

数学课程电子教案

数值分析

Numerical Analysis



任课教师：王琼 李红

教材 (Text Book)

数值分析 (第二版) 李红 编著 (华中科技大学出版社)

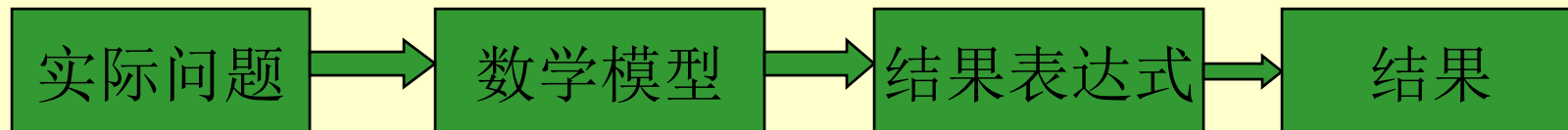
参考书目 (Reference)

➤ **数值分析学习辅导 李红、徐长发编著**
(华工出版社)

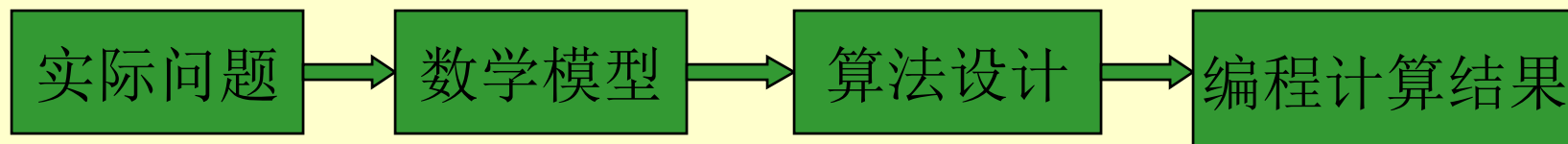
绪 论

数值分析概括为用计算机求解数学问题的**数值方法和理论**。我们在工程计算和科学实验中会遇到诸如线性方程组的求解、微分、积分、微分方程的求解等常见的数学问题。这些问题大家已经在相应的数学课程中学过，通常的思维是求出（或推导出）结果的表达式，然后应用表达式寻求答案。但在实际中结果的表达式（又称解析解），例如满足某个微分方程的函数，某个函数的原函数是不易求得的，这就需要数学理论与计算机结合起来，寻求（设计）合适的算法以期得到问题的近似数值解，这就是**数值分析**所要研究的问题。下面是两种思维过程的对比：

•通常解决数学问题的思维方式



•数值分析的思维方式



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 计算任意角的三角函数，如 $\sin x$ 。不调用库函数，计算机是不能直接计算 $\sin x$ 的。根据微分学的Taylor公式，我们有：

$$\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \cdots + (-1)^n \frac{x^{2n+1}}{(2n+1)!} + R_{2n+1}(x)$$

等式的右端就只是乘法与加法的循环运算。取

$$\sin x \approx x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \cdots + (-1)^n \frac{x^{2n+1}}{(2n+1)!}$$

通过编制程序我们就可以计算 $\sin x$ 的近似值。事实上，计算机语言常用的数学运算的标准函数也可用这种方法写成。

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。

例 解线性方程组

$$a_{11}x_1 + a_{12}x_2 + \dots + a_{1n}x_n = b_1$$

$$a_{21}x_1 + a_{22}x_2 + \dots + a_{2n}x_n = b_2$$

.....

$$a_{n1}x_1 + a_{n2}x_2 + \dots + a_{nn}x_n = b_n$$

按Cramer法则求解, 即 $x_k = \frac{D_k}{D}, k = 1, 2, \dots, n$

其中

$$D = \begin{vmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{vmatrix}$$

D_k 是把 D 中第 k 列

$$(a_{1k} \ a_{2k} \ \dots \ a_{nk})^T$$

换为 $(b_1 \ b_2 \ \dots \ b_n)^T$

这要计算 $n+1$ 行列式, 做 n 法。

而每个行列式包含 $n!$ 乘积, 每个乘积
需做 $n-1$ 次乘法. 这样共需做

$$A_n = (n+1)!(n-1) + n$$

次乘除法。当 $n=20$ 时, $A_{20} \approx 9.7 \times 10^{20}$ 这意味着在每秒做一亿次乘除法的计算机上, 要做
30多万年!

因此, 在构造算法时, 还应考虑如何
计算, 才能**既快又省**。

The following problem can be solved either the easy way or the hard way.

Two trains 200 miles apart are moving toward each other; each one is going at a speed of 50 miles per hour. A fly starting on the front of one of them flies back and forth between them at a rate of 75 miles per hour. It does this until the trains collide and crush the fly to death. What is the total distance the fly has flown?

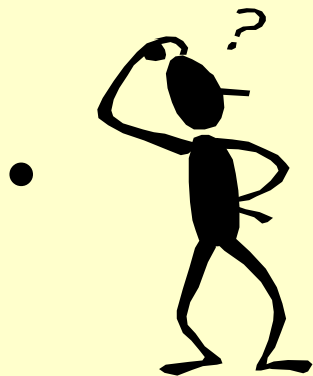
The fly actually hits each train an infinite number of times before it gets crushed, and one could solve the problem the hard way with pencil and paper by summing an infinite series of distances. The easy way is as follows: Since the trains are 200 miles apart and each train is going 50 miles an hour, it takes 2 hours for the trains to collide. Therefore the fly was flying for two hours. Since the fly was flying at a rate of 75 miles per hour, the fly must have flown 150 miles. That's all there is to it.

When this problem was posed to John von Neumann, he immediately replied, "150 miles."

"It is very strange," said the poser, "but nearly everyone tries to sum the infinite series."

"What do you mean, strange?" asked Von Neumann. "That's how I did it!"





提问：数值分析是做什么用的？

输入复杂问题或运算

$$\sqrt{x}, a^x, \ln x, A\bar{x} = \bar{b},$$

$$\int_a^b f(x)dx, \frac{d}{dx} f(x), \dots$$



数值
分析



+

-

×

÷



计算机



近似解

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 */

例：近似计算 $\int_0^1 e^{-x^2} dx = 0.747\dots\dots$

解法之一： 将 e^{-x^2} 作Taylor展开后再积分

$$\int_0^1 e^{-x^2} dx = \int_0^1 \left(1 - x^2 + \frac{x^4}{2!} - \frac{x^6}{3!} + \frac{x^8}{4!} - \dots\right) dx$$

$$= \underbrace{1 - \frac{1}{3} + \frac{1}{2!} \times \frac{1}{5} - \frac{1}{3!} \times \frac{1}{7}}_{S_4} + \underbrace{\frac{1}{4!} \times \frac{1}{9} - \dots}_{R_4} \quad \text{/* Remainder */}$$

取 $\int_0^1 e^{-x^2} dx \approx S_4$,

则 $R_4 = \int_0^1 \left(\frac{x^8}{4!} - \frac{x^{10}}{5!} + \dots \right) dx$ 由留下部分引起截断误差 /* Truncation Error */
/* included terms */

这里 $|R_4|$

引起

$$S_4 = 1 - \frac{1}{3} + \frac{1}{10} - \frac{1}{42} \approx 1 - 0.333 + 0.1$$

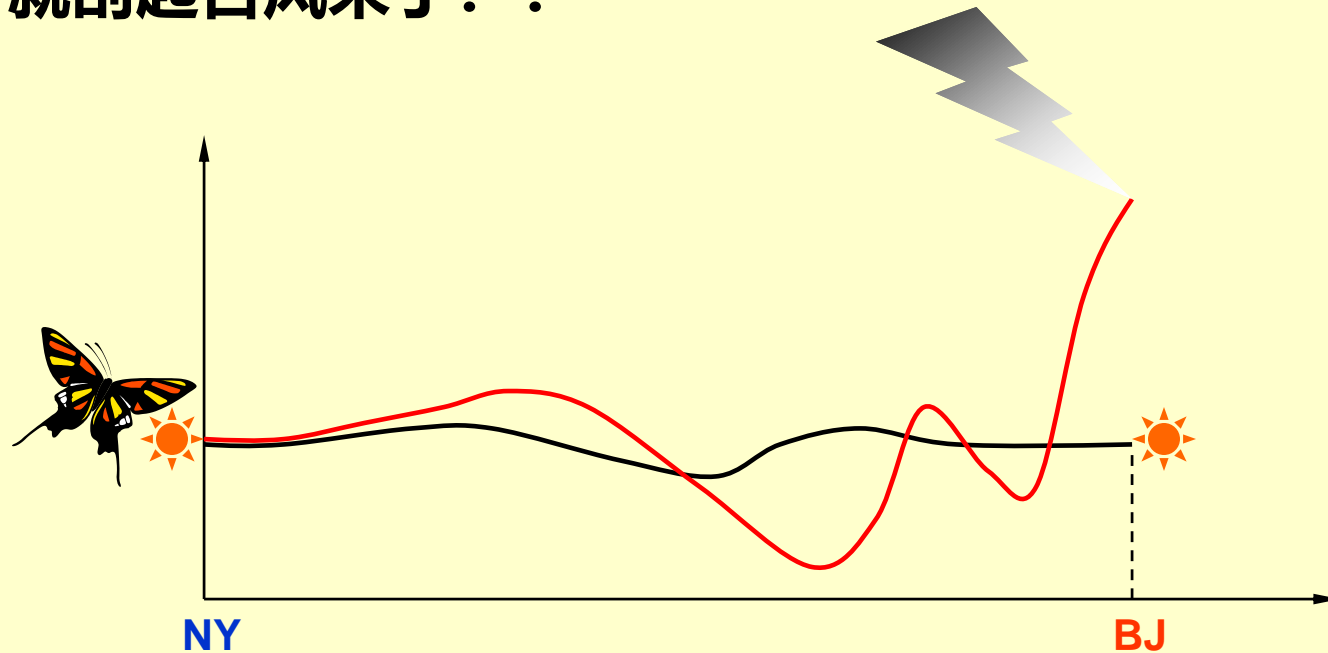
由截去部分引起
/* excluded terms */

| 舍入误差 /* Roundoff Error */ | $< 0.0005 \times 2 = 0.001$

| 计算 $\int_0^1 e^{-x^2} dx$ 的总体误差 | $< 0.005 + 0.001 = 0.006$

2. 传播与积累 /* Spread & Accumulation */

例：蝴蝶效应 —— 纽约的一只蝴蝶翅膀一拍，风和日丽的北京就刮起台风来了？！



以上是一个**病态问题** /* ill-posed problem*/

关于本身是病态的问题，我们还是留给数学家去头痛吧！

例： 计算 $I_n = \frac{1}{e} \int_0^1 x^n e^x dx$, $n = 0, 1, 2, \dots$

公式一： $I_n = 1 - n I_{n-1}$

$$I_0 = \frac{1}{e} \int_0^1 e^x dx = 1 - \frac{1}{e} \approx 0.63212056 \xrightarrow{\text{记为}} I^*$$

注意此公式精确成立

则初始误差 $|E_0| = |I_0 - I_0^*| < 0.5 \times 10^{-8}$

$$\frac{1}{e} \int_0^1 x^n \cdot e^0 dx < I_n < \frac{1}{e} \int_0^1 x^n \cdot e^1 dx \quad \therefore \frac{1}{e(n+1)} < I_n < \frac{1}{n+1}$$

$$I_1^* = 1 - 1 \cdot I_0^* = 0.36787944$$

... ..

$$I_{10}^* = 1 - 10 \cdot I_9^* = 0.08812800$$

$$I_{11}^* = 1 - 11 \cdot I_{10}^* = 0.03059200$$

$$I_{12}^* = 1 - 12 \cdot I_{11}^* = 0.63289600 \text{ ?}$$

$$I_{13}^* = 1 - 13 \cdot I_{12}^* = -7.2276480 \text{ ??}$$

$$I_{14}^* = 1 - 14 \cdot I_{13}^* = 94.959424 \text{ ? !}$$

$$I_{15}^* = 1 - 15 \cdot I_{14}^* = -1423.3914 \text{ !!}$$



What
happened
?!

考察第 n 步的误差 $|E_n|$

$$|E_n| = |I_n - I_n^*| = |(1 - nI_{n-1}) - (1 - nI_{n-1}^*)| = n|E_{n-1}| = \cdots = n!|E_0|$$

可见初始的小扰动 $|E_0| < 0.5 \times 10^{-8}$ 迅速积累，误差呈递增走势。

造成这种情况的是**不稳定的算法** /* unstable algorithm */
我们有责任改变。

✎ **公式二：** $I_n = 1 - n I_{n-1} \Rightarrow I_{n-1} = \frac{1}{n} (1 - I_n)$

方法：先估计一个 I_N ，再反推要求的 I_n ($n \ll N$)。

注意此公式与公式 $\frac{1}{e(N+1) + \frac{1}{N+1}}$
在理论上**等价**。

$$\text{可取 } I_N^* = \frac{1}{2} \left[\frac{1}{e(N+1)} + \frac{1}{N+1} \right] \approx I_N$$

$$\text{当 } N \rightarrow +\infty \text{ 时, } |E_N| = |I_N - I_N^*| \rightarrow 0$$

$$\text{取 } I_{15}^* = \frac{1}{2} \left[\frac{1}{e \cdot 16} + \frac{1}{16} \right] \approx 0.042746233$$

$$\Rightarrow I_{14}^* = \frac{1}{15} (1 - I_{15}^*) \approx 0.063816918$$

$$I_{13}^* = \frac{1}{14} (1 - I_{14}^*) \approx 0.066870220$$

$$I_{12}^* = \frac{1}{13} (1 - I_{13}^*) \approx 0.071779214$$

$$I_{11}^* = \frac{1}{12} (1 - I_{12}^*) \approx 0.077351732$$

$$I_{10}^* = \frac{1}{11} (1 - I_{11}^*) \approx 0.083877115$$

$$\vdots$$

$$I_1^* = \frac{1}{2} (1 - I_2^*) \approx 0.36787944$$

$$I_0^* = \frac{1}{1} (1 - I_1^*) \approx 0.63212056$$

We just got lucky?



考察反推一步的误差：

$$|E_{N-1}| = \left| \frac{1}{N}(1-I_N) - \frac{1}{N}(1-I_N^*) \right| = \frac{1}{N} |E_N|$$

以此类推，对 $n < N$ 有：

$$|E_n| = \frac{1}{N(N-1) \dots (n+1)} |E_N|$$

误差逐步递减，这样的算法称为**稳定的算法** /* stable algorithm */

在我们今后的讨论中，**误差**将不可避免，
算法的**稳定性**会是一个非常重要的话题。

§ 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: $\int_0^1 e^{-x^2} dx = 0.743 \pm 0.006$
Oh yeah? Then tell

注: e^* 理论上讲是唯一确定的, 可能取正, 也可能取负。
 $\varepsilon^* > 0$ 不唯一, 当然 ε^* 越小越具有参考价值。



Of course mine is more accurate! The accuracy relates to not only the absolute error, but also to the size of the exact value.



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

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

A mathematician, a physicist, and an engineer were traveling through Scotland when they saw a sign that read "Simple window of the tree."

Now I wouldn't call it simple.

Say ... what is the relative error of $20\text{cm} \pm 1\text{cm}$?

"Aha," says the

"Hmm," says the physicist, "You mean that some Scottish sheep are black."

"No," says the mathematician, "All we know is that there is at least one sheep in Scotland, and that at least one side of that one sheep is black!"



定义可见, e_r^* 实际上被偷换成了 $\frac{e^*}{x^*}$, 而后才考察上限。那么这样的偷换是否合法?

严格的说法是, $\frac{e^*}{x}$ 与 $\frac{e^*}{x^*}$ 是否反映了同一数量级的误差?



➤ 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 x^* 有 n 位有效数字。

$$\pi = 3.14159265\dots$$

按四舍五入原则若取四位小数得 ≈ 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

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

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: $\pi = 3.1415926535897932 \cdots$; $\pi^* = 3.1415$

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。

例: 以下数字是经四舍五入得到的, 判定各有几位有效数字。

187.9325 0.00369246 3.1415926 2.000072

注: 187.9325 有 4 位有效数字, 而 0.00369246 有 6 位有效数字, 3.1415926 有 8 位有效数字, 2.000072 有 7 位有效数字。

如果写成 0.123×10^5 , 则表示只有 3 位有效数字。

数字末尾的 0 不可随意省去!

► 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

$$\begin{aligned}\varepsilon_r^* &= \left| \frac{\varepsilon^*}{x^*} \right| = \frac{0.5 \times 10^{m-n}}{0.a_1 a_2 \cdots a_n \times 10^m} = \frac{10^{-n}}{2 \times 0.a_1 \cdots} \\ &\leq \frac{1}{2a_1} \times 10^{-n+1}\end{aligned}$$

👉 相对误差限 \Rightarrow 有效数字

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

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

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

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

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

要保证其相对误差小于 0.001%，只要保证其上限满足

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

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



§ 4 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：

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

当 $|x| \ll 1$ 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 $x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$

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：

$$10^9 + 1 = 0.100000000 \times 10^{10} + 0.000000001 \times 10^{10} = 0.100000001 \times 10^{10}$$

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

(Note: A blue callout bubble points to the term $4ac$ in the denominator of x_2 , with the text "大数吃小数" (Large number eats small number) written inside.)

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

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

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

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

4. 先化简再计算, 减少步骤, 避免误差积累。

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

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

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。

HW: p.13-14 #3, #12, #13