# Chapter 3

# The Lifting Scheme

In this chapter we introduce the lifting scheme. The lifting scheme is an efficient implementation of a wavelet transform algorithm. Furthermore it can easily be converted to operate on integer data. The inverse transform is trivial to find and always provides perfect reconstruction of the input data.

All classical wavelet transforms can be implemented using the lifting scheme by a factorization in lifting steps. We work out the details of integer wavelet transforms for a class of biorthogonal wavelets (Cohen-Daubechies-Feauveau).

## 3.1 Introduction

The lifting scheme is an algorithm to calculate wavelet transforms in an efficient way. It was developed first as a method to improve a given wavelet transform to obtain some specific properties. Later it was extended to a generic method to create so-called second-generation wavelets. In the one-dimensional case, these are wavelets that are not necessarily dilates and translates of one function. They are much more flexible and can be used to define a wavelet basis on an interval or on an irregular grid, or even on a sphere.

Second-generation wavelets are more general in the sense that all the classical wavelets can be generated by the lifting scheme. In fact, the decomposition of a classical wavelet filter into lifting steps can be easily obtained via the Euclidean algorithm [14].

The theory behind the type of wavelets described in Section 2.3 relies heavily on Fourier analysis, while the lifting scheme can also be used to introduce wavelets without using the concept of Fourier transform. This allows to explain the basic ideas behind wavelets in a much simpler way.

Several introductions to the lifting scheme are available [48, 49, 40, 47].

## 3.2 Predict and Update

The wavelet transform of a one-dimensional signal is a multi-resolution representation of that signal where the wavelets are the basis functions which at each resolution level decorrelate the signal.

Thus at each level, the (low-pass part of the) signal is split into a high-pass and a low-pass part. These high-pass and low-pass parts are obtained by applying corresponding wavelet filters. In general these filters are coupled if certain conditions are to be fulfilled, like for example perfect reconstruction: there have to exist filters to perform the inverse transform without any data loss.

The lifting scheme is an efficient implementation of these filtering operations at each level when computing a discrete wavelet transform. So suppose that the low-resolution part of a signal at level $j+1$ is given and that it consists of a data set which we represent by $\lambda_{j+1}$. This set is transformed into two other sets at level $j$: the low-resolution part $\lambda_j$ and the high-resolution part $\gamma_j$. This is obtained first by just splitting the data set $\lambda_{j+1}$ into two separate data subsets. Traditionally this is done by separating the set of even samples and the set of odd samples. Such a splitting is sometimes referred to as the *lazy wavelet transform*. Doing just this of course does not improve our representation of the signal. Therefore, the next step is to recombine these two sets in several subsequent lifting steps which decorrelate the two signals.

Lifting steps usually come in pairs of a primal and a dual lifting step. A *dual lifting* step can be seen as a prediction: the data $\gamma_j$ are "predicted" from the data in the subset $\lambda_j$. When the signals are still highly correlated, then such a prediction will usually be very good, and thus we do not have to keep this information in both signals. That is why we can store only the part of $\gamma_j$ that differs from its prediction (the prediction error). Thus $\gamma_j$ is replaced by $\gamma_j - \mathcal{P}(\lambda_j)$ where $\mathcal{P}$ represents the prediction operator. This is the real decorrelating step.

However, the new representation has lost certain basic properties, which one usually wants to keep, like for example the mean value of the signal. To restore this property, one needs a *primal lifting* step, whereby the set $\lambda_j$ is updated with data computed from the (new) subset $\gamma_j$. Thus $\lambda_j$ is replaced by $\lambda_j + \mathcal{U}(\gamma_j)$ with $\mathcal{U}$ some updating operator.

In general, several such lifting steps can be applied in sequence to go from level $j+1$ to level $j$.

The principles that were just explained become very clear in the case of the simple CDF $(2,2)$ transform that we give below in Section 3.2.2. The odd samples are predicted by linear interpolation of the neighboring even samples and are therefore replaced by the interpolation error. The even samples are updated to preserve the mean value of the signal.

To recapitulate, let us consider a simple lifting scheme with only one pair of lifting steps to go from level $j+1$ to level $j$.

**Splitting** (*lazy wavelet transform*) Partition the data set $\lambda_{j+1}$ into two distinct data sets $\lambda_j$ and $\gamma_j$.

**Prediction** (*dual lifting*) Predict the data in the set $\gamma_j$ by the data set $\lambda_j$ and replace $\gamma_j$ by the prediction error:

$$\gamma_j \leftarrow \gamma_j - \mathcal{P}(\lambda_j).$$

**Update** (*primal lifting*) Update the data in the set $\lambda_j$ by the data in set $\gamma_j$:

$$\lambda_j \leftarrow \lambda_j + \mathcal{U}(\gamma_j).$$

These steps can be repeated by iteration on the $\lambda_j$, creating a multi-level transform or multi-resolution decomposition.

### 3.2.1 The Inverse Transform

One of the great advantages of the lifting scheme realization of a wavelet transform is that it decomposes the wavelet filters into extremely simple elementary steps, and each of these steps is very easily invertible. As a result, the inverse wavelet transform can always be obtained immediately from the forward transform. The inversion rules are obvious: revert the order of the operations, invert the signs in the lifting steps, and replace the splitting step by a merging step. Thus, inverting the three step procedure above results in

**Inverse update** $$\lambda_j \leftarrow \lambda_j - \mathcal{U}(\gamma_j),$$

**Inverse prediction** $$\gamma_j \leftarrow \gamma_j + \mathcal{P}(\lambda_j),$$

**Merge** $$\lambda_{j+1} \leftarrow \lambda_j \cup \gamma_j.$$

### 3.2.2 Example: Linear Prediction

From the second-generation viewpoint, one transform step of a discrete one-dimensional signal $x = \{x_k\}$ looks like:

**Splitting** Split the signal $x$ (i.e. $\lambda_{j+1}$) into even samples (i.e. $\lambda_j$) and odd samples (i.e. $\gamma_j$):

$$\begin{aligned} s_i &\leftarrow x_{2i}, \\ d_i &\leftarrow x_{2i+1} \end{aligned}$$

**Prediction** Predict the odd samples using linear interpolation:

$$d_i \leftarrow d_i - \frac{1}{2}(s_i + s_{i+1}).$$

**Update** Update the even samples to preserve the mean value of the samples:

$$s_i \leftarrow s_i + \frac{1}{4}(d_{i-1} + d_i).$$

As a result, the signal $s = \{s_k\}$ is a coarse representation of the original signal $x$, while the signal $d = \{d_k\}$ contains the high-frequency information that is lost when going from resolution level $j+1$ to resolution level $j$.

The same kind of interpolation scheme can be used for interpolating polynomials of higher degree. These lifting schemes need some special care at the boundaries if one wants the wavelets to live explicitly on the discrete set where the data are defined. A whole family of lifting schemes can be constructed in this way [50], of which the above example is just the most simple case.

Note that this transform works on one-dimensional data. For two-dimensional data, it can be applied row and column wise, resulting in a tensor product wavelet transform. This is a separable transform.

Later we will see that this scheme turns out to be the lifting equivalent of the popular Cohen, Daubechies and Feauveau [9] classical biorthogonal wavelets with two vanishing moments for both the primal and dual wavelet (hence named CDF $(2,2)$ wavelet).

Thanks to the lifting scheme, the accompanying wavelet transform can be implemented in an efficient way (see e.g. [57]).

### 3.2.3 Flexibility

This scheme is much more flexible than the one used for classical wavelets, based on Fourier analysis. Using lifting one can create wavelets for higher dimensions, on irregular samplings, intervals, triangularizations, ... [40].

A simple method to construct two-dimensional wavelets using lifting will be given in Section 7.2.

## 3.3 Unifying the Two Paradigms

The previous section introduced the lifting scheme from a second-generation viewpoint. However, all classical wavelet transforms can be implemented using this powerful scheme.

### 3.3.1 The Filter Bank Algorithm

The basic block in a discrete wavelet transform (DWT) [26] is a filter bank, consisting of 2 filters (cfr. Figure 2.3). A discrete signal $S$ is filtered by a high-pass filter $\widetilde{g}$ and a low-pass filter $\widetilde{h}$, and downsampled. The results are a high-pass signal

$HP$ and a low-pass signal $LP$, each containing half as much samples as the input signal $S$.

For the inverse transform, first the signals $HP$ and $LP$ are upsampled by putting zeros in between every sample. After that they are filtered by the filters $g$ and $h$ and the result is added together. In the case of perfect reconstruction — we only consider perfect reconstruction filter banks here — the resulting signal is equal to the original signal $S$, and the filters $g$, $h$, $\widetilde{g}$ and $\widetilde{h}$ are called *wavelet filters*. The filters $\widetilde{g}$ and $\widetilde{h}$ are called dual filters w.r.t. $g$ and $h$.

The signal $LP$ resembles the original signal $S$ very much: it is a representation of $S$ at a lower resolution level. On the other hand, $HP$ contains the detail information that is lost by making the transition to that lower resolution level [34].

If the low-frequency information is treated as a new signal and passed through the same filter bank, and this step is repeated several times, after some iterations a very low-frequency signal is retained. Together with the detail information for the different resolution levels, it represents the original signal decomposed in several resolution levels. This is called a *wavelet transform*.

### 3.3.2 Polyphase Representations

The $z$-transform of a signal $x = \{x_k\}$ is defined as

$$x(z) = \sum_k x_k z^{-k},$$

whenever this makes sense. Thus, the $z$-transform of a finite impulse response (FIR) filter $h = \{h_{k_b}, \ldots, h_{k_e}\}$ is a Laurent polynomial $h(z)$ given by

$$h(z) = \sum_{k=k_b}^{k_e} h_k z^{-k}.$$

Assuming $h_{k_b}, h_{k_e} \neq 0$, we define the degree of such a Laurent polynomial as

$$|h(z)| = k_e - k_b.$$

For consistency we set the degree of the zero polynomial equal to $-\infty$.

Filtering a signal $x$ by a filter $h$ is easily described in the $z$-domain as an ordinary multiplication: $y(z) = h(z)x(z)$.

Subsampling a signal $x = \{x_k\}$ corresponds to keeping only the even samples $x^s = \{x_{2k}\}$. Thus its $z$-transform is given by

$$x^s(z) = x_e(z) = \sum_k x_{2k} z^{-k}$$

where $x_e(z)$ is the even part of $x(z)$, i.e. we define $x_e(z^2)$ as

$$x_e(z^2) = \frac{x(z) + x(-z)}{2}.$$

Defining the odd part $x_o(z^2)$ in a similar way

$$x_o(z^2) = \frac{z}{2}[x(z) - x(-z)] = \sum_k x_{2k+1} z^{-2k}$$

we can decompose $x(z)$ as

$$x(z) = x_e(z^2) + z^{-1} x_o(z^2).$$

The filtering part of the filter bank decomposition described above can be written as

$$\begin{bmatrix} lp(z) \\ hp(z) \end{bmatrix} = \begin{bmatrix} h(z) \\ g(z) \end{bmatrix} x(z).$$

while the subsampling step corresponds to

$$\begin{aligned} LP(z^2) &= \frac{lp(z) + lp(-z)}{2} = lp_e(z^2), \\ HP(z^2) &= \frac{hp(z) + hp(-z)}{2} = lp_o(z^2). \end{aligned}$$

This can be combined as

$$\begin{bmatrix} LP(z^2) \\ HP(z^2) \end{bmatrix} = \frac{1}{2} \begin{bmatrix} h(-z) & h(z) \\ g(-z) & g(z) \end{bmatrix} \begin{bmatrix} x(-z) \\ x(z) \end{bmatrix}.$$

However, by first filtering and then subsampling, we first compute all the coefficients and then throw away half of the work done. It would be more efficient if the subsampling step could be done before the filtering, which means that we only compute the even parts of $lp$ and $hp$. Obviously

$$lp_e(z) = [h(z)x(z)]_e = h_e(z)x_e(z) + z^{-1} h_o(z) x_o(z).$$

Thus $\lambda(z) = lp_e(z)$ and $\gamma(z) = hp_e(z)$ are obtained as

$$\begin{bmatrix} \lambda(z) \\ \gamma(z) \end{bmatrix} = P(z) \begin{bmatrix} x_e(z) \\ z^{-1} x_o(z) \end{bmatrix},$$

where $P(z)$ is the *polyphase matrix*

$$P(z) = \begin{bmatrix} h_e(z) & h_o(z) \\ g_e(z) & g_o(z) \end{bmatrix}.$$

A filter pair $(h, g)$ is *complementary* if the corresponding polyphase matrix $P(z)$ has determinant 1. For wavelet filter pairs, this is always the case because of the perfect reconstruction condition.

The *lazy wavelet transform* is the most simple wavelet transform. It does nothing but split the signal in even and odd components. Its polyphase matrix is the unit matrix:

$$P_{\text{lazy}}(z) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}.$$

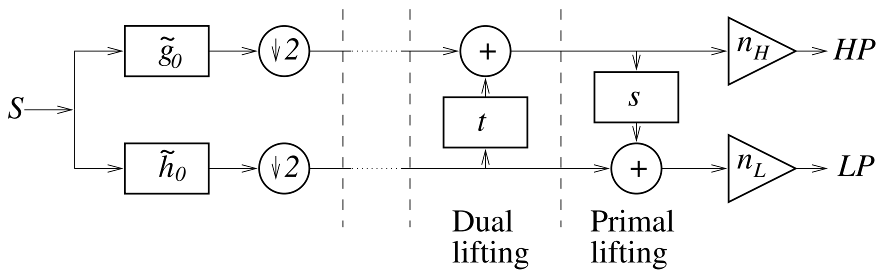


Figure 3.1: The lifting scheme: starting from wavelet filters $\widetilde{g}_0$ and $\widetilde{h}_0$ we *lift* the signal to obtain another wavelet decomposition. The final step is a normalization with factors $n_H$ and $n_L$.

### 3.3.3 Primal and Dual Lifting

The lifting scheme [48, 49, 47] found its roots in a method to improve a given wavelet transform to obtain some specific properties.

With *primal lifting*, starting from two complementary finite filters $h$ and $g$, a new finite filter $h^{\text{new}}$ complementary to $g$ is created:

$$h^{\text{new}}(z) = h(z) + s(z^2)g(z),$$

or, using the polyphase representation:

$$P^{\text{new}}(z) = \begin{bmatrix} h_e^{\text{new}}(z) & h_o^{\text{new}}(z) \\ g_e(z) & g_o(z) \end{bmatrix} = \begin{bmatrix} 1 & s(z) \\ 0 & 1 \end{bmatrix} \begin{bmatrix} h_e(z) & h_o(z) \\ g_e(z) & g_o(z) \end{bmatrix}.$$

With *dual lifting*, starting from two complementary finite filters $h$ and $g$, a new finite filter $g^{\text{new}}$ complementary to $h$ is created:

$$g^{\text{new}}(z) = g(z) + t(z^2)h(z),$$

or, using the polyphase representation:

$$P^{\text{new}}(z) = \begin{bmatrix} h_e(z) & h_o(z) \\ g_e^{\text{new}}(z) & g_o^{\text{new}}(z) \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ t(z) & 1 \end{bmatrix} \begin{bmatrix} h_e(z) & h_o(z) \\ g_e(z) & g_o(z) \end{bmatrix}.$$

This process can be repeated, obtaining a *cakewalk* of alternating primal and dual lifting steps, as shown in Figure 3.1. If the initial $P(z)$ is the unit matrix — i.e. we start from the lazy wavelet transform — the result is a product of unit upper and lower triangular $2 \times 2$ matrices of Laurent polynomials.

### 3.3.4 Decomposition into Lifting Steps

Daubechies and Sweldens [14] proved that *any* polyphase matrix representing a wavelet transform with finite filters can be factored in a finite product of unit upper and lower triangular $2 \times 2$ matrices, and a diagonal normalization matrix:

$$P(z) = \underbrace{\begin{bmatrix} K_1 & 0 \\ 0 & K_2 \end{bmatrix}}_{\text{normalization}} \prod_{i=m}^{1} \left\{ \overbrace{\begin{bmatrix} 1 & s_i(z) \\ 0 & 1 \end{bmatrix}}^{\text{primal lifting}} \underbrace{\begin{bmatrix} 1 & 0 \\ t_i(z) & 1 \end{bmatrix}}_{\text{dual lifting}} \right\}. \qquad (3.1)$$

with $s_i(z)$ and $t_i(z)$ Laurent polynomials and $K_1$ and $K_2$ normalization factors. It can be shown [14] that, based on a Euclidean type algorithm, it is possible to obtain any wavelet transform by performing a lazy wavelet transform followed by alternating primal and dual lifting steps, and a normalization.

From this it immediately follows that the inverse transform can be written as

$$P^{-1}(z) = \prod_{i=1}^{m} \left\{ \begin{bmatrix} 1 & 0 \\ -t_i(z) & 1 \end{bmatrix} \begin{bmatrix} 1 & -s_i(z) \\ 0 & 1 \end{bmatrix} \right\} \begin{bmatrix} 1/K_1 & 0 \\ 0 & 1/K_2 \end{bmatrix}.$$

How can we find the $s_i(z)$ and $t_i(z)$? To decompose the polyphase matrix using primal lifting as

$$P(z) = \begin{bmatrix} h_e(z) & h_o(z) \\ g_e(z) & g_o(z) \end{bmatrix} = \begin{bmatrix} h_e(z) & h_o^{\text{new}}(z) \\ g_e(z) & g_o^{\text{new}}(z) \end{bmatrix} \begin{bmatrix} 1 & s(z) \\ 0 & 1 \end{bmatrix}$$

we have to find Laurent polynomials $s(z)$, $h_o^{\text{new}}(z)$, and $g_o^{\text{new}}(z)$ such that

$$\begin{aligned} h_o(z) &= s(z)h_e(z) + h_o^{\text{new}}(z), \\ g_o(z) &= s(z)g_e(z) + g_o^{\text{new}}(z), \end{aligned}$$

with (recall $|\cdot|$ means degree)

$$\begin{aligned} |h_o^{\text{new}}(z)| &< |h_o(z)|, \\ |g_o^{\text{new}}(z)| &< |g_o(z)|. \end{aligned}$$

This corresponds to two *long division with remainder* schemes of Laurent polynomials, with a common quotient $s(z)$. This corresponds to a Euclidean algorithm applied to $h_o(z)$ and $h_e(z)$. The same Euclidean algorithm can be applied to $g_o(z)$ and $g_e(z)$, as is illustrated in the example below. This is a consequence of the perfect reconstruction condition. For more details see [14]. Note that the division of Laurent polynomials is not unique, so we may have multiple possibilities.

The scheme for dual lifting is similar.

## Example

The analyzing filter pair for the Cohen-Daubechies-Feauveau biorthogonal wavelets [9] with 2 vanishing moments for both the primal and dual wavelet functions (CDF $(2,2)$) is (up to a normalization factor $\sqrt{2}$)

$$\begin{aligned}\widetilde{h}(z) &= -\frac{1}{8}z^{-2}+\frac{1}{4}z^{-1}+\frac{3}{4}+\frac{1}{4}z-\frac{1}{8}z^{2},\\ \widetilde{g}(z) &= \frac{1}{4}z^{-2}-\frac{1}{2}z^{-1}+\frac{1}{4}.\end{aligned}$$

The corresponding polyphase matrix is

$$P(z)=\begin{bmatrix} -\frac{1}{8}z^{-1}+\frac{3}{4}-\frac{1}{8}z & \frac{1}{4}+\frac{1}{4}z \\ \frac{1}{4}z^{-1}+\frac{1}{4} & -\frac{1}{2} \end{bmatrix}.$$

To decompose this matrix using dual lifting as

$$P(z)=\begin{bmatrix} h_e^{\text{new}}(z) & \frac{1}{4}+\frac{1}{4}z \\ g_e^{\text{new}}(z) & -\frac{1}{2} \end{bmatrix}\begin{bmatrix} 1 & 0 \\ t(z) & 1 \end{bmatrix}$$

we have to find Laurent polynomials $t(z)$, $h_e^{\text{new}}(z)$, and $g_e^{\text{new}}(z)$ such that

$$\begin{aligned}-\frac{1}{8}z^{-1}+\frac{3}{4}-\frac{1}{8}z &= h_e^{\text{new}}(z)+t(z)\left(\frac{1}{4}+\frac{1}{4}z\right),\\ \frac{1}{4}z^{-1}+\frac{1}{4} &= g_e^{\text{new}}(z)+t(z)\left(-\frac{1}{2}\right).\end{aligned}$$

Thus we calculate the quotient $t(z)$ and the remainder $h_e^{\text{new}}(z)$ of a long division. This division is not unique: there are three solutions. We have chosen:

$$\begin{array}{rrr|l} -\frac{1}{8}z^{-1} & +\frac{3}{4} & -\frac{1}{8}z & \frac{1}{4}+\frac{1}{4}z \\ \cline{4-4} -\frac{1}{8}z^{-1} & -\frac{1}{8} & & -\mathbf{\frac{1}{2}z^{-1}}-\mathbf{\frac{1}{2}} \\ \cline{1-2} & +\frac{7}{8} & -\frac{1}{8}z & \\ & -\frac{1}{8} & -\frac{1}{8}z & \\ \cline{2-3} & \mathbf{+1} & & \end{array}$$

This choice will allow us to perform further lifting steps later. A solution to this problem is thus given by

$$\begin{aligned} t(z) &= -\frac{1}{2}z^{-1} - \frac{1}{2}, \\ h_e^{\text{new}}(z) &= 1, \\ g_e^{\text{new}}(z) &= 0, \end{aligned}$$

and thus

$$P(z) = \begin{bmatrix} 1 & \frac{1}{4} + \frac{1}{4}z \\ 0 & -\frac{1}{2} \end{bmatrix} \begin{bmatrix} 1 & 0 \\ -\frac{1}{2}z^{-1} - \frac{1}{2} & 1 \end{bmatrix}.$$

We can continue the decomposition to obtain

$$P(z) = \begin{bmatrix} 1 & 0 \\ 0 & -\frac{1}{2} \end{bmatrix} \begin{bmatrix} 1 & \frac{1}{4} + \frac{1}{4}z \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ -\frac{1}{2}z^{-1} - \frac{1}{2} & 1 \end{bmatrix}.$$

Summarized, the analyzing step can be performed by a dual lifting step, followed by a primal lifting step and a normalization. If we omit the normalization factors $K_1$ and $K_2$, it can be implemented using the following algorithm:

| **Forward transform** | | **Inverse transform** | |
|---|---|---|---|
| Splitting | $s_i \leftarrow x_{2i}$<br>$d_i \leftarrow x_{2i+1}$ | Inverse primal lifting | $s_i \leftarrow s_i - \frac{1}{4}(d_{i-1} + d_i)$ |
| Dual lifting | $d_i \leftarrow d_i - \frac{1}{2}(s_i + s_{i+1})$ | Inverse dual lifting | $d_i \leftarrow d_i + \frac{1}{2}(s_i + s_{i+1})$ |
| Primal lifting | $s_i \leftarrow s_i + \frac{1}{4}(d_{i-1} + d_i)$ | Merging | $x_{2i} \leftarrow s_i$<br>$x_{2i+1} \leftarrow d_i$ |

## Other examples

The decompositions in lifting steps of some other wavelet transforms of this family are shown in Appendix A.

### 3.3.5 Advantages of the Lifting Scheme

Summarized, the lifting scheme has the following immediate advantages, when compared to the classical filter bank algorithm:

- It is faster (but still $O(n)$, of course). In the case of CDF $(2,2)$, the original filters $\widetilde{h}(z)$ and $\widetilde{g}(z)$ count $5+3=8$ filter coefficients, while the lifting decomposition gives us $s(z)$ and $t(z)$ counting only $2+2=4$ filter coefficients.

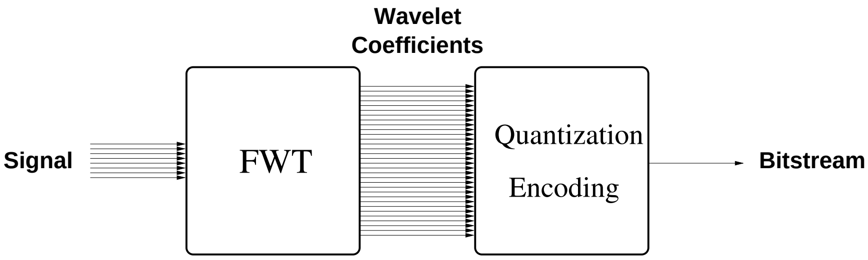


Figure 3.2: The floating point wavelet transform in a compression application.

- The inverse transform is trivial to find. One does not need to explicitly find the corresponding analysis filters.
- The lifting scheme is easier to understand and implement.
- The lifting scheme is more generic and allows for the creation of second-generation wavelets.

Furthermore all wavelet filters can be implemented using the lifting scheme. Thus giving a specific wavelet transform, an implementation using the lifting scheme can always be derived using the Euclidean algorithm. This algorithm forms the bridge between the classical wavelet transforms and the lifting scheme.

## 3.4 The Integer Wavelet Transform

In many applications (e.g. image compression and processing, video, audio, ...) the input data consist of integer samples. In addition the storage and encoding of integer numbers is easier, compared to floating point numbers. Unfortunately all of the above transforms assume that the input samples are floating point values. Even if the input values actually are integer, doing the filtering operations on these numbers will transform them in rational or real numbers because the filter coefficients need not be integers. To obtain an efficient implementation of the discrete wavelet transform, it is of great practical importance that the wavelet transform is represented by a set of integers as well (possibly up to some scaling factor). Of course, this can always be obtained by scaling and rounding the real numbers, but this rounding process will loose information, and after rounding, the original signal can not be reconstructed from its transform without an error.

Especially from a compression point of view, it looks a bit silly to first expand the integer input data from e.g. 8 bits per sample to 32 bit wide floating point numbers, while the goal is to reduce the total amount of data (cfr. Figure 3.2).

### 3.4.1 Integer Transforms

Fortunately the lifting scheme can be modified easily to a transform that maps integers to integers and that is reversible, and thus allows a perfect reconstruction [6, 57]. This is done by adding some rounding operations, at the expense of introducing a non-linearity in the transform.

Since even the most elementary lifting steps involve for example divisions by 2, we obtain in general prediction values $\mathcal{P}(\lambda_j)$ and update values $\mathcal{U}(\gamma_j)$ which are not integer. We shall round these numbers to integers (for example the nearest integer) and indicate this operation by curly braces. Thus, we actually compute rounded values:

$$\begin{aligned} \gamma_j &\leftarrow \gamma_j - \left\{\mathcal{P}(\lambda_j)\right\}, \\ \lambda_j &\leftarrow \lambda_j + \left\{\mathcal{U}(\gamma_j)\right\} \end{aligned}$$

Because an operation of the form

$$a^{\text{new}} \leftarrow a^{\text{old}} + \{b\}$$

does not change $b$, independent of the kind of rounding operation, $a^{\text{old}}$ can always be reconstructed without any errors as

$$a^{\text{old}} \leftarrow a^{\text{new}} - \{b\}.$$

To see that perfect reconstruction is always possible, we note the following. The last computation in a forward transform step is to compute $\lambda_j \leftarrow \lambda_j + \left\{\mathcal{U}(\gamma_j)\right\}$. Thus the first computation in the inverse transform step is to recompute $\left\{\mathcal{U}(\gamma_j)\right\}$, and this will give exactly the same result as before if $\gamma_j$ has been perfectly reconstructed so far. Therefore $\lambda_j \leftarrow \lambda_j - \left\{\mathcal{U}(\gamma_j)\right\}$ reconstructs exactly the original $\lambda_j$. Thus also $\left\{\mathcal{P}(\lambda_j)\right\}$ can be reproduced in the reconstruction step, exactly as in the forward step, and thus also $\gamma_j \leftarrow \gamma_j + \left\{\mathcal{P}(\lambda_j)\right\}$ will give the original $\gamma_j$ back. This shows that each step of the lifting scheme with rounding, is perfectly invertible and thus the whole signal is perfectly reconstructible:

$$\begin{aligned} \lambda_j &\leftarrow \lambda_j - \left\{\mathcal{U}(\gamma_j)\right\}, \\ \gamma_j &\leftarrow \gamma_j + \left\{\mathcal{P}(\lambda_j)\right\} \end{aligned}$$

This is one of the most amazing features of integer lifting: whatever deterministic rounding operation is used, the lifting operation is *always* reversible.

### 3.4.2 Rational Lifting Coefficients

Using the $z$-transform, a lifting step basically looks like

$$x^{\text{new}}(z) \leftarrow x(z) + s(z)y(z).$$

Because the signal part $y(z)$ is not changed by the lifting step, the result of the filter operation can be rounded:

$$x^{\text{new}}(z) \leftarrow x(z) + \{s(z)y(z)\},$$

while retaining a 100% reversible operation:

$$x(z) \leftarrow x(z)^{\text{new}} - \{s(z)y(z)\}.$$

In some cases, all filter coefficients of $s(z)$ are rational and the corresponding lifting step can be written as

$$x_i \leftarrow x_i - \frac{1}{a}\sum_j b_j y_j,$$

where $a, b_j \in \mathbf{Z}$, i.e. they are integers. This is true for various wavelet transforms, among which the Cohen-Daubechies-Feauveau biorthogonal wavelets.

Such a lifting step can be modified in one of the following ways:

**Full rounding** The result of the division by $a$ is rounded:

$$\widetilde{x}_i \leftarrow x_i - \left\{\frac{\sum_j b_j y_j}{a}\right\}.$$

Here $\widetilde{x}_i$ is an integer approximation to the $x_i$ obtained with floating point lifting.

**Without rounding** We avoid the division by $a$ by multiplying the other terms with $a$:

$$\widetilde{x}_i \leftarrow a x_i - \sum_j b_j y_j.$$

Here $\widetilde{x}_i = ax$, and thus the dynamic range of the wavelet coefficients will increase. This has to be taken into account in later steps. Note that in this case no real rounding is performed, and thus this can be considered to be an "exact" implementation of the floating point version, yielding integers.

**Mixed form** We combine the rounding and multiplication steps of both methods:

$$\widetilde{x}_i \leftarrow a_1 x_i - \left\{\frac{\sum_j b_j y_j}{a_2}\right\},$$

with

$$\begin{aligned} a_1, a_2 &\in \mathbf{Z}, \\ a_1 \cdot a_2 &= a. \end{aligned}$$

This variant can be used if we want to have more control over the dynamic range of the resulting $\widetilde{x}_i$.

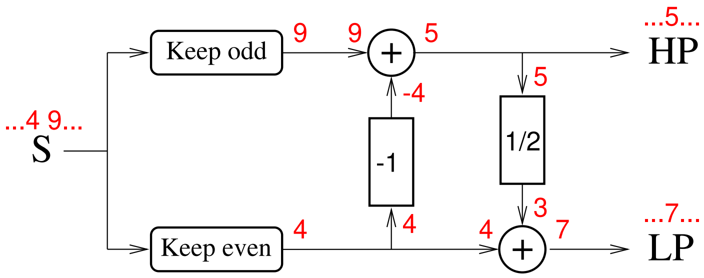


Figure 3.3: Graphical representation of the integer Haar wavelet transform.

Note that in all of the three cases above the modified lifting step is still reversible, and thus the perfect reconstruction feature is still present.

If $a, b_j \notin \mathbf{Z}$, we can still use the "full rounding" method to obtain a transform that maps integers to integers.

In most cases we will use "full rounding", except when we want to control the dynamic range of the result.

## Examples

**CDF** $(1,1)$ This is the well-known Haar transform. The integer version looks like:

| Forward transform | | Inverse transform | |
|---|---|---|---|
| Splitting: | $s_i \leftarrow x_{2i}$<br>$d_i \leftarrow x_{2i+1}$ | Inverse primal lifting: | $s_i \leftarrow s_i - \left\{\frac{d_i}{2}\right\}$ |
| Dual lifting: | $d_i \leftarrow d_i - s_i$ | Inverse dual lifting: | $d_i \leftarrow d_i + s_i$ |
| Primal lifting: | $s_i \leftarrow s_i + \left\{\frac{d_i}{2}\right\}$ | Merging: | $x_{2i} \leftarrow s_i$<br>$x_{2i+1} \leftarrow d_i$ |

A graphical representation of the forward transform is shown in Figure 3.3.

**CDF** $(2,2)$ The algorithm for the integer version is:

| Forward transform | | Inverse transform | |
|---|---|---|---|
| Splitting: | $s_i \leftarrow x_{2i}$<br>$d_i \leftarrow x_{2i+1}$ | Inverse primal lifting: | $s_i \leftarrow s_i - \left\{\frac{1}{4}(d_{i-1} + d_i)\right\}$ |
| Dual lifting: | $d_i \leftarrow d_i - \left\{\frac{1}{2}(s_i + s_{i+1})\right\}$ | Inverse dual lifting: | $d_i \leftarrow d_i + \left\{\frac{1}{2}(s_i + s_{i+1})\right\}$ |
| Primal lifting: | $s_i \leftarrow s_i + \left\{\frac{1}{4}(d_{i-1} + d_i)\right\}$ | Merging: | $x_{2i} \leftarrow s_i$<br>$x_{2i+1} \leftarrow d_i$ |

### 3.4.3 Normalization

In the above examples, we again omitted the normalization factors $K_1$ and $K_2$ (see Equation 3.1). However, in many applications (e.g. compression) it is important to know the magnitudes of the normalized wavelet coefficients. They indicate the importance of the corresponding wavelet components.

Furthermore, if we want the energy of the signal to be retained in the wavelet coefficients — or at least an approximation in the non-orthogonal case — we have to use normalized filter coefficients that are $\sqrt{2}$ times as large as the filter coefficients we used. Thus the real normalization factors are $\sqrt{2}K_1$ for the low-pass band, and $\sqrt{2}K_2$ for the high-pass band. Note that the normalization factors for the low-pass band accumulate, due to the iteration on the low-pass band.

In other cases, we may want that the mean value of the low-pass band coefficients is the same as the mean value of the original signal. This will be true if we make sure that $K_1 = 1$. E.g. if a digital image is transformed, the low-pass band will be a low-resolution version of the original image. Note that the dynamic range of the pixel values may increase due to the non-convexity of the low-pass filter.

## 3.5 Boundary Treatment

Real-world signals do not extend infinitely in time or space, but are limited to a finite interval (Figure 3.4). However, filter bank algorithms assume infinite signal lengths. Padding the signal with zeros is not acceptable, since it would lead to much more wavelet coefficients than original samples. If one truncates the transformed function to yield the same number of wavelet coefficients as samples, this will badly distort the signal and thus render perfect reconstruction impossible. What we need is some kind of signal extension, or take into account the finiteness of the signal by slightly changing the transform near the boundaries of the signal.

### 3.5.1 Classical Extension Methods

In classical signal processing applications, it is common practice to extend the data for the computations near the border, by replicating the signal using one of the following methods:

#### Periodic Extension

A first solution is the periodic extension of the signal: we extend the finite signal by putting copies of itself in front of and behind the original signal (Figure 3.5).

After a wavelet transform we can simply discard the coefficients that lie outside of the interval in which the original signal was defined, since these wavelet

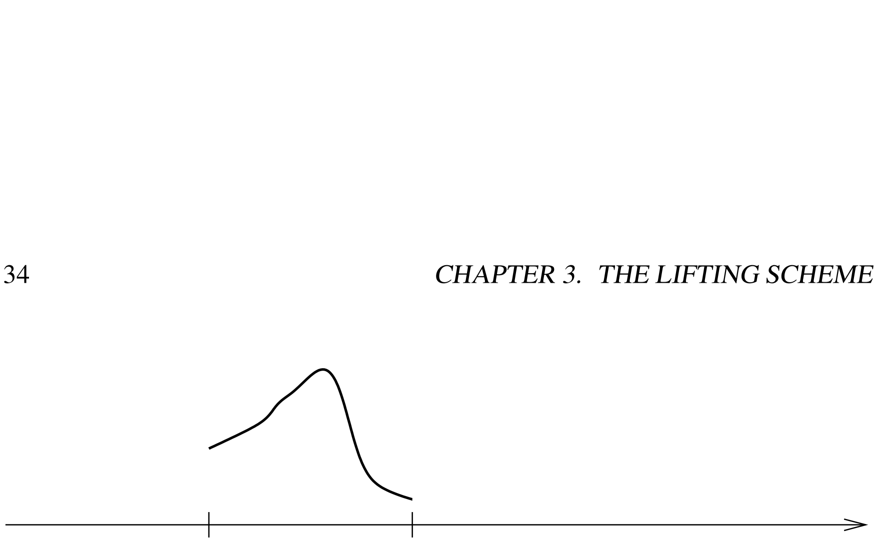

Figure 3.4: A finite signal.

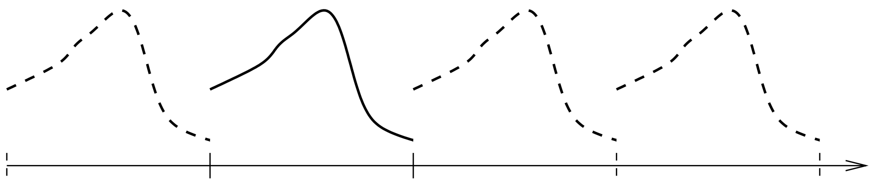

Figure 3.5: Periodic extension of a finite signal.

coefficients are the same as the retained coefficients, and thus can be recovered without any problems.

However, unless the first and the last sample have the same value, we introduce unwanted discontinuities at the boundaries of the original signal. These discontinuities will locally enlarge the wavelet coefficients and make compression of the signal more difficult.

### Symmetric Extension

A better solution is the symmetric extension of the signal: we extend the finite signal by mirroring it around its endpoints, which makes the discontinuities disappear (Figure 3.6) [4]. But the higher order derivatives may still be discontinuous.

Symmetric extension can be considered as periodic extension applied to a concatenation of the original signal and a mirrored copy of the original signal. Hence after filtering one has to retain twice as much coefficients. Fortunately one can discard half of them if the filters are symmetric, yielding the same number of coefficients as the original signal length. That is why we use biorthogonal wavelets instead of orthogonal wavelets, since they can be symmetric.

While the periodic extension of a discrete (sampled) signal is straightforward and unique, there are four possible ways to extend a discrete signal symmetrically. One can duplicate the first and the last sample value or not. These cases are denoted by the notation $(a,b)$. If $a = 1$, the first sample value is not duplicated, while $a = 2$ indicates that the first sample value is duplicated. The value of $b$ indicates the possible duplication of the last sample value (Figure 3.7).

However, to ensure to get a perfect reconstruction, one has to apply the correct

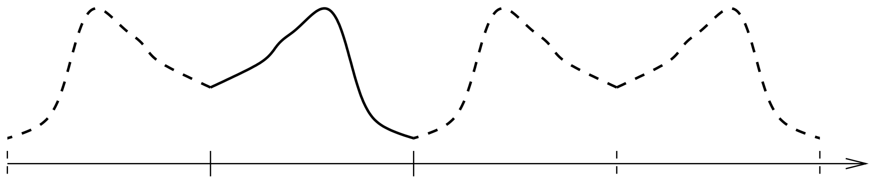

Figure 3.6: Symmetric extension of a finite signal.

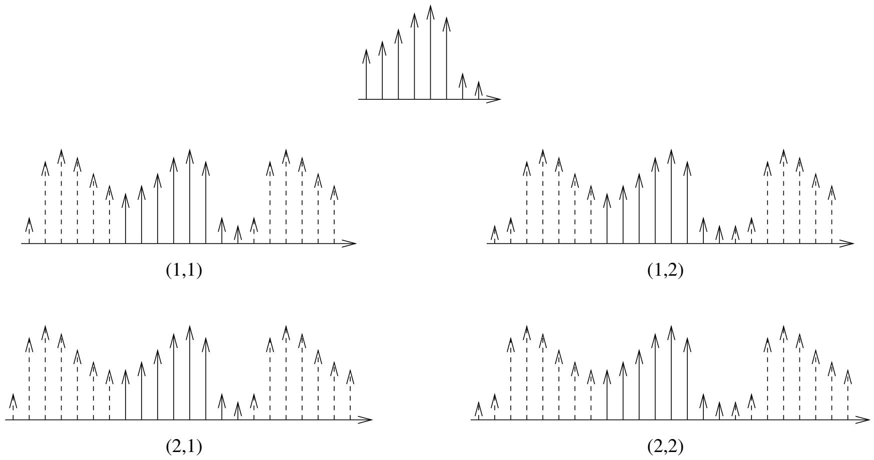


Figure 3.7: The four cases of symmetric signal extension of a discrete finite signal.

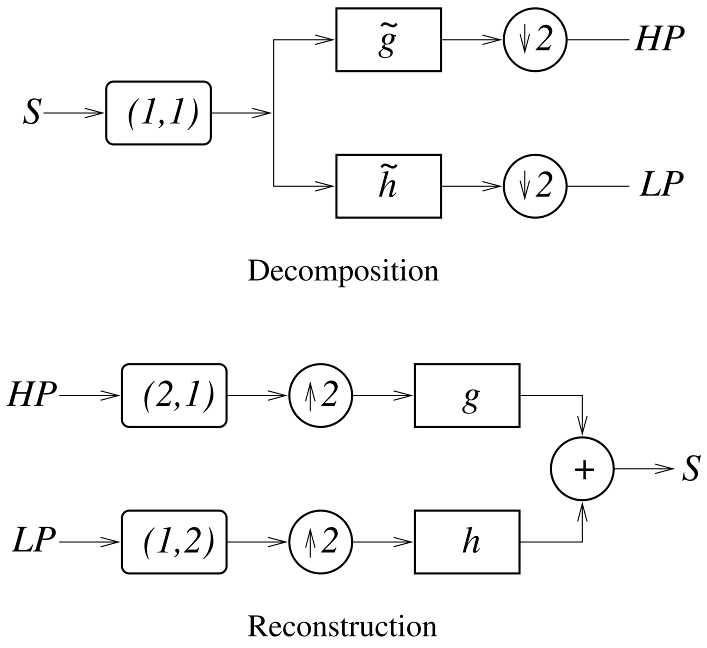


Figure 3.8: The filter bank scheme for the symmetric wavelet transform: filter operations are indicated by boxes, extension operators by rounded boxes and up/downsampling or addition operations by circles.

extension in every step of the decomposition and reconstruction, as indicated in Figure 3.8 [4].

### 3.5.2 Signal Extension with Lifting

Although the lifting scheme allows to transform signals with a finite length without extending the signal, we did not choose to take this approach. Instead we use the classical symmetric extension because it is easier to implement and suffices for the applications we have in mind. In each lifting step, we extend the signal symmetrically. Depending on the first and last sample being "even" or "odd', we can consider four cases, as presented graphically in Figure 3.9:

**FELO** First even, last odd,

**FELE** First even, last even,

**FOLO** First odd, last odd,

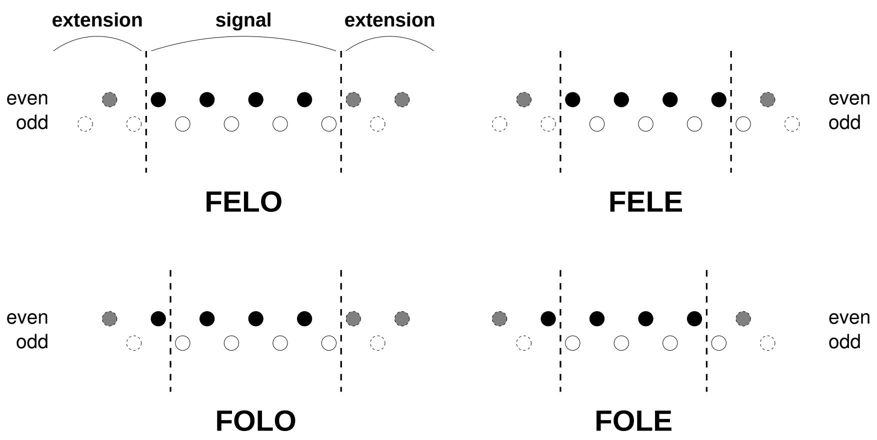


Figure 3.9: The four cases of symmetric signal extension using lifting.

**FOLE** First odd, last even.

This is another nice property of the second-generation viewpoint of the lifting scheme: symmetric extension is always possible! We are no longer limited to symmetric (biorthogonal) wavelets. Even if one has a lifting decomposition for a non-symmetric wavelet filter bank (e.g. orthogonal Daubechies wavelets), the implementation using lifting steps can use symmetric extension.

## 3.6 Conclusions

The lifting scheme allows to implement reversible integer wavelet transforms in an efficient way. All classical wavelet transforms can be implemented using the lifting scheme by a factorization in lifting steps. We worked out the details of integer wavelet transforms for a class of biorthogonal wavelets (Cohen-Daubechies-Feauveau).