Numerical Computing Methods
数值计算方法

李晓鹏
Li Xiao-Peng

电子与信息工程学院
School of Electronic and Information Engineering

Semester–I 24/25

引论

1. 研究对象与特点
2. 数值方法的基本内容
3. 数值算法及其设计
4. 误差分析

引论

1. 研究对象与特点
2. 数值方法的基本内容
3. 数值算法及其设计
4. 误差分析

研究对象与特点

什么是计算机数值方法？

数值计算方法（或数值分析）主要是研究如何运用计算机去获得数学问题的数值解的理论和方法。

研究对象与特点

什么是计算机数值方法?

数值计算方法（或数值分析）主要是研究如何运用计算机去获得数学问题的数值解的理论和方法。

牛顿-拉夫逊（Newton-Raphson）迭代法计算开平方:

```
function y = mysqrt(x)
    guess = x;
    threshold = 1e-9;
    while abs(guess * guess - x) > threshold
        guess = 0.5 * (guess + x / guess);
    end
    y = guess;
end
```

研究对象与特点

现代科学研究的三个重要组成部分

1. 理论计算
2. 科学实验
3. 科学计算

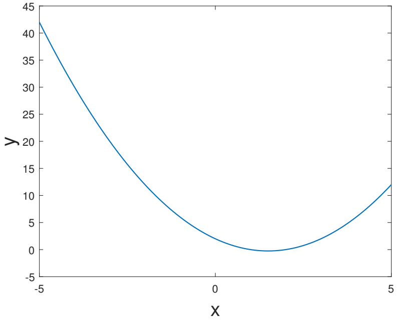
重要性

1. 仅靠数学理论的演绎和推导不能解决实际中的数值问题，只有与计算机科学相结合，才能研制出实用的好算法
2. 好的算法变成数值软件以后才可能为社会创造更大的财富

研究对象与特点

利用数学和计算机知识解决实际问题可分为两步

1. **建立数学模型：** 需要利用有关专业知识和数学理论，这属于应用数学范围
2. **提出数值问题与数值方法：** 将数学模型变成数值问题，进而研究该数值问题的数值方法，并设计有效的数值算法的过程，这属于计算方法的范围



研究对象与特点

现代计算方法的一个显著特点

已产生大量实用的“综合数学软件库”，并逐步形成了数值软件产业,如:

- Mathematica: 综合数学软件包，集符合演算、数值计算和图形演示等
- Maple: 系统内置高级技术解决建模和仿真中的数学问题，符号计算、无限精度数值计算等
- MATLAB: 集数值计算、图形演示等一体的综合数值软件库

Mathematica

```
(* Define variables *)
a = 1;
b = -3;
c = 2;

(* Solve the equation using the solve function *)
solutions = Solve[a x^2 + b x + c == 0, x];
Print["The solutions are: ", solutions];
```

Maple

```
# Define variables
a := 1;
b := -3;
c := 2;

# Defining quadratic equations
equation := a*x^2 + b*x + c = 0;

# Solve the equation using the solve function
solutions := solve(equation, x);

print("The solutions are:", solutions);
```

MATLAB

```matlab
% Define variables
a = 1;
b = 0;
c = 0;

% solution calculation of quadratic equations
solutions = roots([a, b, c]);

disp(['The solutions are: ', num2str(solutions')]);
```

研究对象与特点

本课程包含的主要内容:

1. 数值方法
2. 算法及其设计
3. 误差分析

"数值方法"（Numerical Methods）和"数值算法"（Numerical Algorithm）两个术语在实际应用中常常互相替换使用，但从严格的定义和应用角度来看，它们之间还是存在一些微妙的差异。

研究对象与特点

数值方法（Numerical Method）

数值方法通常是指用于解决数学问题的一套完整的数学框架或理论，这些数学问题通常无法通过精确的、封闭形式的解来解决。例如，微分方程、线性方程组、积分等问题在某些情况下难以得到精确解，因此需要使用数值方法来近似求解。

数值方法往往包括了对问题的建模、理解其数学性质（如稳定性、精度等）、选择适当的近似手段等。

数值算法（Numerical Algorithm）

数值算法则更侧重于实现的细节，它是一种具体的、可以通过计算机编程来实现的操作步骤，用于执行某一数值方法。数值算法考虑了如何高效、准确地进行数值计算，包括迭代次数、计算复杂性、误差累积等因素。

数值算法可以看作是数值方法的一种实现方式，但一个数值方法可以有多种不同的数值算法。

例题

以求解线性方程组 $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ 为例:

- **数值方法:** 可能会涉及使用高斯消元法、矩阵分解（如 LU 分解、QR 分解等）或迭代法（如雅可比迭代、Gauss-Seidel 迭代等）。
- **数值算法:** 对于高斯消元法，具体的算法可能会涉及到如何进行行交换以避免除以零，或者如何优化存储以减少计算量。

简而言之，数值方法通常更注重理论和数学框架，而数值算法则更侧重于实现和计算效率。两者是密切相关的，但侧重点不同。在解决实际问题时，通常需要综合考虑数值方法和数值算法。

数值方法的基本内容

利用数学和计算机知识解决实际问题可分为两步:

1. **建立数学模型：** 需要利用有关专业知识和数学理论，这属于应用数学范围
2. **提出数值问题与数值方法：** 将数学模型变成数值问题，进而研究该数值问题的数值方法，并设计有效的数值算法的过程，这属于计算方法的范围

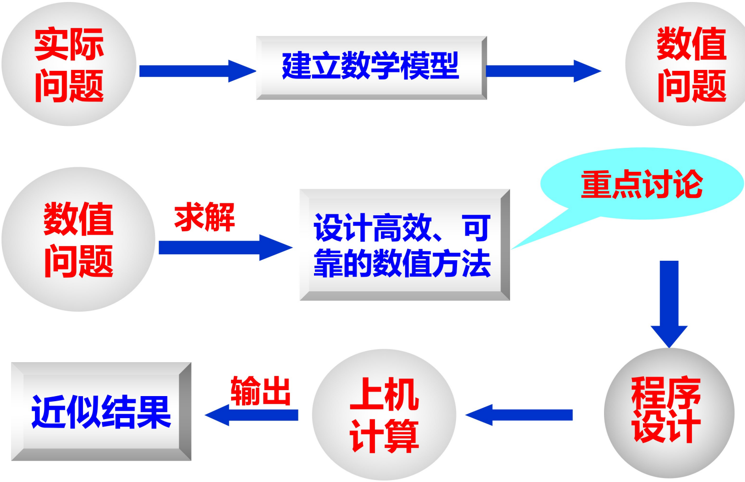
数值问题:

有限个输入数据（问题的自变量、原始数据）与有限个输出数据（待求解数据）之间函数关系的一个明确无歧义的描述。

数值问题的分类:

1. 分析问题（连续问题）
2. 代数问题（离散问题）
3. 概率与统计问题（随机问题）

数值方法的基本内容



什么“数学模型”是“数值问题”

1. 求方程$ax^2+bx+c=0$的解。
   输入a,b,c，则输出数据是根x_1和x_2，故是“数值问题”
2. 求常微分方程

$$\begin{cases} y' = 2x+3, \ x \in [0,a] \\ y(0) = 0 \end{cases}$$

因输入数据为 2 和 3，$x=0$和$y=0$等，输出是函数解析表达式 $y=x^2+3x$, 所以，不是“数值问题”。

数值方法的基本内容

如何将数学模型变成“数值问题”

将非“数值问题”转化为“数值问题”的方法：离散化。

例如：将求解$y = x^2 + 3x$的问题变成求

$$
\begin{gathered}
y(x_1), y(x_2), y(x_3), \ldots, y(x_n) \\
0 < x_1 < x_2 < x_3 < \cdots < x_n = a
\end{gathered}
$$

的问题。即将连续的情况变成在某些点上的值。

Remark： 求函数$y = x^2 + 3x$ 在某些点的近似函数值是数值问题

数值方法的基本内容

数值方法:

将求解“数值问题”的一系列计算公式称为数值方法。也称为计算方法。

计算机数值方法 (数值方法):

指它的一系列计算公式中的运算和数据，必须是可以在计算机上执行。

计算机上可执行的运算：四则运算和逻辑运算（与、或、非等）。

Remark： 计算公式不一定都属于计算机数值方法

例题

下面哪些是数值方法？

$$y = 54 \times 2.5 + 100 \tag{1}$$

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

$$y = \sin(60^\circ) \tag{3}$$

$$y = \log(5) \tag{4}$$

例题

下面哪些是数值方法？

$$y = 54 \times 2.5 + 100 \quad \checkmark \tag{1}$$

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

$$y = \sin(60^\circ) \quad \times \tag{3}$$

$$y = \log(5) \quad \times \tag{4}$$

数值方法的基本内容

计算机上不能直接执行的运算:

开方、超越函数、极限、微分、积分等
超越函数: 三角函数、对数函数，反三角函数，指数函数

几个转化例子

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

$$y' \approx \frac{y(x+h) - y(x)}{h}$$

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

本课程重点是介绍常见的行之有效的计算机数值方法。

1. 研究对象与特点
2. 数值方法的基本内容
3. **数值算法及其设计**
4. 误差分析

数值算法及其设计

数值算法:

有步骤地完成求解数值问题的过程

数值算法特性:

1. **目的性：** 算法目的明确，条件和结论均应明确；
2. **确定性：** 算法必须精确地给出每一步的操作；
3. **可执行性：** 算法中的每个操作都是可执行的；
4. **有穷性：** 算法必须在有限步内结束解题过程。
5. **通用性：** 算法是针对某一类问题的计算，而不是只适用于解决某个具体例题的计算；

例题

牛顿-拉夫逊（Newton-Raphson）迭代法计算开平方：

```
function y = mysqrt(x)
    guess = x;
    threshold = 1e-9;
    while abs(guess * guess - x) > threshold
        guess = 0.5 * (guess + x / guess);
    end
    y = guess;
end
```

目的性？确定性？可执行性？有穷性？通用性？

例题

例：等差数列1,2,3,···,10000的求和算法:

(1) 取 $N = 0$, $S = 0$;

(2) $N + 1 \Rightarrow N$, $S + N \Rightarrow S$;

(3) 若 $N < 10000$ 转（2），否则，转（4）；

(4) 输出 N 和 S

目的性？确定性？可执行性？有穷性？通用性？

计算机上的算法分类:

- 按求解问题的不同,可分为：
  1. 数值算法: 用于求解数值问题的算法
  2. 非数值算法：用于求解非数值问题（公式推导等）的算法
- 按面向计算机的不同，可分为：
  1. 面向串行计算机的串行算法，只有一个进程；
  2. 面向并行计算机的并行算法，含两个以上的进程
- 根据算法内部的特点,可分为：
  1. 确定性算法，每完成一步确切知道下一步该做什么
  2. 非确定性算法（智能算法）

  Remark: 非确定性算法是一种可能产生不同输出（即使输入相同）的算法，或者其内部操作的顺序可能会随机变化。这些算法通常依赖于概率或随机数生成。

例题

MATLAB program

```
syms x
f = sin(x);
df = diff(f, x)
```

数值算法？非数值算法？

例题

MATLAB program

```
% Initialize the vector
original_array = [1, 2, 3, 4, 5];
% Create an empty vector to store the results
squared_array = zeros(1, length(original_array));

% Use a for-loop for serial computing
for i = 1:length(original_array)
    squared_array(i) = original_array(i)^2;
end

disp('Result from serial computing:');
disp(squared_array);
```

串行算法？并行算法？

例题

MATLAB program

```
% Initialize the vector
original_array = [1, 2, 3, 4, 5];
% Create an empty vector to store the results
squared_array = zeros(1, length(original_array));

% Use parfor for parallel computing
parfor i = 1:length(original_array)
    squared_array(i) = original_array(i)^2;
end

disp('Result from parallel computing:');
disp(squared_array);
```

串行算法？并行算法？

例题

MATLAB program

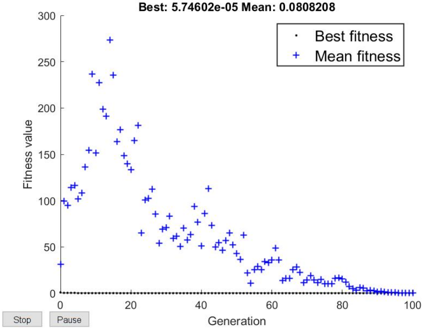
```
% This function takes a solution vector x
% and returns a scalar value
fitnessFunction = @(x) x(1)^2;

% Set algorithm parameters
% In this case 1, as we have only one variable x
nVars = 1;
% Set options to plot the best fitness function
options = gaoptimset('PlotFcns', @gaplotbestf);

% Execute the genetic algorithm
[x, fval] = ga(fitnessFunction, nVars, ...
    [], [], [], [], [], [], [], options);

% Display the result
fprintf('The optimal solution is x = %.4f with a function value of f(x) = 
```

例题



Best: 5.74602e-05 Mean: 0.0808208
Best fitness
Mean fitness
Fitness value
Generation
0
50
100
150
200
250
300
0
20
40
60
80
100
Stop
Pause

数值算法及其设计

不同算法及其计算复杂性有很大差异

例如: (对于大型数值问题)：求解20阶线性方程组

- 利用Gramer法则: 需要乘、除法运算次数 9.7×10^{20}
- 利用Gauss消去法求解，只需2670次乘、除法

例如: 排序算法的计算复杂度

- 冒泡排序：$\mathcal{O}(n^2)$
- 快速排序：$\mathcal{O}(nlogn)$

数值算法及其设计

算法设计的主要目的:

1. 研制可靠性好的数值方法（精度要求）
2. 选择计算复杂性好的数值方法（速度快、存贮少）
3. 方便编码和软件维护

目前流行的软件开发方法:

1. 面向过程的“自顶向下、逐步细化”的结构化方法;
2. 面向对象的“自下而上”的组装开发方法，其主要工具是“类”（特殊模块），利用它可组装数值算法和求解程序。

Remark: 本课程只介绍前者

“自顶向下，逐步细化”法，其关键有三个方面:

- 划分模块，主要原则是模块功能要单一，即独立性好
- 设计或选择模块算法，先总体，后具体
- 充实细节，考虑计算公式的效率和其他要求

例题

例：求解二次方程为例说明算法设计

$$ax^2 + bx + c = 0$$

解上述方程需要考虑三个细节：

1. 判别式：$d = b^2 - 4ab$ 大于零或小于零；
2. 当$d > 0$ 且$\sqrt{d} \approx b$时，会出现两个近似数相减而影响有效数字的位数；
3. 若$|a|$较$|b|$和$|c|$相对小很多时，可能出现舍入误差增大的问题。

数值算法及其设计

在数值软件中，算法常用的表达方法

- 自然语言法；用文字有步骤的表示算法
- 图示法：又分为“流程图”和“结构化框图”

自然语言法:

$$ax^2 + bx + c = 0$$

(1) 输入数据 a, b, c

(2) 如果 $a = 0$, 转（3），否则转（4）

(3) 如果 $b \neq 0$, 则 $x_1 = -c/b$，转（7）；否则，无解停止

(4) 设 $D = b^2 - 4ac$, $SD = \mathrm{SQRT}(|D|)$
如果 $D = 0$, $x_1 = x_2 = -b/2a$，转(7)
如果 $D < 0$, $x_1 = (-b + iSD)/2a$, $x_2 = (-b - iSD)/2a$, 转(7)

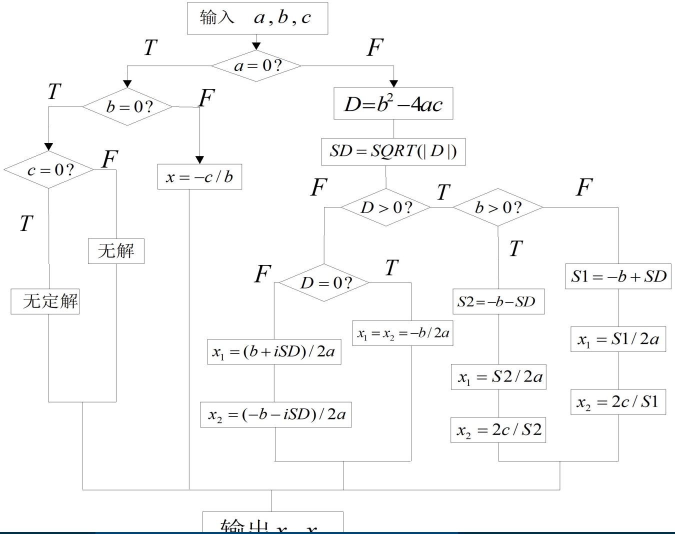
(5) 如果 $b \leq 0$，$S_1 = -b + SD$, $x_1 = S_1/2a$, $x_2 = 2c/S_1$, 转(7)

(6) $S_2 = -b - SD$, $x_1 = S_2/2a$, $x_2 = 2c/S_2$

(7) 输出 x_1 和 x_2

Remark: $x_2 = \frac{2c}{S_1} = \frac{2c}{-b+SD} = \frac{2c(-b-SD)}{(-b+SD)(-b-SD)} = \frac{2c(-b-SD)}{4ac} = \frac{(-b-SD)}{2a}$

图示法-流程图法



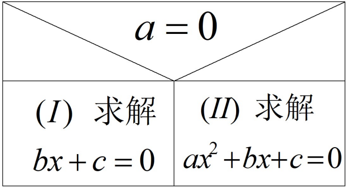
输入 a, b, c
a = 0?
T
F
b = 0?
T
F
D=b²−4ac
SD = SQRT(|D|)
c = 0?
F
x = −c / b
T
无解
无定解
F
D > 0?
T
b > 0?
F
T
F
D = 0?
T
S1 = −b + SD
S2 = −b − SD
x₁ = x₂ = −b / 2a
x₁ = (b + iSD) / 2a
x₁ = S1 / 2a
x₁ = S2 / 2a
x₂ = (−b − iSD) / 2a
x₂ = 2c / S1
x₂ = 2c / S2
输出

图示法-结构化框图法

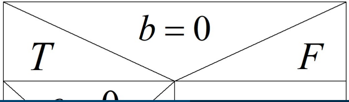
1. 顶层设计：

(I) 输入a,b,c
(II) 求解$ax^2+bx+c=0$
(III) 输出根x_1,x_2

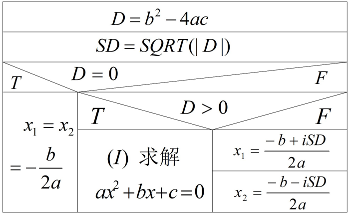
2. 第1层设计：细化 (II)



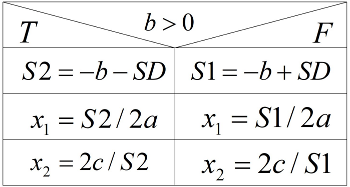
3. 第2层设计： a) 细化 (I)



3．第2层设计： b) 细化 (II)



4. 第3层设计：细化



引论

数值计算中的误差来源

1. **模型误差:** 从实际问题中抽象出数学模型
2. **观测误差:** 通过观测得到模型中某些参数（或物理量）的值
3. **舍入误差:** 由于计算机的字长有限，原始数据在计算机中的表示、运算产生的误差
4. **截断误差:** 由数学问题化成数值问题产生的误差

例题

例: 若计算机仅能表示 6 位十进制数，则将表示为

$$\pi^* = 3.14159$$

$$R_1 = \pi - \pi^* = 0.0000026...$$

若将其与数9.21000进行加法运算，得

$$s = 3.14159 + 9.21000 \approx 1.23516 \times 10$$

$$R_2 = 12.3516 - 12.35159 = 0.00001$$

R_1 和 R_2 都是舍入误差。

例题

计算e^x的数值时

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

由算法的有限性，故利用截断部分和

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

来近似代替，由此产生的误差即为截断误差，估计为

$$R_n(x) = e^x - p(x) = \frac{e^{\zeta}}{(n+1)!} x^{n+1}$$

例题

MATLAB 实验：取 $x = 2, n = 10$

```
clear
close all

x = 2;
S = 1;
for n = 1:10
    S =  S + 1/ factorial(n)*x^n;
end
R1 = exp - S;
```

运行结果： 7.3890，而$e^2 = 7.3891$

若取 $n = 5$, 运行结果： 7.2667

误差的量度

误差的量度

1. 绝对误差
2. 相对误差
3. 有效数字

绝对误差

绝对误差定义：

设 x 为准确值，x^* 为 x 的一个近似值，称

$$E(x^*) = x^* - x$$

为近似值 x^* 的（绝对）误差，简记为E.

此外，圆周率的近似值 $\pi^* = 3.15$, 绝对误差 $E = 0.0084\ldots$

一般来说，E 的准确值很难求出，只能估计出 $|E|$ 的某个上界 $\epsilon(x^*)$，即

$$|E(x^*)| = |x^* - x| \leq \epsilon(x^*)$$

$\epsilon(x^*)$ 称为近似值 x^* 的（绝对）误差限，简记为 ϵ

$$x^* - \epsilon \leq x \leq x^* + \epsilon(x^*)$$

这个不等式有时可以表示为

$$x = x^* \pm \epsilon(x^*)$$

相对误差

相对误差定义：

近似值 x^* 的误差 E 与准确值 x 之比

$$E_r(x^*) = \frac{E(x^*)}{x} = \frac{x^* - x}{x}$$

为近似值 x^* 的**相对误差**，简记为E_r.

E_r绝对值的任一上界$\epsilon_r(x^*)$，称为**相对误差限**，简记为 ϵ_r.

$$|E_r(x^*)| = \left|\frac{E(x^*)}{x}\right| = \left|\frac{x^* - x}{x}\right| \le \epsilon_r(x^*)$$

由于x的准确值难以确定，通常利用

$$E_r^* = E_r^*(x^*) = \frac{E_r(x^*)}{x^*}$$

Remark：绝对误差限与相对误差限不唯一；它们越小越好。

例题

绝对误差：必须为正？

相对误差：必须为正？

例题

绝对误差：误差可正可负，常常是无限位的

相对误差：相对误差也可正可负

有效数字

对于准确值 x 取近似值最常用的方法是采用“四舍五入”的原则。由此产生了一个专有名词–有效数字。

有效数字定义:

如果近似值 x^* 的误差的绝对值不超过某一位数字的半个单位，且该位数字到的第一位非零数字共有 n 位，则称用 x^* 近似 x 时具有 n 位有效数字。

如果使

$$|e^*| = |x^* - x| \leq \frac{1}{2} \times 10^{1-n}$$

成立的最大整数为 n, 则称 x^* 为 x 的具有 n 位有效数字

例题

$\pi_1^* = 3.1416$, $\pi_2^* = 3.14$ 和 $\pi_3^* = 3.1415$ 分别有几位有效数字

例题

$$|3.1416-\pi| = 0.0000074\cdots \le \frac{1}{2}\times 10^{-4} = \frac{1}{2}\times 10^{1-5}$$

$$|3.14-\pi| = 0.0015\cdots \le \frac{1}{2}\times 10^{-2} = \frac{1}{2}\times 10^{1-3}$$

$$|3.1415-\pi| = 0.0000926\cdots \le \frac{1}{2}\times 10^{-3} = \frac{1}{2}\times 10^{1-4}$$

例题

小数点后位数越多，有效数字位数越多？

例题

小数点后位数越多，有效数字位数越多？

$$|3.142-\pi| = 0.0004\cdots \le \frac{1}{2}\times 10^{-3} = \frac{1}{2}\times 10^{1-4}$$

$$|3.1415-\pi| = 0.0000926\cdots \le \frac{1}{2}\times 10^{-3} = \frac{1}{2}\times 10^{1-4}$$

有效数字

在计算机中，x 往往表示为如下标准形式

$$x = \pm 10^m \times 0.x_1 x_2 \cdots x_n x_{n+1} \cdots x_p,$$

其中，m 为整数，$\{x_i\} \in 0, 1, 2, \ldots, 9$ 且 $x_1 \neq 0$, $p \geq n$.

如果

$$x^* = \pm 10^m \times 0.x_1 x_2 \cdots x_n$$

是对 x 的第 $n+1$ 位数字进行四舍五入后得到的近似值，则 x^* 具有 n 位有效数字，且误差的绝对值不超过

$$\frac{1}{2} \times 10^{m-n}$$

例题

$$\pi_1^* = 3.1416 = 10^1 \times 0.31416$$

$$\pi_2^* = 3.14 = 10^1 \times 0.314$$

$$\pi_3^* = 3.1415 = 10^1 \times 0.31415$$

$$|3.1416 - \pi| = 0.0000074\cdots \leq \frac{1}{2} \times 10^{-4} = \frac{1}{2} \times 10^{1-5}$$
$$|3.14 - \pi| = 0.0015\cdots \leq \frac{1}{2} \times 10^{-2} = \frac{1}{2} \times 10^{1-3}$$
$$|3.1415 - \pi| = 0.0000926\cdots \leq \frac{1}{2} \times 10^{-3} = \frac{1}{2} \times 10^{1-4}$$

有效数字

Remark: 有效数字位数与小数点的位置无关，一个数精确到小数点后几位，不能反应它的有效位数的多少，只有它是经四舍五入得到的数且写成如下的规格化形式后，小数点后的位数才能反应出其有效位数的多少。

$$x^* = \pm 10^m \times 0.x_1 x_2 \cdots x_n x_{n+1} \cdots x_p$$

Remark: 从实验仪器所读的近似数（最后一为是估计位）不是有效数，估计最后一位是为了确保对最后一位进行四舍五入得到有效数。

例：从最小刻度为厘米的标尺读得的数据

123.4cm 是为了得到有效数 123cm,
156.7cm 是为了得到有效数 157cm

近似值的有效数字与相对误差之间的关系

设 x^* 是 x 的近似值，它的表达式为

$$x^* = \pm 10^m \times 0.x_1 x_2 \cdots x_n$$

则 x^* 的有效位数和 x^* 的相对误差之间的关系如下：

1. 若 x^* 具有 n 位有效数字，则 x^* 的相对误差 E_r^* 满足

$$|E_r^*| \leq \frac{1}{2} \times 10^{-n+1}$$

2. 若 x^* 的相对误差 E_r^* 满足

$$|E_r^*| \leq \frac{1}{2} \times 10^{-n}$$

   则 x^* 至少具有 n 位有效数字。（证明P19）

Remark: 近似值的有效位数越多，即 n 越大，相对误差（或限）就越小。反之，相对误差（或限）就越小，n 就可能越大，有效数字位数就可能越多。

例题

例：设 $x=1.986$，$x^*=1.98$，则

$$|1.986-1.98|=0.006\leq\frac{1}{2}\times10^{-1}$$

x^* 有 2 位有效数字。

$$|E_r^*(x^*)|=\left|\frac{x^*-x}{x^*}\right|=\frac{|1.986-1.98|}{1.98}\leq\frac{1}{2}\times10^{-2}<\frac{1}{2}\times10^{-2+1}$$

另一方面：

$$|E_r^*(x^*)|\leq\frac{1}{2}\times10^{-2}$$

x^*至少有 2 位有效数字。

Remark： 相对误差的上界 $\frac{1}{2}\times10^{-2+1}$ 并不是最小的。

数值方法的稳定性与算法设计原则

1. 设计和选择算法首要关心的问题是精度要求，要建立一些定性分析准则用于判断结果的可靠性，这是数值稳定性问题。
2. 对于一个数值方法，若对原始数据或某一步有舍入误差，在执行过程中，这些误差能得到控制，则称该数值方法稳定。否则，称为不稳定的。
3. 利用两种数值方法 A 和 B 解输入和舍入误差规则相同的同一问题，若用 A 发比 B 法得到的计算解精度更高，则称 A 法比 B 法具有较大的稳定性。

例题

例：计算积分 $I_n = \frac{1}{e}\int_0^1 x^n e^x dx,\ n = 0, 1, 2, \cdots, 15$

解：由于

$$e = \int_0^1 (x^n e^x)' dx = \int_0^1 x^n e^x dx + n\int_0^1 x^{n-1} e^x dx$$

所以

$$I_n = 1 - nI_{n-1} \tag{1}$$

$$I_0?$$

例题

方法一： 使用公式 (1)先计算 I_0, 然后依次计算 $I_1, I_2, \cdots I_{15}$

$$
\begin{aligned}
I_0 &= \frac{1}{e}\int_0^1 e^x dx = 1 - \frac{1}{e} \approx 0.63212056 \\
I_1 &= 1 - I_0 = 0.36787944 \\
&\vdots \\
I_{10} &= 1 - 10I_9 = 0.08812800 \\
I_{11} &= 1 - 11I_{10} = 0.03059200 \\
I_{12} &= 1 - 12I_{11} = 0.63289600 \ ? \\
I_{13} &= 1 - 13I_{12} = -7.2276480 \ ? \ ? \\
I_{14} &= 1 - 14I_{13} = 94.959424 \ ? \ ? \ ? \\
I_{15} &= 1 - 15I_{14} = -1423.3914 \ ? \ ? \ ? \ ?
\end{aligned}
$$

Remark: 初始误差 $|E_0| < 0.5 \times 10^{-8}$, 然而 $|E_n| = n!|E_0|$

例题

方法二： 使用公式 $I_{n-1} = \frac{1-I_n}{n}$, 先计算 I_{15}, 然后依次计算 $I_{14}, \cdots, I_1, I_0$

$$
\begin{aligned}
I_{15} &= \frac{1}{2}\left(\frac{1}{16e} - \frac{1}{16}\right) \approx 0.042746233 \\
I_{14} &= \frac{1}{15}(1 - I_{15}) = 0.063816918 \\
I_{13} &= \frac{1}{14}(1 - I_{14}) = 0.066870220 \\
&\vdots \\
I_1 &= \frac{1}{2}(1 - I_2) = 0.036787944 \\
I_0 &= \frac{1}{1}(1 - I_1) = 0.63212056
\end{aligned}
$$

例题

因此，方法一为不稳定算法，而方法二为稳定算法。

在我们今后的讨论中，误差将不可回避，算法的稳定性将会是一个非常重要的话题。

蝴蝶效应

蝴蝶效应是气象学家洛伦兹1963年提出来的。其大意为：一只南美洲亚马孙河流域热带雨林中的蝴蝶，偶尔扇动几下翅膀，可能在两周后引起美国德克萨斯一场龙卷风。

蝴蝶效应在社会学界用来说明：一个坏的微小的机制，如果不加以及时地引导、调节，会给社会带来非常大的危害，戏称为“龙卷风”或“风暴”

例题

例: 求线性方程组

$$\begin{cases} x_1 + \frac{1}{2}x_2 + \frac{1}{3}x_3 = \frac{11}{6} \\ \frac{1}{2}x_1 + \frac{1}{3}x_2 + \frac{1}{4}x_3 = \frac{13}{12} \\ \frac{1}{3}x_1 + \frac{1}{4}x_2 + \frac{1}{5}x_3 = \frac{47}{60} \end{cases}$$

其精确解为

$$x_1 = x_2 = x_3 = 1$$

但是，若对数据取3位有效数字，利用 Gauss 消去法求解，则计算解：

$$x_1 \approx 1.09, x_2 \approx 0.0488, x_3 \approx 0.491$$

例题

同样对方程

$$\begin{cases} 2x_1 - x_2 + 3x_3 = 1 \\ 4x_1 + 2 + 5x_3 = 4 \\ 2x_1 + 2x_3 = 6 \end{cases}$$

利用 Gauss 消去法，取 3 位有效数字，则得到准确解：

$$x_1 = 9, x_2 = -1, x_3 - 6$$

1. “数值问题”计算解的精度，不但与数值方法的稳定有关，而且还与数值问题的性态好坏有关。
2. 在数值问题中，若输出数据对输入数据的扰动（误差）很敏感（小的变化会引起较大的变化），称这类数值问题为病态问题；否则称为良态问题。

选用计算公式和设计算法时的普遍原则

一. 四则运算中的稳定性问题

1. 防止大数吃小数

$$0.3684676 + 10^7 \times 0.6327544 + 0.4932001 + 0.4800100 = 10^7 \times 0.6327544$$

预防方法：先加小数，由小到大逐次相加

$$0.3684676 + 0.4800100 + 0.4932001 + 10^7 \times 0.6327544 = 10^7 \times 0.6327545$$

2. 要避免两个相近数相减
   相近数相减会严重损失有效数值的位数 $1 - \cos(x)$ 可改为 $2\sin^2(x/2)$
3. 避免小数作除数和大数作乘法
   这样可避免误差放大

二. 提高算法效率问题

1. 尽量减少运算次数：

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

254次乘法 $\Rightarrow$ 14次乘法

例题

例：计算 $p_n(x) = a_n x^n + a_{n-1}x^{n-1} + a_{n-2}x^{n-2} + \cdots + +a_1 x + a_0$

直接计算需要 $n(n+1)/2$ 次乘法和 n 次加法

若将公式变成递推公式：

$$\begin{cases} s_n = a_n \\ s_k = x s_{k+1} + a_k, k = n-1, n-2, \cdots, 1, 0 \\ p_n(x) = s_0 \end{cases}$$

后，再计算 $p_n(x)$，只需做 n 次乘法和 n 次加法。
充分利用递推公式，可提高算法效率。

选用计算公式和设计算法时的普遍原则

2. 充分利用耗时少的运算
   例： $k+k$ 比 $2k$，$a*a$ 比 a^2，$b*0.25$ 比 $b/4$ 等节省时间。
3. 充分利用存贮空间
   - 节省原始数据的存贮单元；
   - 节省工作单元。

   例： S $=$ SPARSE(X) converts a sparse or full matrix to sparse form by squeezing out any zero elements.

选用计算公式和设计算法时的普遍原则