第24章 快速傅里叶变换原理（FFT）

在数字信号处理中常常需要用到离散傅立叶变换(DFT)，以获取信号的频域特征。尽管传统的 DFT 算法能够获取信号频域特征，但是算法计算量大，耗时长，不利于计算机实时对信号进行处理。因此至 DFT 被发现以来，在很长的一段时间内都不能被应用到实际的工程项目中，直到一种快速的离散傅立叶计算方法——FFT，被发现，离散是傅立叶变换才在实际的工程中得到广泛应用。需要强调的是，FFT 并不是一种新的频域特征获取方式，而是 DFT 的一种快速实现算法。

特别声明：FFT 原理的讲解来自网络和书籍。

24.1 FFT 的由来

离散傅里叶变换（Discrete Fourier Transform，DFT）是数字信号处理最重要的基石之一，也是对信号进行分析和处理时最常用的工具之一。在 200 多年前法国数学家、物理学家傅里叶提出后来以他名字命名的傅里叶级数之后，用 DFT 这个工具来分析信号就已经为人们所知。历史上最伟大的数学家之一。

欧拉是第一个使用“函数”一词来描述包含各种参数的表达式的人，例如：y = f(x)。他是把微积分应用于物理学的先驱者之一。 给出了一个用实变量函数表示傅立叶级数系数的方程； 用三角级数来描述离散声音在弹性媒介中传播，发现某些函数可以通过余弦函数之和来表达。 但在很长时间内，这种分析方法并没有引起更多的重视 ,最主要的原因在于这种方法运算量比较大。直到 1965 年 ,Cooley 和 Tukey 在《计算机科学 》发表著名的《机器计算傅立叶级数的一种算法》论文，FFT 才开始大规模应用。

那个年代，有个肯尼迪总统科学咨询委员会。其中有项研究主题是，对苏联核测试进行检测，Tukey 就是其中一员。美国/苏联核测试提案的批准，主要取决于不实地访问核测试设施而做出检测的方法的发展。其中一个想法是，分析离海岸的地震计情况，这种计算需要快速算法来计算 DFT。其它应用是国家安全，如用声学探测远距离的核潜艇。所以在军事上，迫切需要一种快速的傅立叶变换算法，这也促进了 FFT 的正式提出。

FFT 的这种方法充分利用了 DFT 运算中的对称性和周期性 从而将 DFT 运算量从 N2 减少到 N*log2N。当 N 比较小时，FFT 优势并不明显。但当 N 大于 32 开始，点数越大，FFT 对运算量的改善越明显。比如当 N 为 1024 时，FFT 的运算效率比 DFT 提高了 100 倍。在库利和图基提出的 FFT 算法中，其基本原理是先将一个 N 点时域序列的 DFT 分解为 N 个 1 点序列的 DFT 然后将这样计算出来的 N 个 1 点序列 DFT 的结果进行组合，得到最初的 N 点时域序列的 DFT 值。实际上，这种基本的思想很早就由德国伟大的数学家高斯提出过，在某种情况下，天文学计算（也是现在 FFT 应用的领域之一）与等距观察的有限集中的行星轨道的内插值有关。由于当时计算都是靠手工，所以产生一种快速算法的迫切需要。 而且，更少的计算量同时也代表着错误的机会更少，正确性更高。高斯发现，一个富氏级数有宽度 N=N1*N2，可以分成几个部分。计算 N2 子样本 DFT 的 N1 长度和 N1 子样本 DFT 的 N2 长度。只是由于当时尚欠东风——计算机还没发明。在 20 世纪 60 年代，伴随着计算机的发展和成熟，库利和图基的成果掀起了数字信号处理的革命，因而 FFT 发明者的桂冠才落在他们头上。

之后，桑德（G.Sand）-图基等快速算法相继出现，几经改进，很快形成了一套高效运算方法，这就是现在的快速傅立叶变换（FFT）。这种算法使 DFT 的运算效率提高 1 到 2 个数量级，为数字信号处理技术应用于各种信号的实时处理创造了良好的条件，大大推进了数学信号处理技术。1984 年，法国的杜哈梅（P.Dohamel）和霍尔曼（H.Hollamann）提出的分裂基块快速算法，使运算效率进一步提高。

库利和图基的 FFT 算法的最基本运算为蝶形运算，每个蝶形运算包括两个输入点，因而也称为基-2 算法。在这之后，又有一些新的算法，进一步提高了 FFT 的运算效率，比如基-4 算法，分裂基算法等。这些新算法对 FFT 运算效率的提高一般在 50%以内，远远不如 FFT 对 DFT 运算的提高幅度。从这个意义上说，FFT 算法是里程碑式的。可以说，正是计算机技术的发展和 FFT 的出现，才使得数字信号处理迎来了一个崭新的时代。除了运算效率的大幅度提高外，FFT 还大大降低了 DFT 运算带来的累计量化误差，这点常为人们所忽略。

24.2 直接计算 DFT 的问题及改进路径

24.2.1 问题的提出

设有限长序列 x(n)，非零值长度为 N，若对 x(n)进行一次 DFT 运行，共需要多大的运算工作量。

24.2.2 DFT 的运算量

DFT 和 IDFT 的变换式：

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

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

注意：

1. x(n)为复数，$W_N^{nk} = e^{-j\frac{2\pi}{N}nk}$也为复数。
2. DFT 与 IDFT 的计算量相当。

下面以 DFT 为例说明计算量：

计算机运算时（编程实现）：

k = 0 X(0) = x(0)W_N^{00} + x(1)W_N^{10} + ... + x(N-1)$W_N^{(N-1)0}$

k = 1 X(1) = x(0)W_N^{01} + x(1)W_N^{00} + ... + x(N-1)$W_N^{(N-1)1}$

k = 2 X(2) = x(0)W_N^{00} + x(1)W_N^{00} + ... + x(N-1)$W_N^{(N-1)2}$

.

.

k = N-1 X(N-1) = x(0)$W_N^{0(N-1)}$ + x(1)$W_N^{1(N-1)}$ + ... + x(N-1)$W_N^{(N-1)(N-1)}$

N 个点

N 次复乘，N-1 次复加

由上面的结算可得 DFT 的计算量如下：

	复数乘法	复数加法
一个 X(k)	N	N-1
N 个 X(k)(N 点 DFT)	N^2	N(N-1)

复数乘法的计算量：(a+jb)(c+jd)=(ab-bd)+j(bc+ad)

	实数乘法	复数加法
一次复乘	4	2
一次复加		2
一个 X(k)	4N	2N+2(N-1)=2(2N-1)
N 个 X(k) (N 点 DFT)	$4N^2$	

下面通过两个实例来说明计算量：

例一：计算一个 N 点 DFT，共需N^2次复乘。以做一次复乘 1μs计算，若 N=4096，所需时间为

$$(4096)^2 = 16777216\mu s \approx 17s$$

例二：石油勘探，有 24 个通道的记录，每通道波形记录长度为 5 秒，若每秒抽样 500 点/秒。

1. 每道总抽样点数：500*5 = 2500 点。
2. 24 道总抽样点数：24*2500=6 万点。
3. DFT 复乘运算时间：$N^2=(60000)^2=36*10^8$ 次。

   $(60000)^2=36*10^8\mu s=3600s$

由于计算量大，且要求相当大的内存，难以实现实时处理，限制了 DFT 的应用，人们一直在寻求一种能提高 DFT 运算速度的方法。

FFT 便是 Cooley 和 Tukey 在 1995 年提出来的快速算法，它可以使运算速度提高几百倍，从而使数字信号处理成为一个新兴的应用学科。

24.3 改善 DFT 运算效率的基本途径

1、利用 DFT 运算的系数W_N^{kn}的固有对称性和周期性，改善 DFT 的运算效率。

1）对称性

2）周期性

3）可约性

W_N^{kn}的特性　$W_N^{kn} = e^{-j\frac{2\pi}{N}nk}$

对称性　$(W_N^{kn})^* = W_N^{-nk} = W_N^{(N-n)k} = W_N^{n(N-k)}$

$W_N^{Nk} \cdot W_N^{-nk}$　　$W_N^{nN} \cdot W_N^{-nk}$

周期性　$W_N^{kn} = W_N^{(N+n)k} = W_N^{n(N+k)}$

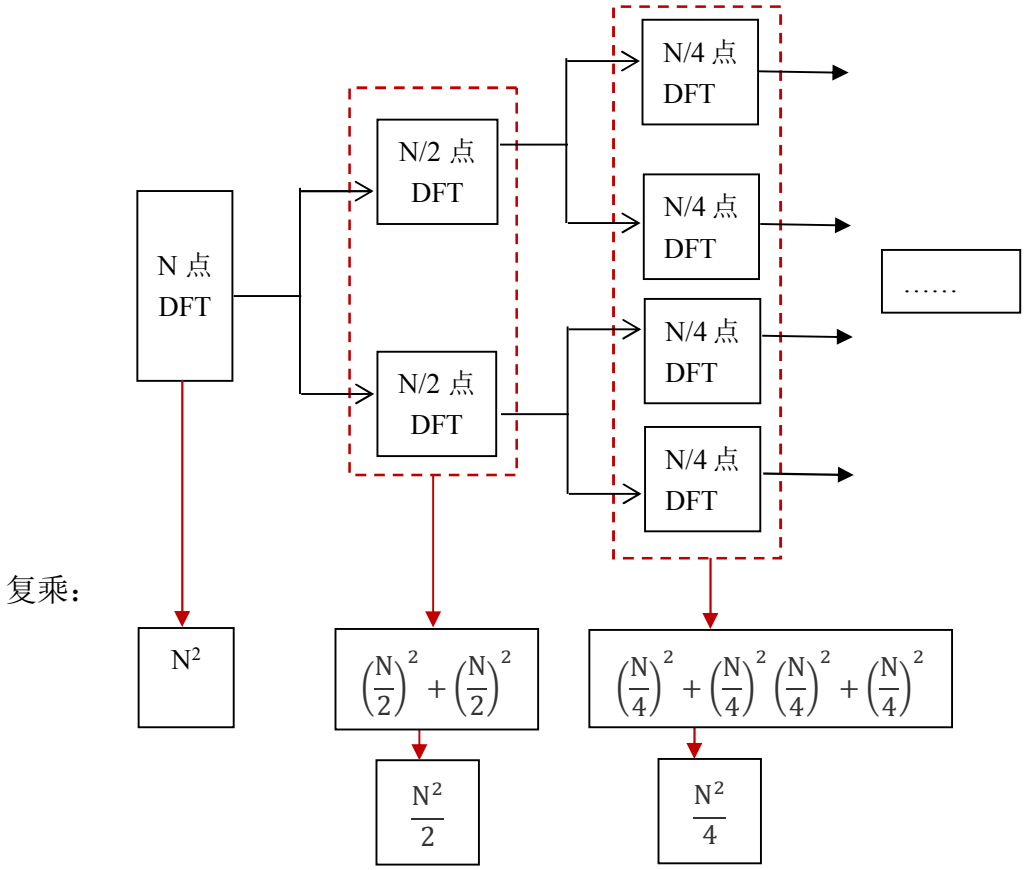
可约性　$W_N^{kn} = W_{mN}^{mnk}$　　$W_N^{kn} = W_{N/m}^{nk/m}$

$e^{-j\frac{2\pi}{mN}mnk}$　　$e^{-j\frac{2\pi}{N}\cdot\frac{N}{2}} = e^{-j\pi}$=-1

特殊点　$W_N^0 = 1$　　$W_N^{N/2} = -1$　　$W_N^{(k+N/2)} = -W_N^k$

2、将长序列 DFT 利用对称性和周期性分解为短序列 DFT 的思路

因为 DFT 的运算量与 N^2 成正比，如果一个大点数 N 的 DFT 能分解为若干小点数 DFT 的组合，则显然可以达到减少运算工作量的效果。



FFT 算法的基本思想：

- 利用 DFT 系数的特性，合并 DFT 运算中的某些项。
- 把长序列 DFT→短序列 DFT，从而减少运算量。

FFT 算法分类：

时间抽选法

DIT: Decimation-In-Time

频率抽选法

DIF: Decimation-In-Frequency

24.4 按时间抽选的基 2-FFT 算法

24.4.1 算法原理

设输入序列长度为 N = 2^M（M 为正整数），将该序列按时间顺序的奇偶分解为越来越短的子序列，称为基 2 按时间抽取的 FFT 算法。也称为 Coolkey-Tukey 算法。

其中基 2 表示：N = 2^M，M 为整数。若不满足这个条件，可以人为地加上若干零值（加零补长）使

其达到 N = 2^M。

24.4.2 算法步骤

- **分组，变量置换**

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

先将 x(n)按 n 的奇偶分为两组，作变量置换：

当 n = 偶数时，令 n = 2r；

当 n = 奇数时，令 n = 2r+1；

得到：

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

$x(2r+1)=x_2(r)$ r = 0,..... $^N/_2$-1

- **分组，变量置换**

X(k) = DFT[x(n)]

$$= \sum_{n=0}^{N-1} x(n)W_N^{nk}$$

$$= \sum_{\substack{n=0 \\ n\text{ 为偶数}}}^{N-1} x(n)W_N^{nk} + \sum_{\substack{n=0 \\ n\text{ 为奇数}}}^{N-1} x(n)W_N^{nk}$$

$$= \sum_{r=0}^{N/2-1} x(2r)W_N^{2rk} + \sum_{r=0}^{N/2-1} x(2r+1)W_N^{(2r+1)k}$$

$$= \sum_{r=0}^{N/2-1} x_1(2r)W_N^{2rk} + W_N^k \sum_{r=0}^{N/2-1} x_2(2r+1)W_N^{(2r+1)k}$$

由于$W_N^{nk} = e^{-j\frac{2\pi}{N}n} = e^{-j\frac{2\pi}{N/2}n} = W_{N/2}^n$

$$X(k) = \sum_{r=0}^{N/2-1} x_1(2r)W_N^{2rk} + W_N^k \sum_{r=0}^{N/2-1} x_2(2r+1)W_N^{(2r+1)k}$$

$$= \sum_{r=0}^{N/2-1} x_1(2r)W_N^{2rk} + W_N^k \sum_{r=0}^{N/2-1} x_2(2r+1)W_{N/2}^{rk}$$

$$= X_1(k) + W_N^k X_2(k)$$

其中 k = 0， 1，.... N/2 − 1。$X_1(k)$和$X_2(k)$只有 N/2 个点，以 N/2 为周期；而 X(k)却有 N 个点，以 N

为周期。要用$X_1(k)$和$X_2(k)$表达全部的 X(k)值。还必须利用W_N系数的周期特性。

因为：$W_{N/2}^{r(\frac{N}{2}+k)} = W_{N/2}^{rk}$

所以：$X_1\left(\frac{N}{2}+k\right) = \sum_{r=0}^{N/2-1} x_1(2r)W_{\frac{N}{2}}^{r\left(\frac{N}{2}+K\right)} = \sum_{r=0}^{N/2-1} x_1(2r+1)W_{N/2}^{rk}$

又考虑到W_N^k的对称性：

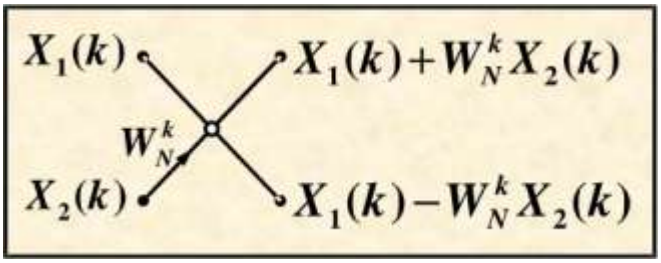
$W_N^{(\frac{N}{2}+k)} = W_N^{(\frac{N}{2})}W_N^k = -W_N^k$，有：

$X(k) = X_1(k) + W_N^k X_2(k) \quad k = 0, 1, \ldots N/2-1$

$X(\frac{N}{2}+k) = X_1\left(\frac{N}{2}+k\right) + W_N^{\left(\frac{N}{2}+K\right)}X_2(\frac{N}{2}+k)$

$= X_1(k) - W_N^k X_2(k) \quad k = 0, 1, \ldots N/2-1$

有了上面的计算结果后，我们可以得到如下的蝶形运算流图符号：



关于这个蝶形运算流图符号说明如下：

1. 1 个蝶形运算需要 1 次复乘，2 次复加。
2. 左边两路为输入。
3. 右边两路为输出。
4. 中间以一个小圆表示加减运算（右上路为相加输出，右下路为相减输出）。

➢ **分解后的运算量**

	复数乘法	复数加法
一个 N 点 DFT	N^2	N(N-1)
一个 N/2 点 DFT	N^2	N/2(N/2-1)
两个 N/2 点 DFT	$N^2/2$	N(N/2-1)
一个蝶形	1	2
N/2 个蝶形	N/2	N
总计	$N2 + N/2 \approx N^2/2$	$N(N/2-1)+N \approx N^2/2$

运算量减少了近一半。

例子：求 $N=2^3=8$ 点 FFT 变化。按 N=8→N/2=4，做 4 点的 DFT，先将 N=8 点的 DFT 分解成 2 个 4

点的 DFT:

可知： 时域上 x(0)，x(2)，x(4)，x(6)为偶子序列。

x(1)，x(3)，x(5)，x(7)为奇子序列。

频域上 X(0) 到 X(3) 由 X(k)给出

X(4) 到 X(7) 由 X(k+N/2)给出

N=8 点的直接 DFT 的计算量为：

复乘：N^2 次=64 次

复加：N(N-1)次=8× 7 = 56 次

$X(k)=X_1(k)+W_N^k X_2(k)$

$X(k+N/2)= X_1(k)-W_N^k X_2(k)$ k=0,…,N/2-1

得到 $X_1(k)$和 $X_2(k)$需要：

复乘：$(N/2)^{2+}(N/2)^2$ 次=32 次

复加：N/2(N/2-1)+ N/2(N/2-1)=12+12=24 次

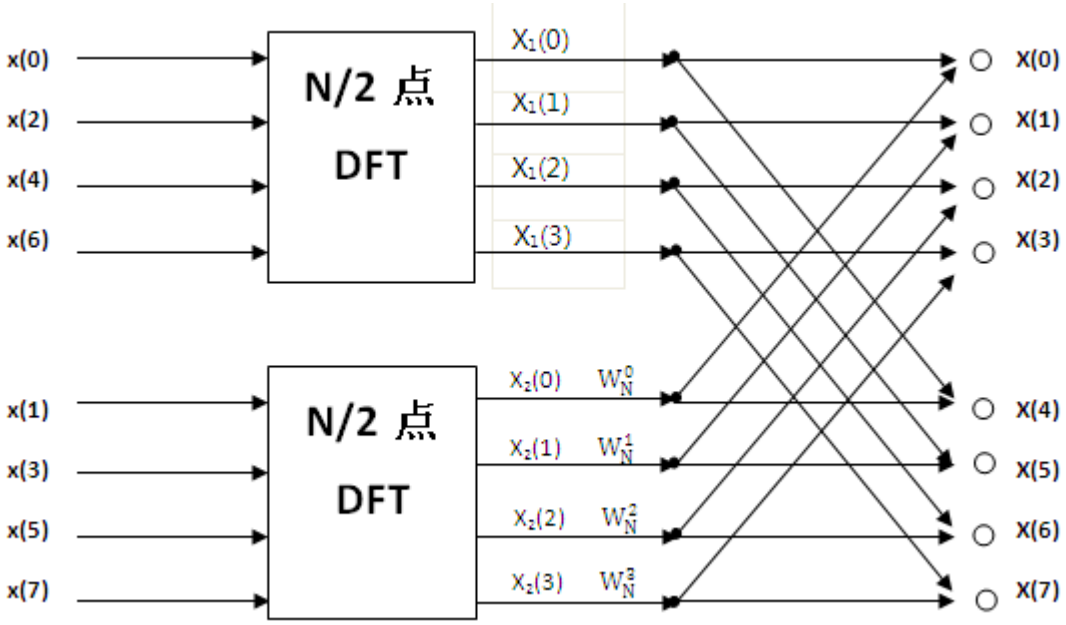
此外，还有 4 个蝶形结，每个蝶形结需要 1 次复乘，2 次复加。一共是：复乘 4 次，复加 8 次。

用分解的方法的得到 X(k)需要：

复乘：32+4=36 次

复加：24+8=32 次

N = 2^3 = 8 按时间抽取的 DFT 分解过程：



因为 4 点 DFT 还是比较麻烦，所以再继续分解。

若将 N/2(4 点)子序列按奇/偶分解成两个 N/4 点(2 点)子序列。即对将 $x_1(r)$和 $x_2(r)$分解成奇、偶两个 N/4 点(2 点)的子序列。

$x_1(r)$：$\begin{cases} x(0)、x(4) \text{ 偶序列} \\ x(2)、x(6) \text{ 奇序列} \end{cases}$ 同理：$x_2(r)$：$\begin{cases} x(1)、x(5) \text{ 偶序列} \\ x(3)、x(7) \text{ 奇序列} \end{cases}$

设：$\begin{cases} x_1(2l) = x_3(l) & \text{偶序列} \\ x_1(2l+1) = x_4(l) & \text{奇序列} \end{cases}$ (l=0…$N/4-1$) 此处，l=0,1

设：$\begin{cases} x_2(2l) = x_5(l) & \text{偶序列} \\ x_2(2l+1) = x_6(l) & \text{奇序列} \end{cases}$ (l=0…$N/4-1$) 此处，l=0,1

那么，$X_1(k)$又可表示为

$$X_1(k)=\sum_{l=0}^{\frac{N}{4}-1} x_1(2l)W_{N/2}^{2lk} + \sum_{l=0}^{\frac{N}{4}-1} x_1(2l+1)W_{N/2}^{(2l+1)k}$$

$$=\sum_{l=0}^{\frac{N}{4}-1} x_3(l)W_{N/4}^{lk} + W_{N/2}^{k}\sum_{l=0}^{\frac{N}{4}-1} x_4(l)W_{N/4}^{lk}$$

$$=X_3(k) + W_{N/2}^{k}X_4(k)$$

$$\left.\begin{aligned} X_1(k)&=X_3(k)+W_{N/2}^{k}X_4(k) \\ X_1(k+N/4)&=X_3(k)- W_{N/2}^{k} X_4(k) \end{aligned}\right\} \quad k=0,1,\ldots\ldots N/4-1$$

$X_2(k)$因为可以进行相同的分解：

$$X_2(k)=\sum_{l=0}^{\frac{N}{4}-1} x_2(2l)W_{N/2}^{2lk} + \sum_{l=0}^{\frac{N}{4}-1} x_2(2l+1)W_{N/2}^{(2l+1)k}$$

$$=\sum_{l=0}^{\frac{N}{4}-1} x_5(l)W_{N/4}^{lk} + W_{N/2}^{k}\sum_{l=0}^{\frac{N}{4}-1} x_6(l)W_{N/4}^{lk}$$

$$=X_5(k) + W_{N/2}^{k}X_6(k)$$

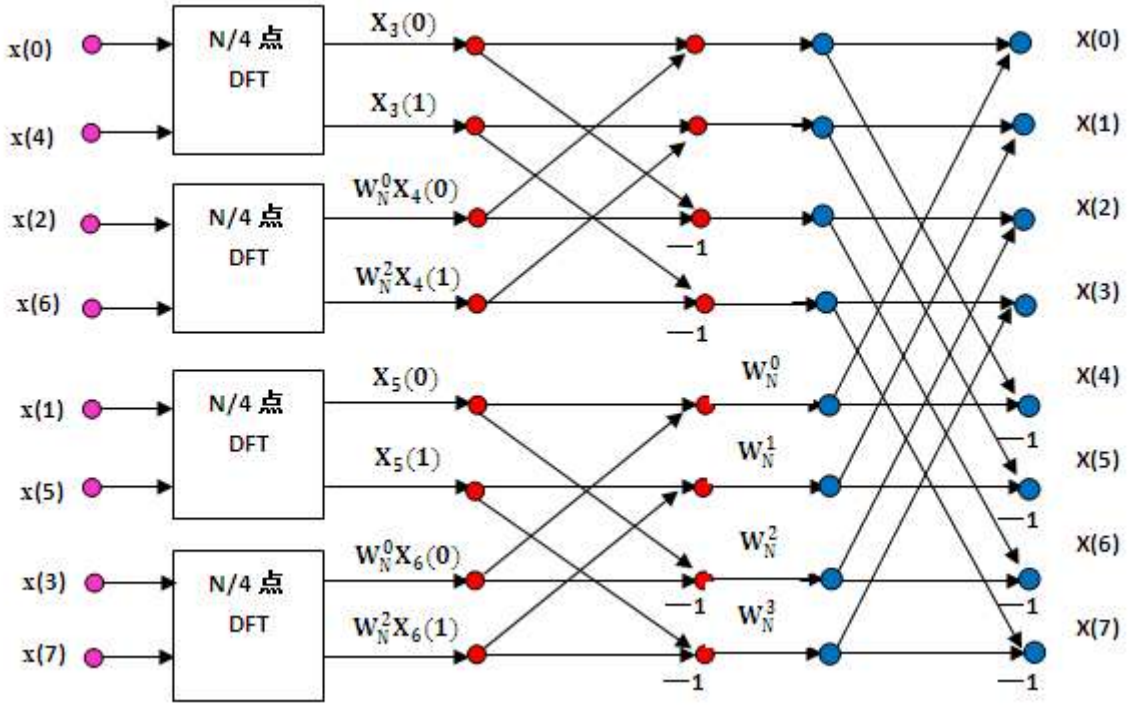
$$\left.\begin{aligned} X_2(k)&=X_5(k)+W_{N/2}^{k}X_6(k) \\ X_2(k+N/4)&=X_5(k)- W_{N/2}^{k} X_6(k) \end{aligned}\right\} \quad k=0,1,\ldots\ldots N/4-1$$

注意：通常我们会把$W_{N/2}^{k}$写成W_N^{2k}。

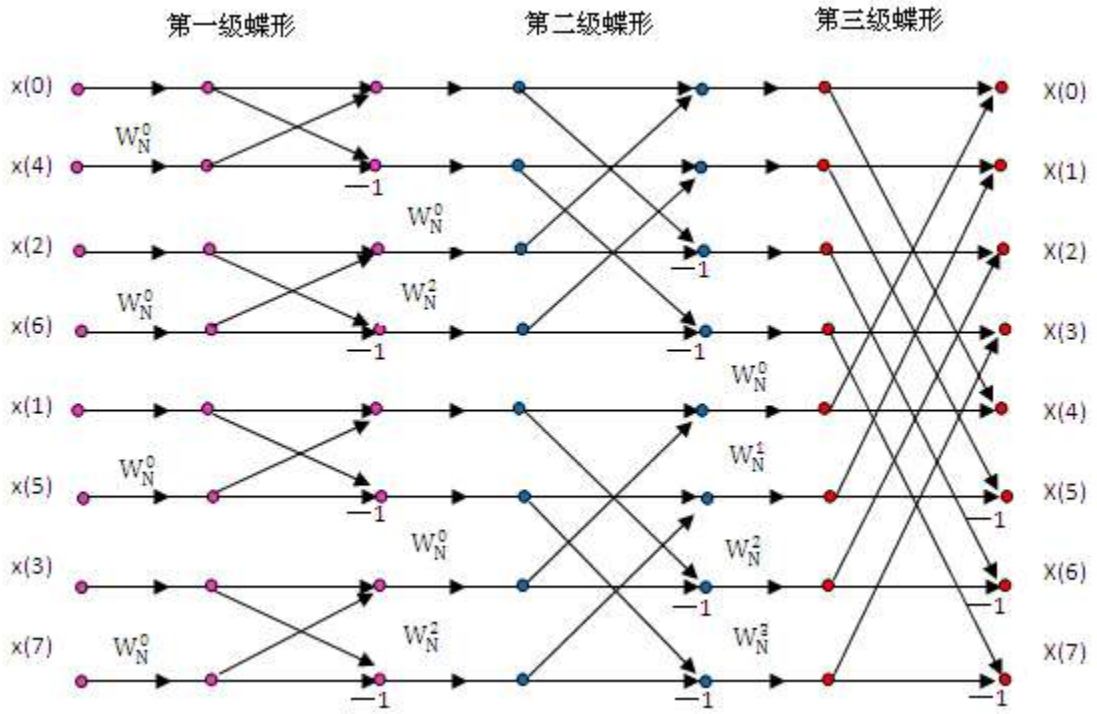
因此可以对两个 N/2 点的 DFT 再分别作进一步的分解。将一个 8 点的 DFT 可以分解成四个 2 点的 DFT,直到最后得到两两点的 DFT 为止。

由于这种方法每一步分解都是按输入序列是属于偶数还是奇数来抽取的，所以称为“按时间抽取的 FFT 算法”。

下图是由 4 个两点 DFT 组成的 8 点 DFT：



下图是按 8 点抽取的 FFT 运算流图：



这里注意观察蝶形图的系数W_N^{nk}

观察 $N=2^3=8$ 点 FFT 的蝶形系数W_N^{nk}：

第一级：有一种类型的蝶形运算系数W_8^0

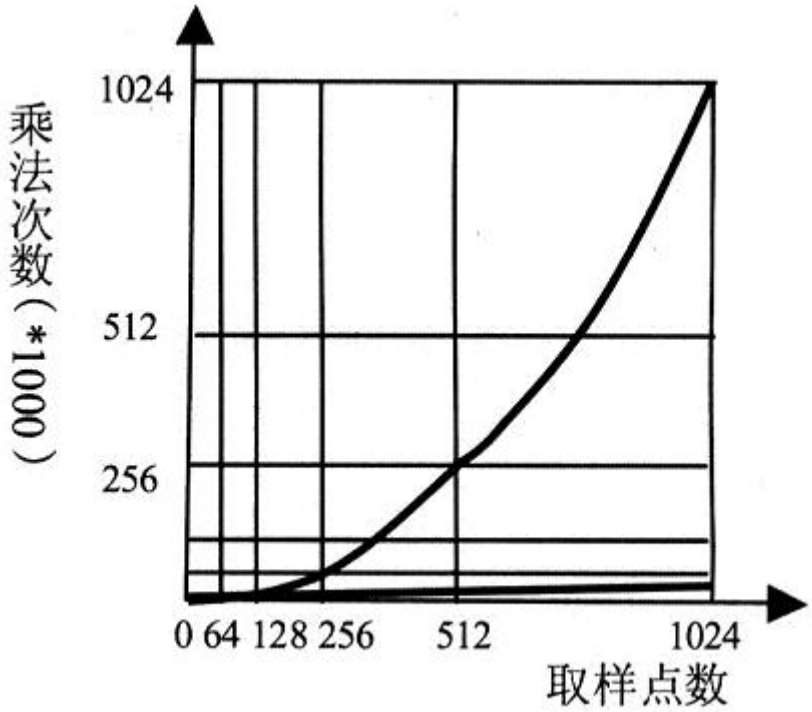
第二级：有两种类型的蝶形运算系数W_8^0、W_8^2

第三级：有四种类型的蝶形运算系数W_8^0、W_8^1、W_8^2、W_8^3

第 L 级：有 2^{L-1} 种蝶形运算系数W_N^0、W_N^1、W_N^2、....、$W_N^{N/2-1}$

24.4.3 FFT 算法和直接计算 DFT 运算量的比较

FFT 算法与直接计算 DFT 所需乘法次数的比较曲线



24.5 按频率抽选的基 2-FFT 算法

在基 2 快速算法中，频域抽取法 FFT 也是一种常用的快速算法，简称 DIF-FFT。

鉴于网上和课本中关于 FFT 原理已经讲解非常详细了，在这里就不再赘述了。有兴趣的查阅相关书籍进行学习即可。

24.6 总结

本章节主要讲解了 FFT 的基 2 算法实现原理，讲解稍显枯燥，不过还是希望初学的同学认真学习，搞懂一种快速傅里叶算法的实现即可。