Advanced Transform Methods

Wavelet Transform from Filter Banks

Andy Watson

Wavelet transform from Filter Banks

From Multiresolution Analysis (MRA),

if signal $s(t)$ is in V_m for fininte m, then $s(t)$ determined by

$$s(t) = \sum_{n=-\infty}^{\infty} c_{m,n}\phi_{m,n}(t) \quad \left[\sum_k c_k^j \phi(2^j t - k) \quad \text{in MRA lecture} \right]$$

Since $V_m = V_{m-1} \oplus W_{m-1}$ this can be written

$$s(t) = \sum_n c_{m_0,n}\phi_{m_0,n}(t) + \sum_{k=m_0}^{m-1} \sum_n d_{k,n}\psi_{k,n}(t) \qquad m > m_0$$

where coefficients $d_{m,n}$ and $c_{m,n}$ are

inner products between $\psi_{m,n}(t)$ and $\phi_{m,n}(t)$ respectively.

Approximating the signal

Using Parseval, we have

$$c_{m,n} = 2^{m/2}\int_{-\infty}^{\infty} s(t)\phi^*(2^m t - n)dt$$

$$= \frac{1}{2\pi}2^{-m/2}\int_{-\infty}^{\infty} S(\omega)\Phi^*(2^{-m}\omega)e^{-j2^{-m}\omega n}d\omega$$

For large scale m and $\Phi(0) = 1$ (i.e. $\phi(t)$ normalised), have

$$c_{m,n} \approx \frac{1}{2\pi}2^{-m/2}\int_{-\infty}^{\infty} S(\omega)e^{-j2^{-m}\omega n}d\omega = 2^{-m/2}s(2^{-m}n)$$

(since $\Phi(\omega/2^m) \approx \Phi(0) = 1$ over the range where $S(\omega)$ exists).

Thus $c_{m,n}$ approximates

$s(t)$ at $t = 2^{-m}n$ with a scaling factor of $2^{-m/2}$.

Recursive computation of coeffs

Let us define

$c_{m,n} \equiv s[n] \equiv s(t)\big|_{t=2^{-m}n}$ for large m.

Using the dilation equation $\phi(t/2) = 2\sum_n h_0[n]\phi(t-n)$ we get

$$c_{m-1,n} = \int_{-\infty}^{\infty} s(t)\phi^*{}_{m-1,n}(t)dt = 2^{(m-1)/2}\int_{-\infty}^{\infty} s(t)\phi^*\left(\frac{2^m t-2n}{2}\right)dt$$

$$= 2^{(m-1)/2}\int_{-\infty}^{\infty} s(t)2\sum_i h_0[i]\phi^*(2^m t-2n-i)dt$$

$$= \sqrt{2}\sum_i h_0[i]\int_{-\infty}^{\infty} s(t)\phi^*{}_{m,2n+i}(t)dt = \sqrt{2}\sum_i h_0[i]c_{m,2n+i}$$

i.e. $$c_{m-1,n} = \sqrt{2}\sum_i h_0[i-2n]c_{m,i}$$

Filtering and downsampling

Given this eqn $c_{m-1,n} = \sqrt{2}\sum_{i} h_0[i-2n]c_{m,i}$

once $c_{m,n}$ is known, we can compute $c_{k,n}$ for $k < m$,

using a low pass filter $H_0{}^*(\omega)$ and downsampling $2n = i$.

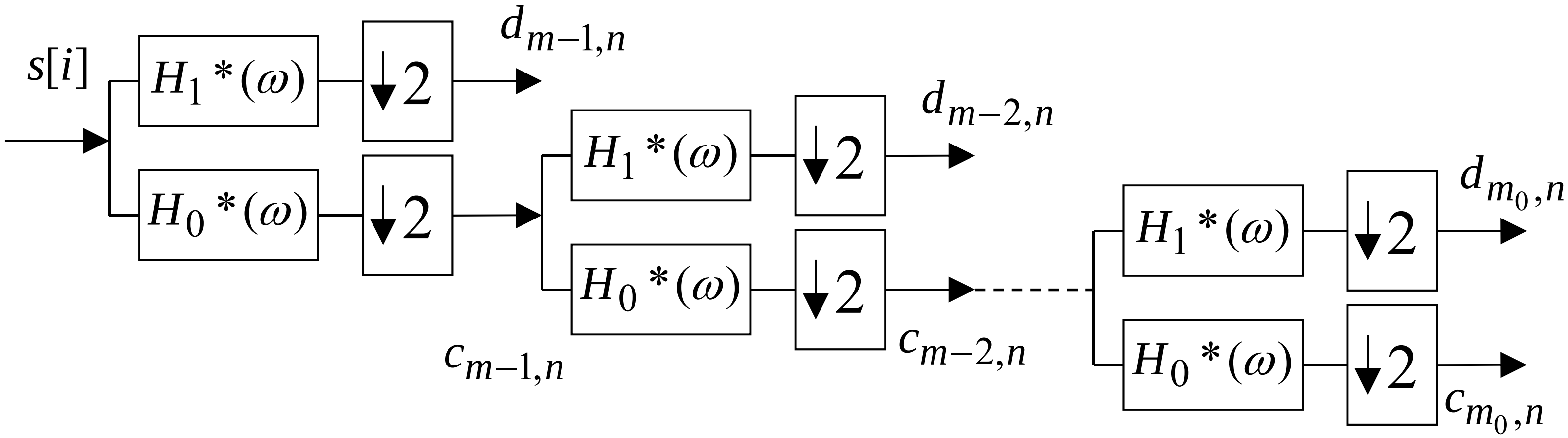
$$c_{m,n} \longrightarrow \boxed{H_0{}^*(\omega)} \longrightarrow \boxed{\downarrow 2} \longrightarrow c_{m-1,n}$$

Similarly, we can show $d_{m-1,n} = \sqrt{2}\sum_{i} h_1[i-2n]c_{m,i}$

i.e. a high pass filter $H_1{}^*(\omega)$ and downsampling.

$$c_{m,n} \longrightarrow \boxed{H_1{}^*(\omega)} \longrightarrow \boxed{\downarrow 2} \longrightarrow d_{m-1,n}$$

Filter Bank for Wavelet Series Coeffs



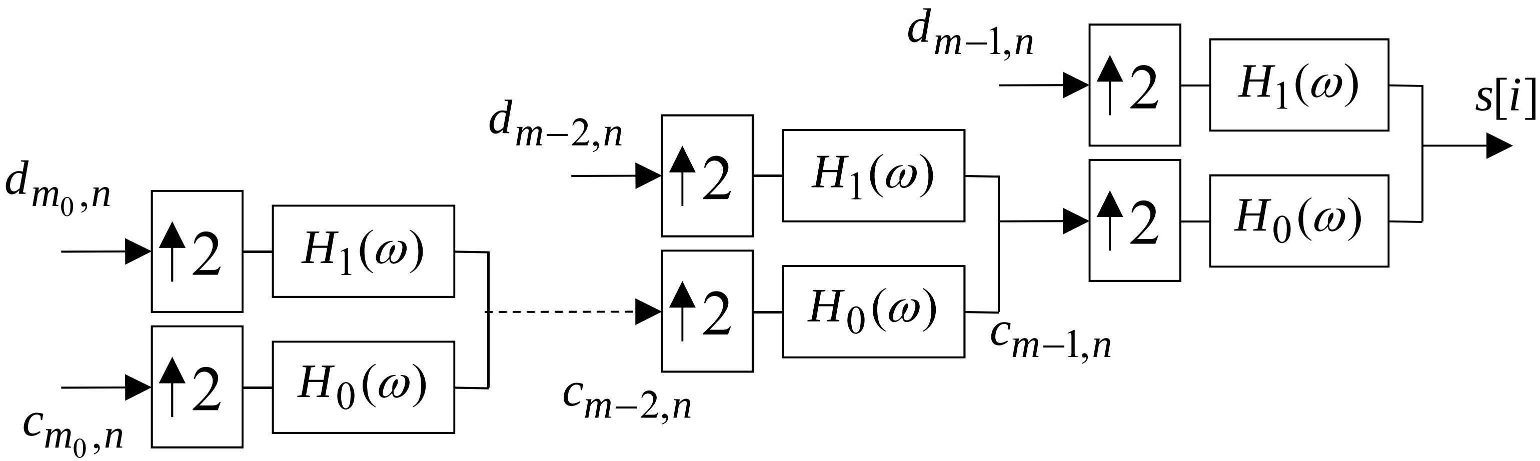
So for discrete-time samples $s[i]$, can compute wavelet transform directly by applying filter banks.

No need to compute the mother wavelet $\psi(t)$.

Signal recovery filterbank

Can also compute high - res coeffs from low - res coeffs :

$$c_{m,n} = \sqrt{2}\left(\sum_i h_0[n-2i]c_{m-1,i} + \sum_i h_1[n-2i]d_{m-1,i} \right)$$

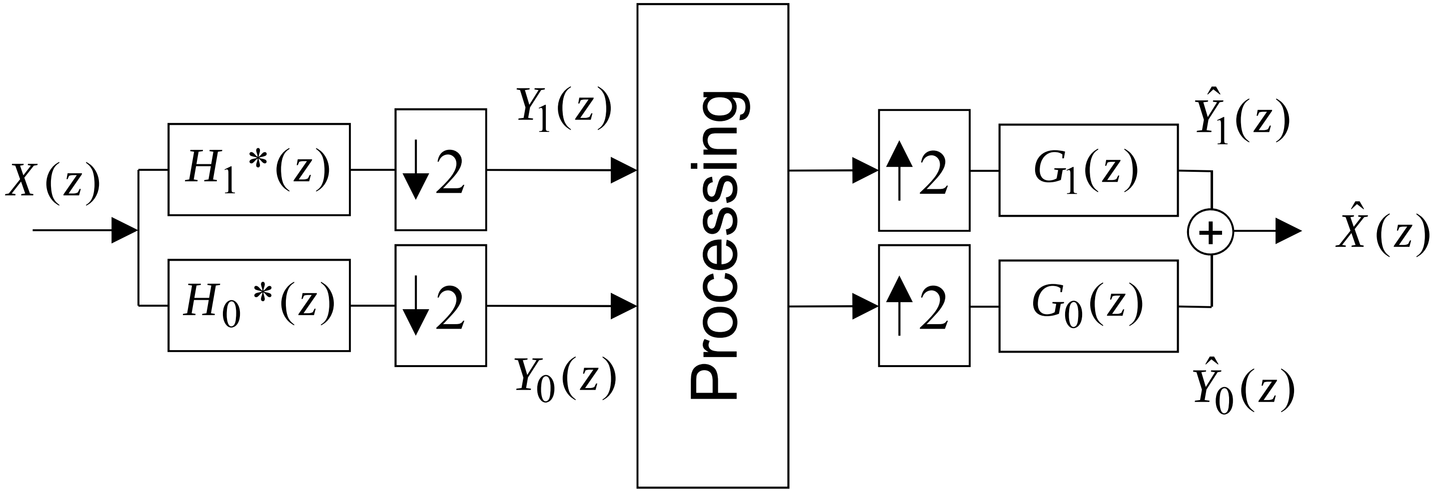


(For proof, see e.g. Qian)

So – don't need scaling functions or wavelets, just filter banks!

More general filterbanks

Analysis filters H_0, H_1 & synthesis filters G_0, G_1 may differ.



Recall z - transform : $H(z) = \sum_{n=0}^{N} h[n] z^{-n}$ (DFT with $z = e^{j\omega}$)

Note e.g. $\omega = 0 \rightarrow z = 1$ and $\omega = \pi \rightarrow z = -1$.

Assuming no processing, we typically want

Perfect Reconstruction (PR) - i.e. that $\hat{X}(z)$ is equal to $X(z)$

with a $+$ ve delay only, i.e. $\hat{X}(z) = z^{-l} X(z)$ for some $l \geq 0$.

Perfect Reconstruction Filterbanks

It turns out that (see e.g. Qian, sec 6.1)

$$X(z) = \frac{1}{2}[G_0(z)H_0(z) + G_1(z)H_1(z)]X(z)$$

$$+ \frac{1}{2}[G_0(z)H_0(-z) + G_1(z)H_1(-z)]X(-z)$$

For PR, want 2nd ("alias") term to be zero:

$$G_0(z)H_0(-z) + G_1(z)H_1(-z) = 0$$ *biorthogonal filter bank*

a possible solution is

$G_0(z) = H_1(-z)$ and $G_1(z) = -H_0(-z)$

For PR, also want 1st term to be a delay, e.g.

$$2z^{-l} = G_0(z)H_0(z) + G_1(z)H_1(z) = G_0(z)H_0(z) - H_0(-z)G_0(-z)$$

i.e.

$P_0(z) - P_0(-z) = 2z^{-l}$ where $P_0(z) = H_0(z)G_0(z)$

Daubechies Wavelet family

So, wavelet design method is :

1) Design product filter $P_0(z)$ to satisfy $P_0(z) - P_0(-z) = 2z^{-l}$

2) Factorize $P_0(z)$ into $H_0(z)$ and $G_0(z)$

Example : The kth order Daubechies wavelets ("db$k$"),

$$H_0(z) = (1 + z^{-1})^k \prod_{i=1}^{k-1} (z_i - z^{-1})$$

$$G_0(z) = (1 + z^{-1})^k \prod_{i=1}^{k-1} (\frac{1}{z_i} - z^{-1})$$

where z_i and $1/z_i$ are roots of a polynomial of degree $2k-2$

Daubechies Wavelet

Using $k=1$ for db$k$ wavelet we get

$$H_0(z) = G_0(z) = (1+z^{-1}) \quad \text{or } H_0(\omega) = (1+e^{-j\omega n})$$

i.e. the Haar wavelet (apart from a scaling factor).

Using $k=2$ for dbk wavelet we get

$$H_0(z) = (1+z^{-1})^2(c-z^{-1}) \text{ and } G_0(z) = (1+z^{-1})^2(\frac{1}{c}-z^{-1})$$

where $c = 2-\sqrt{3}$ and $1/c$ are the roots of the polynomial

$$Q(z) = -1+4z^{-1}-z^{-2}$$

Daubechies wavelets are actually *orthogonal*.

(E.g. check the power complementary condition)

Orthogonal filter banks

Orthogonal filter banks are orthogonal in the sense that

$$\sum_n h_i[n-2k]h_i[n] = \delta(k) \text{ and } \sum_n h_i[n-2k]h_l[n] = 0 \text{ for } i \neq l$$

which can be achieved by e.g. (given without proof)

$$H_1(z) = (-z)^{-N} H_0(-z^{-1})$$

i.e. that high-pass analysis filter h_1 is alternating flip of h_0 :

$$(h_1[0], h_1[1], h_1[2], \ldots, h_1[N]) = (h_0[N], -h_0[N-1], h_0[N-2], \ldots)$$

Now, since $G_0(z) = H_1(-z)$ and $G_1(z) = -H_0(-z)$, we get e.g

$$G_0(z) = z^{-N} H_0(z^{-1})$$

so e.g. the resynthesis filter $\gamma_0[n] \Leftrightarrow G_0(z)$ is flip of $h_0[n]$:

$$(\gamma_0[0], \gamma_0[1], \gamma_0[2], \ldots, \gamma_0[N]) = (h_0[N], h_0[N-1], \ldots, h_0[0])$$

Example: Daubechies for k=2

Daubechies wavelet : analysis filters

$$h_0[n] = (h_0[0], h_0[1], h_0[2], h_0[3]) = \left(\frac{1+\sqrt{3}}{4\sqrt{2}}, \frac{3+\sqrt{3}}{4\sqrt{2}}, \frac{3-\sqrt{3}}{4\sqrt{2}}, \frac{1-\sqrt{3}}{4\sqrt{2}}\right)$$

$$h_1[n] = (h_1[0], h_1[1], h_1[2], h_1[3]) = (h_0[3], -h_0[2], h_0[1], -h_0[0])$$

Synthesis filters :

$$\gamma_0[n] = (h_0[3], h_0[2], h_0[1], h_0[0])$$

$$\gamma_1[n] = (h_1[3], h_1[2], h_1[1], h_1[0]) = (h_0[0], -h_0[1], h_0[2], -h_0[3])$$

Example (cont)