快速傅里叶变换和卷积算法

[法] H. J. 努斯鲍耿 著

胡广钦 译
徐俊毅

上海科学技术文献出版社

Fast Fourier Transform and Convolution Algorithms
Henri J. Nussbaumer
Springer-Verlag Berlin Heidelberg New York 1981

快速傅里叶变换和卷积算法
[法] H. J. 努斯鲍默 著
胡光锐 译
徐俊荣 校
*
上海科学技术文献出版社出版
（上海市武康路2号）
新华书店上海发行所发行
德清洛舍印刷厂印刷
*
开本 787×1092 1/32 印张 9.5 字数 227,000
1984年12月第1版 1984年12月第1次印刷
印数：1—11,800

书号：13192·63 定价：1.48元

《科技新书目》80-222

目　录

第一章 绪 言

1.1 引言

近几年来，数字卷积和离散傅里叶变换 (DFT) 的实际应用更加广泛，其意义愈来愈重要. 其直接原因是由于数字滤波和 DFT 在数字信号处理中的重要作用以及数字硬件的价格迅速下降后使数字信号处理的应用更加普及. 促使快速卷积和 DFT 算法发展的主要原因是由于直接计算长度为 N 的卷积和 DFT 需要正比于 N^2 的运算次数，因此当数据很多时运算次数将迅速猛增. 这意味着在用计算机实现这些算法时将会提出苛刻的要求.

在快速算法发展的历史中最重要的一件事是由库利 (Cooley) 和图基 (Tukey) 在 1965 年提出的快速傅里叶变换 (FFT)，用 FFT 计算 DFT 的运算次数正比于 $N \log N$，因此对于很长的变换能显著地减少计算的复杂性. 因为卷积可以利用 DFT 来计算，故 FFT 算法也可以用正比于 $N \log N$ 的运算次数来计算卷积，因此自从 FFT 出现以来，它在数字信号处理中就起着关键性的作用. 最近，又提出了许多新的快速卷积和 DFT 算法，可以进一步减少相应运算的计算复杂性. 1976 年由 Winograd 提出的快速 DFT 算法可能是这些方法中最重要的一种，因为它在理论上所达到的计算复杂性的减少已超过了 FFT，而且这种方法能将 DFT 作为卷积来计算，故可看作是 FFT 的一种相反形式. 实际上，正如我们在本书中将要看到的，卷积和 DFT 之间的关系包括许多方面，因此这种关系的含

意远超出了单纯的算法步骤的范围.

新算法发展的另一重要因素是认识到卷积和 DFT 可以看作在整数及多项式的有限环和域中定义的运算. 这个新观点可以用来导出一些更低的计算复杂性界限和设计一些以多项式变换和数论变换为基础的新的和改进的算法.

许多卷积和 DFT 算法除了它们的实用意义外还具有理论意义,这是因为能使我们更好地理解数学结构,这些结构除了在卷积和 DFT 范畴外还可以有许多应用. 例如, 多项式变换很可能成为将多维问题变换成一维问题的一种很通用的数学工具.

本书很多内容讨论的是执行同一功能的各种不同算法的比较. 在许多情形下, 我们按执行某种算法所需要的运算次数来衡量计算的复杂性. 但是某种算法的总的复杂性和它的代数复杂性之间只是一种粗略的关系, 因此计算方法的实际价值将取决于许多因素. 除运算次数外, 算法的效率还取决于其他许多参数,例如数据移动次数、辅助运算的耗费、整体结构的复杂性、执行算法的那台计算机的工作能力以及程序员的熟练程度等. 因此, 按照计算机执行算法的时间作为实际效率来评价不同算法是一件困难的工作,为此,按运算次数进行比较时必须根据具体的实现方式加以权衡.

1.2 记号

在表示各种不同的 DFT 和卷积算法时, 总是不可避免地要扩大使用不同的符号和下标. 为了简化起见我们采用了一些常规方法. 离散数据序列通常用小写字母表示, 例如 x_n. 我们不用 $\{x_n\}$ 来表示数据序列, 这是为了简化记号, 而且根据上下文的信息就能够避免序列和序列的第 n 个元素之间的混淆. 因

此，在我们的表示式中，离散时间信号 x_n 是连续信号 $x(t)$ 在 $t=nT$ 时的取样值组成的序列，并用数字来表示．多项式用大写字母来表示，例如

$$X(z)=\sum_{n=0}^{N-1}x_nz^n. \tag{1.1}$$

我们用记号 $\overline{X}_k$ 来表示变换，对于 DFT，$\overline{X}_k$ 的表示形式为

$$\overline{X}_k=\sum_{n=0}^{N-1}x_nW^{nk}. \tag{1.2}$$

我们有时也采用 Rader 用过的符号 $\langle x\rangle_p$ 来表示 x 以 p 为模时的剩余．

1.3 本书的组成

第二章讨论数论和多项式代数方面的基本内容．这一章用直观方式叙述各种不同问题，如整数和多项式的整除性，同余，在有限域和环中定义的根．这部分数学基础对于理解本书的其它部分是必需的，而对于已经熟悉数论和现代代数的读者则可以跳过．

第三章讨论快速卷积算法．证明了这些算法的大多数可以表示成多项式代数，并可以认为是不同的嵌套形式．

第四章讨论一般的快速傅里叶变换算法的发展简况，并介绍了这种算法的一些新形式，例如 Rader-Brenner 算法．

第五章的内容是离散傅里叶变换的卷积计算方法，主要讨论 Winograd 傅里叶变换算法，这是计算离散傅里叶变换的一种非常有效的算法．

在第六章和第七章中，我们介绍多项式变换，这些变换是定义在多项式有限环和域中的 DFT．我们将证明这些变换不需要乘法运算，因此对于计算多维卷积和 DFT 是一种很有效的

数学工具.

在第八章中.我们将讨论用模算术来实现的算法,我们还要介绍数论变换,这些变换是在数的有限环和域中定义的DFT.我们将证明这些变换用专用硬件来实现时可能会有重要的应用.

第二章　数论和多项式代数基础

许多新的数字信号处理算法是从初等数论或多项式代数导出的，为了理解这些算法并在实践中能够应用，应该具备一些初等数论和多项式代数方面的知识.

为了理解上述新算法，本章采用一种简单而直观的方法来介绍一些必需的基础知识，我们的意图是想使一般的工程师能熟悉本书中最常用的数学原理. 我们不打算作完整的严格数学分析，而只提供一些尽可能简明的数学工具，并希望这些基础能使某些读者进一步学习参考文献[2.1]～[2.4]所指出的一些更好的参考书，这些书都已经出版.

这一章所包括的内容可以分成两个主要部分：初等数论和多项式代数. 在初等数论中，对数字信号处理应用而言最重要的内容是中国余数定理和原根. 利用中国余数定理可产生一种独特的数的表示法，并已用于数论变换(NTT)和一维问题映射为多维问题中的标号处理. 原根在NTT的定义中起着关键作用，也可用于将离散傅里叶变换(DFT)转换成卷积，后者在发展 Winograd 傅里叶变换算法中是一个重要步骤.

在多项式代数这一节中，我们简要地介绍本书中常用的环和域的概念. 我们将说明多项式代数和卷积、相关等一些大家熟悉的信号处理运算之间的联系. 我们还要介绍多项式的中国余数定理，并且提出一些在卷积和相关计算中用到的复杂性理论的结果.

2.1 初等数论

在这一节中，我们将扼要地论述整数的性质. 我们从整数除法的简单概念开始讨论.

2.1.1 整数的整除性

令 a 和 b 为两个整数，且 b 为正数. 用 b 除 a 的除法可表示为

$$a=bq+r,\quad 0\leqslant r<b, \tag{2.1}$$

式中 q 称为商，r 称为余数. 当 $r=0$ 时，b 和 q 为 a 的因子或除数，且称 b 整除 a，并用 $b|a$ 来表示. 除 1 和 a 外，a 没有其他因子时，则 a 是一个素数. 在其他所有情形时，a 为复合数.

当 a 为复合数时，它总是能分解为素数的幂 $p_i^{c_i}$ 的乘积，其中 c_i 为一正整数，且

$$a=\prod_i p_i^{c_i}. \tag{2.2}$$

算术的基本定理告诉我们这个分解是唯一的.

能整除两个整数 a 及 b 的最大正整数 d 称为最大公因子 (GCD)，并表示为

$$d=(a,\ b); \tag{2.3}$$

当 $d=(a,\ b)=1$ 时，a 和 b 除 1 外没有其他公因子，我们就称 a 和 b 互素或相互为素的.

利用称为欧几里德算法的辗转相除法能够很容易求出最大公因子. 在讨论这种算法时，我们将假定 a 和 b 为正整数. 这样做没有失去普遍性，因为 $(a,\ b)=(-a,\ b)=(a,\ -b)=(-a,\ -b)$. 当 a 除以 b 时得到

$$a=bq_1+r_1,\quad r_1<b \tag{2.4}$$

根据定义，$d=(a,\ b)\leqslant a$ 或 b. 因此，若 $r_1=0$，则 $b|a$，且

$(a, b)=b$. 若 $r_1\neq 0$, 连续应用这种运算，可得到如下一组方程：

$$\begin{aligned} b &= r_1q_2+r_2,\ r_2<r_1\\ r_1 &= r_2q_3+r_3,\ r_3<r_2\\ &\cdots\\ r_{k-2} &= r_{k-1}q_k+r_k,\ r_k<r_{k-1}\\ r_{k-1} &= r_kq_{k+1} \end{aligned} \tag{2.5}$$

因为 $r_1>r_2>r_3\cdots$，故最后一个余数为零. 因而，按照最后一个方程，得 $r_k|r_{k-1}$. 因为 $r_k|r_{k-1}$，故前面的方程就意味着 $r_k|r_{k-2}$. 最后我们可推得 $r_k|b$ 及 $r_k|a$. 因此，r_k 为 a 和 b 的因子. 现在假定，c 为 a 和 b 的任一因子，按照(2.4)式，c 也能整除 r_1. 于是，(2.5)式意味着 c 能整除 r_2, r_3, $\cdots$, r_k. 因而，a 和 b 的任一因子 c 能整除 r_k，因此 $c\leqslant r_k$. 故 r_k 为 a 和 b 的最大公因子.

欧几里德算法的一个重要结果是：两个整数 a 和 b 的最大公因子为 a 和 b 的线性组合. 将(2.4)及(2.5)式改写，便可看到这一点

$$\begin{aligned} r_1 &= a-bq_1\\ r_2 &= b-r_1q_2\\ &\cdots\\ r_k &= r_{k-2}-r_{k-1}q_k. \end{aligned} \tag{2.6}$$

第一个方程说明 r_1 为 a 和 b 的线性组合. 第二个方程说明 r_2 为 b 和 r_1 的线性组合，因此也是 a 和 b 的线性组合. 而最后一个方程意味着 r_k 是 a 和 b 的线性组合. 因为 $r_k=(a, b)$，故我们得到

$$(a, b)=ma+nb, \tag{2.7}$$

式中 m 和 n 为整数. 当 a 和 b 互素时，(2.7)式就化为 Bezout

关系式

$$1=ma+nb. \tag{2.8}$$

现在，我们考虑一个具有整系数 a, b 和 c 的线性方程

$$ax+by=c, \tag{2.9}$$

式中 x 和 y 为一对整数，并且是一个所谓丢番图(Diophantine)方程的解. 当且仅当 $(a, b)|c$ 时，这个方程才有解. 现证明如下.

从(2.9)式显然可知，当 $a=0$ 或 $b=0$ 时，我们必须有 $b|c$ 或 $a|c$.

当 $a\neq 0$ 和 $b\neq 0$ 时，很明显，若对于整数 x 和 y, (2.9)式成立，则从 $d=(a, b)$ 可得 $d|c$. 反之，若 $d|c$，则 $c=c_1d$，因此(2.7)式意味着存在两个整数 m 及 n，使 $d=ma+nb$. 因此 $c=c_1d=c_1ma+c_1nb$，且丢番图方程的解就由 $x=c_1m$ 和 $y=c_1n$ 给出. 因而，当 $(a, b)|c$ 时，丢番图方程的解可由欧几里德算法给出. 但是，丢番图方程的解不是唯一的. 如考虑有一个特殊解 $c\equiv ax_0+by_0$，我们就可以看到这一点. 假定 x, y 为另一个解，我们有

$$a(x-x_0)=b(y_0-y), \tag{2.10}$$

对这个方程除以 d，我们得到

$$(a/d)(x-x_0)=(b/d)(y_0-y). \tag{2.11}$$

因为 $[(a/d), (b/d)]=1$，这意味着 $(b/d)|(x-x_0)$ 及 $x=x_0+(b/d)s$，其中 s 为一整数. 代入(2.11)式，我们得到

$$\begin{aligned} y&=y_0-(a/d)s \\ x&=x_0+(b/d)s. \end{aligned} \tag{2.12}$$

这样就对(2.9)式确定了一组与整数 s 有关的线性相关解.

作为一个具体的数值例子，考虑方程

$$15x+9y=21.$$

我们首先利用欧几里德算法确定最大公因子 d, 这时 $a=15$ 和 $b=9$,

$$15=9\cdot 1+6$$
$$9=6\cdot 1+3$$
$$6=3\cdot 2$$

因此 $d=3$. 因为 $3|21$, 故丢番图方程有一个解. 我们现在改写前面这组方程来确定 3 是 15 和 9 的线性组合

$$6=15-9\cdot 1$$
$$3=9-6\cdot 1=-15+2\cdot 9$$

因而, $m=-1$ 和 $n=2$. 用 $d=3$ 除 $c=21$ 得到 $c_1=7$. 由此得到一个特殊的解 $x_0=-7$, $y_0=14$. 如果我们用 $d=3$ 除 $a=15$ 及 $b=9$, 我们得到 $(a/d)=5$ 和 $(b/d)=3$. 因此, 丢番图方程的一般解为

$$y=14-5s$$
$$x=-7+3s,$$

式中 s 为任一整数.

2.1.2 同余和剩余

在(2.1)式中, 整数 a 除以整数 b 的除法产生余数 r. 当所有整数 a 除以 b 时得到相同余数时, 则相对于等价关系 $a=bq+r$ 而言可看作属于相同的等价类.

属于相同类的两个整数 a_1 和 a_2 称为对模 b 同余, 且这种等价关系表示为

$$a_1\equiv a_2 \bmod{}^{(1)} b. \tag{2.13}$$

因此, 如果有

$$b|(a_1-a_2), \tag{2.14}$$

(1) 原书中的公式均为 modulo, 现按照通常的符号, 译文全部改为 mod. ——译者注