Week 1: Introduction to Wavelets

(Burrus et al, ch. 1)

Norms and Inner Products

x = vector = $\begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}$

inner product $\langle x, y \rangle = \sum_i \overline{x_i}\, y_i$ ($\overline{\ }$ = complex conjugate)

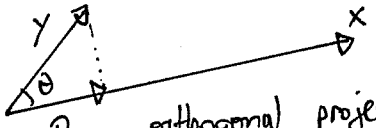
norm $\|x\| = \sqrt{\langle x, x \rangle}$

$f(t)$ = function

$\langle f, g \rangle = \int \overline{f(t)}\, g(t)\, dt$

$\|f\| = \sqrt{\langle f, f \rangle}$

orthonormal expansion



P_y = orthogonal projection of y onto x

$\langle x, y \rangle = \|x\| \cdot \|y\| \cdot \cos\theta$

$\Rightarrow P_y = \underbrace{(\|y\| \cos\theta)}_{\text{length}} \cdot \underbrace{\frac{x}{\|x\|}}_{\text{direction}} = \frac{\langle y, x \rangle}{\langle x, x \rangle} x$

if x is unit vector (i.e. $\|x\| = 1$)

$P_y = \langle y, x \rangle x$

Likewise, if $e_1, e_2 \ldots$ are orthonormal (i.e. length 1, mutually orthogonal)

$$Py = \langle y, e_1 \rangle e_1 + \langle y, e_2 \rangle e_2 + \ldots = \sum_i \langle y, e_i \rangle e_i$$

(orthogonal projection onto $\text{span}(e_i)$)

Wavelet expansion

$\psi(t)$ = mother wavelet

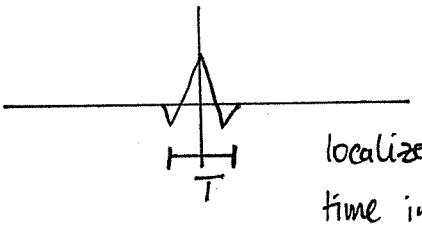
$$\psi_{jk}(t) = 2^{j/2} \psi(2^j t - k)$$

assume $\{\psi_{jk}\}$ are orthonormal, then

$$f(t) \sim \sum_{jk} a_{jk} \psi_{jk}(t), \quad a_{jk} = \langle f, \psi_{jk} \rangle$$

Assume ψ is localized in both time and frequency

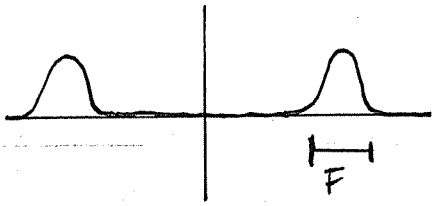
$\psi(t)$



localized in some time interval T

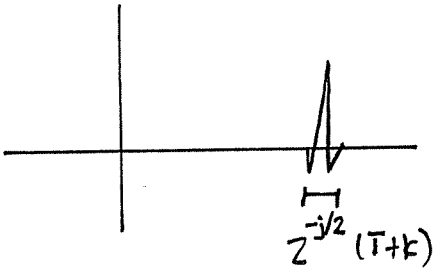
$$\hat{\psi}(\xi) = \frac{1}{\sqrt{2\pi}} \int \psi(t) e^{-it\xi} dt$$

Fourier transform

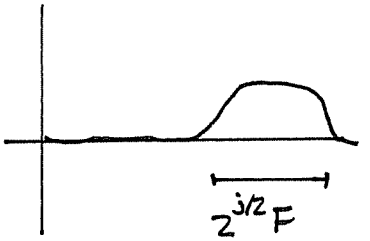


localized in some frequency interval F

Then



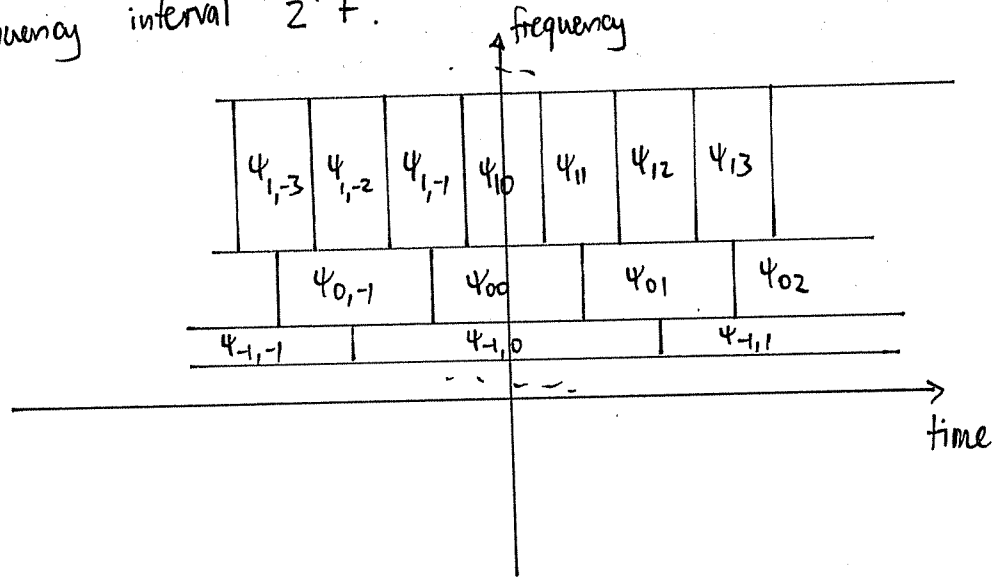
$\psi_{jk}(t)$



$\hat{\psi}_{jk}(\xi)$

$$\langle f, \psi_{jk} \rangle = \sqrt{2\pi}\ \langle \hat{f}, \hat{\psi}_{jk} \rangle$$

interpreted as energy content of f in time interval $2^{-j/2}(T+k)$, frequency interval $2^{j/2}F$.



Consider $F(t,\xi)$ = time-frequency distribution of $f(t)$

$F(t,\xi)$ = frequency content of f at time t, frequency ξ

The Heisenberg uncertainty principle says that point values of F are undefined. The only thing that makes sense are local averages of F over areas of a certain minimal size (area $\geq \frac{1}{2}$, with my definition of $\hat{f}$).

So, the wavelet coefficients $a_{jk} = \langle f, \psi_{jk} \rangle$ are interpreted as local averages of the (hypothetical) F. The boxes change shape, but they all have the same area.

low frequency $\leftrightarrow$ good frequency resolution
bad time resolution

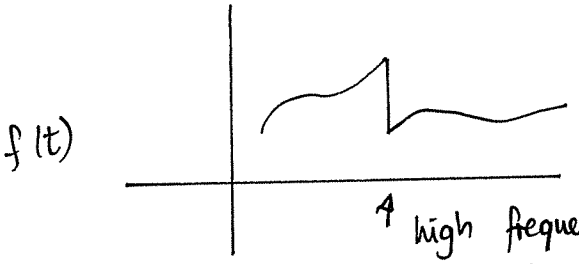
high frequency $\leftrightarrow$ bad frequency resolution
good time resolution

The frequency resolution is arranged in octaves

$\ldots, [\frac{1}{2}\xi_0, \xi_0], [\xi_0, 2\xi_0], [2\xi_0, 4\xi_0], [4\xi_0, 8\xi_0] \ldots$

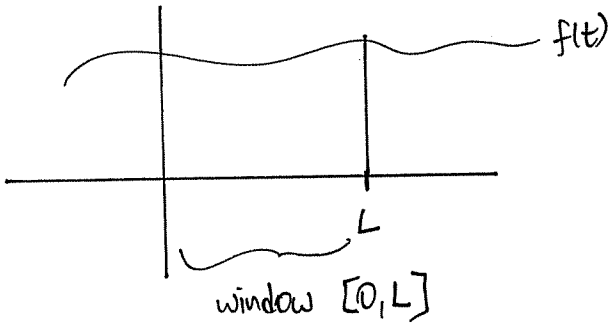
Expressed differently: the time resolution is proportional to the wavelength. This makes sense: to reliably measure a frequency, you have to observe the signal over a time period comparable to its period.

The wavelet transform is good for resolving discontinuities in the signal.



↑ high frequency components present for a short time

In contrast: Short-Time Fourier Transform



$f(t)$ restricted to $[0,L]$ can be expanded in Fourier series

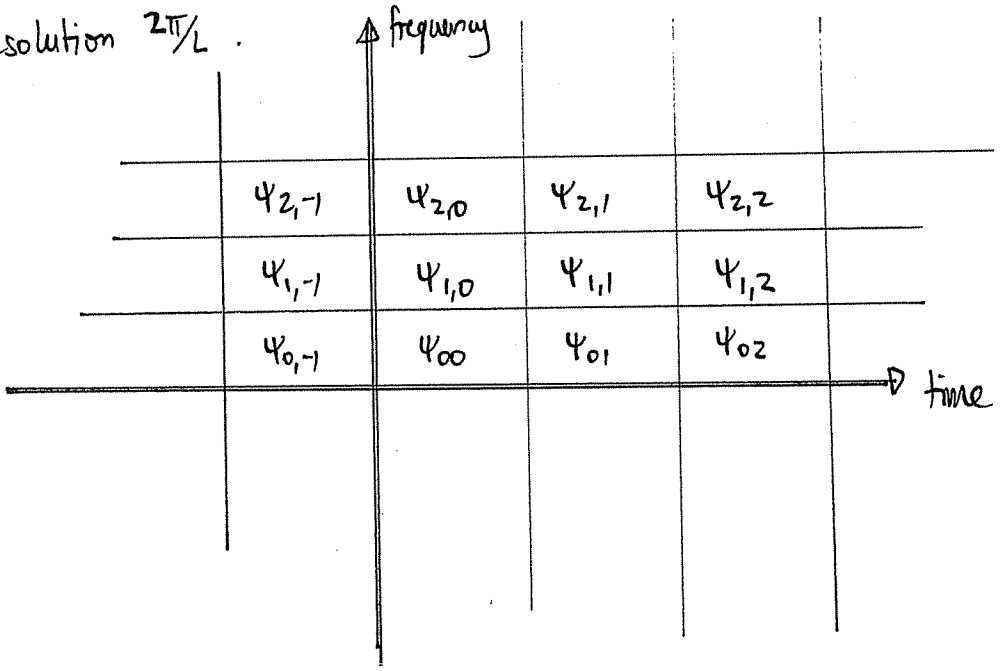
$$f(t) = \frac{1}{L} \sum_j \hat{f}_j \, e^{i \frac{2\pi}{L} j t}, \qquad \hat{f}_j = \int_0^L f(t) \, e^{-i \frac{2\pi}{L} j t} \, dt$$

By splitting the time axis into intervals $[kL, (k+1)L]$, we can get a different time-frequency expansion:

$$f(t) \sim \sum_{jk} c_{jk} \psi_{jk}(t), \qquad c_{jk} = \langle f, \psi_{jk} \rangle$$

$$\psi_{jk}(t) = \begin{cases} \frac{1}{\sqrt{L}} \, e^{i \frac{2\pi}{L} j t}, & t \in [kL, (k+1)L] \\ 0 & \text{otherwise} \end{cases}$$

The difference between this and the wavelet transform is that all ψ_{jk} have the same time resolution L, and the same frequency resolution $2\pi/L$.



Week 2: Multiresolution Approximation

(Burrus et al, ch. 2)

We take a different approach as in ch. 1, but end up in the same place.

Def: A multiresolution approximation (MRA) of L^2 is a nested chain of subspaces

$$\cdots \subset V_{-1} \subset V_0 \subset V_1 \subset V_2 \subset \cdots$$

with properties

(i) $\overline{\bigcup_{j=-\infty}^{\infty} V_j} = L^2(\mathbb{R})$, $\quad \bigcap V_j = \{0\}$

(ii) $f(t) \in V_j \iff f(2t) \in V_{j+1}$

(iii) $f(t) \in V_0 \iff f(t-k) \in V_0 \quad \forall k \in \mathbb{Z}$

(iv) There is a single function $\varphi(t)$, called a scaling function, so that its integer translates $\varphi(t-k)$ form an unconditional basis of V_0.

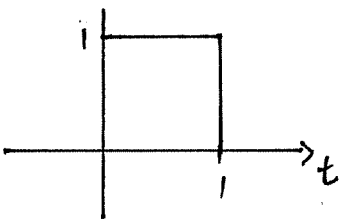
For now, we also assume that $\{\varphi(t-k) : k \in \mathbb{Z}\}$ are orthonormal.

Immediate consequence:

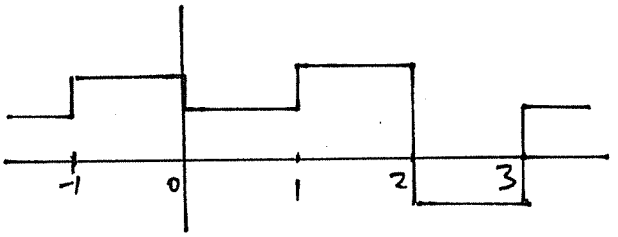
let $\varphi_{jk}(t) = 2^{j/2} \varphi(2^j t - k)$

then $\{\varphi_{jk}(t) : k \in \mathbb{Z}\}$ is a basis of V_j.

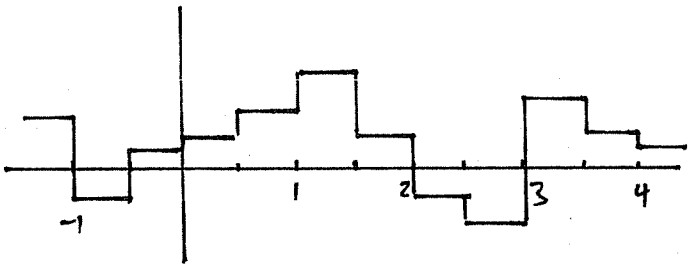
standard example:

$\varphi(t) =$ 

The shifted box functions form a basis of V_0. Every function in V_0 is a sum of shifted boxes:



Functions in V_1 are the functions in V_0, compressed by a factor of 2:



It is obvious that any function in V_0 (constant on integer intervals) is also constant on half-integer intervals, so it is in V_1: $V_0 \subset V_1$

The functions in V_2 are constant on quarter-integer intervals, etc.
Functions in V_{-1} are constant on double-integer intervals.

Interpretation: The scaling function φ is localized in some interval of size L ($L=1$ in the example).
The orthogonal projection of $f(t)$ onto V_j

$$P_j f(t) = \sum_k \langle f, \varphi_{jk} \rangle \varphi_{jk}(t)$$

is an approximation to f at resolution $2^j L$.

Our textbook has a better picture than I can draw:

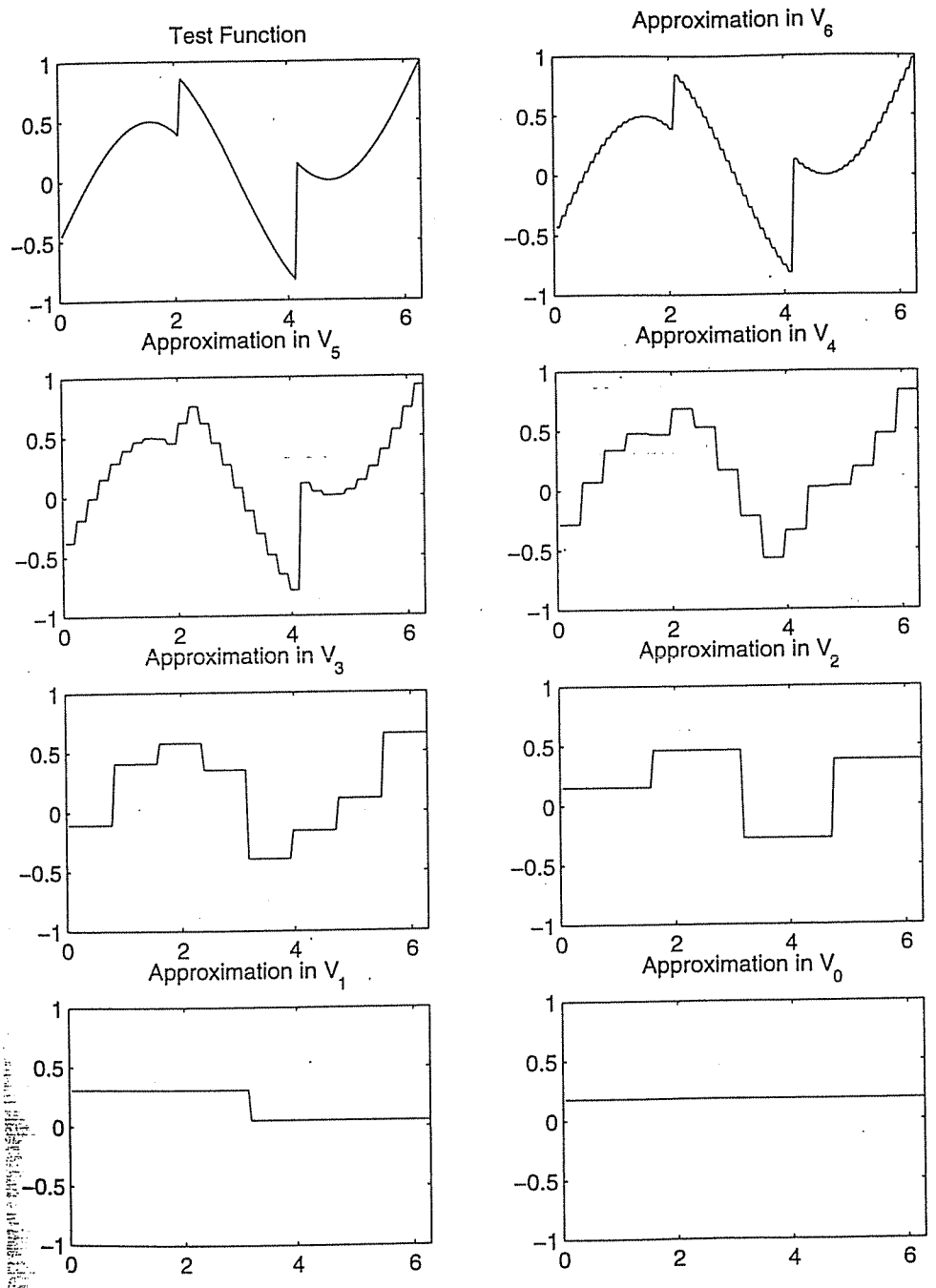


Figure 2.15. Haar Function Approximation in $\mathcal{V}_j$

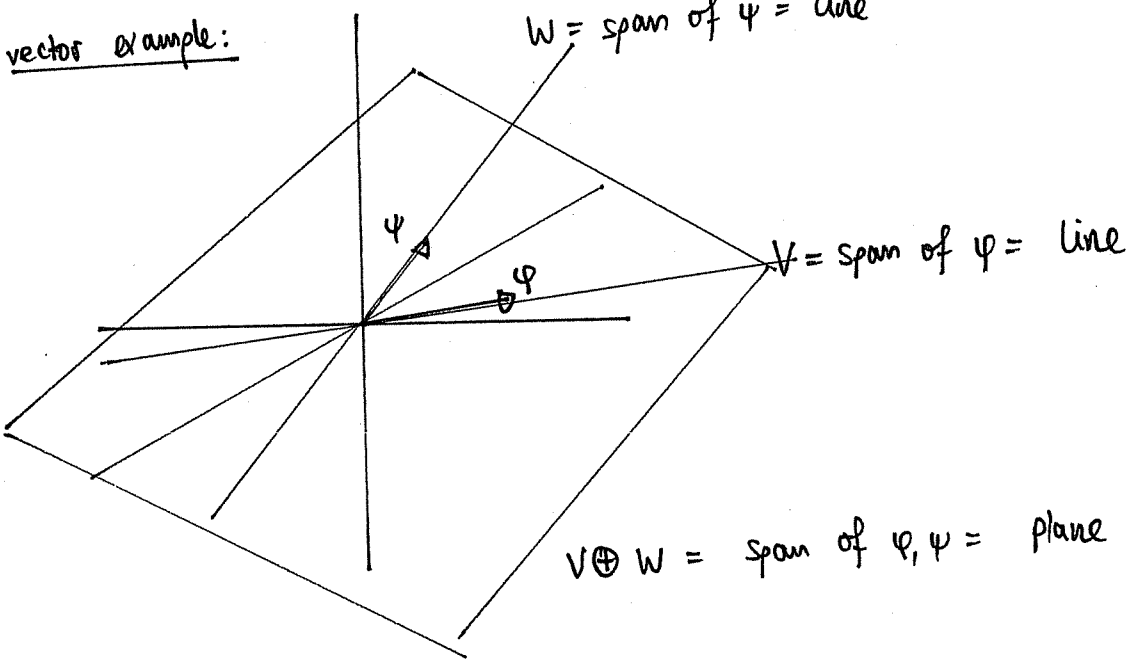
property (i): $\overline{\bigcup V_j} = L^2$ says: for any $f \in L^2$, the projection $P_j f$ gets arbitrarily close to f, as $j \to \infty$.

Since $V_0 \subset V_1$, we can look for another space W_0 so that

$$V_0 \oplus W_0 = V_1$$

(The $\oplus$ means "direct sum of vector spaces"; "direct" means that V_0, W_0 have no vectors in common (except 0), and the sum of vector spaces is the set of all possible sums of elements).

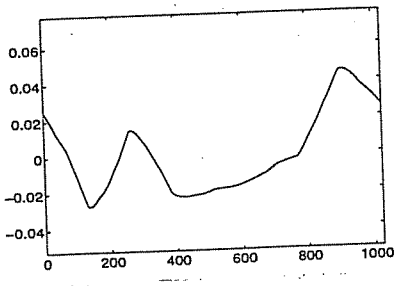
vector example:



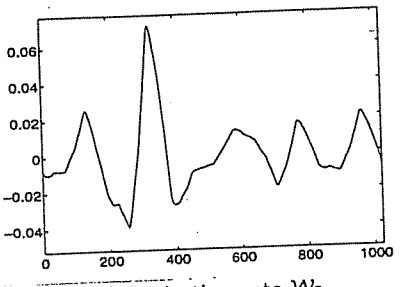
As you can see, ψ is not unique: for a fixed φ and fixed plane, we can choose ψ = any vector in the plane linearly independent of φ.
There is one special vector ψ perpendicular to φ. That one is unique (up to a constant multiplier).

In the general case, there is likewise a unique W_0 orthogonal to V_0, but many other possibilities as well. For now, we'll stick with the orthogonal choice.
Under some mild technical conditions, it turns out that W_0 is spanned by integer translates of a single function ψ, called the mother wavelet. That is the same ψ as before, in lecture 1.

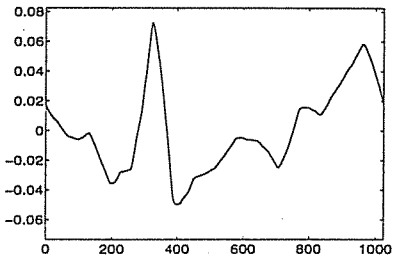
Interpretation: $Q_j f = P_{j+1} f - P_j f$ = fine detail in f at resolution 2^{-j}



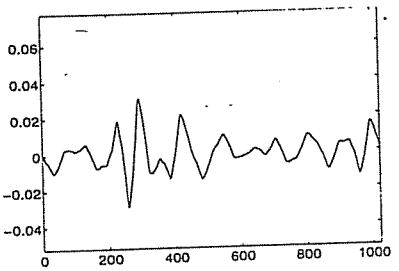
(a) Projection onto $\mathcal{V}_0$



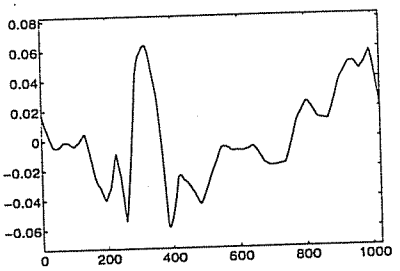
(b) Projection onto $\mathcal{W}_0$



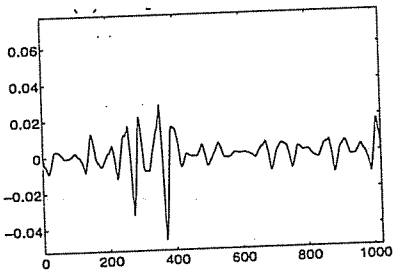
(b) Projection onto $\mathcal{V}_1$



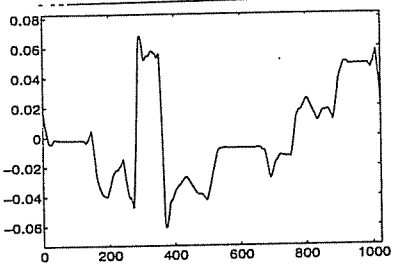
(c) Projection onto $\mathcal{W}_1$



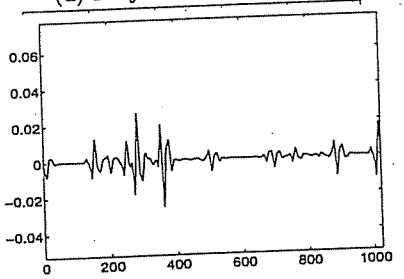
(c) Projection onto $\mathcal{V}_2$



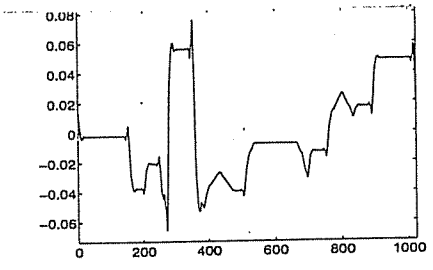
(d) Projection onto $\mathcal{W}_2$



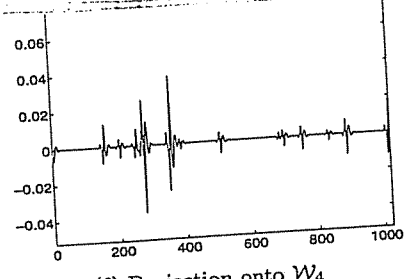
(d) Projection onto $\mathcal{V}_3$



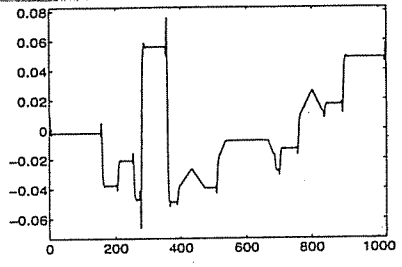
(e) Projection onto $\mathcal{W}_3$



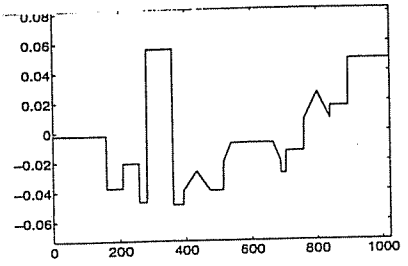
(e) Projection onto $\mathcal{V}_4$



(f) Projection onto $\mathcal{W}_4$

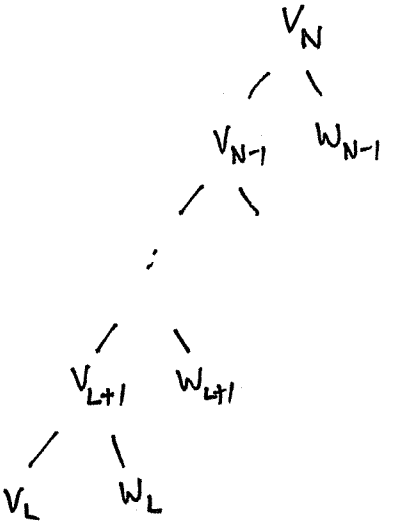


(f) Projection onto $\mathcal{V}_5$



(b) Projection onto $\mathcal{W}_5$

In practice, the multiresolution tree is finite:



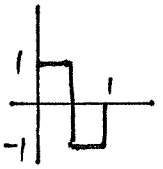
resolution of original signal is 2^{-N}

decompose it into a slightly blurred signal at resolution $2^{-(N-1)}$, and the fine detail at resolution $2^{-(N-1)}$

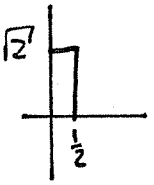
⋮

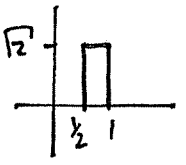
blurred signal at resolution 2^{-L}, and fine detail at resolution 2^{-L}

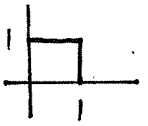
Mathematically, the decomposition is simply a change of basis.
No information is created or destroyed.

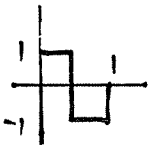
example: use φ = box function, ψ = 

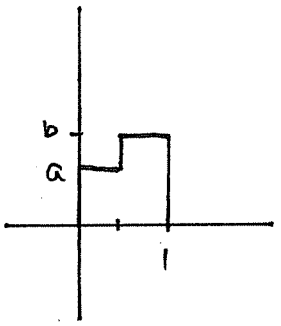
restrict attention to interval $[0,1]$.

basis of V_1 is φ_{10} = 

φ_{11} = 

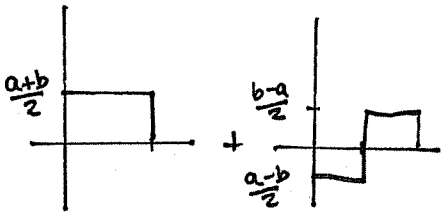
basis of V_0 is φ_{00} = 

W_0 ψ_{00} = 



$$= \frac{1}{\sqrt{2}} a \cdot \varphi_{10} + \frac{1}{\sqrt{2}} b \cdot \varphi_{11}$$

$=$



$$= \frac{a+b}{2} \cdot \varphi_{00} + \frac{a-b}{2} \cdot \psi_{00}$$

<u>discrete example:</u> $V_1 = \mathbb{R}^2$, basis $\varphi_{10} = \sqrt{2}\begin{pmatrix}1\\0\end{pmatrix}$, $\varphi_{11} = \sqrt{2}\begin{pmatrix}0\\1\end{pmatrix}$

$V_0 = \text{span}\left\{\begin{pmatrix}1\\1\end{pmatrix}\right\}$ basis $\varphi_{00} = \begin{pmatrix}1\\1\end{pmatrix}$

$W_0 = \text{span}\left\{\begin{pmatrix}1\\-1\end{pmatrix}\right\}$ basis $\psi_{00} = \begin{pmatrix}1\\-1\end{pmatrix}$

arbitrary vector $v = \begin{pmatrix}a\\b\end{pmatrix}$ can be expressed as $\frac{1}{\sqrt{2}} a\varphi_{10} + \frac{1}{\sqrt{2}} b\varphi_{11}$, using the basis of V_1, or as $\frac{a+b}{2} \cdot \varphi_{00} + \frac{a-b}{2}\psi_{00}$, using the bases of V_0, W_0.

One step of the wavelet decomposition is taking a signal in V_N, and expressing it in terms of the bases of V_{N-1}, W_{N-1}.

$$f(t) = \sum_j b_{Nj}\, \varphi_{Nj}(t)$$

$$= \sum_j b_{N-1,j}\, \varphi_{N-1,j}(t) + \sum_j c_{N-1,j}\, \psi_{N-1,j}(t)$$

ultimately,

$$V_N = V_{N-1} \oplus W_{N-1} = \ldots = V_L \oplus W_L \oplus W_{L+1} \oplus \ldots \oplus W_{N-1}$$

if we let $L \to -\infty$, $N \to \infty$:

$$L^2 = \ldots \oplus W_{-1} \oplus W_0 \oplus W_1 \oplus \ldots = \bigoplus_{j=-\infty}^{\infty} W_j$$

in theory

$$f = \sum_{jk} c_{jk} \psi_{jk}(t)$$ (same as in lecture 1) infinite expansion

but in practice

$$f(t) = \sum_k b_{Lk} \varphi_{Lk}(t) + \sum_{j=L}^{N-1} c_{jk} \psi_{jk}(t)$$ (finite expansion)

Finally, in preparation for lecture 3, note that since $V_0 \subset V_1$, the ~~exact~~ scaling function $\varphi = \varphi_{00}$ can be expressed in terms of the basis of V_1:

$$\varphi(t) = \sum_k h_k \varphi_{1k}(t) = \sqrt{2} \sum_k h_k \varphi(2t-k)$$

Likewise

$$\psi(t) = \sqrt{2} \sum_k g_k \varphi(2t-k)$$

These are the <u>two-scale recursion relations</u>, which will form the basis of the algorithm.

Lecture 3: Filter Banks and the Discrete Wavelet Transform

(Burrus et al, ch. 3)

As we saw in lecture 2, the Discrete Wavelet Transform (DWT) takes a function in some space V_N, and decomposes it into its components in $V_L \oplus W_L \oplus W_{L+1} \oplus \cdots \oplus W_{N-1}$. It suffices to work out the details for $V_1 = V_0 \oplus W_0$. The algorithm works the same at all levels, and is then applied recursively.

Recall:

$\varphi(t)$ = scaling function

$\varphi_{jk}(t) = 2^{j/2}\varphi(2^j t - k)$

$\psi(t)$ = mother wavelet

$\psi_{jk}(t) = 2^{j/2}\psi(2^j t - k)$

We assume for now that the basis functions are orthonormal:

$\langle \varphi_{jk}, \varphi_{lm} \rangle = \delta_{jl}\,\delta_{km}$

$\langle \psi_{jk}, \psi_{lm} \rangle = \delta_{jl}\,\delta_{km}$

$\langle \varphi_{jk}, \psi_{lm} \rangle = 0$

$\delta_{jk} = \begin{cases} 1 & \text{if } j=k \\ 0 & j \neq k \end{cases}$ (Kronecker δ)

Recall:

$$\varphi(t) = \sum_k h_k \varphi_{1k}(t) = \sqrt{2}\sum_k h_k \varphi(2t-k) \qquad (1)$$

$$\psi(t) = \sum_k g_k \varphi_{1k}(t) = \sqrt{2}\sum_k g_k \varphi(2t-k)$$

Recall:

$P_0 f(t) = \sum_k \langle f, \varphi_{0k} \rangle \varphi_{0k}(t)$ orthogonal projection onto V_0

$P_1 f(t) = \sum_k \langle f, \varphi_{1k} \rangle \varphi_{1k}(t)$ ... V_1

$Q_0 f(t) = \sum_k \langle f, \psi_{0k} \rangle \psi_{0k}(t)$ ... W_0

Let's generalize the two-scale recursion relations (1) a little:

$$\varphi_{0j}(t) = \varphi(t-j) = \sqrt{2}\sum_k h_k \varphi(2(t-j)-k)$$
$$= \sqrt{2}\sum_k h_k \varphi(2t-(2j+k))$$
$$= \sum_k h_k \varphi_{1,2j+k}(t) = \sum_k h_{k-2j}\varphi_{1,k}(t)$$

Now $P_1 \varphi_{0k}(t) = \varphi_{0k}(t)$, since $\varphi_{0k} \in V_0 \subset V_1$

So
$$\varphi_{0j}(t) = \sum_k \langle \varphi_{0j}, \varphi_{1k}\rangle \varphi_{1k}(t)$$
$$= \sum_k h_{k-2j}\varphi_{1k}(t)$$
} compare

$$\boxed{\langle \varphi_{0j}, \varphi_{1k}\rangle = h_{k-2j}}$$

Likewise
$$\boxed{\langle \psi_{0j}, \varphi_{1k}\rangle = g_{k-2j}}$$

(2)

Now take some $f \in V_1$

$$f(t) = \underbrace{\sum_k c_{1k}\varphi_{1k}(t)}_{\text{in } V_1} = \underbrace{\sum_j c_{0j}\varphi_{0j}(t) + \sum_j d_{0j}\psi_{0j}(t)}_{\text{in } V_0 \oplus W_0}$$

$$f(t) = P_0 f(t) + Q_0 f(t)$$
$$= \sum_j \langle f, \varphi_{0j}\rangle \varphi_{0j}(t) + \sum_j \langle f, \psi_{0j}\rangle \psi_{0j}(t)$$
$$= \sum_j \underbrace{\langle \sum_k c_{1k}\varphi_{1k}, \varphi_{0j}\rangle}_{c_{0j}} \varphi_{0j}(t) + \sum_j \underbrace{\langle \sum_k c_{1k}\varphi_{1k}, \psi_{0j}\rangle}_{d_{0j}} \psi_{0j}(t)$$

By (2)

$$c_{0j} = \sum_k c_{1k} \langle \varphi_{1k}, \varphi_{0j} \rangle = \sum_k h_{k-2j}\, c_{1k}$$

This also works for d_{0j} (with g coefficients), and at all levels:

$$\boxed{\begin{aligned} c_{n-1,j} &= \sum_k h_{k-2j}\, c_{n,k} \\ d_{n-1,j} &= \sum_k g_{k-2j}\, c_{n,k} \end{aligned}} \qquad (3)$$

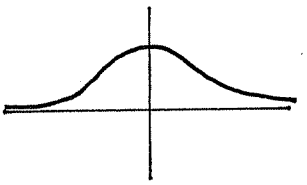
Def: The convolution of two functions $f(t)$, $g(t)$ is

$$(f * g)(t) = \int f(s)\, g(t-s)\, ds = \int f(t-s)\, g(s)\, ds$$

In engineering, this is called filtering. g is the filter, and $f * g$ is the filtered version of f.

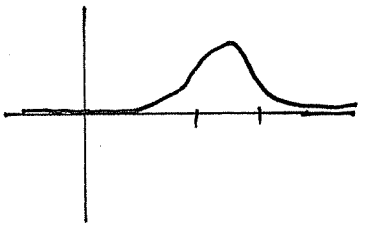
In terms of the Fourier transform,

$$(f * g)\hat{}(\xi) = \sqrt{2\pi}\, \hat{f}(\xi)\, \hat{g}(\xi)$$

$\hat{g}$: 

low-pass filter

results in a smoothing operation

$\hat{g}$: 

bandpass filter

selects frequencies in a certain range

If $\{x_i\}$, $\{y_i\}$ are two sequences, their discrete convolution is

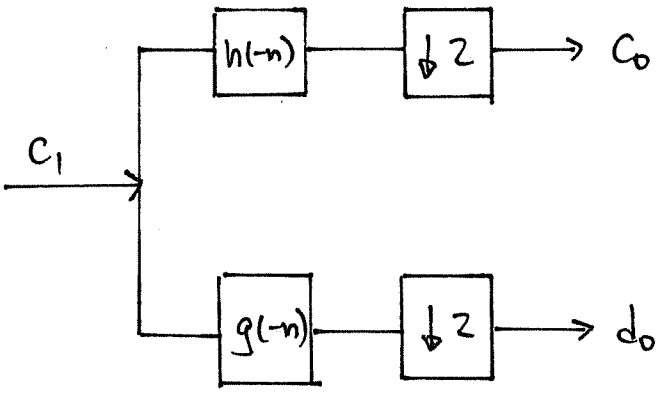
$$(x*y)_j = \sum_k x_k y_{j-k} = \sum_k x_{j-k} y_k$$

Now look at ③: These are discrete filtering operations (convolutions), except (a) the subscripts of h, g have wrong signs

(b) the subscripts on the right contain $2j$ instead of j.

That means that we only use the even-numbered coefficients.

Engineers like to draw this like this:

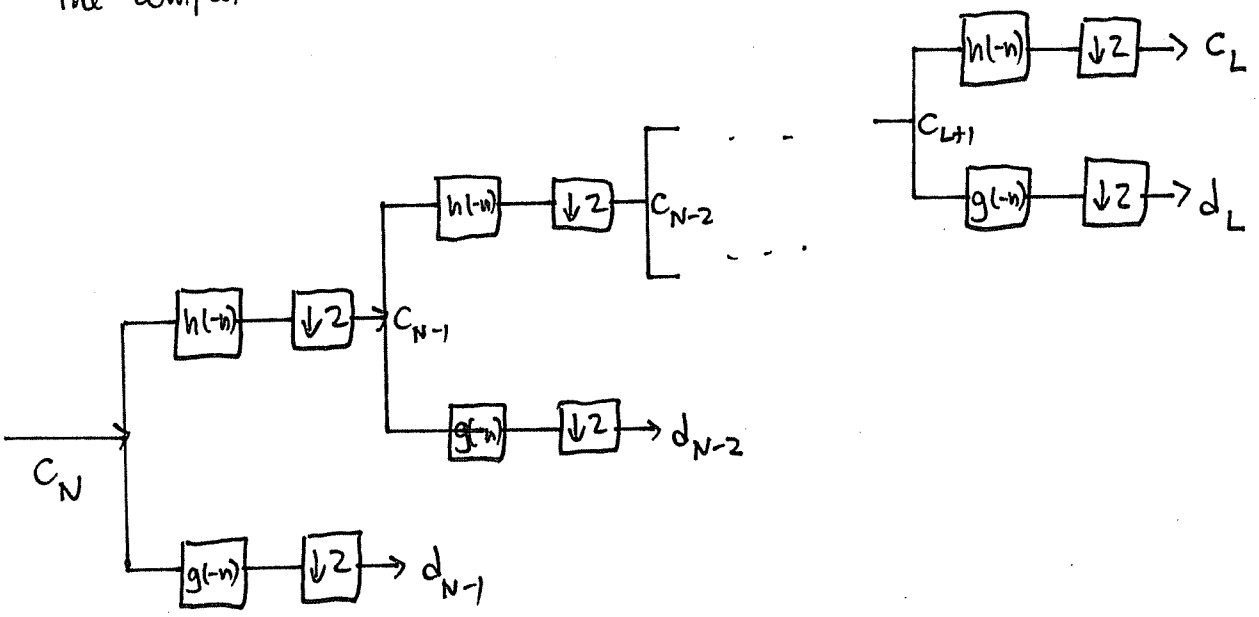


filtering with h, g coefficients in reverse order

downsampling by 2

The number of coefficients has not changed: each of c_0, d_0 has only half as many coefficients as c_1.

The complete DWT consists of repeated applications of this:



<u>operation count</u>:

If h, g each have k coefficients, each filtering operation takes k multiplications. If input sequence c_N has L coefficients, we need kL multiplications at the first level.

c_{N-1} only has length $\frac{L}{2}$ $\Rightarrow$ $\frac{kL}{2}$ multiplications at second level

Altogether: $$kL\left(1 + \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \ldots\right) < 2kL$$

The amount of work is <u>$O(L)$</u> (proportional to L)

The Fast Fourier Transform has a similar tree structure, but you have to continue with <u>both</u> branches at each level. The total work is $kL + kL + \ldots + kL = kL \cdot$ (number of levels)

$$= kL \cdot \log_2 L$$

(here k is some constant; I think it is 5, but that is not important).

For long sequences, the DWT is faster than the FFT.

Now let's go in the opposite direction: assume c_0, d_0 are given, we want c_1.

$$f(t) = P_1 f(t) = \sum_k \langle f, \varphi_{1k} \rangle \varphi_{1k}(t)$$

$$= \sum_k \underbrace{\langle \sum_j c_{0j} \varphi_{0j} + \sum_j d_{0j} \psi_{0j}, \varphi_{1k} \rangle}_{c_{1k}} \varphi_{1k}(t)$$

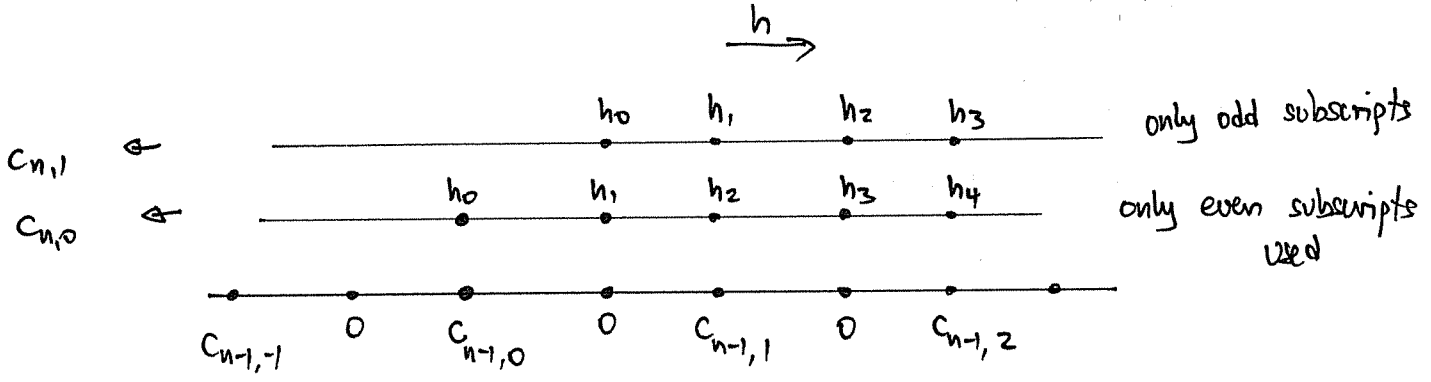
so $$c_{1k} = \sum_j c_{0j} \underbrace{\langle \varphi_{0j}, \varphi_{1k} \rangle}_{h_{k-2j}} + \sum_j d_{0j} \underbrace{\langle \psi_{0j}, \varphi_{1k} \rangle}_{g_{k-2j}}$$

At level n:

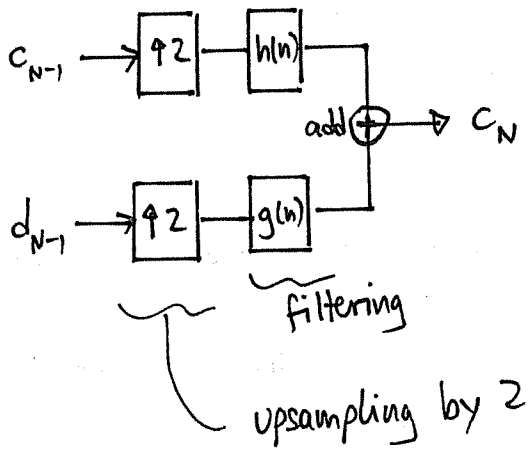
$$\boxed{c_{n,k} = \sum_j h_{k-2j} c_{n-1,j} + \sum_j g_{k-2j} d_{n-1,j}} \quad (4)$$

This is again a discrete convolution, with h, g in natural order this time. However, the presence of $2j$ on the right means that at one point, we filter with $h_1, h_3, h_5 \ldots$, at the next point with $h_0, h_2, h_4 \ldots$ etc.

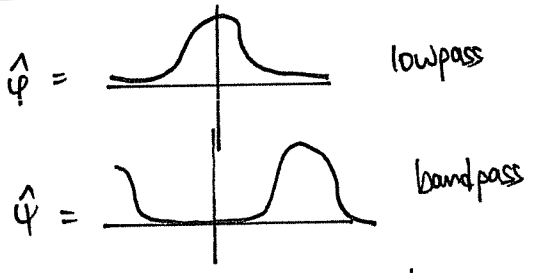
An easy way to implement this is to insert a zero in between any two elements of c_{n-1}, d_{n-1}:



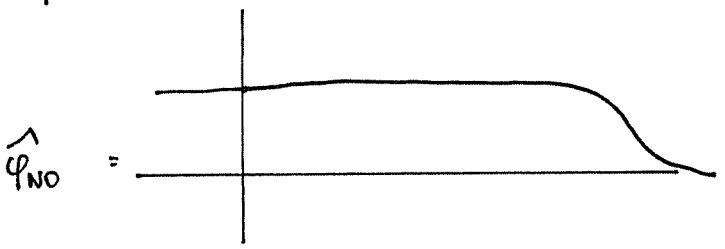
Engineers like to draw this like this:



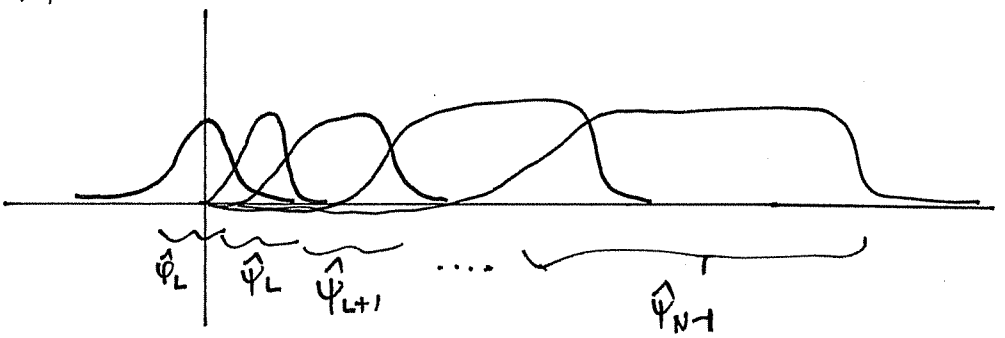
Another filtering viewpoint:



f in V_N:



f in $V_L \oplus W_L \oplus .. \oplus W_{N-1}$



split the frequency range of f into octaves

Notation

signal in time domain: $\{x(n)\}$ or $\{x_n\}$

frequency domain: $X(\omega) = \sum x_n e^{-in\omega}$

z-transform: $X(z) = \sum x_n z^{-n}$

} basically the same, with $z = e^{i\omega}$

Filtering in time domain corresponds to multiplication in frequency:

if $x \leftrightarrow X$

$y \leftrightarrow Y$

then $x * y \leftrightarrow X \cdot Y$

We want to express the DWT in the frequency domain. For that, we need to check the effect of some time-domain operations on the frequency domain:

time reversal

$\{h_n\} \leftrightarrow H(\omega)$ or $H(z)$

$\{h_{-n}\} \leftrightarrow \overline{H(\omega)}$ or $H\left(\frac{1}{z}\right)$

(note how $\overline{z} = \frac{1}{z}$ if $z = e^{i\omega}$)

upsampling

$\{x_n\} \leftrightarrow X(\omega)$ or $X(z)$

$\{(\uparrow 2x)_n\} \leftrightarrow X(2\omega)$ or $X(z^2)$

downsampling

$\{x_n\} \leftrightarrow X(\omega)$ or $X(z)$

$\{(\downarrow 2x)_n\} \leftrightarrow \frac{1}{2}\left[X\left(\frac{\omega}{2}\right) + X\left(\frac{\omega}{2} + \pi\right)\right]$

or $\frac{1}{2}\left[X(\sqrt{z}) + X(-\sqrt{z})\right]$

(usually written $(\downarrow 2X)(z^2) = \frac{1}{2}[X(z) + X(-z)]$ to avoid roots)

Now we are ready to apply this to the DWT:

decomposition $C_0(\xi) \to \overline{H(\xi)}\, C_1(\xi)$ (filtering with time-reversed h)

$$\to \tfrac{1}{2}\left[\overline{H(\tfrac{\xi}{2})}\, C_1(\tfrac{\xi}{2}) + \overline{H(\tfrac{\xi}{2}+\pi)}\, C_1(\tfrac{\xi}{2}+\pi)\right] = C_0(\xi)$$

$$C_1(\xi) \to \tfrac{1}{2}\left[\overline{G(\tfrac{\xi}{2})}\, C_1(\tfrac{\xi}{2}) + \overline{G(\tfrac{\xi}{2}+\pi)}\, C_1(\tfrac{\xi}{2}+\pi)\right] = D_0(\xi)$$

reconstruction

$C_0(\xi) \to C_0(2\xi)$ (upsampling)

$D_0(\xi) \to D_0(2\xi)$

$$C_0, D_0 \to H(\xi)\, C_0(2\xi) + G(\xi)\, D_0(2\xi) = C_1(\xi)$$

One more notation:

If $\{h_n\}$ is a filter, then

$$H(\xi) = \sum h_n e^{-in\xi}$$

is called its frequency response. From now on, I will be working instead with the symbol

$$h(\xi) = \frac{1}{\sqrt{2}} \sum_n h_n e^{-in\xi}$$

or

$$h(z) = \frac{1}{\sqrt{2}} \sum_n h_n z^{-n}.$$

Just like you always get a factor of 2π somewhere in the Fourier transform, you always get a factor of 2 somewhere in the DWT. This is where I choose to hide it.

Lecture 4. Coefficients and Functions

(Burrus et al., ch. 5)

Recall from lecture 3:

The two-scale recursion relations

$$\varphi(t) = \sqrt{2} \sum_k h_k \varphi(2t-k)$$

$$\psi(t) = \sqrt{2} \sum_k g_k \varphi(2t-k)$$

plus the orthogonality assumptions lead to a decomposition algorithm

$$c_{n-1,j} = \sum_k h_{k-2j} c_{nk}$$

$$d_{n-1,j} = \sum_k g_{k-2j} c_{nk}$$

and a reconstruction algorithm

$$c_{nk} = \sum_j h_{k-2j} c_{n-1,j} + \sum_j g_{k-2j} d_{n-1,j}$$

For the algorithm, we don't actually need to know φ, ψ (except to interpret the meaning of c_{nj}, d_{nj})

In this section, we consider the connections between h_k, g_k and φ, ψ. Specifically, we address three questions:

1. What properties of $\{h_k\}$, $\{g_k\}$ make the algorithm work?
2. Under what additional conditions on $\{h_k\}$, $\{g_k\}$ is there a function φ? When do φ, ψ generate a MRA?
3. How to find φ from $\{h_k\}$?

1. What makes the algorithm work?

In frequency domain notation, recall

$$C_0(\xi) = \frac{1}{\sqrt{2}}\left[\overline{h(\tfrac{\xi}{2})}\, C_1(\tfrac{\xi}{2}) + \overline{h(\tfrac{\xi}{2}+\pi)}\, C_1(\tfrac{\xi}{2}+\pi)\right]$$

$$D_0(\xi) = \frac{1}{\sqrt{2}}\left[\overline{g(\tfrac{\xi}{2})}\, C_1(\tfrac{\xi}{2}) + \overline{g(\tfrac{\xi}{2}+\pi)}\, C_1(\tfrac{\xi}{2}+\pi)\right]$$

$$C_1(\xi) = \sqrt{2}\left[h(\xi)\, C_0(2\xi) + g(\xi)\, D_0(2\xi)\right].$$

Putting these equations together, we get

$$C_1(\xi) = \left[|h(\xi)|^2 + |g(\xi)|^2\right] C_1(\xi) + \left[h(\xi)\overline{h(\xi+\pi)} + g(\xi)\overline{g(\xi+\pi)}\right] C_1(\xi+\pi)$$

So: the algorithm works if and only if

$$\begin{cases} |h(\xi)|^2 + |g(\xi)|^2 = 1 \\ h(\xi)\,\overline{h(\xi+\pi)} + g(\xi)\,\overline{g(\xi+\pi)} = 0 \end{cases} \qquad (0)$$

I will call this "Condition 0" (for "orthogonality").

Let $$M(\xi) = \begin{pmatrix} h(\xi) & h(\xi+\pi) \\ g(\xi) & g(\xi+\pi) \end{pmatrix}$$ "modulation matrix"

then (0) $\Leftrightarrow$ $M^* M = I$

(M^* = complex conjugate transpose

$$M^* = \begin{pmatrix} \overline{h(\xi)} & \overline{g(\xi)} \\ \overline{h(\xi+\pi)} & \overline{g(\xi+\pi)} \end{pmatrix}$$)

An M which satisfies $M^*M = I$ is called unitary.

The equivalent condition $M \cdot M^* = I$

leads to

$$\begin{cases} |h(\xi)|^2 + |h(\xi+\pi)|^2 = 1 \\ |g(\xi)|^2 + |g(\xi+\pi)|^2 = 1 \\ h(\xi)\overline{g(\xi)} + h(\xi+\pi)\overline{g(\xi+\pi)} = 0 \end{cases} \quad (0)$$

which is equivalent to

$$\begin{cases} \sum_j h_j h_{j+2k} = \sum_j g_j g_{j+2k} = \delta_{0k} \\ \sum_j h_j g_{j+2k} = 0 \end{cases} \quad (0)$$

We will call all of these "condition 0". They are just different ways to express the same condition.

You can also get another interesting fact out of condition 0:

$$g_n = \alpha \cdot (-1)^n \cdot h_{(2k+1)-n}, \qquad |\alpha| = 1$$

In words: the g-coefficients are the h-coefficients in reverse order, with alternating sign, and shifted by an odd number of places.

Sketch of proof:

If $A = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$ is any 2×2 matrix, then

$$A^{-1} = \frac{1}{ad-bc}\begin{pmatrix} d & -b \\ -c & a \end{pmatrix} \qquad (*)$$

if $ad-bc = \det(A)$ is nonzero.

Let $\mu(\xi) = \det M(\xi)$, then $|\mu(\xi)| = 1$ (true for any unitary matrix). Both μ and $\frac{1}{\mu}$ are trig polynomials

$\Rightarrow$ μ is monomial

$$\mu(\xi) = \alpha \cdot e^{-in\xi} \qquad \text{for some } n \in \mathbb{Z},\ |\alpha| = 1$$

Formula $(*)$, applied to $M^{-1} = M^*$, gives

$$\frac{1}{\mu(\xi)}\begin{pmatrix} g(\xi+\pi) & -h(\xi+\pi) \\ -g(\xi) & h(\xi) \end{pmatrix} = \begin{pmatrix} \overline{h(\xi)} & \overline{g(\xi)} \\ \overline{h(\xi+\pi)} & \overline{g(\xi+\pi)} \end{pmatrix}$$

The first column gives two formulas for g in terms of h.

Compare them $\Rightarrow$ $\mu(\xi+\pi) = -\mu(\xi)$

$\Rightarrow$ $n = 2k+1$ is odd

and

$$g(\xi) = -\mu(\xi)\overline{h(\xi+\pi)}$$

which leads to the stated formula.

2. When do φ, ψ exist?

If $\{g_k\}$ is finite, ψ is just a finite linear combination of φs.
It suffices to concentrate on φ.

We are not interested in pathological cases. φ should be an L^1-function, an L^2-function, or a tempered distribution: something that has a Fourier transform.

Theorem: $\hat{\varphi}(\xi) = h\left(\frac{\xi}{2}\right)\hat{\varphi}\left(\frac{\xi}{2}\right)$

proof: Take the Fourier transform of $\varphi(t) = \sqrt{2}\sum_k h_k \varphi(2t-k)$.

Theorem: If $\varphi \in L^1$, $\int \varphi(t)dt \neq 0$, then $\sum_k h_{2k} = \sum_k h_{2k+1} = \frac{1}{2}\sqrt{2}$. (M)

proof: You can integrate both sides of

$$\varphi(t) = \sqrt{2}\sum_k h_k \varphi(2t-k)$$

That will give you $\sum h_k = \sqrt{2}$, but not more.

But then, we use $|h(\xi)|^2 + |h(\xi+\pi)|^2 = 1$.

$$\sum h_k = \sqrt{2} \iff h(0) = 1 \implies h(\pi) = 0 \iff \sum h_{2k} = \sum h_{2k+1}.$$

Note: I will call this "Condition M" (for "moment").
It leads to $\sum g_{2k} = -\sum g_{2k+1} = \frac{1}{2}\sqrt{2}\cdot\alpha$, $|\alpha| = 1$

which leads to $\int \psi(t)dt = 0$ (the zeroth moment of ψ is 0)

Conditions O and M are necessary conditions: if they are violated, φ will certainly not exist.

Conditions O, M + decay condition guarantee the existence of φ.

Something like: $\sum |n^{1+\varepsilon} h_n| < \infty$

For finite length $\{h_k\}$, the decay condition is automatically satisfied.

To insure that φ, ψ generate MRA, we need another condition that says that $h(\xi)$ should stay away from 0:

$$h(\xi) \neq 0 \quad \text{except at} \quad \xi = \pi + 2\pi k$$

or

compact support + $h(\xi) \neq 0$ on $[-\frac{\pi}{3}, \frac{\pi}{3}]$

---

3. How to find φ from $\{h_k\}$?

We want to find $\varphi(t)$ which satisfies

$$\varphi(t) = \sqrt{2} \sum_k h_k \varphi(2t-k) \qquad \circledast$$

Observation: if φ satisfies $\circledast$, so does $\alpha\varphi$ for any α, so φ is only determined up to a multiple.

Usual normalization: $\|\varphi\|_2 = 1$

method 1: (not very practical, but has theoretical uses)

recall $$\hat{\varphi}(\xi) = h\left(\frac{\xi}{2}\right)\hat{\varphi}\left(\frac{\xi}{2}\right)$$
$$= h\left(\frac{\xi}{2}\right)h\left(\frac{\xi}{4}\right)\hat{\varphi}\left(\frac{\xi}{4}\right)$$
$$= \cdots = \left[\prod_{j=1}^{n} h\left(2^{-j}\xi\right)\right]\hat{\varphi}\left(2^{-n}\xi\right)$$

assume $\hat{\varphi}$ is continuous at 0, then

$$\hat{\varphi}(\xi) = \underbrace{\hat{\varphi}(0)}_{\text{constant}} \cdot \left[\prod_{j=1}^{\infty} h\left(2^{-j}\xi\right)\right]$$

method 2: (Cascade algorithm)

start with $\varphi^{(0)}(t)$ = arbitrary starting function with $\int \varphi^{(0)}(t)dt \neq 0$
(usually ⎍ (box) or ⩚ (triangle))

iterate: $$\varphi^{(n+1)}(t) = \sqrt{2}\sum_{k} h_k \varphi^{(n)}(2t-k)$$

then $\varphi^{(n)} \to \varphi$

method 3: (eigenvalue method)

Fact: (proved later) If coefficients are $h_0 .. h_n$, then φ is nonzero only on interval $[0, n]$.

Write out recursion relation for integers $0..n$:

$t=0$: $\quad \varphi(0) = \sqrt{2}\, h_0 \varphi(0)$

$t=1$: $\quad \varphi(1) = \sqrt{2}\left[h_0 \varphi(2) + h_1 \varphi(1) + h_2 \varphi(0)\right]$

$\vdots$

$t = n-1$: $\quad \varphi(n-1) = \sqrt{2}\left[h_{n-2} \varphi(n) + h_{n-1} \varphi(n-1) + h_n \varphi(n-2)\right]$

$t=n$ $\quad \varphi(n) = \sqrt{2}\, h_n \varphi(n)$

or

$$T \emptyset = \emptyset$$

where

$$T = \sqrt{2} \begin{pmatrix} h_0 & & & & 0 \\ h_2 & h_1 & h_0 & & \\ h_4 & h_3 & h_2 & h_1 & h_0 \\ & & & \ddots & \\ & & h_n & h_{n-1} & h_{n-2} \\ 0 & & & & h_n \end{pmatrix}$$

$$\emptyset = \begin{pmatrix} \varphi(0) \\ \varphi(1) \\ \vdots \\ \varphi(n) \end{pmatrix}$$

This is an eigenvalue problem

If $h_0 \neq \frac{1}{\sqrt{2}}$, $h_n \neq \frac{1}{\sqrt{2}}$ (usually the case), then we can make it shorter: $\varphi(0) = \varphi(n) = 0$

$$\sqrt{2}\begin{pmatrix} h_1 & h_0 & & 0 \\ h_3 & h_2 & h_1 & h_0 \\ & \ddots & & \\ 0 & & h_{n-1} & h_{n-2} \end{pmatrix}\begin{pmatrix} \varphi(1) \\ \vdots \\ \varphi(n-1) \end{pmatrix} = \begin{pmatrix} \varphi(1) \\ \vdots \\ \varphi(n-1) \end{pmatrix}$$

Once we have $\varphi(j)$, we get $\varphi(\frac{j}{2})$, $\varphi(\frac{j}{4})\ldots$ by applying recursion formula.

<u>advantage of this method:</u>

- gives exact function values
- easy to apply

<u>disadvantage</u>

- fails if there is a jump at an integer (example: Haar function, $T = I$)

<u>Remark:</u> Condition M implies that $(1,1\ldots 1)$ is left eigenvector with eigenvalue 1 $\Rightarrow$ right eigenvector exists.

Lemma $\sum_n \varphi(t-n) = \text{Constant}$

Sketch of proof: Obviously, $\emptyset(t) = \sum_n \varphi(t-n)$ is a 1-periodic function

restrict it to $[0,1]$, use recursion relations to find

$$\emptyset(t) = \emptyset(2t) + \emptyset(2t-1)$$

$\Rightarrow \emptyset$ is box function.

Theorem For orthogonal scaling functions, the normalizations

$$\sum \varphi(n) = 1$$

$$\int \varphi(t)dt = 1$$

$$\int |\varphi(t)|^2 dt = 1$$

are identical.

Sketch of proof: normalize φ so that $\sum_n \varphi(t-n) = 1$

Then obviously $\sum_n \varphi(n) = 1$

apply recursion relation

$$\sum_n \varphi(n) = \frac{1}{2} \sum_n \varphi\left(\frac{n}{2}\right) = \cdots = 2^{-j} \sum_n \varphi(2^{-j}n) \to \int \varphi(t)dt$$

trapezoidal rule with $h = 2^{-j}$

Finally,

$$1 = \int \varphi(t)dt = \langle \varphi, 1 \rangle = \langle \varphi, \sum_n \varphi(t-n) \rangle$$

$$= \int |\varphi(t)|^2 dt + \underbrace{\sum_{j \neq 0} \langle \varphi, \varphi(t-j) \rangle}_{=0 \text{ by orthogonality}}$$

Lecture 5. Biorthogonal Wavelets

Def: Two sets of vector $\{v_i\}$, $\{\tilde{v}_i\}$ are biorthogonal to each other, if

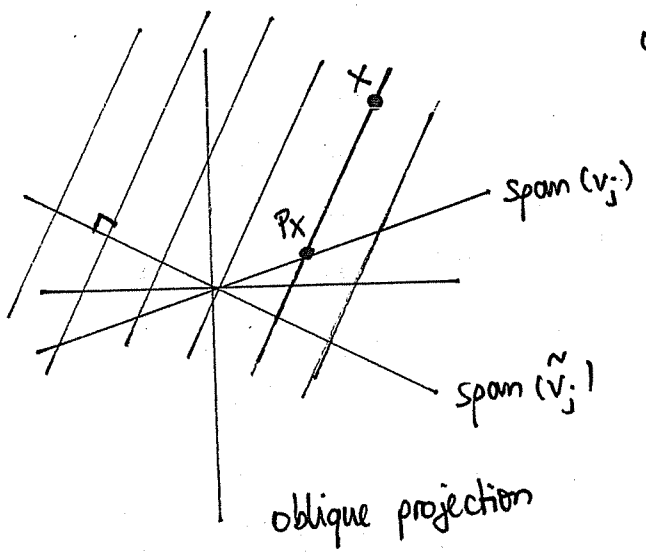
$$\langle v_i, \tilde{v}_j \rangle = \delta_{0j}$$

Examples:

$$v_1 = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \quad v_2 = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$$

$$\tilde{v}_1 = \begin{pmatrix} 2 \\ -1 \end{pmatrix} \quad \tilde{v}_2 = \begin{pmatrix} -1 \\ 1 \end{pmatrix}$$

Fact: $Px = \sum_j \langle x, \tilde{v}_j \rangle v_j$ is a projection of x onto span (v_j) but orthogonal to span $(\tilde{v}_j)$ instead of span (v_j)



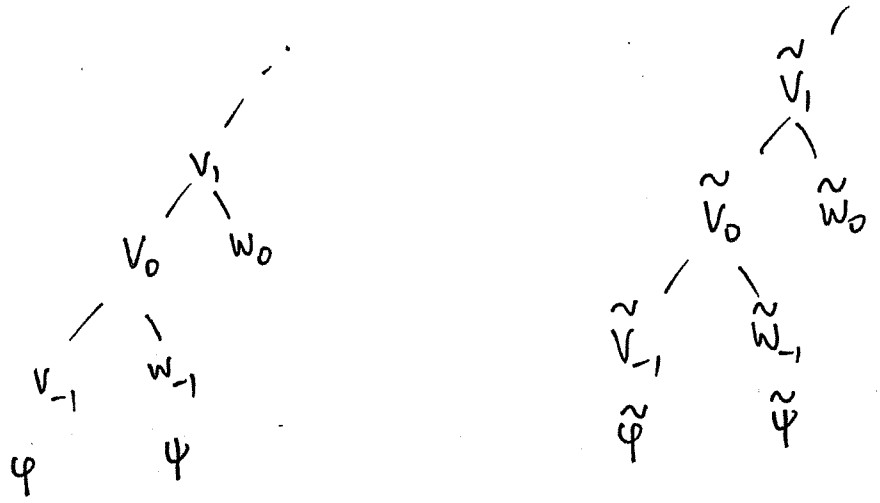
oblique projection

$\tilde{P}x = \sum_j \langle x, v_j \rangle \tilde{v}_j$ is projection of x onto span $(\tilde{v}_j)$, perpendicular to span (v_j)

Note: We are really only interested in real scaling functions and wavelets, but for this section only, I will pay attention to complex conjugation.

$$\langle x, y \rangle = \sum_i x_i \overline{y_i}$$

We assume the same MRA setup as before, except for orthogonality. Instead, we have <u>two</u> MRAs, with biorthogonal basis functions:



$$\langle \varphi_{jk}(t), \tilde{\varphi}_{lm}(t) \rangle = \delta_{jl}\,\delta_{km}$$

$$\langle \psi_{jk}, \tilde{\psi}_{lm} \rangle = \delta_{jl}\,\delta_{km}$$

$$\langle \varphi_{jk}, \tilde{\psi}_{lm} \rangle = 0$$

$$\varphi(t) = \sqrt{2} \sum_k h_k \varphi(2t-k)$$

$$\psi(t) = \sqrt{2} \sum_k g_k \varphi(2t-k)$$

$$\tilde{\varphi}(t) = \sqrt{2} \sum_k \tilde{h}_k \tilde{\varphi}(2t-k)$$

$$\tilde{\psi}(t) = \sqrt{2} \sum_k \tilde{g}_k \tilde{\varphi}(2t-k)$$

recursion relations

assume $f(t) = \sum_j c_{nj}\, \varphi_{nj}(t) \in V_n$

then

$$c_{n-1,k} = \sum_j \overline{\tilde{h}}_{j-2k}\, c_{nj}$$

$$d_{n-1,k} = \sum_j \overline{\tilde{g}}_{j-2k}\, c_{nj}$$

$$c_{nj} = \sum_k h_{j-2k}\, c_{n-1,k} + \sum_k g_{j-2k}\, d_{n-1,k}$$

original algorithm

φ, ψ = synthesis functions

$\tilde{\varphi}, \tilde{\psi}$ = analysis functions

or if we assume $f(t) = \sum_j c_{nj}\, \tilde{\varphi}_{nj}(t) \in \tilde{V}_n$

then

$$c_{n-1,k} = \sum_j \overline{h}_{j-2k}\, c_{nj}$$

$$d_{n-1,k} = \sum_j \overline{g}_{j-2k}\, c_{nj}$$

$$c_{nj} = \sum_k \tilde{h}_{j-2k}\, c_{n-1,k} + \sum_k \tilde{g}_{j-2k}\, d_{n-1,k}$$

dual algorithm

φ, ψ = analysis

$\tilde{\varphi}, \tilde{\psi}$ = synthesis

Everything involving biorthogonal wavelets is symmetric: to every formula or algorithm, there is a dual formula with h, g and $\tilde{h}, \tilde{g}$ interchanged.

What we usually want:

analysis functions: - good lowpass/bandpass cutoffs
- vanishing moments (for compression)

synthesis functions: - smoothness

} related

Condition D

$$M(\xi)\,\tilde{M}(\xi)^* = I$$

$$\Leftrightarrow \quad h(\xi)\overline{\tilde{h}(\xi)} + g(\xi)\overline{\tilde{g}(\xi)} = 1$$
$$h(\xi)\overline{\tilde{h}(\xi+\pi)} + g(\xi)\overline{\tilde{g}(\xi+\pi)} = 0$$

$$\Leftrightarrow \quad h(\xi)\overline{\tilde{h}(\xi)} + h(\xi+\pi)\overline{\tilde{h}(\xi+\pi)} = 1$$
$$g(\xi)\overline{\tilde{g}(\xi)} + g(\xi+\pi)\overline{\tilde{g}(\xi+\pi)} = 1$$
$$h(\xi)\overline{\tilde{g}(\xi)} + h(\xi+\pi)\overline{\tilde{g}(\xi+\pi)} = 0$$

$$\Leftrightarrow \quad \sum_j h_j \overline{\tilde{h}_{j+2k}} = \sum_j g_j \overline{\tilde{g}_{j+2k}} = \delta_{0k}$$
$$\sum_j h_j \overline{\tilde{g}_{j+2k}} = \sum_j \tilde{h}_j \overline{g_{j+2k}} = 0$$

which implies

$$g_n = \alpha \cdot (-1)^n \tilde{h}_{(2k+1)-n} \qquad |\alpha| \neq 0$$

$$\tilde{g}_n = \tilde{\alpha} \cdot (-1)^n h_{(2k+1)-n}$$

condition M same as before:

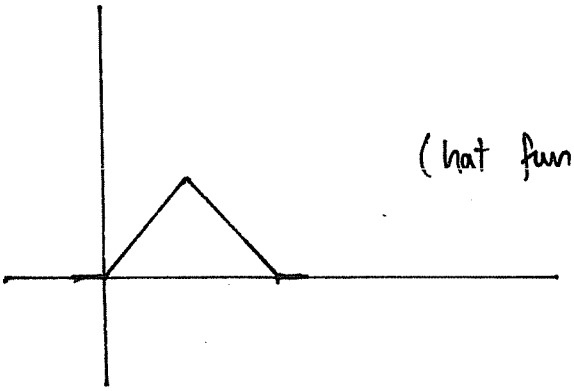
$$h(0) = 1,\ h(\pi) = 0 \Leftrightarrow \sum h_{2k} = \sum h_{2k+1} = \tfrac{1}{2}\sqrt{2}$$

$$\tilde{h}(0) = 1,\ \tilde{h}(\pi) = 0 \Leftrightarrow \sum \tilde{h}_{2k} = \sum \tilde{h}_{2k+1} = \tfrac{1}{2}\sqrt{2}$$

Example (bior 2.2 in Matlab wavelet toolbox)

h_k		1	2	1			$\cdot \sqrt{2}/4$
g_k		1	2	−6	2	1	$\cdot \sqrt{2}/8$
$\tilde{h}_k$	−1	2	6	2	−1		$\cdot \sqrt{2}/8$
$\tilde{g}_k$			1	−2	1		$\cdot \sqrt{2}/4$
k	−1	0	1	2	3	4	

φ =



(hat function)

Note: Except for the Haar wavelets, orthogonal wavelets cannot be symmetric.
Biorthogonal wavelets can.

Lecture 6. Intro to the Wavelet Toolbox

In this lecture, we will learn some of the basic commands for doing a wavelet decomposition / reconstruction

waveinfo

The waveinfo function lists all available wavelets.

1. gaus, morlet, mexihat } mostly useful for continuous wavelet transform
2. meyr

3. orthogonal wavelets
   - db_ (minimum phase Daubechies wavelets)
   - sym_ (most symmetric ------)
   - coif_ (Coiflets)

4. biorthogonal wavelets
   - bior_._ (Daubechies - Cohen spline wavelets)
   - rbio_._ (same, with ψ, $\tilde{\psi}$ interchanged)

wfilters

```
[lo_d, hi_d, lo_r, hi_r] = wfilters('wavelet')
```

$\tilde{h}_{-n}$ $\tilde{g}_{-n}$ h_n g_n

This routine gives you the filter coefficients for a wavelet.

wavefun

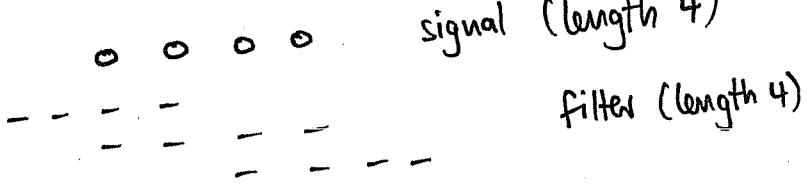
This routine produces point values of φ, ψ

$[\varphi, \psi, t] = \text{wavefun}('\text{wavelet}', n);$ (stepsize 2^{-n}, default 2^{-8})

$[\varphi, \psi, \tilde{\varphi}, \tilde{\psi}, t] = \text{wavefun}('\text{wavelet}', n);$

dwtmode

For a finite length signal (which includes all signals you can actually process), and any wavelet with more than 2 coefficients, there will be some samples that partially overlap the signal.

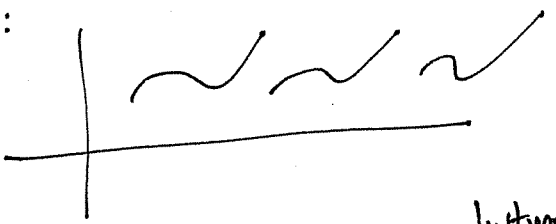


How do you handle that?

(a) methods that exactly cut the signal length in half

- special endpoint basis functions
  works, but is complicated and slows down the algorithm.
  Also, the special endpoint functions won't have the same properties as the rest (not in wavelet toolbox)

- periodization
  works, easy to implement, but introduces artificial jumps in signal:

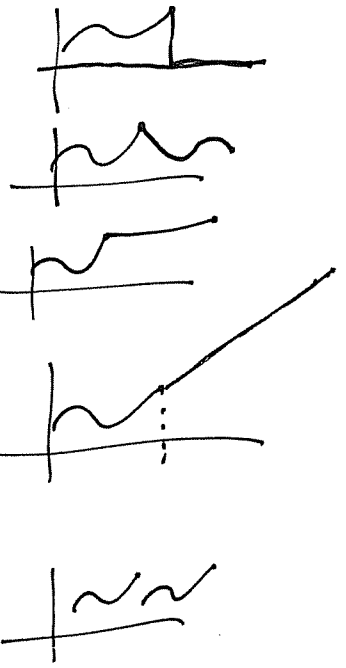


dwtmode('per')

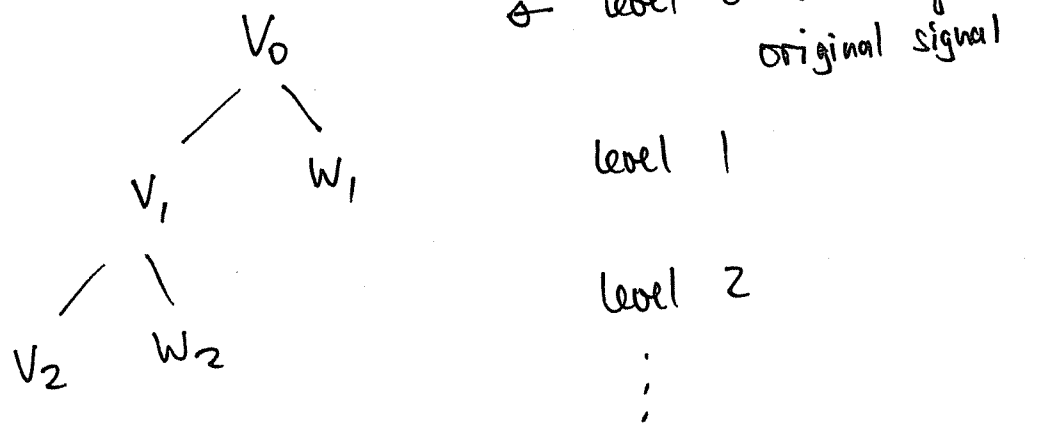
(b) methods that make the signal longer

Basically, you extend the signal in some fashion, just far enough to accommodate the filter.

- dwtmode('zpd') zero padding (default)
- dwtmode('sym') symmetric extension
- dwtmode('spd') constant padding
- dwtmode('sp1') = 'spd' straight line padding
- dwtmode('ppd') periodic padding



Now to the actual decomposition/reconstruction routines. the Matlab toolbox numbers the levels in the opposite order from what I have been using:



dwt one level of decomposition

[c1, d1] = dwt(s, 'wavelet')

or dwt(s, lo_d, hi_d)

in any of these routines, you can either specify the wavelet name, or give decomposition or reconstruction filters

idwt one level of reconstruction

s = idwt(c1, d1, 'wavelet')

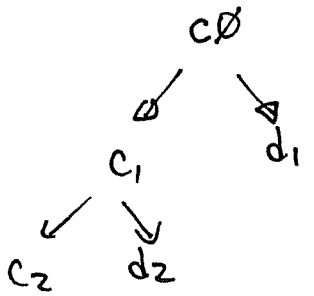
or idwt(c1, d1, lo_r, hi_r)

wavedec multiple levels of decomposition

[C, L] = wavedec(s, n, 'wavelet')

↑ number of levels

example:



C = [c2, d2, d1]

L = [length of c2, length of d2, length of d1, length of cØ]

wavedec multiple levels of reconstruction

```
s= wavedec (C,L,'wavelet')
```

detcoef extract detail coefficients

```
dn = detcoef (C,L,n)
```

appcoef extract approximation coefficients

```
an = appcoef (C,L,n)
                  'wavelet'
```

(may involve partial reconstruction)

wrcoef sum up series

```
F = wrcoef ('a' or 'd', C,L, 'wavelet', n)
```

if C,L is a decomposition, then

```
d2 = detcoef (C,L,2)
```

will produce the coefficients d_{2j}, while

```
F2 = wrcoef ('d', C,L, 'wavelet', 2)
```

will produce the function

$$Q_2 f = \sum d_{2,j} \psi_{2j}(t)$$

(sampled at the same rate as level 0).

In the end,

$$Q_1 f + Q_2 f + \dots + Q_n f + P_n f = f \quad \text{(original signal)}$$

upcoef F = upcoef ('a' or 'd', a or d, 'wavelet', n)

I think it does essentially the same as wrcoef, except the input is a single set of coefficients (a_{nj} or d_{nj}), instead of the C, L - structure

upwlev [Cnew, Lnew, An] = upwlev (C, L, 'wavelet')

turn a decomposition of depth n into one of depth (n-1).

Lecture 7: Moments and Regularity

Def: The continuous moments of φ, ψ are

$$m_k = \int t^k \varphi(t)\, dt \qquad k = 0, 1, 2 \ldots$$

$$n_k = \int t^k \psi(t)\, dt$$

The discrete moments of $\{h_k\}, \{g_k\}$ are

$$\mu_k = \frac{1}{\sqrt{2}} \sum_j j^k h_j \qquad k = 0, 1, 2 \ldots$$

$$\nu_k = \frac{1}{\sqrt{2}} \sum_j j^k g_j$$

φ is called k-regular if $h(z)$ has a zero of order k at π, that is

$$h(\pi) = h'(\pi) = \ldots = h^{(k-1)}(\pi) = 0$$

ψ has k vanishing moments if $m_j = 0$, $j = 0 \ldots k-1$.

Theorem The following are equivalent

(a) $\nu_j = 0$, $j = 0 \ldots k-1$

(b) $n_j = 0$, $j = 0 \ldots k-1$, (that is, ψ has k vanishing moments)

(c) φ is k-regular.

sketch of proof:

(a) $\Leftrightarrow$ (b) Substitute the recursion relations into

$$n_j = \int t^j \psi(t)\,dt$$

exercise!

and derive

$$n_j = 2^{-j} \sum_{l=0}^{j} \binom{j}{l} \nu_l \, m_{j-l} \qquad (*)$$

In matrix notation

$$\begin{pmatrix} n_0 \\ n_1 \\ \vdots \\ \vdots \end{pmatrix} = \begin{pmatrix} m_0 & & 0 \\ & \frac{1}{2}m_0 & \\ \times & & \frac{1}{4}m_0 \\ & & \ddots \end{pmatrix} \begin{pmatrix} \nu_0 \\ \nu_1 \\ \nu_2 \\ \vdots \end{pmatrix}$$

The matrix is nonsingular, since $m_0 \neq 0$.

Thus all $n_j = 0 \Leftrightarrow$ all $\nu_j = 0$

(a) $\Leftrightarrow$ (c) We observe

$$\nu_j = i^j D^j g(0) \qquad D = \text{derivative}$$

so $\nu_j = 0$, $j = 0 \ldots k-1$ $\Leftrightarrow$ $g(\xi)$ has a zero of order k at 0

From condition 0:

$$g(0) = 0 \Rightarrow |g(\pi)| = 1$$

Now differentiate the relation (also from condition 0)

$$h(\xi)\overline{g(\xi)} + h(\xi+\pi)\overline{g(\xi+\pi)} = 0$$

and evaluate at $\xi = 0$

$\Rightarrow$ $h(\xi)$ has zero of order k at π

For future reference

$$m_k = i^k D^k \hat{\varphi}(0) \qquad \mu_k = i^k D^k h(0)$$
$$n_k = i^k D^k \hat{\psi}(0) \qquad \nu_k = i^k D^k g(0)$$

Why are vanishing moments so interesting?

Reason 1: smoothness.

Regularity and smoothness are related:

φ k times differentiable $\Rightarrow$ φ is k-regular

φ k-regular $\Rightarrow$ φ is αk times differentiable for some α ($\alpha \approx 0.4$ for Daubechies wavelets)

Reason 2: compression

Theorem: φ is k-regular $\Leftrightarrow$ all polynomials up to degree $k-1$ are contained in V_0.

proof: Take a polynomial $p(t)$ of degree $\leq k-1$

$P_n p = P_{n-1} p + Q_{n-1} p$

$P_n p$: approximation at resolution 2^{-n}

but $Q_j p = 0$ for all j $\Rightarrow$ $P_n p$ is independent of n

but $P_n \to I$ as $n \to \infty$ $\Rightarrow$ $P_n p = p$ for all n

Assume that $f(t)$ is smooth for t near 0, and that ψ has k vanishing moments. Then

$$f(t) \approx f(0) + f'(0)t + \cdots + \frac{1}{(k-1)!} f^{(k-1)}(0)\, t^{k-1} + c \cdot t^k$$

fairly small constant

so $$\langle f, \psi_{j0} \rangle \approx c \cdot \langle t^k, \psi_{j0} \rangle = c \cdot n_j \cdot 2^{-(j+\frac{1}{2})k}$$

You get the same result for $\langle f, \psi_{jL} \rangle$, if f is smooth near $L \cdot 2^{-j}$.

<u>in words:</u> near a point where f is smooth, the detail coefficient d_{jL} at resolution 2^{-j} for a wavelet with k vanishing moments behaves like 2^{-jk}. This is useful for applications in data compression.

Scaling Function Coefficients versus Point Values

All wavelet algorithms begin with a signal at some fine resolution 2^{-n}:

$$\text{Signal} = P_n f(t) = \sum_j c_{nj}\, \varphi_{nj}(t) \in V_n$$

In real life, the signal most likely consists of equally spaced samples of f:

$$\text{Signal} = \{ f(n \cdot h) \} \qquad h = \text{stepsize}$$

So far, we have pretended not to notice that, but now is an appropriate time to consider this.

φ is only determined up to a multiple, so let's assume

$$m_0 = \int \varphi(t)\,dt = 1$$

(For orthogonal wavelets, that is the standard normalization, anyway)

Assume $m_1 = m_2 = \cdots = m_{k-1} = 0$, then by the same argument as above,

$$\langle f, \varphi_{jL} \rangle \approx f(2^{-j}L) + c \cdot 2^{-jk}$$

so the coefficients and the point values are virtually the same.

In general, let $m_1 = \tau$, and expand f around $2^{-j}(L+\tau)$ instead of $2^{-j}L$. It turns out that you then want

$$m_1 = \tau, \quad m_2 = \tau^2, \quad \cdots \quad m_{k-1} = \tau^{k-1}$$

to get $$\langle f, \varphi_{jL} \rangle \approx f(2^{-j}(L+\tau))$$

still an equally spaced sample.

<u>Surprise Bonus:</u> If ψ has k vanishing moments, $k \geq 2$, then $m_2 = m_1^2$.

The orthogonal Daubechies wavelets

Daubechies considered the following problem:

For a given length $n = 2L$, find recursion coefficients $h_0 \ldots h_{n-1}$ which produce orthogonal wavelets which are optimal in some sense.

choice 1: maximum number of vanishing moments of ψ

It turns out that with length $n = 2L$, we can achieve L vanishing moments.

For any L, there is a small number of distinct solutions; we can select one based on other criteria.

db- family: choose the one closest to linear phase
sym- : choose the most symmetric one
} same for $L = 1, 2, 3$

: choose smoothest one (not in toolbox)

choice 2: make vanishing moments for φ and ψ.

$$n_0 = n_1 = \ldots = n_{k-1} = 0 \qquad k \approx L$$

$$m_1 = 1, \quad m_2 = \ldots = m_{L-1} = 0$$

These are called coiflets (after Ravi Coifman, who requested them).

More generally, you could consider

$$m_1 = \tau, \quad m_2 = \tau^2 \ldots m_{L-1} = \tau^{L-1}$$

Lecture 8. The Daubechies Wavelets

We will now go over Ingrid Daubechies' construction of the shortest possible orthogonal wavelets with M vanishing moments.

Assume $\{h_k\} = \{h_0, h_1 \dots h_{2L-1}\}$

Then $\{g_k\} = \{-(-1)^k h_{2L-1-k}\}$ ... (see p. 4-3)

We need condition O:

$$|h(\xi)|^2 + |h(\xi+\pi)|^2 = 1$$

$$|g(\xi)|^2 + |g(\xi+\pi)|^2 = 1$$

$$h(\xi)\overline{g(\xi)} + h(\xi+\pi)\overline{g(\xi+\pi)} = 0$$

and condition M:

$$h(\pi) = 0$$

We also want M vanishing moments:

$$h(\pi) = h'(\pi) = \dots = h^{(M-1)}(\pi) = 0$$

which obviously includes condition M.

Using the above formula for g, you can check that the second and third parts of condition O are automatic.

Altogether, we need

$$\begin{cases} h(\xi) = \left(\dfrac{1+e^{-i\xi}}{2}\right)^M L(\xi), \quad L(0)=1 \\ |h(\xi)|^2 + |h(\xi+\pi)|^2 = 1 \end{cases}$$

Use $$\frac{1+e^{-i\zeta}}{2} = e^{-\frac{i\zeta}{2}} \frac{e^{\frac{i\zeta}{2}} + e^{-\frac{i\zeta}{2}}}{2} = e^{-\frac{i\zeta}{2}} \cos\frac{\zeta}{2}$$

to find $$|h(\zeta)|^2 = \left(\cos^2\frac{\zeta}{2}\right)^M |L(\zeta)|^2$$

We now require that the $\{h_k\}$ are all <u>real</u>.

If $$L(\zeta) = \sum l_j e^{-ij\zeta},$$

then $$|L(\zeta)|^2 = \sum l_j l_k e^{-i(j-k)\zeta} = \sum l_j l_k [\cos(j-k)\zeta - i \sin(j-k)\zeta]$$

When we group the jk and kj terms together, the sin terms drop out, so $$|L(\zeta)|^2 = \sum \lambda_n \cos n\zeta.$$

Using trig identities, we can convert that to

$$|L(\zeta)|^2 = \sum \mu_n \cos^n \zeta.$$

Using the identity

$$\cos\zeta = 1 - 2\sin^2\frac{\zeta}{2}$$

we can further write that as

$$|L(\zeta)|^2 = \sum \nu_n \left(\sin^2\frac{\zeta}{2}\right)^n = P\left(\sin^2\frac{\zeta}{2}\right).$$

Altogether, we then get

$$\left(\cos^2\frac{\zeta}{2}\right)^M P\left(\sin^2\frac{\zeta}{2}\right) + \left(\sin^2\frac{\zeta}{2}\right)^M P\left(\cos^2\frac{\zeta}{2}\right) = 1$$

or, if we let $z = \sin^2\frac{\zeta}{2}$,

$$\boxed{(1-z)^M P(z) + z^M P(1-z) = 1} \quad (*)$$

Theorem (Bezout)

For any $M \geq 1$, equation ⊛ has a unique solution of degree $(M-1)$. All other solutions have higher degrees.

The shortest solution can be found by a trick:

Solve ⊛ for $P(z)$:

$$P(z) = (1-z)^{-M} - (1-z)^{-M} z^M P(1-z)$$

If we expand $(1-z)^{-M}$ into a power series in z, then $P(z)$ and that series have to match up to the z^{M-1} term, since the second term above starts with a z^M power. Using the Bezout theorem, the shortest possible $P(z)$ = first M terms in the power series for $(1-z)^{-M}$:

$$P(z) = \sum_{k=0}^{M-1} \binom{-M}{k} (-z)^k \cdot 1^{M-1-k}$$

We note

$$\binom{-M}{k} \cdot (-1)^k = \frac{-M(-M-1)\cdots(-M-k+1)}{1 \cdot 2 \cdots k} \cdot (-1)^k = \frac{M(M+1)\cdots(M+k-1)}{1 \cdot 2 \cdots k}$$

$$= \binom{M+k-1}{k}$$

so

$$\boxed{P(z) = \sum_{k=0}^{M-1} \binom{M+k-1}{k} z^k}$$

M=1: $P(z) = \binom{0}{0} z^0 = 1$

M=2: $P(z) = \binom{1}{0} z^0 + \binom{2}{1} z^1 = 1 + 2z$

M=3: $P(z) = \binom{2}{0} z^0 + \binom{3}{1} z^1 + \binom{4}{2} z^2 = 1 + 3z + 6z^2$

Remark: Since $P(\sin^2\frac{\xi}{2}) = |L(\xi)|^2 \geq 0$, we must have $P(z) \geq 0$ for $z \in [0,1]$. The Bezout theorem cannot guarantee that, but fortunately that turns out to be true.

Now we need to find our way back to $L(\xi)$.
I will illustrate the procedure for $M=2$:

① Substitute $z = \sin^2\frac{\xi}{2} = \frac{1}{2}(1-\cos\xi)$

$$P(z) = 1 + 2z = 1 + 2\cdot\tfrac{1}{2}(1-\cos\xi) = 2 - \cos\xi$$
$$= -\tfrac{1}{2}e^{-i\xi} + 2 - \tfrac{1}{2}e^{i\xi} = |L(\xi)|^2$$

② A theorem of Riesz states that any trigonometric polynomial which is a polynomial in $\cos\xi$ only, and is ≥ 0, can be factored.
One way to do it goes as follows:
set $w = e^{i\xi}$, and pull out a power of w to make it into a standard polynomial:

$$P(z) = -\tfrac{1}{2}w^{-1} + 2 - \tfrac{1}{2}w = w^{-1}\underbrace{\left(-\tfrac{1}{2} + 2w - \tfrac{1}{2}w^2\right)}_{R(w)}$$

Theorem If w is a root of $R(w)$, so are $\frac{1}{w}$, $\overline{w}$, $\frac{1}{\overline{w}}$.
Thus, if a root is real or on the unit circle, it is part of a pair $(w, \frac{1}{w})$. If it is complex and not on the unit circle, it is part of a quadruple $(w, \frac{1}{w}, \overline{w}, \frac{1}{\overline{w}})$.

We can now select one member of each pair, and one pair $(w, \frac{1}{w})$ from each quadruple. Build the polynomial with exactly those roots:

$$L(\xi) = c \cdot (w - w_1)(w - w_2) \cdots (w - w_k)$$

and fix the constant c so that $L(0) = 1$. This will work.

Example: $R(w) = -\frac{1}{2} + 2w - \frac{1}{2}w^2$

has roots $2 \pm \sqrt{3}$. Choose $2 + \sqrt{3}$ at random,

then $L(\xi) = c \cdot (w - (2+\sqrt{3})) = c \cdot (e^{-i\xi} - (2+\sqrt{3}))$

$$L(0) = c \cdot (1 - (2+\sqrt{3})) = 1 \Rightarrow c = -\frac{1}{\sqrt{3}+1}$$

Altogether, $$h(\xi) = \left(\frac{1+e^{-i\xi}}{2}\right)^2 \left(-\frac{1}{\sqrt{3}+1}\right)\left(e^{-i\xi} - (2+\sqrt{3})\right)$$

After you sort everything out:

$$h_0 = (1-\sqrt{3})/4\sqrt{2}$$
$$h_1 = (3-\sqrt{3})/4\sqrt{2}$$
$$h_2 = (3+\sqrt{3})/4\sqrt{2}$$
$$h_3 = (1+\sqrt{3})/4\sqrt{2}$$

which is backwards from the usual ordering. If we had chosen $2-\sqrt{3}$, we would have found the usual Daubechies coefficients.

General rule: if you make one choice of roots, you get one scaling function; if you make exactly the opposite choice, you get the same function backwards.

For $M=2$, we have a pair of roots $\Rightarrow$ one choice (up to reflection)

$M=3$ quadruple $\Rightarrow$ one choice (up to reflection)

$M=4$ pair + quadruple $\Rightarrow$ 2 choices

$\vdots$

Daubechies' original choice: choose roots inside unit circle, in upper half plane (not in toolbox)

db_ in toolbox: choose the wavelet (among 2^K possibilities) which is closest to linear phase. This is done by brute force (generate all 2^K, then check which one is best), but only <u>once</u>.

sym_ in toolbox: choose the most symmetric one

Lecture 9. The Polyphase Representation

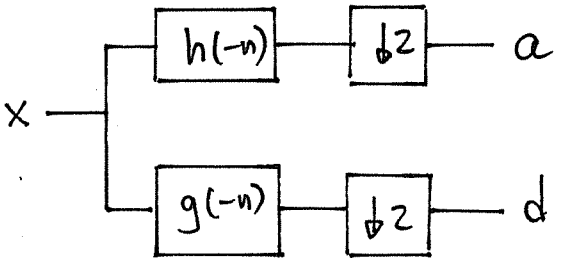
To make the notation easier, I will use

x = original signal

a = approximation coefficients

d = detail

DWT:



If we write this out in detail

$$\begin{array}{llllll} x_0 & x_1 & x_2 & x_3 & x_4 & \cdots \\ h_0 & h_1 & h_2 & h_3 & & \rightarrow a_{-1} \\ & h_0 & h_1 & h_2 & h_3 & \rightarrow a_0 \\ & & h_0 & h_1 & h_2 \cdots & \rightarrow a_1 \\ & & & \vdots & & \end{array}$$

we note that x_n with even n only multiply h_k with even k, and odd n only go with odd k.

This suggests a different implementation: split both x and the filters h, g into even and odd parts, called <u>phases</u>.

I will use the z-transform in this section, except I like positive powers of z, so $z = e^{-i\xi}$.

$$\{x_n\} \leftrightarrow \begin{aligned} x_0(z) &= \sum x_{2j} z^j \\ x_1(z) &= \sum x_{2j+1} z^j \end{aligned}$$

$$\{h_n\} \leftrightarrow \begin{aligned} h_0(z) &= \sum h_{2j} z^j \\ h_1(z) &= \sum h_{2j+1} z^j \end{aligned}$$

"polyphase symbols"

Note: The symbol was $h(z) = \frac{1}{\sqrt{2}} \sum h_j z^j$. The polyphase symbols do not get a factor of $1/\sqrt{2}$.

Verify:
$$\begin{cases} a(z) = h_0\left(\frac{1}{z}\right) x_0(z) + h_1\left(\frac{1}{z}\right) x_1(z) \\ d(z) = g_0\left(\frac{1}{z}\right) x_0(z) + g_1\left(\frac{1}{z}\right) x_1(z) \end{cases}$$

(the $\frac{1}{z}$ instead of z produces the reversal of coefficients)

conversely:
$$\begin{cases} x_0(z) = h_0(z)\, a(z) + g_0(z)\, d(z) \\ x_1(z) = h_1(z)\, a(z) + g_1(z)\, d(z) \end{cases}$$

or
$$\begin{pmatrix} a \\ d \end{pmatrix} = \begin{pmatrix} h_0\left(\frac{1}{z}\right) & h_1\left(\frac{1}{z}\right) \\ g_0\left(\frac{1}{z}\right) & g_1\left(\frac{1}{z}\right) \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \end{pmatrix}$$

$$\begin{pmatrix} x_0 \\ x_1 \end{pmatrix} = \begin{pmatrix} h_0 & g_0 \\ h_1 & g_1 \end{pmatrix} \begin{pmatrix} a \\ d \end{pmatrix}$$

Def: If $M(z)$ is a matrix polynomial, then $M^*(z)$ $= \left[M\left(\frac{1}{z}\right)\right]^T$ (or Hermitian transpose, if M is complex).

Def: $P(z) = \begin{pmatrix} h_0(z) & h_1(z) \\ g_0(z) & g_1(z) \end{pmatrix}$ polyphase matrix

Then

condition O = perfect reconstruction condition

$\Leftrightarrow \quad P(z)\, P^*(z) = I$

condition M $\Leftrightarrow \quad h_0(1) = h_1(1) = \frac{1}{2}\sqrt{2}$

Def: A matrix $M(z)$ with $M \cdot M^* = I$ is called paraunitary. For $z = e^{-i\xi}$ on the unit circle, it is unitary.

Compare $P(z)$ to the modulation matrix

$$M(z) = \begin{pmatrix} h(z) & h(-z) \\ g(z) & g(-z) \end{pmatrix}$$

① If $\{h_k\} = \{h_0 \dots h_{2L-1}\}$, then M has degree $2L-1$, P has degree $L-1$.

② M is redundant: the second column can be determined from the first. P is not redundant.

③ Polyphase matrices can be multiplied: if $P_1 P_1^* = I$, $P_2 P_2^* = I$, then $(P_1 P_2)(P_1 P_2)^* = I$.

If you multiply two modulation matrices, you don't get the correct structure.

Summary: polyphase matrices have lower degrees, and can be produced from smaller building blocks.

Remark: when I say "polynomial", I really mean "Laurent polynomial". That means that both positive and negative powers are allowed.

Condition 0 for biorthogonal wavelets is

$$P(z)\ \tilde{P}^*(z) = I$$

Theorem: If all filters have finite length, then

$$\det P(z) = c \cdot z^d$$

proof: $\det P(z)$ is a scalar polynomial, and so is $\det \tilde{P}^*(z) = \frac{1}{\det P(z)}$.

Def: A matrix polynomial $M(z)$ is called unimodular (in the narrow sense) if $\det M(z) = 1$. In a wider sense, we call it unimodular if $\det M(z) = c \cdot z^d$.

Summary: A polyphase matrix $P(z)$ produces finite-length filterbanks if and only if it is unimodular. It produces orthogonal filterbanks if and only if it is paraunitary.

Example: $h_0 .. h_3$ = Daubechies wavelet with two vanishing moments.

$$P(z) = \begin{pmatrix} h_0 + h_2 z & h_1 + h_3 z \\ g_0 + g_2 z & g_1 + g_3 z \end{pmatrix}$$

$$\det P(z) = -z$$

Def: The power d in $\det P(z) = c \cdot z^d$ is called the McMillan degree of $P(z)$.

Examples of unimodular matrices

① unit triangular: $L(z) = \begin{pmatrix} 1 & p(z) \\ 0 & 1 \end{pmatrix}$ or $\begin{pmatrix} 1 & 0 \\ p(z) & 1 \end{pmatrix}$

② nilpotent factors: $N(z) = I + Az$, $A^d = 0$ for some d

$$(I+Az)^{-1} = I - Az + A^2z^2 - + \cdots \pm A^d z^d$$

③ projection factors: Q = projection, that is $Q^2 = Q$

$$F(z) = (I-Q) + Qz$$

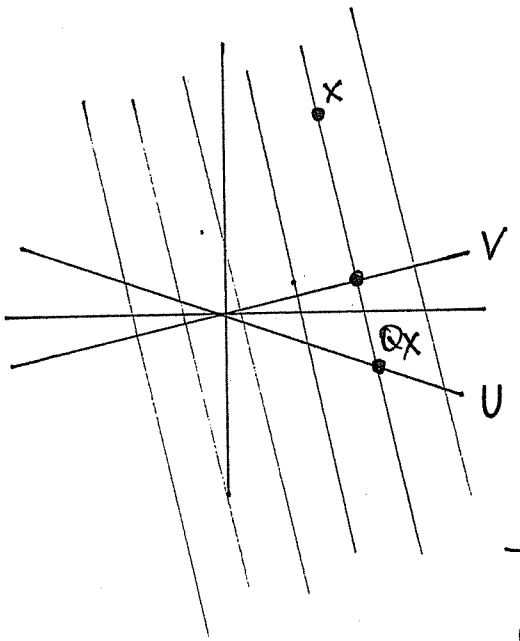
$$F^{-1}(z) = Qz^{-1} + (I-Q)$$

If $\tilde{F} = (I-Q^T) + Q^T z$, then $F \cdot \tilde{F}^* = I$

If Q is orthogonal (i.e. $Q^T = Q$) then F is paraunitary.

---

sideline on projections:



U, V = subspaces of same dimension

Q = projection onto U, perpendicular to V

Fact: Let $\{u_1 .. u_k\}$ be a basis of U, $\{v_1 .. v_k\}$ basis of V

U = matrix with columns $u_1 .. u_k$

V — — — $v_1 .. v_k$

Then

$$Q = U(V^TU)^{-1}V^T$$

Def: rank of Q = dimension of U (and V)

Special case: $U = V$, $\{u_1 .. u_k\}$ orthonormal basis

Then $Q = UU^T$, and $Q^T = Q$

Theorem (independently discovered many times)

The polyphase matrix of any orthogonal wavelet can be factored into

$$P(z) = H \cdot F_1(z) F_2(z) .. F_n(z)$$

where H = constant orthogonal matrix, $F_j(z)$ = (orthogonal) projection factors.

method 1 (Vaidyanathan)

use only projection factors of rank 1,

that is $Q = v \cdot v^T$, v = unit vector

In this case, n = number of factors = Mc Millan degree.

method 2

allow higher rank projection factors

In this case, n = degree of $P(z)$ (as polynomial)

Note: if $P(z)$ is 2×2 matrix, the only nontrivial projections have rank 1, so the two methods are the same.

Def: If $P(z) = \sum P_k z^k$ is any polynomial (scalar or matrix polynomial, standard polynomial or Laurent, ...), the j^{th} moment of P is

$$\sum_k k^j P_k$$

The zeroth moment is $\sum_k P_k = P(1)$.

In particular, the polyphase moments are the moments of the polyphase symbols:

$$h_0(z) = \sum h_{2k} z^k \qquad m_{j,0} = \sum k^j h_{2k}$$

$$h_1(z) = \sum h_{2k+1} z^k \qquad m_{j,1} = \sum k^j h_{2k+1}$$

Alternative Derivation of Daubechies wavelets

Let $P(z)$ = polyphase matrix of Daubechies wavelets with 2 vanishing moments.

By theorem,

$$P(z) = H \cdot F(z),$$

where H = constant orthogonal matrix

$$F(z) = (I - Q) + Qz,$$

$$Q = UU^T, \quad U = \text{unit vector}$$

so $U = \begin{pmatrix} \cos\alpha \\ \sin\alpha \end{pmatrix}$ for some α

condition 0 is automatic

condition M was $h_0(1) = m_{0,0} = \frac{1}{2}\sqrt{2}$

$h_1(1) = m_{0,1} = \frac{1}{2}\sqrt{2}$

but $P(1) = H \cdot F(1) = H$, since $F(1) = I$

$$\Rightarrow \quad H = \begin{pmatrix} \frac{1}{2}\sqrt{2} & \frac{1}{2}\sqrt{2} \\ * & * \end{pmatrix}$$ ← irrelevant

affects only wavelet, not scaling function

<u>So</u>: we have a one-parameter family of orthogonal wavelets of length 4 with one vanishing moment.
The second vanishing moment will select for a particular α.

The one drawback of working with polyphase matrices is that the vanishing moment conditions get complicated.

We'll do that next, for now I'll just quote the condition:

The second moment vanishes if

$$m_{1,0} - m_{1,1} = \frac{1}{2\sqrt{2}}$$

Now, $U = \begin{pmatrix} \cos\alpha \\ \sin\alpha \end{pmatrix} = \begin{pmatrix} c \\ s \end{pmatrix}$ $\qquad Q = UU^T = \begin{pmatrix} c^2 & cs \\ cs & s^2 \end{pmatrix}$

$$F = (I - Q) + Qz = \left[\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} - \begin{pmatrix} c^2 & cs \\ cs & s^2 \end{pmatrix}\right] + \begin{pmatrix} c^2 & cs \\ cs & s^2 \end{pmatrix} z$$

$$= \begin{pmatrix} s^2 & -cs \\ -cs & c^2 \end{pmatrix} + \begin{pmatrix} c^2 & cs \\ cs & s^2 \end{pmatrix} z$$

$$P = H \cdot F = \frac{1}{\sqrt{2}} \begin{pmatrix} s^2 - cs + (c^2 + cs)z & -cs + c^2 + (cs + s^2)z \\ * & * \end{pmatrix}$$

$$m_{00} = \frac{1}{\sqrt{2}}\left[(s^2 - cs) + (c^2 + cs)\right] = \frac{s^2+c^2}{\sqrt{2}} = \frac{1}{\sqrt{2}}, \text{ like it should}$$

$$m_{1,0} = \frac{1}{\sqrt{2}}\left[0\cdot(s^2-cs) + 1\cdot(c^2+cs)\right] = \frac{c^2+cs}{\sqrt{2}}$$

$$m_{0,1} = \frac{1}{\sqrt{2}}$$

$$m_{1,1} = \frac{1}{\sqrt{2}}\left[0\cdot(-cs+c^2) + 1\cdot(cs+s^2)\right] = \frac{cs+s^2}{\sqrt{2}}$$

$$m_{1,0} - m_{1,1} = \frac{c^2-s^2}{\sqrt{2}} = \frac{1}{2\sqrt{2}}$$

$$\Rightarrow \quad c^2 - s^2 = \frac{1}{2}$$

$$\cos^2\alpha - \sin^2\alpha = \cos 2\alpha = \frac{1}{2}$$

$$\Rightarrow 2\alpha = \frac{\pi}{3}, \quad \alpha = \frac{\pi}{6} \Rightarrow \begin{matrix}\cos\alpha = \frac{1}{2}\sqrt{3} \\ \sin\alpha = \frac{1}{2}\end{matrix}$$

so $$P(z) = \begin{pmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ * & * \end{pmatrix} \cdot \left[\begin{pmatrix} \frac{1}{4} & -\frac{\sqrt{3}}{4} \\ -\frac{\sqrt{3}}{4} & \frac{3}{4} \end{pmatrix} + \begin{pmatrix} \frac{3}{4} & \frac{\sqrt{3}}{4} \\ \frac{\sqrt{3}}{4} & \frac{1}{4} \end{pmatrix} z \right]$$

$$= \begin{pmatrix} \frac{1-\sqrt{3}}{4\sqrt{2}} + \frac{3+\sqrt{3}}{4\sqrt{2}} z, & \frac{3+\sqrt{3}}{4\sqrt{2}} + \frac{1+\sqrt{3}}{4\sqrt{2}} z \\ * & * \end{pmatrix}$$

That's the Daubechies wavelet in reverse.
We should have chosen $2\alpha = -\frac{\pi}{3}$ instead.

The moment conditions for the polyphase matrix (very brief summary)

Assume ψ has p vanishing moments. This means that x^j, $j=0..p-1$ can be represented exactly by translates of φ:

$$x^j = \sum_k y_k^{(j)} \varphi(t-k)$$

$\leftarrow$ suitable coefficients

It suffices to know $y_0^{(j)}$, since other $y_k^{(j)}$ can be computed by

$$y_k^{(j)} = \sum_{l=0}^{j} \binom{j}{l} k^{j-l} y_0^{(l)}$$

In particular $y_1^{(0)} = y_0^{(0)}$

$y_1^{(1)} = y_0^{(0)} + y_0^{(1)}$

$y_1^{(2)} = y_0^{(0)} + 2y_0^{(1)} + y_0^{(2)}$

$\vdots$

$y_0^{(0)}$ depends on the scaling of φ, so we can assume $y_0^{(0)} = 1$.

Vanishing Moment Condition

$$\boxed{\sum_{s=0}^{j} \binom{j}{s} (-1)^{j-s} m_{j-s,\mu} y_0^{(s)} = 2^{-j-\frac{1}{2}} y_\mu^{(j)}} \qquad \mu = 0,1 \quad j = 0..p-1$$

special cases: $j=0$

$$m_{0,0} y_0^{(0)} = \frac{1}{\sqrt{2}} y_0^{(0)}$$

$$m_{0,1} y_0^{(0)} = \frac{1}{\sqrt{2}} y_1^{(0)} = \frac{1}{\sqrt{2}} y_0^{(0)}$$

$$\Rightarrow \boxed{m_{0,0} = m_{0,1} = \frac{1}{\sqrt{2}}}$$

$j=1$:

$$m_{0,0} y_0^{(1)} - m_{1,0} y_0^{(0)} = 2^{-\frac{3}{2}} y_0^{(1)}$$

$$m_{0,1} y_0^{(1)} - m_{1,1} y_0^{(0)} = 2^{-\frac{3}{2}} y_1^{(1)} = 2^{-\frac{3}{2}} \left[y_0^{(0)} + y_0^{(1)} \right]$$

subtract, use $y_0^{(0)} = 1$:

$$\boxed{m_{1,0} - m_{1,1} = \frac{1}{2\sqrt{2}}}$$

$j=2$: (assuming I didn't make a mistake)

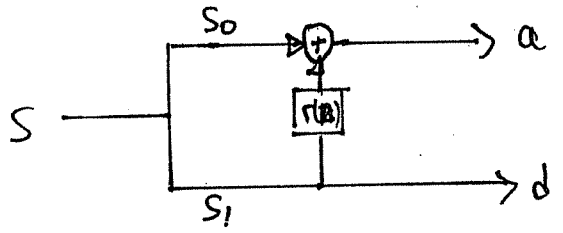
$$\boxed{m_{2,0} - m_{2,1} = m_{1,0} - 2^{-3/2}}$$

Lecture 10. Lifting

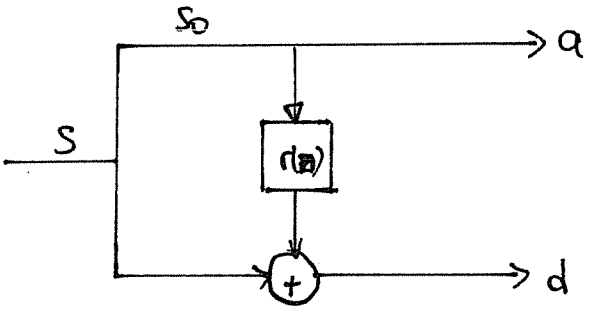
"Lifting" is the term Wim Sweldens coined to describe the factorization of a wavelet polyphase matrix into unit triangular steps.

Let's look at a single such step: $P(z) = \begin{pmatrix} 1 & r(z) \\ 0 & 1 \end{pmatrix}$

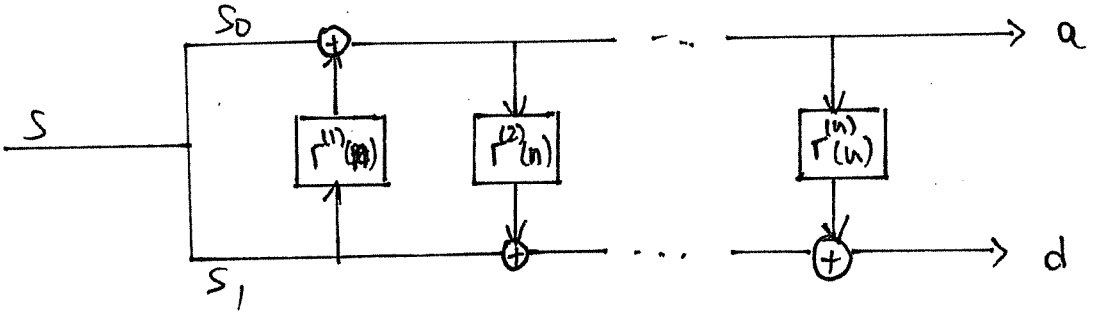
$$\begin{pmatrix} a(z) \\ d(z) \end{pmatrix} = \begin{pmatrix} 1 & r(z) \\ 0 & 1 \end{pmatrix} \begin{pmatrix} s_0(z) \\ s_1(z) \end{pmatrix}$$



or $P(z) = \begin{pmatrix} 1 & 0 \\ r(z) & 1 \end{pmatrix}$



Altogether $P(z) = \begin{pmatrix} 1 & 0 \\ r^{(N)}(z) & 1 \end{pmatrix} \cdots \begin{pmatrix} 1 & 0 \\ r^{(2)}(z) & 1 \end{pmatrix} \begin{pmatrix} 1 & r^{(1)}(z) \\ 0 & 1 \end{pmatrix}$



Factoring polynomials

example: $(x-1)\overline{)\,2x^2-3x+2}$ quotient $(2x-1)$

$2x^2-2x$

$-x+2$

$-x+1$

1 remainder

$$\frac{2x^2-3x+2}{x-1} = 2x-1 + \frac{1}{x-1}$$

or $$(2x^2-3x+2) = (2x-1)(x-1)+1$$

In general $\frac{p(x)}{q(x)} = r(x) + \frac{s(x)}{q(x)}$ $\quad r$ = quotient, s = remainder

notation: ∂p = degree of polynomial p

Then $\boxed{\partial s < \partial q}$

The factoring algorithm

concentrate on the top row of $P(z)$, that is $[h_0(z) \;\; h_1(z)]$

Assume $\partial h_1 \leq \partial h_0$. Let $h_0^{(0)} = h_0$, $h_1^{(0)} = h_1$.

Divide h_0 by h_1:

$$h_0^{(0)} = h_1^{(0)} r^{(0)} + h_0^{(1)}$$

with $\partial h_0^{(1)} < \partial h_1^{(0)}$

$$[h_0^{(0)}, h_1^{(0)}] = [h_0^{(1)}, h_1^{(0)}] \begin{pmatrix} 1 & 0 \\ r^{(0)} & 1 \end{pmatrix}$$

then divide $h_1^{(0)}$ by $h_0^{(1)}$:

$$h_1^{(0)} = h_0^{(1)} r^{(1)} + h_1^{(1)} \qquad \partial h_1^{(1)} < \partial h_0^{(1)} < \partial h_1^{(0)} \leq \partial h_0^{(0)}$$

$$[h_0^{(0)}\ h_1^{(0)}] = [h_0^{(1)}\ h_1^{(1)}] \begin{pmatrix} 1 & r^{(1)} \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ r^{(0)} & 1 \end{pmatrix}$$

continue until you reach degree 0. In the end

$$[h_0^{(0)}\ h_1^{(0)}] = \left([\alpha\ 0] \text{ or } [0\ \alpha] \right) \cdot (\text{upper and lower unit triangular terms})$$

α = const.

If you add the second row again:

$$P(z) = \begin{pmatrix} \alpha & 0 \\ \beta & \gamma \end{pmatrix} \cdot (\text{upper and lower unit triangular terms})$$

γ = monomial

we could factor $\begin{pmatrix} \alpha & 0 \\ \beta & \gamma \end{pmatrix} = \begin{pmatrix} \alpha & 0 \\ 0 & \gamma \end{pmatrix} \begin{pmatrix} 1 & 0 \\ \beta/\gamma & 1 \end{pmatrix}$

Application 1 speedup

for Daubechies wavelet with 2 vanishing moments,

$$P(z) = \begin{pmatrix} \frac{1+\sqrt{3}}{4\sqrt{2}} + \frac{3-\sqrt{3}}{4\sqrt{2}} z & \frac{3+\sqrt{3}}{4\sqrt{2}} + \frac{1-\sqrt{3}}{4\sqrt{2}} z \\ \frac{1-\sqrt{3}}{4\sqrt{2}} + \frac{3+\sqrt{3}}{4\sqrt{2}} z & -\frac{3-\sqrt{3}}{4\sqrt{2}} - \frac{1+\sqrt{3}}{4\sqrt{2}} z \end{pmatrix}$$

To calculate $\begin{pmatrix} a \\ d \end{pmatrix} = P \begin{pmatrix} s_0 \\ s_1 \end{pmatrix}$, this requires 8 multiplications and 6 additions per entry in $\begin{pmatrix} s_0 \\ s_1 \end{pmatrix}$.

In factored form:

$$P(z) = \begin{pmatrix} \frac{1+\sqrt{3}}{\sqrt{2}} & 0 \\ 0 & \frac{1-\sqrt{3}}{\sqrt{2}} \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 1 & z \end{pmatrix} \begin{pmatrix} 1 & \frac{\sqrt{3}}{4} + \frac{\sqrt{3}-2}{4} z \\ & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ -\sqrt{3} & 1 \end{pmatrix}$$

this takes 5 multiplications and 4 additions.
Sweldens showed that asymptotically (for very long filters) you can expect a speedup of 50%.

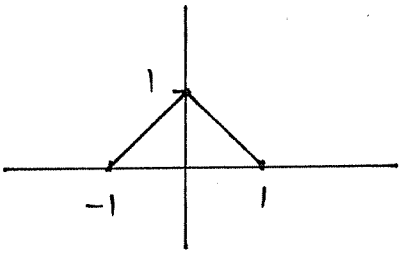
Application 2: Completion

Suppose you have φ, and φ is not orthogonal. How do you find ψ, $\tilde{\varphi}$, $\tilde{\psi}$?

example: the hat function is refinable:

$$\{h_{-1}, h_0, h_1\} = \frac{1}{2\sqrt{2}}(1, 2, 1)$$

φ



$$P(z) = \frac{1}{2\sqrt{2}} \begin{pmatrix} 2 & \frac{1}{z}+1 \\ z & z \end{pmatrix} = \frac{1}{2\sqrt{2}} \begin{pmatrix} 2 & 0 \\ \beta & \gamma \end{pmatrix} \begin{pmatrix} 1 & \frac{1}{2z} + \frac{1}{2} \\ 0 & 1 \end{pmatrix}$$

β unknown

γ = monomial
β = arbitrary

Lecture 11: Denoising and Compression

The basic idea is the same for both: Given a signal

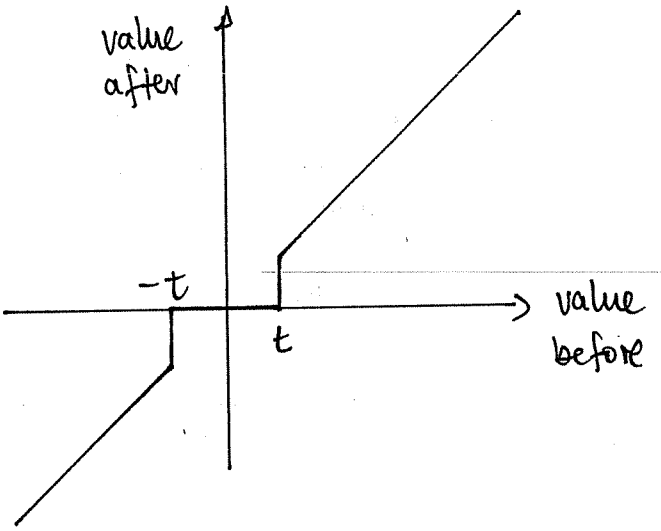
1. take DWT
2. get rid of some of the coefficients, especially small detail coefficients at fine resolution
3. reconstruct the signal from the remaining coefficients

Thresholding Let t = threshold value. There are two ways of getting rid of values smaller than t.

(1) hard thresholding

values below t (in absolute value) are set to 0.

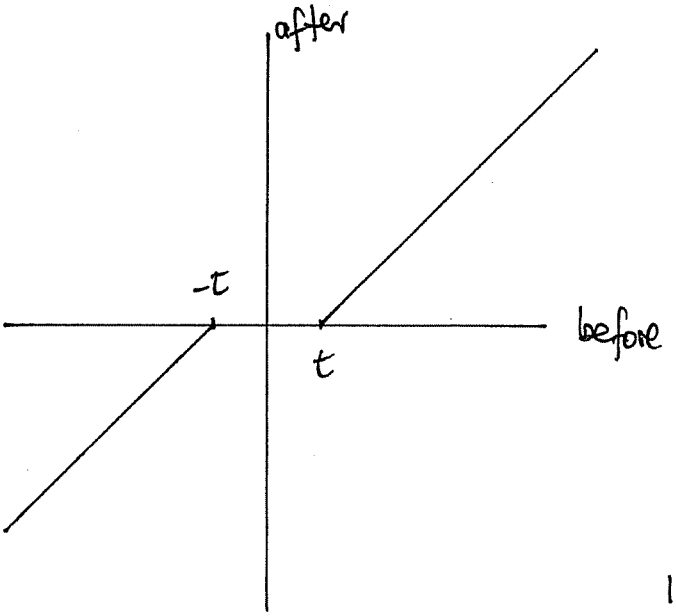
values above t unchanged.



(2) soft thresholding

values below t are set to zero.

values above t are decreased by t



Denoising:

Donoho investigated the following setup:

$\{x_k\}$ = "true" signal

$\{n_k\}$ = random signal (independent normal distributions, mean 0, standard deviation 1)

$$\{y_k\} = \{x_k + \varepsilon n_k\} = \text{observed signal}$$

Facts: ① The optimal threshold is $t = \varepsilon$, in the following sense:

Fix ε (= noise level). Let $x_k(t)$ = de-noised signal using (hard) threshold t. Let

$$R(t) = \| x_k(t) - x_k \|_2^2$$

This is a random variable, since $\{y_k\}$ is random.

Let error $(t) = E(R(t))$ (expected value).

Then error(t) is a minimum if $t = \varepsilon$.

② Donoho has a method of estimating t which is almost optimal. (The error using this estimate, and the smallest error using $t = \varepsilon$, are not more than a factor of $\log N$ apart, N = signal length).

③ Hard thresholding results in a smaller error in the L_2-sense. Soft thresholding results in a smoother, better-looking signal.

This denoising procedure can be improved in another way:

If you shift the signal by 1, all wavelet coefficients change

- — — — — — — 2, the coefficients at the next level shift by 1, then they change

You could take the signal and denoise it
shift the signal by 1, denoise again
- — — — 2 - — —

---

average over all shifts.

This works, but requires $O(N^2)$ operations, N = signal length.

You can obtain the same effect from denoising the

shift-invariant discrete wavelet transform.

The SIDWT

Start with an original signal s

s_0 s_1 s_2 s_3 s_4 ...

convolve (filter) with $h(-n)$:

$(sh)_0$ $(sh)_1$ $(sh)_2$ $(sh)_3$ ...

The approximation coefficients at the next level are found by downsampling $\{sh\}$ by 2:

$\{a_0, a_1, a_2 ..\} = \{(sh)_0, (sh)_2, (sh)_4 ..\}$

If we shift the signal by 1, we get

$\{a_0, a_1, a_2 \ldots\} = \{(sh)_1, (sh)_3, (sh)_5 \ldots\}$ (new numbers)

If we shift the signal by 2, we get

$\{a_0, a_1, a_2 \ldots\} = \{(sh)_2, (sh)_4, (sh)_6 \ldots\}$

= same coefficients as for original signal, but shifted by 1.

If we don't downsample (sh), we retain information about the coefficients for all shifts of s.
We could reconstruct $\{s\}$ from $\{sh\}$ by reconstructing from even and odd phases separately and averaging.

Now take (sh) again (without downsampling), and filter with $h(-n)$ again, with added zeros:

$$\{sh^2\} = sh * \{h_0, 0, h_1, 0, h_2, 0, \ldots\}$$

we get

$(sh^2)_0 \quad (sh^2)_1 \quad (sh^2)_2 \quad (sh^2)_3 \quad (sh^2)_4 \ldots$

$\{(sh^2)_0 \quad 4 \quad 8 \ldots\}$ = second level of coefficients for original signal

$\{(sh^2)_1 \quad 5 \quad 9 \ldots\}$ = . . . for signal shifted by 1

$\{(sh^2)_2 \quad 6 \quad 10 \ldots\}$ = . . . shifted by 2

This is the SIDWT: don't downsample, use spacing 2^k between filter coefficients at step k. It contains information on the wavelet coefficients for all possible shifts of the signal, but contains only $(N \log N)$ coefficients instead of N^2.

You can then denoise this transform and reconstruct. This has (at least approximately) the same effect as denoising all shifts of a signal, but is more efficient.

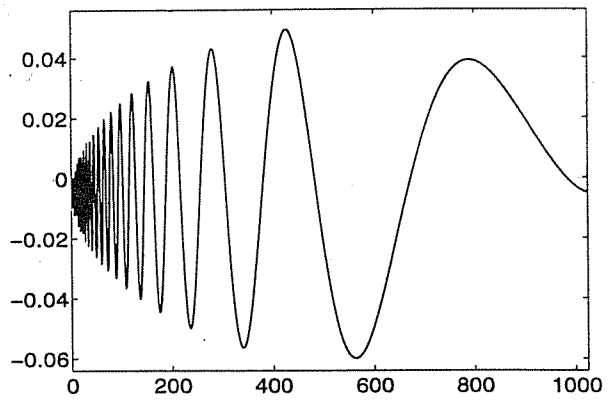
Observation: This works better in practice than just denoising the signal itself.

Note: The SIDWT is not in the wavelet toolbox, as far as I know.

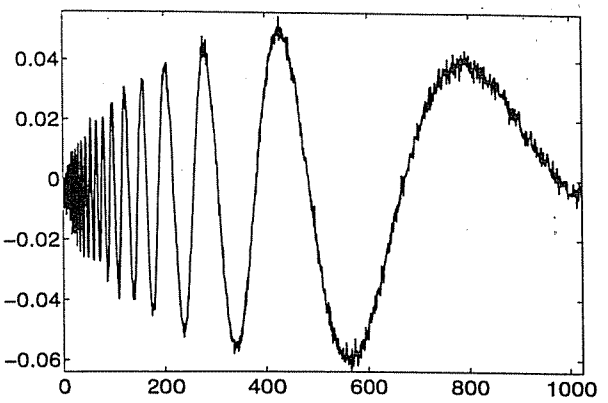
Compression

Recall that if the signal is smooth, and the wavelet has several vanishing moments, then the detail coefficients at fine resolution are very small.

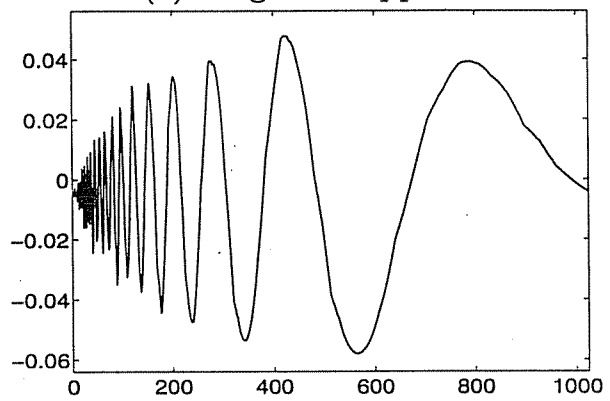
Wavelets with several vanishing moments are especially suited for signal compression.



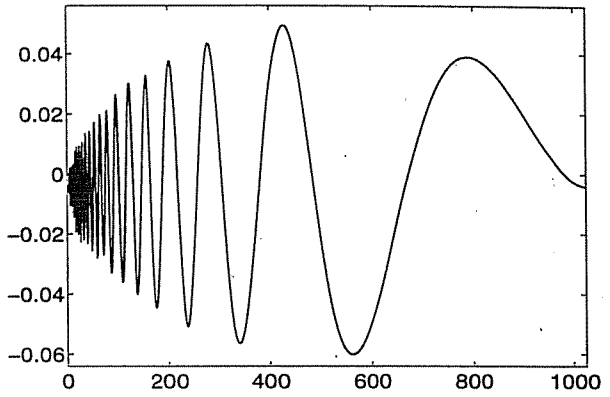
(a) Original Doppler



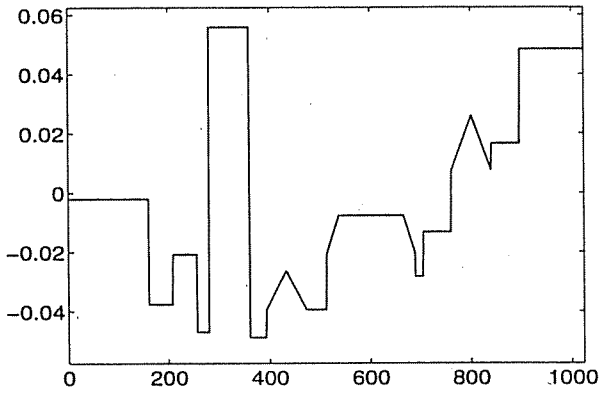
(b) Noisy Doppler



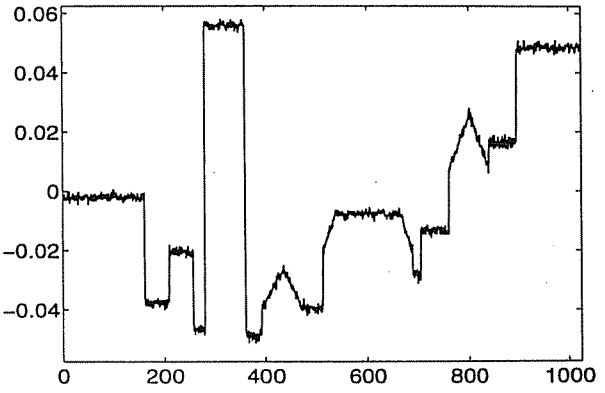
(c) DWT Denoised Doppler



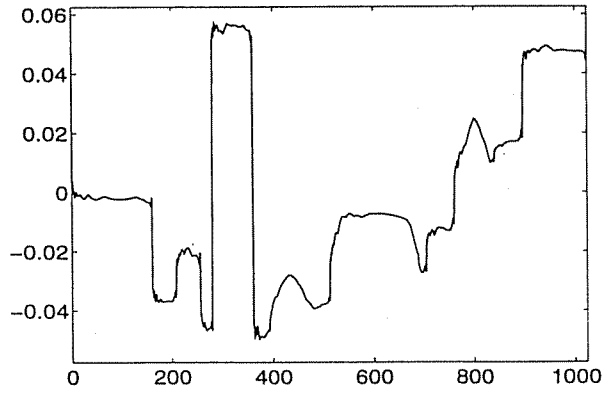
(d) RDWT Denoised Doppler



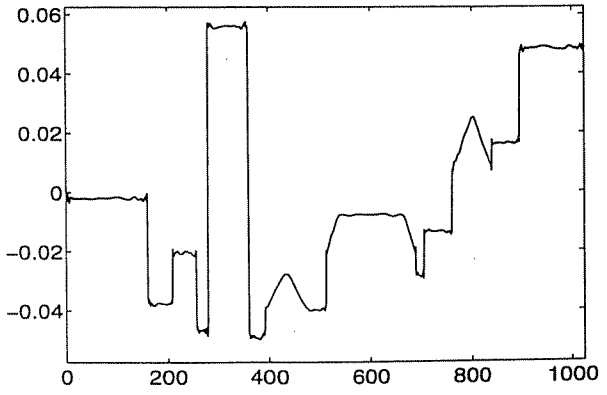
(e) Original Houston Skyline



(f) Noisy Houston Skyline



(g) DWT Denoised Skyline



(h) RDWT Denoised Skyline

RDWT = redundant DWT = SIDWT

Compression is a bit of an art. There are various ways to do it:
- thresholding (soft or hard)
- keeping the top x % of coefficients
:

Obviously, more compression is related to more distortion. You need to decide how much is acceptable.

To do either denoising or compression in the wavelet toolbox, you can ask Matlab to estimate the correct threshold for you:

```
[t, sorh, keepapp] = ddencmp('den' or 'cmp', 'wv', s)
                                                  ↑ Signal
```

t = threshold
sorh = 's' or 'h' (soft or hard)
keepapp = keep approximation coefficients ?
1 = yes (don't threshold a coefficients)
0 = false (threshold both a and d)

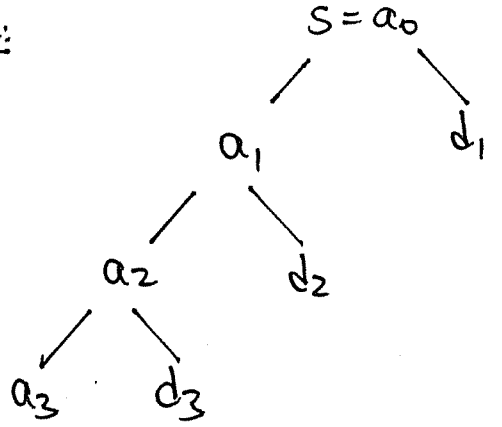
The actual denoising or compression is done by

```
c_new = wthcoef('t', c, L, N, T, sorh)
```

N = integer vector containing the levels to be thresholded
T = corresponding thresholds

only d coefficients get thresholded

example:



$C = [a_3, d_3, d_2, d_1]$

$L = [\#a_3, \#d_3, \#d_2, \#d_1, \#a_0]$

C_{new} = wthcoef('t', C, L, [1,3], [.1, .05], 'h')

use hard threshold 0.1 in d_1, threshold 0.05 in d_3.

You can also do percentage-based zeroing:

C_{new} = wthcoef('d', C, L, [1,3], [25, 50]);

zero 25% of coefficients in d_1, 50% of coefficients in d_3

default is to zero 100%

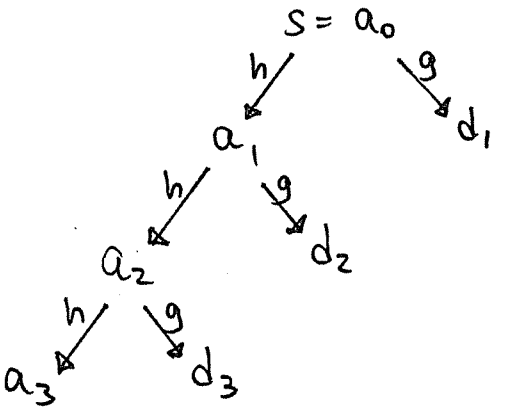
C_{new} = wthcoef('d', C, L, 3): wipe out d_3

wthcoef('a', C, L): wipe out a_3

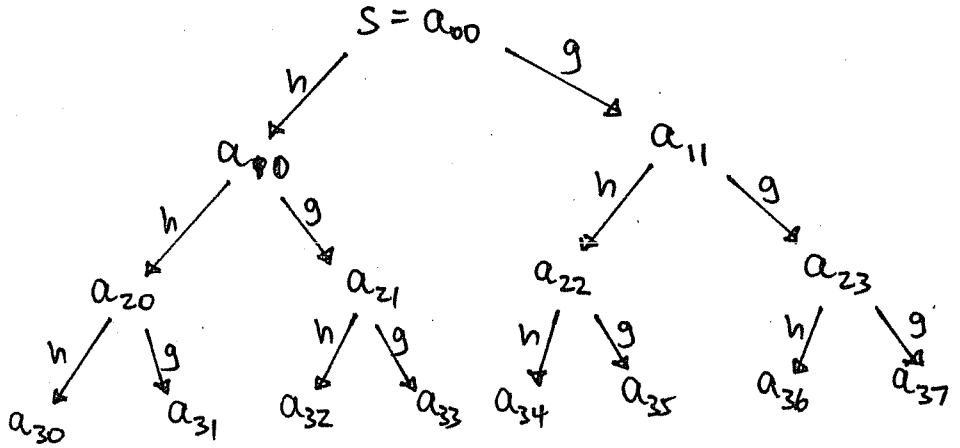
For applications to compression, there is yet another variation on the DWT which is widely used, and built into the wavelet toolbox: wavelet packets, or simply wave packets.

The basic idea is: instead of decomposing only the "a" signal at every level, we decompose both "a" and "d".

Wavelet decomposition

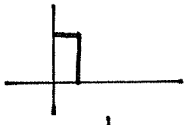


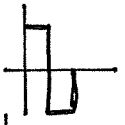
wave packet decomposition

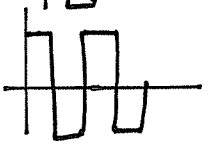


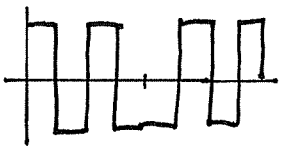
The basis functions that go with the various levels can be found by applying the recursion formula with either the h or g coefficients as you go from top to bottom.

Example: Haar wavelet $h = \{1, 1\}$, $g = \{1, -1\}$ (ignore scaling)

$a_{00} \Leftrightarrow$  φ_{00}

$a_{11} \Leftrightarrow \varphi_{11} =$  $(1 \cdot \varphi_{00} + (-1) \cdot \text{shifted } \varphi_{00})$

$a_{22} \Leftrightarrow \varphi_{22} =$  $(1 \cdot \varphi_{11} + 1 \cdot \text{shifted } \varphi_{11})$

$a_{35} \Leftrightarrow \varphi_{35} =$  $(1 \cdot \varphi_{22} + (-1) \cdot \text{shifted } \varphi_{22})$

The wavelet toolbox supports the use of the wave packet decomposition for denoising, but I am not convinced that is appropriate.

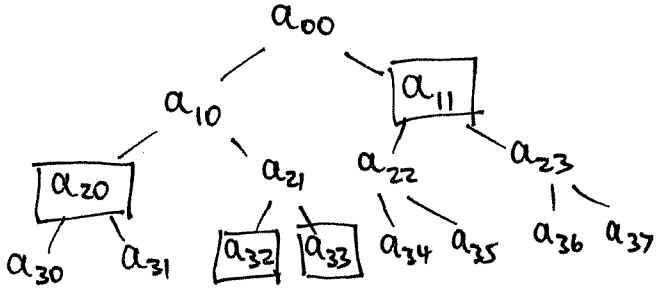
In the wavelet decomposition, the a_j, d_j have interpretations as approximation and detail coefficients, or parts of the spectrum of S.

A similar interpretation of the components of the wave packet decomposition is not so obvious.

For compression, on the other hand, wave packets are very well suited. We don't care what the coefficients mean, as long as a lot of them are negligible.

The true power of the wave packet decomposition lies not in the complete decomposition, but in partial decompositions.

For example:

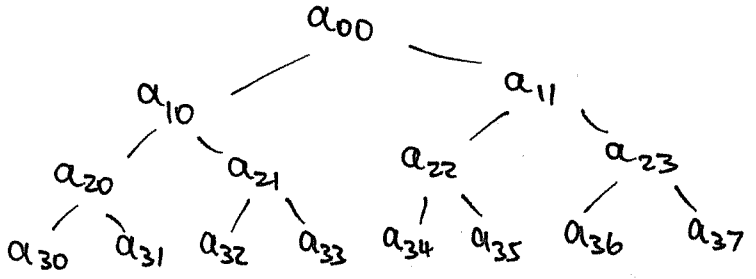


For every node, you have a choice of keeping the coefficients at that node, or to make sure that each of the two branches can be reconstructed.

For example, $\{a_{20}, a_{32}, a_{33}, a_{11}\}$ is a valid decomposition tree.

The wave packet compression algorithm

① do a complete wave packet decomposition:



② go through the levels, bottom to top.

level 3: compress (by thresholding)

level 2: compress each node, and compare to level 3. Decide whether to keep a_{20}, or $a_{30}+a_{31}$, then a_{21} or $(a_{32}+a_{33})$, etc.

level 1: keep going like this: should we keep a_{10}, or the subtrees $a_{20}+a_{21}$?

⋮

Overall effort: $O(N \log N)$.

The implementation of all this in the wavelet toolbox is a bit complicated, unfortunately.

Each node in the decomposition tree is assigned a numerical value called an "entropy". You have to specify an entropy, and it gets stored in the data structure along with the coefficients.

The compression algorithm described above is implemented as follows:

① select the subtree you want to use by applying the above algorithm, using entropy, not the number of retained coefficients.

② compress the selected subtree by thresholding.

available entropies are

default → 'shannon': $E(s) = -\sum_i s_i^2 \log s_i^2$

'log energy': $E(s) = \sum_i \log(s_i^2)$

} by definition $\log 0 = 0$

'norm', p: $E(s) = \sum_i |s_i|^p$ $\quad 1 \le p < 2$

'threshold', t: $E(s) =$ number of s_i with $|s_i| > t$

'user', 'f': $E(s) = f(s)$, f = user-supplied subroutine

'sure', t: $E(s) =$ (number of s_i with $|s_i| > t$) $+ \sum_i \min(s_i^2, t^2)$

Relevant Matlab subroutines

decomposition: `[T,D] = wpdec(s, n, 'wavelet', 'entropy', entropy-parameter)` (if needed)

(s = signal, n = number of levels)

reconstruction: `s = wprec(T,D)`

compare this to `[C,L] = wavedec(s,n,'wavelet')`, `s = waverec(C,L,'wavelet')`

C = coefficients

L = structure information

T = "tree" = structure information

D = "data" = coefficients + entropy + wavelet coefficients + who knows what else

For the wave packet reconstruction, you don't need to specify the wavelet, since that is stored in the tree itself.

change entropy: D_{new} = entrupd (T, D, 'entropy', e_par)

coefficients and tree structure are unchanged

select tree with smallest total entropy

$[T_{new}, D_{new}]$ = besttree (T, D)

note that "number of coefficients above threshold t" is one of the available entropies, so the algorithm on page 11-11 can be implemented.

find recommended thresholds

[t, sorh, keepapp] = ddencmp ('den' / 'cmp', 'wp', s)

keepapp ↑ entropy

see page 11-7 (same routine for wavelets)

Question: does this routine ever consider 'norm' as a possible entropy? If yes, how would it return p?

Note: for entropies 'threshold' and 'sure', the t parameter does double duty: as threshold for the entropy, and as threshold for the compression.

perform thresholding

$[s_{new}, T_{new}, D_{new}, perf0, perfL2]$

= wpdencmp (T, D, sorh, 'entropy', t, keepapp)

perf0 = achieved compression, in percent

perfL2 = retained L_2- energy, in percent

alternatively, you can call wpdencmp with the original signal:

$[s_{new}, T_{new}, D_{new}, perf0, perfL2]$

= wpdencmp (s, sorh, n, 'wavelet', 'entropy', t, keepapp)

The complete procedure:

[t, sorh, keepapp, entropy] = ddencmp(...)

[T,D] = wpdec (S, ...)

$[T_{new}, D_{new}]$ = besttree (T,D)

$[S_{new}, \ldots]$ = wpdencmp (T,D,...)

} same as

$[S_{new}, \ldots]$ = wpdencmp (s, ...)

(no reconstruction needed, since wpdencmp already does that)

Remark: "besttree" may not be needed.

If wpdencmp kept the same tree, it wouldn't need to know what the entropy is. Since 'entropy' is a parameter, wpdencmp is probably capable of refining the tree.