XI. Hilbert Huang Transform (HHT)

Proposed by 黃鍔院士 (AD. 1998)
黃鍔院士的生平可參考

http://djj.ee.ntu.edu.tw/%E9%BB%83%E9%8D%94%E9%99%A2%E5%A3%AB.pdf

References

[1] N. E. Huang, Z. Shen, S. R. Long, M. C. Wu, H. H. Shih, Q. Zheng, N. C. Yen, C. C. Tung, and H. H. Liu, "The empirical mode decomposition and the Hilbert spectrum for nonlinear and non-stationary time series analysis," *Proc. R. Soc. Lond. A*, vol. 454, pp. 903-995, 1998.

[2] N. E. Huang and S. Shen, *Hilbert-Huang Transform and Its Applications*, World Scientific, Singapore, 2005.

(PS: 謝謝 2007 年修課的趙逸群同學和王文阜同學)

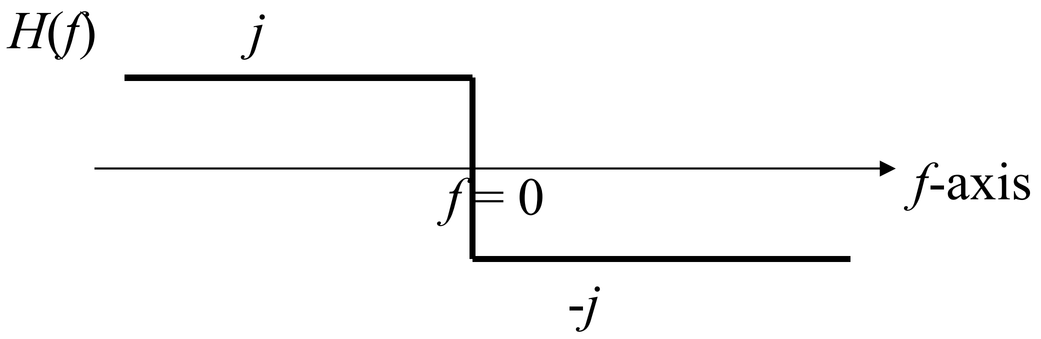
11-A The Origin of the Concept

Another instantaneous frequency analysis method： Hilbert transform

- Hilbert transform

$$x_H(t) = \frac{1}{\pi}\int_{-\infty}^{\infty} \frac{x(\tau)}{t-\tau}\,d\tau$$

or
$$x_H(t) = IFT\left\{FT\left[x(t)\right]H(f)\right\}$$



Applications of the Hilbert Transform

- analytic signal

$$x_a(t) = x(t) + jx_H(t)$$

- edge detection
- another way to define the instantaneous frequency:

$$\text{instantaneous frequency} = \frac{1}{2\pi}\frac{d}{dt}\theta$$

$$\text{where} \quad \theta = \tan^{-1}\frac{x_H(t)}{x(t)}$$

Example:

$$\cos(2\pi ft) \xrightarrow{Hilbert} \sin(2\pi ft) \qquad \theta = 2\pi ft$$

$$\sin(2\pi ft) \xrightarrow{Hilbert} -\cos(2\pi ft) \qquad \theta = 2\pi ft + \pi/2$$

Problem of using Hilbert transforms to determine the instantaneous frequency:

This method is only good for cosine and sine functions with <u>single component.</u>

Not suitable for (1) complex function

(2) non-sinusoid-like function

(3) multiple components

Moreover, (4) θ has multiple solutions.

Example:

$$\cos(2\pi f_1 t) + \cos(2\pi f_2 t) \xrightarrow{Hilbert} \sin(2\pi f_1 t) + \sin(2\pi f_2 t)$$

• Hilbert-Huang transform 的基本精神：

先將一個信號分成多個 sinusoid-like components + trend

(和 Fourier analysis 不同的地方在於，這些 sinusoid-like components 的 period 和 amplitude 可以不是固定的)

再運用 Hilbert transform (或 STFT，number of zero crossings) 來分析每個 components 的 instantaneous frequency

完全不需用到 Fourier transform

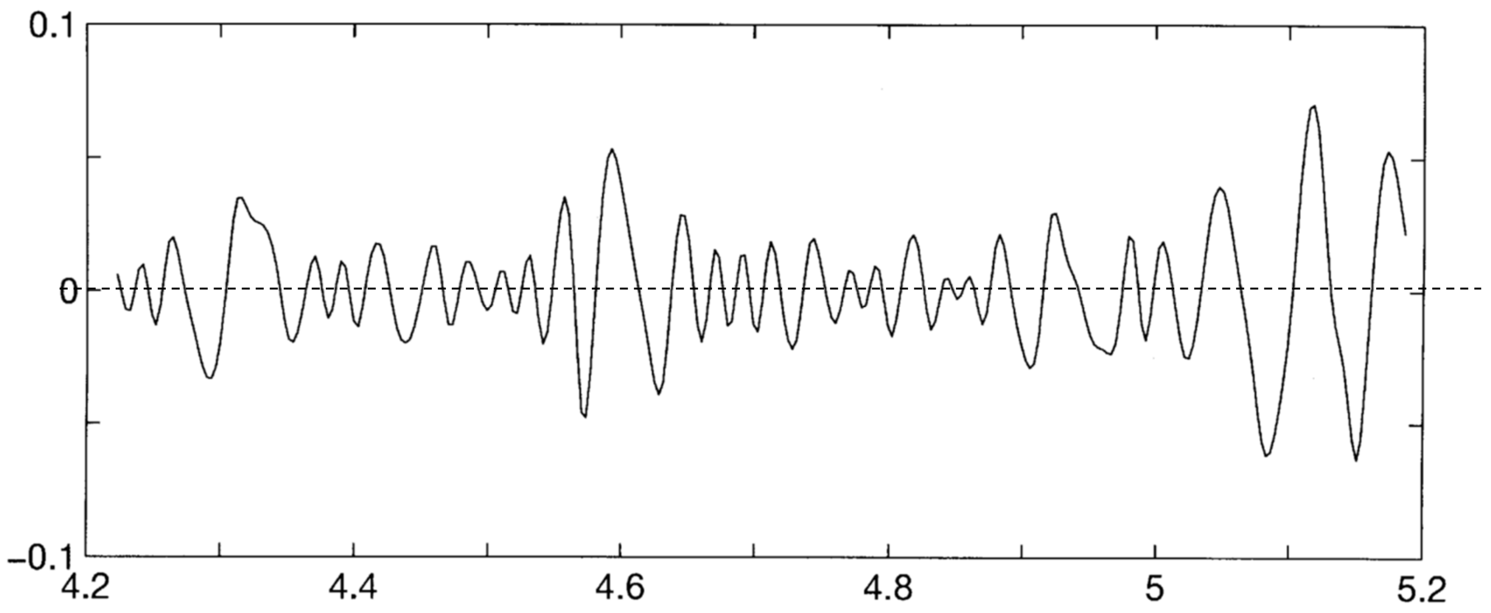
11-B Intrinsic Mode Function (IMF)

Amplitude and frequency can vary with time.

但要滿足 local maximums & local minimums

(1) The number of extremes and the number of zero-crossings must either equal or differ at most by one.

(2) At any point, the mean value of the envelope defined by the local maxima and the envelope defined by the local minima is near to zero.

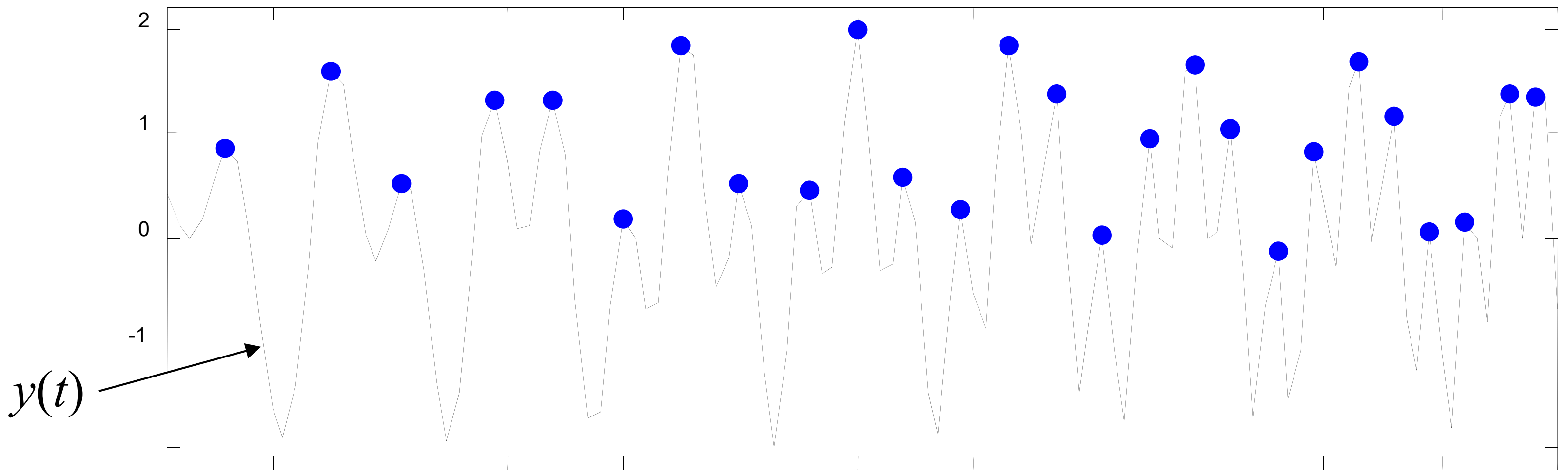


11-C Procedure of the Hilbert Huang Transform

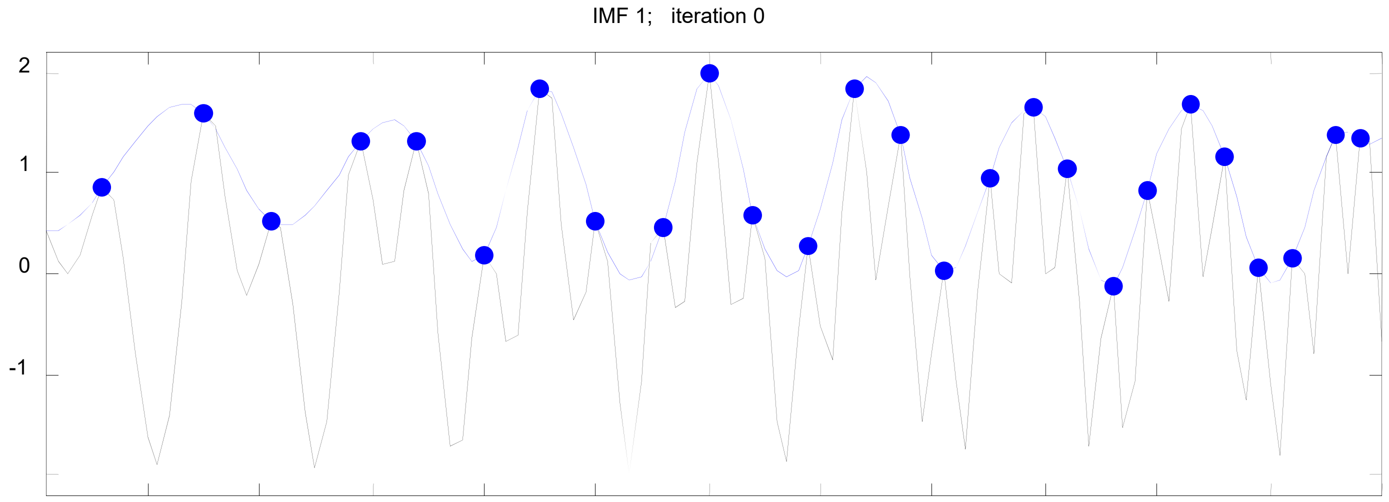
Steps 1~8 are called Empirical Mode Decomposition (EMD)

(Step 1) Initial: $y(t) = x(t)$, ($x(t)$ is the input) $n = 1$, $k = 1$

(Step 2) Find the local peaks



(Step 3) Connect local peaks

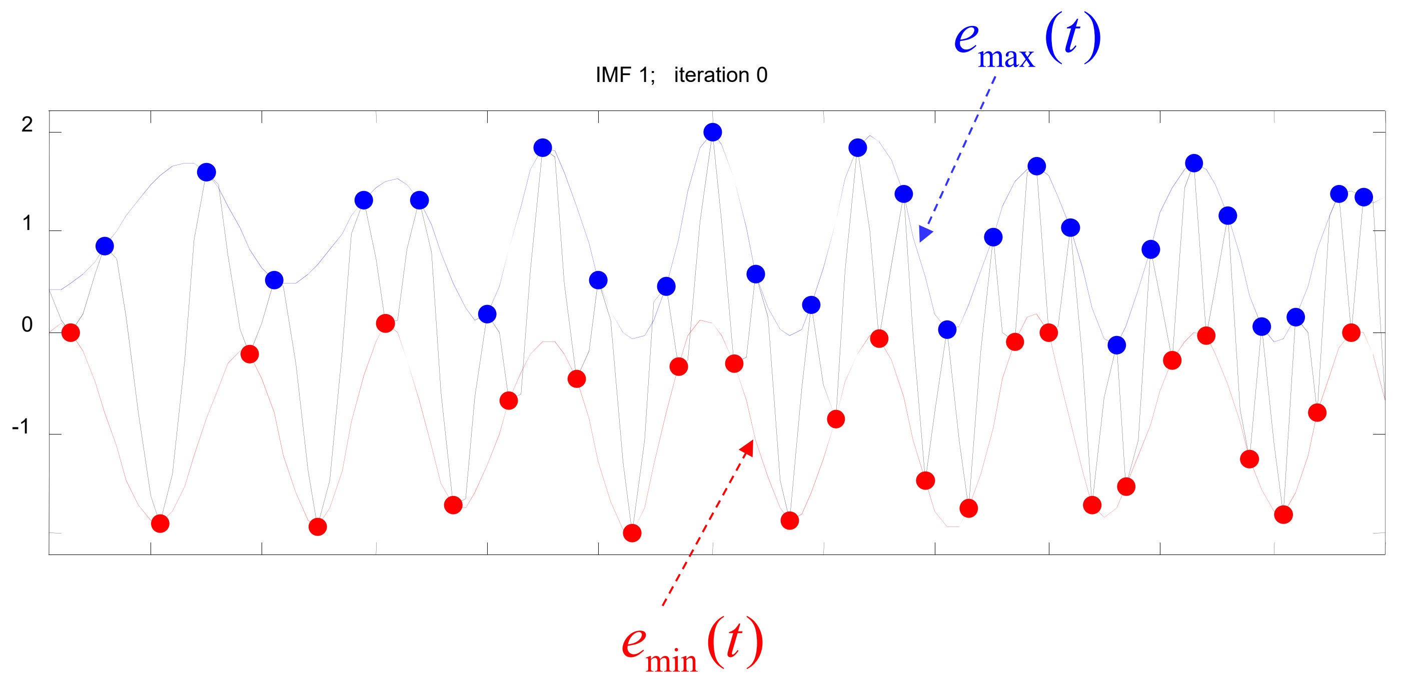


通常使用 B-spline，尤其是 cubic B-spline 來連接

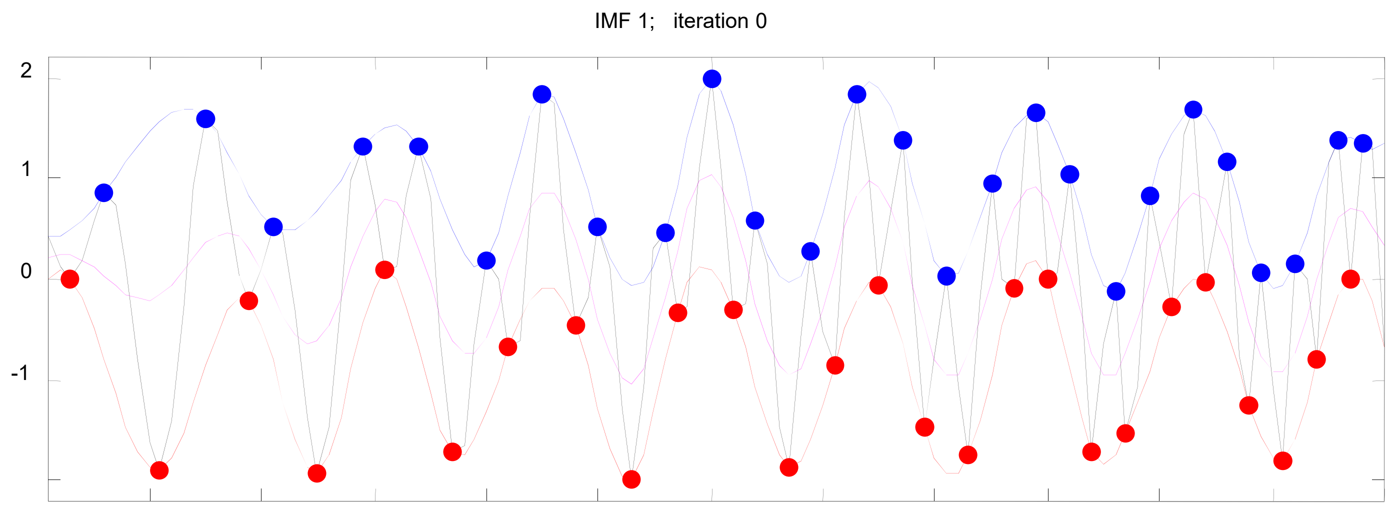
(參考附錄十一)

(Step 4) Find the local dips

(Step 5) Connect the local dips



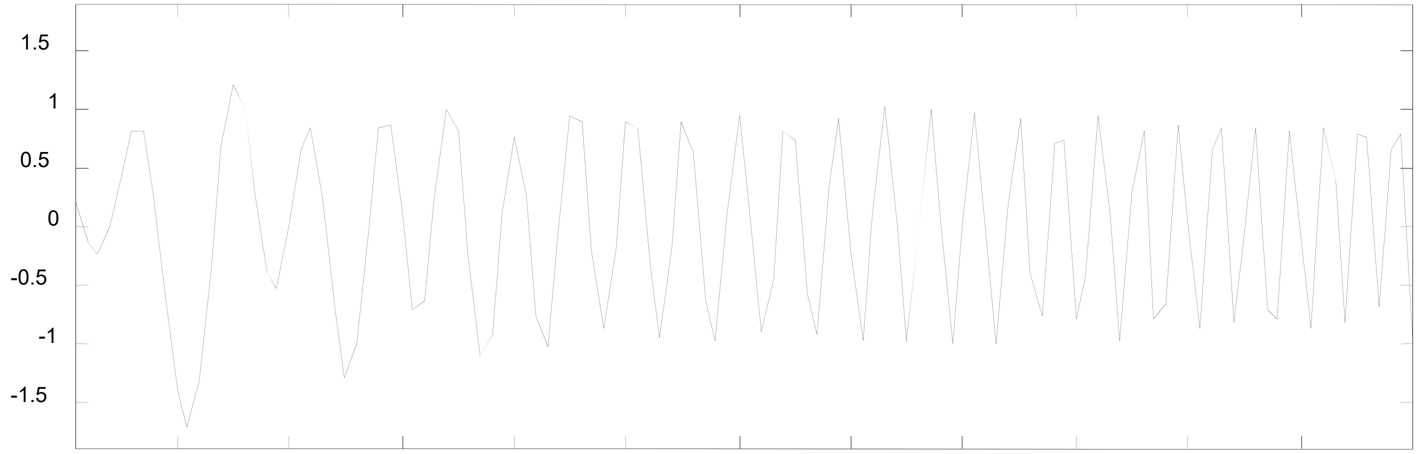
(Step 6-1) Compute the mean



$$z(t) = \frac{e_{\min}(t) + e_{\max}(t)}{2}$$

(pink line)

(Step 6-2) Compute the residue



$$h_k(t) = y(t) - z(t)$$

(Step 7) Check whether $h_k(t)$ is an intrinsic mode function (IMF)

(1) 檢查是否 local maximums 皆大於 0

local minimums 皆小於 0

(2) 上封包： $u_1(t)$，　下封包： $u_0(t)$

檢查是否 $\left|\frac{u_1(t)+u_0(t)}{2}\right| < threshold$ 　 for all t

If they are satisfied (or $k \geqq K$), set $c_n(t) = h_k(t)$ and continue to Step 8

$c_n(t)$ is the n^{th} IMF of $x(t)$.

If not, set $y(t) = h_k(t)$,

$k = k + 1$, and repeat Steps 2~6

(為了避免無止盡的迴圈，可以定 k 的上限 K)

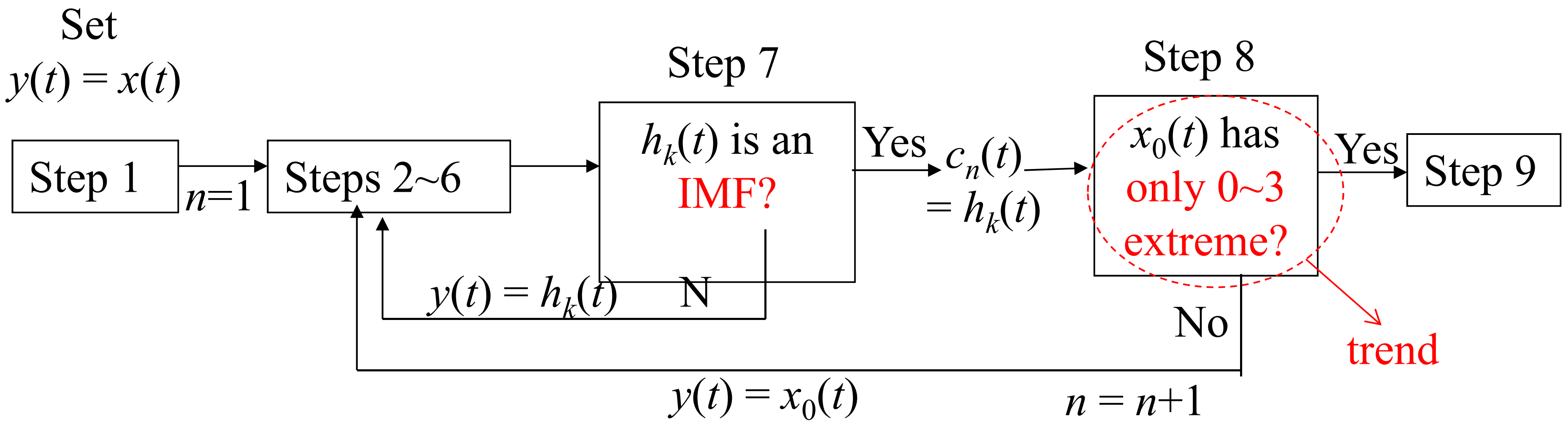
(Step 8) Calculate $x_0(t) = x(t) - \sum_{s=1}^{n} c_s(t)$

and check whether $x_0(t)$ is a function with no more than one extreme point.

If not, set $n = n+1$, $y(t) = x_0(t)$

and repeat Steps 2~7

If so, the empirical mode decomposition is completed.



$$x(t) = x_0(t) + \sum_{s=1}^{n} c_s(t)$$

(Step 9) Find the instantaneous frequency for each IMF $c_s(t)$ (s = 1, 2, …, n).

Method 1: Using the Hilbert transform

Method 2: Calculating the STFT for $c_s(t)$.

Method 3: Furthermore, we can also calculate the instantaneous frequency from the number of zero-crossings directly.

$$\text{instantaneous frequency } F_s(t) \text{ of } c_s(t)$$
$$= \frac{\text{the number of zero-crossings of } c_s(t) \text{ between } t-B \text{ and } t+B}{4B}$$

Technique Problems of the Hilbert Huang Transform

(A) 邊界處理的問題：

目前尚未有一致的方法，可行的方式有

(1) 只使用非邊界的 extreme points

(2) 將最左、最右的點當成是 extreme points

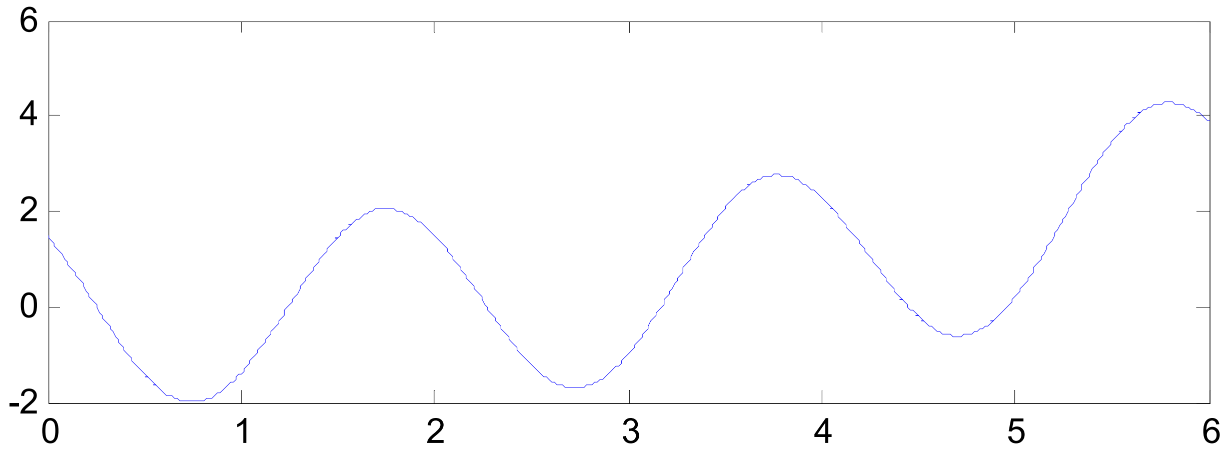
(3) 預測邊界之外的 extreme points 的位置和大小

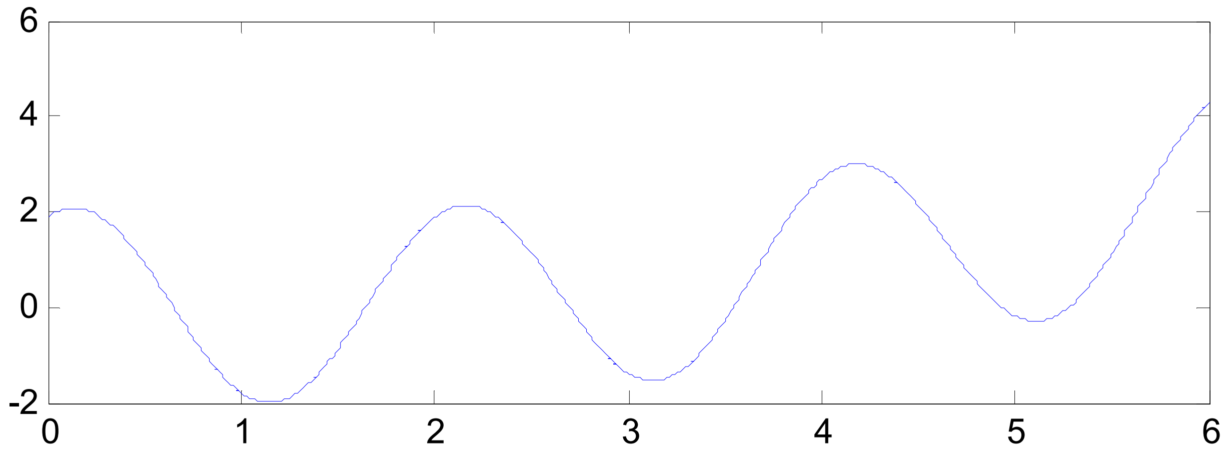
(4) 用邊界和最近的 extreme point 的距離來判斷是否邊界要當成 extreme points

(B) Noise 的問題：

先用 pre-filter 來處理

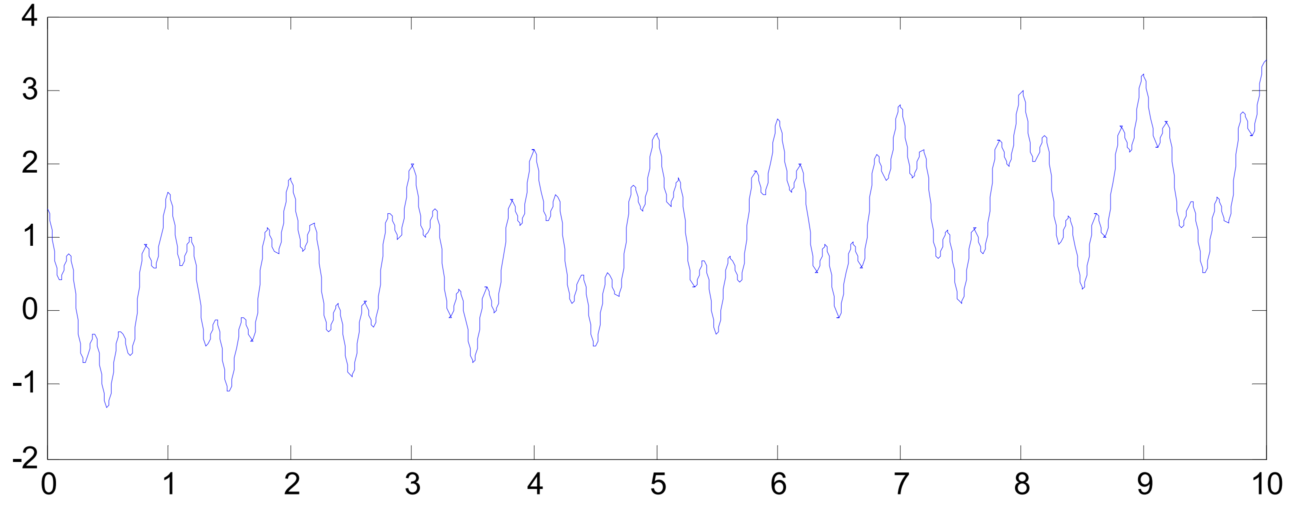
最左、最右的點是否要當成是 extreme points



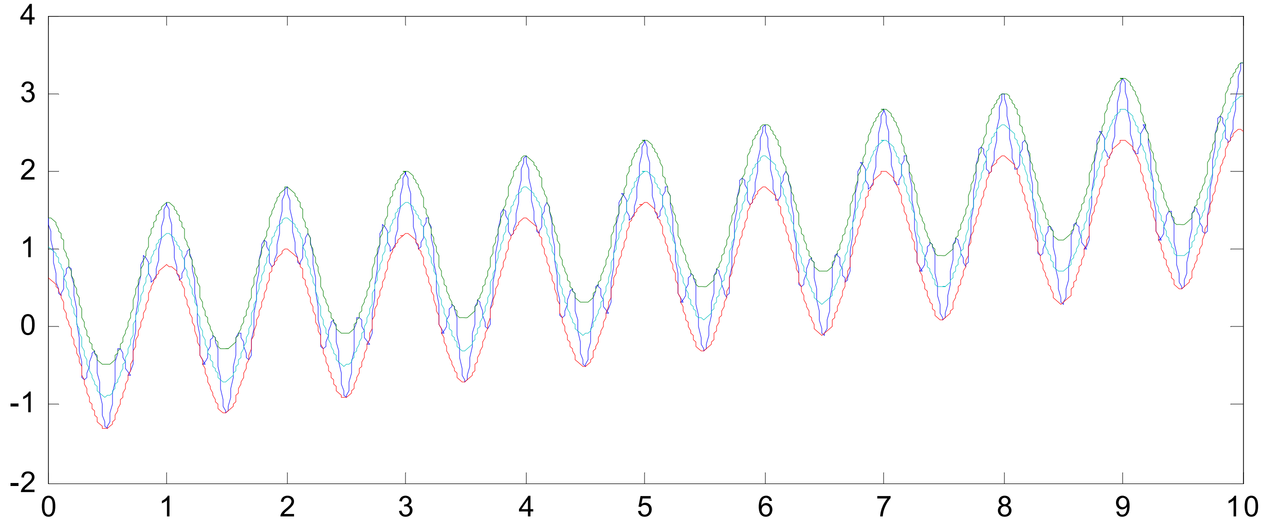


11-D Example

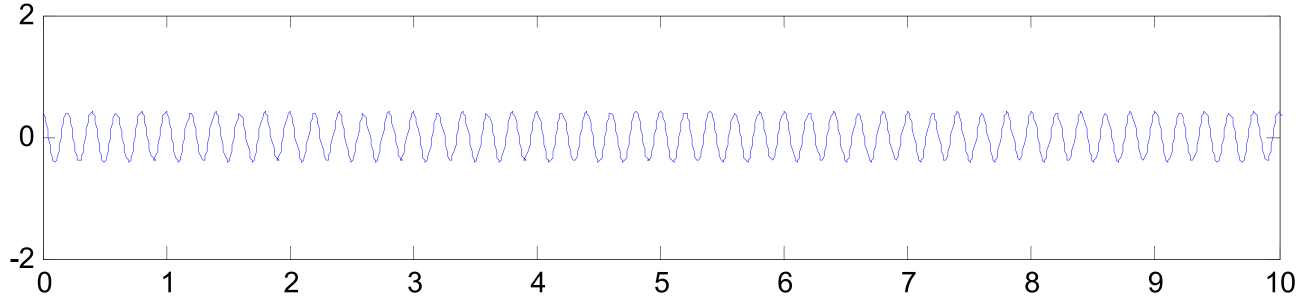
Example 1 $x(t) = 0.2t + \cos(2\pi t) + 0.4\cos(10\pi t)$



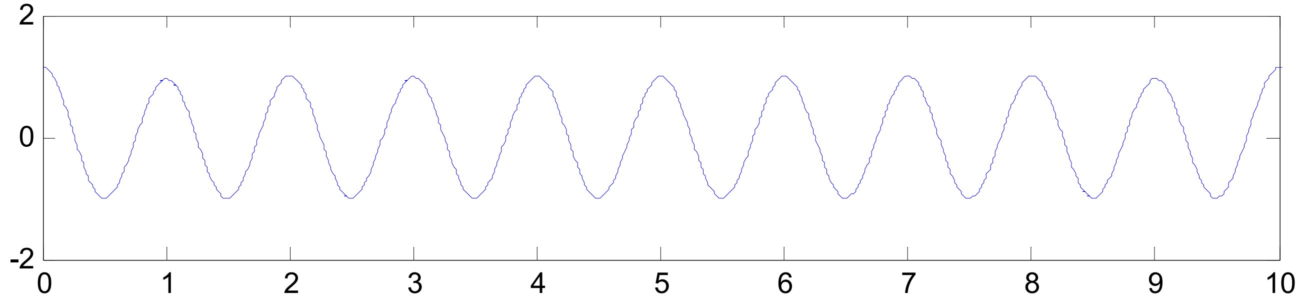
After Step 6



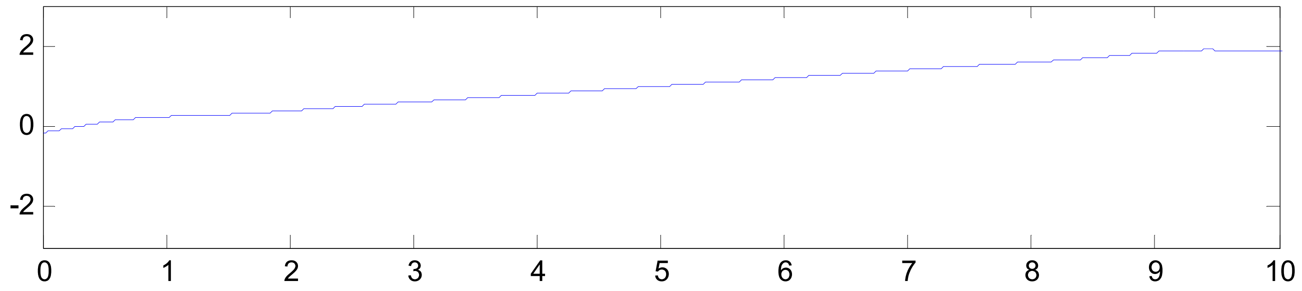
IMF1



IMF2

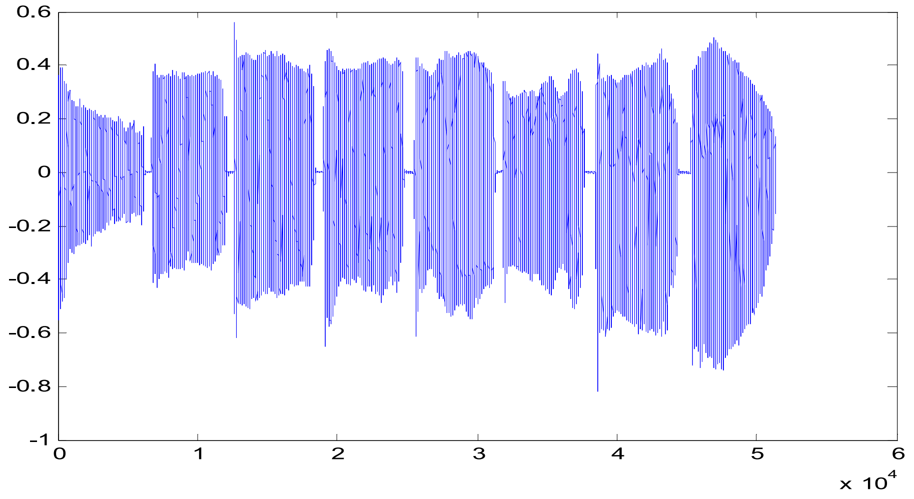


$x_0(t)$

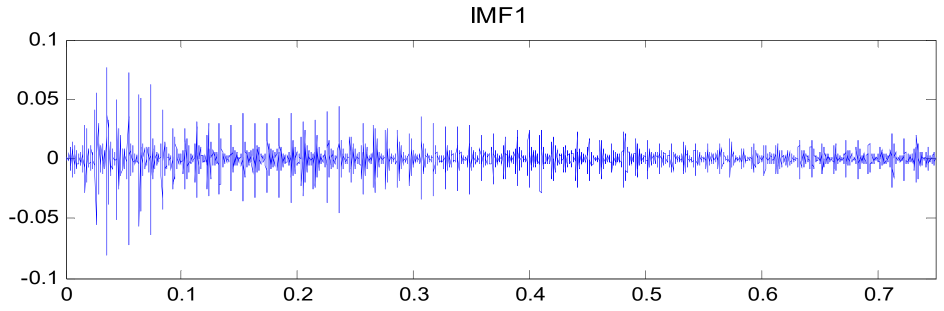


Example 2

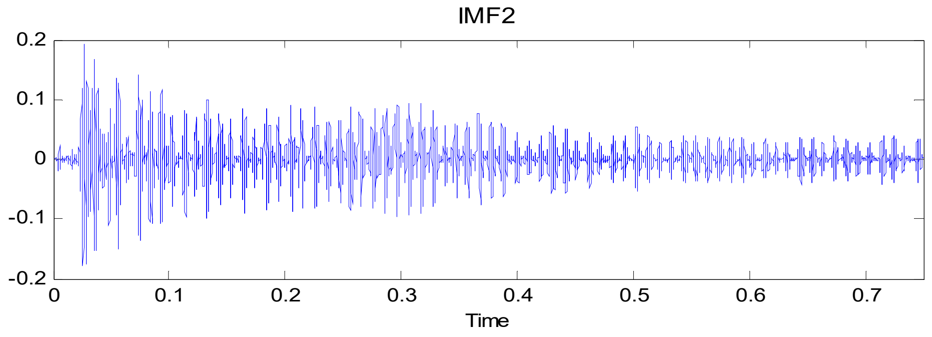
hum signal



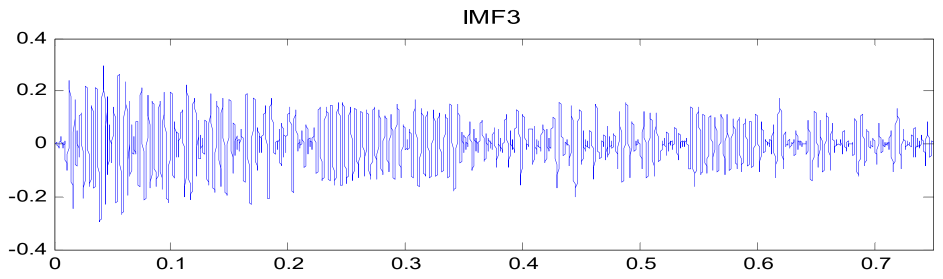
IMF1



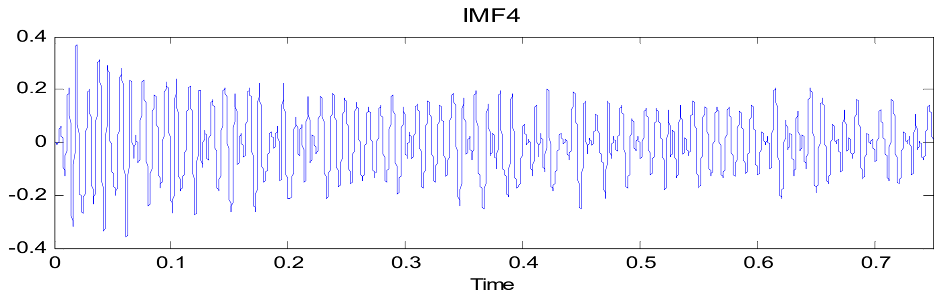
IMF2



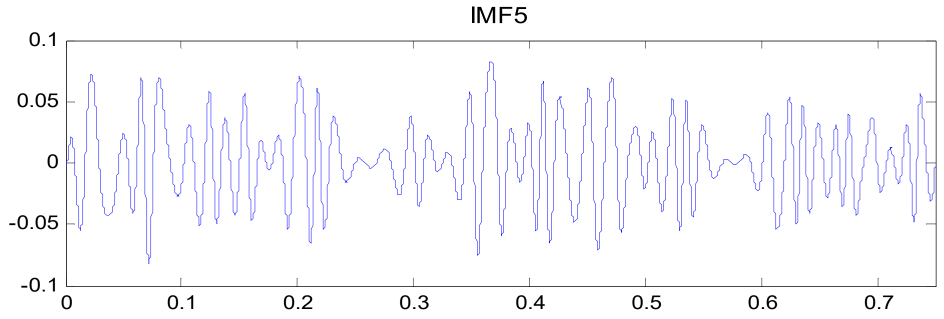
IMF3



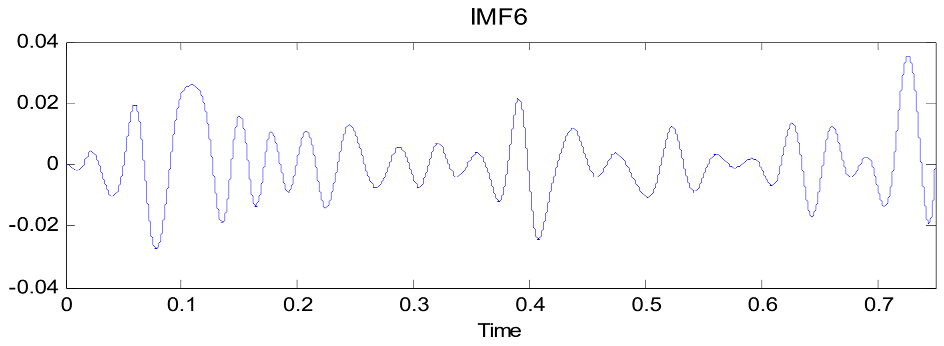
IMF4



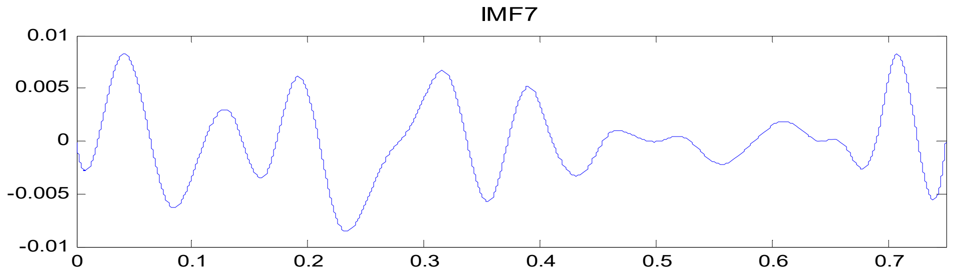
IMF5



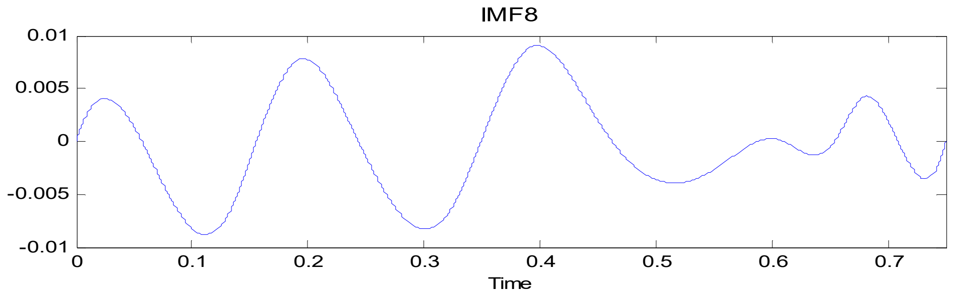
IMF6



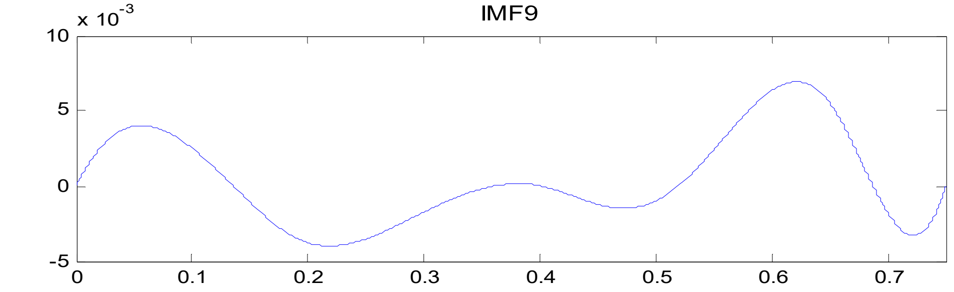
IMF7



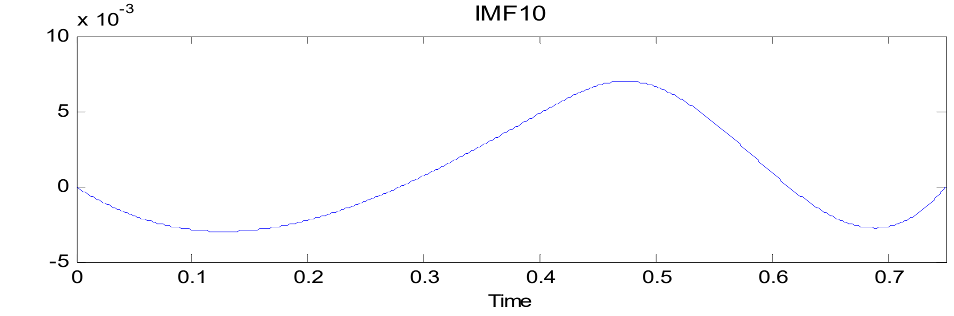
IMF8



IMF9

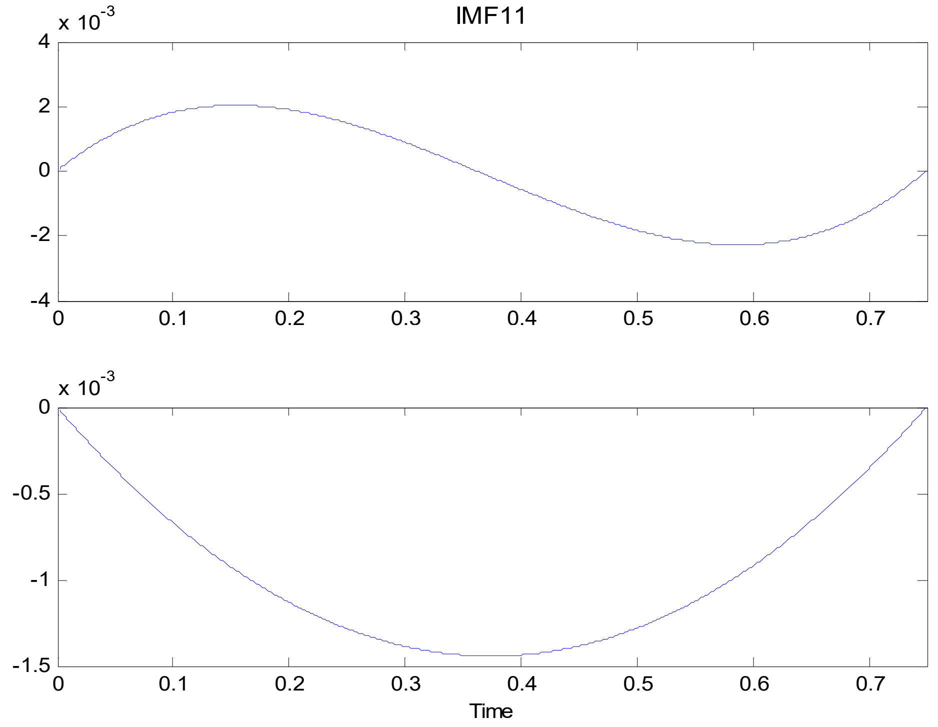


IMF10



IMF11

$x_0(t)$



11-E Comparison

(1) 避免了複雜的數學理論分析

(2) 可以找到一個 function 的「趨勢」

(3) 和其他的時頻分析一樣，可以分析頻率會隨著時間而改變的信號

(4) 適合於
- Climate analysis
- Economical data
- Geology
- Acoustics
- Music signal

- Conclusion

當信號含有「趨勢」

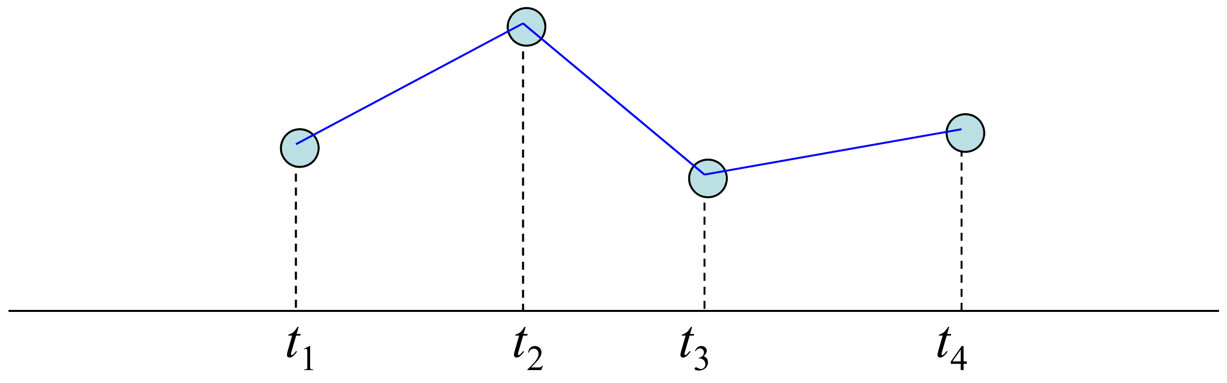
或是由少數幾個 sinusoid functions 所組合而成，而且這些sinusoid functions 的 amplitudes 相差懸殊時，可以用 HHT 來分析

附錄十一 Interpolation and the B-Spline

Suppose that the sampling points are t_1, t_2, t_3, …, t_N
and we have known the values of $x(t)$ at these sampling points.

There are several ways for interpolation.

(1) The simplest way: Using the straight lines (i.e., linear interpolation)



(2) Lagrange interpolation

$$x(t) = \sum_{n=1}^{N} \frac{\prod_{\substack{j=1 \\ j\neq n}}^{N} (t - t_j)}{\prod_{\substack{j=1 \\ j\neq n}}^{N} t_n - t_j} x(t_n)$$

$\prod$ 指的是連乘符號，

$$\prod_{j=1}^{N} h_j = h_1 h_2 h_3 \cdots\cdots h_N$$

Example: When $N = 4$,

$$x(t) = \frac{(t-t_2)(t-t_3)(t-t_4)}{(t_1-t_2)(t_1-t_3)(t_1-t_4)} x(t_1) + \frac{(t-t_1)(t-t_3)(t-t_4)}{(t_2-t_1)(t_2-t_3)(t_2-t_4)} x(t_2)$$

$$+ \frac{(t-t_1)(t-t_2)(t-t_4)}{(t_3-t_1)(t_3-t_2)(t_3-t_4)} x(t_3) + \frac{(t-t_1)(t-t_2)(t-t_3)}{(t_4-t_1)(t_4-t_2)(t_4-t_3)} x(t_4)$$

(3) Polynomial interpolation

$$x(t) = \sum_{n=1}^{N} a_n t^{n-1},$$ solve a_1, a_2, a_3, ……, $a_{N\text{-}1}$ from

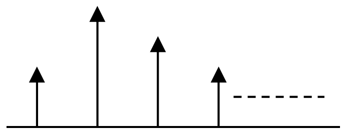
$$\begin{bmatrix} 1 & t_1 & t_1^2 & \cdots & t_1^{N-1} \\ 1 & t_2 & t_2^2 & \cdots & t_2^{N-1} \\ 1 & t_3 & t_3^2 & \cdots & t_3^{N-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & t_N & t_N^2 & \cdots & t_N^{N-1} \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ \vdots \\ a_N \end{bmatrix} = \begin{bmatrix} x(t_1) \\ x(t_2) \\ x(t_3) \\ \vdots \\ x(t_N) \end{bmatrix}$$

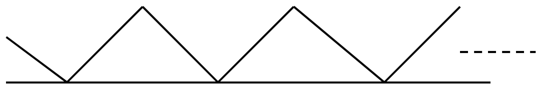
(4) Lowpass Filter Interpolation

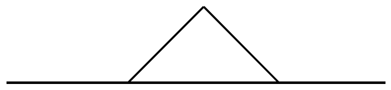
適用於 sampling interval 為固定的情形 $t_{n+1} - t_n = \Delta_t$ for all n

$$x(t) = \sum_{n=1}^{N} x(t_n)\,\mathrm{sinc}\left(\frac{t-t_n}{\Delta_t}\right)$$

$$\begin{tikzcd}
x(t_n) \arrow[r, "\text{discrete time Fourier transform}"] & X_1(f) \arrow[r, "\text{lowpass mask}"] & X(f) \arrow[d, "\begin{array}{c}\text{inverse discrete time}\\\text{Fourier transform}\end{array}"'] \\
& & x(t)
\end{tikzcd}$$







(5) B-Spline Interpolation

B-spline 簡稱為 spline

$$B_{n,0}(t) = 1 \quad \text{for } t_n < t < t_{n+1}$$

$$B_{n,0}(t) = 0 \quad \text{otherwise}$$

$$B_{n,m}(t) = \frac{t - t_n}{t_{n+m} - t_n} B_{n,m-1}(t) + \frac{t_{n+m+1} - t}{t_{n+m+1} - t_{n+1}} B_{n+1,m-1}(t)$$

for $t_n < t < t_{n+m+1}$

$$\boxed{x(t) = \sum_{n=1}^{N} x(t_n) B_{n,m}(t)}$$

$m = 1$: linear B-spline

$m = 2$: quadratic B-spline

$m = 3$: cubic B-spline (通常使用) $x(t)$, $x'(t)$, $x''(t)$ are continuous

In **Matlab**，the command “spline” can be used for spline interpolation.

(Note： In the command, the cubic B-spline is used)

Cubic B-Spline Interpolation by Matlab:

Generating a sine-like spline curve and samples it over a finer mesh:

```
x = 0:1:10;         % original sampling points
y = sin(x);         % values of the function at x
xx = 0:0.1:10.2;    % new sampling points
yy = spline(x,y,xx);
plot(x,y,'o',xx,yy)
```

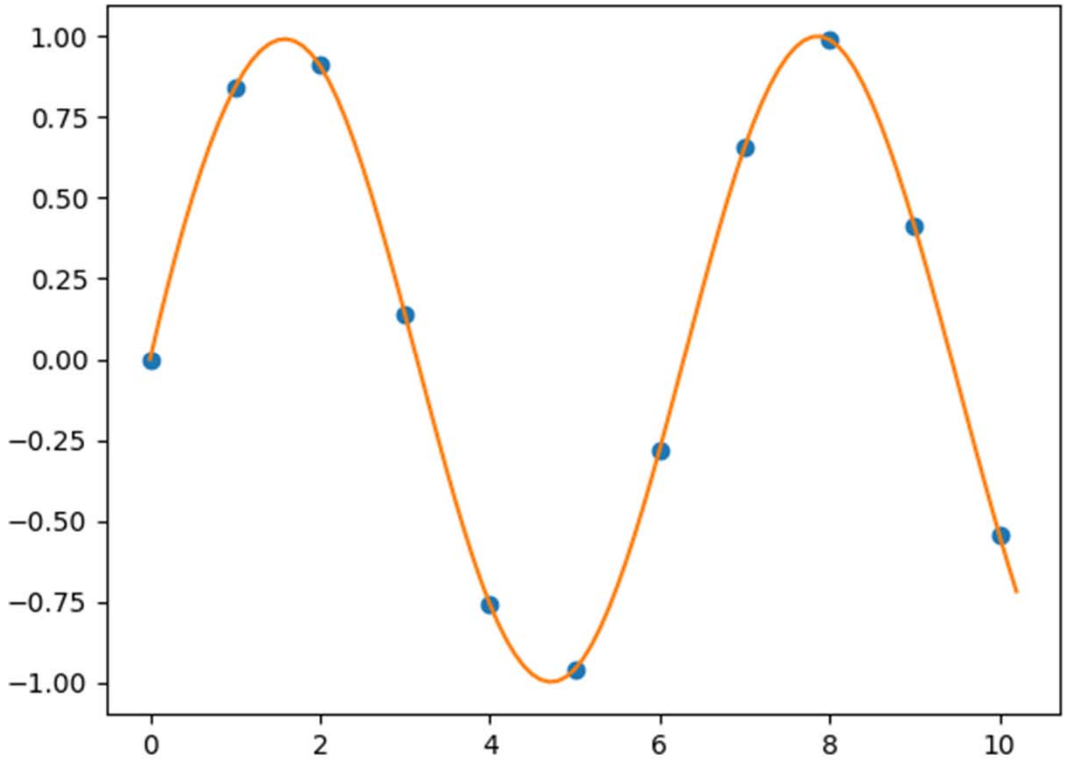
In **Python**, we can use the following way to perform cubic B-spline interpolation.

事前安裝模組

pip install numpy

pip install scipy

pip install matplotlib



Reference :
https://docs.scipy.org/doc/scipy/reference/reference/generated/scipy.interpolate.interp1d.html#scipy.interpolate.interp1d

Cubic B-Spline Interpolation by Python

```
import numpy as np
import scipy.interpolate as interpolate
import matplotlib.pyplot as plt
x = np.arange(0, 11)    # original sample points, [0, 1, 2, …, 9, 10]
y = np.sin(x)
t, c, k = interpolate.splrep(x, y, k=3)
x_new = np.arange(0, 10.3, 0.1)
# new sample points, [0, 0.1, 0.2, ….., 10, 10.1, 10.2]
f = interpolate.BSpline(t, c, k)
y_new = f(x_new)
plt.plot(x,y,'o',x_new, y_new)
plt.show()
```