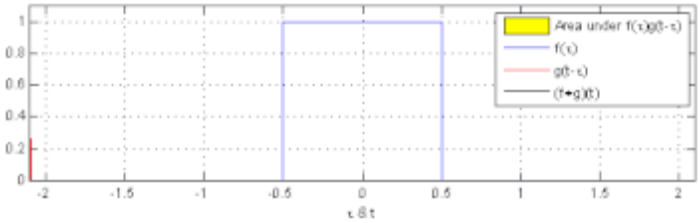
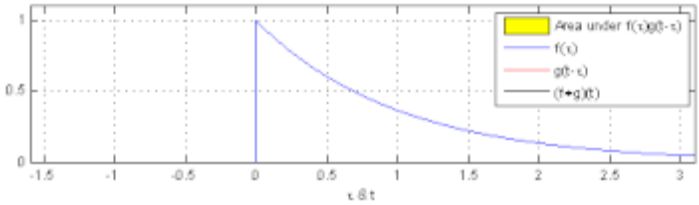
卷积

维基百科，自由的百科全书

在泛函分析中，**捲積**、**疊積**（convolution）、**褶積**或**旋積**，是通过两个函数*f*和*g*生成第三个函数的一种数学算子，表徵函数*f*与经过翻转和平移的*g*的乘積函數所圍成的曲邊梯形的面積。如果将参加卷积的一个函数看作区间的指示函数，卷积还可以被看作是“滑動平均”的推广。



图示两个方形脈衝波的捲積。其中函数"g"首先对$\tau = 0$反射，接著平移"t"，成為$g(t - \tau)$。那么重叠部份的面积就相当于"t"处的卷积，其中横坐標代表待变量τ以及新函數$f * g$的自變量"t"。



圖示方形脈衝波和指數衰退的脈衝波的捲積（後者可能出現於RC電路中），同樣地重疊部份面積就相當於"t"處的捲積。注意到因為"g"是對稱的，所以在這兩張圖中，反射並不會改變它的形狀。

目录

简单介绍

卷积是数学分析中一种重要的运算。设：$f(x)$、$g(x)$是$\mathbb{R}$上的两个可积函数，作积分：

$$\int_{-\infty}^{\infty} f(\tau)g(x-\tau)\,\mathrm{d}\tau$$

可以证明，关于几乎所有的$x \in (-\infty, \infty)$，上述积分是存在的。这样，随着x的不同取值，这个积分就定义了一个新函数$h(x)$，称为函数f与g的卷积，记为$h(x) = (f * g)(x)$。我們可以輕易验证：$(f * g)(x) = (g * f)(x)$，并且$(f * g)(x)$仍为可积函数。这就是说，把卷积代替乘法，$L^1(R^1)$空间是一个代数，甚至是巴拿赫代数。雖然這裡為了方便我們假設

$f, g \in L^1(\mathbb{R})$，不過捲積只是運算符號，理論上並不需要對函數 f, g 有特別的限制，雖然常常要求 f, g 至少是可測函數（measurable function）（如果不是可測函數的話，積分可能根本沒有意義），至於生成的卷積函數性質會在運算之後討論。

卷积与傅里叶变换有着密切的关系。例如两函数的傅里叶变换的乘积等于它们卷积后的傅里叶变换，利用此一性質，能簡化傅里叶分析中的许多问题。

由卷积得到的函数$f * g$一般要比f和g都光滑。特别当g为具有紧支集的光滑函数，f为局部可积时，它们的卷积$f * g$也是光滑函数。利用这一性质，对于任意的可积函数f，都可以简单地构造出一列逼近于f的光滑函数列f_s，这种方法称为函数的光滑化或正则化。

卷积的概念还可以推广到数列、测度以及广义函数上去。

定义

函数 f, g 是定義在 $\mathbb{R}^n$ 上的可測函數（measurable function），f与g的卷积记作$f * g$，它是其中一个函数翻转并平移后与另一个函数的乘积的积分，是一个对平移量的函数，也就是：

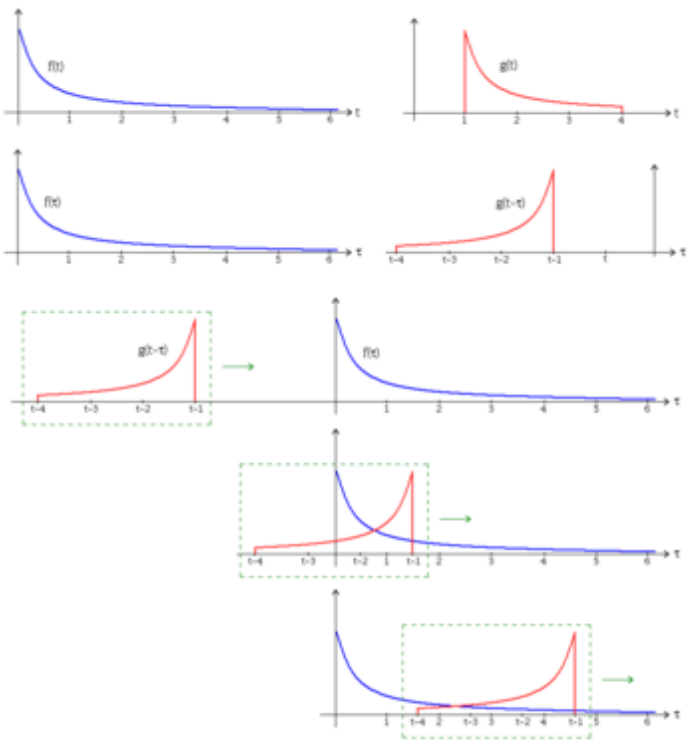
$$(f * g)(t) \ \stackrel{\text{def}}{=} \int_{\mathbb{R}^n} f(\tau) g(t - \tau)\, d\tau。$$

如果函數不是定義在 $\mathbb{R}^n$ 上，可以把函數定義域以外的值都規定成零，這樣就變成一個定義在 $\mathbb{R}^n$ 上的函數。

圖解摺積

1. 已知两函数f(t)和g(t)。右图第一行两图分别为f(t)和g(t)。
2. 首先將兩個函數都用τ來表示，从而得到f(τ)和g(τ)。将函数g(τ)向右移动t个单位，得到函数g(τ-t)的图像。将g(τ-t)翻转至纵轴另一侧，得到g(-(τ-t))即g(t-τ)的图像。右图第二行两图分别为f(τ)和g(t-τ)。
3. 由于t非常数（实际上是时间变量），当时间变量（以下简称“时移”）取不同值时，$g(t - \tau)$能沿着τ轴“滑动”。右图第三四五行可理解为“滑动”。
4. 讓τ從-∞滑動到+∞。兩函數交會時，計算交會範圍中兩函數乘積的積分值。換句話說，我們是在計算一個滑動的的加權總和(weighted-sum)。也就是使用$g(t - \tau)$當做加權函數，來對$f(\tau)$取加權值。

   最後得到的波形（未包含在此圖中）就是f和g的摺積。如果f（t）是一個單位脈衝，我們得到的乘積就是g（t）本身，稱為衝激響應。



离散卷积

对于定义在整数 $\mathbb{Z}$ 上的函数 f, g，卷积定义为

$$(f*g)[n] \overset{\text{def}}{=} \sum_{m=-\infty}^{\infty} f[m]g[n-m]$$

$$= \sum_{m=-\infty}^{\infty} f[n-m]\, g[m].$$

這裡一樣把函數定義域以外的值當成零，所以可以擴展函數到所有整數上（如果本來不是的話）。

當$g[n]$ 的支撐集（support）為有限長度M，上式會變成有限和：

$$(f*g)[n] = \sum_{m=-M}^{M} f[n-m]g[m].$$

計算离散卷積的方法

計算卷積$f[n]*g[n]$有三種主要的方法，分別為

1. 直接計算（Direct Method）
2. 快速傅立葉轉換（FFT）
3. 分段卷積（sectioned convolution）

方法1是直接利用定義來計算卷積，而方法2和3都是用到了FFT來快速計算卷積。也有不需要用到FFT的作法，如使用數論轉換。

方法1：直接計算

- 作法：利用卷積的定義

$$y[n] = f[n]*g[n] = \sum_{m=0}^{M-1} f[n-m]g[m]$$

- 若$f[n]$和$g[n]$皆為實數信號，則需要MN個乘法。
- 若$f[n]$和$g[n]$皆為更一般性的複數信號，不使用複數乘法的快速演算法，會需要$4MN$個乘法;但若使用複數乘法的快速演算法，則可簡化至$3MN$個乘法。

  因此，使用定義直接計算卷積的複雜度為$O(MN)$。

方法2：快速傅立葉轉換（FFT）

- 概念：由於兩個離散信號在時域（time domain）做卷積相當於這兩個信號的離散傅立葉轉換在頻域（frequency domain）做相乘：

  $$y[n] = f[n]*g[n] \leftrightarrow Y[f] = F[f]G[f]$$

  ，可以看出在頻域的計算較簡單。

- 作法：因此這個方法即是先將信號從時域轉成頻域：

  $$F[f] = DFT_P(f[n]), G[f] = DFT_P(g[n])$$

  ，於是

$$Y[f] = DFT_P(f[n])DFT_P(g[n])$$

，最後再將頻域信號轉回時域，就完成了卷積的計算：

$$y[n] = IDFT_P DFT_P(f[n])DFT_P(g[n])$$

總共做了2次DFT和1次IDFT。

- 特別注意DFT和IDFT的點數P要滿足$P \geq M + N - 1$。
- 由於DFT有快速演算法FFT，所以運算量為$O(P \log_2 P)$
- 假設P點DFT的乘法量為a，$f[n]$和$g[n]$為一般性的複數信號，並使用複數乘法的快速演算法，則共需要$3a + 3P$個乘法。

方法3：分段卷積（sectioned convolution）

- 概念：將$f[n]$切成好幾段，每一段分別和$g[n]$做卷積後，再將結果相加。
- 作法：先將$f[n]$切成每段長度為L的區段（$L > M$），假設共切成S段：

$$f[n](n = 0, 1, \ldots, N-1) \rightarrow f_1[n], f_2[n], f_3[n], \ldots, f_S[n](S = \left\lceil \frac{N}{L} \right\rceil)$$

Section 1: $f_1[n] = f[n], n = 0, 1, \ldots, L-1$

Section 2: $f_2[n] = f[n + L], n = 0, 1, \ldots, L-1$

$$\vdots$$

Section r: $f_r[n] = f[n + (r-1)L], n = 0, 1, \ldots, L-1$

$$\vdots$$

Section S: $f_S[n] = f[n + (S-1)L], n = 0, 1, \ldots, L-1$

，$f[n]$為各個section的和

$$f[n] = \sum_{r=1}^{S} f_r[n + (r-1)L]。$$

因此，

$$y[n] = f[n] * g[n] = \sum_{r=1}^{S} \sum_{m=0}^{M-1} f_r[n + (r-1)L - m]g[m] \text{ ，}$$

每一小段作卷積則是採用方法2，先將時域信號轉到頻域相乘，再轉回時域：

$$y[n] = IDFT(\sum_{r=1}^{S} \sum_{m=0}^{M-1} DFT_P(f_r[n + (r-1)L - m])DFT_P(g[m])), P \geq M + L - 1。$$

- 總共只需要做P點FFT $2S + 1$次，因為$g[n]$只需要做一次FFT。

- 假設P點DFT的乘法量為a，$f[n]$和$g[n]$為一般性的複數信號，並使用複數乘法的快速演算法，則共需要$(2S+1)a+3SP$個乘法。
- 運算量：$\frac{N}{L}3(L+M-1)[\log_2(L+M-1)+1]$
- 運算複雜度：$O(N)$，和N呈線性，較方法2小。
- 分為 Overlap-Add 和 Overlap-Save 兩種方法。

分段卷積: Overlap-Add

欲做$x[n]*h[n]$的分段卷積分，$x[n]$ 長度為 N，$h[n]$ 長度為 M,

Step 1: 將$x[n]$每 L 分成一段

Step 2: 再每段 L 點後面添加 $M-1$ 個零，變成長度 $L+M-1$

Step 3: 把 $h[n]$ 添加 $L-1$個零，變成長度 $L+M-1$的 $h'[n]$

Step 4: 把每個 $x[n]$ 的小段和 $h'[n]$ 做快速卷積，也就是$IDFT_{L+M-1}\{DFT_{L+M-1}(x[n])DFT_{L+M-1}(h'[n])\}$，每小段會得到長度 $L+M-1$ 的時域訊號

Step 5: 放置第 i 個小段的起點在位置 $L\times i$ 上, $i=0,1,\dots,\lceil\frac{N}{L}\rceil-1$

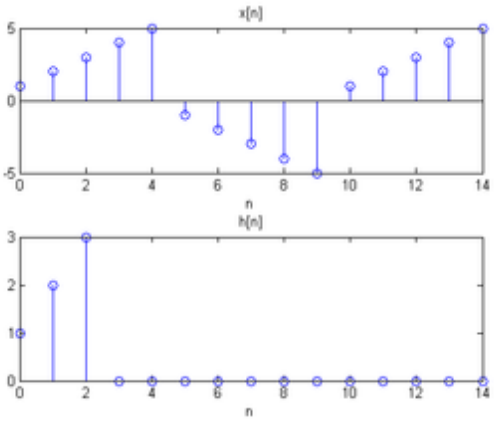
Step 6: 會發現在每一段的後面 $M-1$ 點有重疊，將所有點都相加起來，顧名思義 Overlap-Add，最後得到結果

舉例來說:

$x[n]=[1,2,3,4,5,-1,-2,-3,-4,-5,1,2,3,4,5]$, 長度 $N=15$

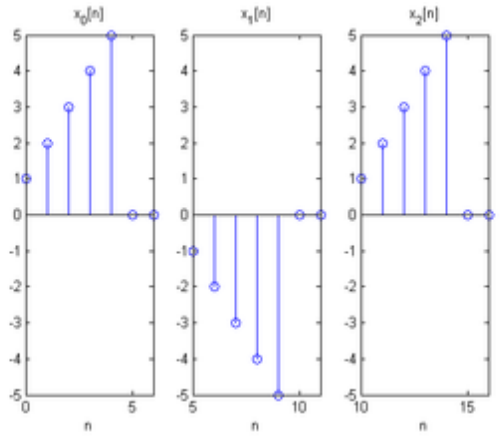
$h[n]=[1,2,3]$, 長度 $M=3$

令 $L=5$



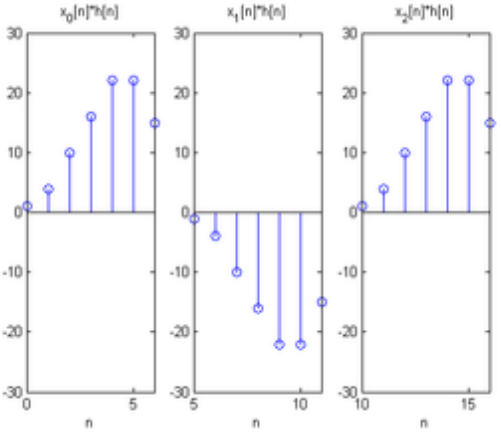
x[n]和h[n]

令 $L=5$ 切成三段，分別為 $x_0[n],x_1[n],x_2[n]$, 每段填 $M-1$ 個零，並將 $h[n]$ 填零至長度 $L+M-1$



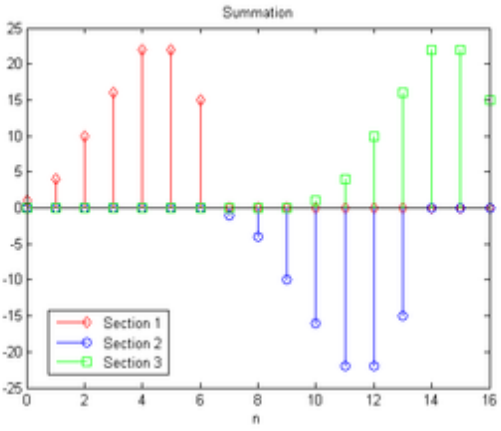
分段x[n]

將每一段做 $IDFT_{L+M-1}\{DFT_{L+M-1}(x[n])DFT_{L+M-1}(h'[n])\}$



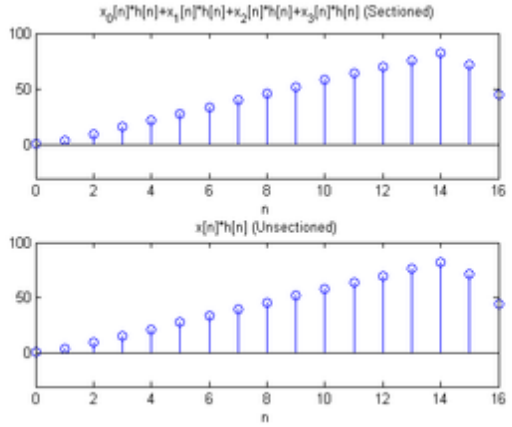
分段運算結果

若將每小段擺在一起，可以注意到第一段的範圍是 $0 \sim 6$ ，第二段的範圍是 $5 \sim 11$，第三段的範圍是 $10 \sim 16$，三段的範圍是有重疊的



合併分段運算結果

最後將三小段加在一起，並將結果和未分段的卷積做比較，上圖是分段的結果，下圖是沒有分段並利用快速卷積所算出的結果，驗證兩者運算結果相同。



結果比較圖

分段卷積: Overlap-Save

欲做$x[n] * h[n]$的分段卷積分，$x[n]$ 長度為 N，$h[n]$ 長度為 M,

Step 1: 將 $x[n]$ 前面填 $M-1$ 個零

Step 2: 第一段 $i=0$, 從新的 $x[n]$ 中 $L \times i-(M-1) \times i$ 取到 $L \times (i+1)-(M-1) \times i-1$ 總共 L 點當做一段，因此每小段會重複取到前一小段的 $M-1$ 點，取到新的一段全為零為止

Step 3: 把 $h[n]$ 添加 $L-M$ 個零，變成長度 L 的 $h'[n]$

Step 4: 把每個 $x[n]$ 的小段和 $h'[n]$ 做快速卷積，也就是$IDFT_L\{DFT_L(x[n])DFT_L(h'[n])\}$，每小段會得到長度 L 的時域訊號

Step 5: 對於每個 i 小段，只會保留末端的 $L-(M-1)$ 點，因此得名 Overlap-Save

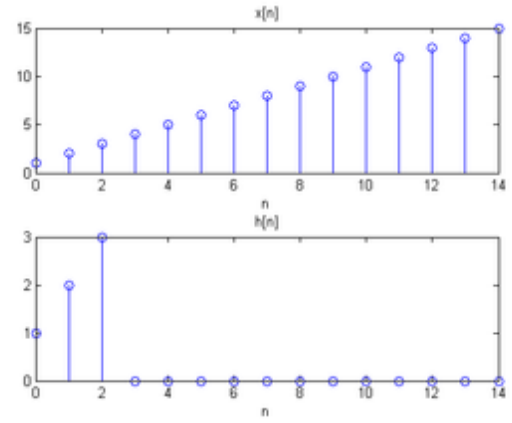
Step 6: 將所有保留的點合再一起，得到最後結果

舉例來說:

$x[n]=[1,2,3,4,5,6,7,8,9,10,11,12,13,14,15]$, 長度 $N=15$

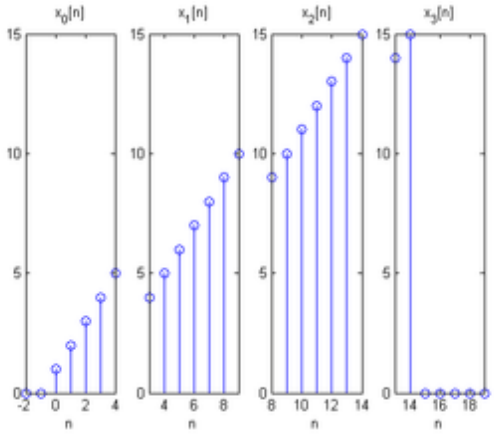
$h[n]=[1,2,3]$, 長度 $M=3$

令 $L=7$



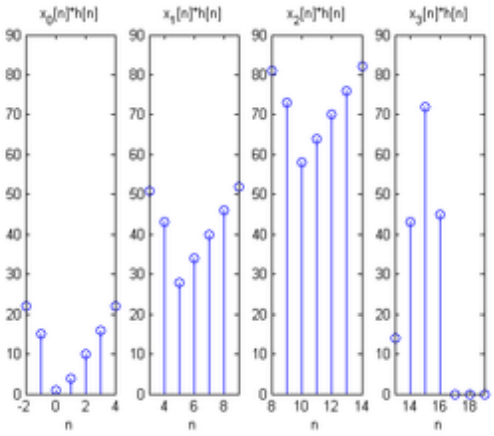
x[n]和h[n]

將 $x[n]$ 前面填 $M-1$ 個零以後，按照 Step 2 的方式分段，可以看到每一段都重複上一段的 $M-1$ 點



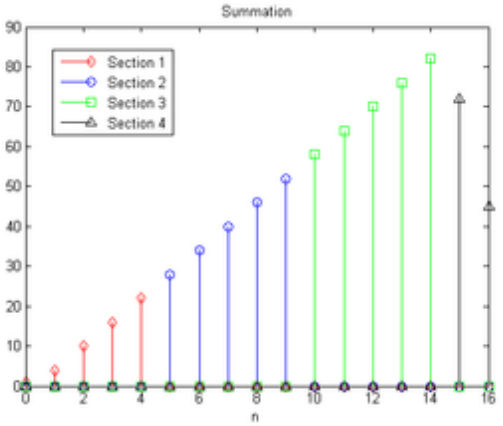
分段x[n]

再將每一段做 $IDFT_L\{DFT_L(x[n])DFT_L(h'[n])\}$ 以後可以得到



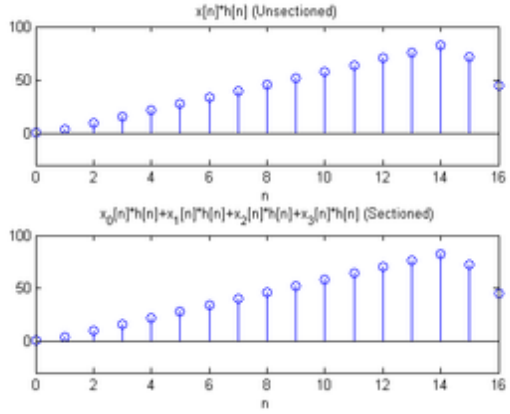
分段運算結果

保留每一段末端的 $L-(M-1)$ 點，擺在一起以後，可以注意到第一段的範圍是 $0\sim4$ ，第二段的範圍是 $5\sim9$，第三段的範圍是 $10\sim14$，第四段的範圍是 $15\sim16$，四段的範圍是沒有重疊的



合併分段運算結果

將結果和未分段的卷積做比較，下圖是分段的結果，上圖是沒有分段並利用快速卷積所算出的結果，驗證兩者運算結果相同。



結果比較圖

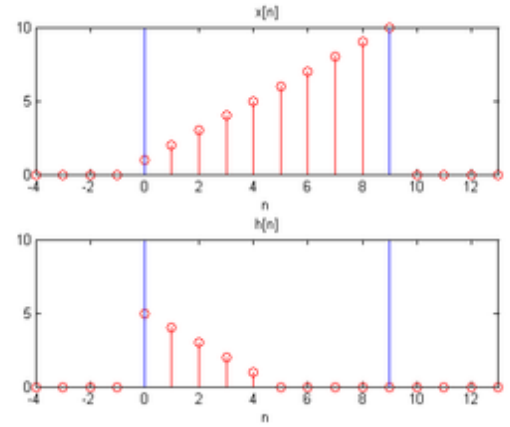
至於為什麼要把前面 $M-1$ 丟掉?

以下以一例子來闡述:

$x[n]=[1,2,3,4,5,6,7,8,9,10]$, 長度 $L=10$,

$h[n]=[1,2,3,4,5]$, 長度 $M=5$,

第一條藍線代表 y 軸，而兩條藍線之間代表長度 L，是在做快速摺積時的週期



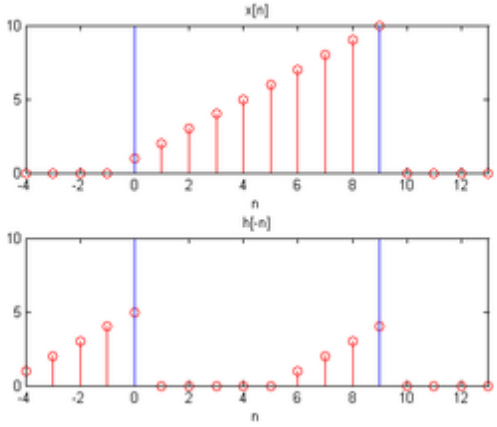
x[n]和h[n]

當在做快速摺積時 $IDFT_L\{DFT_L(x[n])DFT_L(h'[n])\}$ ，是把訊號視為週期 L，在時域上為循環摺積分,

而在一開始前 $M-1$ 點所得到的值，是 $h[0],h[6],h[7],h[8],h[9]$ 和 $x[0],x[6],x[7],x[8],x[9]$ 內積的值，

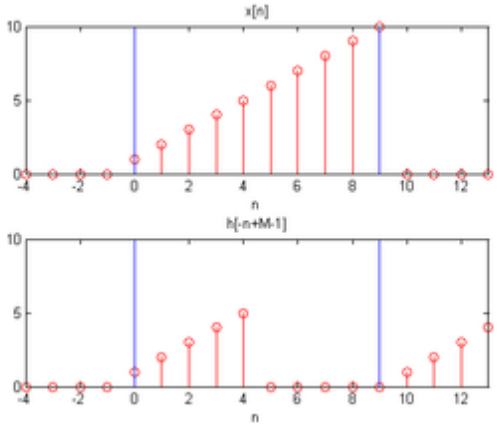
然而 $h[6],h[7],h[8],h[9]$ 這 $M-1$ 個值應該要為零，以往在做快速摺積時長度為 $L+M-1$ 時不會遇到這些問題，

而今天因為在做快速摺積時長度為 L 才會把這 $M-1$ 點算進來，因此我們要丟棄這 $M-1$ 點內積的結果



循環摺積

為了要丟棄這 $M-1$ 點內積的結果，位移 $h[-n]$ $M-1$ 點，並把位移以後內積合的值才算有效。



位移以後內積

應用時機

以上三種方法皆可用來計算卷積，其差別在於所需總體乘法量不同。基於運算量以及效率的考量，在計算卷積時，通常會選擇所需總體乘法量較少的方法。

以下根據$f[n]$和$g[n]$的長度（N, M）分成5類，並列出適合使用的方法：

1. M為一非常小的整數 - 直接計算
2. $M \ll N$ - 分段卷积
3. $M \approx N$ - 快速傅里叶变换
4. $M \gg N$ - 分段卷积
5. N為一非常小的整數 - 直接計算

基本上，以上只是粗略的分類。在實際應用時，最好還是算出三種方法所需的總乘法量，再選擇其中最有效率的方法來計算卷積。

例子

Q1：當$N = 2000, M = 17$，適合用哪種方法計算卷積?

Ans：

方法1：所需乘法量為$3MN = 102000$

方法2：$P \geq M + N - 1 = 2016$，而2016點的DFT最少乘法數$a = 12728$，所以總乘法量為$3(a + P) = 44232$

方法3：

若切成8塊（$S = 8$），則$L = 250, P \geq M + L - 1 = 266$。選$P = 288$，則總乘法量為$(2S + 1)a + 3SP = 26632$，比方法1和2少了很多。
但是若要找到最少的乘法量，必須依照以下步驟

(1)先找出L:解L：$\frac{\partial \frac{N}{L} 3(L + M - 1)[\log_2(L + M - 1) + 1]}{\partial L} = 0$
(2)由$P \geq L + M - 1$算出點數在P附近的DFT所需最少的乘法量，選擇DFT的點數
(3)最後由$L = P + 1 - M$算出L_{opt}

因此，

(1)由運算量對L的偏微分為0而求出$L = 85$
(2)$P \geq L + M - 1 = 101$，所以選擇101點DFT附近點數乘法量最少的點數$P = 96$或$P = 120$。
(3-1)當$P = 96 \rightarrow a = 280, L = P + 1 - M = 80 \rightarrow S = 25$，總乘法量為$(2S + 1)a + 3SP = 21480$。
(3-2)當$P = 120 \rightarrow a = 380, L = P + 1 - M = 104 \rightarrow S = 20$，總乘法量為$(2S + 1)a + 3SP = 22780$。

由此可知，切成20塊會有較好的效率，而所需總乘法量為21480。

- 因此，當$N = 2000, M = 17$，所需總乘法量：分段卷積<快速傅立葉轉換<直接計算。故，此時選擇使用分段卷積來計算卷積最適合。

Q2：當$N = 1024, M = 3$，適合用哪種方法計算卷積?

Ans：

方法1：所需乘法量為$3MN = 9216$

方法2：$P \geq M + N - 1 = 1026$，選擇1026點DFT附近點數乘法量最少的點數，
$\rightarrow P = 1152, a = 7088$。

因此，所需乘法量為$3(a + P) = 24342$

方法3：

(1)由運算量對L的偏微分為0而求出$L = 5$
(2)$P \geq L + M - 1 = 7$，所以選擇7點DFT附近點數乘法量最少的點數$P = 8$或$P = 6$或$P = 4$。
(3-1)當$P = 8 \rightarrow a = 4, L = P + 1 - M = 6 \rightarrow S = 171$，總乘法量為$(2S + 1)a + 3SP = 5476$。
(3-2)當$P = 6 \rightarrow a = 4, L = P + 1 - M = 4 \rightarrow S = 256$，總乘法量為$(2S + 1)a + 3SP = 6660$。
(3-3)當$P = 4 \rightarrow a = 0, L = P + 1 - M = 2 \rightarrow S = 512$，總乘法量為$(2S + 1)a + 3SP = 6144$。

由此可知，切成171塊會有較好的效率，而所需總乘法量為5476。

- 因此，當$N=1024, M=3$，所需總乘法量：分段卷積<直接計算<快速傅立葉轉換。故，此時選擇使用分段卷積來計算卷積最適合。
- 雖然當M是個很小的正整數時，大致上適合使用直接計算。但實際上還是將3個方法所需的乘法量都算出來，才能知道用哪種方法可以達到最高的效率。

Q3：當$N=1024, M=600$，適合用哪種方法計算卷積?

Ans：

方法1：所需乘法量為$3MN=1843200$

方法2：$P \geq M+N-1=1623$，選擇1026點DFT附近點數乘法量最少的點數，
$\rightarrow P=2016, a=12728$。

因此，所需乘法量為$3(a+P)=44232$

方法3：

(1)由運算量對L的偏微分為0而求出$L=1024$
(2)$P \geq L+M-1=1623$，所以選擇1623點DFT附近點數乘法量最少的點數$P=2016$。
(3)當$P=2016 \rightarrow a=12728, L=P+1-M=1417 \rightarrow S=1$，總乘法量為$(2S+1)a+3SP=44232$。

由此可知，此時切成一段，就跟方法2一樣，所需總乘法量為44232。

- 因此，當$N=1024, M=600$，所需總乘法量：快速傅立葉轉換 = 分段卷積<直接計算。故，此時選擇使用分段卷積來計算卷積最適合。

多元函数卷积

按照翻转、平移、积分的定义，还可以类似的定义多元函数上的积分：

$$(f*g)(t_1,t_2,\cdots,t_n)=\int\int\cdots\int f(\tau_1,\tau_2,\cdots,\tau_n)g(t_1-\tau_1,t_2-\tau_2,\cdots,t_n-\tau_n,)\,d\tau_1 d\tau_2\cdots d\tau_n$$

性质

各种卷积算子都满足下列性质：

交换律

$$f*g=g*f$$

结合律

$$f*(g*h)=(f*g)*h$$

分配律

$$f*(g+h)=(f*g)+(f*h)$$

数乘结合律

$$a(f*g)=(af)*g=f*(ag)$$

其中a为任意实数（或复数）。

微分定理

$$\mathcal{D}(f*g) = \mathcal{D}f*g = f*\mathcal{D}g$$

其中D*f*表示*f*的微分，如果在离散域中则是指差分算子，包括前向差分与后向差分两种：

- 前向差分：$\mathcal{D}^{+}f(n) = f(n+1) - f(n)$
- 后向差分：$\mathcal{D}^{-}f(n) = f(n) - f(n-1)$

卷积定理

卷积定理指出，函数卷积的傅里叶变换是函数傅里叶变换的乘积。即，一个域中的卷积相当于另一个域中的乘积，例如时域中的卷积就对应于频域中的乘积。

$$\mathcal{F}(f*g) = \mathcal{F}(f)\cdot\mathcal{F}(g)$$

其中$\mathcal{F}(f)$表示*f*的傅里叶变换。

这一定理对拉普拉斯变换、双边拉普拉斯变换、Z变换、Mellin变换和Hartley变换（参见Mellin inversion theorem）等各种傅里叶变换的变体同样成立。在调和分析中还可以推广到在局部紧致的阿贝尔群上定义的傅里叶变换。

利用卷积定理可以简化卷积的运算量。对于长度为n的序列，按照卷积的定义进行计算，需要做$2n-1$组对位乘法，其计算复杂度为$\mathcal{O}(n^2)$；而利用傅里叶变换将序列变换到频域上后，只需要一组对位乘法，利用傅里叶变换的快速算法之后，总的计算复杂度为$\mathcal{O}(n\log n)$。这一结果可以在快速乘法计算中得到应用。

在群上的卷积

若*G*是有某*m* 测度的群（例如豪斯多夫空间上哈尔测度下局部紧致的拓扑群），对于*G*上*m*-勒贝格可积的实数或复数函数*f*和*g*，可定义它们的卷积：

$$(f*g)(x) = \int_G f(y)g(xy^{-1})\,dm(y)$$

对于这些群上定义的卷积同样可以给出诸如卷积定理等性质，但是这需要对这些群的表示理论以及调和分析的彼得-外尔定理。

应用

卷积在科学、工程和数学上都有很多应用：

- 代数中，整数乘法和多项式乘法都是卷积。
- 图像处理中，用作图像模糊、锐化、边缘检测。
- 统计学中，加权的滑动平均是一种卷积。
- 概率论中，两个统计独立变量X与Y的和的概率密度函数是X与Y的概率密度函数的卷积。
- 声学中，回声可以用源声与一个反映各种反射效应的函数的卷积表示。
- 电子工程与信号处理中，任一个线性系统的输出都可以通过将输入信号与系统函数（系统的冲激响应）做卷积获得。
- 物理学中，任何一个线性系统（符合叠加原理）都存在卷积。

参见

- 反卷积

- 自相关函数
- 傅里叶变换

外部链接

- PlanetMath上Convolution (http://planetmath.org/?op=getobj&from=objects&id=2790)的資料。
- Visual convolution Java Applet (http://www.jhu.edu/~signals/convolve/index.html)
- Lecture notes, Jian-Jiun Ding (2013), Advanced Digital Signal Processing (http://djj.ee.ntu.edu.tw/ADSP.htm)