准确计算方法

邓健新 著

科学出版社

计算方法丛书

准确计算方法

邓健新 著

科学出版社

1996

（京）新登字 092 号

内 容 简 介

本书系统地介绍了计算机中的无误差数值方法及其应用. 前三章的内容包括：整数的准确运算、单模剩余算法、多模剩余算法、p-adic 数系统和 Hensel 码运算法. 第四章至第十一章内容包括：准确求解线性方程组、矩阵求逆、广义逆、多项式运算、多项式矩阵、矩阵特征多项式、矩阵约当型、数论变换 FNT、素数与素数识别、整数因子分解及信息安全与数字密码.

本书读者范围：高校计算机科学、系统科学专业的师生、科研人员、工程技术人员.

计算方法丛书

准 确 计 算 方 法

邓健新 著

责任编辑 林 鹏

科学出版社出版

北京东黄城根北街 16 号

邮政编码：100717

中国科学院印刷厂印刷

新华书店北京发行所发行 各地新华书店经售

*

1996 年 3 月第 一 版 开本：850×1168 1/32

1996 年 3 月第 一 次印刷 印张：10 1/2

印数：1—1 200 字数：271 000

ISBN 7-03-004758-3/O・810

定价：24.50 元

前　言

准确计算方法是用计算机进行无误差计算的数值方法. 在数论、图论、数学规划、化学反应、符号计算、信息安全……等等领域中，有许多计算问题不允许计算结果存在误差，要求得到问题的准确解. 大量的数值计算问题虽然只要近似解，但是有些问题是高度病态的，差之毫厘谬已千里，十分棘手. 如果用准确计算，问题可能迎刃而解. 还有许多数学方法，具有计算过程简单、计算复杂度小、并行度高、存贮量小等等优点，但是因为对舍入误差十分敏感，导出的算法数值不稳定，在数值计算中被淘汰. 如果用准确计算方法，这些数学方法会成为最有用的算法. 这都表明准确计算的必要性.

通常，准确计算的运算量大，要求的存贮量也大. 过去计算机的性能-价格比太低，准确计算不能得到实际应用. 随着计算机性能大幅度提高，并行计算技术的迅速发展，准确计算的实际应用已成为可能. 近年来，对准确计算方法的研究发展很快、实际应用面愈来愈广，显示出十分广阔的前景. 现在，准确计算方法已成为计算数学研究的一个活跃的方向.

本书的目的是系统地叙述准确计算的基本思想、基本方法与技巧，介绍准确计算方法的几个重要的应用，为开展准确计算方法的研究与推广应用提供一本参考资料. 前三章是方法论，叙述的是整数的准确四则运算法、单模剩余法、多模剩余法、p-adic 数与 Hensel 码运算法. 这些是用计算机进行不带误差地进行整数、有理数、有理多项式运算的基本方法. 第 4 章至第 11 章是应用部分，包括线性代数方程组、矩阵求逆、多项式、多项式矩阵、矩阵特征多项式、矩阵约当型等重要的矩阵计算，还有数论变换、整数素数识别、因子分解和数字密码与信息安全等问题.

本书是从用计算机实现算法的角度来写的，因此在叙述上力求初等、简明和易懂。内容仅涉及初等数论、线性代数、矩阵论和计算方法的基础知识. 在应用篇中涉及方面广，为了便于直接选读某些章节，我们保持了各章具有较强的独立性. 准确计算方法具有较好的并行性，有些算法有自然的并行性，计算的大部分可以并行进行，在有关算法的叙述中已给出一些说明，但是设计算法的并行实现已超出本书的范围，有兴趣的读者可参阅书末列出的文献.

在各章中列出的C语言计算机程序可用于试验一些简单的算法，帮助理解内容和算法设计.

本书的编写与有关的研究工作得到王元院士、石钟慈院士的鼓励和中国科学院原计算中心同行的帮助. 高辉同志为书中的程序编写与上机试算作了很大努力. 在此，我表示衷心的感谢. 我还衷心地感谢国家自然科学基金、中国科学院“八五”重点项目“中大规模并行算法应用研究”和科学出版基金的资助.

邓健新

目　录

第一章 整数运算

整数运算是准确计算方法的基础，要解决科学研究或工程设计中的准确计算问题，进行大整数的准确运算是不可避免的.

计算机的字长有限，通常的算法语言只提供单字长和双字长精度运算. 有些语言能提供 3 倍，4 倍字长精度运算，但是没有一种计算机语言支持大整数准确的运算. 因此建立一个具有高效率的整数运算系统是必需的.

本章叙述整数运算法，快速乘法、整数运算软件的性能设计和大整数运算程序 LIAS.

1.1 整数运算

我们用计算机进行整数准确运算时，首先遇到的问题是怎样存贮很大的整数. 一个大整数要用许多机器字存贮. 通常用一维数组存贮一个数，用二维数组存贮一批数. 存贮的方式直接影响运算的效率和存贮量. 存贮方式应该根据算法和机器的软件、硬件环境来决定，而算法的具体实现与存贮方式密切相关. 本节先以数学的观点叙述算法，在节 1.4 中，我们将介绍一个实用的存贮方式.

适合在计算机上用的大整数准确运算算法是复杂的. 为了计算快，存贮量省，对整数运算已作了很深入的研究并且提出了多种算法. 最简单的是多倍字长运算. 这种算法不但十分冗繁、程序复杂而且效率低，缺乏选择字长的灵活性和程序的通用性. 本节介绍一种以 p 进制记数法为基础的算法. 我们先约定数的表示法，然后叙述算法的原理.

在本章以及以后各章中，除了有特别的说明之外，我们讨论的数都是整数.

数的表示法

任何一个数 a，都可以表示为基数 p 的多项式：

$$a=\pm\sum_{i=0}^{m-1}a_ip^i, \tag{1}$$

或

$$a=\pm p^m\sum_{i=1}^{m}a_ip^{-i}. \tag{2}$$

当 $p=10$ 时，a 就是一个常用的 m 位的十进制数，系数 a_i 满足条件 $0\leqslant a_i\leqslant 9,(i=0,1,\cdots,m)$. 在一般情况下，$p$ 是一个正数，a_i 也是正数，并且满足条件 $0\leqslant a_i<p$. 我们称 a 是一个 m 位的 p 进制数.

为了简化记号和叙述的方便、我们常用(2)式表示数并且假定两个正数 a,b 的表达式如下：

$$a=p^m\sum_{i=1}^{m}a_ip^{-i},b=p^n\sum_{i=1}^{n}b_ip^{-i}. \tag{3}$$

a 与 b 运算的结果是 c，c 的表达式是：

$$c=p^k\sum_{i=1}^{k}c_ip^{-i}.$$

不失一般性，我们假设

$$m\geqslant n,a\geqslant b. \tag{4}$$

加法

加法运算分为下述三步：

(一) 逐位加

为了叙述方便，我们把数 b 形式上看作和 a 一样是个 m 位数. 定义 b_i' 如下：

$$b_i'=\begin{cases}0, & (i=1,2,\cdots,m-n),\\ b_{i-m+n} & (i=m-n+1,\cdots,m).\end{cases} \tag{5}$$

逐位加可以统一地记为

$$c_i'=a_i+b_i',(i=1,2,\cdots,m). \tag{6}$$

实际上，对于在 b_i' 中数值为零的数是不需要进行运算的.

（二）进位

定义

$$c_0'=e_{-1}=e_m=0, e_{i-1}=[(c_i'+e_i)/p], (i=m,m-1,\cdots,1),$$

其中符号$[x]$表示不大于 x 的最大整数. 进位法可表示为

$$c_i''=c_i'+e_i-e_{i-1}p, (i=m,m-1,\cdots,0), \tag{7}$$

（三）标准化

$$c_i=\begin{cases} c_{i-1}'', & i=1,2,\cdots,m+1, \text{当 } e_0\neq 0, \\ c_i'', & i=1,2,\cdots,m, \text{当 } e_0=0. \end{cases} \tag{8}$$

综上所述，当 $e_0=0$ 时，a,b 的和 c 可记为

$$c=p^m\sum_{i=1}^{m}c_ip^{-i}. \tag{9}$$

当 $e_0\neq 0$ 时，c 可表示为

$$c=p^{m+1}\sum_{i=1}^{m+1}c_ip^{-i}. \tag{10}$$

减法

按照(4),(5)式的假定和记号，减法的三个部分如下：

（一）逐位减

$$c_i'=a_i-b_i', \quad (i=1,2,\cdots,m). \tag{11}$$

（二）进位

因为逐位减没有采用借位的方法，在上式中 c_i' 的值域是 $[-p+1,p-1]$. 为了把数规格化为 p 进制数的标准形式，我们要执行进位运算. 进位法与加法中的进位法相同. 我们假定进位之后的数仍用 c_i' 表示.

（三）标准化

在进位运算之后，假如在 c_i' 表示的差数中，开头有 l 位数是零，我们记

$$c_i=c_{l+i}', (i=1,2,\cdots,m-l), \tag{12}$$

并且可把 c 表示为

$$c=p^{m-l+1}\sum_{i=1}^{m-l+1}c_ip^{-i} \tag{13}$$

乘法

用(4)(5)式的假定与记号，乘法的三个组成部分是：

(一) 逐位乘

逐位乘与累积在形式上可表示为

$$c_1'=0, c_k'=\sum_{\substack{i,j=1\\ i+j=k}}^{m} a_i b_j', (k=2,\cdots,2m), \tag{14}$$

上式中的乘积 $a_i b_j'$ 或它们的和都可能大于 p，甚至大于机器字长所能准确表示的最大整数 M. 实现(14)式的运算效率与 p 及 M 的选择有密切的关系. 通常我们假定 M 是给定的数，p 可以按条件被自由选择. 如果 $p^2\leqslant M$，则 $a_i b_j+p<M$. 这时，(14)式中的乘法可以进行，但是在累加的过程中仍然可能会超过机器精度范围. 当 $p^2>M$ 时，$a_i b_j$ 会超过 M，(14)式的乘积及累加均不能准确进行. 实际上，运算先要用 $2M$ 精度作下述分解，

$$a_i b_j=xp+y. \tag{15}$$

然后把 x, y 分别加到 c_{i+j-1}' 和 c_{i+j}' 上.

(二) 进位

进位法与加法中的进位法相同. 为了简化记号，我们仍然用 c_i' 来记进位后的数字.

(三) 标准化

当 $c_1'=0$ 时，c 有 $m+n-1$ 位，我们可记

$$c_i=c_{i+1}', (i=1,2,\cdots,m+n-1). \tag{16}$$

当 $c_1'\neq 0$ 时，c 有 $m+n$ 位，我们可得

$$c_i=c_i', (i=1,2,\cdots,m+n). \tag{17}$$

除法

在整数环上没有除法运算，计算机也不能准确地进行除法运算. 在准确计算方法中，进行除法运算($c=a/b$; $a=b.t+r, r<b$)的目的是要知道商 b 和余数 r 而不是通常想要的 c 的近似值.

为了叙述方便和记号简单. 我们采用浮点记数法，并且只需要讨论尾数的除法.

假定

$$a'>b'>0,\quad a'=p^m a=p^m\sum_{i=1}^{m}a_i p^{-i},\tag{18}$$

$$b'=p^m\cdot b\sum_{i=1}^{m}b_i p^{-i},c'=a'/b'=p^{m-n}a/b,c=a/b,\tag{19}$$

并且记 c' 的整数部分为 $[c']$. 当 $a\geqslant b$ 时，$[c']$ 是一个 $m-n$ 位的数. 当 $a<b$ 时，$[c]$ 是一个 $m-n-1$ 位的数，在下面，我们要准确地定出 $[c']$ 和 $r=a-b[c']$. 整个运算的主要过程由下述三部分组成.

（一）逐位求商

记 $a^0=a_0^{(0)}\cdot a_1^{(0)}\cdots a_m^{(0)}=0.a_1a_2\cdots a_m$，并称其为第 0 次被除数.

$$a^{(k)}=a_k^{(k)}\cdot a_{k+1}^{(k)}\cdots a_{k+m}^{(k)}\tag{20}$$

是第 k 次被除数. 取 $a^{(k)}$ 的不足近似值 $\bar{a}^{(k)}$ 及 b 的过剩近似值 $\tilde{b}$，假定它们都能用机器单字长准确地表示并且满足下述两组条件之一：

$$\begin{cases}0\leqslant(a^{(k)}-\bar{a}^{(k)})/a^{(k)}\leqslant p^{-1}/4,(k=0,1,\cdots),\\ a\leqslant(\tilde{b}-b)/b\leqslant p^{-1}/4\end{cases}\tag{21}$$

或

$$\begin{cases}0\leqslant a^{(k)}-\bar{a}^{(k)}\leqslant p^{-1}/4,\\ 0\leqslant(\tilde{b}-b)/b\leqslant 3p^{-1}/8,\end{cases}(k=0,1,\cdots).\tag{22}$$

记 $\bar{c}^{(k)}$ 形如下式：

$$\bar{c}^{(k)}=\bar{a}^{(k)}/\tilde{b}=c_k^{(k)}\cdot c_{k+1}^{(k)}\cdots,(k=0,1,\cdots),\tag{23}$$

定义第 k 次近似商 $c^{(k)}$ 为

$$c^{(k)}=\sum_{i=0}^{k}c_i^{(i)}p^{-i}=c_0^{(0)}\cdot c_1^{(1)}\cdots c_k^{(k)},(k=0,1,\cdots,),\tag{24}$$

第 $k+1$ 次被除数 $a^{(k+1)}$ 如下，并且规定 $a^{(k+1)}$ 是准确的值

$$a^{(k+1)}=p(a^{(k)}-bc_k^{(k)}),\tag{25}$$

因为 c 的整数部分不超过 $m-n+1$ 位，因此上述递推计算只需要进行至 $k=m-n+1$.

（二）进位

当我们计算了第 k 次近似商，(24)式中的 $c^{(k)}$ 就可以用加法运

算中的进位法作进位运算. 假定仍然用进位前的记号表示进位后的当前值. 当 $c_0^{(0)}=0$ 时，$[c']$有 $m-n$ 位数，若 $c_0^{(0)}\neq 0$ 时，$[c']$有 $m-n+1$ 位数，定义

$$c''=c_1'c_2'\cdots c_k'.$$

因为计算带有舍入误差，我们不能保证计算的$[c']$是准确的，因此要进行检验. 这是第三步的工作.

（三）检验及标准化

计算 $r=a-bc'$. 若$|r|\geqslant 2b$ 则计算错误. 若 $b<r<2b$ 则记$[c']=c''+1$，若$-b<r<0$，则记$[c']=c''-1$. 若$-2b<r<-b$ 则记$[c']=c''-2$. 若 $0\leqslant r<b$，则记$[c']=c''$. 计算了准确的$[c']$之后，可以得到准确的剩余 $|a|_b=a-b\cdot[c']$.

现在，我们用一个数值例子来说明逐位除法的主要过程.

假设 $m=6,n=4,p=10,a'=200\ 000,b'=1411$. 按照(19)的记号，$a=0.200\ 000,b=0.1411$，除法 $c=a/b$ 的算法如下：取 $\tilde{b}=0.144$， 当 $k=0$ 时，

$$a^{(0)}=0.200\ 000,\bar{a}^{(0)}=0.20,\bar{c}^{(10)}=1.38,c_0^{(0)}=1,c^{(0)}=0;$$

当 $k=1$ 时，

$$a^{(1)}=10(0.200\ 000-0.1411*1.)=0.589000,a^{(1)}=0.58,$$

$$\bar{c}^{(1)}=4.02,c_1^{(1)}=4,c^{(1)}=1.4;$$

当 $k=2$ 时，

$$a^{(2)}=10(0.589000-0.1411*4)=0.246000,\bar{a}^{(2)}=0.24,$$

$$\bar{c}^{(2)}=1.66,c_2^{(2)}=1,c^{(2)}=1.41;$$

当 $k=3$ 时，

$$a^{(3)}=10(0.246000-0.1411*1)=1.04900,\bar{a}^{(3)}=1.04,$$

$$\bar{c}^{(3)}=7.22,c_3^{(3)}=7,c^{(3)}=1.417.$$

进位运算后的结果 $c_0^{(0)}=1\neq 0$，$[c']$的位数为 3，并且有 $c''=141$.

检验

$$r=200000-1411*141=1049,0<r<1411,$$

于是可得商及余数为：

$$[200000/1411]=141,$$

$$|200000|_{1411}=1049.$$

基数 p 的选择

假设浮点数的尾数字长为二进制 l 位，即机器单字长精度可准确表示的最大整数不超过 $M=2^l$. 为了提高运算效率和节省存贮量，应选取较大的 p. 但是 p 不能太大，必需满足一定的条件. 在运算之前，为了数据能准确地存贮，可以假定

$$p\leqslant M. \tag{26}$$

对于加法和减法运算，要使计算准确，我们可假定

$$p\leqslant M/2. \tag{27}$$

而对于乘法运算，p 的限制条件是

$$p(p-1)\leqslant M. \tag{28}$$

对于除法运算，要得到高精度计算结果，因为要满足除法运算中的条件(19)，(20)式，通常 p 的限定条件为

$$p^2\leqslant M/16. \tag{29}$$

实际上，p 的值与算法的实现有密切的关系，一个精心设计的程序可以选 p 接近 M. 例如

$$p\leqslant M/45. \tag{30}$$

为了使用方便，人们习惯用十进制数，可以选 p 为 10^s 的形式. s 是正整数. 在节 1.4 中，我们介绍一种实用的 p 值.

1.2 快速乘法

假设 $m\geqslant n$，一个 m 位数与一个 n 位数相加需要作 n 个逐位加和 n 次进位运算. 若进行乘法，则要执行 $m.n$ 次逐位乘和 $m(n-1)$ 次逐位加以及进位运算. 一般地说，加法运算量为 $O(n)$，乘法运算量为 $O(m.n)$. 对于单字长的运算，计算机硬件设备已能使执行乘法的时间与执行加法的时间差别不大. 但是，对于大整数的运算，无论软件或硬件都没有能力使计算乘法的时间与计算加法的时间接近. 要节省计算乘法的时间，只有从算法着手. 我们假定单字长

精度运算的乘法时间与加法的时间相同，寻找快速乘法算法.

为叙述简单起见，我们先用两位 p 进制数的乘法为例说明一个快速算法的主要思想.

快速乘法(一)

假设

$$a=a_1 \cdot p+a_2, \qquad b=b_1 p+b_2, \tag{1}$$

可得

$$c=ab=a_1b_1p^2+(a_1b_2+a_2b_1)p+a_2b_2. \tag{2}$$

假定 a_1,a_2,b_1,b_2 都是单字长精度的数. 上述运算包含 4 次单精度的乘法. 结果是 4 个双精度的数 a_1b_1,a_1b_2,a_2b_1 和 a_2b_2. 如果改写上式为等价的公式

$$c=a_1b_1(p^2+p)+(a_1-a_2)(b_2-b_1)p+a_2b_2(p+1), \tag{3}$$

(3)式中包括三次单精度的乘法，得到三个双精度的数 a_1b_1，$(a_1-a_2)(b_2-b_1)$和 a_2b_2. 因此，我们有理由希望用上式执行乘法运算会比一般的算法节省时间. 为了说明用两个公式执行乘法的实际运算量，下面给出一个数值例子：

给定 $a=5678$，$b=8765$，$p=100$. 逐位乘得出

$$\begin{cases} a_1b_1=56*87=4872, \\ a_1b_2=56*65=3640, \\ a_2b_1=78*87=6786, \\ a_2b_2=78*65=5070, \end{cases} \tag{4}$$

累加得出

$$\begin{array}{rcccc} & 48 & 72 & & \\ & & 36 & 40 & \\ & & 67 & 86 & \\ + & & & 50 & 70 \\ \hline & 49 & 76 & 76 & 70 \end{array} \tag{5}$$

上述运算包含两位数字的 4 次乘法和 4 次加法. 按照(3)我们可得到下列结果：