Constructing and implementing orthogonal and biorthogonal wavelet transforms via liftings

WTBV 2017/18 (short version)

January 30, 2018

- *Lifting* is a particular method for constructing and implementing orthogonal and biorthogonal (pairs of) filters and discrete wavelet transforms
- *Lifting* has close relations to classical topis in computer algebra: polynomial arithmetic and Euclid's algorithm in particular

- Haar WT uses the filters

$$\text{low-pass} \qquad \boldsymbol{h} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \end{bmatrix}$$

$$\text{high-pass} \qquad \boldsymbol{g} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & -1 \end{bmatrix}$$

- orthogonal matrix

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}.$$

- operation on signals

$$\boldsymbol{a} = (a_n)_{n\in\mathbb{Z}} \longmapsto \begin{cases} \tilde{\boldsymbol{a}} = (\tilde{a}_n)_{n\in\mathbb{Z}} \\ \tilde{\boldsymbol{d}} = (\tilde{d}_n)_{n\in\mathbb{Z}} \end{cases}$$

where

$$\tilde{a}_n = \frac{1}{\sqrt{2}} \left(a_{2n} + a_{2n+1}\right), \qquad \tilde{d}_n = \frac{1}{\sqrt{2}} \left(a_{2n} - a_{2n+1}\right).$$

- Using the technique of z-transforms (alias power series), one gets by separating the sequences of even- and odd-indexed coefficients

$$(\ldots, a_{-2}, a_{-1}, a_0, a_1, a_2, a_3, \ldots) \longleftrightarrow \begin{cases} (\ldots, a_{-2}, a_0, a_2, \ldots) \\ (\ldots, a_{-1}, a_1, a_3, \ldots) \end{cases}$$

the decomposition

$$a(z) = \sum_{n \in \mathbb{Z}} a_n \, z^n = a_{even}(z^2) + z \cdot a_{odd}(z^2),$$

where

$$a_{even}(z) = \sum_{n \in \mathbb{Z}} a_{2n} \, z^n = \left. \frac{a(z) + a(-z)}{2} \right|_{z^2 \leftarrow z},$$

$$a_{odd}(z) = \sum_{n \in \mathbb{Z}} a_{2n+1} \, z^n = \left. \frac{a(z) - a(-z)}{2z} \right|_{z^2 \leftarrow z}.$$

- The series for approximation and for detail are

$$\tilde{a}(z) = \sum_{n\in\mathbb{Z}} \tilde{a}_n\, z^n = \frac{1}{\sqrt{2}}\left(a_{even}(z) + a_{odd}(z)\right),$$

$$\tilde{d}(z) = \sum_{n\in\mathbb{Z}} \tilde{d}_n\, z^n = \frac{1}{\sqrt{2}}\left(a_{even}(z) - a_{odd}(z)\right).$$

- Writing this in matrix form (with power series as coefficients)

$$\begin{bmatrix} \tilde{a}(z) \\ \tilde{d}(z) \end{bmatrix} = H \cdot \begin{bmatrix} a_{even}(z) \\ a_{odd}(z) \end{bmatrix}$$

- This simple cases is not typical, however, because the matrix entries are constants. In general they will be polynomials (including terms with negative z-powers)

- Now consider the following decomposition of the H-matrix:

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} = \begin{bmatrix} \sqrt{2} & 0 \\ 0 & -\frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} 1 & \frac{1}{2} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix}$$

- Only very simple (2×2)-matrices occur as factors:
  - the diagonal matrix $\begin{bmatrix} \sqrt{2} & 0 \\ 0 & -\frac{1}{\sqrt{2}} \end{bmatrix}$
  - the special upper triangular matrix $\begin{bmatrix} 1 & \frac{1}{2} \\ 0 & 1 \end{bmatrix}$
  - the special lower triangular matrix $\begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix}$
- “special” means: entries in the main diagonal are 1

- The decomposition can be interpreted as a *straight-line program* for parallel execution of the approximation-detail transformation

$$(a_{2n}, a_{2n+1}) \mapsto (\tilde{a}_n, \tilde{d}_n) \qquad (n \in \mathbb{Z})$$

$$x \leftarrow a_{2n}$$

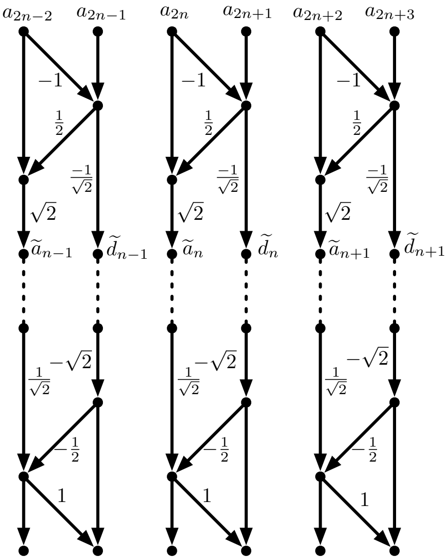
$$y \leftarrow a_{2n+1}$$

$$y \leftarrow y - x \qquad \text{multiplication by } \begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix}$$

$$x \leftarrow x + \frac{1}{2} y \qquad \text{multiplication by } \begin{bmatrix} 1 & \frac{1}{2} \\ 0 & 1 \end{bmatrix}$$

$$\begin{array}{rcl} \tilde{a}_n & \leftarrow & \sqrt{2}\, x \\ \tilde{d}_n & \leftarrow & -\frac{1}{\sqrt{2}}\, y \end{array} \qquad \text{multiplication by } \begin{bmatrix} \sqrt{2} & 0 \\ 0 & -\frac{1}{\sqrt{2}} \end{bmatrix}$$

Graphical representation as a data flow diagram (including the inverse transformation)



- The decomposition of H immediately gives the decomposition of the matrix of the *inverse transformation* H^{-1}

$$H^{-1} = \begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix}^{-1} \begin{bmatrix} 1 & \frac{1}{2} \\ 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} \sqrt{2} & 0 \\ 0 & -\frac{1}{\sqrt{2}} \end{bmatrix}^{-1}$$

$$= \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} 1 & -\frac{1}{2} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{2}} & 0 \\ 0 & -\sqrt{2} \end{bmatrix}$$

- Note: the diagonal and triangular matrices are easy to invert!
- The Haar situation is particular because $H = H^{-1}$, but the previous remark applies in general
- The decompositions of H and H^{-1} can be read as blueprints for implementation!

- Filter coefficients of the D4 transform

$$
\begin{aligned}
(h_0, h_1, h_2, h_3) &= \left(\frac{1+\sqrt{3}}{4\sqrt{2}}, \frac{3+\sqrt{3}}{4\sqrt{2}}, \frac{3-\sqrt{3}}{4\sqrt{2}}, \frac{1-\sqrt{3}}{4\sqrt{2}}\right) \\
&= (0.482963, 0.836516, 0.224144, -0.129410) \\
(g_{-2}, g_{-1}, g_0, g_1) &= \left(\frac{-1+\sqrt{3}}{4\sqrt{2}}, \frac{3-\sqrt{3}}{4\sqrt{2}}, \frac{-3-\sqrt{3}}{4\sqrt{2}}, \frac{1+\sqrt{3}}{4\sqrt{2}}\right) \\
&= (0.129410, 0.224144, -0.836516, 0.482963)
\end{aligned}
$$

- polyphase matrix

$$
\begin{bmatrix} h_{even}(z) & h_{odd}(z) \\ g_{even}(z) & g_{odd}(z) \end{bmatrix} = \begin{bmatrix} h_0 + h_2 z & h_1 + h_3 z \\ g_{-2} z^{-1} + g_0 & g_{-1} z^{-1} + g_1 \end{bmatrix}
$$

- Decomposing the matrix of the D4 transform

$$\begin{bmatrix} h_{even}(z) & h_{odd}(z) \\ g_{even}(z) & g_{odd}(z) \end{bmatrix} =$$

$$\begin{bmatrix} \frac{\sqrt{3}-1}{\sqrt{2}} & 0 \\ 0 & \frac{\sqrt{3}+1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} 1 & -z \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ -\frac{\sqrt{3}}{4} - \frac{\sqrt{3}-2}{4} z^{-1} & 1 \end{bmatrix} \begin{bmatrix} 1 & \sqrt{3} \\ 0 & 1 \end{bmatrix}$$

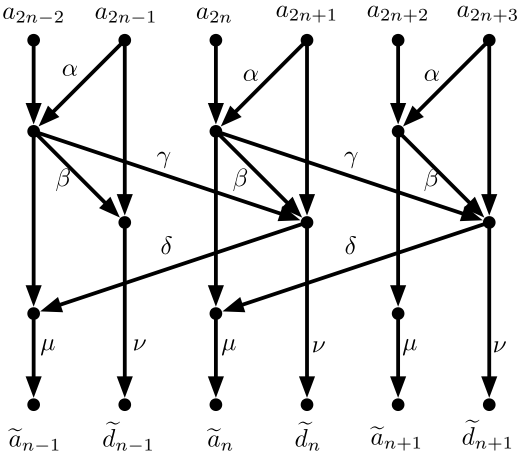
- define parameters

$$(\alpha, \beta, \gamma, \delta, \mu, \nu) = \left(\sqrt{3}, -\frac{\sqrt{3}}{4}, -\frac{\sqrt{3}-2}{4}, -1, \frac{\sqrt{3}-1}{\sqrt{2}}, \frac{\sqrt{3}+1}{\sqrt{2}} \right)$$

- $D4$ transform in terms of these parameters

$$\begin{aligned} \widetilde{a}_n &= \mu \left[(1 + \gamma\delta) a_{2n} + (\alpha + \alpha\gamma\delta) a_{2n+1} \right. \\ &\qquad \left. + \delta\beta\, a_{2n+2} + (\alpha\beta\delta + \delta) a_{2n+3} \right] \\ \widetilde{d}_n &= \nu \left[(1 + \alpha\beta) a_{2n+1} + \beta\, a_{2n} + \alpha\gamma\, a_{2n-1} + \gamma\, a_{2n-2} \right] \end{aligned}$$

- *D*4 analysis as data flow diagram



- D4 synthesis by inverting the decomposition

- General considerations
  - $\downarrow$ = downsampling (factor 2)

$$\downarrow a(z) = a_{even}(z) = \left.\frac{a(z) + a(-z)}{2}\right|_{z^2 \leftarrow z}$$

$$a_{odd}(z) = \downarrow (z^{-1} \cdot a(z)) = \left.\frac{a(z) - a(-z)}{2z}\right|_{z^2 \leftarrow z}$$

  - downsampling a convolution $\boldsymbol{a} \star \boldsymbol{b}$ of two signals (or filters)

$$\begin{aligned}\downarrow \big(a(z) \cdot b(z)\big) &= \left.\frac{1}{2}\big(a(z)b(z) + a(-z)b(-z)\big)\right|_{z^2 \leftarrow z} \\ &= \frac{1}{2}\Big((a_{even}(z^2) + z\, a_{odd}(z^2))(b_{even}(z^2) + z\, b_{odd}(z^2)) \\ &\qquad \left. + \,(a_{even}(z^2) - z\, a_{odd}(z^2))(b_{even}(z^2) - z\, b_{odd}(z^2))\Big)\right|_{z^2 \leftarrow z} \\ &= a_{even}(z) \cdot b_{even}(z) + z \cdot a_{odd}(z) \cdot b_{odd}(z),\end{aligned}$$

- filter bank with filters $\boldsymbol{h}, \boldsymbol{g}$ followed by downsampling (reversed filters denoted by $\overleftarrow{h}$ and $\overleftarrow{g}$)

$$\boldsymbol{a} = (a_n)_{n\in\mathbb{Z}} \longmapsto \begin{cases} \tilde{\boldsymbol{a}} = (\tilde{a}_n)_{n\in\mathbb{Z}} = a_{even}(z)\cdot \overleftarrow{h}_{even}(z) + z\cdot a_{odd}(z)\cdot \overleftarrow{h}_{odd}(z) \\ \tilde{\boldsymbol{d}} = (\tilde{d}_n)_{n\in\mathbb{Z}} = a_{even}(z)\cdot \overleftarrow{g}_{even}(z) + z\cdot a_{odd}(z)\cdot \overleftarrow{g}_{odd}(z) \end{cases}$$

- matrix version of this transform

$$\begin{bmatrix} a_{even}(z) \\ a_{odd}(z) \end{bmatrix} \longmapsto \begin{bmatrix} \tilde{a}(z) \\ \tilde{d}(z) \end{bmatrix} = \begin{bmatrix} \overleftarrow{h}_{even}(z) & z\cdot \overleftarrow{h}_{odd}(z) \\ \overleftarrow{g}_{even}(z) & z\cdot \overleftarrow{g}_{odd}(z) \end{bmatrix} \begin{bmatrix} a_{even}(z) \\ a_{odd}(z) \end{bmatrix}$$

- Note that

$$\overleftarrow{h}_{even}(z) = h_{even}(z^{-1}), \quad z \cdot \overleftarrow{h}_{odd}(z) = h_{odd}(z^{-1})$$

- so that the matrix of this transform can be written as

$$P(z^{-1})^{\mathtt{t}} = \begin{bmatrix} h_{even}(z^{-1}) & h_{odd}(z^{-1}) \\ g_{even}(z^{-1}) & g_{odd}(z^{-1}) \end{bmatrix}$$

- The matrix

$$P(z) = \begin{bmatrix} h_{even}(z) & g_{even}(z) \\ h_{odd}(z) & g_{odd}(z) \end{bmatrix}$$

  is known as the *polyphase matrix* belonging to the pair $(\boldsymbol{h}, \boldsymbol{g})$ of filters

- Orthogonality of a pair $(\boldsymbol{h}, \boldsymbol{g})$ of filters can be characterized in terms of the polyphase matrix as follows:
  - A pair $(\boldsymbol{h}, \boldsymbol{g})$ of filters is orthogonal if an only if

$$P(z^2)^{-1} = P(z^{-2})^{\mathtt{t}}$$

  - Proof: see the Lecture Notes

- Bi-Orthogonalitity of a pair of filters $(\boldsymbol{h}, \widetilde{\boldsymbol{h}})$ can also be cast in terms of the polyphase matrix:
  - A pair of filters $(\boldsymbol{h}, \widetilde{\boldsymbol{h}})$ (together with filters $\boldsymbol{g}$ und $\widetilde{\boldsymbol{g}}$ constructed as usual) is a biorthogonal pair if and only if

$$\widetilde{P}(z^2)^{-1} = P(z^{-2})^{\mathtt{t}}$$

  - Proof: see the Lecture Notes

- What does "lifting" mean?
  - The goal is to write the polyphase matrix $P(z)$ of a filter pair $(\boldsymbol{h}, \boldsymbol{g})$ as a product of very simple matrices

$$P(z) = \begin{bmatrix} h_{even}(z) & g_{even}(z) \\ h_{odd}(z) & g_{odd}(z) \end{bmatrix} =$$
$$\begin{bmatrix} 1 & q_1(z) \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ q_2(z) & 1 \end{bmatrix} \begin{bmatrix} 1 & q_3(z) \\ 0 & 1 \end{bmatrix} \cdots \begin{bmatrix} 1 & 0 \\ q_k(z) & 1 \end{bmatrix} \begin{bmatrix} K & 0 \\ 0 & 1/K \end{bmatrix}$$

  - w.l.o.g. assume that k is even
  - The $q_i(z)$ are power series $\sum_{j \in \mathbb{Z}} q_j\, z^j$ with positive and possibly also with negative z-exponents
  - If $\boldsymbol{h}$ and $\boldsymbol{g}$ are finite filters (the only case of interest for us) then the $q_i(z)$ are polynomials in z in an extended sense:
    $\sum_{j=s}^{t} q_j z^j$ with positive or negative bounds of summation $s \leq t$
    Generalized polynomials of this type are often called
    Laurent-polynomials (LP)

- Graphical representation of the analysis part of a filter bank using the product decomposition with coefficient polynomials $q_j(z)$ of the polyphase matrix $P(z)$

$$P(z) = \begin{bmatrix} h_{even}(z) & g_{even}(z) \\ h_{odd}(z) & g_{odd}(z) \end{bmatrix} =$$

$$\begin{bmatrix} 1 & q_1(z) \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ q_2(z) & 1 \end{bmatrix} \begin{bmatrix} 1 & q_3(z) \\ 0 & 1 \end{bmatrix} \cdots \begin{bmatrix} 1 & 0 \\ q_k(z) & 1 \end{bmatrix} \begin{bmatrix} K & 0 \\ 0 & 1/K \end{bmatrix}$$

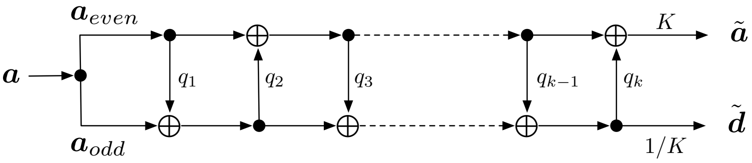


Figure: Lifting analysis

- Graphical representation of the synthesis part of a filter bank using the polynomials $q_j(z)$ and the decompostion of the inverse of the polyphase matrix $P(z)$

$$P(z)^{-1} = \begin{bmatrix} 1/K & 0 \\ 0 & K \end{bmatrix} \begin{bmatrix} 1 & 0 \\ -q_k(z) & 1 \end{bmatrix} \cdots \begin{bmatrix} 1 & -q_3(z) \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ -q_2(z) & 1 \end{bmatrix} \begin{bmatrix} 1 & -q_1(z) \\ 0 & 1 \end{bmatrix}$$

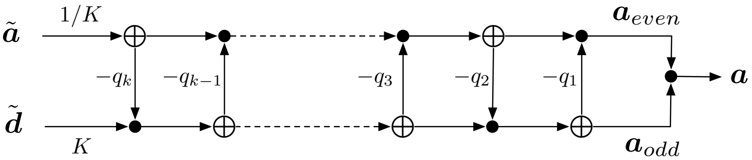


Figure: Lifting synthesis

- How to obtain the lifting decomposition of a polyphase matrix?

$$P(z) = \begin{bmatrix} h_{even}(z) & g_{even}(z) \\ h_{odd}(z) & g_{odd}(z) \end{bmatrix} =$$

$$\begin{bmatrix} 1 & q_1(z) \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ q_2(z) & 1 \end{bmatrix} \begin{bmatrix} 1 & q_3(z) \\ 0 & 1 \end{bmatrix} \cdots \begin{bmatrix} 1 & 0 \\ q_k(z) & 1 \end{bmatrix} \begin{bmatrix} K & 0 \\ 0 & 1/K \end{bmatrix}$$

- Euclids's algorithm can be used for that!

- Consider polynomials (in the usual sense) with real or complex coefficients
- Fundamental property (division with remainder):
  - For any polynomials $a(X), b(X)$ with $b(X) \neq 0$ there exist (unique!) polynomials $q(X)$ (the *quotient*) and $r(X)$ (the *remainder*) s.th.

    $$a(X) = b(X) \cdot q(X) + r(X),$$

    where $r(X) = 0$ or $\deg r(X) < \deg b(X)$
  - In perfect analogy to the situation with integers:
    iterated division with remainder can be used to compute the greatest common divisor (gcd) of two polynmials

- Scheme of Euclid's algorithm

$$
\begin{aligned}
\texttt{input} \quad r_0 &= r_0(X) = a(X) \\
r_1 &= r_1(X) = b(X) \\
r_0 &= r_1 \cdot q_1 + r_2 \\
r_1 &= r_2 \cdot q_2 + r_3 \\
r_2 &= r_3 \cdot q_3 + r_4 \\
&\vdots \\
r_{k-2} &= r_{k-1} \cdot q_{k-1} + r_k \quad (r_k \neq 0) \\
r_{k-1} &= r_k \cdot q_k + 0 \\
\texttt{output} \quad r_k(X) &= \mathrm{ggT}(a(X), b(X))
\end{aligned}
$$

- Scheme of Euclid's algorithm in matrix form

$$\begin{bmatrix} r_1 \\ r_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & -q_1 \end{bmatrix} \begin{bmatrix} r_0 \\ r_1 \end{bmatrix} \qquad \begin{bmatrix} r_0 \\ r_1 \end{bmatrix} = \begin{bmatrix} q_1 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} r_1 \\ r_2 \end{bmatrix}$$

$$\begin{bmatrix} r_2 \\ r_3 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & -q_2 \end{bmatrix} \begin{bmatrix} r_1 \\ r_2 \end{bmatrix} \qquad \begin{bmatrix} r_1 \\ r_2 \end{bmatrix} = \begin{bmatrix} q_2 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} r_2 \\ r_3 \end{bmatrix}$$

$$\vdots \qquad \qquad \vdots$$

$$\begin{bmatrix} r_{k-1} \\ r_k \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & -q_{k-1} \end{bmatrix} \begin{bmatrix} r_{k-2} \\ r_{k-1} \end{bmatrix} \qquad \begin{bmatrix} r_{k-2} \\ r_{k-1} \end{bmatrix} = \begin{bmatrix} q_{k-1} & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} r_{k-1} \\ r_k \end{bmatrix}$$

$$\begin{bmatrix} r_k \\ 0 \end{bmatrix} = \begin{bmatrix} q_k & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} r_{k-1} \\ r_k \end{bmatrix} \qquad \begin{bmatrix} r_{k-1} \\ r_k \end{bmatrix} = \begin{bmatrix} q_k & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} r_k \\ 0 \end{bmatrix}$$

- Putting things together one gets

$$\begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} r_0 \\ r_1 \end{bmatrix} = \begin{bmatrix} q_1 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} q_2 & 1 \\ 1 & 0 \end{bmatrix} \cdots \begin{bmatrix} q_{k-1} & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} q_k & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} r_k \\ 0 \end{bmatrix}$$

- An easy modification is necessary to rewrite this decomposition in the way which is needed for the decomposition of the polyphase matrix

$$\begin{bmatrix} q & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ q & 1 \end{bmatrix} = \begin{bmatrix} 1 & q \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

- The lifting-version of Euclid's algorithm is

$$\begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} 1 & q_1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ q_2 & 1 \end{bmatrix} \begin{bmatrix} 1 & q_3 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ q_4 & 1 \end{bmatrix} \cdots \begin{bmatrix} r_k \\ 0 \end{bmatrix}$$

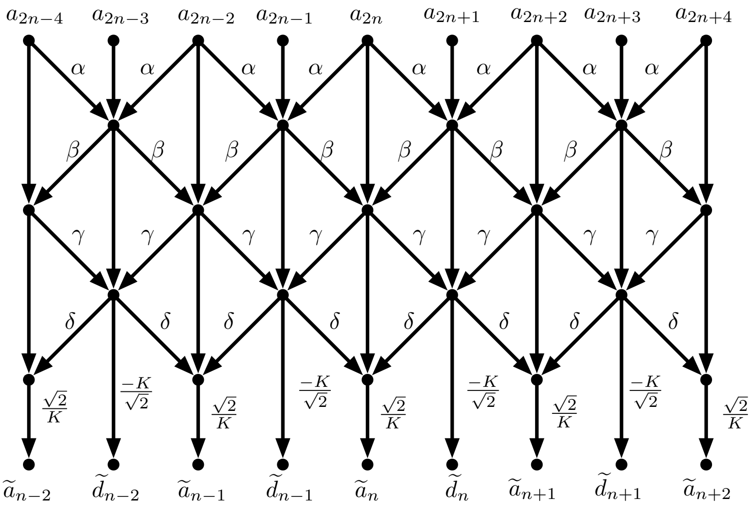
- Details how to employ Euclids's algorithm for the decomposition of the polyphase matrix will not be given here.
- The principle is important:
  - For a pair $(\boldsymbol{h}, \boldsymbol{g})$ of orthogonal filters Euclid's algorithm applied the LPs $h_{even}(z), h_{odd}(z)$ produces a lifting decomposition of the polyphase matrix

$$
\begin{aligned}
P(z) &= \begin{bmatrix} h_{even}(z) & g_{even}(z) \\ h_{odd}(z) & g_{odd}(z) \end{bmatrix} \\
&= \begin{bmatrix} 1 & q_1(z) \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ q_2(z) & \end{bmatrix} \cdots \begin{bmatrix} 1 & q_{k-1}(z) \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ q_k(z) & 1 \end{bmatrix} \begin{bmatrix} K & 0 \\ 0 & 1/K \end{bmatrix}
\end{aligned}
$$

- Polynomials $g_{even}(z)$ and $g_{odd}(z)$ are not used for computing this decomposition. The sequence of quotients $q_1(z)$, $q_2(z)$, ... depends on $h_{even}(z)$ and $h_{odd}(z)$ only
- From the lifting decomposition and its quotient sequence one obtains automatically $g_{even}(z)$ and $g_{odd}(z)$

- The JPEG2000 standard for encoding, storing, compression and transmission of images employs two wavelet transforms:
  - The biorthogonal symmetric COHEN-DAUBECHIES-FEAUVEAU-(7,9) filter pair for *lossy* compression
  - the biorthogonal symmetric LEGALL-(5,3) filter pair for *lossless* compression

- Lifting scheme for the CDF-(9,7) wavelet



- Parameters for the biorthogonal CDF-(9,7) filter pair

$$\begin{aligned} \alpha &= -1.586134342 \\ \beta &= -0.052980118 \\ \gamma &= 0.882911075 \\ \delta &= 0.443506852 \\ K &= 1.230174105 \end{aligned}$$

$$\begin{aligned} h_0 &= \frac{\sqrt{2}}{K}\left[\, 6\alpha\beta\gamma\delta + 2\alpha\beta + 2\gamma\delta + 2\alpha\delta + 1 \,\right] \\ h_1 &= \frac{\sqrt{2}}{K}\left[\, 3\beta\gamma\delta + \beta + \delta \,\right] \\ h_2 &= \frac{\sqrt{2}}{K}\left[\, 4\alpha\beta\gamma\delta + \gamma\delta + \alpha\delta + \alpha\beta \,\right] \\ h_3 &= \frac{\sqrt{2}}{K}\left[\, \beta\gamma\delta \,\right] \\ h_4 &= \frac{\sqrt{2}}{K}\left[\, \alpha\beta\gamma\delta \,\right] \\ g_1 &= \frac{-K}{\sqrt{2}}\left[\, 1 + 2\beta\gamma \,\right] \\ g_2 &= \frac{-K}{\sqrt{2}}\left[\, 3\alpha\beta\gamma + \alpha + \gamma \,\right] \\ g_3 &= \frac{-K}{\sqrt{2}}\left[\, \beta\gamma \,\right] \\ g_4 &= \frac{-K}{\sqrt{2}}\left[\, \alpha\beta\gamma \,\right] \end{aligned}$$

- The biorthogonal LeGall-(5,3) filter pair
  - The filters are

$$\boldsymbol{h} = \frac{1}{8}\begin{bmatrix}-1 & 2 & 6 & 2 & -1\end{bmatrix}_{-2..2}$$
$$\tilde{\boldsymbol{h}} = \frac{1}{2}\begin{bmatrix}1 & 2 & 1\end{bmatrix}_{-1..1}$$
$$\boldsymbol{g} = \frac{1}{2}\begin{bmatrix}-1 & 2 & -1\end{bmatrix}_{0..2}$$
$$\tilde{\boldsymbol{g}} = \frac{1}{8}\begin{bmatrix}-1 & -2 & 6 & -2 & -1\end{bmatrix}_{-2..2}$$

  - Obviously $\tilde{\boldsymbol{h}}$ is a spline filter.
    The lifting scheme uses parameters $\alpha = -\frac{1}{8}$ and $\beta = \frac{1}{4}$

- Lifting scheme for the LeGall transformation

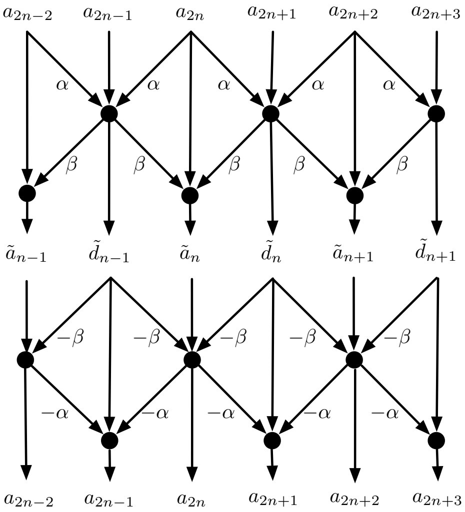


Figure: Lifting scheme for the LeGall transformation

- LeGall transformation

analysis	synthesis
$\tilde{d}_n = a_{2n+1} + \alpha\left(a_{2n} + a_{2n+2}\right)$	$a_{2n} = \tilde{a}_n - \beta\left(\tilde{d}_{n-1} + \tilde{d}_n\right)$
$\tilde{a}_n = a_{2n} + \beta\left(\tilde{d}_{n-1} + \tilde{d}_n\right)$	$a_{2n+1} = \tilde{d}_n - \alpha\left(a_{2n} + a_{2n+2}\right)$

- An interesting aspect of this filter pair is that it can be used to define and invertible *integer-to-integer* transformation – that is why it can be used for lossless compression!
- For analysis do

$$d_n^* = a_{2n+1} - \left\lfloor \frac{1}{2}\left(a_{2n} + a_{2n+2}\right)\right\rfloor, \qquad a_n^* = a_{2n} + \left\lfloor \frac{1}{4}\left(d_{n-1}^* + d_n^*\right)\right\rfloor$$

- and check that the synthesis transformation is given by

$$a_{2n} = a_n^* - \left\lfloor \frac{1}{4}\left(d_{n-1}^* + d_n^*\right) + \frac{1}{2}\right\rfloor, \qquad a_{2n+1} = d_n^* + \left\lfloor \frac{1}{2}\left(a_{2n} + a_{2n+2}\right)\right\rfloor$$