从 DFT 到 FFT

关键词：离散傅里叶变换 快速傅里叶变换

摘要：快速傅里叶变换算法是现代工程技术中广泛使用的一种算法。文章从离散傅里叶变换与复指数函数引入，分析了快速傅里叶变换的基本思路，最后介绍了实际运算用到的蝶形网络。

傅里叶变换广泛应用于信号的频谱分析中，在现代工程中有着重大的意义。但是，作为信息时代核心的计算机系统只能够处理离散的数值，为了充分利用计算机的性能进行工程分析，很有必要把傅里叶变换从连续空间带到离散空间上来。

一 离散傅里叶变换

离散傅里叶变换（Discrete Fourier Transform，缩写为 DFT），是傅里叶变换在时域和频域上都呈离散的形式，将信号的时域采样变换为其 DTFT 的频域采样。从连续傅里叶变换可以推导到 DFT 的正变换式为：

$$X(k) = DFT[x(n)] = \sum_{n=0}^{N-1} x(n)W^{nk} \qquad (0 \le k \le N-1)$$

其中 $W = e^{-j(\frac{2\pi}{N})}$

为了分析方便，转换成下面的矩阵运算：

$$\begin{bmatrix} X(0) \\ X(1) \\ \dots \\ X(N-1) \end{bmatrix} = \begin{bmatrix} W^0 & W^0 & W^0 & \dots & W^0 \\ W^0 & W^{1\times1} & W^{2\times1} & \dots & W^{(N-1)\times1} \\ \dots & \dots & \dots & & \dots \\ W^0 & W^{1\times(N-1)} & W^{2\times(N-1)} & \dots & W^{(N-1)\times(N-1)} \end{bmatrix} \begin{bmatrix} x(0) \\ x(1) \\ \dots \\ x(N-1) \end{bmatrix}$$

显然，为了计算 X(k)，需要进行 N 次乘法（忽略 W_N^k 本身值的计算）与 N-1 次加法。在计算机中，一般乘法运算要明显慢于加法，所以下面都仅仅考虑乘法的次数。那么完成整个 DFT 需要进行 N^2 次乘法。

这种朴素方法易于理解与计算，但是 $O(N^2)$ 的时间复杂度意味着计算时间随序列长度增加而增加的速度是相当高的平方级。这严重阻碍了 DFT 的实际应用。

二 W_N^k 的性质

我们用 W_N^k 表示 $e^{2\pi j\frac{k}{N}}$ 以简化公式，通过对复指数 $e^{2\pi j\frac{k}{N}}$ 的研究，可以发现 W_N^k 的一些性质。通过欧拉公式可以发现其表示复平面中的单位圆，那么显然有：

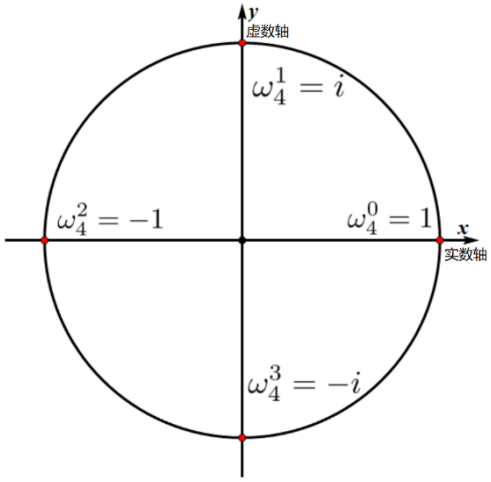
$$W_N^N = W_N^0 = 1$$

在单位圆上转整数圈回到原处（周期性）：

$$W_N^{k+N} = W_N^k$$

在单位圆上转半圈矢量方向相反：

$$W_N^{k+\frac{N}{2}} = -W_N^k \text{ (消去引理)}$$



利用这些性质可以消去朴素的 DFT 算法中的重复步骤。

三 快速傅里叶变换

快速傅里叶变换（Fast Fourier Transform，缩写为 FFT），是快速计算串行的离散傅里叶变换或其逆变换的方法[1]。快速傅里叶变换广泛的应用于工程、科学和数学领域。这里的基本思想在 1965 年才得到普及，但早在 1805 年就已推导出来[2]。库利-图基算法是最常用的快速傅里叶变换（Fast Fourier Transform，缩写为 FFT）算法。这一算法利用分治法把 DFT 求解优化到了 O(NlogN)。在 1965 年 J.W. Cooley 和 J. W. Tukey 合作发表的 An algorithm for the machine calculation of complex Fourier series 中介绍这种方法以及 FFT 的基本思路。虽然后来发现，实际上高斯在 1805 年就已经提出了这样的算法。

其基本思路是先将偶数与奇数样本点序列分开，

$$X\left[k\right]=\sum_{n=0}^{N-1}x\left[n\right]W_N^{kn}=\sum_{n=0}^{\frac{N}{2}-1}x\left[2n\right]W_N^{k(2n)}+W_N^k\sum_{n=0}^{\frac{N}{2}-1}x\left[2n+1\right]W_N^{k(2n)}$$

其中$W_N^{k(2n)}=e^{-j\frac{2\pi}{N}2n}=e^{-j\frac{2\pi}{\frac{N}{2}}n}=W_{\frac{N}{2}}^{kn}$

方便起见，令：

$$X(k)=G(k)+W_N^kH(k)$$

$$G(k)=\sum_{r=0}^{\frac{N}{2}-1}x(2r)W_{\frac{N}{2}}^{rk}$$

$$H(k)=\sum_{r=0}^{\frac{N}{2}-1}x(2r+1)W_{\frac{N}{2}}^{rk}$$

由消去引理易得：

$$G(k+\frac{N}{2})=G(k)$$

$$H(k+\frac{N}{2})=H(k)$$

$$X(k+\frac{N}{2})=G(k+\frac{N}{2})-W_N^kH(k+\frac{N}{2})=G(k)-W_N^kH(k)$$

这意味着 具备着一定的周期性，我们只需要计算一半的 。但是这还没有结束，留意 G(k)、H(k) 与 X(k) 的形式非常相似，令：

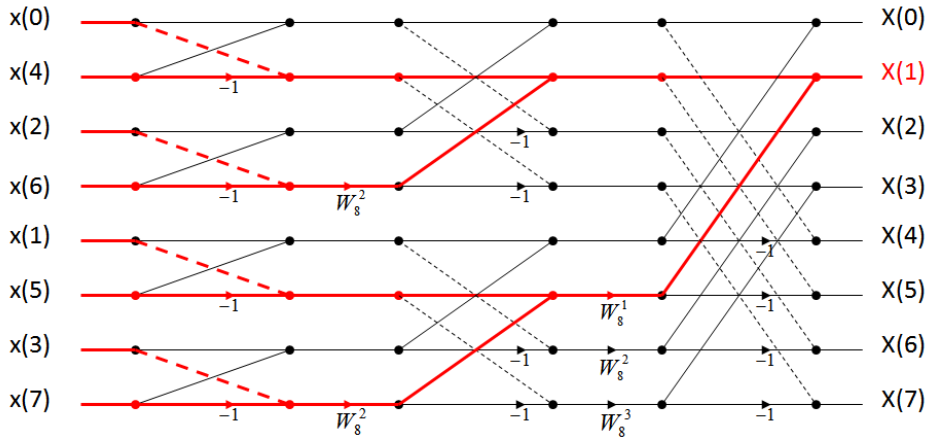
$$x_1(r)=x(2r)$$

则：

$$G(k)=\sum_{r=0}^{\frac{N}{2}-1}x_1(r)W_{\frac{N}{2}}^{rk}$$

由此可得，G(k) 可以像 X(k) 一样继续奇偶分解，H(k) 同理。

实际计算时正向进行，为了方便递推引出蝶形运算：



$$X(1) = x(0) + x(1)(W_8^1) + x(2)(W_8^2) + x(3)(W_8^2 W_8^1) + x(4)(-1) + \ldots$$
$$x(5)(-W_8^1) + x(6)(-W_8^2) + x(7)(-W_8^2 W_8^1)$$

其中输入点列按位反转（Bit Reversal）排序：首先依据二进制进行编号，然后对各个编号按位倒置并按此重新排序。例如，对于一个 8 点变换：

001 倒置以后变成 100
000 → 000
001 → 100
010 → 010
011 → 110
100 → 001
101 → 101
110 → 011
111 → 111

倒置后的编号为 {0,4,2,6,1,5,3,7}。依此顺序输入蝶形网络计算出的变换序列就是正序的。

相较于 DFT，FFT 在 N 很大时，计算量节省相当可观。不同点数时 DFT 与 FFT 的复数乘法次数和比较见下表[3]

N	8	16	32	64	128	256	512	1024	2048
DFT	64	256	1024	4096	16384	65536	262144	1048576	4194304
FFT	12	32	80	192	448	1024	2034	5120	11264

参考文献：

[1] 杨毅明. 数字信号处理（第 2 版）. 北京: 机械工业出版社. 2017 年: 第 95 页. ISBN 9787111576235.
[2] Heideman, M. T.; Johnson, D. H.; Burrus, C. S. Gauss and the history of the fast Fourier transform. IEEE ASSP Magazine. 1984, 1 (4): 14–21. doi:10.1109/MASSP.1984.1162257.
[3] 李勇. 数字信号处理原理与应用 西安 西北工业大学出版社 2016 年: 第 96 页. ISBN 9787561248287