

V. Homomorphic Signal Processing

同質

◎ 5-A Homomorphism

Homomorphism is a way of “carrying over” operations from one algebra system into another.

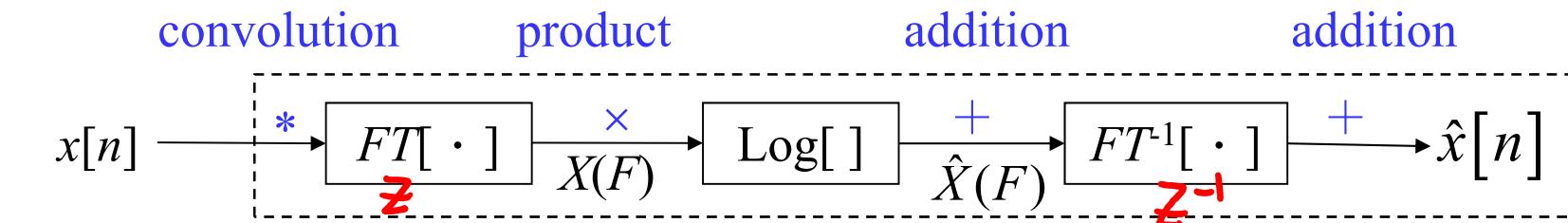
Ex. convolution $\xrightarrow{\text{Fourier}}$ multiplication $\xrightarrow{\log}$ addition

把複雜的運算，變成效能相同但較簡單的運算

◎ 5-B Cepstrum 倒頻譜

$$\hat{X}(Z) \Big|_{z=e^{i2\pi F}} = \log X(Z) \Big|_{z=e^{i2\pi F}} = \log |X(Z)| \Big|_{z=e^{i2\pi F}} + j \arg[X(e^{i2\pi F})]$$

For the process of cepstrum (denoted by $D_*(\cdot)$)



$$\hat{X}(F) = \log(X(F)) \quad \hat{H}(F) = \log(H(F))$$

FT : discrete-time Fourier transform

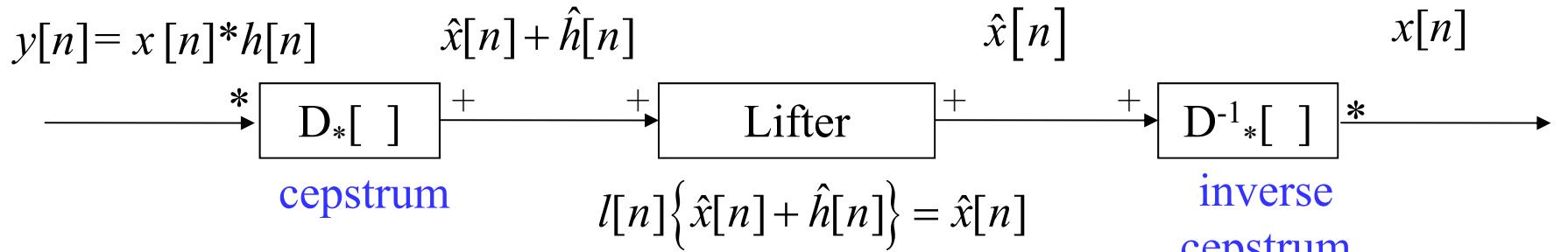
$$\hat{x}[n] = \text{IFT}(\log(X(F)))$$

$$\hat{h}[n] = \text{IFT}(\log(H(F)))$$

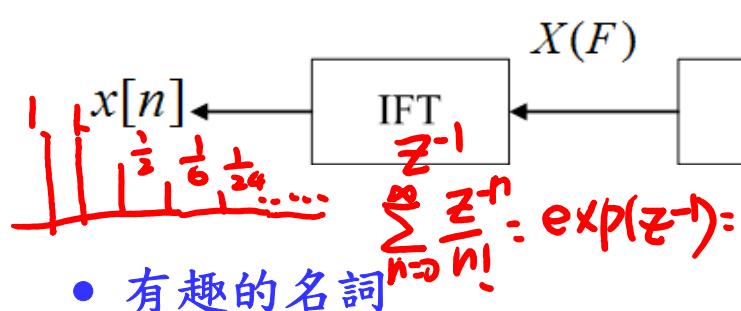
$$e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!}$$

183

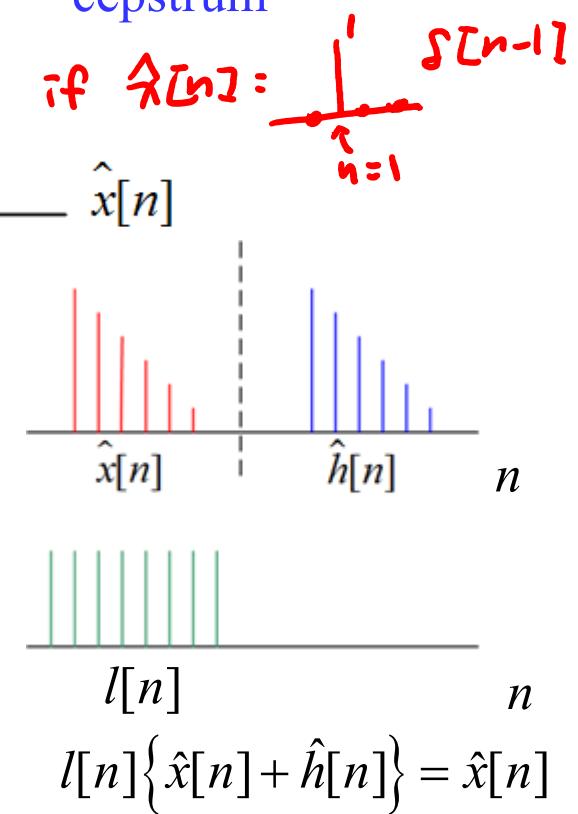
- 由 $y[n] = x[n] * h[n]$ 重建 $x[n]$



For the process of the inverse cepstrum $D^{-1}_*(\cdot)$



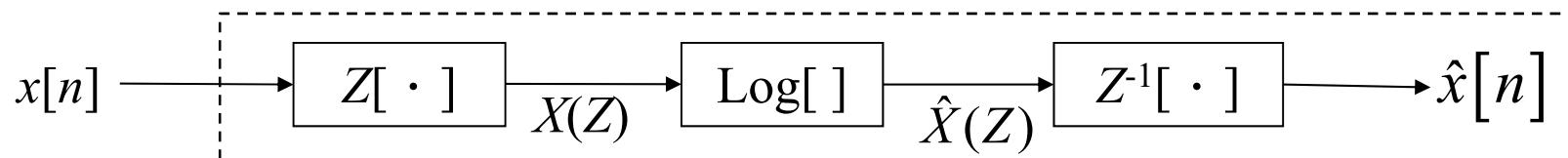
$$\hat{x}(z) = \sum_n \hat{x}[n] z^{-n}$$



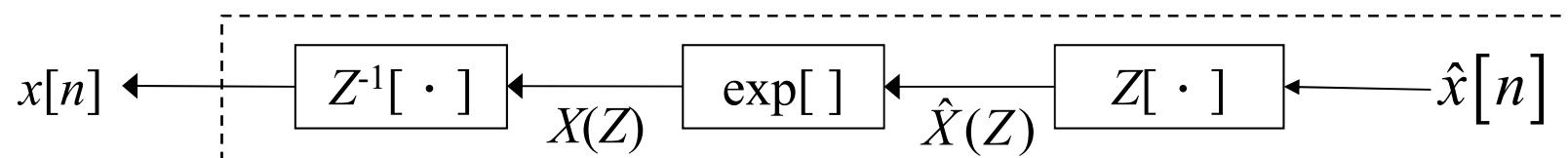
$\hat{x}[n]$	cepstrum	$\xrightarrow{\text{spectrum}}$ 倒頻譜
n	quefrency	$\xrightarrow{\text{frequency}}$ 倒頻率
$l[n]$	lifter	$\xrightarrow{\text{filter}}$ 倒濾波器

Using the Z transforms instead of the Fourier transforms:

For the process of [cepstrum](#)



For the process of the [inverse cepstrum](#)



◎ 5-C Methods for Computing the Cepstrum

- **Method 1:** Compute the inverse discrete time Fourier transform:

$$\hat{x}[n] = \int_{-1/2}^{1/2} \hat{X}(F) e^{i2\pi n F} dF \quad : \text{inverse F.T}$$

where $\hat{X}(F) = \log|X(F)| + j \arg[X(F)]$

ex: $X(F) = \bar{J}$ 通解? $|X(F)| = 1$, $\arg(X(F)) = \frac{\pi}{2}, \frac{-3\pi}{2}, \dots$ ambiguity for phase

$$\hat{X}(F) = \log X(F)$$

$$X(F) = |X(F)| e^{j \arg X(F)}$$

- Problems:
- (1) $\log|X(F)| \rightarrow -\infty$ if $|X(F)| \rightarrow 0$
 - (2) $\arg(X(F))$ has infinite number of solutions.

Actually, the COMPLEX Cepstrum is REAL for real input

• Method 2 (From Poles and Zeros of the Z Transform)

實際上計算
cepstrum的方法

$$X(Z) = \frac{A \cancel{\prod_{k=1}^{m_i} (1 - a_k Z^{-1})}}{\prod_{k=1}^{P_i} (1 - c_k Z^{-1})} \frac{\prod_{k=1}^{m_0} (1 - b_k Z)}{\prod_{k=1}^{P_0} (1 - d_k Z)}$$

where $|a_k|, |b_k|, |c_k|, |d_k| \leq 1$

a_k : zeros inside unit circle
 c_k : poles inside unit circle

b_k^{-1} : zeros outside unit circle
 d_k^{-1} : poles outside unit circle

Π: 連乘
 Σ: 連加

$$\begin{aligned}\therefore \hat{X}(Z) &= \log X(Z) = \log A + r \cdot \cancel{\log Z} + \sum_{k=1}^{m_i} \log (1 - a_k Z^{-1}) + \sum_{k=1}^{m_0} \log (1 - b_k Z) \\ &\quad - \sum_{k=1}^{P_i} \log (1 - c_k Z^{-1}) - \sum_{k=1}^{P_0} \log (1 - d_k Z)\end{aligned}$$

$$\begin{aligned}ex: \quad &z^2 - 0.7z + 0.1 \\ &= (z - 0.5)(z - 0.2) \\ &= z^2(1 - 0.5z^{-1})(1 - 0.2z^{-1})\end{aligned}$$

$$\therefore \hat{X}(Z) = \log X(Z) = \log A + r \cdot \cancel{\log Z} + \sum_{k=1}^{m_i} \log (1 - a_k Z^{-1}) + \sum_{k=1}^{m_0} \log (1 - b_k Z)$$

$$- \sum_{k=1}^{P_i} \log (1 - c_k Z^{-1}) - \sum_{k=1}^{P_0} \log (1 - d_k Z)$$

↓
 (inverse Z transform)
 ?

Taylor series

$$f(t) = f(t_0) + \sum_{n=1}^{\infty} \frac{f^{(n)}(t_0)}{n!} (t - t_0)^n$$

$$\begin{aligned} \log(1 - b_k z) &= \sum_{n=1}^{\infty} \frac{(-1)^{n-1}}{n} (-1)^n b_k^n z^n \\ &= \sum_{n=1}^{\infty} -\frac{1}{n} b_k^n z^n \quad n \rightarrow -n \\ &= \sum_{n=-\infty}^{-1} \frac{1}{n} b_k^{-n} z^{-n} \end{aligned}$$

$$\frac{d^n}{dt^n} \log(1+t) = \frac{(-1)^{n-1} (n-1)!}{c(1+t)^n}$$

$$\log(1+t) = \sum_{n=1}^{\infty} \frac{(-1)^{n-1} (n-1)!}{n!} t^n \quad (t_0=0)$$

$$\log(1+t) = \sum_{n=1}^{\infty} \frac{(-1)^{n-1}}{n} t^n$$

$$\begin{aligned} \log(1 - a_k z^{-1}) &= \sum_{n=1}^{\infty} \frac{(-1)^{n-1}}{n} (-1)^n a_k^n z^{-n} \\ &= \sum_{n=1}^{\infty} \frac{-1}{n} a_k^n z^{-n} \end{aligned}$$

Taylor series expansion

Z^{-1}

(Suppose that $r = 0$)

$$\hat{x}[n] = \begin{cases} \log(A) & , n = 0 \\ -\sum_{k=1}^{m_i} \frac{a_k^{-n}}{n} + \sum_{k=1}^{P_i} \frac{c_k^{-n}}{n} & , n > 0 \\ \sum_{k=1}^{m_0} \frac{b_k^{-n}}{n} - \sum_{k=1}^{P_0} \frac{d_k^{-n}}{n} & , n < 0 \end{cases}$$

Poles & zeros inside unit circle, right-sided sequence

Poles & zeros outside unit circle, left-sided sequence

Note:

- (1) $\hat{x}[n]$ always decays with $|n|$.
- (2) 在 complex cepstrum domain
Minimum phase 及 maximum phase 之貢獻以 $n = 0$ 為分界切開
- (3) For FIR case, there is no c_k and d_k
- (4) The complex cepstrum is unique and of infinite duration for both positive & negative n , even though $x[n]$ is causal & of finite durations

$\hat{x}[n]$ is always IIR

● Method 3

$$Z \cdot \hat{X}'(Z) = Z \cdot \frac{X'(Z)}{X(Z)}$$

$$\therefore ZX'(Z) = Z\hat{X}'(Z) \cdot X(Z)$$

$$\downarrow Z^{-1}$$

$$n x[n] = \sum_{k=-\infty}^{\infty} k \hat{x}[k] x[n-k]$$

$$\therefore x[n] = \sum_{k=-\infty}^{\infty} \frac{k}{n} \hat{x}[k] x[n-k] \quad \text{for } n \neq 0$$

Suppose that $x[n]$ is causal and has minimum phase, i.e. $x[n] = \hat{x}[n] = 0, n < 0$

$$x[n] = \sum_{k=-\infty}^{\infty} \frac{k}{n} \hat{x}[k] x[n-k] \quad \text{for } n \neq 0$$

$$\Rightarrow x[n] = \sum_{k=0}^n \frac{k}{n} \hat{x}[k] x[n-k] \quad \text{for } n > 0 \quad (\text{causal sequence})$$

$$x[n] = \hat{x}[n] x[0] + \sum_{k=0}^{n-1} \frac{k}{n} \hat{x}[k] x[n-k]$$

For a minimum phase sequence $x[n]$

$$\hat{x}[n] = \begin{cases} 0 & , n < 0 \\ \frac{x[n]}{x[0]} - \sum_{k=0}^{n-1} \left(\frac{k}{n} \right) \hat{x}[k] \frac{x[n-k]}{x[0]} & , n > 0 \\ \log A & , n = 0 \end{cases} \quad \text{recursive method}$$

Determining $\hat{x}[n]$ from $\hat{x}[0], \hat{x}[1], \dots, \hat{x}[n-1]$

For anti-causal and maximum phase sequence, $x[n] = \hat{x}[n] = 0, n > 0$

$$\begin{aligned} x[n] &= \sum_{k=n}^0 \frac{k}{n} \hat{x}[k] x[n-k] \quad , n < 0 \\ &= \hat{x}[n] x[0] + \sum_{k=n+1}^0 \frac{k}{n} \hat{x}[k] x[n-k] \end{aligned}$$

For maximum phase sequence,

$$\hat{x}[n] = \begin{cases} 0 & , n > 0 \\ \log A & , n = 0 \\ \frac{x[n]}{x[0]} - \sum_{k=n+1}^0 \left(\frac{k}{n}\right) \hat{x}[k] \frac{x[n-k]}{x[0]} & , n < 0 \end{cases}$$

◎ 5-D Properties

P.1) The complex cepstrum decays at least as fast as $\frac{1}{n}$

$$|\hat{x}[n]| < c \left| \frac{\alpha^n}{n} \right| \quad -\infty < n < \infty$$

$$\alpha = \max(|a_k|, |b_k|, |c_k|, |d_k|)$$

P.2) If $X(Z)$ has no poles and zeros outside the unit circle, i.e. $x[n]$ is minimum phase, then

$$\hat{x}[n] = 0 \quad \text{for all } n < 0$$

because of no b_k, d_k

P.3) If $X(Z)$ has no poles and zeros inside the unit circle, i.e. $x[n]$ is maximum phase, then

$$\hat{x}[n] = 0 \quad \text{for all } n > 0$$

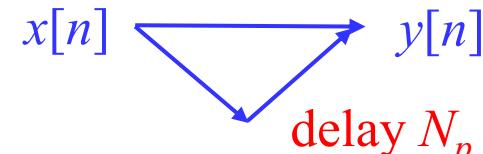
because of no a_k, c_k

P.4) If $x[n]$ is of finite duration, then
 $\hat{x}[n]$ has infinite duration

◎ 5-E Application of Homomorphic Deconvolution

(1) Equalization for Echo

$$y[n] = x[n] + \alpha x[n - N_p]$$



Let $p[n]$ be $p[n] = \delta[n] + \alpha\delta[n - N_p]$

$$y[n] = x[n] + \alpha x[n - N_p] = x[n] * p[n]$$

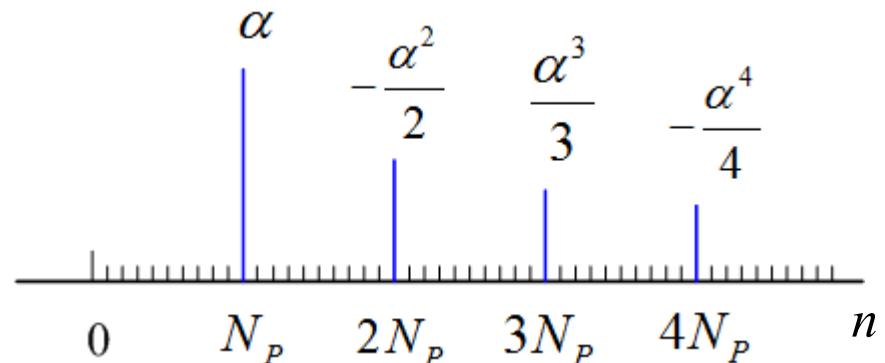
$$\log(1+t) = \sum_{k=1}^{\infty} \frac{(-1)^{k+1}}{k} t^k$$

$$P(Z) = 1 + \alpha Z^{-N_p}$$

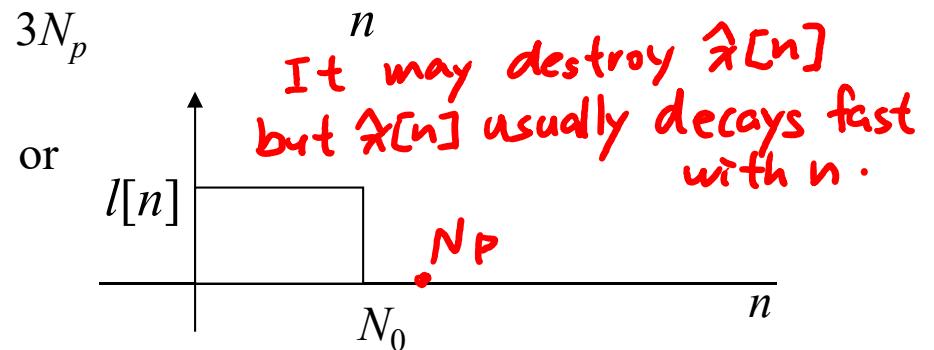
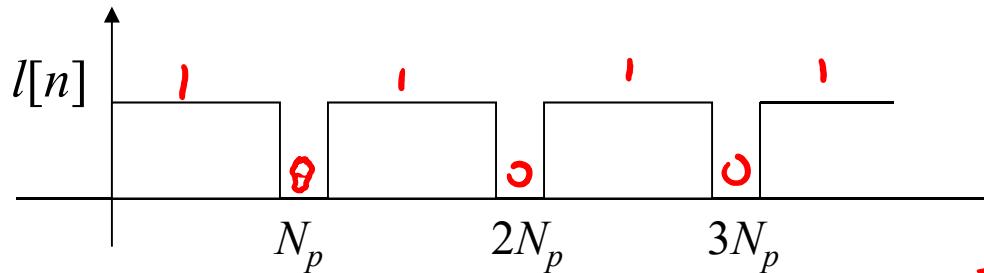
$$\hat{P}(Z) = \log(1 + \alpha Z^{-N_p}) = \sum_{k=1}^{\infty} (-1)^{k+1} \frac{\alpha^k}{k} Z^{-kN_p}$$

$$\downarrow Z^{-1}$$

$$\hat{p}[n] = \sum_{k=1}^{\infty} (-1)^{k+1} \frac{\alpha^k}{k} \delta(n - k \cdot N_p)$$



Filtering out the echo by the following “lifter”:



Q: For the case where N_p is unknown

(2) Representation of acoustic engineering

$$N_p > N_0$$

$$y[n] = x[n] * h[n]$$

Synthesiz
ed music

music
building effect : e.g. 羅馬大教堂的
impulse response

(3) Speech analysis

$$s[n] = g[n] * v[n] * p[n]$$

Speech wave Global wave shape Vocal tract impulse Pitch

聲道

只要 path 是彎來彎去的，就一定會有 multipath 的問題產生



把聲音經過聲道再從口腔發出來，就會是非直線的 path
>> 講話也有 multipath 的問題

They can be separated by filtering in the complex cepstrum domain

(4) Seismic Signals

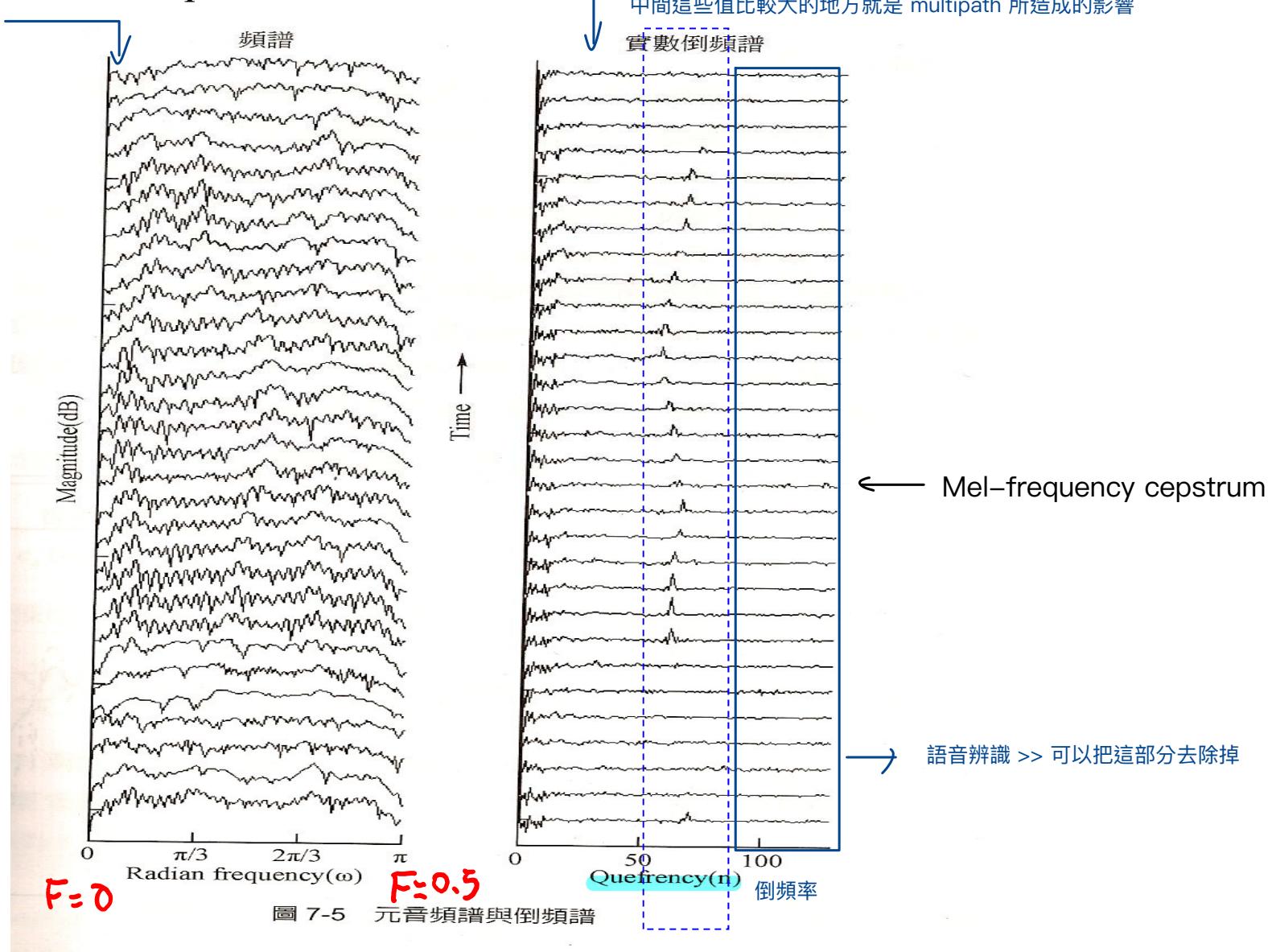
地震波

只要是物質波（要用到介質才能傳播的波）都會有 multipath 的問題
ex: 地震波 >> 經過密度不同的介質

- 不過電磁波也有 multipath 的問題

(5) Multiple-path analysis for any wave-propagation problem

用 cepstrum 將 multipath 的影響去除



omega 除 2π 換算成
normalized frequency

From 王小川，“語音訊號處理”，全華出版，台北，民國94年。

From 王小川，“語音訊號處理”，全華出版，台北，民國94年。

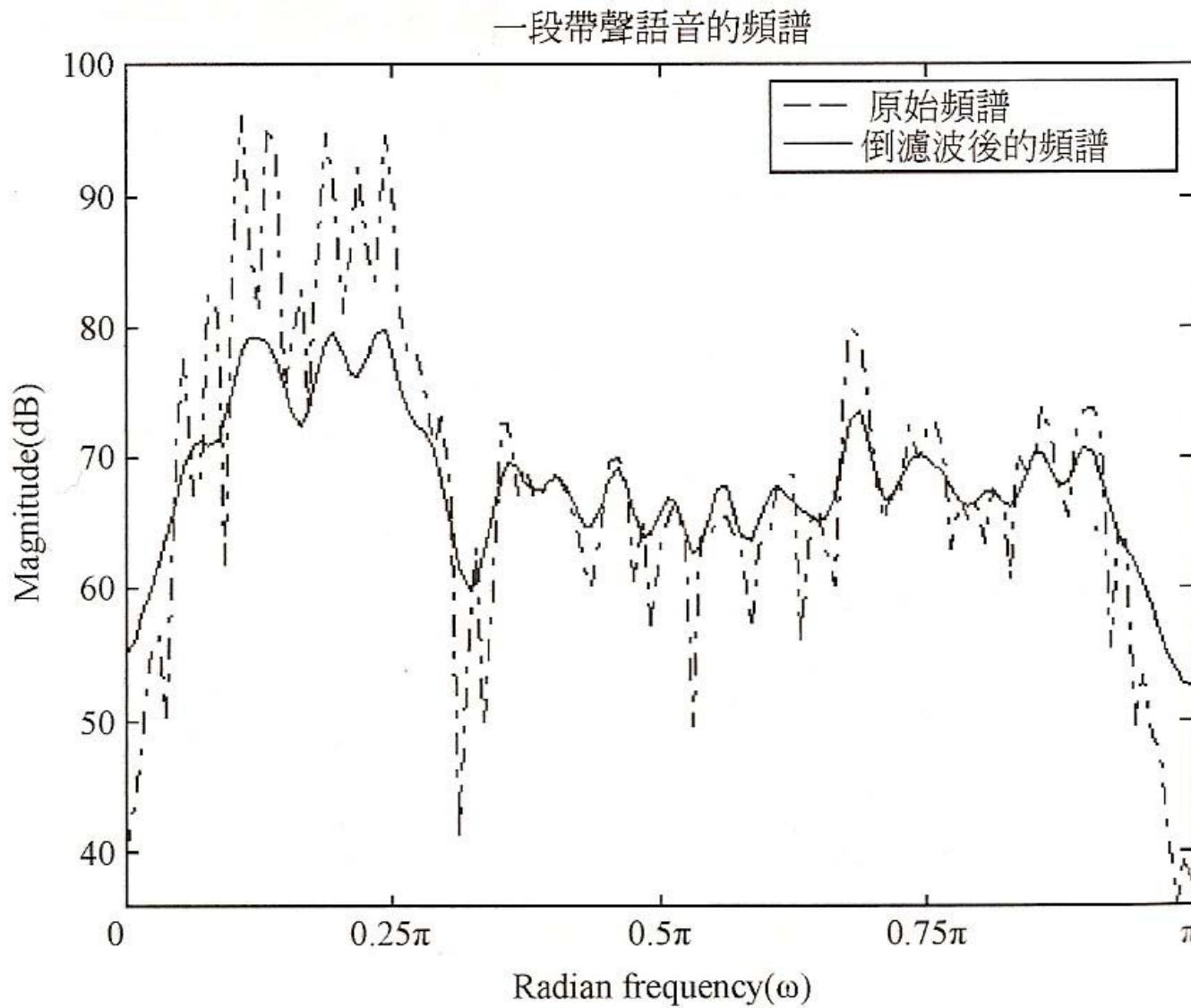


圖 7-6 經過倒濾波器作平滑處理的頻譜

◎ 5-F Problems of Cepstrum

- (1) $|\log(X(Z))|$
- (2) Phase
- (3) Delay Z^{-k}
- (4) Only suitable for the multiple-path-like problem

◎ 5-G Differential Cepstrum 微分倒頻譜

z transform 的微分除 z transform , 再取 inverse z transform

$$\hat{x}_d(n) = Z^{-1} \left[\frac{X'(Z)}{X(Z)} \right] \quad \text{或} \quad \hat{x}_d[n] = \int_{-1/2}^{1/2} \frac{X'(F)}{X(F)} e^{j2\pi F} dF$$

↑
inverse Z transform

$$\text{Note: } \frac{d}{dZ} \hat{X}(Z) = \frac{d}{dZ} \log(X(Z)) = \frac{X'(Z)}{X(Z)}$$

乘 A (可能是正實數、負實數、甚至是複數)
對微分倒頻譜沒有影響

$$\text{If } x(n) = x_1(n) * x_2(n)$$

$$X(Z) = X_1(Z) \cdot X_2(Z)$$

$$X'(Z) = X'_1(Z) \cdot X_2(Z) + X_1(Z) \cdot X'_2(Z)$$

*$x_1[n]$ and $Ax_1[n]$
has the same differential
cepstrum*

$$\boxed{\frac{X'(Z)}{X(Z)} = \frac{X'_1(Z)}{X_1(Z)} + \frac{X'_2(Z)}{X_2(Z)}}$$

同取 inverse
z transform

∴

$$\hat{x}_d(n) = \hat{x}_{1d}(n) + \hat{x}_{2d}(n)$$

Advantages: no phase ambiguity

able to deal with the delay problem

$$z^{-1} \left[\frac{x'(z)}{x(z)} \right] = z^{-1} \left[\frac{x'_1(z)}{X_1(z)} \right] + z^{-1} \left[\frac{x'_2(z)}{X_2(z)} \right]$$

- Properties of Differential Cepstrum

(1) The differential Cepstrum is shift & scaling invariant

不只適用於 multi-path-like problem

也適用於 pattern recognition

If $y[n] = A X[n - r]$

$$\Rightarrow \hat{y}_d(n) = \begin{cases} \hat{x}_d(n) & , n \neq 1 \\ -r + \hat{x}_d(1) & , n = 1 \end{cases}$$

(Proof): $Y(z) = Az^{-r} X(z)$

$$Y'(z) = Az^{-r} X'(z) - rA z^{-r-1} X(z)$$

$$\frac{Y'(z)}{Y(z)} = \frac{X'(z)}{X(z)} - rz^{-1}$$

$$\hat{y}_d(n) = \hat{x}_d(n) - r\delta(n-1)$$

(2) The complex cepstrum $\hat{C}[n]$ is closely related to its differential cepstrum $\hat{x}_d[n]$ and the signal original sequence $x[n]$

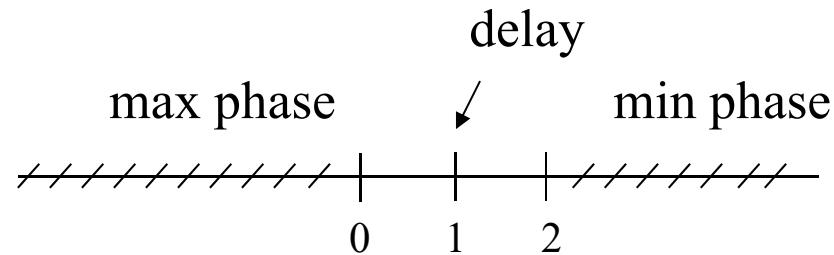
$$\hat{C}(n) = \frac{-\hat{x}_d(n+1)}{n} \quad n \neq 0 \quad \text{diff cepstrum}$$

$$\text{and} \quad -(n-1)x(n-1) = \sum_{k=-\infty}^{\infty} \hat{x}_d(n)x(n-k) \quad \text{recursive formula}$$

Complex cepstrum 做得到的事情, differential cepstrum 也做得到 !

- (3) If $x[n]$ is minimum phase (no poles & zeros outside the unit circle), then
 $\hat{x}_d[n] = 0$ for $n \leq 0$

- (4) If $x[n]$ is maximum phase (no poles & zeros inside the unit circle), then
 $\hat{x}_d[n] = 0$ for $n \geq 2$



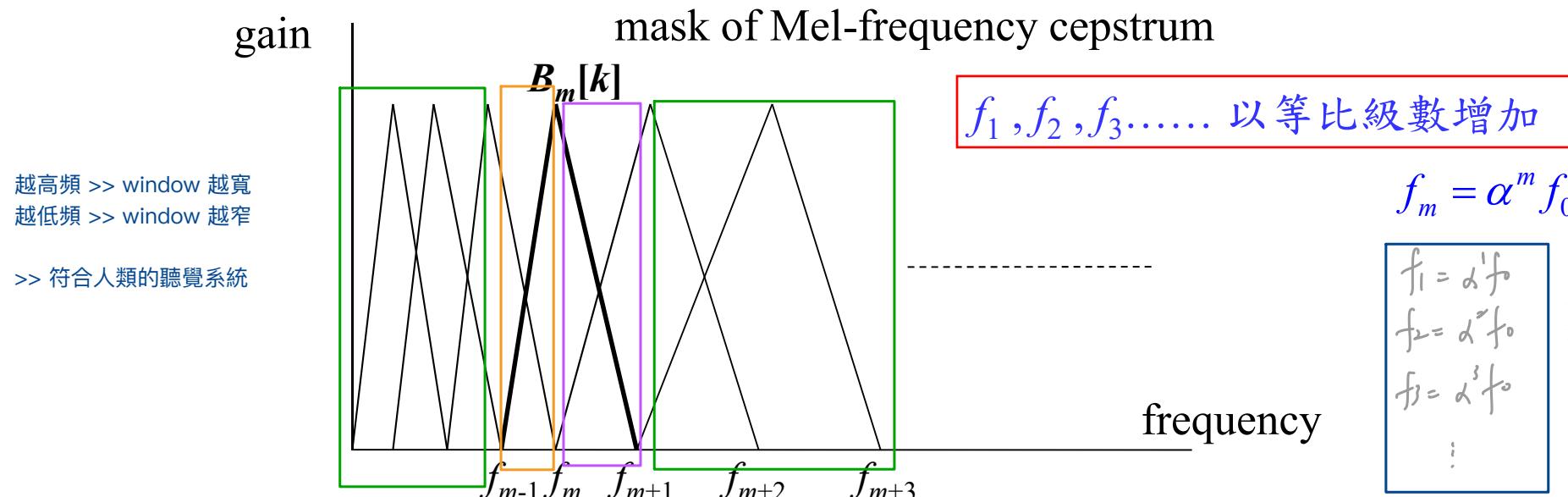
- (5) If $x(n)$ is of finite duration, $\hat{x}_d[n]$ has infinite duration

Complex cepstrum decay rate $\propto \frac{1}{n}$

Differential Cepstrum decay rate 變慢了, $\therefore \hat{x}_d(n+1) = n \cdot \hat{c}(n) \propto n \cdot \frac{1}{n} = 1$

◎ 5-H Mel-Frequency Cepstrum (梅爾頻率倒頻譜)

Take log in the frequency mask



$$B_m[k] = 0 \quad \text{for } f < f_{m-1} \text{ and } f > f_{m+1}$$

$$B_m[k] = (k - f_{m-1}) / (f_m - f_{m-1}) \quad \text{for } f_{m-1} \leq k \leq f_m$$

$$B_m[k] = (f_{m+1} - k) / (f_{m+1} - f_m) \quad \text{for } f_m \leq f \leq f_{m+1}$$

$$f = k f_s / N$$

ex: 兩組頻率：

1. 100, 200 hz
 2. 1000, 1100 hz

>> 第一組聽起來差異較大，儘管兩組頻率都是相差 100 hz，但前者相差兩倍，後者差 11 倍

>> 人耳的感知是根據等比級數來呈現的，所以才會把 window 設成等比級數（下頁

(一、二、三點的優點是依此
advantages 第三點)

Process of the Mel-Frequency Cepstrum

一樣 input 仍然是 fourier transform 的架構

$$(1) \quad x[n] \xrightarrow{FT} X[k]$$

加總起來還是 = 0 的機率下降
 >> 減少 log 0 的機率

取絕對值 >> 一定會是正實數
 >> phase 設為 0

$$(2) \quad Y[m] = \log \left\{ \sum_{k=f_{m-1}}^{f_{m+1}} |X[k]|^2 B_m[k] \right\}$$

但是我們之所以是用 z transform 而不是 fourier transform 的原因就是因為 FT:
 • 可能有 log0 >> -\infty
 • phase 有無限多的解

$$(3) \quad c_x[n] = \frac{1}{M} \sum_{m=1}^M Y[m] \cos \left(\frac{\pi n(m-1/2)}{M} \right)$$

>> 這裡就是用步驟 (2) 解決

用 inverse cosine transform 取代 inverse fourier transform (原本第一步做 FT 這裡應該要做 IFT) >> 改 cosine 的好處：

FT: 複數運算，inverse cosine：實數運算 >> 運算量較小

Q: What are the difference between the Mel-frequency cepstrum and the original cepstrum?

Advantages : (i) $\sum_{k=f_m}^{f_{m+1}} |X[k]|^2 B_m[k]$ has much smaller probability to be 0
 (ii) $|X[k]|^2$ is real and positive, which avoids phase ambiguity
 (iii) cutoff frequencies $f_m = N^m f_0$ match the human hearing system
 (iv) Using the cosine transform instead of the IFT to reduce the complexity

Mel-frequency cepstrum 更接近人耳對語音的區別性

用 $c_x[1], c_x[2], c_x[3], \dots, c_x[13]$ 即足以描述語音特徵

summation of the effect
 inside the m^{th} mask

Mel-frequency cepstrum 比較偏實務導向
犧牲了原本 cepstrum 的一些漂亮的數學性質

ex: p.188 cepstrum $\hat{x}[n]$ 隨著 $|n|$ 遞減，但 Mel-frequency cepstrum 不會有這個性質

Mel-frequency cepstrum 會和 p.188 的結果相近，但不會完全相同

◎ 5-I References

- R. B. Randall and J. Hee, “Cepstrum analysis,” *Wireless World*, vol. 88, pp. 77-80. Feb. 1982
- 王小川 , “語音訊號處理” , 全華出版 , 台北 , 民國94年。
- A. V. Oppenheim and R. W. Schafer, *Discrete-Time Signal Processing*, London: Prentice-Hall, 3rd ed., 2010.
- S. C. Pei and S. T. Lu, “Design of minimum phase and FIR digital filters by differential cepstrum,” *IEEE Trans. Circuits Syst. I*, vol. 33, no. 5, pp. 570-576, May 1986.
- S. Imai, “Cepstrum analysis synthesis on the Mel-frequency scale,” *ICASSP*, vol. 8, pp. 93-96, Apr. 1983.

附錄六：聲音檔和影像檔的處理 (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 $\text{size}(x) = 122868 \quad 2$
變成有兩個 column
- 思考：所以，取樣間隔多大？
- 這個聲音檔有多少秒？

一開始音訊的檔案是一維的

$fs = 22050$

為了節省資料量，將取樣頻率砍半（反正太高其實人耳聽不太到，大部分聲音都在 3000 Hz 以下，而且就算聽到聽了也不舒服）

雙聲道 (Stereo)，俗稱立體聲

這兩個波形會很像，但不太一樣（見下頁）

standard:

$fs = 44100 \text{ Hz}$

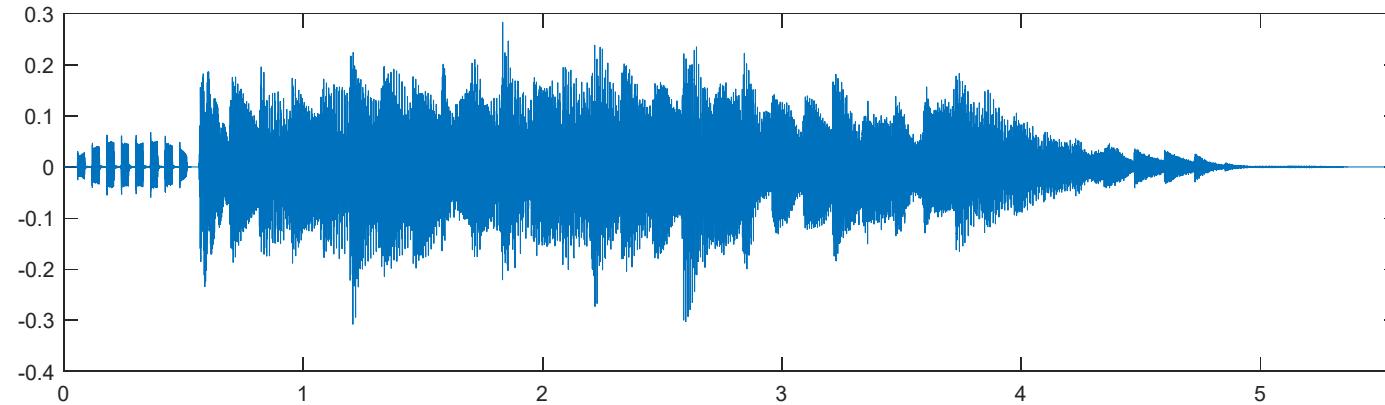
因為人耳可以聽到的頻率為 20~20000 Hz，所以根據定理，取樣至少要兩倍（四萬點），但電腦的取樣頻率多取一點，令標準為 44100

>> 意思就是每秒要取 44100 點

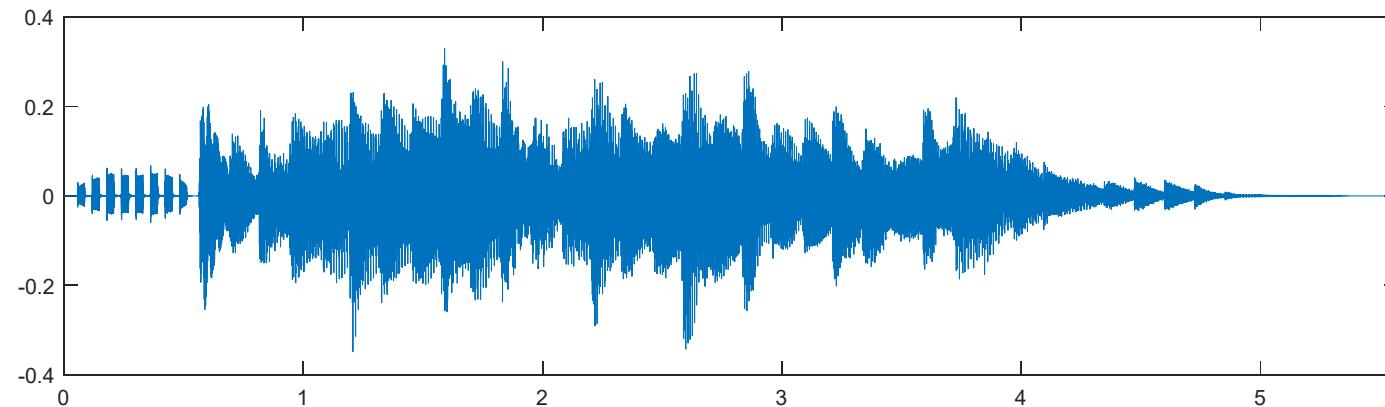
>> 如果是雙聲道則每秒要取 88200 點

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 (詳細方法請參考附錄二)

X = fft(x(:,1)); % 只做這一步無法得出正確的頻譜

```
X=X.';

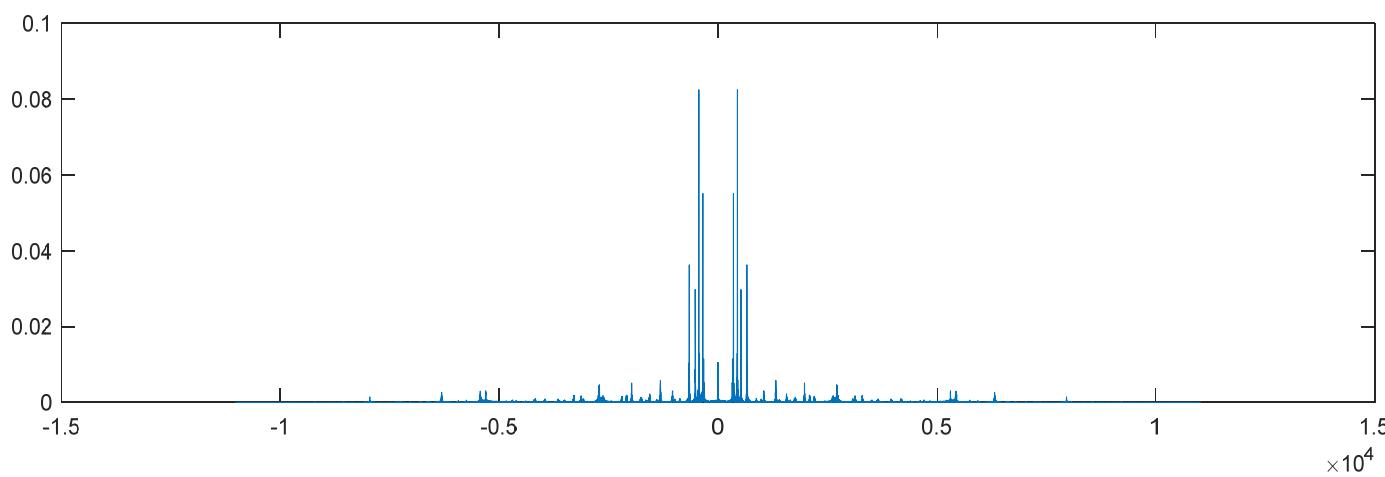
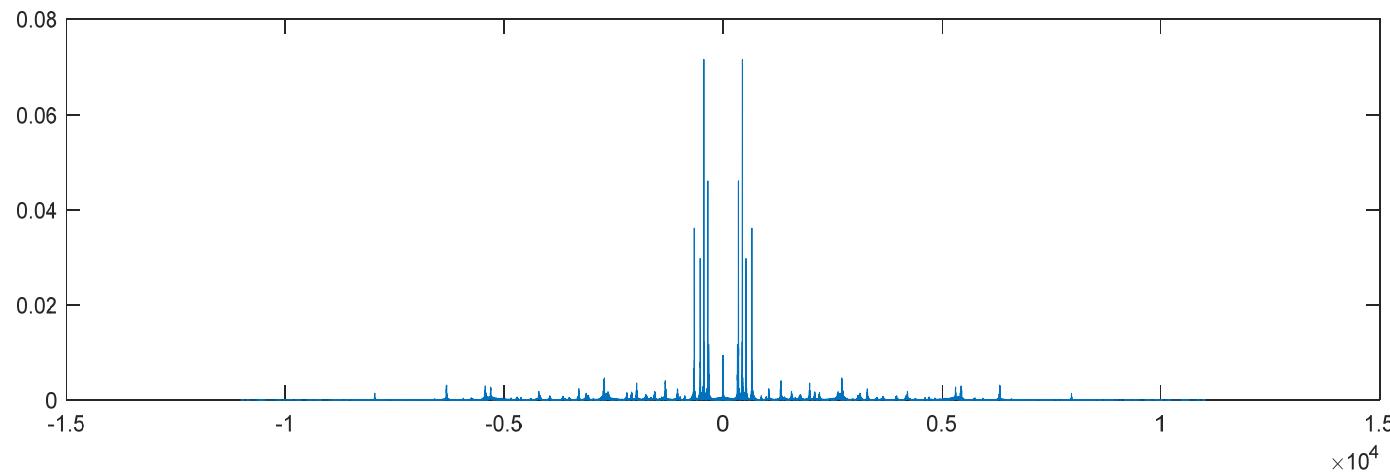
N=length(X); N1=round(N/2);

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

能量大部分集中於 1000 Hz 以下

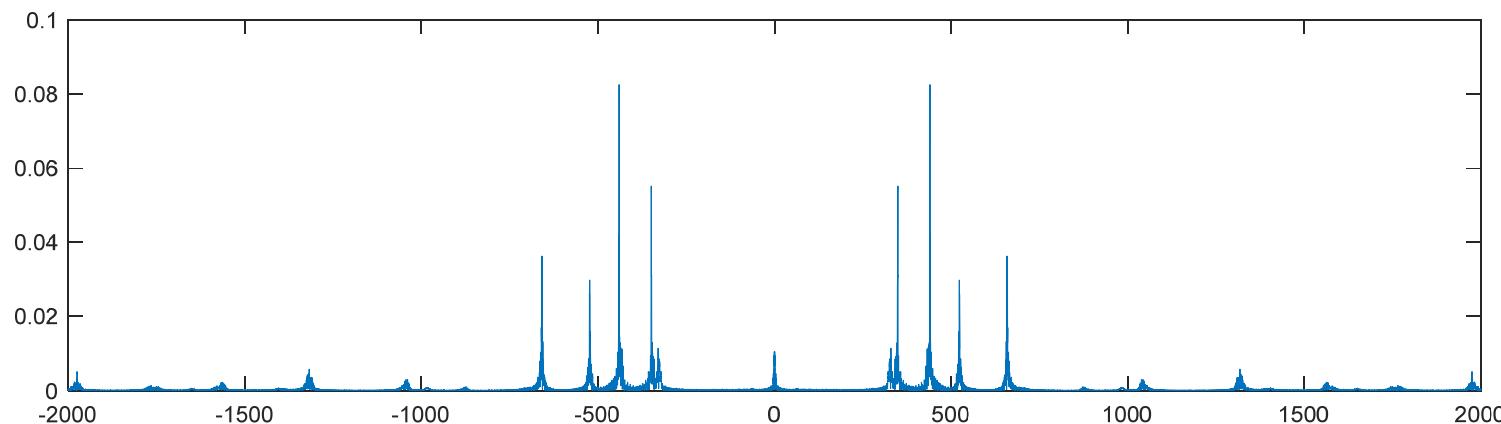
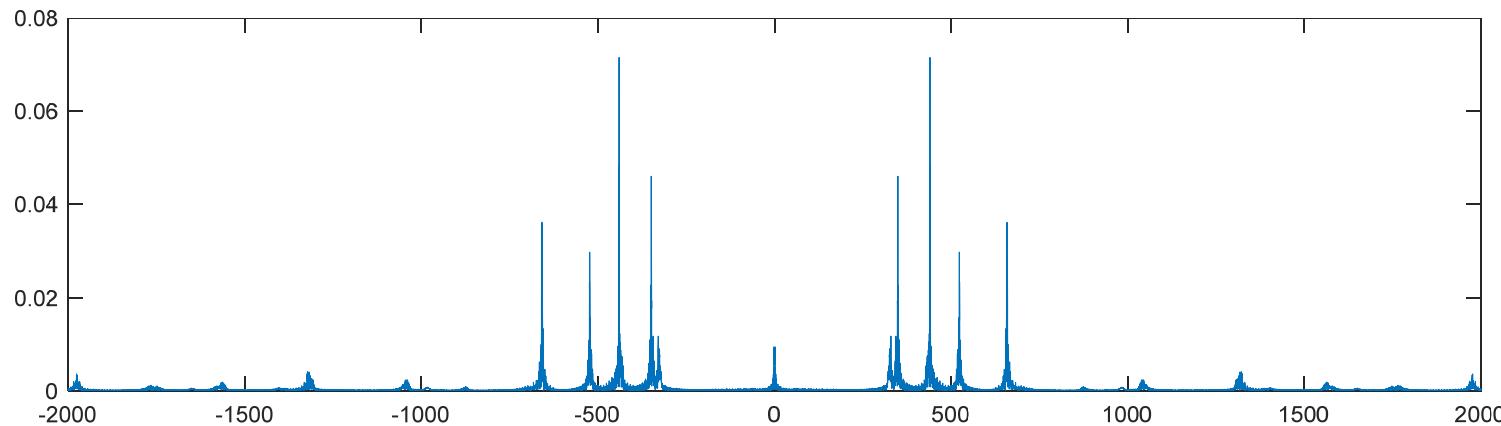


if $x[n]$ is real
then $X(f) = X^*(-f)$

211

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. 用 Matlab 製作 *.wav 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. 用 Matlab 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) :

```
sound(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('test.wav', audioarray, Fs) % 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

F：影像檔的處理

Image 檔讀取: `imread`

Image 檔顯示: `imshow`, `image`, `imagesc`

Image 檔製作: `imwrite`

基本概念：灰階影像在 Matlab 當中是一個**矩陣**

彩色影像在 Matlab 當中是三個**矩陣**，分別代表 Red,
Green, Blue

*.bmp: 沒有經過任何壓縮處理的圖檔

*.jpg: 有經過 JPEG 壓縮的圖檔

Video 檔讀取: `aviread`

範例一：(黑白影像)

217

```
im=double(imread('C:\Program Files\MATLAB\pic\Pepper.bmp'));
```

(注意，如果 Pepper.bmp 是個灰階圖，im 將是一個矩陣)

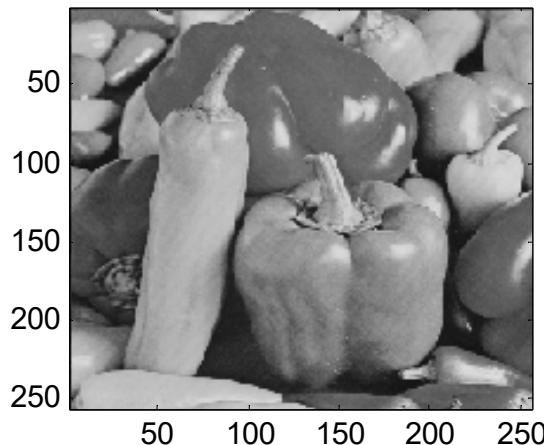
```
size(im)      (用 size 這個指令來看 im 這個矩陣的大小)
```

```
ans =
```

```
256 256
```

```
image(im);
```

```
colormap(gray(256))
```



範例二：(彩色影像)

```
im2=double(imread('C:\Program Files\MATLAB\pic\Pepper512c.bmp'));
```

```
size(im2)
```

```
ans =
```

```
512 512 3
```

(注意，由於這個圖檔是個彩色的，所以 im2 將由
三個矩陣複合而成)

```
imshow(im); or image(im/255);
```

注意：要對影像做運算時，要先變成 double 的格式

否則電腦會預設影像為 integer 的格式，在做浮點運算時會產生誤差

例如，若要對影像做 2D Discrete Fourier transform

```
im=imread('C:\Program Files\MATLAB\pic\Pepper.bmp');  
im=double(im);  
Imf=fft2(im);
```

附錄七 聲音檔和影像檔的處理 (by Python)

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
```

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
```

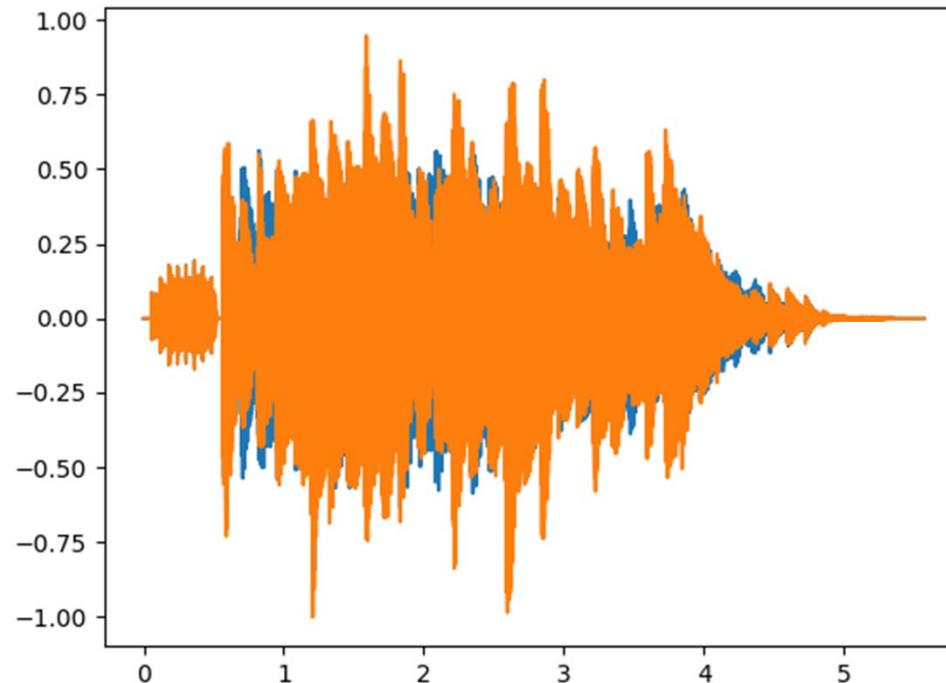
```
>>> num_frame, n_channel
```

```
(1150416, 2)
```

畫出音訊波形圖

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

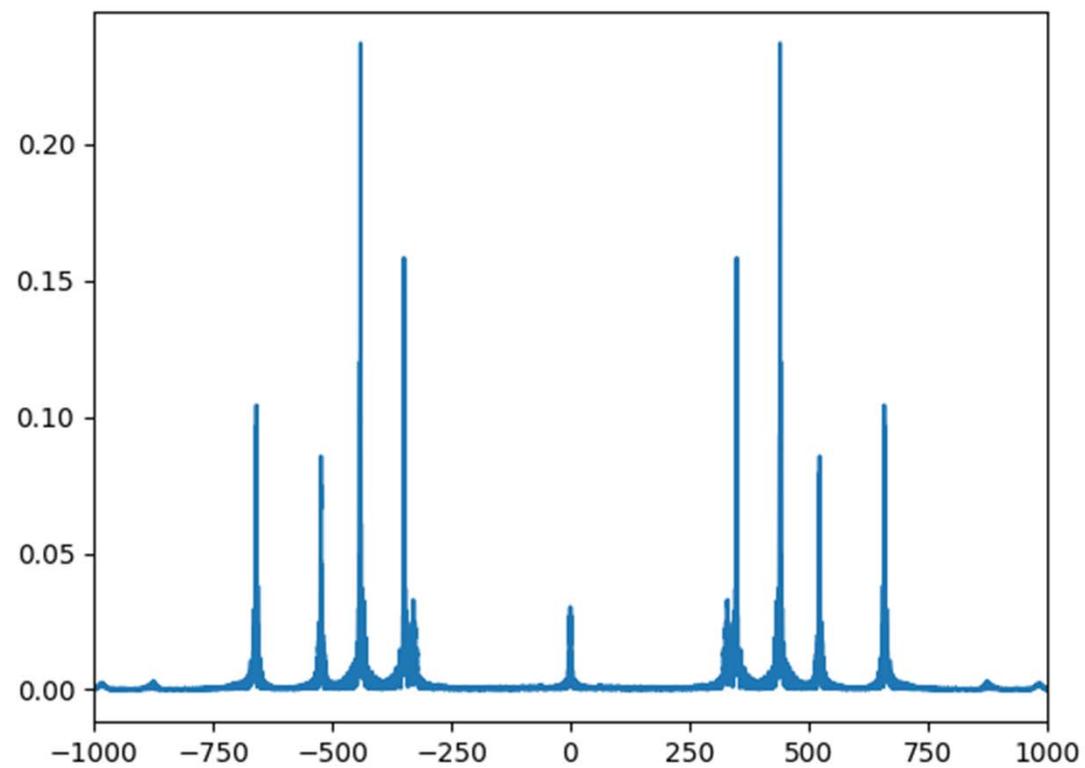
- time = np.arange(0, num_frame)*1/fs
- plt.plot(time, wave_data)
- plt.show()



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} ~ 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/>

F. 影像檔的處理

可以先安裝幾個模組

pip install numpy

pip install matplotlib

1. 讀取影像檔

```
import cv2  
image = cv2.imread('D:/Pic/peppers.bmp')
```

或

```
import matplotlib.pyplot as plt  
image = plt.imread('D:/Pic/peppers.bmp')
```

注意

(1) 寫入圖片若為彩色圖片，需要注意 `cv2.imread` 預設channels 順序為 BGR，

`image[:, :, 0] => B, image[:, :, 1] => G, image[:, :, 2] => R`

(2) 若使用 `plt.imread`，則 3 個 channels 的順序仍為 RGB

`image[:, :, 0] => R, image[:, :, 1] => G, image[:, :, 2] => B`

(3) 若讀檔讀不出來，有時要將路徑的 \ 改為 /

(4) `image.shape` 可以看出影像之大小

```
>>> image.shape
```

```
(512, 512, 3)
```

(5) 可讀 jpg, bmp, png 檔，但不能讀 gif 檔

2. 顯示影像

Case 1: 圖的值格式為 int

以下的指令要配合使用 (彩色，灰階皆可)

```
cv2.imshow('test', image) # 以 test 為顯示的圖的名稱  
cv2.waitKey(0)  
cv2.destroyAllWindows()
```

亦可用以下之指令

```
import matplotlib.pyplot as plt  
plt.imshow(image)  
plt.show()
```

若一開始用 cv.imread 讀圖但要用 plt.imshow 來顯示彩色圖，要先將 BGR 的順序轉回 RGB，將第二行改為

```
plt.imshow(image[:, :, [2, 1, 0]])
```

Case 2: 圖的值格式為 double (非整數)

此時，無論用 cv2.imshow 或是 plt.imshow，皆要先除以 255，再將圖顯示出來

Example 1:

```
image = cv2.imread('D:/Pic/peppers.bmp')
Image1 = image*0.5 + 127.5 # lighten the image
cv2.imshow('test', image) # int 不用除255
cv2.waitKey(0)
cv2.destroyAllWindows()
cv2.imshow('test', image1/255) # 非整數要除255
cv2.waitKey(0)
cv2.destroyAllWindows()
```



Example 2:

```
import matplotlib.pyplot as plt  
image = plt.imread('D:/Pic/peppers.bmp')  
image1 = image*0.5 + 127.5 # lighten the image  
plt.imshow(image) # int 不用除255  
plt.show()  
plt.imshow(image1/255) # 非整數要除255  
plt.show()
```



3. 寫入圖片檔

`cv2.imwrite('D:/Pic/jpg', image)`

或

`plt.imsave('D:/Pic/jpg', image)`

注意

(1) 寫入圖片若為彩色圖片，在使用 `cv2.imwrite` 時需要注意

`image[:, :, 0] => B, image[:, :, 1] => G, image[:, :, 2] => R`

若用 `plt.imsave` 則

`image[:, :, 0] => R, image[:, :, 1] => G, image[:, :, 2] => B`

(2) 若用 `cv2.imwrite('D:\Pic\jpg', image)` 可能無法存檔，要將 \ 改為 /

(3) 若是使用 `plt.imshow` 和 `plot.show()` 來顯示，可以用右下角的“save the figure”來存檔

VI. Brief Introduction for Acoustics (聲學)

[參考資料]

- 王小川，“語音訊號處理”，第三版，全華出版，台北，民國98年。
- T. F. Quatieri, *Discrete-Time Speech Signal Processing: Principle and Practice*, Pearson Education Taiwan, Taipei, 2005.
- L. R. Rabiner and R. W. Schafer, *Digital Processing of Speech Signals*, Prentice-Hall, 1978.
- P. Filippi, *Acoustics : Basic Physics, Theory, and Methods*, Academic Press, San Diego, 1999.

◎ 6-A 聲音的相關常識

235

人耳可以辨識頻率：20Hz ~ 20000Hz

說話：150~2000Hz

電話系統頻域：小於 4000Hz

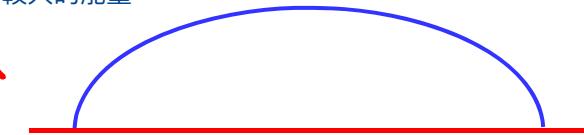
電腦音效卡取樣頻率：44100Hz (最新技術可達192K)

(一般用 22050Hz, 11025Hz 即可)

> 20000Hz: 超音波 (ultrasound)

< 20Hz: 次聲波 (infrasound)

如果聲音信號是 f_0 ，在 f_0 倍數的地方都會有比較大的能量



music file

$$250 \text{ sec} \times 22050 \times 2 \times 8 / 8 \text{ bytes} \\ = 11 \text{ M}$$

MP3: 3M~4M

30%

如果音強夠強的話，即使 < 20 Hz 也有可能聽得到 (p.238)

- 聲波就是空氣的震動，所以每秒鐘揮一次手其實也會產生 1Hz 的聲波，只是我們聽不到而已，但是假如有人可以一秒揮手 20 下，我們就有機會聽得到 (ex: 蚊子震動翅膀每秒鐘可以震動六十次，所以我們聽得到)

波長較長 -> 傳播距離較遠，但容易散射

(頻率比較低)

ex: 海底沒辦法有很多基地台，所以為了傳播較遠，會用較低的頻率通訊

波長較短 -> 衰減較快，但傳播方向較接近直線

$$\text{正比於 } e^{-kz} \quad k = \frac{2\pi}{\lambda}$$

k: 波速

假設一個 mp3 檔案長度為 250 sec，通常會是雙聲道來產生立體聲的效果，聲音檔通常每個 sample 都會用八個 bit 來代表 (-127~+128)

>> mp3 是一種壓縮格式，所以這個例子中把 11 M 壓縮到 3~4 M

>> 壓縮三倍算少 (關於壓縮後面會講)

>> ex: 影像壓縮 jpeg 可以壓縮到 1/30

>> 直觀的理由：音訊是 1 dimension，所以可以用來做預測的東西變少

- 一般聲音檔格式：
 - (1) 取樣頻率 22050Hz
 - (2) 單聲道或雙聲道
 - (3) 每筆資料用8個bit來表示
- 電腦中沒有經過任何壓縮的聲音檔： *.wav

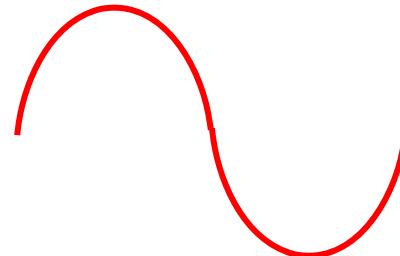
Q: What is the data size of a song without compression?

- 數位電話取樣頻率：8000Hz

聲音在空氣中傳播速度：每秒 340 公尺 (15°C 時)

所以，人類對3000Hz 左右頻率的聲音最敏感 人耳朵可以看成一個接收器（共振箱）

(一般人，耳翼到鼓膜之間的距離：2.7公分)



$\lambda/4$

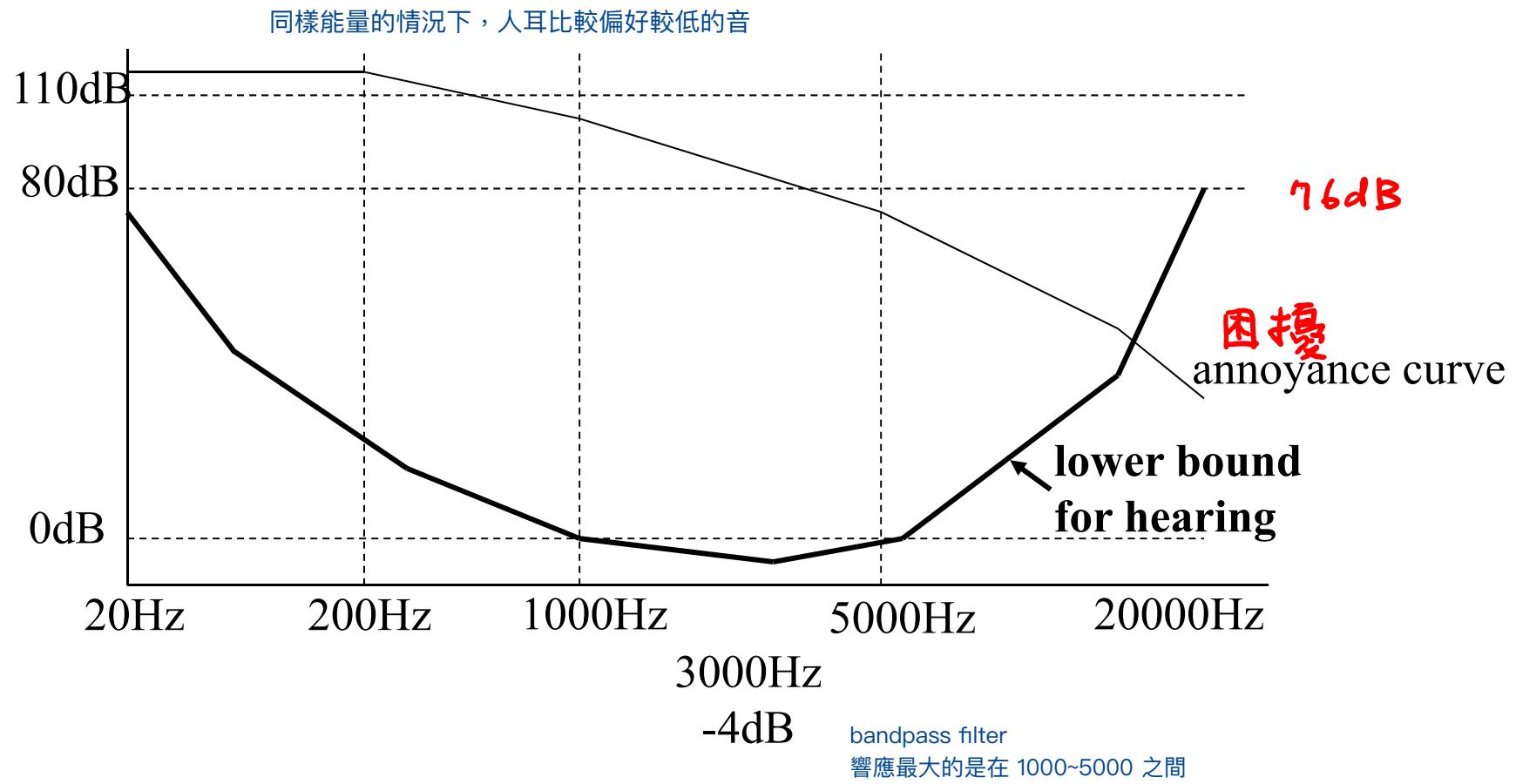
$$\lambda = 0.108 \text{ m}$$

如果 $1/4$ 波長 = 2.7 cm，波長 = 10.8 cm = 0.108 m

附：(1) 每增加 1°C ，聲音的速度增加 0.6 m/sec

(2) 聲音在水中的傳播速度是 1500 m/sec

在鋁棒中的傳播速度是 5000 m/sec



因為嬰兒沒辦法發出那麼大的聲音，所以音頻比較高，就可以不用那麼大的 dB 就能讓人聽得見

- dB: 分貝 $10\log_{10}(P/C)$ ，其中 P 為音強(正比於振幅的平方)；C 為 0dB 時的音強

每增加 10dB，音強增加 10 倍，振幅增加 $10^{0.5}$ 倍；

每增加 3dB，音強增加 2 倍，振幅增加 $2^{0.5}$ 倍；

所幸，內耳的振動不會正比於聲壓

- 人對於頻率的分辨能力，是由頻率的「比」決定

所以即使不同的人唱 Do 發出來的頻率也會不同，但我們仍然可以辨識出他在唱哪一首歌，因為每半音差 $2^{\{1/12\}}$ 是固定的

對人類而言，300Hz 和 400 Hz 之間的差別，與 3000Hz 和 4000 Hz 之間的差別是相同的



一般而言語音訊號處理的難度會遠大於音樂訊號
(因為一個音階裡只有 12 個半音，語音訊號以中文來說有 1095 個以上可以發出來的音，英文甚至有 5000 多個)

◎ 6-B Music Signal

240

電子琴 Do 的頻率：低音 Do: 131.32 Hz

不同樂器的 Do 的頻率也會不同
(越大的樂器 >> 共振箱越大 >> 頻率越低)

中音 Do: 261.63 Hz 每增加八度音頻率變成兩倍

高音 Do: 523.26 Hz

更高音 Do: 1046.52 Hz,

音樂每增加八度音，頻率變為 2 倍

每一音階有12個半音

If Do: 200 Hz
then Re $200 \times 2^{\frac{2}{12}}$ Hz
Mi $200 \times 2^{\frac{4}{12}}$ Hz

增加一個半音，頻率增加 $2^{1/12}$ 倍 (1.0595 倍)

	Do	升Do	Re	升Re	Mi	Fa	升Fa	So	升So	La	升La	Si
Hz	262	277	294	311	330	349	370	392	415	440	466	494

$$261.63 \times 2^{\frac{2}{12}}$$

$$261.63 \times 2^{\frac{4}{12}}$$

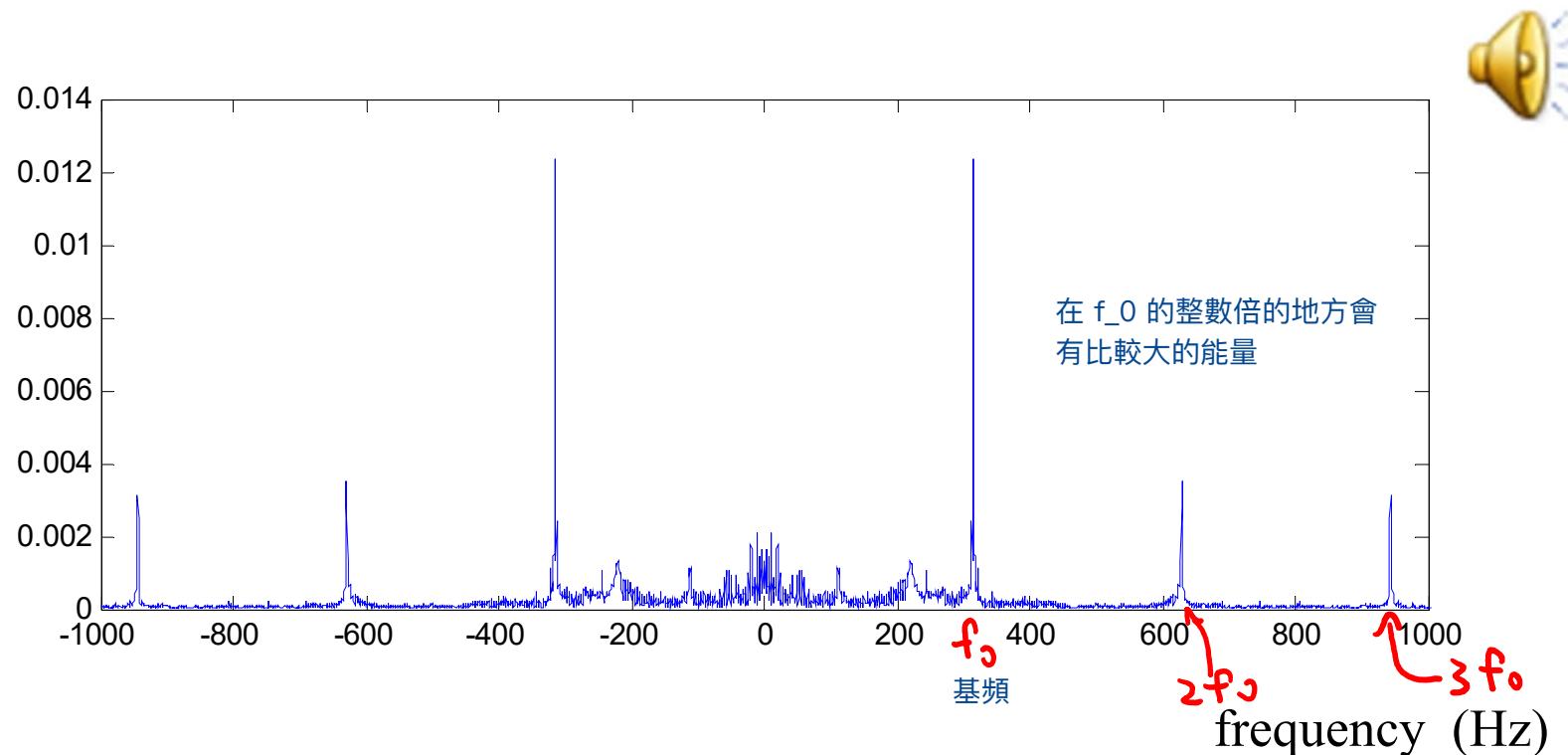
音樂通常會出現「和弦」(chord)的現象

因為音樂是由共振的現象產生的 (ex: 弦樂器：弦的共振：管樂器：共振箱的共振)

>> 任何用共振方式產生的聲音都會有倍頻的現象

除了基頻 f_0 Hz 之外，也會出現 $2f_0$ Hz, $3f_0$ Hz, $4f_0$ Hz, 的頻率

Mi:330



為什麼會產生和弦？

以共振的觀點：lambda：波長

假設今天有一個
弦長度為 L

$$L = \frac{1}{2}\lambda, \lambda = 2L$$

$$f = \frac{340}{2L}$$

$$L = \lambda$$

$$f = \frac{340}{L} = 2f_0$$

$$L = \frac{3}{2}\lambda, \lambda = \frac{2}{3}L$$

$$f = \frac{3}{2} \frac{340}{L} = 3f_0$$

$$L = 2\lambda, \lambda = \frac{L}{2}, f = 2 \frac{340}{L} = 4f_0$$

⋮

$$L = \frac{n}{2}\lambda, \lambda = \frac{2L}{n}, f = \frac{n}{2} \frac{340}{L} = nf_0$$

只要 L 是 lambda 的 $\frac{n}{2}$ 倍，就會產生共振
=> 當 n 越大，代表波長越小，產生的頻率就越高

聲音信號是一個 periodic signal，但是不一定是 sinusoid

cosine / sine function

如果要產生 $Do = 261.63 \text{ Hz}$

>> 可以設計 L 的長度： $340 / 261.63 = 1.29 \text{ (m)} = 2L$

>> L 約為 $129 \text{ (cm)} / 2 = 65 \text{ cm}$

- 要發出越高頻率 >> 弦越短

$$\text{set } f_0 = \frac{340}{2L}$$

假設固定

◎ 6-C 語音處理的工作

- (1) 語音編碼 (Speech Coding) 與壓縮有關
- (2) 語音合成 (Speech Synthesis)
- (3) 語音增強 (Speech Enhancement) 把背景的噪音去除，讓說話的聲音增強

前三項目前基本上已經很成功

- (4) 語音辨認 (Speech Recognition)

聲音辨識的基本單位：
音素 → 音節 → 詞 → 句 → 整段話
目前已很高的辨識率

- (5) 說話人辨認 (Speaker Recognition)
- (6) 其他：語意，語言，情緒

人類在發出聲音的過程會經過非直線的路徑，所以會有 multipath 的問題
>> fourier transform : 語音辨識難有很剛的辨識率
>> cepstrum : 解決 multipath 問題，語音辨識率大幅提升（超過 90%）
>> 近年機器學習 (SVM, DL...) : 語音辨識提升到 99% 以上

◎ 6-D 語音的辨認

子音 + 母音 = 音節

音素 → 音節 → 詞 → 句 → 整段話

音素：相當於一個音標

(1) Spectrum Analysis

Time-Frequency Analysis

(2) Cepstrum

(3) Correlation for Words

◎ 6-E 子音和母音

ㄅ ㄆ ㄇ ㄈ ㄉ ㄋ ㄎ ㄊ ㄃ ㄕ ㄔ ㄗ ㄕ ㄔ
ㄚ ㄛ ㄜ ㄝ ㄢ ㄤ ㄦ ㄩ ㄱ ㄲ ㄳ ㄵ ㄶ ㄷ ㄸ

母音：ㄚ ㄛ ㄜ ㄝ ㄢ ㄤ ㄦ ㄩ ㄱ ㄲ ㄳ ㄵ ㄶ ㄷ ㄸ

單母音：a, e, i, o, u ㄚ ㄛ ㄜ ㄝ ㄢ ㄤ ㄦ ㄩ ㄱ ㄲ ㄳ ㄵ ㄶ ㄷ ㄸ

雙母音：ㄞ ㄠ ㄯ ㄵ

母音 + 濁音：ㄻ ㄽ ㄹ ㄷ

子音：ㄅ ㄆ ㄇ ㄈ ㄉ ㄋ ㄎ ㄊ ㄃ ㄕ ㄔ ㄗ ㄕ ㄔ

	ㄅ	ㄆ	ㄇ	ㄈ	ㄉ	ㄊ	ㄋ	ㄎ	ㄏ	ㄔ	ㄕ	ㄕ	ㄔ	
漢語拼音	b	p	m	f	d	t	n	l	g	k	h	j	q	x
通用拼音	b	p	m	f	d	t	n	l	g	k	h	j	c	s

	ㄓ	ㄔ	ㄕ	ㄖ	ㄗ	ㄘ	ㄙ	ㄚ	ㄛ	ㄜ	ㄝ	ㄢ	ㄤ	ㄩ
漢語拼音	zh	ch	sh	r	z	c	s	a	o	e	e	ai	ei	ao
通用拼音	jh	ch	sh	r	z	c	s	a	o	e	e	ai	ei	ao

	ㄡ	ㄞ	ㄣ	ㄤ	ㄥ	ㄦ	ㄧ	ㄨ	ㄩ
漢語拼音	ou	an	en	ang	eng	er	i, y	u, w	yu, iu
通用拼音	ou	an	en	ang	eng	er	i, y	u, w	yu, iu

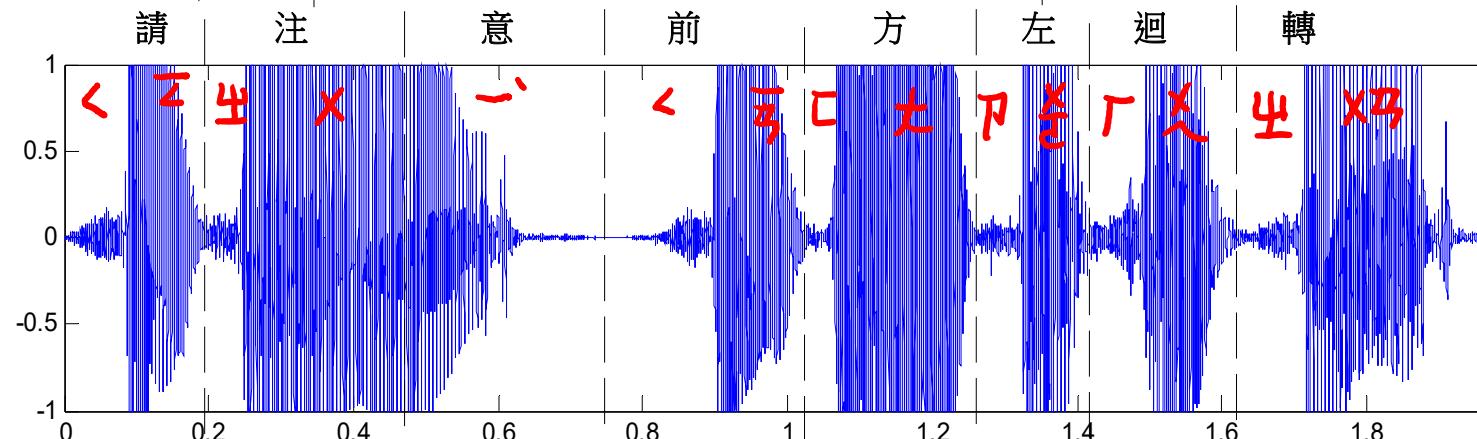
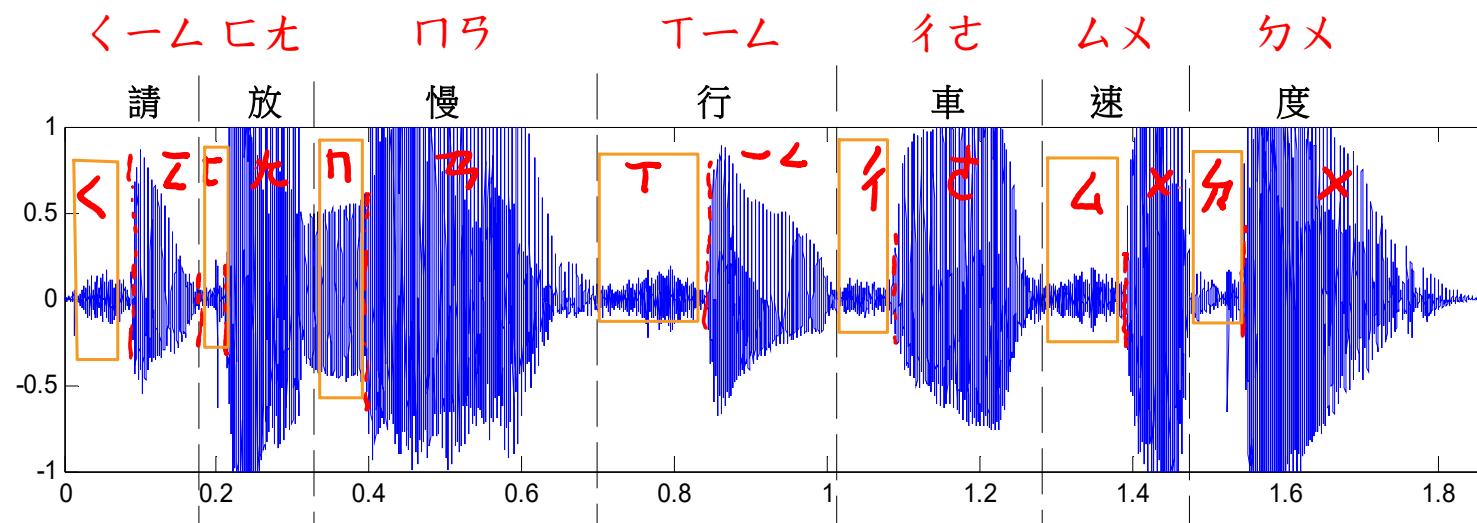
母音：依唇型而定

子音：在口腔，鼻腔中某些部位將氣流暫時堵住後放開

子音的能量小，頻率偏高，時間較短，出現在母音前

母音的能量大，頻率偏低，時間較長，出現在子音後或獨立出現

每個能量比較大的地方對應到一個字
 每個能量比較大的地方之中有能量比較小的地方 >> 對應到子音



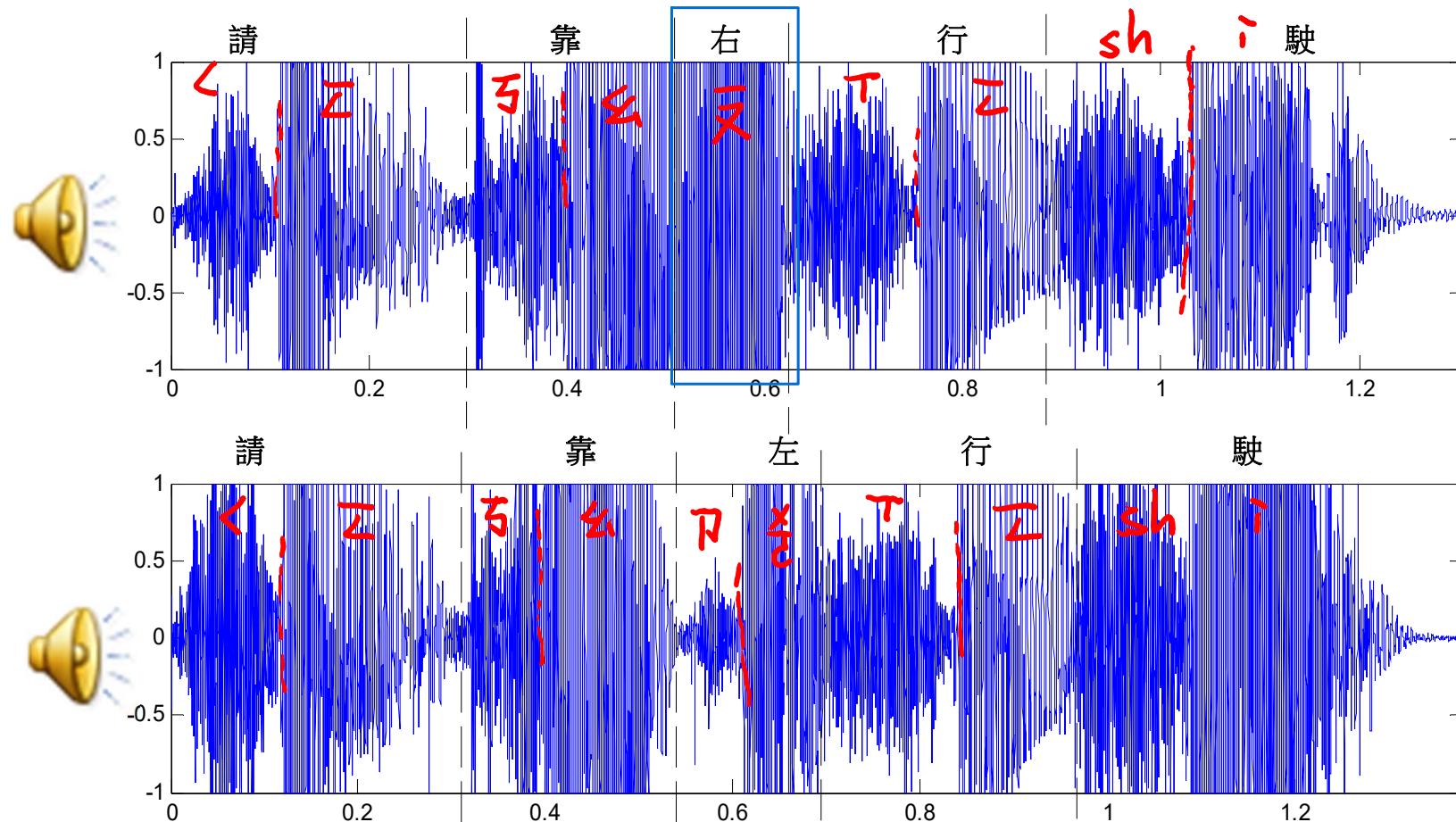
有八個字但只有七個片段
 >> 因為「意」沒有子音，所以沒有前面能量比較小的地方

我們在講子音的時候還不是正確的因，所以給他比較小的音，等到唇形等調對以後發母音聲音才比較大

雖然 尸 只有子音，
但是以拼音來看
sh：子音
i：母音

249

一又都是母音，沒有子音 >> 沒有前面比較小的片段



- 根據能量的變化判斷音節（但沒有子音的部分就要靠其他方法判斷）

- 所有的 LTI 都可以用 convolution 表示

250

發音模型 (線性非時變近似)

$$x[n] = e_p[n] * g[n] * h[n] * r[n], \quad * \text{ means the convolution}$$

音訊的 z transform $X(z) = E_p(z) G(z) H(z) R(z)$ 相當於 $e_p[n], g[n], h[n], r[n]$ 的 z transform 相乘

在 time domain 做 convolution 相當於在 frequency domain 做相乘

$r[n]$: 嘴唇模型 , $h[n]$: 口腔模型 , $g[n]$: 聲帶模型

$e_p[n]$: 輸入(假設為週期脈衝)

音量和 $e_p[n], g[n]$ 有關

頻率和 $g[n]$ 有關

子音和 $h[n], r[n]$ 有關

母音和 $r[n]$ 有關

$de \rightarrow te \rightarrow tea \rightarrow tca$

- 分析一個聲音信號的頻譜：
用 **Windowed Fourier Transform**
或稱作 **Short-Time Fourier Transform**

- Fourier transform

$$G(f) = \int_{-\infty}^{\infty} g(t) e^{-j2\pi f t} dt$$

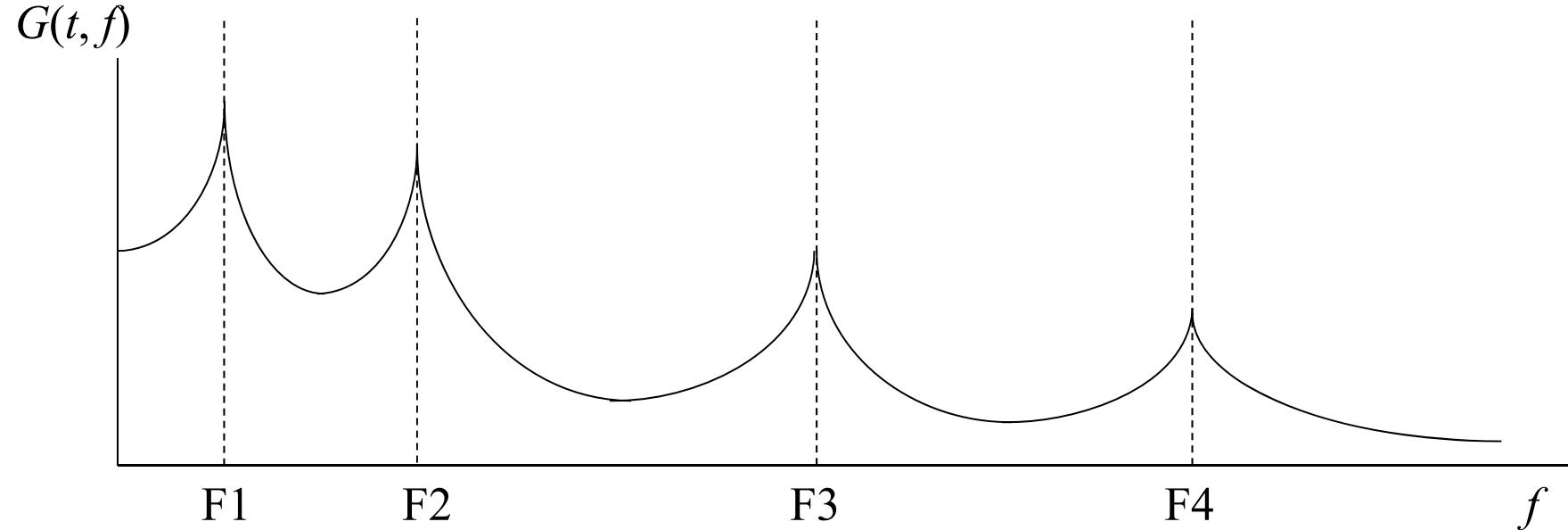
Windowed Fourier transform

以 t_0 為中心取一段做 short time fourier transform

$$G(f) = \int_{t_0-B}^{t_0+B} g(t) e^{-j2\pi f t} dt \quad \text{強調 } t = t_0 \text{ 附近的區域}$$

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

典型的聲音頻譜(不考慮倍頻)：
語音信號不會像音樂信號那麼乾淨，雖然也會有倍頻現象，但是
基頻的整數倍的地方之外不會完全為零
>> 會干擾到 f_1, f_2, f_3 的找尋



頻譜上，大部分的地方都不等於0。

出現幾個 peaks 值

可以依據 peaks 的位置來辨別母音

母音 peaks 處的頻率 (Hz) (不考慮倍頻)：

	男聲			女聲		
	F1	F2	F3	F1	F2	F3
ㄚ	900	1200	2900	1100	1350	3100
ㄛ	560	800	3000	730	1100	3200
ㄔ	560	1090	3000	790	1250	3100
ㄝ	500	2100	3100	600	2400	3300
ㄧ	310	2300	3300	360	3000	3500
ㄨ	370	540	3400	460	820	3700
ㄩ	300	2100	3400	350	2600	3200
ㄦ	580	1500	3200	760	1700	3200

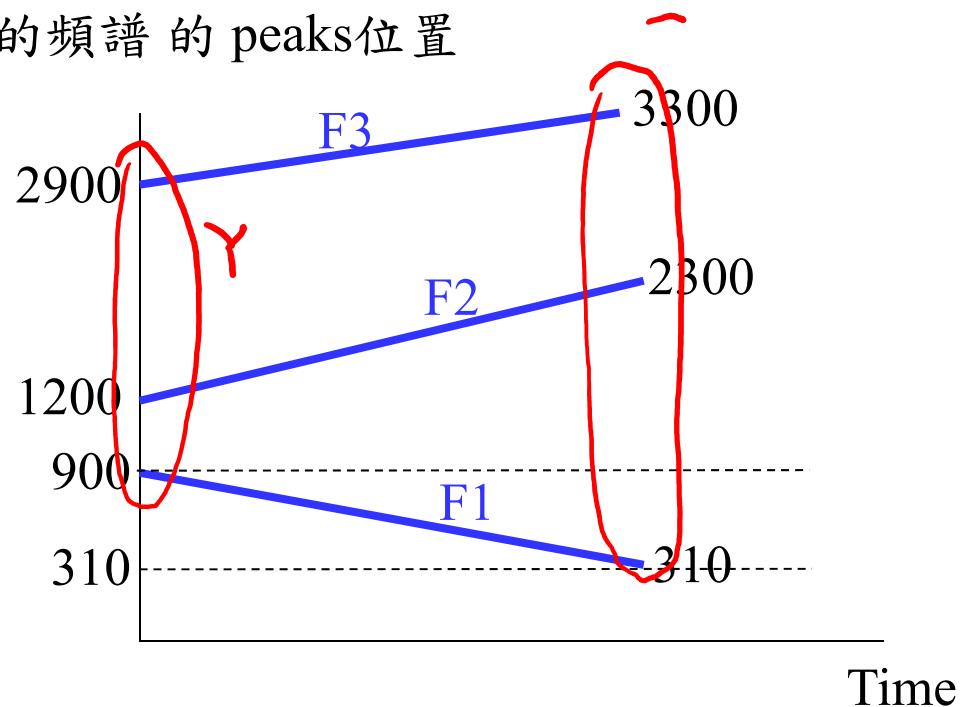
原則上：(1) 嘴唇的大小，決定F1
 (2) 舌面的高低，決定 F2 – F1

[Ref] 王小川，“語音訊號處理”，第三版，全華出版，台北，民國98年

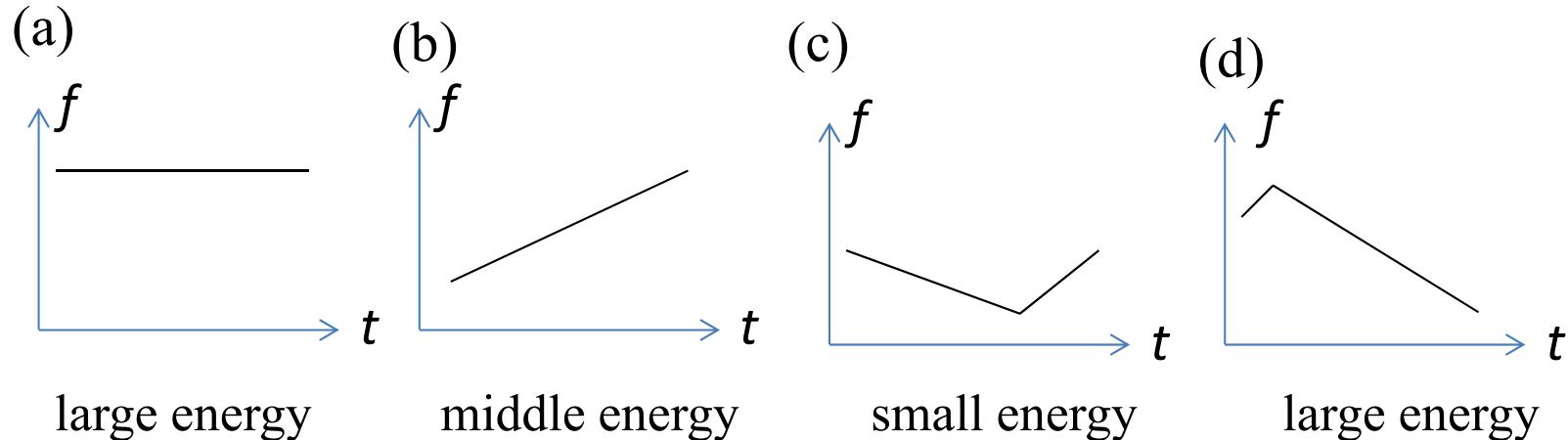
- 雙母音：
ㄞ (ai) , ㄟ (ei) , ㄠ (ao) , ㄡ (ou)

頻譜隨時間而改變，一開始像第一個母音，後變得像另一個母音

ㄞ 的頻譜的 peaks 位置



6-F Tone Analysis



Typical relations between time and the instantaneous frequencies for (a) the 1st tone, (b) the 2nd tone, (c) the 3rd tone, and (d) the 4th tone in Chinese.

X. X. Chen, C. N. Cai, P. Guo, and Y. Sun, “A hidden Markov model applied to Chinese four-tone recognition,” *ICASSP*, vol. 12, pp. 797-800, 1987.

◎ 6-G 語意學的角色

以「語意學」或「機率」來補足語音辨識的不足

例如：經過判定，一個聲音可能是

ㄅ一 ㄔㄎ
ㄅ一 ㄔㄎ

ㄅ一 ㄔㄎ
ㄅ一 ㄔㄎ

這個聲音是「必然」的機率比較大。

ㄅㄛ ㄅㄛ
ㄅㄛ ㄅㄛ

可能是「伯伯」，也可能是「婆婆」，看上下文

儲存詞庫

- 當前主流的語音辨識技術：

Mel-Frequency Cepstrum + Tone Analysis + 語意分析 + Machine Learning

附錄八：線性代數觀念補充

- (1) \mathbf{x} 和 \mathbf{y} 兩個向量的內積可表示成 $\langle \mathbf{x} | \mathbf{y} \rangle$
- (2) 兩個互相正交(orthogonal)或垂直(perpendicular)的向量，其內積為0。可表示成： $\langle \mathbf{x} | \mathbf{y} \rangle = 0$ 或 $\langle \mathbf{x}, \mathbf{y} \rangle = 0$
- (3) 令 S 為內積空間 V 的一組正交集合(set)且由非零向量構成，

$$\text{其中 } \mathbf{x} = \sum_{\mathbf{y} \in S} a_y \mathbf{y}, \quad a_y = \frac{\langle \mathbf{x} | \mathbf{y} \rangle}{\langle \mathbf{y} | \mathbf{y} \rangle}$$

如果 S 是由一組正規集合(orthonormal set)構成，那麼 $a_y = \langle \mathbf{x} | \mathbf{y} \rangle$

(4) Gram-Schmidt algorithm: 對於內積空間V的任意一組基底 $\langle \mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n \rangle$ ，我們可以透過這演算法找到一組正交基底 $\langle \mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_n \rangle$

$$\mathbf{y}_j = \mathbf{x}_j - \sum_{i=1}^{j-1} \frac{\langle \mathbf{x}_j | \mathbf{y}_i \rangle}{\langle \mathbf{y}_i | \mathbf{y}_i \rangle} \mathbf{y}_i \quad \text{for each } j = 2, \dots, n$$

幾何意義: 把 \mathbf{x}_j 在 $\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_{j-1}$ 上面的分向量全都從向量 \mathbf{x}_j 身上扣掉之後，剩下的向量 \mathbf{y}_j 自然就會跟 $\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_{j-1}$ 垂直。

(5) Solving $\mathbf{Ax} = \mathbf{b}$ but $\text{size}(\mathbf{A}) = m \times n$ and $\mathbf{b} \in F^m$, $m > n$

Interpolation Theorem (插值定理)

1. For any inner-product function of F^m , there exists a vector \mathbf{z} that minimizes

$$\| \mathbf{Az} - \mathbf{b} \| \quad \text{where } \mathbf{z} \in F^n$$

2. If $\text{rank}(\mathbf{A}) = n$, then $\mathbf{z} = (\mathbf{A}^H \mathbf{A})^{-1} \mathbf{A}^H \mathbf{b}$ is the unique minimizer of $\| \mathbf{Az} - \mathbf{b} \|$

附錄九：PCA and SVD

PCA (principal component analysis) 是資料分析和影像處理當中常用到的數學方法，用來分析資料的「主要成分」或是影像中物體的「主軸」。

它其實和各位同學在高中和大一線代所學的回歸線 (regressive line) 很類似。回歸線是用一條一維 (one-dimensional) 的直線來近似二維 (two-dimensional) 的資料，而 PCA 則是用 M -dimensional data 來近似 N -dimensional data，其中 M 小於等於 N

在講解 PCA 之前，先介紹什麼是 SVD (singular value decomposition)

我們在大一的時候，都已經學到該如何對於 $N \times N$ 的矩陣做 eigenvector-eigenvalue decomposition

那麼.....

當一個矩陣的 size 為 $M \times N$ ，且 M 和 N 不相等時，我們該如何對它來做 eigenvector-eigenvalue decomposition?

SVD 的流程：

假設 \mathbf{A} 是一個 $M \times N$ 的矩陣。

(Step 1) 計算

$$\mathbf{B} = \mathbf{A}^H \mathbf{A} \quad \mathbf{C} = \mathbf{A} \mathbf{A}^H$$

注意， \mathbf{B} 是 $N \times N$ 的矩陣，而 \mathbf{C} 是 $M \times M$ 的矩陣。上標 H 代表 Hermitian matrix，相當於做共軛轉置。

(Step 2) 接著，對 \mathbf{B} 和 \mathbf{C} 做 eigenvector-eigenvalue decomposition

$$\mathbf{B} = \mathbf{V} \mathbf{D} \mathbf{V}^{-1} \quad \mathbf{C} = \mathbf{U} \Lambda \mathbf{U}^{-1}$$

其中 \mathbf{V} 的每一個 column 是 \mathbf{B} 的 eigenvector (with normalization)， \mathbf{U} 的每一個 column 是 \mathbf{C} 的 eigenvector (with normalization)， Λ 和 \mathbf{D} 都是對角矩陣， Λ 和 \mathbf{D} 對角線上的 entries 是 \mathbf{B} 和 \mathbf{C} 的 eigenvalues。並假設 eigenvectors 根據 eigenvalues 的大小排序 (由大到小)

Note: 值得注意的是，由於 $\mathbf{B} = \mathbf{B}^H$ 且 $\mathbf{C} = \mathbf{C}^H$ ，所以 \mathbf{B} 和 \mathbf{C} 的 eigenvectors 皆各自形成一個 orthogonal set。經過適當的 normalization 使得 \mathbf{U} 和 \mathbf{V} 的 column 自己和自己的內積為 1 之後， $\mathbf{U}^{-1} = \mathbf{U}^H$ 和 $\mathbf{V}^{-1} = \mathbf{V}^H$ 將滿足。因此， \mathbf{B} 和 \mathbf{C} 可以表示成

$$\mathbf{B} = \mathbf{V} \mathbf{D} \mathbf{V}^H \quad \mathbf{C} = \mathbf{U} \Lambda \mathbf{U}^H$$

注意， \mathbf{V} 和 \mathbf{U} 是 unitary matrix

(Step 3) 計算

$$\mathbf{S}_1 = \mathbf{U}^H \mathbf{A} \mathbf{V}$$

\mathbf{S}_1 是一個 $M \times N$ 的矩陣，只有在 $\mathbf{S}_1[n, n]$ ($n = 1, 2, \dots, \min(M, N)$) 的地方不為 0

(Step 4) $\mathbf{S} = |\mathbf{S}_1|$ 取絕對值

若 $\mathbf{S}_1[n, n] < 0$ ，改變 \mathbf{U} 第 n 個 column 的正負號

即完成 SVD

Note: Since \mathbf{V} is bound to be real,

$$\mathbf{A} = \mathbf{U} \mathbf{S} \mathbf{V}^H$$

$$\mathbf{A} = \mathbf{U} \mathbf{S} \mathbf{V}^T$$

\mathbf{A} 也可以表示為

$$\mathbf{A} = \lambda_1 \mathbf{u}_1 \mathbf{v}_1^T + \lambda_2 \mathbf{u}_2 \mathbf{v}_2^T + \dots + \lambda_k \mathbf{u}_k \mathbf{v}_k^T$$

其中 $\lambda_n = \mathbf{S}[n, n]$, $k = \min(M, N)$

註：Matlab 有內建的 svd 指令可以計算 SVD

從 SVD 到 PCA (principal component analysis , 主成份分析)

$$\mathbf{A} = \lambda_1 \mathbf{u}_1 \mathbf{v}_1^T + \lambda_2 \mathbf{u}_2 \mathbf{v}_2^T + \dots + \lambda_k \mathbf{u}_k \mathbf{v}_k^T \quad k = \min(M, N)$$

若 $\lambda_1 \geq \lambda_2 \geq \lambda_3 \geq \dots \geq \lambda_k$

$\lambda_1 \mathbf{u}_1 \mathbf{v}_1^T$ 是 \mathbf{A} 矩陣的最主要的成份

$\lambda_2 \mathbf{u}_2 \mathbf{v}_2^T$ 是 \mathbf{A} 矩陣的第二主要的成份

⋮

⋮

$\lambda_k \mathbf{u}_k \mathbf{v}_k^T$ 是 \mathbf{A} 矩陣的最不重要的成份

若為了壓縮或是去除雜訊的考量，可以選擇 $h < k$ ，使得 \mathbf{A} 可以近似成

$$\mathbf{A} \approx \lambda_1 \mathbf{u}_1 \mathbf{v}_1^T + \lambda_2 \mathbf{u}_2 \mathbf{v}_2^T + \dots + \lambda_h \mathbf{u}_h \mathbf{v}_h^T$$

PCA 的流程

假設現在有 M 筆資料，每一筆資料 為 N dimension

$$\mathbf{g}_1 = [f_{1,1} \ f_{1,2}, \dots, f_{1,N}]$$

$$\mathbf{g}_2 = [f_{2,1} \ f_{2,2}, \dots, f_{2,N}]$$

⋮

$$\mathbf{g}_M = [f_{M,1} \ f_{M,2}, \dots, f_{M,N}]$$

(Step 1) 扣掉平均值，形成新的 data

$$\mathbf{d}_m = [e_{m,1} \ e_{m,2} \ \dots \ e_{m,N}] \quad m = 1, 2, \dots, M$$

其中 $e_{m,n} = f_{m,n} - \tilde{f}_n$, $\tilde{f}_n = \frac{1}{M} \sum_{m=1}^M f_{m,n}$

(Step 2) 形成 $M \times N$ 的矩陣 \mathbf{A}

\mathbf{A} 的第 m 個 row 為 d_m , $m = 1, 2, \dots, M$

(Step 3) 對 A 做 SVD 分解

$$\begin{aligned} \mathbf{A} &= \mathbf{U}\mathbf{S}\mathbf{V}^H \\ &= \lambda_1 \mathbf{u}_1 \mathbf{v}_1^T + \lambda_2 \mathbf{u}_2 \mathbf{v}_2^T + \dots + \lambda_k \mathbf{u}_k \mathbf{v}_k^T & k = \min(M, N) \\ \lambda_1 &\geq \lambda_2 \geq \lambda_3 \geq \dots \geq \lambda_k \end{aligned}$$

(Step 4) 將 A 近似成

$$\mathbf{A} \approx \lambda_1 \mathbf{u}_1 \mathbf{v}_1^T + \lambda_2 \mathbf{u}_2 \mathbf{v}_2^T + \dots + \lambda_h \mathbf{u}_h \mathbf{v}_h^T$$

則每一筆資料可以近似為

$$g_m \approx \lambda_1 u_1[m] \mathbf{v}_1^T + \lambda_2 u_2[m] \mathbf{v}_2^T + \dots + \lambda_h u_h[m] \mathbf{v}_h^T + [\tilde{f}_1 \quad \tilde{f}_2 \quad \dots \quad \tilde{f}_N]$$

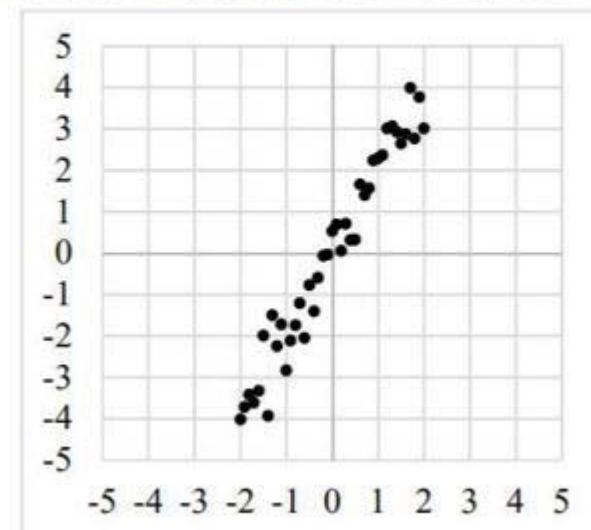
除了平均值 $[\tilde{f}_1 \quad \tilde{f}_2 \quad \dots \quad \tilde{f}_N]$ 之外

\mathbf{v}_1^T 是資料的最主要成分， \mathbf{v}_2^T 是資料的次主要成分，
 \mathbf{v}_3^T 是資料的第三主要成分，以此類推

Example of PCA

3. 在處理二維數據時，有種方法是將數據垂直投影到某一直線，並以該直線為數線，進而了解投影點所成一維數據的變異。下圖的一組二維數據，試問投影到哪一選項的直線，所得之一維投影數據的變異數會是最小？

- (1) $y = 2x$
- (2) $y = -2x$
- (3) $y = -x$
- (4) $y = \frac{x}{2}$
- (5) $y = -\frac{x}{2}$



From 2022 大考中心官網

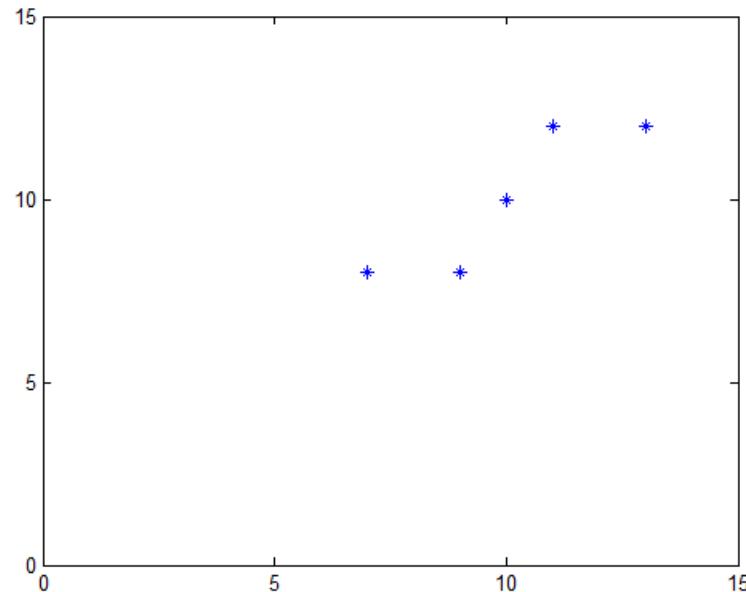
Example of PCA

假設在一個二維的空間中，有5個點
，座標分別是

$(7,8), (9,8), (10,10), (11,12), (13,12)$

$M = 5, N = 2$

試求這五個點的 PCA (即回歸線)



(Step 1) 將這五個座標點減去平均值 $(10, 10)$

$(-3, -2), (-1, -2), (0, 0), (1, 2), (3, 2)$

(Step 2) 形成 5×2 的 matrix

$$\mathbf{A} = \begin{bmatrix} -3 & -2 \\ -1 & -2 \\ 0 & 0 \\ 1 & 2 \\ 3 & 2 \end{bmatrix}$$

(Step 3) 計算 SVD

$$\mathbf{A} = \mathbf{U}\mathbf{S}\mathbf{V}^H$$

$$\mathbf{U} = \begin{bmatrix} -0.6116 & 0.3549 & 0 & 0.0393 & 0.7060 \\ -0.3549 & -0.6116 & 0 & 0.7060 & -0.0393 \\ 0 & 0 & 1 & 0 & 0 \\ 0.3549 & 0.6116 & 0 & 0.7060 & -0.0393 \\ 0.6116 & -0.3549 & 0 & 0.0393 & 0.7060 \end{bmatrix} \quad \mathbf{S} = \begin{bmatrix} 5.8416 & 0 \\ 0 & 1.3695 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$$

$$\mathbf{V} = \begin{bmatrix} 0.7497 & -0.6618 \\ 0.6618 & 0.7497 \end{bmatrix}$$

主成分

次要成分

$$\mathbf{A} = 5.8416 \begin{bmatrix} -0.6116 \\ -0.3549 \\ 0 \\ 0.3549 \\ 0.6116 \end{bmatrix} + 1.3695 \begin{bmatrix} 0.3549 \\ -0.6116 \\ 0 \\ 0.6116 \\ -0.3549 \end{bmatrix} + [0.7497 \quad 0.6618] \begin{bmatrix} 0.7497 & 0.6618 \end{bmatrix} + [-0.6618 \quad 0.7497]$$

(Step 4)

$$\mathbf{A} \cong 5.8416 \begin{bmatrix} -0.6116 \\ -0.3549 \\ 0 \\ 0.3549 \\ 0.6116 \end{bmatrix} [0.7497 \quad 0.6618]$$

$$\begin{bmatrix} 7 & 8 \\ 9 & 8 \\ 10 & 10 \\ 11 & 12 \\ 13 & 12 \end{bmatrix} \cong [10 \quad 10] + 5.8416 \begin{bmatrix} -0.6116 \\ -0.3549 \\ 0 \\ 0.3549 \\ 0.6116 \end{bmatrix} [0.7497 \quad 0.6618]$$

得到主成分 $[0.7497 \quad 0.6618]$

這五個座標點可以近似成

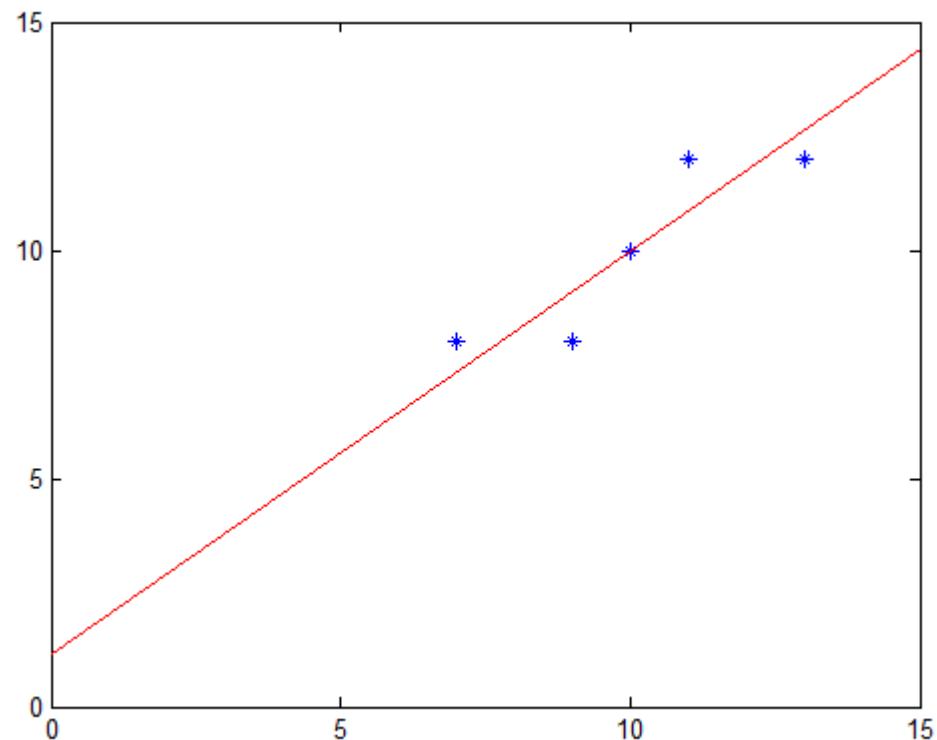
$$5.8416 \cdot u_m [0.7497 \quad 0.6618] + [10 \quad 10] \quad m = 1, 2, \dots, 5$$

$$u_1 = -0.6116, \quad u_2 = -0.3549, \quad u_3 = 0, \quad u_4 = 0.3549, \quad u_5 = 0.6116$$

回歸線

$$[10 \ 10] + c[0.7497 \ 0.6618]$$

$$c \in (-\infty, \infty)$$



Using the PCA method can obtain the best approximation result.

(Proof):

Without the loss of generalization, we discuss the problem in the 2D case (i.e., $N = 2$). Suppose that the location of the M points are

$$(x_1, y_1), (x_2, y_2), \dots, (x_M, y_M)$$

We want to find a line passing through the origin such that the projection of $(x_1, y_1), (x_2, y_2), \dots, (x_M, y_M)$ on the line has the maximal sum of the square norm. That is, to [find a unit vector](#)

$$\mathbf{e} = (e_1, e_2) \quad \text{where} \quad \|\mathbf{e}\| = 1 \quad (\text{The line passing through the origin is } \alpha\mathbf{e}.) \quad (1)$$

such that

$$\|\langle (x_1, y_1), \mathbf{e} \rangle \mathbf{e}\|^2 + \|\langle (x_2, y_2), \mathbf{e} \rangle \mathbf{e}\|^2 + \dots + \|\langle (x_M, y_M), \mathbf{e} \rangle \mathbf{e}\|^2 \quad (2)$$

[is maximal](#). Note that

$$\begin{aligned} & \|\langle (x_1, y_1), \mathbf{e} \rangle \mathbf{e}\|^2 + \|\langle (x_2, y_2), \mathbf{e} \rangle \mathbf{e}\|^2 + \dots + \|\langle (x_M, y_M), \mathbf{e} \rangle \mathbf{e}\|^2 \\ &= (\langle (x_1, y_1), \mathbf{e} \rangle)^2 + (\langle (x_2, y_2), \mathbf{e} \rangle)^2 + \dots + (\langle (x_M, y_M), \mathbf{e} \rangle)^2 \end{aligned} \quad (3)$$

Suppose that for the matrix

$$\mathbf{A} = \begin{bmatrix} x_1 & y_1 \\ x_2 & y_2 \\ \vdots & \vdots \\ x_M & y_M \end{bmatrix}$$

we have performed the SVD for \mathbf{A} and decompose it into

$$\begin{aligned} \mathbf{A} &= \mathbf{U}\mathbf{S}\mathbf{V}^T \\ \mathbf{U} &= [\mathbf{u}_1 \quad \mathbf{u}_2 \quad \cdots \quad \mathbf{u}_M] \quad \mathbf{V} = [\mathbf{v}_1 \quad \mathbf{v}_2] \quad \mathbf{S} = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \\ \vdots & \vdots \\ 0 & 0 \end{bmatrix} \\ \mathbf{A} &= \lambda_1 \mathbf{u}_1 \mathbf{v}_1^T + \lambda_2 \mathbf{u}_2 \mathbf{v}_2^T \end{aligned} \tag{4}$$

$$\text{If } \mathbf{v}_1 = \begin{bmatrix} v_{1,1} \\ v_{1,2} \end{bmatrix}, \quad \mathbf{v}_2 = \begin{bmatrix} v_{2,1} \\ v_{2,2} \end{bmatrix} \text{ then } \mathbf{v}_1 \text{ and } \mathbf{v}_2 \text{ are orthonormal } \mathbf{v}_1^T \mathbf{v}_2 = \mathbf{v}_2^T \mathbf{v}_1 = 0 \\ \mathbf{v}_1^T \mathbf{v}_1 = \mathbf{v}_2^T \mathbf{v}_2 = 1$$

Therefore,

$$\mathbf{A}\mathbf{v}_1 = \lambda_1 \mathbf{u}_1 \mathbf{v}_1^H \mathbf{v}_1 + \lambda_2 \mathbf{u}_2 \mathbf{v}_2^H \mathbf{v}_1 = \lambda_1 \mathbf{u}_1 \quad \mathbf{A}\mathbf{v}_2 = \lambda_1 \mathbf{u}_2 \tag{5}$$

Since \mathbf{v}_1 and \mathbf{v}_2 are orthonormal, any two-entry vector \mathbf{e} can be expressed as

$$\mathbf{e} = c_1 \mathbf{v}_1^T + c_2 \mathbf{v}_2^T \quad \text{where} \quad c_1^2 + c_2^2 = 1$$

Therefore, from (3),

$$\begin{aligned} & \| \langle (x_1, y_1), \mathbf{e} \rangle \mathbf{e} \|^2 + \| \langle (x_2, y_2), \mathbf{e} \rangle \mathbf{e} \|^2 + \dots + \| \langle (x_M, y_M), \mathbf{e} \rangle \mathbf{e} \|^2 \\ &= \left(\langle (x_1, y_1), c_1 \mathbf{v}_1^T + c_2 \mathbf{v}_2^T \rangle \right)^2 + \left(\langle (x_2, y_2), c_1 \mathbf{v}_1^T + c_2 \mathbf{v}_2^T \rangle \right)^2 + \dots + \left(\langle (x_M, y_M), c_1 \mathbf{v}_1^T + c_2 \mathbf{v}_2^T \rangle \right)^2 \end{aligned} \quad (6)$$

Moreover, from (5),

$$\left(\langle (x_m, y_m), c_1 \mathbf{v}_1^T + c_2 \mathbf{v}_2^T \rangle \right)^2 = (\lambda_1 c_1 u_{1,m} + \lambda_2 c_2 u_{2,m})^2 \quad (7)$$

where $u_{1,m}$ and $u_{2,m}$ are the m^{th} entries of \mathbf{u}_1 and \mathbf{u}_2 , respectively. Therefore,

$$\begin{aligned} & \| \langle (x_1, y_1), \mathbf{e} \rangle \mathbf{e} \|^2 + \| \langle (x_2, y_2), \mathbf{e} \rangle \mathbf{e} \|^2 + \dots + \| \langle (x_M, y_M), \mathbf{e} \rangle \mathbf{e} \|^2 \\ &= \sum_{m=1}^M (c_1 \lambda_1 u_{1,m} + c_2 \lambda_2 u_{2,m})^2 = c_1^2 \lambda_1^2 \sum_{m=1}^M u_{1,m}^2 + c_2^2 \lambda_2^2 \sum_{m=1}^M u_{2,m}^2 + 2c_1 \lambda_1 c_2 \lambda_2 \sum_{m=1}^M u_{1,m} u_{2,m} \end{aligned}$$

Since \mathbf{u}_1 and \mathbf{u}_2 are orthonormal,

$$\sum_{m=1}^M u_{1,m}^2 = \sum_{m=1}^M u_{2,m}^2 = 1, \quad \sum_{m=1}^M u_{1,m} u_{2,m} = 0$$

we have

$$\|\langle (x_1, y_1), \mathbf{e} \rangle \mathbf{e}\|^2 + \|\langle (x_2, y_2), \mathbf{e} \rangle \mathbf{e}\|^2 + \dots + \|\langle (x_M, y_M), \mathbf{e} \rangle \mathbf{e}\|^2 = c_1^2 \lambda_1^2 + c_2^2 \lambda_2^2$$

Since $c_1^2 + c_2^2 = 1$ and $\lambda_1 > \lambda_2$, the best way to assign c_1 and c_2 is

$$c_1 = 1, c_2 = 0$$

That is, we can choose

$$\mathbf{e} = \mathbf{v}_1^T$$

and the projection of (x_m, y_m) on \mathbf{e} is $\lambda_1 u_{1,m} \mathbf{v}_1^T$

$$\mathbf{A} = \begin{bmatrix} x_1 & y_1 \\ x_2 & y_2 \\ \vdots & \vdots \\ x_M & y_M \end{bmatrix} \cong \begin{bmatrix} \lambda_1 u_{1,1} \mathbf{v}_1^T \\ \lambda_1 u_{1,2} \mathbf{v}_1^T \\ \vdots \\ \lambda_1 u_{1,M} \mathbf{v}_1^T \end{bmatrix} = \lambda_1 \mathbf{u}_1 \mathbf{v}_1^T$$