shown, that each of the two lifting filters has the form (x, x) if working with real values. We start at the filters

$$\begin{aligned} h'' =& (0, 0, \boldsymbol{\frac{1}{a}}, 0, 0) \\ g'' =& \quad\;\; (\boldsymbol{0}, a, 0) \end{aligned}$$

and the first lifting filter is

$$s = (\boldsymbol{-\frac{1}{2}a^2}, -\frac{1}{2}a^2)$$

which does the linear interpolation, because the resulting filter

$$\begin{aligned} g' &= g'' + s(\circ^2) \cdot h'' \\ &= (\boldsymbol{-\frac{a}{2}}, a, -\frac{a}{2}) \\ &= a \cdot (\boldsymbol{-\frac{1}{2}}, 1, -\frac{1}{2}) \end{aligned}$$

has the required form.

The second lifting filter is set to

$$t = (b, \boldsymbol{b})$$

with still unknown b.

The filter pair built from these lifting steps is:

$$\begin{array}{lclccclclccl} h_e & := & (& -\frac{ab}{2} & \boldsymbol{\frac{1}{a} - ab} & -\frac{ab}{2} &) & h_o & := & (& ab & \boldsymbol{ab} &) \\ g_e & := & (& & \boldsymbol{-\frac{a}{2}} & -\frac{a}{2} &) & g_o & := & (& & \boldsymbol{a} &) \end{array}$$

Now, our aim can be expressed by the determination of

$$\operatorname*{argmin}_{a,b} \max_r \widetilde{p}(r)$$

It is easy to predict that the coefficient of the square term of $\widetilde{p}$ (defined in (2.1.19) and (2.1.12)) is $\frac{1}{8}(ab)^2$. Thus $\widetilde{p}$ is convex again, its maximum is either at $r = -1$ or $r = 1$ and the maximum of p is at $z = -1$ or $z = 1$ respectively.

$$\max_r \widetilde{p}(r) \quad = \quad \max\{p(-1), p(1)\}$$

To process both cases together, we will use the pair notation $\{a, b\}$. Each operation applies to both members of the tuple. It is not really a set, because the order is important. E.g. $\{a, b\} \geq \{c, d\}$ means $a \geq c \;\wedge\; b \geq d$ and a relation between either a and d or b and c is not given.

$$\begin{aligned}
2\{p(-1),p(1)\} &= 2p(\{-1,1\}) \\
&= \left(\{1,-1\}\cdot ab + \frac{1}{a} - ab\right)^2 + (\{0,2\}\cdot ab)^2 + (\{0,-1\}\cdot a)^2 + a^2 \\
&= \left(\{0,-2\}\cdot ab + \frac{1}{a}\right)^2 + (\{0,2\}\cdot ab)^2 + \{1,2\}\cdot a^2 \\
&= \{0,8\}\cdot(ab)^2 + \frac{1}{a^2} + \{0,-4\}\cdot b + \{1,2\}\cdot a^2 \\
&= \{0,1\}\cdot\left(8(ab)^2 - 4\cdot b + a^2\right) + \frac{1}{a^2} + a^2 \\
&= \{0,1\}\cdot\left(2\left(2ab - \frac{1}{2a}\right)^2 - \frac{1}{2a^2} + a^2\right) + \frac{1}{a^2} + a^2 \\
&\text{equality for } b = \frac{1}{4a^2} \\
&\geq \{0,1\}\cdot\left(a^2 - \frac{1}{2a^2}\right) + a^2 + \frac{1}{a^2} \\
&= \left\{a^2 + \frac{1}{a^2}, 2a^2 + \frac{1}{2a^2}\right\}
\end{aligned}$$

This implies

$$\begin{aligned}
\forall b: \quad 2\max\{p(-1),p(1)\} &\geq \max\left\{a^2 + \frac{1}{a^2}, 2a^2 + \frac{1}{2a^2}\right\} \\
\min_b\left(2\max\{p(-1),p(1)\}\right) &\geq \max\left\{a^2 + \frac{1}{a^2}, 2a^2 + \frac{1}{2a^2}\right\}
\end{aligned}$$

Because for given a, $p(-1)$ and $p(1)$ are minimal for the same b, we can exchange $\min$ and $\max$:

$$\begin{aligned}
\min_b\left(2\max\{p(-1),p(1)\}\right) &= 2\max\left\{\min_b p(-1), \min_b p(1)\right\} \\
&= \max\left\{a^2 + \frac{1}{a^2}, 2a^2 + \frac{1}{2a^2}\right\}
\end{aligned}$$

Thus we will set

$$b = \frac{1}{4a^2} \tag{2.3.3}$$

for further considerations of $p(-1)$ and $p(1)$. What remains is the calculation of

$$\operatorname*{argmin}_a \max\{p(-1),p(1)\}$$

We note that both $2p(-1) = a^2 + \frac{1}{a^2}$ and $2p(1) = 2a^2 + \frac{1}{2a^2}$ are convex in a, so $\max 2p(\{-1,1\})$ is, too. That is why there is only one minimum and we guess, that it is reached when both terms are equal.

$$\begin{aligned} 2a^2 + \frac{1}{2a^2} &= a^2 + \frac{1}{a^2} \\ a^2 &= \frac{1}{2a^2} \\ a^4 &= \frac{1}{2} \\ a &= \sqrt[4]{\frac{1}{2}} \end{aligned}$$

Because the function $2(p(1) - p(-1)) = a^2 - \frac{1}{2a^2}$ is monotonically increasing in a, we can verify easily, that

$$\begin{cases} p(-1) < p(1) & : a > \sqrt[4]{\frac{1}{2}} \\ p(-1) > p(1) & : a < \sqrt[4]{\frac{1}{2}} \end{cases}$$

In other words

$$\max\{p(-1), p(1)\} = \begin{cases} p(1) & : a > \sqrt[4]{\frac{1}{2}} \\ p(-1) & : a < \sqrt[4]{\frac{1}{2}} \end{cases}$$

That is why the left and right hand derivatives of $\max 2p(\{-1, 1\})$ at $a = \sqrt[4]{\frac{1}{2}}$ are

$$\begin{aligned} 2p_{a-}(-1) &= 2a - \frac{2}{a^3} \\ &= \frac{2a^4 - 2}{a^3} \qquad \Big| \qquad a^4 < \tfrac{1}{2} \\ &< 0 \end{aligned}$$

$$\begin{aligned} 2p_{a+}(1) &= 4a - \frac{1}{a^3} \\ &= \frac{4a^4 - 1}{a^3} \qquad \Big| \qquad a^4 > \tfrac{1}{2} \\ &> 0 \end{aligned}$$

This proves that $a = \sqrt[4]{\frac{1}{2}}$ minimizes $\max 2p(\{-1, 1\})$. This means

$$\begin{aligned} a &= \sqrt[4]{\frac{1}{2}} \\ (2.3.3) \Rightarrow \qquad b &= \frac{\sqrt{2}}{4} \\ p &= \frac{3}{4}\sqrt{2} \\ \lambda_{\max/\min} &= \frac{3}{4}\sqrt{2} \pm \sqrt{\frac{18}{16} - 1} = \frac{3 \pm 1}{4}\sqrt{2} \end{aligned}$$

$$\lambda_{\max} = \sqrt{2} \qquad \|F\|_2 = \sqrt[4]{2}$$
$$\lambda_{\min} = \frac{1}{\sqrt{2}} \qquad \left\|F^{-1}\right\|_2^{-1} = \frac{1}{\sqrt[4]{2}}$$

$$\begin{aligned} ab &= \frac{\sqrt[4]{2}}{4} \\ h &= \frac{\sqrt[4]{2}}{8}(-1, 2, \mathbf{6}, 2, -1) \\ g &= \frac{1}{2\sqrt[4]{2}}(\mathbf{-1}, 2, -1) \end{aligned}$$

We see that the result is a CDF-2,2 wavelet weighted by $\sqrt[4]{2}$. Refer to section 3.2.1 for further considerations of weighting.

Chapter 3

Construction of image specific wavelets

3.1 Minimization of entropy

Preprocessing images with wavelet transforms is a good choice to achieve better compression rates. The goal of transforming an image is to gain only a few values of significance, so that good compression is simply done by throwing away the less significant values. Further compression can be achieved by quantifying the transformed values. Care must be taken that modifying wavelet coefficients does not damage the original image visually.

A technique that saves space by quantifying values and suppressing low values and which pays attention to the special structure of wavelet transformed images (figure 3.1) is the Embedded Zero Tree method [14, 15]. In this encoding algorithm, the wavelet transformed image is treated as a multi-rooted directed tree. Each node of the tree corresponds to a pixel of the multi-scale representation. The tree is defined in such a way that each node v has either no offspring or four offspring which are 'refinements' of node v.

The `EZT` encoding is an iterative procedure: In the j. iteration, it starts by encoding the roots of the tree, i.e., the tree nodes in which normally most of a wavelet transformed image's energy is concentrated. Then for each pixel of this level, the corresponding subtree is considered. If all the nodes of the subtree are insignificant with respect to the j. threshold T_j, then the offspring of the pixel are not encoded and the subtree is pruned away. If the subtree is not such a zerotree with respect to T_j, the offspring are encoded and the procedure is recursively applied to the offspring. Here, a pixel of the wavelet transformed image is called *insignificant with respect to a threshold* T_j if its magnitude is smaller than T_j. A subtree is called *zerotree with respect to* T_j, if all of its nodes are insignificant with respect to T_j. [12]

Wavelet transformation is designed to produce as low as possible transformed values on the high frequency bands HL, LH, HH, if the images are natural, which means that the assumption "neighboured pixels are similar" holds in general. The lower the values on these bands are, the more often can the `EZT` encoder make use of the efficient coding of a complete sub-tree as zerotree. If small values can be obtained on a local image part over many levels, a deep sub-tree can be encoded as zerotree. If the values of the high frequency bands can be made small in general, the zerotree encoding can be applied even for small thresholds T_j.

In general, standard wavelets are used, which are constructed under some theoretical aspects and which have proven on much example images to satisfy the criterion of high frequency bands with low amplitude. Newer approaches (as sketched in [13]) select wavelet filters which provides the best result for a specific image or part of it, from a discrete set of standard wavelets. Another possible extension

original transformed

Figure 3.1: Structure of wavelet transformed images – Each transformed level consists of three high frequency bands HL (right top corner), LH (left bottom), HH (right bottom) for the scale corresponding to the level. The HL bands emphasize vertical edges, the LH bands emphasize horizontal edges. The very small part in the left top corner is a scaled down version of the original image. The original image is transformed over 6 levels with the CDF-2,2 wavelet. Values with small magnitude are white, values with big magnitude (independent from the sign) are drawn black.

is to further divide and transform the H bands of some transformation step results. This is associated with the concept of *wavelet packet bases* and *best bases selection*. Refer to [3, 10, 9] for details.

In the further considerations, we want to keep the simple scheme of dividing the input signal into high and low frequency bands, but we will drop the restriction to predefined wavelets. This should make it possible to react on image characteristics like intensive noise, which can not be handled by most standard wavelets. Further on, the adaptive creation of the wavelet would save an explicit algorithmic distinction, e.g. between smooth and noisy images. If multiple wavelets are used, each for a part of the image, more local features e.g. edges or local patterns of an image can be respected. But the more wavelets have to be created, the more wavelet coefficients have to be stored in the compressed image file, which may impair the enhancement that was achieved by making the wavelets adaptive to the image.

3.1.1 Goal of optimization

As stated above, on natural images, standard wavelets reduce the values on the H band. This effect is utilized for compression. With adaptively created wavelets we want to optimize this reduction. So first of all, we have to select a measurement for a total amount of all signal values. The definition of vector norms fits best to our intuitive concept of total amounts. Some of the vector norms as the sum norm, the maximum norm and the EUCLIDean norm (which is equivalent to the PSNR measurement for optimization, see section 2) are explored quite well. The EUCLIDean norm weights big vector components more than small components. This differs from the sum norm which weights all component values equally. But the EUCLIDean norm does not over-emphasize big components in opposition to the maximum norm, where only one big component is sufficient to increase the norm to the component's value.

Thus the EUCLIDean norm warrants a good balance and furthermore it allows the application of straightforward optimization algorithms sometimes.

For simplicity we will start with one-dimensional signals. Then, the EUCLIDean norm is defined as

$$\|x\|_2 = \sqrt{\sum_{i=0}^{n-1} {x_i}^2}$$

At first, we try to not to restrict the norm reduction to the H bands for more generality. Let W be the matrix that describes the whole wavelet transform, then we can express this goal as finding a W where

$$\|Wx\|_2 \rightarrow \text{minimum!}$$

with the requirement that W^{-1} must exist.

But the requirement of the reversibility of W (equivalent to $\forall x \neq 0 : \|Wx\|_2 > 0$) is not enough. Since $\|Wx\|_2$ depends continuously on a varying wavelet transformation W, there is no minimum if we exclude $\|Wx\|_2 = 0$, only. To avoid this problem, we could e.g. claim wavelet transformations with balanced norms of the transformations in both directions, $\|W\|_2 = \left\|W^{-1}\right\|_2$. Note that this may contradict to a normalization of the polyphase determinant to 1 (section 2.1.2). As the normalization of the polyphase determinant, the normalization of the norms should also have good numerical properties, because the optimization can not lead to wavelet transformations with low (good) bounds for the one direction which has to be paid with high (bad) bounds for the reverse transformation. The balancing results in a normalization which can be assured by scaling. For every reversible W a scaled

$$W' = \sqrt{\frac{\|W^{-1}\|_2}{\|W\|_2}} \cdot W$$

would be such a balanced wavelet transformation. Thus a better optimization approach would be

$$\left\|W'x\right\|_2 \rightarrow \text{minimum!}$$

with the requirement that $\|W'\|_2 = \left\|W'^{-1}\right\|_2$ (which includes the existence of W'^{-1}).

How can we do this optimization, respecting that the linear transformation W is a wavelet transformation and is normalized to $\|W\|_2 = \left\|W^{-1}\right\|_2$? The generic representation of any finite dimensional linear operation is the matrix representation. But the whole wavelet transformation written as a matrix has a complex structure. The matrix coefficients depend non-linearly on the filter coefficients, and it is difficult to retrieve the filter coefficients from the matrix representation. This is even more complicated, if different filters on different transformation levels are allowed. Also, the restriction to minimize all H bands only would not simplify the problem. Thus it should be easier to optimize within one transformation step.

3.1.2 The optimization scheme

Keep in mind that we have to construct invertible wavelets, which means that the restriction for the filter pair h, g must be respected, i.e. the polyphase matrix has to be regular, which is equal to a monomial determinant of the polyphase matrix. Additionally, we will require the determinant to be normalized to the constant 1 (refer to section 2.1.2). But with this normalization it makes no longer sense to normalize the transformation norms. So we will drop the condition of balanced matrix norms.

Under the restriction of the normalized polyphase determinant we have to minimize the norm of the transformed signal after one transformation step. Since the polyphase determinant depends non-linearly on the filter coefficients, the problem is still too complex. We also do not know how modifying the L band (the one which is transformed in the next step again) can influence following transformation steps. So we will content ourselves to minimizing the norm of the H band, leaving the L band unchanged and rely on the reduction of the norm of the L band in subsequent transformation steps.

The greatest common divisor of the filters g_e, g_o (those that form g) is always a divisor of the polyphase determinant. Thus g_e and g_o must be relatively prime, otherwise the polyphase determinant can not be a monomial. This restriction complicates the optimization still too much. At this point the lifting scheme [4] comes to a rescue, because wavelet transformations constructed with it are always reversible. Thus we will try to design lifting steps from L band x_e to H band x_o to reduce the energy on the H band as depicted in figure 3.2.

Lifting with s as lifting filter has the following effect, depending on the lift direction:

$$\begin{array}{rcl|l} x'_o & := & x_o + x_e * s & \text{L} \to \text{H lifting} \\ x'_e & := & x_e + x_o * s & \text{H} \to \text{L lifting} \end{array}$$

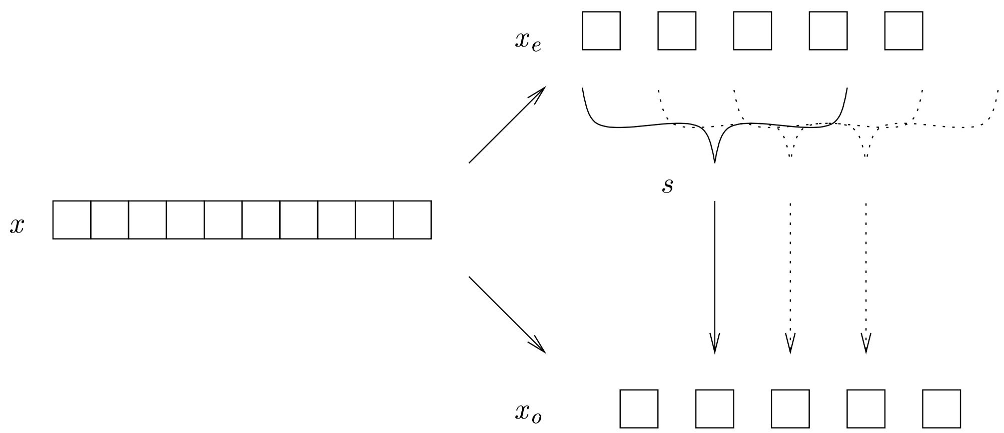


Figure 3.2: Idea of lifting – How pixels in one band predict the ones in the other

Multiple lifting steps are applied alternating between the two bands, because two following lifting steps in the same direction can be merged to one. If multiple lifting steps are applied, either x'_e or x'_o replaces x_e and x_o, respectively, as input for the succeeding step. Figure 3.3 may demonstrate that.

The big advantage of the lifting method is, that we automatically receive invertible transformations as results. We can even use non-linear lift functions, anyhow we can be sure that we can do a reverse transformation.

In the context of reducing the norm of the H band we can interpret the lifting step as predicting the H band x_o signal by filtering the L band x_e. The filter s is optimal if it filters the L band x_e and produces the prediction $y_e := s * x_e$ that is as similar as possible to the H band x_o. In other words $\|x'_o\|_2 = \|x_o - s * x_e\|_2$ should be small.

$$\|s * x_e - x_o\|_2 \to \text{minimum!}$$

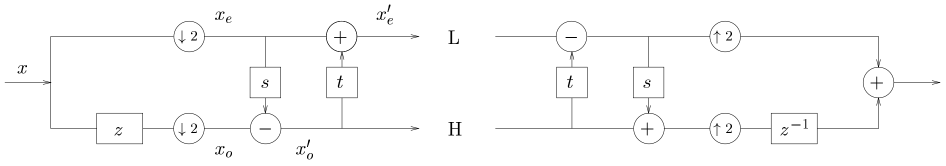


Figure 3.3: The lifting scheme in one dimension – Here for example with two steps s and t and the corresponding back transformation on the right side

3.1.3 Least mean square optimization

Now, let us express our strategy in terms of Linear Algebra. For simpler notation we will use x as the identifier for the source signal, $y = s * x$ for the predicting signal derived from it and u for the signal to be predicted. Thus it is $x = x_e, y = y_e, u = x_o$, and the optimization goal is expressed by

$$\|s * x - u\|_2 \to \text{minimum!}$$

Let S be the transformation matrix corresponding to the filter $s = (s_m, s_{m+1}, \ldots, s_{n-1}, s_n), m \leq 0, n \geq 0$, that means $Sx = s * x$, and let l be the length of the H band. Then S can be written as

$$\begin{pmatrix}
s_n & s_{n-1} & s_{n-2} & \cdots & s_m & 0 & 0 & \cdots & 0 & 0 & 0 \\
0 & s_n & s_{n-1} & \cdots & s_{m+1} & s_m & 0 & \cdots & 0 & 0 & 0 \\
0 & 0 & s_n & \cdots & s_{m+2} & s_{m+1} & s_m & \cdots & 0 & 0 & 0 \\
\vdots & \vdots & \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots & \vdots & \vdots \\
0 & 0 & 0 & \cdots & s_n & s_{n-1} & s_{n-2} & \cdots & s_m & 0 & 0 \\
0 & 0 & 0 & \cdots & 0 & s_n & s_{n-1} & \cdots & s_{m+1} & s_m & 0 \\
0 & 0 & 0 & \cdots & 0 & 0 & s_n & \cdots & s_{m+2} & s_{m+1} & s_m
\end{pmatrix}$$

where $S \in \mathbb{R}^{l, l+|s|}$

$$\begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \\ y_{l-3} \\ y_{l-2} \\ y_{l-1} \end{pmatrix} = S \cdot \begin{pmatrix} x_m \\ x_{m+1} \\ x_{m+2} \\ \vdots \\ x_{l+n-3} \\ x_{l+n-2} \\ x_{l+n-1} \end{pmatrix}$$

If it is inconvenient that the filtered signal is shorter than the input, the first $-m$ and last n columns of S could be cut off, which is equivalent to filling the unknown input values $x_m, \ldots, x_{-1}$ and $x_l, \ldots, x_{l+n-1}$ with zeros. Finding a filter which minimizes the norm of $y - u$, means finding the corresponding filter matrix, which is not a very clever approach. But we remember that the convolution is commutative, so it is the same if we ask for an s where

$$\|x * s - u\|_2 \to \text{minimum!}$$

That means we express x as a matrix $X \in \mathbb{R}^{l,|s|+1}$ and s as a vector.

$$\begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \\ y_{l-3} \\ y_{l-2} \\ y_{l-1} \end{pmatrix} = X \cdot \begin{pmatrix} s_m \\ s_{m+1} \\ s_{m+2} \\ \vdots \\ s_{n-2} \\ s_{n-1} \\ s_n \end{pmatrix}$$

$$\begin{aligned} X &= (x_{jk})_{j=0\ldots l-1, k=m\ldots n} \\ & \quad x_{jk} = \begin{cases} x_{j-k} & : 0 \leq j-k < l \\ 0 & : \text{otherwise} \end{cases} \\ &= \begin{pmatrix} x_{-m} & x_{-m-1} & x_{-m-2} & \cdots & x_1 & x_0 & 0 & \cdots & 0 & 0 & 0 \\ x_{-m+1} & x_{-m} & x_{-m-1} & \cdots & x_2 & x_1 & x_0 & \cdots & 0 & 0 & 0 \\ x_{-m+2} & x_{-m+1} & x_{-m} & \cdots & x_3 & x_2 & x_1 & \cdots & 0 & 0 & 0 \\ \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \\ 0 & 0 & 0 & \cdots & x_{l-2} & x_{l-3} & x_{l-4} & \cdots & x_{l-n-1} & x_{l-n-2} & x_{l-n-3} \\ 0 & 0 & 0 & \cdots & x_{l-1} & x_{l-2} & x_{l-3} & \cdots & x_{l-n} & x_{l-n-1} & x_{l-n-2} \\ 0 & 0 & 0 & \cdots & 0 & x_{l-1} & x_{l-2} & \cdots & x_{l-n+1} & x_{l-n} & x_{l-n-1} \end{pmatrix} \end{aligned}$$

Finally we obtain a standard linear least mean square problem in s,

$$\|Xs - u\|_2 \to \text{minimum!}$$

which is solvable with several algorithms, such as normal equations, the QR decomposition method, the singular value decomposition of X and the pseudo inverse of X [19].

In our software package we follow the simple approach of normal equations:

$$s_{\min} = \operatorname*{argmin}_s \|Xs - u\|_2 \quad \Leftrightarrow \quad X^T X s_{\min} = X^T u$$

Assumed that $X^T X$ is regular the linear equation system on the right hand can be solved with a CHOLESKY decomposition. Tests with long filters (20 taps and more) have shown that the implementation of the CHOLESKY decomposition in `TNT` may fail sometimes, so that the iterative *conjugate gradient* method [19] is implemented in our software package, too, which replaces CHOLESKY's method now. Unfortunately it is not known, how the *conjugate gradient* iteration reacts on an almost singular matrix $X^T X$, which can occur when a filter s is requested, that has more coefficients than necessary for exact prediction of a given signal structure.

Table 3.1 presents the compression ratios which will be achieved with the least mean square prediction. Note that the transformation is done with a predicting step only, whereas an update step is left. We see that simple predicting may even damage the compression effect compared to a standard wavelet. It has to be explored what the problem is. Indeed the energy of the H band is decreased by longer prediction filters. It is well below the decrease of energy from the original signal to the one with the subtracted predicted signal. The results show that the decrease of energy in the H band does not necessarily lead to better pack rates.

The exception is the `lena noisy` image which consists of a mixture of `lena` with a synthetical pattern like that of a coarse printing. One can see that a certain filter length is needed to adapt the pattern. But synthetical patterns are also dangerous, because the matrix $X^T X$ of the normal equation system may become singular.

image	CDF 2,2	2	4	10	50
baboon	**6.25**	6.34	6.32	6.30	6.30
goldhill	**5.06**	5.12	5.11	5.11	5.11
lena noisy	8.92	8.53	8.63	5.56	**5.47**
lena	**4.56**	4.62	4.67	4.66	4.67
mountain	**7.38**	7.52	7.45	7.44	7.44
parrot	**4.44**	4.72	4.65	4.65	4.74
pepper	**4.97**	5.22	5.21	5.20	5.18
sar_pripuls	7.21	7.24	7.22	7.21	**7.21**

Table 3.1: Least mean square prediction – How does increasing the size of the predictor influence the pack rate? CDF-2,2 is compared to wavelets which consist of a single prediction lifting step. The predictor is computed for every image and for every transformation level and direction (horizontal, vertical). The predictors have the sizes 2, 4, 10, 50, respectively. Transformation is done over 6 levels. Thereafter every image is lossless compressed with EZT method. The bits per pixel rate achieved is presented here.

3.1.4 Special cases

The described optimization scheme can be modified to better fit particular applications. Every modification to the scheme that leaves the linear characteristic unchanged can be handled with the linear least mean square optimization algorithm.

Treatment of undefined input values: As mentioned above $l + |s|$ input values are necessary to get l output values, when using a filter with $|s| + 1$ coefficients. But normally you expect the same number of input and output values. The solution used above was to fill extra input values with zero. But this introduces discontinuities into the input signal. Instead, values can be reused at the start and the end in reversed order. This results in the matrix:

$$
\begin{aligned}
X &= (x_{jk})_{j=0\ldots l-1,k=m\ldots n} \\
& \quad x_{jk} = \begin{cases} x_{(j-k) \bmod l} & : 0 \leq (j-k) \bmod 2l < l \\ x_{(k-j-1) \bmod l} & : \text{otherwise} \end{cases} \\
&= \begin{pmatrix}
x_{-m} & x_{-m-1} & \cdots & x_2 & x_1 & x_0 & x_0 & x_1 & \cdots & x_{-m-2} & x_{-m-1} \\
x_{-m+1} & x_{-m} & \cdots & x_3 & x_2 & x_1 & x_0 & x_0 & \cdots & x_{-m-3} & x_{-m-2} \\
x_{-m+2} & x_{-m+1} & \cdots & x_4 & x_3 & x_2 & x_1 & x_0 & \cdots & x_{-m-4} & x_{-m-3} \\
\ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \\
x_{l-n+2} & x_{l-n+3} & \cdots & x_{l-1} & x_{l-2} & x_{l-3} & x_{l-4} & x_{l-5} & \cdots & x_{l-n-2} & x_{l-n-3} \\
x_{l-n+1} & x_{l-n+2} & \cdots & x_{l-1} & x_{l-1} & x_{l-2} & x_{l-3} & x_{l-4} & \cdots & x_{l-n-1} & x_{l-n-2} \\
x_{l-n} & x_{l-n+1} & \cdots & x_{l-2} & x_{l-1} & x_{l-1} & x_{l-2} & x_{l-3} & \cdots & x_{l-n} & x_{l-n-1}
\end{pmatrix}
\end{aligned}
$$

The assumption of periodic input signal can be made to legitimate the FOURIER analysis. Then

X looks like this:

$$
\begin{aligned}
X &= (x_{jk})_{j=0\ldots l-1,k=m\ldots n} \\
&\qquad x_{jk} = x_{(j-k) \bmod l} \\
&= \begin{pmatrix}
x_{-m} & x_{-m-1} & x_{-m-2} & \cdots & x_1 & x_0 & x_{l-1} & \cdots & x_{l-n+2} & x_{l-n+1} & x_{l-n} \\
x_{-m+1} & x_{-m} & x_{-m-1} & \cdots & x_2 & x_1 & x_0 & \cdots & x_{l-n+3} & x_{l-n+2} & x_{l-n+1} \\
x_{-m+2} & x_{-m+1} & x_{-m} & \cdots & x_3 & x_2 & x_1 & \cdots & x_{l-n+4} & x_{l-n+3} & x_{l-n+2} \\
\ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \\
x_{-m-3} & x_{-m-4} & x_{-m-5} & \cdots & x_{l-2} & x_{l-3} & x_{l-4} & \cdots & x_{l-n-1} & x_{l-n-2} & x_{l-n-3} \\
x_{-m-2} & x_{-m-3} & x_{-m-4} & \cdots & x_{l-1} & x_{l-2} & x_{l-3} & \cdots & x_{l-n} & x_{l-n-1} & x_{l-n-2} \\
x_{-m-1} & x_{-m-2} & x_{-m-3} & \cdots & x_0 & x_{l-1} & x_{l-2} & \cdots & x_{l-n+1} & x_{l-n} & x_{l-n-1}
\end{pmatrix}
\end{aligned}
$$

Equated coefficients: The coefficients of the filter to be designed may depend on each other. The simplest dependency is that some coefficients have to be equal. You can modify X to generate symmetrical filters automatically. Say, two filter coefficients s_j and s_k shall be equal, then the result is the same if you remove s_k from the vector s and add the k. column to the j. one, removing the k. column from X hereafter.

For image processing it might be desirable to avoid visible distortions caused by unsymmetrical wavelet filters. It is no problem to restrict the optimization to symmetrical filters ($n = -m$):

$$\forall j \in \{0, \ldots, n\} : s_j = s_{-j}$$

You only need to determine the coefficients $s_0, \ldots, s_n$. Note that the following definition of X contains matrix elements with indices out of $0, \ldots, l-1$. Fill them as you like or follow the description of the previous section.

$$
\begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \\ y_{l-3} \\ y_{l-2} \\ y_{l-1} \end{pmatrix} = X \cdot \begin{pmatrix} s_0 \\ s_1 \\ \vdots \\ s_{n-1} \\ s_n \end{pmatrix}
$$

$$
\begin{aligned}
X &= (x_{jk})_{j=0\ldots l-1,k=0\ldots n} \\
&\qquad x_{jk} = \begin{cases} x_j & : k = 0 \\ x_{j-k} + x_{j+k} & : \text{otherwise} \end{cases} \\
&= \begin{pmatrix}
x_0 & x_{-1} + x_1 & x_{-2} + x_2 & \cdots & x_{-n} + x_n \\
x_1 & x_0 + x_2 & x_{-1} + x_3 & \cdots & x_{-n+1} + x_{n+1} \\
x_2 & x_1 + x_3 & x_0 + x_4 & \cdots & x_{-n+2} + x_{n+2} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
x_{l-3} & x_{l-4} + x_{l-2} & x_{l-5} + x_{l-1} & \cdots & x_{l-n-3} + x_{l+n-3} \\
x_{l-2} & x_{l-3} + x_{l-1} & x_{l-4} + x_l & \cdots & x_{l-n-2} + x_{l+n-2} \\
x_{l-1} & x_{l-2} + x_l & x_{l-3} + x_{l+1} & \cdots & x_{l-n-1} + x_{l+n-1}
\end{pmatrix}
\end{aligned}
$$

Table 3.2 shows how the compression rates change when restricting the prediction filters to symmetric coefficients. It was to expect that the visual improvement must be paid with less compression efficiency, but the results show that the compression rates are comparable to those of the unrestricted predictors (table 3.1).

image	CDF 2,2	2		4		10	
		asym	sym	asym	sym	asym	sym
baboon	**6.25**	6.34	6.34	6.32	6.32	6.30	6.31
goldhill	**5.06**	5.12	5.11	5.11	5.11	5.11	5.13
lena noisy	8.92	8.53	8.53	8.63	8.62	**5.56**	5.58
lena	**4.56**	4.62	4.62	4.67	4.66	4.66	4.66
mountain	**7.38**	7.52	7.52	7.45	7.46	7.44	7.44
parrot	**4.44**	4.72	4.72	4.65	4.65	4.65	4.63
pepper	**4.97**	5.22	5.20	5.21	5.19	5.20	5.18
sar_pripuls	7.21	7.24	7.24	7.22	7.22	**7.21**	7.21

Table 3.2: Least mean square prediction with symmetrical predictors – It is the same situation as in table 3.1, except that now symmetrical predictor filters are computed and applied additionally. They are compared to the asymmetric predictors from above.

3.1.5 Lifting variants

Our approach is not limited to one dimensional (1D) lifting. Some extensions can be made easily. The idea is always the same: Filtering data means calculating some linear combinations of input values and filter coefficients. So look which input values x_{j_k} are included to calculate one output value y_j and write them as the j. row into X. Then solve $X^T X s = X^T u$ for s.

2D lifting: There are several possibilities to extend the optimization scheme to two dimensions. The restriction to pictures that have only one component per pixel, e.g. grey scale pictures, is retained.

The easiest extension emerges if the image is considered as a set of columns or rows and operations are performed on the columns and rows like on one dimensional signals. This results in separable wavelets and does not give us more choices for lifting scheme generalizations. But it can still be chosen how many filters shall be constructed. One per line, one per image or one for some line groups, for groups of equal or different sizes. Since optimization is based on the result of the previous transformation step, the order whether you start on rows or on columns will influence the result as well.

The other possibility to move towards 2D is to think about generalized lifting schemes. We remember that the approach in one dimension was to predict values of the odd band using values of the even band. On images we have not only two bands but four sub-pictures. After sorting the pixels, odd to the right/bottom, even to the left/top, the right bottom picture now plays the role of the H band. It is possible to design filters that predict the values of the right bottom picture HH by filtering values from the other three pictures. The case of small filters can be better illustrated on the original image (refer to figure 3.4). Every pixel of the HH part $u_{j,k}$ with $j \equiv k \equiv 1 \mod 2$ is predicted with

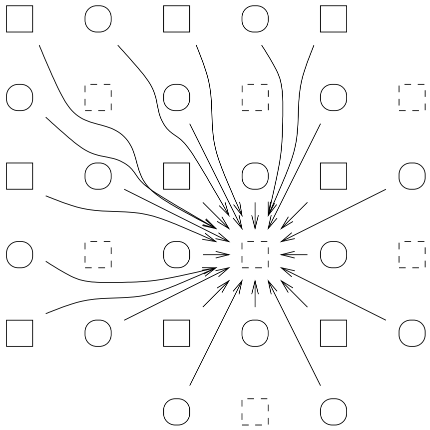
$$y_{j,k} = \sum_{\substack{a \equiv 0 \mod 2 \ \vee \\ b \equiv 0 \mod 2}} s_{j-a,k-b} \cdot x_{a,b}$$

E.g. with 3×3 filters you predict the value of one pixel by a linear combination of the pixels in its 8-neighbourhood.

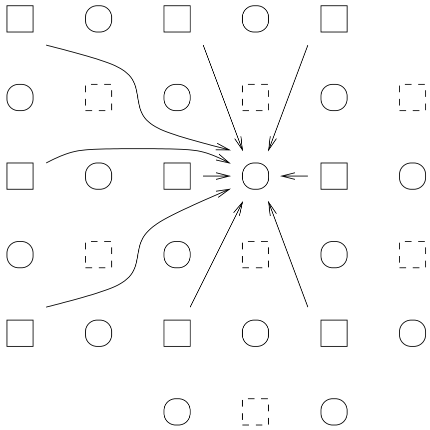
In a second step you design filters to predict the right top picture LH with the LL band, i.e. for all $j \equiv 1 \mod 2, k \equiv 0 \mod 2$ you calculate

$$y_{j,k} = \sum_{\substack{a \equiv 0 \mod 2 \ \wedge \\ b \equiv 0 \mod 2}} s_{j-a,k-b} \cdot x_{a,b}$$

and analogous, you predict the left bottom picture HL with the LL part. This method makes sure that the result is independent of the optimization order. This differs from the adaptive lifting step creation for separable wavelets!



Pixels which predict the ones in the H H part



Pixels which predict the ones in the L H part

Figure 3.4: Pixel prediction in two dimensions – Boxes corresponds to LL band pixels, dashed boxes corresponds to HH band pixels, rounded boxes corresponds to either LH or HL band pixels.

The ideas of 2D lifting (*interpolation* from *sub-sampled images*) and least mean square optimization (BURG*s algorithm*) are always covered by [8] separately. Different to what we consider here, updating lifting steps are left and the transformed data is compressed with HUFFMAN, arithmetical coding and LEMPEL-ZIV-WELSH (LZW) compression instead of EZT. The least mean square prediction which is performed line by line and from left to right in every row with appended HUFFMAN or arithmetical coding achieved the best results in that test.

Non-linear lifting: According to table 3.1 and table 3.2 using bigger filters does not warrant to improve the results. So it is of interest if one can get better results by extracting more information from pixels which lay close around the one that shall be predicted. For doing the optimization it is not necessary to restrict to linear dependencies from the input values. As shown in figure 3.5 it is also possible to use linear combinations of terms which depend non-linearly on the input values. E.g. it would be a special case if one band would be processed in a non-linear way into a temporary buffer and then used as the source for a filter based lifting step. As example for non-linear lifting we consider additional terms like $x_n x_{n+1}$ (which has the wrong physical unit, what means that its coefficient depends on the amplitude of input signal) or $\sqrt{x_n x_{n+1}}$ (which makes trouble when the radicand is negative and it is not clear how to choose the sign of the root). Such non-linear terms miss some nice characteristics: They depend on the offset of input data, which means you can not reuse coefficients on the same data if their offset is varying. Given, an image has a characteristic which is described very good by such terms but has a slightly changing offset, using the additional term will not lead to an advantage.

Assuming that $x_n x_{n+1} < 0$ indicates a zero between n and $n+1$, we quickly write a function which works around the root problems and we can build a new X:

$$c(u,v) \quad := \quad \begin{cases} \operatorname{sgn}(u) \cdot \sqrt{uv} & : uv >= 0 \\ 0 & : \text{otherwise} \end{cases}$$

$$\begin{aligned} X &= (x_{jk})_{j=0\ldots l-1, k=0\ldots 2} \\ & \qquad x_{jk} = \begin{cases} x_j & : k=0 \\ c(x_j, x_{j+1}) & : k=1 \\ x_{j+1} & : k=2 \end{cases} \\ &= \begin{pmatrix} x_0 & c(x_0,x_1) & x_1 \\ x_1 & c(x_1,x_2) & x_2 \\ x_2 & c(x_2,x_3) & x_3 \\ \vdots & \vdots & \vdots \\ x_{l-3} & c(x_{l-3},x_{l-2}) & x_{l-2} \\ x_{l-2} & c(x_{l-2},x_{l-1}) & x_{l-1} \\ x_{l-1} & c(x_{l-1},x_l) & x_l \end{pmatrix} \end{aligned}$$

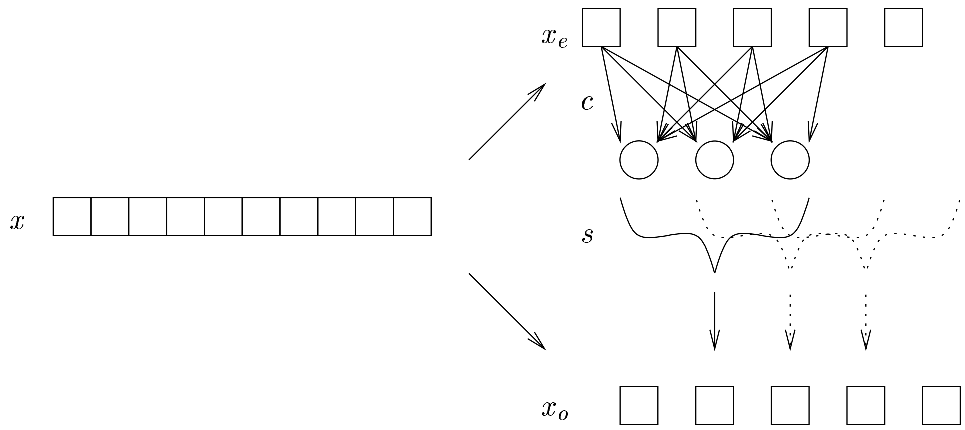


Figure 3.5: Non-linear prediction of the other band – The non-linear mapping c is inserted before the filter s.

3.2 Improve norm bounds

Since we are not working with orthogonal/unitary wavelets, it is not exactly known how big the change of the image is if some values of the transformed image are changed. Such changes happen if the transformed image is compressed in a lossy manner. In the following we will spend our interest on reducing this uncertainty (which can be expressed by tight lower and upper transformation bounds).

At first glance this goal seems to be contrary to the good pack rates we want to achieve, since the latter requires small values in the transformed image in contrast to the original image. But since the transformation is designed to be exactly reversible, it is natural that many small values require a few very big values somewhere in the transformed image. This must be similar in the behaviour of wavelets, which have close norm bounds, too.

We will discuss two approaches to design lifting steps that can be performed after other lifting steps, e.g. energy minimizing predictor lifting steps, to make the norm bounds closer. On the one hand, splitting the prediction and the norm narrowing step leads to sub-optimal results, on the other hand, when optimizing them together, it is not clear how to balance both optimization criteria.

You might ask, why we determine general norm bounds for wavelets we constructed just for a specific image. E.g., if you construct a predictor for a given image, in the worst case the H band is not touched and in the best case it vanishes, which means that the norm of the image after applying the predictor is between $\|x_e\|_2$ and $\sqrt{\|x_e\|_2^2 + \|x_o\|_2^2}$. Why general bounds if the transformation is only used for this signal?

The reason is the change of the transformed image caused by lossy compression, again. When transforming back we start on an image that differs more or less from the one we obtained by the analysis transformation. That is why we have to deal with the signal independent operator norms.

3.2.1 Weight filters

One possible way to improve the bounds is to weight the filters for both channels: One filter is amplified and the other must be weakened. This keeps the balance expressed by the polyphase determinant as given in (2.1.6) which was fixed to 1.

That is what we have to solve:

$$\alpha_{\min} = \operatorname*{argmin}_{\alpha\in\mathbb{R}\setminus\{0\}} \max_{\substack{z\in\mathbb{C}\\|z|=1}} p_\alpha(z)$$

p_α is the p polynomial as in (2.1.12) corresponding to the weighted polyphase matrix $\begin{pmatrix}\alpha h_e & \alpha h_o\\ \frac{1}{\alpha}g_e & \frac{1}{\alpha}g_o\end{pmatrix}$ which can be shortly expressed as:

$$p_\alpha := \alpha^2\cdot\underbrace{\tfrac{1}{2}(\overline{h_e}h_e+\overline{h_o}h_o)}_{=:v} + \frac{1}{\alpha^2}\cdot\underbrace{\tfrac{1}{2}(\overline{g_e}g_e+\overline{g_o}g_o)}_{=:u}$$

Weighting problem is convex: We are now going to show that this optimization problem is convex, which yields to some nice features. E.g. any local minimum of a convex problem is the global minimum, too, and the set of local minima forms an interval (maybe containing ∞).

Since $\overline{h_e(z)}h_e(z) \geq 0, \overline{h_o(z)}h_o(z) \geq 0, \ldots$ the v, u as defined have only non-negative real values on the complex unit circle. We know that the following defines a norm in the space of such polynomials:

$$\begin{aligned}\|v\|_\infty &:= \max_{\substack{z\in\mathbb{C}\\|z|=1}} |v(z)| \qquad\qquad \mid \qquad v(z)\geq 0\\ &= \max_{\substack{z\in\mathbb{C}\\|z|=1}} v(z)\end{aligned}$$

We will show, that

$$f(\beta) := \left\|\beta v + \frac{1}{\beta}u\right\|_\infty$$

is convex in $\beta \in \mathbb{R}_+$, thus f has exactly one minimum on an interval, which will often consist of one point only. Calling one of the minimum arguments $\beta_{\min}$ we obtain $\alpha_{\min} = \pm\sqrt{\beta_{\min}}$.

For all $t \in [0,1], \beta_0, \beta_1 \in \mathbb{R}_+$ it is true that

$$\begin{aligned}
f(t\beta_0 + (1-t)\beta_1) &= \left\|(t\beta_0 + (1-t)\beta_1)v + \frac{1}{t\beta_0 + (1-t)\beta_1}u\right\|_\infty \\
&\underset{v\geq 0 \ \wedge \ u\geq 0}{\overset{(3.2.1)}{\leq}} \left\|(t\beta_0 + (1-t)\beta_1)v + \left(t\frac{1}{\beta_0} + (1-t)\frac{1}{\beta_1}\right)u\right\|_\infty \\
&\leq t \cdot \left\|\beta_0 v + \frac{1}{\beta_0}u\right\|_\infty + (1-t)\cdot\left\|\beta_1 v + \frac{1}{\beta_1}u\right\|_\infty \\
&\leq t\cdot f(\beta_0) + (1-t)\cdot f(\beta_1)
\end{aligned}$$

which is the convexity condition. We have used this auxiliary calculation:

$$\begin{aligned}
0 &\leq t(1-t)(\beta_0 - \beta_1)^2 \\
0 &\leq t(t-1)\cdot 2\beta_0\beta_1 + t(1-t)(\beta_0^2 + \beta_1^2) \\
0 &\leq (t^2 + (t-1)^2 - 1)\cdot\beta_0\beta_1 + t(1-t)(\beta_0^2+\beta_1^2) \qquad \mid \quad +\beta_0\beta_1 \\
\beta_0\beta_1 &\leq (t\beta_0 + (1-t)\beta_1)(t\beta_1 + (1-t)\beta_0) \\
& \qquad\qquad \mid \quad : (\beta_0\beta_1)(t\beta_0 + (1-t)\beta_1) \\
\frac{1}{t\beta_0 + (1-t)\beta_1} &\leq \frac{t}{\beta_0} + \frac{1-t}{\beta_1}
\end{aligned} \tag{3.2.1}$$

We can also predict some limits of the upper bound minimization by weighting. How small can the upper bound become in the best case?

$$\begin{aligned}
\|p_\alpha\|_\infty &= \left\|\alpha^2 v + \frac{1}{\alpha^2}u\right\|_\infty \qquad \mid \qquad v \geq 0 \ \wedge \ u \geq 0 \\
&= \left\|\left(\alpha\sqrt{v} - \frac{1}{\alpha}\sqrt{u}\right)^2 + 2\sqrt{vu}\right\|_\infty \\
&\text{because of non-negative summands} \\
&\geq \left\|2\sqrt{vu}\right\|_\infty \\
&= 2\sqrt{\|vu\|_\infty}
\end{aligned}$$

With (2.1.15) you can compute the transformation bounds associated with this estimation. Note that this bound will not always be assumed.

Exact solution: $f(\beta)$ is continuous but not necessarily smooth, the graph may have pikes. A pike may occur whenever two maxima regarding z of $p_\alpha(z)$ have the same magnitude. Figure 3.6 visualizes this case. The minimum regarding β may be at such a pike, in this case we will call it a non-smooth minimum, otherwise a smooth minimum.

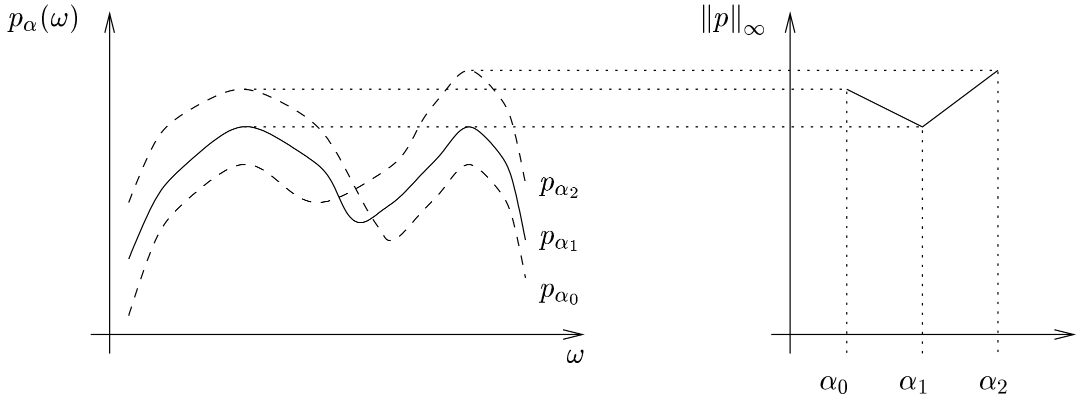


Figure 3.6: How non-smooth minima arise

With $z = e^{i\omega}$ let

$$y(\beta, z) := \beta v(z) + \frac{1}{\beta} u(z)$$

Then f can be expressed with the help of y:

$$f(\beta) = \max_{\substack{z\in\mathbb{C} \\ |z|=1}} y(\beta, z)$$

and we ask for

$$\beta_{\min} = \operatorname*{argmin}_{\beta} f(\beta)$$

What we could do, is to find all candidates of β for locations of pikes and all candidates for smooth minima. Then we had to verify which β of this set is the global minimum. Unfortunately, it is not clear, how to locate the pikes, but it is possible to find a small (i.e. finite) set of candidates for smooth minima by determining the set of all stationary points of y regarding β and z,

$$M := \{(\beta, z) : y_\beta(\beta, z) = 0, y_\omega(\beta, z) = 0\}$$

M contains all local minima, maxima and saddle points of y regarding β and z. Thus it contains the saddle point for $\beta_{\min}$, which is a minimum regarding β and a maximum regarding z.

For stationary points (β, z) it holds:

$$\begin{pmatrix} 0 \\ 0 \end{pmatrix} \stackrel{!}{=} \begin{pmatrix} y_\beta(\beta, z) \\ y_\omega(\beta, z) \end{pmatrix} \tag{3.2.2}$$

$$= \begin{pmatrix} v(z) - \frac{1}{\beta^2} u(z) \\ \beta v_\omega(z) + \frac{1}{\beta} u_\omega(z) \end{pmatrix} \tag{3.2.3}$$

This implies for every stationary point:

$$\begin{aligned} (3.2.2) \Rightarrow \qquad & 0 = y_\beta(\beta, z) u_\omega(z) + y_\omega(\beta, z) \frac{u(z)}{\beta} \\ (3.2.3) \Rightarrow \qquad & 0 = v(z) u_\omega(z) + v_\omega(z) u(z) \\ & = (vu)_\omega(z) \end{aligned}$$

For every z with $(vu)_\omega(z) = 0$ we can determine $\beta = \sqrt{\frac{u(z)}{v(z)}}$. These β's are the only candidates for a smooth minimum of $f(\beta)$. If for a fixed β, $y(\beta, z)$ is globally maximal in regard to z, then, since $y(\beta, z)$ is convex in regard to β, $f(\beta)$ is the global minimum and it is smooth. Otherwise the minimum is non-smooth and we do not know how to locate it exactly.

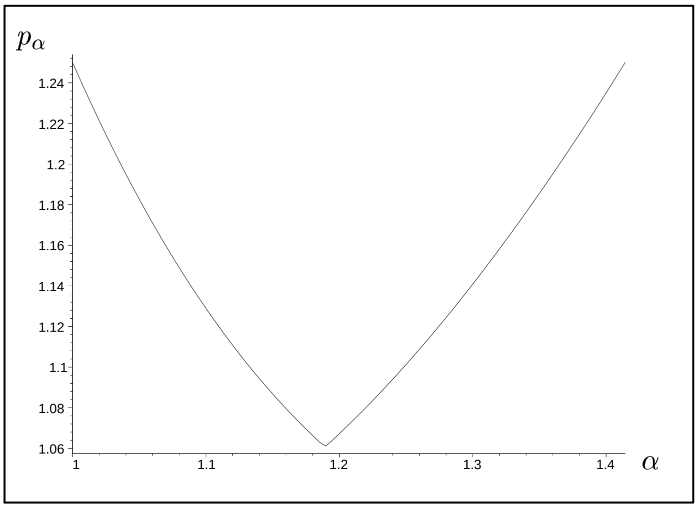


Figure 3.7: Graph of p_α

Example: Weight CDF-2,2: To verify the result we will weight the well-known CDF-2,2 wavelet filter pair in order to minimize the norm bounds. The graph of p_α displayed in figure 3.7 let us assume, that the minimum is non-smooth and lies around 1.19. But we will locate it exactly. According to section 2.1.6 it is

$$\begin{aligned}
128 p_\alpha &= \alpha^2 (1, -8, \mathbf{46}, -8, 1) + \frac{16}{\alpha^2}(1, \mathbf{6}, 1) \\
128 \widetilde{p}_\alpha(r) &= \alpha^2((2r)^2 - 8 \cdot (2r) + 44) + \frac{16}{\alpha^2}((2r) + 6) \\
32 \widetilde{p}_\alpha(r) &= \alpha^2(r^2 - 4r + 11) + \frac{8}{\alpha^2}(r + 3)
\end{aligned}$$

We see again that the coefficient of r^2 is positive, which is the reason that $\widetilde{p}_\alpha(r)$ is convex with respect to r and the maximum is at the borders of $[-1, 1]$.

$$\begin{aligned}
16 \widetilde{p}_\alpha(-1) &= 8\alpha^2 + \frac{8}{\alpha^2} \\
16 \widetilde{p}_\alpha(1) &= 4\alpha^2 + \frac{16}{\alpha^2}
\end{aligned}$$

We check the candidates for a smooth minimum ...

$\widetilde{p}_\alpha(-1)$	is minimal for	$\alpha^2 = 1$	and	$16 \left\| p_1 \right\|_\infty$	$= \max\{16, 20\} = 20$
$\widetilde{p}_\alpha(1)$	is minimal for	$\alpha^2 = 2$	and	$16 \left\| p_{\sqrt{2}} \right\|_\infty$	$= \max\{20, 16\} = 20$

…and the candidates for a non-smooth minimum

$$
\begin{aligned}
\widetilde{p}_\alpha(-1) &\stackrel{!}{=} \widetilde{p}_\alpha(1) \\
8\alpha^2 + \frac{8}{\alpha^2} &= 4\alpha^2 + \frac{16}{\alpha^2} \\
4\alpha^2 &= \frac{8}{\alpha^2} \\
\alpha^4 &= 2
\end{aligned}
$$

It is $16 \left\| p_{\sqrt[4]{2}} \right\|_\infty = 12\sqrt{2}$.

$$
\begin{aligned}
\min_\alpha \|p_\alpha\|_\infty &= \min \left\{ \|p_1\|_\infty, \left\|p_{\sqrt[4]{2}}\right\|_\infty, \left\|p_{\sqrt{2}}\right\|_\infty \right\} \\
&= \frac{3}{4}\sqrt{2} \\
\operatorname*{argmin}_\alpha \|p_\alpha\|_\infty &= \sqrt[4]{2}
\end{aligned}
$$

This is the result we expected referring to section 2.3.

Since the intention of narrowing the norm bounds was to give more safety when deciding about which pixels of the transformed image are relevant and which are not, we can expect that compression artifacts are reduced on lossy compression. With table 3.3 this hypothesis can be verified.

image	1	$\sqrt[8]{2}$	$\sqrt[4]{2}$	$\sqrt[8]{2}^3$	$\sqrt{2}$	Chebyshev
baboon	20.71	23.20	24.24	**24.58**	24.48	24.42
goldhill	28.89	30.07	31.57	**32.05**	32.00	31.78
lena noisy	9.54	10.20	**11.12**	10.97	10.43	10.79
lena	32.01	34.45	35.52	**35.84**	35.80	35.55
mountain	15.01	17.72	19.07	19.49	**19.54**	19.31
parrot	25.45	30.18	31.59	32.17	**32.20**	31.88
pepper	30.88	33.43	34.57	**34.95**	34.94	34.67
sar_pripuls	18.29	19.48	20.01	20.18	**20.29**	19.92

Table 3.3: Weighting and CHEBYSHEV wavelet – Do closer norm bounds assure, that the energy of compression artifacts is reduced? – After applying a CDF-2,2, the bands are weighted by the factors 1, $\sqrt[8]{2}$, $\sqrt[8]{4}$, $\sqrt[8]{8}$, $\sqrt[8]{16}$, respectively. For comparison the CHEBYSHEV wavelet constructed in section 2.2.3 is used in the right column. Transformation includes 6 levels. Thereafter the images are compressed to 0.5 bpp, decompressed and compared to the original image. The differences measured as PSNR (section 2) are presented here. Higher values mean better matching of original and decompressed image. According to the theory, from all weighted CDF-2,2 wavelets the one with weighting factor $\sqrt[4]{2}$ should give the best results.

As we can see, the weighting really reduces the difference between original and processed image. Apparently, the optimal weighting factor is always a bit above the theoretical predicted value of $\sqrt[4]{2}$.

$\sqrt[8]{2}^3$ seems to be a better choice. This could be explained since we considered one transformation step for optimization only, and thus did not get the optimum of the whole transformation. We realize, that the linear interpolating CHEBYSHEV wavelet achieves an improvement comparable to that of an optimally weighted CDF-2,2 wavelet, but it is not as good as the latter.

The reduction of the image difference is not only a technical improvement, it can be visually verified, too. Examples are given in figure 3.8.

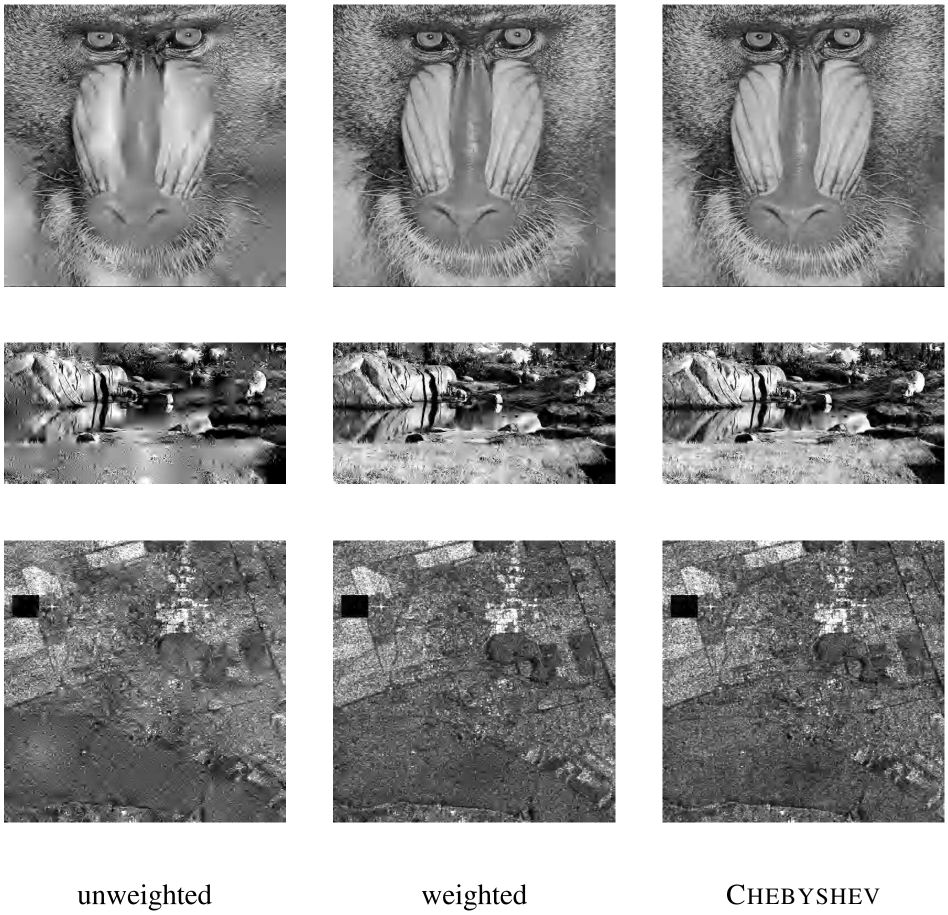


Figure 3.8: Weighting and CHEBYSHEV wavelet – Does reduced differences between original and lossily compressed images lead to visual improvement? The images in the first two columns are transformed with CDF-2,2, in the second column it is additionally weighted by $\sqrt[4]{2}$ after every transformation step. The right column contains images processed with the linear interpolating CHEBYSHEV wavelet of section 2.2.3. Transformation is repeated over 6 levels. Thereafter the transformed images are compressed to 0.5 bpp and decompressed again. The results are shown here.

We discover an interesting connection: With higher weighting factors the L band is amplified and the H band is weakened. One might expect that low frequencies will more and more dominate the decompressed image when the weighting factor increases. But figure 3.8 shows that the opposite is true – Up to a certain point the amplification of low frequency coefficients preserves more high frequencies in the image. But the bigger safety when evaluating the significance of a wavelet coefficient which is targeted by weighting may explain this behaviour.

Although the approach of reducing the norm bound differences was made to improve the results of lossy image compression, there might also be an increase of the pack rate when using wavelets of this approach for lossless compression. Table 3.4 validates this assumption. We can see that for the most images, the optimal weighting factors for good compression ratios are close to that for minimal distortion on lossy compression (table 3.3).

image	1	$\sqrt[8]{2}$	$\sqrt[4]{2}$	$\sqrt[8]{2}^3$	$\sqrt{2}$	Chebyshev
baboon	6.25	6.23	6.21	**6.20**	6.21	6.24
goldhill	5.06	5.06	**5.05**	5.06	5.07	5.13
lena noisy	8.92	8.80	**8.67**	8.69	8.69	8.82
lena	4.56	4.57	4.56	**4.56**	4.57	4.68
mountain	7.38	7.33	7.29	7.27	**7.26**	7.30
parrot	**4.44**	4.48	4.51	4.50	4.56	4.77
pepper	4.97	4.94	4.92	4.90	**4.90**	4.96
sar_pripuls	7.21	7.18	7.15	**7.15**	7.15	7.18

Table 3.4: Weighting and CHEBYSHEV wavelet – Can the amplitude of different transformation levels be weighted in a way that increases compression efficiency? – After applying a CDF-2,2 it is weighted by the factors 1, $\sqrt[8]{2}$, $\sqrt[8]{4}$, $\sqrt[8]{8}$, $\sqrt[8]{16}$, respectively. The pack rates for the CHEBYSHEV wavelet are given for comparison in the last column. Transformation is done over 6 levels. Thereafter every image is lossless compressed with EZT method. The bits per pixel rate achieved is presented here.

Approximative solution: Due to the convexity of $f(\beta)$, there are not many difficulties for implementing an iterative algorithm for finding the minimum, since there is only one. The only problem is the big computation effort, because all maxima of a polynomial have to be determined only to compute one value of $f(\beta)$. Finding the maxima of a polynomial again requires an iteration process for finding the zeros of the derivative. This means a nested iteration, which is pretty slow. Nevertheless, this is the current implementation in our software package. For speedup, further implementations could reuse the zeros found in one step as start approximations in the next step.

In the first phase the algorithm tries to find a lower and an upper bound for the minimum β. $f(0.5), f(1), f(2)$ are calculated. If they form a falling sequence it is continued with computing values to the right: $f(4), f(8), f(16), \dots,$ if they form a rising sequence it is searched to the left: $f(0.25), f(0.125), f(0.0625), \dots$. When a β is reached where $f(\beta) < f(\frac{\beta}{2}) \ \wedge \ f(\beta) < f(2\beta)$ then $\frac{\beta}{2}$ and 2β are obviously bounds for the minimum.

The second phase narrows the bounds step by step. Starting with the bounds β_- and β_+ and a β_0 between them, some new β within the interval are guessed and a triple $\beta'_-, \beta'_0, \beta'_+$ of neighboured points with $f(\beta'_0) < f(\beta'_-) \ \wedge \ f(\beta'_0) < f(\beta'_+)$ is selected and passed to the next iteration step. The points between are guessed as follows:

1. a point between β_- and β_0: $\frac{1}{2}(\beta_- + \beta_0)$

2. a point between β_0 and β_+: $\frac{1}{2}(\beta_0 + \beta_+)$
3. the vertex of the parabola laid through $(\beta_-, f(\beta_-)), (\beta_0, f(\beta_0)), (\beta_+, f(\beta_+))$

The iteration is aborted when $|\beta_+ - \beta_-|$ appear to be below a given threshold.

Weight by lifting: How can we weight in a way that allows perfect reconstruction, even when computing with integers? [4] shows that weighting can also be done by lifting with four lifting steps and that it is possible to choose the direction of the first lifting step. Additionally, it is pointed out that the weighting is often part of a longer lifting sequence and it is suggested to choose the start direction so that the first or last lifting step of the weighting can be joined with previous or subsequent lifting steps. This way one can always reduce the weighting lifting sequence to three steps. We will leave out this optimization and we will see, that the choice of the start direction is also a matter of computation accuracy.

Let us follow an approach to determine four lifting steps which can perform the weighting process. Each lifting step is represented by its polyphase matrix. The way the polyphase matrix is defined we have to order the lifting step matrices from the right to the left. We start with lifting from the L band to the H band. If starting with lifting in reverse direction is preferred, lifting steps can be designed to weight by $\frac{1}{\alpha}$ and then the direction of every lifting step must be reversed, too.

$$
\begin{aligned}
\begin{pmatrix} \alpha & 0 \\ 0 & \frac{1}{\alpha} \end{pmatrix} &\overset{!}{=} \begin{pmatrix} 1 & d \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ c & 1 \end{pmatrix} \begin{pmatrix} 1 & b \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ a & 1 \end{pmatrix} \\
& \quad \Big| \quad \alpha \cdot \begin{pmatrix} 1 & -d \\ 0 & 1 \end{pmatrix} \cdot () \cdot \begin{pmatrix} 1 & 0 \\ -a & 1 \end{pmatrix} \\
\begin{pmatrix} 1 & -d \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \alpha^2 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ -a & 1 \end{pmatrix} &= \alpha \begin{pmatrix} 1 & b \\ c & 1 + bc \end{pmatrix} \\
\begin{pmatrix} \alpha^2 & -d \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ -a & 1 \end{pmatrix} &= \alpha \begin{pmatrix} 1 & b \\ c & 1 + bc \end{pmatrix} \\
\begin{pmatrix} \alpha^2 + ad & -d \\ -a & 1 \end{pmatrix} &= \alpha \begin{pmatrix} 1 & b \\ c & 1 + bc \end{pmatrix}
\end{aligned}
$$

Comparison of matrix entries yields:

$$\alpha^2 + ad = \alpha \tag{3.2.4}$$

$$-d = \alpha b \tag{3.2.5}$$

$$-a = \alpha c \tag{3.2.6}$$

$$1 = \alpha(1 + bc) \tag{3.2.7}$$

But we observe, that if we insert (3.2.5) and (3.2.6) into (3.2.4)

$$
\begin{aligned}
\alpha^2 + \alpha^2 bc &= \alpha \\
\alpha(1 + bc) &= 1
\end{aligned}
$$

we obtain, that the equation (3.2.7) depends on the three other equations formed by the matrix equation. Does this mean that we can save one of the lifting steps?

If we remove one of the lifting steps associated with b or c ($b = 0$ or $c = 0$, respectively), the steps for a and c as well as b and d could be joined, which would force the other coefficients to be zero and $\alpha = 1$. If we remove one of the lifting steps associated with a or d (set $a = 0$ or $d = 0$) it follows immediately that $c = 0$ and $b = 0$, respectively and in both cases $\alpha = 1$. Summarized, this means that four lifting steps are necessary for weighting (different from $\alpha = 1$) and there remains one degree of freedom. What can this parameter be used for?

Minimize rounding errors: If we choose $a = 1$ we obtain the lifting steps proposed in [4]:

$$\left(1, \quad \alpha - 1, \quad -\frac{1}{\alpha}, \quad \alpha - \alpha^2\right) \tag{3.2.8}$$

But is this sensible? What about numerical properties?

Let us recall that we need the lifting scheme primarily for integer calculation. For floating point arithmetic direct multiplication is faster and probably more accurate. With integer calculation the main advantage of the lifting used for weighting is the possibility of exact reconstruction. But how can this be achieved?

Imagine, we have two numbers x and y which have to be weighted by a factor c, i.e. we want to calculate $c \cdot x$ and $\frac{1}{c} \cdot y$. Say, c is a shift coefficient $c = 2^n, n > 0$, then $c \cdot x$ has n trailing zeros in its binary representation, whereas y is shifted n bits to the right and its n trailing bits are lost. But the lifting operations are reversible! That means that the weighting performed by lifting will lead to results different from those obtained by exact weighting with subsequent rounding in general. Weighting by lifting will usually calculate approximations that are worse than the integer rounding error which can be at most 0.5. Thus it is of interest how these rounding errors can be minimized.

We want to get an error estimation which depends on the filters but not on the signal we will process. Asking for an absolute error will fulfill this. We will use an ε which describes the computing precision. Because we work with integers, ε varies between -0.5 and 0.5 – the maximum absolute rounding error possible. Each time we add a value of the filtered source band to the destination band, we have to increment the error by one ε because of rounding occurs. Of course, we always use the absolute values of scalars that has to be multiplied with the error accumulated so far.

For simplification we choose $\alpha > 0$ and because for every lifting sequence (a, b, c, d), which weights by α, the sequence $(-a, -b, -c, -d)$ does it as well, we can choose $b \geq 0$ without loss of generality. To avoid worrying about absolute values, we check with (3.2.5) and (3.2.6) that it is always true that $\operatorname{sgn}(a) = -\operatorname{sgn}(c)$, $\operatorname{sgn}(b) = -\operatorname{sgn}(d)$ and $d \leq 0$ because of the choice of b. Because $\alpha(1 + bc) = 1, 1 + bc = \frac{1}{\alpha}$ it must be $\alpha \geq 1$ if and only if $c \leq 0$. We will use the shortcut

$$\sigma = \begin{cases} 1 & : \alpha \geq 1 \\ -1 & : \alpha < 1 \end{cases}$$

to be able to process everything together. It is sure now that $b, -d, -\sigma c, \sigma a$ are all non-negative. Now let us explore what happens if we apply the lifting steps respecting rounding:

$$
\begin{array}{rcl}
\text{L band} & & \text{H band} \\
0 & & 0 \\
 & \cdot a \rightarrow & \\
0 & & \varepsilon \\
 & \leftarrow b\cdot & \\
(1+b)\varepsilon & & \varepsilon \\
 & \cdot c \rightarrow & \\
(1+b)\varepsilon & & (1+1-\sigma c\cdot(1+b))\varepsilon \\
 & \leftarrow d\cdot & \\
(2+b-d\cdot(2-\sigma c\cdot(1+b)))\varepsilon & & (2-\sigma c\cdot(1+b))\varepsilon
\end{array}
$$

The sum of both errors is

$$
\begin{aligned}
& (4+b-2d+\sigma c(b+1)(d-1))\,\varepsilon \qquad \Big| \quad (3.2.5) \ \wedge \ (3.2.7) \\
= & \left(4+b+2\alpha b+\sigma\frac{1}{b}(\frac{1}{\alpha}-1)(b+1)(-\alpha b-1)\right)\varepsilon \\
= & \left(4+(1+2\alpha)b+\sigma(1-\frac{1}{\alpha})(1+\frac{1}{b})(\alpha b+1)\right)\varepsilon \\
= & \left(4+(1+2\alpha)b+\left|1-\frac{1}{\alpha}\right|\left((1+\alpha)+\frac{1}{b}+\alpha b\right)\right)\varepsilon \\
= & \left(4+\left|\alpha-\frac{1}{\alpha}\right|+\underbrace{\left|1-\frac{1}{\alpha}\right|}_{K:=}\frac{1}{b}+\underbrace{(1+2\alpha+|\alpha-1|)b}_{L:=}\right)\varepsilon
\end{aligned}
$$

$$
K=\begin{cases}\frac{\alpha-1}{\alpha} & :\alpha\ge 1\\ \frac{1-\alpha}{\alpha} & :\alpha<1\end{cases} \qquad\qquad L=\begin{cases}3\alpha & :\alpha\ge 1\\ 2+\alpha & :\alpha<1\end{cases}
$$

$$
\begin{aligned}
\frac{K}{b}+Lb &= \left(\sqrt{\frac{K}{b}}-\sqrt{Lb}\right)^2+2\sqrt{KL} \\
&\ge 2\sqrt{KL} \qquad \text{equality for } b=\sqrt{\frac{K}{L}} \qquad (3.2.9)\\
&= \begin{cases}2\sqrt{3(\alpha-1)} & :\alpha\ge 1\\ 2\sqrt{\frac{(1-\alpha)(2+\alpha)}{\alpha}} & :\alpha<1\end{cases}
\end{aligned}
$$

We have observed that we can either weight by α or weight by $\frac{1}{\alpha}$ with changed roles of the H band and L band to achieve the same. That means that we can restrict the weighting method for either $\alpha \geq 1$ or $\alpha \leq 1$ dependent on the higher precision.

$$
\begin{aligned}
(\alpha-1)^2 &\geq 0 \\
\alpha^2-2\alpha+1 &\geq 0 \\
-3\alpha+3 &\geq -\alpha^2-\alpha+2 \\
3(1-\alpha) &\geq (1-\alpha)(2+\alpha) \\
3\left(\frac{1}{\alpha}-1\right) &\geq \frac{(1-\alpha)(2+\alpha)}{\alpha} \\
4+\left|\frac{1}{\alpha}-\alpha\right|+2\sqrt{3\left(\frac{1}{\alpha}-1\right)} &\geq 4+\left|\alpha-\frac{1}{\alpha}\right|+2\sqrt{\frac{(1-\alpha)(2+\alpha)}{\alpha}} \qquad \Bigg| \quad \alpha' := \frac{1}{\alpha} \\
\underbrace{4+\left|\alpha'-\frac{1}{\alpha'}\right|+2\sqrt{3(\alpha'-1)}}_{\text{error estimation for } \alpha'\geq 1} &\geq \underbrace{4+\left|\alpha-\frac{1}{\alpha}\right|+2\sqrt{\frac{(1-\alpha)(2+\alpha)}{\alpha}}}_{\text{error estimation for } \alpha\leq 1}
\end{aligned}
$$

We see that the error is smaller when we choose weighting by $\alpha \leq 1$ instead of weighting by $\alpha' = \frac{1}{\alpha} \geq 1$ in the reverse direction. So we will continue with determining the lifting steps for $\alpha \leq 1$:

$$
\begin{aligned}
(3.2.9) \Rightarrow b &= \sqrt{\frac{1-\alpha}{(2+\alpha)\alpha}} \\
(3.2.7) \Rightarrow bc = \frac{1-\alpha}{\alpha} \Rightarrow c &= \sqrt{\frac{(2+\alpha)(1-\alpha)}{\alpha}} \\
(3.2.5) \Rightarrow d &= -\sqrt{\frac{(1-\alpha)\alpha}{2+\alpha}} \\
(3.2.6) \Rightarrow a &= -\sqrt{(2+\alpha)(1-\alpha)\alpha}
\end{aligned}
\tag{3.2.10}
$$

Examples for weighting by lifting: First, let us verify if there is some obvious advantage of this solution over the one given in (3.2.8). We consider the case $\alpha = 1$. You might argue, that the lifting step sequence can be compressed to length zero for both solutions. But we imagine that we are working with values of α only close to 1, where such compressions are not possible. The lifting sequences obtained by both methods are:

- (3.2.8) – traditional: $(1, \quad 0, \quad -1, \quad 0)$
- (3.2.10) – error minimized: $(0, \quad 0, \quad 0, \quad 0)$

It can be seen easily, that the second variant really does not do anything, whereas the first variant adds the L band to the H band and subtracts it immediately, again. We can see that the error minimizing algorithm tends to use smaller lifting factors which should decrease the influences of the bands to each other.

The weighting by $\sqrt[4]{2}$ as post-processing of the CDF-2,2 wavelet has shown former in this section, that it optimizes the norm bounds of the CDF-2,2 wavelet. It should serve as a second example for a lifting decomposition here. Since $\sqrt[4]{2} > 2$ we have to work with $\alpha = \frac{1}{\sqrt[4]{2}}$ and we can obtain the coefficients which minimizes rounding errors:

$$\begin{array}{rcrcr} a & \approx & -0.617 & \approx & -\frac{158}{256} \\ b & \approx & 0.258 & \approx & \frac{66}{256} \\ c & \approx & 0.733 & \approx & \frac{188}{256} \\ d & \approx & -0.217 & \approx & -\frac{56}{256} \end{array}$$

The fractions with a power of two as denominators are of interest for machine-oriented implementations.

The lifting sequence is:

$$\begin{array}{ccc} \text{L band} & & \text{H band} \\ & \leftarrow -0.617\cdot & \\ & \cdot 0.258 \rightarrow & \\ & \leftarrow 0.733\cdot & \\ & \cdot - 0.217 \rightarrow & \end{array}$$

With this lifting sequence we now have a method to apply the weighted CDF-2,2 wavelet to a signal in a fully reversible form, even if rounding errors occur.

3.2.2 Update lifting steps

We will now explore how the norm bounds can be narrowed by appending an update lifting step.

Updating problem is convex, too: We consider the filters h_e, h_o, g_e, g_o before and the filters h'_e, g'_e after the lifting step with the lifting filter s:

$$\begin{aligned} h'_e &= h_e + g_e s \\ h'_o &= h_o + g_o s \\ 2p'(s) &= \overline{h'_e} h'_e + \overline{h'_o} h'_o + \overline{g_e} g_e + \overline{g_o} g_o \\ &= 2p + \underbrace{(\overline{h'_e} g'_e + \overline{h'_o} g'_o)}_{v:=} s + (h'_e \overline{g'_e} + h'_o \overline{g'_o}) \overline{s} + \underbrace{(\overline{g_e} g_e + \overline{g_o} g_o)}_{u:=} \overline{s} s \end{aligned}$$

We want to verify, that p' is convex regarding s:

$$
\begin{aligned}
&2p'(ts_0 + (1-t)s_1) \\
&\quad = \; 2p + v(ts_0 + (1-t)s_1) + \overline{v(ts_0 + (1-t)s_1)} + \\
&\quad\quad\;\; u\overline{(ts_0 + (1-t)s_1)}(ts_0 + (1-t)s_1) \\
&\quad = \; 2p + v(ts_0 + (1-t)s_1) + \overline{v(ts_0 + (1-t)s_1)} + \\
&\quad\quad\;\; u\left(\overline{ts_0}ts_0 + \overline{(1-t)s_1}(1-t)s_1 + \overline{ts_0}(1-t)s_1 + ts_0\overline{(1-t)s_1}\right) \\
&\quad = \; (1-t)\cdot 2p + (1-t)vs_1 + (1-t)\overline{vs_1} + (1-t)^2u\overline{s_1}s_1 + \\
&\quad\quad\;\; t\cdot 2p + tvs_0 + t\overline{vs_0} + t^2u\overline{s_0}s_0 + t(1-t)u\left(\overline{s_0}s_1 + s_0\overline{s_1}\right) \\
&\quad = \; t\cdot 2p'(s_0) + (1-t)\cdot 2p'(s_1) + (t^2-t)\overline{s_0}s_0 + (t^2-t)\overline{s_1}s_1 + t(1-t)(\overline{s_0}s_1 + s_0\overline{s_1}) \\
&\quad = \; t\cdot 2p'(s_0) + (1-t)\cdot 2p'(s_1) + \underset{\geq 0}{t}\,\underset{\leq 0}{(t-1)}\underbrace{\overline{(s_0-s_1)}(s_0-s_1)}_{\geq 0} \\
&\quad \leq \; t\cdot 2p'(s_0) + (1-t)\cdot 2p'(s_1)
\end{aligned}
$$

Approximative solution: The algorithm implemented in our software package for determining an update lifting step works as follows:

Start with $s = 0$. In every iteration cycle a direction Δs with a given norm (increment) is guessed. If $\|p'(s+\Delta s)\|_\infty < \|p'(s)\|_\infty$, it is assumed that further following this direction is successful, the increment is increased and a further step towards this direction is tried. If the step does not reduce $\|p'\|_\infty$, the increment is decreased and the step is canceled. This is repeated until the increment falls below a given value.

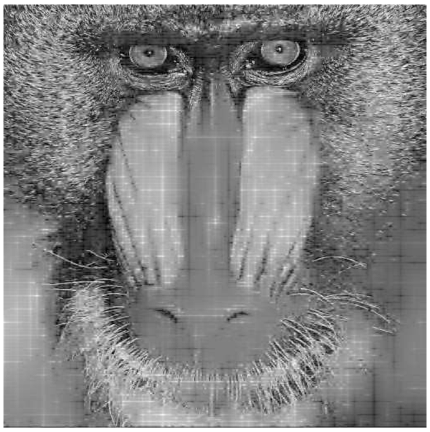
image	CDF 2,2	2	4	10	weight
baboon	**6.25**	6.29	6.30	6.32	6.28
goldhill	**5.06**	5.08	5.09	5.11	5.10
lena noisy	8.92	5.74	5.74	5.68	**5.66**
lena	**4.56**	4.58	4.57	4.60	4.64
mountain	**7.38**	7.41	7.43	7.46	7.38
parrot	**4.44**	4.61	4.62	4.73	4.66
pepper	**4.97**	4.99	5.02	5.05	5.10
sar_pripuls	7.21	7.23	7.25	7.25	**7.18**

Table 3.5: Improving the norm bounds – How do further steps which improve the norm bounds influence the pack rate? – The CDF-2,2 is compared to automatically generated wavelets, each consisting of a symmetric least-mean-square predictor of size 6 and an update lifting step of a symmetrical filter of size 2, 4, 10, respectively and the same predictor combined with a norm minimizing weighting. Transformation is done over 6 levels. Thereafter every image is lossless compressed with `EZT` method. The bits per pixel rate achieved is presented here.

The results in table 3.5 are generated with lifting step sequences which summarize the methods developed in this chapter. The first lifting step reduces the values on H band by linear least mean square prediction, the second lifting step narrows the norm bounds. For comparison the second step is substituted with optimized weighting in the last column.

The artifacts introduced to the image when working with least mean square predictors and lossy compression are shown in figure 3.9. The distortions of the CDF-2,2 appear quite soft, whereas the artifacts of the least mean square predictor are peaks and grids. The artifacts of the least mean square predictor look comparable for different update lifting steps.

CDF-2,2



Least mean square prediction

Figure 3.9: Artifacts caused by linear prediction wavelets – The left image is transformed with an unweighted CDF-2,2 wavelet, the right image is transformed with a 10 tap symmetric least mean square predictor and a 4 tap norm minimizing update. Transformation is repeated over 6 levels. Thereafter the transformed images are compressed to 0.5 bpp and decompressed again. The results are shown here.

Chapter 4

Summary

4.1 Results and perspectives

Let us summarize expectations and results found in this work and how the exploration could be continued:

- **Operator norm of a wavelet transformation**

  In section 2.1.2 we derived a method for determining the exact EUCLIDean operator norm of a filter matrix application which is equivalent to a wavelet transformation step. Further explorations may result in a method that can compute the norm of a complete wavelet transformation with arbitrary number of levels.

  In the case of two bands we could simplify the method in a way that even allowed optimization of the norm bounds. This was done by introducing the p polynomial in (2.1.12). In the case of more than two bands an analogous simplification is not obvious. Further research may result in the needed simplification or in optimization approaches that work without it.

- CHEBYSHEV **wavelets**

  In section 2.2 we made an approach for symmetric wavelets that have naturally close norm bounds. For the case of the filter lengths (5,3) we have seen, that there are really wavelets that fulfill the requirements. The one that performs linear interpolation as the CDF-2,2 leads to compression rates comparable with CDF-2,2 but no superior ones.

  For CHEBYSHEV wavelets with longer filters we found a basic structure in section 2.2.4. By the way, CHEBYSHEV wavelets exist at other lengths, too, but it is more difficult to assign them additional properties.

  Future explorations may find other applications, where the outermost filter coefficients are given and symmetrical wavelet filters with close norm bounds are requested.

- **Linear prediction**

  It was expected that we can improve the compression rate considerably when using image dependent linear prediction (section 3.1.3) instead of predicting with predefined filters.

  As we can see, the plain prediction does not increase the compression efficiency at the expected extent, in general. The filters generated do not differ very much from a linear interpolating one and the energy is decreased slightly better compared to the decrease achieved with a static linear interpolation. Conversely this means, that the linear interpolation is a good choice for most pictures.

An improvement of the compression can be achieved by a following update lifting step (see below), a technique that is widely used already for common wavelets. It has to be checked, if multiple adaptive prediction/update combinations can further increase compression efficiency.

A general disadvantage of wavelets designed by least mean square lifting is, that they do not fulfill any smoothness conditions. This may make compression artifacts more annoying.

A general advantage is: If we use a wavelet which is found through standard assumptions about the regularity of the input signal, this will fail on images which do not fulfill this assumptions. E.g. this is the case for noise. Since the least mean square optimization will never produce data with more energy than before, it is not possible to amplify noise with it. As soon as the algorithm "detects" noise, what means that a prediction is impossible, it will weaken the filter coefficients accordingly. This theoretical argument can be verified with the noisy satellite photography `sar_pripuls`. But as you can see, the compression rates are not mentionable better than with standard wavelets. The noise seems to be too smooth, so that the linear interpolation succeeds in the first levels, nevertheless.

One can do tests with more high frequent noise. Then the least mean square prediction performs better than e.g. the CDF-2,2, but it exceeds the limit of a rate of 8 bits per pixel. Instead of a "compression" to a bit rate above 8 bits, one would better store the original image in this case.

A type of image which can be processed very good by least mean square prediction are images with very regular patterns superposed. Such patterns could originate e.g. from a coarse printing.

- **Weighting**

  The compression results show that weighting the wavelet filters as in section 3.2.1 can increase the compressibility of a transformed image. The visual comparison of images that are lossily compressed with plain CDF-2,2 and optimally weighted CDF-2,2 proves the better properties of a weighted CDF-2,2 wavelet.

  For general wavelet pairs we know, how to decide if the minimum of the wavelet operator norm with regard to the weighting factor is smooth or not. If it is smooth, it can be located. A method for locating a non-smooth minimum has to be derived, yet. At least a faster iteration which is not nested, should be possible. An alternative may be interleaved iterations.

  Again, we may ask for optimal weightings for filter matrices larger than 2×2.

- **Norm minimizing update**

  An additional update lifting step was introduced in section 3.2.2 for reducing the norm bounds of the filter operation. Weighting the filters (see above) has shown, that minimizing the norm bounds may also lead to better pack results. The benchmarks show that this is true for updating steps, too, but the effect is smaller.

  As stated for the weighting problem, the optimization iteration has to be made faster in future or necessary conditions have to be found, that allow a restriction to a small set of candidates for the optimum. The current implementation needs too much computational power.

  Both least mean square prediction and norm minimizing update steps are methods, that do not increase the overall compression efficiency at an extent that would excuse the computational effort of the current implementation.

4.2 Implementation

The methods found in this work are implemented as extensions to the C++ program `ubk` by Jörg Ritter which was originally developed for testing the CDF-2,2 wavelet transformation with embedded zero tree compression, including transformation and compression in partitioned images. You can access these methods by specifying command line options when invoking the compression program. The most important options are:

- `-l` n
  Specifies the number of transformation levels.
  With the new functions it is possible to specify different wavelets at successive transformation levels. Each `-l` option starts a new block of wavelet specifications which are used for the next n transformation levels.

- `-lift` *type parameters*
  Specifies a lifting step.
  type selects one of the lifting step types listed below,
  parameters contains parameters specific to the lifting step type.

- `-c`
  Compress the transformed image with an embedded zero tree algorithm similar to the SPIHT algorithm of Said/Pearlman [14].

- `-bpp` *number_of_bits_per_pixel*
  Abort the bit stream of compressed data, as soon as the file reaches the size it had, if it would be saved uncompressed but with *number_of_bits_per_pixel* . That is the way the lossy compression with `EZT` works. The most relevant data is transferred first, the less relevant data is transferred last. If you abort the stream you obtain a good approximation of the original image.
  Note that the file size measurement does not include header information. Among other things, the header contains all coefficients of the used lifting filters, which consumes space of

$$O\left(J \cdot \sum_k (1 + |s_k|)\right)$$

  where J is the number of levels and s_k are the lifting filters within each level.

The type of wavelet is specified by its lifting step decomposition. The transformation is performed directly by applying these lifting steps. The lifting steps can be composed of custom filters or filters that are calculated at run-time. The types which can be passed to the `-lift` option are:

- `weight` *factor*
  Weight filters by a the scalar *factor*

- `weight minbound`
  Weight in order to minimize the difference between lower and upper transformation norm bound. See section 2 for explanation.
  The necessary factor is approximated at run time.

- `cdf` n, m
  The standard CDF-2,2 wavelet (developed by COHEN, DAUBECHIES, FEAUVEAU, see [2]) can be accessed with `cdf 2,2`, other wavelets of the CDF family are not implemented yet, but the option specification allows this future extension.

- `cheby` n, c
  CHEBYSHEV wavelet with a low pass filter h of $2n + 1$ taps and an outer coefficient c. Only members with $n = 2$ out of this family are supported currently, the ones we considered in section 2.2.2 in detail.

- `minenergy` n, `minenergysym` n
  Do a predicting lifting step which minimizes the energy on the H band
  The required lifting filter is determined after the previous steps has been processed. Arbitrary filter lengths n are supported and restriction to symmetric filters is possible with the `minenergysym` option.

- `minbound` n, `minboundsym` n
  Do an update lifting step which minimizes the difference between lower and upper wavelet norm bound.
  The previous lifting steps are merged to a filter pair, and the desired filter is approximated for that. Arbitrary filter lengths n and restriction to symmetric filters are allowed by `minboundsym`.

In the benchmark tables that compare pack rates, values are given that are average numbers of bits per pixel. This does not match exactly the mathematical definition of the first order entropy, which can be calculated from the histogram of the image. The pack rates given are the ones achieved with `EZT` compression and the measurement is the same as for the `-bpp` option.

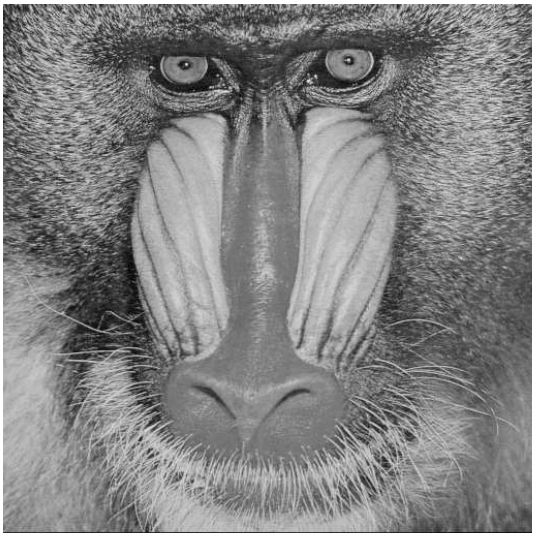
There is another parameter which can not be influenced by command line arguments: The type of numbers used for the transformation and image storage. The number type can be switched between integers and floating point numbers, but switching requires recompilation. Working with floating point numbers gives better packing results (about 0.3 bpp improvement are possible). But you can not compare the compression rates of the floating point and the integer arithmetic directly, because a transformed image with floating point values can not be restored exactly after compression. To keep the option of lossless compression, all benchmarks are made with integer transformations only.

The computations are supported by the Template Numerical Toolkit `TNT` [11] which provides functions for doing Linear Algebra in C++. This includes management of matrices and vectors of arbitrary types, basic matrix/vector operations and various matrix factorization algorithms (LU, CHOLESKY, QR decomposition).

Appendix A

Test images

Here are the images used for compression tests. Some of them were clipped to a size of a power of two in respect of the limitations of the current implementation of the `EZT` pack algorithm.



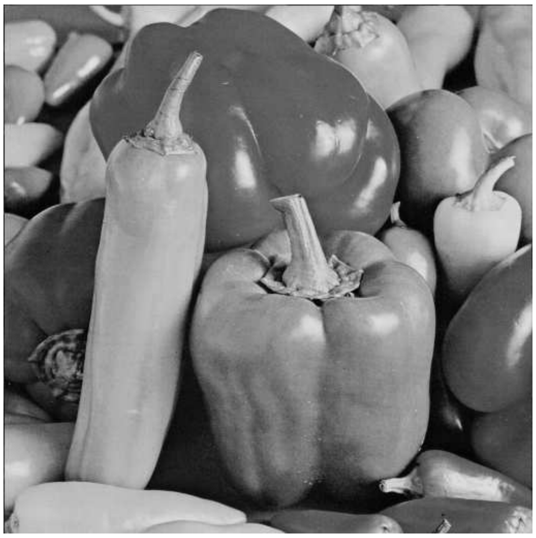
baboon, 512×512 pixel

goldhill, 512×512 pixel


lena, 512 × 512 pixel


lena noisy, 512 × 512 pixel


sar_pripuls, 512 × 512 pixel


pepper, 512 × 512 pixel


mountain, 512 × 256 pixel


parrot, 256 × 128 pixel

List of Tables

List of Figures

Bibliography

[1] E.J. Barbeau. *Polynomials*. Problem books in mathematics. Springer, 2nd edition, 1995.

[2] A. Cohen, I. Daubechies, and J.C. Feauveau. Biorthogonal bases of compactly supported wavelets. *Communications on Pure and Applied Mathematics*, 45:485–560, 1992.

[3] I. Daubechies. *Ten Lectures on Wavelets*. Number 61 in CBMS/NSF Series in Applied Math. SIAM, 1992.

[4] I. Daubechies and W. Sweldens. Factoring wavelet transforms into lifting steps. *J. Fourier Anal. Appl.*, 4(3):245–267, 1998. `http://cm.bell-labs.com/who/wim/papers/papers.html#factor`.

[5] `http://bmrc.berkeley.edu/courseware/cs294/fall97/assignment/psnr.html`.

[6] Richard W. Hamming. *Digital Filters*. Signal Processing Series. Prentice–Hall, Englewood Cliffs, New Jersey 07632, second edition, 1977, 1983.

[7] J. Kovačević and W. Sweldens. Wavelet families of increasing order in arbitrary dimensions. *IEEE Transactions on Image Processing*, 9(3):480–496, March 2000. `http://cm.bell-labs.com/who/wim/papers/mdlift/index.html`.

[8] G.R. Kuduvalli and R.M. Rangayyan. Performance analysis of reversible image compression techniques for high-resolution digital teleradiology. *IEEE Transactions on Medical Imaging*, 11:430–445, September 1992.

[9] D. Marpe, H. Cycon, and W. Li. Energy constraint scarce wavelet packet libraries for image compression, January 1997.

[10] D. Marpe, H. Cycon, and W. Li. A complexity constraint best-basis wavelet packet algorithm for image compression. *IEEE Proceedings-Vision, Image and Signal Processing*, 1998.

[11] Roldan Pozo. Template Numerical Toolkit, August 2000. `http://math.nist.gov/tnt/`.

[12] J. Ritter. A pipelined architecture for partitioned DWT based lossy image compression using FPGAs. *Ninth ACM International Symposium on Field Programmable Gate Arrays*, February 2001. `http://nirvana.informatik.uni-halle.de/~ritter/`.

[13] S. Saha and R. Vemuri. Adaptive wavelet coding of multimedia images. *Proc. ACM Multimedia*, 1999. Orlando, Florida.

[14] A. Said and W. Pearlman. A new fast and efficient image codec based on set partitioning in hierarchical trees. *IEEE Transactions on Circuits and Systems for Video Technology*, 6, June 1996.

[15] J.M. Shapiro. Embedded image coding using zerotrees of wavelet coefficients. In *Trans. Signal Processing*, volume 11, pages 3115–3162. IEEE, 1993.

[16] W. Sweldens. The lifting scheme: A custom-design construction of biorthogonal wavelets. *Appl. Comput. Harmon. Anal.*, 3(2):186–200, 1996. `http://cm.bell-labs.com/who/wim/papers/papers.html#lift2`.

[17] W. Sweldens. Wavelets and the lifting scheme: A 5 minute tour. *Z. Angew. Math. Mech.*, 76 (Suppl. 2):41–44, 1996.

[18] H. Thielemann. Lifting als Konstruktionsmethode für Wavelets sowie schnelle Berechnungsvorschrift (lifting used for constructing wavelets, as well as for fast computation). *Seminar Datenkompression*, December 1999. `http://nirvana.informatik.uni-halle.de/~ritter/Seminar19992000/seminar.html`.

[19] R. Weiner. Numerische Mathematik I,II (Numerical mathematics), 1998-1999. `http://www.mathematik.uni-halle.de/~numerik/`.