

计算方法

郑太英

taiying_zheng@zju.edu.cn

13588066482

课程教材

计算方法

主编：易大义

出版社：浙江大学出版社

▣ 推荐教材或主要参考书：

- ▣ 《数值分析》 Richard L.Burden Thomson Learning
- ▣ 《数值分析》英文版 John H.Mathews 电子工业出版社
- ▣ 《MATLAB 数值分析与应用》 张德丰 国防工业出版社
- ▣ 《工程数值方法》 Steven C.Chapra Mc Graw Hill

网络资源

▣ Online Course Material

- *Numerical Methods, Stuart Dalziel University of Cambridge*
- *Lectures on Numerical Analysis, Dennis Deturck and Herbert S. Wilf University of Pennsylvania*
- *Numerical methods, John D. Fenton University of Karlsruhe*
- *Numerical Methods for Science, Technology, Engineering and Mathematics, Autar Kaw University of South Florida*
- *Numerical Analysis Project, John H. Mathews California State University, Fullerton*
- *Numerical Methods - Online Course, Aaron Naiman Jerusalem College of Technology*
- *Numerical Methods for Physicists, Anthony O'Hare Oxford University*
- *Lectures in Numerical Analysis, R. Radok Mahidol University*
- *Introduction to Numerical Analysis for Engineering, Henrik Schmidt Massachusetts Institute of Technology*

课件下载：学在浙大平台

考核标准

- ▣ 线上学习：20分（学校规定）
- ▣ 作业：5分
- ▣ 上机：15分
- ▣ 期末考试：60分
 - ▣ 填空、简答或小计算题、大计算题。
- ▣ Bonus：5分
 - ▣ 视课堂参与度而定，每次0.5或1分

课程内容

1	绪论（误差）	1次课
----------	---------------	------------

2	非线性方程求解	2次课
----------	----------------	------------

3	线性方程组求解	4次课
----------	----------------	------------

4	拟合插值	3次课
----------	-------------	------------

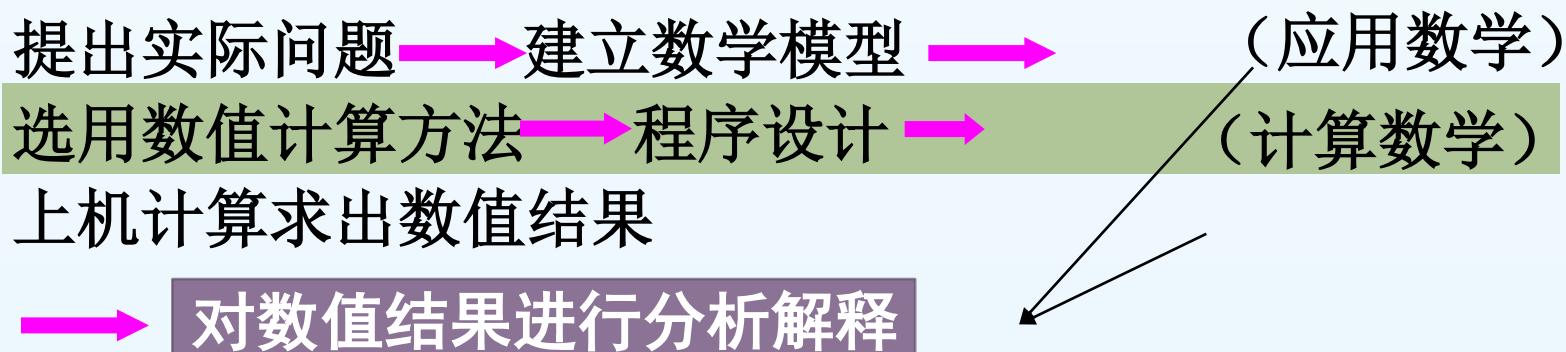
5	数值积分与微分	2次课
----------	----------------	------------

6	微分方程数值解法	3次课
----------	-----------------	------------

数值计算：研究怎样利用算盘、算尺、计算器、**计算机**等工具，来求出**数学问题的数值解答**的学问。

本课程介绍计算机上常用的数值计算方法。

计算机解决科学计算问题时经历的几个过程：



数值计算的研究对象：求解各种数学问题的数值方法的设计、分析；有关的数学理论和软件实现。

计算机实质上只会做加减乘除等算术运算和一些逻辑运算。由这些基本运算及运算顺序的规定构成的完整的解题步骤，称为算法。它可用框图、算法语言、数学语言或自然语言描述。用计算机算法语言描述的算法称为计算机程序。

数值计算是用计算机进行数学计算的，而计算机的运算速度快，可以承担各种计算工作。因此，很多人甚至为数不少的一些科研人员，常常认为只要把涉及到的一些数学公式，用一种计算机语言正确编程，计算机就一定能给出正确的结果。

例1.1 求解线性方程组

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2 \\ \cdots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n = b_n \end{cases}$$

克莱姆（Cramer）法则 求解

克莱姆 (Cramer) 法则

1.1 数值计算的任务与特点(续)

如果 $\det(\mathbf{A}) \neq 0$ ，则方程组有唯一解 $x_i = \frac{\mathbf{D}_i}{\det(\mathbf{A})} \quad (i = 1, 2, \dots, n)$

$$\det(\mathbf{A}) = \begin{vmatrix} a_{11} & a_{12} & \cdots & a_{1,i-1} & a_{1i} & a_{1,i+1} & \cdots & a_{1,n} \\ a_{21} & a_{22} & \cdots & a_{2,i-1} & a_{2i} & a_{2,i+1} & \cdots & a_{2,n} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ a_{n1} & a_{n2} & \cdots & a_{n,i-1} & a_{ni} & a_{n,i+1} & \cdots & a_{n,n} \end{vmatrix}$$
$$\mathbf{D}_i = \begin{vmatrix} a_{11} & a_{12} & \cdots & a_{1,i-1} & b_1 & a_{1,i+1} & \cdots & a_{1,n} \\ a_{21} & a_{22} & \cdots & a_{2,i-1} & b_2 & a_{2,i+1} & \cdots & a_{2,n} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ a_{n1} & a_{n2} & \cdots & a_{n,i-1} & b_n & a_{n,i+1} & \cdots & a_{n,n} \end{vmatrix}$$

用这种方法解一个 n 元方程组,要算 $n+1$ 个 n 阶行列式的值, **总共需要 $(n+1)n!(n-1)$ 次乘法。**

当 n 较大时, 计算量相当惊人。

如: $n=20$ 时, $(n+1)n!(n-1)\approx 9.7*10^{20}$

工作量太大, 不适用在计算机上求解高维方程组。

解线性方程组有许多实用的算法, 可大大减少计算工作量。
在数值计算中要注意算法计算量的分析, 并尽量节省存储量。

例 1.2 求一元二次方程 $x^2 - (10^9 + 4)x + 4 * 10^9 = 0$ 的根时

常用的公式是：
$$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

若用这个公式编程并在字长为8位的计算机上计算，得到的结果为：

$$x_1 = 10^9, x_2 = 0$$

本题的两个根应为： $x_1 = 10^9, x_2 = 4$

出现这一错误的原因是受机器字长的限制所引起的误差造成的。

在设计算法时，要注意算法的误差分析。



例 1.3 计算数列 $I_n = \int_0^1 \frac{x^n}{x+5} dx$

解 因为

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

所以可以得到 I_n 的递推公式:

$$I_n = -5I_{n-1} + \frac{1}{n} \quad (n=1, 2, \dots)$$

由 $I_0 = \int_0^1 \frac{1}{x+5} dx = \ln 6 - \ln 5$ 可依次计算 I_1, I_2, \dots



如果按上述方法用计算机编程计算，会发现当n较大时， I_n 的计算结果会出现负数。

例如在字长为8的计算机上编程计算，可出现

$$I_{12} = -0.32902110 \times 10^{-2}$$

这显然是错误的

$$I_n = \int_0^1 \frac{x^n}{x+5} dx \geq 0$$

出现这一错误，是由于受机器字长限制引起的误差在计算过程中的传播造成的。

一个好的算法应能控制误差的传播，即应是所谓**数值稳定**的算法

数值计算方法的任务：为各种数学问题的数值解答，提供最有效的算法。

最有效的算法：适用范围广、运算工作量少，需用存储单元少，逻辑结构简单，便于编写计算机程序，而且计算结果可靠。

误差估计：由于计算机计算通常是近似的，因而一般要求算法能估计误差

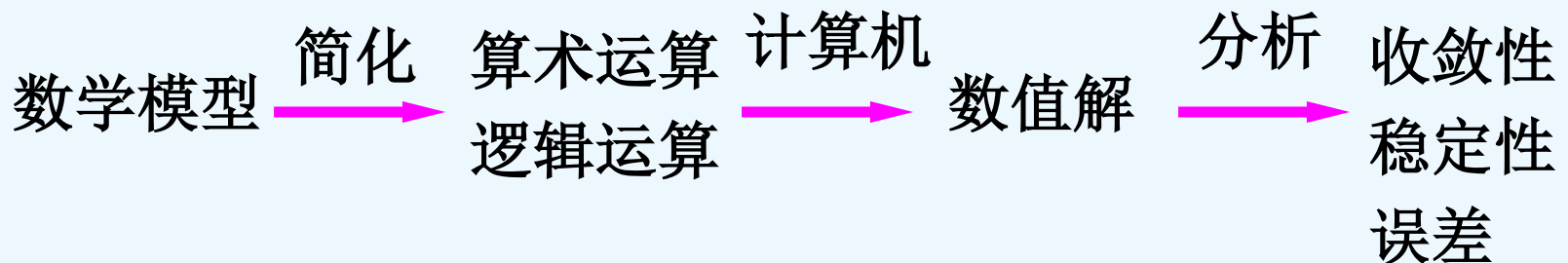
稳定性：计算过程中，误差能得到控制，各步误差对计算结果不致产生过大影响；不合适的算法会导致计算误差达到不能容许的地步，而使计算最终失败。

收敛性：通过增加计算量，能使近似计算解充分接近理论解。

没有一种算法处处有效，而且各种算法在计算过程中往往都会出现某些问题。因此，为用计算机解决实际问题，除正确提出数学问题（数学模型）外，必须针对具体问题选择或改造适当的算法，清楚计算过程中可能发生的问题。

了解算法的设计原理，充分熟悉算法的运算过程，知道算法的长处和短处；

分析算法的优劣，特别是算法的稳定性、收敛性和误差估计，需要较多的数学推导，只需知道结论。



计算方法的特点

- ▣ 面向计算机，要根据计算机特点提供实际可行的有效算法，如加、减、乘、除和逻辑运算；
- ▣ 有可靠的理论分析，能任意逼近并达到精度要求，对近似算法要保证收敛性和数值稳定性，还要对误差进行分析。这都建立在相应数学理论的基础上；
- ▣ 节省时间、节省存储量；
- ▣ 要有数值实验，即任何一个算法除了从理论上要满足上述三点外，还要通过数值试验证明是行之有效的。

1.2 计算机中的数系与运算特点

1.2.1

计算机的数系

在计算机中，任一实数 x 可表示成

$$x = \pm 10^c \times 0.a_1a_2a_3\cdots \quad \text{十进制浮点数}$$

其中 $a_i \in \{0, 1, 2, 3, \dots, 9\} (i = 1, 2, 3, \dots)$ c 为整数

一般地，可定义 β 进制浮点数

$$x = \pm \beta^c \times 0.a_1a_2a_3\cdots$$

其中 $a_i \in \{0, 1, 2, 3, \dots, \beta - 1\} (i = 1, 2, 3, \dots)$

在计算机中，由于机器本身的限制，一般实数被表示为：

阶码

尾数

正整数，
计算机的
字长

$$x = \pm \beta^c \times 0.a_1a_2a_3 \cdots a_t$$

其中 $a_i \in \{0, 1, 2, 3, \cdots, \beta - 1\} (i = 1, 2, \cdots, t)$

c 为整数满足 $L \leq c \leq U$, L 和 U 为固定整数。

对不同的计算机， t ， L 和 U 是不同的。

t位β进制浮点数

t位β进制浮点数

这样一些数的全体

$$F(\beta, t, L, U) = \{\pm \beta^c \times 0.a_1 a_2 \cdots a_t \mid a_i \in \{0, 1, 2, \dots, \beta - 1\} \\ (i = 1, 2, \dots, t), L \leq c \leq U\}$$

➡ 机器数系

它是计算机进行实数运算所用的数系，一般 β 取**2**，**8**，**10**和**16**。

机器数系 F 是一个离散的有限集合，分布也不均匀。
在 F 中有一个最大正数 M 和最小正数 m 。

如数系 $F(10, 4, -99, 99)$ 中

$$M = 10^{99} \times 0.9999,$$

$$m = 10^{-99} \times 0.0001$$

计算机存储或运算的数不能过大或过小。

若一个非零实数的绝对值大于 M ，则计算机产生上溢错误；若绝对值小于 m ，则计算机产生下溢错误。

上溢时，计算机中断程序处理，下溢时，计算机将此数用零表示继续执行程序。

无论是上溢还是下溢，都称为溢出错误。

1.2.2 计算机对数的接受与处理

问题：计算机是怎样接收与处理一般实数的？

设非零实数 x 是计算机接收的数，则计算机对其的处理方法是：

1) 若 $x \in F(\beta, t, L, U)$ ，则原样接收 x

2) 若 $x \notin F(\beta, t, L, U)$ ，但 $m \leq |x| \leq M$ ，则用 $F(\beta, t, L, U)$

中最接近 x 的数 $fl(x)$ 表示并记录 x ，以便后面处理。→四舍五入

计算机对接收到的数只能做加减乘除四则运算，其运算方式是：

1) 加减法：先向上对阶，后运算，再舍入。

2) 乘除法：先运算，再舍入。

如数系 $F(10, 4, -90, 90)$

$$fl(x_1) = 0.2337 \times 10^{-1}, fl(x_2) = 0.3364 \times 10^2$$

是计算机接收到的两个实数，则有

$$fl(x_1 + x_2) = fl(0.2337 \times 10^{-1} + 0.3364 \times 10^2)$$

$$\text{对阶} = fl(0.0002337 \times 10^2 + 0.3364 \times 10^2)$$

$$\text{运算} = fl(0.3366337 \times 10^2)$$

$$\text{舍入} = 0.3366 \times 10^2$$

$$fl(x_1 x_2) = fl(0.2337 \times 10^{-1} \times 0.3364 \times 10^2)$$

$$\text{运算} = fl(0.7861668 \times 10^0)$$

$$\text{舍入} = 0.7862 \times 10^0 = 0.7862$$

计算机接收和处理实数的上述特点，往往使得数学上完美的公式在计算机编程计算时，却得不到正确的结果。

在计算机的数系 $F(\beta, t, L, U)$ 中，把尾数第一位 $a_1 \neq 0$ 的数称为规格化的浮点数。

用规格化的浮点数表示一个实数，具有形式唯一和精度高的特点，

但并不是 $F(\beta, t, L, U)$ 中每一个数都可以用规格化浮点数表示。

例如数 $0.00...1 \times \beta^L$ ，就不能表示为规格化浮点数。

1.3 数值计算的误差

在研究算法时，
必须注重误差分析，
否则，一个合理的算
法也可能得出错误的
结果

1.3.1

误差的来源

误差：科学和工程计算中的数通常都是近似的。**近似值与真正值之差称为误差。**

过失误差（疏忽误差）：由于算题者在工作中的粗心大意而产生的，例如笔误将886误写成868，以及误用公式等。它完全是人为造成的，只要工作中仔细、谨慎，是完全可以避免的，本课程不予讨论。

非过失误差：在数值计算中往往无法避免。包括：**模型误差、观测误差、截断误差和舍入误差。**对这类误差应该设法降低其数值，尤其要控制住经过多次运算后误差的积累，以确保计算结果的精度。

例：近似值带来的误差和算法的选择对计算结果的精度所产生的巨大影响。

$x = \left(\frac{\sqrt{2}-1}{\sqrt{2}+1} \right)^3$	$\sqrt{2} \approx 7/5 = 1.4$	$\sqrt{2} \approx 17/12 = 1.4166\dots$
$x = (\sqrt{2}-1)^6$	$(\frac{2}{5})^6 = 0.0040960$	$(\frac{5}{12})^6 = 0.00523278$
$x = 99 - 70\sqrt{2}$	1	$-\frac{1}{6} = -0.16666667$
$x = \left(\frac{1}{\sqrt{2}+1} \right)^6$	$(\frac{5}{12})^6 = 0.00523278$	$(\frac{12}{29})^6 = 0.00501995$
$x = \frac{1}{99+70\sqrt{2}}$	$\frac{1}{197} = 0.00507614$	$\frac{12}{2378} = 0.00504626$

用数值计算的方法来解决工程实际和科学技术中的具体技术问题，首先必须将具体问题抽象为数学问题，例如各种微分方程、积分方程、代数方程……

从实际问题提炼出数学问题时，欲将复杂的物理现象抽象，归结为数学模型，总是在一定条件下抓住主要因素，忽略许多次要因素的影响，而对问题作某些必要的简化。

这样建立起来的数学模型实际上必定只是所研究的复杂客观现象的一种近似的理想化了的数学描述，与实际之间总存在误差。

因而即使数学问题能求出准确解，也与实际问题的真正解不同。

在建模和具体运算过程中所用的一些初始数据往往都是通过人们实际观察、测量得来的，由于受到所用观测仪器、设备精度的限制，这些测得的数据都只能是近似的，即存在着误差。

在不少数值运算中常遇到超越计算，如微分、积分和无穷级数求和等，它们需用极限或无穷过程来求得。

然而计算机却只能完成有限次算术运算和逻辑运算，因此需将解题过程化为一系列有限的算术运算和逻辑运算。

这样就要对某种无穷过程进行“截断”，即仅保留无穷过程的前段有限序列而舍弃它的后段。

例：

$$\begin{aligned}\sin x &= x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \cdots \\ \ln(1+x) &= x - \frac{x^2}{2} + \frac{x^3}{3} - \frac{x^4}{4} + \cdots \\ (-1 < x \leq 1)\end{aligned}$$

$$e^x = 1 + x + \frac{x^2}{2!} + \cdots + \frac{x^n}{n!} \cdots,$$

计算机只能截取有限项求出：

$$S_n(x) = 1 + x + \frac{x^2}{2!} + \cdots + \frac{x^n}{n!}$$

在数值计算过程中还会用到一些无穷小数，例如无理数和有理数中某些分数化出的无限循环小数，如

$$\pi = 3.14159265\dots; \sqrt{2} = 1.41421356\dots$$

$$\frac{1}{3!} = \frac{1}{6} = 0.166666\dots$$

计算机数系 $F(\beta, t, L, U)$ 受机器字长的限制，是离散的有限集，只能对有限的数进行运算。

这时就需要把数据按四舍五入成一定位数的近似的有理数来代替。产生舍入误差。

每一步的舍入误差是微不足道的，但在计算机上经过千百万次运算后所积累起来的总误差不容忽视，有时可能会大得惊人，甚至“淹没”真值，使计算结果失去意义。

数学模型一旦建立，进入具体计算时所要考虑和分析的就是截断误差和舍入误差。而模型误差和观测误差不是计算工作者所能独立解决的。本课程介绍算法时，将对其截断误差和舍入误差作适当的介绍。

在数值计算中，误差虽然不可避免，但人们总是希望计算结果能足够准确。—— 误差估计

为了从不同的侧面表示近似数的精确程度，通常运用绝对误差、相对误差和有效数字的概念。

绝对误差和绝对误差限

1.3.2 绝对误差、相对误差与准确数字 (续)

定义1.1 设 x^* 是准确值, x 是 x^* 的一个近似值, 称

$$e = x^* - x$$

为近似值 x 的绝对误差, 简称误差。 (注意与书本上的差异)

准确值 x^* 一般无法知道, 绝对误差 e 自然也无法知道。但一般可以根据测量工具或计算过程, 设法估计出此绝对误差的取值范围, 即误差绝对值的一个上界 ε 。

定义1.2 设存在一个正数 ε , 使

$$|e| = |x^* - x| \leq \varepsilon,$$

称 ε 是近似数 x 的绝对误差限 (绝对误差界), 简称误差限、精度

若近似数 x 的误差限为 ε ，则 $x - \varepsilon \leq x^* \leq x + \varepsilon$

表明准确值 x^* 必落在区间 $[x - \varepsilon, x + \varepsilon]$ 上， 常用

$x^* = x \pm \varepsilon$ 表示近似值 x 的精度或准确值 x^* 所在的范围。

例如，用具有毫米刻度的米尺测量不超过1m的某个物体的长度 l^* 时。

读数方法如下：如果长度 l^* 接近于毫米刻度 l ，就读出那个刻度数 l 作为长度 l^* 的近似值。

这个近似值的绝对误差不会超过半个毫米，即测量长度的精度为0.5mm。即：

$$|e| = |l^* - l| \leq 0.5mm$$

如果读出的长度是**513mm**，则

$$|l^* - 513| \leq 0.5mm$$

得： $512.5 \leq l^* < 513.5(mm)$

即 l^* 在[512.5,513.5]毫米区间内。

用绝对误差还不能完全评价近似值的精确度。

$$l_1 = (1000 \pm 1)cm \quad l_2 = (100 \pm 1)cm$$

l_1 和 l_2 的绝对误差限都是 $1cm$ ，但显然 l_1 的近似程度 l_2 比好。

要评价一个近似值的精确度，除了要看其绝对误差的大小外，还需考虑该量本身的大小。即相对误差。

定义1.3 称 $e_r = \frac{e}{x^*} = \frac{x^* - x}{x^*}$ 为近似值 x 的相对误差。

相对误差是一个无量纲量，通常用百分数表示，相对误差的绝对值越小，近似程度越高。如：

$$|e_r(l_1)| = 0.1\% \quad |e_r(l_2)| = 1\%$$

所以 l_1 的近似程度 l_2 比好。

同样，由于准确值 x^* 一般无法知道，不能定出相对误差 e_r 的准确值，而只能估计它的大小范围。

定义1.4 如果存在一个正数 ε_r ，使

$$|e_r| = \left| \frac{x^* - x}{x^*} \right| = \left| \frac{e}{x^*} \right| \leq \varepsilon_r,$$

则称正数 ε_r 为近似数 x 的相对误差限。

相对误差限不如绝对误差限容易得到，在实际计算中常借助绝对误差限来求之，并取分母中的准确值 x^* 为近似值 x ，即取

$$\varepsilon_r = \frac{\varepsilon}{|x|}$$

有效数字 (准确数字)

一种近似数的表示方法，

既能表示其大小，又能表示其精确程度

在计算中常按四舍五入原则得到数 x^* 的前几位近似值 x ，例如设

$$x^* = \pi = 3.1415926 \dots$$

经四舍五入，

若取三位得 $x = 3.14$,

若取五位得 $x = 3.1416$,

它们的绝对误差限都不超过末位的半个单位，即

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

—————→ 准确到了末位

定义1.5 设近似数 $x = \pm 0.a_1a_2 \cdots a_n \times 10^m$,

其中 $a_i \in \{0, 1, 2, 3, \cdots, 9\} (i = 1, 2, \cdots, n), a_1 \neq 0$,
 m 为整数, 如果 $|e| = |x^* - x| \leq \frac{1}{2} \times 10^{m-n}$,

则称近似值 x 有 n 位有效数字, 其中 a_1, a_2, \dots, a_n 都是 x 的有效数字, 也称 x 为有 n 位有效数字的近似值。

$x = 3.14$ 三位有效数字

$x = 3.1416$ 五位有效数字

有效数字越多, 绝对误差越小。

有效数字与相对误差的关系

定理1.1 设近似数 $x = \pm 0.a_1a_2 \cdots a_n \times 10^m$, 有 n 位有效数字
 则其相对误差限为 $\varepsilon_r = \frac{1}{2a_1} \times 10^{-n+1}$

证：由 x 有 n 位有效数字知 $|e| = |x^* - x| \leq \frac{1}{2} \times 10^{m-n}$,
 而 $|x| \geq a_1 \times 10^{m-1}$, 故有

$$|e_r| = \left| \frac{x^* - x}{x} \right| \leq \frac{\frac{1}{2} \times 10^{m-n}}{a_1 \times 10^{m-1}} = \frac{1}{2a_1} \times 10^{-n+1}$$

即相对误差限 $\varepsilon_r = \frac{1}{2a_1} \times 10^{-n+1}$

证毕。

定理1.2 设近似数 $x = \pm 0.a_1a_2 \cdots a_n \times 10^m$ 的相对误差限

$$\varepsilon_r = \frac{1}{2(a_1 + 1)} \times 10^{-n+1} \quad \text{则它至少有 } n \text{ 位有效数字。}$$

证：由于 $\varepsilon = |x| \varepsilon_r$ ，而 $|x| \leq (a_1 + 1) \times 10^{m-1}$ ，所以

$$\varepsilon \leq (a_1 + 1) \times 10^{m-1} \times \frac{1}{2(a_1 + 1)} \times 10^{-n+1} = \frac{1}{2} \times 10^{m-n}$$

因此， x 至少有 n 位有效数字。证毕。

例1.4 要使 $\sqrt{20}$ 的近似值的相对误差限小于0.1%，要取几位有效数字？

解 设取 n 位有效数字，由定理1.1知 $\varepsilon_r = \frac{1}{2a_1} \times 10^{-n+1}$

由 $\sqrt{20} = 10 \times 0.44 \dots$ 知 $a_1 = 4$ ，依题意应使 $\frac{1}{8} \times 10^{-n+1} < 0.1\%$ ，即 $10 \times 10^{-n} < 8 \times 10^{-3}$ ，

故只要取 $n=4$ ，即只要对 $\sqrt{20}$ 的近似值取4位有效数字，其相对误差限小于0.1%。 $\sqrt{20} \approx 4.472$

定理1.1 设近似数 $x = \pm 0.a_1 a_2 \dots a_n \times 10^m$ ，有 n 位有效数字

则其相对误差限为

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

1.3.3 计算机的舍入误差

设计算机的数系为 $F(\beta, t, L, U)$, 某数

$$x = \pm \beta^c \times 0.a_1 a_2 \cdots a_t \cdots$$

其中 $a_i \in \{0, 1, 2, 3, \dots, \beta - 1\} (i = 1, 2, \dots)$,

$$a_1 \neq 0, x \notin F(\beta, t, L, U), m < |x| < M$$

m 及 M 是 $F(\beta, t, L, U)$ 中的最小正数和最大正数。

计算机经舍入处理后以数 $fl(x)$ 接收, 即

$$fl(x) = \pm \beta^c \times \tilde{a}$$

$$fl(x) = \pm \beta^c \times \tilde{a}$$

$$\tilde{a} = \begin{cases} 0.a_1a_2\cdots a_t. & \text{当 } 0 \leq a_{t+1} < \beta/2, \\ 0.a_1a_2\cdots a_t + \beta^{-t}, & \text{当 } \beta/2 \leq a_{t+1}. \end{cases}$$

因此计算机对 x 的舍入绝对误差满足

$$|e| = |x - fl(x)| \leq 0.5 \times \beta^{c-t},$$

舍入相对误差满足

$$|e_r| = \frac{|x - fl(x)|}{|x|} \leq \frac{0.5 \times \beta^{c-t}}{0.1 \times \beta^c} = 0.5 \times \beta^{1-t},$$

$$|e_r| = \frac{|x - fl(x)|}{|x|} \leq \frac{0.5 \times \beta^{c-t}}{0.1 \times \beta^c} = 0.5 \times \beta^{1-t},$$

计算机对任何实数的舍入相对误差限与实数本身无关，只与计算机字长 t 有关，因此通常定义数

$$eps = 0.5 \times \beta^{1-t} \quad \longrightarrow \quad \text{计算机的精度}$$

给定一台计算机后，其舍入误差的精度也就给定了。

计算机的精度只与字长有关，计算机字长 t 越大，其精度越高。

任何数学问题的解 y 总与某些参量 $x^*_1, x^*_2, \dots, x^*_n$ 有关:

$$y^* = \varphi(x^*_1, x^*_2, \dots, x^*_n)$$

参量的值（即数据）若有误差，解也一定有误差。

设 $x^*_1, x^*_2, \dots, x^*_n$ 的近似值为 x_1, x_2, \dots, x_n ，相应的解为 y
则近似解 y 的绝对误差

$$\begin{aligned} e(y) &= y^* - y \\ &= \varphi(x^*_1, x^*_2, \dots, x^*_n) - \varphi(x_1, x_2, \dots, x_n) \end{aligned}$$

相对误差:

$$e_r(y) = e(y) / y^* = e(y) / \varphi(x^*_1, x^*_2, \dots, x^*_n)$$

当数据误差较小时，由于函数增量近似等于其微分，得误差估计式

$$dx_i \approx x_i^* - x_i = e(x_i) \quad dy \approx y^* - y = e(y)$$

$$e(y) \approx dy \approx \sum_{i=1}^n \frac{\partial \varphi(x_1, x_2, \dots, x_n)}{\partial x_i} dx_i \approx \sum_{i=1}^n \frac{\partial \varphi(x_1, x_2, \dots, x_n)}{\partial x_i} e(x_i),$$

$$e_r(y) \approx \sum_{i=1}^n \frac{\partial \varphi(x_1, x_2, \dots, x_n)}{\partial x_i} \frac{x_i}{\varphi} \frac{e(x_i)}{x_i} = \sum_{i=1}^n \frac{\partial \varphi(x_1, x_2, \dots, x_n)}{\partial x_i} \frac{x_i}{\varphi} e_r(x_i)$$

$$y = \varphi(x_1, x_2, \dots, x_n)$$

$$e(y) \approx \sum_{i=1}^n \frac{\partial \varphi(x_1, x_2, \dots, x_n)}{\partial x_i} e(x_i),$$

$$e_r(y) \approx \sum_{i=1}^n \frac{\partial \varphi(x_1, x_2, \dots, x_n)}{\partial x_i} \frac{x_i}{\varphi} e_r(x_i)$$

绝对误差系数

相对误差系数

系数绝对值: $\left| \frac{\partial \varphi}{\partial x_i} \right|, \left| \frac{x_i}{\varphi} \frac{\partial \varphi}{\partial x_i} \right|$ ——求y问题的条件数

表示解的误差相对参量 x_i 的误差的放大或缩小“倍数”。
它们的值如果很大, 则 x_i 的小误差可能导致解y很大的误差。

条件数很大的问题称为坏条件问题或病态问题。

例：用电表测得一个电阻两端的电压和流过的电流范围分别为 $V = 220 \pm 2$ (伏特) 和 $I = 10 \pm 0.1$ (安培)，求这个电阻的阻值 R ，并估计其绝对误差和相对误差。

解 由欧姆定律，有 $R = V / I$

可求出 R^* 的近似值 $R = \frac{220}{10} = 22$ (欧姆)

R 的绝对误差：

$$e(R) \approx \left(\frac{\partial R}{\partial V}\right) \times e(V) + \left(\frac{\partial R}{\partial I}\right) \times e(I) = \frac{1}{I} e(V) - \frac{V}{(I)^2} e(I)$$
$$|e(R)| \leq \left|\frac{1}{I} e(V)\right| + \left|\frac{V}{(I)^2} e(I)\right| \leq \frac{1}{10} \times 2 + \frac{220}{(10)^2} \times 0.1 = 0.42 \text{ (欧姆)}$$

R 的相对误差：

$$|e_r(R)| = \frac{e(R)}{R} \leq \frac{0.42}{22}$$

$$e(y) \approx \sum_{i=1}^n \frac{\partial \varphi(x_1, x_2, \dots, x_n)}{\partial x_i} e(x_i)$$

误差在运算中的传播

1.3.4 误差的传播(续)

1) 加、减运算

$$e\left(\sum_{i=1}^n x_i\right) \approx \sum_{i=1}^n e(x_i), e_r\left(\sum_{i=1}^n x_i\right) \approx \sum_{i=1}^n \frac{x_i}{\sum_{i=1}^n x_i} e_r(x_i)$$

近似值之和的绝对误差等于各近似值的绝对误差的代数和。

两数 x_1 和 x_2 相减时, 有

$$e_r(x_1 - x_2) \approx \frac{x_1}{x_1 - x_2} e_r(x_1) - \frac{x_2}{x_1 - x_2} e_r(x_2)$$

$$|e_r(x_1 - x_2)| \leq \left| \frac{x_1}{x_1 - x_2} \right| |e_r(x_1)| + \left| \frac{x_2}{x_1 - x_2} \right| |e_r(x_2)|$$

$$e(y) \approx \sum_{i=1}^n \frac{\partial \varphi(x_1, x_2, \dots, x_n)}{\partial x_i} e(x_i), e_r(y) \approx \sum_{i=1}^n \frac{\partial \varphi(x_1, x_2, \dots, x_n)}{\partial x_i} \frac{x_i}{\varphi} e_r(x_i)$$

2) 乘法运算

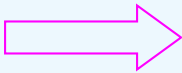
1.3.4 误差的传播(续)

$$e\left(\prod_{i=1}^n x_i\right) \approx \sum_{i=1}^n \left[\left(\prod_{\substack{j=1 \\ j \neq i}}^n x_j \right) e(x_i) \right]$$
$$e_r\left(\prod_{i=1}^n x_i\right) \approx \sum_{i=1}^n [e_r(x_i)]$$

近似值之积的相对误差等于相乘各因子的相对误差的代数和。当乘数的绝对值很大时，乘积的绝对误差会很大。

尽量避免用绝对值很大的数作乘数。

$$e(x_1 x_2) \approx x_2 e(x_1) + x_1 e(x_2)$$

若 x_1 或 x_2 很大  计算结果的绝对误差可能很大

$$e(y) \approx \sum_{i=1}^n \frac{\partial \varphi(x_1, x_2, \dots, x_n)}{\partial x_i} e(x_i), e_r(y) \approx \sum_{i=1}^n \frac{\partial \varphi(x_1, x_2, \dots, x_n)}{\partial x_i} \frac{x_i}{\varphi} e_r(x_i)$$

3) 除法运算

$$e\left(\frac{x_1}{x_2}\right) \approx \frac{e(x_1)}{x_2} - \frac{x_1}{x_2^2} e(x_2) = \frac{x_1}{x_2} [e_r(x_1) - e_r(x_2)]$$

$$e_r\left(\frac{x_1}{x_2}\right) \approx e_r(x_1) - e_r(x_2)$$

两近似值之商的相对误差等于被除数的相对误差与除数的相对误差之差。

当除数的绝对值很小，接近于零时，商的绝对误差可能会很大，甚至造成计算机的“溢出”错误。

应当尽量避免用接近于零的数作除数。

$$e(y) \approx \sum_{i=1}^n \frac{\partial \varphi(x_1, x_2, \dots, x_n)}{\partial x_i} e(x_i), e_r(y) \approx \sum_{i=1}^n \frac{\partial \varphi(x_1, x_2, \dots, x_n)}{\partial x_i} \frac{x_i}{\varphi} e_r(x_i)$$

4) 乘方及开方运算

1.3.4 误差的传播(续)

$$\begin{aligned} e(x^p) &\approx p(x)^{p-1} e(x) & e(x^{1/q}) &\approx \frac{1}{q} (x)^{\frac{1}{q}-1} e(x) \\ e_r(x^p) &\approx p e_r(x) & e_r(x^{1/q}) &\approx \frac{1}{q} e_r(x) \end{aligned}$$

乘方运算将使计算结果的相对误差增大为原值的 p 倍，降低了精度；

开方运算则使结果的相对误差缩小为原值的 $1/q$ (q 为开方次数)，精度得到提高。

大小相近的同号数相减，乘数的绝对值很大，以及除数接近于零等，在数值计算中都应设法避免。

$$e(y) \approx \sum_{i=1}^n \frac{\partial \varphi(x_1, x_2, \dots, x_n)}{\partial x_i} e(x_i), e_r(y) \approx \sum_{i=1}^n \frac{\partial \varphi(x_1, x_2, \dots, x_n)}{\partial x_i} \frac{x_i}{\varphi} e_r(x_i)$$

	$x = \left(\frac{\sqrt{2}-1}{\sqrt{2}+1}\right)^3$	$\sqrt{2} \approx 7/5 = 1.4$	$\sqrt{2} \approx 17/12 = 1.4166\dots$
1)	$x = (\sqrt{2}-1)^6$	$(\frac{2}{5})^6 = 0.0040960$	$(\frac{5}{12})^6 = 0.00523278$
2)	$x = 99 - 70\sqrt{2}$	1	$-\frac{1}{6} = -0.16666667$
3)	$x = \left(\frac{1}{\sqrt{2}+1}\right)^6$	$(\frac{5}{12})^6 = 0.00523278$	$(\frac{12}{29})^6 = 0.00501995$
4)	$x = \frac{1}{99+70\sqrt{2}}$	$\frac{1}{197} = 0.00507614$	$\frac{12}{2378} = 0.00504626$

第一、第二种算法都是相近数相减，使计算结果的有效数字位数显著减少。近似值的P次乘方的相对误差是该近似值本身的相对误差的p倍。第四种方法误差最小。

1.4 算法的稳定性

算法：不仅仅是单纯的数学公式，是对一些已知数据按某种规定的顺序进行有限次四则运算，求出所关心的未知量的整个计算步骤。

解决一个计算问题往往有多种算法，用不同算法计算的结果其精度往往大不相同。

初始数据的误差或计算中的舍入误差在计算过程中的传播，因算法不同而异。

一个算法如果输入数据有误差，而在计算过程中舍入误差不增长，则称此算法是数值稳定的，否则称此算法为不稳定的。

构造数值稳定的算法。

算法数值稳定的几个原则

(1) 要防止大数“吃掉”小数
在数值运算中：
参加运算的数有时数量级相差很大，
计算机字长有限，
如不注意运算次序

大数“吃掉”
小数的现象

影响计算结
果的可靠性

例：在4位十进制计算机上，两数相加：

$$\begin{aligned} 10^4 \times 0.1995 + 10^{-1} \times 0.4270 &= 10^4 \times 0.1995 + 10^4 \times 0.00000427 \\ &= 10^4 \times 0.19950427 = 10^4 \times 0.1995 \end{aligned}$$

小数被大数“吃掉”，小数对
和数不起作用，和数产生误差。

为减少舍入误差，实际计算时需注意运算顺序，避免大数吃掉小数。

例： $31.97 + 2.456 + 0.1352$ 真值： 34.5612

先加前两数： $(31.97 + 2.456) + 0.1352 = 34.43 + 0.1352 = 34.57$


先加后两数： $31.97 + (2.456 + 0.1352) = 31.97 + 2.591 = 34.56$

结论：若干数相加，最好先加绝对值较小的数。

例1.2中计算的 x_2 严重失真，也是由大数“吃掉”小数造成的。

$$x_{1,2} = \frac{(10^9 + 4) \pm \sqrt{(10^9 + 4)^2 - 16 \times 10^9}}{2}$$

在字长为8的计算机上计算时，由于加减运算是先对阶，后运算，

故实际计算时 10^9 “吃掉”了4， $(10^9 + 4)^2$ 又“吃掉”了 16×10^9 ， 计算结果为：

$$fl(x_1) = \frac{0.100000000 \times 10^{10} + 0.100000000 \times 10^{10}}{0.2 \times 10} = 0.100000000 \times 10^{10}$$

$$fl(x_2) = \frac{0.100000000 \times 10^{10} - 0.100000000 \times 10^{10}}{0.2 \times 10} = 0.000000000$$

为了改善 x_2 的计算效果，可以用根与系数的关系式
 $x_1 x_2 = 4 \times 10^9$ ，得

$$x_2 = \frac{4 \times 10^9}{x_1}, \text{ 这样}$$

$$fl(x_2) = \frac{0.400000000 \times 10^{10}}{0.100000000 \times 10^{10}} = 0.400000000 \times 10$$



(2) 要控制舍入误差的积累和传播

计算机计算过程中，原始数据可能有误差，每次运算又可能产生舍入误差，误差积累起来，很可能淹没真正解，使计算结果根本不可靠。

例如要在4位计算机上计算8个积分。

$$I_i = e^{-1} \int_0^1 x^i e^x dx, \quad i = 0, 1, \dots, 7 \quad \text{递推关系: } I_i = 1 - iI_{i-1}$$

$$I_0 = 1 - e^{-1} \quad I_i = \int_0^1 x^i e^{-(1-x)} dx < \int_0^1 x^i dx = \frac{1}{i+1} \rightarrow 0 \quad (i \rightarrow \infty)$$

算法一：令 $I_0 = 0.6321 \approx 1 - e^{-1}$ ，再算 $I_i = 1 - iI_{i-1}$ ， $i = 1, 2, \dots, 7$

算法二：令 $I_{11} = 0$ ，再算 $I_{i-1} = (1 - I_i)/i$ ， $i = 11, 10, \dots, 1$

算法一结果：

0.6321, 0.3679, 0.2642, 0.2074

0.1704, 0.1480, 0.1120, 0.2160

算法二 结果：

0.6321, 0.3679, 0.2642, 0.2073

0.1709, 0.1455, 0.1268, 0.1124

可靠的算法，各步误差不应对计算结果产生过大影响，即具有稳定性。

研究稳定性，应当考察每步误差的影响；为简单，通常只考虑一步（如运算开始时）误差的影响。

将算法稳定性的研究，转化为初始数据误差对算法影响的分析。

对算法一：设 $I_0 \approx I_0^*$ 有误差，此后计算无误差，则

$$\begin{cases} I_i^* = 1 - iI_{i-1}^* \\ I_i = 1 - iI_{i-1} \end{cases}, \quad i = 1, 2, \dots, 7$$

两式相减得 $I_i^* - I_i = (-i)(I_{i-1}^* - I_{i-1})$, $i = 1, 2, \dots, 7$

由此递推，得 $I_7^* - I_7 = (-7!)(I_0^* - I_0) = -5040(I_0^* - I_0)$

对算法二：设 $I_7 \approx I_7^*$ 有误差，此后计算无误差，则

$$\begin{cases} I_{i-1}^* = (1 - I_i^*) / i \\ I_{i-1} = (1 - I_i) / i \end{cases}, \quad i = 7, 6, \dots, 1$$

两式相减得 $I_{i-1}^* - I_{i-1} = -(I_i^* - I_i) / i, \quad i = 7, 6, \dots, 1$

由此递推，得 $I_0^* - I_0 = -(I_7^* - I_7) / 7! = -(I_7^* - I_7) / 5040$

例1.3 用 $I_n = -5I_{n-1} + \frac{1}{n} \quad (n = 1, 2, \dots)$ 计算

I_0 的舍入误差在计算过程中迅速传播，每次扩大5倍，致使 I_{12} 严重失真

变换计算公式： $I_{n-1} = -\frac{1}{5}I_n + \frac{1}{5n} \quad (n = m, m-1, \dots, 2, 1)$

$$I_{n-1} = -\frac{1}{5}I_n + \frac{1}{5n} \quad (n = m, m-1, \dots, 2, 1)$$

数值稳定的算法

取充分大的 m 对应的 I_m 的一个估计值作为计算初值，
在逐步算出 I_{m-1} ， I_{m-2} ， \dots ， I_1 。
可使计算的误差不增加

(3) 要避免两个相近的数相减

两个相近的数相减会引起有效数字的严重损失，导致相对误差增大。

当 $x_1 \approx x_2$ ，即大小接近的两个同号近似值相减时，

$$|e_r(x_1 - x_2)| \leq \left| \frac{x_1}{x_1 - x_2} \right| |e_r(x_1)| + \left| \frac{x_2}{x_1 - x_2} \right| |e_r(x_2)| \rightarrow \infty$$

计算结果的有效数字严重丢失，计算精度很低。

通过变换计算公式, 尽量避开相近数的相减

例: 当 $x \gg 1$

$$y = \ln(x + \frac{1}{x}) - \ln(x - \frac{1}{x}), \quad z = \arctan(x + 1) - \arctan x$$

采用计算公式: $y = \ln \frac{x^2 + 1}{x^2 - 1}, \quad z = \arctan \frac{1}{1 + x(x + 1)}$

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

二次方程: $ax^2 + bx + c = 0$ ($b^2 \gg 4ac$)

$$x_1 = (-b + \sqrt{b^2 - 4ac}) / 2a, \quad x_2 = (-b - \sqrt{b^2 - 4ac}) / 2a$$

采用计算公式:

$$x_1 = [-b - \operatorname{sgn}(b)\sqrt{b^2 - 4ac}] / 2a, \quad x_2 = c / ax_1$$

(4) 要避免绝对值很小的数作除数

绝对值很小的数作除数，也会直接影响计算结果的精度。

$$e\left(\frac{x_1}{x_2}\right) \approx \frac{e(x_1)}{x_2} - \frac{x_1}{x_2^2} e(x_2) = \frac{x_1}{x_2} [e_r(x_1) - e_r(x_2)]$$

$$\text{当 } |x_2| \rightarrow 0 \text{ 时 } \left| e\left(\frac{x_1}{x_2}\right) \right| \rightarrow \infty$$

通过变换计算公式, 尽量避开小除数

$$\text{例 } |x| \rightarrow 0 \text{ 时, 计算 } \frac{1 - \cos x}{\sin x}$$

相近数相减

小除数

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

(5) 减少运算次数, 避免误差积累

为减少舍入误差, 也为节省计算机时间, 实际计算时应当设法减少运算次数。

例计算多项式:

$$P_n(x) = a_0 + a_1x + a_2x^2 + \cdots + a_nx^n$$

可用两种方法进行:

算法一:

$$\text{令 } t_k = x^k, u_k = \sum_{i=1}^k a_i x^i, \text{ 则}$$

$$\begin{cases} t_k = xt_{k-1}, \\ u_k = u_{k-1} + a_k t_k \end{cases} \quad (k = 1, 2, \cdots, n),$$

初始值 $t_0 = 1, u_0 = a_0$, 则 $P_n(x) = u_n$

➡ 共需 $2n$ 次乘法


算法二：秦九韶算法

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

$$\text{令 } v_k = (\cdots((a_n x + a_{n-1})x + a_{n-k+1})x + a_{n-k},$$

$$\text{设初值 } v_0 = a_n, \text{ 则: } v_1 = a_n x + a_{n-1} \cdots$$

$$v_k = x v_{k-1} + a_{n-k} \quad (k = 1, 2, \cdots, n) \rightarrow P_n(x) = v_n$$

 共需n次乘法和n次加法

$$\text{多项式: } 0.0625x^4 + 0.4250x^3 + 1.215x^2 + 1.192x + 2.129$$

直接计算需: **4+3+2+1=10**次乘法和**4**次加法

$$\text{秦九韶算法: } (((0.0625x + 0.4250)x + 1.215)x + 1.192)x + 2.129$$

只需: **4**次乘法和**4**次加法

除了以上若干原则外，在编程时也应该注意误差的传播，以免出错。例如：

1) **If $A=0$ Then Goto B Else Goto C;**

2) **If $|A|\leq 10^{-12}$ Then Goto B Else Goto C;**

如果单元A中的结果是由前面运算结果得到的，按准确的结果应有 $A=0$ ，

而由于误差的影响，实际上 $A\neq 0$ 但 $|A|$ 很小，
因此按语句1) 原来要执行B，但去选择了C；
而按语句2) 就不会出问题。

故在编制程序时，切忌用“等号”作为条件语句中的条件，以免误差的影响使程序失真。

减少计算误差的措施

减少运算次数，避免相近数相减，避免小除数、大乘数和“溢出”，注意运算顺序（如若干数相加时先加绝对值较小的数，避免大数吃小数），防止误差影响扩大（采用稳定算法）。

作业:习题一(p26)第1题,第2题,第9题

小结

本章阐明了误差理论的基本概念，误差在近似值运算中的传播规律及其估算方法，以及数值稳定性的概念。

按照误差产生的来源不同，有“过失误差”和“非过失误差”之分。后者又可分为“模型误差”、“观测误差”、“截断误差”和“舍入误差”等。

误差的表示法有两种：绝对误差和相对误差。

计算机中进行的都是有限位数的运算，因此有效数字的概念是重要的，它与误差之间有着密切的关联。



THANK YOU

