IX. Basic Implementation Techniques and Fast Algorithm

◉ 9-A 快速演算法設計的原則

- **Fast Algorithm Design**

Goals: Saving Computational Time

Number of Additions

Number of Multiplications

Number of Time Cycles

Saving the Hardware Cost for Implementation

Saving the buffer size

Repeated Using a Structure

Four important concepts that should be learned from fast algorithm design:

(1) N-point DFT

(2) Complexity of LTI Systems

(3) Replacement of DFTs

(4) Simplification Techniques

◉ 9-B 對於簡單矩陣快速演算法的設計

如何簡化下面四個運算

(1) $y_1 = ax_1 + 2ax_2$

(2) $\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} a & a \\ a & a \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$

(3) $\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} a & b \\ b & a \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$

(4) $\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} a & b \\ c & a \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$

(4)

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} a & b \\ c & a \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} a & a \\ a & a \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 & b-a \\ c-a & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

$$\begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = \begin{bmatrix} a & a \\ a & a \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \qquad \begin{bmatrix} z_3 \\ z_4 \end{bmatrix} = \begin{bmatrix} 0 & b-a \\ c-a & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

(i) $z_1 = a[x_1 + x_2], \quad z_2 = z_1$

(ii) $z_3 = (b-a)x_2, \qquad z_4 = (c-a)x_1$

(iii) $y_1 = z_1 + z_3, \qquad y_2 = z_2 + z_4$

問題思考：如何對 complex number multiplication 來做 implementation？

◉ 9-C General Way for Simplifying Calculation

假設一個 $M \times N$ sub-rectangular matrix **S** 可分解為 column vector 及 row vector 相乘

$$\mathbf{S} = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_M \end{bmatrix} \begin{bmatrix} b_1 & b_2 & \cdots & b_N \end{bmatrix}$$

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} = S \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix}$$

$$\begin{cases} z = b_1 x_1 + b_2 x_2 + \ldots + b_N x_N \\ y_n = a_n z \end{cases}$$

若 $[a_1, a_2, \ldots., a_M]^T$ 有 M_0 個相異的 non-trivial values

$$(a_m \neq \pm 2^k, \quad a_m \neq \pm 2^k a_h \text{ where } m \neq h)$$

$[b_1, b_2, \ldots., b_N]$ 有 N_0 個相異的 non-trivial values

則 **S** 共需要 $M_0 + N_0$ 個乘法

$$\begin{bmatrix} z[1] \\ z[2] \\ \vdots \\ z[N] \end{bmatrix} = \mathbf{S} \begin{bmatrix} x[1] \\ x[2] \\ \vdots \\ x[N] \end{bmatrix} = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_M \end{bmatrix} \begin{bmatrix} b_1 & b_2 & \cdots & b_N \end{bmatrix} \begin{bmatrix} x[1] \\ x[2] \\ \vdots \\ x[N] \end{bmatrix}$$

Step 1 $z_a = b_1 x[1] + b_2 x[2] + \ldots + b_N x[N]$

Step 2 $z[1] = a_1 z_a$, $z[2] = a_2 z_a$, ……, $z[N] = a_M z_a$

簡化理論的變型

$$\mathbf{S} = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_M \end{bmatrix} \begin{bmatrix} b_1 & b_2 & \cdots & b_N \end{bmatrix} + \mathbf{S_1}$$

$\mathbf{S_1}$ 也是一個 $M \times N$ matrix

若 $\mathbf{S_1}$ 有 P_1 個值不等於 0, 則 S 的乘法量上限為 $M_0 + N_0 + P_1$

$$\mathbf{S} = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_M \end{bmatrix} \begin{bmatrix} b_1 & b_2 & \cdots & b_N \end{bmatrix} + \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_M \end{bmatrix} \begin{bmatrix} d_1 & d_2 & \cdots & d_N \end{bmatrix} + \mathbf{S_1}$$

以此類推

思考：對於如下的情形需要多少乘法

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} = \begin{bmatrix} a & b & c & d \\ e & f & f & e \\ f & e & e & f \\ d & c & b & a \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}$$

◉ 9-D Examples

DFT: $$X[m]=\sum_{n=0}^{N-1}x[n]e^{-j\frac{2\pi mn}{N}}$$

Without any simplification, the DFT needs $4N^2$ real multiplications ($x[n]$ may be complex)

- 3×3 DFT 可以用特殊方法簡化

$$\begin{bmatrix}1 & 1 & 1\\ 1 & -1/2 & -1/2\\ 1 & -1/2 & -1/2\end{bmatrix}+j\begin{bmatrix}0 & 0 & 0\\ 0 & -\sqrt{3}/2 & \sqrt{3}/2\\ 0 & \sqrt{3}/2 & -\sqrt{3}/2\end{bmatrix}$$

- 5×5 DFT 的例子

real part **imaginary part**

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & a & b & b & a \\ 1 & b & a & a & b \\ 1 & b & a & a & b \\ 1 & a & b & b & a \end{bmatrix} - j \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & c & d & -d & -c \\ 0 & d & -c & c & -d \\ 0 & -d & c & -c & d \\ 0 & -c & -d & d & c \end{bmatrix}$$

$$a = \cos(2\pi/5)$$

$$b = \cos(4\pi/5)$$

$$c = \sin(2\pi/5)$$

$$d = \sin(4\pi/5)$$

8-point DCT

$$\begin{bmatrix} y[0] \\ y[1] \\ y[2] \\ y[3] \\ y[4] \\ y[5] \\ y[6] \\ y[7] \end{bmatrix} = \begin{bmatrix} 0.7010 & 0.7010 & 0.7010 & 0.7010 & 0.7010 & 0.7010 & 0.7010 & 0.7010 \\ 0.9808 & 0.8315 & 0.5556 & 0.1951 & -0.1951 & -0.5556 & -0.8315 & -0.9808 \\ 0.9239 & 0.3827 & -0.3827 & -0.9239 & -0.9239 & -0.3827 & 0.3827 & 0.9239 \\ 0.8315 & -0.1951 & -0.9808 & -0.5556 & 0.5556 & 0.9808 & 0.1951 & -0.8315 \\ 0.7010 & -0.7010 & -0.7010 & 0.7010 & 0.7010 & -0.7010 & -0.7010 & 0.7010 \\ 0.5556 & -0.9808 & 0.1951 & 0.8315 & -0.8315 & -0.1951 & 0.9808 & -0.5556 \\ 0.3827 & -0.9239 & 0.9239 & -0.3827 & -0.3827 & 0.9239 & -0.9239 & 0.3827 \\ 0.1951 & -0.5556 & 0.8315 & -0.9808 & 0.9808 & -0.8315 & 0.5556 & -0.1951 \end{bmatrix} \begin{bmatrix} x[0] \\ x[1] \\ x[2] \\ x[3] \\ x[4] \\ x[5] \\ x[6] \\ x[7] \end{bmatrix}$$

觀察對稱性質之後，令

$$\begin{bmatrix} z[0] \\ z[1] \\ z[2] \\ z[3] \end{bmatrix} = \begin{bmatrix} x[0]+x[7] \\ x[1]+x[6] \\ x[2]+x[5] \\ x[3]+x[4] \end{bmatrix} \qquad \begin{bmatrix} z[4] \\ z[5] \\ z[6] \\ z[7] \end{bmatrix} = \begin{bmatrix} x[0]-x[7] \\ x[1]-x[6] \\ x[2]-x[5] \\ x[3]-x[4] \end{bmatrix}$$

Part 1:
$$\begin{bmatrix} y[0] \\ y[2] \\ y[4] \\ y[6] \end{bmatrix} = \begin{bmatrix} 0.7010 & 0.7010 & 0.7010 & 0.7010 \\ 0.9239 & 0.3827 & -0.3827 & -0.9239 \\ 0.7010 & -0.7010 & -0.7010 & 0.7010 \\ 0.3827 & -0.9239 & 0.9239 & -0.3827 \end{bmatrix} \begin{bmatrix} z[0] \\ z[1] \\ z[2] \\ z[3] \end{bmatrix}$$

$$\begin{bmatrix} z[0] \\ z[1] \\ z[2] \\ z[3] \end{bmatrix} = \begin{bmatrix} x[0]+x[7] \\ x[1]+x[6] \\ x[2]+x[5] \\ x[3]+x[4] \end{bmatrix}$$

Part 2:
$$\begin{bmatrix} y[1] \\ y[3] \\ y[5] \\ y[7] \end{bmatrix} = \begin{bmatrix} 0.9808 & 0.8315 & 0.5556 & 0.1951 \\ 0.8315 & -0.1951 & -0.9808 & -0.5556 \\ 0.5556 & -0.9808 & 0.1951 & 0.8315 \\ 0.1951 & -0.5556 & 0.8315 & -0.9808 \end{bmatrix} \begin{bmatrix} z[4] \\ z[5] \\ z[6] \\ z[7] \end{bmatrix}$$

$$\begin{bmatrix} z[4] \\ z[5] \\ z[6] \\ z[7] \end{bmatrix} = \begin{bmatrix} x[0]-x[7] \\ x[1]-x[6] \\ x[2]-x[5] \\ x[3]-x[4] \end{bmatrix}$$

[Ref] B. G. Lee, “A new algorithm for computing the discrete cosine transform,” *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 32, pp. 1243-1245, Dec. 1984.

◉ 9-E Summary of the Complexity

- N-point DFT: $O(N \log_2 N)$
- N-point DCT, DST, DHT: $O(N \log_2 N)$
- Two-dimensional (2-D) $N_x \times N_y$-point DFT: $O((N_x N_y)\log_2(N_x N_y))$ Why?
- Convolution of an M-point sequence and an N-point sequence:

$O((M+N-1)\log_2(M+N-1))$ when M/N and N/M are not large,

$O(N)$ when $N >> M$ and M is a fixed constant.

$O(M)$ when $M >> N$ and N is a fixed constant.

- 2-D Convolution of an $(M_x \times M_y)$-point matrix and an $(N_x \times N_y)$-point matrix:

$$O\Big((M_x + N_x - 1)(M_y + N_y - 1)\log_2\big((M_x + N_x - 1)(M_y + N_y - 1)\big)\Big)$$

when M_xM_y/N_xN_y and N_xN_y/M_xM_y are not large,

$O(M_xM_y)$ when $M_xM_y >> N_xN_y$

$O(N_xN_y)$ when $N_xN_y >> M_xM_y$,
and Mx, M_y are fixed constants.

X. Fast Fourier Transform

- C. S. Burrus and T. W. Parks, “DFT / FFT and convolution algorithms”, John Wiley and Sons, New York, 1985.
- R. E. Blahut, *Fast Algorithm for Digital Signal Processing*, Addison Wesley Publishing Company.

$$X[m]=\sum_{n=0}^{N-1}x[n]e^{-j\frac{2\pi mn}{N}}$$

N-point Fourier Transform:　運算量為 N^2

FFT (with the Cooley Tukey algorithm):　運算量為 $N\log N$

要學到的概念：(1)快速演算法不是只有 Cooley Tukey algorithm

(2) 不是只有 $N=2^k$ 有時候才有快速演算法

◉ 10-A Other DFT Implementation Algorithms

(1) Cooley-Tukey algorithm (Butterfly form)

(2) Radix-4, 8, 16, …. Algorithms

(3) Prime Factor Algorithm

(4) Goertzel Algorithm

(5) Chirp Z transform (CZT)

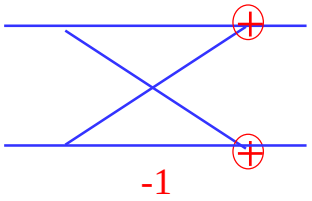
(6) Winograd algorithm

Reference

- J. W. Cooley and J. W. Tukey, "An algorithm for the machine computation of complex Fourier series," *Mathematics of Computation*, vol. 19, pp. 297-301, Apr. 1965. (Cooley-Tukey)
- C. S. Burrus, "Index Mappings for multidimensional formulation of the DFT and convolution," *IEEE Trans. Acoustics, Speech, and Signal Processing*, vol. 25, pp. 1239-242, June 1977. (Prime factor)
- G. Goertzel, "An algorithm for the evaluation of finite trigonometric series," *American Math. Monthly*, vol. 65, pp. 34-35, Jan. 1958. (Goertzel)
- C. R. Hewes, R. W. Broderson, and D. D. Buss, "Applications of CCD and switched capacitor filter technology," *Proc. IEEE*, vol. 67, no. 10, pp. 1403-1415, Oct. 1979. (CZT)
- S. Winograd, "On computing the discrete Fourier transform," *Mathematics of Computation*, vol. 32, no. 141, pp. 179-199, Jan. 1978. (Winograd)
- R. E. Blahut, *Fast Algorithm for Digital Signal Processing*, Reading, Mass., Addison-Wesley, 1985.

◉ 10-B Cooley Tukey Algorithm

When $N = 2^k$



$$X[m] = \sum_{n=0}^{N-1} x[n] e^{-j\frac{2\pi mn}{N}}$$

$$= \sum_{n=0}^{N/2-1} x[2n] e^{-j\frac{2\pi m(2n)}{N}} + \sum_{n=0}^{N/2-1} x[2n+1] e^{-j\frac{2\pi m(2n+1)}{N}}$$

$$= \sum_{n=0}^{N/2-1} x_1[n] e^{-j\frac{2\pi mn}{N/2}} + e^{-j\frac{2\pi m}{N}} \sum_{n=0}^{N/2-1} x_2[n] e^{-j\frac{2\pi mn}{N/2}}$$

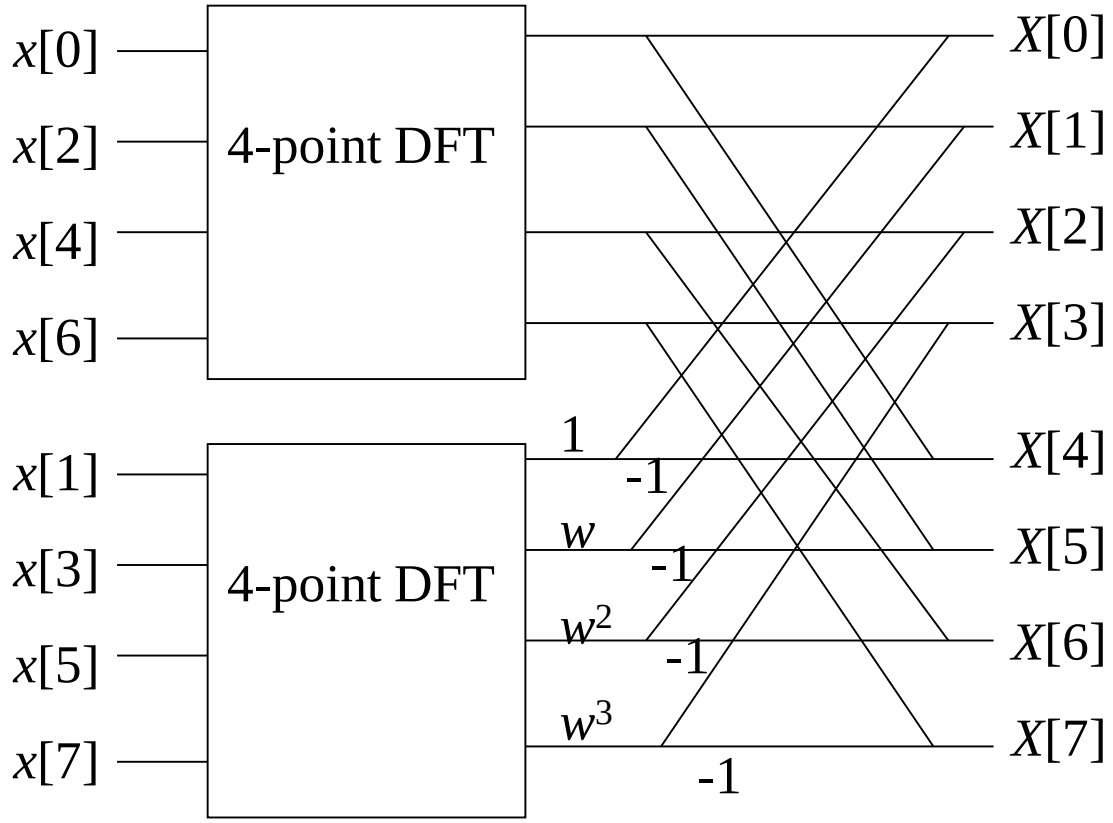
twiddle factors

$$x_1[n] = x[2n], \quad x_2[n] = x[2n+1]$$

Therefore,

one N-point DFT = two $(N/2)$-point DFT + twiddle factors

8-point DFT



$$w = e^{-j\frac{2\pi}{N}}$$

$$w^{(m+\frac{N}{2})} = e^{-j\frac{2\pi}{N}(m+\frac{N}{2})} = e^{-j\frac{2\pi}{N}m} e^{-j\frac{2\pi}{N}\frac{N}{2}} = -e^{-j\frac{2\pi}{N}m} = -w^m$$

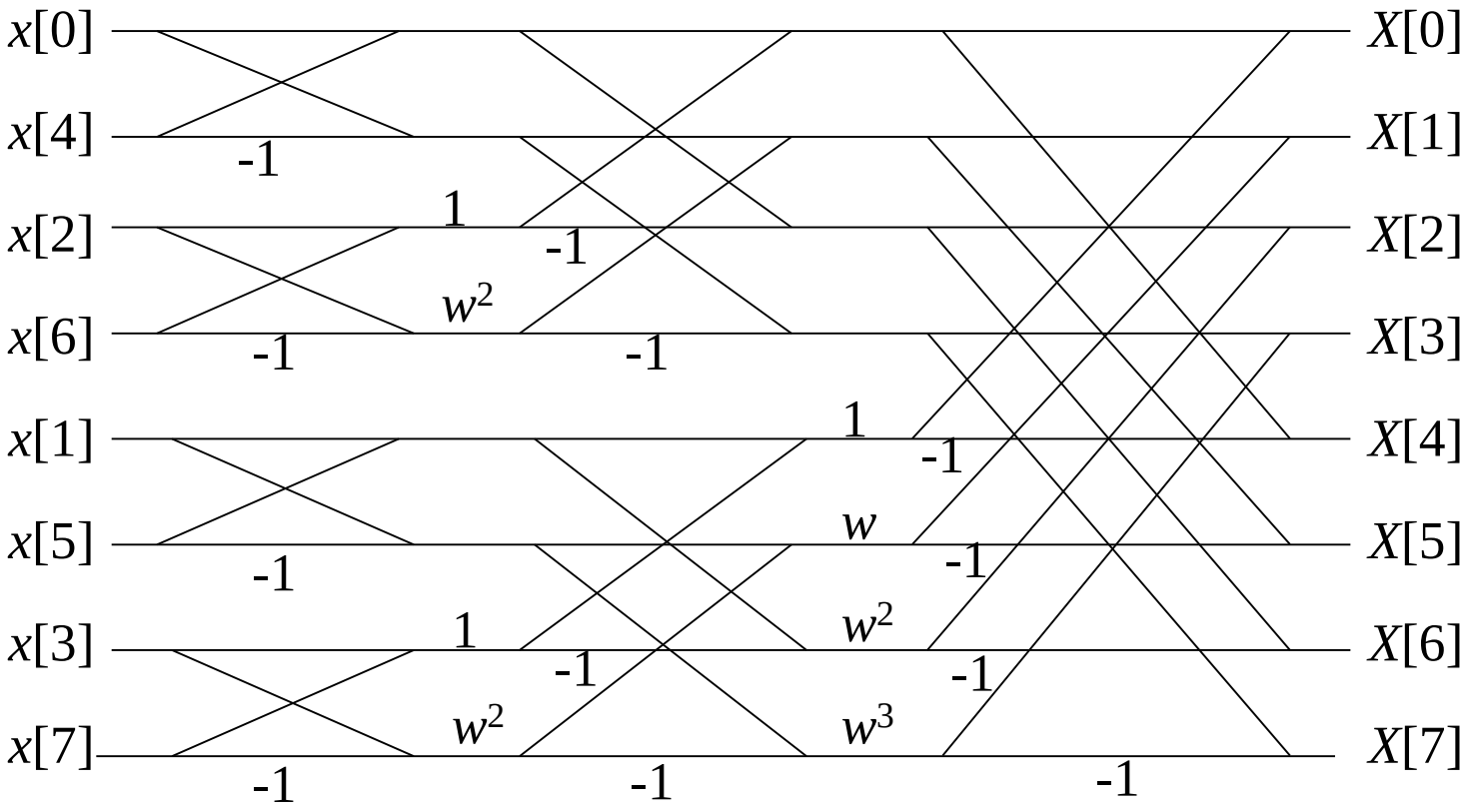
When $N = 8$

$$w = e^{-j\frac{2\pi}{8}}$$

$$w^4 = -1$$

$$w^8 = 1$$

8-point DFT



$$w = e^{-j\frac{2\pi}{8}}$$

- Number of real multiplications 的估算

2^k-point DFT 一共有 k 個 stages　　k-1 次 decomposition

每個 stage 和下一個 stage 之間有 2^{k-1} 個 twiddle factors

所以，一共有 $2^{k-1}(k-1)$ 個 twiddle factors

一般而言，每個 twiddle factor 需要 3 個 real multiplications

∴ 2^k-point DFT 需要

$$3\left(2^{k-1}(k-1)\right)=\frac{3}{2}N\left(\log_2 N-1\right)$$ 個 real multiplications

Complexity of the N-point DFT: $O(N\log_2 N)$

- 8-point DFT 只需要 4 個 real multiplications (Why?)

- 更精確的分析，使用 Cooley-Tukey algorithm 時，N-point DFT 需要

$$\frac{3}{2}N\log_2 N - 5N + 8$$ 個 real multiplications

(Why?)

◉ 10-C Radix-4 Algorithm

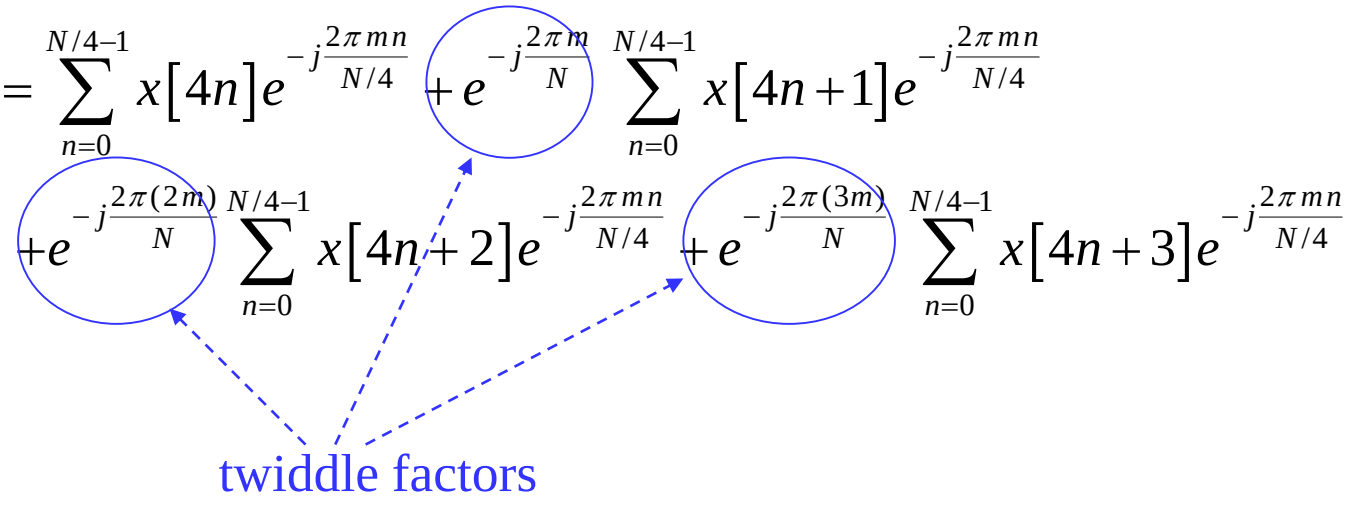
限制： $N = 4^k$

or $N = 2 \cdot 4^k$ (此時 Cooley-Tukey algorithm 和 radix-4 algorithm 並用)

$$X[m] = \sum_{n=0}^{N-1} x[n] e^{-j\frac{2\pi mn}{N}}$$

$$= \sum_{n=0}^{N/4-1} x[4n] e^{-j\frac{2\pi mn}{N/4}} + e^{-j\frac{2\pi m}{N}} \sum_{n=0}^{N/4-1} x[4n+1] e^{-j\frac{2\pi mn}{N/4}}$$

$$+ e^{-j\frac{2\pi(2m)}{N}} \sum_{n=0}^{N/4-1} x[4n+2] e^{-j\frac{2\pi mn}{N/4}} + e^{-j\frac{2\pi(3m)}{N}} \sum_{n=0}^{N/4-1} x[4n+3] e^{-j\frac{2\pi mn}{N/4}}$$



One N-point DFT = four $(N/4)$-point DFTs + twiddle factors

Note:

(1) radix-4 algorithm 最後可將 $N = 4^k$-point DFT 拆解成 4-point DFTs 的組合

4-point DFTs 不需要任何的乘法

(2) 使用 radix-4 algorithm 時，N-point DFT 需要

$$\frac{9}{4} N \log_4 N - \frac{43}{12} N + \frac{16}{3}$$ 個 real multiplications

- Number of real multiplications for the N-point DFT

N	乘法數	加法數	N	乘法數	N	乘法數	N	乘法數
1	0	0	11	46	24	28	39	182
2	0	4	12	8	25	148	40	100
3	2	12	13	52	26	104	42	124
4	0	16	14	32	27	114	44	160
5	10	34	15	40	28	64	45	170
6	4	36	16	20	30	80	48	92
7	16	72	18	32	32	72	52	208
8	4	52	20	40	33	160	54	228
9	16	72	21	62	35	150	56	156
10	20	88	22	80	36	64	60	160

N	乗法數	N	乗法數	N	乗法數	N	乗法數
63	256	96	280	192	752	360	1540
64	204	104	468	204	976	420	2080
66	284	108	456	216	1020	480	2360
70	300	112	396	224	1016	504	2300
72	164	120	380	240	940	512	3180
80	260	128	560	252	1024	560	3100
81	480	144	436	256	1308	672	3496
84	248	160	680	288	1160	720	3620
88	412	168	580	312	1324	784	4412
90	340	180	680	336	1412	840	4580

N	乘法數	*N*	乘法數	*N*	乘法數	*N*	乘法數
1008	5356	1440	8680	2520	16540	4032	29488
1024	7436	1680	10420	2688	19108	4096	37516
1152	7088	2016	12728	2880	20060	4368	35828
1260	7640	2048	16836	3369	24200	4608	36812
1344	8252	2304	15868	3920	29900	5040	36860

附錄九：新的相似度測量工具：結構相似度 Structural Similarity (SSIM)

傳統量測兩個信號 (including images, videos, and vocal signals) 之間相似度的方式：

(1) maximal error $Max\left(\left|y[m,n]-x[m,n]\right|\right)$

(2) mean square error (MSE) $\frac{1}{MN}\sum_{m=0}^{M-1}\sum_{n=0}^{N-1}\left|y[m,n]-x[m,n]\right|^2$

(3) normalized mean square error (NMSE) $\frac{\sum_{m=0}^{M-1}\sum_{n=0}^{N-1}\left|y[m,n]-x[m,n]\right|^2}{\sum_{m=0}^{M-1}\sum_{n=0}^{N-1}\left|x[m,n]\right|^2}$

(4) normalized root mean square error (NRMSE) $\sqrt{\frac{\sum_{m=0}^{M-1}\sum_{n=0}^{N-1}\left|y[m,n]-x[m,n]\right|^2}{\sum_{m=0}^{M-1}\sum_{n=0}^{N-1}\left|x[m,n]\right|^2}}$

(5) L_α-Norm

$$\|y-x\|_\alpha = \left(\sum_{m=0}^{M-1} \sum_{n=0}^{N-1} |y[m,n]-x[m,n]|^\alpha \right)^{1/\alpha}$$

$$\frac{1}{MN}\|y-x\|_\alpha = \frac{1}{MN}\left(\sum_{m=0}^{M-1} \sum_{n=0}^{N-1} |y[m,n]-x[m,n]|^\alpha \right)^{1/\alpha}$$

(6) signal to noise ratio (SNR)，信號處理常用

$$10\log_{10}\left(\frac{\sum_{m=0}^{M-1} \sum_{n=0}^{N-1} |x[m,n]|^2}{\sum_{m=0}^{M-1} \sum_{n=0}^{N-1} |y[m,n]-x[m,n]|^2} \right)$$

(7) peak signal to noise ratio (PSNR)，影像處理常用

$$10\log_{10}\left(\frac{X_{Max}^2}{\frac{1}{MN}\sum_{m=0}^{M-1}\sum_{n=0}^{N-1}\left|y[m,n]-x[m,n]\right|^2}\right)$$

X_{Max}: the maximal possible value of $x[m, n]$

In image processing, $X_{Max} = 255$

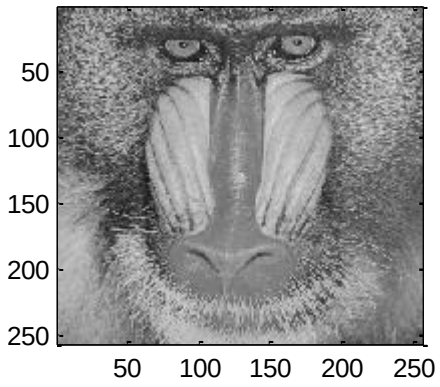
for color image:

$$10\log_{10}\left(\frac{X_{Max}^2}{\frac{1}{3MN}\sum_{R,G,B}\sum_{m=0}^{M-1}\sum_{n=0}^{N-1}\left|y_{color}[m,n]-x_{color}[m,n]\right|^2}\right)$$

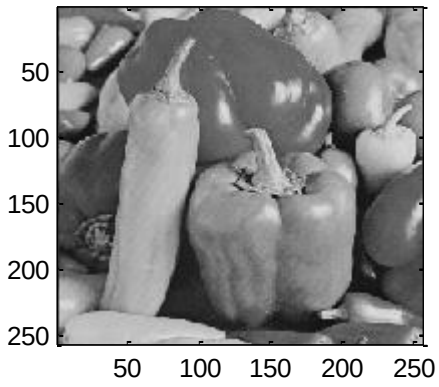
color = R, G, or B

然而，MSE 和 NRMSE 雖然在理論上是合理的，但卻無法反應出實際上兩個信號之間的相似度

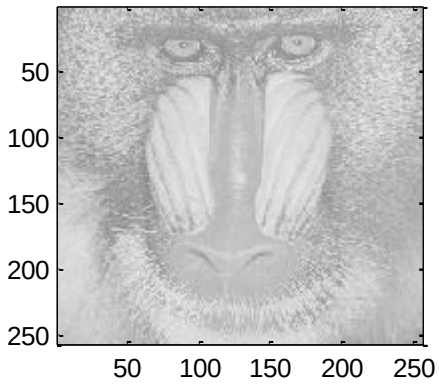
例如：以下這三張圖



圖一



圖二



圖三

圖三 = 圖一 × 0.5 + 255.5 × 0.5

照理來說，圖一和圖三較相近

然而，圖一和圖二之間的 NRMSE 為 0.4411

圖一和圖三之間的 NRMSE 為 0.4460

(8) Structural Similarity (SSIM)

有鑑於 MSE 和 PSNR 無法完全反應人類視覺上所感受的誤差，在 2004 年被提出來的新的誤差測量方法

$$SSIM(x,y) = \frac{\left(2\mu_x\mu_y + (c_1L)^2\right)}{\left(\mu_x^2 + \mu_y^2 + (c_1L)^2\right)} \frac{\left(2\sigma_{xy} + (c_2L)^2\right)}{\left(\sigma_x^2 + \sigma_y^2 + (c_2L)^2\right)}$$

$$DSSIM(x,y) = 1 - SSIM(x,y)$$

μ_x, μ_y: means of x and y σ_x^2, σ_y^2: variances of x and y

σ_{xy}: covariance of x and y c_1, c_2: adjustable constants

L: the maximal possible value of x − the minimal possible value of x

Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, “Image quality assessment: from error visibility to structural similarity,” *IEEE Trans. Image Processing*, vol. 13, no. 4, pp. 600−612, Apr. 2004.

若使用 SSIM，且前頁的 c_1, c_2 皆選為 1

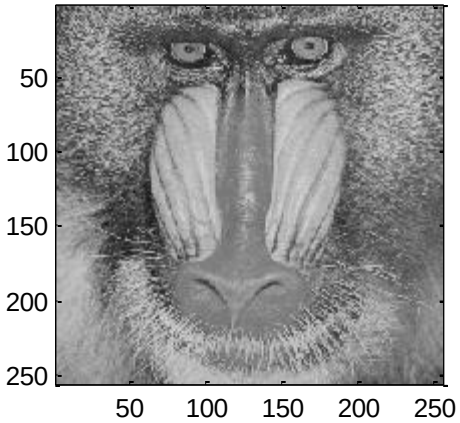
圖一、圖二之間的 SSIM 為 0.1040

圖一、圖三之間的 SSIM 為 0.7720

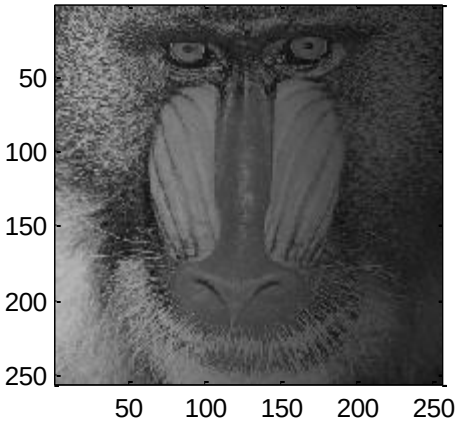
反應出了圖一、圖三之間確實有很高的相似度

其他幾個用 MSE 和 NRMSE 無法看出相似度，但是可以用 SSIM 看出相似度的情形

影子 shadow



圖四

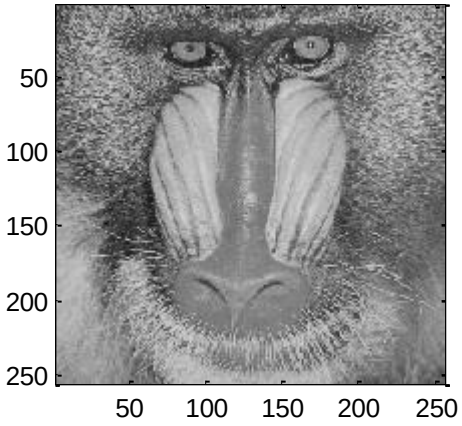


圖五

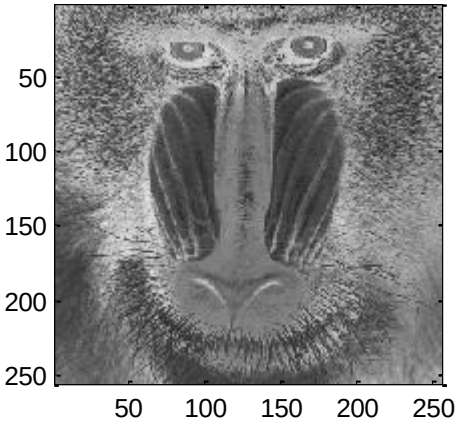
NRMSE = 0.4521 (大於圖一、圖二之間的 NRMSE)

SSIM = 0.6010

底片 the negative of a photo



圖六



圖七

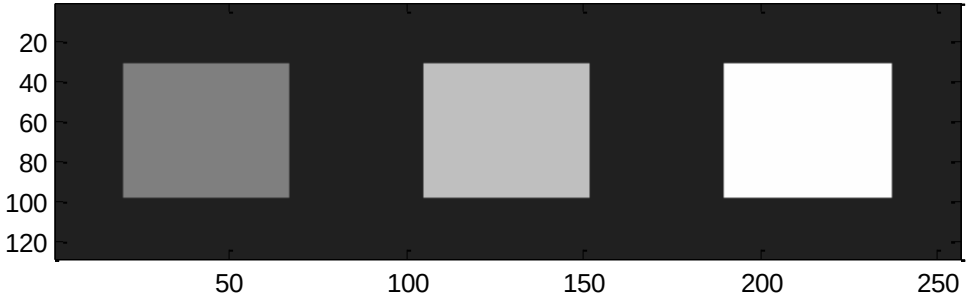
圖七 = 255 − 圖六

NRMSE = 0.5616 (大於圖一、圖二之間的 NRMSE)

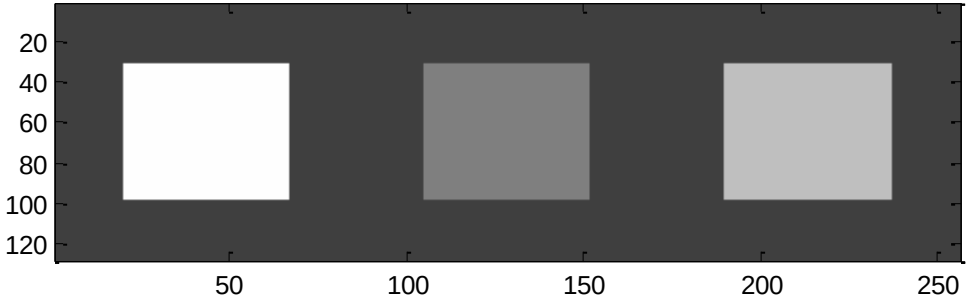
SSIM = -0.8367 (高度負相關)

同形，但亮度不同 (Same shape but different intensity)

圖八



圖九



NRMSE = 0.4978 (大於圖一、圖二之間的 NRMSE)

SSIM = 0.7333

思考：對於 vocal signal (聲音信號而言)

MSE 和 NRMSE 是否真的能反應出兩個信號的相似度？

為什麼？