数值分析

Numerical Analysis

- 通常解决数学问题的思维方式

实际问题 → 数学模型 → 结果表达式 → 结果

- 数值分析的思维方式

实际问题 → 数学模型 → 算法设计 → 编程计算结果

后者也正是利用计算机解决科学计算的过程。

众所周知，电子计算机实质上只会做加减乘除等基本运算，研究怎样通过计算机所能执行的基本运算，求得各类数学问题的数值解或近似解就是数值计算的根本课题。由基本运算及运算顺序的规定所构成的完整的解题步骤，称为算法。数值计算的根本任务就是研究算法。

例　计算多项式

$$0.0625x^4 + 0.425x^3 + 1.215x^2 + 1.912x + 2.1296$$

的值。

- 算法1：按原形计算：
  需做十次乘法、四次加法

- 算法2：上述多项式化为

$$(((0.0625x + 0.425)x + 1.215)x + 1.912)x + 2.1296$$

则需做四次乘法、四次加法。

第一章 误差 /* Error */

§1 误差的背景介绍 /* Introduction */

1. 来源与分类 /* Source & Classification */

- 从实际问题中抽象出数学模型

  —— 模型误差 /* Modeling Error */

- 通过测量得到模型中参数的值

  —— 观测误差 /* Measurement Error */

- 求近似解 —— 方法误差 (截断误差 /* Truncation Error */)

- 机器字长有限 —— 舍入误差 /* Roundoff Error */

§2 误差与有效数字 /* Error and Significant Digits */

- 绝对误差 /* absolute error */

$e^* = x^* - x$ 其中x为精确值，x^*为x的近似值。

$|e^*|$的上限记为ε^*，称为绝对误差限 /* accuracy */，工程上常记为 $x = x^* \pm \varepsilon^*$，例如：$\int_0^1 e^{-x^2} dx = 0.743 \pm 0.006$

$$\int_0^1 e^{-x^2} dx \approx 0.743$$

- 相对误差 /* relative error */ $e_r^* = \dfrac{e^*}{x}$

x 的相对误差上限 /* relative accuracy */ 定义为 $\varepsilon_r^* = \dfrac{\varepsilon^*}{|x^*|}$

➢有效数字 /* significant digits */

定义 （有效数字）若近似值 x^* 的误差限是某一位的半个单位，该位到 x^* 的第一位非零数字共有 n 位，则 e^* 有 n 位有效数字。

$$\pi = 3.14159265\cdots$$

按四舍五入原则若取四位小数得 $\pi \approx 3.1416$，取五位小数则有 $\pi \approx 3.14159$，它们的绝对误差不超过末位数的半个单位，即

$$|\pi - 3.1416| \le \frac{1}{2} \times 10^{-4}, |\pi - 3.14159| \le \frac{1}{2} \times 10^{-5}$$

用科学计数法，记$x^* = \pm 0.a_1a_2\cdots a_n \times 10^m$（其中$a_1 \neq 0$）。若$|x - x^*| \leq 0.5 \times 10^{m-n}$(即$a_n$的截取按四舍五入规则），则称$x^*$为有$n$位有效数字，精确到$10^{m-n}$。

例：$\pi = 3.14159265358979 32\cdots\cdots$；　$\pi^* = 3.1415$

问：π^*有几位有效数字？请证明你的结论。

证明：$\because \pi^* = 0.31415 \times 10^1$，

$$\text{and } |\pi^* - \pi| < 0.5 \times 10^{-3} = 0.5 \times 10^{1-4}$$

$\therefore \pi^*$有4位有效数字，精确到小数点后第3位。

➢有效数字与相对误差的关系

有效数字 $\Rightarrow$ 相对误差限

已知 x^* 有 n 位有效数字，则其相对误差限为

$$\varepsilon_r^* = \left|\frac{\varepsilon^*}{x^*}\right| = \frac{0.5\times 10^{m-n}}{0.a_1 a_2 \cdots a_n \times 10^m} = \frac{10^{-n}}{2\times 0.a_1 \cdots}$$

$$\le \frac{1}{2a_1}\times 10^{-n+1}$$

相对误差限 $\Rightarrow$ 有效数字

已知 x^* 的相对误差限可写为 $\varepsilon_r^* = \dfrac{1}{2(a_1+1)}\times 10^{-n+1}$

则 $|x - x^*| \le \varepsilon_r^* \cdot |x^*| = \dfrac{10^{-n+1}}{2(a_1+1)}\times 0.a_1 a_2 \cdots \times 10^m$

$$< \frac{10^{-n+1}}{2(a_1+1)}\cdot(a_1+1)\times 10^{m-1} = 0.5\times 10^{m-n}$$

可见 x^* 至少有 n 位有效数字。

例：为使π^*的相对误差小于0.001%, 至少应取几位有效数字?

解：假设 π^* 取到 n 位有效数字，则其相对误差上限为

$$\varepsilon_r^* \le \frac{1}{2a_1} \times 10^{-n+1}$$

要保证其相对误差小于0.001%，只要保证其上限满足

$$\varepsilon_r^* \le \frac{1}{2a_1} \times 10^{-n+1} < 0.001\%$$

已知 $a_1 = 3$，则从以上不等式可解得 $n > 6 - \log 6$，即 $n \ge 6$，应取 $\pi^* = 3.14159$。$\blacksquare$

§ 3 几点注意事项 /* Remarks */

1. 避免相近二数相减

例：$a_1 = 0.12345$，$a_2 = 0.12346$，各有5位有效数字。

而 $a_2 - a_1 = 0.00001$，只剩下1位有效数字。

几种经验性避免方法：

$$\sqrt{x+\varepsilon} - \sqrt{x} = \frac{\varepsilon}{\sqrt{x+\varepsilon} + \sqrt{x}}; \qquad \ln(x+\varepsilon) - \ln x = \ln\left(1 + \frac{\varepsilon}{x}\right);$$

当 $|x| << 1$ 时：$1 - \cos x = 2\sin^2 \frac{x}{2}$；

$$e^x - 1 = x\left(1 + \frac{1}{2}x + \frac{1}{6}x^2 + \dots\right)$$

2. 避免小分母：分母小会造成浮点溢出 /* over flow */

3. 避免大数吃小数

例：用单精度计算 $x^2-(10^9+1)x+10^9=0$ 的根。

精确解为 $x_1=10^9$，$x_2=1$

算法1：利用求根公式 $x=\dfrac{-b\pm\sqrt{b^2-4ac}}{2a}$

在计算机内，10^9存为0.1×10^{10}，1存为0.1×10^1。做加法时，两加数的指数先向大指数对齐，再将浮点部分相加。

即1 的指数部分须变为10^{10}，则：$1 = 0.0000000001 \times 10^{10}$，取单精度时就成为：

$10^9+1=0.10000000\times10^{10}+0.00000000\times10^{10}=0.10000000\times10^{10}$

$$\Rightarrow x_1=\frac{-b+\sqrt{b^2-4ac}}{2a}=10^9,\quad x_2=\frac{-b-\sqrt{b^2-4ac}}{2a}=0$$

大数吃小数

✍算法2：先解出 $x_1 = \dfrac{-b - sign(b)\cdot\sqrt{b^2-4ac}}{2a} = 10^9$

再利用 $x_1 \cdot x_2 = \dfrac{c}{a} \Rightarrow x_2 = \dfrac{c}{a\cdot x_1} = \dfrac{10^9}{10^9} = 1$

注：求和时从小到大相加，可使和的误差减小。

例：按从小到大、以及从大到小的顺序分别计算

$$1 + 2 + 3 + \ldots + 40 + 10^9$$

4. 先化简再计算，减少步骤，避免误差积累。

$$algorithm\,1: x^{16} = \underbrace{x\cdot x\cdot x\cdot x\cdots x}_{16}$$

$$algorithm\,2: x^{16} = (((x^2)^2)^2)^2$$

一般来说，计算机处理下列运算的速度为 $(+,-) > (\times,\div) > (\exp)$

5. 选用稳定的算法。