计算数学丛书

线性代数与多项式的快速算法

游兆永

计算数学丛书

线性代数与多项式的快速算法

游 兆 永

上海科学技术出版社

内 容 提 要

六十年代以来，在计算数学中出现了一批可以称之为“快速算法”的计算方法．其中，有的算法在所涉及的计算问题中已经发挥了很大的作用，快速富里叶变换(FFT)就是一个典型的例子．本书综合介绍了其中的两个重要部分，即线性代数中的快速算法以及有关多项式的快速算法．全书共分四章：第一章介绍一般矩阵向量的快速算法；第二章介绍特殊矩阵向量的快速算法；第三章介绍有关多项式的快速算法；第四章总结前三章内容之间的联系．

本书可供高等学校计算数学专业的学生及研究生参考，也可供高等学校数学系和计算机科学系的师生以及计算数学工作者参考．

计 算 数 学 丛 书

线性代数与多项式的快速算法

游 兆 永

上海科学技术出版社出版

(上海瑞金二路450号)

新华书店上海发行所发行　江苏溧水印刷厂印刷

开本 787×1092 1/32　印张 4　字数 85,000

1980年2月第1版　1983年7月第2次印刷

印数 28,001—39,400

书号：13119·822　　定价：0.40元

出 版 说 明

《计算数学丛书》是为了适应计算数学和计算机科学的发展，配合高等院校计算数学教学的需要而组织的一套参考读物。读者对象主要是高等院校数学系和计算机科学系的学生、研究生，亦可供高等院校数学系和计算机科学系的教师以及工矿企业、科研单位从事计算工作的技术人员参考。

本丛书向读者介绍近代计算方法的一些主要进展及其适用范围和实用效果。每种书集中介绍一个专题，针对本专题的近代发展作综合性的介绍，内容简明扼要，重点突出，有分析，有评价，力图使读者对该专题的动向和发展趋势得到一个完整的了解。

本丛书已拟定的选题计有：《线性代数与多项式的快速算法》、《数论变换》、《数值有理逼近》、《矩阵特征值问题》、《索伯列夫空间引论》、《计算组合数学》、《样条与插值》、《有限条形法》、《广义逆矩阵及其计算方法》、《非线性方程迭代解法》、《奇异摄动中的边界层校正法》、《沃尔什函数理论与应用》、《多项式最佳逼近的实现》、《坏条件常微分方程数值解》、《误差分析》、《最小二乘问题的数值解法》、《板壳问题非协调方法》、《外推法及其应用》、《Monte Carlo 方法》、《差分格式理论》、《高维偏微分方程数值解》等二十余种，于一九八〇年初起陆续出版。

引　　言

六十年代以来，不到二十年的时间，在计算数学中出现了一批可以称之为"快速算法"的计算方法．其中有的算法在所涉及的计算问题中已经发挥了很大的作用，大部分算法虽然在目前的发展情况和计算条件下还没有显著的计算效果，但已有的结果使人确信，在不久的将来也会发展到象前述的算法一样地给予有关计算问题以很大的推动．一个典型的例子就是六十年代中期出现的快速富里叶变换(FFT)，即根据关系式

$$x(j)=\sum_{k=0}^{N-1}A(k)W^{jk},\quad W=e^{2\pi\sqrt{-1}/N},$$

$$j=0\sim N-1,$$

从数组 $A(k)$ 计算数组 $x(j)$，运算量可由传统算法的 $O(N^2)$ 次算术运算(包括加减乘除)降为 $O(N\log_2 N)$ 次．由于 $\frac{N}{\log_2 N}\to\infty(N\to\infty)$，用这个快速算法提高工效的倍数是巨大的，并且大大地扩大了用现有计算工具所能解决的有关问题的规模．因此，这个快速算法是当前实际应用得最多的一种．到七十年代，又出现了比快速富里叶变换具有更多优点的快速数论变换．另一个简单而又有趣的例子是求复数 $a+b\sqrt{-1}$ 与 $c+d\sqrt{-1}$ 的乘积．传统的算法是做四个(实数)乘法：ac, bd, ad, bc, 用以组成乘积的实部 $ac-bd$ 与虚部 $ad+bc$．这个问题能否用三个乘法来组成？又最少要用多少个乘法才能组成？如果算出 ac 与 bd(两个乘法)，并用

$$ad+bc=(a+b)(c+d)-ac-bd \quad (\text{再用一个乘法}),$$

便可少用一个乘法（但加减法次数由 2 增加到 5）. 事实上，已经有了证明，乘法的次数不能再少了.

显然，对计算数学中的每个计算问题，不仅要研究它的解算方法，而且要研究其中的快速算法. 这样看来，快速算法的研究范围就非常广. 目前，快速算法在线性代数与多项式这两个方面发展得很快，获得了较多的重要成果. 当然，这样说也是相对的. 事实上，上述两个方面的研究仅涉及它们的一部分基本问题，还有很多问题需要进一步研究. 即使是对已有的成果（所构成的快速算法）来说，也是在发展中. 人们还在研究更快速的算法，并希望能达到最优的程度.

根据目前发展的情况，我们力图比较系统地综合介绍有关线性代数与多项式的快速算法. 全书分为四章：第一章，介绍一般矩阵向量的快速算法；第二章，介绍特殊矩阵向量的快速算法. 这两章显然是介绍有关线性代数的快速算法. 线性代数计算方法是计算数学的基础算法，构造线性代数问题的快速算法对大型问题来说尤为重要. 另一方面，一般矩阵的研究固然重要，但目前已有若干种特殊矩阵（如 Toeplitz 矩阵）显得也很重要，因此用两章的篇幅分别叙述它们的快速算法. 第三章介绍有关多项式问题的快速算法. 主要包括多项式的快速乘除法（这是最基本的），快速求值与求导数值以及快速插值等. 最后，由于线性代数和多项式这两个方面的快速算法之间，以及每个方面的各个快速算法之间都是密切相关的，故在第四章中研究它们的某些共同特点以及它们之间的联系. 又在第四章的“结束语”中，结合全书的内容，对本专题的发展试作一些初步的分析与评论.

本书的内容限于线性代数与多项式的快速算法，也限于

考虑串行计算而不考虑并行计算的快速算法，后者可归在“并行算法”的专题中．本书对分析为了解出一个计算问题所需的运算量的界限，对研究所构造的快速算法是否最优以及构造最优算法等问题均不予考虑，因为它们同样可归于“计算复杂度”等别的专题中．

本书中“O 记号”的含义和使用将比通常的定义更灵活些．例如在

$$\sum_{k=1}^{n} O(k^2)=O(1^2)+O(2^2)+\cdots+O(n^2)$$

中，通项 $O(k^2)$ 表示略去相对的低阶项后就成为 ck^2，而 c 是与 k 无关的非零常数．即有

$$O(k^2)=ck^2+\alpha_k,$$

$$\sum_{k=1}^{n} O(k^2)=\sum_{k=1}^{n}(ck^2+\alpha_k)$$

$$=\frac{1}{6}cn(n+1)(2n+1)+\sum_{k=1}^{n}\alpha_k,$$

又 $\sum_{k=1}^{n}\alpha_k$ 的阶低于 n^3，当 $\alpha_k=2k$ 时，$\sum_{k=1}^{n}\alpha_k=n(n+1)$ 的阶为 n^2（即上式右边第二项的阶满足低于 n^3 的要求）．在上述情况下，我们就直接写成

$$O(1^2)+O(2^2)+\cdots+O(n^2)=O(1^2+2^2+\cdots+n^2)=O(n^3).$$

目　录

第 1 章

一般矩阵向量的快速算法

一般矩阵向量的快速乘法运算是矩阵向量的快速算法的基础. 本章主要介绍向量的内积以及矩阵与矩阵相乘的快速算法. 在这一方面, S. Winograd 与 V. Strassen 等人有了重要的结果, 这是我们介绍的重点. 此外还介绍一些以矩阵快速乘法运算为基础的矩阵快速算法, 如快速 Schmidt 正交化算法等.

§1 Winograd 的内积算法

设有两个向量

$$x=\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \quad 与 \quad y=\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix},$$

其内积为

$$(x,\ y)=x_1y_1+x_2y_2+\cdots+x_ny_n.$$

在通常的算法中, 计算一个内积要用 n 个乘法 (即计算 x_iy_i, $i=1\sim n$) 与 $n-1$ 个加法.

1967 年, S. Winograd 提出了内积的一个新算法. 首先算出

$$\xi=\sum_{j=1}^{[n/2]} x_{2j-1}x_{2j}, \quad \eta=\sum_{j=1}^{[n/2]} y_{2j-1}y_{2j}.$$

这两个乘积的和式称为"奇偶积". 例如 $n=4$ 或 5 时, $[n/2]=2$, 并有

$$\xi=x_1x_2+x_3x_4,\quad \eta=y_1y_2+y_3y_4.$$

ξ 和 η 的计算工作量都是 $[n/2]$ 个乘法与 $[n/2]-1$ 个加法. 在 ξ 与 η 的基础上, 内积可以用

$$(x,\ y)=\begin{cases}\sum_{j=1}^{[n/2]}(x_{2j-1}+y_{2j})(x_{2j}+y_{2j-1})-\xi-\eta, & n\text{ 为偶数},\\ \sum_{j=1}^{[n/2]}(x_{2j-1}+y_{2j})(x_{2j}+y_{2j-1})-\xi-\eta+x_ny_n, & n\text{ 为奇数}\end{cases}$$

算出. 例如, 当 $n=4$ 时, 有

$$(x,\ y)=(x_1+y_2)(x_2+y_1)+(x_3+y_4)(x_4+y_3)-(x_1x_2+x_3x_4)-(y_1y_2+y_3y_4);$$

又如, 当 $n=5$ 时, 上式右边添上一项 x_5y_5 即可.

记

$$\omega=\sum_{j=1}^{[n/2]}(x_{2j-1}+y_{2j})(x_{2j}+y_{2j-1}),$$

则计算 ω 要用 $[n/2]$ 个乘法与 $3[n/2]-1$ 个加法, 而当 n 为奇数时, 计算 $\omega+x_ny_n$ 要用 $[n/2]+1$ 个乘法与 $3[n/2]$ 个加法. 当 n 为偶数时,

$$\left[\frac{n}{2}\right]=\frac{n}{2}=\left[\frac{n+1}{2}\right],$$

又当 n 为奇数时,

$$\left[\frac{n}{2}\right]+1=\frac{n-1}{2}+1=\frac{n+1}{2}=\left[\frac{n+1}{2}\right].$$

于是当 n 为偶数时计算 ω, 以及当 n 为奇数时计算 $\omega+x_ny_n$ 都用 $\left[\frac{n+1}{2}\right]$ 个乘法与 $2\left[\frac{n}{2}\right]+\left[\frac{n+1}{2}\right]-1$ 个加法. 我们还有如下的公式:

$$\left[\frac{n}{2}\right]+\left[\frac{n+1}{2}\right]=n,\quad n\text{ 为任意正整数,}$$

于是上述所用的加法的次数可记为 $n+[n/2]-1$. 连同 ξ 与 η 的计算工作量，由此得内积 (x, y) 的新算法要用

$$\left[\frac{n+1}{2}\right]+2\left[\frac{n}{2}\right]=n+\left[\frac{n}{2}\right]$$

个乘法. 这反而比通常算法所要的 n 个乘法多 $[n/2]$ 个. 但如有一批向量要用作构成多个内积(例如两个 n 阶矩阵相乘，可以看作 $2n$ 个 n 维向量用作构成 n^2 个内积)，其中便有文章可做.

设有 N 个 n 维向量用作构成 T 个内积. 首先把这 N 个向量的奇偶积算出，其乘法次数为 $N[n/2]$，加法次数为 $N([n/2]-1)$. 然后构成 T 个内积，其乘法次数为 $T\left[\frac{n+1}{2}\right]$. 由于利用 ξ 与 η 计算内积所用的加法(或减法)次数为

$$n+[n/2]-1+2=n+[n/2]+1,$$

故构成 T 个内积还要用加减法 $T(n+[n/2]+1)$ 个. 因此，新算法要用的乘法次数总数为

$$N\left[\frac{n}{2}\right]+T\left[\frac{n+1}{2}\right]=Nn+(T-N)\left[\frac{n+1}{2}\right].$$

与通常算法的乘法次数

$$Tn=Nn+(T-N)n$$

相比较，如果 $T>N$，则新算法所用的乘法次数较少. 又新算法的运算总次数(包括加、减、乘、除)为

$$N\left(\left[\frac{n}{2}\right]-1\right)+T\left(n+\left[\frac{n}{2}\right]+1\right)+N\left[\frac{n}{2}\right]+T\left[\frac{n+1}{2}\right]$$

$$=N\left(2\left[\frac{n}{2}\right]-1\right)+T(2n+1),$$

而通常算法的总次数为 $T(2n-1)$. 如果 $T\gg N$ (即 T 比 N

大得多)，则新算法与通常算法的运算总次数都约等于 $2nT$，因做乘法所需的计算时间比加法长，故新算法较快.

现在把这个内积的新算法应用于矩阵与矩阵相乘. 设 A 为 r 行 s 列，B 为 s 行 t 列矩阵，求矩阵乘积 AB 相当于用 $r+t$ 个 s 维向量构成 rt 个内积. 用新算法，乘法次数为

$$(r+t)s+(rt-r-t)\left[\frac{s+1}{2}\right]$$

(通常算法要 rst 次)，又加减法次数为

$$(r+t)\left(\left[\frac{s}{2}\right]-1\right)+rt\left(s+\left[\frac{s}{2}\right]+1\right)$$

(通常算法要 $rt(s-1)$ 次). 如果 r、s、t 都是大数，则新算法约用 $\frac{1}{2}rst$ 个乘法与 $\frac{3}{2}rst$ 个加减法，而通常算法约用 rst 个乘法与 rst 个加减法. 此即两者的运算总次数大体相同，但在通常算法中乘法次数约占运算总次数的一半而新算法仅占四分之一.

特殊情形，考虑两个 n 阶矩阵相乘(即 $r=s=t=n$)，新算法用

$$2n^2+(n^2-2n)\left[\frac{n+1}{2}\right]$$

个乘法，当 n 为偶数时，化为 $\frac{1}{2}n^3+n^2$ 个.

如果 $n=2$，则 $\frac{1}{2}\times 2^3+2^2=8$，即新算法要用 8 个乘法，这和通常的矩阵相乘算法所用的乘法次数一样多. 但新算法要用 16 个加减法，而通常算法只用 4 个.

两个二阶矩阵

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \quad 与 \quad \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix}$$

相乘的新算法的具体过程是：首先算出

$$a_{11}a_{12},\ a_{21}a_{22},\ b_{11}b_{21},\ b_{12}b_{22}\quad\text{（四个乘法）}.$$

然后算出

$$a_{11}b_{11}+a_{12}b_{21}=(a_{11}+b_{21})(a_{12}+b_{11})-a_{11}a_{12}-b_{11}b_{21},$$

$$a_{11}b_{12}+a_{12}b_{22}=(a_{11}+b_{22})(a_{12}+b_{12})-a_{11}a_{12}-b_{12}b_{22},$$

$$a_{21}b_{11}+a_{22}b_{21}=(a_{21}+b_{21})(a_{22}+b_{11})-a_{21}a_{22}-b_{11}b_{21},$$

$$a_{21}b_{12}+a_{22}b_{22}=(a_{21}+b_{22})(a_{22}+b_{12})-a_{21}a_{22}-b_{12}b_{22}.$$

以上四个式子的每一式都用一个乘法与四个加减法，连同先算的，共用 8 个乘法与 16 个加减法．乘法次数没有减少，而加减法次数反而增多．

当 $n=3$ 时，由于

$$2n^2+(n^2-2n)\left[\frac{n+1}{2}\right]=2\times3^2+(3^2-2\times3)\left[\frac{3+1}{2}\right]=24,$$

这时才比通常算法所用的 $3^3=27$ 个乘法少 3 个．

有了矩阵相乘的新算法后，结合分块运算的方法，可以建立矩阵求逆和解线代数方程组的新算法．先看 n 阶矩阵 A 的求逆，如果 $n=m\cdot k$，则可把矩阵 A 看成 m 阶分块矩阵，其中每块都是 k 阶矩阵．对这个 m 阶分块矩阵 A，象通常 m 阶矩阵一样，用一种方法求逆，例如用高斯消去法．消去过程中要做 k 阶矩阵的加法、乘法与求逆，其中 k 阶矩阵相乘可用新算法进行，而 k 阶矩阵求逆仍用高斯消去法，这便构成矩阵求逆的一种新算法．再看求解线代数方程组 $Ax=b$，其中 A 是 n 阶矩阵，b 是 n 维向量，x 为未知向量（n 维）．如果 $n=mk$，也把矩阵 A 分为 m 阶分块矩阵，其中每块都是 k 阶矩阵．对这个线代数方程组用分块的高斯消去法，其中 k 阶矩阵相乘用新算法进行，便构成求解线代数方程组的一种新算法．

§2 Strassen 的矩阵乘法

上节基于内积新算法的矩阵相乘的新算法，在二阶矩阵情形时，乘法次数没有减少(仍为 8 个). 1969 年, V. Strassen 提出只用 7 个乘法来做矩阵

$$A=\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \quad 与 \quad B=\begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix}$$

的乘法运算. 首先按下列 7 个式子 (每式用一个乘法) 算出 $c_i(i=1\sim7)$, 即

$$\begin{aligned}
c_1&=(a_{11}+a_{22})(b_{11}+b_{22}),\\
c_2&=(a_{21}+a_{22})b_{11},\\
c_3&=a_{11}(b_{12}-b_{22}),\\
c_4&=a_{22}(-b_{11}+b_{21}),\\
c_5&=(a_{11}+a_{12})b_{22},\\
c_6&=(-a_{11}+a_{21})(b_{11}+b_{12}),\\
c_7&=(a_{12}-a_{22})(b_{21}+b_{22}).
\end{aligned}$$

再按下式组成矩阵乘积 AB 的元素(这时只用到加减法):

$$\begin{aligned}
a_{11}b_{11}+a_{12}b_{21}&=c_1+c_4-c_5+c_7,\\
a_{11}b_{12}+a_{12}b_{22}&=c_3+c_5,\\
a_{21}b_{11}+a_{22}b_{21}&=c_2+c_4,\\
a_{21}b_{12}+a_{22}b_{22}&=c_1+c_3-c_2+c_6.
\end{aligned}$$

以上共用 7 次乘法与 18 次加减法.

上述做两个二阶矩阵乘法的计算式不是唯一的，还可以作出类似的计算式. 下面给出构成一组计算式的思路. 以下记

$$AB=C=\begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{bmatrix}.$$

首先注意到矩阵 A、B 的元素的下标 1 与 2 都作对调变换时(即 a_{11} 换为 a_{22}, b_{12} 换为 b_{21} 等等),矩阵 C 的元素有

$$c_{11}=a_{11}b_{11}+a_{12}b_{21} \quad 与 \quad c_{22}=a_{21}b_{12}+a_{22}b_{22} \text{ 对调},$$

$$c_{12}=a_{11}b_{12}+a_{12}b_{22} \quad 与 \quad c_{21}=a_{21}b_{11}+a_{22}b_{21} \text{ 对调}.$$

任取矩阵乘积 C 的一个元素,例如

$$c_{11}=a_{11}b_{11}+a_{12}b_{21},$$

直接计算要用两个乘法(即 $a_{11}b_{11}$ 与 $a_{12}b_{21}$). 利用"架桥法"的技巧(加一项 $a_{12}b_{11}$ 同时又减去同一项)计算,

$$a_{11}b_{11}+a_{12}b_{21}=(a_{11}+a_{12})b_{11}+a_{12}(b_{21}-b_{11}),$$

这样仍要两个乘法,即 $(a_{11}+a_{12})b_{11}$ 与 $a_{12}(b_{21}-b_{11})$. 从只计算矩阵 C 的一个元素来看,乘法次数没有减少(加减法次数反而增加),但从计算矩阵 C 的四个元素来看,减少乘法次数的途径增加了,详细说明如下.

在四个元素 c_{11}、c_{12}、c_{21}、c_{22} 原来的表达式($c_{11}=a_{11}b_{11}+a_{12}b_{21}$ 等)中,8 个乘积项($a_{11}b_{11}$, $a_{12}b_{21}$, …, $a_{22}b_{22}$)没有相同的,因此按原来的表达式计算要用 8 个乘法. 如果对 c_{11} 等都用架桥法或其它方法组合出新的表达式,并使它们有相同的乘积项,就可以减少乘法次数. 把

$$c_{11}=a_{11}b_{11}+a_{12}b_{21}=(a_{11}+a_{12})b_{11}+a_{12}(b_{21}-b_{11})$$

的下标 1 与 2 对调,得

$$c_{22}=a_{21}b_{12}+a_{22}b_{22}=(a_{21}+a_{22})b_{22}+a_{21}(b_{12}-b_{22}).$$

如果算出(共用 4 个乘法)

$$r_1=(a_{11}+a_{12})b_{11}, \quad r_2=a_{12}(b_{21}-b_{11}),$$

$$r_3=(a_{21}+a_{22})b_{22}, \quad r_4=a_{21}(b_{12}-b_{22}),$$

则可算出(不再用乘法)