International Journal of Wavelets, Multiresolution and Information Processing

DISCRETE WAVELET TRANSFORM OF FINITE SIGNALS: DETAILED STUDY OF THE ALGORITHM

PAVEL RAJMIC

Dept. of Telecommunications, FEEC, Brno University of Technology
Purkyňova 118, 602 00 Brno, Czech Republic
rajmic@feec.vutbr.cz

ZDENEK PRUSA

Acoustics Research Institute, Austrian Academy of Sciences,
Wohllebengasse 12–14, 1040 Wien, Austria
zdenek.prusa@oeaw.ac.at

Received (Day Month Year)
Revised (Day Month Year)
Communicated by (xxxxxxxxxx)

The paper presents a detailed analysis of algorithms used for the forward and the inverse discrete wavelet transform (DWT) of finite-length signals. The paper provides answers to questions such as "how many wavelet coefficients are computed from the signal at a given depth of the decomposition" or conversely, "how many signal samples are needed to compute a single wavelet coefficient at a given depth of the decomposition" or "how many coefficients at a given depth are influenced by the selected type of boundary treatment" or "how many samples of the input signal simultaneously influence two neighbouring wavelet coefficients at a given depth of the decomposition". As a byproduct, the rigorous analysis of the algorithms gives details needed for the implementation. The paper is accompanied by several Matlab functions.

Keywords: DWT; wavelet transform; analysis; study; algorithm; details; computation of coefficients; boundary handling; border effect; blocking effect

1. Introduction

In the paper, a detailed analysis of the Discrete-time Wavelet Transform (DWT) is presented. It consists of the description of the algorithms for computing the forward and the inverse transforms, including details that are useful for implementation, and, in particular, it contains a comprehensive analysis of the properties of the algorithms with a number of implications. The reader can find answers to questions such as

- How many wavelet coefficients are computed from the signal at different depths of decomposition?
- How many signal samples are needed to compute one single wavelet coefficient at the defined depth of decomposition?

- How many coefficients at a given depth of decomposition are affected by the choice of the type of the boundary treatment?
- How many signal samples simultaneously influence two neighbouring wavelet coefficients?
- Which particular wavelet coefficients are affected by the choice of the boundary treatment method?

Answers to the listed (and other related) questions are presented in the form of theorems and proofs. The results are depicted and commented in a number of graphs, examples, and tables. Several Matlab functions come with the paper to implement and demonstrate the theoretical findings. The reader is expected to be familiar with the basics of the wavelet signal decomposition [28,20,8,4].

Although the intent of the article could seem theoretical, some of the results can be readily transferred into practice. Consider, for example, the case when a real-time algorithm based on wavelet decomposition has to be implemented on a low level, e.g. in a microcontroller. In such a case the programmer definitely needs to know an exact count of the number of coefficients to store, etc. [24,13]

Only finite-length discrete-time signals are considered throughout the text. The coefficients of the wavelet filters used are assumed to be known; however, in most cases, their particular values will not be used, as long as the information about the filter length (i.e. the number of filter taps) is absolutely sufficient. The inverse transform will be denoted iDWT. We narrow our considerations to the *dyadic* DWT case only, which is the case when all the up- and downsample factors are equal to two.

In Section 1.1, most of the symbols used are established. Section 2 deals with the forward DWT. It consists of part 2.1 dealing with types of treatment used for signal extension at the boundaries (which is necessary for the finite-length, non-periodic signals), and of parts 2.2 and 2.3, which deal with two alternative formulations of the DWT computation: the first one being in the form of a matrix multiplication, the second using the recursive filtering "pyramid" (Mallat's algorithm). Section 2.4 is the most original and the most contributive part of the paper, containing a thorough analysis of the forward DWT algorithm. This part answers the questions stated above, among other things. Section 3 then brings a detailed analysis of the inverse iDWT in a similar manner. The last part is dedicated to the description of Matlab functions.

1.1. *Symbols and Assumptions*

The vectors are considered as rows, unless stated otherwise. Indexing the vector elements is assumed to start with the number one, i.e. $\mathbf{x} = [x_1, \ldots, x_n]$. The symbol $\mathrm{len}(\mathbf{x})$ denotes the length of vector $\mathbf{x}$, i.e. the number of its elements, for example $\mathrm{len}([x_1, \ldots, x_n]) = n$. When talking of the *effective length*, possible zeros at the borders are excluded and the effective length can thus be shorter. Formally, this is defined by $(\max\{k \,|\, x_k \neq 0\} - \min\{k \,|\, x_k \neq 0\} + 1)$ for nonzero vectors. Rounding

to the nearest integer will often be employed; $\lfloor x \rfloor$ represents rounding towards the negative infinity: $\lfloor x \rfloor = \max\{k \,|\, k \in \mathbb{Z}, k \leq x\}$. By analogy, $\lceil x \rceil$ rounds x towards the positive infinity.

In the context of the DWT algorithm, the following symbols need to be established in particular:

- $s \in \mathbb{N}$ represents the signal length,
- $J \in \mathbb{N}$ represents the depth of the decomposition,
- $m \in \mathbb{N}$ represents the length of the wavelet filters.

1.1.1. *Wavelet Filters*

Only the finite impulse response (FIR) filters are considered. Hence $m < \infty$. The decomposition low-pass filter is denoted by $\mathbf{h}$, the high-pass filter by $\mathbf{g}$, the reconstructing low-pass filter by $\tilde{\mathbf{h}}$, and the high-pass filter by $\tilde{\mathbf{g}}$.

The filters can be of both odd and even length. (Satisfying $m \geq 2$ at the same time, to make the filtering meaningful.) The so-called quadrature mirror filters, which are always orthogonal, have all four identical lengths. The biorthogonal filters nevertheless can have different effective lengths. According to [27], one of the following cases is true (both for the decomposition and the reconstruction stages):

(1) Both filters have odd lengths which differ by an odd multiple of two.
(2) Both filters have even lengths, being either equal or differing by an even multiple of two.
(3) One filter is of odd length, the other is of even length, and the zeros of both the filters are located at the unit circle.

To work with filters of different lengths consistently, the shorter one is zero-padded to the length of the longer one. Zeros, of course, do not affect the values at the output of the filter. (The "lifting scheme" [3,14] which would make use of the shorter length, is not exploited.) The rules for padding the shorter filter at both its ends follow immediately and they correspond with the Matlab Wavelet Toolbox [9] behavior.

In the following text only the first two of the cases mentioned are considered — case 3 is of no practical interest [27]. Two nonnegative variables l_0 and r_0 are defined, denoting the number of zeros to be added from the left and the right end, respectively. Denoting the effective length of the shorter filter by $\underline{m}$, the following naturally holds: $m = l_0 + \underline{m} + r_0$.

In case 1 (i.e. odd $m, \underline{m}$) the extensions are chosen so that $l_0 = r_0 - 2$, leading to

$$l_0 = \frac{m - \underline{m}}{2} - 1, \quad r_0 = \frac{m - \underline{m}}{2} + 1. \tag{1.1}$$

In case 2 (i.e. even $m, \underline{m}$) the extensions l_0, r_0 are equal, which induces

$$l_0 = r_0 = \frac{m - \underline{m}}{2}. \tag{1.2}$$

Whenever a particular wavelet filter is mentioned in the paper, its abbreviated labeling is taken over from [9].

Example 1.1. The biorthogonal filter bank `bior2.2` comprises the decomposing low-pass filter $\mathbf{h} = [h_1, \ldots, h_5]$ of length $m = \text{len}(\mathbf{h}) = 5$ and the high-pass filter $\mathbf{g} = [g_1, g_2, g_3]$ of effective length $\underline{m} = \text{len}(\mathbf{g}) = 3$. This corresponds to case 1, and according to (1.1), the extensions to be used are $l_0 = 0$ a $r_0 = 2$. Thus, the resultant padded high-pass filter is $[g_1, g_2, g_3, 0, 0]$.

Example 1.2. The biorthogonal filter bank `bior1.5`: the decomposing low-pass filter $\mathbf{h} = [h_1, \ldots, h_{10}]$ of length $m = \text{len}(\mathbf{h}) = 10$, the high-pass filter $\mathbf{g} = [g_1, g_2]$ of only the effective length $\underline{m} = \text{len}(\mathbf{g}) = 2$. Case 2 should be used now, and according to (1.2), the final extensions are $l_0 = r_0 = 4$. Thus the padded filter takes the form $[0, 0, 0, 0, g_1, g_2, 0, 0, 0, 0]$.

From now on, $\mathbf{h}$, $\mathbf{g}$, $\tilde{\mathbf{h}}$, and $\tilde{\mathbf{g}}$ will denote filters already extended to have an equal length m.

Remark 1.1. When filters of odd lengths are considered, there is one difference between the Matlab Wavelet Toolbox [9] and the process just described. The Wavelet Toolbox inserts an extra zero at the beginning of both the filters to make their lengths be always even.

1.1.2. *Discrete Linear Convolution*

The convolution is the core of the DWT algorithm. The result of the discrete linear convolution of two signals $\mathbf{x} = [x_1, \ldots, x_s]$ and $\mathbf{h} = [h_1, \ldots, h_m]$ is the vector $\mathbf{y}$ composed of elements at indexes $n \in \{1, \ldots, s + m - 1\}$, where the elements are defined by the formula

$$y_n = \sum_{k=0}^{m-1} x_{n-k} h_{1+k}. \tag{1.3}$$

Certain combinations of n and k in the formula push the index $n - k$ outside the support of $\mathbf{x}$. Naturally, such values (whose count is $m-1$ at both ends) are assumed to be zeros.

1.1.3. *Parity*

The *parity* of a number indicates whether the number is even or odd. For $x \in \mathbb{Z}$,

$$\text{par}(x) = \begin{cases} 0 & \text{for } x \text{ even}, \\ 1 & \text{for } x \text{ odd}. \end{cases} \tag{1.4}$$

Figuratively, the parity of a vector will indicate the parity of its length.

1.1.4. *Downsampling and Upsampling*

An important operation used within the DWT is the dyadic *downsampling* of a vector, which means removing "every other sample" from it. Such an operator will be denoted ${}^{\varepsilon}\!\downarrow 2$, where ε indicates the variant of the downsampling: $\varepsilon = 0$ for the so-called *even type*, when the removal starts with the first sample, and $\varepsilon = 1$ for the *odd type*. Their effect is best shown on examples: let $\mathbf{x} = [1, 2, 3, 4, 5]$ and $\mathbf{y} = [1, 2, 3, 4, 5, 6]$ be vectors, then $({}^{0}\!\downarrow 2)\mathbf{x} = [2, 4]$, $({}^{1}\!\downarrow 2)\mathbf{x} = [1, 3, 5]$, $({}^{0}\!\downarrow 2)\mathbf{y} = [2, 4, 6]$, $({}^{1}\!\downarrow 2)\mathbf{y} = [1, 3, 5]$.

Lemma 1.1. *Given a vector of length d, the ε-downsampled result consists of*

$$\frac{d}{2} + \left(\varepsilon - \frac{1}{2}\right) \mathrm{par}(d) \tag{1.5}$$

samples.

Proof. If d is even, $\mathrm{par}(d) = 0$, then both variants of decimation $\varepsilon = 0$ and $\varepsilon = 1$ produce $d/2$ samples. If d is odd, the decimation clearly leaves $\left\lfloor \frac{d}{2} \right\rfloor + \varepsilon$ samples as the result. Formula (1.5) connects both the results because

$$\frac{d}{2} + \left(\varepsilon - \frac{1}{2}\right) \mathrm{par}(d) = \begin{cases} \frac{d}{2} & \text{for } d \text{ even} \\ \left\lfloor \frac{d}{2} \right\rfloor + \varepsilon = \frac{d-1}{2} + \varepsilon & \text{for } d \text{ odd.} \end{cases} \tag{1.6}$$

□

Similarly, the *upsampling* operator inserts zeros "between every two samples" and is denoted ${}^{\varepsilon}\!\uparrow 2$. Its action on vector $\mathbf{z} = [2, 4]$ results in $({}^{0}\!\uparrow 2)\mathbf{z} = [2, 0, 4]$, $({}^{1}\!\uparrow 2)\mathbf{z} = [0, 2, 0, 4, 0]$. Let us point out that in the case of $\varepsilon = 1$ zeros are appended both at the beginning and at the end of the signal. The following clearly holds.

Lemma 1.2. *The vector produced by ε-upsampling of a vector of length d has*

$$2d - 1 + 2\varepsilon \tag{1.7}$$

samples.

Remark 1.2. Due to the presence of the dyadic down- and upsampling in the DWT algorithm, the powers of two will play the central role in the analysis.

Remark 1.3. For the same reason, the DWT is not a shift-invariant transform. Actually, it is a 2^J-shift-invariant transform, which means that shifting the signal by 2^J samples results simply in a shift in the coefficient domain. The referred property can sometimes cause problems in applications; when shift-invariance has to be kept, the stationary wavelet transform (also called the undecimated DWT) [11] has to be used. Its classical application for denoising is presented e.g. in [2].

At a certain stage of the article, our analysis will neglect the variant $\varepsilon = 1$. This, however, not goes against the generality, since any ε can be mimicked by simply shifting the signal before the transform is applied.

1.1.5. *Most important symbols and denotations*

s ..... length of the input signal
m ..... length of the wavelet filters
J, j ..... depth of decomposition, particular levels of decomposition, $j \leq J$
$\mathrm{len}(\cdot)$ ..... length of a vector
$\mathrm{par}(\cdot)$ ..... parity of the number, or parity of the length of a vector
$\lfloor\cdot\rfloor$, $\lceil\cdot\rceil$ ..... nearest integer downwards, upwards
$|\mathcal{A}|$ ..... number of elements in set $\mathcal{A}$
${}^{\varepsilon}\!\downarrow 2$, ${}^{\varepsilon}\!\uparrow 2$ ..... dyadic down- and upsampling
$\mathbf{g}, \mathbf{h}, \tilde{\mathbf{g}}, \tilde{\mathbf{h}}$ ..... wavelet filters: high-pass, low-pass decomposing, high-pass, low-pass reconstructing
$\mathbf{a}^{(j)}$, $a_k^{(j)}$ ..... vector of the approximation wavelet coefficients at level j, and its kth element
$\mathbf{d}^{(j)}$, $d_k^{(j)}$ ..... vector of the detail wavelet coefficients at level j; its kth element
$\mathbf{c}^{(j)}$, $c_k^{(j)}$ ..... vector of wavelet coefficients at level j (no distinction what kind); its kth element
$n_{\mathrm{coef}}(s, m, j, \varepsilon)$ ..... number of coefficients produced by the DWT; the amount is equal to both $\mathrm{len}(\mathbf{a}^{(j)})$ and $\mathrm{len}(\mathbf{d}^{(j)})$
$n_{\mathrm{coef}}(s, m, j)$ ..... substitutes $n_{\mathrm{coef}}(s, m, j, 0)$
$n_{\mathrm{coef}}^{\Sigma}(s, m, J, \varepsilon)$ ..... overall number of coefficients produced by the DWT
$n_{\mathrm{samp}}(q, j, m)$ ..... number of signal samples needed for the computation of q subsequent wavelet coefficients
$\mathrm{ran}(k, j, u, m)$ ..... range of the influence of a single wavelet coefficient
$n_{\mathrm{shift}}(k_1, k_2, j, m)$ ..... time-shift in the signal domain for two given coefficients
$n_{\mathrm{shared}}(k_1, k_2, j, u, m)$ .. number of coefficients at depth u that are shared by two specified coefficients from level j
$n_{\mathrm{affect}}(j, m)$ ..... number of coefficients affected by the choice of the method of boundary treatment.

2. Forward Discrete Wavelet Transform (DWT)

The theory of wavelet transform admits cases when the finite-energy signal can be of infinite length and, consequently, the number of wavelet coefficients can be also (countably) infinite. By contrast, the signal is in practice usually localized on some time interval. Moreover, most of the signals cannot be considered periodic as well.

Following these statements, the methods of signal boundary treatment are presented briefly in Section 2.1. Then in Section 2.2 the basic formulation of DWT via matrix multiplication is introduced. Such a formulation is in fact not important from the computational efficiency point of view but it helps to understand in depth the mechanism of DWT, which is essential for Section 2.3 and mainly for Section 2.4, which encompasses the promised analysis.

2.1. *Treatment of Signal Boundaries*

There is a question that immediately comes to mind when working with time-limited signals: Since the fundamental part of DWT is convolution and since convolution is known to exhibit "boundary artifacts", how should one compute the wavelet coefficients located "near the boundaries"?

Although it is not the main focus of this study, a summary of possible methods is presented in this section, which answer the above question. Let us say in advance that all of the approaches suffer from some shortcoming [20,9,22,1,5,26,16]. In this part of text we assume (without loss of generality) just a single level of the wavelet decomposition $J = 1$.

(1) *Using special border filters.* In this case, special filters are constructed for the signal samples in the neighbourhood of the borders. The signal is not extended in any way.

(2) *Assuming periodicity.* The signal is considered to be a single period of an infinite-length periodic signal. If, in addition, the signal length is even, then the total number of coefficients produced at the first level of decomposition is equal to the original number of samples.

(3) *Defining samples outside of the original domain.* The idea here is that the samples beyond the signal domain are extrapolated using a more or less suitable and/or a more or less computationally demanding method. It is convenient to divide the possibilities into several groups:

   (a) *Symmetrization/Mirroring.* The edge-samples are "mirrored". Such an approach brings "discontinuities" of the signals first difference. If symmetric filters (only the biorthogonal filters can be symmetric) are used, it is possible to make the DWT representation non-redundant (for redundancy/expansivity see below).

   (b) *Point-symmetric extension.* Using point-symmetry one can get rid of the discontinuity mentioned.

   (c) *Smooth extension using polynomials.* The method tries to "guess" samples outside of the signal domain using a polynomial of a specified order (kth order polynomial preserves the "continuity" of kth difference).

   (d) *Extending with zeros.* This is the simplest method — the signal is considered to be zero beyond its original borders.

(4) *Using samples from the neighbouring segment.* This approach is natural when the signal to be processed is in fact a time-limited portion cut from a longer signal. For example, in real-time speech processing the buffer holds 256 samples. This type of method reasonably uses samples directly from its neighbour(s) to extend the borders.

   In this sense, such a method could be considered a special case of group 3. Nevertheless it is listed separately because in the case of the decomposition depth being $J > 1$, the recursive nature of the DWT makes the necessary extension length greater when compared with the other methods. Such a

situation requires more detailed treatment and modification of DWT as can be found in [15,10,12,14,18], for example.

(5) *Cutting off.* The goal of this naive approach is to keep the wavelet representation non-redundant. The DWT computation is performed using any of the above methods and then the "border" coefficients are simply discarded. Therefore the reconstruction cannot be exact near the borders any more.

Each of the stated methods suffers from at least one shortcoming from the following list:

- the necessity of having special border filters (which is not effective algorithmically),
- the deviation from (bi)orthogonality of the transform,
- inexact reconstruction from the transform coefficients,
- redundancy (expansivity) of the wavelet representation, which means that the wavelet representation of a signal of length s will have the total number of coefficients slightly higher than s,
- bringing possible errors to the "other end" due to periodicity.

Hence, in choosing a method one always has to make a compromise.

We find the extension methods given under item 3 (and possibly 4) to be the most natural and the most generally utilizable in practice; such methods have only one drawback — expansivity. As mentioned in 3a, there exist special situations when expansivity does not appear — this is typical of image processing with biorthogonal filters, for example in JPEG2000 compression scheme, see [23,25]. The expansivity can also be avoided in cases 3d and 3c ($k = 1$) for particular signal lengths via pre-processing near-border coefficients before the inverse transform [21]. Because of its universality, the analysis in Section 2.4 considers almost exclusively the generally expansive case 3.

In the context of this part other questions might arise, hand in hand with each other:

- What is the difference between the types of extension with respect to the effect on the values of the wavelet coefficients?
- How many boundary coefficients are affected by the choice of the method?
- If these coefficients were altered, how many signal samples would be affected after the reconstruction?

Answers to these questions can be found later in Sec. 2.4.5.

2.2. *DWT as Matrix Multiplication*

For convenience, both the signals and sets of coefficients will be treated as column vectors in this section. The DWT of a signal of length $s = 2^J$ can be formulated as a matrix multiplication [29,26] on the assumption of the periodic-type border extension. This is in fact the same as if $\mathbf{x}$ was s-periodic.

Denoting $\mathbf{x} = [x_1, \ldots, x_s]^\top$ the vector to be decomposed, it is possible to repre-

sent it as a linear combination of "wavelet" vectors arranged as column vectors of matrix $\mathbf{W}$ of size $s \times s$; the weights in the linear combination are the elements of $\mathbf{y} = [y_1, \ldots, y_s]^\top$,

$$\mathbf{x} = \mathbf{W}\mathbf{y}. \tag{2.1}$$

The columns of $\mathbf{W}$ form the basis in $\mathbb{R}^s$, thus $\mathbf{y}$ represents the coordinates of $\mathbf{x}$ in the basis, given by

$$\mathbf{W}^{-1}\mathbf{x} = \mathbf{y}, \tag{2.2}$$

which corresponds to finding the coordinates $\mathbf{x}$ in the dual basis for $\mathbf{y}$. In the orthogonal case $\mathbf{W}^\top = \mathbf{W}^{-1}$ holds right, meaning that the original and the dual bases are identical.

The DWT matrix for decomposition depth $J = 1$ can be written as

$$\mathbf{W}^{-1} = \begin{bmatrix} \mathbf{H}_1 \\ \mathbf{G}_1 \end{bmatrix}. \tag{2.3}$$

Here the rows of $\mathbf{H}_1$ and $\mathbf{G}_1$ are constructed using vectors $\mathbf{h} = [h_1, \ldots, h_m]$ and $\mathbf{g} = [g_1, \ldots, g_m]$, respectively, in a "circulant way": each row is a copy of the previous row except that it is shifted to the right by two positions (the shift comes from the downsampling). The size of both the matrices is $\gamma_j \times 2\gamma_j$, where $\gamma_j = 2^{-j}s$. The construction of $\mathbf{H}_1$ is shown here:

$$\mathbf{H}_j = \begin{bmatrix} \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \\ \cdots & h_m & h_{m-1} & h_{m-2} & \cdots & h_1 & 0 & 0 & \cdots \\ \cdots & 0 & 0 & h_m & \cdots & h_3 & h_2 & h_1 & \cdots \\ & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \end{bmatrix}, \tag{2.4}$$

and $\mathbf{G}_j$ is built using $\mathbf{g}$ in a similar way.

The rows of the matrices contain time-reversed and shifted versions of the filters (i.e. convolution, Sec. 1.1.2). If the filter length is $m < s_j$, the remaining elements of the matrix row are zeros, but, conversely, if $m > s_j$, the first s_j samples are used to fill the row and the rest is cyclically folded so that the values are added to the previous ones. The process of building the transform matrix is implemented in the file `dwtmatrix.m` described in Sec. 4.

Example 2.1. Working with orthogonal filters $\mathbf{h} = [h_1, h_2, h_3, h_4]$ and $\mathbf{g} = [g_1, g_2, g_3, g_4]$ (which may correspond to the Daubechies `db2` wavelet, for example)

and a signal of length $s = 8$, the decomposition matrix is of the form:

$$\mathbf{W}^{-1} = \begin{bmatrix} h_3 & h_2 & h_1 & 0 & 0 & 0 & 0 & h_4 \\ 0 & h_4 & h_3 & h_2 & h_1 & 0 & 0 & 0 \\ 0 & 0 & 0 & h_4 & h_3 & h_2 & h_1 & 0 \\ h_1 & 0 & 0 & 0 & 0 & h_4 & h_3 & h_2 \\ g_3 & g_2 & g_1 & 0 & 0 & 0 & 0 & g_4 \\ 0 & g_4 & g_3 & g_2 & g_1 & 0 & 0 & 0 \\ 0 & 0 & 0 & g_4 & g_3 & g_2 & g_1 & 0 \\ g_1 & 0 & 0 & 0 & 0 & g_4 & g_3 & g_2 \end{bmatrix}. \tag{2.5}$$

Example 2.2. The decomposing matrix with biorthogonal filters $\mathbf{h} = [h_1, h_2, h_3, h_4, h_5]$ and $\mathbf{g} = [g_1, g_2, g_3, 0, 0]$ (`bior2.2` spline wavelet, for example) and signal length $s = 8$ is

$$\mathbf{W}^{-1} = \begin{bmatrix} h_3 & h_2 & h_1 & 0 & 0 & 0 & h_5 & h_4 \\ h_5 & h_4 & h_3 & h_2 & h_1 & 0 & 0 & 0 \\ 0 & 0 & h_5 & h_4 & h_3 & h_2 & h_1 & 0 \\ h_1 & 0 & 0 & 0 & h_5 & h_4 & h_3 & h_2 \\ g_3 & g_2 & g_1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & g_3 & g_2 & g_1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & g_3 & g_2 & g_1 & 0 \\ g_1 & 0 & 0 & 0 & 0 & 0 & g_3 & g_2 \end{bmatrix}. \tag{2.6}$$

For greater decomposition depths $j = 1, 2, \ldots, J$, let the dual basis matrix be denoted $\mathbf{W}_J^{-1}$. There are two ways how to construct it. The first one relies on the cascade algorithm [7], the second one is more elegant and uses just matrix operations as shown in (2.7), using (2.4) [26].

$$\begin{aligned} \mathbf{W}_1^{-1} &= [\quad & & & \mathbf{H}_1^\top \ \mathbf{G}_1^\top \ ]^\top \\ \mathbf{W}_2^{-1} &= [\quad & & \mathbf{H}_1^\top\mathbf{H}_2^\top \ \mathbf{H}_1^\top\mathbf{G}_2^\top \ \mathbf{G}_1^\top \ ]^\top \\ \mathbf{W}_3^{-1} &= [\quad & \mathbf{H}_1^\top\mathbf{H}_2^\top\mathbf{H}_3^\top \ \mathbf{H}_1^\top\mathbf{H}_2^\top\mathbf{G}_3^\top \ \mathbf{H}_1^\top\mathbf{G}_2^\top \ \mathbf{G}_1^\top \ ]^\top \\ &\vdots \\ \mathbf{W}_J^{-1} &= [\ \mathbf{H}_1^\top \ldots \mathbf{H}_J^\top \ \mathbf{H}_1^\top \ldots \mathbf{H}_{J-1}^\top\mathbf{G}_J^\top \quad \ldots \quad \mathbf{H}_1^\top\mathbf{G}_2^\top \ \mathbf{G}_1^\top \ ]^\top. \end{aligned} \tag{2.7}$$

2.3. *Connection Between DWT and Linear Filtering, Mallat's Algorithm*

The computation of the DWT via the matrix multiplication (which has quadratic computational complexity) can be replaced by a less demanding pyramidal algorithm by S. G. Mallat [6,7]. Mallat's algorithm exploits the structure of matrix $\mathbf{W}$ (as seen in the previous section) and performs recursively:

The input vector $\mathbf{x}$ is filtered by the low-pass filter $\mathbf{h}$ and high-pass filter $\mathbf{g}$, respectively. Both the outputs are then downsampled. This way two new series of coefficients are obtained, containing exactly or approximately half the number of

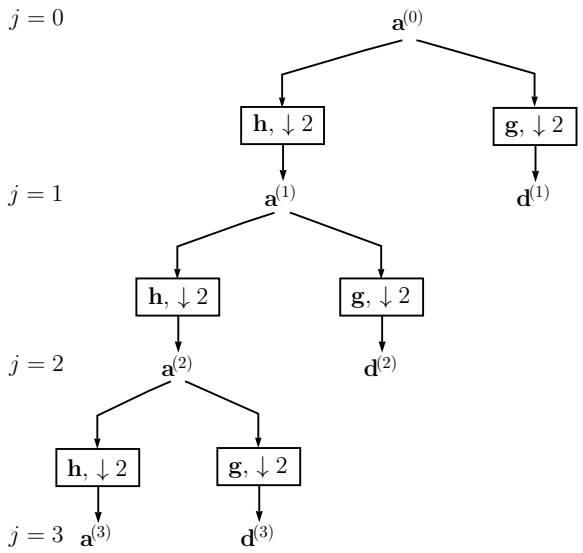

Fig. 1. Mallat's DWT pyramidal algorithm with depth $J = 3$. Signal $\mathbf{x}$ is decomposed using $\mathbf{g}, \mathbf{h}$ iteratively into the wavelet coefficients. The respective level of decomposition is shown on the left.

input samples (the number depends on the selected border extension type, see Sec. 2.1). The resultant coefficients produced by $\mathbf{h}$ are called *approximation coefficients* and those produced by $\mathbf{g}$ *detail coefficients* of the signal at level $j = 1$.

The detail coefficients are stored and the process just described is repeated with the approximation coefficients playing the role of the input signal. The number of such iterations, the *depth of decomposition*, J, is optional. Fig. 1 shows the described procedure for $J = 3$.

The limit for the choice of J usually presented in the literature is

$$J_{\max} = \lfloor \log_2 s \rfloor \tag{2.8}$$

for a signal of length s. This comes from $s/2^J \geq 1$, or in words: there is no sense in continuing the decomposition if there is only one coefficient left. Worth noting is that (2.8) holds for arbitrary m but only in the case of periodic border extension. If any other border extension is used, it is theoretically possible to continue the recursions infinitely many times, but this lacks any practical meaning from a certain level on. Such situations are discussed starting from Theorem 2.5 up to Corollary 2.3.

The DWT process has been described only briefly until now. The details have been intentionally postponed to the next section.

2.4. *DWT Pyramidal Algorithm Analysis*

In this part, only border extensions of type 3 are considered (see Sec. 2.1 for details). However, some of the results hold for the other types of border extension as well.

Algorithm 1 (DWT with defining samples outside of the original signal domain).
Input: input signal $\mathbf{x}$ of length s, two wavelet decomposition filters of equal length m (either odd or even) — the high-pass $\mathbf{g}$ and the low-pass $\mathbf{h}$ —, the downsampling type $\varepsilon \in \{0, 1\}$, the depth of decomposition J.
Output: Vectors of coefficients $\mathbf{a}^{(J)}, \mathbf{d}^{(J)}, \mathbf{d}^{(J-1)}, \ldots, \mathbf{d}^{(1)}$.

(1) Denote $\mathbf{x}$ as $\mathbf{a}^{(0)}$. Set $j = 0$.
(2) One step of decomposition:
    (a) *Input signal border extension.* Extend $\mathbf{a}^{(j)}$ by $(m-1)$ samples from both sides, whose values reflect the chosen type of border extension.
    (b) *Filtering.* Filter the extended signal with $\mathbf{g}$, which could be done by means of convolution.
    (c) *Cropping off.* Crop off $m-1$ samples from both ends of the vector.
    (d) *Downsampling.* Downsample the cropped vector, ε determines the type used.

    Store the resultant vector in $\mathbf{d}^{(j+1)}$.
    Repeat steps (b)–(d), now with the filter $\mathbf{h}$, and store the result in $\mathbf{a}^{(j+1)}$.
(3) Increase j by 1. If $j < J$ start the next level beginning with step 2, else the algorithm ends.

Remark 2.1. In contrast to Sec. 2.2, where the wavelet coefficients of different depths were in fact kept in a single vector, one after another, here we rather think of the output as of $J+1$ vectors (or sets) of coefficients. This is mainly for the sake of convenience in addressing the individual coefficients.

Remark 2.2. Step 2(a) is done in each iteration of the algorithm. The values of the samples used for extending the borders at level $j = 0$ are determined directly from the input signal. For higher j the extensions are calculated from the actual (approximation) coefficients by analogy. This means, among other things, that when the extension is done incorrectly in some sense, the originated error accumulates with increasing j. Section 2.4.5 deals with this issue.

In the following text, the identity $\mathbf{x} = \mathbf{a}^{(0)}$ is assumed. The individual elements of wavelet coefficient vectors are referenced using their respective lowercase letters — for example, $a_{15}^{(2)}$ denotes the 15th element of the approximate coefficients at depth 2, i.e. of vector $\mathbf{a}^{(2)}$. For purposes of the statements, level $j \in \{0, \ldots, J\}$ and a fixed length of filters m are assumed.

2.4.1. *How many coefficients does DWT produce*

The denotation $n_{\mathrm{coef}}(s, m, j, \varepsilon)$ is used to represent the number of coefficients at level j produced by Algorithm 1 from a signal of length s with filters of length m, and with ε-type downsampling. Hence $n_{\mathrm{coef}}(s, m, J, \varepsilon) = \mathrm{len}(\mathbf{a}^{(J)}) = \mathrm{len}(\mathbf{d}^{(J)})$. The *total* number of coefficients at depth j will thus be $2\, n_{\mathrm{coef}}(s, m, j, \varepsilon)$ and the *overall*

number of coefficients produced will be $\text{len}(\mathbf{a}^{(J)}) + \text{len}(\mathbf{d}^{(J)}) + \text{len}(\mathbf{d}^{(J-1)}) + \ldots + \text{len}(\mathbf{d}^{(1)})$, or

$$n^{\Sigma}_{\text{coef}}(s, m, J, \varepsilon) = n_{\text{coef}}(s, m, J, \varepsilon) + \sum_{j=1}^{J} n_{\text{coef}}(s, m, j, \varepsilon). \tag{2.9}$$

Theorem 2.1. *Algorithm 1 produces*

$$n_{\text{coef}}(s, m, 1, \varepsilon) = \frac{s+m-1}{2} + \left(\varepsilon - \frac{1}{2}\right) \text{par}(s+m-1) \tag{2.10}$$

coefficients from a signal of length s at decomposition depth $J = 1$. Equivalently,

$$n_{\text{coef}}(s, m, 1, \varepsilon) = \begin{cases} \frac{s+m-1}{2} & \text{for } s+m-1 \text{ even,} \\ \left\lfloor \frac{s+m-1}{2} \right\rfloor + \varepsilon & \text{for } s+m-1 \text{ odd.} \end{cases} \tag{2.11}$$

Proof. According to 2(a), in the first recursion of the algorithm $\mathbf{a}^{(0)}$ is extended to $s + 2(m-1)$. After convolving it with the filter in step 2(b), the output length is $[s + 2(m-1) + m - 1] = s + 3(m-1)$. After step 2(c), the cropped central part contains $s+m-1$ samples, the same length that the convolution of the not-extended signal $\mathbf{a}^{(0)}$ would result in. Using Lemma 1.1, after ε-type subsampling in step 2(d) the final length is given by (2.10). □

Remark 2.3. It is clear from the algorithm description that using either the low-pass or high-pass filter results in the same number of coefficients. Nevertheless, when biorthogonal filters are considered, it is known in advance that several border coefficients are zero (due to the zero padding of the shorter filter, see Section 1.1.1).

Corollary 2.1. *When even-type downsampling is utilized equation* (2.10) *becomes*

$$n_{\text{coef}}(s, m, 1, 0) = \left\lfloor \frac{s+m-1}{2} \right\rfloor. \tag{2.12}$$

Proof. This comes directly from (2.11). □

From this point up to the end of Section 2.4 we will assume only even-type sampling $\varepsilon = 0$, mainly because otherwise the formulas would get unnecessarily complicated. The symbol ε will be omitted in most of such cases. Such a restriction does not impair generality, see Remark 1.3.

Theorem 2.2. *Given a vector of length s, the number of wavelet coefficients produced at level $j \geq 1$ is given by*

$$n_{\text{coef}}(s, m, j) = \left\lfloor 2^{-j} s + (1 - 2^{-j})(m-1) \right\rfloor. \tag{2.13}$$

Proof. The proof is done using induction with respect to j. For $j = 1$, Eq. (2.13) holds true because substituting $j = 1$ into it becomes (2.12) after a short manipulation. Assume that (2.13) holds for a fixed $j \geq 1$. Then the number of coefficients

Table 1. Numbers of coefficients returned by Algorithm 1 from signal of length s at different depths of decomposition J. See Theorem 2.2 and the subsequent statements.

(a) $n_{\mathrm{coef}}(s, m, J)$ for $m = 8$

$J \backslash s$	**4**	**6**	**7**	**8**	**9**	**15**	**30**	**45**	**90**	**125**	**180**	**250**	**335**
1	5	6	7	7	8	11	18	26	48	66	93	128	171
2	6	6	7	7	7	9	12	16	27	36	50	67	89
3	6	6	7	7	7	8	9	11	17	21	28	37	48
4	6	6	7	7	7	7	8	9	12	14	17	22	27
5	6	6	7	7	7	7	7	8	9	10	12	14	17
6	6	6	7	7	7	7	7	7	8	8	9	10	12
7	6	6	7	7	7	7	7	7	7	7	8	8	9
8	6	6	7	7	7	7	7	7	7	7	7	7	8
J_{const}	–	–	0	1	2	4	5	6	7	7	8	8	9

(b) $n^{\Sigma}_{\mathrm{coef}}(s, m, J)$ for $m = 8$

$J \backslash s$	**4**	**6**	**7**	**8**	**9**	**15**	**30**	**45**	**90**	**125**	**180**	**250**	**335**
1	10	12	14	14	16	28	36	52	96	132	186	256	342
2	17	18	21	21	22	34	42	58	102	138	193	262	349
3	23	24	28	28	29	40	48	64	109	144	199	269	356
4	29	30	35	35	36	46	55	71	116	151	205	276	362
5	35	36	42	42	43	53	61	78	122	157	212	282	369
6	41	42	49	49	50	60	68	84	129	163	218	288	376
7	47	48	56	56	57	67	75	91	135	169	225	294	382
8	53	54	63	63	64	74	82	98	142	176	231	300	389

produced at depth $j+1$ by a single recursion of Alg. 1 is given by (2.12):

$$\begin{aligned}\left\lfloor \frac{n_{\mathrm{coef}}(s,m,j)+m-1}{2} \right\rfloor &= \left\lfloor \frac{\left\lfloor 2^{-j}s + (1-2^{-j})(m-1)\right\rfloor + m - 1}{2} \right\rfloor \\ &= \left\lfloor \frac{\left\lfloor 2\{2^{-(j+1)}s + (1-2^{-(j+1)})(m-1)\}\right\rfloor}{2} \right\rfloor \\ &= \left\lfloor 2^{-(j+1)}s + (1-2^{-(j+1)})(m-1) \right\rfloor,\end{aligned}$$

exploiting the fact that $\left\lfloor \frac{\lfloor 2a \rfloor}{2} \right\rfloor = \lfloor a \rfloor$ holds true for any $a \in \mathbb{R}$. □

Tables 1(a) and 2(a) show the numbers of coefficients, computed by Alg. 1 for various choices of s and m. Theorems now follow that prove the declared expansivity of the analyzed type of DWT.

Theorem 2.3. *It holds*

$$n^{\Sigma}_{\mathrm{coef}}(s, m, 1) = \begin{cases} s + (m-2) & \textit{for } \mathrm{par}(s) = \mathrm{par}(m) \\ s + (m-1) & \textit{for } \mathrm{par}(s) \neq \mathrm{par}(m)\,. \end{cases} \tag{2.14}$$

Table 2. Numbers of coefficients returned by Algorithm 1 from signal of length s at different depths of decomposition J. See Theorem 2.2 and the subsequent statements.

(a) $n_{\mathrm{coef}}(s, m, J)$ for $m = 15$

$J \backslash s$	**4**	**13**	**14**	**15**	**16**	**21**	**30**	**45**	**90**	**125**	**180**	**250**	**335**
1	9	13	14	14	15	14	22	29	52	69	97	132	174
2	11	13	14	14	14	14	18	21	33	41	55	73	94
3	12	13	14	14	14	14	16	17	23	27	34	43	54
4	13	13	14	14	14	14	15	15	18	20	24	28	34
5	13	13	14	14	14	14	14	14	16	17	19	21	24
6	13	13	14	14	14	14	14	14	15	15	16	17	19
7	13	13	14	14	14	14	14	14	14	14	15	15	16
8	13	13	14	14	14	14	14	14	14	14	14	14	15
J_{const}	–	–	0	1	2	3	5	5	7	7	8	8	9

(b) $n^{\Sigma}_{\mathrm{coef}}(s, m, J)$ for $m = 15$

$J \backslash s$	**4**	**13**	**14**	**15**	**16**	**21**	**30**	**45**	**90**	**125**	**180**	**250**	**335**
1	18	26	28	28	30	34	44	58	104	138	194	264	348
2	31	39	42	42	43	47	58	71	118	151	207	278	362
3	44	52	56	56	57	60	72	84	131	164	220	291	376
4	58	65	70	70	71	74	86	97	144	177	234	304	390
5	71	78	84	84	85	88	99	110	158	191	248	318	404
6	84	91	98	98	99	102	113	124	172	204	261	331	418
7	97	104	112	112	113	116	127	138	185	217	275	344	431
8	110	117	126	126	127	130	141	152	199	231	288	357	445
9	123	130	140	140	141	144	155	166	213	245	302	371	458
10	136	143	154	154	155	158	169	180	227	259	316	385	472
11	149	156	168	168	169	172	183	194	241	273	330	399	486

Proof. Look at the difference between the number of coefficients at depth $J = 1$ and the number of signal samples: $n^{\Sigma}_{\mathrm{coef}}(s, m, 1) - s = 2\, n_{\mathrm{coef}}(s, m, 1) - s = 2\left\lfloor \frac{s+m-1}{2} \right\rfloor - s$.

In the case of both s and m being even, this can be reformulated:

$$\begin{aligned} 2\left\lfloor \frac{s+m-1}{2} \right\rfloor - s &= 2\left\lfloor \frac{s+m-1}{2} - \frac{s}{2} \right\rfloor \\ &= 2\left\lfloor \frac{m-1}{2} \right\rfloor \\ &= 2\left\lfloor \frac{2\ell-1}{2} \right\rfloor \\ &= 2(\ell-1) \\ &= m-2. \end{aligned} \tag{2.15}$$

In the case of odd m the approach would be the same up to equation (2.15), from where we would continue with $(2.15) = 2\left(\frac{m-1}{2}\right) = m-1$.

In the case of both s and m being odd, more tricks must be utilized:

$$\begin{aligned} 2\left\lfloor\frac{s+m-1}{2}\right\rfloor - s &= 2\left\lfloor\frac{s+m-1}{2} - \left\lfloor\frac{s}{2}\right\rfloor\right\rfloor - 1 \\ &= 2\left\lfloor\frac{s}{2} - \left\lfloor\frac{s}{2}\right\rfloor + \frac{m-1}{2}\right\rfloor - 1 \\ &= 2\left\lfloor\frac{m}{2}\right\rfloor - 1 \qquad (2.16) \\ &= 2\left\lfloor\frac{2\ell+1}{2}\right\rfloor - 1 \\ &= 2\ell - 1 = m - 2, \end{aligned}$$

where the fact $\frac{s}{2} - \left\lfloor\frac{s}{2}\right\rfloor = \frac{1}{2}$ was exploited. In the case of even m, Eq. (2.16) would continue as $(2.16) = 2\left(\frac{m}{2}\right) - 1 = m - 1$. □

Theorem 2.4. *The overall number of DWT coefficients, $n_{\mathrm{coef}}^{\Sigma}(s,m,J)$, increases with each depth of decomposition, J, by $(m-1)$ or by $(m-2)$.*

Proof. This follows immediately from Theorem 2.3 due to the DWT recursivity: at depths greater than 1, the symbol s plays the role of the number of coefficients from the previous depth. □

These facts can be tracked down in Tables 1(b) and 2(b). The sequence of the increments can be characterized even deeper, but before we do it in Theorem 2.6, we first introduce a corollary and several other results.

Corollary 2.2. *DWT via Algorithm 1 is expansive, i.e., $n_{\mathrm{coef}}^{\Sigma}(s,m,J) > s$, for any depth $J \geq 1$ with $m \geq 3$. More precisely, for even s, an expansion occurs for $m \geq 3$, and for odd s, it occurs already for $m \geq 2$.*

Proof. These conditions follow from Theorems 2.3 and 2.4, where possible combinations of parities of s and m are taken into consideration. □

Remark 2.4. Therefore, only a wavelet transform using filters of length $m = 2$ (e.g. the Haar filters) can be non-expansive, and this holds only up to transform depth j such that s is divisible by 2^j.

Theorem 2.5. *When the signal length is $s = m-2$ or $s = m-1$, one recursion of DWT produces precisely twice the number of coefficients (approximation and detail), formally written*

$$n_{\mathrm{coef}}(m-1,m,1) = m-1, \quad n_{\mathrm{coef}}(m-2,m,1) = m-2. \qquad (2.17)$$

In the case of $s < m-2$ a single recursion of DWT generates more than twice the number of coefficients, i.e. $n_{\mathrm{coef}}(s,m,1) > s$. Finally, when $s \geq m$, DWT produces coefficients whose number is less than $2s$.

Proof. The first part of the theorem: We evaluate $n_{\text{coef}}(s, m, 1)$ using (2.12) and we ask when it is equal to the number of signal samples:

$$\left\lfloor \frac{s+m-1}{2} \right\rfloor = s$$
$$\left\lfloor \frac{s+m-1}{2} - s \right\rfloor = 0$$
$$\left\lfloor \frac{-s+m-1}{2} \right\rfloor = 0.$$

This can happen only when $s = m-2$ or $s = m-1$. The second and third part of the theorem can be proved analogously. □

These facts are again visible in Tables 1 and 2, together with all the ongoing results.

Remark 2.5. Due to the recursive nature of the transform, Eq. (2.17) can be generalized. For *any* depth $j \geq 0$ it holds

$$n_{\text{coef}}(s, m, j) = s \quad \text{for either } s = m-1 \text{ or } s = m-2. \tag{2.18}$$

Theorem 2.6. *If $s \geq m-1$, then there exists a finite depth of decomposition $J \geq 0$ such that*

$$n^{\Sigma}_{\text{coef}}(s, m, j+1) = n^{\Sigma}_{\text{coef}}(s, m, j) + (m-1) \ \textit{for all } j > J, \tag{2.19}$$

to put it in words: the increment of the total number of coefficients remains $m-1$ for $j > J$. The minimum depth J satisfying (2.19) *is obtained as*

$$J_{\text{const}} = J_{\text{const}}(s, m) = \min\{j | j > \log_2(s-m+1)\}, \tag{2.20}$$

and we define $J_{\text{const}} = 0$ if $s = m-1$.

Proof. The goal is to find the minimum J for which (2.19) holds. The problem can be, however, stated in a simpler way. In Theorem 2.5 it was proved that if $s = m-2$ or $s = m-1$, the DWT produces precisely double the number of coefficients during a single recursion. Therefore, provided that there exists a depth J satisfying either $n_{\text{coef}}(s, m, J) = m-1$ or $m-2$, then it must hold also for levels $j \geq J$ (see Remark 2.5). This means that the generally valid formula

$$\begin{aligned} n^{\Sigma}_{\text{coef}}(s, m, j+1) - n^{\Sigma}_{\text{coef}}(s, m, j) = \\ = 2n_{\text{coef}}(s, m, j+1) - n_{\text{coef}}(s, m, j) \end{aligned} \tag{2.21}$$

could be simplified to $(2.21) = n_{\text{coef}}(s, m, j) = s$ for the considered depths $j \geq J$. Thus the desired J_{const} is the smallest j which accomplishes $n_{\text{coef}}(s, m, j) = m-1$ or $m-2$. These two possibilities will now be explored in detail, in order to see, in the end, that only the first of them can actually occur.

In the first case,

$$\begin{aligned} n_{\mathrm{coef}}(s,m,j) &= m-1 \\ \left\lfloor 2^{-j}s + (1-2^{-j})(m-1) \right\rfloor &= m-1 \\ \left\lfloor 2^{-j}s + (1-2^{-j})(m-1) - (m-1) \right\rfloor &= 0 \\ \left\lfloor 2^{-j}(s-m+1) \right\rfloor &= 0, \end{aligned}$$

which means that any suitable j complies with

$$\begin{aligned} 0 \leq 2^{-j}(s-m+1) < 1, \text{ i.e.} \\ 0 \leq \quad (s-m+1) < 2^{j}. \end{aligned} \tag{2.22}$$

The left inequality is fulfilled due to the assumption $s \geq m-1$. Thus, the smallest j satisfying (2.22) is just J_{const} from (2.20); the logarithm is defined for $s \geq m$, therefore all complying levels j are positive, $j > 0$.

In the second case, the following inequalities can be derived analogously:

$$\begin{aligned} -1 \leq 2^{-j}(s-m+1) < 0, \text{ i.e.} \\ -2^{j} \leq \quad (s-m+1) < 0. \end{aligned} \tag{2.23}$$

The right-side inequality contradicts the assumption. Thus, in the case $s \geq m-1$ the increment can never stabilize at $m-2$. □

Example 2.3. For a filter of $m = 8$ and signal length $s = 180$ we have $J_{\mathrm{const}} = J_{\mathrm{const}}(180, 8) = 8$ and the series of increments is
$\{n^{\Sigma}_{\mathrm{coef}}(s,m,j+1) - n^{\Sigma}_{\mathrm{coef}}(s,m,j)\}_{j=0}^{\infty} = \{6,7,6,6,7,6,7,6,7,7,7,\ldots\}$.

Remark 2.6. Similarly, a theorem could be formulated that by $s \leq m-2$ the series of increments first alternates between $m-2$ and $m-1$ until it stabilizes at $m-2$.

Remark 2.7. In contrast to the periodic case of boundary handling, where the maximum decomposition depth is given by (2.8), here it can be seen that the transform recursions can be theoretically repeated as many times as one likes. (Although from the practical point of view it is not of interest.)

Corollary 2.3. *Assuming $s \geq m-1$, J_{const} is a depth of decomposition such that it holds*

$$n_{\mathrm{coef}}(s,m,j) = m-1 \quad \textit{for all } j \geq J_{\mathrm{const}}. \tag{2.24}$$

2.4.2. *How many input signal samples are needed for computing a specified number of coefficients*

This section is devoted to the backward look — from the coefficient domain into the time domain. For these purposes, $n_{\mathrm{samp}}(q,j,m)$ will denote the *number* of sam-

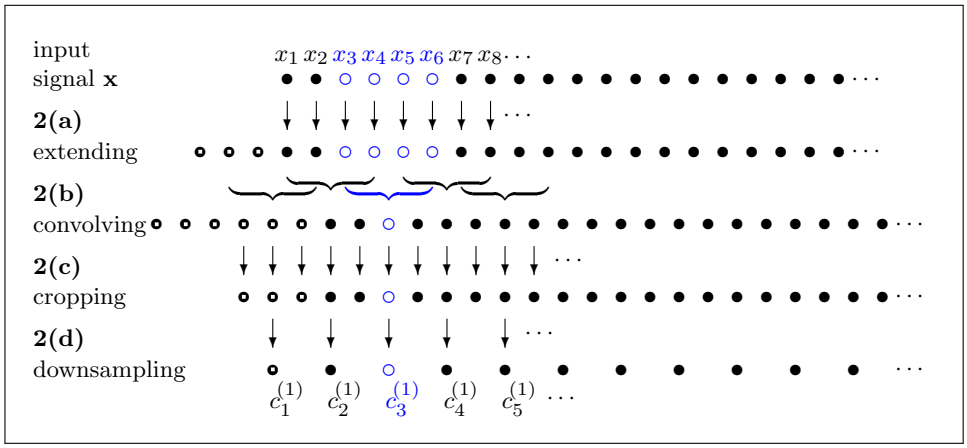

Fig. 2. Depiction of steps 2(a) to 2(d) of Alg. 1 for the filter length $m = 4$ and even downsampling $\varepsilon = 0$. Each circle stands for a single vector entry (i.e., signal sample or coefficient). The circles with squares inside stand for samples/coefficients induced/affected by the boundary extension. Marked with empty circles, the procedure of obtaining the third output coefficient $c_3^{(1)}$ is calculated using signal samples at indexes 3 to 6.

ples from the input signal that are needed to compute q consecutive coefficients at decomposition level j.

Lemma 2.1. *To calculate a single wavelet coefficient at level j, m coefficients at level $j-1$ are needed.*

Proof. This is clear from the description of the algorithm; a wavelet coefficient at depth j is just a single element of the convolution of vector $\mathbf{a}^{(j-1)}$ with the wavelet filter of length m; see Fig. 2. □

Lemma 2.2. *To calculate $q \geq 1$ consecutive coefficients at level j it is necessary to provide $m + 2(q-1)$ consecutive coefficients at level $(j-1)$.*

Proof. To calculate a single coefficient at level j it is necessary to have m coefficients at level $j-1$. Every other neighbouring coefficient at level j requires two extra coefficients at level $j-1$ due to the downsampling; see also Figs. 2 and 3. □

Theorem 2.7. *To calculate $q \geq 1$ consecutive wavelet coefficients at level of decomposition $j \geq 1$, it is necessary to have $n_{\mathrm{samp}}(q, j, m)$ samples of the input signal, where*

$$n_{\mathrm{samp}}(q, j, m) = (2^j - 1)(m-2) + 2^j q. \tag{2.25}$$

Proof. Recursive application of Theorem 2.2 leads to

$$n_{\mathrm{samp}}(q, j, m) = m + 2[n_{\mathrm{samp}}(q, j-1, m) - 1], \tag{2.26}$$

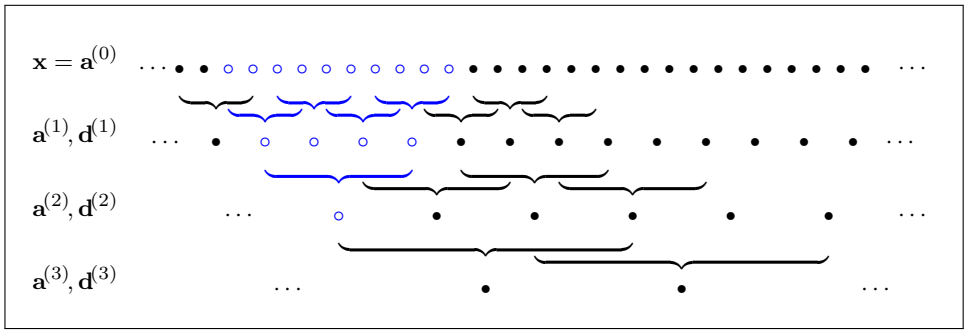

Fig. 3. Depiction of Alg. 1 in three levels of decomposition, $J = 3$. Each coefficient at level j is calculated using m coefficients from level $j-1$. The filter length used is $m = 4$. Empty circles indicate the "range of influence" of one coefficient from depth $j = 2$ at depths $j = 1$ and $j = 0$.

where we define $n_{\text{samp}}(q, 0, m) = q$. It will be shown by induction that (2.25) is the non-recursive form of (2.26). For $j = 1$ it holds $n_{\text{samp}}(q, 1, m) = m + 2[n_{\text{samp}}(q, 0, m) - 1] = m + 2(q-1)$ due to (2.26). According to (2.25), we also have $n_{\text{samp}}(q, 1, m) = m + 2q - 2 = m + 2(q-1)$.

Assume from now on that for a chosen $j \geq 1$ Eq. (2.25) holds. Using (2.26),

$$\begin{aligned} n_{\text{samp}}(q, j+1, m) &= m + 2[n_{\text{samp}}(q, j, m) - 1] \\ &= m + 2[(2^j - 1)(m-2) + 2^j q - 1] \\ &= [2(2^j - 1) + 1](m-2) + 2^{j+1} q \\ &= (2^{j+1} - 1)(m-2) + 2^{j+1} q. \end{aligned}$$

□

Corollary 2.4. *To calculate $q \geq 1$ consecutive coefficients at level j it is necessary to have $n_{\text{samp}}(q, j-u, m)$ coefficients at level u, $0 \leq u < j$.*

Proof. The statement can be proved via Theorem 2.7 — when talking about the numbers of coefficients, the situation $0 \leq u < j$ is actually equivalent to the situation of enumerating the number of input signal samples (i.e. the coefficients at level 0) for q coefficients at depth $j - u$. □

Corollary 2.5. *To calculate a single wavelet coefficient at level j, it is necessary to have*

$$n_{\text{samp}}(1, j, m) = (2^j - 1)(m-1) + 1. \tag{2.27}$$

samples of the input signal.

Proof. Plugging $q = 1$ into (2.25) quickly leads to this result. □

Table 3 shows the numbers of signal samples necessary for calculating a single wavelet coefficient at different depths of decomposition. Fig. 4 shows the table data in graphs; it is clear that the increase in filter length leads to increasing

Table 3. Numbers of input signal samples necessary for computation of a single coefficient, $n_{\mathrm{samp}}(1, j, m)$. Wavelet filter length m in horizontal direction, depth of decomposition j vertically.

$n_{\mathrm{samp}}(1, j, m)$

$j \setminus m$	**2**	**3**	**4**	**5**	**6**	**8**	**10**	**12**	**14**	**16**	**18**
1	2	3	4	5	6	8	10	12	14	16	18
2	4	7	10	13	16	22	28	34	40	46	52
3	8	15	22	29	36	50	64	78	92	106	120
4	16	31	46	61	76	106	136	166	196	226	256
5	32	63	94	125	156	218	280	342	404	466	528
6	64	127	190	253	316	442	568	694	820	946	1072
7	128	255	382	509	636	890	1144	1398	1652	1906	2160
8	256	511	766	1021	1276	1786	2296	2806	3316	3826	4336

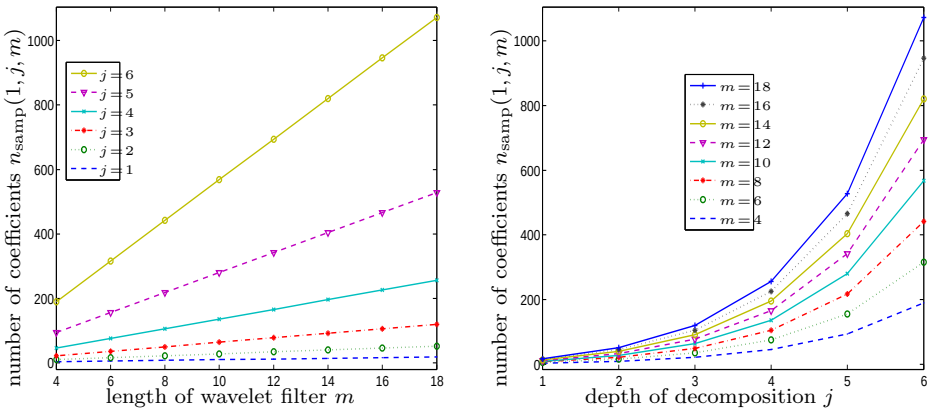

Fig. 4. The numbers of signal samples necessary for computation of a single coefficient: (a) depth j is fixed, (b) filter length m is fixed.

$n_{\mathrm{samp}}(1, j, m)$ linearly; the increase of the decomposition depth leads to an exponential growth.

2.4.3. *Other theorems*

Prior to the following considerations, several new terms and notation need to be established:

- Two coefficients at a given level of decomposition will be called *neighbouring coefficients* if their indexes differ by one. For example, $d_{23}^{(3)}$ and $d_{24}^{(3)}$ are neighbouring at level 3. Two coefficients with q coefficients in between will be termed *q-distant*. For example, $a_{19}^{(2)}$ and $a_{26}^{(2)}$ are 6-distant coefficients.

Clearly, coefficients with a distance $q = 0$ are neighbouring coefficients.

- The *range of influence* of a coefficient from level j at level u, $0 \leq u \leq j$, is the set of indexes of coefficients at level u, which contribute to the value of the specified coefficient at level j. This term is chosen because if the particular coefficient were changed, all the coefficients in its range would be "influenced" after performing the inverse transform (and vice versa). The range of influence of the kth coefficient from level j at level u will be denoted $\mathrm{ran}(k, j, u, m)$.
  For example, for the coefficient $d_{11}^{(3)}$ we have $\mathrm{ran}(11, 3, 0, 5) = \{60, 61, \ldots, 88\}$.
  The lower boundary can turn out to be negative; such a case corresponds to those coefficients that were appended by the border extension step in the DWT algorithm. This can also be seen in Fig. 2; the range of influence of $c_1^{(1)}$ starts with the index -1 in the input signal $\mathbf{x}$: $\mathrm{ran}(1, 1, 0, 4) = \{-1, 0, 1, 2\}$. Similar considerations hold for the upper boundary of the range.
- *Shift in terms of signal samples* for given two coefficients at level j is the number of input signal samples by which the ranges of influence of the respective two coefficients differ. For example, while the range of influence of a given coefficient in the time domain (i.e., $u = 0$) would be $\{8, 9, \ldots, 23\}$ and the other one would have its own range $\{12, 13, \ldots, 27\}$, their shift in terms of signal samples would be simply 4. Formally, for $k_1 \leq k_2$, this will be denoted $n_{\mathrm{shift}}(k_1, k_2, j, m) = \min(\mathrm{ran}(k_1, j, 0, m)) - \min(\mathrm{ran}(k_2, j, 0, m))$.
- Given a pair of coefficients at level j, the *shared coefficients* at level u, $0 \leq u < j$, of these two are the coefficients at level u such that they simultaneously belong to both the ranges of influence of the coefficients from level j. Returning to the above example, the samples in the time domain (i.e. choosing $u = 0$) that are shared are $x_{12}, x_{13}, \ldots, x_{23}$. We are interested in the indexes, not in their values; formally, $\mathrm{shared}(k_1, k_2, j, u, m) = \mathrm{ran}(k_1, j, u, m) \cap \mathrm{ran}(k_2, j, u, m)$, while allowing negative indexes in the intersection. The number of elements of such a set of indexes will be denoted $n_{\mathrm{shared}}(k_1, k_2, j, u, m)$.

Theorem 2.8. *The length of the range of influence of a coefficient from level j at level u is*

$$|\mathrm{ran}(k, j, u, m)| = (2^{j-u} - 1)(m - 1) + 1. \tag{2.28}$$

Proof. Corollary 2.4 says that the cardinality of such a set is equal to $|\mathrm{ran}(k, j, u, m)| = n_{\mathrm{samp}}(1, j - u, m)$, which becomes the desired formula after an easy manipulation. □

Remark 2.8. It is also immediately apparent that for even m, the length of the range is also even, and for odd m, the length of the range is odd.

Remark 2.9. It is not surprising that the quantity (2.28) is equal to (2.27) when we take $u = 0$.

Lemma 2.3. *The range of influence of the kth coefficient from level j at level u, $j \geq u$, satisfies*

$$\mathrm{ran}(k, j, u, m) = \mathrm{ran}(1, j, u, m) + 2^{j-u}(k-1). \tag{2.29}$$

Here, the addition means adding the number to each each element of the set.

Proof. It is clear that the range of influence is translated by 2^{j-u} from coefficient k to coefficient $k-1$ at each level of decomposition due to the downsampling (see Fig. 3). □

Theorem 2.9. *The range of influence of the kth coefficient from level j at level u is*

$$\mathrm{ran}(k, j, u, m) = \left\{2^{j-u}k - (2^{j-u}-1)(m-1), \ldots, 2^{j-u}k\right\}. \tag{2.30}$$

Proof. First, the upper boundary of the range will be derived. Based on Lemma 2.3 and an obvious definition $\mathrm{ran}(1, u, u, m) = \{1\}$ we can write $\max[\mathrm{ran}(k, u, u, m)] = \max[\mathrm{ran}(1, u, u, m)] + (k-1) = k$, which corresponds with the theorem, since $k = 2^{j-u}k$ for $j = u$.

Assume that the theorem is valid for $j > u$; from that the validity for $j+1$ will be derived:

$$\begin{aligned}
\max[\mathrm{ran}(k,&j+1, u, m)] = \\
&= \max[\mathrm{ran}(1, j+1, u, m)] + 2^{j+1-u}(k-1) \\
&= \max[\mathrm{ran}(2, j, u, m)] + 2^{j+1-u}(k-1) \\
&= 2^{j-u} \cdot 2 + 2^{j+1-u}(k-1) \\
&= 2^{j+1-u}k.
\end{aligned}$$

The manipulations above were based mainly on the observation that the range of influence of the very first coefficient from level $j+1$ at level j always ends at the element with index 2, which carries the analysis one level down. This fact can be seen in Fig. 2 as well.

The length of the range of influence is known from Theorem 2.8, thus it is enough to subtract that number from the above proved upper boundary $2^{j-u}k$ to get the desired starting index. □

Theorem 2.10. *The shift in terms of signal samples of two neighbouring coefficients at level j is*

$$n_{\mathrm{shift}}(k, k+1, j, m) = 2^j. \tag{2.31}$$

Two neighbouring coefficients at level j share

$$n_{\mathrm{shared}}(k, k+1, j, 0, m) = (2^j - 1)(m-2) \tag{2.32}$$

samples of the input signal.

Proof. The first part is clear due to the downsampling at each level. Next, two neighbouring coefficients at level j share $m-2$ coefficients at level $j-1$. Now, using Theorem 2.7 it can be calculated how many signal samples are necessary to get these coefficients; we get

$$\begin{aligned} n_{\text{shared}}(k, k+1, j, 0, m) &= |\text{shared}(k, k+1, j, 0, m)| \\ &= n_{\text{samp}}(m-2, j-1, m) \\ &= (2^{j-1}-1)(m-2) + 2^{j-1}(m-2) \\ &= (2^j - 1)(m-2). \end{aligned}$$ □

Example 2.4. For a filter of length $m = 4$, there are $n_{\text{shared}}(k, k+1, 2, 0, 4) = (2^2-1)(4-2) = 6$ time-domain samples shared by the neighbouring coefficients at level 2, which can be evidenced in Fig. 3. For the Haar wavelet ($m = 2$) there are no shared samples at any depth.

Theorem 2.11. *For two q-distant coefficients at level j it holds:*
1/ The shift in terms of signal samples of these coefficients is

$$\begin{aligned} n_{\text{shift}}(k, k+q+1, j, m) &= n_{\text{shift}}(k, k+1, j, m) \cdot (q+1) \\ &= 2^j(q+1). \end{aligned} \tag{2.33}$$

2/ The number of their shared samples in the time domain is given by

$$n_{\text{shared}}(k, k+q+1, j, 0, m) = \max\{0,\ (2^j-1)(m-2) - 2^j q\}. \tag{2.34}$$

Proof. The first part is clearly true as every coefficient in the line causes a shift by the jth power of two, see Fig. 3. Next, using relations (2.32) and (2.33), it is possible to derive that the number of time-domain samples that are shared by two q-distant coefficients at level j is $|\text{shared}(k, k+1, j, 0, m)| - q \cdot n_{\text{shift}}(k, k+1, j, m) = (2^j-1)(m-2) - 2^j q$. Should this number come out negative, we naturally set it to zero. □

Example 2.5. Given $m = 4, j = 2, q = 1$, the shift is $n_{\text{shift}}(k, k+2, 2, 4) = 8$ and the number of shared signal samples is $n_{\text{shared}}(k, k+2, 2, 0, 4) = \max\{0, (2^2-1)(4-2) - 2^2 \cdot 1\} = 2$. When we change the level from $j = 2$ to $j = 1$, it comes out that $n_{\text{shared}}(k, k+2, 1, 0, 4) = 0$. Again, this can be seen in Fig. 3.

Corollary 2.6. *Let the filter length be $m \geq 3$. In order to have a nonzero number of shared samples in the input signal, the maximum allowed distance between a pair of coefficients at level of decomposition j is*

$$q_{\max}(j, m) = \max\left\{q \,\middle|\, q < (1-2^{-j})(m-2),\ q \in \{0, 1 \ldots\}\right\}. \tag{2.35}$$

Proof. Equation (2.34) results in a positive number if $(2^j-1)(m-2) > 2^j q$ holds. The desired $q_{\max}$ follows directly from this inequality. □

Example 2.6. For $m = 4, j = 2$ we have $q_{\max} = 1 < \frac{3}{2}$. For $m = 6, j = 2$ we have $q_{\max} = 2 < 3$.

Remark 2.10. Theorem 2.10 is actually a special case of Theorem 2.11 when choosing $q = 0$.

Remark 2.11. The quantities (2.31) and (2.33) do not depend on the filter length, m.

2.4.4. *Example of Application*

Although it is not the main goal of the article, an example where such an analysis can be useful in practice is presented, following up on the results about the range of influence.

Imagine that a signal is stored in a compressed way using its wavelet coefficients (in the case of images this may roughly correspond to the SPIHT or JPEG2000 codecs). Imagine that the signal is subject to "local" editing in the original domain (image retouching, for example). In a standard approach, the DWT would be performed again on the whole signal after the edit has been done to store the modified signal. But this is not necessary: we can save much computation thanks to Theorem 2.9, as we can utilize the knowledge of which coefficients only actually have been changed by the editing.

Assume that only a single signal sample with index n has been edited. To derive which coefficients at different levels of decomposition are possibly affected, we utilize the mentioned Theorem: Regarding the right-hand side, we look for a coefficient whose leftmost sample of the range of influence is less than or equal to n, formally we wish to determine the maximum k such that $\min[\mathrm{ran}(k, j, 0, m)] \leq n$. An easy manipulation leads to

$$k_{\max} = k_{\max}(n, j, m) = \left\lfloor \frac{n + (2^j - 1)(m - 1)}{2^j} \right\rfloor . \tag{2.36}$$

Similarly, regarding the left-hand side, we end up with the desired starting index at a specified level j equal to

$$k_{\min} = k_{\min}(n, j) = \left\lceil \frac{n}{2^j} \right\rceil . \tag{2.37}$$

Thus only values of coefficients indexed from $k_{\min}(n, j)$ to $k_{\max}(n, j, m)$ at levels $j = 1, \ldots, J$ have to be recoded after editing.

See the program `demo_coefs_affected_by_signal_edit` for the implementation (in a slightly more general setup).

2.4.5. *Situation at signal boundaries*

The goal of this section is to determine how the result of the transform depends on the chosen type of border extension (see Sec. 2.1). Such an information can

be valuable e.g. in the case when the extension method would have been chosen inappropriately by mistake. Then the coefficients "near the boundaries" can have "incorrect" values. We are looking for an answer to this question:

How many wavelet coefficients at decomposition level j are affected by the values of samples by which the signal is extended according to a selected boundary extension method?

Such a number will be denoted $n_{\text{affect}}(j,m)$. At first, the situation when the depth of decomposition is just 1 will be explored at the "left" and the "right" boundaries of the signal.

Theorem 2.12. *The boundary extension type affects*

$$\begin{aligned} n^{\text{left}}_{\text{affect}}(1,m) &= \left\lfloor \frac{m-1}{2} \right\rfloor, \quad \textit{and} \\ n^{\text{right}}_{\text{affect}}(1,m) &= \left\lfloor \frac{m-1}{2} \right\rfloor + \operatorname{par}(s)\operatorname{par}(m-1) \end{aligned} \tag{2.38}$$

"left" and "right" wavelet coefficients, respectively. Thus, the scope of affection is the same at both ends of coefficient vectors, except the case when the signal length s is odd and m is even — then the number of the affected coefficients at the right side is greater by one coefficient.

Proof. In the first iteration of DWT (Algorithm 1), the input signal ($j = 0$) is initially extended by $m-1$ samples from both sides. Therefore, the number of coefficients affected by these $m-1$ samples at level $j = 1$ is $\left\lfloor \frac{m-1}{2} \right\rfloor$ at the left side. This fact can be seen in Fig. 2, where $m = 4$ results in just a single affected coefficient at the first depth.

The situation at the right side is slightly more complicated. The parities of s and m determine whether the range of influence of the rightmost wavelet coefficient encompasses the last sample of the input signal after its extension or just the last but one sample (this twofold option is due to the downsampling).

The last but one sample is encompassed in the case that $1 = \operatorname{par}(s+m-1)$. Then it can be stated that

$$n^{\text{right}}_{\text{affect}}(1,m) = \max\{q \,|\, (m-2) - 2(q-1) > 0\}.$$

The inequality inside the brackets can be simplified to $q < \frac{m}{2}$, which means, for m being odd, that the greatest such q equals to $\frac{m-1}{2} = \left\lfloor \frac{m-1}{2} \right\rfloor$, and thus it corresponds to the theorem. For m being even the highest such q is $\frac{m}{2} - 1 = \frac{m-1}{2} + \frac{1}{2} = \left\lfloor \frac{m-1}{2} \right\rfloor$ which again satisfies the theorem.

The very last sample falls into the range of influence of the coefficient if $\operatorname{par}(s+m-1) = 0$. Then we can write

$$n^{\text{right}}_{\text{affect}}(1,m) = \max\{q \,|\, (m-1) - 2(q-1) > 0\}.$$

The inequality can be simplified to $q < \frac{m+1}{2}$, which means that the greatest such q equals to $\frac{m+1}{2} - 1 = \frac{m-1}{2} = \left\lfloor \frac{m-1}{2} \right\rfloor$ for m being odd. For m being even, the greatest such q is $n^{\text{right}}_{\text{affect}}(1,m) = \frac{m}{2} = \left\lfloor \frac{m}{2} \right\rfloor = \left\lfloor \frac{m}{2} - \frac{1}{2} \right\rfloor + 1 = \left\lfloor \frac{m-1}{2} \right\rfloor + 1$. □

How many coefficients at an *arbitrary* level of decomposition $j \geq 1$ are affected? The following theorem gives the answer for the left-side case.

Theorem 2.13. *The choice of signal boundary extension type affects*

$$n_{\text{affect}}^{\text{left}}(j,m) = \left\lfloor (1-2^{-j})(m-1) \right\rfloor \tag{2.39}$$

"left-side" wavelet coefficients at decomposition level j.

Proof. According to Theorem 2.12, $n_{\text{affect}}^{\text{left}}(1,m) = \left\lfloor \frac{m-1}{2} \right\rfloor$, which is in correspondence with (2.39) being proved. At the next level of decomposition, $m-1$ coefficients are newly appended to the current ones. We add $\left\lfloor \frac{m-1}{2} \right\rfloor$ already affected coefficients to them to obtain the total number of affected coefficients at depth $j = 2$, by analogy with (2.38):

$$n_{\text{affect}}^{\text{left}}(2,m) = \left\lfloor \frac{m-1+\left\lfloor \frac{m-1}{2} \right\rfloor}{2} \right\rfloor .$$

The value of the inner term depends on the parity of m, therefore we distinguish between two scenarios. For m odd,

$$n_{\text{affect}}^{\text{left}}(2,m) = \left\lfloor \frac{m-1}{2} + \frac{m-1}{4} \right\rfloor = \left\lfloor \frac{3}{4}(m-1) \right\rfloor .$$

For m even (thus $m = 2k$ for some k) the term is more complicated. We exploit the fact that $\left\lfloor \frac{3}{2}k \right\rfloor = \left\lfloor \frac{3}{2}k + \frac{1}{4} \right\rfloor$ for any $k \in \mathbb{Z}$ and we obtain

$$\begin{aligned} n_{\text{affect}}^{\text{left}}(2,m) &= \left\lfloor \frac{m-1}{2} + \frac{m-2}{4} \right\rfloor \\ &= \left\lfloor \frac{3}{4}(m-1) - \frac{1}{4} \right\rfloor \\ &= \left\lfloor \frac{3}{2}k \right\rfloor - 1 = \left\lfloor \frac{3}{2}k + \frac{1}{4} \right\rfloor - 1 \\ &= \left\lfloor \frac{3}{2}\frac{m}{2} - \frac{3}{4} \right\rfloor = \left\lfloor \frac{3}{4}(m-1) \right\rfloor , \end{aligned}$$

which satisfies the theorem for $j = 2$.

For higher decomposition levels j, the procedure is similar but there are more possibilities to be taken into account. To be precise, there are 2^{j-1} branches depending not only on the parity of m, but also on its binary representation (let us call it the higher-order parity). In any case, we exploit techniques from above and a generalized formula $\left\lfloor \frac{2^j+1}{2^j}k \right\rfloor = \left\lfloor \frac{2^j+1}{2^j}k + \frac{1}{2^{j+1}} \right\rfloor$ for $k \in \mathbb{Z}$ and $j > 0$. □

Remark 2.12. For a filter of length $m \geq 3$, the number of affected coefficients is always nonzero, $n_{\text{affect}}^{\text{left}}(j,m) \geq 1$. This fact, which immediately follows from (2.39), can be seen in Table 4.

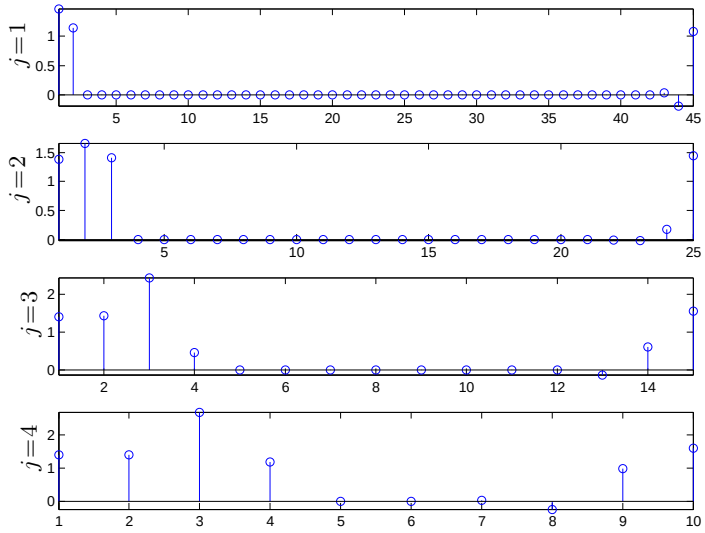

Fig. 5. Illustration of Theorem 2.13. Approximation wavelet coefficients of the signal consisting from $s = 85$ zero samples are shown. At each level of the decomposition, $j = 1, \ldots, 4$, the DWT was performed in a way that the existing coefficients were extended by ones as a dummy boundary-handling method. Filter `db3` which is of length $m = 6$ was used. From the left side, the number of the affected (nonzero) coefficients increases up to four coefficients (which is the upper limit according to Remark 2.13), from the right side it increases up to five coefficients. This in turn means that there is only a single unaffected coefficient remaining at depth four.

Table 4. Numbers of coefficients $n_{\text{affect}}^{\text{left}}(J, m)$ of the respective coefficient vectors, affected by the choice of the boundary extension type. According to Theorem 2.13.

$n_{\text{affect}}^{\text{left}}(J, m)$

$J \backslash m$	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
1	0	1	1	2	2	3	3	4	4	5	5	6	6	7
2	0	1	2	3	3	4	5	6	6	7	8	9	9	10
3	0	1	2	3	4	5	6	7	7	8	9	10	11	12
4	0	1	2	3	4	5	6	7	8	9	10	11	12	13
5	0	1	2	3	4	5	6	7	8	9	10	11	12	13

Remark 2.13. From (2.39) it is also clear that the value of $n_{\text{affect}}^{\text{left}}(j, m)$ gets closer to $(m - 2)$ with increasing j, and hits this value for j sufficiently large, see again Table 4.

Theorem 2.14. *The choice of the signal boundary extension type affects*

$$n_{\text{affect}}^{\text{right}}(j, m) = n_{\text{affect}}^{\text{left}}(j, m) + \delta_j \tag{2.40}$$

right-side coefficients at the decomposition level j, where $\delta_j \in \{0, 1\}$. Thus the number of affected right-side coefficients is always either equal to the number of

affected left-side coefficients or greater by one.

Proof. It is not quite possible to express the number of right-side affected coefficients in an explicit form like it was in the case of Theorem 2.13, due to the term $[\mathrm{par}(s)\,\mathrm{par}(m-1)]$ contained in $n_{\mathrm{affect}}^{\mathrm{right}}(1,m)$ in Eq. (2.38).

For $j=1$, (2.40) corresponds to (2.38), when $\delta = \mathrm{par}(s)\,\mathrm{par}(m-1)$ is chosen. Thus, at the first decomposition depth there are $\left(\left\lfloor\frac{m-1}{2}\right\rfloor+\delta\right)$ right-side affected coefficients. The following approach is analogous to the proof of Theorem 2.12.

If the range of influence of the rightmost coefficient from level $j=2$ contains the rightmost coefficient at level $j=1$ then the number of affected coefficients is given by

$$n_{\mathrm{affect}}^{\mathrm{right}}(2,m) = \max\left\{q \,\middle|\, (m-1)+\left\lfloor\frac{m-1}{2}\right\rfloor+\delta-2(q-1)>0\right\}.$$

The inequality can be rewritten as $q < \left\lfloor\frac{3(m-1)}{2}\right\rfloor/2+1+\frac{\delta}{2}$, which becomes $q < \frac{3(m-1)}{4}+1+\frac{\delta}{2}$ for odd m; the greatest such integer q is $\left\lfloor\frac{3(m-1)}{4}\right\rfloor$ or $\left\lfloor\frac{3(m-1)}{4}\right\rfloor+1$, which proves (2.40). For even m, the inequality becomes $q < \frac{3(m-1)}{4}-\frac{1}{4}+1+\frac{\delta}{2}$, which means that the greatest such q is, again, equal to either $\left\lfloor\frac{3(m-1)}{4}\right\rfloor$ or $\left\lfloor\frac{3(m-1)}{4}\right\rfloor+1$.

In the case of range of influence containing just the last but one coefficient from level $j=1$, the proving procedure is analogous. The proof for higher levels j is analogous. □

In the next theorem, an interesting simple property of the series of δ_j is stated.

Theorem 2.15. *If, in the sense of Theorem 2.14, there is such j for which $\delta_j = 1$ holds true, then $\delta_i = 1$ holds also for all $i \geq j$.*

Proof. By similar means as the above proofs. □

Remark 2.14. One can ask for a characterization of a setup in which *all* of the wavelet coefficients at depth J are affected. Such a situation appears when

$$0 \geq n_{\mathrm{coef}}(s,m,J) - [2\,n_{\mathrm{affect}}^{\mathrm{left}}(J,m)+\delta_J].$$

For example, the expression at the right side is precisely zero for $m=12, s=80$ and $J=2$. At depth $J=1$, 11 coefficients still remain unaffected, while at $J=3$, 6 coefficients are affected even from both sides.

Remark 2.15. Although the number of affected coefficients $n_{\mathrm{affect}}(j,m)$ can be seemed small in comparison with the input signal length, we point out that the ratio between $n_{\mathrm{affect}}(j,m)$ and the number of wavelet coefficients at level j is increasing exponentially in j! This is due to the reduction of the number of wavelet coefficients to approximately one half of the previous length at each level of decomposition. The just described dependency is shown in Fig. 5.

Table 5. Numbers of left-side samples of the reconstructed signal which can be affected by the choice of border extension type. According to Theorem 2.16.

$J\backslash^m$	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
1	0	2	2	4	4	6	6	8	8	10	10
2	0	4	8	12	12	16	20	24	24	28	32
3	0	8	16	24	32	40	48	56	56	64	72
4	0	16	32	48	64	80	96	112	128	144	160
5	0	32	64	96	128	160	192	224	256	288	320
6	0	64	128	192	256	320	384	448	512	576	640
7	0	128	256	384	512	640	768	896	1024	1152	1280
8	0	256	512	768	1024	1280	1536	1792	2048	2304	2560
9	0	512	1024	1536	2048	2560	3072	3584	4096	4608	5120
10	0	1024	2048	3072	4096	5120	6144	7168	8192	9216	10240

Theorem 2.16 corresponds with this notion and shows that the perturbation of the "border" coefficients has an extensive impact on the signal after its reconstruction.

Theorem 2.16. *Perturbation of coefficients which are affected by the choice of the boundary extension type results in change of*

$$2^j \left\lfloor (1-2^{-j})(m-1) \right\rfloor \tag{2.41}$$

original input signal samples from the left side.

Proof. Assume that all the coefficients of interest are perturbed. Theorem 2.9 defines the interval in the time-domain signal which contributes to the value of a single wavelet coefficient, or, vice versa, what range in the time-domain a single coefficient influences. For the kth coefficient at level of the decomposition j, such an interval finishes at index $2^{j-u}k = 2^j k$ in the input signal level ($u = 0$). Now it suffices to plug $n^{\text{left}}_{\text{affect}}(j,m)$ from (2.39) to get the maximum range of the modified coefficients in the time domain. □

Remark 2.16. Although (2.39) gives the *total* number of affected coefficients, it is clear that the closer the coefficients are to the "border", the greater the affect is. This fact comes from the calculation of the convolution inside the DWT, where more of the extra-appended coefficients contribute to the resultant coefficient at the border, compared to the resultant coefficient located more in the center of the respective vector.

Similarly, the *total* number of affected reconstructed signal samples is given by (2.41), but those samples closer to the signal border are affected more strongly, as can be evidenced in Fig. 6.

Remark 2.17. Similar to Remark 2.14, one can ask when a situation appears such that all the signal samples are affected by the boundary treatment method. It appears in setups which satisfy $\left\lfloor (1-2^{-j})(m-1) \right\rfloor \geq \frac{s}{2^{j+1}}$.

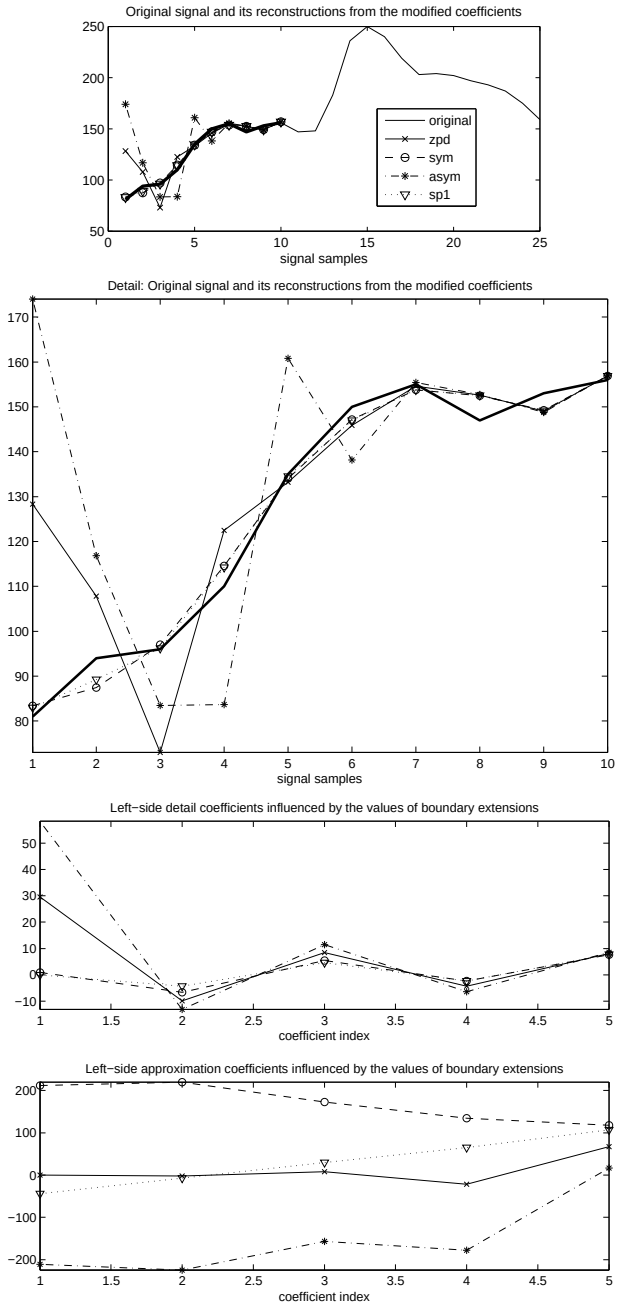

Fig. 6. Illustration of Theorem 2.16. The top graph shows the left end of the processed signal. Its original behaviour is emphasized. The signal was subject to DWT, depth $J = 1$, filter `db6` of length $m = 12$, four times with four different types of boundary extension (corresponding to the legend: extension with zeros, symmetrical, antisymmetrical, first order polynomial). The second graph brings the close-up view of the situation. The first $n^{\mathrm{left}}_{\mathrm{affect}}(1, 12) = 5$ detail and approximation coefficients are shown in the bottom two graphs. Prior to the reconstruction(s), which are depicted using the respective line styles, these five detail coefficients were hard-thresholded with threshold $\lambda = 9$ and then magnified five times to bring out the differences in the first and second graphs.

3. Inverse Discrete Wavelet Transform (iDWT)

This section contains a detailed analysis of the iDWT algorithm. Similar to the forward transform, the inverse transform can be performed by a matrix multiplication. For orthogonal filters, this matrix will be just the transpose of the forward one. Nevertheless, Mallat's algorithm for the inverse transform is again more effective to be used:

The approximation and detail coefficient vectors are upsampled (i.e., zeros are inserted in between pairs of coefficients) and convolved with reconstructing filters $\tilde{\mathbf{h}}$ and $\tilde{\mathbf{g}}$, respectively. The results are summed up, and after cutting out the central part, we end up with approximation coefficients belonging to a level smaller by one than the initial level. This procedure is again done iteratively until the zero level of decomposition is reached, which means we get the reconstructed signal in the time domain. A single recursion of the inverse pyramidal algorithm is shown schematically in Fig. 7.

When performing iDWT, the border extension type used in the forward DWT (see Section 2.1) does not need to be known; for purposes of iDWT, it makes no difference whether, for example, symmetric or antisymmetric extension was used. On the other hand, the knowledge of the *category* of the border extension used *must be known* (items 1 to 5 in the list). This is for the reason of correctly handling the numbers of coefficients to be cut at the "borders". Like in the analysis of DWT, below we will consider only category 3, i.e., extending signal boundaries by defining samples outside of the domain.

Remark 3.1. It is worth mentioning that, even though a very bad boundary handling method is used, the reconstructed signal is identical to the original signal until there is a perturbation of the wavelet coefficients. However, the values of the "border" coefficients are "wrong", and as such they are suitable neither for analysis nor for processing (namely nonlinear processing, recall the thresholding example in Fig. 6).

3.1. *Analysis of pyramidal iDWT algorithm*

Algorithm 2 (pyramidal algorithm for iDWT).
Input: a pair of the wavelet reconstruction filters of length m (in the case of biorthogonal filters, the shorter one is zero-padded to the length of the longer one, as described in Sec. 1.1.1) — the highpass filter $\tilde{\mathbf{g}}$ and the lowpass filter $\tilde{\mathbf{h}}$; the number of signal samples in the time domain s; the even/odd type of the upsampling ε, and especially the $J+1$ vectors of wavelet coefficients $\mathbf{a}^{(J)}, \mathbf{d}^{(J)}, \mathbf{d}^{(J-1)}, \ldots, \mathbf{d}^{(1)}$, which arose from DWT, Alg. 1, using the same choice of ε.
Output: the reconstructed signal, stored in vector $\mathbf{a}^{(0)}$.

(1) Set $j := J$.
(2) A single reconstruction level:

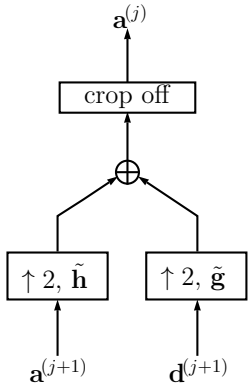

Fig. 7. Single-level wavelet reconstruction. The vector of approximation coefficients at level j, i.e. $\mathbf{a}^{(j)}$, is calculated from the vectors $\mathbf{a}^{(j+1)}$ and $\mathbf{d}^{(j+1)}$ using upsampling, filtering with $\tilde{\mathbf{h}}$ and $\tilde{\mathbf{g}}$, and cropping off the unnecessary border coefficients.

(a) *Upsampling.* Perform ε-type upsampling of vectors $\mathbf{a}^{(j)}$ and $\mathbf{d}^{(j)}$.
(b) *Filtering.* Filter the upsampled vectors, i.e., perform the convolution with reconstruction filters $\tilde{\mathbf{h}}$ and $\tilde{\mathbf{g}}$, respectively.
(c) *Summing.* Add up the outcomes of both convolutions.
(d) *Cropping.* For the resultant vector, keep just its "middle" part. The starting index is $m - \mathrm{par}(m) - 1 + 2\varepsilon$ and the length of the portion to be kept equals the length of vector $\mathbf{d}^{(j-1)}$. When $j = 1$, define s to be the length of the (nonexisting) vector $\mathbf{d}^{(0)}$.

Denote the result $\mathbf{a}^{(j-1)}$.

(3) Decrease j by one. If $j > 0$, skip to step 2., otherwise the algorithm ends.

Remark 3.2. According to Eq. (1.7), the ε-type upsampling in step 2(a) results in a vector with $2p - 1 + 2\varepsilon$ entries, where p denotes the length of the initial vector of coefficients at depth J, thus $p = n_{\mathrm{coef}}(s, m, J, \varepsilon)$. The vector after filtering in step 2(b) has the length $o = (2p - 1 + 2\varepsilon) + m - 1 = 2p + m + 2\varepsilon - 2$. After the cutting off in step 2(d), the length of the vector is equal to the length of the vector of detail coefficients from the appropriate level of decomposition and it is therefore suitable for calculations in the subsequent iteration.

Remark 3.3. We will show now why the knowledge of the original signal length s is actually needed for the reconstruction algorithm 2. Then, in Remark 3.4, it will be shown that in fact its parity $\mathrm{par}(s)$ is sufficient.

Formula (2.10) gives the number of coefficients that are calculated from the signal using a filter of length m in a single iteration: $p = \frac{s+m-1}{2} + \left(\varepsilon - \frac{1}{2}\right) \mathrm{par}(s + m - 1)$. In turn, for example, signals of lengths $s = 255$ and $s = 256$ produce the same number of wavelet coefficients, assuming filters of even length and

$\varepsilon = 0$, because

$$\frac{255+m-1}{2} - \frac{1}{2}\cdot 0 = \frac{256+m-1}{2} - \frac{1}{2}\cdot 1.$$

From this it is clear that in each single reconstruction level it is necessary to know the number of coefficients in the subsequent level, to avoid a mistake in the reconstruction length. But such information is naturally available for all $j \in \{J, \ldots, 3, 2\}$ as they can be easily identified from the lengths of the (stored) detail wavelet vectors $\mathbf{d}^{(j-1)}$. However, the vector denoted $\mathbf{d}^{(0)}$ does not exist and therefore the length of the original signal must be supplied.

Remark 3.4. In fact, it is sufficient to supply just the parity of the original signal length, i.e. whether it is odd or even. Indeed, because using (2.11) we get $p = \frac{s+m-1}{2}$ and thus $s = 2p - m + 1$ for $s + m - 1$ even. For $s + m - 1$ odd we get $p = \left\lfloor \frac{s+m-1}{2} \right\rfloor + \varepsilon = \frac{s+m-2}{2} + \varepsilon = \frac{s+m+2\varepsilon-2}{2}$ and therefore $s = 2p - m - 2\varepsilon + 2$. To summarize, if m and the parity of the original signal s are known, then the original signal length can be calculated by

$$s = \begin{cases} 2p - m + 1 & \text{for even } s + m - 1, \\ 2p - m - 2\varepsilon + 2 & \text{for odd } s + m - 1. \end{cases} \tag{3.1}$$

Remark 3.5. The coefficients cropping in step 2d in Alg. 2 pertains coefficients starting at index $m - \mathrm{par}(m) - 1 + 2\varepsilon$. This remark shows the derivation of this formula and offers its interpretation.

Speaking in simple terms, the "middle" part that is to be retained has length s and it is cut off from the vector after the convolution, which has length o. The numbers of coefficients that are left out at both ends may differ and depend on the values of s, m and ε. E.g. for $s = 6$, $m = 6$, $\varepsilon = 0$ we have $o = 16$ and the number of coefficients that are left out is equal to 5 at both ends. But for $s = 6$, $m = 9$, $\varepsilon = 0$ the convolution results in $o = 23$ entries and the number of coefficients that are left out is equal to 7 from the left side and 10 from the right side.

A general formula for the starting index is

$$start = \left\lfloor \frac{o-s}{2} \right\rfloor - \mathrm{par}(m) + \varepsilon[1 - \mathrm{par}(s+m-1)] + 2\varepsilon, \tag{3.2}$$

which suggests that the smallest possible number of left-side coefficients to be left out is preferred (owing to the $\lfloor\cdot\rfloor$ operator) in the cropping process, nevertheless this amount is still corrected by parities of m, s and ε. Using (3.1) and assuming

$(s+m-1)$ to be even, formula (3.2) becomes

$$\begin{aligned} start &= \left\lfloor \frac{o-s}{2} \right\rfloor - \mathrm{par}(m) + \varepsilon \cdot 1 + 2\varepsilon \\ &= \left\lfloor \frac{(2p+m+2\varepsilon)-(2p-m+1)}{2} \right\rfloor - \mathrm{par}(m) + 3\varepsilon \\ &= \left\lfloor \frac{2m-2\varepsilon-1}{2} \right\rfloor - \mathrm{par}(m) + 3\varepsilon \\ &= m - \varepsilon + \left\lfloor -\frac{1}{2} \right\rfloor - \mathrm{par}(m) + 3\varepsilon \\ &= m - \mathrm{par}(m) - 1 + 2\varepsilon. \end{aligned} \tag{3.3}$$

The simplification of (3.2) to (3.3) can be obtained also for $(s+m-1)$ being odd.

Remark 3.6. What happens when distinct types of up-/downsampling ε are used for the forward and inverse transforms, respectively? In Section 1.1.4 we stated that the even-type and odd-type sampling differ by two extra zero samples, one at the beginning and one at the end of the vector. Thus the convolution leads to the same values using either $\varepsilon = 0$ or $\varepsilon = 1$, the only difference is the lengths of the resulting vectors (the longer one contains two extra zeros). However, the left and right borders of the cut in step 2d are derived (see Remark 3.5) assuming the agreement of ε between the forward and inverse transforms, so in the other case the cropping step would result in a failure of the reconstruction procedure.

4. Software

The paper is accompanied by several MATLAB functions, which implement key parts of DWT and iDWT and demonstrate the properties described throughout the paper. This set is not intended to form a comprehensive toolbox for computing the wavelet transform like for example [19]. Presented functions are independent on the official Wavelet Toolbox [9].

To obtain particular wavelet filters, the `w_filters.m` function can be used; it contains impulse responses of filters available from the just mentioned commercial toolbox (which provides function called `wfilters`).

The functions are compatible with biorthogonal filters of odd lengths (in contrast to [9], see Remark 1.1 and description of `short_filt.m` as well). The files can be used as building blocks for other programs under the "BSDFree" licence. Testing was performed in Matlab versions R2008b a R2010a.

The functions are downloadable from URL [17]. Description of the `m`-files follows:

`dwtmatrix` .... generates the matrix for performing the forward DWT, see Section 2.2, Eq. (2.7)

`sldwt` ..... performs single-level DWT, the border extension is by zeros

`slidwt` ..... performs single-level iDWT

`mldwt` ..... performs multi-level DWT, the border extension is by zeros

`mlidwt` ..... performs multi-level iDWT
`downsample` ... performs dyadic downsampling with chosen ε-parity
`upsample` ..... performs dyadic upsampling using ε-parity
`parity` ..... returns the parity of the input number
`n_coef` ..... returns the number of coefficients at a chosen decomposition produced by the DWT; corresponds to $n_{\mathrm{coef}}(s, m, J)$ as defined in (2.13)
`n_coef_total` . returns the total number of coefficients of the transform, see (2.9)
`n_samp` ..... returns the number of signal samples necessary for computation of q subsequent wavelet coefficients; corresponds to $n_{\mathrm{samp}}(q, j, m)$ in (2.25).
`ran` ..... returns the range of influence of a specified coefficient, $\mathrm{ran}(k, j, u, m)$ from (2.30)
`n_shift` ..... returns the shift in terms of signal samples for specified wavelet coefficients; corresponds to (2.33).
`n_shared` ..... returns the number of coefficients/signal samples that are shared by a pair of coefficients; corresponds to (2.34).
`n_affect_l` .... returns the number of left-side coefficients that are affected by the boundary handling; corresponds to (2.39).
`short_filt` ... if both the impulse responses of a pair of input biorthogonal filters contain zeros as a first sample, this function shortens the responses by this sample; this is an auxiliary function intended mainly for manipulation with Matlab's Wavelet Toolbox filters, which are always padded with zeros to even lengths.
`w_filters` .... returns four impulse responses $\mathbf{h}, \mathbf{g}, \tilde{\mathbf{h}}, \tilde{\mathbf{g}}$ according to the input wavelet name. The biorthogonal filters are shortened as described in `short_filt`.

There are more files in the archive, mainly for testing and demo purposes. For example, `test_dwt.m` compares our transforms with Matlab's. (Matlab's `wavedec` does not allow to choose ε, so again, we cannot forget that Matlab Wavelet Toolbox uses the non-shortened filters, and thus if a comparison involving the biorthogonal filters is performed, we have to use $\varepsilon = 1$ to get the "compatibility" back.)

5. Conclusion

The algorithms of forward and inverse discrete-time wavelet transform for finite-length signals were presented in detail, so that the reader can easily implement them, or/and utilize the attached Matlab files. However, optimization of the algorithms (there are several possibilities [30,18]) is beyond the scope of the article.

The principal contribution of the article is the detailed analysis of the algorithms' properties and consequences. The text answers the questions about the precise relationship between the signal samples and its wavelet coefficients, as they were asked in the Introduction.

Several generalizations of the results can be easily made:

- Wavelet packets differ from the ordinary DWT just in the branching of the decomposition tree. This means that the values of the packet coefficients are different but the actual lengths of the sets of coefficients in the individual levels remain the same as in the DWT. Thus all the results trivially apply to wavelet packets as well.
- Multi-dimensional wavelet transforms are in most cases performed as a sequence of the one-dimensional ones. This is called separability and the presented theorems apply to the multidimensional case dimension-by-dimension using this principle. (A typical question from the field of image processing that is answered by the article might be: Which coefficients are affected by editing this region of pixels?)
- As was already stated above, many of the results apply not only to the boundary handling method selected for our analysis, but also for the other methods.

Acknowledgments

This work was supported by projects CZ.1.05/2.1.00/03.0072, CZ.1.07/2.3.00/20.0094 and CZ.1.07/2.2.00/28.0062 co-financed by the European Social Fund and the state budget of the Czech Republic and by the Austrian Science Fund (FWF) START-project FLAME (Frames and Linear Operators for Acoustical Modeling and Parameter Estimation; Y 551-N13).

References

1. A. Cohen, I. Daubechies and P. Vial, Wavelets on the interval and fast wavelet transforms, *Applied and Computational Harmonic Analysis* **1**(1) (1993) 54–81.
2. R. R. Coifman and D. L. Donoho, Translation invariant de-noising, *Lecture Notes in Statistics* **103** (1995) 125–150.
3. I. Daubechies and W. Sweldens, Factoring wavelet transforms into lifting steps, *J. Fourier Anal. Appl.* **4**(3) (1998) 245–267.
4. A. Jensen and A. l. Cour-Harbo, *Ripples in Mathematics: The Discrete Wavelet Transform* (Springer-Verlag, 2001).
5. I. Kharitonenko, X. Zhang and S. Twelves, A wavelet transform with point-symmetric extension at tile boundaries, *IEEE Transactions on Image Processing* **11**(12) (2002) 1357–1364.
6. S. Mallat, A theory for multiresolution signal decomposition: The wavelet representation, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **11** (1989) 674–693.
7. S. Mallat, *A Wavelet tour of signal processing*, 2 edn. (Academic Press, 1999).
8. S. Mallat, *A Wavelet tour of signal processing, The Sparse way*, 3 edn. (Academic Press, Boston, 2008).
9. M. Misiti, Y. Misiti, G. Oppenheim and J.-M. Poggi, *Wavelet Toolbox Users Guide.* The Mathworks, Inc., (2000).
10. H. Mota, F. Vasconcelos and R. Silva, Real-time wavelet transform algorithms for the processing of continuous streams of data, *Intelligent Signal Processing, 2005 IEEE International Workshop on*, (2005), pp. 346–351.

11. G. P. Nason and B. Silverman, The stationary wavelet transform and some statistical applications, *Lecture Notes in Statistics* **103** (1995) 281–299.
12. J. Nealand, A. Bradley and M. Lech, Overlap-save convolution applied to wavelet analysis, *IEEE Signal Processing Letters* **10**(2) (2003) 47–49.
13. D. Onchis and E. Sanchez, The flexible Gabor-wavelet transform for car crash signal analysis, *Int. J. Wavelets Multiresolut. Inf. Process.* **7**(4) (2009) 481–490.
14. Z. Prusa and P. Rajmic, Segmented computation of wavelet transform via lifting scheme, *34th International Conference on Telecommunications and Signal Processing*, Budapest (2011), pp. 433–437.
15. P. Rajmic, Algorithms for segmentwise computation of forward and inverse discrete-time wavelet transform, *Journal of Concrete and Applicable Mathematics, Special Issues on Applied Mathematics and Approximation Theory* **8**(3) (2010) 393–406.
16. P. Rajmic and J. Maly, Boundary effects in the wavelet transform of finite discrete signals, *Proceedings of the conference TSP'2007*, (Brno University of Technology, Brno, 2007), pp. 81–84.
17. P. Rajmic, Z. Prusa and R. Konczi, Matlab codes accompanying the paper Discrete Wavelet Transform of Finite Signals: Detailed Study of the Algorithm (2012).
18. P. Rajmic, Z. Prusa and R. Konczi, VST plugin module performing wavelet transform in real-time, *Proceedings of the 15th international conference on digital audio effects, DAFx12*, (University of York, 2012).
19. P. L. Søndergaard, B. Torrésani and P. Balazs, The Linear Time Frequency Analysis Toolbox, *International Journal of Wavelets, Multiresolution Analysis and Information Processing* **10**(4) (2012).
20. G. Strang and T. Nguyen, *Wavelets and Filter Banks* (Wellesley College, 1996).
21. C. Taswell and K. C. McGill, Wavelet transform algorithms for finite-duration discrete-time signals, *Progress in Wavelet Analysis and Applications, Proceedings of the International Conference on Wavelets and Applications*, eds. Y. Meyer and S. Roques (1993), pp. 221–224.
22. C. Taswell and K. C. McGill, Algorithm 735: Wavelet transform algorithms for finite-duration discrete-time signals, *ACM Trans. Math. Softw.* **20** (September 1994) 398–412.
23. D. S. Taubman and M. W. Marcellin, *JPEG2000: Image compression fundamentals, standards, and practice* (Kluwer Academic Publishers, USA, 2002).
24. F. Truchetet and O. Laligant, Review of industrial applications of wavelet and multiresolution-based signal and image processing, *J. Electronic Imaging* **17**(3) (2008) p. 031102.
25. M. Unser and T. Blu, Mathematical properties of the JPEG2000 wavelet filters, *IEEE Transactions on Image Processing* **12** (2003) 1080–1090.
26. R. Vargic, *Wavelets and filter banks* (STU in Bratislava, Bratislava, 2004). in Slovak.
27. M. Vetterli and C. Herley, Wavelets and filter banks: theory and design, *Signal Processing, IEEE Transactions on* **40** (sep 1992) 2207–2232.
28. M. Vetterli and J. Kovačević, *Wavelets and Subband Coding*, 2 edn. (Prentice Hall, 2007).
29. B. Vidakovic, *Statistical modeling by wavelets* (John Wiley & Sons, 1999).
30. M. V. Wickerhauser, *Adapted wavelet analysis from theory to software* (IEEE, 1994).