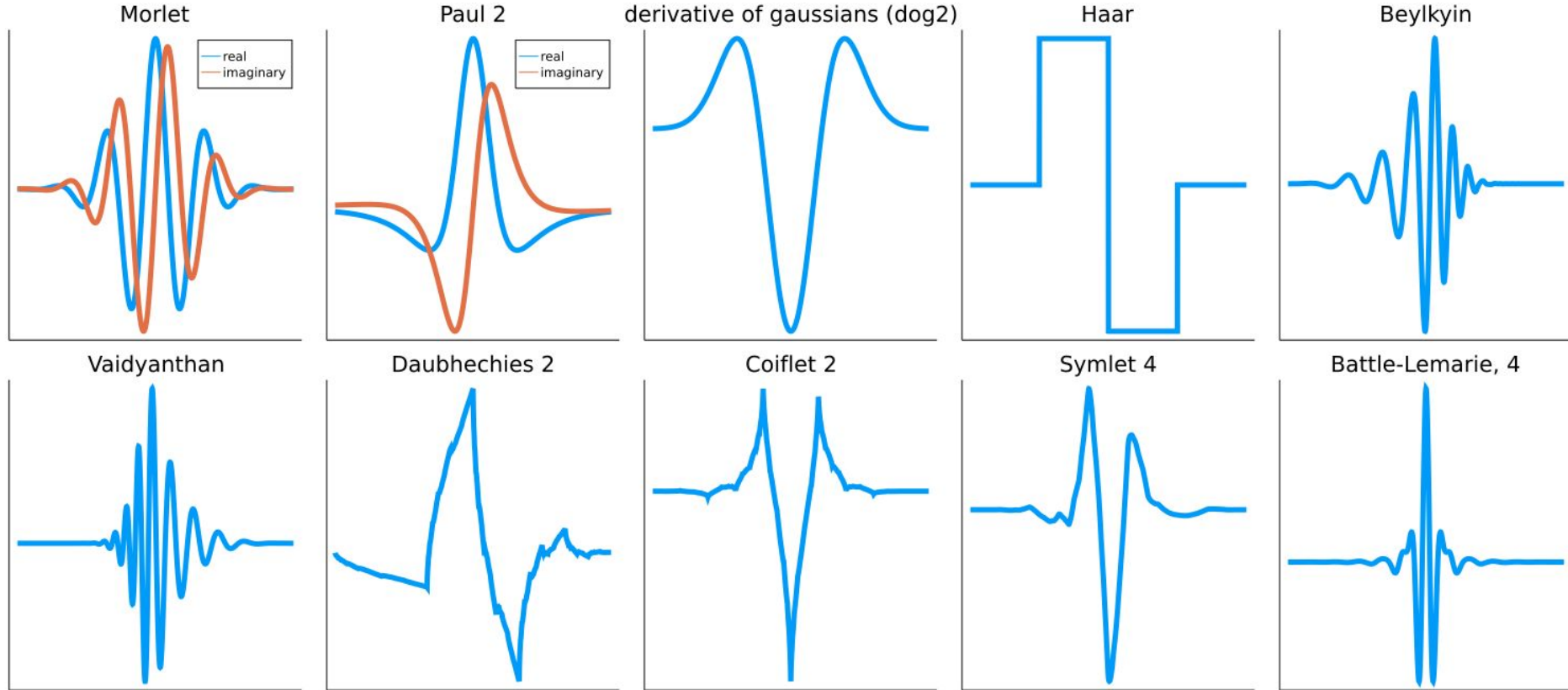


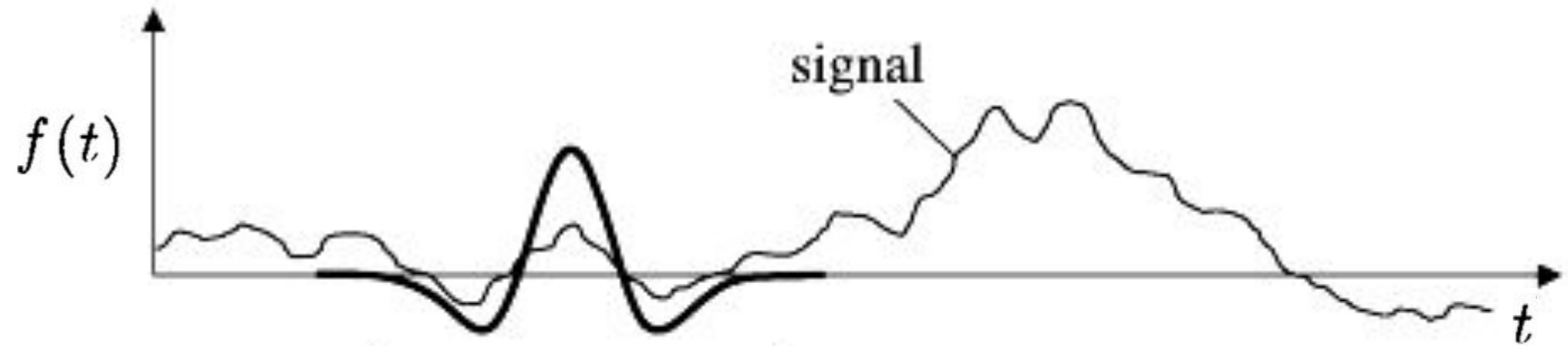
Catch the Rhythm with Wavelets

SS4202 Space Astronomy
Spring 2023
IISER Kolkata

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



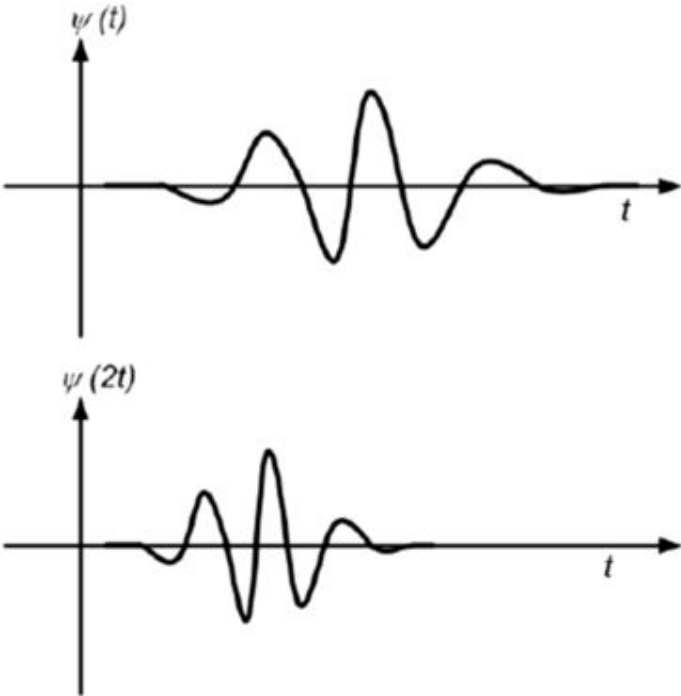
Local matching of wavelet and signal can extract useful information.



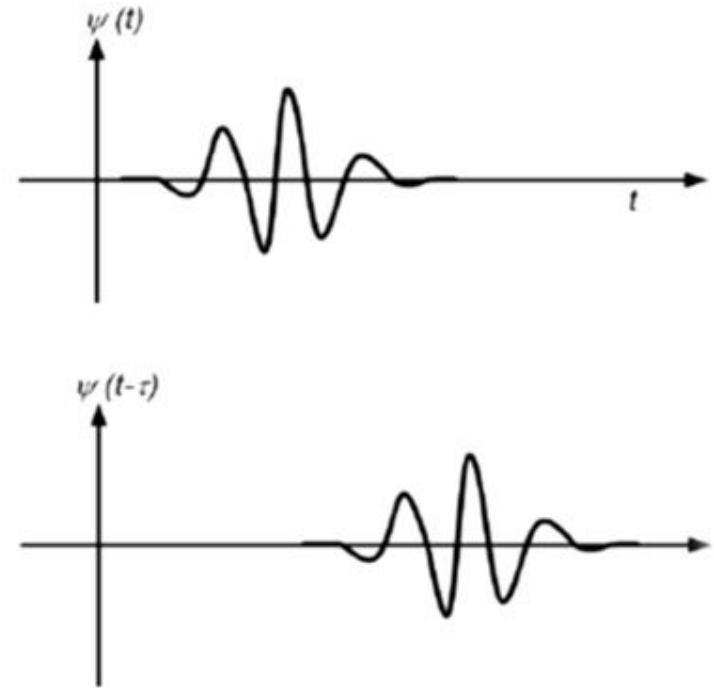
$$f(t) = \sum_{k=-\infty}^{\infty} c_k \phi_k(t) + \sum_{k=-\infty}^{\infty} \sum_{j=0}^{\infty} d_{j,k} \psi_{j,k}(t)$$

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

Scaling



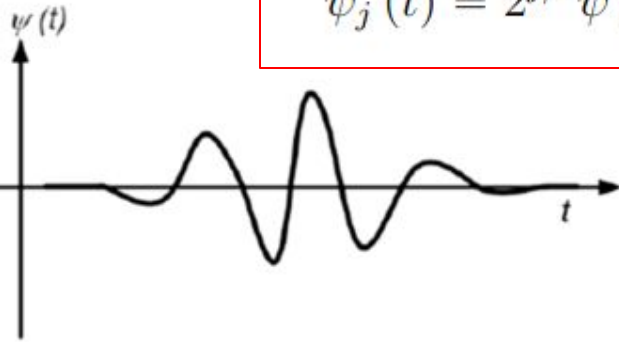
Translation



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

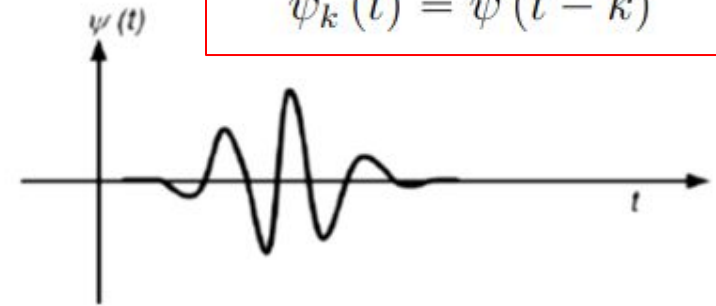
Scaling

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

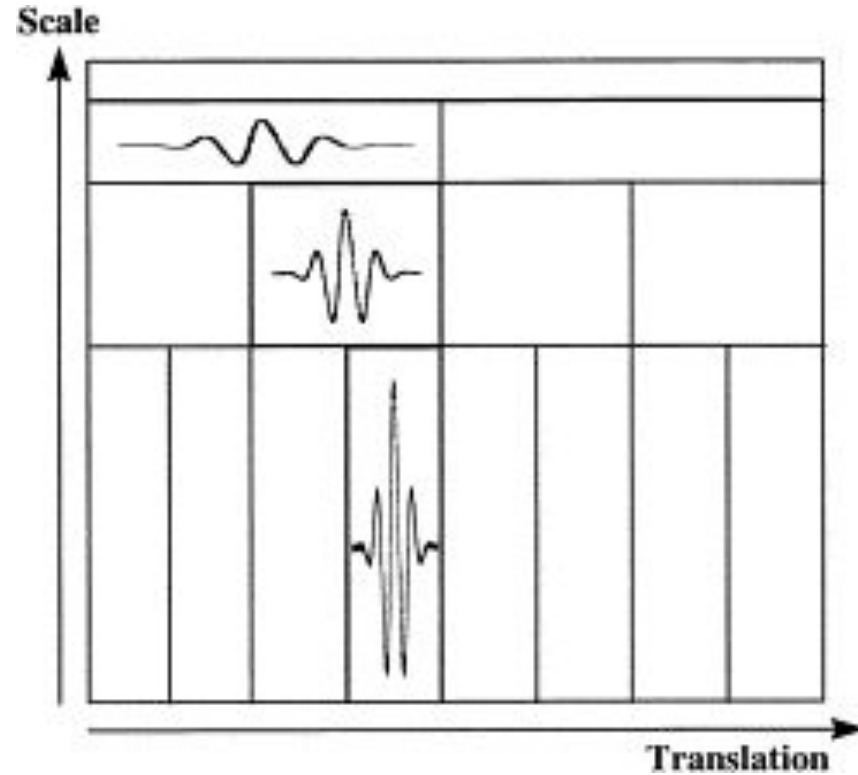


Translation

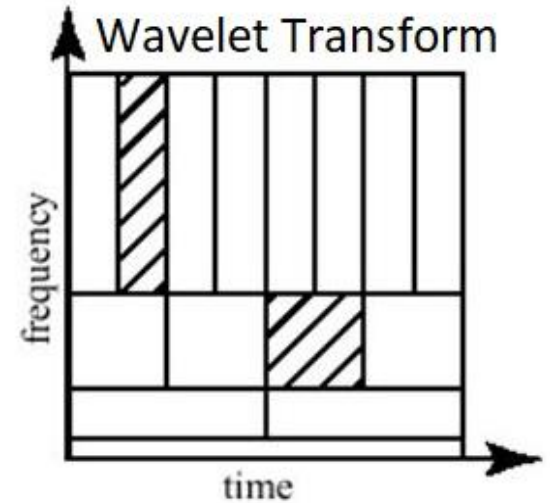
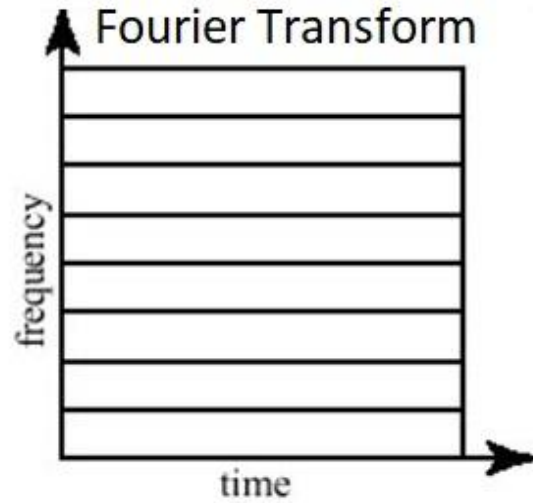
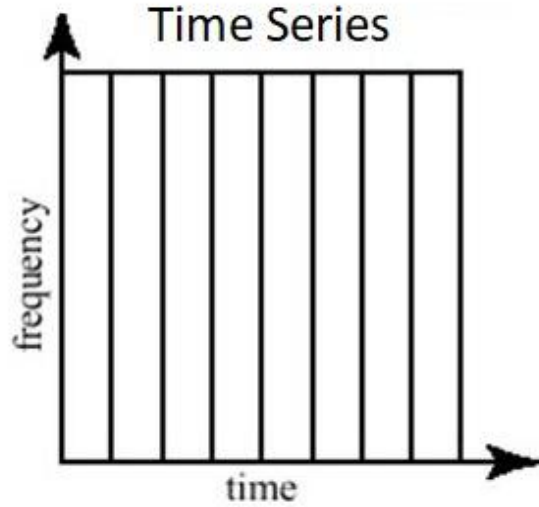
$$\psi_k(t) = \psi(t - k)$$



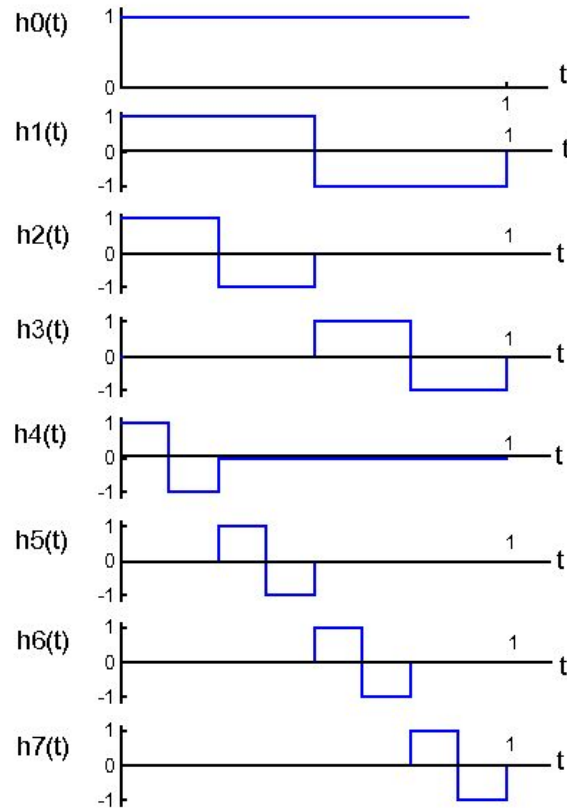
Use a **Thorn** to ~~remove~~ Detect a **Thorn**



Time-Frequency Localization



Haar Wavelet Family



Father $\phi(t)$

Mother $\psi(t)$

Daughter

Daughter

Translation

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

Translation + Scaling

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

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

Wavelet Family – Some Characteristics

Father $\phi(t)$

Mother $\psi(t)$

Def.

$$\int_{-\infty}^{\infty} \phi(t) dt = \text{constant}$$

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

Square
integrability

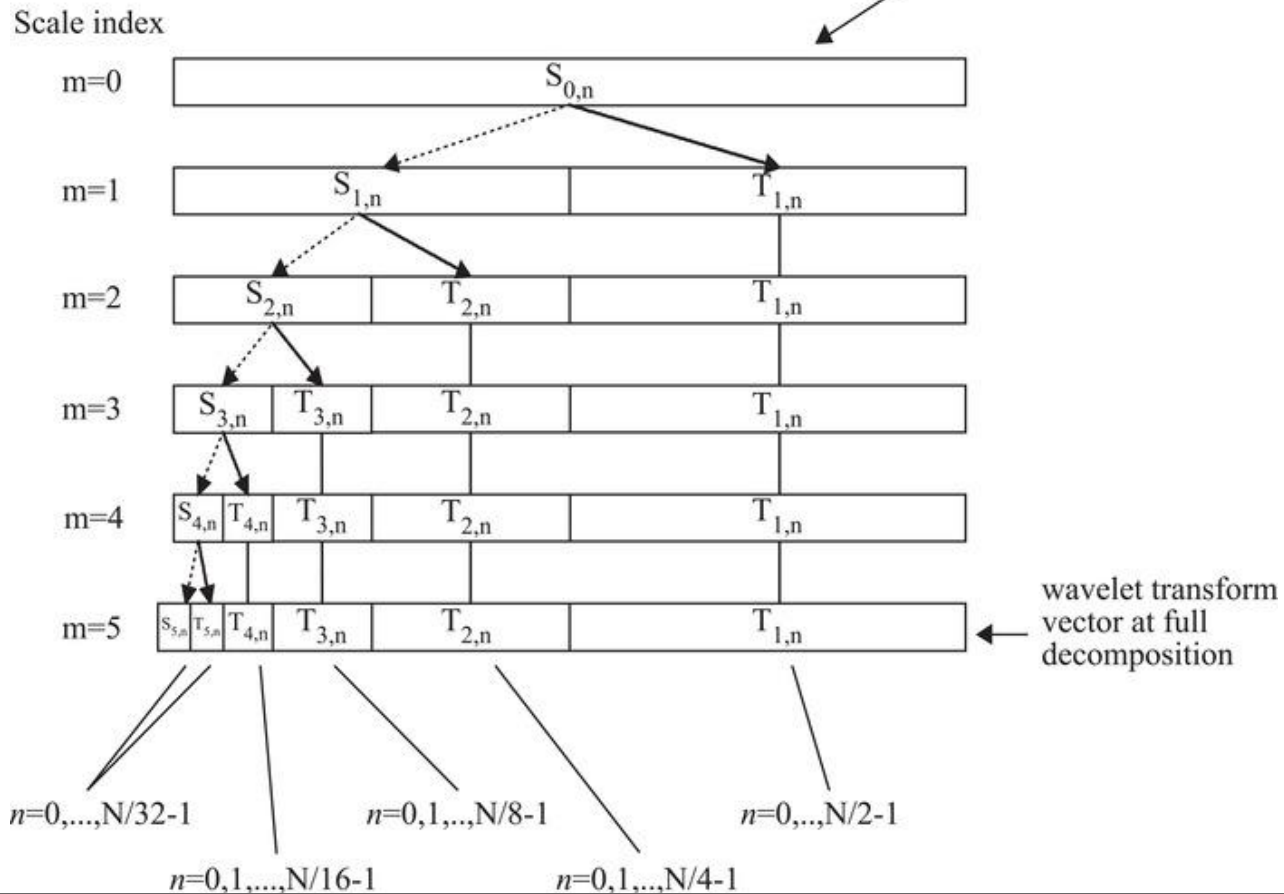
$$\int_{-\infty}^{\infty} |\phi(t)|^2 dt < \infty$$

$$\int_{-\infty}^{\infty} |\psi(t)|^2 dt < \infty$$

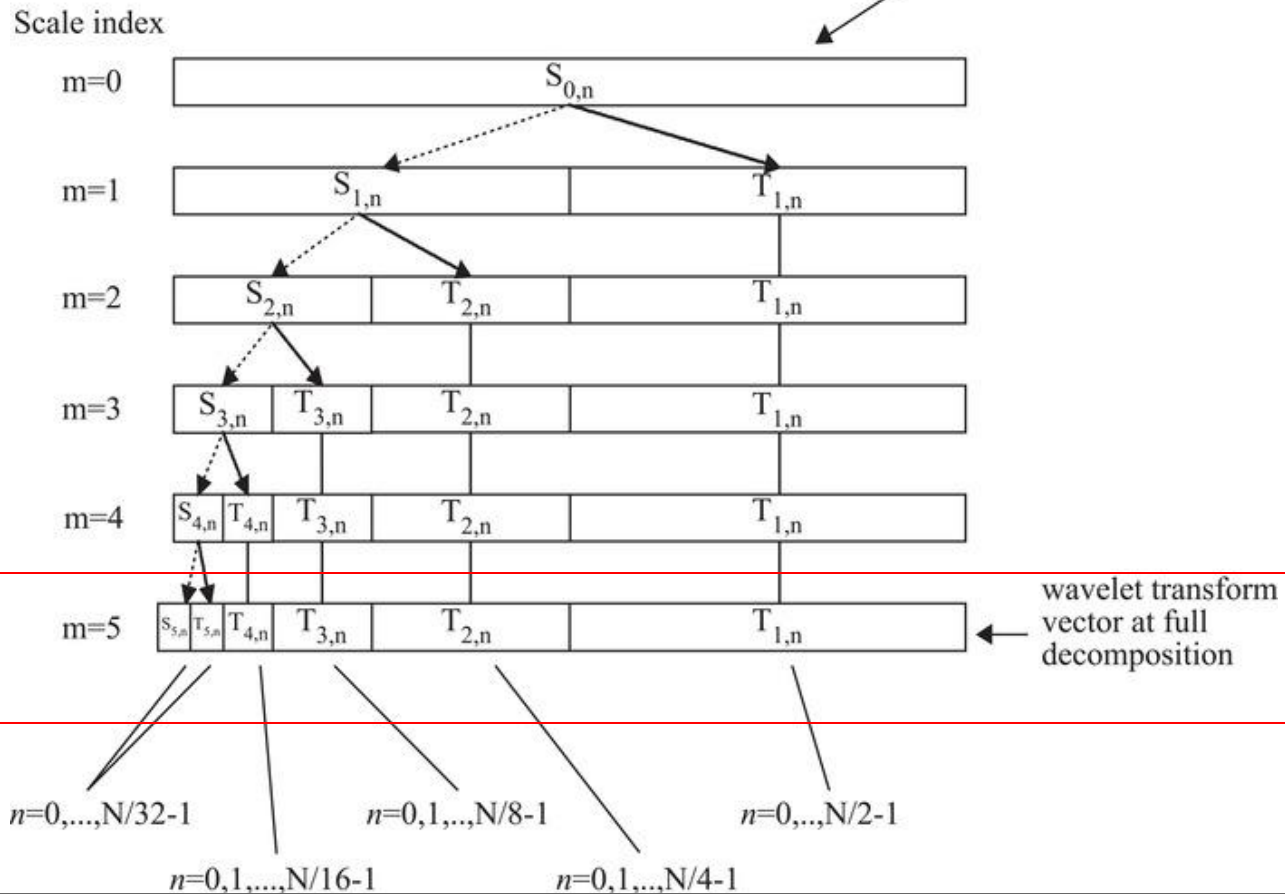
Orthogonality

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

The Big Picture: Wavelet Decomposition



The Big Picture: Wavelet Decomposition



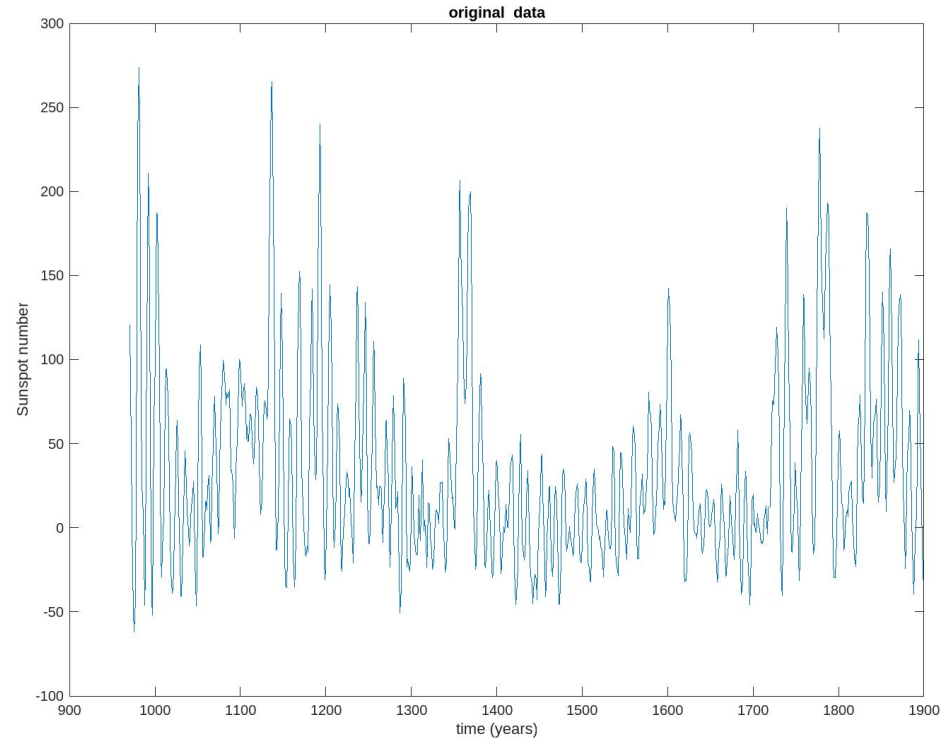
Hands On MATLAB

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

- 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)

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

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



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

```

13  [c,l] = wavedec(ssn,4,'db4');           % level 4 decomposition
14  approx = appcoef(c,l,'db4');           % approximation coefficients
15  [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](#).


```

13  [c,l] = wavedec(ssn,4,'db4');           % level 4 decomposition
14  approx = appcoef(c,l,'db4');           % approximation coefficients
15  [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 $0 \leq N \leq \text{length}(L)-2$.

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

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](#).

```

13  [c,l] = wavedec(ssn,4,'db4');           % level 4 decomposition
14  approx = appcoef(c,l,'db4');           % approximation coefficients
15  [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 $1 \leq N \leq \text{NMAX}$

where $\text{NMAX} = \text{length}(L)-2$.

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

If N is a vector of integers such that $1 \leq N(j) \leq \text{NMAX}$:

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

If $\text{length}(N) > 1$, DCELL = detcoef(C,L,N) is equivalent to

DCELL = detcoef(C,L,N,'cells').

DCELL = detcoef(C,L,'cells') is equivalent to

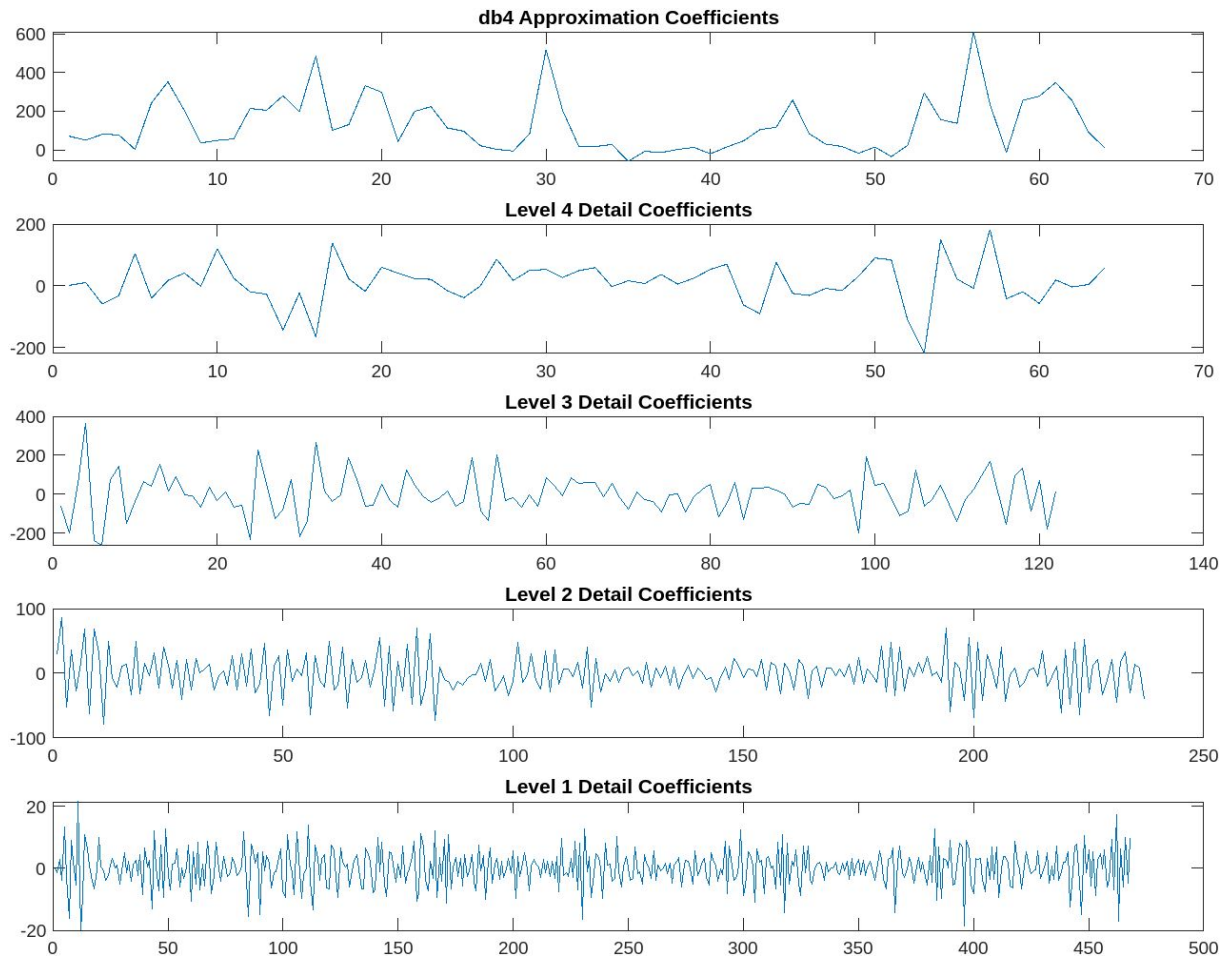
DCELL = detcoef(C,L,[1:NMAX])

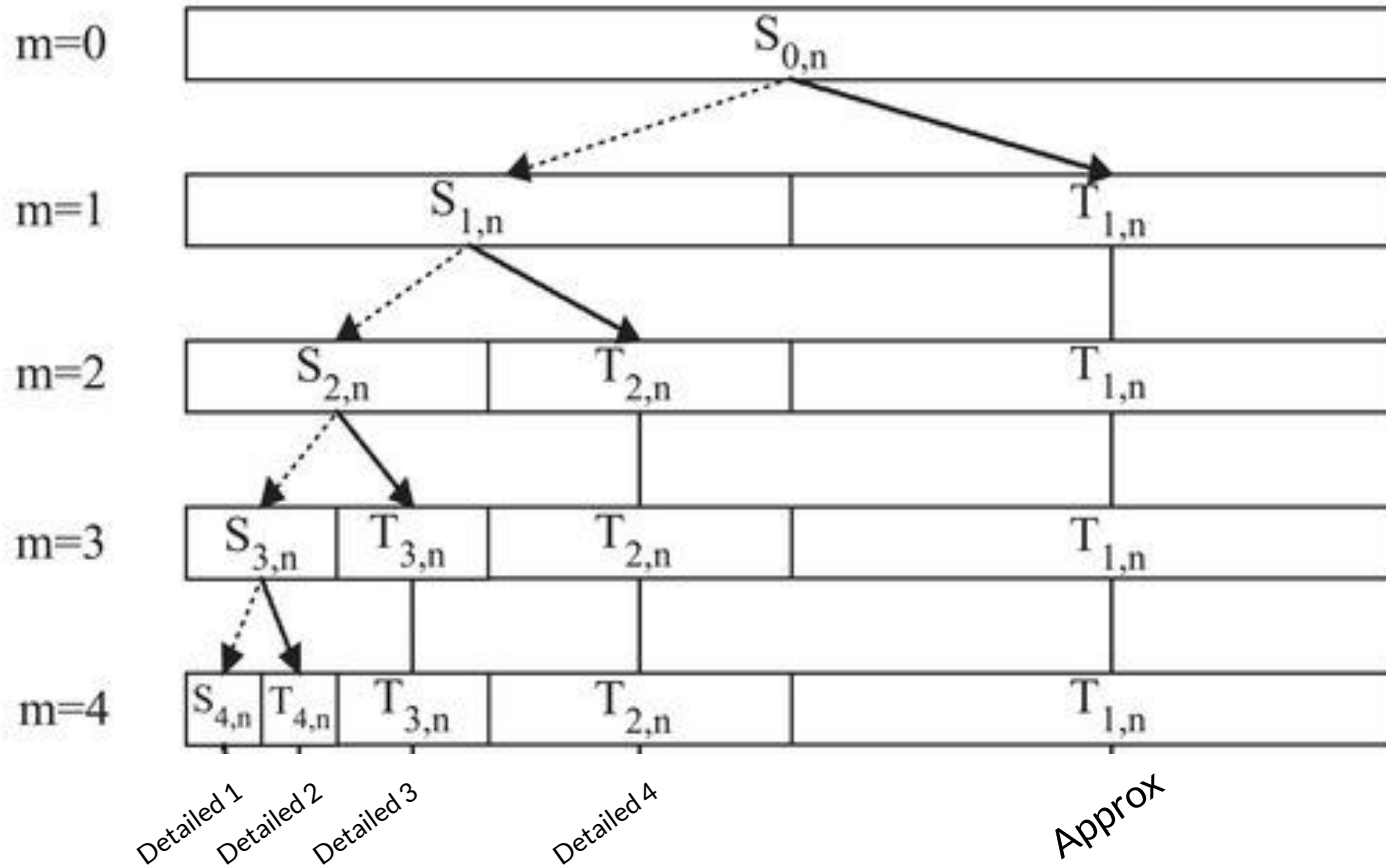
[D1,...,Dp] = detcoef(C,L,[N(1),...,N(p)]) extracts the details coefficients at levels [N(1),...,N(p)].

```

13  [c,1] = wavedec(ssn,4,'db4');           % level 4 decomposition
14  approx = appcoef(c,1,'db4');           % approximation coefficients
15  [cd1,cd2,cd3,cd4] = detcoef(c,1,[1 2 3 4]); % detailed coefficients
16
17  %coefficients plot
18  figure()
19  subplot(5,1,1)
20  plot(approx)
21  title('db4 Approximation Coefficients','linewidth' ,1.5)
22  subplot(5,1,2)
23  plot(cd4)
24  title('Level 4 Detail Coefficients','linewidth' ,1.5)
25  subplot(5,1,3)
26  plot(cd3)
27  title('Level 3 Detail Coefficients','linewidth' ,1.5)
28  subplot(5,1,4)
29  plot(cd2)
30  title('Level 2 Detail Coefficients','linewidth' ,1.5)
31  subplot(5,1,5)
32  plot(cd1)
33  title('Level 1 Detail Coefficients','linewidth' ,1.5)

```





```

34 % low pass and high pass filtered signal after reconstruction of db4 coefficients
35 for i =1:5
36 level=i;
37 [c,l]=wavedec(ssn,level,'db4'); %wavelet decomposition
38 lp=wrcoef('a',c,l,'db4', level); %reconstruction of average (low-pass) coefficient
39 low_pass(:,i)=lp;
40 hp1=ssn-lp; %reconstruction of high pass filtered data (fluctuations)
41 high_pass(:,i)=(hp1);
42 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 $N \leq \text{length}(L)-2$.

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

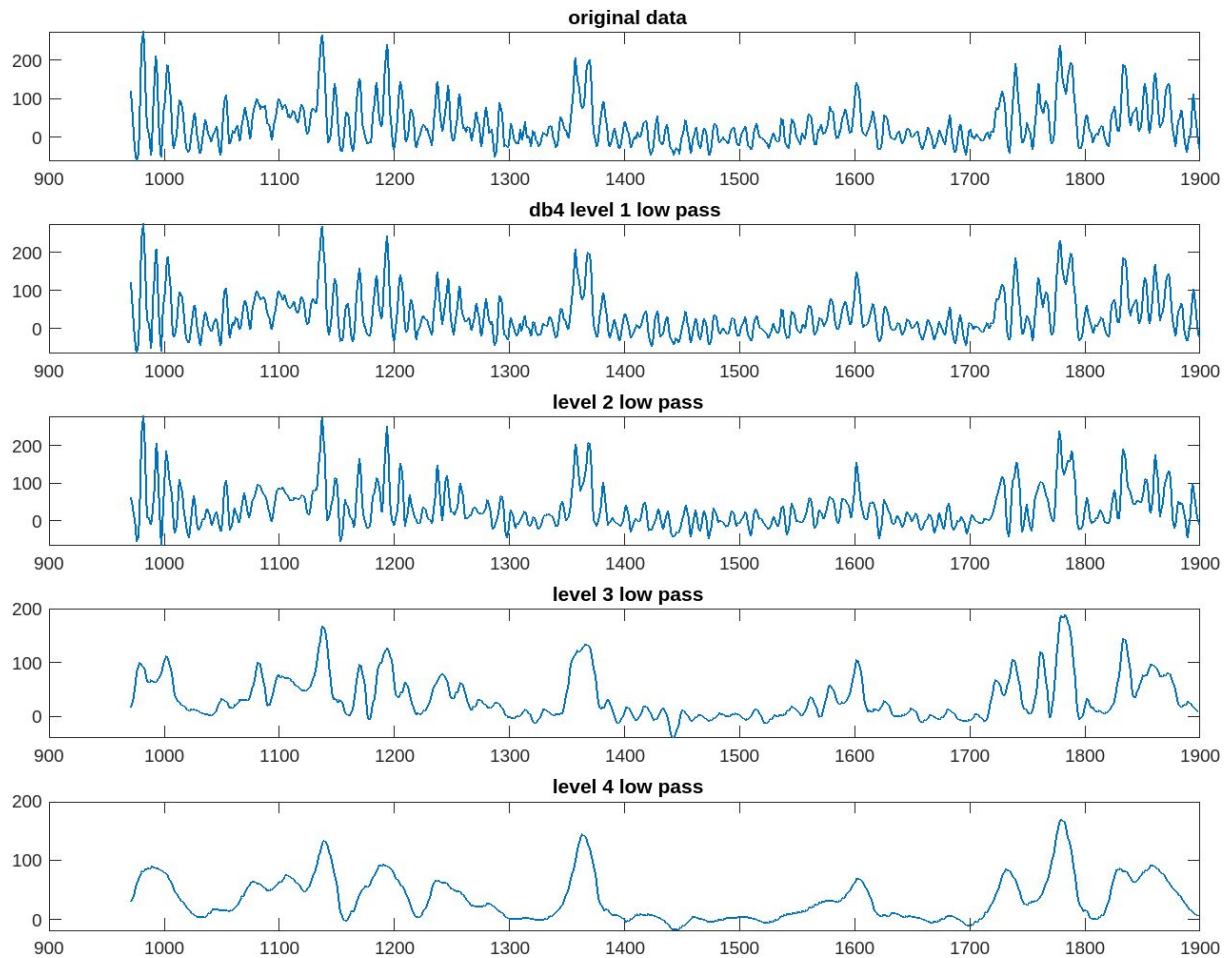
X = wrcoef('type',C,L,Lo_R,Hi_R) reconstruct coefficients of maximum level $N = \text{length}(L)-2$.

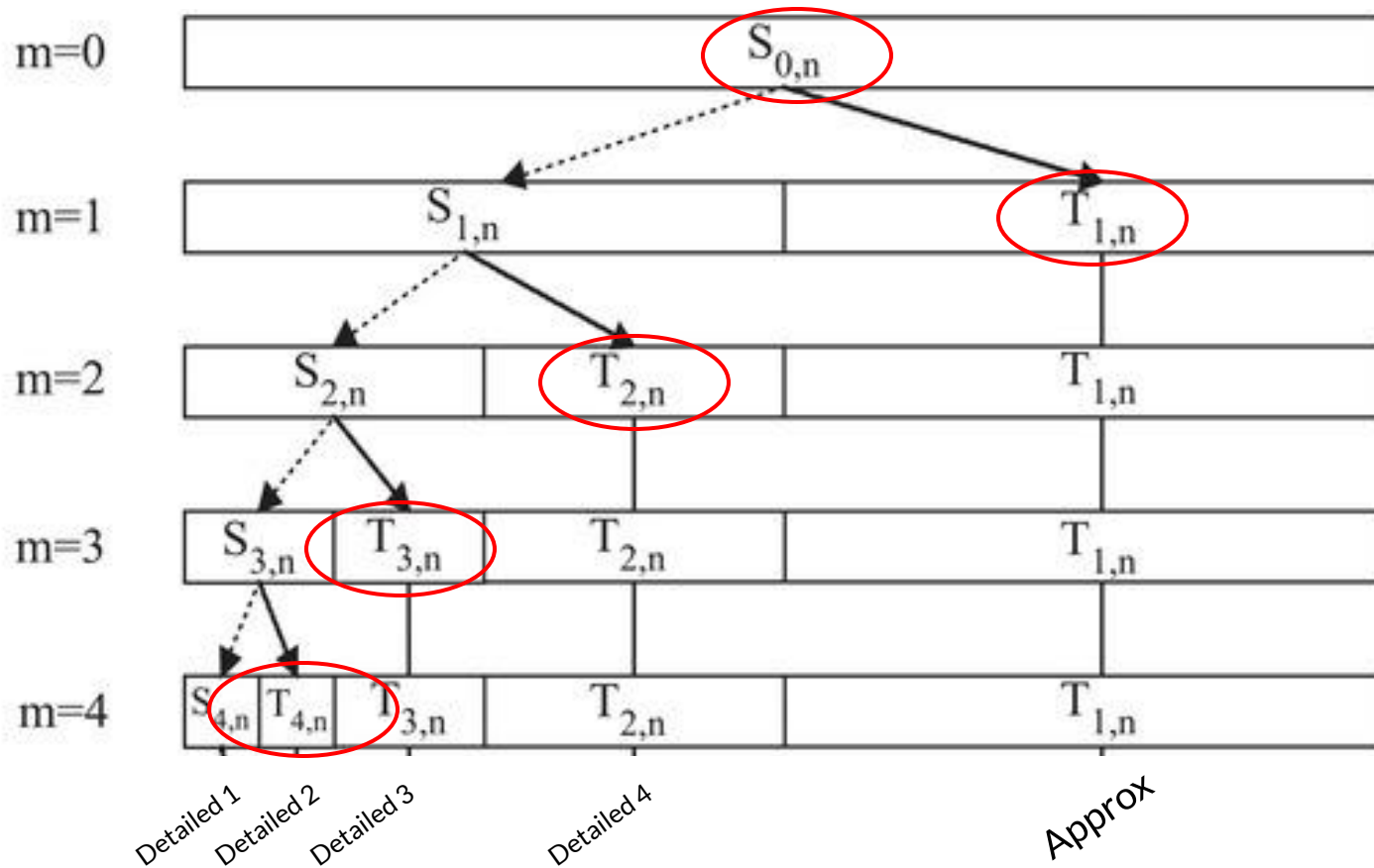
```

44 figure() %low pass filtered data plot
45 subplot(5,1,1)
46 plot(time,ssn,'linewidth',1)
47 title(' original data')
48 subplot(5,1,2)
49 plot(time,low_pass(:,1),'linewidth',1)
50 title('db4 level 1 low pass')
51 subplot(5,1,3)
52 plot(time,low_pass(:,2),'linewidth',1)
53 title(' level 2 low pass')
54 subplot(5,1,4)
55 plot(time,low_pass(:,3),'linewidth',1)
56 title(' level 3 low pass')
57 subplot(5,1,5)
58 plot(time,low_pass(:,4),'linewidth',1)
59 title(' level 4 low pass')

```

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



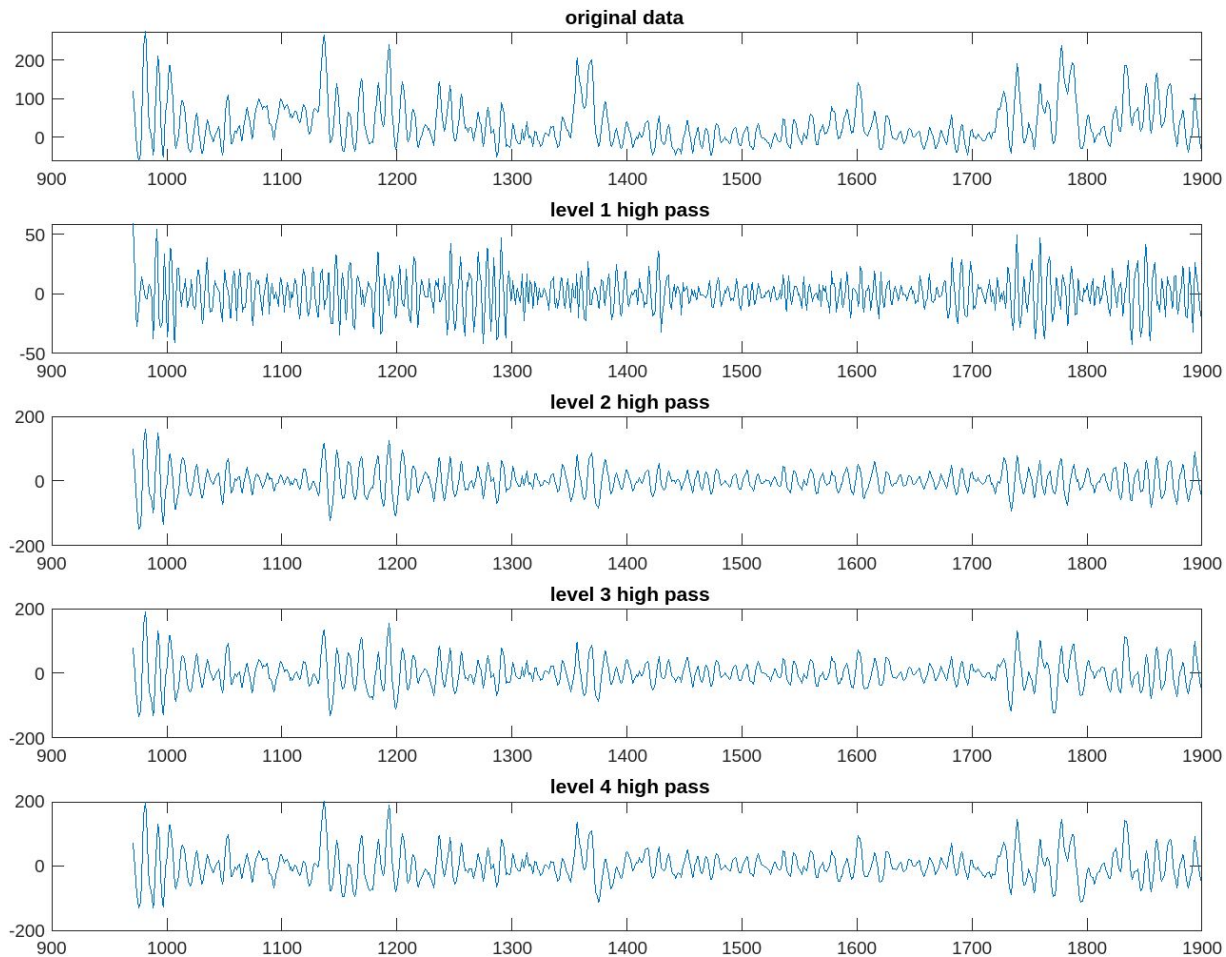


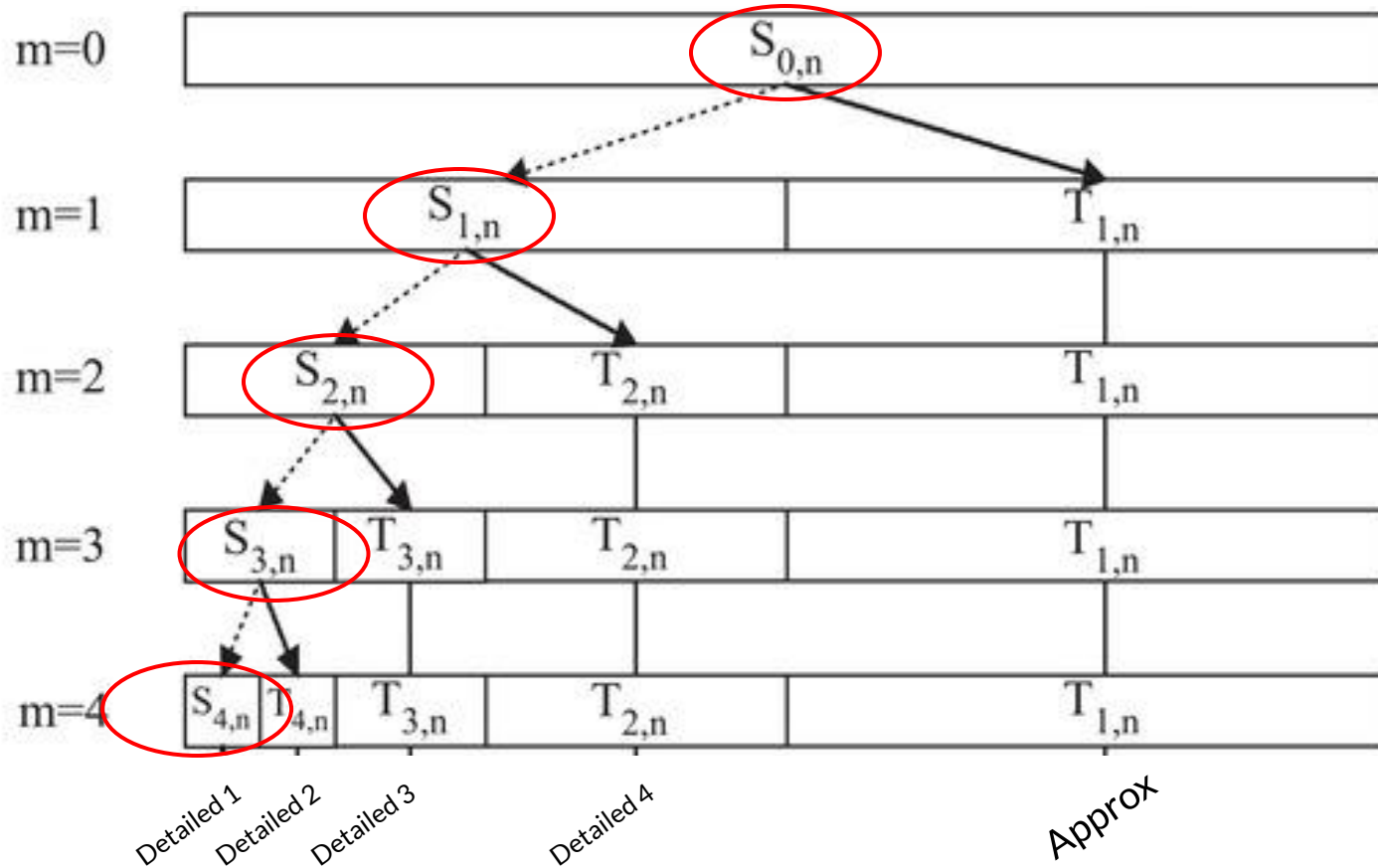
```

60 figure() %fluctuation plot
61 subplot(5,1,1)
62 plot(time,ssn(:,1))
63 title('original data ','linewidth' ,1.5)
64 subplot(5,1,2)
65 plot(time,high_pass(:,2))
66 title(' level 1 high pass','linewidth' ,1.5)
67 subplot(5,1,3)
68 plot(time,high_pass(:,3))
69 title(' level 2 high pass','linewidth' ,1.5)
70 subplot(5,1,4)
71 plot(time,high_pass(:,4))
72 title(' level 3 high pass','linewidth' ,1.5)
73 subplot(5,1,5)
74 plot(time,high_pass(:,5))
75 title(' level 4 high pass','linewidth' ,1.5)

```

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





```

76 %continuous wavelet
77 figure()
78 cwt(ssn, "amor", years(1));
79 [WT,F,COI] =cwt(ssn, "amor", years(1));

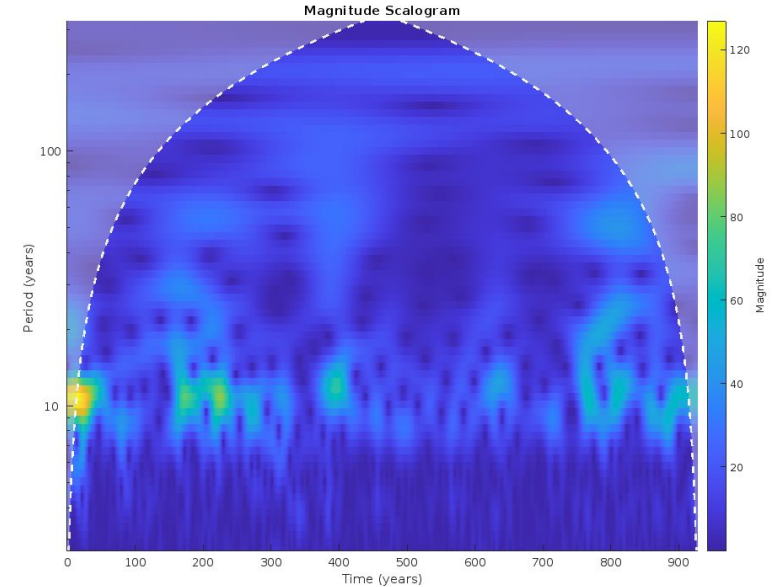
```

```
>>> 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, F_s , 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.



```
76 %continuous wavelet
77 figure()
78 cwt(ssn, "amor", years(1));
79 [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.

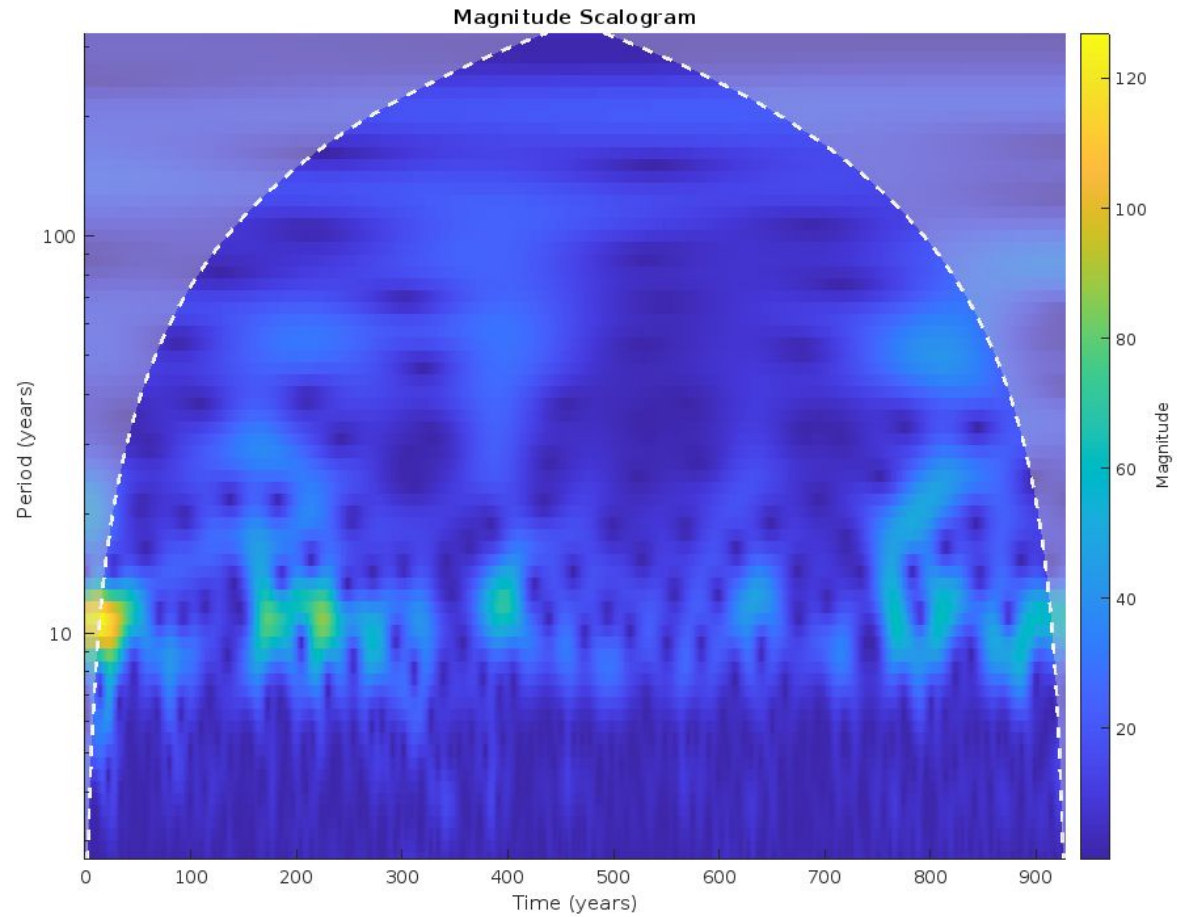
[...] = cwt(X,WAVNAME) uses the wavelet corresponding to the string WAVNAME. Valid options for WAVNAME are: 'morse', 'amor', or 'bump'. If you do not specify WAVNAME, WAVNAME defaults to 'morse'.

[...,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.



The background of the slide features a close-up of two hands, palms facing each other, with a bright green, ethereal glow emanating from the center between them. The hands are positioned symmetrically, with fingers slightly curled. The overall color palette is dark, with deep blues and purples, creating a mysterious and focused atmosphere.

Rhythms in Your Hands!