

## IV. Implementation

### IV-A Method 1: Direct Implementation

以 STFT 為例

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

Converting into the Discrete Form

$$t = n\Delta_t, \quad f = m\Delta_f, \quad \tau = p\Delta_t$$



$$X(n\Delta_t, m\Delta_f) = \sum_{p=-\infty}^{\infty} w((n-p)\Delta_t) x(p\Delta_t) e^{-j2\pi pm\Delta_t \Delta_f \Delta_t}$$

Suppose that  $w(t) \cong 0$  for  $|t| > B$ ,  $B/\Delta_t = Q$

$$X(n\Delta_t, m\Delta_f) = \sum_{p=\textcolor{red}{n-Q}}^{\textcolor{red}{n+Q}} w((n-p)\Delta_t) x(p\Delta_t) e^{-j2\pi pm\Delta_t \Delta_f \Delta_t}$$

Problem : 對 scaled Gabor transform 而言 ,  $Q = ?$

- **Constraint for  $\Delta_t$**  (The only constraint for the direct implementation method)

To avoid the aliasing effect,

$$\Delta_t < 1/2\Omega, \quad \Omega \text{ is the bandwidth of ?}$$

There is no constraint for  $\Delta_f$  when using the direct implementation method.

## Four Implementation Methods

(1) Direct implementation

Complexity:

假設  $t$ -axis 有  $T$  個 sampling points,  $f$ -axis 有  $F$  個 sampling points

(2) FFT-based method

Complexity:

(3) FFT-based method with recursive formula

Complexity:

(4) Chirp-Z transform method

Complexity:

### (A) Direct Implementation

Advantage : simple, flexible

Disadvantage : higher complexity

### (B) DFT-Based Method

Advantage : lower complexity

Disadvantage : with some constraints

### (C) Recursive Method

Advantage :

Disadvantage :

### (D) Chirp Z Transform

Advantage :

Disadvantage :

## IV-B Method 2: FFT-Based Method

Constraints : (ii)  $\Delta_t \Delta_f = 1/N$ ,

(iii)  $N = 1/(\Delta_t \Delta_f) \geq 2Q + 1$ : ( $\Delta_t \Delta_f$  是整數的倒數)

Standard form of the DFT 
$$Y[m] = \sum_{n=0}^{N-1} y[n] e^{-j \frac{2\pi mn}{N}}$$

$$X(n\Delta_t, m\Delta_f) = \sum_{p=n-Q}^{n+Q} w((n-p)\Delta_t) x(p\Delta_t) e^{-j 2\pi pm \Delta_t \Delta_f} \Delta_t$$

$$\Delta_t \Delta_f = 1/N$$

$$X(n\Delta_t, m\Delta_f) = \sum_{p=n-Q}^{n+Q} w((n-p)\Delta_t) x(p\Delta_t) e^{-j \frac{2\pi pm}{N}} \Delta_t$$

$$q = p - (n - Q) \rightarrow p = (n - Q) + q$$

$$X(n\Delta_t, m\Delta_f) = \Delta_t e^{j \frac{2\pi(Q-n)m}{N}} \sum_{q=0}^{2Q} w((q - Q)\Delta_t) x((q + n - Q)\Delta_t) e^{-j \frac{2\pi q m}{N}}$$

Note that the input of the  $N$ -point FFT should have  $N$  points (others are set to zero).

$$X(n\Delta_t, m\Delta_f) = \Delta_t e^{j\frac{2\pi(Q-n)m}{N}} \sum_{q=0}^{N-1} x_1(q) e^{-j\frac{2\pi q m}{N}}, \quad q = p - (n-Q) \rightarrow p = (n-Q) + q$$

where  $x_1(q) = w((Q-q)\Delta_t)x((n-Q+q)\Delta_t)$

$$x_1(q) = 0$$

$$\downarrow \quad k = q - Q$$

for  $0 \leq q \leq 2Q,$

for  $2Q < q < N.$

$$n-Q \leq n-Q+q \leq n+Q$$

$$Q \geq Q - q \geq -Q$$

$$X(n\Delta_t, m\Delta_f) = \Delta_t e^{j\frac{2\pi(Q-n)m}{N}} DFT(x_1(q))$$

where  $x_1(q) = w(k\Delta_t)x((n+k)\Delta_t) \quad \text{for } 0 \leq q \leq 2Q, \quad -Q \leq k \leq Q \quad (k = q - Q)$

$$x_1(q) = 0 \quad \text{for } 2Q < q < N.$$

(Suppose that  $w(t) = w(-t)$ )

$$X(n\Delta_t, m\Delta_f) = \Delta_t e^{j \frac{2\pi(Q-n)m}{N}} DFT(x_1(q))$$

注意：

(1) 可以使用 Matlab 的 FFT 指令來計算  $\sum_{q=0}^{N-1} x_1(q) e^{-j \frac{2\pi q m}{N}} = X_1(m)$

(2) 對每一個固定的  $n$ ，都要計算一次下方的式子

$$X(n\Delta_t, m\Delta_f) = \Delta_t e^{j \frac{2\pi(Q-n)m}{N}} \sum_{q=0}^{N-1} x_1(q) e^{-j \frac{2\pi q m}{N}} = \Delta_t e^{j \frac{2\pi(Q-n)m}{N}} X_1(m)$$

(fixed  $n$ )

(3) Complexity = ?

假設  $t = n_0\Delta_t, (n_0+1)\Delta_t, (n_0+2)\Delta_t, \dots, (n_0+T-1)\Delta_t$

$$f = m_0\Delta_f, (m_0+1)\Delta_f, (m_0+2)\Delta_f, \dots, (m_0+F-1)\Delta_f$$

Step 1: Calculate  $n_0, m_0, T, F, N, Q$

Step 2:  $n = n_0$

Step 3: Determine  $x_1(q)$

Step 4:  $X_1(m) = \text{FFT}[x_1(q)]$

Step 5: Convert  $X_1(m)$  into  $X(n\Delta_t, m\Delta_f)$

$$X(n\Delta_t, m\Delta_f) = X_1(?) \times ?$$

} page 120

$$m = f/\Delta_f$$

$$m_1 = \text{mod}(m, N) + 1$$

$$X_1[m] = \sum_{q=0}^{N-1} x_1(q) e^{-j \frac{2\pi q m}{N}}$$

$$X_1[m] = X_1[m+N]$$

Step 6: Set  $n = n+1$  and return to Step 3 until  $n = n_0+T-1$ .

## IV-C Method 3: Recursive Method

- A very fast way for implementing the rec-STFT

( $n$  和  $n-1$  有 recursive 的關係)

$$X(n\Delta_t, m\Delta_f) = \sum_{p=n-Q}^{n+Q} x(p\Delta_t) e^{-j\frac{2\pi pm}{N}} \Delta_t$$

$$X((n-1)\Delta_t, m\Delta_f) =$$

- (1) Calculate  $X(\min(n)\Delta_t, m\Delta_f)$  by the  $N$ -point FFT

$$X(n_0\Delta_t, m\Delta_f) = \Delta_t e^{j\frac{2\pi(Q-n_0)m}{N}} \sum_{q=0}^{N-1} x_1(q) e^{-j\frac{2\pi qm}{N}}, \quad n_0 = \min(n),$$

$$x_1(q) = x((n-Q+q)\Delta_t) \quad \text{for } q \leq 2Q, \quad x_1(q) = 0 \quad \text{for } q > 2Q$$

- (2) Applying the recursive formula to calculate  $X(n\Delta_t, m\Delta_f)$ ,

$$n = n_0 + 1 \sim \max(n)$$

$$\begin{aligned} X(n\Delta_t, m\Delta_f) = & X((n-1)\Delta_t, m\Delta_f) - x((n-Q-1)\Delta_t) e^{-j2\pi(n-Q-1)m/N} \Delta_t \\ & + x((n+Q)\Delta_t) e^{-j2\pi(n+Q)m/N} \Delta_t \end{aligned}$$

## IV-D Method 4: Chirp Z Transform

$$\exp(-j2\pi pm\Delta_t\Delta_f) = \exp(-j\pi p^2\Delta_t\Delta_f)\exp(j\pi(p-m)^2\Delta_t\Delta_f)\exp(-j\pi m^2\Delta_t\Delta_f)$$

For the STFT

$$X(n\Delta_t, m\Delta_f) = \sum_{p=n-Q}^{n+Q} w((n-p)\Delta_t) x(p\Delta_t) e^{-j2\pi pm\Delta_t\Delta_f} \Delta_t$$

$$X(n\Delta_t, m\Delta_f) = \Delta_t e^{-j\pi m^2\Delta_t\Delta_f} \sum_{p=n-Q}^{n+Q} w((n-p)\Delta_t) x(p\Delta_t) e^{-j\pi p^2\Delta_t\Delta_f} e^{j\pi(p-m)^2\Delta_t\Delta_f}$$

Step 1 multiplication

---

Step 2 convolution

---

Step 3 multiplication

$$\text{Step 1 } x_1[p] = w((n-p)\Delta_t) x(p\Delta_t) e^{-j\pi p^2 \Delta_t \Delta_f} \quad n-Q \leq p \leq n+Q$$

$$\text{Step 2 } X_2[n, m] = \sum_{p=n-Q}^{n+Q} x_1[p] c[m-p] \quad c[m] = e^{j\pi m^2 \Delta_t \Delta_f}$$

$$\text{Step 3 } X(n\Delta_t, m\Delta_f) = \Delta_t e^{-j\pi m^2 \Delta_t \Delta_f} X_2[n, m]$$

Step 2 在計算上，需要用到 linear convolution 的技巧

**Question:** Step 2 要用多少點的 DFT?

- Illustration for the Question on Page 124

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

- Case 1

When  $\text{length}(x[n]) = N$ ,  $\text{length}(h[n]) = K$ ,  $N$  and  $K$  are finite,

—————>  $\text{length}(y[n]) = N+K-1$ ,

Using the  $(N+K-1)$ -point DFTs (學信號處理的人一定要知道的常識)

- Case 2

$x[n]$  has finite length but  $h[n]$  has infinite length ????

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

• Case 2

$x[n]$  has finite length but  $h[n]$  has infinite length

$x[n]$  的範圍為  $n \in [n_1, n_2]$ ，範圍大小為  $N = n_2 - n_1 + 1$

$h[n]$  無限長

$$y[n] = \sum_k x[n-k]h[k] \quad y[n] \text{ 每一點都有值 (範圍無限大)}$$

但我們只想求出  $y[n]$  的其中一段

希望算出的  $y[n]$  的範圍為  $n \in [m_1, m_2]$ ，範圍大小為  $M = m_2 - m_1 + 1$

$h[n]$  的範圍？

要用多少點的 FFT？

$$y[n] = x[n] * h[n] = \sum_{k=-\infty}^{\infty} x[n-k]h[k]$$

改寫成  $y[n] = x[n] * h[n] = \sum_{s=n_1}^{n_2} x[s]h[n-s]$

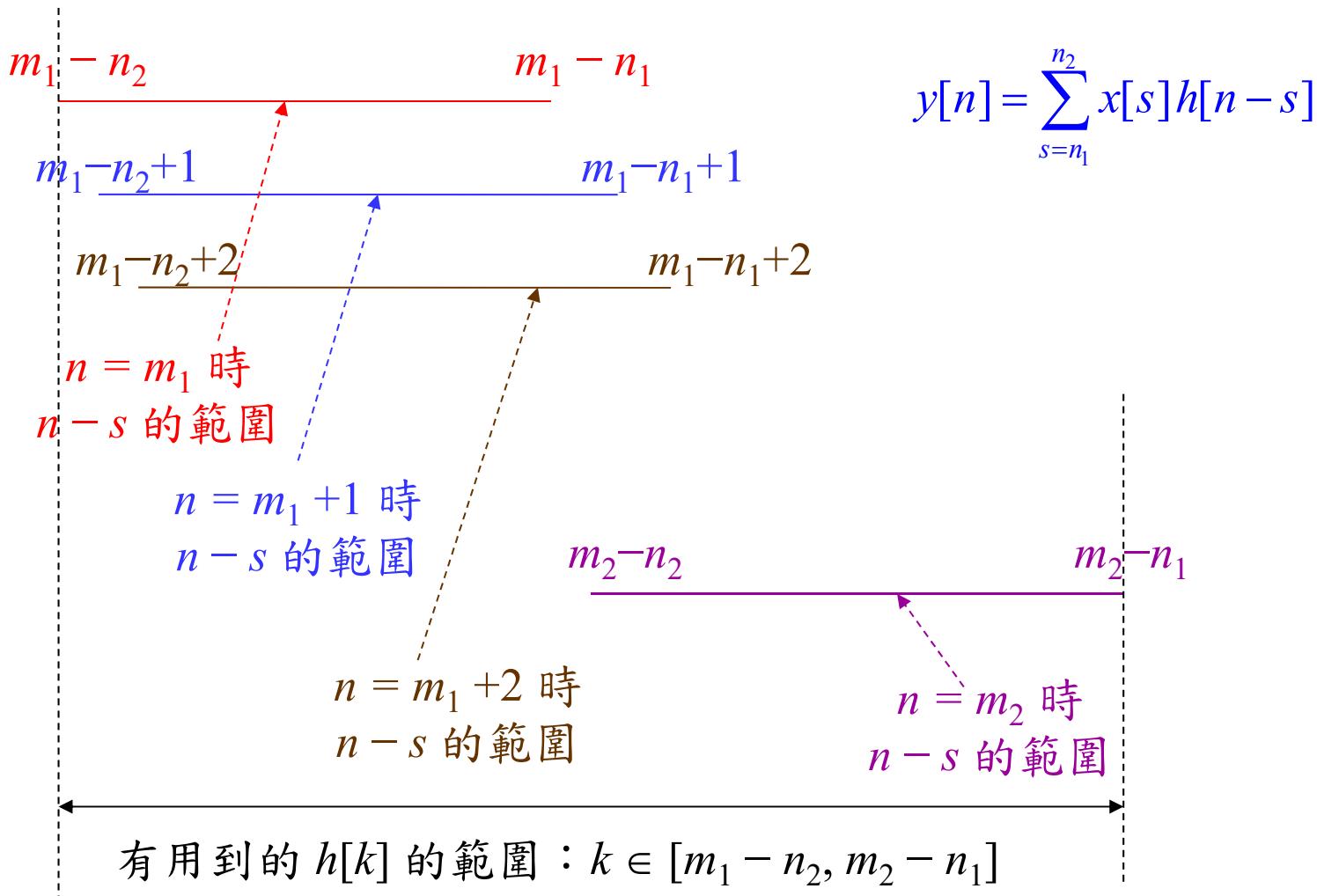
$$\begin{aligned} y[n] &= x[n_1]h[n-n_1] + x[n_1+1]h[n-n_1-1] + x[n_1+2]h[n-n_1-2] \\ &\quad + \dots + x[n_2]h[n-n_2] \end{aligned}$$

當  $n = m_1$

$$\begin{aligned} y[m_1] &= x[n_1]h[m_1-n_1] + x[n_1+1]h[m_1-n_1-1] + x[n_1+2]h[m_1-n_1-2] \\ &\quad + \dots + x[n_2]h[m_1-n_2] \end{aligned}$$

當  $n = m_2$

$$\begin{aligned} y[m_2] &= x[n_1]h[m_2-n_1] + x[n_1+1]h[m_2-n_1-1] + x[n_1+2]h[m_2-n_1-2] \\ &\quad + \dots + x[n_2]h[m_2-n_2] \end{aligned}$$



所以，有用到的  $h[k]$  的範圍是  $k \in [m_1 - n_2, m_2 - n_1]$

範圍大小為  $m_2 - n_1 - m_1 + n_2 + 1 = N + M - 1$

FFT implementation for Case 2

$$x_1[n] = x[n + n_1] \quad \text{for } n = 0, 1, 2, \dots, N-1$$

$$x_1[n] = 0 \quad \text{for } n = N, N+1, N+2, \dots, L-1 \quad L = N + M - 1$$

$$h_1[n] = h[n + m_1 - n_2] \quad \text{for } n = 0, 1, 2, \dots, L-1$$

$$y_1[n] = IFFT_L \left( FFT_L \{x_1[n]\} FFT_L \{h_1[n]\} \right)$$

$$y[n] = y_1[n - m_1 + N - 1] \quad \text{for } n = m_1, m_1+1, m_1+2, \dots, m_2$$

## IV-E Unbalanced Sampling for STFT and WDF

將 pages 115 and 119 的方法作修正

$$\begin{aligned} X(t, f) &= \int_{-\infty}^{\infty} w(t - \tau) x(\tau) e^{-j2\pi f \tau} d\tau \\ &\downarrow \\ X(n\Delta_t, m\Delta_f) &= \sum_{p=nS-Q}^{nS+Q} w((nS - p)\Delta_\tau) x(p\Delta_\tau) e^{-j2\pi pm\Delta_\tau \Delta_f} \Delta_\tau \end{aligned}$$

where  $t = n\Delta_t$ ,  $f = m\Delta_f$ ,  $\tau = p\Delta_\tau$ ,  $B = Q\Delta_\tau$  (假設  $w(t) \cong 0$  for  $|t| > B$ ),

$$S = \Delta_t / \Delta_\tau \quad \Delta_t \neq \Delta_\tau$$

註： $\Delta_\tau$  (sampling interval for the **input signal**)

$\Delta_t$  (sampling interval for the **output t-axis**) can be different.

However, it is better that  $S = \Delta_t / \Delta_\tau$  is an integer.

When (1)  $\Delta_\tau \Delta_f = 1/N$ , (2)  $N = 1/(\Delta_\tau \Delta_f) > 2Q + 1$ : ( $\Delta_\tau \Delta_f$  只要是整數的倒數即可)

(3)  $\Delta_\tau < 1/2\Omega$ ,  $\Omega$  is the bandwidth of  $w(\tau - t)x(\tau)$

i.e.,  $|FT\{w(\tau - t)x(\tau)\}| = |X(t, f)| \approx 0$  when  $|f| > \Omega$

$$X(n\Delta_t, m\Delta_f) = \sum_{p=nS-Q}^{nS+Q} w((nS-p)\Delta_\tau) x(p\Delta_\tau) e^{-j\frac{2\pi pm}{N}} \Delta_\tau$$

令  $q = p - (nS - Q) \rightarrow p = (nS - Q) + q$

$$X(n\Delta_t, m\Delta_f) = \Delta_\tau e^{j\frac{2\pi(Q-nS)m}{N}} \sum_{q=0}^{N-1} x_1(q) e^{-j\frac{2\pi qm}{N}}$$

$$x_1(q) = w((Q-q)\Delta_\tau) x((nS-Q+q)\Delta_\tau) \quad \text{for } 0 \leq q \leq 2Q,$$

$$x_1(q) = 0 \quad \text{for } 2Q < q < N.$$

If  $w(t) = w(-t)$ )

$$x_1(q) = w(k\Delta_\tau) x((nS+k)\Delta_\tau) \quad \text{for } 0 \leq q \leq 2Q, \quad k = q - Q$$

$$x_1(q) = 0 \quad \text{for } 2Q < q < N.$$

假設  $t = c_0\Delta_t, (c_0+1)\Delta_t, (c_0+2)\Delta_t, \dots, (c_0+C-1)\Delta_t$

$$= c_0S\Delta_\tau, (c_0S+S)\Delta_\tau, (c_0S+2S)\Delta_\tau, \dots, [c_0S+(C-1)S]\Delta_\tau$$

$$f = m_0\Delta_f, (m_0+1)\Delta_f, (m_0+2)\Delta_f, \dots, (m_0+F-1)\Delta_f$$

$$\tau = n_0\Delta_\tau, (n_0+1)\Delta_\tau, (n_0+2)\Delta_\tau, \dots, (n_0+T-1)\Delta_\tau$$

$$S = \Delta_t / \Delta_\tau$$

Step 1: Calculate  $c_0, m_0, n_0, C, F, T, N, Q$

Step 2:  $n = c_0$

Step 3: Determine  $x_1(q)$

Step 4:  $X_1(m) = \text{FFT}[x_1(q)]$

Step 5: Convert  $X_1(m)$  into  $X(n\Delta_t, m\Delta_f)$

Step 6: Set  $n = n+1$  and return to Step 3 until  $n = c_0+C-1$ .

**Complexity = ?**

## IV-F Non-Uniform $\Delta_t$

(A) 先用較大的  $\Delta_t$

(B) 如果發現  $|X(n\Delta_t, m\Delta_f)|$  和  $|X((n+1)\Delta_t, m\Delta_f)|$  之間有很大的差異  
則在  $n\Delta_t, (n+1)\Delta_t$  之間選用較小的 sampling interval  $\Delta_{t1}$

$(\Delta_\tau < \Delta_{t1} < \Delta_t, \Delta_t/\Delta_{t1} \text{ 和 } \Delta_{t1}/\Delta_\tau \text{ 皆為整數})$

再用 page 131 的方法算出

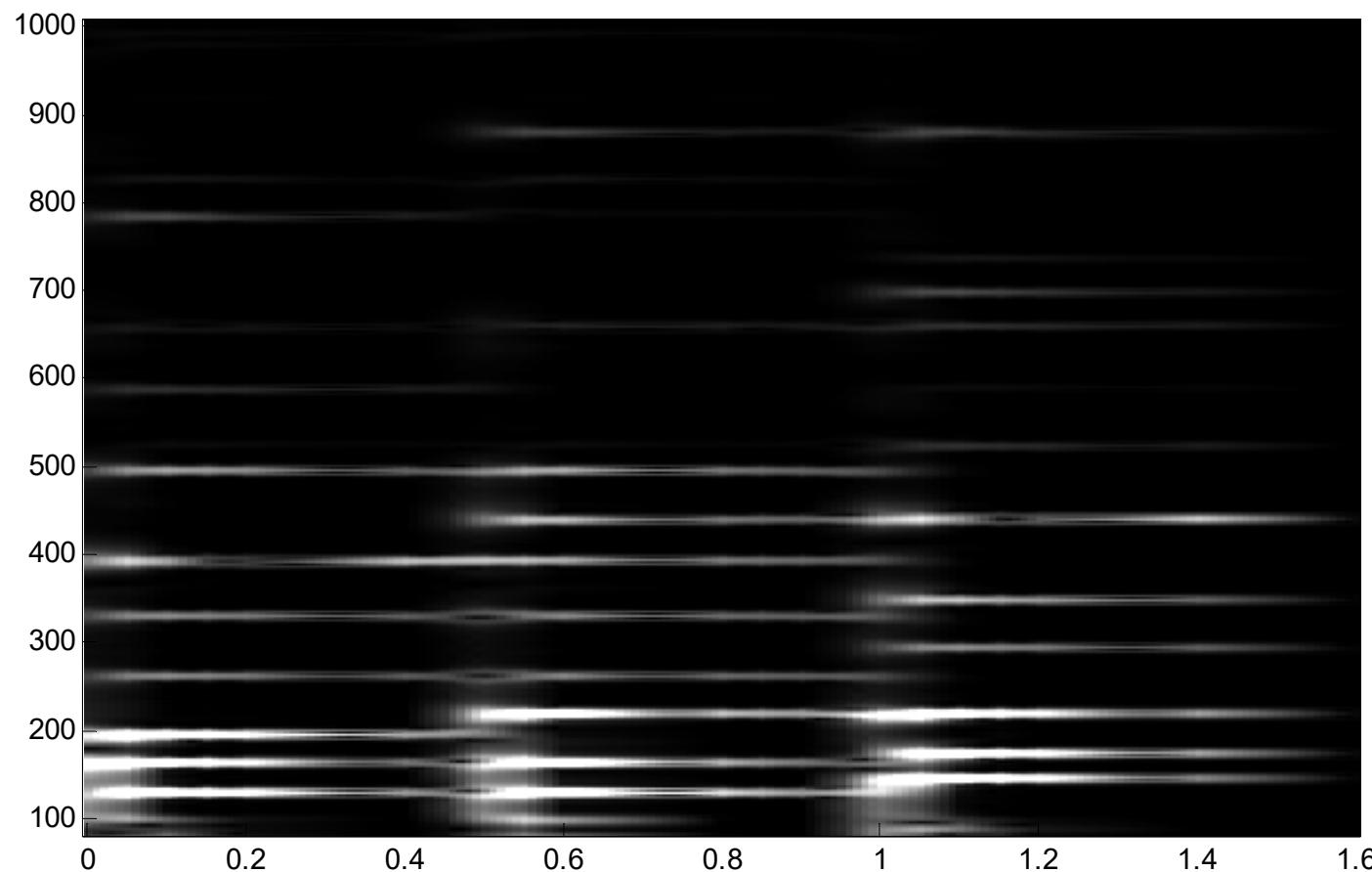
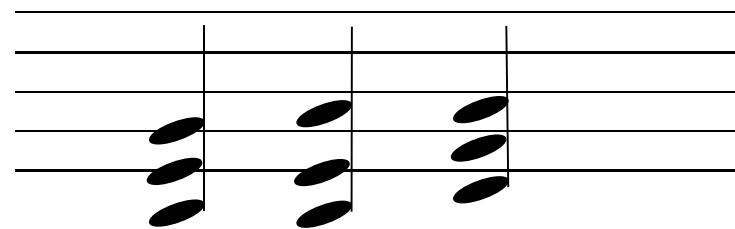
$$X(n\Delta_t + \Delta_{t1}, m\Delta_f), X(n\Delta_t + 2\Delta_{t1}, m\Delta_f), \dots, X((n+1)\Delta_t - \Delta_{t1}, m\Delta_f)$$

(C) 以此類推，如果  $|X(n\Delta_t + k\Delta_{t1}, m\Delta_f)|, |X(n\Delta_t + (k+1)\Delta_{t1}, m\Delta_f)|$

的差距還是太大，則再選用更小的 sampling interval  $\Delta_{t2}$

$(\Delta\tau < \Delta_{t2} < \Delta_{t1}, \Delta_{t1}/\Delta_{t2} \text{ 和 } \Delta_{t2}/\Delta\tau \text{ 皆為整數})$

Gabor transform of a music signal



$$\Delta_\tau = 1/44100 \text{ (總共有 } 44100 \times 1.6077 \text{ sec} + 1 = 70902 \text{ 點)}$$

(A) Choose  $\Delta_t = \Delta_\tau$

running time = out of memory (2008年) 15.262140 sec (2022年)

(B) Choose  $\Delta_t = 0.01 = 441\Delta_\tau$  ( $1.6/0.01 + 1 = 161$  points)

running time = 1.0940 sec (2008年) 0.041053 sec (2022年)

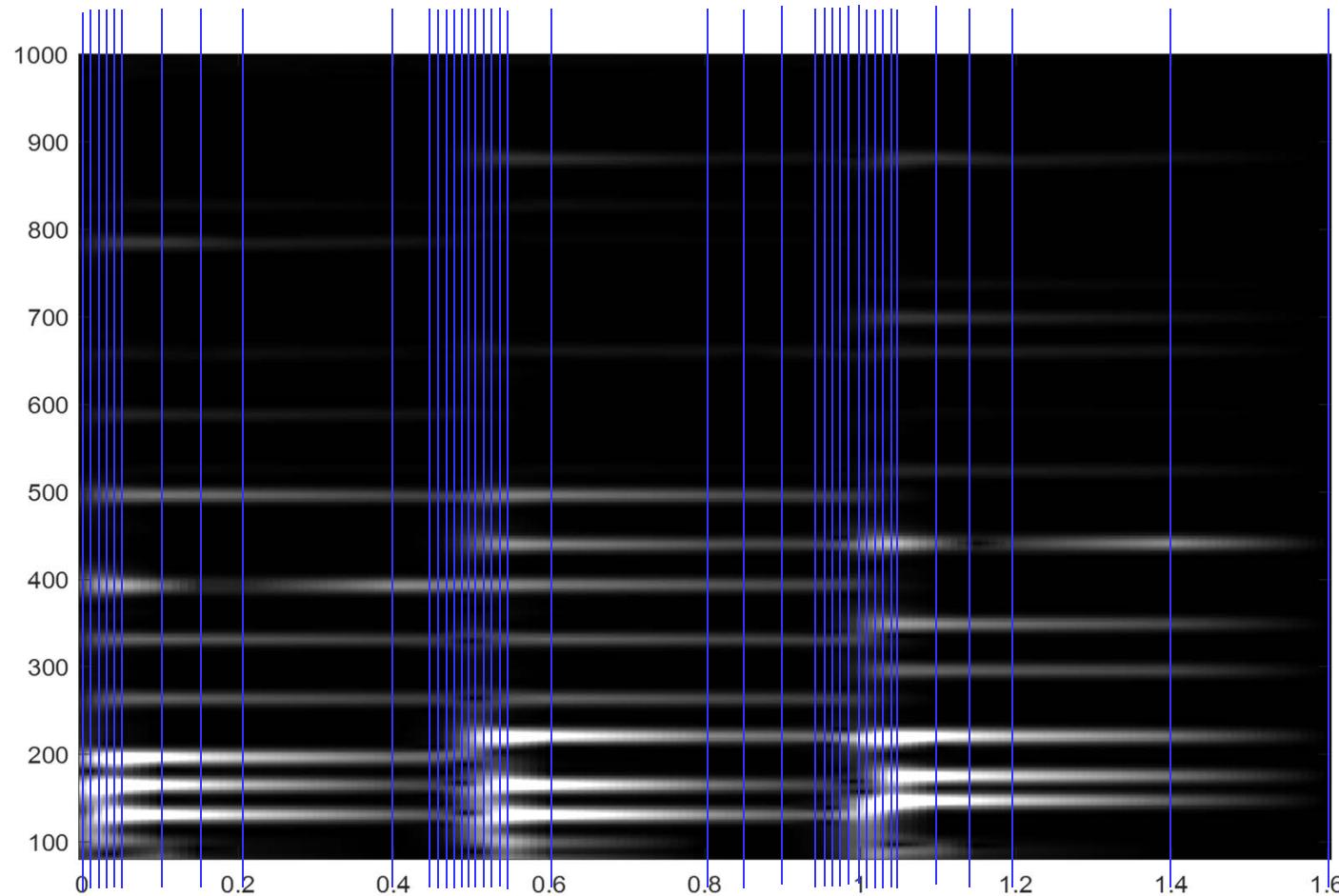
(C) Choose the sampling points on the  $t$ -axis as

$t = 0, 0.01, 0.02, 0.03, 0.04, 0.05, 0.1, 0.15, 0.2, 0.4, 0.45, 0.46, 0.47, 0.48,$   
 $0.49, 0.5, 0.51, 0.52, 0.53, 0.54, 0.55, 0.6, 0.8, 0.85, 0.9, 0.95, 0.96, 0.97,$   
 $0.98, 0.99, 1, 1.01, 1.02, 1.03, 1.04, 1.05, 1.1, 1.15, 1.2, 1.4, 1.6$

(41 points)

running time = 0.2970 sec (2008年) 0.010594 sec (2022年)

with adaptive output sampling intervals



## 附錄七 和 Dirac Delta Function 相關的常用公式

$$(1) \quad \int_{-\infty}^{\infty} e^{-j2\pi t f} dt = \delta(f)$$

$$(2) \quad \delta(t) = |a| \delta(at) \quad (\text{scaling property})$$

$$(3) \quad \int_{-\infty}^{\infty} e^{-j2\pi t g(f)} dt = \delta(g(f)) = \sum_n |g'(f_n)|^{-1} \delta(f - f_n)$$

where  $f_n$  are the zeros of  $g(f)$

$$(4) \quad \int_{-\infty}^{\infty} \delta(t - t_0) y(t, \dots) dt = y(t_0, \dots) \quad (\text{sifting property I})$$

$$(5) \quad \delta(t - t_0) y(t, \dots) = \delta(t - t_0) y(t_0, \dots) \quad (\text{sifting property II})$$