高等学校教材

基础课程系列

数值分析教程

刘长安　编著

Fundamental Courses

西北工业大学出版社

高等学校教材

数值分析教程

刘长安　编著

西北工业大学出版社

【内容简介】 本书首先介绍了数值分析研究的内容，计算机数系的特点，误差理论，数值问题的适定性、条件以及算法稳定性等概念，然后介绍了在计算机上求解各种数值问题的常用算法，对算法的基本原理、收敛性、收敛速度、误差估计、数值稳定性以及算法的优劣给出了详细的理论分析。全书共分十章，包括引论、解线性代数方程组的直接法、方程组的条件和不相容方程组求解、解线性方程组的迭代法、矩阵特征问题的求解、插值法、函数逼近、数值积分和数值微分、非线性方程(组)的求解、常微分方程的数值解法。

图书在版编目 (CIP) 数据

数值分析教程/刘长安编著．—西安：西北工业大学出版社，2005.8
ISBN 7-5612-1988-1

Ⅰ．数… Ⅱ．刘… Ⅲ．数值计算 Ⅳ．0241

中国版本图书馆 CIP 数据核字 (2005) 第 092192 号

出版发行：西北工业大学出版社
通信地址：西安市友谊西路 127 号 邮编：710072
电 话：(029)88493844 88491757
网 址：www.nwpup.com
印 刷 者：陕西丰源印务有限公司
开 本：787 mm×1 092 mm 1/16
印 张：23.25
字 数：565 千字
版 次：2005 年 8 月第 1 版 2005 年 8 月第 1 次印刷
定 价：29.00 元

前　言

上世纪中期数字式计算机的发明无疑是人类最伟大的成就之一，计算机的出现对人类产生了极其深刻和深远的影响，改变着并继续改变着人类的生存条件和生活方式. 计算机在本世纪将会进一步显示它的巨大威力和作用.

人类在不断改变自身生存条件和生活方式的同时，创造了灿烂的文化和人类的文明. 作为人类文化以及科学技术基础的数学已有几千年的历史，近几个世纪有了长足的发展，但作为数学中最重要的计算方面却苦于计算工具的落后长期停滞不前，严重地影响了社会的进步和科学技术甚至数学本身的发展，这一状况在计算机的出现后才得到彻底改变.

数值分析是研究数值问题算法的一门学科，又称为数值计算方法或数值方法，随着计算机的广泛应用，数值分析这门学科焕发了勃勃生机，和计算有关的各种学科相继产生，科学研究和工程设计从模型试验向数值计算方面转变，数值分析给科学研究和工程设计提供了新的手段，数值方法与传统的理论方法、实验方法一起已成为现代科学研究的重要支柱.

目前数值分析是理工科所有专业的大学生和研究生的一门重要的必修课. 这门课不但要求学生掌握一些解决实际问题中数值计算的常用方法，学会误差分析和误差估计，并对实际计算中出现的异常现象能给出科学的分析和处理. 毫无异议，数值分析的重要性在今后将会更加突出，在新世纪的高等院校中加强数值分析这门学科的教学和研究势在必行，为此编写一本具有时代特色的教材十分必要.

本书作为讲义在过去给本院研究生和本科生使用过多年，为适应新世纪的要求，在此基础上进行了较大的修改和完善，使它能适应不同需要，它既可以作为理工科非计算数学专业的本科生（带“*”号标记的章节作为选学内容）和研究生学习“数值分析”这门课所用的教材，还能提供给任课教师备课参考以及作为工程技术人员知识更新的资料.

本书内容包括在计算机上对来自科学研究和工程技术实际领域中出现的各种常用数值问题求解算法的介绍，以及对算法的基本原理，算法的可靠性，算法的优劣的理论分析. 阅读本书需要具备高等数学和线性代数的基本知识以及在计算机上编程上机的初步经验.

虽然本书大部分内容是数值分析的传统内容，但在材料的组织和处理

上，在理论的推导和分析上却有它的鲜明特色，不需要读者具有较深的数学知识的情况下能给出理论较为严格和完整的分析，对书中涉及的内容能尽可能详尽和深入地讨论，理论的推导和证明尽可能简明和新颖，材料和内容的安排尽可能达到逻辑上的完备. 精选的习题分为二类，一类侧重于数值计算，通过演算使读者熟练掌握数值计算的基本方法，另一类侧重于理论分析，通过它能使读者更好地掌握数值分析的基本理论或对本书中的内容进行理论上的加深和扩充.

读者不难发现，书中有一部分内容不常见于其他同类书籍，这一部分是作者自己多年教学的一些经验和体会. 由于作者理论水平和实践经验有限，本书内容上不足之处甚至谬误难以避免，恳请广大读者给予批评指正.

本书的出版得到西安工业学院研究生部的领导与同志的大力支持，西北工业大学出版社的同志对本书提出一些有益的建议，在此对他们的支持、鼓励和帮助表示衷心感谢.

编著者

2005 年 4 月 13 日于西安工业学院

目 录

第一章 引 论

1.1 数值分析研究的内容

数值分析又称为数值计算方法或数值方法，是研究数值问题算法的一门学问，它研究如何借用计算工具求得数学问题的数值解答. 这里的数学问题是指数值问题，所谓数值问题，是给出一组数值型数据，通常是一些实数，称为初始数据，去求另外一组数值型数据，问题的本身反映了这两组数据之间的一定关系. 函数计算、方程求根就是数值问题的典型例子. 除此之外，数学中还存在着大量的非数值问题，如定理的证明、几何的作图和组态的枚举等问题都不是研究的对象.

自有数学以来就有关于数值计算方面的研究，古代巴比伦人(公元前 2000 年左右)就有了关于数的运算和 2 次方程求解的研究，我国古代数学家刘徽(公元 263 年)利用割圆术求得圆周率 π 的近似值，而后祖冲之(公元 400 多年)求得圆周率的高精度的值都是数值计算方面的杰出成就. 数值分析是在解决数值问题的长期实践过程中逐步形成和发展起来的. 但在计算机出现以前，它的理论和方法发展十分缓慢，甚至长期停滞不前，由于受到当时计算工具的限制，无法进行大量的复杂的计算. 数值分析的进一步发展是和计算机的出现以及它的广泛使用分不开的.

无论计算机在数据处理、信息加工等方面取得了多么辉煌的成就，科学计算始终是计算机应用的一个重要方面. 而数值分析是计算机进行科学计算的依据，它不但为科学计算提供了理论基础，并为科学计算提供了大量行之有效的数值问题的算法. 另一方面，计算机在进行科学计算的过程中，又不断给数值分析的理论提出一些新的问题，促进了它的进一步发展.

由于计算机对数值分析这门学科的推动和影响，使数值分析的重点转移到用计算机算题这方面上来. 现代的数值分析的理论和方法主要是面向计算机的，研究和寻求适合于计算机上各种数值问题的算法是数值分析这门学科研究的主要内容.

一般地讲，算法就是解决问题的步骤. 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，他清醒地认识到了计算的本质，他的 Turing 机和现代的数字式计算机在原理和功能上没有本质的区别.

计算机虽是运算速度极高的现代化的计算工具，但是它本质上仅仅能完成一系列具有一定位数的基本算术运算和逻辑运算，所以进行数值计算首先要将各种类型的数值问题归结为一系列计算机能执行的基本运算. 通常的数值问题是在实数范围内提出的，而计算机所能表示的仅仅是有限位小数，误差不可避免，这些误差对计算结果能起什么影响是需要考虑的. 如果给出一种算法，在计算机上进行计算时，误差在成千上万次的运算过程中得不到控制，初始

数据的误差，由中间结果的舍入产生误差，这些误差在计算过程中的积累越来越大，“淹没”了真值，这样的计算结果毫无意义，这种算法是不可靠的. 现代的计算机无论在运算速度上还是存储能力上是传统计算工具无法比拟的，但即使这样，在设计算法时，对运算次数和所需存储量的大小要给予足够重视. 实际上存在大量这样的问题，所提供的解决这些问题的算法因为运算次数大得惊人，就是利用高速运行的计算机也不能在有效时间内给出问题的答案.

在进行数值计算时，一定要采用那些运算次数较少，所需存储量较小，逻辑结构简单的可靠算法. 数值分析的理论要对这些问题给出分析，在此基础上为数值问题提供良好的算法.

1.2 数在计算机中的表示

研究适合于计算机上使用的数值计算方法，一定要对数在计算机中的表示有一个初步了解. 数在计算机中的表示有两种：定点表示和浮点表示.

计算机使用的数常是 2 进制(数的基数为 2)或是它的变形 8 进制和 16 进制，很少直接使用 10 进制，下面采用大家熟悉的 10 进制讨论，并不失去一般性.

一、定点表示

定点表示是固定小数点位置的一种表示方法，它的每个表示就是一个定点数. 它的一般形式为

$$x = \mathrm{sign}(x)a_{m-1}a_{m-2}\cdots a_1a_0.a_{-1}a_{-2}\cdots a_{-n} \tag{1.2.1}$$

其中 $\mathrm{sign}(x)$ 表示 x 的符号，$a_i(-n \leqslant i \leqslant m-1)$ 表示 0 ～ 9 的数，它有 m 位整数，n 位小数，$m+n$ 称为定点数的字长. 这时 x 代表的值为

$$x = \mathrm{sign}(x)\sum_{i=-n}^{m-1} a_i \times 10^i \tag{1.2.2}$$

例如 8 位定点数，其中 $m = n = 4$，有下列一些数：

$$0201.5760, \quad -0000.3785, \quad 1462.0000$$

定点表示的数范围很小，如在上例中，绝对值最大和最小的非零数是 ± 9999.9999 和 ± 0000.0001，因此在计算机中更常用数的浮点表示.

二、浮点表示

浮点表示的数称为浮点数，浮点数的一般形式为

$$x = \mathrm{sign}(x)(0.a_1a_2\cdots a_n) \times 10^s \tag{1.2.3}$$

其中 $\mathrm{sign}(x)$ 表示 x 的符号，$a_i(1 \leqslant i \leqslant n)$ 表示 0 ～ 9 的数字. $0.a_1a_2\cdots a_n$ 称为浮点数的尾数，s 称为其阶码，n 称为浮点数 x 的字长或机器字长，$m \leqslant n \leqslant M$，其中 M，m 称为阶码 s 的上、下界，对一台确定的计算机是给定的. 式(1.2.3) 中的 x 代表的值为

$$x = \mathrm{sign}(x)\sum_{i=1}^{n} \frac{a_i}{10^i} \times 10^s \tag{1.2.4}$$

例如 $n = 4$，$-99 \leqslant s \leqslant 99$，有如下一些浮点数：

$$0.1234 \times 10^5, \quad 0.0123 \times 10^6, \quad -0.5831 \times 10^{-4}$$

$a_1 \neq 0$ 的浮点数称为规格化的浮点数. 在上面的表示中，小数点的位置由 s 的值确定，这

种小数点的位置可以改变的数的表示方法称为浮点表示. 因此对同一个数,它的浮点表示一般不是惟一的,例如

$$0.1200\times10^{5}=0.0120\times10^{6}=0.0012\times10^{7}$$

但是规格化的浮点数表示是惟一的,上面三个数中仅有 0.1200×10^{5} 为规格化浮点数. 在上例中,绝对值最大和最小的非零浮点数分别为 $\pm0.999\,9\times10^{99}$ 和 $\pm0.000\,1\times10^{-99}$. 由此可见,浮点表示数的范围较定点表示大得多.

三、机器数系

计算机所能表示的全体浮点数构成的集合,称为计算机数系,又称机器数系. 记为

$$F=\{x\mid x=\mathrm{sign}(x)(0.a_1a_2\cdots a_n)\times10^{s},m\leqslant s\leqslant M\} \tag{1.2.5}$$

式中 $a_i(1\leqslant i\leqslant n)$ 为 0 ~ 9 的数字,$\mathrm{sign}(x)\in\{+,-\}$,$n,M,m$ 为确定的值.

计算机数系和实数系有很大区别,实数系是一个连续、稠密的无限集合,对算术运算封闭,运算服从一般的运算法则. 然而计算机数系是一个离散的有限集合,它是有理数集合的真子集,对算术运算一般不封闭,一般的运算法则也并非总是成立的. 实数均匀分布在实数轴上,而计算机的浮点数在数轴上分布是不均匀的.

计算机的浮点数字长 n,阶码的上、下界 M 和 m 的值给定,计算机中所有十进制规格化浮点数(包括零)的个数为

$$18\times10^{n-1}\times(M-m+1)+1$$

一般基数为 β 的规格化浮点数(包括零)的个数为

$$2(\beta-1)\beta^{n-1}(M-m+1)+1$$

设 $n=4$,$-99\leqslant s\leqslant99$,$x_1=0.1234\times10^{5}$,$x_2=0.5678\times10^{3}$,显然 $x_1,x_2\in F$,但 $x_1+x_2=0.129\,078\times10^{5}\notin F$,$F$ 对加法不封闭.

在上述假设下,浮点数 0.1234×10^{s} 和 0.1235×10^{s} 为相邻的两个浮点数,它们之间再没有其他 F 中的数,但它们之间的间隔(距离)却依赖于 s 的值,s 的值大,之间的间隔就大,s 的值小,之间的间隔就小,实际上两数距离为 10^{s-4},这说明浮点数分布是不均匀的.

今后进行的数值计算就是在这样一个残缺不全的数系上进行的,认识到这一点是重要的.

1.3 误差理论

一、误差的来源

在用计算机解决实际问题进行科学计算时,误差是不可避免的. 按误差产生的原因可将实际中常遇到的误差分为下列几种.

1. 模型误差

在解决实践中提出的问题,常常是建立该问题的数学模型,任何模型都是实际问题的近似表述,我们往往抓住实际问题的主要因素,而略去次要的方面,把问题理想化、简单化,这样建立的简单数学模型和实际问题存在一些差异,这就是模型误差.

2. 观测误差

在进行科学技术的计算时,问题提供的初始数据往往是经过观察测量得到的,人们受着测

量工具的局限，得到的往往是近似值，这种误差称为观测误差.

3. 截断误差

数学中很多问题的解，从理论上分析，需要通过无限次运算才能得到其精确值，而人们不管利用什么计算工具只能完成有限次运算. 像这样通过有限次算术运算得到的数值问题的近似解的方法称为数值方法. 由数值方法产生的误差称为方法误差或截断误差.

例如：已知 $x>0$，求 e^{-x}，利用 e^{-x} 的 Taylor 展开式

$$e^{-x}=1-x+\frac{x^2}{2}-\frac{x^3}{6}+\cdots$$

这是一个无穷级数，需要无限次运算，可取部分和为

$$s(x)=1-x+\frac{x^2}{2}-\frac{x^3}{6}+\cdots+(-1)^n\frac{x^n}{n!}$$

作为 e^{-x} 的近似值，这样差

$$e^{-x}-s(x)=(-1)^{n+1}\frac{e^{-\xi}x^{n+1}}{(n+1)!},\quad 0<\xi<x$$

称为用 $s(x)$ 近似 e^{-x} 的截断误差.

4. 舍入误差

从 1.2 节知道计算机数系是一个有限集合，是有理数的真子集. 在实际问题中出现的很多数据不在计算机数系 F 中，无理数 $\sqrt{2},\sqrt{3},\pi\notin F$，就像 $1/3,1/7$ 这样的有理数也不在 F 中，因此需要用 F 中的数近似表示它们，这样产生的误差称为舍入误差. 关于舍入误差后面还要讨论.

二、绝对误差和相对误差

1. 绝对误差

给定一实数，它的近似数为 $\tilde{x}$，$\tilde{x}-x$ 反映了近似数和精确值差异的大小，因此称 $\varepsilon(x)=\tilde{x}-x$ 为近似数 $\tilde{x}$ 的绝对误差.

精确值往往是无法知道的，因此近似数的绝对误差也无法得到，但有时却能估计出 $\varepsilon(x)$ 的绝对值的上界，如果有一正数 η，使

$$|\varepsilon(x)|\leqslant\eta$$

则称 η 为 $\tilde{x}$ 的绝对误差界，此时

$$x-\eta\leqslant\tilde{x}\leqslant x+\eta$$

将上式简记为 $\tilde{x}=x\pm\eta$.

2. 相对误差

绝对误差通常还不能完全反映近似数的精确程度，它还依赖于该数本身的大小，因此引进相对误差的概念，近似数的相对误差定义为

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

和绝对误差情况一样，引进相对误差界的概念，如果有一正数 δ，使

$$|\varepsilon_r(x)|\leqslant\delta$$

则称 δ 为 $\tilde{x}$ 的相对误差界，将 $\varepsilon_r(x)$ 简记为 ε，根据相对误差和其界的定义可得

$$\tilde{x}=x(1+\varepsilon),\quad |\varepsilon|\leqslant\delta$$

3. 向量的误差

实际问题中给出的数据(初始数据)往往不是孤立的一个数,有时给出的是相关联的一组实数 $x_1, x_2, \cdots, x_n$,为了统一处理,作为一个整体将它们看成 n 维空间 $\boldsymbol{R}^n$ 的向量或点,记为 $\boldsymbol{x} = [x_1\ x_2\ \cdots\ x_n]^{\mathrm{T}}$,设 $\tilde{x}_i$ 是 $x_i (1 \leqslant i \leqslant n)$ 的近似数,$\tilde{\boldsymbol{x}} = [\tilde{x}_1\ \tilde{x}_2\ \cdots \tilde{x}_n]^{\mathrm{T}}$ 是 $\boldsymbol{x}$ 的近似向量,反映这两组数据的整体误差可用 $\tilde{\boldsymbol{x}} - \boldsymbol{x}$ 的 Euclid 范数,即

$$\| \tilde{\boldsymbol{x}} - \boldsymbol{x} \| = \left[\sum_{i=1}^{n} (\tilde{x}_i - x_i)^2 \right]^{1/2}$$

来表示,它实际上是向量 $\tilde{\boldsymbol{x}}$ 和 $\boldsymbol{x}$ 在 $\boldsymbol{R}^n$ 空间的 Euclid 距离,用 $\| \tilde{\boldsymbol{x}} - \boldsymbol{x} \|$ 的值表示 $\tilde{\boldsymbol{x}}$ 的绝对误差,类似地用 $\| \tilde{\boldsymbol{x}} - \boldsymbol{x} \| / \| \boldsymbol{x} \|$ 表示 $\tilde{\boldsymbol{x}}$ 的相对误差,当 $n = 1$ 时就化为上面通常单个数的误差了.

除了上面利用 Euclid 范数来表示向量的大小外,还可以利用其他范数,后面将会看到它的推广.

三、数据在机器数系中的近似表示和近似数的有效数位

1. 数据的近似表示和舍入误差

计算机数系 F 既然是有理数的真子集,在利用计算机进行运算时,初始数据和中间结果都可能不在 F 中,这样就需要用 F 中的数近似地表示相应的数据,设任意实数 $x \notin F$,要用 F 中的一个浮点数作为 x 的近似,记这个浮点数为 $\mathrm{fl}(x)$,它应有最佳逼近的性质,即应有

$$| x - \mathrm{fl}(x) | = \min_{g \in F} | x - g | \tag{1.3.1}$$

在 F 中的所有数中 $\mathrm{fl}(x)$ 和 x 的误差最小.

如果 x 的绝对值大于 F 中的最大正数,或小于 F 中最小正数,就会发生上溢或下溢现象.在下面的讨论中,假定给出的数据和中间结果都在 F 的范围内,而不发生溢出现象.

假设计算机浮点数字长为 n,任给实数 x,将 x 可写成下面惟一形式:

$$x = \mathrm{sign}(x) a \times 10^s$$

其中 $a = 0.a_1 a_2 \cdots a_n a_{n+1} \cdots$,$a_i$ 是 $0 \sim 9$ 的数,并规定 $a_1 \neq 0$,显然 $a \geqslant 10^{-1}$. 设

$$a' = 0.a_1 a_2 \cdots a_n, \quad a'' = 0.a_1 a_2 \cdots a_n + 10^{-n}$$

$$x' = \mathrm{sign}(x) a' \times 10^s, \quad x'' = \mathrm{sign}(x) a'' \times 10^s$$

显然 $x', x'' \in F$,且实数 x 位于这两个相邻的浮点数 x', x'' 之间,为了满足最佳逼近性质式(1.3.1),按照下面"四舍五入"的原则选择 x' 和 x'' 之一作为 x 的浮点近似值 $\mathrm{fl}(x)$,即当 $0 \leqslant a_{n+1} \leqslant 4$,则 $\mathrm{fl}(x) = x'$,当 $5 \leqslant a_{n+1} \leqslant 9$,则 $\mathrm{fl}(x) = x''$. 由此可知

$$\min\{ | a - a' |, | a - a'' | \} \leqslant \frac{1}{2} \times 10^{-n} \tag{1.3.2}$$

此时绝对误差满足下式:

$$| \varepsilon(x) | = | \mathrm{fl}(x) - x | \leqslant \frac{1}{2} \times 10^{-n} \times 10^s = \frac{1}{2} \times 10^{-n+s} \tag{1.3.3}$$

而相对误差满足

$$| \varepsilon_r(x) | = \left| \frac{\varepsilon(x)}{x} \right| \leqslant \frac{\frac{1}{2} \times 10^{-n+s}}{10^{-1} \times 10^s} = \frac{1}{2} \times 10^{-(n-1)} \tag{1.3.4}$$

设 $\mathrm{eps} = 0.5 \times 10^{-(n-1)}$,eps 称为计算机精度,它作为浮点数的相对误差界,仅和浮点数的字长 n 有关. 由此可见,机器数系虽然分布不均匀,但按上述原则用机器数近似表示任意实数其相

对误差的分布却是均匀的. 由式(1.3.4)可得

$$\mathrm{fl}(x) = x(1+\varepsilon), \quad |\varepsilon| \leqslant \mathrm{eps}$$

在计算机中,有时采用"切断"的原则来确定 $\mathrm{fl}(x)$,此时,无论 a_{n+1} 的值为多少,一律规定 $\mathrm{fl}(x) = x'$,这样确定的 $\mathrm{fl}(x)$ 不再具有最佳逼近性质式(1.3.1),此时绝对误差和相对误差分别满足下式:$|\varepsilon(x)| \leqslant 10^{-n+s}$,$|\varepsilon_r(x)| \leqslant 10^{-(n-1)}$,它的误差界恰是上面"舍入"得到的近似数误差界的2倍,因此计算机中不常采用这种方法.

在计算机上进行计算时,计算机会自动地将初始数据和中间结果表示为 F 中的浮点数,因此在 F 中进行算术运算和通常的算术运算是有区别的. 为了和通常运算区分开来,F 中的加法和乘法运算在这里暂时采用运算符号 $\oplus$,$\otimes$,实际上

$$x_1 \oplus x_2 = \mathrm{fl}(x_1 + x_2), \quad x_1 \otimes x_2 = \mathrm{fl}(x_1 \times x_2)$$

设 $n = 4$,$-99 \leqslant s \leqslant 99$,$x_1 = 0.123\,4 \times 10^5$,$x_2 = 0.567\,8 \times 10^2$,$x_3 = 0.878\,7 \times 10^2$,此时

$$x_1 \oplus x_2 = \mathrm{fl}(x_1 + x_2) = 0.124\,0 \times 10^5, \quad x_2 \oplus x_3 = \mathrm{fl}(x_2 + x_3) = 0.144\,7 \times 10^3$$

$$(x_1 \oplus x_2) \oplus x_3 = \mathrm{fl}((x_1 \oplus x_2) + x_3) = 0.124\,9 \times 10^5$$

$$x_1 \oplus (x_2 \oplus x_3) = \mathrm{fl}(x_1 + (x_2 \oplus x_3)) = 0.124\,8 \times 10^5$$

故有 $(x_1 \oplus x_2) \oplus x_3 \neq x_1 \oplus (x_2 \oplus x_3)$,由此可见,在 F 中浮点数的加法并不满足结合律.

2. 近似数的有效数位

相对误差可以反映一个近似数的精度,反映近似数精度的还有一种更加简便的方法,下面首先介绍近似数的有效数字的概念.

设 x 为一实数,像上面一样,将 x 写成惟一的形式为

$$x = \mathrm{sign}(x) \times 0.a_2a_2\cdots a_m a_{m+1}\cdots \times 10^s$$

其中 $a_1 \neq 0$,$\tilde{x}$ 是由 x 对 a_{m+1} 按"四舍五入"的原则得来的近似数,此时称 $\tilde{x}$ 有 m 位有效数字. 显然,$\tilde{x}$ 的有效数字的位数等于它的第1个非零数字 a_1 到它的最后一位数字 a_m 的数字的个数. $\tilde{x}$ 有 m 位有效数字的充要条件是 $\tilde{x}$ 的绝对误差满足

$$|\varepsilon(x)| \leqslant \frac{1}{2} \times 10^{-m+s} \tag{1.3.5}$$

由式(1.3.3)可知,字长为 n 的浮点数能对数据提供 n 位有效数字.

定理1 近似数 $\tilde{x}$ 有不少于 m 位有效数字,则它的相对误差满足

$$|\varepsilon_r(x)| \leqslant \frac{1}{2} \times 10^{-(m-1)} \tag{1.3.6}$$

证明 设 $\tilde{x}$ 有 m' 位有效数字,且 $m' \geqslant m$,由式(1.3.5)得

$$|\varepsilon(x)| \leqslant \frac{1}{2} \times 10^{-m'+s} \leqslant \frac{1}{2} \times 10^{-m+s}$$

故

$$|\varepsilon_r(x)| = \left|\frac{\varepsilon(x)}{x}\right| \leqslant \frac{\frac{1}{2} \times 10^{-m+s}}{10^{-1} \times 10^s} = \frac{1}{2} \times 10^{-(m-1)}$$

定理2 近似数 $\tilde{x}$ 的相对误差 $|\varepsilon_r(x)| \leqslant \frac{1}{2} \times 10^{-m}$,则 $\tilde{x}$ 至少有 m 位有效数字.

证明 由 $|\varepsilon_r(x)| \leqslant \frac{1}{2} \times 10^{-m}$,$|x| \leqslant 10^s$ 得

$$|\varepsilon(x)|=|\varepsilon_r(x)||x|\leqslant\frac{1}{2}\times10^{-m}\times10^{s}=\frac{1}{2}\times10^{-m+s}$$

于是 $\tilde{x}$ 至少具有 m 位有效数字.

由此可见,近似数有效数字位数越多,相对误差越小;反之,相对误差越小,近似数有效数位越多,有效数字位数的多少反映了近似数的精度.

在实际计算中,如果精确值的绝对值不大,可用精确到小数点后第几位来表示近似数的精度,如果近似数 $\tilde{x}$ 的绝对误差满足式(1.3.5),则称它精确到小数点后第 $m-s$ 位,即近似数精确到小数点后某位是指其绝对误差不超过该位值的一半.

四、问题的适定性和问题的条件

因为在计算机上对数值问题进行计算,误差不可避免,所以要提出这样的问题,当初始数据(或中间结果的数据)有所变化时,解的情况如何. 首先希望初始数据有了变化(扰动),变化了的数据所对应的解仍然存在而且惟一,而且当初始数据变化任意小时,其对应的解和精确解之间的差也任意小;另外,还希望初始数据有小的变化,能确保解的改变也不要太大. 把前者称为问题的适定性,后者称为问题的条件. 实际中出现的大多数问题都是我们所希望的,但也确实存在一些问题不具有适定性,或者满足适定性,但条件很坏. 下面给出问题适定性和问题条件的精确定义.

1. 问题的适定性

给定数据 $x\in\boldsymbol{R}^n$(可以看成由 n 个数构成的数据),问题的解为 $f(x)\in\boldsymbol{R}^m$,它是由 m 个数构成的. 设 $\delta x\in\boldsymbol{R}^n$ 为 x 的误差(扰动),$x+\delta x$ 对应的解为 $f(x+\delta x)$,如果

(1) 对数据 x 和任意接近 x 的数据 $x+\delta x$,$f(x+\delta x)$ 存在且惟一.

(2) 解 $f(x)$ 连续依赖于 x,即当 $\|\delta x\|\to0$ 时有

$$\|f(x+\delta x)-f(x)\|\to0$$

则称这种问题对数据 x 是适定的.

如果问题不适定,尽管数据任意接近 x,而对应 $x+\delta x$ 的解 $f(x+\delta x)$ 或者不存在,或者不惟一,或者和 $f(x)$ 完全不一样,这样的问题用计算机进行数值计算是非常困难的. 下面举例说明.

例 1.1 解方程组

$$\begin{cases}x+y=3\\2x+y=4\\x/6+y/6=1/2\end{cases}$$

这个方程有惟一的解 $x=1,y=2$,但是用 $n=4$ 的计算机去求解,方程变为

$$\begin{cases}x+\quad\ y=3\\2x+\quad\ y=4\\0.166\ 7x+0.166\ 7y=0.5\end{cases}$$

这个方程组却变为不相容的,它没有解. 即使取较大的字长,情况仍不会改变,这是因为原方程组增广矩阵的秩为 2,即增广矩阵的行列式为零,初始数据的任何一点变化就会使它的值非零,对应的增广矩阵为满秩,变为不相容方程组.

2. 问题的条件

给定数据 $x \in \boldsymbol{R}^n$，$f(x)$ 为适定问题，如果对数据 x 的每一小扰动 δx，问题 $f(x)$ 解的变化 $\| f(x+\delta x)-f(x) \|$ 也不大，这个问题称为条件是好的，或称为是良态的. 否则，如果对数据的一个小扰动 δx，使解的变化很大，或者说解对初始数据的扰动十分敏感，称这个问题为坏条件的，或称为病态的. 对病态问题也很难在计算机上用数值方法求解. 在实际问题中，如何判断问题的条件好坏，对有一些问题是容易确定的，大多数适定问题，往往存在和 x 无关的常数 $L>0$，$\delta>0$，对所有满足 $\| \delta x \| < \delta$ 的 δx 均有

$$\| f(x+\delta x)-f(x) \| \leqslant L \| \delta x \|$$

称为 Lipschitz 条件，此时可看出 L 不很大，问题是好条件的，这时当 δx 很小，可以确保 $\| f(x+\delta x)-f(x) \|$ 也很小，如果 L 很大，或者不存在，则 x 小的变化将会引起解的大的变化，这时问题是病态的，满足 Lipschitz 条件的常数 L 称为 Lipschitz 常数，由于 L 反映了问题条件的好坏，故也称为条件数.

下面通过几个实例来说明病态问题.

例 1.2 计算 $g(x)=x^2+x-111\,442$ 在 $x=1\,000/3$ 的值.

解 $g(1\,000/2)=22/9=2.\dot{4}$，利用 $n=4$ 的浮点数计算，此时 $g(333.3)=-19.81$，再利用 $n=5$ 的浮点数计算，此时 $g(333.33)=0.218\,7$，由此可见，上面的计算误差很大，没有一位有效数字和精确值相同，计算结果不可靠.

例 1.3 (Hilbert) 解线性方程组

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

解 此方程组的精确解为 $x_1=x_2=x_3=1$. 如果在 $n=2$ 的计算机上计算，上面方程化为

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

求得的解为 $x_1=-6.222\cdots$，$x_2=38.25\cdots$，$x_3=-33.65\cdots$. 如果在 $n=3$ 的计算机上计算，求得的解为 $x_1=1.090\cdots$，$x_2=0.4880\cdots$，$x_3=1.491\cdots$，计算结果和精确解比较相差甚远，计算仍是不可靠的.

例 1.4 (Wilkinson) 求解 20 次代数方程

$$p(x)=(x-1)(x-2)(x-3)\cdots(x-20)=x^{20}-210x^{19}+\cdots=0$$

解 其方程的根为 $1,2,3,\cdots,20$. 现在将 x^{19} 的系数换为 $-210+2^{-23}$，其余系数不变，此时方程变为 $p(x)+2^{-23}x^{19}=0$，方程的系数变化很小，$2^{-23}\approx 0.119\times 10^{-6}$，但它的根变化很

大，Wilkinson 在 $\beta=2, n=90$ 的计算机上用精密方法计算出这个方程的根为

1.000 000 000	6.000 006 944	$0.095\ 266\ 145 \pm 0.643\ 500\ 904i$
2.000 000 000	6.999 697 234	$11.793\ 633\ 881 \pm 1.652\ 329\ 728i$
3.000 000 000	8.007 267 603	$13.992\ 358\ 137 \pm 2.518\ 830\ 070i$
4.000 000 000	8.917 250 259	$16.730\ 737\ 466 \pm 2.812\ 624\ 894i$
4.999 999 928	20.846 908 101	$19.502\ 439\ 400 \pm 1.940\ 330\ 347i$

由此可见，初始数据微小变化，而解变化很大，出现了 10 个复根，解对初始数据的扰动十分敏感.

问题的适定性和问题的条件均是问题本身固有的属性，它和用以解决此类问题的算法或采用的手段没有任何关系，它和后面将要介绍的算法稳定性有着本质区别.

五、误差对计算结果的影响

1. 误差在一般计算问题中的传播

给一组数据 $x_1, x_2, \cdots, x_n$，要求的值 y 可以看成是 $x_1, x_2, \cdots, x_n$ 的函数，即

$$y = \varphi(x_1, x_2, \cdots, x_n)$$

设 $\tilde{x}_i$ 为 $x_i (i=1,2,\cdots,n)$ 的近似值. 记 $\Delta x_i = \tilde{x}_i - x_i$ 为 x_i 的绝对误差，由于初始数据的误差，因此计算的值应为

$$\tilde{y} = \varphi(\tilde{x}_1, \tilde{x}_2, \cdots, \tilde{x}_n)$$

现在讨论 $\tilde{x}_i$ 的误差对解的误差有什么影响，记 $\tilde{y}$ 的绝对误差为

$$\varepsilon(y) = \tilde{y} - y = \varphi(\tilde{x}_1, \tilde{x}_2, \cdots, \tilde{x}_n) - \varphi(x_1, x_2, \cdots, x_n)$$

假设函数 φ 连续 2 阶可微，将上式等号右边 Taylor 展开，如果所有绝对误差 Δx_i 较小，因而 Δx_i 的高阶量可略去不计，上式变为

$$\varepsilon(y) \approx \frac{\partial \varphi}{\partial x_1}\Delta x_1 + \frac{\partial \varphi}{\partial x_2}\Delta x_2 + \cdots + \frac{\partial \varphi}{\partial x_n}\Delta x_n = \sum_{i=1}^{n} \frac{\partial \varphi}{\partial x_i}\Delta x_i \tag{1.3.7}$$

其中 $\dfrac{\partial \varphi}{\partial x_i} = \dfrac{\partial}{\partial x_i}\varphi(x_1, x_2, \cdots, x_n)$，设 $\varphi = y = \varphi(x_1, x_2, \cdots, x_n)$，于是由式(1.3.7) 有

$$\varepsilon_r(y) = \frac{\varepsilon(y)}{y} \approx \sum_{i=1}^{n} \frac{x_i}{y}\frac{\partial \varphi}{\partial x_i}\varepsilon_r(x_i) = \sum_{i=1}^{n} \frac{x_i}{\varphi}\frac{\partial \varphi}{\partial x_i}\varepsilon_r(x_i) \tag{1.3.8}$$

$$|\varepsilon_r(y)| \leqslant \sum_{i=1}^{n} \left|\frac{x_i}{\varphi}\frac{\partial \varphi}{\partial x_i}\right| |\varepsilon_r(x_i)| = \sum_{i=1}^{n} \alpha_i \varepsilon_i \tag{1.3.9}$$

这里 $\alpha_i = \left|\dfrac{x_i}{\varphi}\dfrac{\partial \varphi}{\partial x_i}\right|$，$\varepsilon_i = |\varepsilon_r(x_i)|$，$i=1,2,\cdots,n$，式(1.3.8)、式(1.3.9) 反映了初始数据的误差传播到计算结果的误差 $\varepsilon_r(y)$ 的情况. 式(1.3.9) 中的系数 α_i 反映了问题 φ 的条件的好坏，如果 α_i 均很小，由初始数据 x_i 的小的误差 ε_i 能确保计算结果的误差 $\varepsilon_r(y)$ 也很小，而 α_i 较大，初始数据的误差 ε_i 可导致计算结果大的误差.

设 $\boldsymbol{\alpha} = [\alpha_1\ \alpha_2\ \cdots\ \alpha_n]^{\mathrm{T}}$，$\boldsymbol{\varepsilon} = [\varepsilon_1\ \varepsilon_2\ \cdots\ \varepsilon_n]^{\mathrm{T}}$，利用 Cauchy 不等式，有

$$\sum_{i=1}^{n} \alpha_i \varepsilon_i \leqslant \left(\sum_{i=1}^{n} \alpha_i^2\right)^{1/2} \left(\sum_{i=1}^{n} \varepsilon_i^2\right)^{1/2} = \|\boldsymbol{\alpha}\|\ \|\boldsymbol{\varepsilon}\|$$

故由式(1.3.9) 得

$$|\varepsilon_r(y)| \leqslant \|\boldsymbol{\alpha}\|\ \|\boldsymbol{\varepsilon}\|$$

这里 $L = \|\boldsymbol{\alpha}\|$ 为 Lipschitz 常数，称为问题 φ 的条件数.