快速付里叶变换及其实现

D.Butler 和 G.Harvey

（英国Plessey公司）

一、引　　言

本文旨在讨论快速付里叶变换的基本概念和树状图的数学推导。在第二节中用树状图表示法的明显特性来表明，怎样才能得到一种新的FFT处理机，使其在速度和简单性方面均优于现有的技术。预期这种技术可以有别的应用。

本文第三节讨论了实现FFT通行的方法和它们的优缺点对比。

在第四节讨论了实现诸如互相关一类附加功能的某些方法。

如果不用快速付里叶变换（FFT）算法来计算离散付里叶变换（DFT），则具有N个抽样的算术运算次数正比于N^2。由于FFT算法的使用，运算次数正比于$N\log_2 N$，所以改善的因子是

$$N^2/N\log_2 N^* = N/\log_2 N$$

抽样数很大时，其改善可很大。

于是，当N为2的乘幂时，FFT是以$N\log_2 N$次运算完成N个抽样的离散付里叶变换（DFT）计算的一种方法。

1965年，Cooley和Tukey发表了概述这一方法的论文。那时，该文大多数读者认为，这是一种计算DFT的新方法。而N个抽样的DFT的运算次数与N^2成正比，其比例常数取决于加权函数中正弦和余弦的对称性。按照此种N^2方法的算法，计算机占去大量的计算时间。

这篇文章发表后不久人们就意识到，从更一般的意义上来看，事实上它是下述方法的重新发现；该方法是1942年Danielson和Lanczos参照德国人Ronge和Konig 1924年的文章所采用的。Ronge和Konig的文章利用正弦和余弦级数来叙述变换N点的方法，此处N是2的乘幂，通过形成$\log_2 N$个子序列，该算法以$\log_2 N$倍数成倍形成这类DFT。因此，$N\log_2 N$次运算总数是必需的。

在使用初等计算装置的时期，由于可能的抽样数小，所以节省的时间很少。因此，直到现代计算装置的出现以前，该算法一直被遗忘了。

二、树状图的推导

本讲演向大家介绍FFT算法的几种形式，作为提出FFT处理机建议的一种基础。并且，

*此式原文有误。一译者

在数据的编排方式上，对这几种形式进行比较。

FFT算法，仅是一种计算离散付里叶变换（DFT）的有效方法。

$$F(k)=\sum_{n=0}^{N-1} f(n)\, W_N^{kn}{}^{*} \qquad k=0,\ 1,\ \cdots,\ N-1$$

式中

$$W_N=exp\left(-\frac{2\pi j}{N}\right)$$

这是一个N点变换，$f(n)$是数据点，n通常是一个时间标号。

上述的DFT直接算需要$4N^2$次实数乘法和加法，而FFT只需要$4Nlog_2N$次实数乘法和加法。

FFT算法之一可以这样推得：把上面的DFT和数一分为二，一个为级数的$\frac{N}{2}$个奇数项组成的和数，一个为级数的$\frac{N}{2}$个偶数项组成的和数（这里假设N是偶数）。这就给出FFT算法的“按时间抽取”形式。

显然

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

其中$G(k)$和$H(k)$是$\frac{N}{2}$点变换，因此它们以$\frac{N}{2}$为重复周期。这一点在图1中加以说明，图中g标示f的偶号点，h标示奇号点，并用大写字母来写变换。$N=4$ FFT 框的输出以一种明显

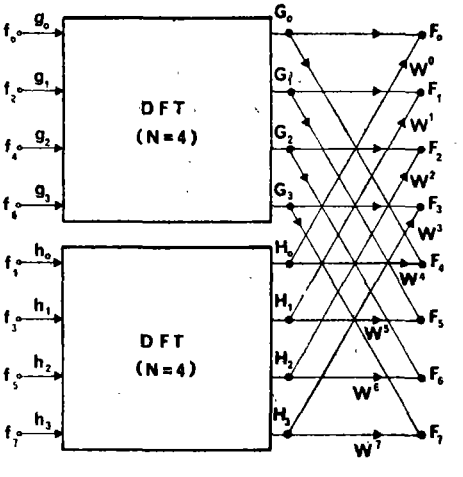


图 1

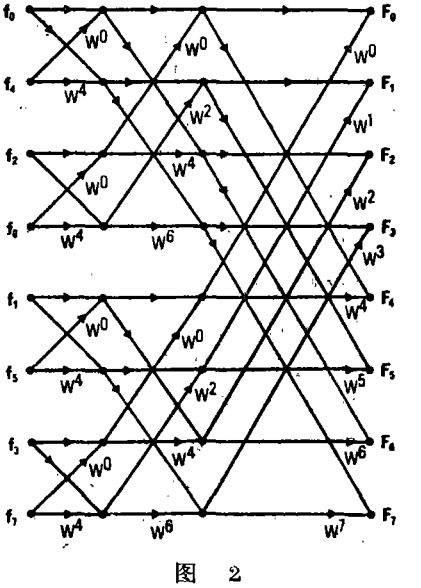


图 2

*此式原文有误。—译者

的记号组合为

$$F_0=G_0+W_8^0H_0$$

$$F_1=G_1+W_8^1H_1$$

等。

如果N是2的乘幂，这一过程经log_2N步的重复，便得到图2所示的“树状图”。在每个结点处，有两条线自左边达到，取每条线另一端的数值，且乘上该线相应的权重，组成两数之和。在框图中除了另加表示的地方外，权重是1。

这种树状图是由一定的基本单位构成的，这种单位常常称作“蝶形”，如图3所示。

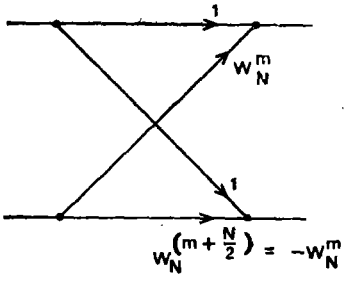


图 3

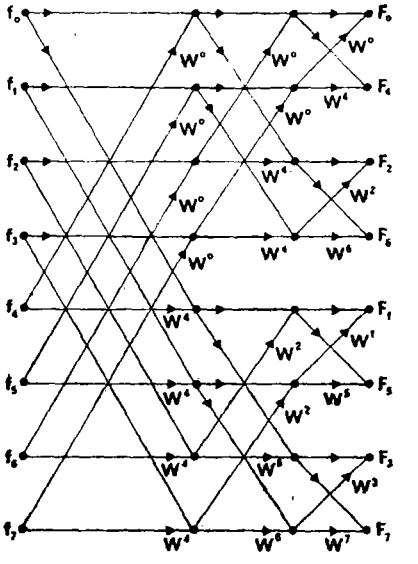


图 4

如果把图2的结点看作在某个存贮器里自然存贮的位置，则树状图的这种形式，对“位反序”（或BRO）输入有“位正序”（或BO）输出，亦即，用二进制数写出存贮的位置，对给定的输入数据，存贮位置的标号有倒位的二进制数。

简单的重排导出图4，它表明对BO输入有BRO输出。决不改变算法，只是重排存贮器。

算法的这两种形式共同的一个特点是计算的“置换”（“$in\ place$”）性质，亦即，用f_0和f_4去产生相同位置的中间数据，而且在计算中不再需要它们。中间数据同样可存贮在f_0和f_4的位置，这样丢去了f_0和f_4，所以一套存贮器就够了。通常并不是这种情况。

图5中重新排列的算法，给出BO输入和BO输出，计算不再按“位”，且对计算来说，现在一套存贮器就不够了。这种特殊的图表是不易编排的，然而不是“置换”的一些算法确实有其另外所需的特性。

在本讲演先前说明的算法中，对存贮器要求随机存取，例如图2，在不同步的计算中，要求不同位置的存取。在图6所示的树状图中，所有各步计算要求顺序地存取。后者的算法不是“置换”的，且必须用两套存贮器。在第一步，位置1和2的f_0和f_4给出下一步的头一个点。从这一步到下一步，其变换的几何形状是相同的，但加权函数是不同的。对每一步来说，容纳输入数据的存贮器的再循环比输出数据用的存贮器快一倍。这种特殊的算法是BRO到BO。

图7给出为硬设备使用而提出的算法型式。它也需要两套存贮器。输入存贮器所需的两个存取点相隔$\frac{N}{2}$个点。在图7中，点f_0、f_1、f_2、f_3存放在存贮器的一半，f_4、f_5、f_6、

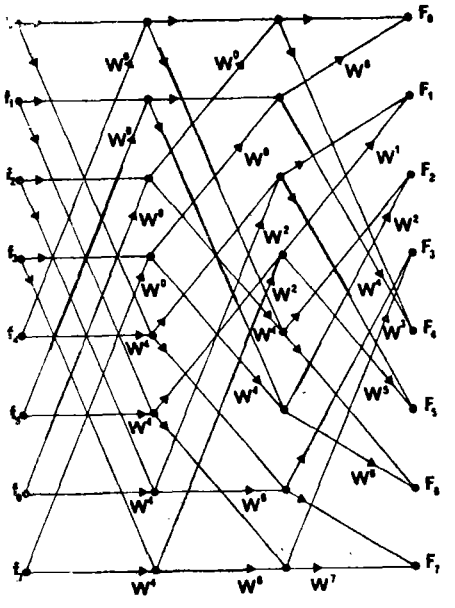


图 5

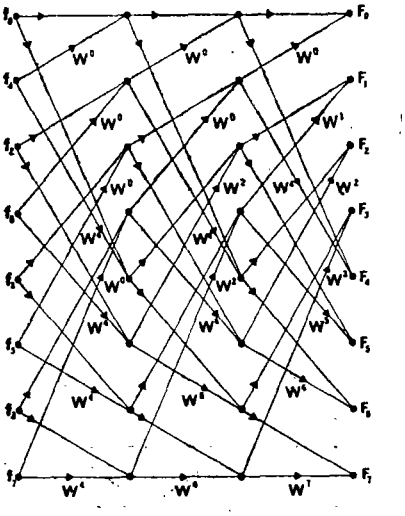


图 6

f_7存放在存贮器的另一半。用f_0和f_4给出输入的头两个点，等等。只是对输入存贮器下面一半的输出应用加权。在这种情形中，输出存贮器的再循环比输入存贮器快一倍。一步一步变换的几何图形是相同的，使得硬设备实现更加简单。该算法是BO到BRO。

采用称做为"按频率抽取"方法，可以推导出一种不同的算法。不是把初始数据分成一个对偶数编号项求和，一个对奇数编号项求和，而是变为由一个$\frac{N}{2}$点序列的DFT推导出每一个

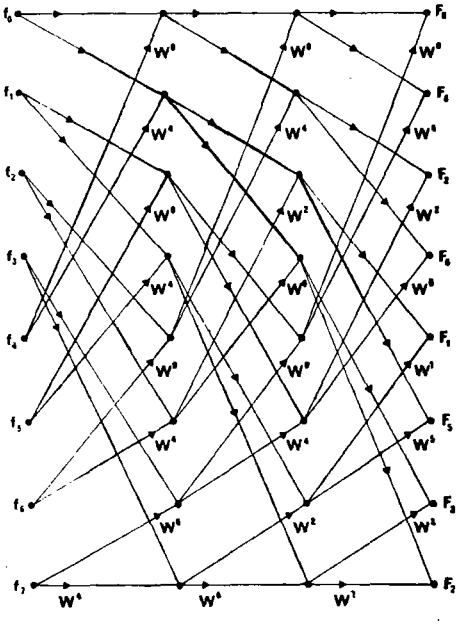


图 7

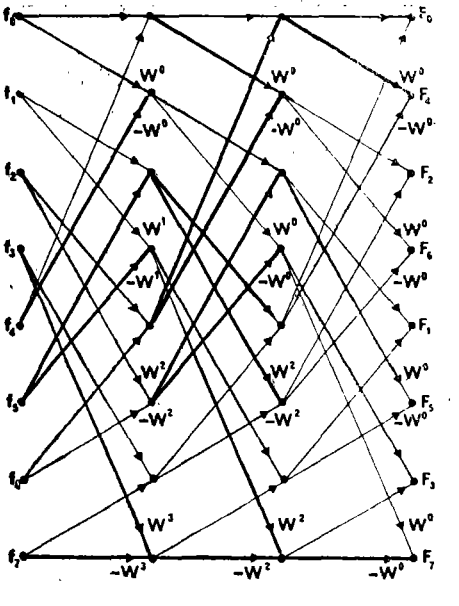


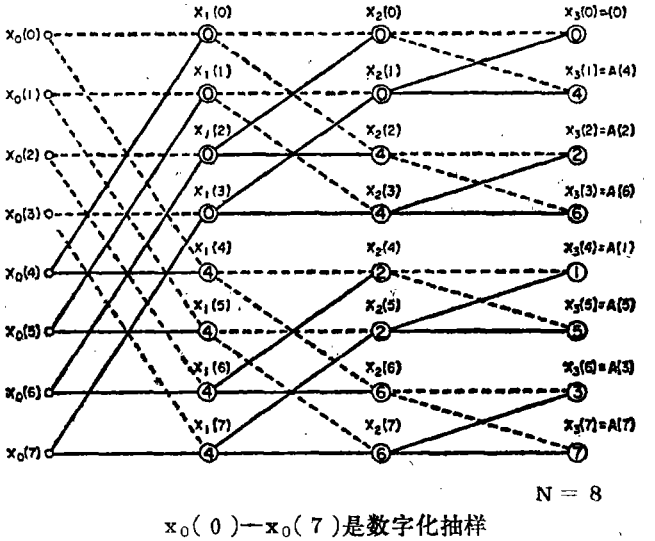
图 8

偶数编号变换函数F_0、F_2、F_4、F_6，由另外$\frac{N}{2}$点序列推导出奇数编号变换函数。这样导致算法相同，但结构不同。

在图8中给出的是等价于图7的一种算法。这也是BO到BRO，但是各自所需的加权函数的次序不同，现在是自然次序。在硬设备的一些形式中，这可是方便的。当需求卷积和互相关时，兼有按时间抽取和按频率抽取两种算法是特别方便的。这时一种算法可用于变换，而另一种算法可用于反变换，因此数据不需要重排。而如果这两个变换用同一种算法，数据就要重排了。

三、一种串行的FFT处理方法

如上所述，FFT算法可用树状图的几何图形来表示，图9中的8点变换是一个例子。为了简化硬设备讨论，这里已画成一种稍不同于图1—8的形式。



$x_0(0)$—$x_0(7)$是数字化抽样
A(0)—A(7)是输出付里叶系数

图 9

下面给出的说明，概括了该图的基本性质，以及它们同处理机设计的关系。

（i）用左边结点的垂直列代表抽样数据：如示出的8个样点$x_0(0)$到$x_0(7)$（x的下标系指垂直列）。第二垂直列（x_1）相当于在第一列上运算所得的中间系数，第三列（x_2）相当于在第一和第二列上运算所得的系数。最后的垂直列是一些最后的系数（x_3），就是最后的变换系数A（n）（n＝0—7）。

(ii)所需的算术运算是用连接结点的线和围着该结点的数一起来确定的。每个结点代表一个复数乘法和一个复数加法。进入一结点的两条线确定了由前一列来的两个系数，将它们相加，其中的一个先乘以加权函数。虚线代表前面系数的简单传送。实线代表前面的系数乘以加权函数，加权函数的值与结点圆内写出的数有关。（精确的关系式是W^z形式，其中W^z是加权函数的值，z是结点圆内所示的一个整数。它通常称为加权函数指数值。）

例如，系数$x_1(0)$是由系数$x_0(0)$和$x_0(4)$运算形成的，它由下式确定：

$$x_1(0)=x_0(0)+W^0\cdot x_0(4)$$

(iii) 第二个系数$x_1(4)$也是由系数$x_0(0)$和$x_0(4)$运算来确定，即

$$x_1(4)=x_0(0)+W^4\cdot x_0(4)$$

这样一对加权函数之间存在着一种有规则的关系式，简化了处理机的结构和加权函数存贮。每对加权函数用来由前面的两个系数产生两个系数，其指数值相差$N/2$，此处N是变换中的点数。

对这样的一对加权函数来说，加权函数的值大小相等，但符号相反。

对$N=8$而言，$W^4=-W^0$，$W^5=-W^1$，等等。

这一特性导致与变换有关的“和与差”处理方法，即由一对系数同时形成下一步中两个相应的系数，在形成一个系数时权函数用正值，在形成另一个时用负值。因此，用$x_0(0)$分别乘以W^{0*}和$-W$，所得结果加上$x_0(0)$，便形成$x_1(0)$和$x_1(4)$。在所有提出的系统中，同时提取两个系数，同时形成两个系数。

(iv) 在所用的树状图的这种情形，它相当于一种特殊的FFT算法，输入数据是按时间次序的形式（参看图7）。由于算法固有的性质，得到的输出系数不是按频率次序的；事实上，他们是按“位反序”（BRO），即如果用二进制记数法来表示最后的系数x的位值，把该二进制数左右掉换，这样得到的二进制数是系数A的位值。以8点为例，则$x_3(0)$对应于A(0)，$x_3(1)$对应于A(4)，$x_3(3)$对应于A(6)，等等。

(a) 树状图性质在处理机设计方面的作用

假设起始输入存贮器存满抽样。由树状图表明，对于第一垂直列的运算，在变换的第一次“通过”期间，所需的系数对相隔N/2个字。在第二次通过期间，对第二列的运算产生第三列，系数对相隔N/4个字。这种间隔收缩是逐步进行的，一直到最后一次通过时系数对相邻为止。在处理机存贮器的设计上这些性质的含意是，或者需要某种重复存取的存贮器，或者数据的存贮和结构必须按某种补偿方法排列。

(b) 存贮器的结构

为了利用串联的MOST**个寄存器成本低的优点，必须进一步考察图9的树状图。

图中左边一列抽样数据，即$x_0(0)$，$x_0(1)$，…$x_0(7)$是按字对计算所必需的。在后面各步计算中，这些字对是变化的，但在任何特定的步总是确定的。

就8点变换而言，表1列出每一步所需的字对。

表 1

	第一步	第二步	第三步
字对	$X_0(0)$和$X_0(4)$	$X_1(0)$和$X_1(2)$	$X_2(0)$和$X_2(1)$
	$X_0(1)$和$X_0(5)$	$X_1(1)$和$X_1(3)$	$X_2(2)$和$X_2(3)$
	$X_0(2)$和$X_0(6)$	$X_1(4)$和$X_1(6)$	$X_2(4)$和$X_2(5)$
	$X_0(3)$和$X_0(7)$	$X_1(5)$和$X_1(7)$	$X_2(6)$和$X_2(7)$

联系表和树状图来看，第一步变换所需的字对相隔4个抽样，需要全部的字对；在第二步变换，这些字对相隔2个抽样，不必要全部的字对；在第三步变换，这些字对是邻接的，又不必要全部可能的字对。如果最初的抽样$x_0(0)$，…$x_0(7)$是串联存贮的，则对第一次通过来说，需要在$x_0(0)$和$x_0(4)$位置存取。第二次通过需要在$x_0(0)$和$x_0(2)$位置存取，第二次通过的后一步在$x_0(4)$和$x_0(6)$位置存取。第三次通过需要在每个字位置存取。这在下面的表2中说明，表中*代表变换的各步的存取点。

---

* 原文误为W_0。——译者

** 金属氧化物半导体场效应晶体管。——译者

表　2

抽　样	$X_0(0)$	$X_0(1)$	$X_0(2)$	$X_0(3)$	$X_0(4)$	$X_0(5)$	$X_0(6)$	$X_0(7)$	
需要存	*				*				1
取的所在	*		*		*		*		2
步	*	*	*	*	*	*	*	*	3

这些结果推广到2048点的变换，即11步变换，要实现该系统就需总数23的存取点。

另一个方法是，存取点数不变，改变MOST寄存器内的数据，使所需的字对在存取点总可得到。这种不含重复移位的构造方法示于图10。在记忆存贮器的每一段只需两个存取点，供多路复用的每个记忆存贮器，在下次变换阶段和与它相配的记忆存贮器交换。

寄存器的钟频是f和2f。

对照表1和树状图，正确的字对可看作是在变换的每一步存取的。利用“和与差”方法，相应的计算字对是顺序存贮的，后面的字对是从输入寄存器存取的。二者同时进行。由于变动数据，存取在不重叠的时间内得到。完成这一过程的速度上的限度，现在由MOST移位寄存器最大工作频率F_t来确定。目前这一方法的计算时间是几个毫微秒，与提出的MOST存贮器存取时间一致。

实现这种串联的FFT，其完整的系统框图示于图11。

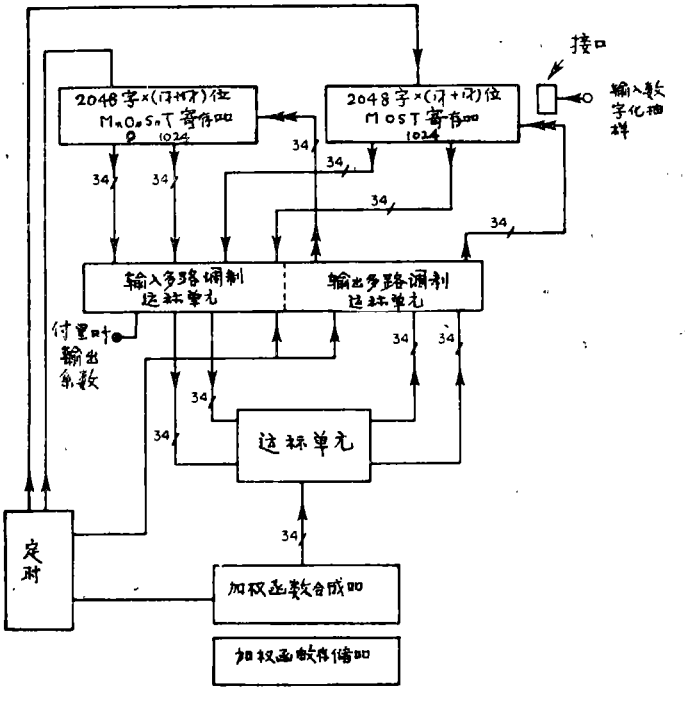
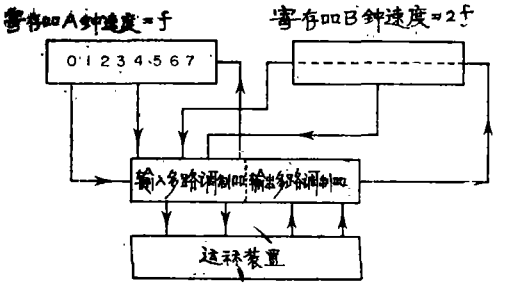


图10　FFT框图

从上文可以看到，能够实现一种串联的FFT系统。它与传统的构造方法相比主要有高速计算的优点。在2.5毫秒内能处理2048个复数点。



寄存器A 在 f　　寄存器B 在2f

存取＊	0 1 2 3 4 5 6 7		
	＊　　＊		
	1 2 3 4 5 6 7 －	－ － － － － － 0 4	in←
	＊　　＊		
第一次通过	2 3 4 5 6 7 － －	－ － － － 0 4 1 5	in←
	＊　　＊		
	3 4 5 6 7 － － －	－ － 0 4 1 5 2 6	in←
	＊　　＊		
	4 5 6 7 － － － －	0 4 1 5 2 6 3 7	in←

in—表示在寄存器B 的末端串行输入两个计算的值

	寄存器A 在 2f		寄存器B 在f	
			＊　　＊	
	6 7 － － － － 0	2 in	4 1 5 2 6 3 7 －	
第二次通过	－ － － － 0 2 4	6 in	1 5 2 6 3 7 － －	
交换寄存器	－ － 0 2 4 6 1	3 in	5 2 6 3 7 － － －	
	0 2 4 6 1 3 5	7 in	2 6 3 7 － － － －	

	寄存器A 在f	寄存器 B 在2f	
	＊　　＊		
第三次通过	2 4 6 1 3 5 7 －	3 7 － － － － 0 1	in←
交换寄存器	4 6 1 3 5 7 － －	－ － － － 0 1 2 3	in←
	6 1 3 5 7 － － －	－ － 0 1 2 3 4 5	in←
	1 3 5 7 － － － －	0 1 2 3 4 5 6 7	in←

f和2f系统

图 11

四、FFT处理方法

1．软设备

按照前面的解释，首先是用通用计算机和软件来执行FFT。这意味着，对系数和加权函数的存贮使用随机存取磁芯存贮器。由于磁芯循环时间是1微秒（这对于大多数通用计算

机是典型的），对于实现FFT算法存在一个速度的限制。对大多数通用计算机来说，实现FFT的软件是容易做到或书写的，但缺点是执行1024个字变换要用若干秒时间。

2．平行处理机

从树状图看到，在运算单元中引入平行化可提高处理速度。采用不同程度平行化的处理机已经构成，上限是阵列处理机，其中全部算术运算是平行进行的。虽然，采用此种处理器可达到极快的处理时间，但是，由于该设备的费用和大小的原因使它局限于最特殊的应用。

3．FFT外围设备

处理机的范围已扩展到相当于通用计算机的外围设备。

最简单的形式是一个与通用计算机联用的高速运算单元。

更完善的FFT外围设备有其自己的存贮器和运算单元。一台通用计算机可用于数据的编排、与外围设备间的字组转移和系统编排。——这些专用外围设备比单纯FFT能完成更多的功能。这些功能在第五节中说明。可实现高的处理速度。

五、其它功能

基本的FFT处理机可用来完成如下所说的其它功能。

1．褶积

若两个频率函数G(f)、H(f)之积是C(f)，即

$$C(f)=G(f)\cdot H(f) \qquad (A)$$

这个乘法可用于由频域到时域的反变换形式。得到的时间函数c(t)叫做时间序列g(t)、h(t)（分别为G(f)、H(f)的时间函数）的褶积。其数学描述是：

$$c(\tau)=\frac{1}{N}\sum_{t=0}^{N-1} g(t)h(\tau-t) \qquad (B)$$

为使这个和数有意义，对时间函数h(t)有一个规定，就是

$$h(t)=0 \qquad \text{对} t<0 \qquad (C)$$

如果这种处理是利用FFT处理机来进行的，则表面看似乎只是一项建立方程（A），然后由反变换形成方程（B）的简单任务。事实上并不是这种情形，因为对h(t)的附加条件没有满足。这是因为FFT处理机假定了时间函数h(t)，g(t)在抽样区间外边是周期地延续的，即：

$$h(t+rN)=h(t) \qquad 0\leqslant t<N-1$$
$$r=0,1,\cdots,S,\cdots$$

因此，为了在系统中克服这一缺点，用下法对信号添加0：

令

$$g_1(t)=g(t) \qquad 0\leqslant t<N/2 \qquad (D)$$
$$0 \qquad N/2\leqslant t<N \qquad (E)$$

$$h_1(t)=h(t) \qquad 0\leqslant t<N/2 \qquad (F)$$

$$0 \qquad N/2\leqslant t<N \qquad (G)$$

（1）建立方程(D)、(E)、(F)、(G)。

（2）形成$g_1(t)$、$h_1(t)$的FFT，即$G_1(f)$、$H_1(f)$

（3）形成乘积$C(f)=G_1(f)\cdot H_1(f)$ (H)

（4）形成反变换c(t)，从而确定方程(B)。完成这些处理总的时间是三次FFT 时间 加上形成方程(H)的N个乘法的时间。

2．互相关

用类似于上述褶积的和数来定义两个信号g(t)和h(t)之间的互相关和数，即

$$c(\tau)=\frac{1}{N}\sum_{\tau=0}^{N-1} g(t)\cdot h^*(t-\tau) \qquad (\alpha)$$

式中＊是复数共轭，且对t＜0来说，h(t)＝0。

用FFT处理机来求得这个乘积，所采用的方法类似于上面在褶积中叙述的方法。

当完成由(α)式描写的时间序列变换到频率序列的时候，方程的形式是

$$C(f)=G(f)\cdot H^*(f) \qquad (\beta)$$

所以，与前面所述方法不同之处只是用方程(β)来代替方程(H)。

3．自相关

自相关乘积用以下和数定义：

$$c(\tau)=\frac{1}{N}\sum_{t=0}^{N-1} g(t)\cdot g^*(t-\tau) \qquad (a)$$

方程(a)的变换是

$$C(f)=G(f)\cdot G^*(f) \qquad (b)$$

我们立刻可看出，这个处理肯定是较快的，因它省去了第二个函数FFT的计算。

该处理过程叙述如下：

（1）形成$g_1(t)=g(t)$ $0\leq t<N/2$ (c)

0 $N/2\leq t<N$ (d)

（2）形成$g_1(t)$的FFT，即$G_1(f)$。

（3）形成乘积$G_1(f)\cdot G_1^*(f)$，即$G_1(f)$的模。

（4）形成这个模的FFT反变换，给出自相关系数。

如同在互相关中一样，当抽样持续时间超过$N/2$个抽样时，该处理可扩展。

4．互譜分析

两个时间函数$x(t)$、$y(t)$的互谱是用互相关系数$C_{xy}(\tau)$来定义的。

$$R_{xy}(f)=\frac{1}{N}\sum_{\tau=0}^{N-1} C_{xy}(\tau)e^{-2\pi i f\tau/N} \qquad (\alpha)$$

为了采用FFT处理机来做到这点，利用下面事实：

$$R_{xy}(f)=X_1(f)\cdot Y_1^*(f) \qquad (\beta)$$

式中$X_1(f)$和$Y_1(f)$是用下面的输入函数$x(t)$、$y(t)$来定义的：

$$x_1(t)=x(t) \qquad 0\leqslant t<N/2 \qquad (\gamma)$$

$$0 \qquad N/2\leqslant t<N \qquad (\delta)$$

$$y_1(t)=y(t) \qquad 0\leqslant t<N/2 \qquad (\iota)$$

$$0 \qquad N/2\leqslant t<N \qquad (\mu)$$

$X_1(f)$,$Y_1(f)$分别是$x_1(t)$和$y_1(t)$的变换。

所以该方法等同于互相关分析，一直到它包括的第（3）步，但没有FFT反变换。

六、谱 频 谱

用FFT处理机和外围计算机计算谱频谱是容易实现的。

方法是：

（1）对时间序列$x(t)$作FFT处理。

（2）再排处理机输出。

（3）结果写成复数形式$a+ib$。然后，根据语言分析的谱频谱的定义，计算机取这种复数的对数。

$$a+ib=Re^{i\theta} \qquad \text{其中}R=\sqrt{a^2+b^2}$$

$$\text{和}\theta=tan^{-1}\left(\frac{b}{a}\right)$$

所以$\log_e(a+ib)=\log_e R+i\theta$。

这仅仅在开头$N/2$个结果上进行。

（4）序列$\log_e R$按频率次序置放在实数寄存器里，θ放在虚数寄存器里。

（5）对这个按频率次序排的序列进行一次FFT，以形成谱频率（quefrency）序列。

（6）再次排成一个谱频率次序的序列。

自FFT的第一次通过以后只使用$N/2$个抽样的理由等价于在一个时间次序序列上的Nyquist抽样准则。

若最高可分辨频率是f，则谱频率序列以f的倒数即$\frac{1}{f}$作为它的“频率”。谱频谱则形成序列

$$\left(0,\ \frac{1}{f},\ \frac{2}{f},\ \cdots\cdots\frac{f/2}{f}\right)\text{秒}$$

作为谱频率序列。如果$f<1$，则最大可分辨的谱频率是$\frac{1}{2}$秒，即在频谱上是2赫。

討　　論

问：根据我在FFT硬设备方面的有限经验，似乎这些设备比与之配合工作的计算机更昂贵，因此它们局限于非常专用的模拟。鉴于这一点，你们考虑过更有限的程序，例如基4的算法吗？

G.Harvey：我们同意有关FFT分析器价格的看法。市場上早期的FFT分析器含有许多设备，因为那时制造商不确知需要的是什么。这样促使了价格的提高。我们沒有考虑过基4的算法。

在市場上腾贵了一个时期，价格就会回跌。硬设备的制造很容易做到在一个基片上做大量的运算。

I.G. Libbell：請你谈一谈，鉴于什么基本理由，不能有一种位正序输入、位正序输出且是“置換”的FFT？象快速Walsh变換就是这样一种算法。

D.Butler：基本的道理我不知道，我也不知道有这些性质的算法。

（单荣华译　丁达夫　孙允恭校）