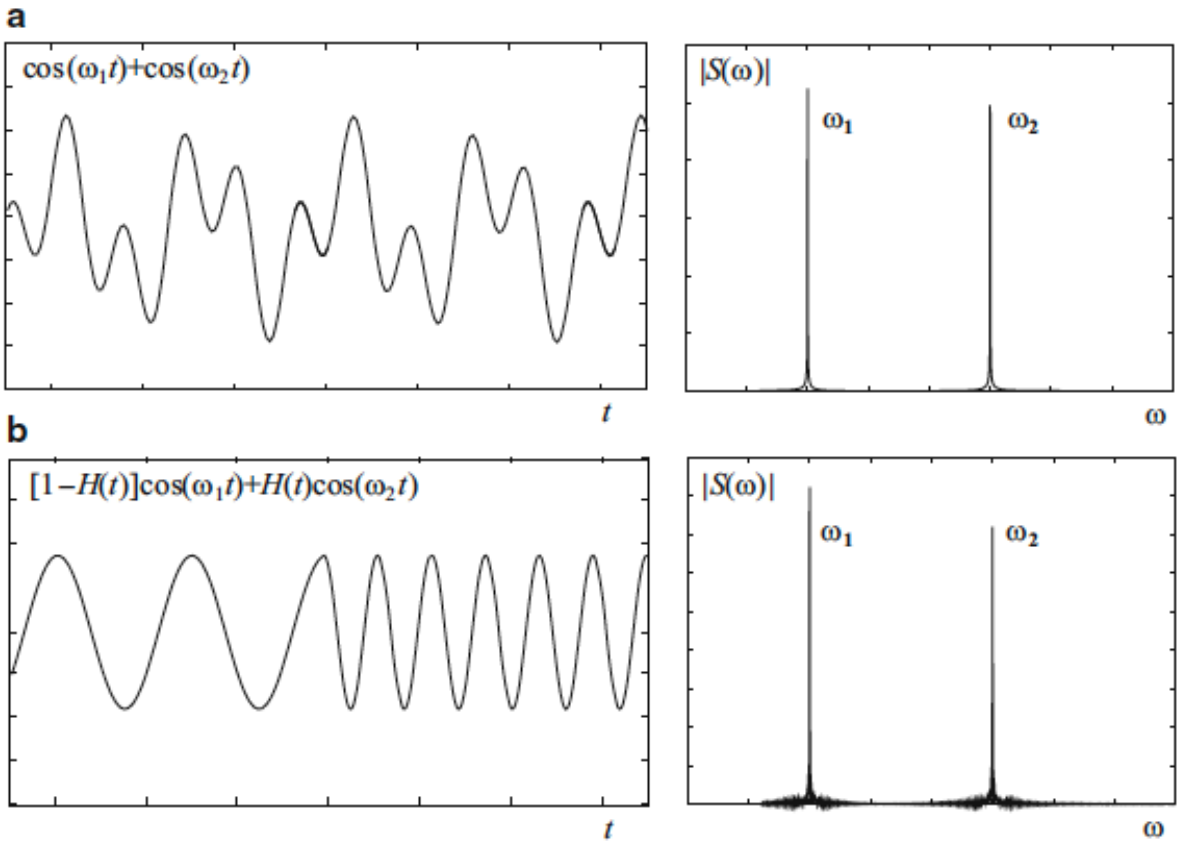
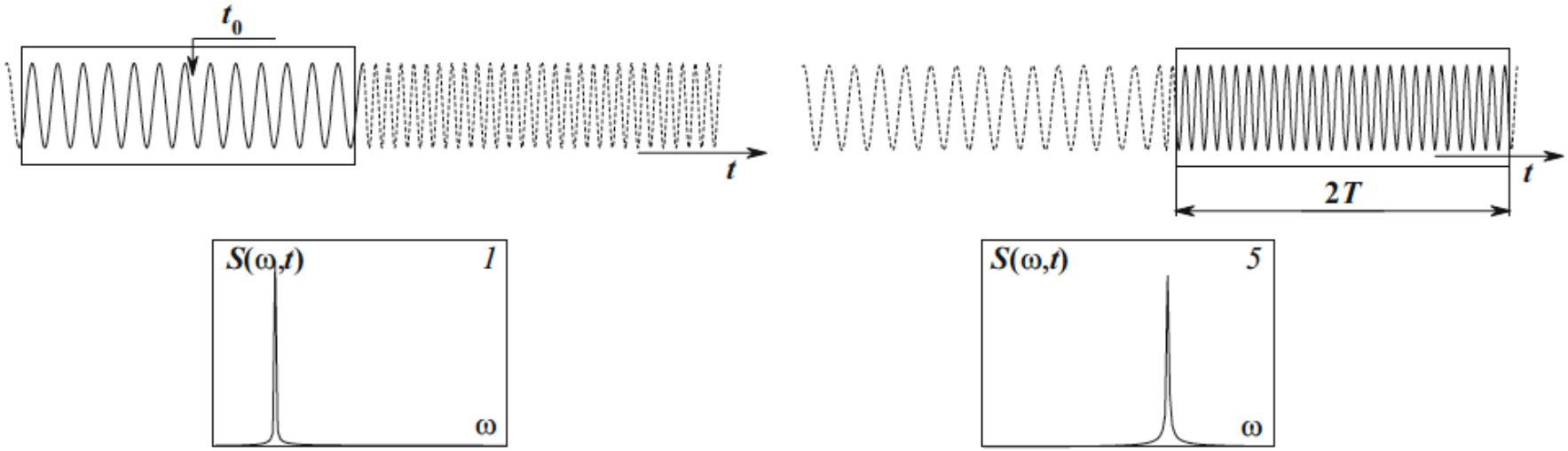
Time-Frequency Analysis & Wavelets

Lecture Notes

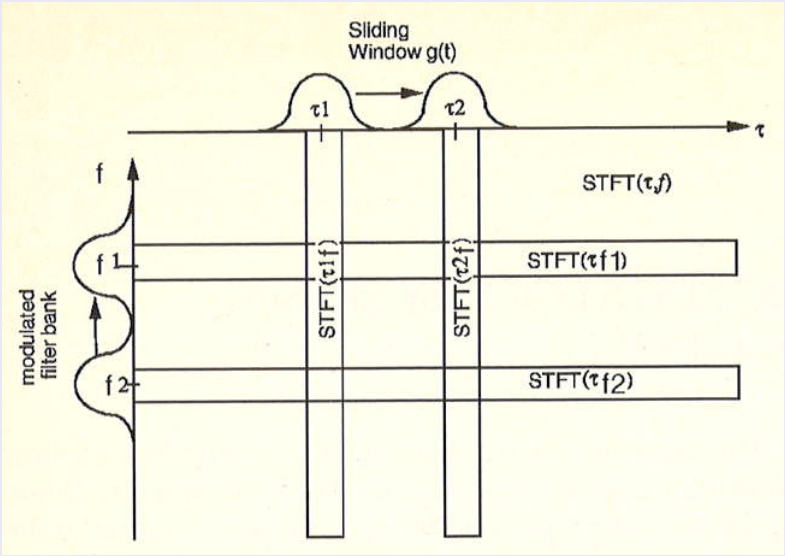
Ahmet Ademoglu, *PhD*
Bogazici University
Institute of Biomedical Engineering


a
cos(ω1t)+cos(ω2t)
t
|S(ω)|
ω1
ω2
ω
b
[1−H(t)]cos(ω1t)+H(t)cos(ω2t)
t
|S(ω)|
ω1
ω2
ω


t_0
t
$2T$
t
$S(\omega,t)$
1
ω
$S(\omega,t)$
5
ω

SHORT-TIME FOURIER TRANSFORM (GABOR)

$$STFT(\tau, f) = \int x(t) g^*(t-\tau) e^{-j2\pi ft} dt$$



- Fourier Transform of a windowed signal
- Filtering the signal with a window modulated at that particular frequency

The Fourier Transform of a Gaussian function is

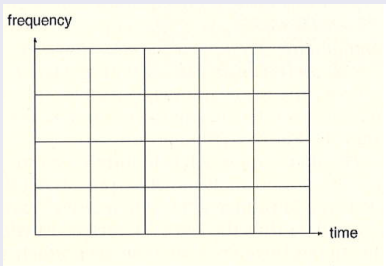
$$\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2}{2\sigma^2}} \overset{\mathcal{F}}{\longleftrightarrow} e^{-\omega^2\sigma^2/2}$$

Heisenberg's Uncertainty Principle

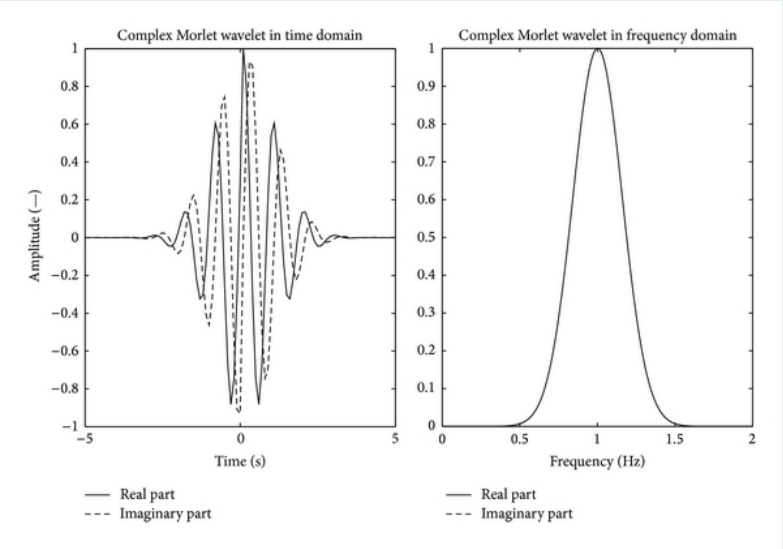
- Bandwidth of a filter is defined as $\Delta f^2 = \frac{\int f^2 |G(f)^2| df}{\int |G(f)^2| df}$
  Two sinusioids are identified if they are Δf apart.
- The spread in time is $\Delta t^2 = \frac{\int t^2 |g(t)^2| dt}{\int |g(t)^2| dt}$
  Two pulses are identified if they are Δt apart.

Time-frequency resolution: Fixed

Time-bandwidth product $= \Delta t \Delta f \geq \frac{1}{4\pi}$



Modulated Gaussian Window: Morlet Wavelet



$w(t|\sigma, t_0, f_0) = e^{-\frac{(t-t_0)^2}{2\sigma^2}} e^{j2\pi f_0 t}$

σ : window length

in time and in frequency domains

t_0 : center of the window in time

f_0 : center of the window in frequency

Determine the σ and f_0 of Morlet function to filter a signal sampled at 100 Hz through a $[15 - 25]$ Hz interval with a FWHM of 10 Hz.

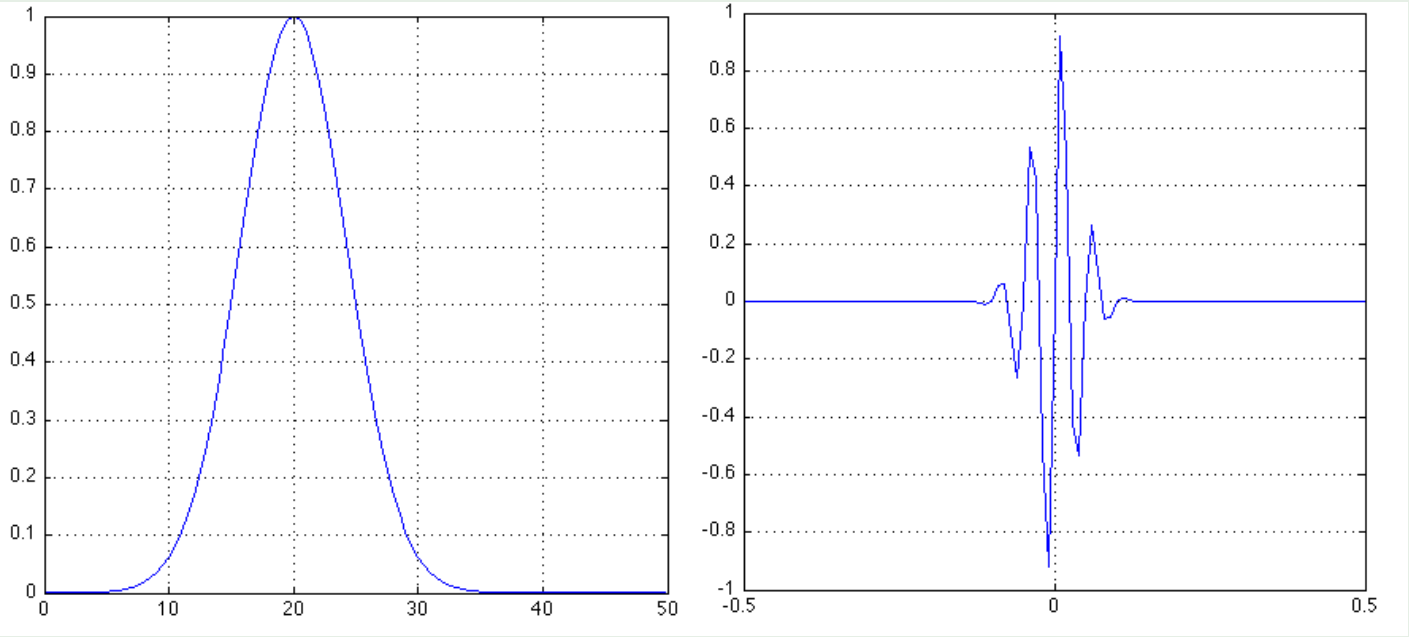
$f_0 = \frac{25+15}{2} = 20$ Hz. $\hat{f} = FWHM/2 = 10/2$ Hz.

The Fourier transform of $w(t|\sigma, f_0)$ is $W(f|\sigma, f_0) = e^{-(2\pi(f-f_0))^2\sigma^2/2}$

$(1/2) = e^{-(2\pi(\hat{f}-f_0))^2\sigma^2/2}/e^{-(2\pi(f_0-f_0))^2\sigma^2/2}$, $(2\pi 5)^2\sigma^2/2 = ln(2)$,

$\sigma^2 = 0.0014$.

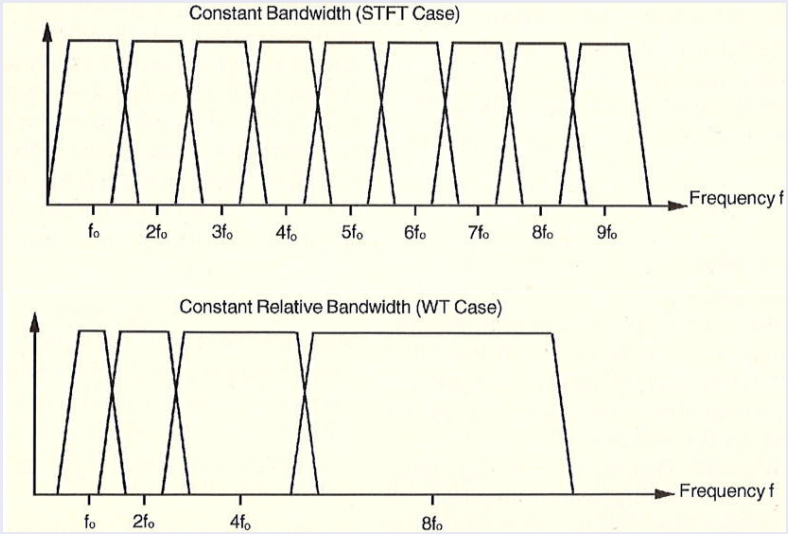
```
t = -.5:0.01:0.5;
s2=0.0014;
w= exp(-1/(2*s2)*t.^2 ).*sin(2*pi*20*t);
f = [0:0.01:1]*pi;
H1=abs(fft(w));
H=freqz(w,1,f);
subplot(1,2,1);plot(f/pi*50,abs(H)/max(abs(H)));grid
subplot(1,2,2);plot(t,w);grid
```



Multiresolution Analysis

To overcome the resolution limitation of STFT

A filter bank with time resolution increasing with the modulating frequency $\frac{\Delta f}{f} = \text{constant}$

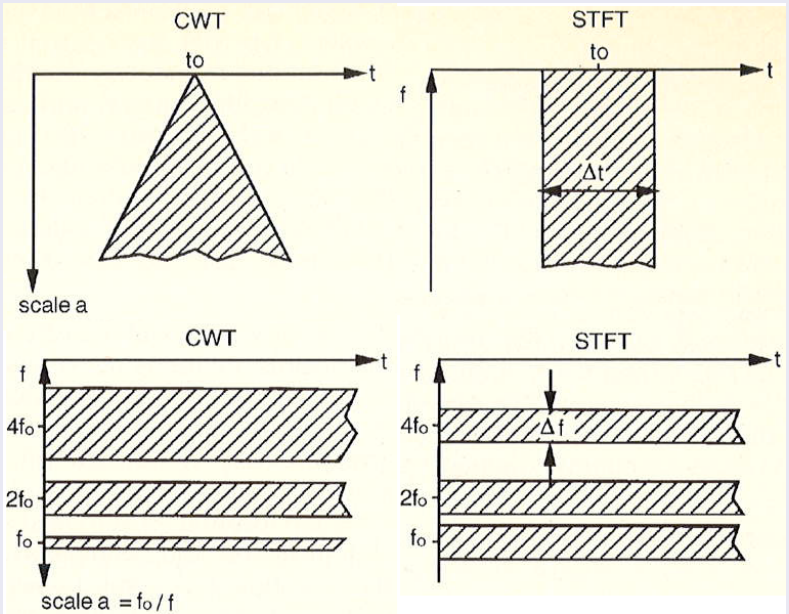


The notion of scale and resolution

Scale : a temporal quantity that qualifies the signal extent

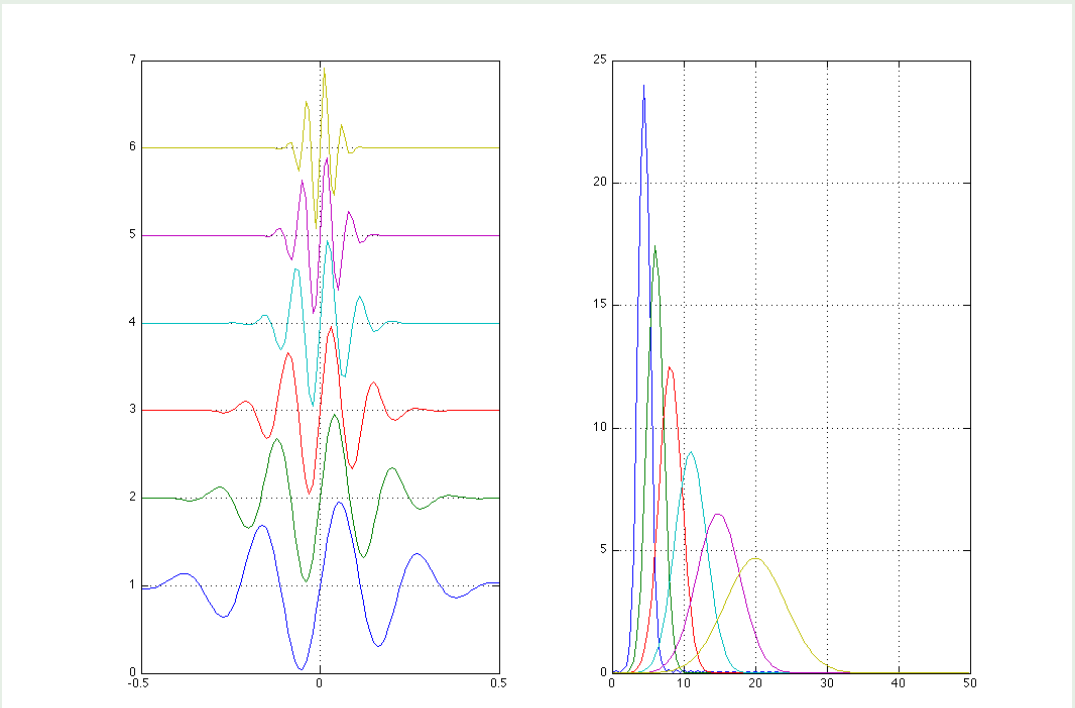
Resolution : a spectral quantity that qualifies the frequency

Scaleogram and Spectrogram



Scaled versions of Morlet filter: σ^2 and f_0 are sampled on a logarithmic interval of 0.1 from scale 0.5 to 1.

```
t = -.5:0.01:0.5;
s2=0.0014;
f = [0:0.01:1]*pi;
S2 = s2.^[0.5:0.1:1];
F2 = 20.^[0.5:0.1:1];
w2 = exp(- (ones(size(S2')))./(2*S2')*t.^2 ).* sin(2*pi* F2'*t) ;
plot(t,w2'+ones(size(w2,2),1)*[1:6] );grid
for i=1:size(w2,1),Hp(i,:)= freqz(w2(i,:),1,f); end;
subplot(1,2,2);plot(f/pi*50,abs(Hp') );grid
```



Continuous Wavelet Transform

Wavalet Analysis

$$CWT_x(\tau, a) = \frac{1}{\sqrt{a}} \int x(t) h^*(\frac{t-\tau}{a}) dt$$

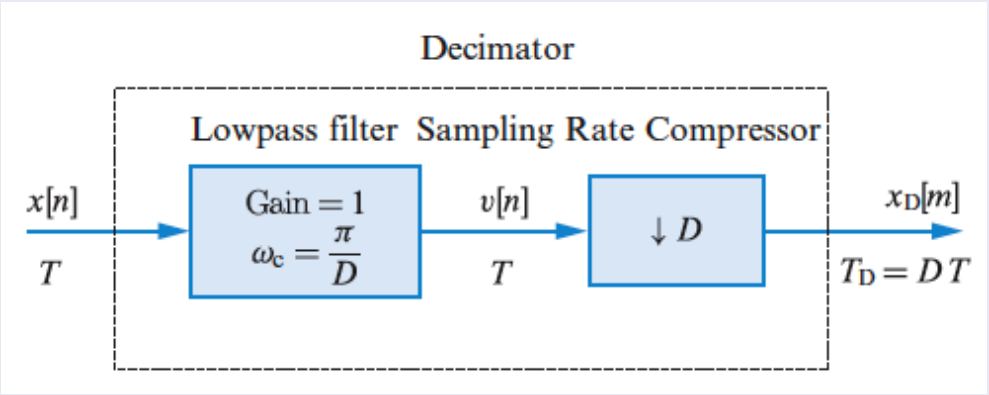
$h(t)$: *Basic Wavelet* which is a band pass filter
a : *Scale Factor* such that $h_{a,\tau}(t) = \frac{1}{\sqrt{a}} h(\frac{t-\tau}{a})$

Wavelet Synthesis

$$x(t) = \frac{c}{a^2} \int_{a>0} CWT_x(\tau, a) h'_{a,\tau}(t) da d\tau$$

$h'_{a,\tau}(t)$: synthesis wavelet

Decimation and Interpolation



$$x_D[m] = v[mD] = \sum_{k=0}^{N} h[k]x[mD-k]$$

The continuous bandlimited signal $x_c(t)$ can be perfectly reconstructed using a sinc function

$$x_c(t) = \sum_{m=-\infty}^{\infty} x_D[m] sin(\frac{\pi(t-mDT)/DT}{\pi(t-mDT)DT})$$

and if we sample $x_c(t)$ at $t = nT$, we can recover $x[n]$ from $x_D[m]$

$$x[n] = \sum_{m=-\infty}^{\infty} x_D[m] sin(\frac{\pi(nT-mDT)/DT}{\pi(nT-MDT)DT})$$

Wavelet Frames and Orthonormal Bases

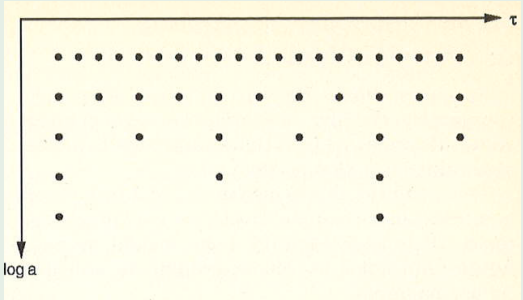
Time-scale parameters are discretized : $a = a_0^j,\ b = k a_0^j T$

Corresponding wavelet function $h_{j,k}(t) = a_0^{-j/2} h(a_0^{-j} t - kT)$

Wavelet Coefficients : $c_{j,k} = \int x(t) h_{j,k}^*(t) dt$

Reconstruction : $x(t) \approx c \sum_j \sum_k c_{j,k} h_{j,k}(t),$

Dyadic Sampling $a_0 = 2,\ a = 2^j,\ b = k 2^j T$



Orthogonal Wavelet Bases

Orthonormality

$\int h_{j,k} h^*_{j',k'} dt = \delta(j - j', k - k')$

The basis functions $h_{j,k}$ are orthonormal to their shifted as well as its scaled versions.

The signal can be perfectly reconstructed as

$x(t) = \sum_{j,k} c_{j,k} h_{j,k}(t)$

Biorthonormality

$$\int h_{j,k} h^*_{j,k'} dt = \delta(k - k')$$

The wavelet functions $h_{j,k}(t)$ are orthonormal to their shifted versions but not their scaled ones.

Decomposition into scaling and wavelet basis set

On a certain arbitrary scale, for any function $x(t)$

$$x(t) = \sum_k c_{j,k}\phi_{j,k}(t) + \sum_{j \le j_n} \sum_k d_{j,k}\Psi_{j,k}(t)$$

where $c_{j,k} = \int_{-\infty}^{\infty} x(t)\phi_{j,k}(t)dt$ and $d_{j,k} = \int_{-\infty}^{\infty} x(t)\Psi_{j,k}(t)dt$

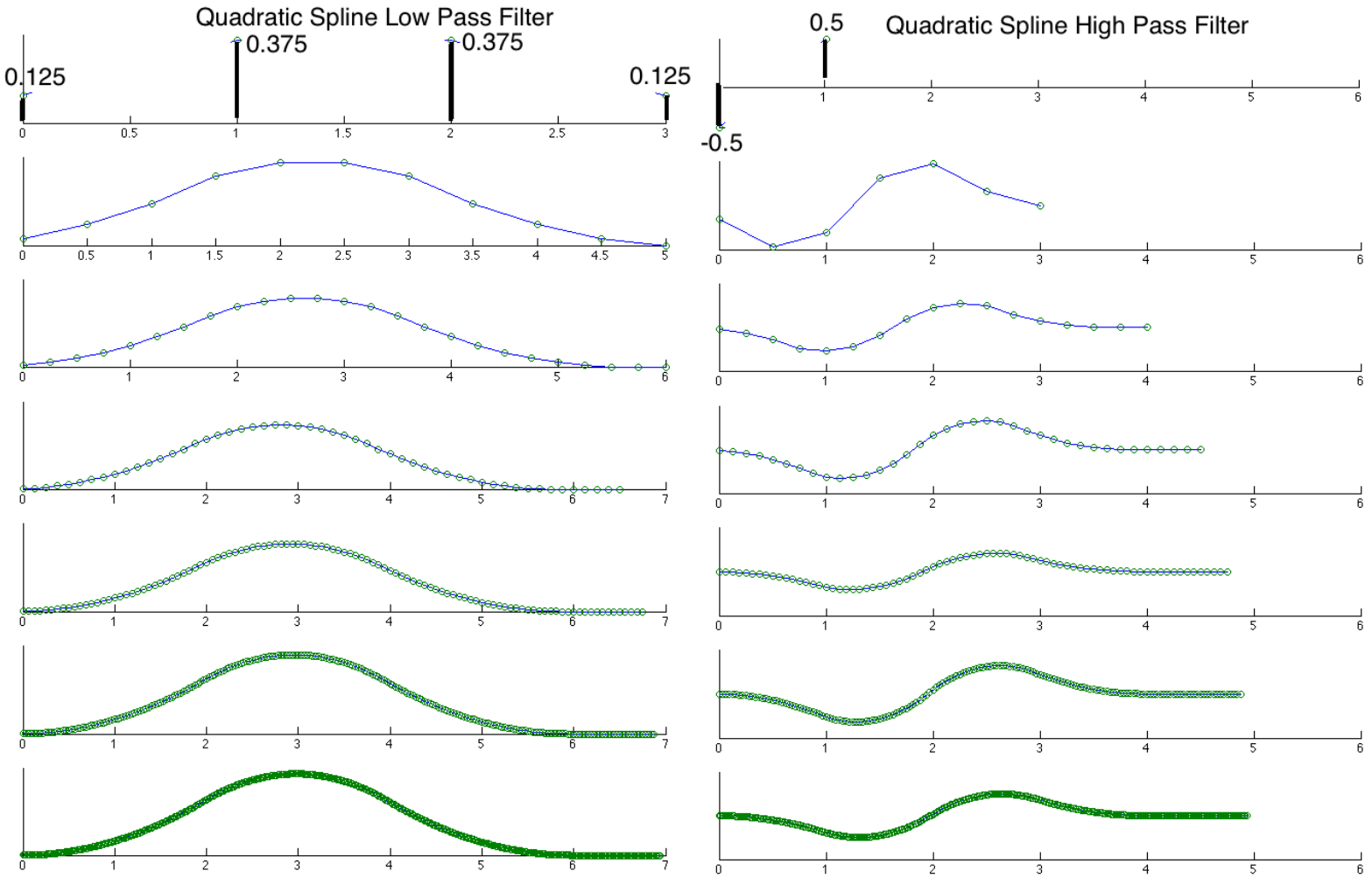
2-Scale Iteration Equation

$$\phi_{j,k}(t) = \sum_{k=-\infty}^{\infty} h[k]\phi_{j,k}(2t-k)$$

$$\Psi_{j,k}(t) = \sum_{k=-\infty}^{\infty} g[k]\phi_{j,k}(2t-k)$$

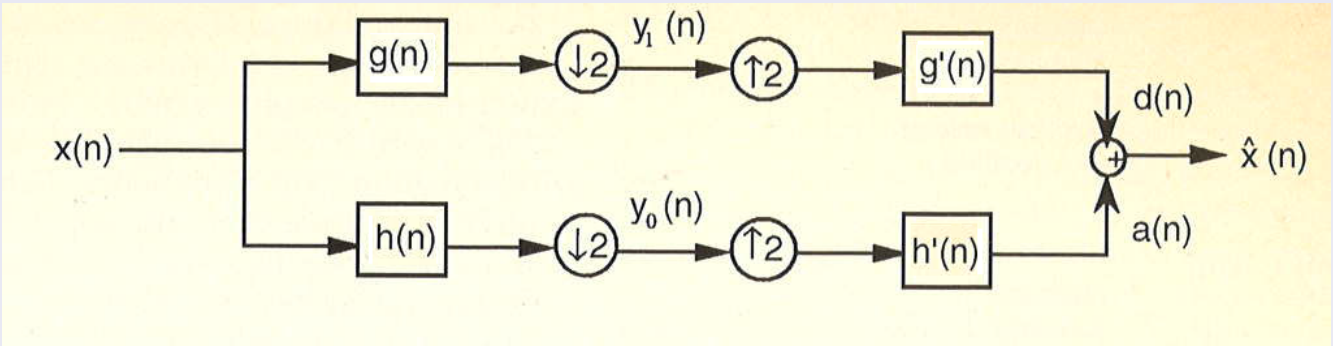
MATLAB Demonstration of Repeated 2-Scale iterations to obtain quadratic spline scaling $\phi(t)$ and wavelet functions $\psi(t)$

```
h = [   1 3 3 1  ]'/8; % quadratic spline scaling filter (LP)
g = [-2 2 ]'/4;% quadratic spline wavelet filter (HP)
Xr1 =h; Xr2=g;
Ts =1;  Jmax = 7;
t_Ts1 = 0:length(Xr1)-1;
t_Ts2 = 0:length(Xr2)-1;
for J=1:Jmax,
subplot(Jmax,2,2*J-1);
plot(t_Ts1*Ts,Xr1,t_Ts1*Ts,Xr1,'o'); set(gca,'ytick',[]);set(gca,'box','off');
subplot(Jmax,2,2*J);
plot(t_Ts2*Ts,Xr2,t_Ts2*Ts,Xr2,'o'); set(gca,'ytick',[]);set(gca,'box','off');
Ts = Ts/2;
W1 = zeros(length(Xr1)*2,1);  W1(1:2:end) = Xr1; % upsample the LP filter by 2
W2 = zeros(length(Xr2)*2,1);  W2(1:2:end) = Xr2; % upsample the HP filter by 2
Xr1 = Ts*conv(W1,h);% Interpolate with LP filter
Xr2 = Ts*conv(W2,h); % Interpolate with LP filter
t_Ts1 = 0:length(Xr1)-1;
t_Ts2 = 0:length(Xr2)-1;
xlim([0 6]);
end;
```


Quadratic Spline Low Pass Filter
0.125
0.375
0.375
0.125
Quadratic Spline High Pass Filter
0.5
-0.5

Scale and Resolution

Low pass filter with $h[n]$, high pass filter with $g[n]$ and downsample by 2



$$y_0[n] = \sum_{k=-\infty}^{\infty} x[k]h[-k+2n] \text{ and } y_1[n] = \sum_{k} x[k]g[-k+2n]$$

Upsample by 2: $y_0'[2n] = y_0[n]$, $y_0'[2n+1] = 0$, $y_1'[2n] = y_1[n]$, $y_1'[2n+1] = 0$ and interpolate with low pass filter $h'[n]$ and $g'[n]$, respectively

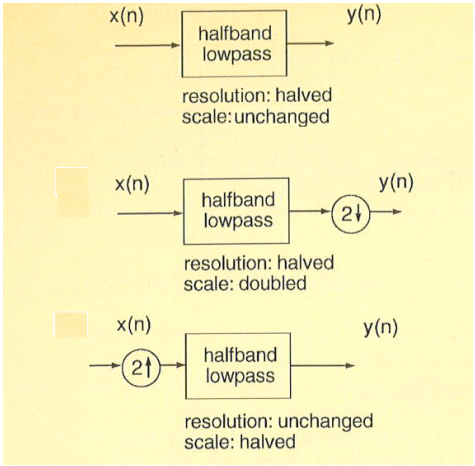
$$\hat{x}[n] = \sum_{k=-\infty}^{\infty} \Big[y_0[k]h'[n-2k] + y_1[k]g'[n-2k] \Big]$$

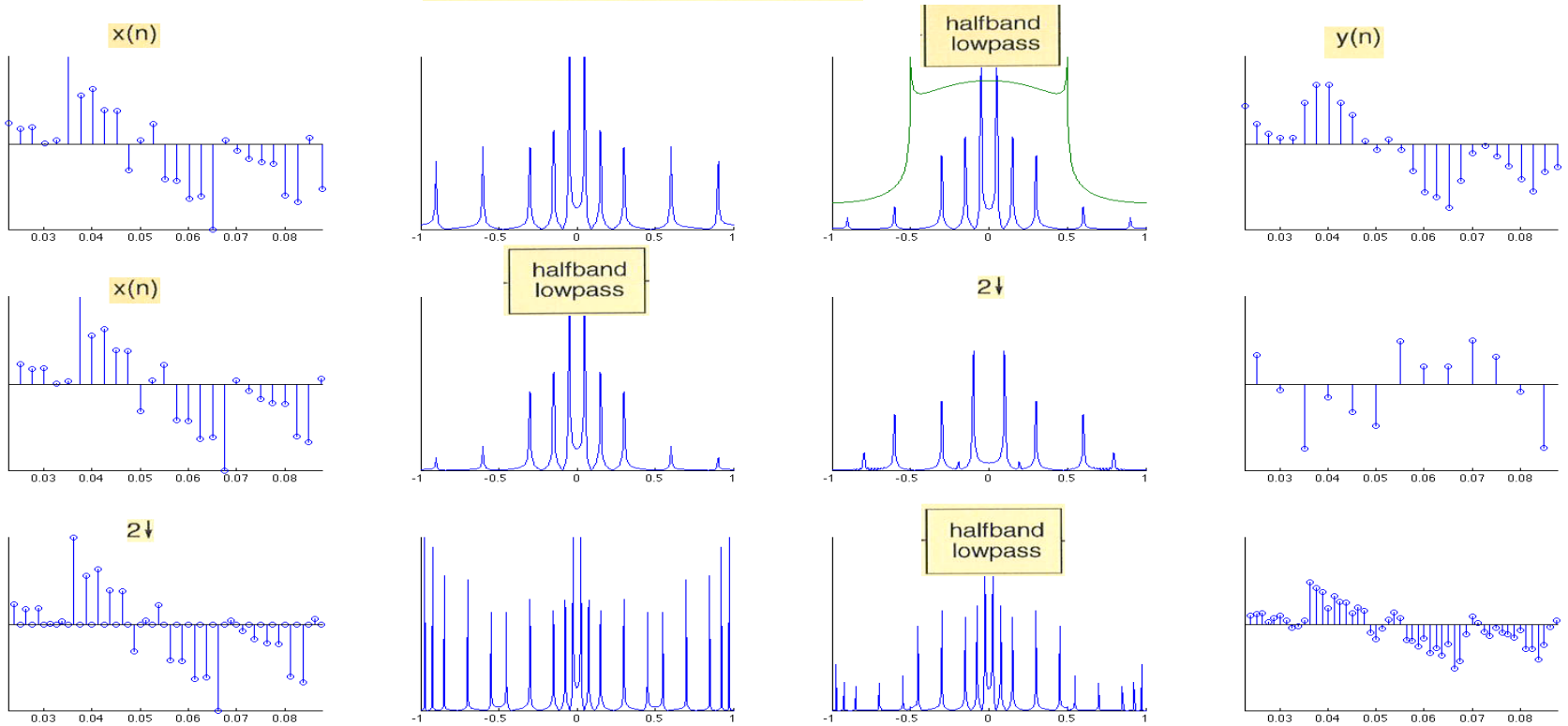
MATLAB Demonstration of Half Band Filtering and Upsampling/Downsampling

```
Nf=5; % number of sinusoids
Fs = 400; % Original sampling rate
Td=1; % signal duration
Af = [10 6 5 5 4]' ; %  amplitudes of the sinusoids
f = [  10 30 60 120  180];% frequencies of individual components in the signal
T  =1/Fs; % sampling interval of the original signal
%t  =  [0:delta_t:T-delta_t]';
t = linspace(0,Td,Td*Fs)'; % time axis of the resampled signal
x =  sin(t*2*pi*f)*Af; % original signal sampled as a sum of the sinusoids

D = 2 ; % no downsampling
td=1:D:length(x); % downsampled time index
L = length(td) ; % number of samples in the downsampled signal
M = length(t) ;  % number of samples in the original signal
xd=x(td); % downsampled version of the original signal by a factor of D
h_s=  0.5*sinc( (t - (length(100)/2)*T)/(D*T) );   %  halfband filter
y = real(ifft(fft(x).*fft(h_s) ));  %  halfband filtered signal

% Resolution is halved, scale is unchanged
Df = Fs; % sampling rate of frequency spectrum
w = linspace(-1,1,Df);
H_x = freqz(x,1,w*pi);
H_y = freqz(y,1,w*pi);
H_S =freqz(h_s,1,w*pi);
```

```
t1 = 10:36;
subplot(3,4,1); stem(t(t1),x(t1)) ;
axis([min(t(t1))  max(t(t1)) min(x(t1)) max(x(t1))  ]);set(gca,'ytick',[]);set(gca,'box','off');
subplot(3,4,2); plot(w,abs(H_x)) ;
axis([min(w)  max(w) min(abs(H_x)) max(abs(H_x))  ]);set(gca,'ytick',[]);set(gca,'box','off');
subplot(3,4,3); plot(w,abs(H_y),w,abs(H_S)* max(abs(H_x))/max(abs(H_S))) ;
axis([min(w)  max(w) min(abs(H_x)) max(abs(H_x))  ]);set(gca,'ytick',[]);set(gca,'box','off');
subplot(3,4,4); stem(t(t1),y(t1)) ;
axis([min(t(t1))  max(t(t1)) min(x(t1)) max(x(t1))  ]);set(gca,'ytick',[]);set(gca,'box','off');

% downsampling after half band filtering
D = 2;
y_d = y(1:D:end);
t2=[round((min(t1)-1)/D)+1 :D:max(t1)];

% Resolution is halved, scale is doubled
H_yd = freqz(y_d,1,w*pi);

subplot(3,4,5); stem(t1/Fs,x(t1)) ;
axis([min(t(t1))  max(t(t1)) min(x(t1)) max(x(t1))  ]); set(gca,'ytick',[]);set(gca,'box','off');
subplot(3,4,6); plot(w,abs(H_y)) ;
axis([min(w)  max(w) min(abs(H_y)) max(abs(H_y)) ]); set(gca,'ytick',[]);set(gca,'box','off');
subplot(3,4,7); plot(w,abs(H_yd));
axis([min(w)  max(w) min(abs(H_y)) max(abs(H_y)) ])   ; set(gca,'ytick',[]);set(gca,'box','off');
subplot(3,4,8); stem((t2/(Fs)),y_d(t2));
axis([min(t(t1))  max(t(t1)) min(x(t1)) max(x(t1))  ]); set(gca,'ytick',[]);set(gca,'box','off');
```

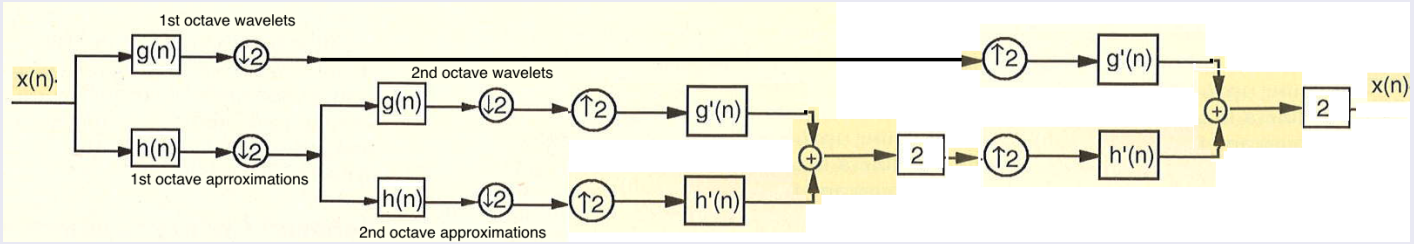
```
% Half band filtering after upsampling
x_ups = zeros(length(x)*2,1);  x_ups(1:2:end) = x; % upsample by 2
y_ups = real(ifft(fft(x_ups).*fft(h_s,length(x_ups)) ));  %  halfband filter  the  signal
% frequency responses
Df = Fs*D; % sampling rate of frequency spectrum
w = linspace(-1,1,Df);
H_x = freqz(x_ups,1,w*pi);
H_yups = freqz(y_ups,1,w*pi);

t3=[(D*(min(t1)-1)+1 ):D*max(t1)];
subplot(3,4,9); stem(t3/(D*Fs),x_ups(t3));
axis([min(t(t1))  max(t(t1)) min(x(t1)) max(x(t1))  ]);set(gca,'ytick',[]);set(gca,'box','off');
subplot(3,4,10); plot(w,abs(H_x)) ;
axis([min(w)  max(w) min(abs(H_y)) max(abs(H_y))  ]); set(gca,'ytick',[]);set(gca,'box','off');
subplot(3,4,11); plot(w,abs(H_yups)) ;
axis([min(w)  max(w) min(abs(H_y)) max(abs(H_y))  ]);set(gca,'ytick',[]);set(gca,'box','off');
subplot(3,4,12); stem((t3/(D*Fs)),y_ups(t3)) ;
axis([min(t(t1))  max(t(t1)) min(x(t1)) max(x(t1))  ]);set(gca,'ytick',[]);set(gca,'box','off');
```


x(n)
halfband lowpass
y(n)
resolution: halved
scale:unchanged
x(n)
halfband lowpass
2↓
y(n)
resolution: halved
scale: doubled
x(n)
2↑
halfband lowpass
y(n)
resolution: unchanged
scale: halved


x(n)
halfband lowpass
y(n)
x(n)
halfband lowpass
2↓
2↓
halfband lowpass

Multiresolution Pyramid

2 Octave Wavelet Decomposition and Reconstruction by the Pyramidal Scheme

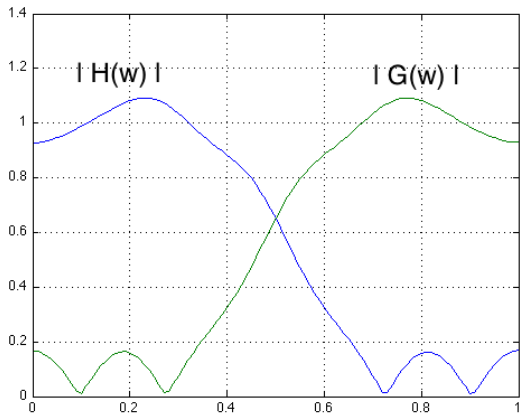


The lowpass $h[n]$ and high pass $g[n]$ FIR filters are related by $g[L-1-n] = (-1)^n h[n]$, L is the filter length

Analysis filters $h[n]$, $g[n]$ and synthesis filters, $h'[n]$, $g'[n]$ are the time reversal of each other because of orthonormality.

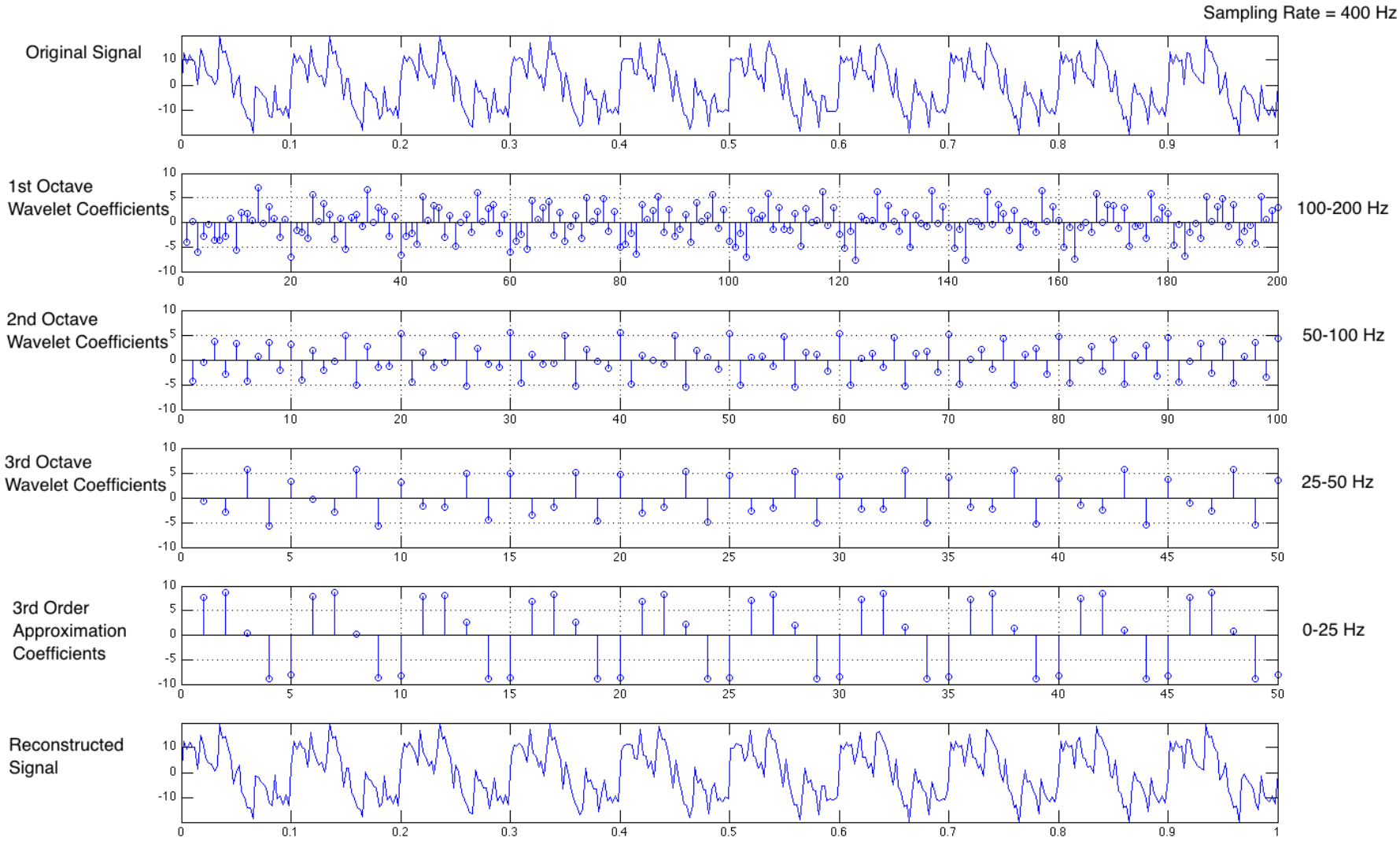
MATLAB Demonstration of 3 Octave Pyramidal Decomposition and Reconstruction

```
% Tutorial for Multiresolution Analysis
% Mallat wavelet
% low pass filter
h = [0.542 0.307 -0.035 -0.078 0.023 -0.030  0.012 -0.013 0.006 0.006  -0.003 -0.002 ];
n0=length(h)-1;
np= [1:2*n0-1]'-n0;
h = [fliplr(h(2:end)) h]';
n = [-n0:n0]';
g = ((-1).^(1-n)); % high pass filter
g = g.* [ 0; h(end:-1:2)  ];
stem(n,[h  g])
f =[0:0.01:1];
H= (freqz(h,1,f*pi));
G= (freqz(g,1,f*pi));
plot(f,abs(H),f,abs(G));grid;pause;
```



```
%%%%%%%%%%%%%%%%%%%%%
% Generate a signal as a sum of sinusiods
Nf=5; % number of sinusoids
Fs = 400; % Original sampling rate
Td=1; % signal duration
Af = [10 6 5 4 3]' ; %  amplitudes of the sinusoids
f = [  10 30 60 120  180]; % frequencies of individual components in the signal
T  =1/Fs; % sampling interval of the original signal
%t  =  [0:delta_t:T-delta_t]';
t = linspace(0,Td,Td*Fs)'; % time axis of the resampled signal
x =  sin(t*2*pi*f)*Af; % original signal sampled as a sum of the sinusoids


L = length(x);
Xr = x;
clear W
% Analysis
for J = 1: 3,
XF = fft(Xr);
W1 = real(ifft( fft(g,length(Xr)).*XF  )  );
R1  = real(ifft( fft(h,length(Xr)).*XF  )  );
W{J}= W1(1:2:end);
R1 = R1(1:2:end);
Xr = R1;
end;
W{J+1}=R1;
subplot(6,1,1);plot(t,x); axis([ min(t) max(t)  min(x)  max(x) ]);
for J =1:4, subplot(6,1,J+1);stem(W{J});grid;end
```

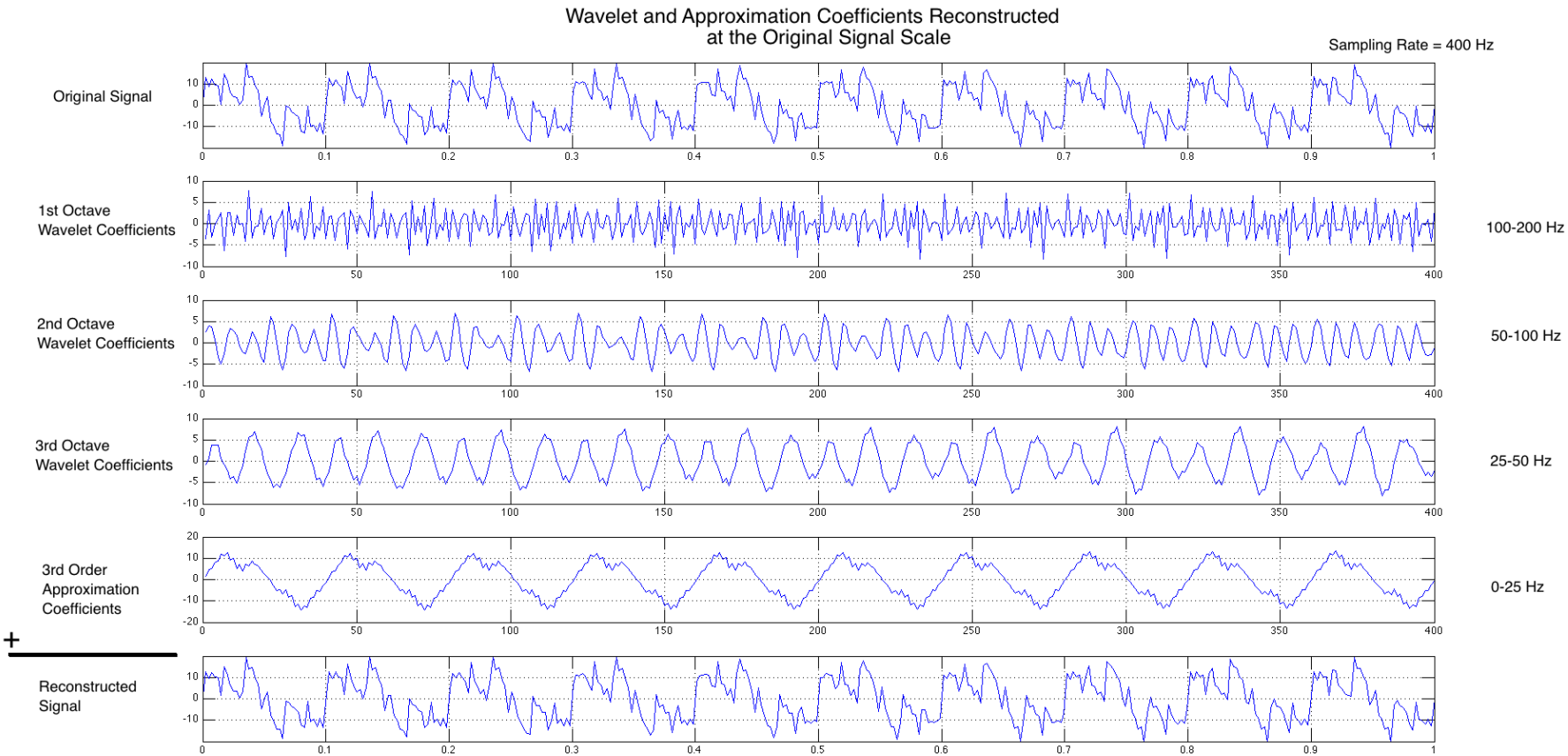
```
% Synthesis
Xr = W{end};
L =length(Xr);
for J = 3:-1:1,
W1 = zeros(length(Xr)*2,1);
R1 =  zeros(length(Xr)*2,1);
W1(1:2:end) = W{J};
R1(1:2:end)  = Xr;
XF = fft(Xr,length(W1));
W1 = real(ifft( conj(fft(g,length(W1))).*fft(W1)  )  );
R1  = real(ifft( conj(fft(h,length(W1))).*fft(R1) )  );
Xr = 2*(W1+R1);
end;
subplot(6,1,6);plot(t,Xr); axis([ min(t) max(t)  min(x)  max(x) ]);
```


Sampling Rate = 400 Hz
Original Signal
1st Octave
Wavelet Coefficients
100-200 Hz
2nd Octave
Wavelet Coefficients
50-100 Hz
3rd Octave
Wavelet Coefficients
25-50 Hz
3rd Order
Approximation
Coefficients
0-25 Hz
Reconstructed
Signal

MATLAB Demonstration of Reconstructed Wavelet and Approximation Coefficients at the Original Signal Scale for 3 Octave Pyramidal Decomposition

```
% Reconstruction of residual coefficients at original scale
Xr = W{4};
for J=3:-1:1,
W1 = zeros(length(Xr)*2,1); W1(1:2:end) = Xr;
XF = fft(Xr,length(W1));
W1 = real(ifft( conj(fft(h,length(W1))).*fft(W1)  )  );
Xr   = 2*W1;
end;
W_I{4}=Xr;

% Reconstruction of wavelet coefficients at original scale
for K = 3:-1:1,
Xr = W{K};
W1 = zeros(length(Xr)*2,1);
W1(1:2:end) = Xr;
XF = fft(Xr,length(W1));
W1 = real(ifft( conj(fft(g,length(W1))).*fft(W1)  )  );
Xr   = 2*W1;
for J=K-1:-1:1,
W1 = zeros(length(Xr)*2,1);
W1(1:2:end) = Xr;
XF = fft(Xr,length(W1));
W1 = real(ifft( conj(fft(h,length(W1))).*fft(W1)  )  );
Xr   = 2*W1;
end;
W_I{K}=Xr
end;
```

```
WT = zeros(size(x));
subplot(6,1,1);plot(t,x);grid; axis([ min(t) max(t)  min(x)  max(x) ]);
for J =1:4,
subplot(6,1,J+1);plot(W_I{J});grid; WT = WT + W_I{J};
end
subplot(6,1,6);plot(t, [WT ]);grid; axis([ min(t) max(t)  min(x)  max(x) ]);
```


Wavelet and Approximation Coefficients Reconstructed
at the Original Signal Scale
Sampling Rate = 400 Hz
Original Signal
1st Octave
Wavelet Coefficients
100-200 Hz
2nd Octave
Wavelet Coefficients
50-100 Hz
3rd Octave
Wavelet Coefficients
25-50 Hz
3rd Order
Approximation
Coefficients
0-25 Hz
+
Reconstructed
Signal

Restoring the phase delay caused by filtering

```
n1=0:63;
n2=length(n1)+n1;
n3=length(n1)+length(n2)+n1;
n4= length(n1)+length(n2)+length(n3)+n1;
x= zeros(1,length(n1)+length(n2)+length(n3) +length(n4));
x(1+n1) = 2*cos(0.02*pi*n1) ;  x(1+n3) = 3*cos(0.36*pi*n3);
% Tutorial for Multiresolution Analysis
% Mallat wavelet
% low pass filter
h = [0.542 0.307 -0.035 -0.078 0.023 -0.030 0.012 -0.013 0.006 0.006 -0.003 -0.002 ];
n0=length(h)-1;
np= [1:2*n0-1]'-n0;
h = [fliplr(h(2:end)) h]';
n = [-n0:n0]';
g = ((-1).^(1-n)); % high pass filter
g = g.* [ 0; h(end:-1:2) ];
x=x';
t=0:length(x)-1;
Xr = x;
h_shift = fix(length(h)/2)+1;   g_shift = fix(length(g)/2)+1;
% Analysis
for J = 1:3,XF = fft(Xr);
W1 = real(ifft( fft(g,length(Xr)).*XF ) ); W1 = circshift(W1,-g_shift);
R1 = real(ifft( fft(h,length(Xr)).*XF ) ); R1 = circshift(R1,-h_shift);
W{J}= W1(1:2:end);
R1 = R1(1:2:end);
Xr = R1;
end;
```

Restoring the phase delay caused by filtering

```
W{J+1}=R1;
subplot(6,1,1);plot(t,x); axis([ min(t) max(t) min(x) max(x) ]);
for J =1:3, subplot(6,1,J+1); WU = zeros(length(x),1); WU(1:2^J:end) = W{J}; bar(WU);
grid;axis([ min(t) max(t) min(x) max(x) ]);
end
subplot(6,1,J+2); WU = zeros(length(x),1); WU(1:2^(J):end) = W{J+1}; bar(WU);
grid;axis([ min(t) max(t) min(x) max(x) ]);
```