现代数值计算方法

第一章 数值计算的基本概念

第一章 数值计算的基本概念

§1.1 数值计算的研究对象和内容

数值计算是数学中关于计算的一门学问, 它研究如何借助于计算工具求得数学问题的数值解答. 这里的数学问题仅限于数值问题, 即给出一组数值型的数据 (通常是一些实数, 称为初始数据), 去求另一组数值型数据, 问题的本身反映了这两组数据之间的某种确定关系. 如函数的计算、方程的求根都是数值问题的典型例子.

数值计算的历史源远流长, 自有数学以来就有关于数值计算方面的研究. 古代巴比伦人在公元前 2000 年左右就有了关于二次方程求解的研究, 我国古代数学家刘徽利用割圆术求得圆周率的近似值, 而

后祖冲之求得圆周率的高精度的值都是数值计算方面的杰出成就. 数值计算的理论与方法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. 但在电子计算机出现以前, 它的理论与方法发展十分缓慢, 甚至长期停滞不前. 由于受到计算工具的限制, 无法进行大量的复杂的计算.

科学技术的发展与进步提出了越来越多的复杂的数值计算问题, 这些问题的圆满解决已远非人工手算所能胜任, 必须依靠电子计算机快速准确的数据处理能力. 这种用计算机处理数值问题的方法, 称之为科学计算. 今天, 科学计算的应用范围非常广泛, 天气预报, 工程设计, 流体计算, 经济规划和预测以及国防尖端的一些科研项目, 如核武器的研制、导弹和火箭的发射等等, 始终是科学计算最为活跃的领域.

现代数值计算的理论与方法是与计算机技术的发展与进步一脉相承的. 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, 而数值计算的理论与方法是计算机进行科学计算的依据. 它不但为科学计算提供了可靠的理论基础, 并且提供了大量行之有效的数值问题的算法.

由于计算机对数值计算这门学科的推动和影响, 使数值计算的重点转移到使用计算机编程算题的方面上来. 现代的数值计算理论与方法主要是面对计算机的. 研究与寻求适合在计算机上求解各种数值问题的算法是数值计算这门学科的主要内容.

§1.2 数值算法的基本概念

粗略地说, 数值算法就是求解数值问题的计算步骤. 由一些基本运算及运算顺序的规定构成的一个 (数值) 问题完整的求解方案称为 (数值) 算法.

计算机的运算速度极高, 可以承担大运算量的工作, 这是否意味

着人们可以对计算机上的算法随意选择呢？

我们知道, 在线性代数中, 克兰姆规则原则上可用来求解线性方程组. 用这种方法求解一个 n 阶方程组, 要计算 $n+1$ 个 n 阶行列式的值, 这意味着总共需要做 $A_n = n!(n-1)(n+1)$ 次乘法. 当 n 充分大时, 这个计算量是相当惊人的. 譬如: 对于一个 20 阶的方程组, 大约需要做 $A_{20} \approx 10^{21}$ 次乘法, 现在假设这项计算用每秒十亿次 (10^9) 的计算机去做, 每年只能完成大约 $3.15 \times 10^{16}(365 \times 24 \times 3600 \times 10^9)$, 故所需时间约为 $10^{21} \div (3.15 \times 10^{16}) \approx 3.2 \times 10^4$ (年), 即大约需要三万二千年才能完成. 当然, 解线性方程组我们有许多实用的算法 (参看本书的后续章节). 这个简单的例子说明, 能否正确地制定算法是科学计算成败的关键.

计算机虽然是运算速度极高的现代化计算工具, 但它本质上仅能完成一系列具有一定位数的基本的算术运算和逻辑运算. 故而在进行

数值计算时, 首先要将各种类型的数值问题转化为一系列计算机能够执行的基本运算.

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数仅仅是有限位小数, 误差不可避免. 这些误差对计算结果的影响是需要考虑的. 如果给出一种算法, 在计算机上运行时, 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的误差, 这些误差在计算过程中的累计越来越大, 以致淹没了真值, 那么这样的计算结果将变得毫无意义. 相应地, 我们称这种算法是不可靠的, 或者数值不稳定的. 现在的计算机无论在运算速度上还是在存储能力上都是传统计算工具所无法比拟的. 但即使这样, 我们在设计算法时, 也必须对算法的运算次数和存储量大小给予足够的重视. 实际中存在大量的这样的问题, 由于所提供的解决这些问题的算法的运算量大得惊人, 即使利用最尖端的计算机也无法在有效时间内求得问

题的答案.

那么, 一个好的算法一般应该具备什么特征呢? (1) 必须结构简单, 易于计算机实现; (2) 理论上必须保证方法的收敛性和数值稳定性; (3) 计算效率必须要高, 即计算速度快且节省存储量; (4) 必须经过数值实验检验, 证明行之有效.

§1.3 误差的基本理论

§1.3.1 误差的来源

误差是描述数值计算中近似值的精确程度的一个基本概念, 在数值计算中十分重要, 误差按来源可分为模型误差、观测误差、截断误差和舍入误差四种.

1. 模型误差

数学模型通常是由实际问题抽象得到的, 一般带有误差, 这种误差称为模型误差.

2. 观测误差

数学模型中包含的一些物理参数通常是通过观测和实验得到的, 难免带有误差, 这种误差称为观测误差.

3. 截断误差

求解数学模型所用的数值方法通常是一种近似方法, 这种因方法产生的误差称为截断误差或方法误差. 例如, 利用 $\ln(x+1)$ 的 Taylor 公式:

$$\ln(x+1) = x - \frac{1}{2}x^2 + \frac{1}{3}x^3 - \frac{1}{4}x^4 + \cdots + (-1)^{n-1}\frac{1}{n}x^n + \cdots$$

实际计算时只能截取有限项代数和计算, 如取前 5 项有:

$$\ln 2 \approx 1 - \frac{1}{2} + \frac{1}{3} - \frac{1}{4} + \frac{1}{5}.$$

这时产生误差 (记作 R_5)

$$R_5 = -\frac{1}{6} + \frac{1}{7} - \frac{1}{8} + \frac{1}{9} - \cdots .$$

4. 舍入误差

由于计算机只能对有限位数进行运算, 在运算中象 e, $\sqrt{2}$, $1/3$ 等都要按舍入原则保留有限位, 这时产生的误差称为舍入误差或计算误差. 关于舍入误差, 以后我们还要讨论.

在数值分析中, 我们总假定数学模型是准确的, 因而不考虑模型误差和观测误差, 主要研究截断误差和舍入误差对计算结果的影响.

§1.3.2 绝对误差和相对误差

1. 绝对误差

给一实数 x, 它的近似值为 x^*, $x^* - x$ 反映了近似值和精确值差

异的大小, 因此称

$$\varepsilon(x) = x^* - x$$

为近似数 x^* 的绝对误差. 由于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 $\varepsilon(x)$ 的绝对值的一个上限. 如果存在一个正数 η, 使得

$$|\varepsilon(x)| \leq \eta,$$

则称 η 为 x^* 的绝对误差限（或误差界）. 此时

$$x - \eta \leq x^* \leq x + \eta.$$

通常将上式简记为 $x^* = x \pm \eta$.

2. 相对误差

绝对误差通常不能完全反映近似数的精确程度, 它还依赖于此数本身的大小, 因此有必要引进相对误差的概念. 近似数 x^* 的相对误差

定义为

$$\varepsilon_r(x) = \frac{\varepsilon(x)}{x} = \frac{x^* - x}{x}.$$

由于 x 未知, 实际使用时总是将 x^* 的相对误差取为

$$\varepsilon_r(x) = \frac{\varepsilon(x)}{x^*} = \frac{x^* - x}{x^*}.$$

和绝对误差的情况一样, 引进相对误差限（或相对误差界）的概念. 如果存在一个正数 δ, 使得

$$|\varepsilon_r(x)| \leq \delta,$$

则称 δ 为 x^* 的相对误差限.

根据上述定义可知, 当 $|x^* - x| \leq 1$ cm 时, 测量 10 m 物体时的相对误差为

$$|\varepsilon_r(x)| \leq \frac{1}{1000} = 0.1\%.$$

而测量 100 m 物体时的相对误差为

$$|\varepsilon_r(x)| \leq \frac{1}{10000} = 0.01\%.$$

可见后者的测量结果要比前者精确. 所以, 在分析误差时, 相对误差更能刻画误差的特性.

例 1.1 设 $x^* = 4.32$ 是由精确值 x 经过四舍五入得到的近似值, 求 x^* 的绝对误差限和相对误差限.

解 由已知可得: $4.315 \leq x < 4.325$, 所以

$$-0.005 \leq x - x^* < 0.005.$$

因此, 绝对误差限为 $\eta = 0.005$, 相对误差限为 $\delta = 0.005 \div 4.32 \approx 0.12\%$.

3. 向量的误差

实际问题中给出的数据（初始数据）往往不是一个孤立的数, 有时给出的是一组相关联的实数 $x_1, x_2, \cdots, x_n$, 为了便于统一处理, 通常将它们看成 n 维欧氏空间R^n 中的向量或点, 记为 $x = (x_1, x_2, \cdots, x_n)^T$. 设 x_i^* 是 $x_i (1 \leq i \leq n)$ 的近似数, $x^* = (x_1^*, x_2^*, \cdots, x_n^*)^T$ 是 x 的近似向量, 反映这两组数据的整体误差可用 $x^* - x$ 的欧氏范数

$$\|x^* - x\| = \left(\sum_{i=1}^{n} (x_i^* - x_i)^2 \right)^{\frac{1}{2}}$$

来表示, 它实际上是向量 x^* 与 x 在 R^n 上的欧氏距离. 用 $\|x^* - x\|$ 的值表示 x^* 的绝对误差, 类似地用 $\|x^* - x\|/\|x\|$ 表示 x^* 的相对误差. 容易发现, 当 $n = 1$ 时, 就化成前述通常的单个数的误差了.

§1.3.3 近似数的有效数字

为了能给出一种数的表示法, 使之既能表示其大小, 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 有很多位数时, 我们通常按四舍五入得到 x 的近似值 x^*. 例如无理数

$$\pi = 3.14159265358 97\cdots,$$

若按四舍五入原则分别取二位和四位小数时, 则得

$$\pi \approx 3.14, \quad \pi \approx 3.1416.$$

不管取几位得到的近似数, 其绝对误差不会超过末位数的半个单位, 即

$$|\pi - 3.14| \leq \frac{1}{2} \times 10^{-2}, \quad |\pi - 3.1416| \leq \frac{1}{2} \times 10^{-4}.$$

定义 1.1 设数 x^* 是数 x 的近似值, 如果 x^* 的绝对误差限是它的某一数位的半个单位, 并且从 x^* 左起第一个非零数字到该数位共有 n 位, 则称这 n 个数字为 x^* 的有效数字, 也称用 x^* 近似 x 时具有 n 位有效数字.

例 1.2 已知下列近似数

$$a = 24.1357, \quad b = -0.250, \quad c = 0.00016$$

的绝对误差限都是 0.0005, 问它们具有几位有效数字?

解 由于 0.0005 是小数点后第 3 数位的半个单位, 所以 a 有 5 位有效数字 2、4、1、3、5, b 有 3 位有效数字 2、5、0, c 没有有效数字.

一般地, 任何一个实数 x 经过四舍五入后得到的近似值 x^* 都可以写成如下标准形式

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

所以, 当其绝对误差限满足

$$|x^* - x| \leq \frac{1}{2} \times 10^{m-n}$$

时, 则称近似值 x^* 具有 n 位有效数字, 其中 m 为整数, a_1 是 1 到 9 中的某个数字, $a_2, \cdots, a_n$ 是 0 到 9 中的数字.

根据上述有效数字的定义, 不难验证 π 的近似值 3.1416 具有 5 位有效数字. 事实上, $3.1416 = 0.31416 \times 10^1$, 这里 $m = 1,\ n = 5$, 由于

$$|\pi - 3.1416| = 0.0000073465\cdots < \frac{1}{2} \times 10^{-4},$$

所以它具有 5 位有效数字.

有效数字与绝对误差、相对误差有如下关系:

定理 1.1 (1) 若有数 x 的近似值 $x^* = \pm 0.a_1a_2\cdots a_n \times 10^m$ 有 n 位有效数字, 则此近似值 x^* 的绝对误差限为

$$|x^* - x| \leq \frac{1}{2} \times 10^{m-n}. \tag{1.2}$$

(2) 若近似数 x^* 有 n 位有效数字, 则其相对误差限满足

$$|\varepsilon_r(x)| \leq \frac{1}{2a_1} \times 10^{-(n-1)}. \tag{1.3}$$

反之, 若近似数 x^* 的相对误差限满足

$$|\varepsilon_r(x)| \leq \frac{1}{2(a_1+1)} \times 10^{-(n-1)}. \tag{1.4}$$

则 x^* 至少有 n 位有效数字.

证明 (1) 结论显然.

(2) 由 (1.1) 式可知

$$a_1 \times 10^{m-1} \leq |x^*| \leq (a_1+1) \times 10^{m-1}$$

故

$$|\varepsilon_r(x)| = \frac{|x^*-x|}{|x^*|} \leq \frac{\frac{1}{2} \times 10^{m-n}}{a_1 \times 10^{m-1}} = \frac{1}{2a_1} \times 10^{-(n-1)}.$$

反之, 由

$$|x^*-x| = |x^*|\cdot|\varepsilon_r(x)| \le (a_1+1)\times 10^{m-1}\cdot\frac{1}{2(a_1+1)}\times 10^{-(n-1)} = \frac{1}{2}\times 10^{m-n}$$

知, x^* 至少有 n 位有效数字. □

由此可见, 当 m 一定时, n 越大（即有效数字位数越多）, 其绝对误差限越小.

例 1.3 为使 $\sqrt{26}$ 的近似值的相对误差小于 0.1%, 问应取几位有效数字?

解 $\sqrt{26}$ 的近似值的首位非零数字是 $a_1 = 5$. 假设应取 n 位有效数字, 则由 (1.3) 式有

$$|\varepsilon_r(x)| \le \frac{1}{2\times 5}\times 10^{-(n-1)} < 0.1\%,$$

解之得 $n > 3$, 故取 $n = 4$ 即可满足要求. 也就是说, 只要 $\sqrt{26}$ 的近似值具有 4 位有效数字, 就能保证 $\sqrt{26} \approx 5.099$ 的相对误差小于 0.1%.

例 1.4 已知近似数 x^* 的相对误差界为 0.0002. 问 x^* 至少有几位有效数字?

解 由于 x^* 首位数未知, 但必有 $1 \leq a_1 \leq 9$. 则由 (1.4) 式有

$$|\varepsilon_r(x)| \leq \frac{1}{2 \times (a_1 + 1)} \times 10^{-(n-1)} = 0.0002,$$

得

$$10^{n-1} = \frac{1}{4 \times (a_1 + 1)} \times 10^4, \quad n = 5 - \lg 4 - \lg(a_1 + 1),$$

可得

$$4 - \lg 4 \leq n \leq 5 - \lg 8, \quad 3.3979 \leq n \leq 4.0969,$$

故取 $n = 3$.

§1.4 数值算法设计的若干原则

为了减少舍入误差的影响, 设计算法时应遵循如下的一些原则.

1. 避免两个相近的数相减

如果 x^*, y^* 分别是 x, y 的近似值分别, 则 $z^* = x^* - y^*$ 是 $z = x - y$ 的近似值, 此时有

$$|\varepsilon_r(z)| = \frac{|z^* - z|}{|z^*|} \leq \left|\frac{x^*}{x^* - y^*}\right| \cdot |\varepsilon_r(x)| + \left|\frac{y^*}{x^* - y^*}\right| \cdot |\varepsilon_r(y)|,$$

可见, 当 x^* 与y^* 很接近时, z^* 的相对误差有可能很大. 例如, 当 $x = 5000$ 时, 计算

$$\sqrt{x+1} - \sqrt{x}$$

的值, 若取 4 位有效数字计算

$$\sqrt{x+1} - \sqrt{x} = \sqrt{5001} - \sqrt{5000} = 71.72 - 71.71 = 0.01.$$

这个结果只有一位有效数字, 损失了三位有效数字, 从而绝对误差和相对误差都变得很大, 严重影响了计算精度. 但如果将公式改变为

$$\sqrt{x+1}-\sqrt{x}=\frac{1}{\sqrt{x+1}+\sqrt{x}}=\frac{1}{\sqrt{5001}+\sqrt{5000}}\approx 0.006972.$$

它仍然有四位有效数字, 可见改变计算公式可以避免两个相近数相减而引起的有效数字的损失, 从而得到比较精确的计算结果.

因此, 在数值计算中, 如果遇到两个相近的数相减运算, 应考虑改变一下算法以避免两数相减.

2. 避免绝对值太小的数作除数

由于除数很小, 将导致商很大, 有可能出现“溢出”现象. 另外, 设 x^*, y^* 分别是 x, y 的近似值, 则 $z^*=x^*\div y^*$ 是 $z=x\div y$ 的近似值, 此时, z 的绝对误差满足

$$|\varepsilon(z)|=|z^*-z|=\left|\frac{(x^*-x)y+x(y-y^*)}{y^*y}\right|\approx\frac{|x^*|\cdot|\varepsilon(y)|+|y^*|\cdot|\varepsilon(x)|}{(y^*)^2}.$$

由此可见, 若除数太小, 则可能导致商的绝对误差很大.

3. 防止大数“吃掉”小数

因为计算机上只能采用有限位数计算, 若参加运算的数量级差很大, 在它们的加、减运算中, 绝对值很小的数往往被绝对值较大的数“吃掉”, 造成计算结果失真.

例 1.5 求方程

$$x^2 - (10^9 + 1)x + 10^9 = 0$$

的根.

解 显然, 方程的两个根为: $x_1 = 10^9$, $x_2 = 1$. 如果用 8 位数字的计算机计算, 使用二次方程的求根公式

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

得到

$$\begin{aligned} -b &= 10^9+1 = \underbrace{0.10000000}_{8\ \text{位}} \times 10^{10} + \underbrace{0.10000000}_{8\ \text{位}} \times 10^1 \\ &= \underbrace{0.10000000}_{8\ \text{位}} \times 10^{10} + \underbrace{0.00000000}_{8\ \text{位}} \times 10^{10} \\ &\overset{\triangle}{=} 0.10000000 \times 10^{10} \quad (\text{第 } 9,10 \text{ 位舍去}) \\ &= 10^9 \quad (\ \overset{\triangle}{=}\ \text{表示机器中的相等}), \end{aligned}$$

那么有

$$\sqrt{b^2-4ac} = \sqrt{(10^9+1)^2 - 4\times 1\times 10^9} = \sqrt{(10^9-1)^2} \overset{\triangle}{=} 10^9,$$

所以

$$x_1 = \frac{-b+\sqrt{b^2-4qc}}{2a} = \frac{10^9+10^9}{2\times 1} = 10^9, \quad x_2 = \frac{10^9-10^9}{2\times 1} = 0.$$

实际上, x_2 应等于 1. 可以看出 x_2 的误差太大, 原因是在做加减运算过程中要“对阶”, 因而“小数” 1 在对阶过程中, 被“大数” 10^9 吃掉了. 而从上述计算可以看出, x_1 是可靠的, 故可利用根与系数的关系 $x_1 \cdot x_2 = c/a$ 来求 x_2:

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

此方法是可靠的.

4. 尽量简化计算步骤以减少运算次数

同样一个问题, 如果能减少运算次数, 不但可以节省计算时间, 还可以减少舍入误差的传播. 这是数值计算中必须遵循的原则, 也是数值分析要研究的重要内容. 例如: 计算多项式

$$P(x) = a_0x^n + a_1x^{n-1} + \cdots + a_{n-1}x + a_n$$

的值, 若直接计算 $a_{n-k}x^k(k=0,1,\cdots,n)$, 再逐项相加, 一共需做

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

次乘法和 n 次加法. 而若采用

$$P(x)=(\cdots((a_0x+a_1)x+a_2)x+\cdots+a_{n-1})x+a_n$$

所谓的秦九韶算法, 则只需要 n 次乘法和 n 次加法即可. 又如, 计算 x^{63} 的值. 若将 x 的值逐个相乘, 则需做 62 次乘法, 但如果写成

$$x^{63}=x\cdot x^2\cdot x^4\cdot x^8\cdot x^{16}\cdot x^{32},$$

只要 10 次乘法运算就可以了.

5. 应采用数值稳定性好的算法

在计算过程中, 由于原始数据本身就具有误差, 每次运算有可能产生舍入误差. 误差的传播和累积很可能会淹没真解, 使计算结果变

得根本不可靠. 先看下面的例子.

例 1.6 在四位十进制计算机上计算 8 个积分:

$$I_n = \int_0^1 x^n \mathrm{e}^{x-1}\mathrm{d}x, \quad n = 0,\ 1, \cdots, 7.$$

解 利用分部积分公式可得递推关系: $I_n = 1 - nI_{n-1}$. 注意到 $I_0 = 1 - \mathrm{e}^{-1} \approx 0.6321$ 及

$$I_n = \int_0^1 x^n \mathrm{e}^{-(1-x)}\mathrm{d}x < \int_0^1 x^n \mathrm{d}x = \frac{1}{n+1} \to 0 \ \ (n \to \infty),$$

可得两种算法:

算法 (a) 令 $I_0 = 0.6321$, 再算 $I_n = 1 - nI_{n-1}$, $n = 1,\ 2, \cdots, 7$;

算法 (b) 令 $I_{11} = 0$, 再算 $I_{n-1} = (1 - I_n)/n$, $n = 11,\ 10, \cdots, 1$.

按算法 (a) 得 8 个积分的近似值为

0.6321, 0.3679, 0.2642, 0.2074, 0.1704, 0.1480, 0.1120, 0.2160.

按算法 (b) 得 8 个积分的近似值为

$$0.6321,\ \ 0.3679,\ 0.2642,\ 0.2073,\ 0.1709,\ 0.1455,\ 0.1268,\ 0.1124.$$

算法 (b) 中的诸结果均准确到 4 位小数, 而算法 (a) 中的 I_7 没有一位数字是准确的. □

可靠的算法, 各步误差不应对计算结果产生过大的影响, 即具有稳定性. 研究算法是否稳定, 理应考察每一步的误差对算法的影响, 但这相当困难且繁琐. 为简单计, 通常只考虑某一步 (如运算开始时) 误差的影响. 这实质上是把算法稳定性的研究, 转化为初始数据误差对算法影响的分析. 从某种意义上来说, 这种做法是合理的. 可以设想, 一步误差影响大, 多步误差影响更大; 一步误差影响逐步削弱, 多步误差影响也削弱. 因此简化研究得出的结论具有指导意义.

下面用此法分析例 1.6 中的两种算法的稳定性. 对算法 (a), 设

$\bar{I}_0 \approx I_0$ 有误差, 此后计算无误差, 则

$$\begin{cases} I_n = 1 - nI_{n-1}, \\ \bar{I}_n = 1 - n\bar{I}_{n-1}, \end{cases} \qquad n = 1, 2, \cdots, 7.$$

两式相减得

$$I_n - \bar{I}_n = (-n)(I_{n-1} - \bar{I}_{n-1}), \quad n = 1, 2, \cdots, 7.$$

由此递推, 得

$$I_7 - \bar{I}_7 = -7!(I_0 - \bar{I}_0) = -5040(I_0 - \bar{I}_0).$$

同理, 对算法 (b), 设 $\bar{I}_{11} \approx I_{11}$ 有误差, 此后计算无误差, 则有

$$I_0 - \bar{I}_0 = -(I_{11} - \bar{I}_{11})/39916800 \approx -2.5052 \times 10^{-8}(I_{11} - \bar{I}_{11}).$$

由此可见, 按算法 (a), 最后结果误差是初始误差的 5040 倍, 而按算法

(b), 最后结果误差只是初始误差的 39916800 分之一. 这说明算法 (b) 的稳定性较好.