DFT matrix(DFT 矩陣)

→在應用數學中，DFT 矩陣是離散傅立葉變換（DFT）的表示形式，它可以作為變換矩陣，可以通過矩陣乘法將其應用於信號。

(一)定義

一個 N 點 DFT 表示為乘法 X=Wx，在哪裡 x 是原始輸入信號， W 是 $N \times N$ 平方 DFT 矩陣，並且 X 是信號的 DFT。

轉換矩陣 W 可以定義為 $W = \left(\frac{\omega^{jk}}{\sqrt{N}}\right)_{j,k=0,\ldots,N-1}$ ，或等效地：

$$W = \frac{1}{\sqrt{N}}\begin{bmatrix} 1 & 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega & \omega^2 & \omega^3 & \cdots & \omega^{N-1} \\ 1 & \omega^2 & \omega^4 & \omega^6 & \cdots & \omega^{2(N-1)} \\ 1 & \omega^3 & \omega^6 & \omega^9 & \cdots & \omega^{3(N-1)} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{N-1} & \omega^{2(N-1)} & \omega^{3(N-1)} & \cdots & \omega^{(N-1)(N-1)} \end{bmatrix}$$

哪裡 $\omega = e^{-2\pi i/N}$ 是原始的第 N 個統一根，其中 $i^2=-1$ 我們可以避免為 ω 使用任何指數的事實 x 我們有身份 $\omega^x = \omega^{x \bmod N}$ 這是統一根的范德蒙矩陣，直到歸一化因子為止。請注意，總和前面的歸一化因子（1/√N）和 ω 中指數的符號只是約定，在某些處理上有所不同。以下所有討論均適用，無論約定如何，最多進行較小的調整。唯一重要的事情是，正向變換和逆變換具有相反符號的指數，並且它們的歸一化因子的乘積為 1 / N。但是，那 1/√N這裡的選擇使生成的 DFT 矩陣統一，這在許多情況下都很方便。

快速傅立葉變換算法利用矩陣的對稱性來減少向量乘以該矩陣的時間，而通常 $O(N^2)$。類似的技術可以應用於矩陣的乘法，例如 Hadamard 矩陣和 Walsh 矩陣。

(二)舉例

兩點

兩點DFT是一種簡單的情況，其中第一項是DC（求和），第二項是AC（差）。

$$W=\frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}$$

第一行執行求和，第二行執行差。

的因素 $1/\sqrt{2}$ 是使變換統一

四點

四點順時針DFT矩陣如下：

$$W=\frac{1}{\sqrt{4}}\begin{bmatrix}\omega^0 & \omega^0 & \omega^0 & \omega^0\\ \omega^0 & \omega^1 & \omega^2 & \omega^3\\ \omega^0 & \omega^2 & \omega^4 & \omega^6\\ \omega^0 & \omega^3 & \omega^6 & \omega^9\end{bmatrix}=\frac{1}{\sqrt{4}}\begin{bmatrix}1 & 1 & 1 & 1\\ 1 & -i & -1 & i\\ 1 & -1 & 1 & -1\\ 1 & i & -1 & -i\end{bmatrix}$$

哪裡 $\omega=e^{-\frac{2\pi i}{4}}=-i$。

八點

兩種情況的第一個非平凡整數冪是八點：

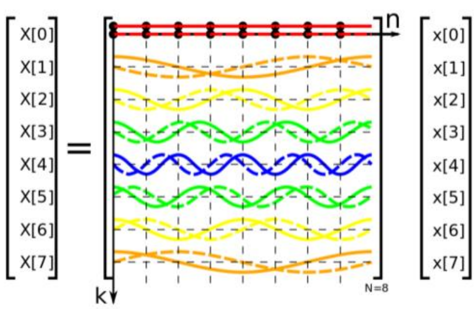
$$W=\frac{1}{\sqrt{8}}\begin{bmatrix}\omega^0 & \omega^0 & \omega^0 & \omega^0 & \omega^0 & \omega^0 & \omega^0 & \omega^0\\ \omega^0 & \omega^1 & \omega^2 & \omega^3 & \omega^4 & \omega^5 & \omega^6 & \omega^7\\ \omega^0 & \omega^2 & \omega^4 & \omega^6 & \omega^8 & \omega^{10} & \omega^{12} & \omega^{14}\\ \omega^0 & \omega^3 & \omega^6 & \omega^9 & \omega^{12} & \omega^{15} & \omega^{18} & \omega^{21}\\ \omega^0 & \omega^4 & \omega^8 & \omega^{12} & \omega^{16} & \omega^{20} & \omega^{24} & \omega^{28}\\ \omega^0 & \omega^5 & \omega^{10} & \omega^{15} & \omega^{20} & \omega^{25} & \omega^{30} & \omega^{35}\\ \omega^0 & \omega^6 & \omega^{12} & \omega^{18} & \omega^{24} & \omega^{30} & \omega^{36} & \omega^{42}\\ \omega^0 & \omega^7 & \omega^{14} & \omega^{21} & \omega^{28} & \omega^{35} & \omega^{42} & \omega^{49}\end{bmatrix}=\frac{1}{\sqrt{8}}\begin{bmatrix}1 & 1 & 1 & 1 & 1 & 1 & 1 & 1\\ 1 & \omega & -i & -i\omega & -1 & -\omega & i & i\omega\\ 1 & -i & -1 & i & 1 & -i & -1 & i\\ 1 & -i\omega & i & \omega & -1 & i\omega & -i & -\omega\\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1\\ 1 & -\omega & -i & i\omega & -1 & \omega & i & -i\omega\\ 1 & i & -1 & -i & 1 & i & -1 & -i\\ 1 & i\omega & i & -\omega & -1 & -i\omega & -i & \omega\end{bmatrix}$$

哪裡

$$\omega=e^{-\frac{2\pi i}{8}}=\frac{1}{\sqrt{2}}-\frac{i}{\sqrt{2}}$$

（注意 $\omega^{8+n}=\omega^n$）

下圖將DFT描述為矩陣乘法，其中矩陣的元素由複指數樣本描述：



實部（餘弦波）用實線表示，虛部（正弦波）用虛線表示。

第一行全為（按 1/√8以確保統一性），因此它“測量” 了輸入信號中的直流分量。下一行是複指數（即分數頻率為-1/8 的一個信號）的負一個週期的八個樣本，因此它“測量”在分數頻率+1/8 中有多少“強度”。信號。回想一下，一個匹配的濾波器將信號與我們想要的東西的時間反向版本進行比較，因此當我們尋找 fracfreq 時。1/8 我們與 fracfreq 進行比較。–1/8，所以這行是負頻率的原因。下一行是一個複雜的指數的負兩個週期，在八個位置採樣，因此它的分數頻率為–1/4，因此“測量”

以下總結了 8 點 DFT 如何按分數頻率逐行工作：

- 0 測量信號中有多少 DC
- –1/8 測量多少信號的分數頻率為+1/8
- −1/4 測量多少信號的分數頻率為+1/4
- −3/8 測量有多少信號的分數頻率為+3/8
- −1/2 測量多少信號的分數頻率為+1/2
- −5/8 測量有多少信號的分數頻率為+5/8
- −3/4 測量多少信號的分數頻率為+3/4
- −7/8 測量多少信號的分數頻率為+7/8

等效地，可以說最後一行的分數頻率為+1/8，因此可以測量多少信號的分數頻率為–1/8。這樣，可以說矩陣的頂部行“測量”信號中的正頻率含量，而底部行測量信號中的負頻率分量。

(三)transform 變換

DFT 是（或者可以是通過適當選擇縮放比例）單一變換，即一種保留能量的變換。實現單一性的適當縮放比例選擇是 1/√N，因此物理域中的能量將與傳立葉域中的能量相同，即滿足 Parseval 定理。

(四)一個極限情況，傅立葉運算幅

傅立葉變換的概念很容易概括。N點 DFT 的這種形式化一般化可以通過將 N任意取大來實現。在極限情況下，嚴格的數學機制將此類線性算符視為所謂的積分變換。在這種情況下，如果我們製作一個非常大的矩陣，在行中具有復雜的指數（即，餘弦實部和正弦虛部），並且無限制地提高了分辨率，那麼我們將接近第二類 Fredholm 積分方程的核，即傅立葉運算符定義連續的傅立葉變換。該連續傅立葉算子的矩形部分可以顯示為圖像，類似於 DFT 矩陣，如右圖所示，其中灰度像素值表示數值。

DFT(離散傅立葉變換)

→離散傅立葉變換，是傅立葉變換在時域和頻域上都呈離散的形式，將信號的時域採樣變換為其 DTFT 的頻域採樣。在形式上，變換兩端（時域和頻域上）的序列是有限長的，而實際上這兩組序列都應當被認為是離散週期信號的主值序列。即使對有限長的離散信號作 DFT，也應當將其看作其週期延拓的變換。在實際應用中通常採用快速傅立葉變換計算 DFT。

(一)定義

對於N點序列$\{x[n]\}_{0\le n<N}$，它的離散傅立葉變換（DFT）為

$$\hat{x}[k]=\sum_{n=0}^{N-1} e^{-i\frac{2\pi}{N}nk}x[n] \qquad k=0,1,\ldots,N-1.$$

其中e是 自然對數的底數 i是 虛數單位 通常以符號$\mathcal{F}$表示這一變換，即

$$\hat{x}=\mathcal{F}x$$

離散傅立葉變換的逆變換（IDFT）為：

$$x\left[n\right]=\frac{1}{N}\sum_{k=0}^{N-1} e^{i\frac{2\pi}{N}nk}\hat{x}[k] \qquad n=0,1,\ldots,N-1.$$

可以記為：

$$x=\mathcal{F}^{-1}\hat{x}$$

實際上，DFT 和 IDFT 變換式中和式前面的歸一化係數並不重要。在上面的定義中，DFT 和 IDFT 前的係數分別為 1 和 1/N。有時會將這兩個係數都改成 1/√N。

(二)從連續到離散

連續時間信號$x(t)$以及對應的連續傅立葉變換$\hat{x}(\omega)$都是連續函數。由於數字系統只能處理有限長的離散信號，因此必須將x和$\hat{x}$都離散化，並且建立對應的傅立葉變換。

假設$x(t)$時限於$[0,L]$，再通過時域採樣將$x(t)$離散化，就可以得到有限長離散信號，記為$x_{discrete}(t)$。設採樣週期為T，則時域採樣點數$N=\frac{L}{T}$。

$$x_{discrete}(t)=x(t)\sum_{n=0}^{N-1}\delta(t-nT)=\sum_{n=0}^{N-1}x(nT)\delta(t-nT)$$

它的傅立葉變換為

$$\hat{x}_{discrete}(\omega)=\sum_{n=0}^{N-1}x(nT)\mathcal{F}\delta(t-nT)=\sum_{n=0}^{N-1}x(nT)e^{-in\omega T}$$

這就是$x(t)$在時域採樣後的連續傅立葉變換，也就是離散時間傅立葉變換，它在頻域依然是連續的。

下面將頻域信號轉化為有限長離散信號。與對時域信號的處理類似，假設頻域信號是帶限的，再經過離散化，即可得到有限長離散信號。依據採樣定理，時域採樣若要能完全重建原信號，頻域信號$\hat{x}(\omega)$應當帶限於$(0,\frac{1}{2T})$。由於時域信號時限於$[0,L]$，由採樣定理以及時頻對偶的關係，頻域的採樣間隔應為$1/L$。故，頻域採樣點數為：

$$\frac{1/T}{1/L}=N$$

即頻域採樣的點數和時域採樣同為N，頻域採樣點為$\{\omega_k=2\pi k/NT\}_{0\le k<N}$。 而DTFT在頻域上採樣：

$$\hat{x}[k]=\hat{x}_{discrete}(\omega_k)=\frac{1}{T}\sum_{n=0}^{N-1}x[nT]e^{-i\frac{2\pi}{N}nk}$$

令$T=1$，將其歸一化，就得到前面定義的離散傅立葉變換。因此，DFT就是先將信號在時域離散化，求其連續傅立葉變換後，再在頻域離散化的結果。

(三)性質

完全性

離散傅立葉變換是可逆的線性變換

$$\mathcal{F}:\mathbf{C}^N\to\mathbf{C}^N$$

其中**C**表示複數集。即，任意*N*-維複向量都存在DFT和IDFT，而且其DFT和IDFT也是*N*-維複向量。

正交性

向量組exp（2π*i kn/N*）是*N*-維複數空間上的一組正交基：

$$\sum_{n=0}^{N-1}\left(e^{\frac{2\pi i}{N}kn}\right)\left(e^{-\frac{2\pi i}{N}k'n}\right)=N\,\delta_{kk'}$$

移位定理

$$若\mathcal{F}(\{x_n\})_k=X_k$$

$$則\mathcal{F}(\{x_n e^{\frac{2\pi i}{N}nm}\})_k=X_{k-m}$$

$$且有\mathcal{F}(\{x_{n-m}\})_k=X_k e^{-\frac{2\pi i}{N}km}$$

(四)應用

1. 快速傅立葉變換
2. 頻譜分析
3. 數據壓縮
4. 解偏微分方程式
5. 長整數與多項式乘法
6. OFDM(正交頻分復用)

(五)特殊例子

數論轉換是離散傅利葉轉換(DFT)的一個特例 $F=Z/p$，此運算是根據模運算 而取得的，p需要是質數 ，如此可以建構出有限體，並且存在 n 個可以整除 $(p-1)$ 的實數根。如此可以得到$p=k\cdot n+1$，k 為正整數。令 ω 為第 $(p-1)$ 個單位根，則第 n 個單位根 α 可以表示成 $\alpha=\omega^k$ 。另外，再令N 為 α 次方 p 模運算之循環個數。

則數論矩陣為

$$\begin{bmatrix} \underline{F} \end{bmatrix} = \begin{bmatrix} \underline{\alpha^{cr}} \end{bmatrix} \begin{bmatrix} \underline{f} \end{bmatrix}$$

c 與 r 各為 Column 與 Row 的指數(index), 令 C 與 R 各代表Column 與 Row 的總數，則$C=R=N$，$c\propto\{0,1,\ldots,C-1\},r\propto\{0,1,\ldots,R-1\}$.

舉個例子來說，當 $p=5,\alpha=2$ 則

$$\alpha^1=2 \qquad (mod \quad 5)$$

$$\alpha^2=4 \qquad (mod \quad 5)$$

$$\alpha^3=3 \qquad (mod \quad 5)$$

$$\alpha^4=1 \qquad (mod \quad 5)$$

因為 α^4 經 5 的模運算為 1 ，因此可以判定此情況為 4 個次方一循環，所以 $N=4$，N可以整除 $(p-1)$。則數論矩陣可以表示成

$$\begin{bmatrix} F(0) \\ F(1) \\ F(2) \\ F(3) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 4 & 3 \\ 1 & 4 & 1 & 4 \\ 1 & 3 & 4 & 2 \end{bmatrix} \begin{bmatrix} f(0) \\ f(1) \\ f(2) \\ f(3) \end{bmatrix}$$

FFT(快速傅立葉變換)

(一)定義和速度

用 FFT 計算 DFT 會得到與直接用 DFT 定義計算相同的結果；最重要的區別是 FFT 更快。令 $x_0, ..., x_{N-1}$ 為複數。DFT 由下式定義

$$X_k = \sum_{n=0}^{N-1} x_n e^{-i2\pi k \frac{n}{N}} \qquad k = 0, \ldots, N-1.$$

直接按這個定義求值需要 O(N^2) 次運算：X_k 共有 N 個輸出，每個輸出需要 N 項求和。直接使用 DFT 運算需使用 N 個複數乘法(4N 個實數乘法)與 N-1 個複數加法(2N-2 個實數加法)，因此，計算使用 DFT 所有 N 點的值需要 N^2 複數乘法與 N^2-N 個複數加法。FFT 則是能夠在 O(N log N) 次操作計算出相同結果的任何方法。更準確的說，所有已知的 FFT 演算法都需要 O(N log N) 次運算（技術上 O 只標記上界），雖然還沒有已知的證據證明更低的複雜度是不可能的。

要說明 FFT 節省時間的方式，就得考慮複數相乘和相加的次數。直接計算 DFT 的值涉及到 N^2 次複數相乘和 $N(N-1)$ 次複數相加（可以通過削去瑣碎運算（如乘以 1）來節省 O(N) 次運算）。眾所周知的基 2 庫利-圖基演算法，N 為 2 的冪，可以只用 $(N/2)\log_2(N)$ 次複數乘法（再次忽略乘以 1 的簡化）和 $N\log_2(N)$ 次加法就可以得到相同結果。在實際中，現代電腦通常的實際效能通常不受算術運算的速度和對複雜主體的分析主導，但是從 O(N^2) 到 O(N log N) 的母體改進仍然能夠體現出來。

(二)一般的簡化理論

假設一個M*N型矩陣**S**可分解成列向量以及行向量相乘：

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

若$\begin{bmatrix}a_1 & a_2 & \cdots & a_m\end{bmatrix}^T$有$M_0$個相異的非平凡值（$a_m \neq \pm 2^k, a_m \neq \pm 2^k a_n$ where $m \neq n$）

$\begin{bmatrix}b_1 & b_2 & \cdots & b_n\end{bmatrix}$有$N_0$個相異的非平凡值

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：$Z_a = b_1X[1]+b_2X[2]+\cdots+b_nX[N]$

Step 2：$Z[1]=a_1Z_a, Z[2]=a_2Z_a, \cdots, Z[N]=a_mZ_a$

簡化理論的變型：

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

S_1也是一個M*N的矩陣。

若S_1有P_1個值不等於0，則**S**的乘法量上限為$M_0+N_0+P_1$。

(三)快速傅立葉變換乘法量的計算

假設$N=P_1\times P_2\times\cdots\times P_k$，其中$P_1, P_2, \cdots, P_k$彼此互質

$\mathbf{P_k}$點DFT的乘法量為$\mathbf{B_k}$，則$\mathbf{N}$點DFT的乘法量為：

$$\frac{N}{P_1}B_1+\frac{N}{P_2}B_2+\cdots\cdots+\frac{N}{P_k}B_k$$

假設$\mathbf{N}=\mathbf{P^c}$，P是一個質數。

若$\mathbf{N_1}=\mathbf{P^a}$點的DFT需要的乘法量為$\mathbf{B_1}$

且$n_1\times n_2$當中（$n_1=0,1,\cdots,N_1-1,\quad n_2=0,1,\cdots,N_2-1$）

有D_1個值不為$\frac{N}{12}$及$\frac{N}{8}$的倍數，

有D_2個值為$\frac{N}{12}$及$\frac{N}{8}$的倍數，但不為$\frac{N}{4}$的倍數，

則N點DFT的乘法量為：

$$\mathbf{N_2B_1+N_1B_2+3D_1+2D_2}$$

(四)複雜度以及運算量的極限

長久以來，人們對於求出快速傅立葉變換的複雜度下限以及需要多少的運算量感到很有興趣，而實際上也還有許多問題需要解決。即使是用較簡單的情形，即2^k點的DFT，也還沒能夠嚴謹的證明出FFT至少需要$\Omega(N \log N)$（不比$N \log N$小）的運算量，目前也沒有發現複雜度更低的演算法。通常數學運算量的多寡會是運算效率好壞最主要的因素，但在現實中，有許多因素也會有很大的影響，如快取記憶體以及CPU均有很大的影響。

在1978年，Winograd率先導出一個較嚴謹的FFT所需乘法量的下限：$\Theta(N)$。當$N = 2^k$時，DFT只需要$4N - 2\log_2^2 N - 2\log_2 N - 4$次無理實數的乘法即可以計算出來。更詳盡，且也能趨近此下限的演算法也一一被提出（Heideman & Burrus, 1986; Duhamel, 1990）。很可惜的是，這些演算法，都需要很大量的加法計算，目前的硬體無法克服這個問題。

對於所需加法量的數目，雖然我們可以在某些受限制的假設下，推得其下限，但目前並沒有一個精確的下限被推導出來。1973年，Morgenstern在乘法常數趨近巨大的情形下（對大部分的FFT演算法為真，但不是全部）推導出加法量的下限：$\Omega\left(N \log N\right)$。Pan（1986）在假設FFT演算法的不同步的情形有其極限下證明出加法量的下限$\Omega(NlogN)$，但一般來說，此假設相當的不明確。長度為$N = 2^k$的情形下，在某些假設下，Papadimitriou（1979）提出使用Cooley-Tukey演算法所需的複數加法量$N \log_2 N$是最少的。到目前為止，在長度為$N = 2^k$情況，還沒有任何FFT的演算法可以讓複數的加法量比$N \log_2 N$還少。

還有一個問題是如何把乘法量與加法量的總和最小化，有時候稱作"演算複雜度"（在這裡考慮的是實際的運算量，而不是漸近複雜度）。同樣的，沒有一個嚴謹下限被證明出來。從1968年開始，$N = 2^k$點DFT而言，split-radix FFT演算法需要最少的運算量，在$N > 1$的情形下，其需要$4N \log_2 N - 6N + 8$個乘法運算以及加法運算。最近有人導出更低的運算量：$\frac{34}{9} N \log_2 N$。（Johnson and Frigo, 2007; Lundy and Van Buskirk, 2007）

大多數嘗試要降低或者證明FFT複雜度下限的人都把焦點放在複數資料輸入的情況，因其為最簡單的情形。但是，複數資料輸入的FFT演算法，與實數資料輸入的FFT演算法，離散餘弦轉換（DCT），離散哈特列轉換（DHT），以及其他的演算法，均有很大的關連性。故任何一個演算法，在複雜度上有任何的改善的話，其他的演算法複雜度也會馬上獲得改善（Duhamel & Vetterli, 1990）。

參考資料

1.

https://en.wikipedia.org/wiki/DFT_matrix

2.

https://zh.wikipedia.org/wiki/%E7%A6%BB%E6%95%A3%E5%82%85%E9%87%8C%E5%8F%B6%E5%8F%98%E6%8D%A2

3.

https://zh.wikipedia.org/wiki/%E5%BF%AB%E9%80%9F%E5%82%85%E9%87%8C%E5%8F%B6%E5%8F%98%E6%8D%A2