理工科研究生通用教材

湖南科技大学研究生课程建设项目资助

数值分析

熊之光　吴晓勤　陈荣华/编

天津大学出版社
TIANJIN UNIVERSITY PRESS

理工科研究生通用教材
湖南科技大学研究生课程建设项目资助

数值分析

熊之光　吴晓勤　陈荣华　编

天津大学出版社
TIANJIN UNIVERSITY PRESS

内 容 提 要

本书是为理工科大学各专业研究生学位课程“数值分析”编写的教材，其内容包括多项式插值法、函数逼近与数据拟合法、数值积分与数值微分、线性方程组的数值解法、非线性方程(组)的数值解法、矩阵特征值问题的数值解法、常微分方程初边值问题的数值解法、椭圆型方程的有限差分法、有限元方法和插值系数有限元法简介，每章附有习题和上机实验题．全书叙述严谨，脉络分明，深入浅出，便于教学．

本书可作为理工科大学各专业研究生和数学专业大学生的数值分析课程教材或教学参考书，也可作为从事科学与工程计算的科技工作者的参考用书．

图书在版编目(CIP)数据

数值分析/熊之光，吴晓勤，陈荣华编．—天津：天津大学出版社，2012.8

理工科研究生通用教材　湖南科技大学研究生课程建设项目资助

ISBN 978-7-5618-4464-9

Ⅰ.①数…　Ⅱ.①熊…②吴…③陈…　Ⅲ.①数值分析—研究生—教材　Ⅳ.①0241

中国版本图书馆 CIP 数据核字(2012)第 205991 号

出版发行	天津大学出版社
出 版 人	杨欢
地　　址	天津市卫津路 92 号天津大学内(邮编：300072)
电　　话	发行部：022-27403647
网　　址	publish.tju.edu.cn
印　　刷	廊坊市长虹印刷有限公司
经　　销	全国各地新华书店
开　　本	185mm×260mm
印　　张	17.75
字　　数	443 千
版　　次	2012 年 9 月第 1 版
印　　次	2012 年 9 月第 1 次
定　　价	39.00 元

前　言

随着计算机软硬件技术的不断提高，计算科学得以迅速发展并在其他科学技术问题中广泛应用，从而导致了数值计算继理论方法和实验方法之后成为科学研究的第三种基本手段．数值分析及其科学计算已经越来越受到数学、计算机科学以及各种工程技术领域的高度重视．数值分析课程现在也已经成为高等院校理工科专业的一门重要基础课程．

本书是作者根据理工科专业研究生教学的要求而为数值分析课程编写的教材．本书的主要内容为数值计算的基本理论与基本方法，主要包括多项式插值法、函数逼近与数据拟合法、数值积分与数值微分、线性方程组的数值解法、非线性方程(组)的数值解法、矩阵特征值问题的数值解法、常微分方程初边值问题的数值解法、椭圆型方程的有限差分法、有限元方法和插值系数有限元法简介．本书在体系结构和内容取材上，致力于取材精练、由浅入深、衔接顺畅，尽量综合国内外同类书的优点；在理论方法的分析和内容表述上，力求叙述严谨、脉络分明、深入浅出，以便于教学．为便于读者及时掌握本书的基本内容，各章给出了适当的习题和上机实验题．

我们希望通过本书的学习能使读者掌握数值计算的基本理论和基本方法，提高数学素养，提高应用计算机进行科学与工程计算的能力，提高应用数学与计算机解决实际问题的能力．本书可作为理工科专业研究生和数学专业大学生的数值分析课程教材或教学参考书，也可作为从事科学与工程计算的科技工作者的参考用书．

本书内容涉及的范围和深度具有一定的弹性，教学时可根据学生的实际情况选用．讲授本书的主要内容需60学时左右，讲授完全部内容需80学时左右．本书的出版得到“湖南科技大学研究生课程建设项目”资助，作者在此深表感谢．天津大学出版社常红女士在本书的编辑过程中给予了大力帮助，在此一并致谢。

本书的选材和内容的叙述可能会有不当甚至错误之处，诚请读者和同行们批评指正．

熊之光、吴晓勤、陈荣华

2012年5月22日于湘潭

目　录

第 1 章　绪论 …… 1
1.1　引论 …… 1
1.2　数值分析的研究内容和特点 …… 2
1.3　数的浮点表示及浮点运算 …… 2
1.4　误差及有效数字 …… 3
1.5　避免误差危害的若干原则 …… 6
1.6　MATLAB 简介 …… 10
习题 1 …… 12
上机实验题 1 …… 13
第 2 章　多项式插值法 …… 15
2.1　多项式插值问题的提法 …… 15
2.2　Lagrange 插值法 …… 16
2.3　Newton 插值法 …… 19
2.4　差分与等距节点的 Newton 插值法 …… 23
2.5　逐次线性插值法 …… 25
2.6　Hermite 插值法 …… 27
2.7　分段多项式插值法 …… 30
2.8　三次样条插值法 …… 33
习题 2 …… 40
上机实验题 2 …… 41
第 3 章　函数逼近与数据拟合法 …… 42
3.1　正交多项式 …… 43
3.2　最佳平方逼近 …… 47
3.3　最佳一致逼近 …… 50
3.4　离散数据的曲线拟合 …… 55
习题 3 …… 62
上机实验题 3 …… 63
第 4 章　数值积分与数值微分 …… 65
4.1　数值积分的基本概念 …… 65
4.2　Newton-Cotes 公式 …… 67
4.3　复化求积公式 …… 70
4.4　变步长求积公式及加速收敛技术 …… 73
4.5　Gauss 求积公式 …… 78
4.6　数值微分 …… 83
习题 4 …… 85

上机实验题 4 …… 86
第 5 章　线性方程组的数值解法 …… 87
5.1　Gauss 消去法 …… 87
5.2　矩阵的三角分解 …… 93
5.3　向量和矩阵的范数 …… 103
5.4　线性方程组的性态和解的误差分析 …… 106
5.5　解线性方程组的迭代法 …… 108
5.6　迭代法的收敛性分析 …… 114
习题 5 …… 119
上机实验题 5 …… 120
第 6 章　非线性方程(组)的数值解法 …… 122
6.1　二分法 …… 122
6.2　迭代法 …… 123
6.3　Newton 法 …… 130
6.4　弦截法和抛物线法 …… 135
6.5　非线性方程组的数值解法 …… 136
习题 6 …… 142
上机实验题 6 …… 143
第 7 章　矩阵特征值问题的数值解法 …… 145
7.1　特征值问题的性质与估计 …… 145
7.2　幂法与反幂法 …… 147
7.3　旋转变换和 Jacobi 方法 …… 154
7.4　QR 方法 …… 160
习题 7 …… 165
上机实验题 7 …… 166
第 8 章　常微分方程初边值问题的数值解法 …… 168
8.1　引论 …… 168
8.2　单步法和 Runge-Kutta 法 …… 174
8.3　单步法的收敛性与稳定性 …… 181
8.4　线性多步法 …… 183
8.5　绝对稳定性与绝对稳定域 …… 192
8.6　方程组和刚性问题 …… 196
习题 8 …… 199
上机实验题 8 …… 200
第 9 章　椭圆型方程的有限差分法 …… 202
9.1　差分逼近的基本概念 …… 202
9.2　一维差分格式 …… 205
9.3　矩形网差分格式 …… 211
9.4　三角网差分格式 …… 215

习题 9 …… 218
上机实验题 9 …… 219
第 10 章　有限元方法 …… 221
10.1　二次函数的极值问题 …… 221
10.2　Sobolev 空间初步 …… 223
10.3　两点边值问题 …… 228
10.4　二阶椭圆边值问题 …… 232
10.5　Ritz-Galerkin 方法 …… 237
10.6　两点边值问题的有限元法 …… 241
10.7　一维高次元 …… 247
10.8　矩形元 …… 251
10.9　三角形元 …… 255
10.10　二阶椭圆方程的有限元法 …… 258
习题 10 …… 265
上机实验题 10 …… 266
第 11 章　插值系数有限元法简介 …… 267
11.1　引论 …… 267
11.2　两类基本正交展开 …… 269
11.3　非线性常微分方程插值系数有限元法 …… 270
11.4　半线性椭圆问题插值系数有限元法 …… 272
上机实验题 11 …… 275
参考文献 …… 276

第1章 绪 论

1.1 引论

当今,科学计算和传统的科学方法——理论和实验,相并列成为第三种科学研究方法.随着计算机软硬件的快速发展,特别是计算机图形学的发展,科学计算可视化的程度越来越高,计算机正深刻地影响着人类的行为.

数值分析也称计算方法(或称数值计算方法)是科学计算的一门基础课程,它研究用计算机求解各种数学问题的数值计算方法及其理论与软件实现.这门学科的主要内容是研究和寻求适合于计算机计算的各种数值问题的算法,即可把数值分析定义为研究连续数学问题的算法.数值分析的核心是为解决某类问题而设计和分析算法.

一般地讲,算法是指解决问题的步骤,由一些基本运算及运算顺序的规定构成的一个完整的解题步骤称为算法.算法的严格定义是由 Turing 给出的,他以自己抽象的计算装置 Turing 机为基础给出了算法的精确定义,规定了算法中允许的基本操作,他清醒地认识到计算的本质,他的 Turing 机和现代的数字计算机在原理和功能上没有本质的区别.

一个可行的、有效的算法必须满足下列基本要求.

(1)符合计算机的要求.我们知道计算机只能对有限位数进行加、减、乘、除与逻辑运算,因此对给定的数学问题提出的数值计算方法也只能包含上述五种运算.对于具体算法还要考虑计算机的内存大小、机器字长、运算速度等.有的算法从纯数学的观点看不够严格和完善,但通过实际计算、对比分析等手段证明是行之有效的,也常被采用.特别地,随着计算机的飞速发展,一些算法在老的计算机上无法实现,而在新型计算机上却可以实现.总之,对于给定的数学模型,在构造算法时要面向计算机,符合计算机的要求.

(2)理论上收敛、稳定,实际计算精度高.由于计算机只能近似地表示实数,不论计算机中的数是定点表示,还是浮点表示,它所表示的数的位数都是有限的,且任一算法只能在有限的时间内通过有限次运算来完成.这说明用计算机运算得到的结果都是近似的,因此需要考虑算法的精度问题.在理论上还要研究用计算机运算得到的结果是否收敛到实际问题的解.此外,还要考虑算法的数值稳定性问题,即在执行算法的过程中,舍入误差的积累不影响产生可靠结果.

(3)计算复杂性尽可能小.从实际需要出发,还需要考虑计算量的大小,即所谓计算复杂性问题.计算复杂性是由以下两个因素决定的:使用中央处理器(CPU)的时间,这主要由四则运算的次数决定;占用内存储器的空间,这主要由使用的数据量决定.有时也称其为时间与空间的复杂性,简称计算复杂性.例如,解线性方程组 $\boldsymbol{Ax}=\boldsymbol{b}$,若 $\det \boldsymbol{A}\neq 0$,则可用 Cramer 法则来解.设 $\boldsymbol{A}$ 为 20 阶矩阵,计算一个 20 阶行列式需要的乘法运算量为 $19\times 20!$,需计算 21 个 20 阶的行列式,总的乘法运算量为 $21\times 19\times 20!\approx 9.71\times 10^{20}$,若用 10 亿次/秒的计算机来运算,解 20 阶的方程组所需乘法运算的时间为 $9.71\times 10^{20}\div(3.15\times 10^{16})\approx$

3.08×10^4(年),即三万零八百年,显然这个运算时间在实际中是不可接受的. 而在实际问题中,例如大型水利工程、天气预报等,需要解的大型方程组的阶数一般都远远大于20,若用上述方法显然无法解决. 这个例子说明解线性方程组的Cramer法则在理论上虽然可行,但在实际应用中却不可行. 有人可能说,随着计算机的发展,运算速度提高、内存增大以及新结构计算机的涌现,以前认为过于复杂而不能求解的问题将会得到解决. 但是,不论计算机如何发展,使用计算机的代价,即计算复杂性,都是要考虑的.

对于给定的数学模型,可能有多种算法,应通过计算机进行数值实验,进行分析、比较来选定算法. 对新提出的算法,有的在理论上虽然还未证明其收敛性,但可以从具体实验中发现其规律,为理论证明提供线索.

总之,对于给定的数学问题所提出的可行的、有效的算法应该符合计算机的要求,即在理论上收敛、稳定,在实际计算中精确度高,计算复杂性小,能通过实验验证的数值方法.

1.2 数值分析的研究内容和特点

数值分析课程的内容包括:函数的插值与逼近、数值积分和数值微分、数值线性代数、微分方程的数值解法等. 本课程的特点是既有纯数学高度抽象性与严密科学性的特点,又有应用的广泛性与实际实验的高度技术性的特点,是一门与使用计算机密切结合的实用性很强的数学课程. 学习时要注意掌握方法的基本原理和思想,注意方法的处理技巧与计算机的结合,重要的是要学习如何用数值分析提供的算法去解决实际中碰到的问题,这就不仅要求在学习中要做一定数量的理论分析和计算练习,更应该做大量的数值实验.

现在,数值分析的实践发生了巨大的变化,几乎很少有人再去写那些繁杂的代码和程序,人们更多的是用像MATLAB以及类似MATHEMATICA、Octave、Gauss那样的软件包. 同时人们所解决问题的规模也越来越大,从这方面说,还要感谢那些高级的计算机硬件以及强有力的软件. 但即使这样,也还要进一步了解算法背后的基本思想以及怎样使这些算法在合理时间内收敛,其中包括选择适当的初值、各种方法的结合及调整算法的参数等. 因此,我们推荐用MATLAB做数值实验,在本章最后一节简要介绍了MATLAB,也可找一本MATLAB的参考书来学习更多的MATLAB知识.

1.3 数的浮点表示及浮点运算

对于任何一个非零实数,用 $x=\pm10^k\times0.a_1a_2a_3\cdots a_n\cdots$ 表示,这就是规格化的十进制科学记数法. 在现代计算机中使用的是二进制(或其变形的十六进制)的数系.

考虑规格化的二进制浮点记数法,由于计算机的字长总是有限的,所以计算机所能表示的数系,并不是连续的集合,而是一个特殊的离散集合,这个集合的数就称为机器数,其二进制规格化浮点表示为

$$\pm2^k\times0.\beta_1\beta_2\cdots\beta_t, \tag{1.3.1}$$

其中整数 k 称为阶码. 用二进制表示 k 有

$$k=\alpha_1\alpha_2\cdots\alpha_s. \tag{1.3.2}$$

式(1.3.2)中,$\alpha_j=0$ 或 $1(j=1,\cdots,s)$,α_1 称为阶符,表示数 k 的符合,取 0 为正数,取 1 为负数;s 是阶码的位数. 小数 $0.\beta_1\beta_2\cdots\beta_t$ 称为尾数,其中 $\beta_1=1$,$\beta_j=0$ 或 $1(j=2,\cdots,t)$,t 是尾数的位数. 再增加一位表示数的符合,机器表示规格化浮点数总共 $s+t+1$ 位,s 和 t 与具体的机器有关.

对于任一非零实数 x,转化为二进制时,将对 t 位后面的数作舍入处理,使得尾数为 t 位,因此一般都有舍入误差.

下面我们看两个数在计算机中是如何运算的. 首先将两个数表示为浮点数,然后对两个浮点数按以下规则作加减乘除运算.

(1)**加减法** 首先比较两数的阶码,将阶码较小的尾数向右移位,每移 1 位阶码加 1,直至两数的阶码相等为止,此过程称为对阶;再将移位后的尾数多于计算机字长的部分进行四舍五入,之后对尾数进行加减运算;最后将尾数写成规格化形式.

(2)**乘法** 阶码相加,尾数相乘,得两倍字长尾数,最后将乘积的尾数舍入成规格化形式,并冠以乘积的符合(同号积为正,异号积为负).

(3)**除法** 阶码相减,将被除数尾数扩大为双倍字长进行除法,得两倍字长的尾数,最后舍入成规格化形式,并冠以商的符合(同号商为正,异号商为负).

我们以十进制数为例说明浮点数的加减乘除法运算.

例 1.3.1 设有两个实数 $x=1.623$,$y=0.189$,假设机器浮点数中阶码是 3 位,尾数也是 3 位,试用机器计算 $z_1=\frac{4}{3}(x+y)$,$z_2=4(\frac{x+y}{3})$.

解 分别记 x,y 规格化浮点数为 $fl(x)$,$fl(y)$,则

$$fl(x)=0.162\times10^1,fl(y)=0.189\times10^0.$$

有

$$\begin{aligned}fl(x+y)&=fl(x)+fl(y)=0.162\times10^1+0.189\times10^0\\&=0.162\times10^1+0.019\times10^1=0.181\times10^1,\end{aligned}$$

$$\begin{aligned}fl(z_1)&=fl(\frac{4}{3}(x+y))=fl(\frac{4}{3})fl(x+y)\\&=0.133\times10^1\times0.181\times10^1=0.133\times0.181\times10^2=0.024\,073\times10^2\\&=0.240\times10^1,\end{aligned}$$

$$\begin{aligned}fl(z_2)&=fl(4(\frac{x+y}{3}))=fl(4)fl(\frac{x+y}{3})\\&=0.400\times10^1\times[(0.181\times10^1)\div(0.300\times10^1)]\\&=(0.400\times10^1)\times(0.603\times10^0)=(0.400\times0.603)\times10^1\\&=0.241\,200\times10^1=0.241\times10^1.\end{aligned}$$

此例说明两种算法得到的答案不同,而且没有一个等于精确值 2.416. 其原因是在计算机中,存储 x 有舍入误差,作加、乘、除法时又引入新的舍入误差.

1.4 误差及有效数字

对数学问题进行数值求解,求得的结果一般都包含有误差,即数值计算绝大多数情况是

近似计算,因此误差分析和估计是数值计算过程中的重要内容. 由它们可以确切地知道误差的性态和误差的界.

1.4.1 误差的来源

(1)**模型误差** 从实际问题提炼出数学模型时往往忽略了许多次要因素,因而即使数学模型能求出精确解,也与实际问题的真解不同,它们之差称为模型误差.

(2)**观测误差** 原始数据是由观测、实验,并加以记录而获得的. 由于仪器的精密性、实验手段的局限性、周围环境的变化以及人的工作态度和能力等因素,而使数据必然带有误差,这种误差称为观测误差.

(3)**截断误差** 理论上的精确值往往要求用无限次的运算才能获得,而实际计算时只能用有限次运算的结果来近似,这样引起的误差称为截断误差或方法误差.

(4)**舍入误差** 计算机只能对有限位数字进行运算,每个超出计算机字长的数据都要经过取舍或截断方法处理,这样引起的误差称为舍入误差.

在本课程中所涉及的误差,一般是指截断误差和舍入误差.

为了对这些误差有更清晰的认识,我们考虑一个实际而简单的问题:求地球的表面积 A. 我们首先需将地球近似地看做球体,可得如下的计算公式:$A=4\pi r^2$,其中 r 为地球半径. 显然,即使由这一公式可以精确计算,所得到的值也与地球的真实表面积有一定的误差,这一步的误差就是模型误差. 假如在计算中取地球的半径 $r=6\ 370$ km,这一值是由前人观测和计算得到的,具有一定的误差,这就是观测误差. 公式中的 π 是圆周率,它是一个无理数,取它的有限位,例如取 $\pi=3.141$,这样就产生截断误差. 最后将 $4\times3.141\times6\ 370^2$ 的计算结果进行四舍五入,得 $A=5.098\ 1\times10^8$ km^2,这一步就产生舍入误差.

1.4.2 绝对误差、相对误差和有效数字

设 x 为某一个量的准(精)确值,x^* 是 x 的近似值,称

$$e(x)=x-x^* \tag{1.4.1}$$

为近似值 x^* 的**绝对误差**,简称**误差**.

显然误差 $e(x)$ 既可为正,也可为负. 一般来说,准确值 x 是不知道的,因此误差 $e(x)$ 的准确值无法求出. 不过在实际工作中,可根据相关领域的知识、经验及测量工具的精度,事先估计出误差绝对值不超过某个正数 ε,即

$$|e(x)|=|x-x^*|\leqslant\varepsilon, \tag{1.4.2}$$

则称 ε 为近似值 x^* 的**绝对误差限**,简称**误差限**或**精度**.

式(1.4.2)去掉绝对值符号得

$$x^*-\varepsilon\leqslant x\leqslant x^*+\varepsilon,$$

这表示准确值 x 在区间$[x^*-\varepsilon,x^*+\varepsilon]$内,有时将准确值 x 写成

$$x=x^*\pm\varepsilon.$$

例如用卡尺测量一个圆杆的直径为 $x=350$ mm,它是圆杆直径的近似值,由卡尺的精度知道这个近似值的误差不会超过 0.5 mm,则有

$$|x-x^*|=|x-350|\leqslant0.5\ \text{mm},$$

于是该圆杆的直径为

$$x = 350 \pm 0.5 \text{ mm}.$$

用 $x = x^* \pm \varepsilon$ 表示准确值可以反映它的准确程度，但不能说明近似值的好坏．例如测量一根 10 cm 长的圆钢时发生了 0.5 cm 的误差和测量一根 10 m 长的圆钢时发生了 0.5 cm 的误差，其绝对误差都是 0.5 cm，但是后者的测量结果显然比前者要准确得多．这说明决定一个量的近似值的好坏，除了要考虑绝对误差的大小，还要考虑准确值本身的大小，这就需要引入相对误差的概念．称绝对误差与准确值之比

$$e_r = \frac{e(x)}{x} = \frac{x - x^*}{x} \tag{1.4.3}$$

为近似值 x^* 的**相对误差**．

在实际计算中，由于准确值 x 总是未知的，因此也把

$$e_r = \frac{e(x)}{x^*} = \frac{x - x^*}{x^*} \tag{1.4.4}$$

称为近似值 x^* 的**相对误差**．

在上面的例子中，前者的相对误差是 0.5/10 = 0.05，而后者的相对误差是 0.5/1 000 = 0.000 5．一般来说，相对误差越小，表明近似程度越好．与绝对误差一样，近似值 x^* 的相对误差的准确值也无法求出．仿照绝对误差限，称相对误差绝对值的上界 ε_r 为近似值 x^* 的**相对误差限**，即

$$|e_r| = \left|\frac{x - x^*}{x^*}\right| \leqslant \varepsilon_r. \tag{1.4.5}$$

注意：绝对误差和绝对误差限有量纲；而相对误差和相对误差限没有量纲，通常用百分数来表示．

为了能使一种数的表示法，既能表示其大小，又能表示其精确程度，于是需要引进有效数字的概念．在实际计算中，当准确值 x 有多位数时，常按四舍五入的原则得到 x 的近似值 x^*．例如无理数 e = 2.718 281 828…，若按四舍五入原则分别取二位和五位小数时，则得 e≈2.72 和 e≈2.718 28.

不管取几位小数得到的近似数，其绝对误差都不超过末位数的半个单位，即

$$|e - 2.72| \leqslant \frac{1}{2} \times 10^{-2}, \quad |e - 2.718\ 28| \leqslant \frac{1}{2} \times 10^{-5}.$$

下面我们将四舍五入抽象成数学语言，从而引进“有效数字”的概念．

若近似值 x^* 的绝对误差限是某一位的半个单位，就称其“准确”到这一位，且从该位直到 x^* 的最左边的第一位非零数字之间的所有数字为**有效数字**．

引入有效数字概念后，我们规定所写出的数都应该是有效数字，且在同一计算问题中，参加运算的数都应有相同的有效数字位．

例如，下列各数 358.467，0.004 275 22 和 8.000 034 中具有 5 位有效数字的近似值分别是 358.47，0.004 275 2，8.000 0.

注意：8.000 034 的 5 位有效数字是 8.000 0，而不是 8，因为 8 只有一位有效数字. 前者精确到小数点后 4 位，而后者仅精确到 1 位整数，两者相差是很大的，显然前者远较后者精确．由此可见，有效数字尾部的零不能随意省去，以免损失精度．

一般地，任何一个实数 x 经四舍五入后得到的近似值 x^* 都可写成如下标准形式

$$x^* = \pm 0.a_1a_2\cdots a_n \times 10^m, \tag{1.4.6}$$

其中 $a_1, a_2, \cdots, a_n$ 是0到9之间的自然数，$a_1 \neq 0$；m 为整数．如果 x^* 的误差限

$$|x - x^*| \leqslant \frac{1}{2} \times 10^{m-n}, \tag{1.4.7}$$

那么称近似值 x^* 具有 n 位有效数字．

根据有效数字的定义，不难验证 e 的近似值 2.718 28 具有6位有效数字．事实上，$2.718\ 28 = 0.271\ 828 \times 10$，这里 $n=6, m=1$，因为 $|e - 2.718\ 28| \leqslant \frac{1}{2} \times 10^{-5}$，所以它具有6位有效数字．

从上面的讨论可以看出，有效数字位数越多，绝对误差限就越小．值得注意的是，从计算机输出的近似值，一般并非每一位数字都是有效数字．至于有效数字与相对误差的关系有如下结论．

定理 1.4.1 设近似值 $x^* = \pm 0.a_1a_2\cdots a_n \times 10^m$（其中 $a_1 \neq 0$），有 n 位有效数字，则 x^* 的相对误差限 $\varepsilon_r \leqslant \frac{1}{2a_1} \times 10^{-n+1}$．

证明 由 x^* 有 n 位有效数字，知式(1.4.7)成立，而 $|x^*| > a_1 \times 10^{m-1}$，故有

$$|e_r| = \left|\frac{x - x^*}{x^*}\right| \leqslant \frac{\frac{1}{2} \times 10^{m-n}}{a_1 \times 10^{m-1}} = \frac{1}{2a_1} \times 10^{-n+1}.$$

即相对误差限 $\varepsilon_r \leqslant \frac{1}{2a_1} \times 10^{-n+1}$．定理得证．

定理 1.4.2 设近似值 $x^* = \pm 0.a_1a_2\cdots a_n \times 10^m$（其中 $a_1 \neq 0$），相对误差限 $\varepsilon_r = \frac{1}{2(a_1+1)} \times 10^{-n+1}$，则 x^* 至少有 n 位有效数字．

证明 由于 $\varepsilon = |x^*|\varepsilon_r$，而 $|x^*| \leqslant (a_1+1) \times 10^{m-1}$，所以

$$\varepsilon \leqslant (a_1+1) \times 10^{m-1} \times \frac{1}{2(a_1+1)} \times 10^{-n+1} = \frac{1}{2} \times 10^{m-n}.$$

因此，x^* 至少有 n 位有效数字．

1.5 避免误差危害的若干原则

在用计算机实现算法时，输入计算机的数据一般是有误差的（如观测误差等），计算机运算过程的每一步又会产生舍入误差，由十进制转化为机器数也会产生舍入误差，这些误差在计算过程中还会逐步传播和积累，因此必须研究这些误差对计算结果的影响．但一个实际问题往往需要亿万次以上的计算，且每一步都可能产生误差，因此不可能对每一步误差进行分析和研究，只能根据具体问题的特点进行研究，提出相应的误差估计．特别地，如果在构造算法的过程中注意了以下一些原则，那么将有效地减少和避免误差的危害、控制误差的传播和积累．

1.5.1 避免两个相近的数相减

在数值计算中两个相近的数相减会造成有效数字的严重损失，从而导致误差增大，影响

计算结果的精度.

例 1.5.1 当 $x=10\ 003$ 时,计算 $\sqrt{x+1}-\sqrt{x}$ 的近似值.

解 若使用6位十进制浮点运算,运算时取6位有效数字,结果

$$\sqrt{x+1}-\sqrt{x}=100.020-100.015=0.005$$

只有1位有效数字,损失了5位有效数字,使得绝对误差和相对误差都变得很大,影响计算结果的精度. 若改用

$$\sqrt{x+1}-\sqrt{x}=\frac{1}{\sqrt{x+1}+\sqrt{x}}=\frac{1}{100.020+100.015}=0.004\ 999\ 13,$$

则其结果有6位有效数字,与精确值 0.004 999 125 231 179 84…非常接近.

类似地,

$$\ln x-\ln y=\ln\frac{x}{y},\sin(x+\varepsilon)-\sin x=2\cos\left(x+\frac{\varepsilon}{2}\right)\sin\frac{\varepsilon}{2},$$

当 ε 很小,x 和 y 很接近时,采用等号右边的算法,有效数字就不损失.

一般地,当 $f(x)\approx f(x^*)$ 时,可用 Taylor 展开,即

$$f(x)-f(x^*)=f'(x^*)(x-x^*)+\frac{f''(x^*)}{2!}(x-x^*)+\cdots,$$

取等号右端的有限项近似等号左端. 若无法改变算法,直接计算时就要多保留几位有效数字.

1.5.2 防止大数"吃"小数

在数值计算中,参加运算的数的数量级有时相差很大,而计算机的字长又是有限的,因此如果不注意运算次序,那么就可能出现小数被大数"吃掉"的现象. 这种现象在有些情况下是允许的,但在有些情况下,这些小数很重要,若它们被"吃掉",就会造成计算结果的失真,从而影响计算结果的可靠性.

例 1.5.2 求二次方程 $x^2-(10^9+1)x+10^9=0$ 的根.

解 显然,方程的两个根为 $x_1=10^9$,$x_2=1$. 如果在浮点数的尾数只有8位的计算机上使用二次方程求根公式

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

首先要对阶,即

$$-b=10^9+1=0.100\ 000\ 00\times10^{10}+0.000\ 000\ 000\ 1\times10^{10}.$$

而计算机上只能达到8位,故计算机上 $0.000\ 000\ 000\ 1\times10^{10}$ 不起作用,即视为0,于是

$$-b=0.1000\ 000\ 0\times10^{10}=10^9.$$

类似地,有 $\sqrt{b^2-4ac}=|b|=10^9$,故所得两个根为 $x_1=10^9$,$x_2=0$. x_2 严重失真的原因是小数被大数"吃掉". 而从上述计算过程可以看出,x_1 是可靠的,这样利用根与系数的关系 $x_1\cdot x_2=c/a$,得

$$x_2=\frac{c}{ax_1}=\frac{10^9}{1\times10^9}=1.$$

需要说明的是大数"吃"小数在有些情况下是允许的,但在有些情况下却会造成失真.

再如,已知 $x=3\times10^{12}$,$y=7$,$z=-3\times10^{12}$,求 $x+y+z$. 如果按 $x+y+z$ 的次序来编程序,x "吃掉" y,而 x 与 z 互相抵消,其结果为零. 若按 $(x+z)+y$ 的次序来编程序,其结果为7. 由此可见,如果事先大致估计一下计算方案中各数的数量级,编制程序时加以合理安排,那么重要的小数就可以避免被"吃掉". 这也说明,用计算机作加减运算时,交换律和结合律往往不成立,不同的运算次序会得到不同的运算结果.

1.5.3 避免除数的绝对值远远小于被除数的绝对值

在用计算机实现算法的过程中,如果用绝对值很小的数作除数,往往会使舍入误差增大. 即在计算 $\frac{y}{x}$ 时,若 $0<|x|\ll|y|$,则可能产生较大的舍入误差,对计算结果带来严重影响,应尽量避免.

例 1.5.3 在4位浮点十进制数下,用消去法解线性方程组

$$\begin{cases}0.000\ 03x_1-3x_2=0.6,\\ x_1+2x_2=1.\end{cases}$$

解 仿计算机实际计算,将上述方程组写成

$$\begin{cases}0.300\ 0\times10^{-4}x_1-0.300\ 0\times10^{1}x_2=0.600\ 0\times10^{0}, & (1)\\ 0.100\ 0\times10^{1}x_1+0.200\ 0\times10^{1}x_2=0.100\ 0\times10^{1}. & (2)\end{cases}$$

用第一个方程消去第二个方程中含 x_1 的项,即(1)÷$(0.300\ 0\times10^{-4})$-(2)(注意:在第一步运算中出现了用很小的数作除数的情形,相应地在第二步运算中出现了大数"吃掉"小数的情形),得

$$\begin{cases}0.300\ 0\times10^{-4}x_1-0.300\ 0\times10^{1}x_2=0.600\ 0\times10^{0},\\ -0.100\ 0\times10^{6}x_2=0.200\ 0\times10^{5}.\end{cases}$$

解得

$$x_1=0,x_2=-0.2.$$

而原方程组的准确解为 $x_1=1.399\ 972\cdots$,$x_2=-0.199\ 986\cdots$. 显然上述结果严重失真.

如果反过来用第二个方程消去第一个方程中含 x_1 的项,那么就可以避免很小的数作除数的情形. 即(2)×$(0.300\ 0\times10^{-4})$-(1),得

$$\begin{cases}-0.300\ 0\times10^{1}x_2=0.600\ 0\times10^{0},\\ 0.100\ 0\times10^{1}x_1+0.200\ 0\times10^{1}x_2=0.100\ 0\times10^{1}.\end{cases}$$

解得

$$x_1=1.4,x_2=-0.2.$$

这是一组相当好的近似解.

1.5.4 减少运算次数

同样一个问题,如果能减少运算次数,那么不但可以节省计算机的计算复杂性,而且还能减少舍入误差. 因此在构造算法时,合理地简化计算公式是一个非常重要的原则.

例 1.5.4 已知 x,计算多项式 $p_n(x)=a_0+a_1x+\cdots+a_{n-1}x^{n-1}+a_nx^n$ 的值.

解 若直接计算,即先计算 $a_kx^k(k=1,2,\cdots,n)$,然后逐项相加,则一共需要作

$$1+2+\cdots+(n-1)+n=\frac{n(n+1)}{2}$$

次乘法和 n 次加法.

若对 $p_n(x)$ 采用如下算法

$$\begin{cases} s_n=a_n, \\ s_k=a_k+x\cdot s_{k+1} \quad (k=n-1,n-2,\cdots,2,1,0), \\ p_n(x)=s_0. \end{cases} \tag{1.5.1}$$

则只要 n 次乘法和 n 次加法,就可得到 $p_n(x)$ 的值. 此即为秦九韶算法(秦九韶是宋代数学家,此法由他最早提出,国外称此法为 Horner 法,比秦九韶算法晚了五六个世纪),而且秦九韶算法计算过程简单、规律性强、适于编程,所占内存也比前一种方法要小. 此外,由于减少了计算步骤,相应地也减少了舍入误差及其积累传播. 此例说明合理地简化计算公式在数值计算中是非常重要的.

1.5.5 注意算法的数值稳定性

为了避免误差在运算过程中的积累增大,在构造算法时,还要考虑算法的稳定性. 一个算法如果输入数据有误差,而在计算过程中舍入误差不增大,那么称此算法是**数值稳定的**,否则称此算法为**数值不稳定的**.

下面的例子说明了算法数值稳定性的重要.

例 1.5.5 计算积分 $I_n=\int_0^1 x^n e^{x-1}dx \quad (n=0,1,2,\cdots)$.

解 (1)用分步积分公式得 $I_n=x^n e^{x-1}\Big|_0^1-\int_0^1 nx^{n-1}e^{x-1}dx=1-nI_{n-1}$,只要知道 I_0,就可逐步求出 $I_n(n=1,2,\cdots)$. 从传统的角度看,这是一个漂亮的递推式. 因此可想到用下列递推公式计算:

$$\begin{cases} I_0=1-e^{-1}, \\ I_n=1-nI_{n-1} \quad (n=1,2,\cdots). \end{cases} \tag{1.5.2}$$

具体计算时,I_0 包含无理数 e,只能取有限位,引进了舍入误差,这就播下了失败的种子. 取 $I_0^*=0.632\ 120\ 559$,$\varepsilon_0=I_0-I_0^*$,其舍入误差 $|\varepsilon_0|\leqslant 0.5\times 10^{-9}$,那么有

$$I_n^*=1-nI_{n-1}^* \quad (n=1,2,\cdots).$$

计算结果为 $I_0^*=0.632\ 120\ 559$,$I_1^*=0.367\ 879\ 441$,$\cdots$,$I_{10}^*=0.084\ 499\ 2$,$\cdots$,$I_{13}^*=-1.000\ 627\ 2$,$\cdots$.

由于被积函数恒正,故它在[0, 1]上的积分恒正,但是结果显示 $I_{13}^*=-1.000\ 627\ 2<0$,这样的计算结果无法相信,计算失败. 是什么原因? 舍入误差的积累和传播!

记 $\varepsilon_n=I_n-I_n^*$ 为计算近似值 I_n^* 的误差,即

$$\varepsilon_n=I_n-I_n^*=(1-nI_{n-1})-(1-nI_{n-1}^*)=-n\varepsilon_{n-1} \quad (n=1,2,\cdots), \tag{1.5.3}$$

所以,$\varepsilon_1=-\varepsilon_0$,$\varepsilon_2=2\varepsilon_0$,$\cdots$,$\varepsilon_n=(-1)^n n!\cdot\varepsilon_0$. 这样计算时,产生的误差居然是 ε_0 的 $n!$ 倍. 计算结果必定不可靠.

(2)改变思路. 因为从式(1.5.3)观察到,$\varepsilon_{n-1}=-\frac{1}{n}\varepsilon_n$,如果先算出 I_n^*,那么 I_{n-1}^*

的误差就缩小为 I_n^* 的误差的 $1/n$. 这就启发我们采用倒递推计算格式 $I_{n-1}=(1-I_n)/n$ $(n=20,19,\cdots,2,1)$,当然这需要估计一个较"好"的 I_n^* 为前提. 由于

$$\min_{0\leqslant x\leqslant 1} e^{x-1}\int_0^1 x^n dx < I_n < \max_{0\leqslant x\leqslant 1} e^{x-1}\int_0^1 x^n dx,$$

也即有

$$\frac{e^{-1}}{n+1}<I_n<\frac{1}{n+1},$$

故有 $0.017\ 518\ 068<I_{20}<0.047\ 619\ 047$,取

$$\begin{cases} I_{20}^*=\dfrac{(0.017\ 518\ 068+0.047\ 619\ 047)}{2}=0.032\ 568\ 558, |I_{20}-I_{20}^*|<0.015\ 051, \\ I_{n-1}^*=(1-I_n^*)/n \quad (n=20,19,\cdots,2,1). \end{cases}$$

这样计算得 $I_{19}^*=0.048\ 371\ 572\ 1,\cdots,I_{13}^*=0.066\ 947\ 702\ 61,\cdots,I_1^*=0.367\ 879\ 441\ 2,I_0^*=0.632\ 120\ 558\ 8$.

尽管得值比较粗糙,但计算结果却越来越精确. 这是因为误差

$$\varepsilon_{n-1}=I_{n-1}-I_{n-1}^*=-\frac{1}{n}\varepsilon_n \quad (n=20,19,\cdots,2,1),$$

每一步的误差是缩小的,所以这样的算法是数值稳定的. 这个例子告诉我们,用数值方法在解决实际问题时一定要选择数值稳定的算法.

1.6 MATLAB 简介

1.6.1 MATLAB 的概况

MATLAB 是矩阵实验室(Matrix Laboratory)之意. 除具备卓越的数值计算能力外,它还提供了专业水平的符号计算、文字处理、可视化建模仿真和实时控制等功能.

MATLAB 的基本数据单位是矩阵,它的指令表达式与数学工程中常用的形式十分相似,故用 MATLAB 来解算问题要比用 C,FORTRAN 等语言完成相同的事情简捷得多.

当前流行的 MATLAB 7.0/Simulink 4.0 包括拥有数百个内部函数的主包和几十种工具包(Toolbox). 工具包又可以分为功能工具包和学科工具包,功能工具包用来扩充 MATLAB 的符号计算、可视化建模仿真、文字处理及实时控制等功能;学科工具包是专业性比较强的工具包,控制工具包、信号处理工具包、通信工具包等都属于此类.

开放性使 MATLAB 广受用户欢迎. 除内部函数外,所有 MATLAB 主包文件和各种工具包都是可读可修改的文件,用户通过对源程序的修改或加入自己编写的程序可构造新的专用工具包.

1.6.2 MATLAB 的发展历程

在 20 世纪 70 年代中期,Cleve Moler 博士和其同事在美国国家科学基金的资助下开发了调用 EISPACK 和 LINPACK 的 FORTRAN 子程序库. EISPACK 是特征值求解的 FORTRAN 程序库,LINPACK 是解线性方程的程序库. 在当时,这两个程序库代表矩阵运算的最高水平.