Transform Methods & Signal Processing

lecture 10

Matthew Roughan
<matthew.roughan@adelaide.edu.au>

Discipline of Applied Mathematics
School of Mathematical Sciences
University of Adelaide

October 26, 2009

Wavelet Filters

The previous Wavelet transforms (even the discrete Wavelet transform) are transforms of continuous functions. In this section we consider the natural approach for applying wavelets to discrete signals, which we give the name Wavelet Filters.

We now take wavelets onto the domain of discrete-time signals (all of the work in lecture 9 concerns wavelets on continuous functions). We call the wavelet transform on a discrete-time signal a Discrete Wavelet Filter. When we consider discrete-time signals, there are significant computational advantages to the wavelet transform, in particular we will discuss the pyramidal filter-bank approach to computing wavelets.

Discrete wavelet filters (DWF)

Previous lecture

- continuous wavelet transform maps functions $f : \mathbb{R} \to \mathbb{R}$ to a new function $W_f : \mathbb{R}^2 \to \mathbb{R}$
- discrete wavelet transform maps the same function $f : \mathbb{R} \to \mathbb{R}$ to a function on the dyadic grid. But its still a transform of a function of a continuous space.
- for signal processing the signals are functions on a discrete space
- we need to have the equivalent of a DFT
  - we will call this the DWF to avoid confusing with the DWT
  - also need fast computation methods

Continuous to discrete

Discrete Wavelet Transform of $f(t)$

$$W_f(u,s) = \langle f, \psi_{u,s} \rangle = \int_{-\infty}^{\infty} f(t) \frac{1}{\sqrt{s}} \psi^* \left(\frac{t-u}{s} \right) dt$$

DWT takes integer scales $s = 2^j$, and translations $u = 2^j n$, so basis functions are $\psi_{n,j}(t) = \frac{1}{\sqrt{2^j}} \psi \left(\frac{t}{2^j} - n \right)$

We get a discrete version of this by sampling the orthogonal basis functions at scale j to get

$$W_f(n,j) = \sum_{m=0}^{N-1} f(m) \psi_j^*(m-n)$$

Transform Methods & Signal Processing (APP MTH 4043): lecture 10 – p.3/75

Transform Methods & Signal Processing (APP MTH 4043): lecture 10 – p.4/75

Continuous to discrete

Wavelet Reconstruction (synthesis), is the same for both DWT and DWF

$$f = \sum_j \sum_n \langle f, \psi_{n,j} \rangle \psi_{n,j}$$

however, the inner products are different in each case

- DWT

$$\langle f, \psi_{u,s} \rangle = \int_{-\infty}^{\infty} f(t) \frac{1}{\sqrt{s}} \psi^* \left(\frac{t-u}{s} \right) dt$$

  for $s = 2^j$ and $u = n2^j$

- DWF

$$\langle f, \psi_{n,j} \rangle = \sum_{m=0}^{N-1} f(m) \psi_j^*(m-n)$$

Wavelets as convolution

Ignoring the edges (or doing a circular convolution)

$$\begin{aligned} \langle f, \psi_{n,j} \rangle &= \sum_{m=0}^{N-1} f(m) \psi_j^*(m-n) \\ &= \left[f * \bar{\psi}_j \right](n) \end{aligned}$$

where we define

$$\bar{\psi}_j(n) = \psi_j^*(-n)$$

i.e. a time reversal (and complex conjugate where needed) of the filter.

Transform Methods & Signal Processing (APP MTH 4043): lecture 10 – p.5/75

Transform Methods & Signal Processing (APP MTH 4043): lecture 10 – p.6/75

How to get wavelet filter: sampling

Start with mother wavelet $\psi(t)$ with support $[0, K]$

- form wavelets (for $n = 0$) at each octave by

$$\psi_{0,j}(t) = \frac{1}{\sqrt{2^j}} \psi\left(\frac{t}{2^j}\right)$$

- support of wavelet at octave j is $[0, 2^j K]$
- sample at unit intervals (i.e., $f_s = 1$)

$$\psi_j(n) = \frac{1}{\sqrt{2^j}} \psi\left(\frac{n}{2^j}\right)$$

- length of filter ψ_j is $2^j K$

Possible range of scales

- Continuous signal $f(t)$, for $t \in [0, T]$
  - sample at times $t = n/N$ for $n = 0, 1 \ldots, TN$
  - sampling interval $t_s = 1/N$
  - sampling frequency $f_s = N$
  - results in discrete signal $x(n)$
  - possible scales for approximation $N^{-1} \leq s \leq T$
- For our purposes here
  - choose $T = N - 1$, so $t \in [0, N-1]$
  - also take $f_s = 1$
  - sample at times $t = n$ for $n = 0, 1 \ldots, N-1$
  - possible scales for approximation $1 \leq s \leq N$

This is bit of a naive guess of how to ge tthe filters, but we will construct them using mathematical arguments later on.

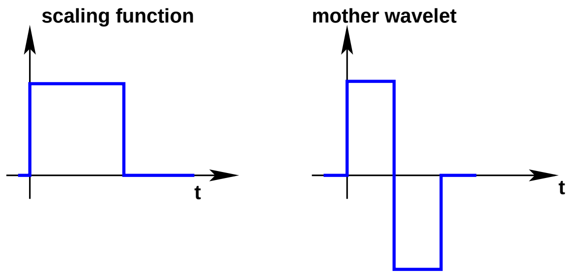
Note that we can do a similar sampling for scaling functions.

Transform Methods & Signal Processing (APP MTH 4043): lecture 10 – p.7/75

Transform Methods & Signal Processing (APP MTH 4043): lecture 10 – p.8/75

Example: Haar wavelets

The Haar wavelet and scaling function are shown below



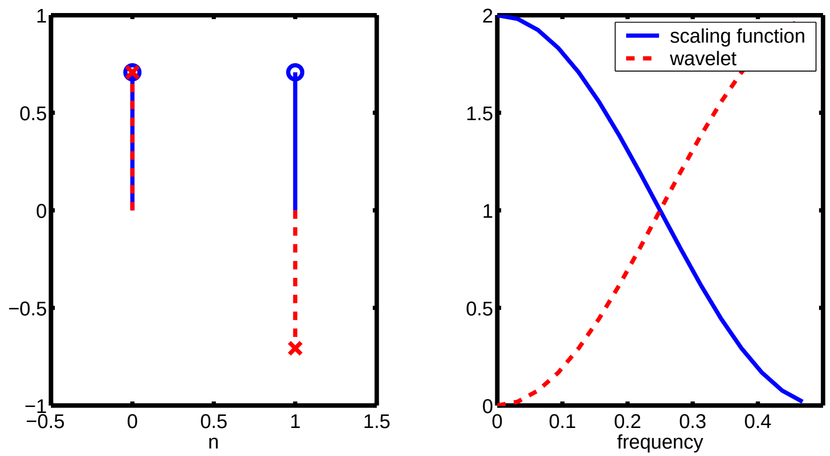
They have support $[0,1]$, so at octave $j=1$ we sample at 2 points, to get

$$\begin{aligned} \psi_1(0) &= \frac{1}{\sqrt{2^1}}\psi(0) &= 1/\sqrt{2} \\ \psi_1(1) &= \frac{1}{\sqrt{2^1}}\psi(1) &= -1/\sqrt{2} \end{aligned}$$

and a similar result for the scaling function.

Example: Haar wavelets

$j=1$



Transform Methods & Signal Processing (APP MTH 4043): lecture 10 – p.9/75

Transform Methods & Signal Processing (APP MTH 4043): lecture 10 – p.10/75

Example: Haar wavelets

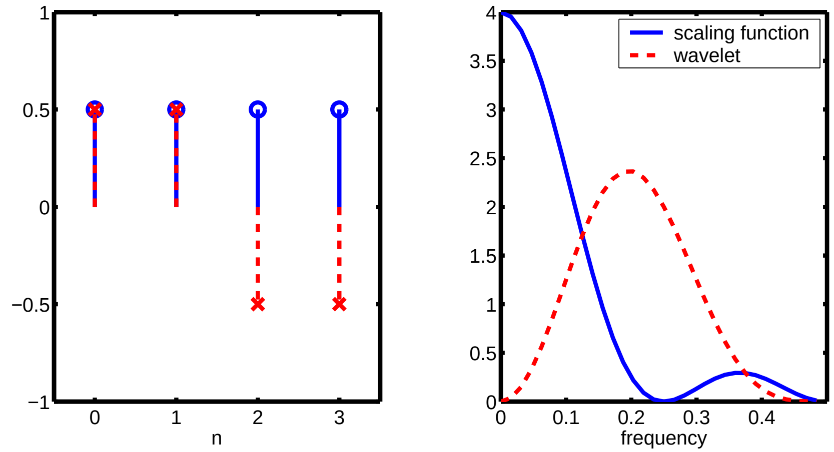
At octave $j = 2$ we sample at 2^2 points, to get

$$
\begin{aligned}
\psi_2(0) &= \frac{1}{\sqrt{2^2}}\psi(0) &= 1/2 \\
\psi_2(1) &= \frac{1}{\sqrt{2^2}}\psi(1/4) &= 1/2 \\
\psi_2(2) &= \frac{1}{\sqrt{2^2}}\psi(1/2) &= -1/2 \\
\psi_2(3) &= \frac{1}{\sqrt{2^2}}\psi(3/4) &= -1/2
\end{aligned}
$$

and a similar result for the scaling function.

Example: Haar wavelets

$j = 2$

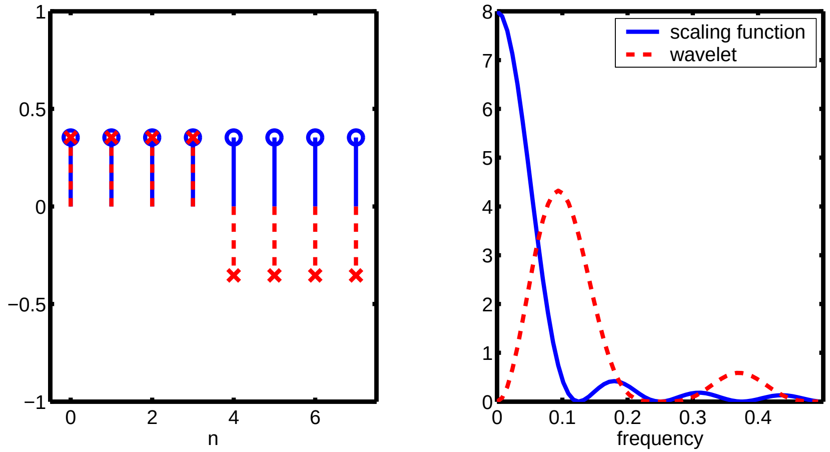


Transform Methods & Signal Processing (APP MTH 4043): lecture 10 – p.11/75

Transform Methods & Signal Processing (APP MTH 4043): lecture 10 – p.12/75

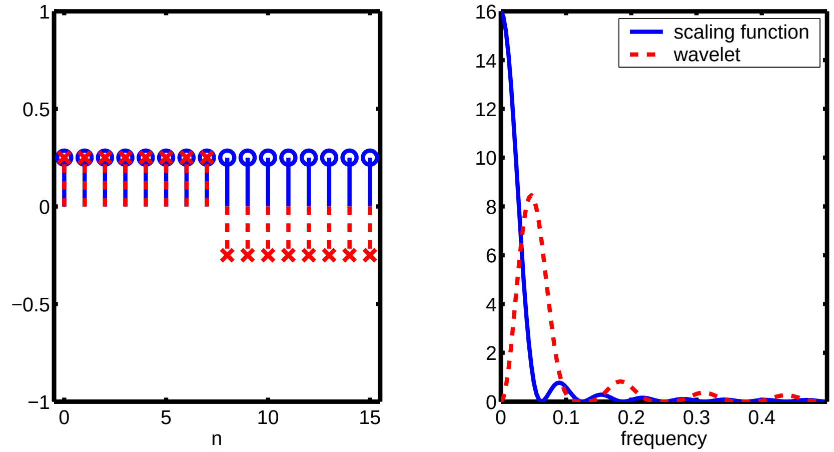
Example: Haar wavelets

Similarly $j = 3$



Example: Haar wavelets

Similarly $j = 4$



Transform Methods & Signal Processing (APP MTH 4043): lecture 10 – p.13/75

Transform Methods & Signal Processing (APP MTH 4043): lecture 10 – p.14/75

Discrete wavelet filters (DWF)

Ignoring edge effects (implies signal periodicity) the wavelet transform becomes

$$W_f(n, 2^j) = \sum_{m=0}^{N-1} f(m)\psi_j^*(m-n) = [f * \bar{\psi}_j](n)$$

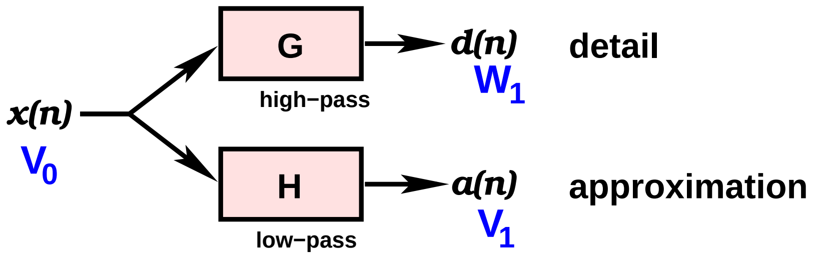
We can do the same with the scaling functions.

- compute in time or frequency domain
  - direct calculation at scale j takes $O(NK2^j)$
  - FFT calculation at scale j takes $O(N \log N)$
- Neither is particular efficient.
- Also need to compute across $O(\log_2(N))$ scales.

Pyramidal decomposition algorithm

Use the fact we can break into approximation and detail

- get approximation using scaling filter H
- get details from wavelet filter G



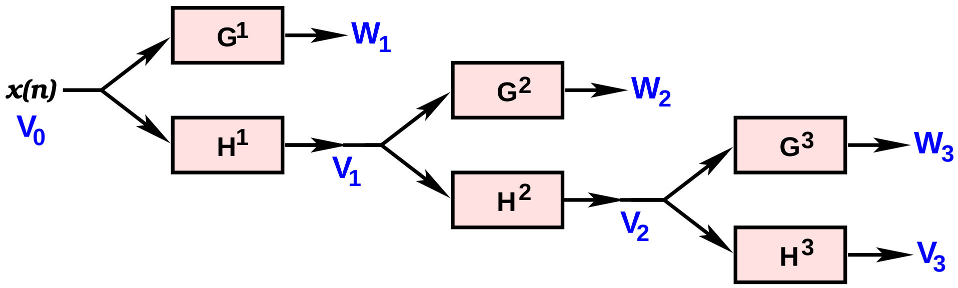
- note $x(n)$ is already sampled to give an approximation of the original function.

Transform Methods & Signal Processing (APP MTH 4043): lecture 10 – p.15/75

Transform Methods & Signal Processing (APP MTH 4043): lecture 10 – p.16/75

Pyramidal decomposition algorithm

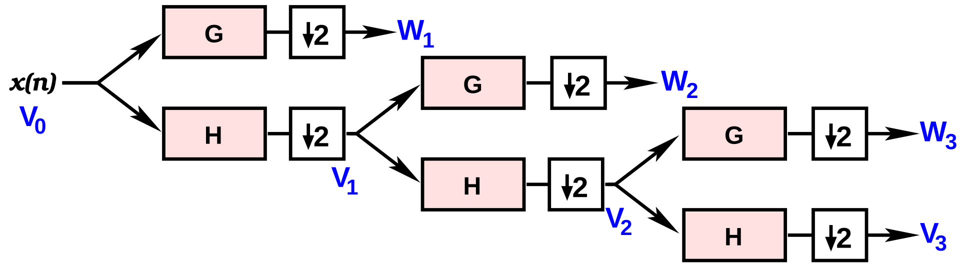
Now use successive approximation



- note though, different filters at each level
- still $O(\log(N))$ levels of filters, so calculations are $O(KN\log(N))$

Pyramidal decomposition algorithm

- there is redundancy
  - coarse approximations have been low-passed, so we could reduce the sample rate (dyadic grid)
  - use conjugate mirror filters, and we get exact reconstruction even with downsampling by 2
  - thanks to downsampling, computation is $O(NK)$

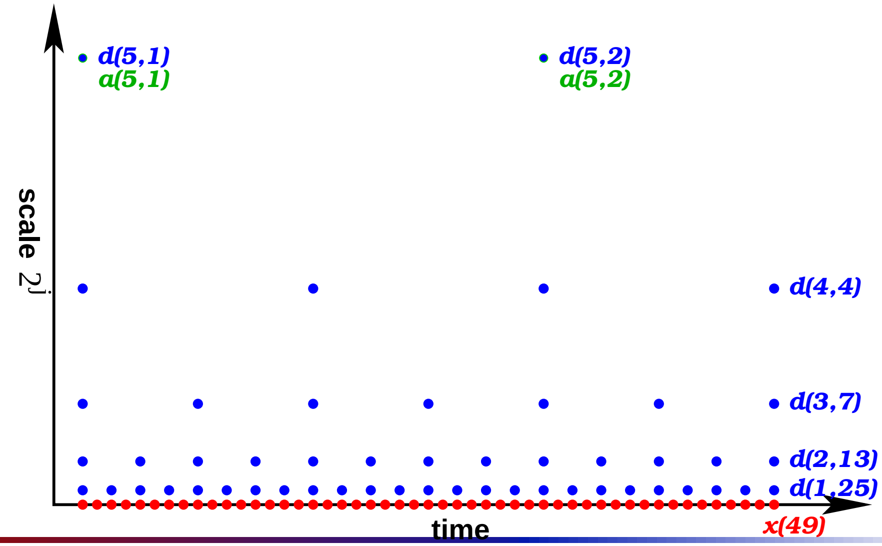


Transform Methods & Signal Processing (APP MTH 4043): lecture 10 – p.17/75

Transform Methods & Signal Processing (APP MTH 4043): lecture 10 – p.18/75

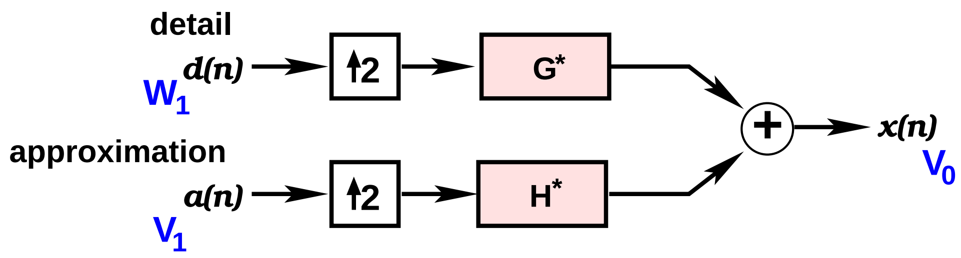
Pyramidal decomposition algorithm

Downsampling automatically results in dyadic grid.



Pyramidal decomposition algorithm

Reconstruction just reverses the process



Even use the same filters (for orthonormal wavelets).

Transform Methods & Signal Processing (APP MTH 4043): lecture 10 – p.19/75

Transform Methods & Signal Processing (APP MTH 4043): lecture 10 – p.20/75

Pyramidal decomposition algorithm

Writing out in full, including downsampling:

$$a_j(n) = \langle f, \phi_{n,j} \rangle \quad \text{and} \quad d_j(n) = \langle f, \psi_{n,j} \rangle$$

Decomposition

$$a_{j+1}(n) = \sum_{m=-\infty}^{\infty} h(m-2n) a_j(m) = \left[a_j * \overline{h}\right](2n)$$

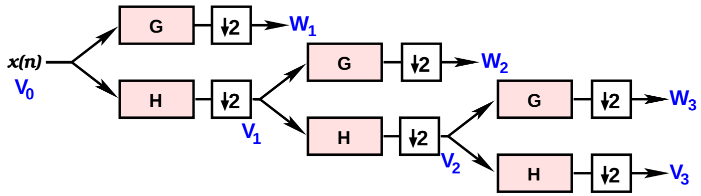
$$d_{j+1}(n) = \sum_{m=-\infty}^{\infty} g(m-2n) a_j(m) = \left[a_j * \overline{g}\right](2n)$$

Reconstruction

$$a_j(n) = \sum_{m=-\infty}^{\infty} h(n-2m) a_{j+1}(m) + \sum_{m=-\infty}^{\infty} g(n-2m) d_{j+1}(m)$$

Example

Example: Find the Haar wavelet coefficients on the dyadic grid (at octaves $j = 1, 2$ and 3) for a signal $(0,0,0,0,1,1,1,1,0,0,0,0,1,1,0,0)$.



The G and H blocks refer to convolution with the discrete filters, which for the Haar wavelets are

$$h = (1,1)/\sqrt{2}$$

$$g = (1,-1)/\sqrt{2}$$

The $\downarrow 2$ blocks refer to downsampling.

Transform Methods & Signal Processing (APP MTH 4043): lecture 10 – p.21/75

Transform Methods & Signal Processing (APP MTH 4043): lecture 10 – p.22/75

Example

So we need to compute convolutions of the form:

$$x * g \downarrow 2 = \sum_{m=-\infty}^{\infty} h(m-2n)x(m)$$

$$x * h \downarrow 2 = \sum_{m=-\infty}^{\infty} g(m-2n)x(m)$$

The first term is the wavelet details $d_1(n)$, and the second term is the approximation $a_1(n)$ (at octave $j=1$), i.e.

$$a_1(n) = \sum_{m=-\infty}^{\infty} h(m-2n)x(m)$$

$$d_1(n) = \sum_{m=-\infty}^{\infty} g(m-2n)x(m)$$

Example

To simplifying the computations, we shall ignore scaling factor of $\sqrt{2}$ until the end. So $h(0)=1$ and $h(1)=1$, and

$$\begin{aligned}
a_1(0) &= h(0)x(0)+h(1)x(1) = x(0)+x(1) = 0 \\
a_1(1) &= h(0)x(2)+h(1)x(3) = 0 \\
a_1(2) &= h(0)x(4)+h(1)x(5) = 2 \\
a_1(3) &= h(0)x(6)+h(1)x(7) = 2 \\
a_1(4) &= h(0)x(8)+h(1)x(9) = 0 \\
a_1(5) &= h(0)x(10)+h(1)x(11) = 0 \\
a_1(6) &= h(0)x(12)+h(1)x(13) = 2 \\
a_1(7) &= h(0)x(14)+h(1)x(15) = 0
\end{aligned}$$

So $a_1 = (0,0,2,2,0,0,2,0)$. Notice there are half as many terms as the original signal because of the $\downarrow 2$.

Transform Methods & Signal Processing (APP MTH 4043): lecture 10 – p.23/75

Transform Methods & Signal Processing (APP MTH 4043): lecture 10 – p.24/75

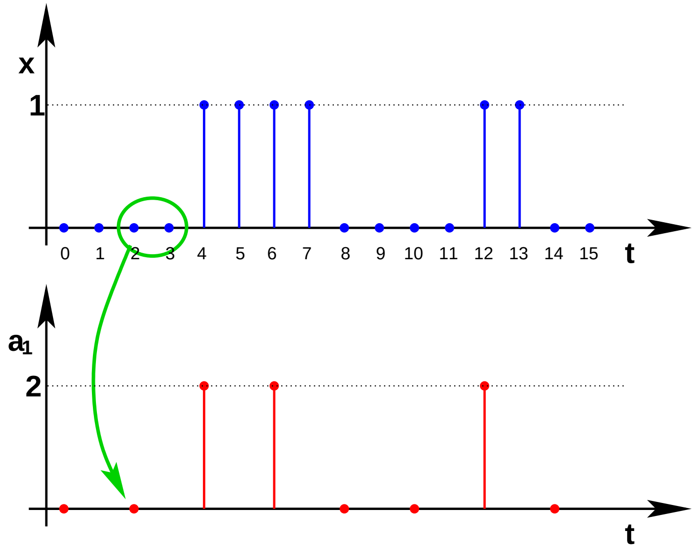
Example

Similarly $g(0) = 1$ and $g(1) = -1$ so

$$\begin{aligned}
d_1(0) &= g(0)x(0) + g(1)x(1) = x(0) - x(1) = 0 \\
d_1(1) &= g(0)x(2) + g(1)x(3) = 0 \\
d_1(2) &= g(0)x(4) + g(1)x(5) = 0 \\
d_1(3) &= g(0)x(6) + g(1)x(7) = 0 \\
d_1(4) &= g(0)x(8) + g(1)x(9) = 0 \\
d_1(5) &= g(0)x(10) + g(1)x(11) = 0 \\
d_1(6) &= g(0)x(12) + g(1)x(13) = 0 \\
d_1(7) &= g(0)x(14) + g(1)x(15) = 0
\end{aligned}$$

So $d_j = (0,0,0,0,0,0,0,0)$, so the approximation at octave 1 is perfect (which we could see from the signal).

Example



Transform Methods & Signal Processing (APP MTH 4043): lecture 10 – p.25/75

Transform Methods & Signal Processing (APP MTH 4043): lecture 10 – p.26/75

Example

Given the pyramidal structure, we can iterate

$$
\begin{aligned}
a_{j+1}(n) &= \sum_{m=-\infty}^{\infty} h(m-2n)a_j(m) = \left[a_j * \overline{h}\right](2n) \\
d_{j+1}(n) &= \sum_{m=-\infty}^{\infty} g(m-2n)a_j(m) = [a_j * \overline{g}](2n)
\end{aligned}
$$

Example

If we use this to compute the second scale approximation we get

$$
\begin{aligned}
a_2(0) &= h(0)a_1(0) + h(1)a_1(1) = 0 \\
a_2(1) &= h(0)a_1(2) + h(1)a_1(3) = 4 \\
a_2(2) &= h(0)a_1(4) + h(1)a_1(5) = 0 \\
a_2(3) &= h(0)a_1(6) + h(1)a_1(7) = 2 \\
\\
d_2(0) &= g(0)a_1(0) + g(1)a_1(1) = 0 \\
d_2(1) &= g(0)a_1(2) + g(1)a_1(3) = 0 \\
d_2(2) &= g(0)a_1(4) + g(1)a_1(5) = 0 \\
d_2(3) &= g(0)a_1(6) + g(1)a_1(7) = 2
\end{aligned}
$$

So $a_2 = (0,4,0,2)$ and $d_2 = (0,0,0,2)$.

Transform Methods & Signal Processing (APP MTH 4043): lecture 10 – p.27/75

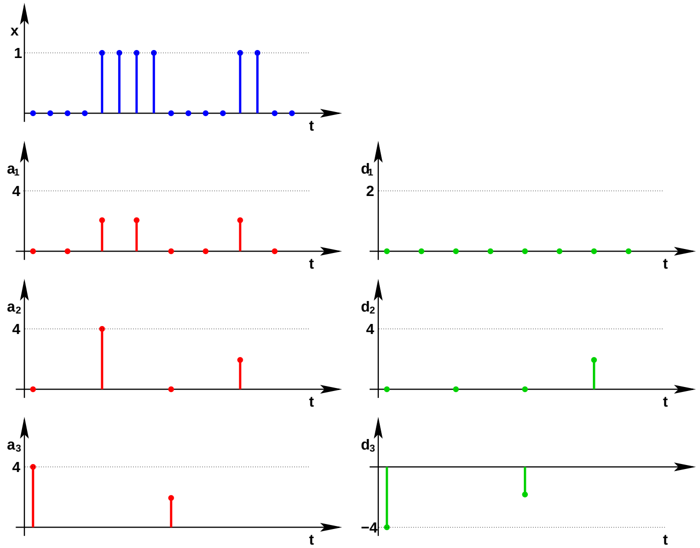
Transform Methods & Signal Processing (APP MTH 4043): lecture 10 – p.28/75

Example

We repeat to get the next higher scale.

$$
\begin{aligned}
a_3(0) &= h(0)a_2(0) + h(1)a_2(1) \\
&= a_2(0) + a_2(1) \\
&= 4 \\
a_3(1) &= h(0)a_2(2) + h(1)a_2(3) = 2 \\
\\
d_3(0) &= g(0)a_2(0) + g(1)a_2(1) \\
&= a_2(0) - a_2(1) \\
&= -4 \\
d_3(1) &= g(0)a_2(2) + g(1)a_2(3) = -2
\end{aligned}
$$

Example



Transform Methods & Signal Processing (APP MTH 4043): lecture 10 – p.29/75

Transform Methods & Signal Processing (APP MTH 4043): lecture 10 – p.30/75

Example

The MRA up to octave 3 (based on the Haar wavelets) including the $\sqrt{2}$ factors is therefore given by $\{a_3, d_1, d_2, d_3\}$ where these are

$$
\begin{aligned}
a_3 &= (4,2)/2^{3/2} \\
&= (\sqrt{2}, 1/\sqrt{2}) \\
d_3 &= (0,-2)/2^{3/2} \\
&= (-2/\sqrt{2}, -1/\sqrt{2}) \\
d_2 &= (0,0,0,2)/2 \\
&= (0,0,0,1) \\
d_1 &= (0,0,0,0,0,0,0,0)/\sqrt{2} \\
&= (0,0,0,0,0,0,0,0)
\end{aligned}
$$

Representation

We can represent (without loss of information or redundancy) the octave L approximation of a signal $x(n)$ by $\{\{d_j\}_{L<j\leq J}, a_J\}$.

- we don't have to go all the way with the transform.
- only need to get the details at scales $2^L < 2^j \leq 2^J$ along with approximation at scale J.

Transform Methods & Signal Processing (APP MTH 4043): lecture 10 – p.31/75

Transform Methods & Signal Processing (APP MTH 4043): lecture 10 – p.32/75

Initialization

Want to associate $x(n)$ the sampled signal with an approximation for the signal $f(t)$ at some scale 2^L.

- sampling interval is N^{-1}
- Initial scale $2^L = N^{-1}$
- need to compute $a_L(n) = \langle f, \phi_{n,L} \rangle$ from $x(n) = f(n/N)$
- Mallat, pp. 257-258, if f is regular

$$a_L(n) = N^{-1/2} x(n)$$

essentially, we are already supposed to have band-passed the signal before sampling.

Deriving the filters directly

Property 1 of MRA: $\mathbf{V}_{j+1} \subset \mathbf{V}_j$ for all $j \in \mathbb{Z}$ must apply to basis functions, so $\phi_{0,j+1} \in \mathbf{V}_j$, and can therefore be written as a linear combination of the basis functions of $\mathbf{V}_j$, i.e.

$$\phi_{0,j+1}(t) = \sum_n h(n) \phi_{n,j}(t)$$

with $h(n) = \langle \phi_{0,j+1}, \phi_{n,j} \rangle$. Take the case with $j = 0$, then this implies

$$\frac{1}{\sqrt{2}} \phi\left(\frac{t}{2}\right) = \sum_n h(n) \phi(t-n) = [h * \phi](t)$$

The relationship defines a filter $h(n) = \left\langle \frac{1}{\sqrt{2}} \phi\left(\frac{t}{2}\right), \phi(t-n) \right\rangle$ for the scaling function ϕ (similar to derivation of relation between ϕ and ψ)

Transform Methods & Signal Processing (APP MTH 4043): lecture 10 – p.33/75

Transform Methods & Signal Processing (APP MTH 4043): lecture 10 – p.34/75

Example: Haar

$$
\begin{aligned}
h(n) &= \left\langle \frac{1}{\sqrt{2}} \phi\left(\frac{t}{2}\right), \phi(t-n) \right\rangle \\
&= \frac{1}{\sqrt{2}} \int_{-\infty}^{\infty} \phi\left(\frac{t}{2}\right) \phi(t-n)\, dt \\
&= \frac{1}{\sqrt{2}} \int_{0}^{2} \phi(t-n)\, dt \\
&= \begin{cases} 1/\sqrt{2} & \text{if } n = 0,1 \\ 0 & \text{otherwise} \end{cases}
\end{aligned}
$$

And we can perform a similar calculation to get the wavelet filter.

$$g(0) = 1/\sqrt{2} \quad \text{and} \quad g(1) = -1/\sqrt{2}$$

Finite signals

Methods to deal with edge effects

- zero padding
- assume periodic signal (do circular convolution)
- Boundary wavelets vanish at the boundary

Haar

- simple two tap filters
  - Scaling filter is just $h = [1,1]/\sqrt{2}$.
  - Wavelet filter is just $g = [1,-1]/\sqrt{2}$.
- only linear phase (symmetric) wavelet with compact support
- filters don't have very good transitions, or stop-band attenuation
- can we design better wavelet filters

Filter properties

Similar to windows, there are many properties, and tradeoffs between different types of wavelet filters

- support
  - compact
  - size
  - number of taps
- transition region, stop band, and roll off
- vanishing moments
- regularity
- real or complex
- linear phase (symmetric)

Vanishing moments

A wavelet has p vanishing moments if

$$\int_{-\infty}^{\infty} t^k \psi(t)\, dt = 0 \quad \text{for } 0 \le k < p$$

This means that $\psi(t)$ will be orthogonal to any polynomial of order $p-1$.

- first $p-1$ derivatives of FT are zero at $freq = 0$
- filter $G(z)$ has p zeros at $z=1$ (in the complex plane)
- polynomials (order < p) are dropped by wavelets with p vanishing moments, so we can examine data with polynomial trends, and these won't effect the detail coefficients (the trend will be in the approximation).

Support

See previous notes on windows.

- scaling function ϕ has same support as filter h (which is its number of taps)
  - as a result of the previous slide
- Support of scaling function ϕ is $[N_1, N_2]$ implies Wavelet support ψ is $[(N_1 - N_2 + 1)/2, (N_2 - N_1 + 1)/2]$
  - same width of support, so filters g and h have same number of taps
- if ψ has p vanishing moments, then it must have support of at least $2p-1$
  - minimal for Daubechies wavelets

Why are vanishing moments important. Think about denoising. Assume we have a signal $P(t)$ which was a polynomial of order $p-1$, to which some (non-polynomial) noise has been added, so that we now have a signal

$$f = P + \varepsilon$$

Apply the wavelet transform (with p vanishing moments) to the signal f, and the details completely ignore the polynomial P, and so they represent only noise ε. Hence, if we set the details to zero, and invert the transform, we will remove some of the ε component, and get closer to the original signal p.
Now, most signals aren't polynomials, but they can often be approximated by a polynomial locally (e.g. using Taylor series). If the local approximation holds over a long enough interval such that the wavelet filters can be applied, then we can get essentially the same result.

Regularity

- cosmetic effect on errors from thresholding or quantization
  - induced errors are smoother
- may be important for subjective quality of compressed image
- often increases with p (but not guaranteed)

Daubechies wavelets

Goal: Minimal support for a given number of vanishing moments p.

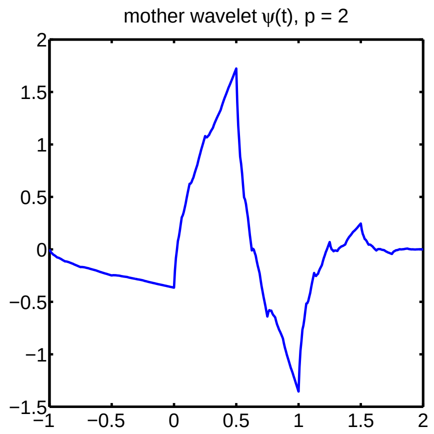
- To ensure p vanishing moments we need to have p zeros at $\omega = \pi$, so z-transform is minimum degree polynomial have p zeros at -1, so should have p factors of $(1+z)$.
- Real conjugate mirror filters, and normalization impose other requirements.
- results is a popular family of wavelets
  - $p = 1$ you get the Haar wavelets

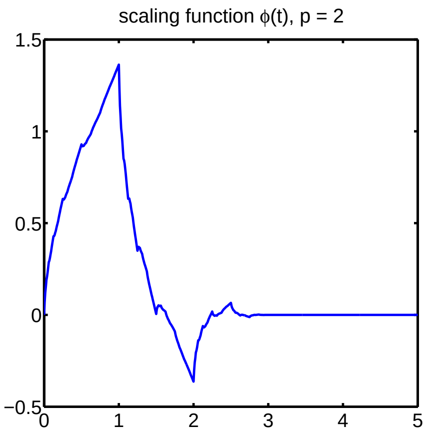
Transform Methods & Signal Processing (APP MTH 4043): lecture 10 – p.41/75

Note there is also a tradeoff between the number of vanishing moments and the support of the wavelet (and the resulting filter length). This relates back to the approximation by polynomials. We often get a better approximation to a signal using a higher order polynomial, but this would require a longer wavelet filter, and hence the approximation would have to be good over a wider range. So the two facts tradeoff, and it is not always obvious what length filter will be best for denoising a sequence.

Transform Methods & Signal Processing (APP MTH 4043): lecture 10 – p.42/75

Daubechies wavelets





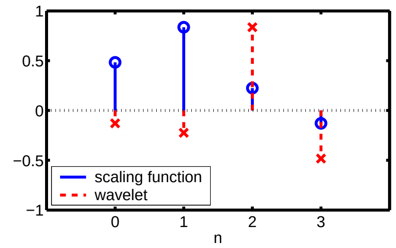
Daubechies wavelet filters

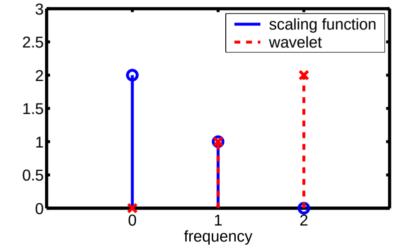
$p = 2$

$$h = \begin{bmatrix} 0.482962913145 \\ 0.836516303738 \\ 0.224143868042 \\ -0.129409522551 \end{bmatrix}$$

$$g(n) = h(2p-n-1)(-1)^n$$

$$g = \begin{bmatrix} -0.129409522551 \\ -0.224143868042 \\ 0.836516303738 \\ -0.482962913145 \end{bmatrix}$$





The figure is similar to that in Mallat, p.253, made using WaveLab `http://www-stat.stanford.edu/~wavelab/`, in particular, see below.

```
% file:       daubechies_2.m, (c) Matthew Roughan, Mon Aug 14 2006
%
opts = struct('height',8, 'width', 8.5, 'Color', 'rgb');
n = 1024;
J = log2(n);
j = 7;
for p=[4 6 8 10]
% for p=[4]
  k = 2^(J-j-1);
  m = MakeWavelet(J-j,k,'Daubechies',p,'Mother',n).*2^(j/2);
  i_m = (((1:n)-n/2)./2^j);
  figure(1)
  p1 = plot(i_m,-m, 'b', 'linewidth', 3);
  set(gca, 'fontsize', 18, 'linewidth', 3);
  title(sprintf('mother wavelet \\psi(t), p = %d', p/2));
  axis([-p/2+1 p/2 -1.5 2])
  exportfig(gcf,sprintf('Plots/daubechies_m_%05d.eps',p), opts, 'format', 'eps');

  f = MakeWavelet(J-j,k,'Daubechies',p,'Father',n).*2^(j/2);
  i_f = (((1:n)-n/2)./2^j)+1;
  figure(2)
  p2 = plot(i_f,f, 'b', 'linewidth', 3);
  set(gca, 'fontsize', 18, 'linewidth', 3);
  title(sprintf('scaling function \\phi(t), p = %d', p/2));
  axis([0 5 -.5 1.5])
  exportfig(gcf,sprintf('Plots/daubechies_f_%05d.eps',p), opts, 'format', 'eps');
end
```


For filter coefficients, see

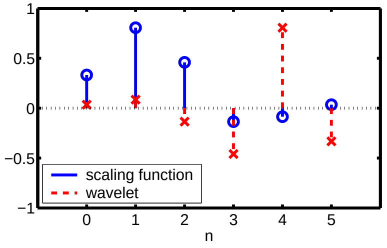
```
function [h,g] = daubechies(p)
% file:       daubechies.m, (c) Matthew Roughan, Sat Oct 16 2004
%
if p==1,
   h=[1/sqrt(2) 1/sqrt(2)];
 %  g=[1/sqrt(2) -1/sqrt(2)];
end
if p==2,
   h(1:2)=[0.482962913145 0.836516303738];
   h(3:4)=[0.224143868042 -0.129409522551];
end
if p==3,
   h(1:2)=[0.332670552950 0.806891509311];
   h(3:4)=[0.459877502118 -0.135011020010];
   h(5:6)=[-0.085441273882 0.035226291882];
end
if p==4,
   h(1:2)=[0.230377813309 0.714846570553];
   h(3:4)=[0.630880767930 -0.027983769417];
   h(5:6)=[-0.187034811719 0.030841381836];
   h(7:8)=[0.032883011667 -0.010597401785];
end
if p==5,
   h(1:2)=[0.160102397974 0.603829269797];
   h(3:4)=[0.724308528438 0.138428145901];
   h(5:6)=[-0.242294887066 -0.032244869585];
   h(7:8)=[0.077571493840 -0.006241490213];
   h(9:10)=[-0.012580751999 0.003335725285];
end
if p==6,
   h(1:2)=[0.111540743350 0.494623890398];
   h(3:4)=[0.751133908021 0.315250351709];
   h(5:6)=[-0.226264693965 -0.129766867567];
   h(7:8)=[0.097501605587 0.027522865530];
   h(9:10)=[-0.031582039318 0.000553842201];
   h(11:12)=[0.004777257511 -0.001077301085];
end
```

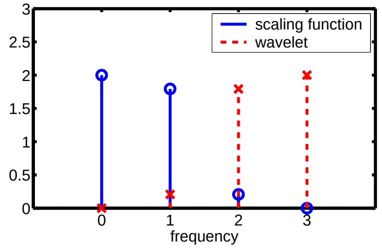
Daubechies wavelet filters

$p = 3$

$$h = \begin{bmatrix} 0.332670552950 \\ 0.806891509311 \\ 0.459877502118 \\ -0.135011020010 \\ -0.085441273882 \\ 0.035226291882 \end{bmatrix}$$

$$g(n) = h(2p - n - 1)(-1)^n$$



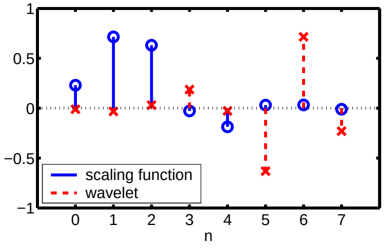


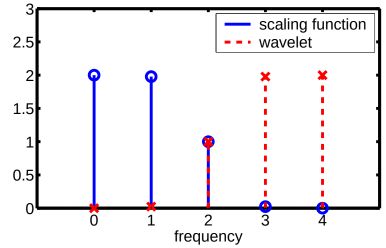
Daubechies wavelet filters

$p = 4$

$$h = \begin{bmatrix} 0.230377813309 \\ 0.714846570553 \\ 0.630880767930 \\ -0.027983769417 \\ -0.187034811719 \\ 0.030841381836 \\ 0.032883011667 \\ -0.010597401785 \end{bmatrix}$$

$$g(n) = h(2p - n - 1)(-1)^n$$





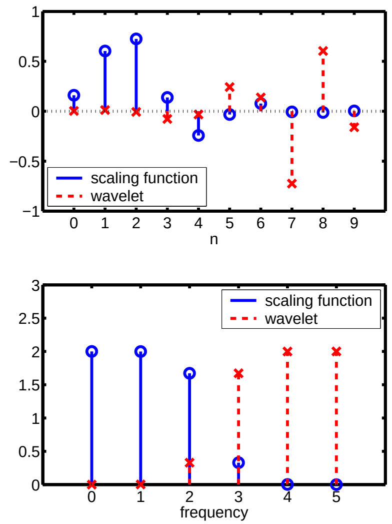
Transform Methods & Signal Processing (APP MTH 4043): lecture 10 – p.45/75

Transform Methods & Signal Processing (APP MTH 4043): lecture 10 – p.46/75

Daubechies wavelet filters

$p = 5$

$$h = \begin{bmatrix} 0.160102397974 \\ 0.603829269797 \\ 0.724308528438 \\ 0.138428145901 \\ -0.242294887066 \\ -0.032244869585 \\ 0.077571493840 \\ -0.006241490213 \\ -0.012580751999 \\ 0.003335725285 \end{bmatrix}$$

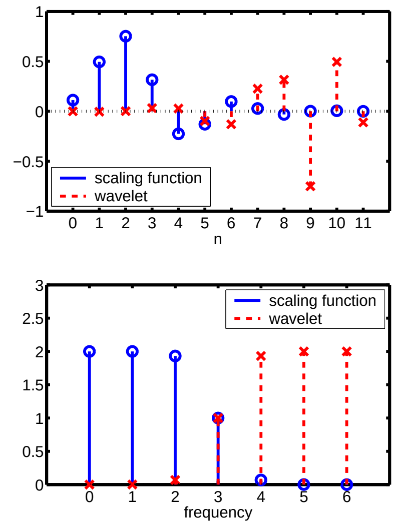


$$g(n) = h(2p-n-1)(-1)^n$$

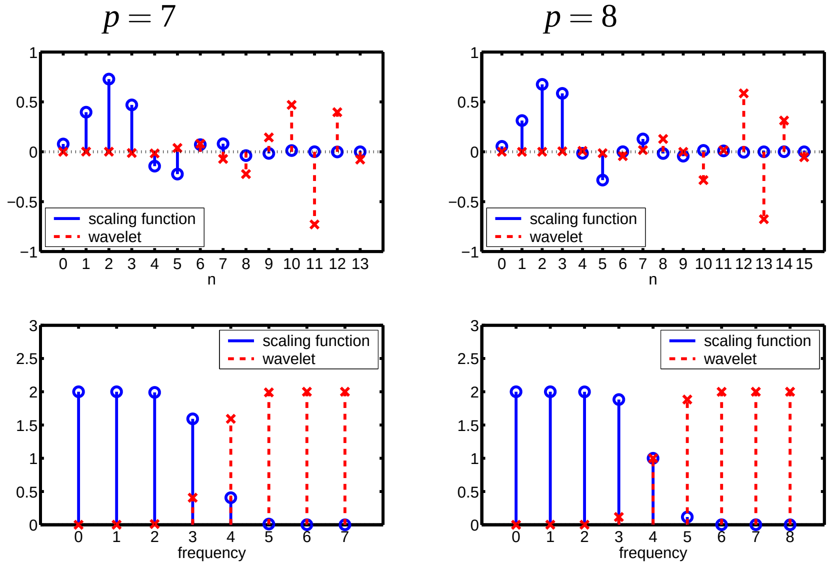
Daubechies wavelet filters

$p = 6$

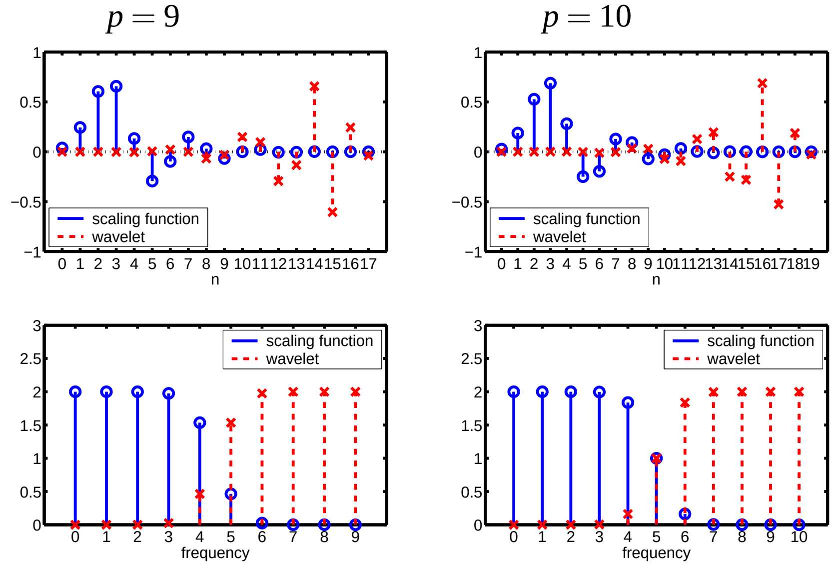
$$h = \begin{bmatrix} 0.111540743350 \\ 0.494623890398 \\ 0.751133908021 \\ 0.315250351709 \\ -0.226264693965 \\ -0.129766867567 \\ 0.097501605587 \\ 0.027522865530 \\ -0.031582039318 \\ 0.000553842201 \\ 0.004777257511 \\ -0.001077301085 \end{bmatrix}$$



Daubechies wavelet filters



Daubechies wavelet filters



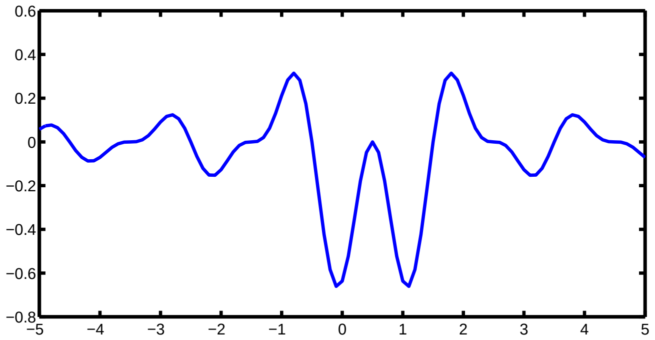
Transform Methods & Signal Processing (APP MTH 4043): lecture 10 – p.49/75

Transform Methods & Signal Processing (APP MTH 4043): lecture 10 – p.50/75

Shannon wavelets

Based on Shannon MRA. Finite support on Fourier domain, so infinite support in time domain.

$$\phi(t) = \frac{\sin(\pi t)}{\pi t}, \quad \psi(t) = \frac{\sin(2\pi(t-1/2))}{2\pi(t-1/2)} - \frac{\sin(\pi(t-1/2))}{\pi(t-1/2)} -$$



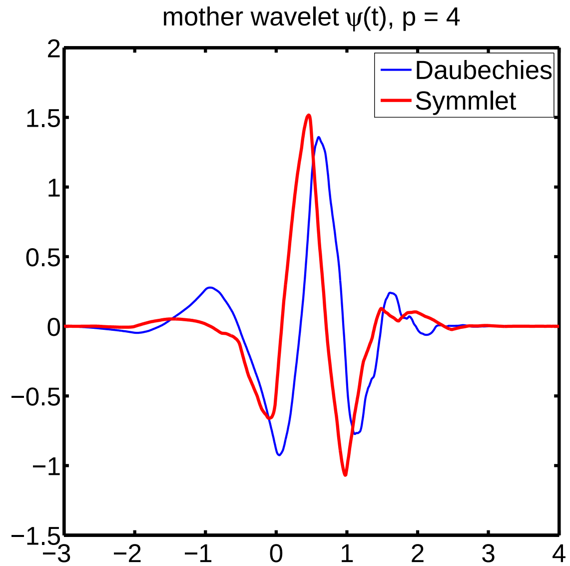
Other wavelets

- Meyer wavelets: like Shannon but FT is smoother, so wavelet and scaling function decay faster.
- Battle-Lemarie wavelets: derived from polynomial splines approximations.
- Mexican hat wavelets: second derivative of a Gaussian (also infinite support).
- Daubechies wavelets symmlets
  - Daubechies wavelets highly asymmetric
  - Haar filter is only real, compactly supported filter with linear phase (symmetry)
  - Symmlets are closest you can get to symmetric for p vanishing moments.
  - Also called Daubechies Least Asymmetric wavelets
- Minimum Bandwidth Discrete-Time (Morris and Pervali)
  - improve approximation to ideal band-pass

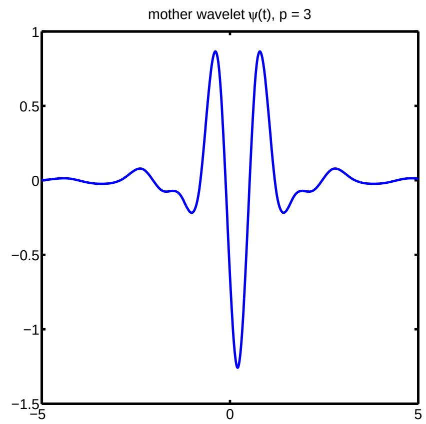
Transform Methods & Signal Processing (APP MTH 4043): lecture 10 – p.51/75

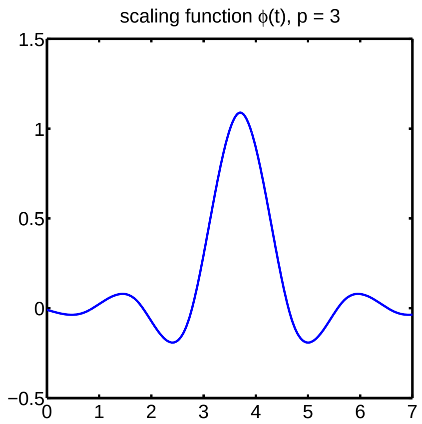
Transform Methods & Signal Processing (APP MTH 4043): lecture 10 – p.52/75

Symmlets



Battle-Lemarié





Cubic spline multiresolution approximation based wavelets.

Made using WaveLab `http://www-stat.stanford.edu/~wavelab/`

```
% file:       symmlets.m, (c) Matthew Roughan, Mon Aug 14 2006
%
opts = struct('height',8, 'width', 8.5, 'Color', 'rgb');
n = 1024;
J = log2(n);
j = 7;
for p=[8 10 16]
  k = 2^(J-j-1);
  m1 = MakeWavelet(J-j,k,'Daubechies',p,'Mother',n).*2^(j/2);
  m2 = MakeWavelet(J-j,k,'Symmlet',p/2,'Mother',n).*2^(j/2);
  i_m = (((1:n)-n/2)./2^j);
  figure(1)
  hold off
  p1 = plot(i_m,-m1, 'b', 'linewidth', 2);
  hold on
  p2 = plot(i_m,-m2, 'r', 'linewidth', 3);
  set(gca, 'fontsize', 18, 'linewidth', 3);
  legend([p1 p2], 'Daubechies', 'Symmlet');
  title(sprintf('mother wavelet \\psi(t), p = %d', p/2));
  axis([-p/2+1 p/2 -1.5 2])
  exportfig(gcf,sprintf('Plots/symmlet_%05d.eps',p), opts, 'format', 'eps');

  pause
end

% interestingly, when p for symmlets=2p for Daubechies, they look similar
```


Made using WaveLab `http://www-stat.stanford.edu/~wavelab/`

```
% file:       battle_lemarie.m, (c) Matthew Roughan, Mon Aug 14 2006
%
opts = struct('height',8, 'width', 8.5, 'Color', 'rgb');
n = 1024;
J = log2(n);
j_i = [5 6];
p_i = [1 3];
for i=1:length(pi)
  j = j_i(i);
  p = p_i(i);
  k = 2^(J-j-1);
  m = MakeWavelet(J-j,k,'Battle',p,'Mother',n).*2^(j/2);
  i_m = (((1:n)-n/2)./2^j);
  figure(1)
  p1 = plot(i_m,-m, 'b', 'linewidth', 3);
  set(gca, 'fontsize', 18, 'linewidth', 3);
  title(sprintf('mother wavelet \\psi(t), p = %d', p));
  axis([-5 5 -1.5 1])
  exportfig(gcf,sprintf('Plots/battle_lemarie_m_%05d.eps',p), opts, 'format', 'eps');

  f = MakeWavelet(J-j,k,'Battle',p,'Father',n).*2^(j/2);
  i_f = (((1:n)-n/2)./2^j)+1;
  figure(2)
  p2 = plot(i_f,f, 'b', 'linewidth', 3);
  set(gca, 'fontsize', 18, 'linewidth', 3);
  title(sprintf('scaling function \\phi(t), p = %d', p));
  axis([0 7 -0.5 1.5])
  exportfig(gcf,sprintf('Plots/battle_lemarie_f_%05d.eps',p), opts, 'format', 'eps');
end
```

Applications

Some applications of Wavelets are image compression and edge detection.

Applications

- edge (and anomaly) detection
- motion detection
- de-noising
- compression,
  - FBI fingerprints
  - JPEG 2000

Transform Methods & Signal Processing (APP MTH 4043): lecture 10 – p.55/75

Transform Methods & Signal Processing (APP MTH 4043): lecture 10 – p.56/75

Tonebursts

Victor Wickerhauser has suggested that sound synthesis is a natural use of wavelets.

- approximate the sound of a musical instrument,
- notes decomposed into wavelet packet coefficients.
- Reproducing the note would then require reloading those coefficients into a wavelet packet generator and playing back the result.
- Transient characteristics such as attack and decay can be controlled separately (for example, with envelope generators), or by using longer wave packets and encoding those properties, as well, into each note.

http://www.amara.com/current/wavesoundfun.html

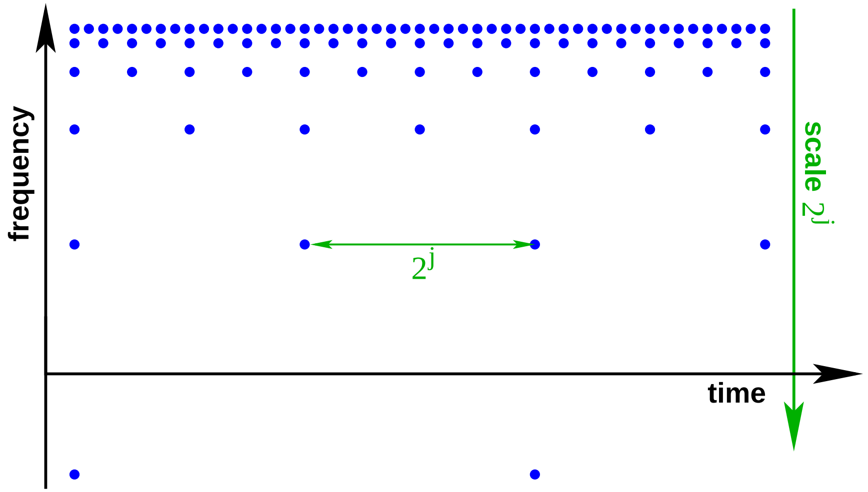
De-noising

Transform a signal $\{x(n)\}_{n\in\mathbb{Z}}$ into wavelet coefficients $\{d(k,j)\}_{k\in\mathbb{Z},1\leq j\leq J}$, and an approximation $\{a(k,j)\}_{k\in\mathbb{Z},j=J}$.

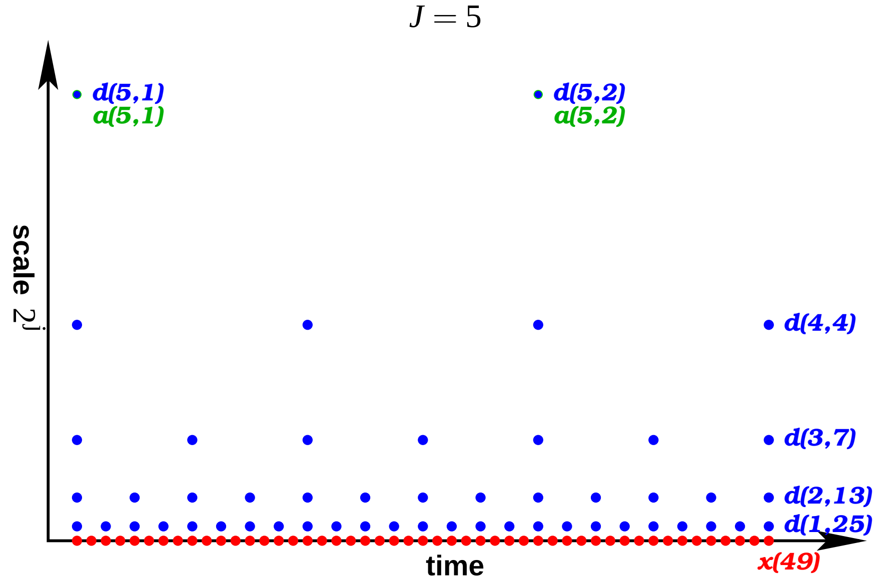
$$
\begin{aligned}
d(k,j) &= \langle x, \psi_{k,j}\rangle \\
a(k,j) &= \langle x, \phi_{k,j}\rangle \\
\psi_{k,j}(n) &= \frac{1}{\sqrt{2^j}}\psi\left(2^{-j}n-k\right) \\
\phi_{k,j}(n) &= \frac{1}{\sqrt{2^j}}\phi\left(2^{-j}n-k\right)
\end{aligned}
$$

Sampled on the dyadic grid.

Dyadic grid



Dyadic grid

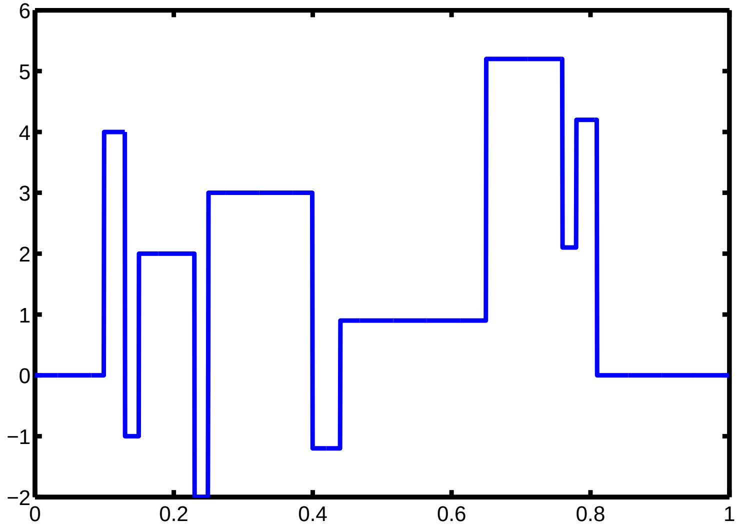


Transform Methods & Signal Processing (APP MTH 4043): lecture 10 – p.59/75

Transform Methods & Signal Processing (APP MTH 4043): lecture 10 – p.60/75

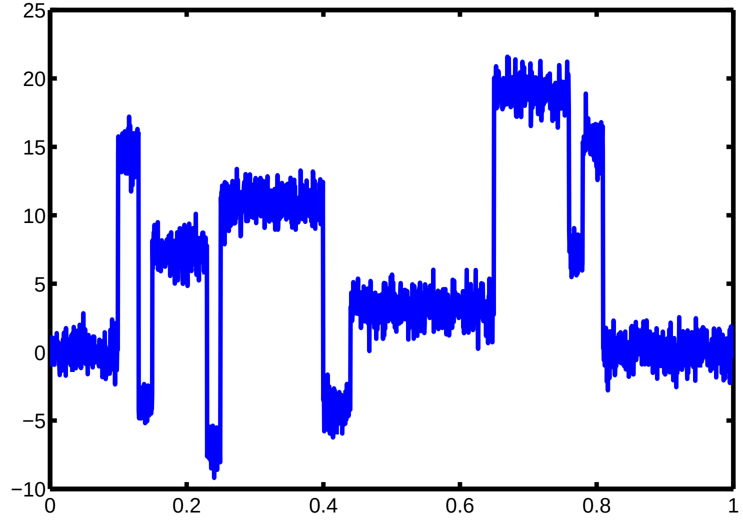
Test signal: Blocks

Test signal: Blocks



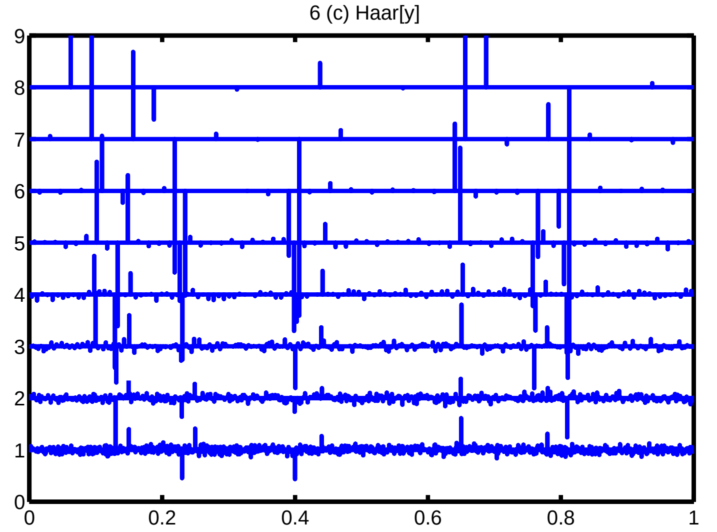
Test signal: Blocks with noise

Test signal: Blocks with noise



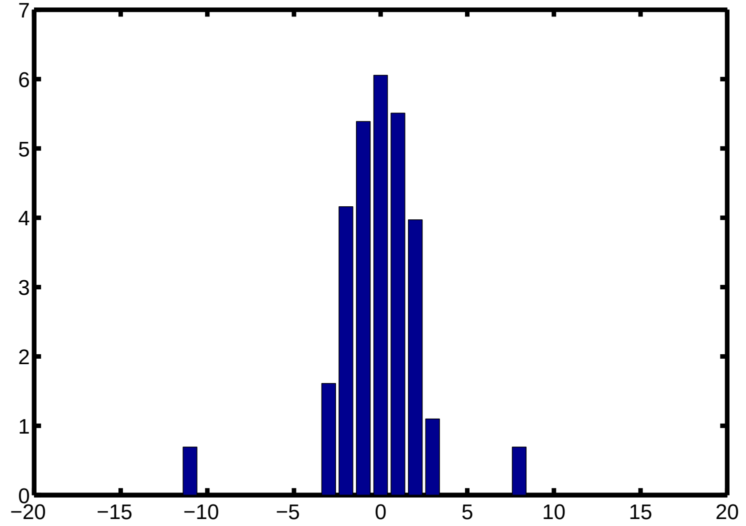
Haar Wavelet transform

Haar Wavelet transform



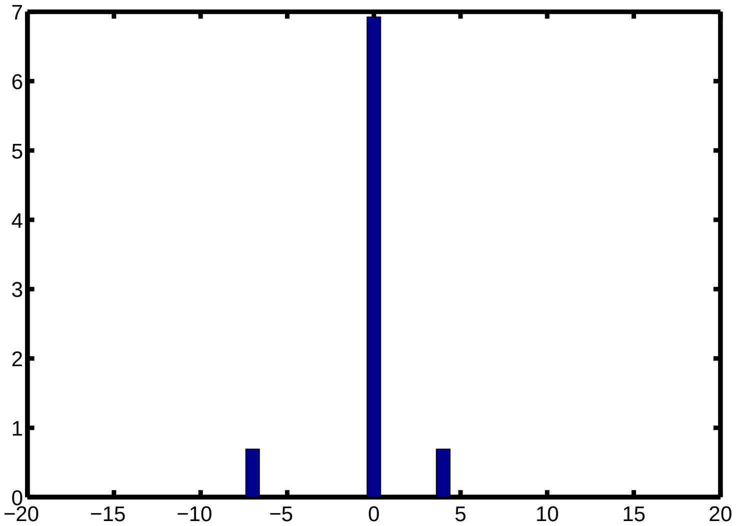
Histogram of details at scale 1

Histogram of details at scale 1



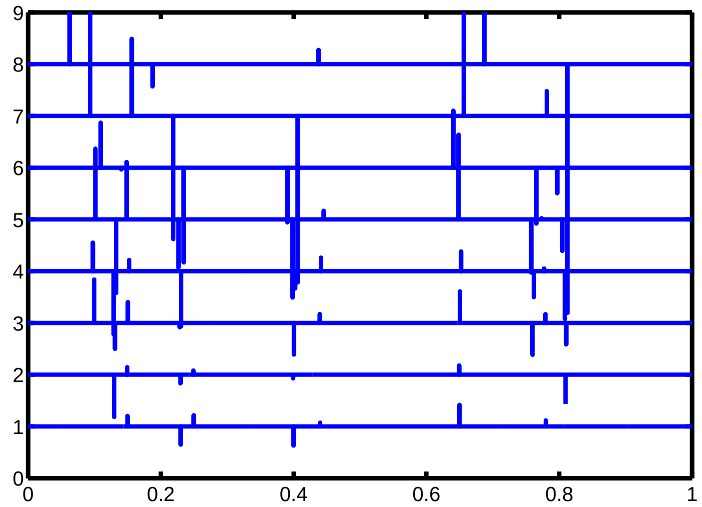
Thresholded details

Thresholded details at scale 1



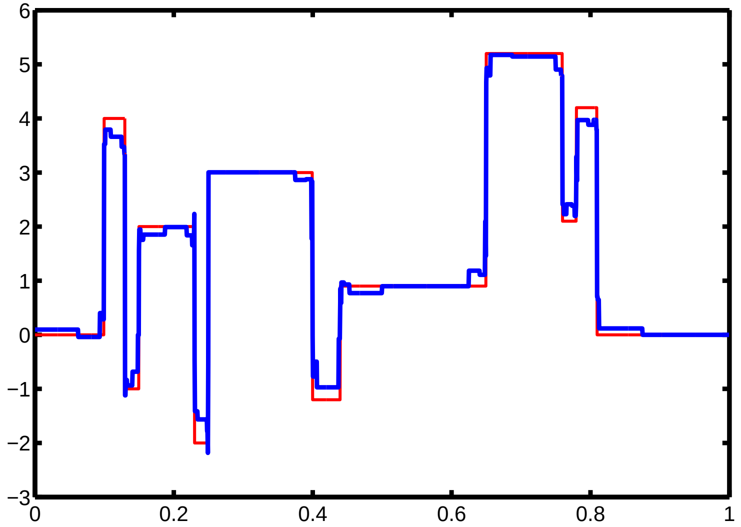
Thresholded transform

Thresholded transform



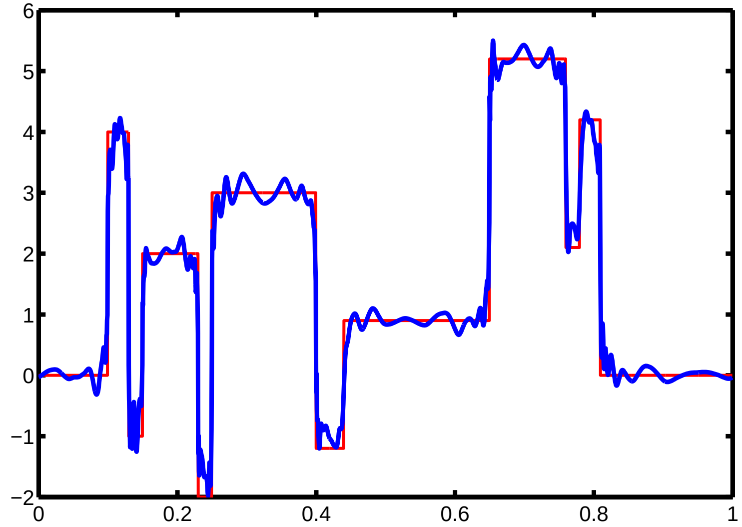
Reconstructed signal

Reconstructed signal



Reconstructed signal

Using smoother wavelets: Symmlets(8)




```
%
% file:       blocks_ex.m, (c) Matthew Roughan, Sun Oct 24 2004
% directory:   /home/mroughan/Classes/Transformations/2004/Matlab/
% created:  Sun Oct 24 2004
% author:   Matthew Roughan
% email:    matthew.roughan@adelaide.edu.au
%
% place help info about BLOCKS_EX here
%
%
clear;

rand('seed', 1);


opengl neverselect;
N = 2048;

sig = MakeSignal('Blocks' , N);
t = (0:(N-1))/N;

figure(1)
plot(t, sig, 'linewidth', 3);
set(gca,'linewidth',3);
set(gca,'fontsize',14);

set(gcf, 'PaperUnits', 'centimeters')
set(gcf, 'PaperOrientation', 'portrait');
set(gcf, 'PaperPosition', [0 0 21 14])
print('-depsc', 'Plots/blocks_ex_1.eps');

rho = 7;
[xblocks,yblocks] = NoiseMaker(sig,rho);
```

De-noising

- Wavelet transform
- Take each scale separately $\{d(j,k)\}_{k\in\mathbb{Z}}$
- (soft) threshold, for $d(j,k) \geq 0$

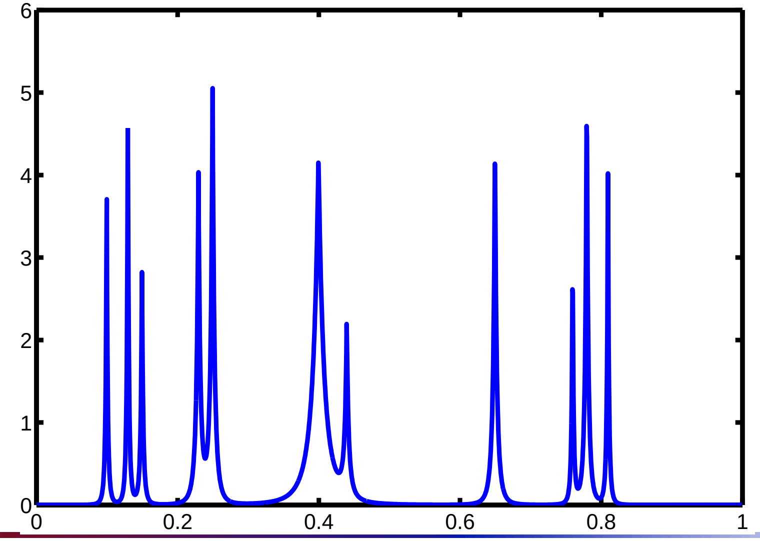
$$\hat{d}(j,k) = \begin{cases} 0, & \text{if } d(j,k) < T_j \\ d(j,k) - T_j & \text{if } d(j,k) \geq T_j \end{cases}$$

  similar approach for $d(j,k) < 0$

- Inverse Wavelet Transform of $\{\hat{d}(j,k)\}_{j=1,\ldots,J,k\in\mathbb{Z}}$ and $\{a(J,k)\}_{k\in\mathbb{Z}}$

Edge Detection

Signal

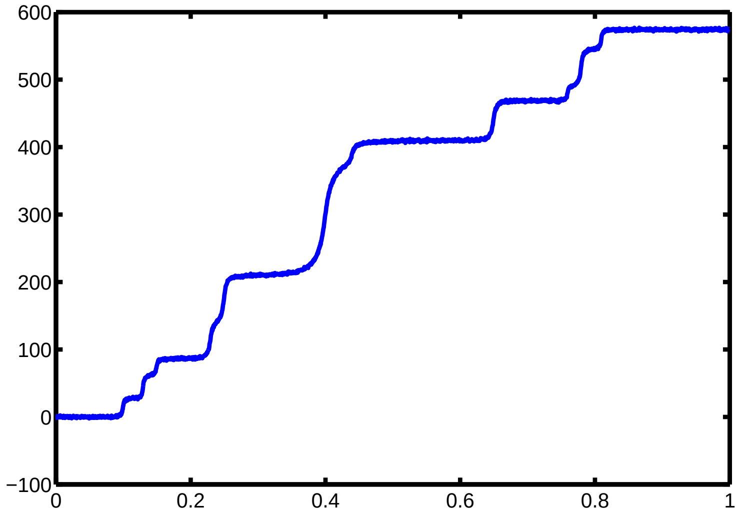


Transform Methods & Signal Processing (APP MTH 4043): lecture 10 – p.69/75

Transform Methods & Signal Processing (APP MTH 4043): lecture 10 – p.70/75

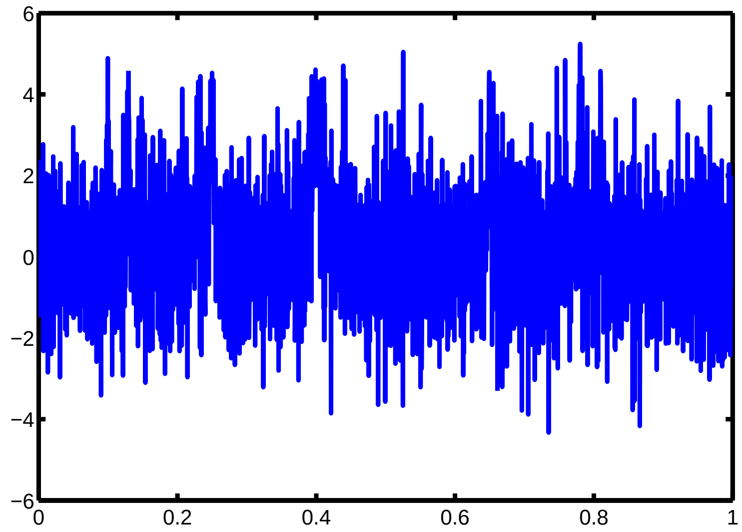
Edge Detection

Cumulative sum + white noise



Edge Detection

Difference filter

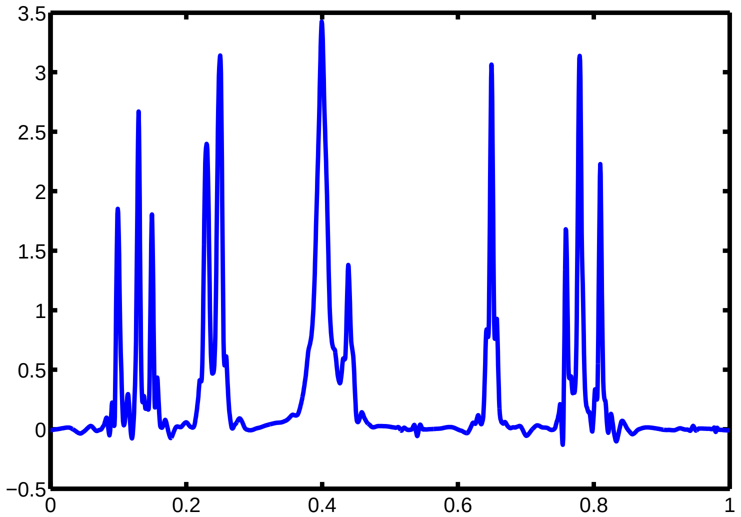


Transform Methods & Signal Processing (APP MTH 4043): lecture 10 – p.71/75

Transform Methods & Signal Processing (APP MTH 4043): lecture 10 – p.72/75

Edge Detection

Wavelet filtered



Edge Detection

Exactly the same algorithm as de-noising.


```
clear;

N = 2048;

Bumps = MakeSignal('Bumps' , N);
t = (0:(N-1))/N;

zBumps = cumsum(Bumps);
N = length(Bumps);
t = (0:(N-1))/N;
x = zBumps + WhiteNoise(zBumps);
z = diff([0 x]);
%
QMF = MakeONFilter('Coiflet',3);
w    = FWT_PO(z, 5,QMF);
ws   = InvShrink(w,5,4,1);
zrec = IWT_PO(ws,5,QMF);
%
wb = FWT_PO(Bumps,5,QMF);
%
clf;

figure(1)
plot(t,Bumps,'linewidth',3);

set(gca,'linewidth',3);
set(gca,'fontsize',14);

set(gcf, 'PaperUnits', 'centimeters')
set(gcf, 'PaperOrientation', 'portrait');
set(gcf, 'PaperPosition', [0 0 21 14])
print('-depsc', 'Plots/edge_detection_1.eps');

figure(2)
```

Transform Methods & Signal Processing (APP MTH 4043): lecture 10 – p.73/75

Transform Methods & Signal Processing (APP MTH 4043): lecture 10 – p.74/75

Code

You will have noticed that I made frequent use of
`http://stat.stanford.edu/~wavelab/`

Transform Methods & Signal Processing (APP MTH 4043): lecture 10 – p.75/75