Wavelets

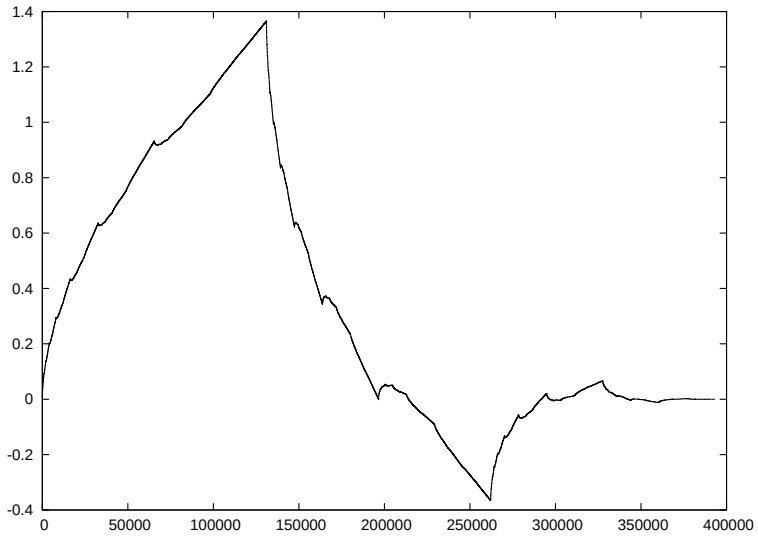
The_Nihilant

June 24, 2011

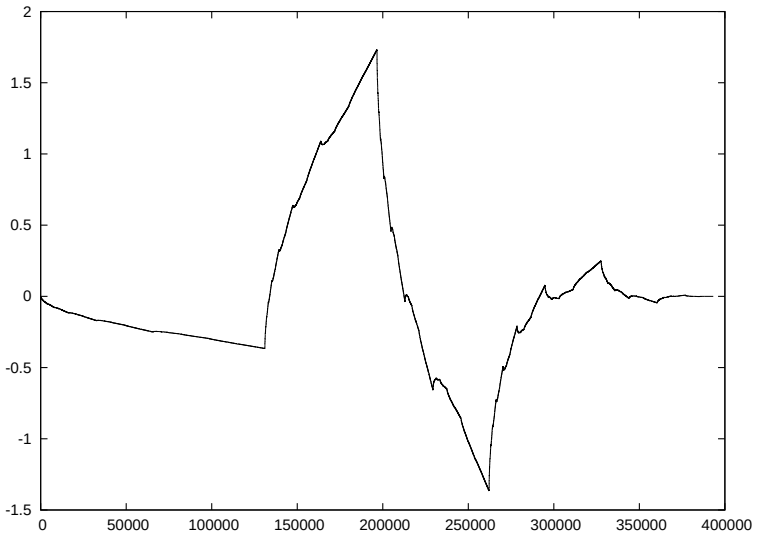
Index

What is a wavelet

This is a wavelet:



(a) Scaling function (ϕ)



(b) Wavelet function (ψ)

Figure: The Daubechies 4 wavelet

How are defined

Both functions have compact support

The scaling function is the solution of the refining equation or dilation equation:

$$\phi(t) = 2\sum_{k=0}^{N} h_0(k)\phi(2t-k)$$

The wavelet function is obtained similarly:

$$\psi(t) = 2\sum_{k=0}^{N} h_1(k)\phi(2t-k)$$

where $h_0(k)$ e $h_1(k)$ are some coefficients which depend on the wavelet.

Formula or coefficients

For many wavelets, the closed form (formula) is available, for other wavelets, like Daubechies wavelets, only coefficients are available.

Wavelet transform

The wavelet transform is similar to the Fourier transform.

Instead of using the various $e^{-i\omega 2\pi}$ as basis, $\phi(t-k)$ and $\psi(2t-k)$ are used, that is, translations and stretchings of those functions.

Wavelets and Fourier

The advantage of the wavelet transform is that it gives better accuracy in the localisation of the frequencies in time.

Heisenberg's uncertainity principle says that:

$$\sigma_t \sigma_\omega \geq \frac{1}{2}$$

where σ_t is the uncertainity in time and σ_ω is the uncertainity in frequencies.

The wavelet transform allows to get closer to that limit.

Fast Wavelet Transform

In practice, projecting the function onto the wavelets is the same as applying a couple of filters, a high-pass (coefficients h_1 of the wavelet) and a low-pass (coefficients h_0 of the scaling function).

Once the filters are applied, the two signals can be downsampled, in order to obtain the same number of samples as the input.

Recontructing

In order to reconstruct the original signal

1. upsample
2. filter the low-pass coefficients with an appropriate dual filter obtained from ϕ and ψ
3. filter the high-pass coefficients with an appropriate dual filter obtained from ϕ and ψ
4. add

In the case of orthogonal wavelers (like the Daubechies wavelets), transformation and reconstruction filters are the same (except for normalizations).

Dyadic wavelet

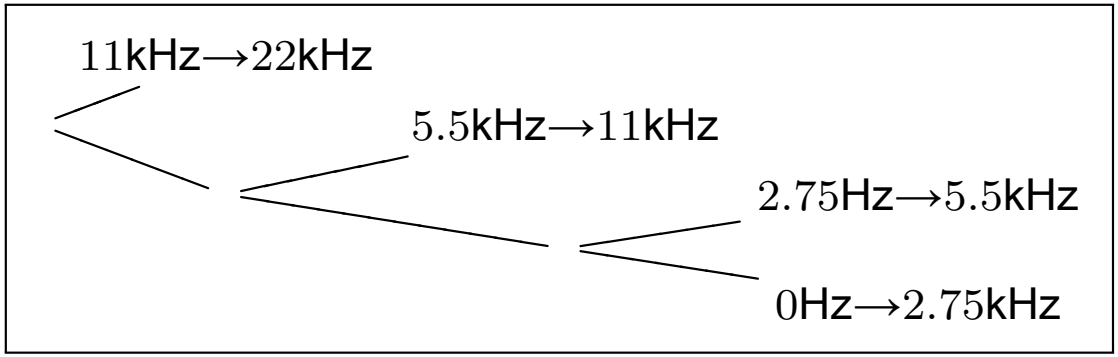


Figure: An example of a 3-stage dyadic wavelet applied to a signal sampled at 44100Hz.

Complexity

It's easy to prove that if the number of stages of the wavelet is fixed, then the complexity of the Fast Wavelet Transform is $O(n)$.

For example, for a simple dyadic wavelet transform, $\sum_{n=0}^{p-1} \frac{n}{2^n} = 2n = O(n)$ operations are performed.

Change of basis matrix

Matrix of the low-pass (2×2):

$$\begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}$$

Matrix of the high-pass:

$$\begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix}$$

Combining the two matrices and subsampling immediately:

$$\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

Let's see the 4×4. Low-pass:

$$\begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \end{pmatrix}$$

High-pass:

$$\begin{pmatrix} 1 & -1 & 0 & 0 \\ 0 & 1 & -1 & 0 \\ 0 & 0 & 1 & -1 \\ -1 & 0 & 0 & 1 \end{pmatrix}$$

Combining the two matrices and subsampling immediately:

$$\begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & -1 \end{pmatrix}$$

Let's pass on the result once more with the 2×2:

$$\begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & -1 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & -1 \end{pmatrix}$$

Let's combine the previous result with the 8×8:

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & -1 & -1 & 0 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \end{pmatrix} =$$

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & 1 & -1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & -1 & -1 \\ 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \end{pmatrix}$$

Description

So, performing a wavelet transform is just applying a couple of filters!

The implementation is trivial, it's just a convolution. Convolution operators can easily be found in math libraries or implemented from scratch.

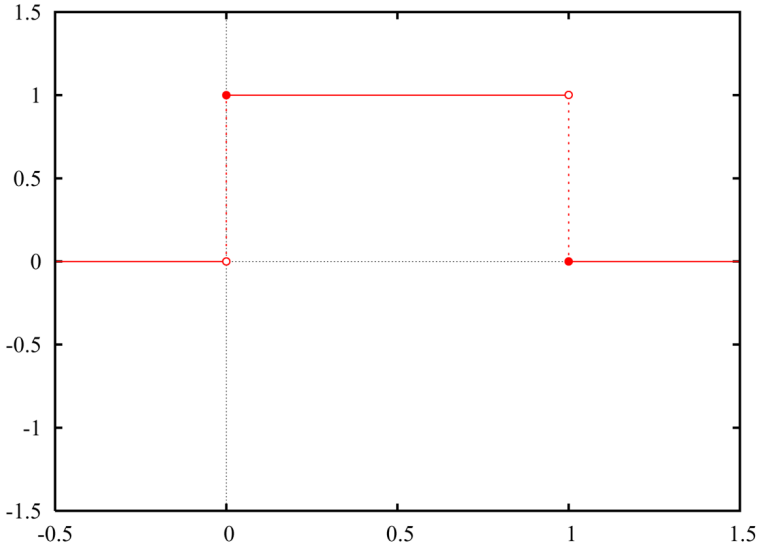
C implementation: FWT

```
static inline void fwt(const int ns, const double*s, const int nw, const double*w,
      const int nx, const double* x, const unsigned int p, double* res){
 double* coarse;
 int cx,dx;

 if(p==0){memcpy(res,x,nx*sizeof(double));return;}
 coarse =calloc((nx/2),sizeof(double));
 for(cx=0;cx<nx/2;cx++) res[cx+nx/2]=0;

 for(cx=0;cx<nx/2;cx++)
  for(dx=0;dx<nw;dx++)
    res[cx+nx/2]+=x[(1+2*cx+dx)%nx]*w[dx];

 for(cx=0;cx<nx/2;cx++)
  for(dx=0;dx<ns;dx++)
    coarse[cx]+=x[(1+2*cx+dx)%nx]*s[dx];

 fwt(ns,s,nw,w,nx/2,coarse,p-1,res);

 free(coarse);
}
```

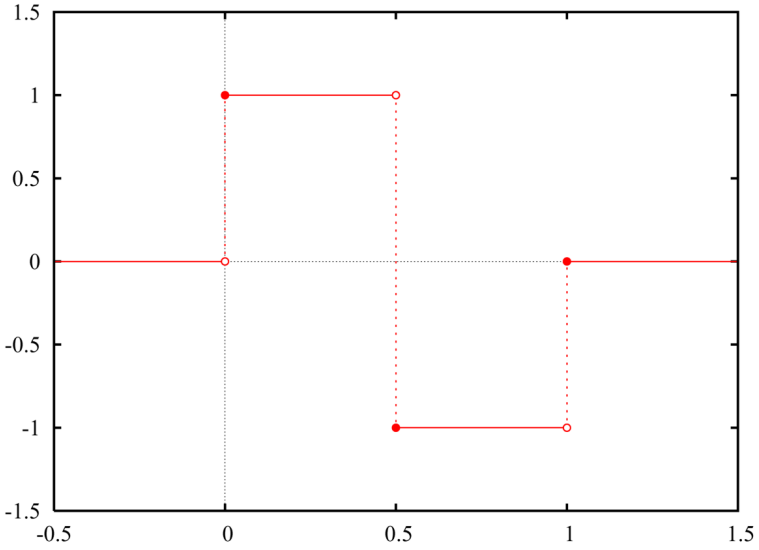
C implementation: Reverse FWT

```
static inline void rfwt(const int ns, const double*s, const int nw, const double*w,
      const int nx, const double* x, const unsigned int p, double* res){
 double* coarse,* details;
 int cx,dx,totdelay = (ns+nw)/2;

 if(p==0){memcpy(res,x,nx*sizeof(double));return;}
 coarse =malloc(nx*sizeof(double));
 details=malloc((nx/2)*sizeof(double));

 rfwt(ns,s,nw,w,nx/2,x,p-1,coarse);

 for(cx=nx/2-1;cx>=0;cx--)coarse[2*cx]=coarse[cx];
 for(cx=1;cx<nx;cx+=2)coarse[cx]=0;

 for(cx=0;cx<nx/2;cx++)details[cx]=x[cx+nx/2];
 for(cx=0;cx<nx;cx++)res[cx]=0;

 for(cx=0;cx<nx;cx++)
  for(dx=cx&1;dx<nw;dx+=2)
   res[(cx+totdelay)%nx]+=details[((cx+dx)/2)%(nx/2)]*w[nw-dx-1];

 for(cx=0;cx<nx;cx++)
  for(dx=cx&1;dx<ns;dx+=2)
   res[(cx+totdelay)%nx]+=coarse[(cx+dx)%nx]*s[ns-dx-1];

 free(coarse); free(details);
}
```

Some wavelets

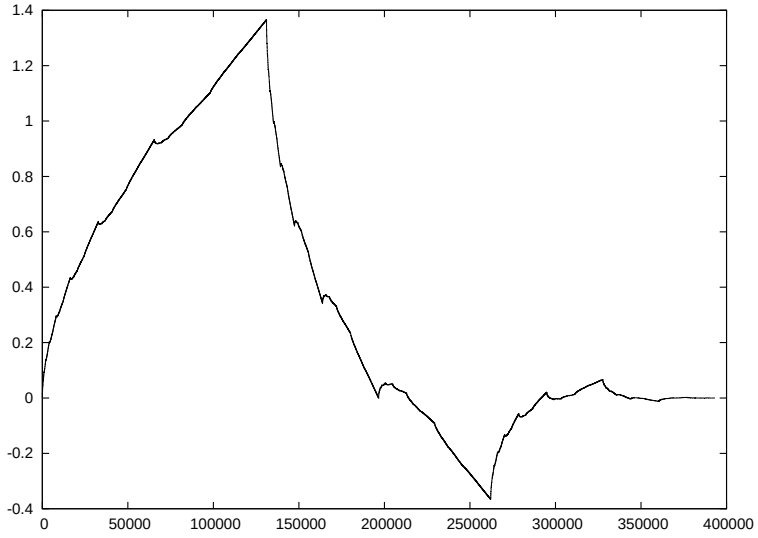


(a) Scaling function

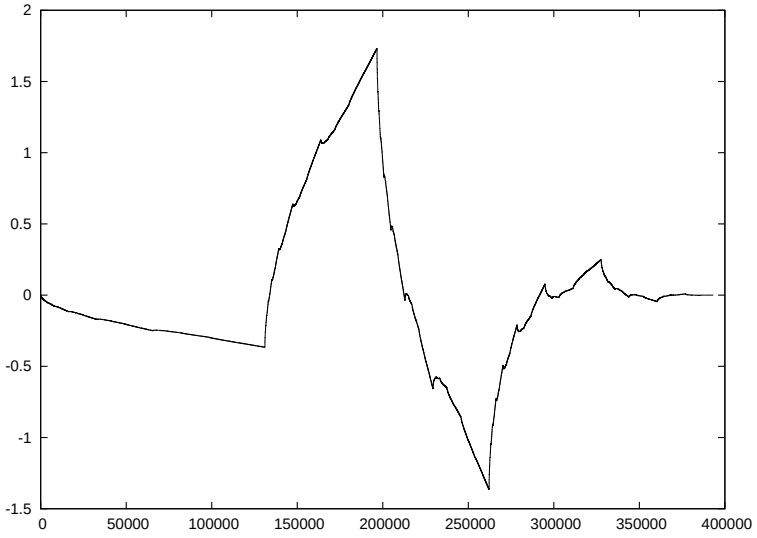


(b) Wavelet function

Figure: Haar Wavelet

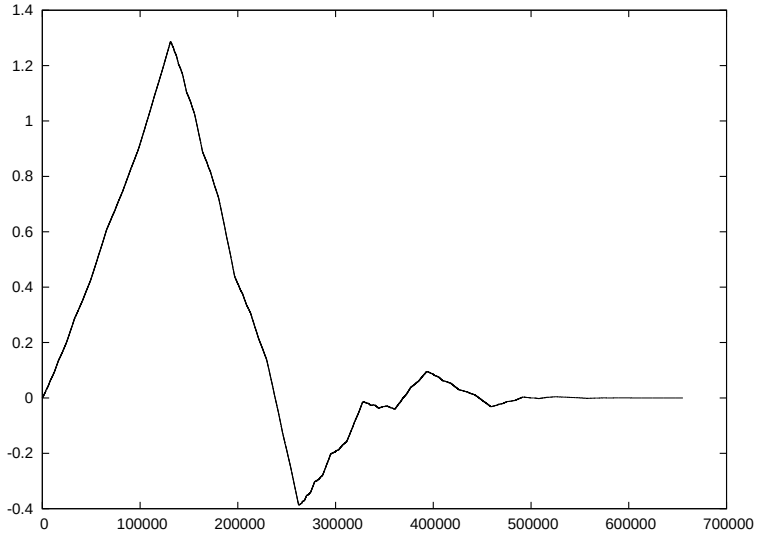


(a) Scaling function

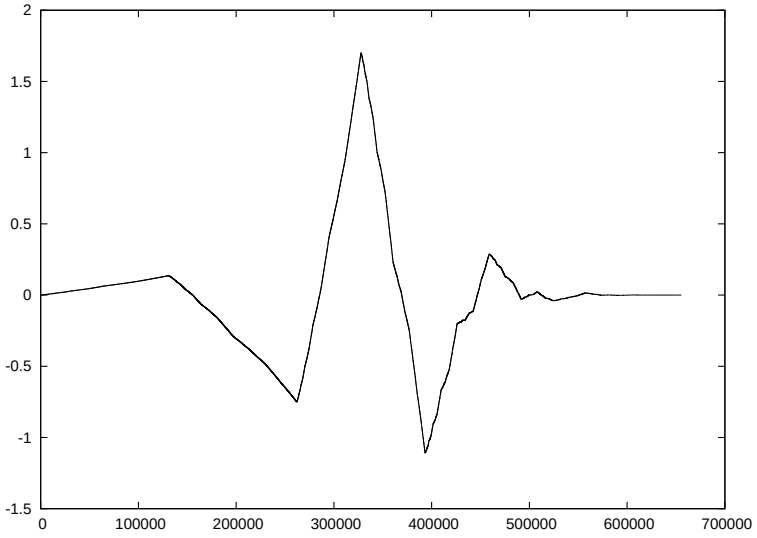


(b) Wavelet function

Figure: Daubechies 4 Wavelet

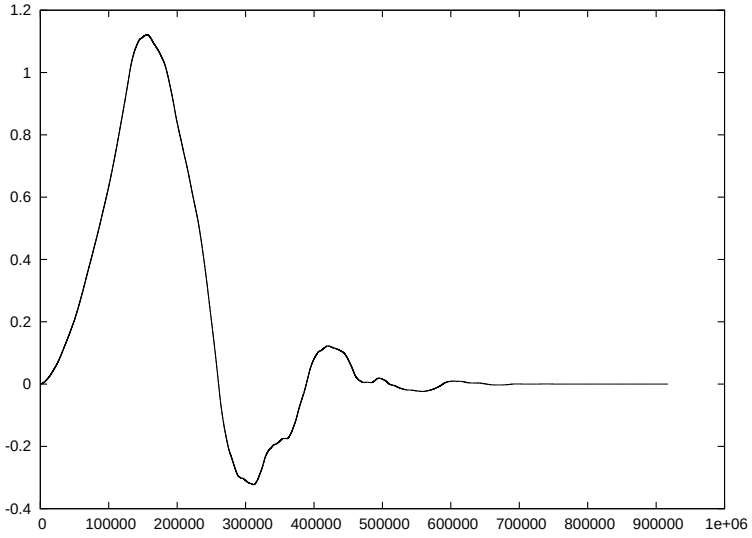


(a) Scaling function

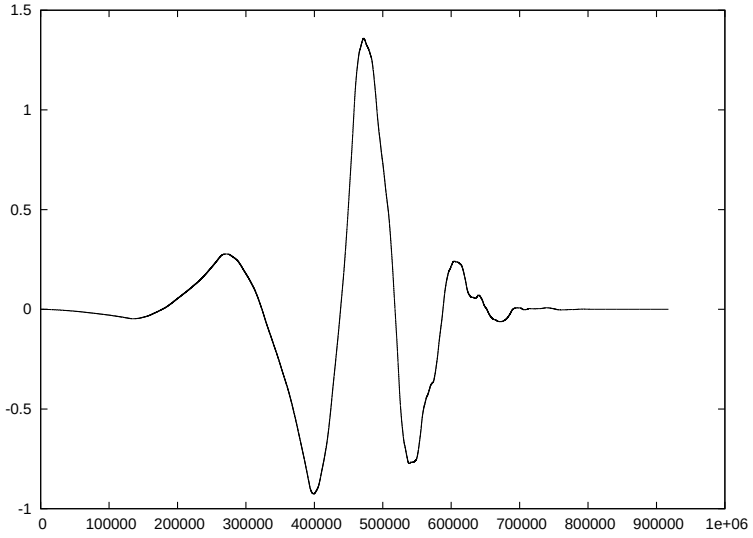


(b) Wavelet function

Figure: Daubechies 6 wavelet

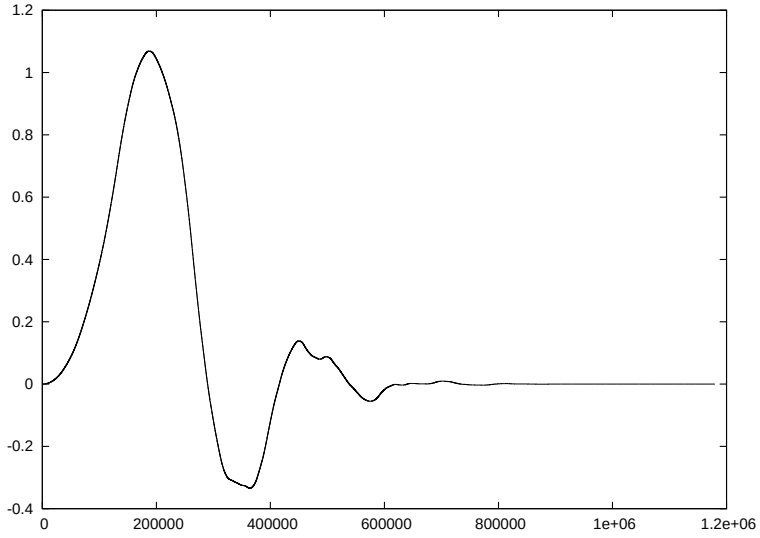


(a) Scaling function

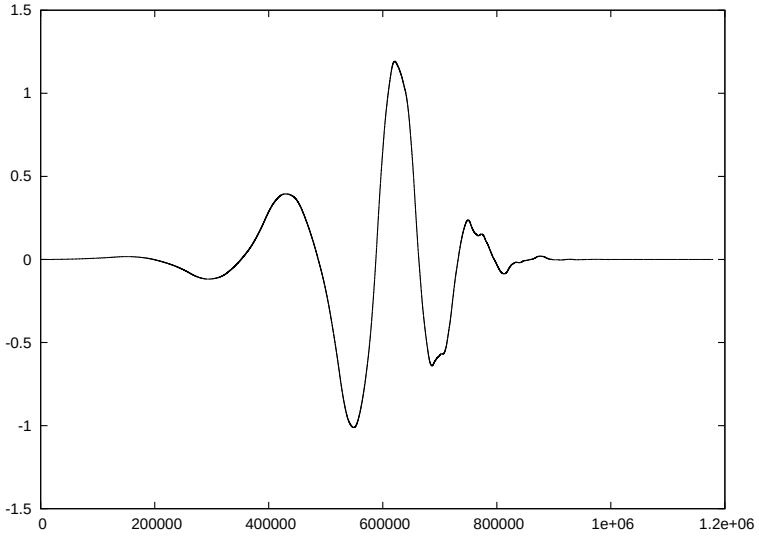


(b) Wavelet function

Figure: Daubechies 8 wavelet

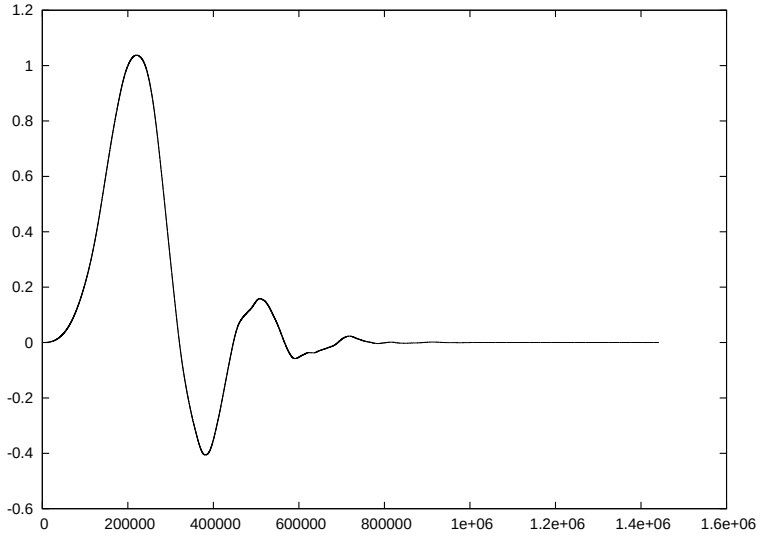


(a) Scaling function

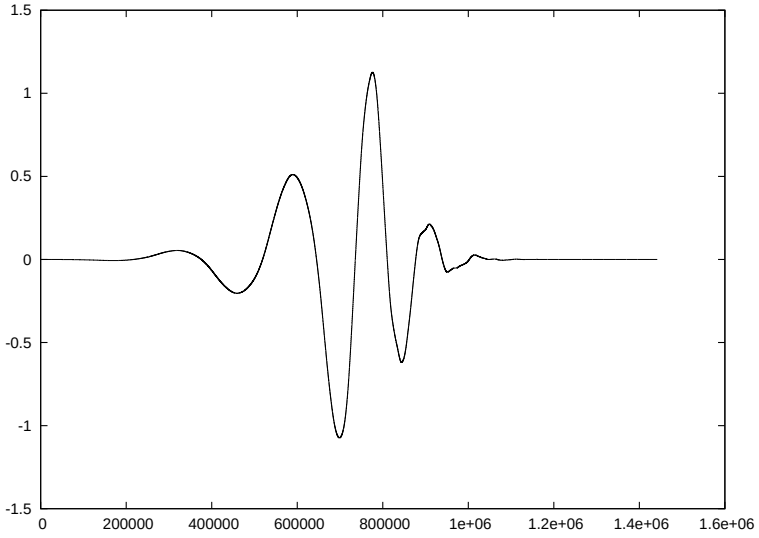


(b) Wavelet function

Figure: Daubechies 10 wavelet

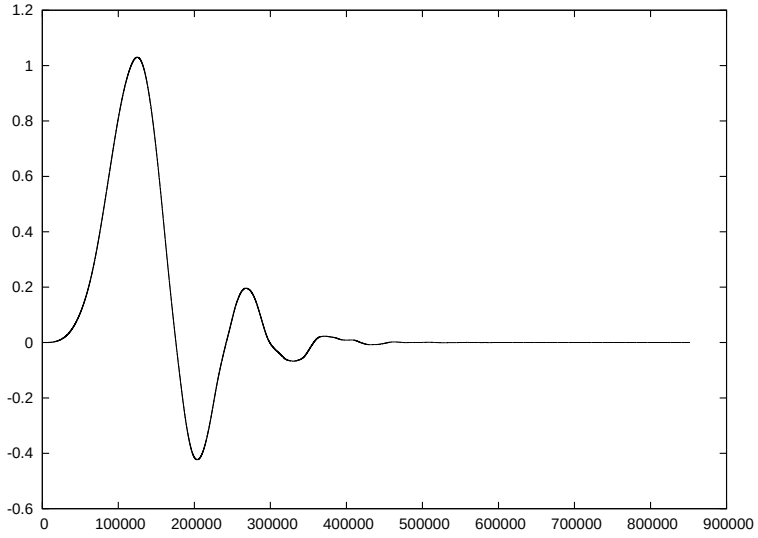


(a) Scaling function

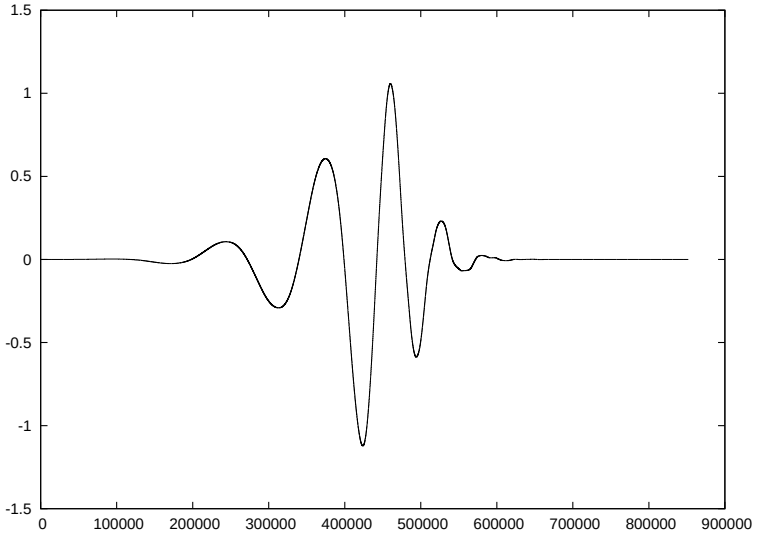


(b) Wavelet function

Figure: Daubechies 12 wavelet



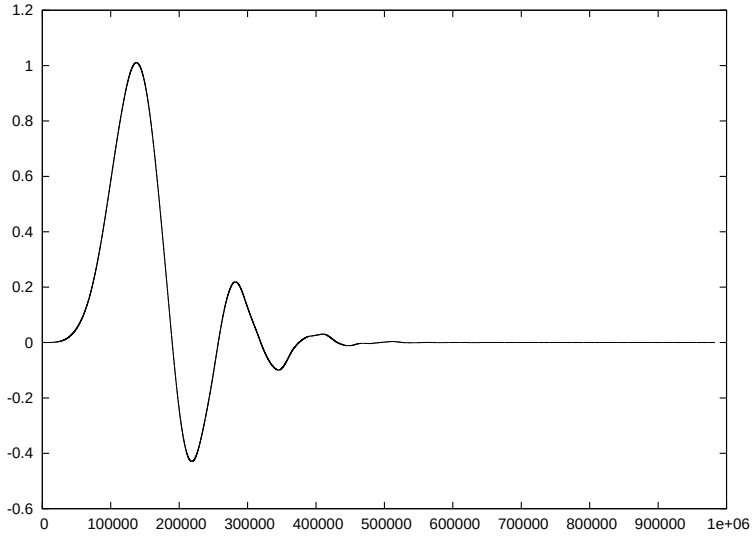
(a) Scaling function



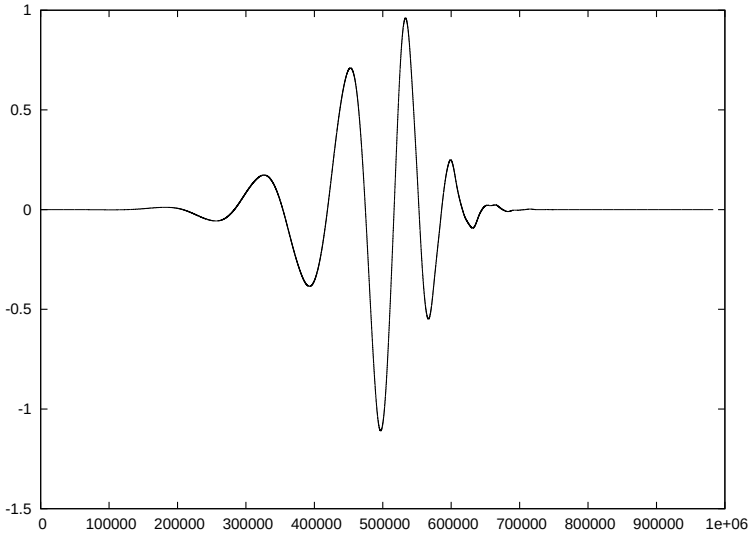
(b) Wavelet function

Figure: Daubechies 14 wavelet

Wavelets



(a) Scaling function



(b) Wavelet function

Figure: Daubechies 16 wavelet