实用计算方法

张善杰 唐 汉 高瑞章 编著

南京大学出版社

实用计算方法

张善杰　唐汉　高瑞章　编著

南京大学出版社

内 容 简 介

实用计算方法是参照本科"计算方法"教学大纲，在总结多年教学经验和科研工作中有关数值计算经验的基础上编写的教材。

本书包括现今常用的各种算法，如函数插值、非线性方程求根、数值积分、线性方程组求解、矩阵特征值问题、快速富里叶变换、微分方程初值问题、二阶偏微分方程差分和有限元数值解法以及最优化方法等。本书内容丰富，取材新，有较强的理论性和系统性，并特别讲究实用，叙述上由浅入深，有算法、程序流图，易读性强，每章末附有一些 FORTRAN 77 实用程序。

本书可供理工科非计算机和数学专业本科二年级学生作教材使用或自学用；也可供从事数值计算的科技人员参考。

实 用 计 算 方 法

张善杰　唐　汉　高瑞章　编著

*

南京大学出版社出版

（南京大学校内　邮政编码：210093）

江苏省新华书店发行　　南京京新印刷厂印刷

*

开本：787×1092　1/16　印张：22.125　字数：552千

1998年4月第1版　　1998年4月第1次印刷

印数：1—2000

ISBN　7—305—02818—5/O·197

定价：25.00元

前　言

随着计算机科学和技术以及计算数学的迅速发展，科学与工程计算已广泛应用于自然科学和工程技术的各个领域，成为在科技工作中与理论分析、科学实验相并列的第三种科学方法。现今无论在传统学科领域还是在高科技领域均少不了数值计算这一类工作，特别是它已成为优化工程设计，进行数值模拟实验以代替耗资巨大的真实试验的一种重要手段，有鉴于此，目前"计算方法"已被大多数理工科院校列为本科生和硕士生的必修基础课程。

本书是参照"计算方法"教学大纲为理工科非计算机和数学专业本科二年级学生而编写的教材。为反映出本书要强调的实用特点，取名为《实用计算方法》；在内容和取材份量上是以课内3小时/周和上机练习2小时/周的一学期教学计划安排为依据的。

本书较详细地介绍了一些现今常用算法，并作为本书特色，在叙述算法时，从简单例子开始，总结计算规律，由浅入深；书中大都给出了便于编程的相应程序流图，并在每章末作为附录给出了与所述算法、程序流图对应的用FORTRAN 77语言编写的实用程序，它们是采用Subroutine子程序形式编写，便于调用，这部分也是作为给同学编写实用程序的范例。程序均可在PC-386,486,586或其兼容机上运行。所有程序，仅供参考。

本书在内容安排和取舍上充分注意了其理论性和系统性，限于学时以及侧重于实用性的考虑，而不得不舍弃一些理论公式、定理和结论的证明；在理论要紧密联系实际的思想指导下，力求少而精，讲究实用；在叙述上保持各章具有一定的相对独立性，并注意到各章之间的联系和前后的呼应。每章中均附有富有启发性的思考题和适量的上机练习题，连同附录中提供的程序，可利于在边学理论、边实践过程中提高编程计算技巧，解决实际复杂算题的应变能力。提供一些程序也是针对在已有很多程序手册和程序汇编出版的现状下，改进计算方法教学，让学生学会如何利用或部分利用现成程序的一种尝试。

本书在叙述算法时配有程序流图和程序，可增加易读性，因而也适用于具有一般高等数学知识包括线性代数知识以及FORTRAN语言知识的读者自学。

本书是编者在近五年来为南京大学信息物理系硕士生和作为公选课开设的"常用算法"的教学笔记，以及编者们多年从事与计算电磁学有关科研工作所取得的数值计算经验的基础上总结、整理而成的。

本书第一至第七章由张善杰执笔编写，第九、十章由唐汉执笔编写，第八章由高瑞章执笔编写。

本书在编写过程中曾得到校、系领导的大力支持，在此表示谢意；编写中曾参阅了有关书籍和文献，受到不少启示，这里也向有关作者表示谢意。由于作者水平有限，书中缺点和错误在所难免，热忱地欢迎广大读者和有关专家学者不吝提出宝贵建议和批评指正。

目　录

第一章 引 论

本章简要地叙述了常用计算方法课程的主要内容；介绍了计算机中数的浮点表示以及计算机数系，以及误差的定义和基本概念，着重讨论了舍入误差、截断误差及作基本运算时误差的传播；讨论了构造算法和编程中应注意的几个问题，结合一些算例，说明由于计算机字长对数值计算精度的影响，强调了研究算法的重要性。

1.1 计算方法的主要内容

计算方法或数值计算方法是数学的一个分支，也称为计算数学或数值分析，它研究的内容是科学和工程中所遇到的各类典型数学问题在计算机上的数值解法及其有关理论，涉及的问题十分广泛。计算数学业已广泛应用于许多学科领域，并形成了一些边缘学科，诸如计算物理、天文、力学、电磁学、化学、生物学等。对于常用计算方法课程所包含的内容则是指最常遇到的各数学问题的基本数值计算方法，它们是

1. 求解线性方程组

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2 \\ \cdots\ \cdots\ \cdots \qquad\qquad \cdots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n = b_n \end{cases}$$

方程组的矩阵形式为：

$$AX = B$$

式中 A 为 $n\times n$ 系数矩阵，X，B 为 $n\times 1$ 列矢量。

2. 函数插值和数据拟合

插值问题是给定一组离散数据

$$(x_i, y_i) \quad i = 0,1,\cdots,n$$

寻求一个插值函数 $P_n(x)$，它应满足条件

$$P_n(x_i) = y_i \quad i = 0,1,\cdots,n$$

数据拟合则是寻求一个函数，它从整体上来看最“贴近”所给各个数据点。

3. 非线性方程和非线性方程组求根

非线性方程包括代数方程和超越方程。

例如求超越方程 $x-\coth x=0$ 的根。

4. 快速富里叶变换(FFT)

问题是寻求计算离散富里叶变换对

$$F_n = \frac{1}{N}\sum_{K=0}^{N-1} f_k e^{-j2\pi kn/N} \qquad \text{(正变换)} \quad n = 0,1,\cdots,N-1$$

$$f_n = \sum_{n=0}^{n-1} F_n e^{j2\pi kn/N} \qquad \text{(逆变换)} \quad k = 0,1,\cdots,N-1$$

的快速算法。

5. 数值积分

积分

$$I \doteq \int_a^b f(x)\mathrm{d}x$$

除非 $f(x)$为某些特殊形式，一般很难积出，因而数值积分法是一个常用的计算积分方法。

6. 矩阵特征值问题

问题是已知矩阵$[A](a_{ij}, i, j=1,2,\cdots,n)$ 要求解方程：$[A][X]=\lambda[X]$的本征值 $\lambda_i(i=1,2,\cdots,n)$和相应的本征矢量$[X_i]$。

6. 常微分方程初值问题的数值解法

求给定方程$\frac{\mathrm{d}y}{\mathrm{d}x}=f(x,y)$及初始条件 $y(x_0)=y_0$的数值解。

7. 偏微分方程边值问题的数值解法

例如采用差分法或有限元法求解二维 Poisson 方程

$$\Delta u = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = f(x,y)$$

满足边界条件

$$u|_\tau = \varphi(x,y)$$

在 D 内的数值解，式中 D 为有限求解域，τ 为 D 的周界。

8. 最优化方法

例如求一元函数 $f(x)$和多元函数 $f(x_1,x_2,\cdots,x_n)$无约束极小值等。

学习计算方法联系编程实践是非常重要的，这不仅有利于对算法原理的深入理解，而且更重要的是将理论用于解决实际问题，并使算法在实践中得到改进和提高。在计算实践中，算法的选择是十分重要的，如果对一个数字问题选择了坏的算法，就很难设想会得出好的结果。另方面，一个好的算法也要通过细心周到的编程才能得到体现和付诸实用。实践中我们也会遇到这样的数学问题：当它的输入数据有微小改变时，却可导致解出现有很大变化，此即所谓病态问题或坏条件问题，此时尽管病态是问题的固有属性与算法无关，但从防止病态的表现更加恶化考虑，则不仅要选择好的算法，而且要注意编程技巧，和采用具有 16 位有效数字位的双精度计算。

学习本课程的目的最终是要求学会根据所学算法编写出实用程序，因此将常用计算方法与编程实践有机结合进行叙述，无疑是有益的。为此，本书在算法选择上对那些在实践中使用频率颇高的常用算法均作了较详细叙述，并给出了方便编程的程序流图。一个好的算法和程序应该是：

程序稳定性好，精度高；可适用参变量范围宽；计算量小，计算需时少，占内存少；使用方便等。这些要求有些往往是矛盾的，例如要求适用范围宽，会导致程序复杂和内存增加，因而需要

折衷考虑或满足算题的重点要求。

此外，人们从实用角度还关心如何掌握使用和利用现今在程序手册、汇编、程序集之类书中的现成程序。由于这些程序说明比较简短，一般来说要读懂它们是不易的。因此，本书在每章末均附有与该章所给程序流图相应的常用算法的 FORTRAN－77 实用程序，并附有范例，便于阅读、理解算法和熟悉调用。相信这对读者日后的科学计算和掌握利用、借鉴程序手册或汇编中的现成程序是有帮助的。

本书取名为实用计算方法，以反映其在内容和叙述上的特点。

1.2 计算机中数的浮点表示

1. 以 β 为基的数系（β 进制数）

β 进制数系有 β 个不同数字 $0,1,2,\cdots,\beta-1$

(1)十进制数($\beta=10$)，十个数字为 $0,1,2,\cdots,9$

例如：$(364)_{10}=(3\times10^2+6\times10^1+4\times10^0)$

$$(5188.51)_{10}=5\times10^3+1\times10^2+8\times10^1+8\times10^0+5\times10^{-1}+1\times10^{-2}$$

(2)二进制数($\beta=2$)，二个数字为 0,1

例如：$(10101)_2=1\times2^4+0\times2^3+1\times2^2+0\times2^1+1\times2^0$

$$=(21)_{10}$$

$$-(10.101)_2=-(1\times2^1+0\times2^0+1\times2^{-1}+0\times2^{-2}+1\times2^{-3})$$

$$=-(2.625)_{10}$$

(3)十六进制数，十六个数字为 $0,1,2,\cdots,9,A,B,C,D,E,F$。

例如：$(5C4)_{16}=5\times16^2+C\times16^1+4\times16^0$

$$=5\times16^2+12\times16^1+4$$

$$=(1476)_{10}$$

2. 定点数

位数有限，有 n 位整数，m 位小数的 β 进制数可表为

$$x=\pm(a_{n-1}\beta^{n-1}+a_{n-2}\beta^{n-2}+\cdots+a_0\beta^0+a_{-1}\beta^{-1}+a_{-2}\beta^{-2}+\cdots+a_{-m}\beta^{-m})$$

例如，$\beta=10$，若取 $n=4,m=4$ 时

109.312	可表为	0109.3120
5.888	可表为	0005.8880
−0.4375	可表为	−0000.4375
4321	可表为	4321.0000

这种把小数点固定在指定位置上，位数有限的数称为定点数。对于 $n=4,m=4$ 情形，绝对值最小和最大的数分别是

$$\pm0000.0001 \quad 和 \pm9999.9999$$

所以定点数所能表示的数的范围是很小的。

3. 浮点数

将非零的十进制数写成 0.1～1 之间的小数乘以 10 为底的幂的形式：

$$x = \pm\ 0.a_1a_2\cdots a_t \times 10^c$$

则小数点的位置随 c 值浮动，而称之为数的浮点表示或浮点数。式中 $0 \leqslant a_1, a_2, \cdots a_t \leqslant 9$ 称为浮点数的尾数，共有 t 位；c 称为浮点数的阶。$a_1 \neq 0$ 时称为规格化的浮点数表示，$a_1 = 0$ 时称为非规格化的浮点数表示。例如

$$-12347800 = -0.123478 \times 10^8$$

$$0.00001993 = 0.1993 \times 10^{-4}$$

$$1993.12 = 0.0199312 \times 10^5 \quad \text{（非规格化浮点表示）}$$

同样地，对其它数制，如

$$(1011.11)_2 = 0.101111 \times 2^4$$

$$(-95.3FA)_{16} = -0.953FA \times 16^2$$

计算机中的数是由有限个二进制数组成，对于特定的计算机，浮点表示的小数部分位数是固定的（共 t 位），t 也称为计算机字长；阶 c 亦有确定的范围：$m_s \leqslant c \leqslant M_s$，一般 $m_s = -M_s$ 或 $m_s = -M_s \pm 1$。

受具体计算机限制的浮点数构成该机器的数系，它由 (β, t, M_s, m_s) 确定，记为 $F(\beta, t, m_s, M_s)$。因此计算机的数系实际上仅是实数系 R 的一个小的离散子集。设正浮点数的最大正数为 γ，最小正数 α，则当实数 $|x| > \gamma$ 时，机器就出现上溢而中断计算；当 $|x| < \alpha$ 时，称为下溢，机器自动作为（机器）零处理。如果对于某实数 x 不属于数系 $F(\beta, t, m_s, M_s)$，但有 $\alpha \leqslant |x| \leqslant \gamma$，则机器将采用四舍五入或直接截断（只舍不入）处理。例如对于数系 $F(10, 7, -99, 98)$，$\frac{1}{3}$ 和 π 的浮点数表示为

$$fl = (\frac{1}{3}) = 0.3333333 \times 10^0$$

$$fl(\pi) = 0.3141593 \times 10^1$$

这里 fl 是 *float* 的缩写，表示浮点数。

因而，对机器而言，凡介于机器容许的最大数与最小数之间的任何数，除非恰好是机器数外，一经送入计算机，即成为近似值参加运算，并且每次运算都可能有舍入误差。

1.3 误差的基本概念

1. 误差来源

利用计算机对某个量进行计算时，数值结果与它的真实值之间总存有差异，其原因是多方面的，除了计算中的人为疏忽或错误，可以归纳为以下几种来源。

(1)原始误差

包括模型误差和原始数据误差。

对具体科技问题的理论分析一般是通过在所建立的能用于描述问题各主要物理量并能够求解的数学模型上进行的。数学模型所描述的问题与原问题总具有差别，由此产生的数值结果误差称为模型误差。

原始数据误差是指计算初始所依据的数据本身就并非准确值而带有的误差，例如万有引

力常数 G、光速 c、空气的介电常数 ε_0 等一些物理常数都是通过测量而得到的、具有一定精度的数据；又如 $\pi, e, \frac{1}{3}, \frac{1}{7}, -\sqrt{2}$ 等一些数学常数由于计算机的浮点数的有限字长，它们也被截成含有误差的一定位数的数。

(2)截断误差和舍入误差

这是计算方法中所要考虑的误差。

在数值计算中经常会遇到计算一个无穷序列(例如迭代法求方程根，迭代法求解线性方程组等)或无穷级数。由于计算次数只能是有限的，这就需要在计算一定次数后终止计算，截断的位置一般取决于计算精度要求。由此产生的误差即为截断误差。例如

$$\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \cdots$$

$$= \sum_{n=0}^{\infty} (-1)^n \frac{x^{2n+1}}{(2n+1)!}$$

计算时取 N 项，则截断误差为

$$\left|\sin x - \sum_{n=0}^{N} (-1)^n \frac{x^{2N+1}}{(2N+1)!}\right| = \left|\sum_{N+1}^{\infty} (-1)^n \frac{x^{2n+1}}{(2n+1)!}\right| \leqslant \left|\frac{x^{2N+3}}{(2N+3)!}\right|$$

由于计算机浮点数的有限字长，计算过程中的数据以及原始数据都是按四舍五入或只舍不入规则截成有限位数的数，由此而引起的计算结果误差即为舍入误差。这种情形对于手算也是存在的。例如

$$\pi = 3.14159265358979392384\cdots$$

$$e = 2.71828182845904523 53\cdots$$

以及 $\frac{1}{3}$。

按四舍五入规则，用具有 7 位的浮点数(单精度数)计算时为

$$\pi = .3141593E + 01$$

$$e = .2718282E + 01$$

$$\frac{1}{3} = .3333333E + 00$$

而用具有 16 位的浮点数(双精度数)计算时为

$$\pi = .3141592653589793D + 01$$

$$e = .2718281828459045D + 01$$

$$\frac{1}{3} = .3333333333333333D + 00$$

2. 误差和误差限

设 x 和 $\tilde{x}$ 分别表示某量的准确值和近似值，则定义 $\tilde{x}$ 的绝对误差(简称误差)为

$$\varepsilon(x) = \tilde{x} - x \tag{1.3.1}$$

而 $\tilde{x}$ 的相对误差为

$$\varepsilon_r(x) = \frac{\tilde{x} - x}{x} = \frac{\varepsilon(x)}{x} \tag{1.3.2}$$

如果 x 为已知，则可由(1.3.1)和(1.3.2)式计算出 $\tilde{x}$ 的绝对误差和相对误差。但准确值 x 通常是不知道的，因而不可能得到 $\varepsilon(x)$ 和 $\varepsilon_r(x)$ 的真值，但可通过计算或从测量数据估计出 $\varepsilon(x)$ 和 $\varepsilon_r(x)$ 的上界 δx 和 $\delta_r x$，即有

$$|\varepsilon(x)| = |\tilde{x}-x| \leqslant \delta(x) \tag{1.3.3}$$

$$|\varepsilon_r(x)| = \left|\frac{\tilde{x}-x}{x}\right| \leqslant \delta_r(x) \tag{1.3.4}$$

$\delta(x)$和$\delta_r(x)$分别称为$\tilde{x}$的绝对误差限和相对误差限。由此可知准确值x所在范围为

$$\tilde{x} - \delta(x) \leqslant x \leqslant \tilde{x} + \delta(x)$$

在实际应用中，相对误差具有更重要的意义，它反映了一个近似数相对于其本身的准确程度。

3. 有效数字

某一实数用四舍五入或只舍不入法取其准确值的前n位作为近似值$\tilde{x}$，则称$\tilde{x}$具有n位有效数字。例如

$$\pi = 3.141592653589793\cdots$$

取五位有效数字为

$\tilde{\pi}_5 = 3.1416$ （四舍五入）

$|\tilde{\pi}_5 - \pi| = |-0.000007346| \leqslant \frac{1}{2} \times 10^{-4}$ （四舍五入误差限）

或 $\tilde{\pi}_5 = 3.1415$ （只舍不入）

$|\tilde{\pi}_5 - \pi| = |-0.00009265| \leqslant 10^{-4}$（只舍不入误差限）

换句话说，我们可将写出的具有有限位数的数，使左边第一个不为零的数字起到它最右边的数字都认为是有效数字，例如

$\tilde{x}$	有效数字	误差限
0.0053	2 位	0.5×10^{-4}
0.123	3 位	0.3×10^{-3}
518.8	4 位	0.5×10^{-1}

注意：写法 1.588 与 1.5880 含义有不同，前者是四位有效数字，后者则有五位有效数字。

一般地，用四舍五入法得到的具有n位有效数字的近似数$\tilde{x}$可表为

$$\tilde{x} = \pm 10^m(a_1 \times 10^{-1} + a_2 \times 10^{-2} + \cdots + a_n \times 10^{-n}) \tag{1.3.5}$$

这里，$a_1, a_2, \cdots$为 0,1,…,9 的某个整数。其绝对误差限为

$$\delta(x) = |\tilde{x} - x| \leqslant \frac{1}{2} \times 10^{m-n} \tag{1.3.6}$$

若已知近似数$\tilde{x}$的有效位数n和第一个非零数a_1，则其相对误差限为

$$\delta_r(x) = \left|\frac{\tilde{x}-x}{x}\right| \simeq \left|\frac{\tilde{x}-x}{\tilde{x}}\right| \leqslant \frac{\frac{1}{2}\times 10^{m-n}}{a_1 \times 10^{m-1}} = \frac{1}{2a_1} \times 10^{-n+1} \tag{1.3.7}$$

例如，$\tilde{x}=518.8=0.5188\times10^3$，有$m=3, n=4$，于是

$$\delta(x) \leqslant \frac{1}{2} \times 10^{-1}; \quad \delta_r(x) \leqslant 10^{-4}$$

反之，若近似数$\tilde{x}$的相对误差满足(1.3.7)式，则$\tilde{x}$至少有n位有效数字。

以下叙述，关于机器的浮点数表示如无特别说明，将认为是采用四舍五入法则。

4. 近似数的和、差、积、商的误差估计

计算机除整型数是采用浮点数运算的，因而运算数据和原始数据均具有舍入误差，都是近似数，现在我们来分析和估计它们在作为和差积商基本算术运算后的误差，亦即分析运算中误差的传播情况。

设数学问题的解 y 与变量 $x_1,x_2,\cdots,x_n$ 有关，即 $y=f(x_1,x_2,\cdots,x_n)$。若所给的是一组相应的变量近似值 $\tilde{x}_1,\tilde{x}_2,\cdots,\tilde{x}_n$，则相应的解 y 就是近似解 $\tilde{y}$。设 $\tilde{x}_1,\tilde{x}_2,\cdots,\tilde{x}_n$ 的误差较小，应用 y 的 Taylor 展开式，略去高阶项后则可得解的绝对误差为

$$\begin{aligned}\delta(y) &= y-\tilde{y}=f(x_1,x_2,\cdots,x_n)-\tilde{f}(\tilde{x}_1,\tilde{x}_2,\cdots,\tilde{x}_n)\\ &\simeq \sum_{i=1}^{n}\frac{\partial f(x_1,x_2,\cdots,x_n)}{\partial x_i}\delta(x_i)\end{aligned} \tag{1.3.8}$$

而解的相对误差为

$$\delta_r(y)=\frac{\delta(y)}{y}=\sum_{i=1}^{n}\frac{\partial f(x_1,x_2,\cdots,x_n)}{\partial x_i}\frac{x_i}{f(x_1,x_2,\cdots,x_n)}\delta_r(x_i) \tag{1.3.9}$$

由(1.3.8)和(1.3.9)式，我们可导得近似数作和差积商基本运算前后误差之间的关系。下面分析 f 仅含 x_1,x_2 两个数据变量的情形。

(1)和、差的误差

$$f(x_1,x_2)=x_1+x_2$$

由(1.3.8)和(1.3.9)式可得

$$\begin{aligned}\delta(x_1+x_2) &\simeq \frac{\partial}{\partial x_1}(x_1+x_2)\delta x_1+\frac{\partial}{\partial x_2}(x_1+x_2)\delta x_2\\ &=\delta x_1+\delta x_2\end{aligned} \tag{1.3.10}$$

$$\begin{aligned}\delta_r(x_1+x_2) &\simeq \frac{\partial}{\partial x_1}(x_1-x_2)\frac{x_1}{x_1+x_2}\delta_r(x_1)+\frac{\partial}{\partial x_2}(x_1+x_2)\frac{x_2}{x_1+x_2}\delta_r(x_2)\\ &=\frac{x_1}{x_1+x_2}\delta_r(x_1)+\frac{x_2}{x_1+x_2}\delta(x_2)\end{aligned} \tag{1.3.11}$$

当 x_1 与 x_2 同号时，相当于加法情形，我们有

$$0\leqslant\left|\frac{x_1}{x_1+x_2}\right|\leqslant 1 \quad 或 \quad 0\leqslant\left|\frac{x_2}{x_1+x_2}\right|\leqslant 1 \tag{1.3.12}$$

于是由(1.3.10)和(1.3.11)式可得

$$|\delta(x_1+x_2)|\leqslant|\delta(x_1)|+|\delta(x_2)| \tag{1.3.13}$$

和

$$|\delta_r(x_1+x_2)|\leqslant|\delta_r(x_1)|+|\delta_r(x_2)| \tag{1.3.14}$$

这就是说加法结果的绝对误差限和相对误差限都不会大于各项的绝对和相对误差限之和。

当 x_1 与 x_2 异号时，相当于减法情形，此时因分式 $\left|\frac{x_1}{x_1+x_2}\right|$ 和 $\left|\frac{x_2}{x_1+x_2}\right|$ 均大于 1，而没有简单的误差限。

由(1.3.11)式可见，当 $x_1+x_2\approx 0$ 时，将导致

$$|\delta_r(x_1+x_2)|>>|\delta_r(x_1)|+|\delta_r(x_2)|$$

即相对误差变大。因此，在计算中应尽可能避免 $x_1+x_2\simeq 0$ 两个相近数相减，以防止有效数字

严重丢失。例如 a,b,c 三个数均是具有七位有效数字，其中 a,b 为首的四位数字相同，因而 $a-b$ 的结果是仅有三位有效数字的数，再与 C 相加其结果也只有三位有效数字了。但这时若改变计算次序就可能使其结果的有效数字不丢失。这表明 $(x+y)+z$ 可与 $x+(y+z)$ 结果大不相同，也就是对计算机数系来说结合律不适用。

(2)积的误差

$$f(x_1,x_2)=x_1\cdot x_2$$

由(1.3.8)和(1.3.9)式可得

$$\begin{aligned}\delta(x_1x_2)&\approx\frac{\partial}{\partial x_1}(x_1x_2)\delta x_1+\frac{\partial}{\partial x_2}(x_1x_2)\delta x_2\\&=x_2\delta x_1+x_1\delta x_2\end{aligned}\tag{1.3.15}$$

和

$$\begin{aligned}\delta_r(x_1x_2)&\simeq\frac{\partial}{\partial x_1}(x_1x_2)\frac{x_1}{x_1x_2}\delta_r(x_1)+\frac{\partial}{\partial x_2}(x_1x_2)\frac{x_2}{x_1x_2}\delta_r(x_2)\\&=\delta_r(x_1)+\delta_r(x_2)\end{aligned}\tag{1.3.16}$$

这表明当 x_1 或 x_2 的绝对值很大时 $|\delta(x_1x_2)|$ 的值可能很大，即导致 x_1 或 x_2 绝对误差放大；而乘法总的相对误差不会大于各项相对误差绝值之和。

(3)商的误差

$$f(x_1,x_2)=x_1/x_2$$

由(1.3.8)和(1.3.9)式可得

$$\begin{aligned}\delta\left(\frac{x_1}{x_2}\right)&\simeq\frac{\partial}{\partial x_1}\left(\frac{x_1}{x_2}\right)\delta(x_1)+\frac{\partial}{\partial x_2}\left(\frac{x_1}{x_2}\right)\delta(x_2)\\&=\frac{1}{x_2}\delta(x_1)-\frac{x_1}{x_2^2}\delta(x_2)\end{aligned}\tag{1.3.17}$$

和

$$\begin{aligned}\delta_r\left(\frac{x_1}{x_2}\right)&\simeq\frac{\partial}{\partial x_1}\left(\frac{x_1}{x_2}\right)x_2\delta_r(x_1)+\frac{\partial}{\partial x_2}\left(\frac{x_1}{x_2}\right)\frac{x_2^2}{x_1}\delta_r(x_2)\\&=\delta_r(x_1)-\delta_2(x_2)\end{aligned}\tag{1.3.18}$$

这表明当作为除数的 x_1 或 x_2 接近于零时，x_1 与 x_2 之商的绝对误差被严重放大而导致精度降低；而除法运算的相对误差不大于各项相对误差限之和。

综上分析可见，在实际计算中两个相近数的相减、大的乘数、接近零的除数都会导致误差的传播和放大，是计算中的禁忌，应尽力避免。

1.4 构造算法与编程中应注意的几个问题

初次接触用计算机进行科学计算的人往往会认为任何复杂的计算问题使用计算机硬算总会算出来的，并且用计算机得到的结果总是精确和可靠的。其实不然，很多情况下不研究具体问题的计算策略靠蛮干硬算是行不通的，计算结果的正确性和所能达到的精度依赖于算法的稳定性，不能无根据地信赖。通常在计算机给出某一算题结果时，都需要采用有效的方法去验证结果的正确性和分析数值结果的精度。有经验的人会感到验证结果的正确性和判断计算结

果精度往往并非易事。

构造算法和编程中需要注意计算机的以下特点：

1. 计算机中的数具有有限字长，含有舍入误差；存在有最大数和最小数限制，超出最大数就出现溢出，低于最小数作机器零处理，此时若作除数，就会导致除零溢出。因而要防止误差传播和积累，注意计算过程的稳定性。

2. 计算机可进行加减乘除及逻辑运算。考虑任何一个复杂问题的算法就是要尽可能地通过逻辑运算功能将复杂计算化为简单算术计算的重复，即构成循环。

本节中将对计算中应注意的几个问题结合算例给予说明。

1. 避免两个相近数的相减

当出现两个相近数相减时，数值结果的有效位数将严重丢失。克服的办法是在可能的情况下对算式进行预处理，变换算式避开两相近数相减，或者采用双精度计算，以保留多的有效位数。

例 1：

$$a=0.1234567;\ b=0.1234531;$$

$$a-b=0.0000036$$

有效位数丢失 5 位。

例 2：计算 $\sqrt{x+1}-\sqrt{x}$ $(x>>1)$

避开两相近数相减，用变换的算式计算。

$$\sqrt{x+1}-\sqrt{x}=(\sqrt{x+1}-\sqrt{x})\left(\frac{\sqrt{x+1}+\sqrt{x}}{\sqrt{x+1}+\sqrt{x}}\right)$$

$$=\frac{1}{\sqrt{x+1}+\sqrt{x}}$$

例 3：计算 $f(x)=\dfrac{1-\cos x}{x^2}$ $(0<x<<1)$

因

$$\cos x=1-\frac{x^2}{2!}+\frac{x^4}{4!}-\frac{x^6}{6!}+\cdots$$

故

$$f(x)=\frac{1}{2!}-\frac{x^2}{4!}+\frac{x^4}{6!}-\cdots$$

值得提醒注意的是 $a-b$ 两个相近数相减，往往 a,b 可能代表两个复杂算式，这时就需针对具体问题寻求对策了；计算时的表现是 a,b 精度较高，而其差值的精度出乎意料的低。

例 4：求 $ax^2+bx+c=0$ 的根

熟知

$$x_{1,2}=\frac{-b\pm\sqrt{b^2-4ac}}{2a} \tag{1.4.1}$$

当 $b^2>>4ac$ 时，有

$$x_1\approx 0;\quad x_2=-\frac{b}{a}$$

实际上这相当于求解方程 $ax^2+bx=0$ 时 c 的作用被“吃”掉了。正确的算法是

$$x_2=\frac{-b-\mathrm{sign}(b)\sqrt{b^2-4ac}}{2a} \tag{1.4.2}$$

而由关系式

$$x_1\cdot x_2=c/a$$

得出

$$x_1=\frac{c}{ax_2} \tag{1.4.3}$$

2. 改进算法，减少计算量

一个算法所需计算量愈小，则使舍入误差的积累机会也愈少，且可节省机时。

例 5：计算多项式

$$P_n(x)=a_0+a_1x+a_2x^2+\cdots+a_nx^n \tag{1.4.4}$$

如果采用直算，计算 a_kx^k 需作 k 次乘法，故计算 $P_n(x)$ 共需 $1+2+3+\cdots+n=\frac{1}{2}n(n+1)$ 次乘法和 n 次加法。但采用如下秦九韶算法（西方称 Hornor 算法）将 $P_n(x)$ 写成：

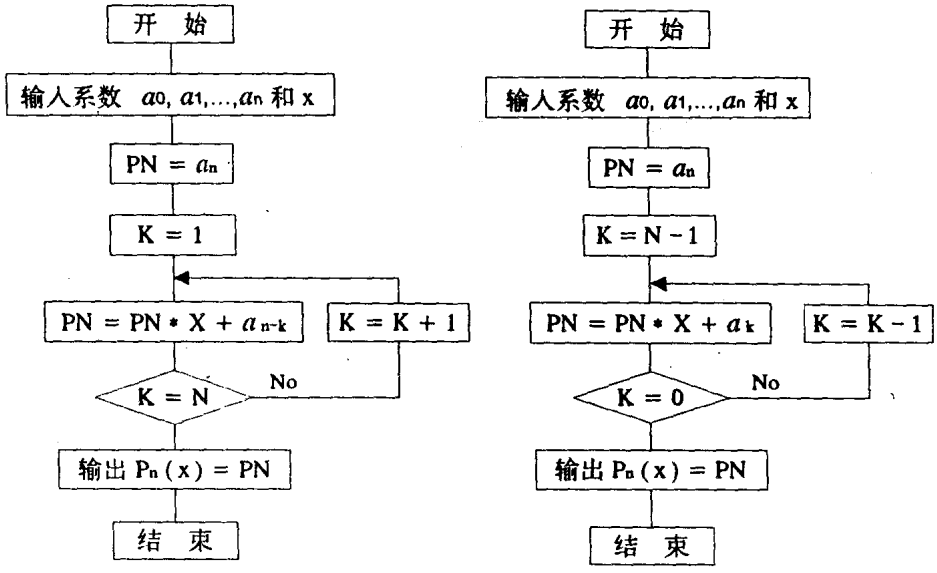
$$P_n(x)=(\underbrace{(\cdots(a_nx+a_{n-1})x+a_{n-2})x+\cdots+a_1}_{n-1\text{个}})x+a_0 \tag{1.4.5}$$

或

$$P_n(x)=a_0+x(a_1+x(a_2+x(a_3+\cdots+\underbrace{a_nx)\ )\cdots)}_{n-1\text{个}} \tag{1.4.6}$$

则仅需作 n 次乘法和 n 次加法。

若多项式系数 $a_0,a_1,\cdots,a_n$ 是存入数组 A，则(1.4.5)式 $P_n(x)$ 可按图 1.1(a)或(b)的程序流图计算。



(a)　　(b)

图 1.1　计算多项式 $P_n(x)$ 的程序流图

例 6：求级数和

$$\cos x=1-\frac{x^2}{2!}+\frac{x^4}{4!}-\frac{x^6}{6!}+\cdots$$