FFT與訊號處理簡介

祁忠勇演講

林聖哲記錄

(一) 何謂訊號處理:

當你要處理訊號時, 首先是必須有測量到的訊號, 而訊號必定有一個產生的過程。假設我們有一輸入訊號$u(n)$, 進入某一個未知的系統後, 而得到一個輸出訊號$x(n)$, 然後我們想要設計一個系統來處理訊號$x(n)$, 以抽出它所攜帶的情報, 這便是我們所謂的「訊號處理」。而如何設計一好的系統, 方能從$x(n)$中正確而完全的獲得我們想要知道的情報, 便是訊號處理者日日在研究的課題。我們現在舉些例子來讓大家更能感覺到什麼是訊號處理。

首先是通訊, 在通訊裡面我們常常傳送0和1兩個數字的訊號, 當然也可能不祇兩個數字, 而所有的通信頻道都有一定的頻寬, 假如你傳送的太快, 那麼前一個訊號和後一個訊號會相互干擾, 從所收到的$x(n)$就再也看不出它原來傳送的信號爲0或1了。因此, 必須設計一個訊號處理系統 (Signal Processing System) 來處理所收到訊號, 以還原所傳送的信號, 這就是一個典型的訊號處理的問題。另一個例子便是語音信號處理, 人的聲音是由聲門產生一振幅不均勻之脈衝串來激發聲道而產生聲音, 且經由聲道形狀的改變產生不同的聲音。我們也可經由電腦來合成這些聲音, 但因爲是合成的, 所以會有些不自然。當然我們也可以設計一個訊號處理系統來抽出一些想知道的訊息, 如所發的是什麼音、頻率多少、以及是由何人所發的, 這便可應用在語音鎖上。另外還有很多應用, 例如: 石油探戡、聲納、影像、雷達· · ·等等。

(二) 何謂訊號和系統:

訊號是一個系統所輸入的函數, 不論是連續變數的函數, 或是離散變數的函數, 而系統的輸出也叫做訊號, 還有系統內部所有可以代表的量, 我們也稱作訊號。那麼什麼又是所謂的連續時間訊號 (Continuous-time Signal), 您可以將它想像成一個以時間t爲變數之函數$x(t)$, 但時間 t 是連續的, 例如: $x(t) = A \cdot \sin(w_0 t)$, 它是一個三角函數, 當然$t$的變化是連續的。而所謂的離散時間訊號 (Discrete-time Signal) 又是什麼呢? 簡單的說, 就是整數的函數$x(n)$。例如: $x(n) = A\sin(w_0 n)$就是一個離散時間訊號了。

接著我們來說明系統是什麼? 簡單的說:「有輸入, 有輸出」的便是系統了。如果輸入與輸出均是離散時間訊號, 此系統就稱作離散時間系統 (Discrete-time System); 同理, 如果輸入與輸出均是連續時間訊號, 此系統就稱爲連續時間系統 (Continuous-time System)。再讓我們介紹一種訊號處理系統, 如此我們才能往下推進。這系統稱爲 Discrete-time Linear Time-invariant System, 而什麼又叫做 Linear System 呢? 就是此系統在輸入與輸出間形成一種線性組合的關係。也就是說, 輸出是輸入的訊號經一種線性的關係組合而成的。那麼什麼又叫做 Time-invariant 呢? 假設某一個輸入訊號進入一系統, 結果產生一個輸出訊號, 但是如果慢一個小時輸入的話, 還是會得到相同的輸出訊號, 並不會因輸入時間不同而得到不同的結果, 這就稱作 Time-invariant。而對於 Discrete-time Linear Time-invariant system, 它的輸出和輸入有一定的關係: 假設輸入訊號是$x(n)$而輸出訊號是$y(n)$, 那麼$y(n) = h(n) * x(n)$, 其中$*$是一種$h(n)$和$x(n)$的運算, 而這運算稱爲 Convolution, 也就是說$y(n) = h(n) * x(n) = \sum_{k=-\infty}^{\infty} h(k)x(n-k) = \sum_{k=-\infty}^{\infty} x(k)h(n-k)$, 而其中$h(n)$稱爲脈衝響應 (Impulse Response), 而它又是什麼呢? 就是此系統, 當輸入訊號是 $\delta(n)$ ($\delta(0) = 1$, $\delta(n \neq 0) = 0$)時, 所得到的輸出訊號。 所以當$h(n)$已知, 這個 Discrete-time Linear Time-invariant System 的輸入輸出訊號間的關係完全確定。

(三)Fast Fourier Transform (FFT):

首先讓我介紹 Discrete-time Fourier Transform。假設我們有一個訊號$x(n)$, 將它乘上 e^{-jwn}, 然後將它從$n = -\infty$至∞連加起來, 我們用$X(e^{jw}) = \sum_{n=-\infty}^{\infty} x(n)e^{-jwn}$表示所得到的結果, 它就形成了一個連續變數$w$的函數。再則, 可以從理論上證明$x(n) = \frac{1}{2\pi}\int_{-\pi}^{\pi} X(e^{jw})\ e^{jwn}dw$, 而 $X(e^{jw})$ 就稱爲 $x(n)$ 的 Fourier Transform, $x(n)$ 就是 $X(e^{jw})$ 的 Inverse Fourier Transform。那麼在物理上它有什麼意思呢? 我們稱$X(e^{jw})$爲分析, 也就是我想知道$x(n)$這個訊號包含的成分是什麼, 也就是某種頻率(w)的成分有多大。例如: 人所發出的聲音之頻率一般最高到4K Hz, 所以我們可以用這種方法分析人的聲音大部分在那一個音頻區。而反過來, 以各種不同頻率的成分組成$x(n)$, 所以它就像一種合成。而 Inverse Fourier Transform 和 Fourier Transform 在工程裡面經常重複地使用, 所以必須發展快速的計算方法來解決這個問題, 也因爲電腦的出現與進步, 也就使得這門學問日漸壯大。

接著讓我們談談什麼是 Fast Fourier Transform(以下簡稱 FFT), 假設我們有一個 Linear Time-invariant System(以下簡稱 LIT 系統), 其脈衝響應爲$h(n)$, 我們將輸出和輸入的訊號做 Fourier Transform 的話, 我們會發現輸出訊號的 Fourier Transform恰巧會等於輸入訊號的 Fourier

Transform 乘上$h(n)$的 Fourier Transform, 也就是說$Y(e^{jw}) = X(e^{jw}) \cdot H(e^{jw})$, 其中$|H(e^{jw})|$ 就稱爲系統之振幅響應 (magnitude response), 而$H(e^{jw})$ 的角度就稱爲系統之相位響應 (phase response), 所以除了用$y(n) = x(n) * h(n)$ 來計算輸出信號以外, 這也提供另一種方法去計算 LTI 系統的輸出$y(n)$。

上述結果和 FFT 又有什麼關係呢? 假設現在有一個接收到的信號, 而其中在某個頻率以 下才是我們想要的信號, 其餘皆是雜訊, 那麼你如何設計一個濾波器 (filter), 它本身即是一個 LTI 系統, 且對某一頻率以下之輸入信號完全接受, 在此一頻率以上之輸入信號完全清除? 在設計這個濾波器的過程, 必須不斷重複計算$H(e^{jw})$, 也就是說, 重複計算 Fourier Transform 許許多多次, 所以必須發展 FFT 以滿足我們的需求。

再接著我們介紹 Discrete Fourier Transform (DFT)。在數位信號處理 (Digital Signal Processing) 的領域裡所使用的 Fourier Transform, 都是指 Discrete-time Signal 的 Fourier Transform。而 DFT 便是指在 Fourier Transform 中的w從0到 2π 的範圍 (一週期) 分成N等分點, 即每兩點的間隔爲$2\pi/N$, 再假設信號$x(n)$的範圍是從0到$N-1$, 因此 DFT 是指$X(e^{j2\pi k/N}) = \sum_{n=0}^{N-1} x(n)e^{-j2\pi kn/N}$, 而且$x(n)$可以證明等於 $\frac{1}{N}\sum_{k=0}^{N-1} X(e^{j2\pi k/N}) \cdot e^{j2\pi kn/N}$。如此一來它便較符合一般實際的應用, 畢竟我們不可能累加一個從 $-\infty$到∞的數列。

計算一個N點的 DFT 運算究竟要花多少的運算量呢? 如果直接來計算, 它需要N^2個複數運算 (指加減乘除), 接下來讓我們告訴您其實不需要那麼多。

假設有一個訊號$x(n)$是一個有限的複數數列, 長度正好是2的指數倍, 即$N = 2^v$, 令$W_N \equiv e^{-j2\pi/N}$, 則我們可以很容易地看出兩個結果: $W_N^{k(N-n)} = W_N^{-kn} = (W_N^{kn})^*$, $W_N^{kn} = W_N^{k(n+N)} = W_N^{(k+N)n}$, 而FFT 便是充分利用這兩個結果來計算 Fourier Transform $X(e^{j2\pi k/N})$, 以減少所需的運算量。下面讓我介紹一下 Decimation-in-time FFT 演算法則。

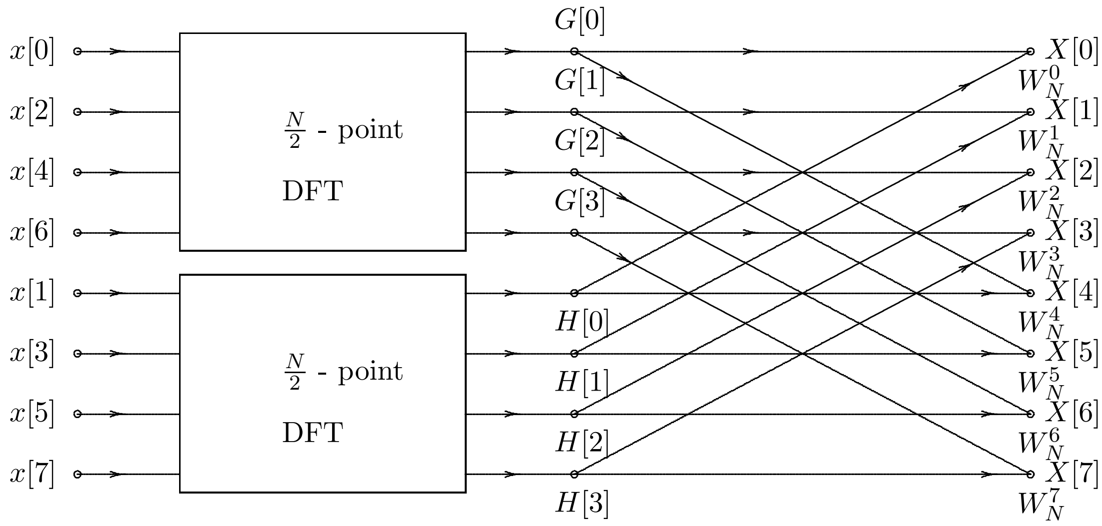
將 DFT 的定義$X(k) \equiv X(e^{j2\pi k/N}) = \sum_{n=0}^{N-1} x(n)W_N^{kn}$, $k = 0, 1, \cdots, N-1$, 表示成下列奇數項的和加上偶數項的和:

$$\begin{aligned} X(k) &= \sum_{r=0}^{N/2-1} x(2r)(W_N^2)^{rk} \\ &\quad + \sum_{r=0}^{N/2-1} x(2r+1)(W_N^2)^{rk} \cdot {W_N}^k \\ &= G(k) + W_N^k \cdot H(k) \end{aligned}$$

其中$W_N^2 = e^{-j2\pi \cdot 2/N} = e^{-j2\pi/(\frac{N}{2})} = W_{N/2}$, $G(k)$和$H(k)$分別是下列兩個$N/2$點的 DFT:

$$\begin{aligned} G(k) &= \sum_{n=0}^{N/2-1} x(2n)W_{N/2}^{kn} \\ H(k) &= \sum_{n=0}^{N/2-1} x(2n+1)W_{N/2}^{kn} \end{aligned}$$

如此一來我們算一個N點的 DFT, 變成算兩個$N/2$點的 DFT, 這樣做究竟是否有減少運算呢? 現在我們用一個流程圖來表示上述的計算過程, (假設$N = 8$):



† 上圖取自 A.V. Oppenheim and R.W. Schafer, Discrete-time Signal Processing, Prentice-Hall, 1989之圖9.3。

因為直接算$G(k)$祇需$(\frac{N}{2})^2$個複數運算, 同理算 $H(k)$ 也需 $(\frac{N}{2})^2$ 個複數運算, 再加上最後 W_N^k 的 N個乘法, 所以我們共需要$2\cdot(\frac{N}{2})^2+N$個複數運算, 與直接算N點的 DFT 所需要的運算量N^2相比, 似乎祇減少了一倍的運算, 沒什麼了不起嘛!但是如果再將上面的$G(k)$和$H(k)$拆成奇數項和偶數項二部份, 我們便又可減少運算量了, 如此一層層的拆下去, 我們便可發現最後事實上我們祇需要$N\cdot\log_2 N$個複數運算, 這將遠小於N^2個複數運算。

有一個值得一提的注意事項是在計算DFT 之前,$x(n)$必須重新排列, 而排列的法則便是將它的二進位表示法, 前後次序顛倒, 例如: $3\longrightarrow 011\longrightarrow 110\longrightarrow 6$。另外, 在圖中的運算形式, 均為蝶形運算, 且 $X(k)=G(k)+W_N^kH(k)$, $X(k+4)=G(k+4)+W_N^{k+4}H(k+4)$, $k=0,1,2,3$。所以, 將圖中的蝶形運算之公因式提出($W_N^{r+N/2}=-W_N^r$), 則$X(k+4)=G(k+4)-W_N^kH(k+4)$, 如此做還可以再減少一倍的運算, 使得我們祇須要$(N/2)\cdot\log_2 N$的運算, 例如: $N=1024=2^{10}$, FFT 的運算量約是直接計算 DFT 的二百分之一。

(四) 信號處理的未來:

信號處理研究可分成二類, 一是須要統計量的信號處理, 另一類是不須統計量的信號處理。但是無論是那一類的信號處理, 快速信號處理演算法則是永遠需要的, 而 FFT 祇是一個典型的例子。各種不同目的的信號處理演算法則不斷地被開發出來, 然而能夠將已開發出來的信號處理演算法則變得更快速也才更能夠符合實際的應用。

—本文作者任教於清華大學電機系—