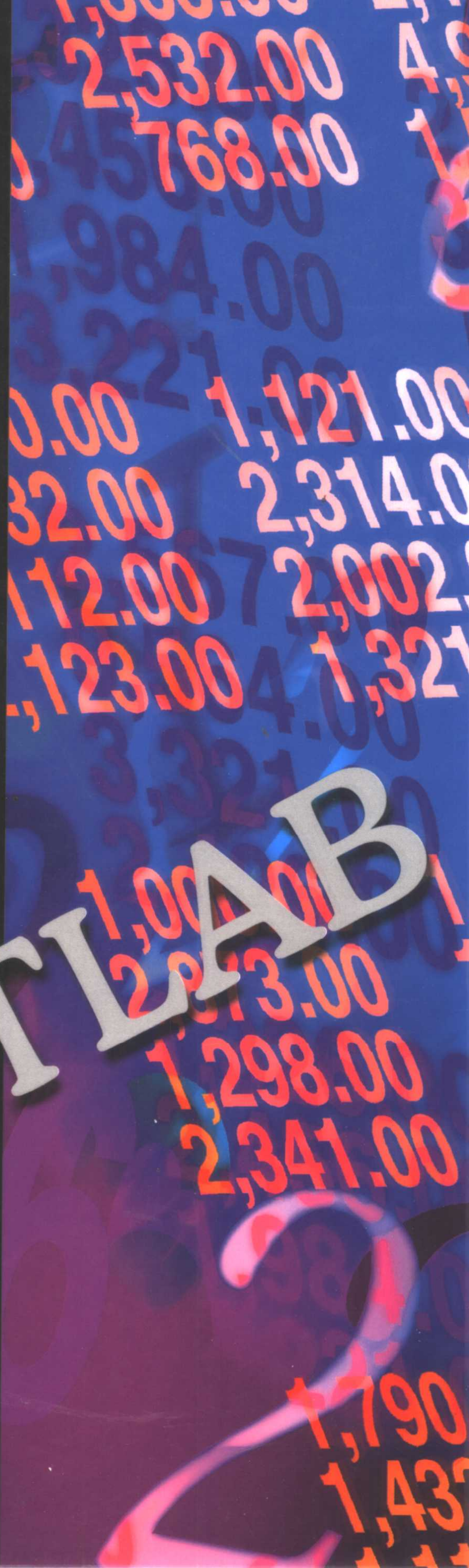
数值分析及其MATLAB实验

姜健飞 胡良剑 唐 俭 编著



科学出版社
www.sciencep.com

上海市研究生教学用书

数值分析及其 MATLAB 实验

姜健飞　胡良剑　唐　俭　编著

科 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，并配有习题解答和多媒体教学资料．

本书可作为理工科研究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，胡良剑，唐俭编著．—北京：科学出版社，2004.6
上海市研究生教学用书
ISBN 7-03-013446-X

Ⅰ．数…　Ⅱ．①姜…　②胡…　③唐…　Ⅲ．计算机辅助计算：数值计算-研究生-教材　Ⅳ．0241

中国版本图书馆 CIP 数据核字(2004)第 044433 号

责任编辑：张　臻／责任校对：连秉亮
责任印制：刘　学／封面设计：一　明

科学出版社 出版
北京东黄城根北街 16 号
邮政编码：100717
http://www.sciencep.com
上海交大印务有限公司印刷
科学出版社发行　各地新华书店经销
*
2004 年 6 月第 一 版　开本：787×1 092 1/16
2004 年 6 月第一次印刷　印张：18
印数：1—3 200　字数：420 000

定价：33.00 元

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. 本教材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：充分利用数学软件功能，加强实验环节，分层次处理. 一方面对于数值计算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.

本教材共有 8 章内容. 第一章是算法和误差的概念. 第二章至第六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的，而上机实验题适合用计算机完成. 第七章和第八章介绍了 MATLAB 的偏微分方程和最优化两个工具箱. 这两个问题尽管在传统的数值分析教材中不常见，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，而是主要提供读者在解决工程计算问题时作参考. 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 E 给出了全书的 MATLAB 命令或函数和 M 文件索引.

本书是作者在东华大学进行了四年教学改革实践基础上编写而成的. 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,可以完成第一至第六章以及附录 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

第一章 数值分析的基本概念

第一节 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,是研究算法的学问.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 $Ax = b$, 其中 A 为 3 阶可逆方阵, $x = (x_1, x_2, x_3)^T$;

(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 Cramer 法则,得 $x_1 = \frac{D_1}{D}$, $x_2 = \frac{D_2}{D}$, $x_3 = \frac{D_3}{D}$, 其中 $D = |A|$, D_j 为由 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 $y(x) = -x - 1 + e^x$.

例 1.2

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

(2) 求超越方程 $xe^x = 1$ 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 Cramer 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 $f(x)=xe^x-1$, 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,可借用例 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-莱布尼兹公式对该问题失效,因为无法找到被积函数 $f(x)=\frac{1}{\ln x}$ 的原函数,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,高等数学中的算法只能求解一些比较简单特殊的数学模型,而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(在第五章我们将有多种方法解此题);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,即使用每秒能做 1 万亿次乘法的计算机也需要 2 000 万亿年! 这就要求我们设计出更有效的算法,事实上,使用第二章的方法,我们用普通的微机可以在 1 秒钟内求解此题.

本课程将会给出解决例 1.2 这类问题的算法. 必须指出的是,实际问题远比例 1.2 复杂得多,这里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,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(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，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，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．数值算法是计算机算法中内容最为丰富、也是最基本的一部分内容，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，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．

第二节　误差分析的概念

在数值计算中，误差往往是不可避免的．怎样估计计算误差以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，$\delta x = x^* - x$，则 δx 为 x 的**误差**或**绝对误差**．在数值计算问题中，x^* 是未知的，从而 δx 也无法精确得到，但我们往往可以得到 δx 的绝对值上界，即 $|x^* - x| \leqslant \varepsilon$，则 ε 为 x 的**误差限**（或**精度**），应用中常记为 $x \pm \varepsilon$．误差与准确值的比值 $\delta_r x$ 称为 x 的**相对误差**，$\delta_r x = (x^* - x)/x^*$．如果 $|(x^* - x)/x^*| \leqslant \varepsilon_r$，则 ε_r 为 x 的**相对误差限**（或**相对精度**）．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，则

$$x = \pm 0.a_1a_2\cdots a_n\cdots \times 10^m. \tag{1.1}$$

其中，m 为整数，$a_1 \sim a_n$ 为 0～9 中的一个数字，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，则称近似值 x **准确到** 10^{-k} 位，并说 x 有 $m+k$ 位**有效数字**.

例 1.3 设圆周率 $\pi = 3.141\,592\,6\cdots$，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) $x_4 = 3.141\,4$.

解 (1) $\pi - x_1 = 0.159\,26\cdots \times 10^{-2}$，$(\pi - x_1)/\pi = 0.507\,2\cdots \times 10^{-2}$，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) $\pi - x_2 = 0.592\,6\cdots \times 10^{-3}$，$(\pi - x_2)/\pi = 0.188\,7\cdots \times 10^{-3}$，由于 $x_2 = 0.314\,1 \times 10^1$，$|\pi - x_2| \leqslant 0.5 \times 10^{-2}$，从而 x_2 有 3 位有效数字；

(3) $\pi - x_3 = -0.407\,3\cdots \times 10^{-3}$，$(\pi - x_3)/\pi = -0.129\,6\cdots \times 10^{-3}$，由于 $x_3 = 0.314\,2 \times 10^1$，$|\pi - x_3| \leqslant 0.5 \times 10^{-3}$，从而 x_3 有 4 位有效数字；

(4) $\pi - x_4 = 0.192\,6\cdots \times 10^{-3}$，$(\pi - x_4)/\pi = -0.613\cdots \times 10^{-4}$，由于 $x_4 = 0.314\,14 \times 10^1$，$|\pi - x_1| \leqslant 0.5 \times 10^{-3}$，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**.

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 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

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

其截断误差可以用余项公式

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

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,我们称式(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:

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

这称为**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 $(j = 2, \cdots, t)$. 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,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(图 1-1). 单精度 $t = 23$, 约相当于十进制 7 位有效数字. 双精度 $t = 52$, 约相当于十进制 15 位有效数字. 机器数中单精度和双精度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 $2^{-1\,024} \sim 2^{1\,024}$(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.

单精度

1 位	阶码 7 位	1 位	尾数 23 位
(阶码正负号)		(尾数正负号)	

双精度

1 位	阶码 10 位	1 位	尾数 52 位
(阶码正负号)		(尾数正负号)	

图 1-1　机器数的表示

由于机器数字长的限制而产生的误差称为**舍入误差**. 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,因而一般都有舍入误差. 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.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?

如果 $f(x_1, x_2, \cdots, x_n)=\sum\limits_{i=1}^{n} a_i x_i$(即线性函数), 那么

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

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,

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

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

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)和(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)和(1.8)得到

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

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

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

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,测得 L 的近似值 $\tilde{L}=(120+0.2)\mathrm{m}$, D 的近似值 $\tilde{D}=(90 \pm 0.2)\mathrm{m}$, 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 $|\delta \tilde{L}|=|\delta \tilde{D}|=0.2\ \mathrm{m}$, 根据式(1.10)得

$$|\delta \tilde{S}| \approx |\tilde{L}\delta \tilde{D}+\tilde{D}\delta \tilde{L}| \leqslant |\tilde{L}||\delta \tilde{D}|+|\tilde{D}||\delta \tilde{L}|=(120+90)\times 0.2=42\ \mathrm{m}^2,$$

$$|\delta_r \widetilde{S}| \leqslant \frac{|\delta \widetilde{S}|}{|\widetilde{S}|} = \frac{42}{120 \times 90} = 0.0039 = 0.39\%.$$

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

$$I_n = \int_0^1 x^n e^{x-1} dx, \ n = 0, 1, \cdots, 20. \tag{1.12}$$

解 这 21 个积分可以各自求解,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

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

从而得到递推公式

$$I_n = 1 - nI_{n-1}, \ n = 1, \cdots, 20. \tag{1.13}$$

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

因为

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

所以

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

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)略做改造,成为

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

首先我们根据式(1.14),取 $I_{20} = 0.5(1/e + 1)/21$, 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(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.

设 I_n 有误差 δ_n, 在算法(1.13)中,有 $\delta_n = -n\delta_{n-1}$, 这样 $\delta_{20} = (20!)\delta_0 = 2.43 \times 10^{18}\delta_0$, 虽然 δ_0 很小,但 δ_{20} 却很大. 而对于算法(1.15),有 $\delta_{n-1} = -\delta_n/n$, 这样 $\delta_0 = \delta_{20}/(20!) = 4.1103 \times 10^{-19}\delta_{20}$, 可见误差 δ_{20} 会在计算过程中逐步缩小.

如果在一个数值算法中,舍入误差会在计算过程中恶性增大,我们就称该算法是**数值不稳定**的,否则称其为**数值稳定**的. 例如,当我们用绝对值很小的数作除数,则算法就是数值不稳定的.

表 1-1 例 1.5 的计算结果

n	算法(1.13)	算法(1.14)
0	0.632 121	0.632 121
1	0.367 879	0.367 879
2	0.264 241	0.264 241
3	0.207 277	0.207 277
4	0.170 893	0.170 893
5	0.145 533	0.145 533
6	0.126 802	0.126 802
7	0.112 384	0.112 384
8	0.100 932	0.100 932
9	0.091 612	0.091 612
10	0.083 877	0.083 877
11	0.077 352	0.077 352
12	0.071 773	0.071 773
13	0.066 948	0.066 948
14	0.062 731	0.062 732
15	0.059 034	0.059 018
16	0.055 459	0.055 719
17	0.057 192	0.052 773
18	−0.029 45	0.050 086
19	1.559 62	0.048 372
20	−30.192 4	0.032 569

4. 数据误差和病态问题

工程问题的数学模型中的参数往往由实验和测量数据取得，所以不可避免地存在数据误差. 正常情况下，这些误差对于解的影响是在允许范围内的. 但对于某些问题，数据误差可能会产生严重影响.

例 1.6 解线性方程组

$$\begin{cases}\dfrac{49}{36}x_1+\dfrac{3}{4}x_2+\dfrac{21}{40}x_3=\dfrac{949}{360}\\[2mm] \dfrac{3}{4}x_1+\dfrac{61}{144}x_2+\dfrac{3}{10}x_3=\dfrac{1\,061}{720}\\[2mm] \dfrac{21}{40}x_1+\dfrac{3}{10}x_2+\dfrac{769}{3\,600}x_3=\dfrac{3\,739}{3\,600}.\end{cases} \tag{1.16}$$

解 容易验证，$x_1=x_2=x_3=1$ 是式(1.16)的惟一准确解. 如果我们将系数保留 4 位有效小数，得到方程组

$$\begin{cases}1.361\,1x_1+0.750\,0x_2+0.525\,0x_3=2.636\\ 0.750\,0x_1+0.4236x_2+0.300\,0x_3=1.474\\ 0.525\,0x_1+0.300\,0x_2+0.213\,6x_3=1.039,\end{cases} \tag{1.17}$$

计算结果为 $x_1=1.220$，$x_2=-0.308\,4$，$x_3=2.298$，与原方程组的解相比已经面目

全非.

与例 1.5 不同的是，这里的误差与算法无关，因为这个解对于式(1.17)有很高的精度. 问题出在式(1.16) 本身对数据的误差极其敏感. 尽管数据只有很小的变化，却导致解产生了很大的变化. 我们称这类问题为**病态问题**.

病态问题也可以用一次近似来作分析. 考虑函数 $y = f(x_1, x_2, \cdots, x_n)$，$x_i^*$ 是模型参数 x_i 的准确值，$\tilde{x}_i$ 是 x_i 的近似值，$i = 1, 2, \cdots, n$，$y^* = f(x_1^*, x_2^*, \cdots, x_n^*)$ 是模型的准确解，$\tilde{y} = f(\tilde{x}_1, \tilde{x}_2, \cdots, \tilde{x}_n)$ 是近似模型的准确解. 根据式(1.7)和(1.8)，$\left(\frac{\partial f}{\partial x_i}\right)^*$ 和 $\left(\frac{\partial f}{\partial x_i}\right)^* \frac{x_i^*}{y^*}$ 的绝对值反映了解对参数数据误差的敏感性程度，我们称 $\left|\left(\frac{\partial f}{\partial x_i}\right)^*\right|$ 和 $\left|\left(\frac{\partial f}{\partial x_i}\right)^* \frac{x_i^*}{y^*}\right|$ 为该问题的**条件数**. 条件数很大的数值问题称为**病态问题**.

第三节 数值算法设计的注意事项

1. 数值算法评价的基本原则

设计一个数值算法主要是处理好计算精度和计算速度两个问题. 计算精度问题就是计算结果的可靠性问题，一般来说可靠的算法应该具有收敛性和数值稳定性. 计算速度问题就是算法的效率问题，包含计算量和存储量两个方面. 时空两个方面的复杂性都会影响计算速度. 通常影响计算速度最关键的因素是算法的收敛速度. 通俗地讲，一个好的算法就是又准又快的算法.

2. 数值算法设计的注意事项

(1) 要保证算法具有收敛性和较高的收敛速度

(2) 要保证算法具有数值稳定性

(3) 小心处理病态问题

(4) 注意影响收敛速度的细节

一般来说，加减法比乘法快，乘法比除法快. 如使用 $x+x$、$x\times x$、$0.5\times x$ 就分别比 $2\times x$、x^2、$x/2$ 的计算速度快.

例 1.7(秦九韶算法或 Horner 法) 考虑多项式 $a_0 + a_1x + \cdots + a_nx^n$ 的算法设计.

解 由于计算机作加减法比乘法快得多，我们只考虑乘法的计算量. 如果直接计算，需要

$$0 + 1 + \cdots + n = n(n+1)/2$$

次乘法. 如果使用递推算法

$$t_0 = 1,\ p_0 = a_0,\ t_k = xt_{k-1},\ p_k = p_{k-1} + a_kt_k,$$

其中，$k = 1, \cdots, n$，则只有 $2n$ 次乘法，计算量减少了 n 的一个阶次. 如果使用秦九韶算法(或称 Horner 算法)

$$p_0 = a_n,\ p_k = xp_{k-1} + a_{n-k},$$

其中 $k = 1, \cdots, n$，则只有 n 次乘法，计算量进一步减少了一半.

(5) 注意影响数值稳定性的细节

例如，应尽量避免相差悬殊的数加减、两个相近的数相减以及绝对值很小的数作除数等.

例 1.8(相近的数相减) 已知 $x = \pi \times 10^{-8}$，考虑采用双精度(15 位十进制)计算

$$y = \frac{1}{1+2x} - \frac{1-x}{1+x} \tag{1.18}$$

的算法设计(其真实解 $0.197\,392 \times 10^{-14}$).

解 直接用式(1.18)求解得 $0.199\,84 \times 10^{-14}$，可见只有 2 位有效数字. 这是由于 x 接近于 0，式(1.18)涉及两个近似数的减法，容易造成相对误差增大. 我们将其改为

$$y = \frac{2x^2}{(1+2x)(1+x)}, \tag{1.19}$$

计算得 $0.197\,39 \times 10^{-14}$，至少有 5 位有效数字，效果就比较好.

例 1.9(大数吃小数) 考虑采用单精度(7 位十进制)计算的算法设计：

$$y = 123\,456 + \sum_{i=1}^{1\,000} x_i,\ 0.01 \leqslant x_i \leqslant 0.04. \tag{1.20}$$

解 显然，不等式 $123\,466 \leqslant y \leqslant 123\,496$ 成立. 但如果我们按式(1.20)的顺序求解，结果为 123 456，也就是后面 1 000 项"小数"x_i 被"大数""123 456""吃掉了". 这是因为计算机在作加减法时，总是先将各项阶码统一到最大的阶码，即

$$123\,456 \to 0.123\,456\,0 \times 10^6,$$

$$0.01 \to 0.1 \times 10^{-1} \to 0.000\,000\,01 \times 10^6 \to 0.000\,000\,0 \times 10^6.$$

由于两个数的阶码相差很大，0.01 的阶码上升到 10^6 时，其尾数很小而超出了字长范围，就被舍去了. 改进的算法是先求 $\sum_{i=1}^{1\,000} x_i$，由于 x_i 数量级相近，不会发生"大数吃小数"，而 1 000项小数的总和超过 10，也已经不太小，就不至于被"123 456""吃掉".

(6) 节省存储空间

例如，考虑迭代

$$p_0 = a_n,\ p_k = xp_{k-1} + a_{n-k},\ k = 1 \sim n.$$

如果将 $p_0, p_1, \cdots, p_n$ 作为一个数组，需要 $n+1$ 个实数单元，但其实我们并不需要 $p_0, p_1, \cdots, p_{n-1}$，且可以更新，所以 $p_0, p_1, \cdots, p_n$ 可以共用一个实数变量 p，这样只需一个实数单元.

(7) 避免死循环

数值算法很多都具有收敛性问题，一旦出现不收敛或收敛速度太慢，普通 while 循环语句往往会进入死循环，所以使用 for 语句更可靠. 如果使用 while 语句，最好设置一个循