计算机数值方法

徐士良 编

计 算 机 数 值 方 法

康金章　黄国柱
宋佩弦　陈荣山　编

厦门大学出版社

1988．5

计 算 机 数 值 方 法

康金章 黄国柱
宋佩弦 陈荣山 编

厦门大学出版社出版发行
厦门大学印刷厂印刷

开本787×1092 1/16 25·375印张 585千字
1988年5月 第1版 1988年5月第1次印刷
印数：0,001—5,000册
ISBN 7—5615—0120—x
T·2 定价：5.90元

前　言

计算机数值方法是计算机科学的重要内容，它的主要作用在于更有效地使用电子计算机，以求得各种数学问题的数值解答。随着科学技术的发展和计算机的广泛应用，建立数值方法的基本概念和介绍计算机上行之有效的常用算法，对计算机使用者来说是完全必要的。为了普及和发展计算机的应用技术，提高计算机的使用效率，缩短解题周期，减少重复劳动，发展数值方法，我们编写了《计算机数值方法》一书，以期满足广大读者的需要。

全书共八章，主要内容有误差与插值；数值积分；逼近与拟合；线代数计算方法；方程求根；常微分方程的数值解法；偏微分方程的数值解法及其它有用的一些算法。书中选入了应用于科学、工程技术、管理和教育等方面的七十多个常用方法，每个方法都按五大部分编写，即（一）方法的功能；方法的用途，方法的适用范围和主要性能。(二)方法原理及其推导：方法的基本思想，方法的数学理论，性质和定理的证明。(三)计算过程概要：给出计算步骤，过程中所做的某些处理和结论，列出有关计算公式。(四)程序及其说明：给出 FORTRAN 程序，使用说明和应注意的事项。（五）应用：给出具体数值例子，理解该过程的使用方法，给出相应的程序和计算结果。

本书在编写过程中，力求阐明构造算法的基本思想和技巧。在叙述上力求对数值方法中某些基本概念和基本原理，尽可能严格精确，便于培养学生具有一定的理论分析能力。力求做到由浅入深，由易到难，循序渐进，便于自学能力的培养。在内容上尽量选取常用的行之有效的算法，选取的范围和深度掌握恰当，推理较严密，并给出典型例子及相应的程序，便于实际应用。鉴于数值方法在近年来发展迅速，提出了许多新方法、新思想，因此，在选材时也注意了推陈出新，适当反映这一分支的新成果。为了适应不同专业的需要，书中列出了选学内容，这些内容，都在标题中加了星号“*”，使用时可酌情取舍。

本书由福州大学，厦门大学《计算机数值方法》编写组的同志共同讨论和审定的。由福州大学计算机系主任康金章教授主编。第一、二、三章由黄国柱同志编写；第四、五章由陈荣山同志编写；第六、七章由宋佩弦同志编写；第八章由康金章教授编写。

本书由厦门大学计算机与系统科学系李文清教授主审。参加审稿的还有福州大学、厦门大学的同志，他们提出了宝贵的意见，厦大计算中心林清溪高级工程师给予热情帮助，在此一并表示衷心感谢。

由于我们水平有限，疏漏和错误之处在所难免，希望广大读者提出宝贵意见，以便改正。

编者

1987年12月

内 容 提 要

本书内容包括误差与插值，数值积分与微分，逼近与拟合，线代数计算方法，求根，常微分方程与偏微分方程的数值解法以及近年来出现的应用数学的一些方法，给出用 FORTRAN 语言编写的、以过程的形式出现的程序，可供直接调用。

本书内容丰富，推理严密，文字简炼，通俗易懂，深入浅出，便于教学。可作为理、工、师范院校和各类大、中专科学校有关专业的教学参考书。也是广大从事计算应用的科技人员、工程设计人员和管理人员所需要的参考读物。

第一章　误差与插值

§1　误差的概念

一、功能

一个计算数学工作者，不但要学会建立方法编制程序，而且还要学会分析计算的结果。大多数数值方法给出的答案，仅是所要求的真解的某种近似。初始数据误差、截断误差和舍入误差对于计算结果的影响，都是计算机数值方法中要研究的问题。误差分析在数值运算中是一个很重要又很复杂的问题，怎样计算才可靠？利用误差基本概念与原理，提出分析误差的若干原则，讨论它们在计算过程中的传播和对计算结果的影响，并找出误差的界。误差的界和估计对于研究误差的渐近特性及改进算法的近似程度，具有重大的实际意义。

二、方法原理及其推导

加减乘除还有什么文章可做吗？给定了计算方案，当程序正确编出后，在电子计算机上实现解题运算，其结果该是足够精确吧！可是实际情况并非如此，请看下面例子：

例 1　要解线性方程组

$$\begin{cases} x_1+\dfrac{1}{2}x_2+\dfrac{1}{3}x_3=\dfrac{11}{6} \\ \dfrac{1}{2}x_1+\dfrac{1}{3}x_2+\dfrac{1}{4}x_3=\dfrac{13}{12} \\ \dfrac{1}{3}x_1+\dfrac{1}{4}x_2+\dfrac{1}{5}x_3=\dfrac{47}{60} \end{cases}$$

该方程组的真实解是 $x_1=x_2=x_3=1$，如果把系数舍入成两位数字的数，于是上述方程组变为

$$\begin{cases} x_1+0.5x_2+0.33x_3=1.8 \\ 0.50x_1+0.33x_2+0.25x_3=1.1 \\ 0.33x_1+0.25x_2+0.20x_3=0.78 \end{cases}$$

然后再求解，则得 $x_1=-6.222\cdots$，$x_2=38.25\cdots$，$x_3=-33.65\cdots$。算得结果可靠吗？若与真实解比较，已是面目全非了。

例 2　给定 $f(x)=10^7(1-\cos x)$，用四位数学用表，求 $f(2^0)$ 的近似值。

甲用下列步骤解题：由于 $\cos 2^0\approx 0.9994$，故

$$f(2^0)\approx 10^7(1-\cos 2^0)=6000.$$

乙用另法计算：由于 $f(x)=10^7(1-\cos x)=2\times 10^7(\sin x/2)^2$，而 $\sin 1^0\approx 0.0175$，故

$$f(2^0)\approx 2\times 10^7\sin^2 1^0=6125.$$

甲、乙共用一本数学用表，表的每个数均准确到小数后第四位，答案为什么不

样？谁的答案较正确呢？

要回答上述问题，在学完本节的若干概念和方法后就会明白的。

1、绝对误差与相对误差

一个物理量的真实值和我们算出的数值往往存在着差异，它们之差称为**误差**。一个数的绝对误差是它的精确值减去它的近似值。若某一数的精确值为 x，其近似值为 x^*，那么 x 与 x^* 之差

$$E(x)=x-x^*, \tag{1}$$

称为近似值 x^* 的**绝对误差**，简称**误差**。

精确值 x 一般是未知的，因而 $E(x)$ 不一定能求出，但往往可以估计出它的大小范围，亦即可以确定一正数 η，使

$$|E(x)|=|x-x^*|\leqslant\eta, \tag{2}$$

此时，η 称为 x^* 的**绝对误差界**。有时也用

$$x=x^*\pm\eta$$

表示 x^* 的精确度或精确值所在的范围。绝对误差是有量纲单位的。

由于对各种不同的问题计算所得的结果，其数值大小相差很大，只用绝对误差还不能说明数的近似程度。如甲打键盘时平均每百个符号错一个，乙打键盘时大约每五百个符号错一个，他们的误差都是错一个，但显然乙打键盘时要精确些。因此，除了要看绝对误差大小外，还必须顾及量本身，这就需要引入相对误差的概念。绝对误差与精确值之比，即

$$E_r(x)=\frac{E(x)}{x}=\frac{x-x^*}{x}, \tag{3}$$

称为 x^* 的**相对误差**。

由于精确值 x 一般不知道，实际计算常用下式作为 x^* 的相对误差

$$E_r(x)=\frac{x-x^*}{x^*}, \tag{4}$$

同样的，若能求出一正数 δ，使 $|E_r(x)|\leqslant\delta$，则 δ 称为 x^* 的**相对误差界**。相对误差是无量纲的数，通常用百分比表示，称为百分误差。

根据上述定义可知，甲打键盘时相对误差 $|E_r^*(x)|\leqslant\frac{1}{100}=1\%$，乙打键盘时的相对误差 $|E_r^*(x)|\leqslant\frac{1}{500}=0.2\%$，乙比甲打键盘时精确，所以，在分析误差时，相对误差更能刻划误差的特性。

2、舍入规则与有效数字

一个实数 x 总可以表示成无穷小数的形式

$$x=\pm(0.a_1a_2\cdots a_na_{n+1}\cdots)_{10}\times10^m \tag{5}$$

这里 $a_1,a_2,\cdots a_n,a_{n+1},\cdots$ 都是 0,1,2,…9 中的一个数字，m、n 均为整数。在实际计算时，只能取有限位数字而把其后的数字都去掉。为了方便起见，假定 $m=0$，$a_1\neq0$，现将（5）式分成两个部分

$$x=(0.\alpha_1\alpha_2\cdots\alpha_n)\times 10^m+(0.\underbrace{00\cdots0}_{n个}\alpha_{n+1}\cdots)\times 10^m$$

称前一部分为有效部分，后一部分为残余部分，为了尽可能减少误差，应根据残余部分的大小来调整有效部分的最后一个数字 α_n，一般**舍入规则**如下：

（1）若 $\alpha_{n+1}<5$，则 α_n 后面数字全部舍去。

（2）若 $\alpha_{n+1}>5$，则 α_n 进 1。

（3）若 $\alpha_{n+1}=5$，当 α_{n+1} 后面有非零数字则 α_n 进 1；当 α_{n+1} 后面无非零数字则视 α_n 奇偶性，采用“奇进偶不进”，即 α_n 是奇进 1，α_n 是偶不进。

为了简单起见，我们把通常所说的“四舍五入”抽象成数学语言，对于形如（5）式的 x，经舍入规则后，得到近似数为

$$x^*=\begin{cases}\pm(0.\alpha_1\alpha_2\cdots\alpha_n)_{10}\times 10^m, & 0\leqslant\alpha_{n+1}<5\\ \pm[(0.\alpha_1\alpha_2\cdots\alpha_n)_{10}+10^{-n}]\times 10^m, & 5\leqslant\alpha_{n+1}<10\end{cases}\qquad(6)$$

其中 $\alpha_1\neq 0$，显然，由舍入规则可知：近似数 x^* 的绝对误差不超过末位数的半个单位，即

$$|x-x^*|\leqslant\frac{1}{2}\times 10^{m-n}\qquad(7)$$

对于形如（6）的 x^*，若其绝对误差满足条件（7），便说 x^* 为具有 n **位有效数字的有效数**，而每一位数字 $\alpha_1,\alpha_2,\cdots\alpha_n$ 都叫 x^* 的**有效数字**。

例如从定义容易验证3.1416是 π 的具有 5 位有效数字的近似值。有效数字不但给出了近似数的大小，而且还给出了它的绝对误差界。如有效数字3587.64，0.158×10^{-2}，0.1580×10^{-2}的绝对误差界分别为 $\frac{1}{2}\times 10^{-2}$，$\frac{1}{2}\times 10^{-5}$，$\frac{1}{2}\times 10^{-6}$，特别要注意有效数字的指数记法，$0.158\times 10^{-2}$为三位有效数字，而$0.1580\times 10^{-2}$是四位有效数字。

例 3 为了求 $x=1-\cos 2^0$，用四位数学用表做工具，采用两种不同算法，现在比较这两种算法结果的误差。

甲算法：$x=1-b$，$b=\cos(2^0)$ 取 $b^*=0.9994$，故

$$|b-b^*|\leqslant\frac{1}{2}\times 10^{-4},$$

从而 $x_1^*=1-b^*=6\times 10^{-4}$，$|E(x_1)|=|x_1-x_1^*|=|b-b^*|\leqslant\frac{1}{2}\times 10^{-4}$，

$$|E_r^*(x_1)|=\frac{|E(x_1)|}{|x_1^*|}=\frac{|b-b^*|}{|x_1^*|}\leqslant\frac{\frac{1}{2}\times 10^{-4}}{6\times 10^{-4}}=\frac{1}{12},$$

乙算法：$x=2c^2$，$c=\sin(1^0)$ 取 $c^*=0.0175$，故

$$|c-c^*|\leqslant\frac{1}{2}\times 10^{-4},$$

从而 $x_2^*=2(C^*)^2=6.125\times 10^{-4}$，

$$E_r^*(x_2)=\frac{|x-x_2^*|}{|x_2^*|}=\frac{2|c^2-c^{*2}|}{|x_2^*|}=\frac{2|c-c^*|\times|c+c^*|}{|x_2^*|},$$

由于 $c=\sin\frac{\pi}{180}<\frac{\pi}{180}<\frac{3.1416}{180}<0.0175=c^*$，所以

$$|E_r^*(x_2)|<\frac{2|c-c^*|\cdot 2c^*}{2(c^*)^2}\leqslant\frac{2\times\frac{1}{2}\times10^{-4}}{0.0175}=\frac{1}{175}$$

于是 $|x_2-x_2^*|\leqslant|x_2^*|\cdot|E_r^*(x_2)|\leqslant6.125\times10^{-4}\times\frac{1}{175}=0.035\times10^{-4}$。

由 x_2 的相对误差界较小，可知 x_2^* 的有效数字较多。

今证 x_2^* 的绝对误差比 x_1^* 小。用反证法，设

$$|E(x_2)|<|E(x_1)|$$

不成立，则有 $|E(x_2)|\geqslant E(x_1)|$，于是

$$0.125\times10^{-4}=|x_1^*-x_2^*|\leqslant|x-x_2^*|+|x_1^*-x|\leqslant2|x-x_2^*|\leqslant0.07\times10^{-4}$$

这个不等式显然是错误的，它是由反证法的假设造成的，所以，

$$|E(x_2)|<|E(x_1)|$$

成立，这说明乙算法比甲算法好。

三、计算过程概要

1、数据误差在算术运算中的传播

设数值问题的解 y 与参量 $x_1,x_2,\cdots x_n$ 有关 $y=\varphi(x_1,x_2,\cdots x_n)$，给定参量的一组初始数据有误差，解也一定有误差，则当初始数据误差较小时，解的绝对误差为

$$E(y)=y-y^*=\varphi(x_1,x_2,\cdots x_n)-\varphi(x_1^*,x_2^*,\cdots x_n^*)$$

$$=\sum_{i=1}^{n}\frac{\partial\varphi(x_1,x_2,\cdots x_n)}{\partial x_i}\cdot E(x_i).$$

解的相对误差为

$$E_r(y)=\frac{E(y)}{y}\approx\sum_{i=1}^{n}\frac{\partial\varphi(x_1x_2,\cdots x_n)}{\partial x_i}\cdot\frac{x_i}{\varphi(x_1,x_2,\cdots x_n)}\cdot E_r(x_i).$$

这里系数 $\frac{\partial\varphi(x_1,x_2,\cdots x_n)}{\partial x_i}$ 或 $\frac{x_i}{\varphi(x_1,x_2,\cdots x_n)}\cdot\frac{\partial\varphi(x_1,x_2,\cdots x_n)}{\partial x_i}$ 表示结果误差相对于数据误差的放大或缩小倍数，它们绝对值大，则 $E(y)$ 或 $E_r(y)$ 可能很大，亦即数据 x_i 的微小变化可能引起结果的误差很大。

对于加减乘除及开平方这几种运算，从这两个公式可推出数据误差与计算结果误差之间的关系：

$$\begin{cases}E(x_1+x_2)\approx E(x_1)+E(x_2)\\ E_r(x_1+x_2)\approx\frac{x_1}{x_1+x_2}\cdot E_r(x_1)+\frac{x_2}{x_1+x_2}\cdot E_r(x_2)\end{cases}\tag{8}$$

$$\begin{cases}E(x_1\cdot x_2)\approx x_2\cdot E(x_1)+x_1\cdot E(x_2)\\ E_r(x_1\cdot x_2)\approx E_r(x_1)+E_r(x_2)\end{cases}\tag{9}$$

$$\begin{cases} E(x_1/x_2)\approx \dfrac{E(x_1)}{x_2}-\dfrac{x_1}{x_2^2}E(x_2) \\ E_r(x_1/x_2)\approx E_r(x_1)-E_r(x_2) \end{cases} \tag{10}$$

$$\begin{cases} E(\sqrt{x})\approx \dfrac{1}{2\sqrt{x}}\cdot E(x) \\ E_r(\sqrt{x})\approx \dfrac{1}{2}E_r(x) \end{cases} \tag{11}$$

当 x_1 与 x_2 同号时，

$$\frac{x_1}{x_1+x_2} \text{ 或 } \frac{x_2}{x_1+x_2} \tag{12}$$

的绝对值都在 0 和 1 之间，由（8）可得

$$|E(x_1+x_2)|\leqslant|E(x_1)|+|E(x_2)|,$$
$$|E_r(x_1+x_2)|\leqslant|E_r(x_1)|+|E_r(x_2)|,$$

这表明，此时加法结果的绝对误差或相对误差界都不超过相加各数的绝对或相对误差界之和，但如果 x_1 与 x_2 异号或（12）中数的绝对值可能大于 1，这个结论就不正确了，特别当 $x_1+x_2\approx 0$ 或（12）中数的绝对值可能很大，就可能有

$$|E_r(x+x_2)|\gg|E_r(x_1)|+|E_r(x_2)| \tag{13}$$

因此，（13）表示，大小接近的异号数相加或大小接近的同号数相减，会严重损失有效数字。从（9）可见，当乘数 x_1 或 x_2 的绝对值很大时，$|E(x_1\cdot x_2)|$ 可能很大，从（10）可见，当除数 x_2 接近于零时，$|E(x_1/x_2)|$ 可能很大，这表明，乘数绝对值很大或除数接近于零，可能会严重扩大误差，减少精确度，在实际计算中应当设法避免上述情况的发生。

2、舍入误差在算术运算中的传播

现代电子计算机常用浮点法表示数，如198.7表示成$10^3\times(0.1987)$，这里 3 称为阶码，0.1987称为尾数，在各种计算机上阶码和尾数的位数均有一定限制，用浮点法表示数的尾数其位数是固定的，并称为字长，假设计算机字长为 t，则任意数 x 按舍入规则表示成如下形式的**浮点数**：

$$f^l(x)=\begin{cases} \pm(0.\alpha_1\alpha_2\cdots\alpha_t)_\beta\times\beta^e & 0\leqslant\alpha_{t+1}<\beta/2 \\ \pm[(0.\alpha_1\alpha_2\cdots\alpha_t)_\beta+\beta^{-t}]\times\beta^e & \beta/2\leqslant\alpha_{t+1}<\beta \end{cases} \tag{14}$$

这里 $\alpha_1\neq 0$，β 称为数的基，$\alpha_1,\alpha_2,\cdots\alpha_t$ 都是 $0,1,2,\cdots\beta-1$ 中的一个数，数 e 称为阶码，β^e 称为定位，用来确定浮点数小数点的真实位置，而 $0.\alpha_1\alpha_2\cdots\alpha_t$ 称为尾数，$f^l(x)$ 称为规格化的浮点数，它作为 x 近似值，其绝对误差为

$$|x-f^l(x)|\leqslant\frac{1}{2}\beta^{e-t}$$

因此，$f^l(x)$ 的相对误差界为

$$\frac{|x-f^l(x)|}{|x|}\leqslant\frac{1}{2}\times\beta^{-t+1} \tag{15}$$

记 $\text{eps}=\dfrac{1}{2}\times\beta^{-t+1}$ 为 $f^l(x)$ 的相对误差界，称它为**计算机的精度**。

由于舍入方法不同，即按“舍入规则”及“只舍不入”，在计算机中进行浮点运算分为舍入运算和切断运算，舍入运算的计算机的精度为

$$eps=\begin{cases}2^{-t} & (\text{二进制的})\\ \frac{1}{2}\times 10^{-t+1} & (\text{十进制的})\end{cases}$$

切断运算的计算机的精度为

$$eps=\begin{cases}2^{-t+1}, & (\text{二进制的})\\ 10^{-t+1}, & (\text{十进制的})\end{cases}$$

如果把 ε 定义为 $f^{l}(x)$ 的相对误差，则由（15）可得

$$f^{l}(x)=(1+\varepsilon)\cdot x,\qquad |\varepsilon|\leqslant\frac{1}{2}\times\beta^{-t+1} \tag{16}$$

这一形式在分析舍入误差传播时是非常有用的。在计算机中，规格化浮点数在运算器中经过运算后结果仍用规格化浮点数存贮，这样，设 x_1,x_2 为规格化浮点数，则

$$\begin{aligned}&f^{l}(x_1+x_2)=(x_1+x_2)(1+\varepsilon_1),\\ &f^{l}(x_1-x_2)=(x_1-x_2)(1+\varepsilon_2),\\ &f^{l}(x_1\cdot x_2)=(x_1\cdot x_2)(1+\varepsilon_3),\\ &f^{l}(x_1/x_2)=(x_1/x_2)(1+\varepsilon_4)\end{aligned} \tag{17}$$

这里 ε_i $i=1,2,3,4$ 的上限由具体计算机运算器的字长，舍入规则等因素决定。

利用（17）可以分析舍入误差对计算结果的影响，如分析舍入误差对几个数相加的影响，由（17）有

$$\begin{aligned}f^{l}(f^{l}(x_1+x_2)+x_3)&=f^{l}((x_1+x_2)(1+\varepsilon_1)+x_3)\\ &=((x_1+x_2)(1+\varepsilon_1)+x_3)(1+\varepsilon_2)\\ &=(x_1+x_2+x_3)\left(1+\varepsilon_2+\frac{x_1+x_2}{x_1+x_2+x_3}\varepsilon_1(1+\varepsilon_2)\right)\\ &=(x_1+x_2+x_3)(1+\varepsilon)\end{aligned} \tag{18}$$

其中

$$\varepsilon=\varepsilon_2+\frac{x_1+x_2}{x_1+x_2+x_3}\varepsilon_1(1+\varepsilon_2) \tag{19}$$

类似地有

$$f^{l}(x_1+f^{l}(x_2+x_3))=(x_1+x_2+x_3)(1+\varepsilon') \tag{20}$$

其中

$$\varepsilon'=\varepsilon_2'+\frac{x_2+x_3}{x_1+x_2+x_3}\varepsilon_1'(1+\varepsilon_2') \tag{21}$$

比较（19）与（21）可知：在 $\varepsilon_1,\varepsilon_2,\varepsilon_1',\varepsilon_2'$ 大体相同的情况下，如果 $|x_1+x_2|<|x_2+x_3|$，则 $|\varepsilon|<|\varepsilon'|$，（18）和（20）说明此时若干数相加，采用绝对值较小者先加的算法，结果的相对误差界较小。如果 $|x_2+x_3|<|x_1+x_2|$，则同样得出绝对值较小者先加比较有利。

四、算法及其说明

算法是为使用电子计算机而提出的，解决数值问题，不仅需要算法，而且要求选用

和设计出好的算法，许多事例说明，如果算法选用不当，计算机的利用效率就得不到充分的发挥，容易看到，在电子计算机进行算术运算时，$a+b+c$ 可能不等于 $a+c+b$，$(a+b)c$ 可能不等于 $a\cdot c+b\cdot c$，发生误差的原因是计算机只能对有限位数进行运算。一个工程或科学计算问题，往往要运算千万次，如果我们对每一步的计算都去分析计算的误差，那是办不到的，也是不必要的，人们往往这样做，一方面针对普遍性问题，提出若干注意事项，另一方面分门别类研究误差的规律。我们在这里提出分析运算误差的若干原则，有助于鉴别计算结果的可靠性，并防止误差危害的现象产生。

1、注意算法步骤简化，减少运算次数

现以多项式求值问题为例，说明减少运算次数的重要性和可能性。例如，计算多项式

$$p_n(x)=\sum_{k=0}^{n}a_kx^k$$

的值。若直接逐项求和运算，算 a_kx^k 要 k 次乘法，一共要 $\frac{1}{2}n(n+1)$ 次乘法和 n 次加法，但若按下列递推关系式

$$\begin{cases}u_0=a_n\\ u_k=u_{k-1}\cdot x+a_{n-k}\end{cases}\qquad(22)$$

对 $k=1,2,\cdots$ 直到 n，反复执行算式（22），最后得到的 u_n 就是所求的结果。只要 n 次乘法和 n 次加法，而且逻辑结构简单，算式（22）就是著名的秦九韶算法。一般说来，在一个工程问题中，通过算法步骤的简化不仅能减少运算次数，提高计算速度，而且还能简化逻辑结构，减少误差的积累。

2、防止两数相似进行减法运算

两个正数之差 $z=x-y$ 的相对误差为

$$E_r(z)=\frac{E(x)-E(y)}{x-y},$$

若两个数 x 和 y 很接近，它们差的相对误差就很大，这是由于 x^* 和 y^* 的前几位相同的有效数字在它们之差 x^*-y^* 内全被减掉了，所以，遇到这种情况，在 x^* 和 y^* 内应多保留几位有效数字或者对公式进行处理，避免减法。特别要避免再用这个差作为除数。

例如求 $z=\sqrt{a+1}-\sqrt{a}$ 之值。当 $a=1000$，以四位数字计算 $\sqrt{a+1}=31.64$，$\sqrt{a}=31.62$，两者直接相减得 $z=0.02$，这个结果只有一位有效数字（z 的实际值为 0.01580），但若对公式处理成

$$z=\sqrt{a+1}-\sqrt{a}=\frac{1}{\sqrt{a+1}+\sqrt{a}},$$

则按后一公式求得的结果 $z=0.01581$ 有三位有效数字，可见改变计算公式能避免相近两数相减引起的有效数字损失，而得出比较精确的结果。

3、设法控制误差的传播和积累

怎样减少计算中误差的积累呢？首先要注意浮点运算的特点，作为一个例子说明误差在以不同方式进行算术运算时积累的情况，我们假设用三位浮点数切断运算，计算

$$y=\sum_{i=1}^{10}\frac{1}{i}=1+\frac{1}{2}+\frac{1}{3}+\frac{1}{4}+\frac{1}{5}+\frac{1}{6}+\frac{1}{7}+\frac{1}{8}+\frac{1}{9}+\frac{1}{10}$$

之值，如果用三位数字表示这十个数 $1/i$ $i=1,2,\cdots10$，我们按照从左到右的次序计算可得 $y=2.91$，而按照从右到左的次序计算可得 $y=2.92$（y 的实际值为2.927），前者误差为0.017，后者误差为0.007，比前者的误差减少了一半以上，这是由于从小的数字加起，切断次数较少，从而减少误差的积累。由此可见，对于符号相同诸数相加时，如果是按绝对值从小到大的次序进行的则绝对误差就较小。

其次，算法的递推性在数值计算中有着重要地位，但多次递推必须注意误差的传播。用不同的算法解决同一问题，所得结果的精度可能大不相同，这是由于初始数据的误差或计算中某一步产生的舍入误差，在计算过程中的传播常因算法而异，于是就产生了算法的数值稳定性问题。

例 4 计算积分

$$I_n=\int_0^1\frac{x^n}{x+5}dx,\quad n=0,1,2,\cdots$$

的近似值。

解： 由 $I_n+5I_{n-1}=\int_0^1\frac{x^n+5x^{n-1}}{x+5}dx=\int_0^1x^{n-1}dx=\frac{1}{n}$，可得递推关系式

$$I_n=\frac{1}{n}-5I_{n-1},\quad n=1,2,\cdots \tag{23}$$

显然，$I_0=\int_0^1\frac{dx}{x+5}=\ln(1.2)\approx0.18232155$，由 I_0 出发，利用（23）式陆续可得表 1—1 中左边那一行。这行结果显然是错误的，因为由 I_n 的表达式可知，对一切自然数 n，I_n 是恒正的，但算出的 I_{10}，I_{12} 等却是负的，为什么会这样呢？原因在于误差所引起的危害。

现在来作误差分析，设 I_0 的近似值 I_0^* 有误差 $E_0=I_0-I_0^*$，并设以后的计算都没有误差，则按（23）计算得到的不是真的 I_n，而是受 E_0 影响了的近似值 I_n^*，即

$$I_n^*=-5I_{n-1}^*+1/n,$$

设 $En=I_n-I_n^*$，得出 $E_n=-5E_{n-1}$，$n=1,2,3,\cdots$ 从而得出

$$E_n=(-5)^nE_0$$

这表明：如果 I_0^* 有误差 E_0，则计算出的 I_n^* 的误差为 E_0 的$(-5)^n$倍。如$|E_0|=10^{-8}$，那么 $|E_9|=5^9|E_0|=\frac{10}{512}>0.01$，由于 $I_9^*=0.03\cdots$，这说明 I_9^* 连一位有效数字都没有了。如果改用递推公式

$$I_{k-1}=\frac{1}{5\cdot k}-\frac{I_k}{5},\quad k=n,n-1,\cdots1 \tag{24}$$

表 1—1

n	用 I_0 及（23）算得的结果	用 I_{14}^* 及（24）算得的结果
0	0.18232155	0.18232155
1	0.08839225	0.08839222
2	0.05803875	0.05803892
3	0.04313958	0.04313873
4	0.03430208	0.03430633
5	0.02848958	0.02846835
6	0.02421875	0.02432491
7	0.02176339	0.02123260
8	0.01618305	0.01883699
9	0.03019588	0.01692617
10	−0.05097941	0.01536914
11	0.34580612	0.01406339
12	−0.64569726	0.01301636
13	8.30540938	0.01184127
14	−41.45561831	0.01222222

取 I_n^* 为初始近似值，同样假定计算过程中除 I_n 的误差 $E_n=I_n-I_n^*$ 外，没有别的误差，从而

$$E_{n-k}=E_n/(-5)^k \qquad k=1,2,\cdots n$$

可见误差是逐步衰减的。由于

$$\frac{1}{6(n+1)}=\int_0^1\frac{x^n}{6}dx<\int_0^1\frac{x^n}{x+5}dx<\int_0^1\frac{x^n}{5}dx=\frac{1}{5(n+1)},$$

所以，

$$\frac{1}{6\times 15}<I_{14}<\frac{1}{5\times 15}$$

因而可取 $I_{14}^*=\frac{1}{2}\left(\frac{1}{6\times 15}+\frac{1}{5\times 15}\right)=0.01222222$ 为初始近似值，再按（24）进行计算，结果列在表1—1中右边一列，这样算出的 I_0 与 $\ln(1.2)$ 之值符合得很好。我们用（23）计算时，尽管 I_0 有八位准确数字，由于误差逐渐扩大，结果越来越不可靠。用（24）计算，虽然 I_{14}^* 很粗糙，但求出的 I_0 有八位准确数字，这是因为误差逐步衰减的关系。

不同算法，对初始数据的误差（或计算过程中某一步的舍入误差）的传播一般不同。一个算法，如果初始数据的误差对计算结果的影响较小，便说这个算法具有较好的数值稳定性。反之，如果初始数据的误差对计算结果的影响很大，便说这个算法的数值稳定性不好，解决数值问题，要选用稳定性较好的算法。

4、保护重要的物理力学的参数

在一个冗长的计算机程序中尽量避免有效数字的损失，以保证计算的精确度是完全必要的，在编制程序时要注意某个重要的物理量在计算时被“吃掉”。例如，考察物体在阻尼介质中的运动时，阻尼系数 K 是个重要的物理参量，若在动力学方程的离散化过程中将 K 置于与一个很大量级的数 a 的加减运算中，那么 K 就被 a 所“吃掉”，引起计算结果的失真。又如若对 A、B、C 三数进行加法运算，$A=10^{12}$，$B=10$，$C\approx -A$，如果按 $(A+B)+C$ 次序来编程序，那么在计算机上（该机只能将数表达到小数后第八位）A“吃掉”B，且 A、C 互相抵消，其结果接近于零。但若按 $(A+C)+B$ 编程序其结果接近于10。所以，在许多大数量级的数相加，它们符号是变化的，且结果又是一个很小的数，此时就会发生有效数字的损失，如果我们事先分析计算方案的数量级，编程序时加以合理安排，注意运算次序，重要的物理参量就不致于在计算中被“吃掉”，以达到避免有效数字的损失。

五、应用

例题 1　设各数由舍入规则得到，计算

$$2.34+0.0215-3.21\times 0.243$$

之值，并估计结果的有效数字位数。

解：令　$x=2.34+0.0215-3.21\times 0.243=1.58147$

由（8）及（9）式可知：

$$\begin{aligned}|E(x)| &\leqslant |E(2.34)|+|E(0.0215)|+3.21\times|E(0.243)|+0.243\times|E(3.21)| \\ &\leqslant 0.005+0.00005+3.21\times 0.0005+0.243\times 0.005=0.00787 \\ &\leqslant \frac{1}{2}\times 10^{-1}\end{aligned}$$

再根据（7）式可知，结果只有两位有效数字，故取 $x\approx 1.6$。

例题 2　已知 $\sqrt{168}=12.961$ 有 5 位有效数字，今试求 $x^2-26x+1=0$ 的两根 $x_1=13+\sqrt{168}$ 及 $x_2=13-\sqrt{168}$，试分析按 $x_{1,2}=13\pm\sqrt{168}$ 求出的根的绝对误差界和相对误差界，怎样去求 x_2 使其相对误差较小？

解：由二次方程求根公式得

$$x_1=13+\sqrt{168},\qquad x_2=13-\sqrt{168},$$

为方程的两个根，已知 $\sqrt{168}=12.961$，因此

$$|\sqrt{168}-12.961|\leqslant 0.0005$$

定义 $x_1^*=25.91$，$x_2^*=0.039$，这是用12.91代替公式中 $\sqrt{168}$ 所得的结果，于是

$$|E(x_1)|=|E(x_2)|\leqslant 0.0005$$

$$|E_r^*(x_1)|\leqslant \frac{0.0005}{x_1^*}\leqslant \frac{0.0005}{25.9605}\approx 1.9\times 10^{-5},$$

$$|E_r^*(x_2)|\leqslant \frac{0.0005}{x_2^*}\leqslant \frac{0.0005}{0.0385}\approx 1.3\times 10^{-2},$$

可见，尽管 x_2^* 中每个数十分准确，但它有较大的相对误差，在计算 x_2^* 时，12.961 的有效数字被丢失了。

现在用另一算法，基于下式定义一个新的 x_2^*：

$$x_2^* = 13-\sqrt{128} = \frac{1}{13+\sqrt{168}}$$

$$\frac{1}{13+\sqrt{168}} \approx \frac{1}{25.961} \approx 0.03851932 \equiv x_2^*$$

这是 x_2 的一个较好的近似值。因为

$$|E(x_2)| = |x_2 - x_2^*| \leqslant \left|x_2 - \frac{1}{25.961}\right| + \left|\frac{1}{25.961} - 0.03851932\right|$$

$$\leqslant \left|E\left(\frac{1}{25.961}\right)\right| + 5\times 10^{-9}$$

故

$$|E_r^*(x_2)| \leqslant \left|E_r^*\left(\frac{1}{25.961}\right)\right| + \frac{5\times 10^{-9}}{0.03859132} \leqslant 1.9\times 10^{-5} + 1.3\times 10^{-7}$$

可见：$|E_r^*(x_2)| \leqslant 1.91\times 10^{-5}$ 与 x_1^* 基本上相同。

例题 3 计算积分

$$I_n = \int_0^1 x^n e^{x-1} dx, \quad n = 1, 2, \cdots 20$$

的近似值。

解：当 $n=1$ 时，$I_1 = \int_0^1 x e^{x-1} dx \approx 0.367879$，当 $n=2,3,\cdots 20$ 时由分部积分法则可得

$$I_n = 1 - n\cdot I_{n-1}$$

于是 $I_2 = 1-2I_1 = 1-2(0.367879) = 0.264262$，$\cdots\cdots$ $I_9 = 1-9I_8 = -0.068480$，由 I_n 的表达式可知：对一切自然数 $I_n > 0$，而上面得出的 $I_9 < 0$ 是错误的，原因正是由于初始数据的误差 I_1 值因舍入（取 6 位数字）而产生的舍入误差（其大小约为 $\varepsilon = 4.41^2 \times 10^{-7}$），其后的计算都是精确的，仅这个舍入误差在计算过程中传播下去，从而影响了后面计算的结果。误差在计算中按下面的规律传播：

$$I_2 = 1-2(I_1+\varepsilon) = 1-2I_1-2!\varepsilon,$$

$$I_3 = 1-3[1-2I_1-2!\varepsilon] = 1-3(1-2I_1)+3!\varepsilon,$$

$$I_4 = 1-4[1-3(1-2I_1)+3!\varepsilon]$$

$$= 1-4[1-3(1-2I_1)]-4!\varepsilon,$$

$$\cdots\cdots\cdots\cdots$$

可见，I_1 有误差 ε，则 I_n 就有 ε 的 $n!$ 倍误差，所以，计算到 I_9 所产生的误差约为

$$9!\varepsilon = 9!\times 4.412\times 10^{-7} \approx 0.1601$$

于是 I_9 取三位数字的精确值为 0.0916。

如果改用递推关系式为

$$I_{n-1} = \frac{1-I_n}{n}, \quad n = 20, 19, \cdots, 3, 2$$

则从后向前计算过程中，求 I_{n-1} 时，I_n 的误差影响较小，I_n 中的误差减少为原来的 $1/n$，所以，若取 n 足够大，误差逐步减少，其影响越来越小。为了得到初始值，考虑关系式

$$I_n=\int_0^1 x^n\cdot e^{x-1}dx\leqslant\int_0^1 x^n dx=\frac{1}{n+1}$$

当 $n\to\infty$ 时，$I_n\to 0$，如取 $I_{20}=0$ 作为初始值，其误差小于 1/21，在计算到 I_{15} 时，误差已减少到不超过 4×10^{-8}，计算结果 $I_9=0.09162$ 已有足够精确，这也说明后一种算法有较好的数值稳定性。

§2 线 性 插 值

一、功能

插值法是计算机数据处理的重要方法。它的重要应用在于提供一些数学工具，这些工具常用于推导逼近论、数值积分以及微分方程数值解各领域中的方法，同时又提供处理列表函数的方法。例如，几乎人人都熟悉中学代数里对数表中的线性插值。所谓线性插值，通俗地说，就是利用两对值 (x_0,y_0) 及 (x_1,y_1) 求得 $y=f(x)$ 的近似值，由于线性插值仅仅利用两个插值节点 x_i 上的信息，精度自然较低，为了改善精度，我们从给定 N 个插值节点 x_i 中选取最靠近插值点 x 的相邻二个插值节点，用线性插值公式对 M 个一元列表函数进行成组插值。从插值余项公式表明，选取的插值节点 x_i 离插值点 x 越近，误差就越小，因而插值效果也就越好。

二、方法原理及其推导

插值法的基本思想是什么呢？就是根据给定函数 $f(x)$ 在一些离散点处的值，想办法去构造某个简单、性质较优的函数 $y=\varphi(x)$ 作为 $f(x)$ 的近似表达式，然后计算 $\varphi(x)$ 的值以得 $f(x)$ 的近似值。例如，给出函数 $y=f(x)$ 在区间 $[a,b]$ 上 $n+1$ 个互不相同的点 x_i 处的函数值 $y_i=f(x_i)(i=0,1,2,\cdots n)$ 或称为给出的列表函数：

x_i	x_0	x_1	x_2	……	x_{n-1}	x_n
y_i	y_0	y_1	y_2	……	y_{n-1}	y_n

要求构造一个次数不超过 n 的代数多项式

$$P_n(x)=a_0+a_1x+a_2x^2+\cdots+x_nx^n$$

使 $P_n(x)$ 在点 x_i 处满足条件

$$P_n(x_j)=f(x_i)=y_i,\quad i=0,1,2,\cdots n \tag{25}$$

其中系数 a_i 待定的，这个问题称为 n **次代数插值**。称 $x_0,x_1,x_2,\cdots x_n$ 为插值节点，包含插值节点的区间 $[a,b]$ 称为插值区间，称 $P_n(x)$ 是函数 $f(x)$ 的插值多项式，称关系式（25）为插值条件。在区间 $[a、b]$ 上用 $y=P_n(x)$ 近似 $y=f(x)$，除了插值节点 x_i 处 $f(x_i)=P_n(x_i)$ 外，在 $[a、b]$ 的其它点 x 处都可能有误差，记 $R_n(x)=f(x)-P_n(x)$ 称为插值多项式的余项，它表示用 $P_n(x)$ 近似 $f(x)$ 的截断误差的大小，一般说，$\max\limits_{a<x<b}|R_n(x)|$ 越小就越近似。

现研究两点插值的简单情形。