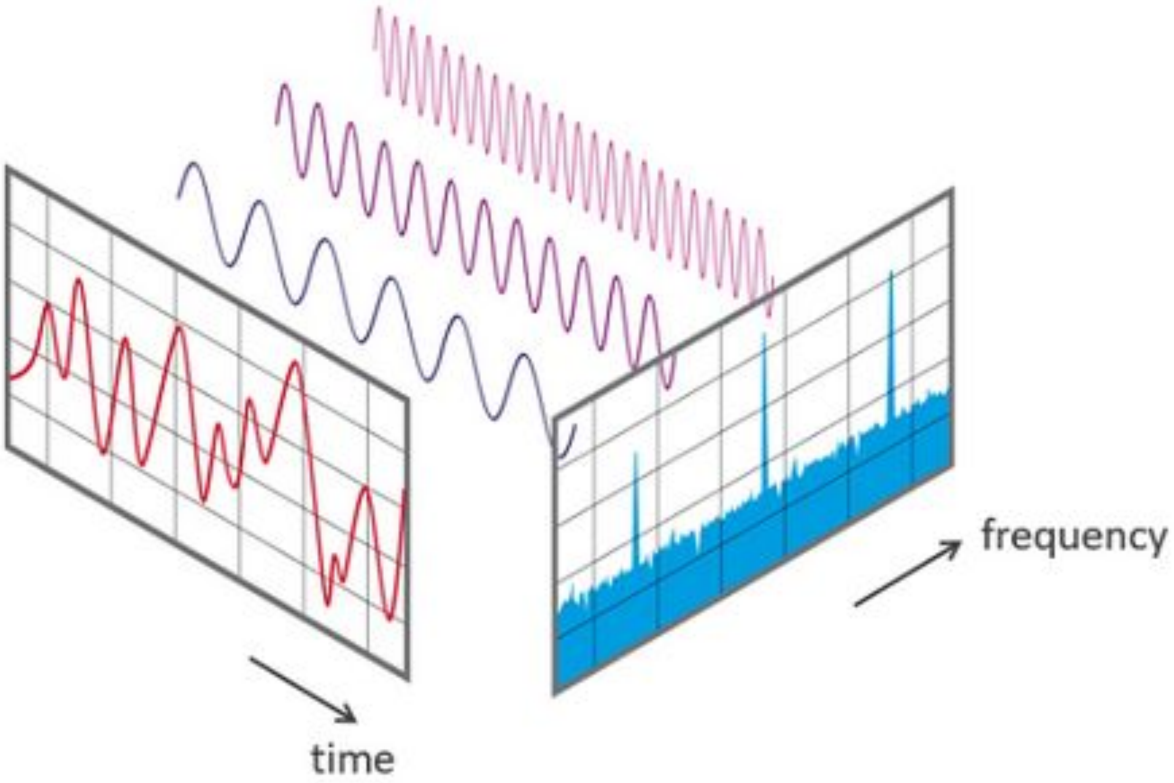
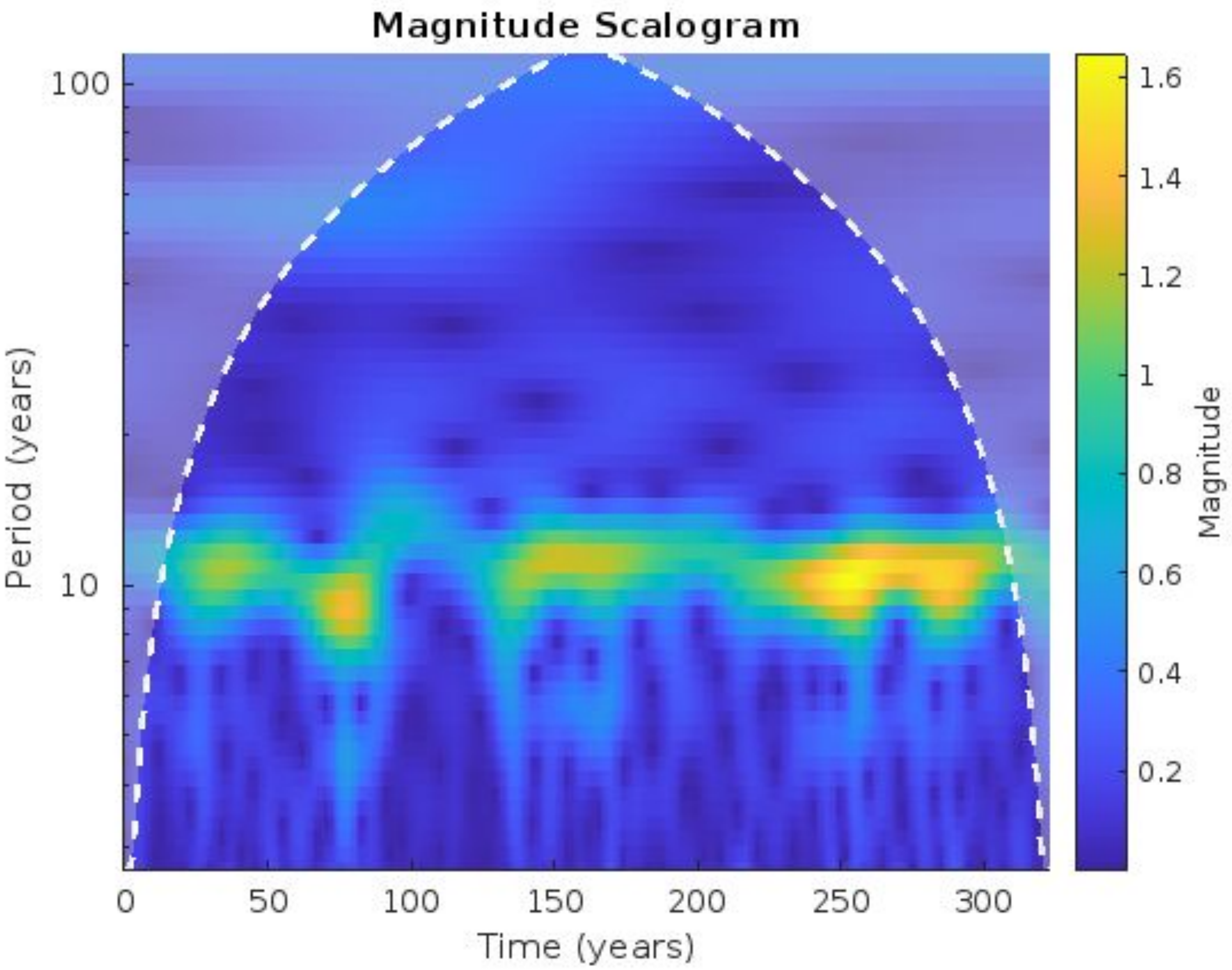
Catch the Rhythm with Wavelets

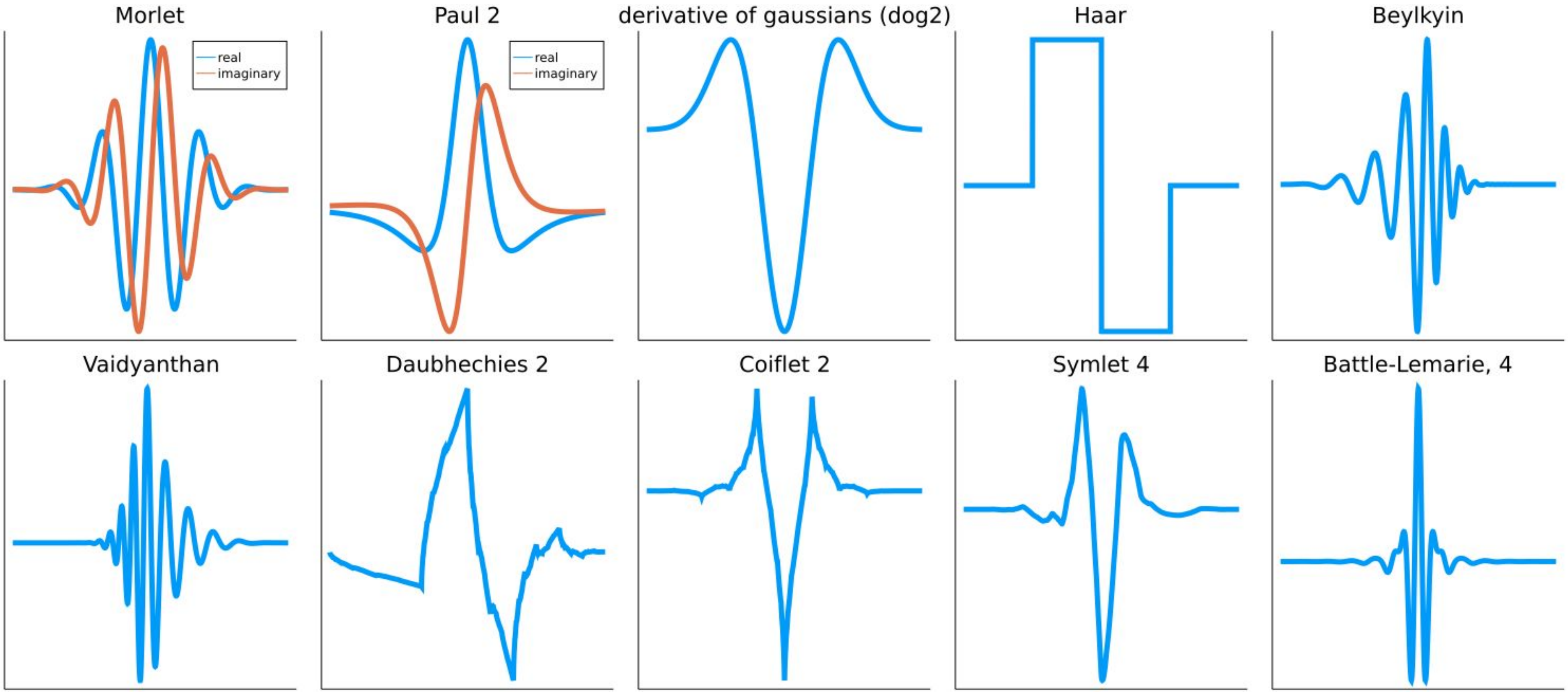
cessi

The big picture

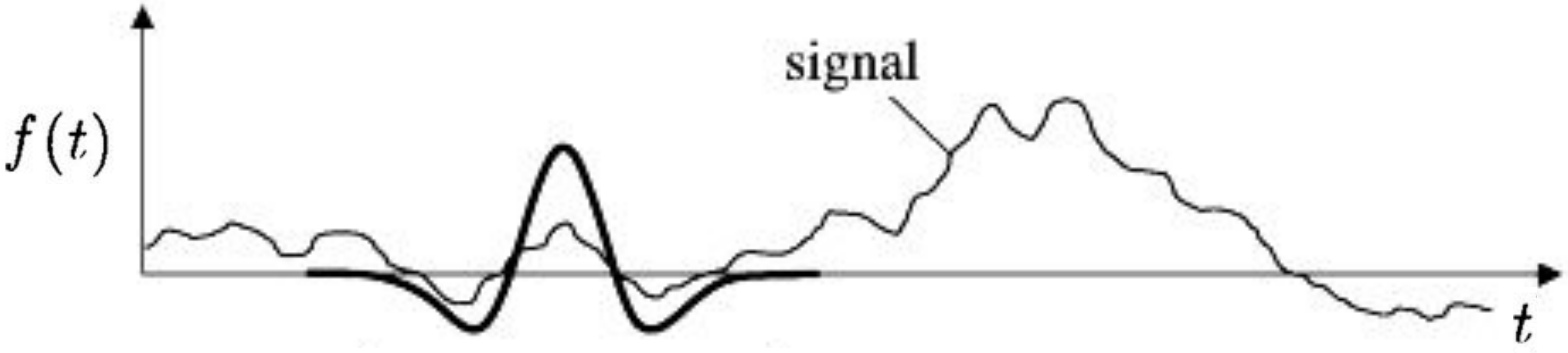



Magnitude Scalogram
100
10
Period (years)
0
50
100
150
200
250
300
Time (years)
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
Magnitude

A wavelet is a *small* wave which oscillates and decays in the time domain.



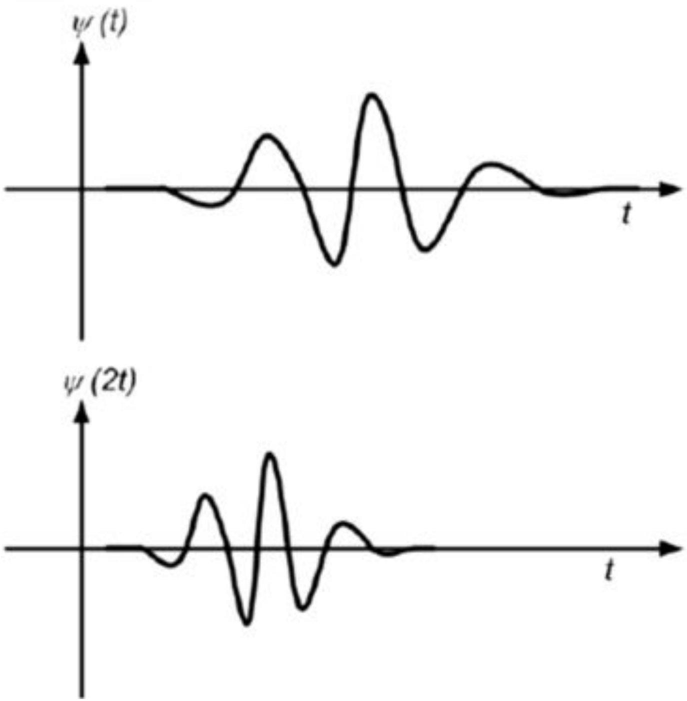
Local matching of wavelet and signal can extract useful information.



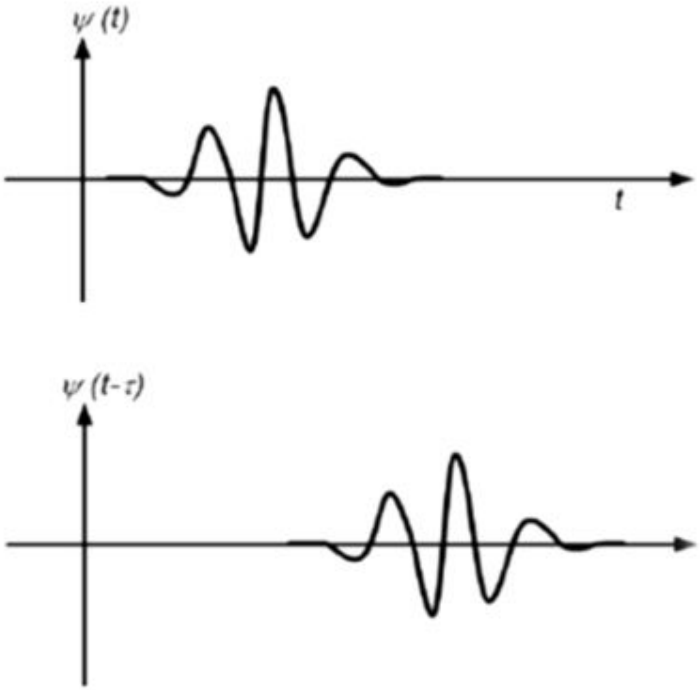
$$f(t) = \sum_{k=-\infty}^{\infty} c_k \phi_k(t) + \sum_{k=-\infty}^{\infty} \sum_{j=0}^{\infty} d_{j,k} \psi_{j,k}(t)$$

Use a Thorn to ~~remove~~ Detect a Thorn

Scaling



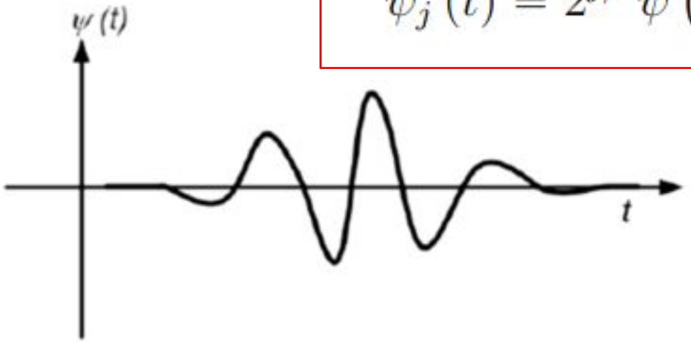
Translation



Use a Thorn to ~~remove~~ Detect a Thorn

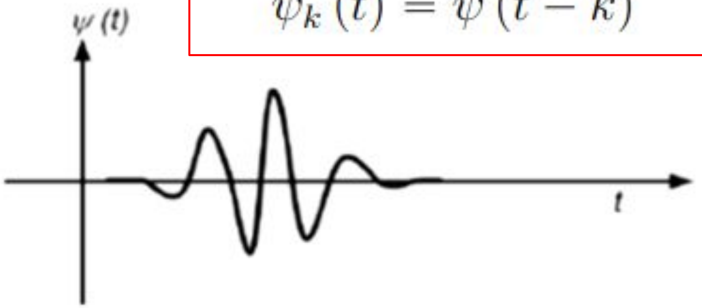
Scaling

$$\boxed{\psi_j(t) = 2^{j/2}\psi\left(2^j t\right)}$$

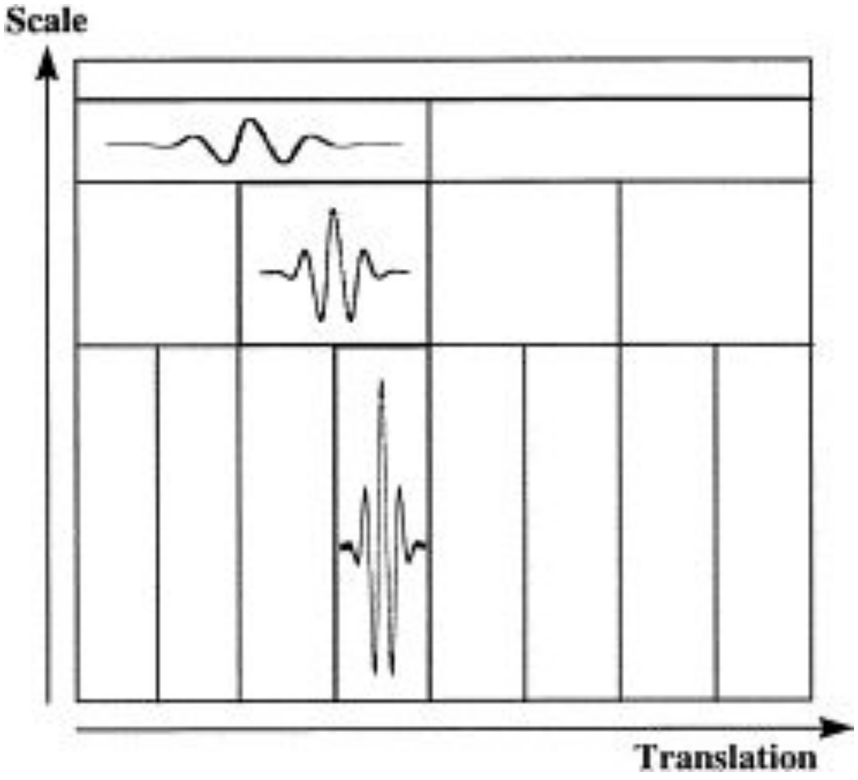


Translation

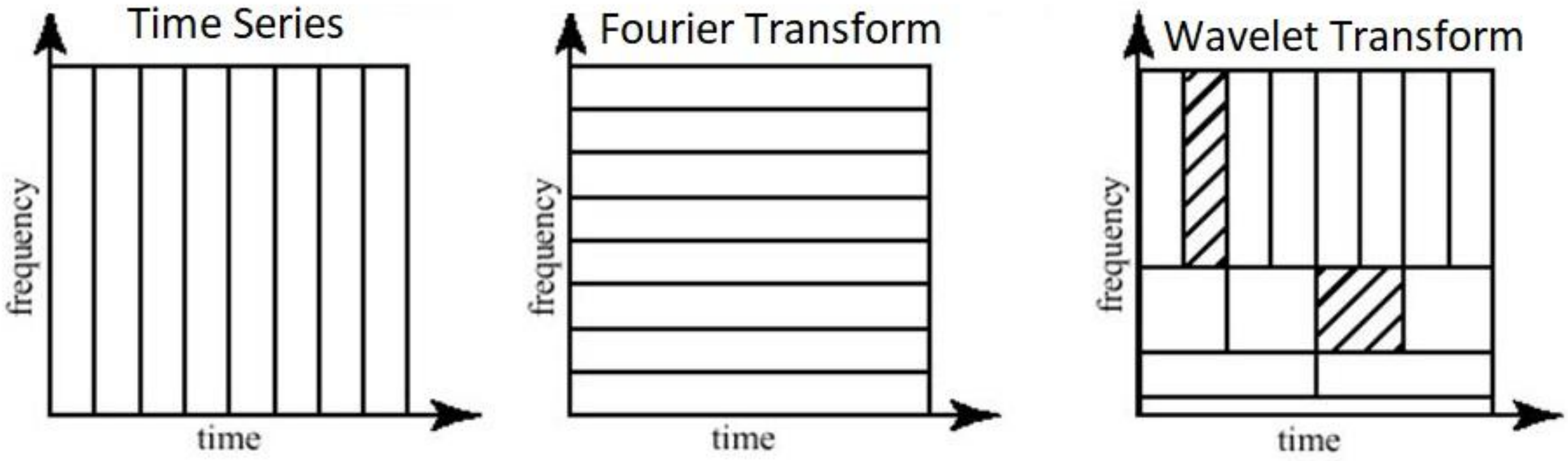
$$\boxed{\psi_k(t) = \psi(t-k)}$$



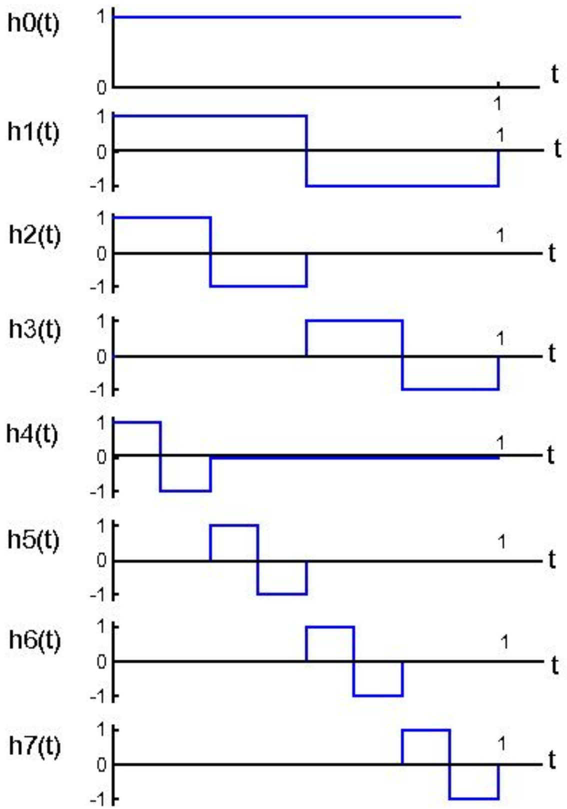
Use a Thorn to ~~remove~~ Detect a Thorn



Time-Frequency Localization



Haar Wavelet Family



Father $\phi(t)$

Mother $\psi(t)$

Daughter

Daughter

Translation

$$\phi_k(t) = \phi(t-k)$$

Translation + Scaling

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

$$\boxed{f(t) = \sum_{k=-\infty}^{\infty} c_k \phi_k(t) + \sum_{k=-\infty}^{\infty} \sum_{j=0}^{\infty} d_{j,k} \psi_{j,k}(t)}$$

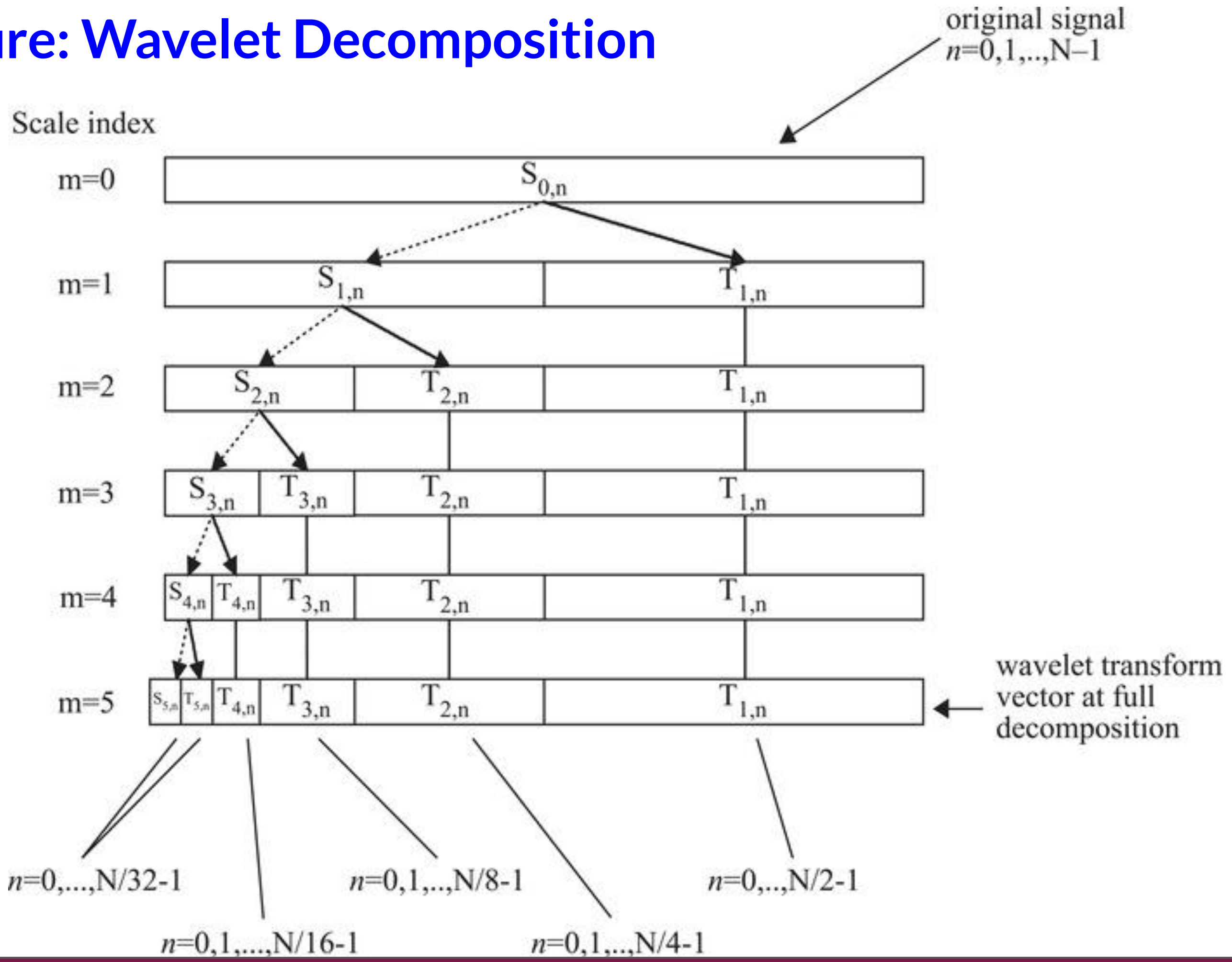
Wavelet Family – Some Characteristics

	Father $\phi(t)$	Mother $\psi(t)$
Def.	$\int_{-\infty}^{\infty} \phi(t)\, dt = constant$	$\int_{-\infty}^{\infty} \psi(t)\, dt = 0$
Square integrability	$\int_{-\infty}^{\infty} \lvert\phi(t)\rvert^2\, dt < \infty$	$\int_{-\infty}^{\infty} \lvert\psi(t)\rvert^2\, dt < \infty$

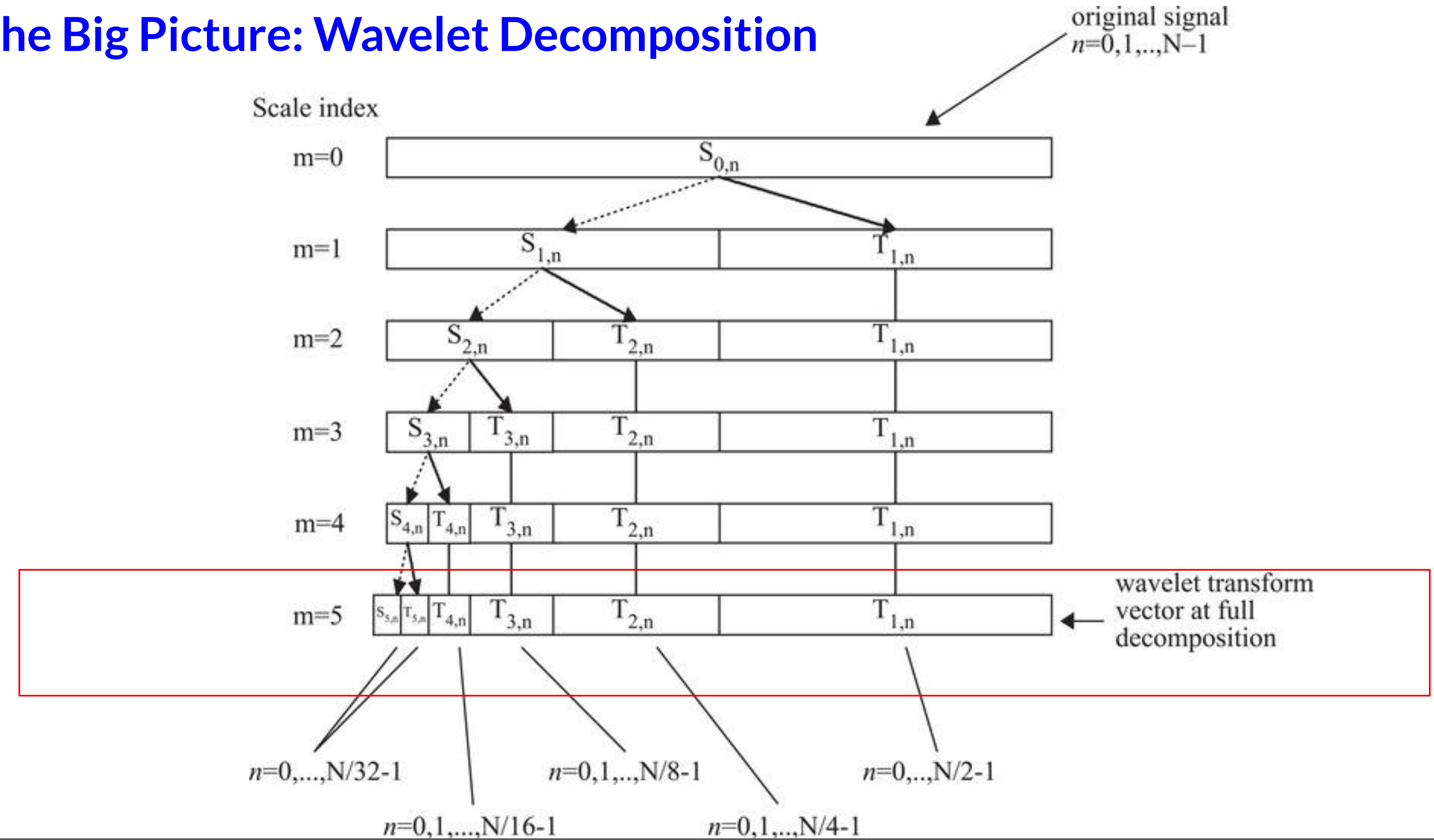
Orthogonality

$$\int_{-\infty}^{\infty} \phi^*(t)\, \psi(t)\, dt = 0$$

The Big Picture: Wavelet Decomposition



The Big Picture: Wavelet Decomposition



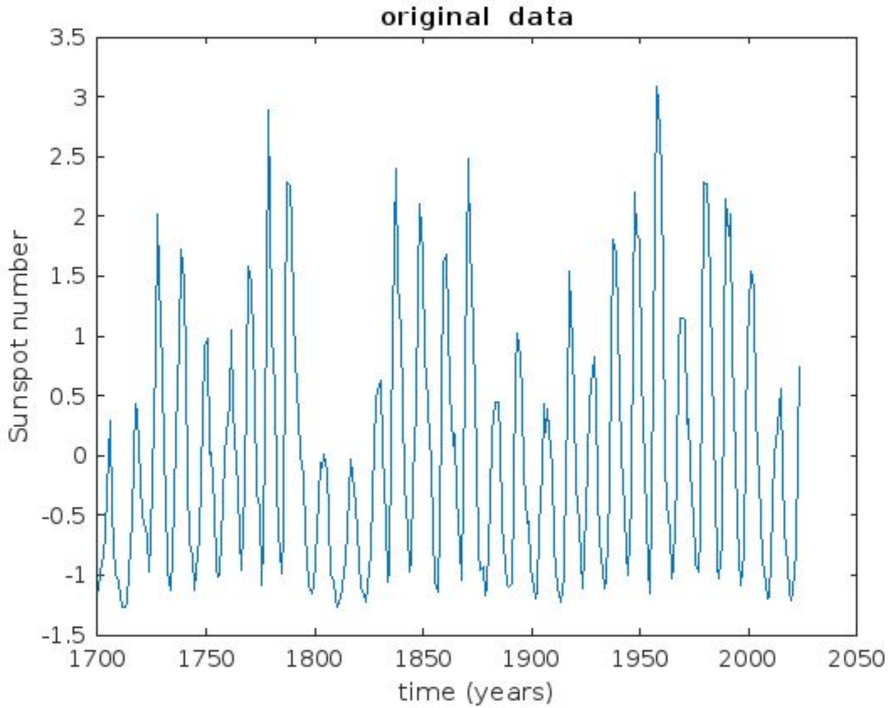
Hands On MATLAB

$$f(t) = \sum_{k=-\infty}^{\infty} c_k \phi_k(t) + \sum_{k=-\infty}^{\infty} \sum_{j=0}^{\infty} d_{j,k} \psi_{j,k}(t)$$

- Plotting the original data in Time domain
- Wavelet decomposition and plotting the coefficients *c*'s and *d*'s
- Reconstructing the signal at different levels
- Continuous Wavelet Transform – Scalogram (Time-Frequency Domain)

```
clc; close all; clear global; clear all
data=readtable('ssn.txt'); %address
data=table2array(data); % conversion to  numeric matrix
time = data(:,1);
ssn=data(:,2);
m = mean(ssn); sd =std(ssn);
ssn =  (ssn-m)/sd;
figure()
plot(time,ssn);
xlabel('time  (years)');
ylabel('Sunspot number');
title('original  data');
```

```
clc; close all; clear global; clear all
data=readtable('/MATLAB Drive/1MillenniumSSN.dat'); %address
data=table2array(data); % conversion to  numeric matrix
time = data(:,1);
ssn=data(:,2);
m = mean(ssn); sd =std(ssn);
ssn = (ssn-m)/sd;
figure()
plot(time,ssn);
xlabel('time (years)');
ylabel('Sunspot number');
title('original  data');
```



```
[c,l] = wavedec(ssn,4,'db4');                 % level 4 decomposition
approx = appcoef(c,l,'db4');                  % approximation coefficients
[cd1,cd2,cd3,cd4] = detcoef(c,l,[1 2 3 4]);   % detailed coefficients
```

```
[c,l] = wavedec(ssn,4,'db4');                   % level 4 decomposition
approx = appcoef(c,l,'db4');                    % approximation coefficients
[cd1,cd2,cd3,cd4] = detcoef(c,l,[1 2 3 4]);     % detailed coefficients
```

```
>> help wavedec
 wavedec Multi-level 1-D wavelet decomposition. wavedec performs a multilevel 1-D wavelet analysis using either a
specific wavelet 'wname' or a specific set of wavelet decomposition filters (see WFILTERS).

      [C,L] = wavedec(X,N,'wname') returns the wavelet decomposition of the signal X at level N, using 'wname'.
wavedec does not enforce a maximum level restriction. Use WMAXLEV to ensure the wavelet coefficients are free from
boundary effects. If boundary effects are not a concern in your application, a good rule is to set N less than or
equal to fix(log2(length(X))).

      The output vector, C, contains the wavelet decomposition. L contains the number of coefficients by level.
      C and L are organized as:
      C      = [app. coef.(N)|det. coef.(N)|... |det. coef.(1)]
      L(1)    = length of app. coef.(N)
      L(i)    = length of det. coef.(N-i+2) for i = 2,...,N+1
      L(N+2) = length(X).

      [C,L] = wavedec(X,N,Lo_D,Hi_D) Lo_D is the decomposition low-pass filter and Hi_D is the decomposition
      high-pass filter.


      See also dwt, waveinfo, waverec, wfilters, wmaxlev.
```

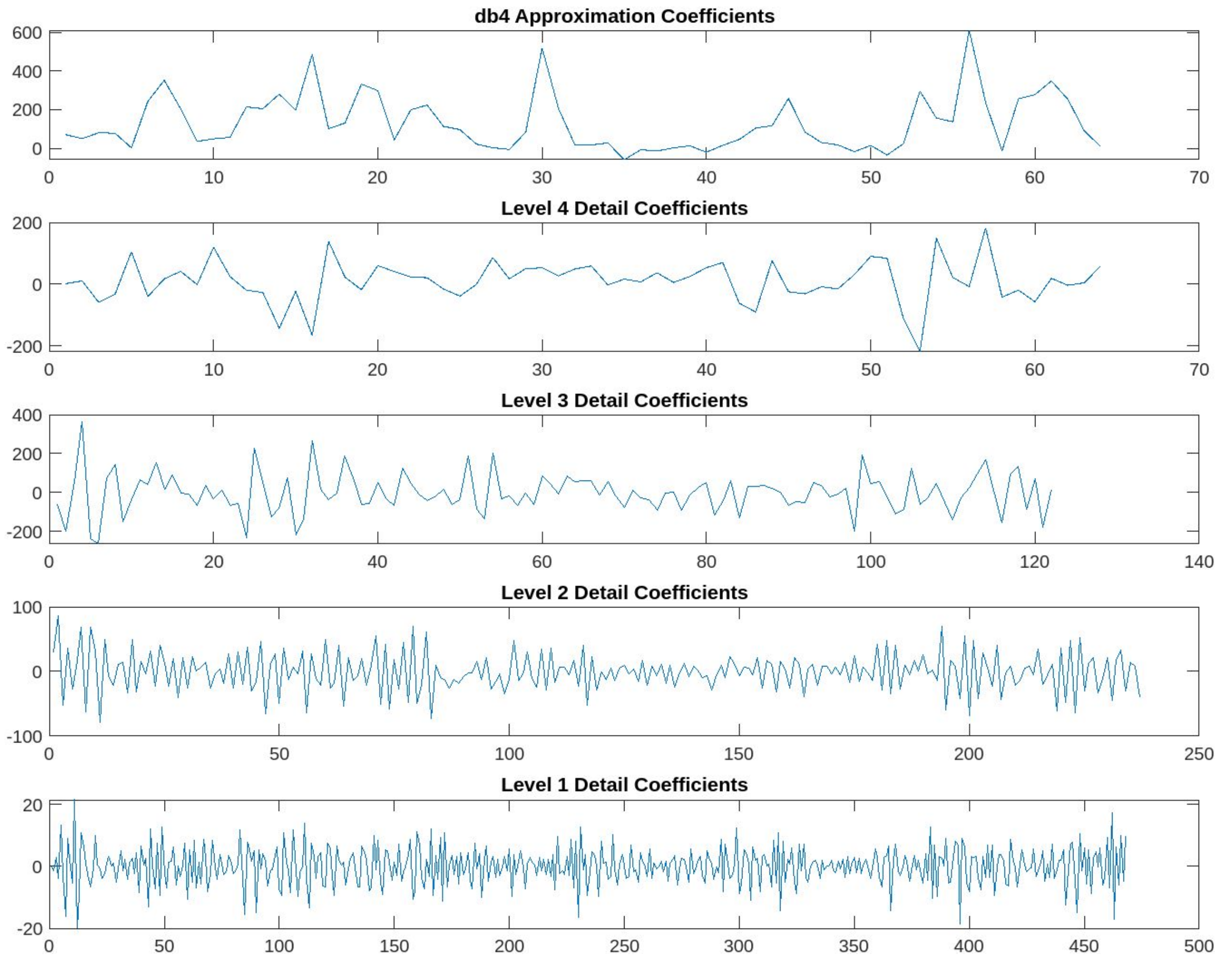
```
[c,l] = wavedec(ssn,4,'db4');                % level 4 decomposition
approx = appcoef(c,l,'db4');                 % approximation coefficients
[cd1,cd2,cd3,cd4] = detcoef(c,l,[1 2 3 4]);  % detailed coefficients
```

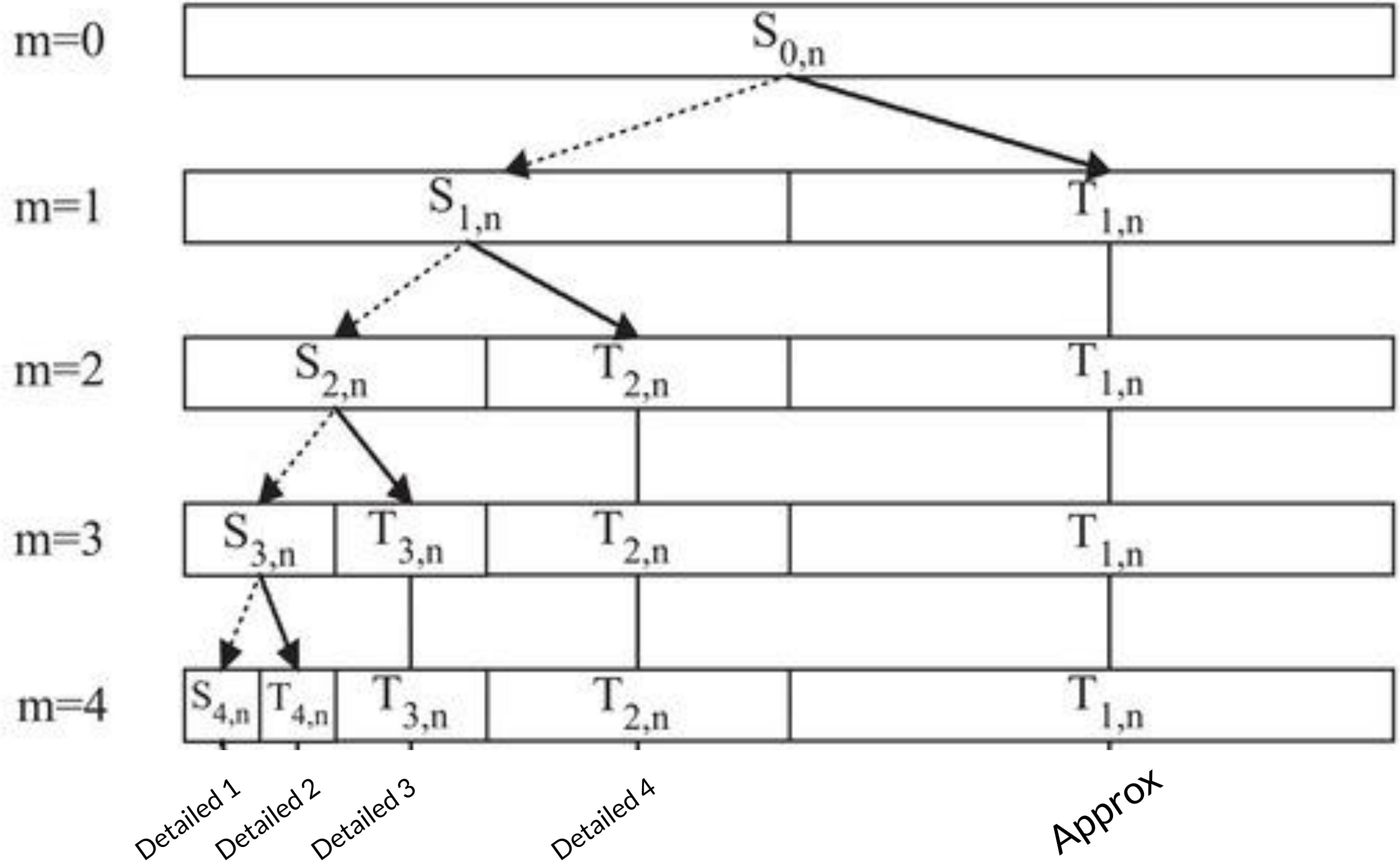
```
>>> help appcoef
 appcoef Extract 1-D approximation coefficients. appcoef computes the approximation coefficients of a
one-dimensional signal.

      A = appcoef(C,L,'wname',N) computes the approximation
      coefficients at level N using the wavelet decomposition
      structure [C,L] (see WAVEDEC).
      'wname' is a character vector containing the wavelet name.
      Level N must be an integer such that 0 <= N <= length(L)-2.

      A = appcoef(C,L,'wname') extracts the approximation
      coefficients at the last level length(L)-2.

      Instead of giving the wavelet name, you can give the filters.
      For A = appcoef(C,L,Lo_R,Hi_R) or
      A = appcoef(C,L,Lo_R,Hi_R,N),
      Lo_R is the reconstruction low-pass filter and
      Hi_R is the reconstruction high-pass filter.

      See also detcoef, wavedec.
```

```
[c,l] = wavedec(ssn,4,'db4');                    % level 4 decomposition
approx = appcoef(c,l,'db4');                     % approximation coefficients
[cd1,cd2,cd3,cd4] = detcoef(c,l,[1 2 3 4]);     % detailed coefficients
```

```
>>> help detcoef
 detcoef Extract 1-D detail coefficients.
     D = detcoef(C,L,N) extracts the detail coefficients at level N from the wavelet decomposition
     structure [C,L].
     See WAVEDEC for more information on C and L.
     Level N must be an integer such that 1 <= N <= NMAX
     where NMAX = length(L)-2.

     D = detcoef(C,L) extracts the detail coefficients at last level NMAX.

     If N is a vector of integers such that 1 <= N(j) <= NMAX:

     DCELL = detcoef(C,L,N,'cells') returns a cell array where DCELL{j} contains the coefficients of detail N(j).

     If length(N)>1, DCELL = detcoef(C,L,N) is equivalent to
     DCELL = detcoef(C,L,N,'cells').

     DCELL = detcoef(C,L,'cells') is equivalent to
     DCELL = detcoef(C,L,[1:NMAX])

     [D1,...,Dp] = detcoef(C,L,[N(1),...,N(p)]) extracts the details
     coefficients at levels [N(1),...,N(p)].
```

```
[c,l] = wavedec(ssn,4,'db4');                  % level 4 decomposition
approx = appcoef(c,l,'db4');                   % approximation coefficients
[cd1,cd2,cd3,cd4] = detcoef(c,l,[1 2 3 4]);    % detailed coefficients

%coefficients plot
figure()
subplot(5,1,1)
plot(approx)
title('db4 Approximation Coefficients','linewidth' ,1.5)
subplot(5,1,2)
plot(cd4)
title('Level 4 Detail Coefficients','linewidth' ,1.5)
subplot(5,1,3)
plot(cd3)
title('Level 3 Detail Coefficients','linewidth' ,1.5)
subplot(5,1,4)
plot(cd2)
title('Level 2 Detail Coefficients','linewidth' ,1.5)
subplot(5,1,5)
plot(cd1)
title('Level 1 Detail Coefficients','linewidth' ,1.5)
```


db4 Approximation Coefficients
Level 4 Detail Coefficients
Level 3 Detail Coefficients
Level 2 Detail Coefficients
Level 1 Detail Coefficients

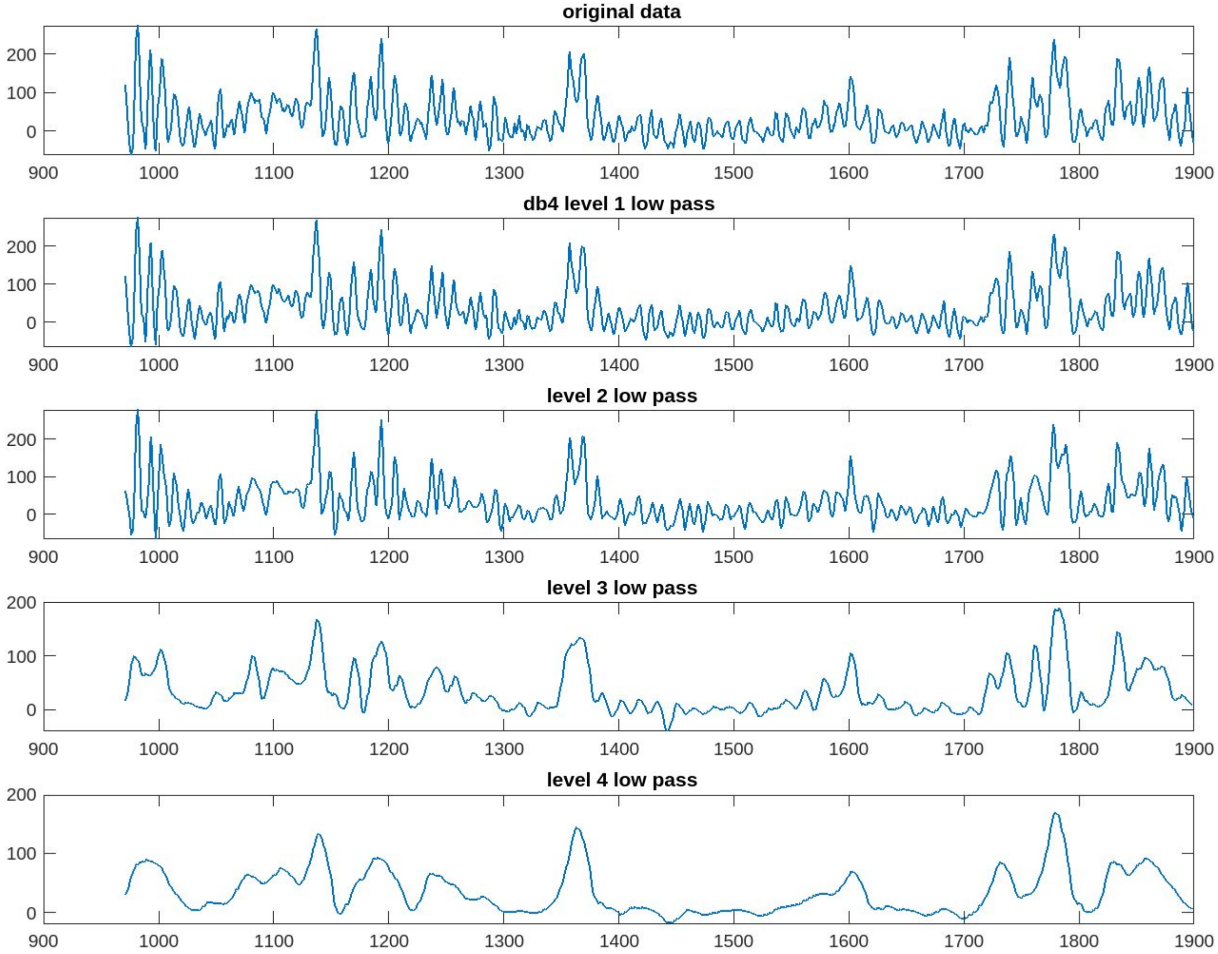

m=0
m=1
m=2
m=3
m=4
S0,n
S1,n
T1,n
S2,n
T2,n
T1,n
S3,n
T3,n
T2,n
T1,n
S4,n
T4,n
T3,n
T2,n
T1,n
Detailed 1
Detailed 2
Detailed 3
Detailed 4
Approx

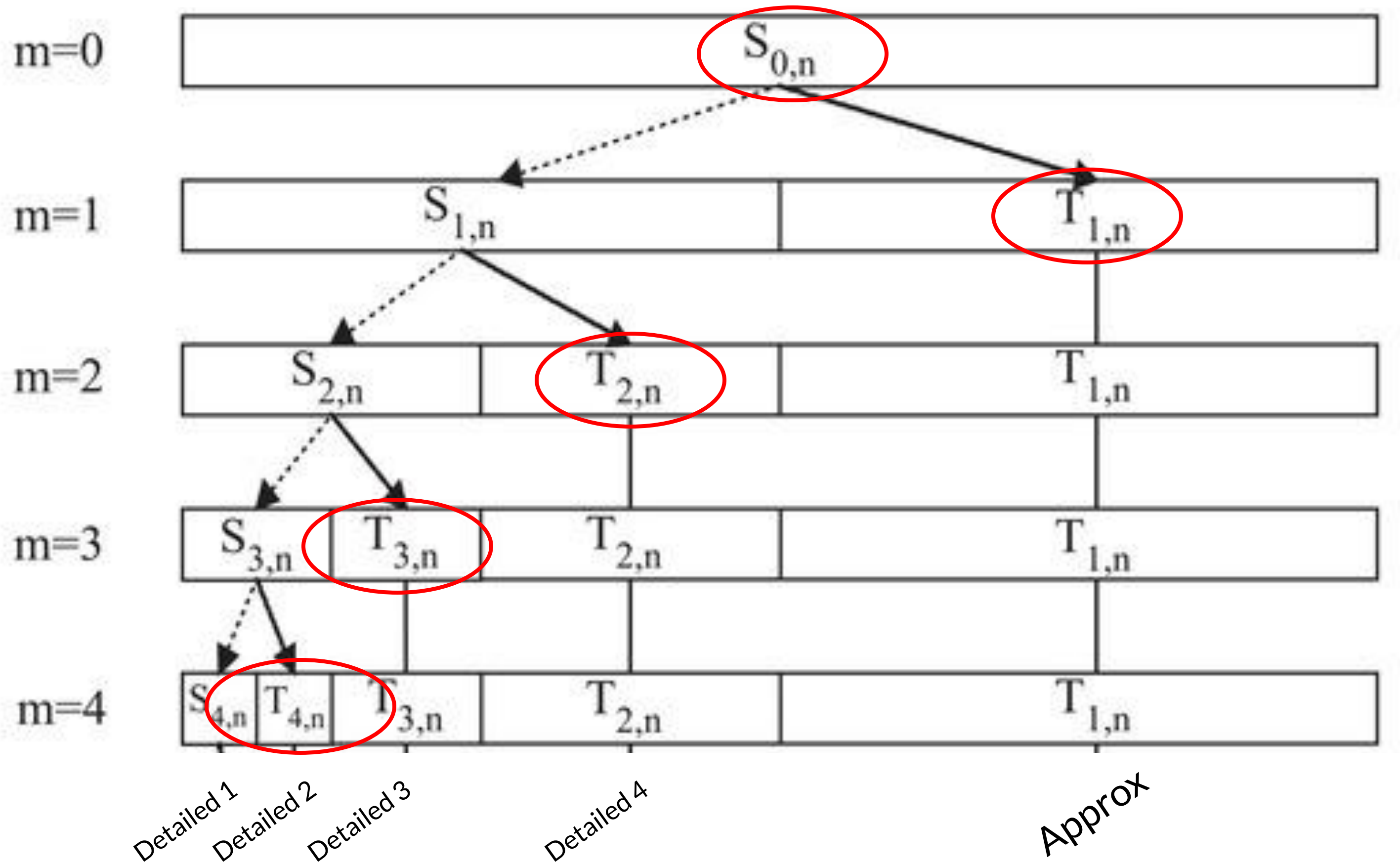
```matlab
% low pass and high pass filtered signal after reconstruction of db4 coefficients
for i =1:5
level=i;
[c,l]=wavedec(ssn,level,'db4'); %wavelet decomposition
lp=wrcoef('a',c,l,'db4', level); %reconstruction of average (low-pass) coefficient
low_pass(:,i)=lp;
hp1=ssn-lp; %reconstruction of high pass filtered data (fluctuations)
high_pass(:,i)=(hp1);
end
```

```
>>> help wrcoef
 wrcoef Reconstruct single branch from 1-D wavelet coefficients.
      wrcoef reconstructs the coefficients of a 1-D signal,given a wavelet decomposition structure (C and L) and
      either a specified wavelet ('wname', see WFILTERS for more information) or specified reconstruction filters
      (Lo_R and Hi_R).

      X = wrcoef('type',C,L,'wname',N) computes the vector of reconstructed coefficients, based on the wavelet
decomposition structure [C,L] (see WAVEDEC for more information), at level N. 'wname' is a character  vector
containing the name of the wavelet.

      Argument 'type' determines whether approximation ('type' = 'a') or detail ('type' = 'd') coefficients are
      reconstructed. When 'type' = 'a', N is allowed to be 0; otherwise, a strictly positive number N is required.
      Level N must be an integer such that N <= length(L)-2.
      X = wrcoef('type',C,L,Lo_R,Hi_R,N) computes coefficient as above, given the reconstruction you specify.
      X = wrcoef('type',C,L,'wname') and
      X = wrcoef('type',C,L,Lo_R,Hi_R) reconstruct coefficients of maximum level N = length(L)-2.
```

```matlab
figure() %low pass filtered data plot
subplot(5,1,1)
plot(time,ssn,'linewidth',1)
title(' original data')
subplot(5,1,2)
plot(time,low_pass(:,1),'linewidth' ,1)
title('db4 level 1 low pass')
subplot(5,1,3)
plot(time,low_pass(:,2),'linewidth' ,1)
title(' level 2 low pass')
subplot(5,1,4)
plot(time,low_pass(:,3),'linewidth' ,1)
title(' level 3 low pass')
subplot(5,1,5)
plot(time,low_pass(:,4),'linewidth' ,1)
title(' level 4 low pass')
```

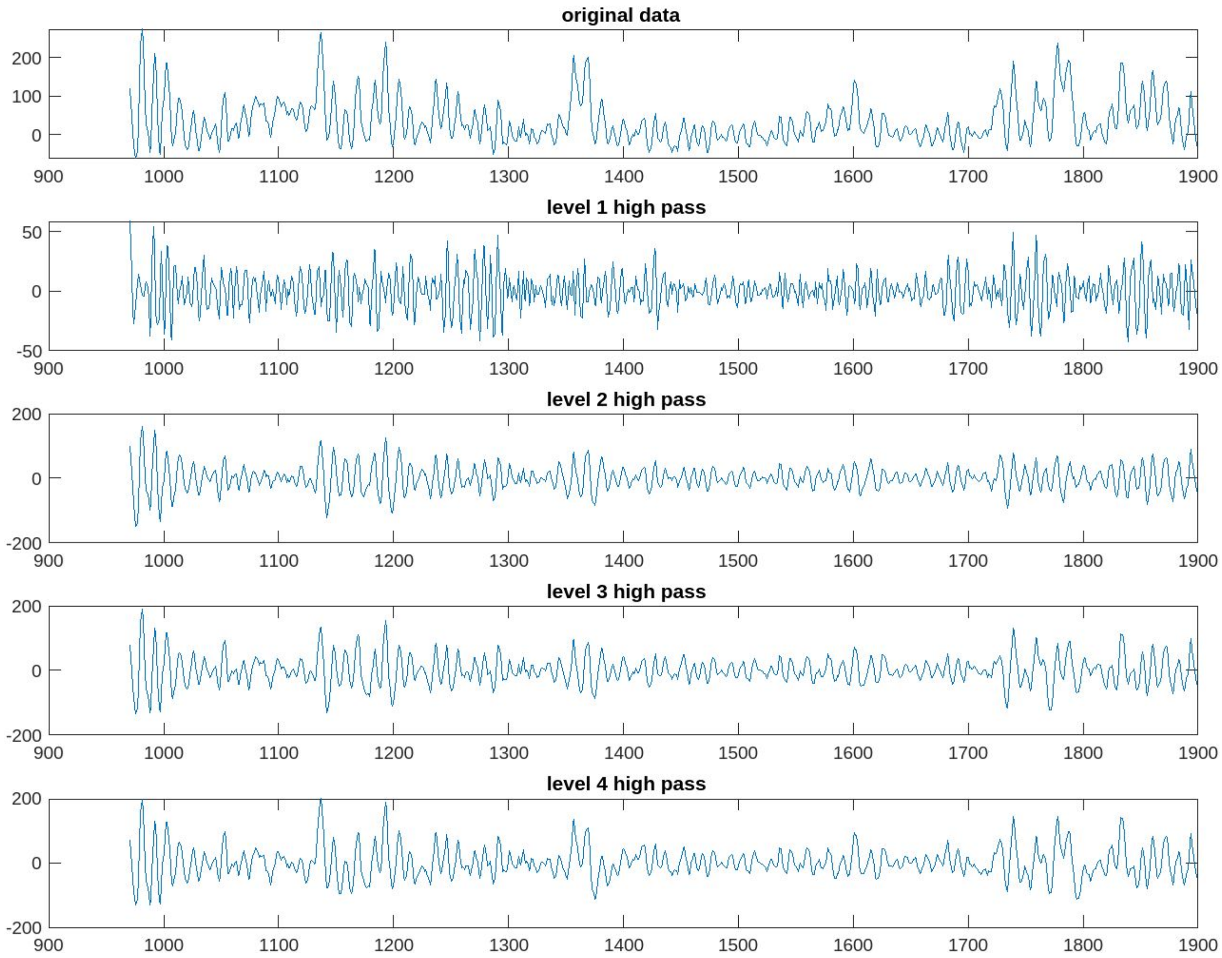
$$f(t) = \sum_{k=-\infty}^{\infty} c_k \phi_k(t) + \sum_{k=-\infty}^{\infty} \sum_{j=0}^{\infty} d_{j,k} \psi_{j,k}(t)$$

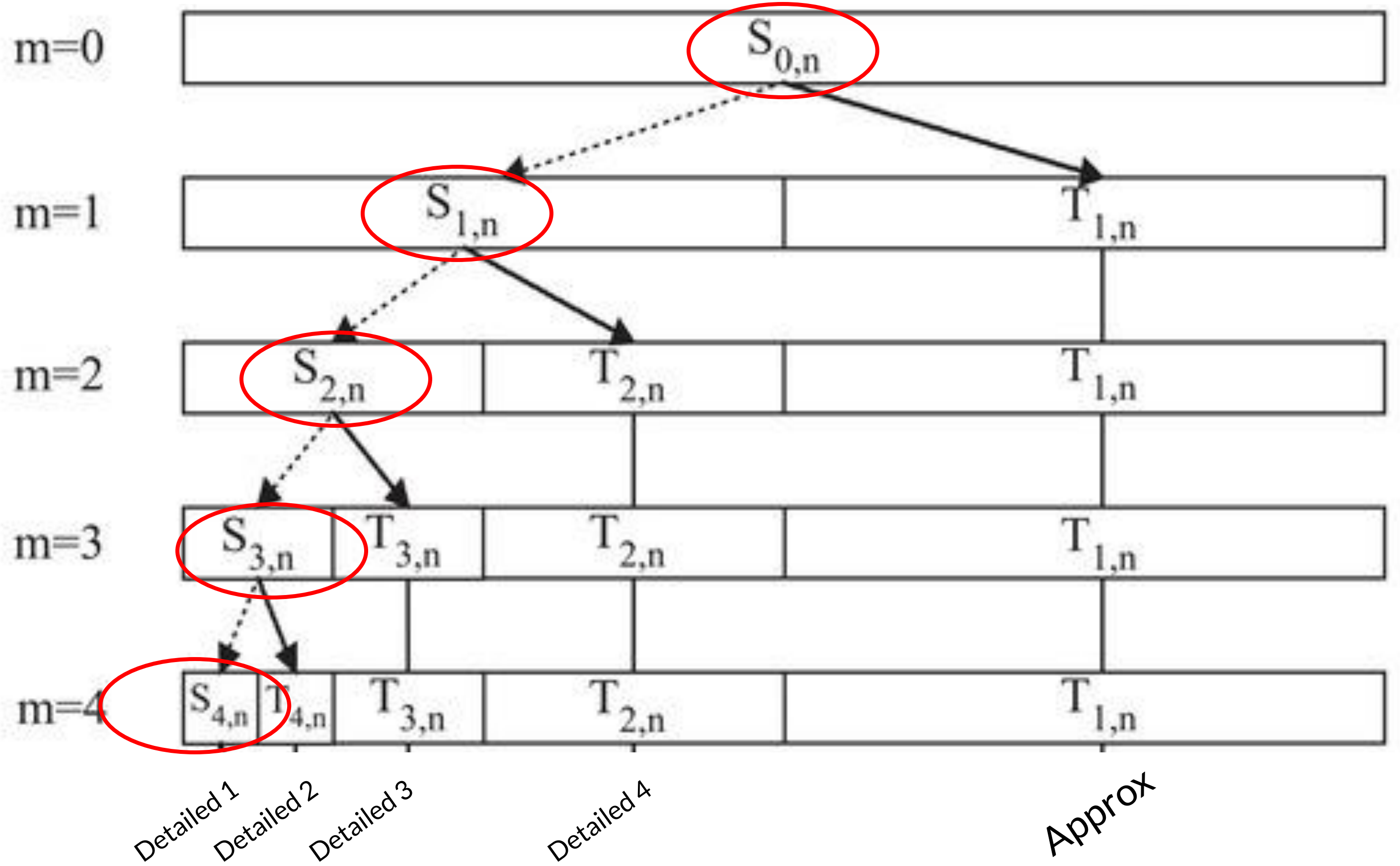

original data
db4 level 1 low pass
level 2 low pass
level 3 low pass
level 4 low pass
200
100
0
900
1000
1100
1200
1300
1400
1500
1600
1700
1800
1900


m=0
$S_{0,n}$
m=1
$S_{1,n}$
$T_{1,n}$
m=2
$S_{2,n}$
$T_{2,n}$
$T_{1,n}$
m=3
$S_{3,n}$
$T_{3,n}$
$T_{2,n}$
$T_{1,n}$
m=4
$S_{4,n}$
$T_{4,n}$
$T_{3,n}$
$T_{2,n}$
$T_{1,n}$
Detailed 1
Detailed 2
Detailed 3
Detailed 4
Approx

```
figure() %fluctuation plot
subplot(5,1,1)
plot(time,ssn(:,1))
title('original data ','linewidth' ,1.5)
subplot(5,1,2)
plot(time,high_pass(:,2))
title(' level 1 high pass','linewidth' ,1.5)
subplot(5,1,3)
plot(time,high_pass(:,3))
title(' level 2 high pass','linewidth' ,1.5)
subplot(5,1,4)
plot(time,high_pass(:,4))
title(' level 3 high pass','linewidth' ,1.5)
subplot(5,1,5)
plot(time,high_pass(:,5))
title(' level 4 high pass','linewidth' ,1.5)
```

$$f(t) = \sum_{k=-\infty}^{\infty} c_k \phi_k(t) + \sum_{k=-\infty}^{\infty} \sum_{j=0}^{\infty} d_{j,k} \psi_{j,k}(t)$$


original data
level 1 high pass
level 2 high pass
level 3 high pass
level 4 high pass
900
1000
1100
1200
1300
1400
1500
1600
1700
1800
1900


m=0
m=1
m=2
m=3
m=4
S0,n
S1,n
T1,n
S2,n
T2,n
S3,n
T3,n
S4,n
T4,n
Detailed 1
Detailed 2
Detailed 3
Detailed 4
Approx

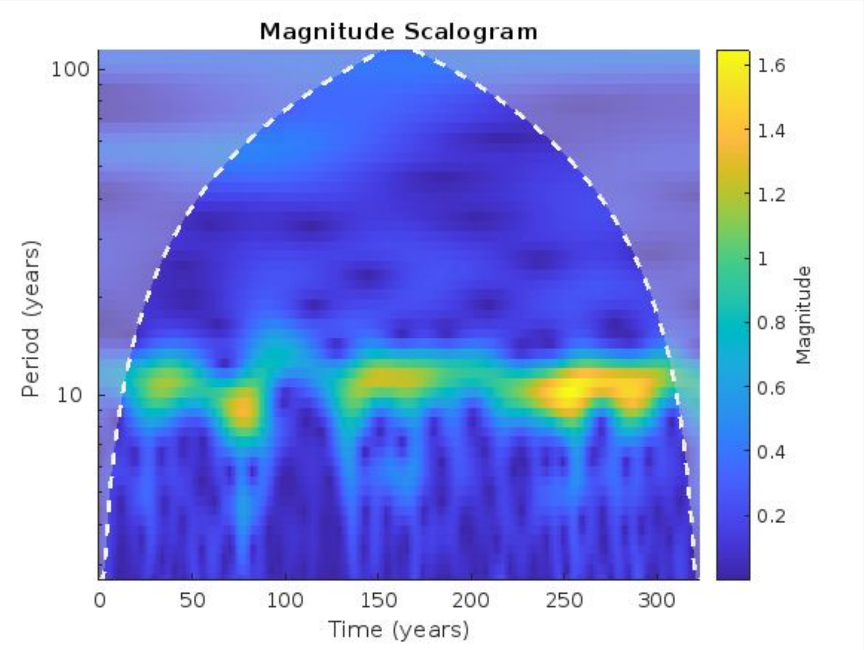
```
%continuous wavelet
figure()
cwt(ssn, "amor", years(1));
[WT,F,COI] =cwt(ssn, "amor", years(1));
```

>>> **help** cwt

cwt Continuous 1-D wavelet transform

cwt(...) with no output arguments plots the absolute value of the continuous wavelet transform, or scalogram, as a function of time and frequency. The cone of influence showing where edge effects become significant is also plotted. Gray regions outside the dashed white lines delineate regions where edge effects are significant. If the input signal is complex-valued, the positive (counterclockwise) and negative (clockwise) components are plotted in separate scalograms. If

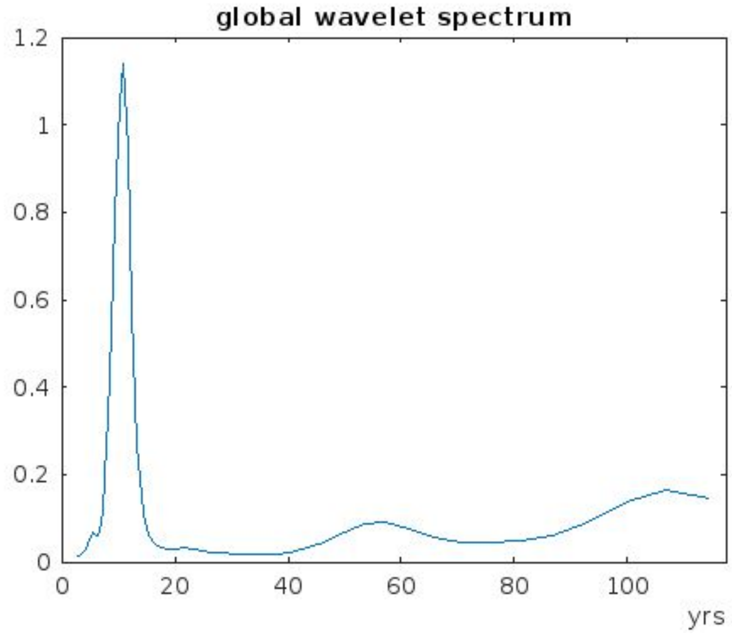
you do not specify a sampling frequency or interval, the frequencies are plotted in cycles/sample. If you supply a sampling frequency, Fs, the scalogram is plotted in hertz.**If you supply a sampling interval using a duration, the scalogram is plotted as a function of time and periods**. If the input to cwt is a timetable, the scalogram is plotted as a function of frequency in hertz and uses the RowTimes of the timetable as the basis for the time axis. The frequency or period axis in the scalogram uses a log10 scale.

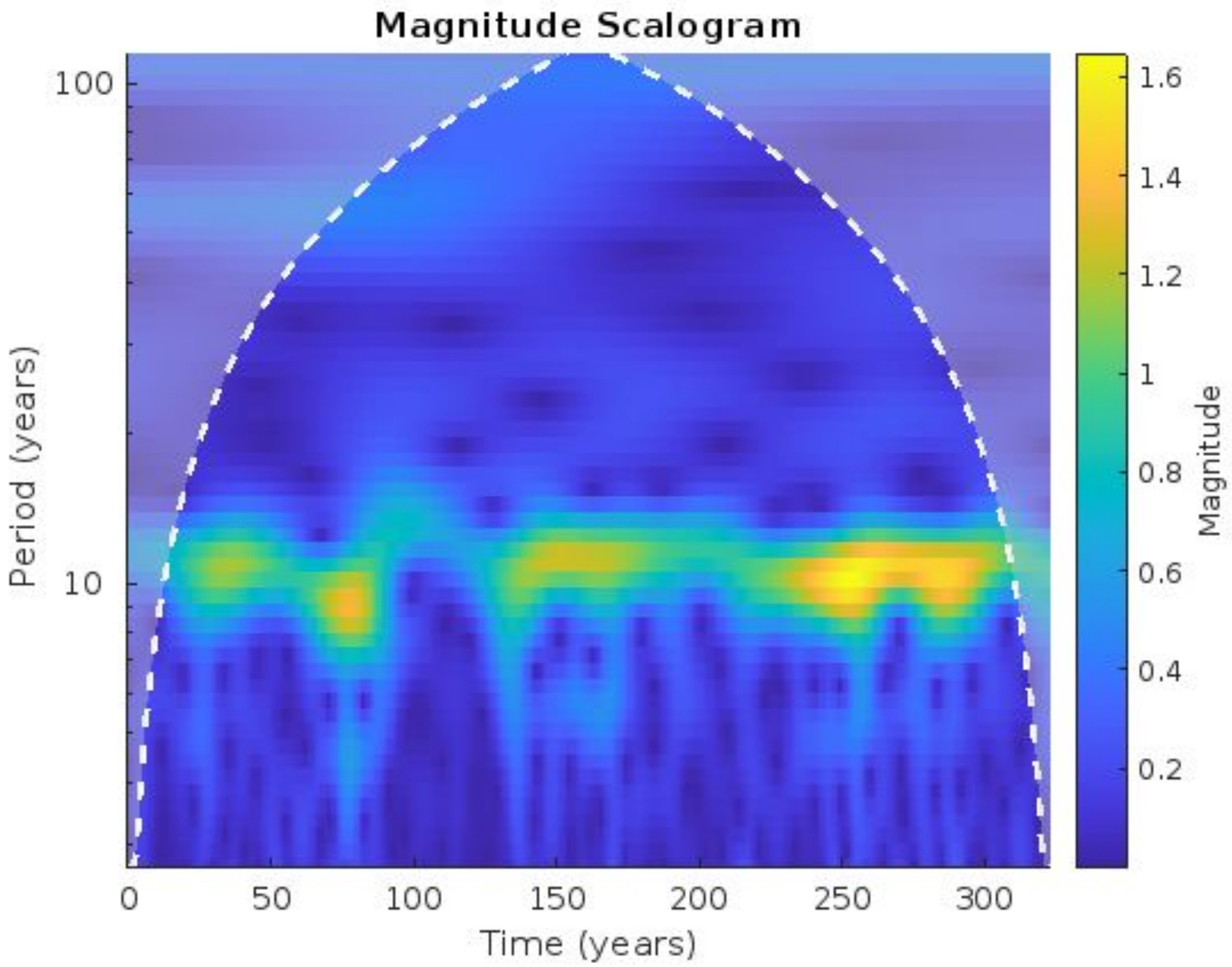


```
%continuous wavelet
figure()
cwt(ssn, "amor", years(1));
[WT,F,COI] =cwt(ssn, "amor", years(1));
```

```
>>> help cwt
 cwt Continuous 1-D wavelet transform
     WT = cwt(X) returns the continuous wavelet transform (cwt) of X.

     [...] = cwt(X,WAVNAME) uses the wavelet corresponding to the string WAVNAME. Valid options for WAVNAME are:
'morse', 'amor', or 'bump'. If you do not specify WAVNAME, WAVNAME defaults to 'morse'.

     [...,F] = cwt(...,Fs) specifies the sampling frequency, Fs, in hertz as a positive scalar and returns the
scale-to-frequency conversions in hertz, F. If you do not specify a sampling frequency, cwt returns F in
cycles/sample. If the input X is complex, the scale-to-frequency conversions apply to both pages of WT.

     [...,PERIOD] = cwt(...,Ts) uses the positive scalar duration, Ts, to compute the scale-to-period conversions,
PERIOD. PERIOD is an array of durations with the same Format property as Ts.  If the input X is complex, the
scale-to-period conversions apply to both pages of WT.

     [...,F,COI] = cwt(...) returns the cone of influence (COI) in cycles/sample for the wavelet transform. Specify
a sampling frequency, Fs, in hertz, to return the cone of influence in hertz. If the input X    is complex, the COI
applies to both pages of WT.

     [...,PERIOD,COI] = cwt(...,Ts) returns the cone of influence in periods for the wavelet transform. Ts is a
positive scalar duration. COI is an array of durations with the same Format property as Ts. If  the input X is
complex, the COI applies to both pages of WT.
```

```
%global wavelet spectrum

WT_norm = WT.*conj(WT);
WT_norm_avg = mean(WT_norm,2);
plot(F, WT_norm_avg);
title('global wavelet spectrum','linewidth' ,1.5)
```





Magnitude Scalogram
Period (years)
100
10
0
50
100
150
200
250
300
Time (years)
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
Magnitude

Rhythms in Your Hands!