**实验一 短时傅里叶变化与小波变换**

1. **实验目的：**
   1. 熟悉并掌握短时傅里叶变换的性质、参数以及不同信号的短时傅里叶变换
   2. 熟悉并掌握小波变换的性质、参数以及不同信号的小波变换
2. **实验内容：**
   1. Matlab中的短时傅里叶变换函数spectrogram

S = spectrogram(x)

S = spectrogram(x,window)

S = spectrogram(x,window,noverlap)

S = spectrogram(x,window,noverlap,nfft)

S = spectrogram(x,window,noverlap,nfft,fs)

调用及参数描述：

**window** is a Hamming window of length nfft.

**noverlap** is the number of samples that each segment overlaps. The default value is the number producing 50% overlap between segments.

**nfft** is the FFT length and is the maximum of 256 or the next power of 2 greater than the length of each segment of x. Instead of nfft, you can specify a vector of frequencies, F. See below for more information.

**fs** is the sampling frequency, which defaults to normalized frequency

* 1. 短时傅里叶变换
     1. 正弦信号
        1. 生成信号长度1秒、采样频率1kHz、周期分别为0.1秒、1秒和10秒的正弦信号s，并画出这些正弦信号
        2. 用spectrogram画出这些正弦信号的短时傅里叶变换：

spectrogram(s,hamming(256),255,256,1000);

* + 1. 窗口的影响
       1. 针对周期为0.1秒的正弦函数，分别调整hamming窗口大小为32、64、128、256，并画出该正弦信号的短时傅里叶变换
       2. 针对周期为0.1秒的正弦函数，窗口大小为128，分别调整窗口类型为hamming、rectwin和blackman，并画出该正弦信号的短时傅里叶变换
    2. 不同信号的短时傅里叶变换
       1. 例四中的离散信号，其前500点是慢变化正弦序列，后500点是快变化正弦序列，在500点处有断点，画出其短时傅立叶变换。



* + - 1. 例五使用STFT分析一个非平稳信号chirp信号



其中n为0~20000的序列。

* + - 1. 用如下命令读取声音文件sealion.wav，并分别用不同的窗口进行STFT，并分析哪种窗口效果更好。

[y,fs]=wavread(‘sealion’);

t=(0:length(y)-1)/fs;

plot(t,y), xlabel(‘time(sec)’)

* 1. 小波变换
     1. 利用matlab函数，生成不同类型的小波
        1. mexihat
        2. meyer
        3. Haar
        4. db
        5. sym
        6. morlet
     2. 一维连续小波变换

利用连续小波变换函数cwt对带白噪声的正弦信号及正弦加三角波进行变换。在对这两组信号利用wavedec函数及db5进行5层和6曾的分解，并利用wrcoef函数对低频和高频分别进行重构。

信号导入load noissin，load trsin。

spectrogram

Spectrogram using short-time Fourier transform

**Syntax**

S = spectrogram(x)  
S = spectrogram(x,window)  
S = spectrogram(x,window,noverlap)  
S = spectrogram(x,window,noverlap,nfft)  
S = spectrogram(x,window,noverlap,nfft,fs)  
[S,F,T] = spectrogram(...)  
[S,F,T] = spectrogram(x,window,noverlap,F)  
[S,F,T] = spectrogram(x,window,noverlap,F,fs)  
[S,F,T,P] = spectrogram(...)  
spectrogram(...,FREQLOCATION)  
spectrogram(...)

**Description**

spectrogram, when used without any outputs, plots a spectrogram or, when used with an S output, returns the short-time Fourier transform of the input signal. To create a spectrogram from the returned short-time Fourier transform data, refer to the [[S,F,T,P] syntax](file:///C:\Program%20Files\MATLAB\R2014a\help\signal\ref\spectrogram.html#f8-808211) described below.

S = spectrogram(x) returns S, the short time Fourier transform of the input signal vector x. By default, x is divided into eight segments. If x cannot be divided exactly into eight segments, it is truncated. These default values are used.

* window is a Hamming window of length nfft.
* noverlap is the number of samples that each segment overlaps. The default value is the number producing 50% overlap between segments.
* nfft is the FFT length and is the maximum of 256 or the next power of 2 greater than the length of each segment of x. Instead of nfft, you can specify a vector of frequencies, F. See below for more information.
* fs is the sampling frequency, which defaults to normalized frequency.

Each column of S contains an estimate of the short-term, time-localized frequency content of x. Time increases across the columns of S and frequency increases down the rows.

If x is a length Nx complex signal, S is a complex matrix with nfft rows and k columns, where for a scalar window

k = fix((Nx-noverlap)/(window-noverlap))

or if window is a vector

k = fix((Nx-noverlap)/(length(window)-noverlap))

For real x, the output S has (nfft/2+1) rows if nfft is even, and (nfft+1)/2 rows if nfft is odd.

S = spectrogram(x,window) uses the window specified. If window is an integer, x is divided into segments equal to that integer value and a Hamming window is used. If window is a vector, x is divided into segments equal to the length of window and then the segments are windowed using the window functions specified in the window vector. For a list of available windows see [Windows](file:///C:\Program%20Files\MATLAB\R2014a\help\signal\ug\windows.html).

|  |
| --- |
| **Note:**   To obtain the same results for the removed specgram function, specify a 'Hann' window of length 256. |

S = spectrogram(x,window,noverlap) overlaps noverlap samples of each segment. noverlap must be an integer smaller than window or if window is a vector, smaller than the length of window.

S = spectrogram(x,window,noverlap,nfft) uses the nfft number of sampling points to calculate the discrete Fourier transform. nfft must be a scalar.

S = spectrogram(x,window,noverlap,nfft,fs) uses fs sampling frequency in Hz. If fs is specified as empty [], it defaults to 1 Hz.

[S,F,T] = spectrogram(...) returns a vector of frequencies, F, and a vector of times, T, at which the spectrogram is computed. F has length equal to the number of rows of S. T has length k (defined above) and the values in T correspond to the center of each segment.

[S,F,T] = spectrogram(x,window,noverlap,F) uses a vector F of frequencies in Hz. F must be a vector with at least two elements. This case computes the spectrogram at the frequencies in F using the Goertzel algorithm. The specified frequencies are rounded to the nearest DFT bin commensurate with the signal's resolution. In all other syntax cases where nfft or a default for nfft is used, the short-time Fourier transform is used. The F vector returned is a vector of the rounded frequencies. T is a vector of times at which the spectrogram is computed. The length of F is equal to the number of rows of S. The length of T is equal to k, as defined above and each value corresponds to the center of each segment.

[S,F,T] = spectrogram(x,window,noverlap,F,fs) uses a vector F of frequencies in Hz as above and uses the fs sampling frequency in Hz. If fs is specified as empty [], it defaults to 1 Hz.

[S,F,T,P] = spectrogram(...) returns a matrix P containing the power spectral density (PSD) of each segment. For real x, P contains the one-sided modified periodogram estimate of the PSD of each segment. For complex x and when you specify a vector of frequencies F, P contains the two-sided PSD.

spectrogram(...,FREQLOCATION) specifies which axis to use as the frequency axis in displaying the spectrogram. Specify FREQLOCATION as a trailing string argument. Valid options are 'xaxis' or 'yaxis'. The strings are not case sensitive. If you do not specify FREQLOCATION, spectrogram uses the x-axis as the frequency axis by default.

The elements of the PSD matrix P are given by C:\Program Files\MATLAB\R2014a\help\signal\ref\eqn1233059860.pngwhere C:\Program Files\MATLAB\R2014a\help\signal\ref\eqn1233059916.pngis a real-valued scalar defined as follows.

* For the one-sided PSD,

C:\Program Files\MATLAB\R2014a\help\signal\ref\eqn1233059974.png

where C:\Program Files\MATLAB\R2014a\help\signal\ref\eqn1233060112.pngdenotes the window function (Hamming by default) and C:\Program Files\MATLAB\R2014a\help\signal\ref\eqn1233060398.pngis the sampling frequency. At zero and the Nyquist frequencies, the factor of 2 in the numerator is replaced by 1.

* For the two-sided PSD,

C:\Program Files\MATLAB\R2014a\help\signal\ref\eqn1233060270.png

at all frequencies.

* If the sampling frequency is not specified, C:\Program Files\MATLAB\R2014a\help\signal\ref\eqn1233060472.pngis replaced in the denominator by C:\Program Files\MATLAB\R2014a\help\signal\ref\eqn1233060494.png.

spectrogram(...) plots the PSD estimate for each segment on a surface in a figure window. The plot is created using

surf(T,F,10\*log10(abs(P)));

axis tight;

view(0,90);

Using spectrogram(...,*'freqloc'*) syntax and adding a *'freqloc'* string (either 'xaxis' or 'yaxis') controls where the frequency axis is displayed. Using 'xaxis' displays the frequency on the *x*-axis. Using 'yaxis' displays frequency on the *y*-axis and time on the *x*-axis. The default is 'xaxis'. If you specify both a *'freqloc'* string and output arguments, *'freqloc'* is ignored.

**References**

[1] Oppenheim, Alan V., Ronald W. Schafer, and John R. Buck. *Discrete-Time Signal Processing*. 2nd Ed. Upper Saddle River, NJ: Prentice Hall, 1999.

[2] Rabiner, Lawrence R., and Ronald W. Schafer. *Digital Processing of Speech Signals*. Englewood Cliffs, NJ: Prentice-Hall, 1978.

# cwt

Continuous 1-D wavelet transform

## Syntax

coefs = cwt(x,scales,'*wname*')  
coefs = cwt(x,scales,'*wname*','plot')  
coefs = cwt(x,scales,'*wname*','coloration')  
[coefs,sgram] = cwt(x,scales,'*wname*','scal')  
[coefs,sgram] = cwt(x,scales,'*wname*','scalCNT')  
coefs = cwt(x,scales,'*wname*','coloration',xlim)

## Description

coefs = cwt(x,scales,'*wname*') computes the continuous wavelet transform (CWT) coefficients of the real-valued signal x at real, positive scales, using wavelet *'wname'* (see [waveinfo](file:///C:\Program%20Files\MATLAB\R2014a\help\wavelet\ref\waveinfo.html) for more information). The analyzing wavelet can be real or complex. coefs is an *la*-by-*lx* matrix, where *la* is the length of scales and *lx* is the length of the input x. coefs is a real or complex matrix, depending on the wavelet type.

coefs = cwt(x,scales,'*wname*','plot') plots the continuous wavelet transform coefficients, using default coloration 'absglb'.

coefs = cwt(x,scales,'*wname*','coloration') uses the specified coloration.

[coefs,sgram] = cwt(x,scales,'*wname*','scal') displays a scaled image of the scalogram.

[coefs,sgram] = cwt(x,scales,'*wname*','scalCNT') displays a contour representation of the scalogram.

coefs = cwt(x,scales,'*wname*','coloration',xlim) colors the coefficients using coloration and xlim, where xlim is a vector, [x1 x2], with 1 ≤ x1 < x2 ≤ length(x).

## Examples

Plot the continuous wavelet transform and scalogram using sym2 wavelet at all integer scales from 1 to 32, using a fractal signal as input:

load vonkoch

vonkoch=vonkoch(1:510);

len = length(vonkoch);

cw1 = cwt(vonkoch,1:32,'sym2','plot');

title('Continuous Transform, absolute coefficients.')

ylabel('Scale')

[cw1,sc] = cwt(vonkoch,1:32,'sym2','scal');

title('Scalogram')

ylabel('Scale')

Compare discrete and continuous wavelet transforms, using a fractal signal as input:

load vonkoch

vonkoch=vonkoch(1:510);

len=length(vonkoch);

[c,l]=wavedec(vonkoch,5,'sym2');

% Compute and reshape DWT to compare with CWT.

cfd=zeros(5,len);

for k=1:5

d=detcoef(c,l,k);

d=d(ones(1,2^k),:);

cfd(k,:)=wkeep(d(:)',len);

end

cfd=cfd(:);

I=find(abs(cfd) <sqrt(eps));

cfd(I)=zeros(size(I));

cfd=reshape(cfd,5,len);

% Plot DWT.

subplot(311); plot(vonkoch); title('Analyzed signal.');

set(gca,'xlim',[0 510]);

subplot(312);

image(flipud(wcodemat(cfd,255,'row')));

colormap(pink(255));

set(gca,'yticklabel',[]);

title('Discrete Transform,absolute coefficients');

ylabel('Level');

% Compute CWT and compare with DWT

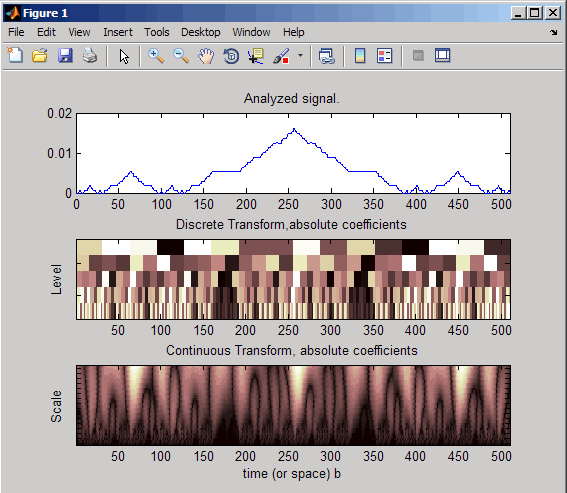
subplot(313);

ccfs=cwt(vonkoch,1:32,'sym2','plot');

title('Continuous Transform, absolute coefficients');

set(gca,'yticklabel',[]);

ylabel('Scale');



Scalograms are plots that represent the percentage energy for each coefficient.

## References

Daubechies, I. Ten Lectures on Wavelets, Philadelphia, PA: Society for Industrial and Applied Mathematics (SIAM), 1992.

Mallat, S. A Wavelet Tour of Signal Processing, San Diego, CA: Academic Press, 1998.