数值计算方法

魏焕彩 郑修才 王 玮

石油大学出版社

1

数值计算方法

魏焕彩　郑修才　王　玮　编

石油大学出版社

鲁新登字10号

内容提要

本书共分九章，主要内容包括：算法与误差，方程求根，线性代数方程组的求解，矩阵的特征值问题，插值与逼近，数值微分与数值积分，常微分方程和偏微分方程的数值解等。每节后附有习题，书末附有部分习题答案，便于读者学习和掌握各节的基本内容。

本书可作为工科院校各专业的教材，也可供从事科学技术计算工作的人员作参考用书和自学用书。

数值计算方法
主编 黄焕彩

石油大学出版社出版
（山东省东营市）
新华书店发行
山东东营新华印刷厂印刷

开本 850×1168 1/32 7.75印张 198千字
1995年7月第1版 1995年7月第1次印刷
印数 1—1100册
ISBN 7-5636-0711-0/O·32
定价：14.75元

前　言

由于计算机在科学领域中应用日益广泛，计算方法课程也受到各院校的重视，因而计算机成为一门必修的基础课。

本书是由我们多年的讲稿改写而成，从实用角度出发，初步阐述了数值计算的基本理论和方法，全文力求通俗易懂，便于自学，对有些复杂的证明及推导，则仅给出公式与结论。本书授课学时为60左右，考虑到有的专业课时较少，故在某些章节前加“*”号，去掉带“*”的章节并不影响后面内容的学习。

在学习本课程之前，应预修高等数学、线性代数，微分方程等课程。

本书在编写过程中，得到许多老师和同志们的关怀和支持，特别是段奇教授主审了全部章节，周学圣教授，孙传灼教授也提出许多修改意见及帮助。在此谨表感谢。

由于我们的水平有限，书中错误之处敬请批评指正。

编　者

1995年6月

目　录

第一章　算法与误差

科学技术的发展提出大量需用计算机才能解决的实际问题，而用计算机解决实际问题则需经图 1-1 的五个过程：

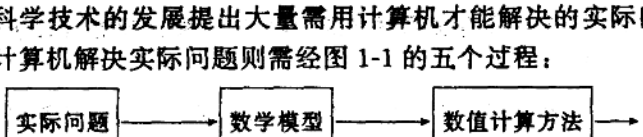

图 1-1

数值计算方法是研究用计算机解决数学问题的方法及其理论，是一门内容丰富、研究方法深刻、实用性较强的数学课程。

§1-1　算　　法

一、什么是算法

为用计算机解决数学问题而构造的能用数值计算的实施方法就称为算法。

算法必须是构造性的，例如数学上常用的反证法这种不是构造性的方法就不能称为算法；算法又必须是能通过数值演算的方法，因而图解法就不能称为算法；算法还必须是一种实施方法，它要完整地描述整个解题过程，因此它不能仅仅是单纯的数学公式，而必须是准确和完整的解题方案。

二、算法的可用性

有些算法在理论上是很完美的，在计算机上却是不可用的，因此有必要考虑算法的可用性。

例如，用克莱姆(Gramer)法则解线性方程组，从理论上说是

可以的，但当求高阶方程组的解时却不可用。这是因为按克莱姆法则求一个 n 阶方程组的解需算 $n+1$ 个行列式的值，而算一个 n 阶行列式的值又需用 $(n-1)\cdot n!$ 次乘法，这样总共需 $(n+1)(n-1)n!$ 次乘法，即使只取 $n=20$，乘法次数也近于 10^{21} 之多，就是使用每秒能做千万次乘法的电子计算机，也需连续工作几百万年才能完成，这显然是没有实际意义的。因而克莱姆法则在计算机上一般是不可用的。

另外，有些算法乍看起来没有什么问题，但实际计算表明却是无效的，因而也是不可用的。

例 1-1　建立积分

$$I_n=\int_0^1 \frac{x^n}{x+5}\mathrm{d}x \qquad (n=0,1,2,\cdots,8)$$

的递推关系并作实际计算。

解：容易导出对 $n=1,2,\cdots,8$ 有

$$\begin{aligned} I_n+5I_{n-1} &= \int_0^1 \frac{x^n+5x^{n-1}}{x+5}\mathrm{d}x \\ &= \int_0^1 x^{n-1}\mathrm{d}x \\ &= \frac{1}{n} \end{aligned}$$

又 $I_0=\int_0^1 \frac{\mathrm{d}x}{x+5}=I_n6-I_n5=I_n1\cdot 2$，于是所求递推公式为

$$\left.\begin{aligned} I_0 &= \mathrm{I_n}1\cdot 2 \\ I_n &= -5I_{n-1}+\frac{1}{n} \quad (n=1,2,\cdots,8) \end{aligned}\right\} \tag{1-1}$$

按式(1-1)取四位小数计算得数据如表(A 列)(见表 1-1)

由于 $I_n>0$，而上述计算值当 $n>4$ 时正负相间，显然计算值不能反映 I_n 的真面目，计算是错误的。

再考虑如下的递推方法：由于 $I_n<I_{n-1}$，故

$$5I_{n-1}<I_n+5I_{n-1}<6I_{n-1}$$

又因 $I_n+5I_{n-1}=\frac{1}{n}$，于是

$$\frac{1}{6n}<I_{n-1}<\frac{1}{5n}$$

表 1-1

n	$I_n(A)$	$I_n(B)$
0	0.1823	0.1823
1	0.0885	0.0884
2	0.0575	0.0580
3	0.0458	0.0431
4	0.0210	0.0343
5	0.0950	0.0285
6	−0.3083	0.0244
7	1.6844	0.0209
8	−8.2970	0.0204

当 $n=9$ 时有

$$\frac{1}{6\times 9}<I_8<\frac{1}{5\times 9}$$

取 $I_8\approx\frac{1}{2}\times(\frac{1}{6\times 9}+\frac{1}{5\times 9})\approx 0.0204$，按与式(1-1)相反方向建立递推关系：

$$\left.\begin{aligned} I_8 &= 0.0204 \\ I_{n-1} &= -\frac{I_n}{5}+\frac{1}{5n} \quad (n=8,7,\cdots,1) \end{aligned}\right\} \tag{1-2}$$

计算得递推数据如上表(B 列)所算，最后得到的 I_0 竟和式(1-1)所取 I_0 值完全一致。

上述分析表明，递推关系式(1-1)是不可用的，而式(1-2)却是可用的，但它们都是由同一个式子 $I_n+5I_{n-1}=\frac{1}{n}$ 按不同的方向推出的。原因在哪里？我们将在下节给予回答。

三、算法的优劣

解决一个数学问题，往往有不同的算法。根据问题的要求和计算机运算的特点，我们可以从以下三个方面来判断一个算法的优劣。

1. 计算量的大小

电子计算机的一个重要特点就是运算速度高，现代科学技术中的许多问题也要求我们很快地给出答案。因此，一个好的算法是计算量应该小，这样运算速度才会快。

例 1-2　试计算多项式

$$P_n(x) = a_0x^n + a_1x^{n-1} + \cdots + a_{n-1}x + a_n \qquad (1\text{-}3)$$

之值。

解：我们当然可以直接计算，但这需要用$\frac{1}{2}n(n+1)$次乘法和 n 次加法。如果我们把式(1-3)写成：

$$P_n(x) = \{[(a_0x + a_1)x + a_2]x + \cdots + a_{n-1}\}x + a_n \qquad (1\text{-}4)$$

按递推公式

$$\left.\begin{aligned} u_0 &= a_0 \\ u_K &= xu_{K-1} + a_K \quad (K = 1,2,\cdots,n) \\ p_n(x) &= a_n \end{aligned}\right\} \qquad (1\text{-}5)$$

计算，则只需 n 次乘法和 n 次加法。

多项式求值的这种算法称作秦九韶算法，它是我国宋代数学家秦九韶提出来的，这种递推方法在数值分析中是经常出现的。

按直接计算法和秦九韶法尽管在计算机上都是可用的，但后者的计算量小，因而较优，实际计算中都采用后者。

例 1-3　计算 x^{255}的值

解：我们当然可以采用逐个相乘的方法，但这需用 254 次乘法。而若写成

$$x^{255} = x \cdot x^2 \cdot x^4 \cdot x^8 \cdot x^{16} \cdot x^{32} \cdot x^{64} \cdot x^{128}$$

则只需做 14 次乘法，显然计算量要小得多。

2. 存贮量的多少

计算程序所占用计算机工作单元的数目称为算法的存贮量。由于现代科学技术中碰到的实际问题大多都是较复杂的问题，就是大型计算机内存也往往相当紧张。因此，尽量节约存贮量，也是设计算法时应考虑的一个重要的因素。比如在存贮大型稀疏矩阵——即高阶且大多数元素为零的矩阵的元素时，我们不是按顺序全部存贮，而是找出非零元素所在位置的规律，尽量只存非零元素。

3. 逻辑结构是否简单

设计算法所要考虑的另一个重要因素是逻辑结构的简单性问题。虽然计算机能自动地执行复杂的算法，但算法的每个细节都需要人来制定。从计算人员的角度来讲，总是希望算法的逻辑结构尽量简化，使得编制程序比较容易，也可以使出错率降低。

§1-2 误　　差

一、误差的来源

我们知道，用计算机解决实际问题，首先要建立数学模型。它通常要加上许多限制，并忽略一些次要的因素，因此数学模型总是近似的，它与实际问题必存在有误差，这种误差称为“模型误差”。

在数学模型确定之后，由于具体问题总含一些观察数据，而这些数据是由人用测量工具观测到的，它们不可能绝对准确，而是有一定的误差，这种误差称为“测量误差”。

以上两种误差不是在数值计算时产生的，因而本课程不讨论它们。下面我们主要考察在数值计算过程中不可避免地产生的两种误差。

1. 截断误差

我们知道，许多数学运算都要求用极限或无穷的过程来求得，而计算机上只能完成有限次的算术运算。这就需将无穷过程加工

成有限过程，由此产生的误差称为“截断误差”或“方法误差”。

例如，函数 f(x)用泰勒多项式表示：

$$P_n(x) = f(x_0) + f'(x_0)(x - x_0) + \frac{f''(x_0)}{2!}(x - x_0)^2 + \cdots + \frac{f^{(n)}(x_0)}{n!}(x - x_0)^n$$

近似代替，其数值方法的截断误差是

$$R_n(x) = f(x) - P_n(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!}(x - x_0)^{n+1}$$

（其中 ξ 在 x_0 和 x 之间）。

2. 舍入误差

计算过程所遇到的数据可能位数很多，甚至是无穷小数，而计算机表示数据是一定的，因而多出的位数则必须“舍入”，这就产生“舍入误差”。

例如，在计算机上连最简单的有理数$\frac{1}{3}$、$\frac{1}{7}$等都需“舍入”成有限位小数，至于无理数 π、e、$\sqrt{2}$ 等就更是如此。

又如，若计算机上只计算 16 位数，两个 16 位数相乘，则乘积也只能保留 16 位数，这就必须“舍入”。

在计算机上“舍入”，一般采用“四舍五入”或“舍去”两种。虽然每个数据的舍入误差不大，但在计算机上完成大量的运算之后，舍入误差的积累就可能会相当惊人。因此我们必须对算法进行误差分析。

二、有关误差的几个概念

1. 绝对误差

若 x 为准确数，x^* 为其近似值，则称

$$e^* = x - x^* \tag{1-6}$$

为近似值 x^* 的绝对误差，简称误差。

注意这样定义的误差 e^* 可正可负，称它是绝对误差，是为区别下面的相对误差，而不是指绝对值。

例如圆周率 π，取近似值为 3.14 时，则其绝对误差为 $\pi-3.14=0.00159\cdots$。

但是，一般我们不能算出准确值 x，因而也不能算出误差的准确值，我们只能根据测量的工具或计算的情况，估计出误差界，这就是下一个定义。

2. 绝对误差限

若 $|e^*|\leqslant\varepsilon^*$，则称 ε^* 为近似数 x^* 的绝对误差限，简称误差限，或称精度。

注意误差限总是正数，有了误差限，就知道了 x 的范围为：

$$x^*-\varepsilon^*\leqslant x\leqslant x^*+\varepsilon^* \tag{1-7}$$

例如用有毫米刻度的米尺测量不到一米长度的物体，如果我们看到长度接近于毫米刻度 x^*，就读出此刻度 x^* 为该物体的近似长度，这时 x^* 的误差限就为二分之一毫米。

又如，真空中光速 C 的最好近似值是：

$$C^*=2.997925\times10^{10}\text{cm/s}$$

误差限是：

$$\varepsilon^*=0.000001\times10^{10}\text{cm/s}$$

通常记成：

$$C=(2.997925\pm0.000001)\times10^{10}\text{cm/s}$$

应该指出，误差限的大小还不能完全反映近似值的好坏。例如，测量一米长度时发生$\frac{1}{2}$毫米的误差，和测量一厘米的长度时发生$\frac{1}{2}$毫米的误差，显然前者的近似程度比后者要好。为能更好地反映近似程度的好坏，我们引入下面的相对误差的概念。

3. 相对误差

称 $e_r^*=\dfrac{e^*}{x}=\dfrac{x-x^*}{x}$ 为近似数 x^* 的相对误差。

由于 x 通常是不知道的，故实际计算时我们取

$$e_r^*=\frac{e^*}{x^*}=\frac{x-x^*}{x^*} \tag{1-8}$$

作为 x^* 的相对误差。

相对误差也是可正可负。由于 x 一般不知道，因而它的精确值也不知道，但我们仍可以估计出其误差限，这就是下一个概念。

4. 相对误差限

若 $|e_r^*| \leqslant \varepsilon_r^*$，则称 ε_r^* 为近似数 x^* 的相对误差限。

例如，量一米时误差限为 $\frac{1}{2}$ 毫米，则其相对误差限为 0.05%，量一厘米时误差限为 $\frac{1}{2}$ 毫米，则其相对误差限为 5%，可见前者的近似程度要好得多。

又如，光速的相对误差限为：

$$\frac{\varepsilon^*}{|x^*|} = \frac{0.000001}{2.997925} < 0.0000004 = 4 \times 10^{-7}$$

在实际问题计算时，有时我们会碰到求函数 $y=f(x)$ 的各种误差限，当 $f(x)$ 计算较复杂而 $f'(x)$ 计算较简单时，我们常采用微分公式 $dy=f'(x)dx$ 的近似公式 $\Delta y \approx f'(x)\Delta x$ 来进行计算。设 x^* 的绝对误差限为 ε^*，这时有

$$|\Delta y| \approx |f'(x^*)| \cdot \varepsilon^*$$

为 $f(x^*)$ 的绝对误差限，于是 $f(x^*)$ 的相对误差限为：

$$\left| \frac{f'(x^*)}{f(x^*)} \right| \cdot \varepsilon^*$$

例 1-4　有一圆半径 $r=21.5$cm，设量其半径时绝对误差限为 0.1cm，求用 $S=\pi r^2$ 计算面积时的绝对误差限和相对误差限。

解：$|\Delta S| \leqslant 2\pi r\varepsilon^* = 2\pi \times 21.5 \times 0.1 = 4.3\pi(\text{cm}^2)$

$$\frac{|\Delta S|}{S} \leqslant \frac{2\pi r\varepsilon^*}{\pi r^2} = \frac{2\varepsilon^*}{r} = \frac{0.2}{21.5} = 0.93\%$$

答：所求绝对误差限为 4.3cm^2，相对误差限为 0.93%。

5. 有效数字

当写出一个数的近似值之后，我们应当指明它的准确程度，为此我们引进有效数字的概念。

设数 x 的近似值 x^* 的标准形式为

$$x^* = \pm 10^m \times O \cdot \alpha_1\alpha_2\cdots\alpha_n$$

其中 $\alpha_1 \neq 0, \alpha_i$ 为整数且 $0 \leqslant \alpha_i \leqslant 9 (i=1,2,\cdots,n)$，$m$ 为整数，n 为正整数，如果 $|x-x^*| \leqslant \frac{1}{2} \times 10^{m-n}$，则称 x^* 为 x 的具有 n 位有效数字的近似值。

例如，若取 3.14 作为 π 的近似值，则有 3 位有效数字；又若取 3.1416 作为 π 的近似值，则它有 5 位有效数字；而若取 3.145 作为 π 的近似值，则它有 3 位有效数字。如果是按“四舍五入”来取有效数字，则从左边第一个非零数字起，到最右边数字止都是有效数字。

例如，如果数 0.203，0.2030，0.20300，2.03，20.3 和 203.0，都是按四舍五入所得的近似数，则其有效数字分别为 3，4，5，3，3 和 4 位。

又如，数 0.2032，0.02032，0.2002，20.32 和 20320 按四舍五入的原则具有 3 位有效数字的近似数分别为 0.203，0.0203，0.200，20.3 和 203×10^2，应该注意，0.200 不能写成 0.2，因为 0.2 只具有一位有效数字。

三、算法的稳定性

现在，我们再来看例 1-1，由于我们取 $I_0 = 0.1823$ 是真值 $\tilde{I}_0$ 的近似值，这就产生误差 e_0，即 $\tilde{I}_0 = I_0 + e_0$，当我们按式(1-1)进行递推时，我们有

$$e_1 = \tilde{I}_1 - I_1 = (1 - 5\tilde{I}_0) - (1 - 5I_0) = -5e_0$$

$$e_2 = \tilde{I}_2 - I_2 = (\frac{1}{2} - 5\tilde{I}_1) - (\frac{1}{2} - 5I_1) = (-5)^2 e_0$$

$$\vdots$$

$$e_n = \tilde{I}_n - I_n = (\frac{1}{n} - 5\tilde{I}_{n-1}) - (\frac{1}{n} - 5I_{n-1}) = (-5)^n e_0$$

其误差传播如此之快，使 I_n 的值很快地就失真，结果 A 列数据后

部就正负相间了。

再看式(1-2),我们有

$$e_n = \tilde{I}_0 - I_0 = \frac{1}{-5}(\tilde{I}_1 - I_1) = \frac{1}{(-5)^n}(\tilde{I}_n - I_n)$$

这时,即使有了 e_8,但由于误差是衰减的,到 e_0 仅是 e_8 的 $\frac{1}{5^8}$,这就是 B 列数据准确的原因。

由上面分析可以看出对算法进行误差分析的重要性。为此我们引入数值稳定性的概念。

我们称在运算过程中舍入误差不增长的算法为稳定的,否则就是不稳定的。

由此定义可以看出例 1-1 的算法中,用式(1-1)是不稳定的,用式(1-2)是稳定的。稳定算法是可以用的,不稳定算法通常不能用,要用也只能用于计算步数少的问题并注意对误差积累的控制。对于稳定的算法,可以不再具体估计舍入误差。下面我们给出在运算中应注意的若干原则,它有助于鉴别计算结果的可靠性并防止误差危害的产生。

1. 要避免相近两数相减

相近两数相减,有效数字会减少,因而相对误差就会增大。遇到这种情况,就要改变计算公式,或多取几位有效数字。

例如,对 $f(x)=\sqrt{x+1}-\sqrt{x}$,要计算 $f(1000)$,取四位有效数字有:$f(1000)\approx 31.64-31.62=0.02$ 仅有一位有效数字,但若用 $f(x)=\frac{1}{\sqrt{x+1}+\sqrt{x}}$,则:$f(1000)\approx\frac{1}{31.64+31.62}=0.01581$ 保留四位有效数字。

2. 要防止大数吃小数

在数值运算中参加运算的数有时数量级相差很大,而计算机位数有限,如不注意运算次序就可能出现大数"吃掉"小数的现象,影响计算结果的可靠性。

例 1-5　计算 $A=52492+\sum_{i=1}^{10000}\delta_i$，其中 $\delta_i=0.4(i=1,2,\cdots,10000)$。取五位有效数字。

解：若按从左到右相加则有

$$A=52492+0.4+\sum_{i=1}^{9999}\delta_i=52492+\sum_{i=1}^{9999}\delta_i$$

$$=\cdots=52492$$

若按右边和先加再加左边一个数，则 $A=56492$，显然前一个结果不正确，这是因为 0.4 太小，每次和 52492 相加都被“吃掉”了。

在除法运算中，要避免除数的绝对值远远小于被除数的绝对值，否则也有可能产生大数“吃掉”小数的现象。

例 1-6　用消去法解方程组

$$\left.\begin{aligned}0.00001x_1+x_2&=1\\2x_1+x_2&=2\end{aligned}\right\}$$

解：取四位有效数字计算，这时方程组变成

$$\left.\begin{aligned}0.1000\times10^{-4}x_1+0.1000\times10^{1}x_2&=0.1000\times10^{1}\\0.2000\times10^{1}x_1+0.1000\times10^{1}x_2&=0.2000\times10^{1}\end{aligned}\right\}$$

若用第一个方程乘以 $\left(-\frac{0.2000\times10^{1}}{0.1000\times10^{-4}}\right)$ 加到第二个方程，可得

$$\left.\begin{aligned}0.1000\times10^{-4}x_1+0.1000\times10^{1}x_2&=0.1000\times10^{1}\\0.2000\times10^{6}x_2&=0.2000\times10^{6}\end{aligned}\right\}$$

由此解得 $x_1=0$，$x_2=1$。显然不是真解，这是因为乘数 $\left(-\frac{0.2000\times10^{1}}{0.1000\times10^{-4}}\right)$ 中除数的绝对值远小于被除数的绝对值，这样数 $\left(\frac{0.2000\times10^{1}}{0.1000\times10^{-4}}\right)$ 相当大，乘以第一个方程使第一个方程系数扩大，再加到第二个方程就把第二个方程后面的系数“吃掉”了。

若用第二个方程乘以 $\left(-\frac{0.1000\times10^{-4}}{0.2000\times10^{1}}\right)$ 加到第一个方程可得