DSP Last HW　　機械系 0811045 黃名諄

以原始 DFT 算法

$$X[k]=\sum_{n=0}^{N-1}x[n]W_N^{kn},\quad 0\le k\le N-1$$

以 matrix 型式表示：

$$\begin{bmatrix}X[0]\\X[1]\\X[2]\\\vdots\\X[N-1]\end{bmatrix}_{N\times 1}=\begin{bmatrix}1&1&1&\cdots&1\\1&W_N^1&W_N^2&\cdots&W_N^{(N-1)}\\1&W_N^2&W_N^4&\cdots&W_N^{2(N-1)}\\\vdots&\vdots&\vdots&\ddots&\vdots\\1&W_N^{(N-1)}&W_N^{2(N-1)}&\cdots&W_N^{(N-1)^2}\end{bmatrix}_{N\times N}\begin{bmatrix}x[0]\\x[1]\\x[2]\\\vdots\\x[N-1]\end{bmatrix}_{N\times 1}$$

對每個 k：① 需 N 次乘法

② 需 N-1 次加法

而整個過程有 N 項 $(0\le k\le N-1)$ ⇒ ① 需共 $N\times N=N^2$ 次乘法

② 需共 $N(N-1)$ 次加法

⇒ 時間複雜度：$O(N^2)$

In Ch5(3) page 1~13 中之 algorithm

將 2N-point sequence 拆成 2個 N-point real sequence

$v[n]$　　偶數項：$g[n]=v[2n]$

奇數項：$h[n]=v[2n+1]$，$0\le n<N$

就能推導出 $v[n]$ DFT 可寫為

$$V[k]=G[\langle k\rangle_N]+W_{2N}^{k}H[\langle k\rangle_N],\quad 0\le k\le 2N-1$$

為了後續推導清楚，將 v[n] 長度設為 N，則 g[n] 和 h[n] 長度為 $\frac{N}{2}$ （N為2的冪次方）

$\Rightarrow V[k] = G[\langle k \rangle_{\frac{N}{2}}] + W_N^k H[\langle k \rangle_{\frac{N}{2}}]$

G[k] 和 h[k] 為長度為 $\frac{N}{2}$ 之 DFT sequence

將長度為 N 之 DFT V[k] 簡化為兩個 $\frac{N}{2}$ 長度之 DFT 的組合

其中 $W_N^{k+\frac{N}{2}} = e^{-j\frac{2\pi}{N}(k+\frac{N}{2})} = \underbrace{e^{-j\frac{2\pi k}{N}}}_{W_N^k} \cdot \underbrace{e^{-j\pi}}_{-1} = -W_N^k$

因此 V[k] 可寫成前半及後半部：

前： $V[k] = G[\langle k \rangle_{\frac{N}{2}}] + W_N^k H[\langle k \rangle_{\frac{N}{2}}]$

後： $V[k+\frac{N}{2}] = G[\langle k+\frac{N}{2} \rangle_{\frac{N}{2}}] - W_N^k H[\langle k+\frac{N}{2} \rangle_{\frac{N}{2}}]$

$= G[\langle k \rangle_{\frac{N}{2}}] - W_N^k H[\langle k \rangle_{\frac{N}{2}}]$

, $0 \le k \le \frac{N}{2}-1$

（可用蝶型圖表示）

對於 N-point DFT 用此 algorithm：

① 乘法部分因前半及後半都用到 $W_N^k H[\langle k \rangle_{\frac{N}{2}}]$，只差在變號

⇒ 可重複利用減少運算時間

⇒ 共需 $\frac{N}{2}$ 次

② 加法部分共需 N 次

而只要 DFT 還是 2 的倍數，就能繼續分一半用相同方法做下去

以下對長度為 N 之 sequence 做此 algorithm：

DFT $_{N\text{-point}} \Longrightarrow$ 2 DFT $_{\frac{N}{2}\text{-point}}$ + $\frac{N}{2}$ 次乘法 + N 次加法

DFT $_{\frac{N}{2}\text{-point}} \Longrightarrow$ 2 DFT $_{\frac{N}{4}\text{-point}}$ + $\frac{N}{4}$ 次乘法 + $\frac{N}{2}$ 次加法

$DFT_{\frac{N}{4}\text{-point}} \Longrightarrow 2DFT_{\frac{N}{8}\text{-point}} + \frac{N}{8}$ 次乘法 $+\frac{N}{4}$ 次加法

$\vdots$

$DFT_{2\text{-point}} \Longrightarrow 2DFT_{1\text{-point}} + 1$ 次乘法 $+2$ 次加法

再代換整理：

$DFT_{N\text{-point}} \Longrightarrow 2DFT_{\frac{N}{2}\text{-point}} + \frac{N}{2}$ 次乘法 $+ N$ 次加法

$\Longrightarrow 4DFT_{\frac{N}{4}\text{-point}} + 2(\frac{N}{2})$ 次乘法 $+ 2N$ 次加法

$\Longrightarrow 8DFT_{\frac{N}{8}\text{-point}} + 3(\frac{N}{2})$ 次乘法 $+ 3N$ 次加法

$\vdots$

$\Longrightarrow N\ DFT_{1\text{-point}} + \log_2(N) \times (\frac{N}{2})$ 次乘法 $+ \log_2(N) \times N$ 次加法

$\because DFT_{1\text{-point}}$ 不需乘法及加法

$\Longrightarrow$ 共需 ① $(\frac{N}{2})\log_2 N$ 次乘法

② $N \cdot \log_2 N$ 次加法

$\Longrightarrow$ 時間複雜度：$O(N\log_2 N)$

(b) 由以上推導比較，可發現此 algorithm 可不斷將 sequence 拆成一半長度去計算 DFT，比起直接做長度為 N 之 DFT 更有效率，此方法也是 FFT 之概念

(c) 由以上比較，DFT 之時間複雜度為 $O(N^2)$
而 FFT 為 $O(N\log_2 N)$，有顯著計算差異，
FFT 有更快的運算速度