

V. Homomorphic Signal Processing

◎ 5-A Homomorphism

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

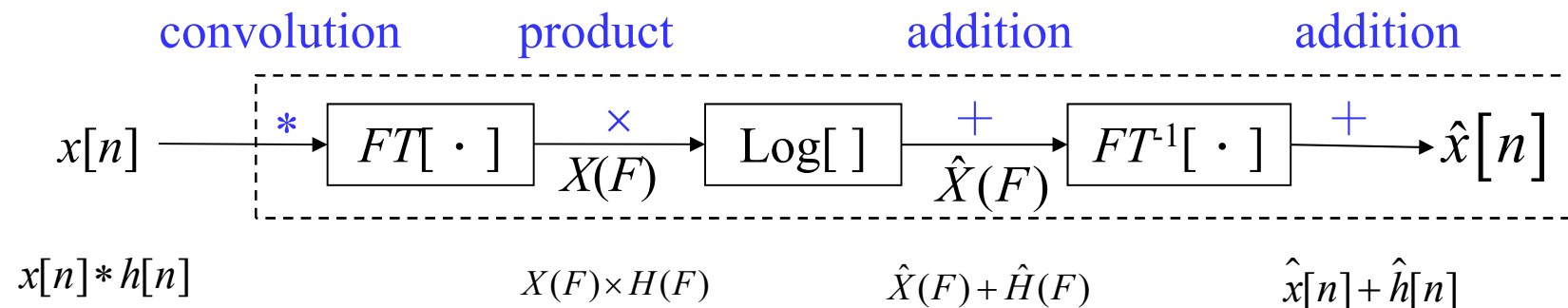
Ex. convolution $\xrightarrow{\text{Fourier}}$ multiplication $\xrightarrow{\text{log}}$ addition

把複雜的運算，變成效能相同但較簡單的運算

◎ 5-B Cepstrum

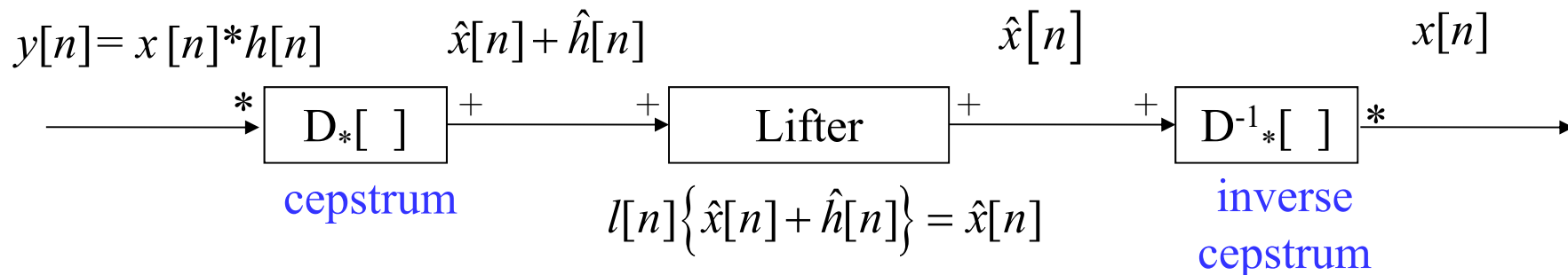
$$\hat{X}(Z)\Big|_{z=e^{j2\pi F}} = \log X(Z)\Big|_{z=e^{j2\pi F}} = \log |X(Z)\Big|_{z=e^{j2\pi F}} + j \arg[X(e^{j2\pi F})]$$

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

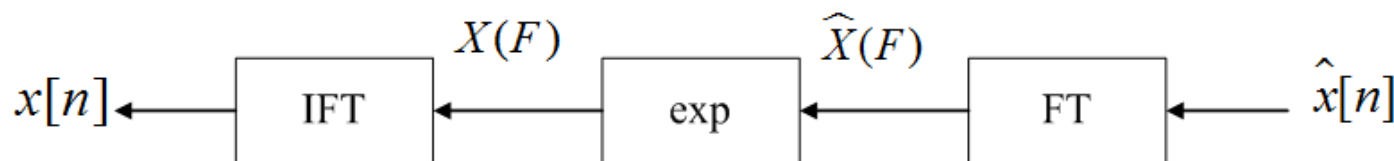


FT : discrete-time Fourier transform

- 由 $y[n] = x[n] * h[n]$ 重建 $x[n]$



For the inverse cepstrum $D^{-1}*[\cdot]$

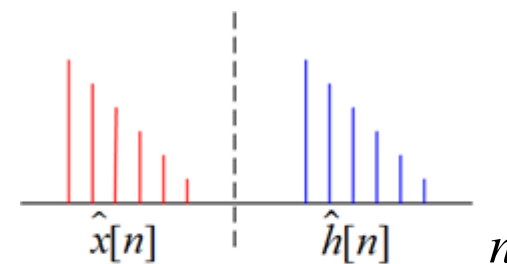


- 有趣的名詞

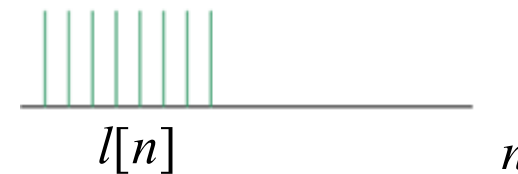
$\hat{x}[n]$ cepstrum

n quefrequency

$l[n]$ lifter



lifter



$$l[n] \{ \hat{x}[n] + \hat{h}[n] \} = \hat{x}[n]$$


© 5-C Methods for Computing the Cepstrum

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

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

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

ambiguity for phase



Problems: (1)

(2)

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

168

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

time delay

where

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

Poles & zeros
inside unit circle

Poles & zeros
outside unit circle

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

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

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

Z^{-1}
 ↓
 (inverse Z transform)
 ↓
 ?

Taylor series

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

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, $c_k = 0, d_k = 0$
- (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,

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

◎ 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

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

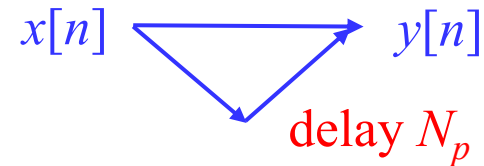
because of no a_k, c_k

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

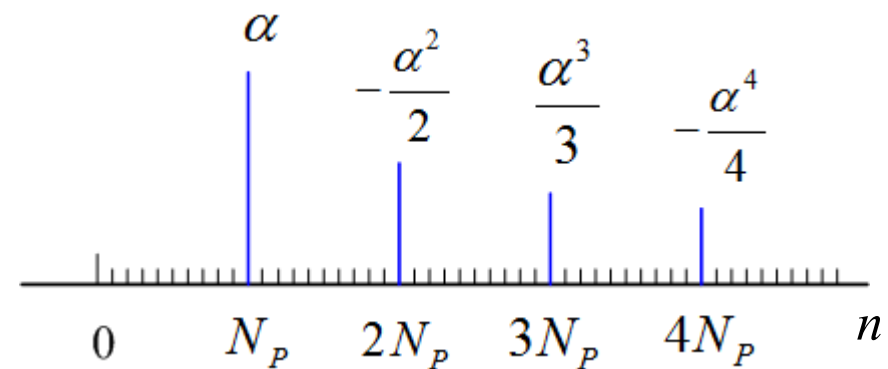
$$y[n] = x[n] + \alpha x[n - N_p] = x[n] * p[n]$$

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

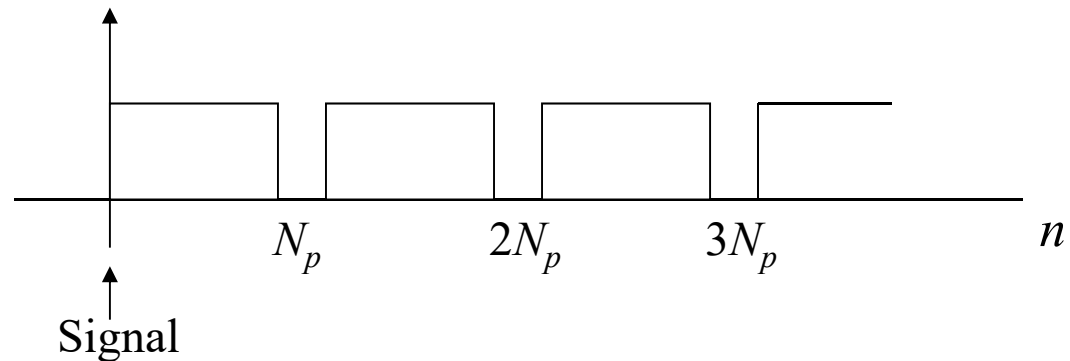
$$\hat{P}(Z) = \log(1 + \alpha Z^{-N_p}) = \sum_{k=1}^{\infty} (-1)^{k+1} \frac{\alpha^k}{k} Z^{-kN_p}$$

↓ 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

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

Synthesized 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 Global Vocal tract Pitch
wave wave impulse Pitch
 shape

They can be separated by filtering in the complex cepstrum domain

(4) Seismic Signals

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

用 cepstrum 將 multipath 的影響去除

From 王小川，「語音訊號處理」，全華出版，台北，民國94年。

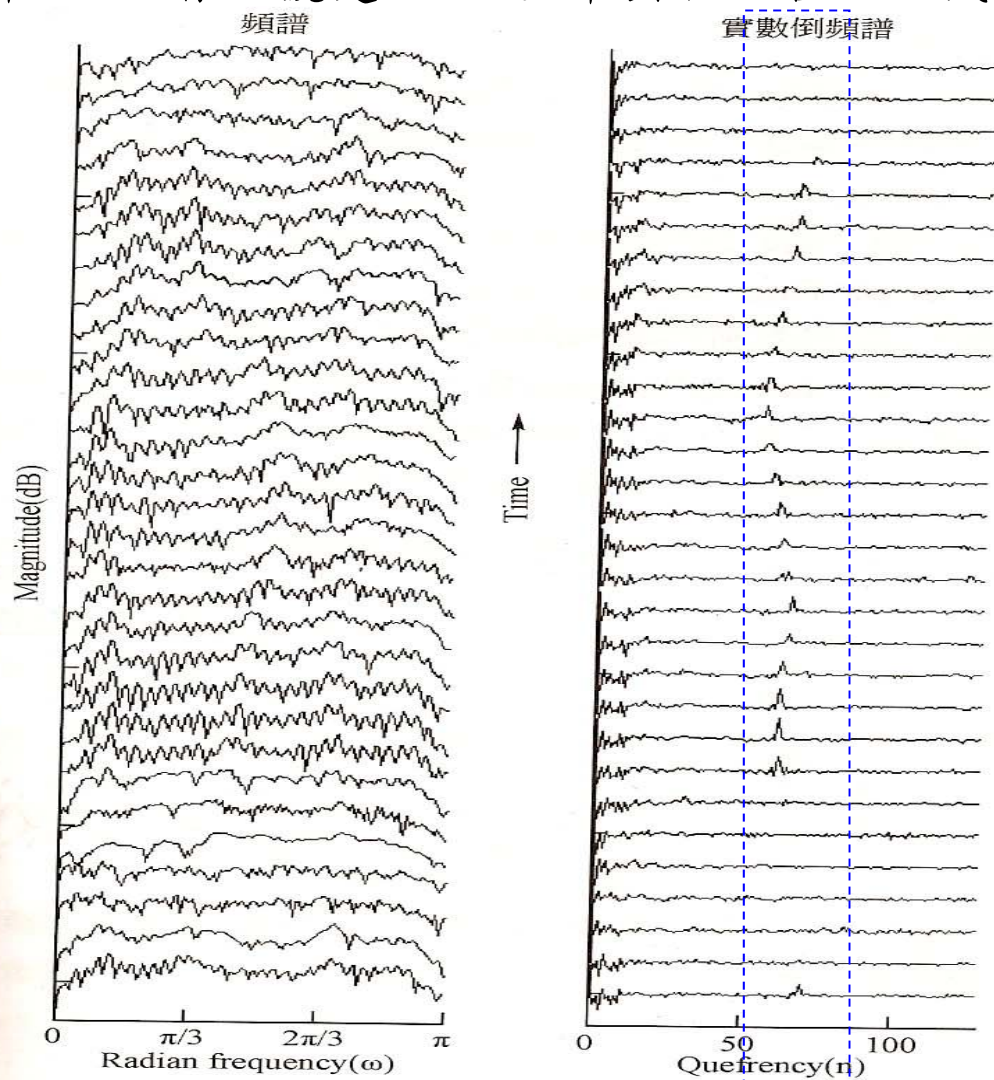


圖 7-5 元音頻譜與倒頻譜

From 王小川，「語音訊號處理」，全華出版，台北，民國94年。

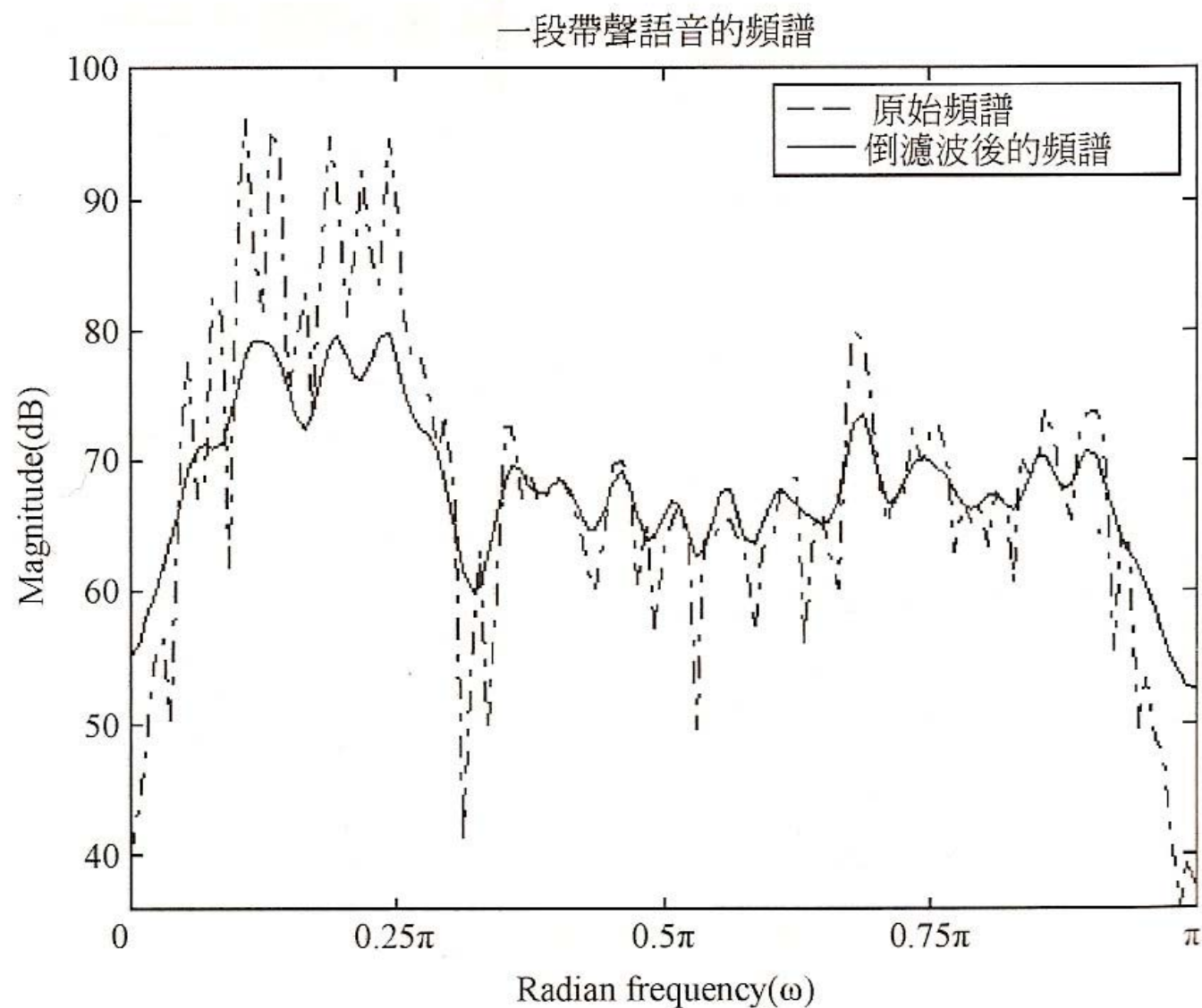


圖 7-6 經過倒濾波器作平滑處理的頻譜

◎ 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

$$\hat{x}_d(n) = Z^{-1} \left[\frac{X'(Z)}{X(Z)} \right] \quad \text{或} \quad \hat{x}_d[n] = \int_{-1/2}^{1/2} \frac{X'(F)}{X(F)} e^{i2\pi F} dF$$

inverse Z transform

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

If $x(n) = x_1(n) * x_2(n)$

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

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

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

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

Advantages: no phase ambiguity

able to deal with the delay problem

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

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

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

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

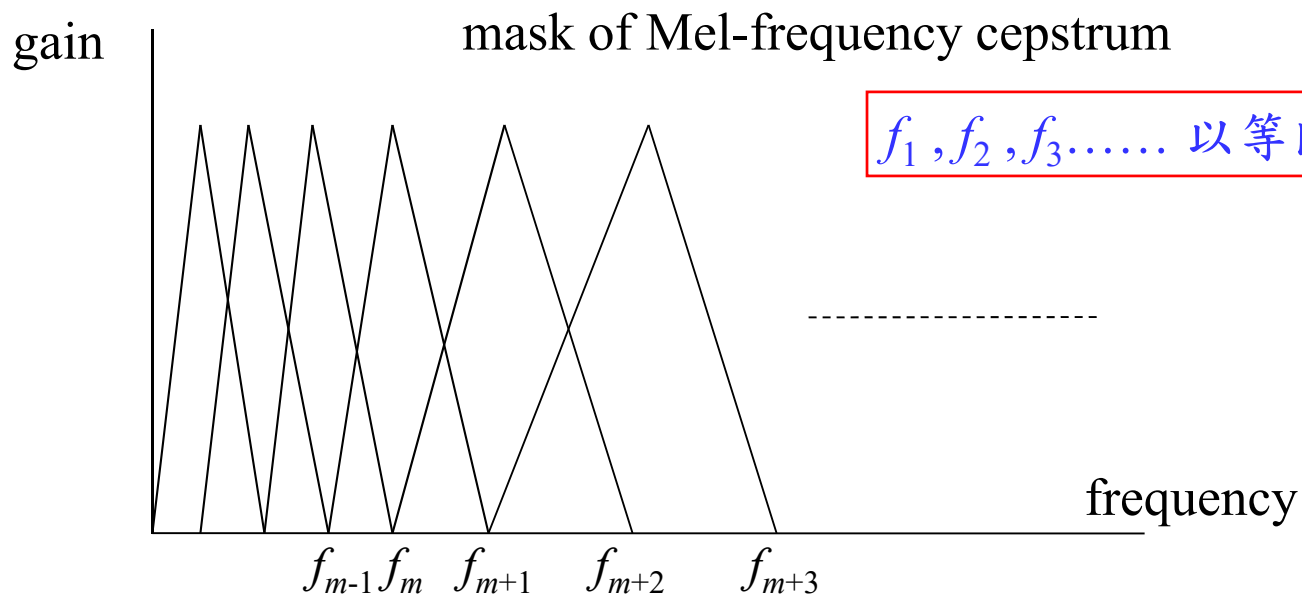
$$\hat{C}(n) = \frac{-\hat{x}_d(n+1)}{n} \quad n \neq 0 \quad \text{diff cepstrum}$$

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

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

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

Take log in the frequency mask



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

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

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

$$f = k f_s / N$$

Process of the Mel-Cepstrum

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

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

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

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

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

Advantages :

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

◎ 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. Schaffer, *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)

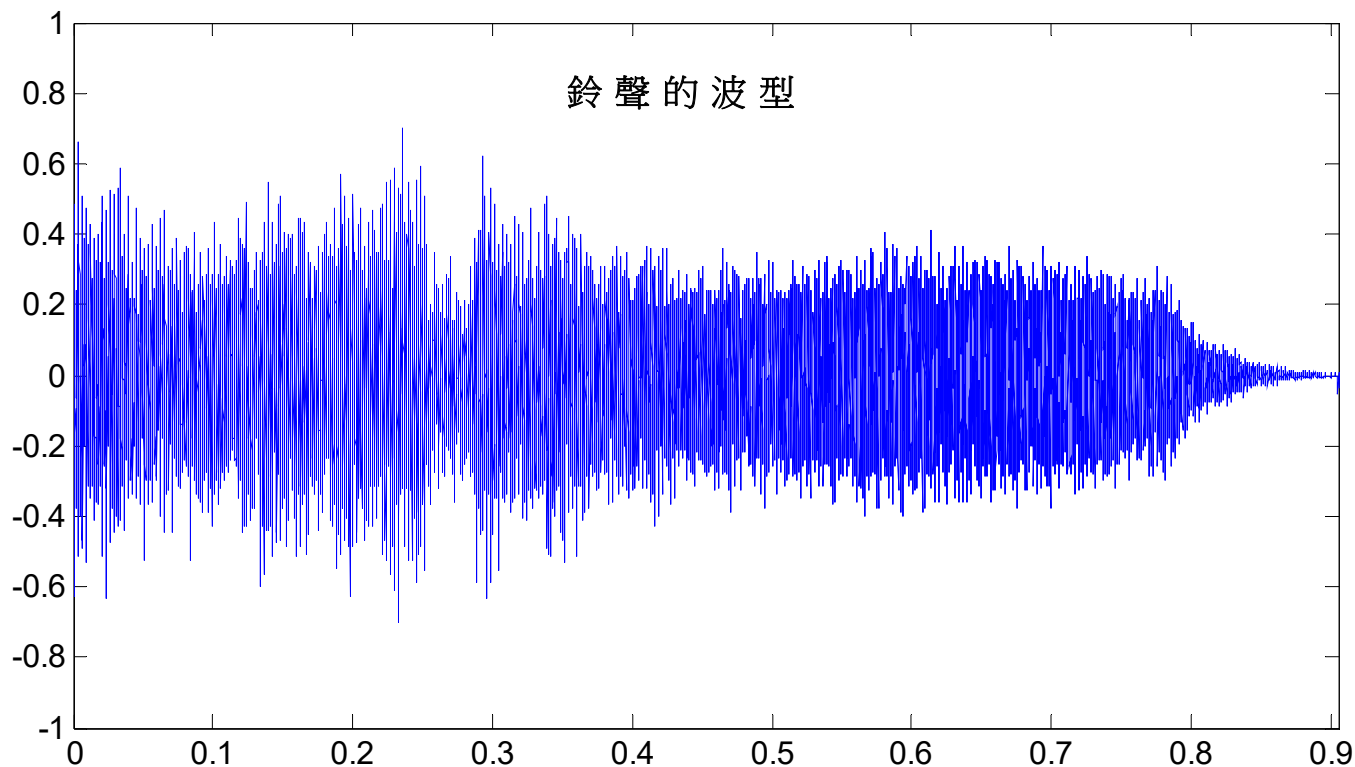
A. 讀取聲音檔

- 電腦中，沒有經過壓縮的聲音檔都是 *.wav 的型態
- 讀取：**wavread** 或 **audioread**
註：2015版本以後的 Matlab，**wavread** 將改為 **audioread**
- 例：`[x, fs] = wavread('C:\WINDOWS\Media\ringin.wav');`
可以將 ringin.wav 以數字向量 **x** 來呈現。 **fs**: sampling frequency
這個例子當中 `size(x) = 9981 1 fs = 11025`
- 思考: 所以，取樣間隔多大?
- 這個聲音檔有多少秒？

畫出聲音的波型

```
time = [0:length(x)-1]/fs; % x 是前頁用 wavread 所讀出的向量
```

```
plot(time, x)
```



注意：*.wav 檔中所讀取的資料，值都在 -1 和 +1 之間

一個聲音檔如果太大，我們也可以只讀取它部分的點

```
[x, fs]=wavread('C:\WINDOWS\Media\ringin.wav', [4001 5000]);
```

```
% 讀取第4001至5000點
```

```
[x, fs, nbits] = wavread('C:\WINDOWS\Media\ringin.wav');
```

nbits: $x(n)$ 的bit 數

第一個bit: 正負號，第二個bit: 2^{-1} ，第三個bit: 2^{-2} ，.....，

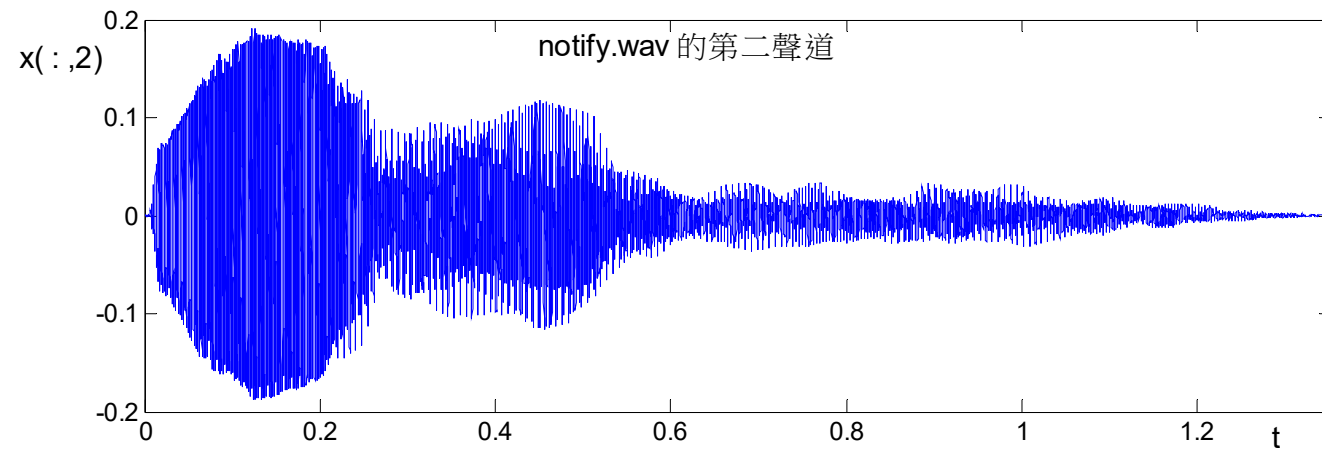
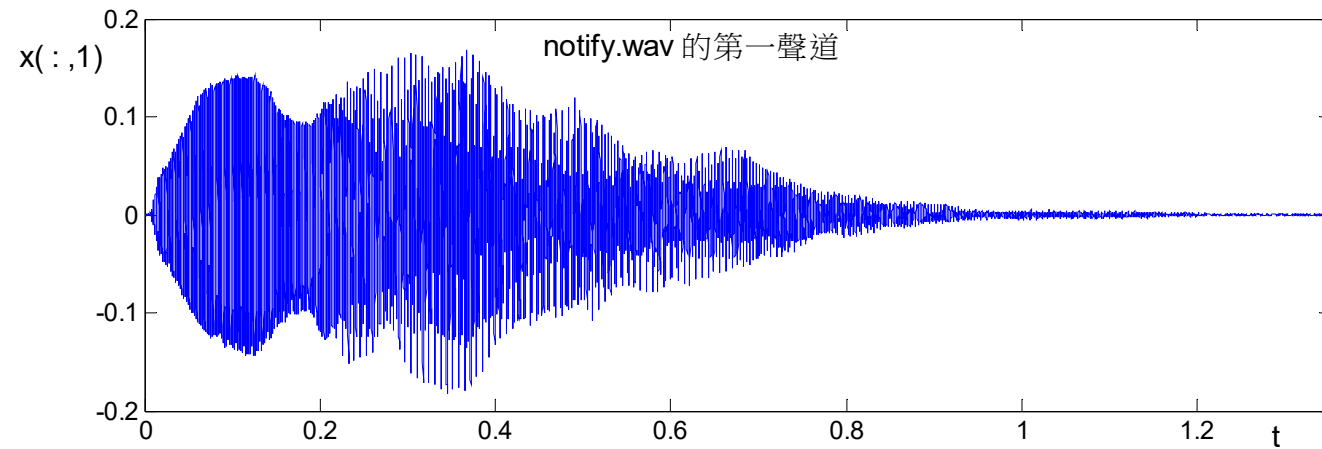
第 n 個bit: $2^{-nbits+1}$ ，所以 x 乘上 $2^{nbits-1}$ 是一個整數

以鈴聲的例子， $nbits = 8$ ，所以 x 乘上 128是個整數

- 有些聲音檔是 **雙聲道 (Stereo)** 的型態 (俗稱**立體聲**)

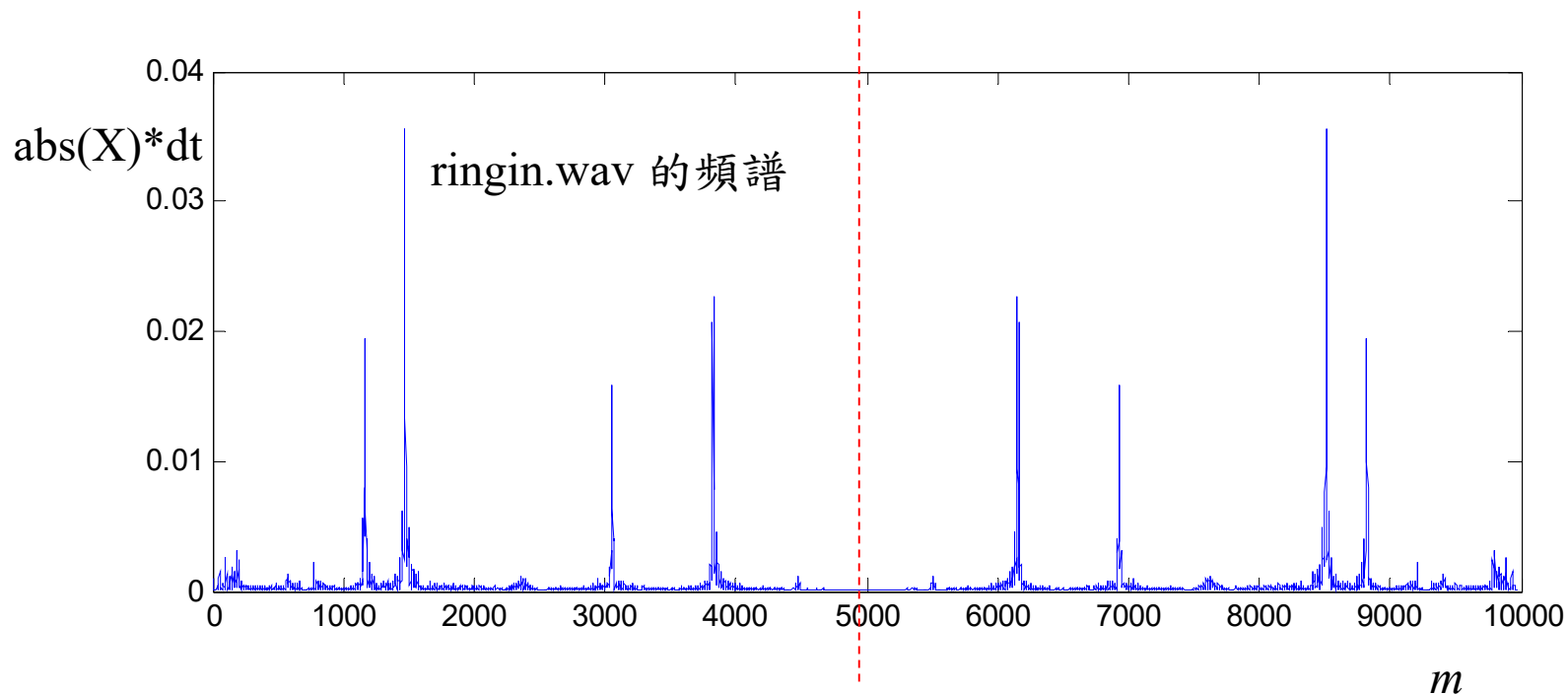
例：`[x, fs]=wavread('C:\WINDOWS\Media\notify.wav');`

`size(x) = 29823 2 fs = 22050`



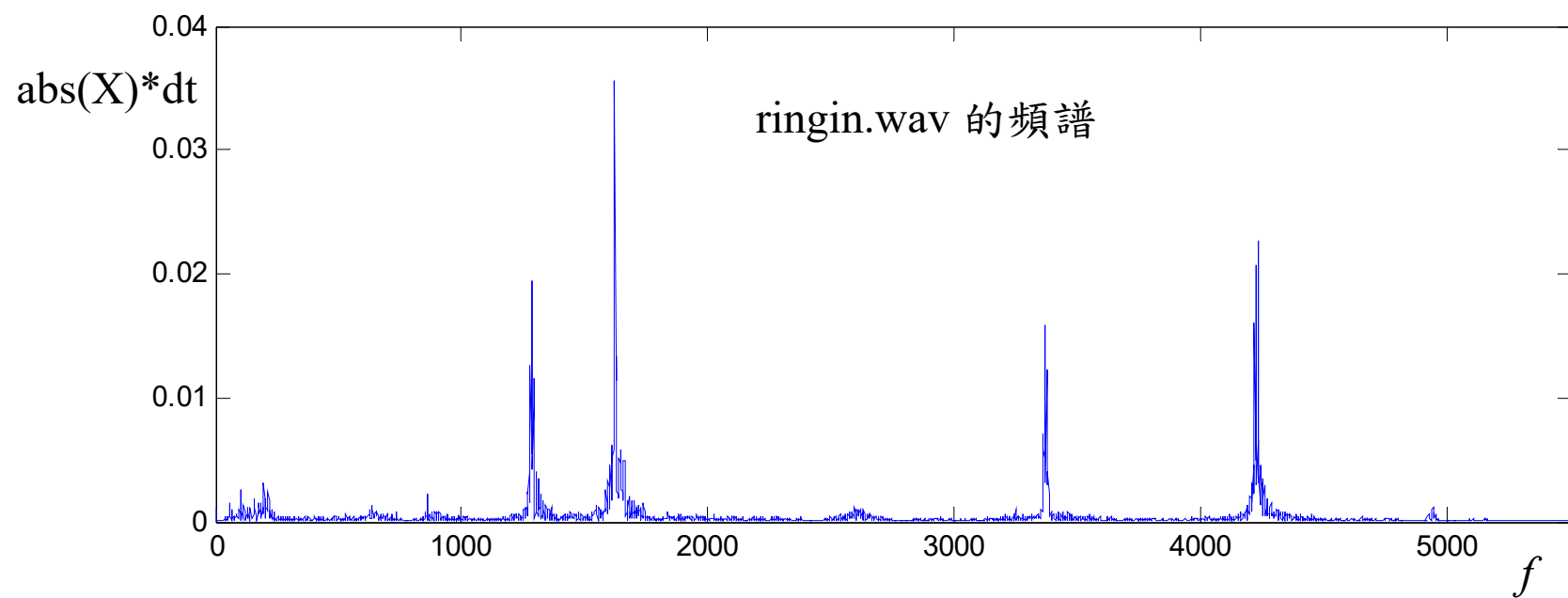
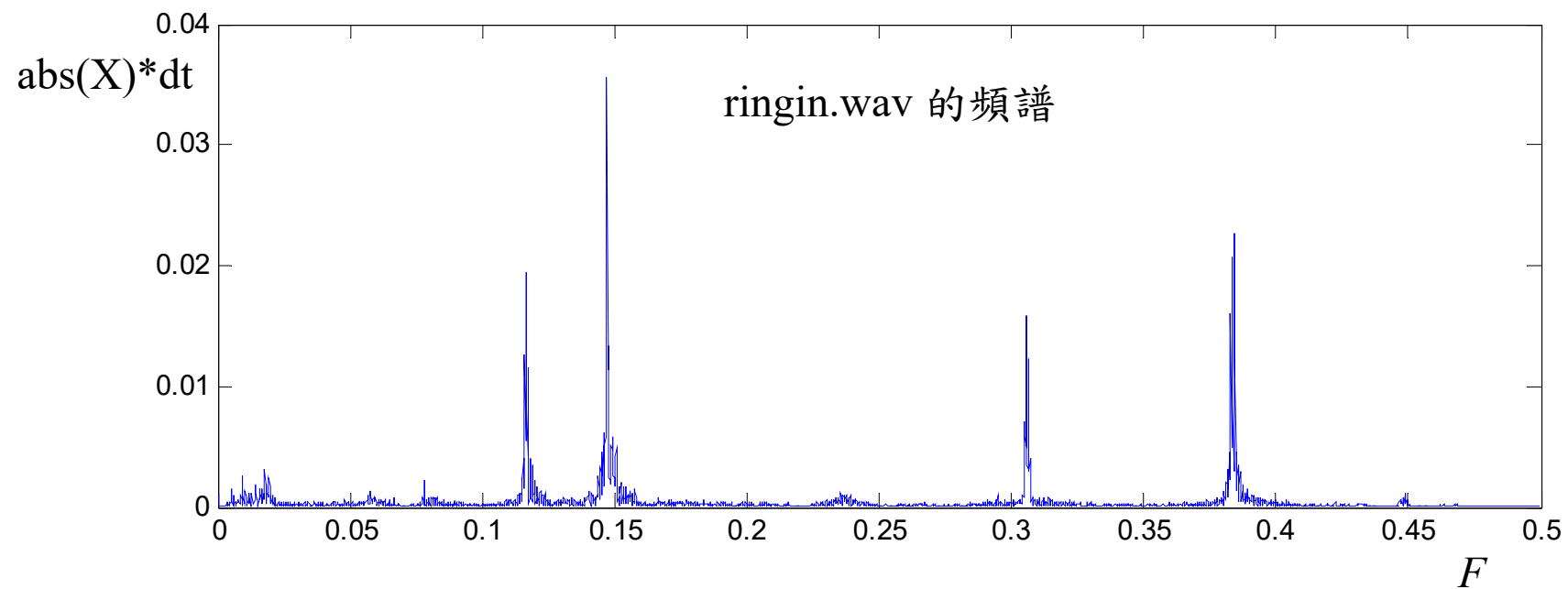
B. 繪出頻譜 (請參考附錄二)

```
X = fft(x);      plot(abs(X)*dt); % dt = 1/fs
```



fft 橫軸 轉換的方法

- (1) Using normalized frequency F : $F = m / N$.
- (2) Using frequency f , $f = F \times f_s = m \times (f_s / N)$.



C. 聲音的播放

- (1) `wavplay(x)`: 將 x 以 11025Hz 的頻率播放
(時間間隔 = $1/11025 = 9.07 \times 10^{-5}$ 秒)
- (2) `sound(x)`: 將 x 以 8192Hz 的頻率播放
- (3) `wavplay(x, fs)` 或 `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
- (4) `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：wavwrite

wavwrite(x, fs, waveFile)

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 之間
- ③ 若沒有設定fs，則預設的fs為 8000Hz

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) :

```
wavplay(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);
wavwrite(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

F、MP3 檔的讀和寫

要先去這個網站下載 mp3read.m, mp3write.m 的程式

<http://www.mathworks.com/matlabcentral/fileexchange/13852-mp3read-and-mp3write>

程式原作者：Dan Ellis

mp3read.m : 讀取 mp3 的檔案

mp3write.m : 製作 mp3 的檔案

不同於 *.wav 檔 (未壓縮過的聲音檔)，*.mp3 是經過 MPEG-2 Audio Layer III 的技術壓縮過的聲音檔

範例：

```
%% Write an MP3 file by Matlab
```

```
fs=8000;          % sampling frequency  
t = [1:fs*3]/3;  
filename = 'test';  
Nbit=32;         % number of bits per sample  
x= 0.2*cos(2*pi*(500*t+300*(t-1.5).^3));  
mp3write(x, fs, Nbit, filename); % make an MP3 file test.mp3
```

```
%% Read an MP3 file by Matlab
```

```
[x1, fs1]=mp3read('phase33.mp3');  
x2=x1(577:end); % delete the head  
sound(x2, fs1)
```

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)

```
im=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)

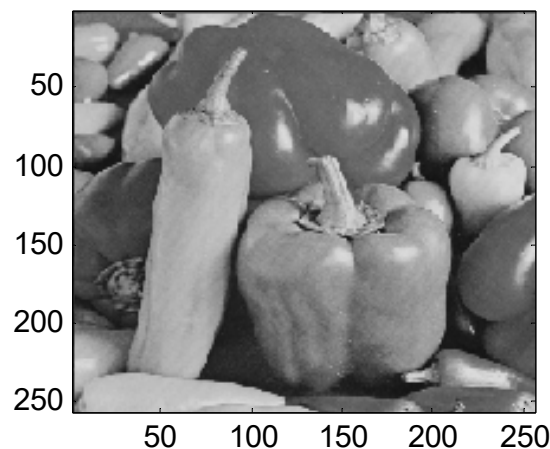
`size(im)` (用 `size` 這個指令來看 im 這個矩陣的大小)

ans =

256 256

```
image(im);
```

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



範例二：(彩色影像)

```
im2=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);
```