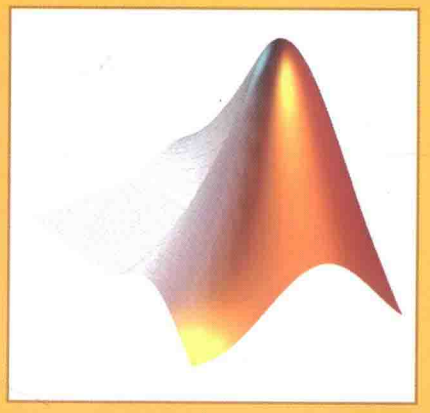

数值分析及其MATLAB实验

（第2版）

姜健飞 吴笑千 胡良剑 编著

清华大学出版社

数值分析及其 MATLAB 实验

（第2版）

姜健飞 吴笑千 胡良剑 编著

清华大学出版社
北京

内 容 简 介

本书详细介绍了数值分析的基本概念和方法，包括数值代数、迭代法、数据建模、数值微积分和常微分方程数值解等，并基于 MATLAB 软件介绍了相应的工程数值算法及 MATLAB 软件的偏微分方程数值解和最优化方法两个专用工具箱。书中提供了大量习题和上机实验题，并配有习题解答、主要算法的流程图和多媒体教学资料。

本书可作为理工科研究生或本科生数值分析课程及其数值实验的教学用书，也可供科研和工程技术人员作为解决数值计算问题的参考书。

图书在版编目(CIP)数据

数值分析及其 MATLAB 实验/姜健飞，吴笑千，胡良剑编著. --2 版. --北京：清华大学出版社，2015
ISBN 978-7-302-40740-9

Ⅰ. ①数… Ⅱ. ①姜… ②吴… ③胡… Ⅲ. ①数值分析－Matlab 软件－高等学校－教材 Ⅳ. ①O241-39

中国版本图书馆 CIP 数据核字(2015)第 160622 号

责任编辑：佟丽霞　洪　英
封面设计：常雪影
责任校对：赵丽敏
责任印制：刘海龙

出版发行：清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦 A 座　　**邮　　编**：100084
社 总 机：010-62770175　　**邮　　购**：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质量反馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
印 装 者：清华大学印刷厂
经　　销：全国新华书店
开　　本：185mm×260mm　　**印　　张**：17.75　　**字　　数**：426 千字
版　　次：2004 年 6 月第 1 版　2015 年 9 月第 2 版　　**印　　次**：2015 年 9 月第1次印刷
印　　数：1～2500
定　　价：36.00 元

产品编号：054831-01

第2版前言

本书的第1版已经在东华大学连续使用11年。数值分析的教学课时已从每学期18周(共54学时)缩减到每学期16周(共48学时)。在课时减少1/9的情况下还要保证教学质量,就要抓住数值分析课程的关键内容,把最重要的概念、理论以及思想方法介绍给学生,内容选择力求少而精,还要删减一些运算烦琐、与本课程关系不大的数学证明和计算。另外,教学内容应该和学生所学专业有机地结合在一起,可以增加一些和所学专业相关的数值分析案例。作者根据多年的教学改革经验以及许多学生的反馈建议和意见,对本书进行了修正和补充。主要修改了第1版的错误,简化了一些定理的证明和公式的计算,增加了一些习题。

本书主要修订内容如下。

(1) 在过去的7年里,MATLAB软件多次更新,从7.0版升级至8.1版。本次修订力图体现有关更新,主要包括:

① MATLAB界面使用更方便,如Command Window的fx函数浏览按钮、doc超文本帮助、Home工具条等,这些变化主要在前两章介绍。

② 新版MATLAB中,inline函数基本不再使用,由匿名函数或函数句柄代替。函数求值指令feval也不再使用,直接使用函数名加括号来求值。对此,本书作了全面更新。

③ 新版MATLAB的数值积分计算使用integral类函数,能求解反常积分和任意区域上的重积分,本书第5章作了相应更新并删除了自编函数dblquad2。

④ MATLAB符号计算引擎由Maple变更为Mupad,因此,附录B全面作了改写。

(2) 把p阶收敛速度定义中的式(1)$\frac{e_{k+1}}{e_k^p}\xrightarrow{k\to\infty}c(\neq 0)$改为式(2)$\frac{|e_{k+1}|}{e_k^p}\xrightarrow{k\to\infty}c(\neq 0)$。已经找到反例,有数列不满足式(1),但它有$p$阶收敛速度。

(3) 简化了按行严格对角占优矩阵的高斯-赛德尔(Gauss-Seidel)迭代法的收敛性证明。

(4) 简化了两点高斯(Gauss)积分公式的证明。方法是构造以x_0,x_1为根的辅助函数$g(x)=(x-x_0)(x-x_1)=x^2+ax+b$。这个方法的优点在于:把关于$x_0$,$x_1$的非线性方程组化为关于$a$,$b$的线性方程组(求以$x_0$,$x_1$为根的多项式系数),很容易求解。

(5) 直接用积分中值定理推导插值型求积公式余项会有问题。插值多项式余项中ξ是x的函数,但不一定是x的连续函数,因此不能直接使用积分中值定理。为此,引入了复合

函数的积分中值定理。只要保证函数 $\xi(x)$ 在 $[a,b]$ 上封闭,就能避免此问题。

(6) 在一些主要专业名词后增加了英文翻译,便于学生查询英文参考文献以及留学生学习。

(7) 增加了 MATLAB 常用语句和主要数值分析算法的流程图,便于学生掌握编程。

作 者

2015 年 7 月

第1版前言

数值分析是工科专业研究生的一门重要公共基础课，主要介绍工程数学问题数值计算的一些基本概念和方法，培养研究生使用计算机技术进行科学与工程计算的能力。长期以来，在数值分析教学中一个很大的遗憾是将数值分析作为一门纯粹的理论课程来讲授，并不进行任何程序设计，这样学生就很难对数值分析的一些重要的特征（如计算速度和稳定性问题等）有深入的理解。究其原因，在于很多数值算法涉及复杂的程序结构。使用FORTRAN、C等通用程序设计语言编程效率比较低，很难在有限的教学课时中包含一些复杂而又非常重要的数值计算问题，加上教学软硬件的限制，造成传统教学中一般都是只讲算法原理和误差分析，而不涉及计算实验。

近年来出现一些优秀的数学软件，如Maple、MATLAB、Mathematica等。这些软件的内核包含了一些关键而又复杂的数值算法，大大提高了编程效率。例如，矩阵特征值问题QR算法用C语言从底层开始编程，需要对算法结构有详尽的了解，全部程序大约需要300句；而用MATLAB编程，不必懂得QR算法的具体细节，只需要一两句简单语句就可以解决问题。从20世纪90年代开始，国内出现了一些基于数学软件的数值分析教材。但遗憾的是，这类教材大都只是用一个附录介绍软件，而没有将数学软件融合到教材中。本书的目标是努力将数值分析理论学习与数学软件编程实验紧密结合起来，提供一本真正基于MATLAB的、适合理工科专业研究生教学实践需要的数值分析教材。

我们面对的现实是，今天研究生需要解决的工程计算问题涉及面越来越广，而他们接受的数学理论知识并没有跟上，这就要求教材涉及面要宽而起点又不能太高。我们解决这一问题的基本思路是：充分利用数学软件的功能，加强实验环节，分层次处理。一方面对于数值计算的一些最基本的概念和思想讲深讲透，并通过实验加强对于数值计算典型特征（如浮点数、计算量、病态问题等）的理解。另一方面对于较复杂的高效率算法不详细讲解，而直接利用MATLAB软件解决。这样，在降低数值分析理论深度的同时增加了实用性，使得研究生在研究课题中遇到工程计算问题时能较快上手并有效地解决问题。

本书详细介绍了数值分析的基本概念和方法，包括数值代数、迭代法、数据建模、数值微积分和常微分方程数值解等，并基于MATLAB软件介绍了相应的工程数值算法及MATLAB软件的偏微分方程数值解和最优化方法两个专用工具箱。书中提供了大量习题、上机实验题以及流程图，并配有习题解答和多媒体教学资料。

本书共有8章。第1章是算法和误差的概念。第2～6章基本按照传统的数值分析教材内容来安排，包含了数值分析课程中最基本的几个问题——线性方程组、非线性方程、插值与拟合、数值微积分和常微分方程。同时，大部分涉及算法的内容都安排一小节给出相应

的 MATLAB 程序和分析,每章最后一节介绍 MATLAB 的相关指令,并包含一些高效率的 MATLAB 算法(如非线性方程组求解、Lobatto 数值积分和常微分方程 RKF 法等),据此可以解决大部分常规的工程计算问题。此外,每章都配有习题和上机实验题,习题适合用手工和计算器完成,而上机实验题适合用计算机完成。第 7 章和第 8 章介绍了 MATLAB 的偏微分方程和最优化方法两个工具箱。尽管这两个问题在传统的数值分析教材中并不常见,但在工程计算问题中却经常遇到。我们并不计划将这些纳入数值分析工科研究生课程基本教学内容,而主要供读者在解决工程计算问题时参考。附录 A 是 MATLAB 入门介绍,没有学过 MATLAB 的读者应该首先学习这部分内容。附录 B 介绍 MATLAB 的符号数学工具箱,这是数学软件最令人叫绝的功能之一,尽管它对于数值分析并没有多大意义。习题解答放在附录 C,而附录 D 和附录 E 给出了全书的 MATLAB 指令或函数和 M 文件索引。

本书是作者在东华大学进行了 4 年教学改革实践的基础上编写而成的。在东华大学教学实践中,54 学时中安排 36 学时上理论课、18 学时上机实验课,实验课主要采用研究生在机房练习、教师辅导的方式,可以完成第 1～6 章以及附录 A 的全部内容。我们制作了实验课部分网上辅导教学课件,需要的读者请与我们联系(对于采用本教材的单位可以免费提供)。

本书的编写和出版得到了上海市重点研究生教学用书建设基金和东华大学研究生主干课程建设基金的大力支持,对此我们深表感谢!

作　者

2004 年 1 月

目 录

第 1 章 数值分析的基本概念 …… 1

1.1 数值算法的研究对象 …… 1
1.2 误差分析的概念 …… 3
1.3 数值算法设计的注意事项 …… 8
习题 …… 10
上机实验题 …… 11

第 2 章 数值代数 …… 13

2.1 高斯消去法 …… 13
2.2 直接三角分解法 …… 21
2.3 范数和误差分析 …… 27
2.4 基于 MATLAB：逆矩阵与特征值问题 …… 32
习题 …… 41
上机实验题 …… 42

第 3 章 迭代法 …… 45

3.1 二分法 …… 45
3.2 迭代法原理 …… 48
3.3 牛顿迭代法和迭代加速 …… 52
3.4 解线性方程组的迭代法 …… 56
3.5 基于 MATLAB：非线性方程组 …… 64
习题 …… 67
上机实验题 …… 68

第 4 章 数据建模 …… 70

4.1 多项式插值 …… 70
4.2 牛顿插值 …… 76
4.3 三次样条插值 …… 79
4.4 最小二乘拟合 …… 86

4.5 基于 MATLAB：非线性拟合与多元插值 …… 94
习题 …… 100
上机实验题 …… 102

第 5 章 数值微积分 …… 105

5.1 数值积分公式 …… 105
5.2 数值积分的余项 …… 112
5.3 复化求积法与步长的选取 …… 115
5.4 数值微分法 …… 123
5.5 基于 MATLAB：数值微积分 …… 125
习题 …… 128
上机实验题 …… 129

第 6 章 常微分方程的数值解法 …… 131

6.1 欧拉法及其改进 …… 131
6.2 龙格-库塔格式 …… 137
6.3 收敛性与稳定性 …… 140
6.4 RKF 格式与亚当斯格式 …… 143
6.5 微分方程组与高阶微分方程 …… 147
6.6 基于 MATLAB：刚性方程组和边值问题 …… 151
习题 …… 157
上机实验题 …… 158

第 7 章 MATLAB 偏微分方程数值解 …… 160

7.1 偏微分方程有限元法 …… 160
7.2 用图形用户界面方式解 PDE …… 164
7.3 用指令方式解 PDE …… 173
7.4 一维问题求解 …… 184
上机实验题 …… 188

第 8 章 MATLAB 最优化方法 …… 190

8.1 最优化方法简介 …… 190
8.2 无约束优化 …… 192
8.3 约束最优化 …… 196
8.4 最小二乘法及多目标优化 …… 200
上机实验题 …… 205

附录 A MATLAB 简介 …… 208

A.1 MATLAB 桌面 …… 208

A.2 数据和变量 …… 210
A.3 数组及其运算 …… 213
A.4 数据类型和数据文件 …… 221
A.5 程序设计 …… 225
A.6 作图 …… 232
A.7 在线帮助和文件管理 …… 237
上机实验题 …… 239

附录 B MATLAB 符号计算 …… 241

B.1 符号对象 …… 241
B.2 符号矩阵和符号函数 …… 243
B.3 符号微积分 …… 245
B.4 符号方程和符号微分方程 …… 249
B.5 符号计算局限性和 Mupad 调用 …… 251
上机实验题 …… 252

附录 C 习题解答 …… 254

附录 D MATLAB 指令或函数索引 …… 267

附录 E M 文件索引 …… 270

参考文献 …… 271

第 1 章

数值分析的基本概念

1.1 数值算法的研究对象

1. 数值算法的研究对象

在解决现代工程技术问题时，常常需要首先建立问题的数学模型，然后合理地设计问题的算法，并通过计算机计算，最后获得问题的解答。本课程正以此为基本研究对象，是研究算法的学科。为使读者对本课程的基本目的和内容有一个大致的了解，我们先讨论下列两组数学模型的计算问题。

例 1.1 (1) 求解线性方程组 $\boldsymbol{Ax}=\boldsymbol{b}$，其中 $\boldsymbol{A}$ 为 3 阶可逆方阵，$\boldsymbol{x}=(x_1,x_2,x_3)^{\mathrm{T}}$；

(2) 求代数方程 $3x^2+8x-3=0$ 在[0,1]上的根 x^*；

(3) 已知 $y=P(x)$ 为 $[x_0,x_1]$ 上的直线，满足 $P(x_0)=y_0$，$P(x_1)=y_1$，$\bar{x}\in(x_0,x_1)$，求 $P(\bar{x})$；

(4) 计算定积分 $I=\int_a^b \frac{1}{x}\mathrm{d}x$ $(1<a<b)$；

(5) 解常微分方程初值问题 $\begin{cases} y'=x+y, \\ y(0)=0。 \end{cases}$

解 应用微积分学和线性代数的基本知识，不难求得问题的解。

(1) 由线性代数克莱姆(Cramer)法则，得 $x_1=\frac{D_1}{D}$，$x_2=\frac{D_2}{D}$，$x_3=\frac{D_3}{D}$，其中 $D=|\boldsymbol{A}|$，D_j 为由 $\boldsymbol{b}$ 置换 D 的第 j 列所得。所以问题归结为计算 4 个 3 阶行列式。

(2) 利用初等求根公式容易得到 $x^*=\frac{1}{3}$。

(3) 由解析几何知识，得 $P(\bar{x})=y_0+\frac{y_1-y_0}{x_1-x_0}(\bar{x}-x_0)$。

(4) 根据积分公式，得 $I=\ln\frac{b}{a}$。

(5) 根据线性常微分方程求解公式，得 $y(x)=-x-1+\mathrm{e}^x$。

例 1.2 (1) 求解线性方程组 $\boldsymbol{Ax}=\boldsymbol{b}$，其中 $\boldsymbol{A}$ 为 30 阶可逆方阵，$\boldsymbol{x}=(x_1,x_2,\cdots,x_{30})^{\mathrm{T}}$；

(2) 求超越方程 $x\mathrm{e}^x=1$ 在[0,1]上的根 x^*；

(3) 已知 $y=f(x)$ 为 $[x_0,x_1]$ 上的函数，满足 $f(x_0)=y_0$，$f(x_1)=y_1$，$\bar{x}\in(x_0,x_1)$，求 $f(\bar{x})$；

(4) 计算定积分 $I=\int_a^b \frac{1}{\ln x}\mathrm{d}x$ $(1<a<b)$;

(5) 解常微分方程初值问题 $\begin{cases} y'=x+y^2, \\ y(0)=0。 \end{cases}$

解 例 1.2 与例 1.1 初步看来"差不多",是否也容易解决呢?试讨论如下。

(1) 由克莱姆法则需要计算 31 个 30 阶行列式,如果按照行列式的展开式计算,由于每个 30 阶行列式有 30! 项,每一项为 30 个元素乘积,共需 30! $\times 29\times 31 \approx 2\times 10^{35}$ 次乘法,计算量非常大。

(2) 设 $f(x)=x\mathrm{e}^x-1$,由于 $f(0)<0$,$f(1)>0$,根据连续函数介值定理,方程在$[0,1]$上的根 x^* 存在,但无法求得 x^* 的解析形式,只能想办法求其近似值。

(3) 此问题看似无理,但又实实在在,借用例 1.1 相应问题的结果求解,$f(\bar{x})\approx P(\bar{x})=y_0+\frac{y_1-y_0}{x_1-x_0}(\bar{x}-x_0)$。

(4) 高等数学的牛顿-莱布尼茨公式对该问题失效,因为无法找到被积函数 $f(x)=\frac{1}{\ln x}$ 的原函数,我们可以考虑一些近似求解方法。

(5) 线性常微分方程比较容易得到解析解,而非线性常微分方程一般都没有解析解,主要依靠数值解法求取近似解。该问题与例 1.1 相应问题似乎只有"一点点"差别,但它没有解析解,只能依赖数值方法。

上述例子提示我们,在高等数学中学习的算法只能求解一些比较简单特殊的数学模型,工程实践中遇到的许多数学问题或者由于计算量太大或者由于没有解析方法,不能有效地使用手工计算,需要借助于计算机的计算能力。同时,我们必须认识到:第一,计算机的认识能力是有限的,这就要求我们设计其能接受的"算法"。例如,我们考虑用 C 语言解例 1.2(4),但 C 语言并不能识别"积分"这个数学概念,需要我们预先将积分转化为初等运算和初等函数构成的计算问题(在第 5 章我们将有多种方法求解此题)。第二,计算机的计算能力也是有限的,这就要求我们为其设计"好的算法"。例如对于例 1.2(1)的计算量,即使用 2013 年全球最快的超级计算机"天河二号"计算,每秒能做 33.86 千万亿次乘法也需要 591 亿年!这就要求我们设计出更有效的算法。事实上,使用第 2 章的方法,我们用普通的计算机可以在 1s 内求解此题。

本课程将会给出解决例 1.2 这类问题的算法。这里必须指出的是,实际问题远比例 1.2 列举的问题复杂得多,本课程只是讨论一些常用的基本数值算法。实际工程中,不同的算法适合不同的问题,所以我们需要更有针对性的算法。课程中所介绍的数值计算的一些基本概念和基本思想是具有普遍意义的。读者在学习中应将着重点放在各种算法的基本思想方法上,而不是死记硬背一些计算公式。

2. 数值算法的特点

对于给定的问题和设备,一个**算法**(algorithm)是用该设备可理解的语言表示的,是对解决这个问题的一种方法的精确刻画。这里的设备主要指计算机软件系统,所以也称计算机算法。对于算法刻画的要求取决于设备的语言能力。如果使用汇编语言之类的低级语言,那么对算法的描述需要极为详尽,包括变量地址、算术运算方式都要表达详尽;如果使用

C、FORTRAN 等高级语言，那么算术运算和初等函数等就成为设备可理解的语言，所以其描述可以粗略一些；而如果使用 MATLAB 这类专业化数学软件，那么就连积分、微分方程等的计算也不必详述。本课程讨论的大部分算法是建立在 C、FORTRAN 等高级语言这个层次上，也有一些算法建立在 MATLAB 软件层次上。

计算机算法主要包含数值算法、非数值算法和软计算方法三类。数值算法主要指与连续数学模型有关的算法，如数值线性代数、方程求解、数值逼近、数值微积分、微分方程数值解和最优化计算方法等；非数值算法主要指与离散数学模型有关的算法，如排序、搜索、分类、图论算法等；软计算方法是近来发展的不确定性算法的总称，包括随机模拟、神经网络计算、模糊逻辑、遗传算法、模拟退火算法和 DNA 算法等。

本课程主要研究的是数值算法，它是计算机算法内容最为丰富、也是最基本的一部分内容。数值算法具有下列几个显著特点。

(1) 有穷性

由于计算机编码的离散性本质，数值算法和所有计算机算法一样，必须在有限步完成，同时其处理对象的规模也是有限的。

(2) 数值性

由于其研究对象的特点，数值算法所涉及的数据类型主要是数值型，特别是以实数型居多。字符类型数据在数值算法中很少出现，整数型也比较少，数值算法中使用的大部分变量都是浮点实数，而且通常都是双精度的。

(3) 近似性

近似性是数值算法最显著的特点，这是由其研究对象和计算机的离散性本质之间的矛盾决定的。与连续数学模型相关的很多概念(如微分、积分)都是数学上的无穷极限，而计算机算法无法确切地表达和执行这些概念，只能得出某种程度的有穷近似，这就决定了它们之间必然存在误差。

1.2　误差分析的概念

在数值计算中，误差往往是不可避免的，那么怎样估计计算误差，判断计算结果的有效性，就成为数值分析极其重要的内容。

1. 误差限和有效数字

首先给出一些误差分析术语的定义。这些术语常常被人们不加定义地使用，但在不同的文献中的含义却有细微差别，我们采用了教科书中比较标准的定义。

定义 1.1(误差和相对误差)　设 x^* 是某量的准确值，x 是 x^* 的近似值，称 $\delta(x)=x^*-x$ 为 x 的**误差**(error)或**绝对误差**(absolute error)。在数值计算问题中，x^* 是未知的，从而 $\delta(x)$ 也无法精确得到，但我们往往可以得到 $\delta(x)$ 的绝对值上界，即 $|x^*-x|\leqslant\varepsilon$，称 ε 为 x 的**误差限**(error bound)或**精度**(accuracy)。应用中常记为 $x\pm\varepsilon$。误差与准确值的比值 $\delta_r(x)=(x^*-x)/x^*$ 称为 x 的**相对误差**(relative error)。如果 $|(x^*-x)/x^*|\leqslant\varepsilon_r$，称 ε_r 为 x 的**相对误差限**(relative error bound)或**相对精度**(relative accuracy)。当 ε_r 很小时，$\varepsilon_r\approx\varepsilon/|x|$。

定义 1.2(准确位数和有效数字)　设 x^* 是某量的准确值，x 是 x^* 的近似值，即

$$x = \pm 0.a_1\, a_2 \cdots a_n \cdots \times 10^m, \tag{1.1}$$

其中,m 为整数,$a_1 \sim a_n$ 为 0～9 中一个数字且 $a_1 \neq 0$。如果

$$|x^* - x| \leqslant 0.5 \times 10^{-k}, \tag{1.2}$$

即 x 的误差不超过 10^{-k} 位的半个单位,则称近似数 x **准确到** 10^{-k} **位**,并说 x 有 $m+k$ 位**有效数字**(significant digit)。

例 1.3 设圆周率 $\pi = 3.1415926\cdots$,求下列近似数的绝对误差、相对误差和有效数字。

(1) $x_1 = 3.14$;

(2) $x_2 = 3.141$;

(3) $x_3 = 3.142$;

(4) $x_4 = 3.1414$。

解 (1) $\pi - x_1 = 0.15926\cdots \times 10^{-2}$,$(\pi - x_1)/\pi = 0.5072\cdots \times 10^{-3}$,由于 $x_1 = 0.314 \times 10^1$,$|\pi - x_1| \leqslant 0.5 \times 10^{-2}$,从而 x_1 有 3 位有效数字;

(2) $\pi - x_2 = 0.5926\cdots \times 10^{-3}$,$(\pi - x_2)/\pi = 0.1887\cdots \times 10^{-3}$,由于 $x_2 = 0.3141 \times 10^1$,$|\pi - x_2| \leqslant 0.5 \times 10^{-2}$,从而 x_2 有 3 位有效数字;

(3) $\pi - x_3 = -0.4073\cdots \times 10^{-3}$,$(\pi - x_3)/\pi = -0.1296\cdots \times 10^{-3}$,由于 $x_3 = 0.3142 \times 10^1$,$|\pi - x_3| \leqslant 0.5 \times 10^{-3}$,从而 x_3 有 4 位有效数字;

(4) $\pi - x_4 = 0.1926\cdots \times 10^{-3}$,$(\pi - x_4)/\pi = -0.613\cdots \times 10^{-4}$,由于 $x_4 = 0.31414 \times 10^1$,$|\pi - x_4| \leqslant 0.5 \times 10^{-3}$,从而 x_4 有 4 位有效数字。

有效数字概念的通俗定义是这样的:设 x^* 是某量的准确值,x 是 x^* 的近似值,如果在从第一个非零数字开始的第 n 位进行四舍五入,x^* 和 x 的结果完全一致,则称 x 有 n 位有效数字。但这一定义在数值分析中无法应用,因为 x^* 是未知的。定义 1.2 是该通俗定义的抽象,其出发点为绝对误差限,而有效数字应该理解为计算精度的一种简略的表达。在例 1.3 中,对于 x_1, x_2, x_3,定义 1.2 与通俗定义一致,而 x_4 不一致,应该说定义 1.2 能更好地表达精度的好坏,因为按照通俗定义,x_4 只有 3 位有效数字,但实际上,x_4 的误差明显比 x_3 的误差小,是 π 更好的近似值,所以通俗定义是不合理的。

2. 截断误差与收敛性

截断误差也称方法误差,是数值算法设计中最主要考虑的误差问题。许多数学概念(如微分、积分等)都具有极限上的意义,不可能经过有限次算术运算计算出来,我们只能构造某种算法用有限次算术运算作近似替代,由此产生的误差称为该数值算法的**截断误差**(truncation error)。

例如,已知 x 在 0 附近,要计算指数函数 e^x 的值。由于 e^x 并不是算术运算,根据微分学的泰勒(Taylor)公式

$$e^x = 1 + x + \frac{x^2}{2!} + \cdots + \frac{x^n}{n!} + R_n(x), \tag{1.3}$$

考虑到一方面计算多项式只用到算术运算,另一方面当 n 充分大时,余项 $R_n(x)$ 的数值很小,这样便可以用式(1.3)右边前面的 n 次多项式来近似替代 e^x,从而得到近似计算公式为

$$e^x \approx S_n(x) = 1 + x + \frac{x^2}{2!} + \cdots + \frac{x^n}{n!}, \tag{1.4}$$

其截断误差可以用余项公式

$$e^x - S_n(x) = R_n(x) = \frac{x^{n+1}}{(n+1)!}e^{\xi} \quad (\xi \text{ 在 } 0 \text{ 与 } x \text{ 之间}) \tag{1.5}$$

来进行分析。根据微分学理论，对于固定的 x，当 $n \to \infty$ 时，余项 $R_n(x) \to 0$，即 $S_n(x) \to e^x$。据此我们称算法(1.4)是**收敛**的。一种算法是收敛的，说明该算法总可以通过提高计算量使得截断误差任意小。

需要指出的是，大多数高级语言已经将指数函数、对数函数、三角函数等许多初等数学函数做成软件的标准库函数，所以我们编程时不需要再作处理。本质上，计算机只能作算术运算，高级语言中的数学函数正是根据式(1.4)这类算法构造的。

3. 舍入误差和数值稳定性

计算机中采用二进制实数系统，并且表示成规格化浮点形式：

$$\pm 2^m \times 0.\beta_1\beta_2\cdots\beta_t,$$

称为**机器数**。其中整数 m 称为**阶码**，用二进制数

$$m = \pm \alpha_1\alpha_2\cdots\alpha_s$$

表示，$\alpha_i = 0$ 或 $1(i = 1,2,\cdots,s)$。小数 $0.\beta_1\beta_2\cdots\beta_t$ 称为**尾数**，$\beta_1 = 1$，$\beta_j = 0$ 或 $1(j = 2,3,\cdots,t)$。正整数 t 称为**字长**。s 决定了机器数的绝对值范围，而 t 决定了机器数的表示精度。由于 s 和 t 都是有限的，所以机器数是离散的而非连续的。

机器数有单精度和双精度之分。通常，单精度为 32 位，双精度为 64 位，它们是正负号、阶码和尾数所占二进制的总长度(见图 1-1)。

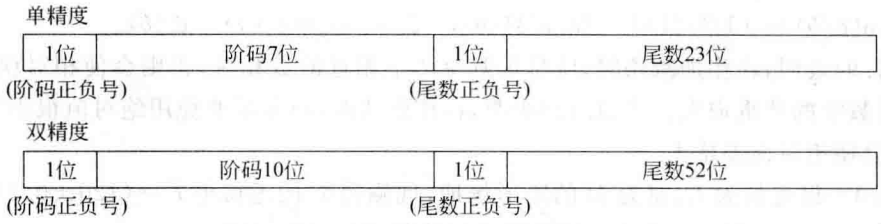

图 1-1　计算机中数的表示

单精度机器数，$t = 23$，大约相当于十进制 7 位有效数字；双精度机器数，$t = 52$，大约相当于十进制 15 位有效数字。单精度机器数和双精度机器数的绝对值范围分别为 $2^{-128} \sim 2^{128}$(即 $2.9 \times 10^{-39} \sim 3.4 \times 10^{38}$)和 $2^{-1024} \sim 2^{1024}$(即 $5.56 \times 10^{-309} \sim 1.79 \times 10^{308}$)。低于该范围的机器数视为 0，高于该范围的机器数视为无穷大。

由于机器字长的限制而产生的误差称为**舍入误差**(rounding error 或者 round-off error)。十进制数进入计算机运算时先转换成二进制机器数，并对 t 位后的数作舍入处理，使得尾数为 t 位，因此一般都有舍入误差。两个二进制机器数作算术运算时，也要作类似的舍入处理，使得尾数为 t 位，从而也有舍入误差。二进制数计算结果输出时再转换成十进制数。所以说，舍入误差遵循的是二进制操作规律(见上机实验题 1)。但为了符合通常的习惯，在以后的讨论中我们仍然采用十进制进行误差分析，而忽略十进制与二进制的差异。

设函数 $y = f(x_1, x_2, \cdots, x_n)$ 是一个算法或模型，x_i^* 是变量 x_i 的准确值，而 $\tilde{x}_i$ 是变量 x_i 的近似值，$i = 1,2,\cdots,n$。如果 $\tilde{x}_i$ 的精度为 ε_i，且 f 的计算过程中没有新的误差产生，那

么计算结果$\tilde{y}=f(\tilde{x}_1,\tilde{x}_2,\cdots,\tilde{x}_n)$具有怎样的精度呢?

如果$f(x_1,x_2,\cdots,x_n)=\sum_{i=1}^{n}a_ix_i$(线性函数),那么

$$|\delta(\tilde{y})|=|f(x_1^*,x_2^*,\cdots,x_n^*)-f(\tilde{x}_1,\tilde{x}_2,\cdots,\tilde{x}_n)|\leqslant\sum_{i=1}^{n}|a_i|\varepsilon_i。\tag{1.6}$$

又如果$f(x_1,x_2,\cdots,x_n)$是非线性函数,此时对于误差传播的严格估计是很困难的,而且不等式估计结果往往过于保守,所以一般采用一次近似估计法。根据微分学理论,绝对误差为

$$\delta(\tilde{y})=f(x_1^*,x_2^*,\cdots,x_n^*)-f(\tilde{x}_1,\tilde{x}_2,\cdots,\tilde{x}_n)\approx\sum_{i=1}^{n}\left(\frac{\partial f}{\partial x_i}\right)^*\delta(\tilde{x}_i)。\tag{1.7}$$

相对误差为

$$\delta_r(\tilde{y})=\frac{\delta(\tilde{y})}{y^*}\approx\sum_{i=1}^{n}\left(\frac{\partial f}{\partial x_i}\right)^*\frac{x_i^*}{y^*}\delta_r(\tilde{x}_i)。\tag{1.8}$$

其中,$y^*=f(x_1^*,x_2^*,\cdots,x_n^*)$,$\left(\frac{\partial f}{\partial x_i}\right)^*=\frac{\partial f}{\partial x_i}(x_1^*,x_2^*,\cdots,x_n^*)$,误差分析中$x_i^*$,$y^*$也可用其近似值代替。根据式(1.7)、式(1.8),绝对误差传播主要取决于$\left(\frac{\partial f}{\partial x_i}\right)^*$,相对误差传播主要取决于$\left(\frac{\partial f}{\partial x_i}\right)^*\frac{x_i^*}{y^*}$($i=1,2,\cdots,n$)。

两个数之间的算术运算是二元函数,根据式(1.7)、式(1.8)得到

$$\delta(a\pm b)=\delta(a)\pm\delta(b),\quad \delta_r(a\pm b)=[a/(a\pm b)]\delta_r(a)+[b/(a\pm b)]\delta_r(b);\tag{1.9}$$

$$\delta(ab)\approx b\delta(a)+a\delta(b),\quad \delta_r(ab)\approx\delta_r(a)+\delta_r(b);\tag{1.10}$$

$$\delta(a/b)\approx(1/b)\delta(a)-(a/b^2)\delta(b),\quad \delta_r(a/b)\approx\delta_r(a)-\delta_r(b)。\tag{1.11}$$

式(1.9)表明,在作加减法时,应尽量避免两个相近的数相减,否则会使相对误差增大,导致有效数字的严重损失。式(1.11)表明,在作除法时,应尽量避免用绝对值很小的数作除数,否则会使绝对误差增大。

例 1.4 设有长为L、宽为D的矩形场地,现测得L的近似值$\tilde{L}=(120\pm0.2)$m,D的近似值$\tilde{D}=(90\pm0.2)$m。求该场地面积的误差限和相对误差限。

解 设场地面积的近似值为$\tilde{S}=\tilde{L}\tilde{D}$,由于$|\delta(\tilde{L})|=|\delta(\tilde{D})|=0.2$m,根据式(1.10)得

$$|\delta(\tilde{S})|\approx|\tilde{L}\delta(\tilde{D})+\tilde{D}\delta(\tilde{L})|\leqslant|\tilde{L}||\delta(\tilde{D})|+|\tilde{D}||\delta(\tilde{L})|=(120+90)\text{m}\times0.2\text{m}=42\text{m}^2,$$

$$|\delta_r(\tilde{S})|\leqslant\frac{|\delta(\tilde{S})|}{|\tilde{S}|}=\frac{42}{120\times90}=0.0039=0.39\%。$$

舍入误差的影响很大程度上取决于计算设备的能力。对于通常的问题,如果我们采用双精度,舍入误差的影响一般不会太明显。但是在某些情况下,舍入误差的影响可能是至关重要的。

例 1.5 计算积分

$$I_n=\int_0^1x^n\mathrm{e}^{x-1}\mathrm{d}x,\quad n=0,1,\cdots,20。\tag{1.12}$$

解 这 21 个积分当然可以各自求解,但计算量比较大。现在我们构造一种递推算法,可以在求得其中一个积分的基础上非常简单地推出其他 20 个积分的值。根据分部积分法得

$$I_n = x^n \mathrm{e}^{x-1} \Big|_0^1 - \int_0^1 n x^{n-1} \mathrm{e}^{x-1} \mathrm{d}x = 1 - nI_{n-1},$$

从而得到递推公式

$$I_n = 1 - nI_{n-1}, \quad n = 1,2,\cdots,20。 \tag{1.13}$$

由于 $I_0 = 1 - 1/\mathrm{e}$，我们可以利用式(1.13)计算 $I_1, I_2, \cdots, I_{20}$。双精度计算结果见表 1-1。

因为当 $0 \leqslant x \leqslant 1$ 时，有

$$x^n/\mathrm{e} \leqslant x^n \mathrm{e}^{x-1} \leqslant x^n,$$

所以

$$\frac{1}{(n+1)\mathrm{e}} \leqslant I_n \leqslant \frac{1}{n+1}。 \tag{1.14}$$

虽然 I_0 有相当高的精度，并且递推公式(1.13)没有任何截断误差，$I_{18} \sim I_{20}$ 的计算结果却明显有很大误差。现在我们将递推公式(1.13)略作改造，成为

$$I_{n-1} = (1 - I_n)/n, \quad n = 20,19,\cdots,1。 \tag{1.15}$$

首先我们根据式(1.14)，取 $I_{20} = 0.5(1/\mathrm{e}+1)/(20+1)$，按式(1.15)递推计算结果见表 1-1。可以看到，尽管 I_{20} 的精度不算很高，递推计算结果 $I_2 \sim I_0$ 却有相当高的精度。表 1-1 告诉我们递推公式(1.13)和式(1.15)尽管在理论上完全等价，其实际计算效果却有天壤之别。为此我们有必要分析一下两种算法中误差的传播情况。

表 1-1　例 1.5 的计算结果

n	算法(1.13)		算法(1.15)	
0	0.632121		0.632121	
1	0.367879		0.367879	
2	0.264241		0.264241	
3	0.207277		0.207277	
4	0.170893		0.170893	
5	0.145533		0.145533	
6	0.126802	↓	0.126802	↑
7	0.112384		0.112384	
8	0.100932		0.100932	
9	0.091612		0.091612	
10	0.083877		0.083877	
11	0.077352		0.077352	
12	0.071773		0.071773	
13	0.066948		0.066948	
14	0.062731		0.062732	
15	0.059034		0.059018	
16	0.055459		0.055719	
17	0.057192		0.052773	
18	−0.02945		0.050086	
19	1.55962		0.048372	
20	−30.1924		0.032569	

设 I_n 有误差 δ_n，在算法(1.13)中，有 $\delta_n = -n\delta_{n-1}$，这样 $\delta_{20} = (20!)\ \delta_0 = 2.43 \times 10^{18} \delta_0$，