1.3 数值稳定性和要注意的若干原则

- 1.3.1 数值方法的稳定性
- 1.3.2 避免有效数字的损失
- 1.3.3 减少运算次数

1.3 数值稳定性和要注意的若干原则

学习目标:

掌握数值运算中避免大误差产生的若干准则。

1.3.1 数值方法的稳定性

定义 1.4 对于某个数值计算方法，如果输入数据的误差在计算过程中迅速增长而得不到控制，则称该算法是数值不稳定的，否则是数值稳定的。

举例说明如下。

例1 计算积分值

$$I_n = \int_0^1 \frac{x^n}{x+5} dx, n = 0,1,\cdots,6。$$

解 由于要计算系列的积分值，我们先推导 I_n 的一个递推公式。由

$$I_n + 5I_{n-1} = \int_0^1 \frac{x^n + 5x^{n-1}}{x+5} dx = \int_0^1 x^{n-1} dx = \frac{1}{n},$$

可得下面两个递推算法。

算法 1：$$I_n = \frac{1}{n} - 5I_{n-1}, n = 1,2,\cdots,6。$$

逆向递推公式

算法 2：$$I_{n-1} = \frac{1}{5}\left(\frac{1}{n} - I_n\right), n = 6,5,\cdots,1。$$

直接计算可得 $I_0 = \ln 6 - \ln 5$。如果我们用四位数字计算，得 I_0 的近似值为 $I_0^* = 0.1823$。记 $E_n = I_n - I_n^*$，I_n^* 为 I_n 的近似值。

对算法 1，有

$$E_n = -5E_{n-1} = \cdots = (-5)^n E_0。$$

按以上初始值 I_0 的取法有 $|E_0| \le 0.5\times10^{-4}$，事实上 $|E_0| \approx 0.22\times10^{-4}$。这样，我们得到 $|E_6| = 5^6|E_0| \approx 0.34$。这个数已经大大超过了 I_6 的大小，所以 I_6^* 连一位有效数字也没有了，误差掩盖了真值。

对算法 2，有

$$E_{k-n}=\left(-\frac{1}{5}\right)^{n}E_k,\left|E_0\right|=\left(\frac{1}{5}\right)^{6}\left|E_6\right|。$$

如果我们能够给出 I_6的一个近似值, 则可由算法2计算 $I_6\left(n=5,4,\ldots 0\right)$ 的近似值. 并且, 即使 E_6较大, 得到的近似值的误差将较小. 由于

$$\frac{1}{6(k+1)}=\int_0^1\frac{x^k}{6}dx<I_k<\int_0^1\frac{x^k}{5}dx=\frac{1}{5(k+1)}$$

可取 I_k的一个近似值为

$$I_k^*=\frac{1}{2}\left[\frac{1}{6(k+1)}+\frac{1}{5(k+1)}\right]。$$

对 $k=6$有 $I_1^*=0.0262$。

按 $I_0^* = 0.1823$ 和 $I_6^* = 0.0262$，分别按算法1和2计算，计算结果如表1-1，其中 $I_n^{(1)}$ 为算法1的计算值，$I_n^{(2)}$ 为算法2的计算值。易知，对于任何自然数 n，都有 $0<I_n<1$，并且 I_n 单调递减。可见，算法1是不稳定的，算法2是稳定的。

表 1 - 1

n	$I_n^{(1)}$	$I_n^{(2)}$	I_n（四位）
0	0.1823	0.1823	0.1823
1	0.0885	0.0884	0.0884
2	0.0575	0.0580	0.0580
3	0.0458	0.0431	0.0431
4	0.0210	0.0344	0.0343
5	0.0950	0.0281	0.0285
6	– 0.3083	0.0262	0.0243

用递推关系进行计算时必须注意误差的积累.

当然，数值不稳定的方法一般在实际计算中不能采用。数值不稳定的现象属于误差危害现象。下面讨论误差危害现象的其他表现及如何避免问题。

1.3.2 避免有效数字的损失

在数值计算中，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，“小数”的作用可能消失，即出现“大数”吃“小数”的现象。

例2 用三位十进制数字计算

$$x = 101 + \delta_1 + \delta_2 + \cdots + \delta_{100},$$

其中 $0.1 \le \delta_i \le 0.4, i = 1,2\cdots,100$。 如果我们自左至右逐个相加，则所有的 δ_i 都会被舍掉，得 $x \approx 101$ 。但若把所有的 δ_i 先加起来，再与101相加，就有

$$111 = 101 + 100 \times 0.1 \le x \le 101 + 100 \times 0.4 = 141$$

可见，计算的次序会产生很大的影响。这是因为用计算机计算时，在运算中要“对阶”，对阶引起了大数吃小数的现象。大数吃小数在有些情况下是允许的，但有些情况下则造成谬误。在数值计算中，两个相近数相减会使有效数字严重损失。

例3 求实系数二次方程 $ax^2+bx+c=0$ 的根，其中 $b^2-4ac<0, ab\neq 0$。

解 考虑两种解法。

算法 1:

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

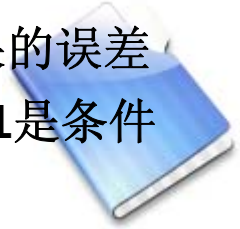
算法2:

$$x_1=\frac{-b-sgn(b)\sqrt{b^2-4ac}}{2a},\ x_2=\frac{c}{ax_1}$$

其中sgn表示取数的符号，即

$$sgn(b)=\begin{cases}1 & b>0\\ -1 & b<0\end{cases}$$

对算法1，若 $b^2>>4ac$ ，则是不稳定的，否则是稳定的。这是因为前一种情况的分子有一个相近数相减，会大量损失有效数字，从而有一个结果的误差很大。算法2不存在这个问题，在任何情况下都是稳定的。因此称算法1是条件稳定的，算法2是无条件稳定的。



例如，对于方程

$$x^2 + 62.10x + 1.000 = 0$$

用4位有效数字计算，结果如下：

算法1： $x_1 = -62.08, x_2 = -0.02000$

算法2： $x_1 = -62.08, x_2 = -0.01611$

准确解是 $x_1 = -62.083892\cdots, x_2 = -0.016107237$。这里 $b^2 >> 4ac$ 所以算法1不稳定，舍入误差对 x_2 的影响大。

遇到两相近数相减的情形，可通过变换计算公式来避免或减少有效数字的损失。例如，我们有如下的变换公式：

$$\frac{1-\cos x}{\sin x}=\frac{\sin x}{1+\cos x}$$

$$\lg x_1-\lg x_2=\lg\frac{x_1}{x_2}$$

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

如果无法改变算法，则采用增加有效位数进行计算，或在计算上采用双精度运算但这要增加机器计算的时间和多占内存单元。

1.3.3 减少运算次数

在数值计算中，要注意简化计算步骤，减少运算次数，这也是数值分析所要研究的重要内容。同样一个计算问题，如果能减少运算次数，不但可以节省计算机的计算时间，还能减少误差的积累。下面举例说明简化计算公式的重要性。

例4 给定x，计算多项式

$$P_n(x)=a_nx^n+a_{n-1}x^{n-1}\cdots+a_0$$

的值。如果我们先求a_kx^k，需要进行k次乘法，在相加，则需要 $n(n+1)/2$ 次乘法和n 次加法才能得到一个多项式的值。如果我们将多项式写成下面的形式

$$P_n(x)=x\{x\cdots[x(a_nx+a_{n-1})+a_{n-2}]+\cdots+a_1\}+a_0$$

则只需n次乘法和n次加法即可得到一个多项式的值，这就是著名的秦九韶算法，可描述为

$$\begin{cases} u_n=a_n \\ u_k=u_{k+1}x+a_k \quad k=n-1,n-2\cdots 0 \end{cases}$$

最后有 $u_0=P_n(x)$

例5 利用级数

$$\ln(1+x)=\sum_{n=1}^{\infty}(-1)^{n+1}\frac{x^n}{n}$$

计算 $\ln 2$ ，若要精确到 10^{-5}，要计算**10**万项求和。这一方面计算量很大，另一方面舍入误差的积累也十分严重。

如果该用级数

$$\ln\frac{1+x}{1-x}=2\left(x+\frac{x^3}{3!}+\frac{x^5}{5!}+\cdots+\frac{x^{2n+1}}{(2n+1)!}+\cdots\right),$$

来计算 $\ln 2$，取 $x=1/3$ ，则只要计算前**9**项，截断误差便小于 10^{-10} 。