

Time Frequency Analysis and Wavelet Transforms

時頻分析與小波轉換

授課者： 丁 建 均

Office：明達館723室，TEL：33669652

E-mail: jjding@ntu.edu.tw

課程網頁：<http://djj.ee.ntu.edu.tw/TFW.htm>

歡迎大家來修課，也歡迎有問題時隨時聯絡！

上課方式

(1) 錄影，影片將藉由 NTU Cool 下載 <http://cool.ntu.edu.tw>

(2) 現場 (明達館231室)

作業和報告繳交方式

用 NTU Cool 來繳交作業與報告的電子檔 <http://cool.ntu.edu.tw>

注意，Tutorial 一定要交 Word 或 Latex 原始碼

Wiki 要寄編輯條目的連結給老師

- 評分方式：

平時分數: 15 scores

基本分 11 分，各位同學皆可拿到

另外，每週都將會在課堂上問一個問題，請特定學號尾數的同學，在作業上回答 (每位同學每學期會輪到四次左右)，每正確回答一次加 0.8 分，參與口頭報告評選再加 0.8 分

Homework: 60 scores

5 times, 每 3 週一次

請自己寫

和同學內容極相似，或重度依賴 AI 且未自己消化，將扣 70% 的分數
就算寫錯但好好寫也會給 40~95% 的分數

遲交分數打 8 折，不交不給分

不知道如何寫，可與老師和助教聯繫

Term paper 25 scores

Term paper 25 scores

方式有五種，可任選其中一種

(1) 書面報告

(10頁以上(不含封面)，中英文皆可，11或12的字體，題目可選擇和課程有關的任何一個主題。

格式不限，但儘量和一般寫期刊論文或碩博士論文相同，包括 abstract, conclusion, 及 references，至少有8個 equations，並且要分 sections，必要時有 subsections。鼓勵多做實驗及模擬。

嚴禁剪刀漿糊 (Ctrl-C, Ctrl-V) 的情形，否則扣 60% 的分數

(2) Tutorial

限十五個名額，和書面報告格式相同，但 17頁以上(若為加強前人的 tutorial，則頁數為 $(2/3)N + 12$ 以上， N 為前人 tutorial 之頁數)，題目由老師指定 (see pages 12 and 13)，以清楚且有系統的介紹一個主題的基本概念和應用為要求，為上課內容的進一步探討和補充，[交Word檔](#)。

選擇這個項目的同學，學期成績加 4分

(3) 口頭報告

限十個名額，每個人 30分鐘，題目可選擇和課程有關的任何一個主題。口頭報告的同學，請在 12月18日之前將報告的影片寄給老師。有意願的同學，請儘早告知，以先登記的同學為優先。

口頭報告鼓勵同學們參與聽講和票選，有參與票選的同學期末報告成績加2分。

選擇這個項目的同學，學期成績加 2分

(4) 編輯 Wikipedia

中文或英文網頁皆可，至少 2 個條目，但不可同一個條目翻成中文和英文。總計80行以上。限和課程相關者，自由發揮，越有條理、有系統的越好

選擇編輯 Wikipedia 的同學，請於12月11日(本學期最後一次上課)前，向我登記並告知我要編輯的條目(2 個以上)，若有和其他同學選擇相同條目的情形，則較晚向我登記的同學將更換要編輯的條目
編輯完成之後，要將連結寄給老師

(1)~(4) 各項期末報告若有做實驗模擬，請附上程式碼，會有額外的加分
(鼓勵不強制)

書面報告、Tutorial、編輯 Wikipedia、和程式編寫的期限是 12月25日

上課時間：14 週

9/4,

9/11, 出 HW1

9/18,

9/25, 交 HW1

10/2, 出 HW2

10/9,

10/16, 交 HW2

10/30, 出 HW3

11/6,

11/13, 交 HW3

11/20, 出 HW4

11/27,

12/4, 交 HW4,

12/11, 出 HW5

12/18, 口頭報告的同學交影片

12/25, 交 HW5 及 term paper

課程大綱：

- (1) Introduction
 - (2) Short-Time Fourier Transform
 - (3) Gabor Transform
 - (4) Implementation of Time-Frequency Analysis
 - (5) Wigner Distribution Function
 - (6) Cohen's Class Time-Frequency Distribution
 - (7) S Transforms, Gabor-Wigner Transforms, Matching Pursuit, and Other Time Frequency Analysis Methods
 - (8) Movement in the Time-Frequency Plane and Fractional Fourier Transforms
 - (9) Applications of Time-Frequency Analysis
- (續)

課程大綱：

- (10) Hilbert Huang Transform
- (11) From Haar Transforms to Wavelet Transforms
- (12) Continuous Wavelet Transforms with Discrete Coefficients
- (13) Discrete Wavelet Transform
- (14) Applications of the Wavelet Transform

- 上課資料：

(1) 講義 (將放在網頁上，請大家每次上課前先印好)

(2) S. Mallat, *A Wavelet Tour of Signal Processing: The Sparse Way*, Academic Press, 3rd ed., 2009.

(3) S. Qian and D. Chen, *Joint Time-Frequency Analysis: Methods and Applications*, Prentice-Hall, 1996.

(4) P. Flandrin, *Time-frequency / Time Scale Analysis*, translated by J. Stöckler, Academic Press, San Diego, 1999.

(5) K. Grochenig, *Foundations of Time-Frequency Analysis*, Birkhauser, Boston, 2001.

(6) L. Debnath, *Wavelet Transforms and Time-Frequency Signal Analysis*, Birkhäuser, Boston, 2001.

(7) F. Hlawatsch and F. Auger, *Time-frequency Analysis Concepts and Methods*, Wiley, London, 2008.

Matlab Program

Download: 請洽台大各系所

參考書目

洪維恩，Matlab 程式設計，旗標，台北市，2013. (合適的入門書)

張智星，Matlab 程式設計入門篇，第四版，碁峰，2016.

預計看書學習所花時間：3~5 天

Python Program

Download: <https://www.python.org/>

參考書目

葉難，Python 程式設計入門，博碩，2015

黃健庭，Python 程式設計：從入門到進階應用，全華，2020

The Python Tutorial <https://docs.python.org/3/tutorial/index.html>

Tutorial 可供選擇的題目(可以略做修改)

- (1) Wave Shape Function Analysis
- (2) Prediction Techniques Using Time-Frequency Analysis
- (3) Quasi-Periodic, Pseudo Periodic, and Almost Periodic Functions
- (4) Multitaper Analysis
- (5) Ridge Analysis for Fundamental Frequency Estimation
- (6) Ensemble Empirical Mode Decomposition (EEMD)
- (7) Learnable Time–Frequency Representation
- (8) Data-Driven Time–Frequency Analysis
- (9) Sparse Time-Frequency Representations
- (10) Tremor Analysis in the Time-Frequency Plane

Tutorial 可供選擇的題目(可以略做修改)

- (11) Electroencephalography (EEG) Analysis Using Wavelet Transforms
- (12) Wavelet Transform for Computed Tomography (CT)
- (13) Wavelet Transform for Geology
- (14) Wavelet Transforms for Encryption
- (15) Wavelet Transforms for Face Detection

I. Introduction

Fourier transform (FT)

$$X(f) = \int_{-\infty}^{\infty} x(t) e^{-j2\pi f t} dt \quad \text{Time-Domain} \rightarrow \text{Frequency Domain}$$

↑ t varies from $-\infty \sim \infty$

Laplace Transform $X(s) = \int_{-\infty}^{\infty} x(t) e^{-st} dt$

Cosine Transform, Sine Transform, Z Transform.

Some things make the FT not practical:

(1) It less happens that a signal has the interval of $(-\infty \sim \infty)$

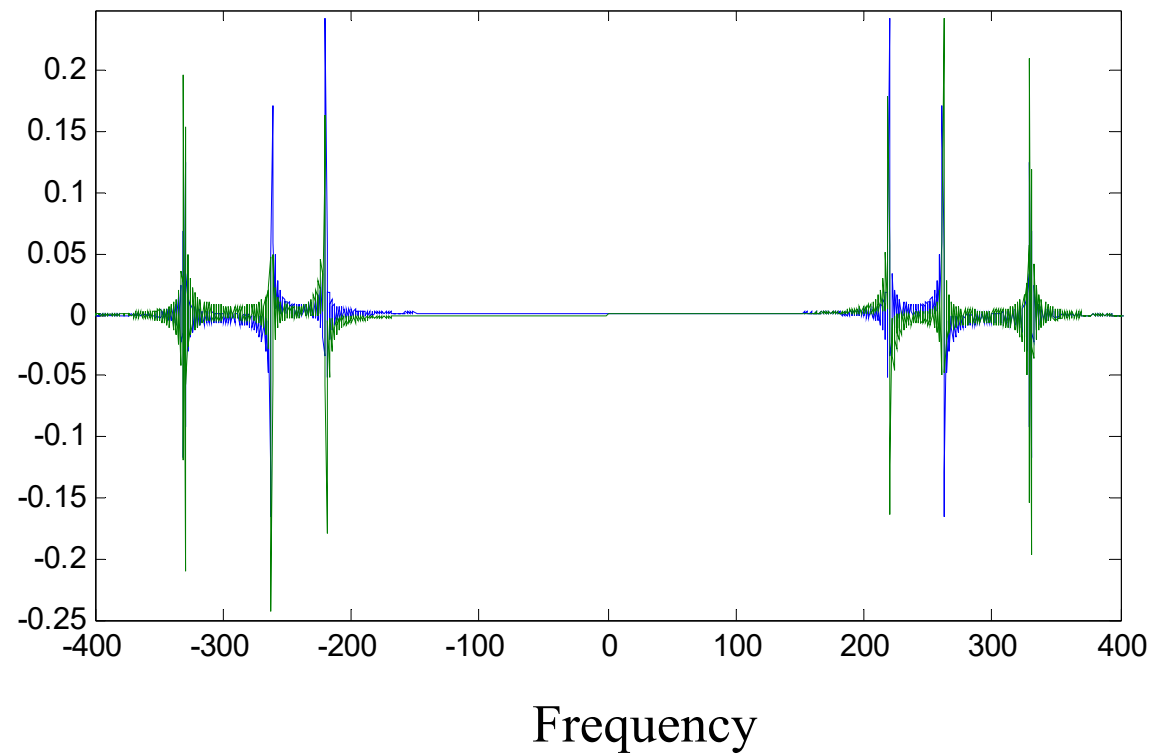
Even for a signal with infinite length, we are only interested in the characteristics in a finite interval.

(2) It is hard to observe the variation of spectrum with time by the FT.

Example 1: $x(t) = \cos(440\pi t)$ when $t < 0.5$,
 $x(t) = \cos(660\pi t)$ when $0.5 \leq t < 1$,
 $x(t) = \cos(524\pi t)$ when $t \geq 1$



The Fourier transform of $x(t)$



(A) Finite-Supporting Fourier Transform

$$X(f) = \int_{t_0-B}^{t_0+B} x(t) e^{-j2\pi f t} dt$$

$$\Rightarrow X(f) = \int_{t-B}^{t+B} x(\tau) e^{-j2\pi f \tau} d\tau = \int_{-\infty}^{\infty} \Pi\left(\frac{\tau-t}{2B}\right) x(\tau) e^{-j2\pi f \tau} d\tau$$

(B) Short-Time Fourier Transform (STFT)

$$X(t, f) = \int_{-\infty}^{\infty} w(t-\tau) x(\tau) e^{-j2\pi f \tau} d\tau$$

$w(t)$: window function 或 mask function

STFT 也稱作 windowed Fourier transform 或

time-dependent Fourier transform

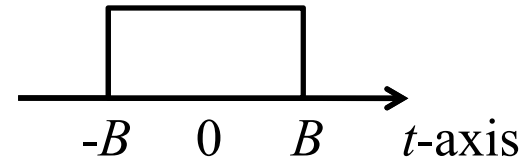
[Ref] L. Cohen, *Time-Frequency Analysis*, Prentice-Hall, New York, 1995.

[Ref] A. V. Oppenheim and R. W. Schaffer, *Discrete-Time Signal Processing*,
London: Prentice-Hall, 3rd ed., 2010.

$$X(t, f) = \int_{-\infty}^{\infty} w(t - \tau) x(\tau) e^{-j2\pi f \tau} d\tau$$

最簡單的例子： $w(t) = 1$ for $|t| \leq B$,

$w(t) = 0$ otherwise

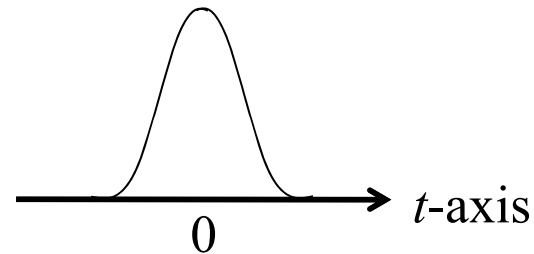


此時 Short-time Fourier transform 可以改寫

$$X(t, f) = \int_{t-B}^{t+B} x(\tau) e^{-j2\pi f \tau} d\tau$$

其他的例子：

$$w(t) = \exp(-\pi\sigma t^2)$$



一般我們把 $\exp(-\pi\sigma t^2)$ 稱作為 Gaussian function 或 Gabor function

此時的 Short-Time Fourier Transform 亦稱作 Gabor Transform

(C) Gabor Transform

$$G_x(t, f) = \int_{-\infty}^{\infty} e^{-\pi\sigma(\tau-t)^2} e^{-j2\pi f\tau} x(\tau) d\tau$$

$$G_x(t, \omega) = \sqrt{\frac{1}{2\pi}} \int_{-\infty}^{\infty} e^{-\frac{\sigma(\tau-t)^2}{2}} e^{-j\omega\tau} x(\tau) d\tau$$

- S. Qian and D. Chen, *Joint Time-Frequency Analysis: Methods and Applications*, Prentice Hall, N.J., 1996.

Without cross term, poor clarity

(D) Wigner Distribution Function

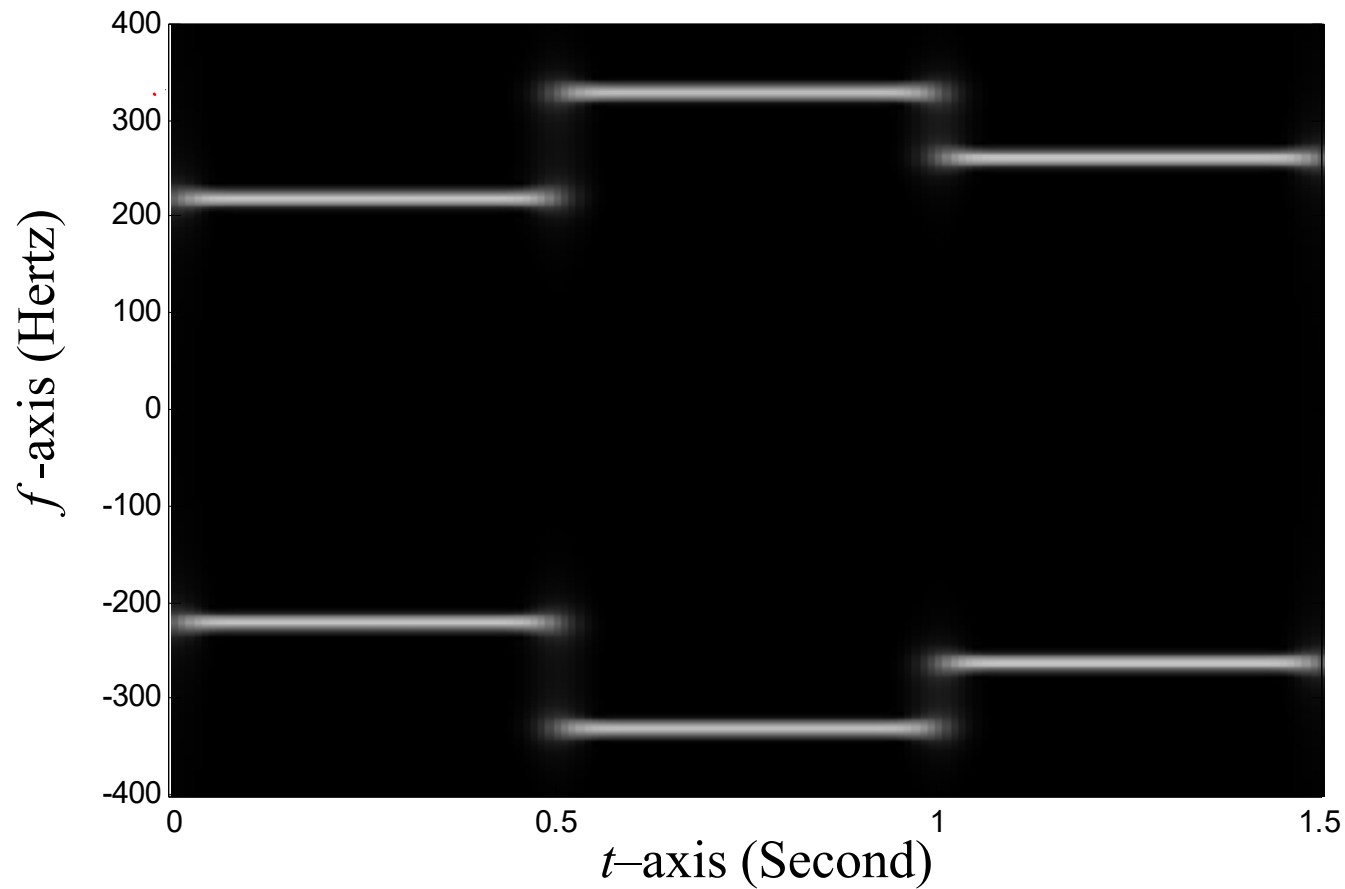
$$G_x(t, f) = \int_{-\infty}^{\infty} e^{-j2\pi f\tau} x(t + \tau / 2) x^*(t - \tau / 2) d\tau$$

$$G_x(t, \omega) = \int_{-\infty}^{\infty} e^{-j\omega\tau} x(t + \tau / 2) x^*(t - \tau / 2) d\tau$$

With cross term, high clarity

Example: $x(t) = \cos(440\pi t)$ when $t < 0.5$,
 $x(t) = \cos(660\pi t)$ when $0.5 \leq t < 1$,
 $x(t) = \cos(524\pi t)$ when $t \geq 1$

The Gabor transform of $x(t)$ ($\sigma = 200$)



用 Gray level 來表示 $X(t, f)$ 的 amplitude

自然界中，頻率會隨著時間而改變的例子

Frequency Modulation

Music

Speech

Others (Animal voice, Doppler effect, seismic waves, radar system, optics, rectangular function)

In fact, in addition to **sinusoid-like functions**, the instantaneous frequencies of other functions will inevitably vary with time.

Instantaneous Frequency 瞬時頻率

If $x(t) = \sum_{k=1}^N a_k \cdot \exp(j \cdot \phi_k(t))$ around t_0

then the instantaneous frequency of $x(t)$ at t_0 are

$$\frac{\phi_1'(t_0)}{2\pi}, \frac{\phi_2'(t_0)}{2\pi}, \frac{\phi_3'(t_0)}{2\pi}, \dots, \frac{\phi_N'(t_0)}{2\pi} \quad (\text{以頻率 frequency 表示})$$

$$\phi_1'(t_0), \phi_2'(t_0), \phi_3'(t_0), \dots, \phi_N'(t_0) \quad (\text{以角頻率 angular frequency 表示}) :$$

If the order of $\phi_k(t) > 1$, then instantaneous frequency varies with time

- Sinusoid Function

- Chirp function

$$\exp\left[j(\alpha_2 t^2 + \alpha_1 t + \alpha_0)\right] \quad \text{Instantaneous frequency} = \frac{\alpha_2}{\pi} t + \frac{\alpha_1}{2\pi}$$

acoustics, wireless communication, radar system, optics

Example : the 2nd, 3rd, 4th, and 5th tones

- Higher order exponential function

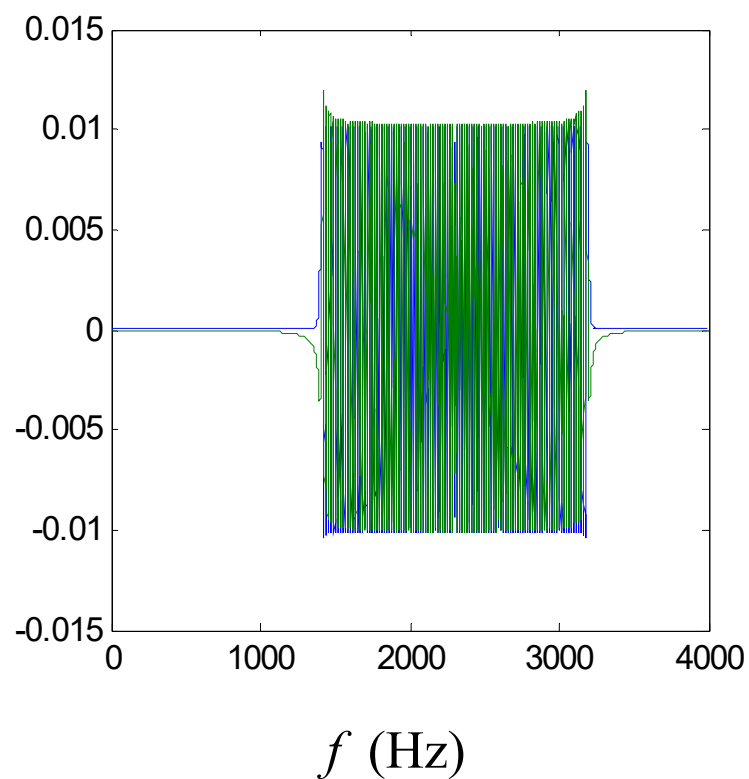
Example 2

$$(1) \quad x(t) = 0.5 \cos(6400\pi t - 600\pi t^2) \quad t \in [0, 3]$$

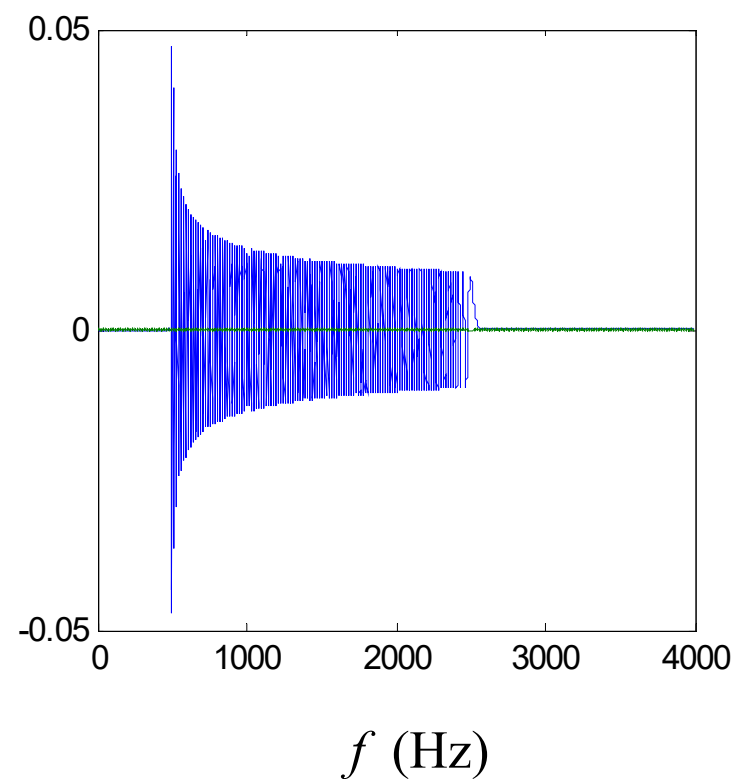
$$(2) \quad x(t) = 0.5 \cos(600\pi t^3 - 2700\pi t^2 + 5050\pi t) \quad t \in [0, 3]$$

Fourier transform

$$x(t) = 0.5 \cos(6400\pi t - 600\pi t^2)$$

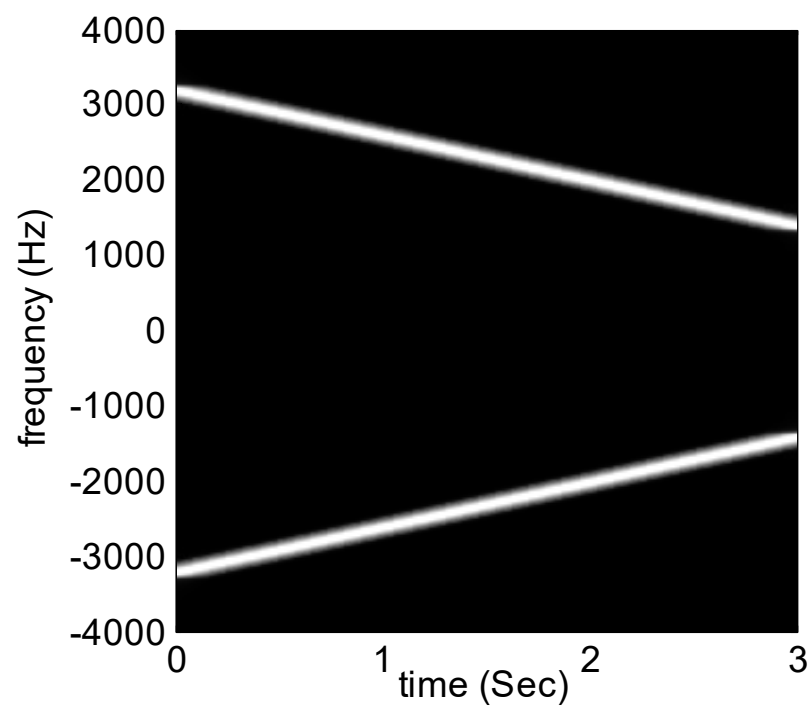


$$x(t) = 0.5 \cos(600\pi t^3 - 2700\pi t^2 + 5050\pi t)$$



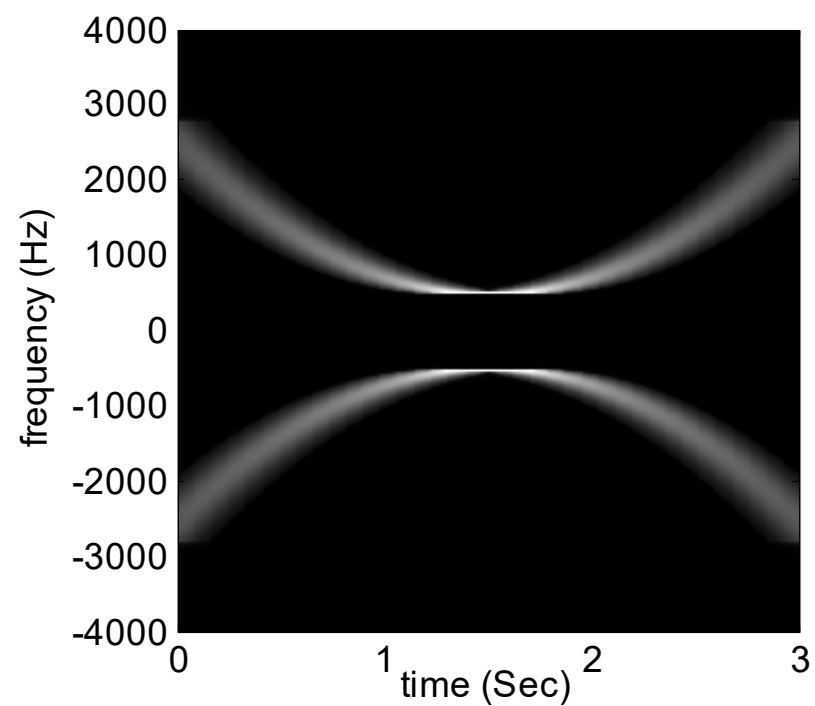
(1)

$$x(t) = 0.5 \cos(6400\pi t - 600\pi t^2)$$



(2)

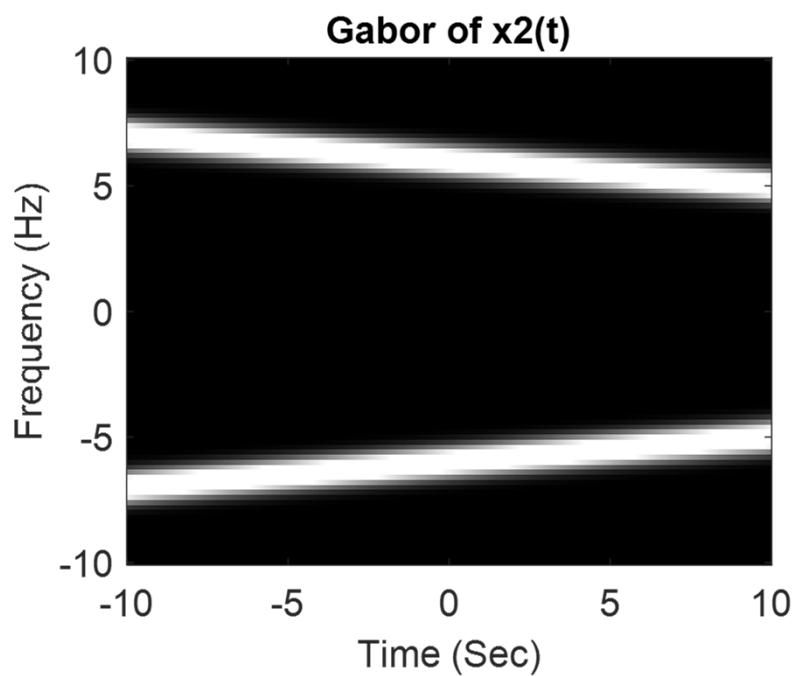
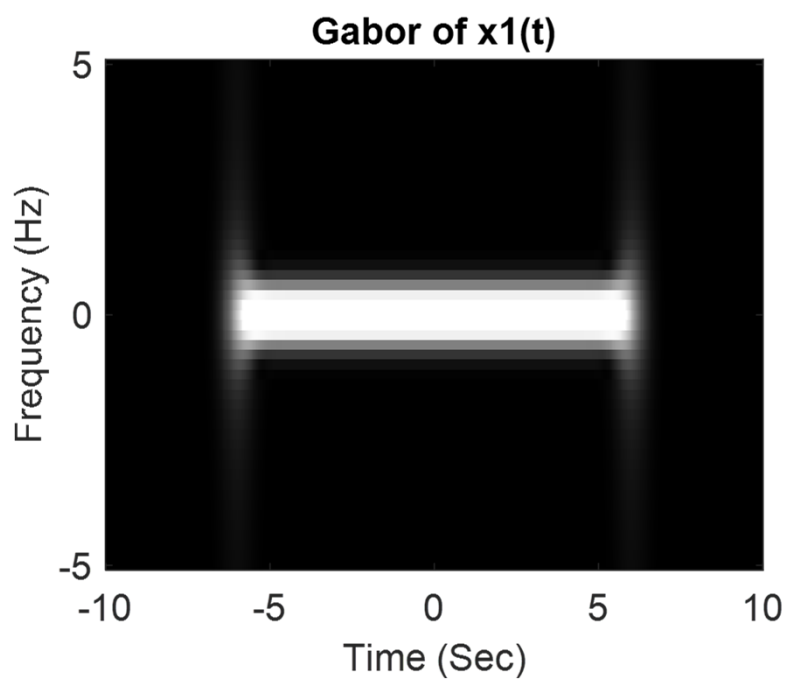
$$x(t) = 0.5 \cos(600\pi t^3 - 2700\pi t^2 + 5050\pi t)$$



Example 3

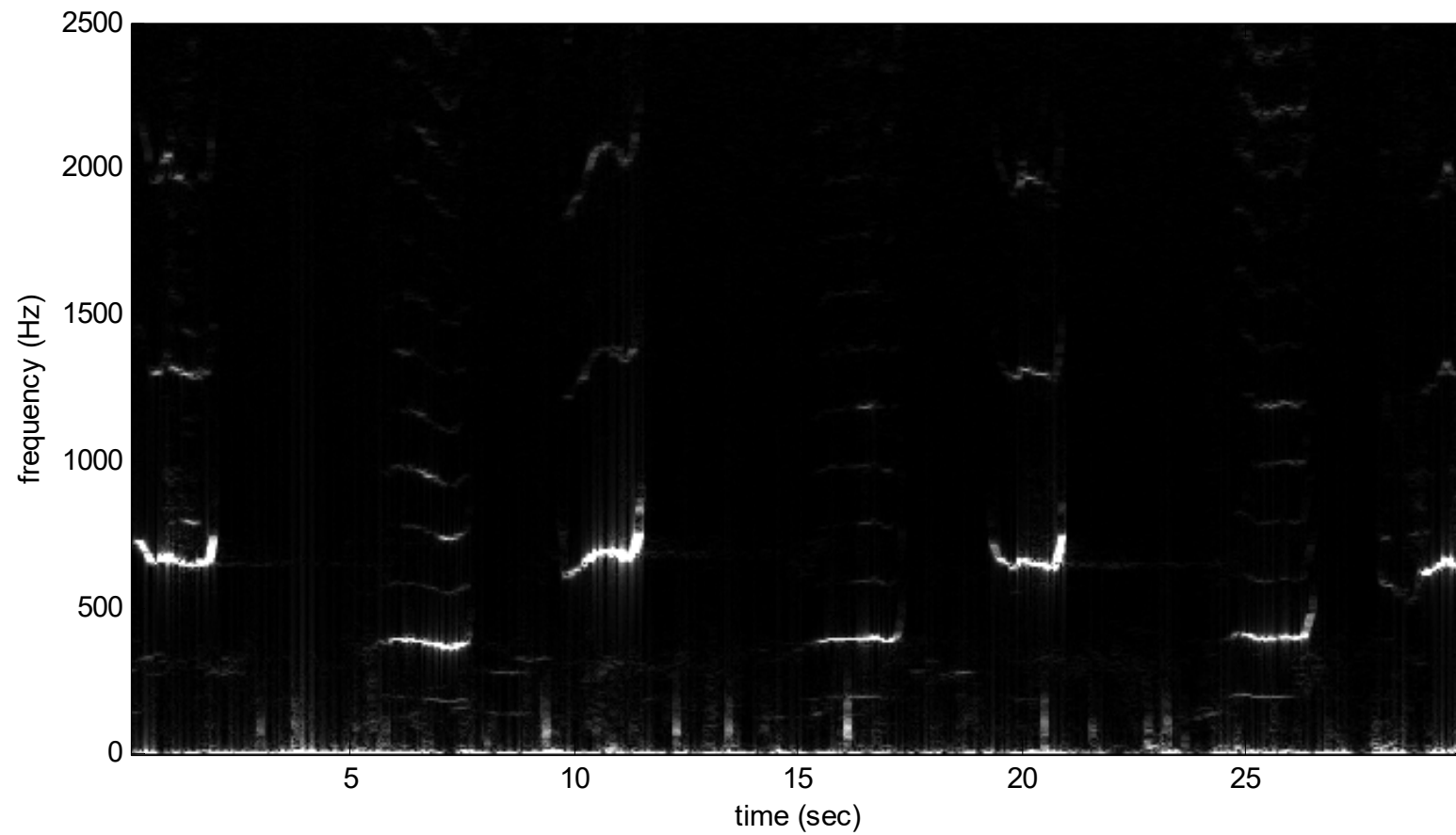
left: $x_1(t) = 1$ for $|t| \leq 6$, $x_1(t) = 0$ otherwise, right: $x_2(t) = \cos(12\pi t - 0.1\pi t^2)$

Gabor transform



Example 4

Data source: <http://oalib.hlsresearch.com/Whales/index.html>



Why Time-Frequency Analysis is Important?

- Many digital signal processing applications are related to the **spectrum** or the **bandwidth** of a signal.
 - If the spectrum and the bandwidth can be determined adaptive, the performance can be improved.
-
- | | |
|---------------------|--------------------------|
| • modulation, | • signal identification, |
| • multiplexing, | • acoustics, |
| • filter design, | • system modeling, |
| • data compression, | • radar system analysis |
| • signal analysis, | • sampling |

Example: Generalization for sampling theory

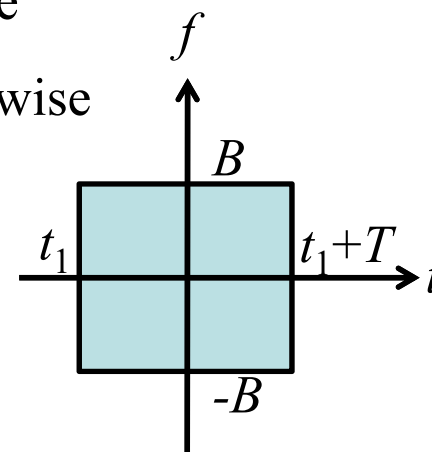
假設有一個信號，

- ① The supporting of $x(t)$ is $t_1 \leq t \leq t_1 + T$, $x(t) \approx 0$ otherwise
- ② The supporting of $X(f) \neq 0$ is $-B \leq f \leq B$, $X(f) \approx 0$ otherwise

根據取樣定理， $\Delta_t \leq 1/F$, $F=2B$, B :頻寬

所以，取樣點數 N 的範圍是

$$N = T/\Delta_t \geq TF$$



重要定理： 一個信號所需要的取樣點數的下限，等於它時頻分佈的面積

Q1 : Scaling 對於一個信號的取樣點數有沒有影響？

Hint:

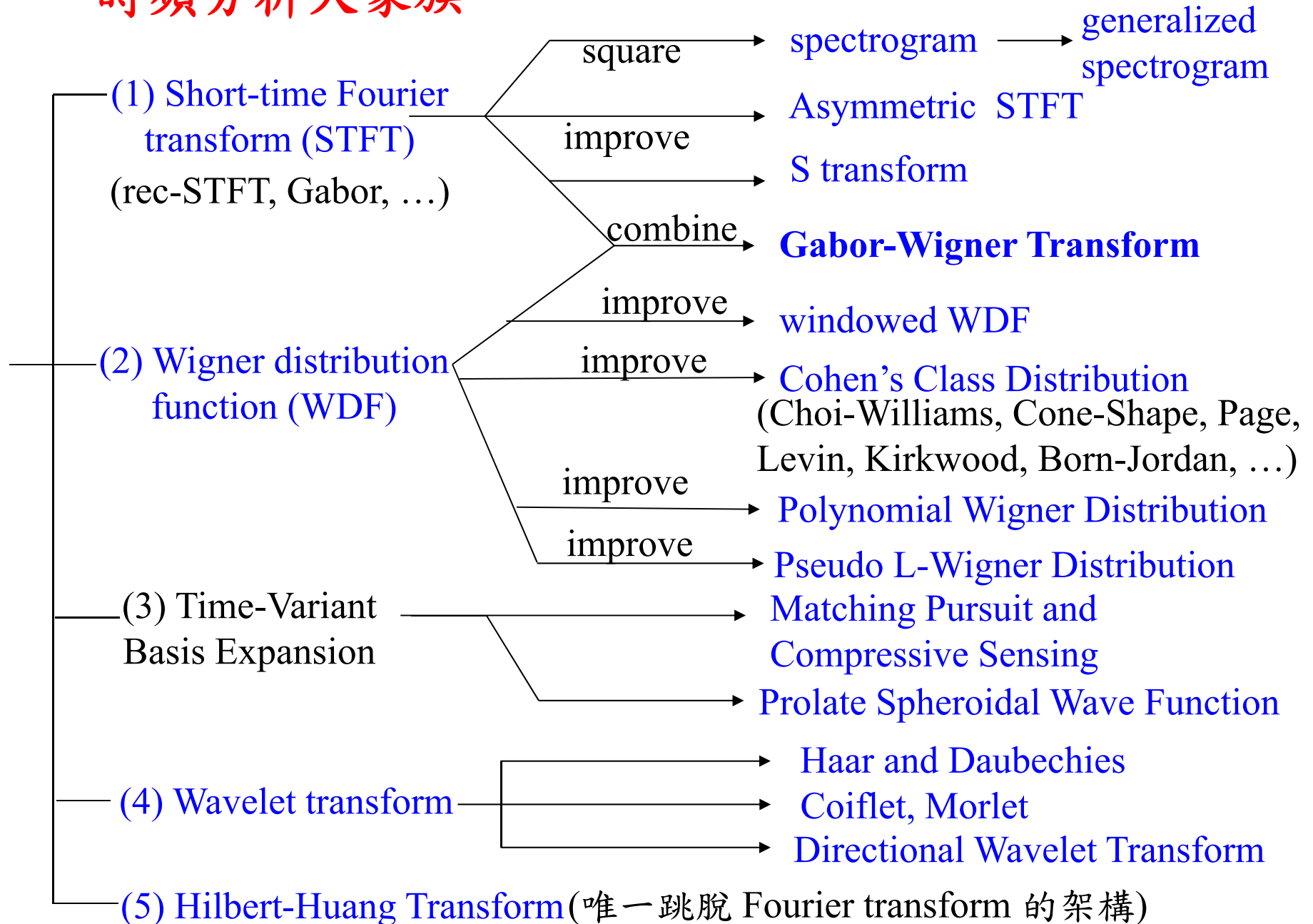
$$g(\sigma t) \xrightarrow{FT} \frac{1}{|\sigma|} G\left(\frac{f}{\sigma}\right)$$

Q2: How to use time-frequency analysis to reduce the number of sampling points?

Time-frequency analysis is an efficient tool for **adaptive signal processing**.

時頻分析大家族

31



- **Wigner distribution function**

$$W_x(t, f) = \int_{-\infty}^{\infty} x(t + \tau/2) \cdot x^*(t - \tau/2) e^{-j2\pi\tau f} d\tau$$

$x(t + \tau/2) \cdot x^*(t - \tau/2)$: signal auto-correlation function:

- **Time-variant basis expansion**

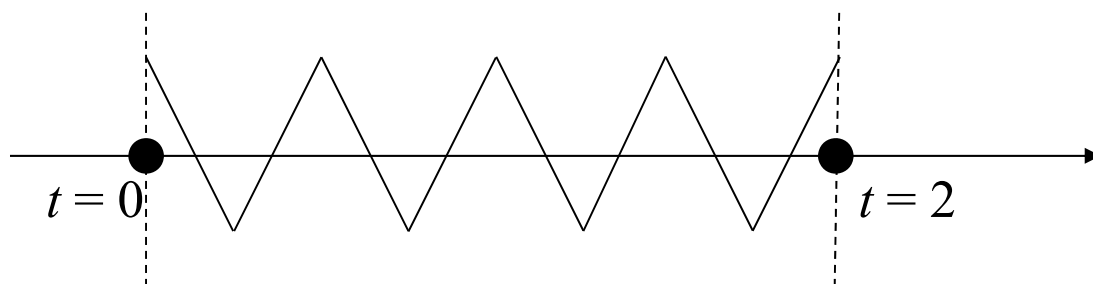
3-parameter atoms

$$x(t) \approx \sum a_{t_0, f_0, \sigma} \varphi_{t_0, f_0, \sigma}(t)$$

$$\varphi_{t_0, f_0, \sigma}(t) = \frac{2^{1/4}}{\sigma^{1/2}} \exp(j2\pi f_0 t) \exp\left(-\frac{\pi(t - t_0)^2}{\sigma^2}\right)$$

- Hilbert Huang Transform

Another way to determine the frequency (without the FT)



$$\text{frequency} = \frac{\text{number of zero crossings}}{2 \times (\text{time duration})}$$

• Continuous Wavelet Transform

forward wavelet transform:

$$X(a, b) = \frac{1}{\sqrt{b}} \int_{-\infty}^{\infty} x(t) \psi\left(\frac{t-a}{b}\right) dt$$

$\psi(t)$: mother wavelet, a : location, b : scaling,

inverse wavelet transform:

$$x(t) = \sum_a \sum_b X(a, b) \varphi_{a,b}(t)$$

$\varphi_{a,b}(t)$ is dual orthogonal to $\psi(t)$.

	output
Fourier transform	$X(f)$, f : frequency
time-frequency analysis	$X(t, f)$, t : time, f : frequency
wavelet transform	$X(a, b)$, a : time, b : scaling

限制：

$$(1) \quad \frac{1}{\sqrt{b}} \int_{-\infty}^{\infty} \varphi_{a_1, b_1}(t) \psi\left(\frac{t-a}{b}\right) dt = 1 \quad \text{when } a_1 = a \text{ and } b_1 = b,$$

$$\frac{1}{\sqrt{b}} \int_{-\infty}^{\infty} \varphi_{a_1, b_1}(t) \psi\left(\frac{t-a}{b}\right) dt = 0 \quad \text{otherwise}$$

(2) $\psi(t)$ has a finite time interval

Two parameters, a : 調整位置, b : 調整寬度

應用：adaptive signal analysis

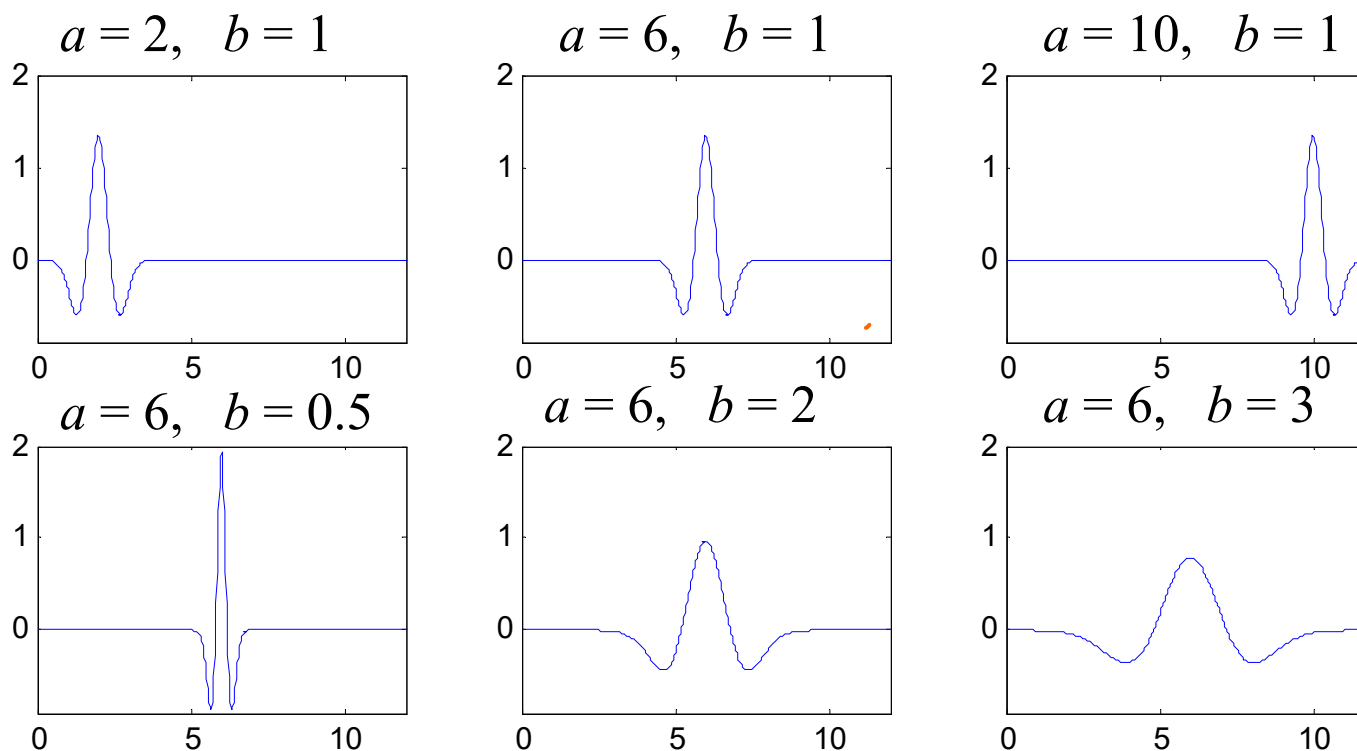
思考：需要較高解析度的地方， b 的值應該如何？

Wavelet 的種類甚多

Mexican hat wavelet, Haar Wavelet, Daubechies wavelet, triangular wavelet,

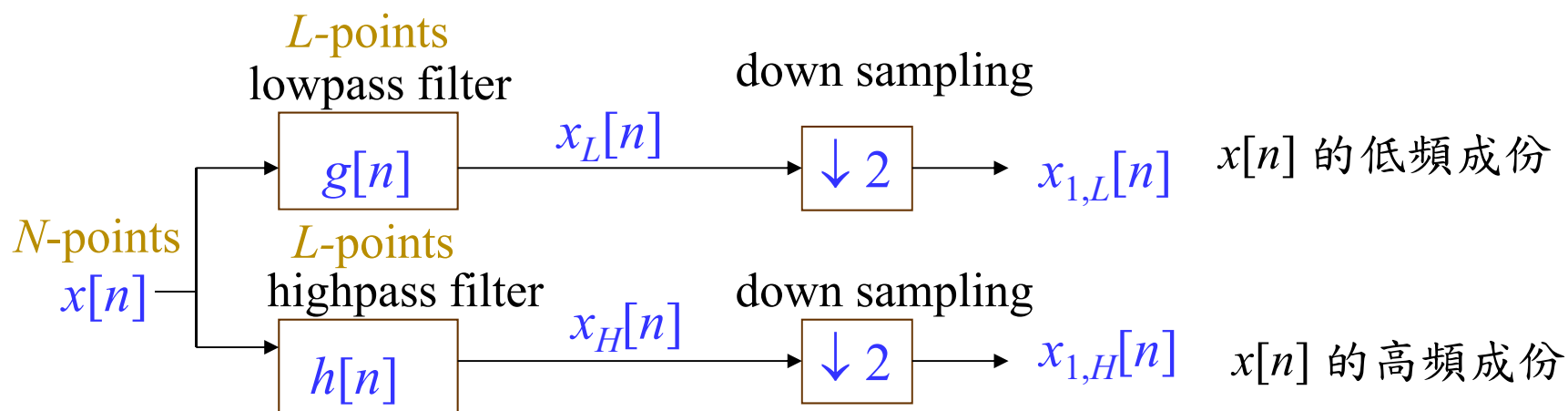
例子：Mexican hat wavelet (Laplacian of Gaussian) 隨 a and b 變化之情形

$$\psi(t) = \frac{2^{5/4}}{\sqrt{3}}(1 - 2\pi t^2)e^{-\pi t^2} \quad \frac{1}{\sqrt{b}}\psi\left(\frac{t-a}{b}\right)$$



• Discrete Wavelet Transform (DWT)

The discrete wavelet transform is **very different** from the continuous wavelet transform. It is **simpler** and **more useful** than the continuous one.



$$x_L[n] = \sum_k x[n-k]g[k]$$

$$x_{1,L}[n] = \sum_k x[2n-k]g[k]$$

$$x_H[n] = \sum_k x[n-k]h[k]$$

$$x_{1,H}[n] = \sum_k x[2n-k]h[k]$$

$$y[n] \longrightarrow \boxed{\downarrow Q} \longrightarrow y[Qn]$$

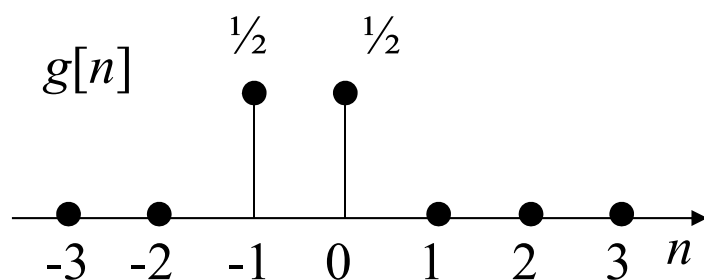
$$x_{1,L}[n] = \sum_k x[2n-k]g[k]$$

$$x_{1,H}[n] = \sum_k x[2n-k]h[k]$$

例子：2-point Haar wavelet

$$g[n] = 1/2 \text{ for } n = -1, 0$$

$$g[n] = 0 \text{ otherwise}$$



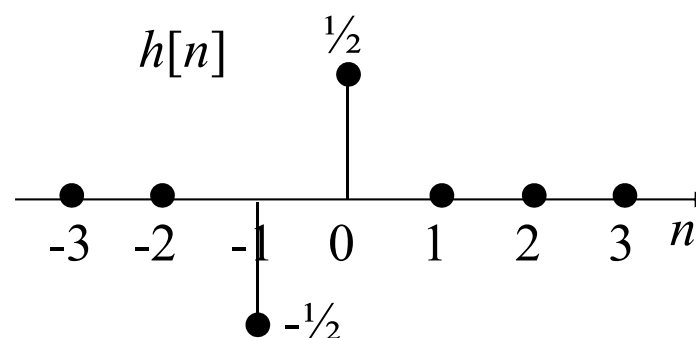
then

$$x_{1,L}[n] = \frac{x[2n] + x[2n+1]}{2}$$

(兩點平均)

$$h[0] = 1/2, \quad h[-1] = -1/2,$$

$$h[n] = 0 \text{ otherwise}$$



$$x_{1,H}[n] = \frac{x[2n] - x[2n+1]}{2}$$

(兩點之差)

Remember: For the 2-point DFT $X[m] = \sum_k e^{-j\frac{2\pi mn}{2}} x[n] = \sum_k (-1)^{mn} x[n]$

39

$$\begin{bmatrix} X[0] \\ X[1] \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} x[0] \\ x[1] \end{bmatrix}$$

一般的 wavelet, $g[n]$ 和 $h[n]$ 點數會多於 2 點

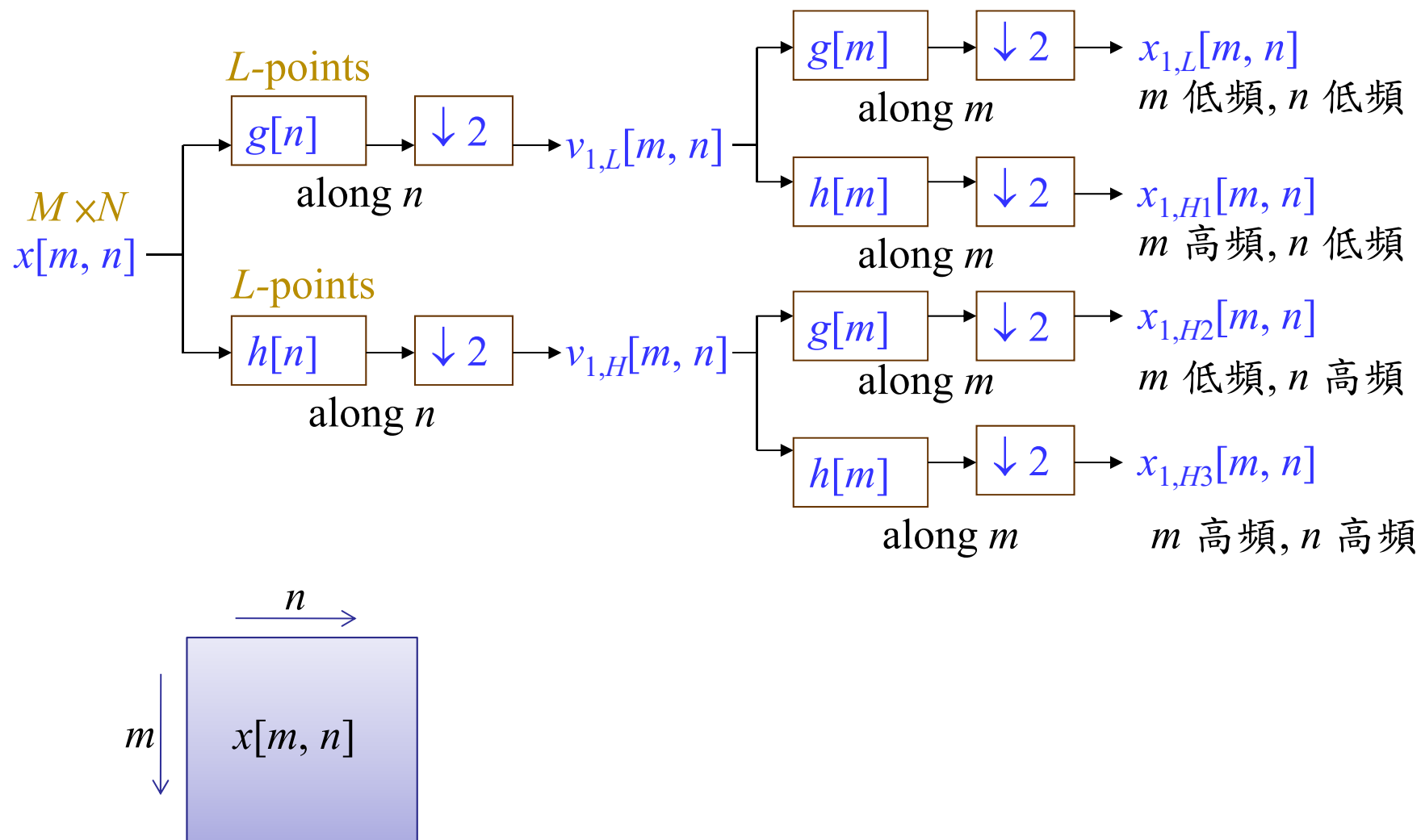
但是 $g[n]$ 通常都是 lowpass filter 的型態

$h[n]$ 通常都是 highpass filter 的型態

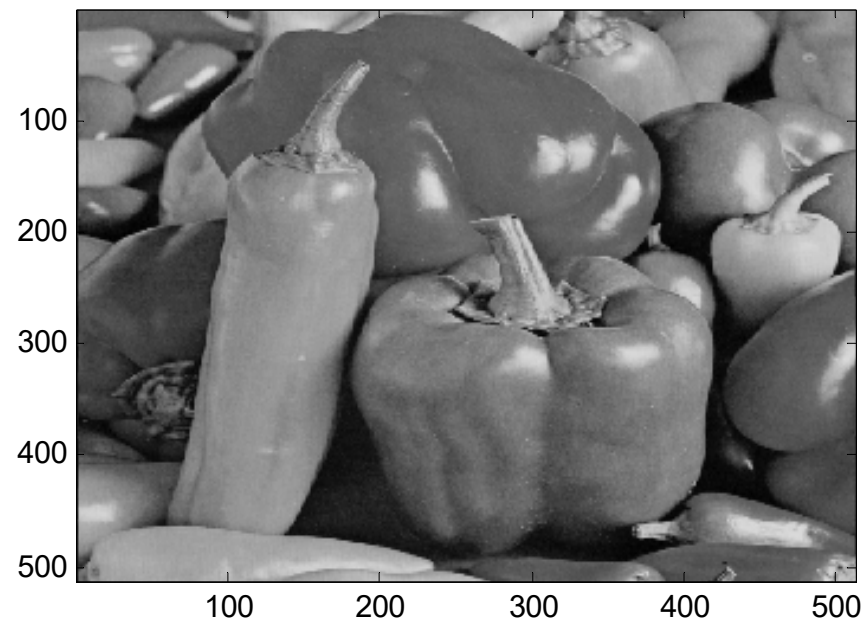
Discrete wavelet transform 有很多種

(discrete Haar wavelet, discrete Daubechies wavelet, B-spline DWT, symlet, coilet,)

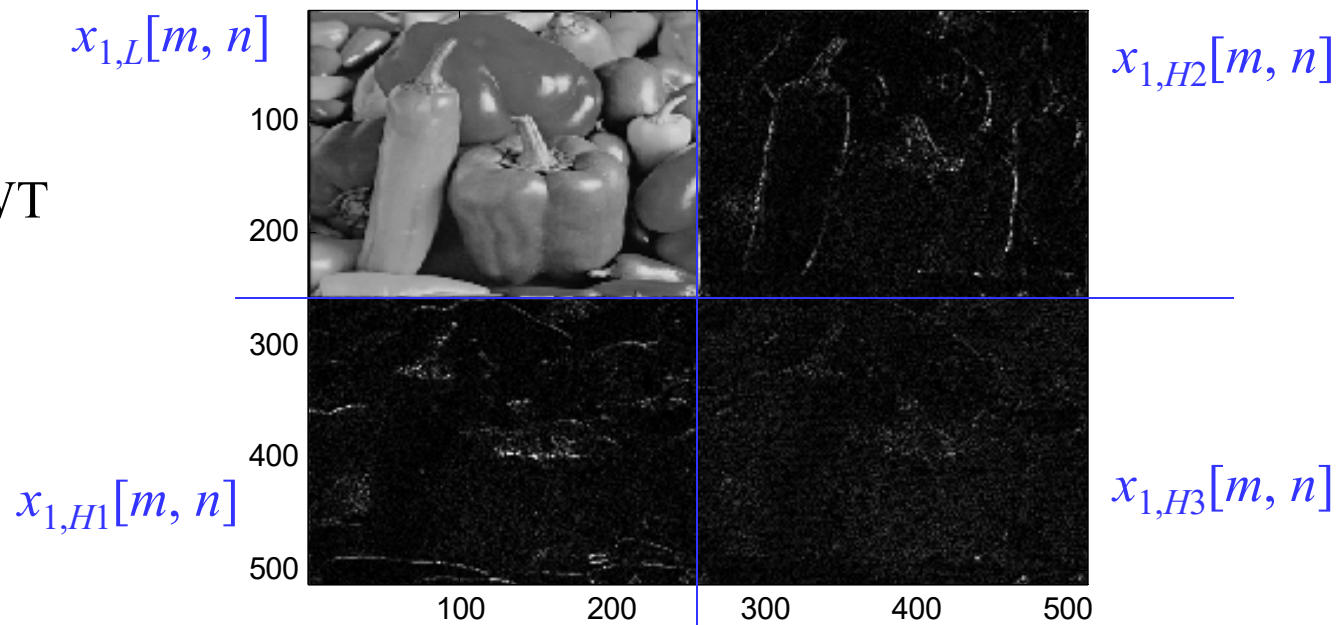
2-D 的情形



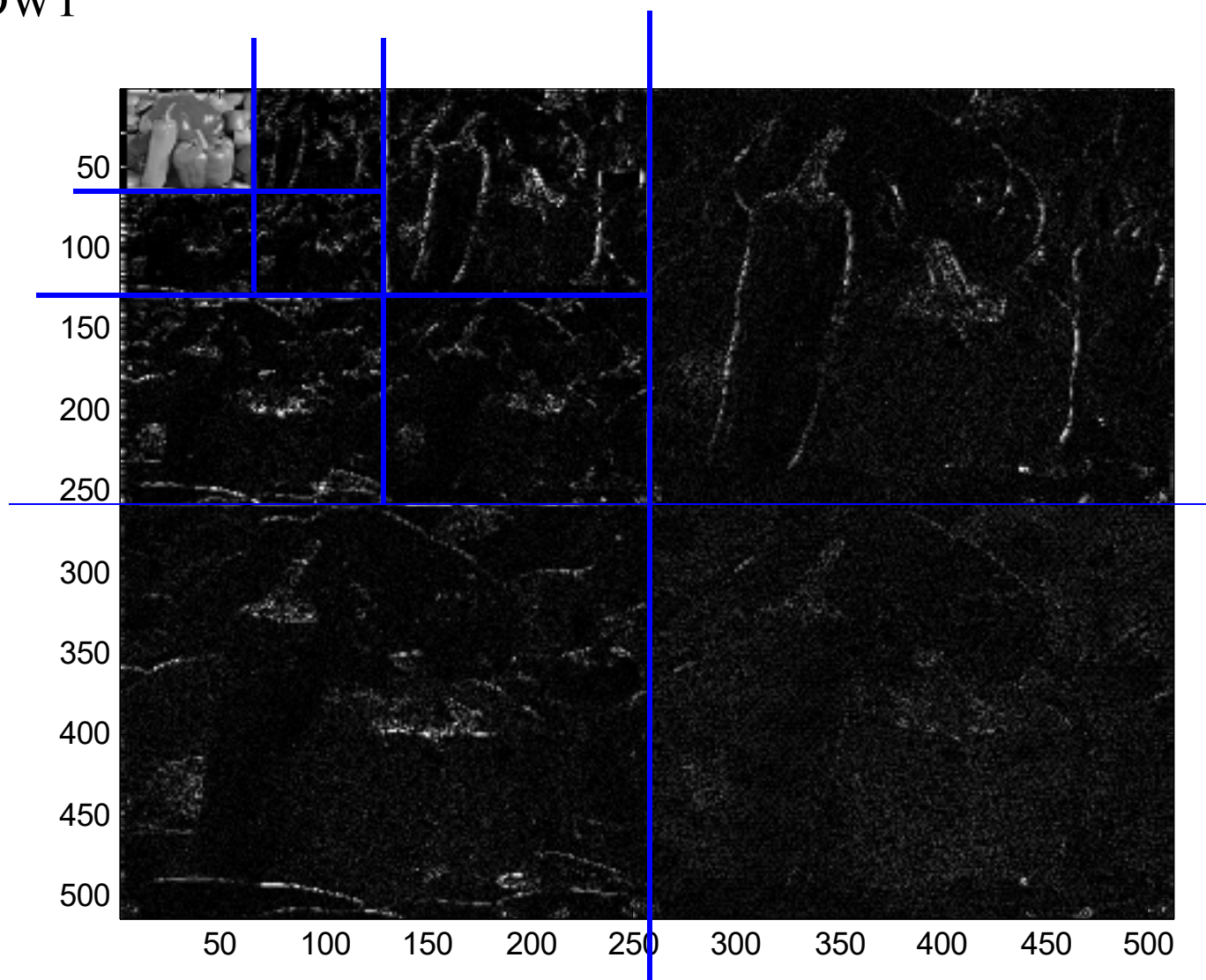
原影像
peppers.bmp



2-D DWT
的結果



3次2-D DWT 的結果



應用：影像壓縮 (JPEG 2000)

其他應用：edge detection

corner detection

filter design

pattern recognition

music signal processing

economical data

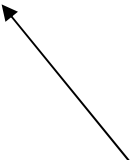
temperature analysis

feature extraction

biomedical signal processing

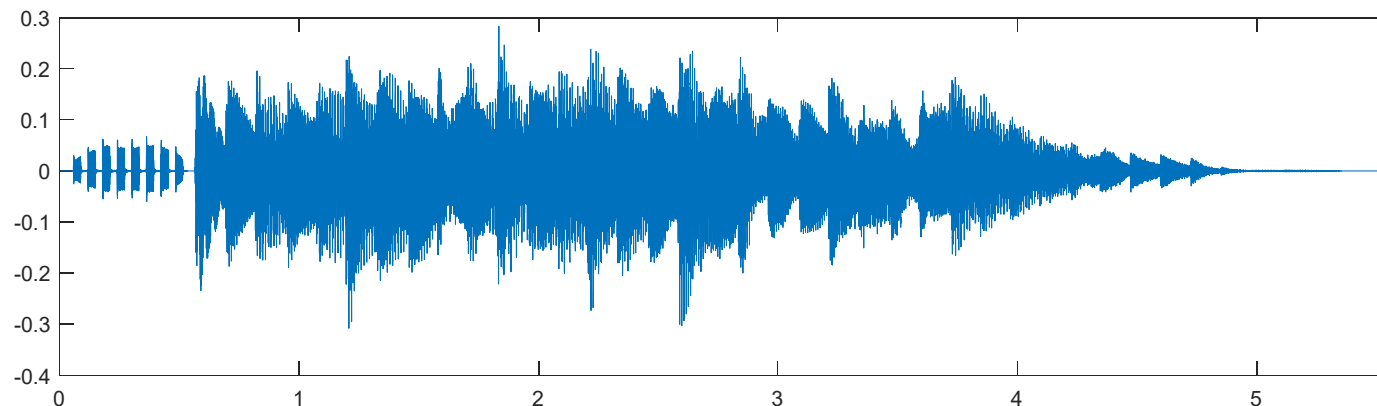
附錄一：聲音檔的處理 by Matlab

A. 讀取聲音檔

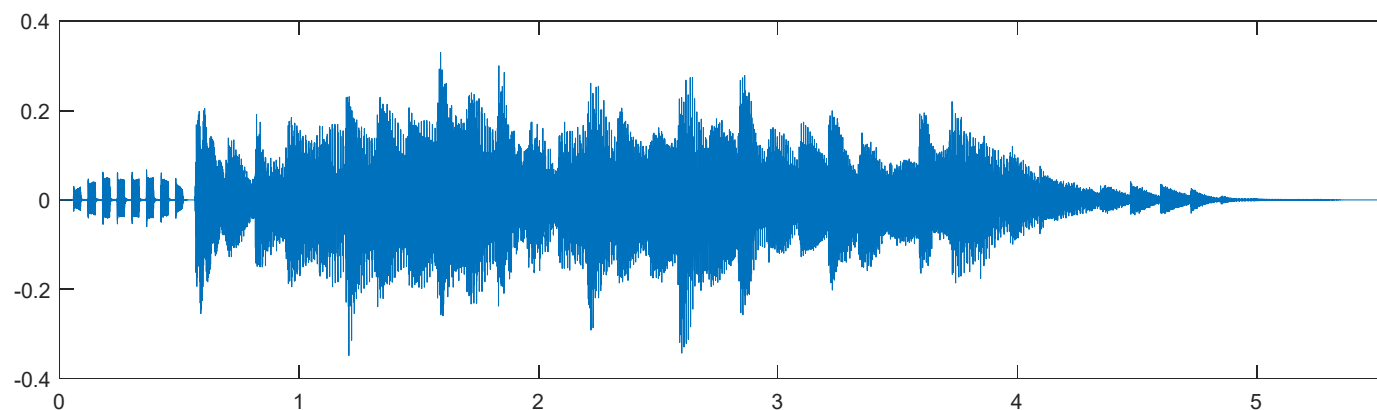
- 電腦中，沒有經過壓縮的聲音檔都是 *.wav 的型態
有經過壓縮的聲音檔是 *.mp3 的型態
 - 讀取：audioread
註：2015版本以後的 Matlab，**wavread** 將改為 **audioread**
 - 例：[**x**, **fs**] = audioread('C:\WINDOWS\Media\Alarm01.wav');
可以將 Alarm01.wav 以數字向量 **x** 來呈現。 **fs**: sampling frequency
這個例子當中 size(**x**) = 122868 2 fs = 22050

 - 思考：所以，取樣間隔多大？
 - 這個聲音檔有多少秒？
- 雙聲道 (Stereo，俗稱立體聲)

畫出聲音的波型

```
time = [0:size(x,1)-1]/fs; % x 是前頁用 audioread 所讀出的向量  
subplot(2,1,1); plot(time, x(:,1)); xlim([time(1),time(end)])
```



```
subplot(2,1,2); plot(time, x(:,2)); xlim([time(1),time(end)])
```



注意：*.wav 檔中所讀取的資料，值都在 -1 和 +1 之間

B. 繪出頻譜

`X = fft(x(:,1));` % 只做這一步無法得出正確的頻譜

`X=X.';`

`N=length(X); N1=round(N/2);`

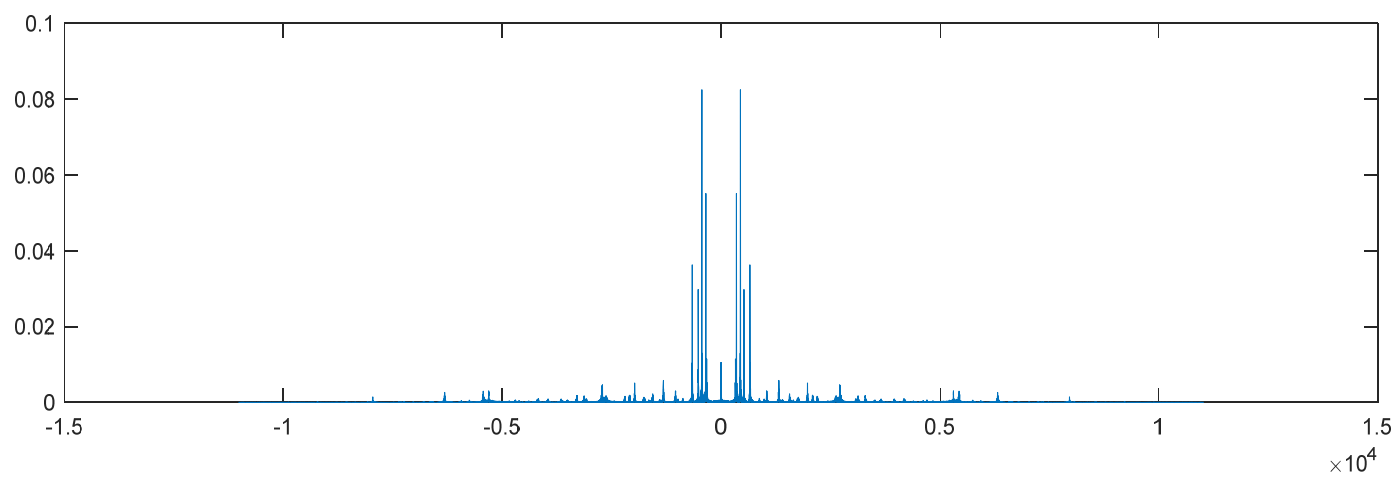
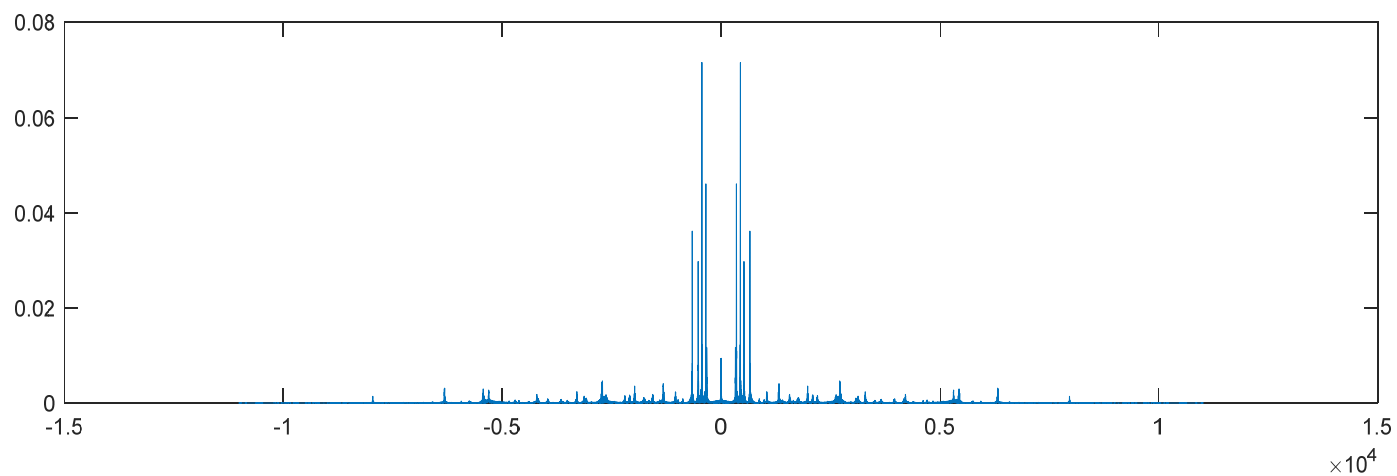
`dt=1/fs;`

`X1=[X(N1+1:N),X(1:N1)]*dt;` % shifting for spectrum

`f=[(N1:N-1)-N,0:N1-1]/N*fs;` % valid f

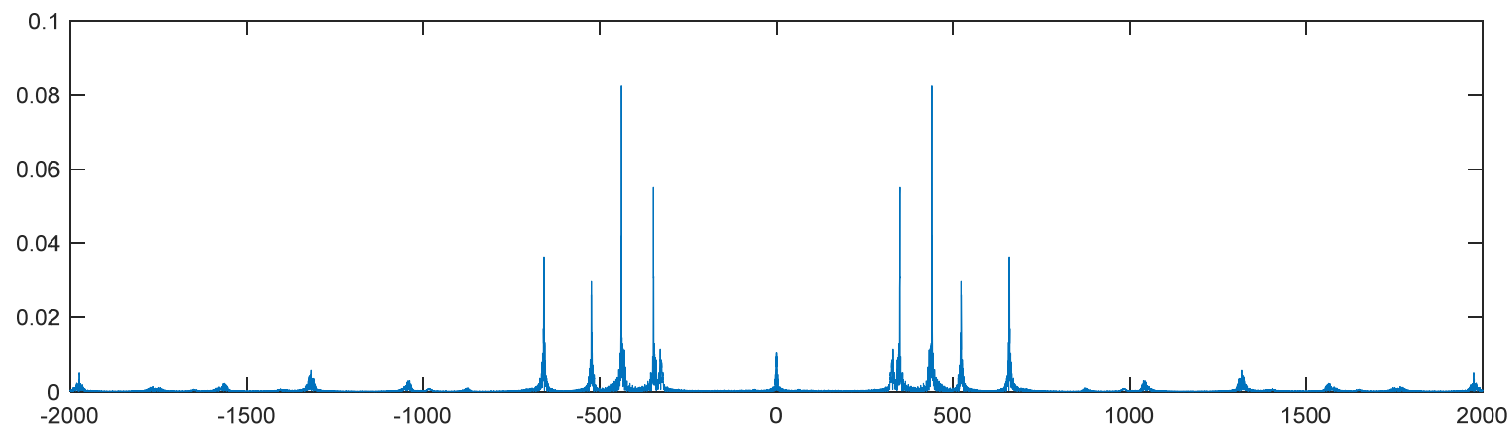
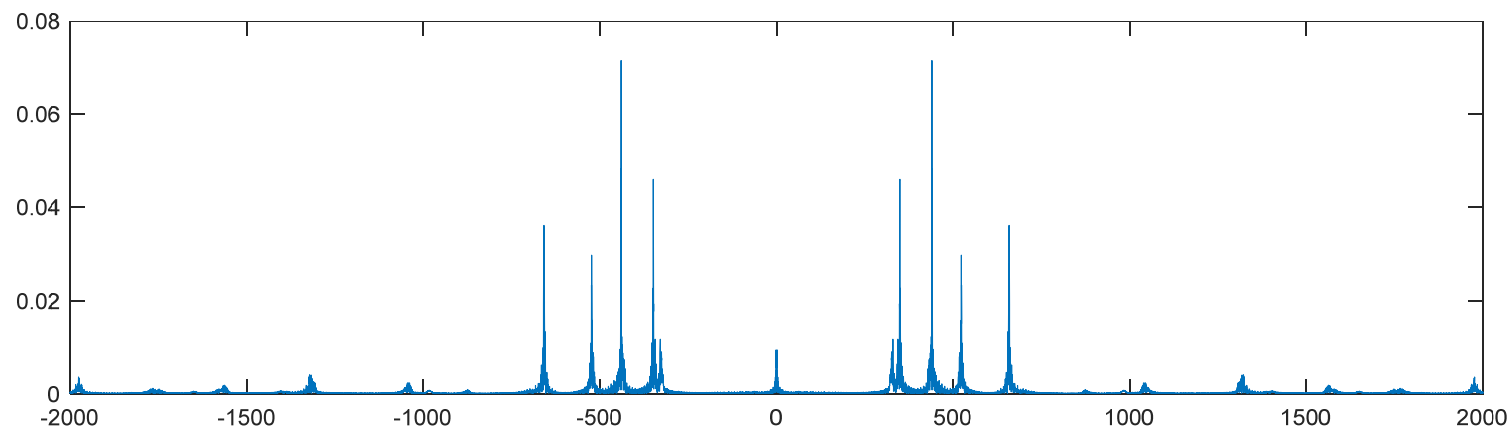
`plot(f, abs(X1));`

Alarm01.wav 的頻譜



Alarm01.wav 的頻譜

`xlim([-2000,2000])` % 只看其中 -2000Hz ~ 2000Hz 的部分



C. 聲音的播放

(1) `sound(x)`: 將 **x** 以 8192Hz 的頻率播放

(2) `sound(x, fs)`: 將 **x** 以 **fs** Hz 的頻率播放

Note: (1)~(3) 中 **x** 必需是1個column (或2個 columns)，且 **x** 的值應該介於 -1 和 +1 之間

(3) `soundsc(x, fs)`: 自動把 **x** 的值調到 -1 和 +1 之間再播放

D. 製作音檔： `audiowrite`

`audiowrite(filename, x, fs)`

將數據 **x** 變成一個 *.wav 檔，取樣速率為 **fs** Hz

① **x** 必需是1個column (或2個 columns)

② **x** 值應該介於 -1 和 +1

E. 錄音的方法

錄音之前，要先將電腦接上麥克風，且確定電腦有音效卡
(部分的 notebooks 不需裝麥克風即可錄音)

範例程式：

```
Sec = 3;  
Fs = 8000;  
recorder = audiorecorder(Fs, 16, 1);  
recordblocking(recorder, Sec);  
audioarray = getaudiodata(recorder);
```

執行以上的程式，即可錄音。

錄音的時間為三秒，sampling frequency 為 8000 Hz

錄音結果為 audioarray，是一個 column vector (如果是雙聲道，則是兩個 column vectors)

範例程式 (續) :

```
audioplay(audioarray, Fs);           % 播放錄音的結果  
t = [0:length(audioarray)-1]./Fs;  
plot (t, audioarray');               % 將錄音的結果用圖畫出來  
xlabel('sec','FontSize',16);  
audiowrite(audioarray, Fs, 'test.wav') % 將錄音的結果存成 *.wav 檔
```

指令說明：

`recorder = audiorecorder(Fs, nb, nch);` (提供錄音相關的參數)

Fs: sampling frequency,

nb: using nb bits to record each data

nch: number of channels (1 or 2)

`recordblocking(recorder, Sec);` (錄音的指令)

recorder: the parameters obtained by the command “audiorecorder”

Sec: the time length for recording

`audioarray = getaudiodata(recorder);`

(將錄音的結果，變成 audioarray 這個 column vector，如果是雙聲道，則 audioarray 是兩個 column vectors)

以上這三個指令，要並用，才可以錄音

附錄二 使用 Python 處理音訊的方法

可以先安裝幾個模組

```
pip install numpy  
pip install scipy  
pip install matplotlib # plot  
pip install pipwin  
pipwin install simpleaudio # vocal files  
pipwin install pyaudio
```

後面將說明使用 Python 讀檔，畫出頻譜，撥放聲音，製作音檔，錄音的方法

PS: 謝謝2021年擔任助教的蔡昌廷同學協助製作

A. 讀音訊檔

要先import 相關模組：`import scipy.io.wavfile as wavfile`

讀取音檔：

```
fs, wave_data = wavfile.read('C:/WINDOWS/Media/Alarm01.wav')
```

```
# fs: sampling frequency
```

```
# If the audio file has one channel, then wave_data is a column vector
```

```
# If the audio file has two channels, then wave_data has two column  
vectors
```

```
num_frame = len(wave_data) # 音訊長度:
```

```
n_channel = int(wave_data.size/ num_frame) # channels 數量
```

```
>>> fs
```

```
22050
```

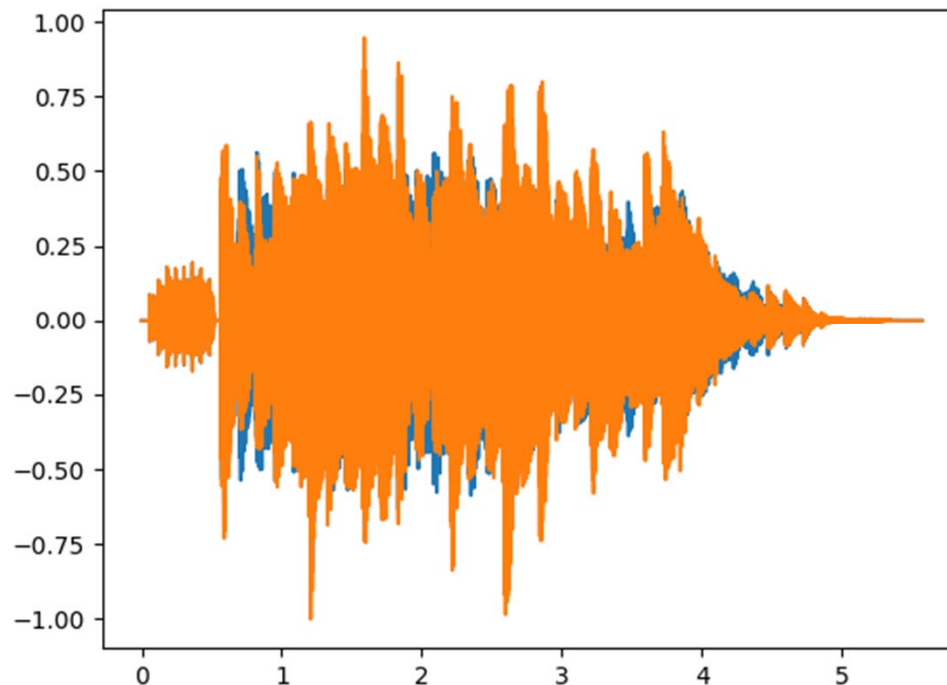
```
>>> num_frame, n_channel
```

```
(1150416, 2)
```

畫出音訊波形圖

要先import 相關模組：`import matplotlib.pyplot as plt`

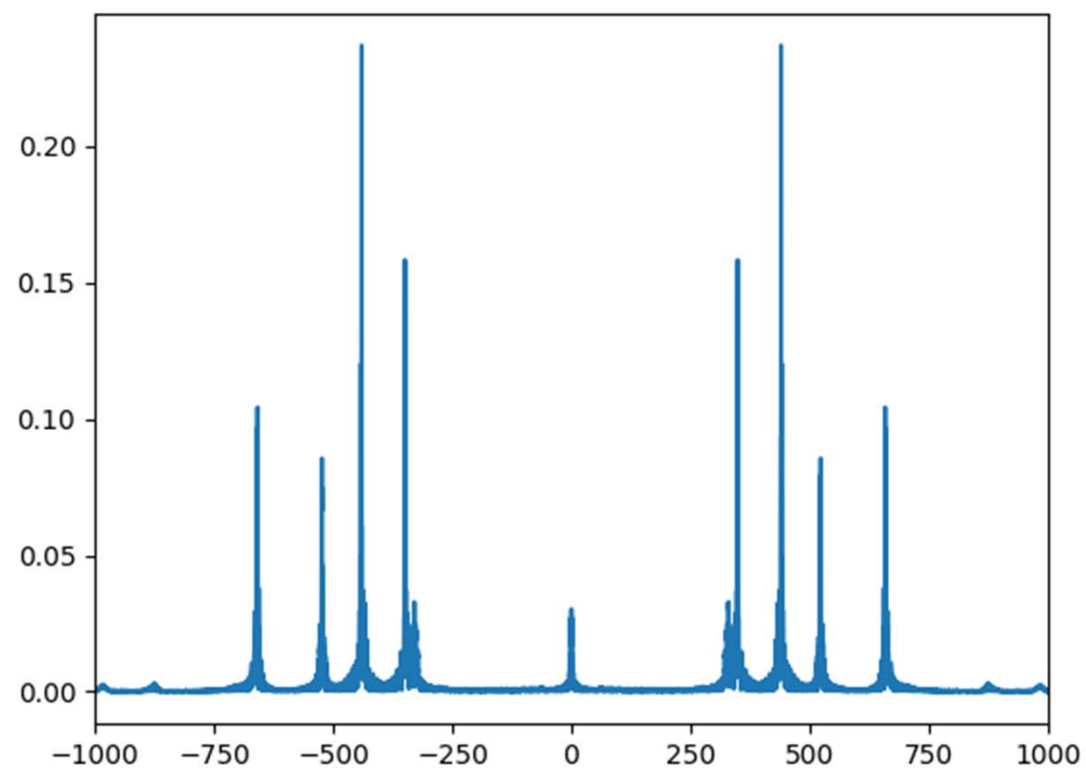
- `time = np.arange(0, num_frame)*1/fs`
- `plt.plot(time, wave_data)`
- `plt.show()`



B. 畫出頻譜

要先import 相關模組： `from scipy.fftpack import fft`

- `fft_data = abs(fft(wave_data[:,1]))/fs` # only choose the 1st channel
注意要乘上 $1/fs$
- `n0=int(np.ceil(num_frame/2))`
- `fft_data1=np.concatenate([fft_data[n0:num_frame],fft_data[0:n0]])`
將頻譜後面一半移到前面
- `freq=np.concatenate([range(n0-num_frame,0),range(0,n0)])*fs/num_frame`
頻率軸跟著調整
- `plt.plot(freq,fft_data1)`
- `plt.xlim(-1000,1000)` # 限制頻率的顯示範圍
- `plt.show()` # 如後圖



C. 播放聲音

要先import 相關模組： `import simpleaudio as sa`

- `n_bytes = 2` # using two bytes to record a data
- `wave_data = (2**15-1)* wave_data`
change the range to $-2^{15} \sim 2^{15}$
- `wave_data = wave_data.astype(np.int16)`
- `play_obj = sa.play_buffer(wave_data, n_channel, n_bytes, fs)`
- `play_obj.wait_done()`

D. 製作音檔

- `wavfile.write(file name, fs, data)`
fs means the sampling frequency
data should be a one-column or two column array

Example:

- `wavfile.write('Alarm01_test.wav', 22050, wave_data)`

E. 錄音

要先import 相關模組： `import pyaudio`

範例程式

```
import pyaudio
pa=pyaudio.PyAudio()
fs = 44100
chunk = 1024
stream = pa.open(format=pyaudio.paInt16, channels=1,
rate=fs, input=True, frames_per_buffer=chunk)

vocal=[]
count=0
```

```
while count < 200: #控制錄音時間
    audio = stream.read(chunk) #一次性錄音取樣位元組大小
    vocal.append(audio)
    count +=1

save_wave_file('testrecord.wav',vocal)
stream.close()
```

參考

<https://codertw.com/%E7%A8%8B%E5%BC%8F%E8%AA%9E%E8%A8%80/491427/>