计算方法及MATLAB实现

郑勋烨　编著

主审：　高世臣　褚宝增　王祖朝　王翠香

副审：　李少琪　李明霞　赵琳琳　赵俊芳

国防工业出版社

一尺之棰，日取其半，万世不竭。

第一章

误差分析

失之毫厘
谬以千里

第 1 节 引言：数值分析和算法

大家好，这学期由我为大家带数值分析课程，英文 Numerical Analysis 。

数值分析 *也称* *计算方法* *或* *科学与工程计算* ，是研究科学与工程计算的数值方法的设计 (Design) 、分析 (Analysis) 与计算机实现 (Realization) 的数学学科．简言之即研究算法的学科。正如话语是思想的表达，我们所熟悉的各种高级计算机语言，从 BASIC 、 Fortran 、 C 、 C++ 到 Java ，本质上都是可被计算机认识的算法的表达。

- 1.1. **算法** Algorithm

定义 1.1 **算法** Algorithm 科学问题的适合于计算机程序演算的构造性的数值方法称为算法。

我们为大家举一个问题。

【例 1 】 **二次方程求根问题** 首一实系数二次方程

$$x^2 + 2bx + c = 0 \tag{1.1}$$

至多有两个不同的实根.

【证明】我们用 3 种不同方法解决。

(1) 反证法：假设存在 3 个互异的实根 x_1, x_2, x_3, 则有

$$\begin{cases} x_1^2 + 2bx_1 + c = 0 \\ x_2^2 + 2bx_2 + c = 0 \\ x_3^2 + 2bx_3 + c = 0 \end{cases} \tag{1.2}$$

从而

$$\begin{cases} x_1^2 - x_2^2 + 2b(x_1 - x_2) = 0 \\ x_3^2 - x_2^2 + 2b(x_3 - x_2) = 0 \end{cases} \tag{1.3}$$

于是有 $x_1 = x_3$. 矛盾。故至多有两个不同的实根.

(2) 图解法：二次方程的图象是抛物线，与 x 轴至多有两个交点，故至多有两个不同的实根.

(3) 算式法：二次方程有求根公式：

$$x = -b \pm \sqrt{b^2 - c} \tag{1.4}$$

考察判别式 $\Delta = 4(b^2 - c)$, 当 $\Delta > 0$, 存在两个互异实根；$\Delta = 0$, 存在两个相等实根； $\Delta < 0$, 无实根.

比较以上解法，方法 (1) 是非构造性的，我们是依据问题结论的否命题做的归谬假设；方法 (2) 是构造性的，我们作出了求根二次函数的图象，但不是数值的：没有根的公式解的表达，无法进行根的计算；方法 (3) 是构造性的而且是数值的方法，我们构造出了公式解且可以通过计算流程进行根的类型判别和计算，因而是一种算法。

- 1.1. **算法** Algorithm
- 1.2. **算法的特点**

【 1 】 可识别　作为面向计算机的算法，必须可被机器有效识别，即转化为程序时只能包括四则运算和逻辑运算；

【 2 】 可分析　算法应当有可靠的理论分析，保证算法的精度、收敛性和数值稳定性，而且能对误差做出估计。

【 3 】 可操作　算法应当有良好的计算复杂性，时间好即速度快以节省运算时间，空间好即节省内存。

【 4 】 可验证　算法最终可以在计算机上进行数值实验，以证明其可靠性和有效性。

- 1.3. **算法的计算量分析**

【 1 】 算法的计算量

一个算法的计算量就是它所包含的 **四则运算** 的次数 (电脑只执行四则运算)，包括加法 (Addition) 、减法 (Difference) 、乘法 (Multiplication) 除法 (Division). 分析算法的计算量叫作 *算法复杂性分析*(Complexity Analysis).

奔腾处理器 (Intel Pentium Processor) 出现以前，通常只计算乘法 (Multiplication) 和除法 (Division) 的计算量；当今由于加减法与乘除法运算占据的内存已相差无几，则将所有的加减乘除 **四则运算** 的次数总和称为算法的计算量.

【 2 】 计算量的“高阶”原则

估计一个算法的计算量通常遵循“高阶”原则或“大 O ”原则，即只取数量级最高的项 kn^M 作为主要计算量，并说此算法的计算量为 kn^M 或 $O(n^M)$.

例如，某算法的计算量为 $2n^3+3n^2+5n$, 则可说此算法的计算量为 $2n^3$ 或 $O(n^3)$.

- 1.4. **算法的要素和解决对象**

【1】 算法的要素

判别算法优良性的要素大致包括可靠性 (Reliability)、精确性 (Precision)、便捷性 (Availability)、有效性 (Efficiency)、误差 (error)、数值稳定性 (Numerical Stability)、收敛性 (Convergence)、自适应性 (Self-adaptability)、复杂性 (Complexibility)、计算量和存贮量。

【2】 算法的解决对象 (1) 函数论：如函数的插值与逼近问题； (2) 微积分：数值积分和数值微分； (3) 代数：线性方程组和非线性方程及方程组的数值解如直接解法和迭代法等； (4) 微分方程：常微分方程和偏微分方程的数值解。

第 2 节　误差分析

- 2.1. **误差来源**

计算机程序演算的基本步骤是 (1) 建模：实际问题建立数学模型； (2) 数值化：将数学问题转化为数值问题； (3) 算法设计； (4) 根据算法编程计算。在以上每一步过程中，都可能出现误差，构成误差的来源。

【 1 】 **模型误差** (Modeling Error)　数学模型是对具体问题忽略次要因素进行抽象而获得，本身即是问题的近似，由此产生的误差称为 *模型误差* 。

【2】 **观测误差** (Observation Error) 数学模型中包含的参数如温度、密度、长度、时间、电压等是由人的观测或工具测量获得，与实际数据存在误差，称为 *观测误差* 。

【3】 **方法误差** (Method Error)(**截断误差** (Truncation Error)) 算法中包含的计算公式如泰勒公式等本身是一种求解的近似公式，由此产生的误差称为 *方法误差* 或 *截断误差* 。

【4】 **舍入误差** (Roundoff Error) 计算机中的数 (机器数) 是具有有限精度的实数的有限子集，称为 浮点数(Floating point) 。由于计算时的 四舍五入 ，或计算机的字长有限而使原始数据只能用有限位数表示，由此产生的误差称为 *舍入误差* 。

【例 1】 截断误差与 Taylor 公式 Taylor 公式

$$\begin{aligned} f(x) &= f(x_0) + f^{'}(x_0)(x-x_0) + \frac{1}{2!}f^{''}(x_0)(x-x_0)^2 + \cdots \\ &+ \frac{1}{n!}f^{(n)}(x_0)(x-x_0)^n + \frac{1}{(n+1)!}f^{(n+1)}(\xi)(x-x_0)^{n+1} \end{aligned} \tag{2.1}$$

截断误差即为余项

$$R_n(x-x_0) = \frac{1}{(n+1)!}f^{(n+1)}(\xi)(x-x_0)^{n+1} \tag{2.2}$$

. 如

(1) 正弦函数：

$$sinx = x - \frac{1}{3!}x^3 + \frac{1}{5!}x^5 - \frac{1}{7!}x^7 + \cdots \tag{2.3}$$

则 $R_n(x) \leq \dfrac{|\ x\ |^{n+1}}{(n+1)!}$;

(2) 指数函数:

$$e^x = 1 + x + \frac{1}{2!}x^2 + \frac{1}{3!}x^3 + \frac{1}{5!}x^5 + \cdots \tag{2.4}$$

则 $R_n(x) \leq \dfrac{e^{\theta x}}{(n+1)!} |\ x\ |^{n+1}, 0 < \theta < 1$;

【例 2 】 **舍入误差与机器精度** eps　考虑 Matlab 程序：

```
format long
x=4/3-1
y=3*x
z=1-y
```

精确值为 $z = 0$; 但实际输出为

```
x=0.33333333333333
y=1.00000000000000
z=2.220446049250313e-016
```

即是由机器精度有限造成的误差现象；这里的 z 即为十进制的机器精度 eps.

- 2.2. **误差与有效数字** Error and Significant Digits

【定义 1】 **绝对误差与绝对误差限** (Absolute Error)
设 x 为真值， x^* 为某近似值，近似值与真值的差
$e^* = x^* - x$ 称为 绝对误差 。并且 $e^* > 0$ 时称 x^* 为 强近似值； $e^* < 0$ 时称 x^* 为 弱近似值；绝对误差绝对值的上限
$\varepsilon^* = \sup \mid x^* - x \mid$ 称为 绝对误差限;

但有时仅用绝对误差并不能准确衡量精确程度：测量地球质量误差 1 千克和买两斤肉误差 1 千克显然不可同日而语；所谓一个人“五十步笑百步”就是说明此君的误差概念不清。因而有必要引入相对误差。

【定义 2】 **相对误差与相对误差限** (Relative Error) 绝对误差与真值的比 $e_r^* = \dfrac{e^*}{x} = \dfrac{x^* - x}{x}$ 称为 相对误差 。因为真值往往未知，有时也用近似值代替，从而得到 $e_r^* = \dfrac{e^*}{x^*} = \dfrac{x^* - x}{x^*}$ ；相对误差绝对值的上限 $\varepsilon_r^* = sup\dfrac{|\, x^* - x \,|}{|\, x \,|} = \dfrac{\varepsilon^*}{|\, x \,|}$ 称为 相对误差限; 或以近似值代替真值，则 $\varepsilon_r^* = sup\dfrac{|\, x^* - x \,|}{|\, x^* \,|} = \dfrac{\varepsilon^*}{|\, x^* \,|}$

【引例】 负幂多项式展开

一年的天数近似为

$$
\begin{aligned}
&365.13 = 300 + 60 + 5 + 0.1 + 0.03 \\
&= (3 \times 10^2 + 6 \times 10^1 + 5 \times 10^0 + 1 \times 10^{-1} + 3 \times 10^{-2}) \\
&= 10^2 \times (3 + 6 \times 10^{-1} + 5 \times 10^{-2} + 1 \times 10^{-3} + 3 \times 10^{-4})
\end{aligned}
$$

【定义 3 】 **有效数字** (Significant Digits) 若某实数值 x^* 可以表示为 10 的负幂多项式展开的形式：

$$x^* = \pm 10^m \times (a_1 + a_2 \times 10^{-1} + a_3 \times 10^{-2} + \cdots\cdots + a_n \times 10^{-(n-1)}) = \pm(a_1 10^m + a_2 10^{m-1} + a_3 10^{m-2} + \cdots\cdots + a_n 10^{m-n+1}) \tag{2.5}$$

则称 x^* 具有 n 位 有效数字；这里 $m \in Z$ 为整数，首位自然数不可取 0， $a_1 \in \{1,2,3,4,5,6,7,8,9\}$,（否则就没有首位了，也就没有这个数，好比“ 0803 ”其实就是 803 。 0 出现在中间的位数当然可以。）其余的 $a_k \in \{0,1,2,3,4,5,6,7,8,9\}, 2 \le k \le n.$

【命题 1】 **有效数字与绝对误差限：** 绝对误差限通常取为

$$\varepsilon^* = \frac{1}{2} \times 10^{m-n+1} \tag{2.6}$$

直观地看，就是绝对误差限通常都是 0.5,0.05,0.005 这些模样。其根源自然是我们熟知的“四舍五入”进位法。

【例 1】 **有效数字与绝对误差限** 一年的天数近似为

$$365.13 = 10^2 \times (3 + 6 \times 10^{-1} + 5 \times 10^{-2} + 1 \times 10^{-3} + 3 \times 10^{-4}) \tag{2.15}$$

则 $m = 2$, 有效数字 $n = 5$, 绝对误差限

$$\varepsilon^* = \frac{1}{2} \times 10^{m-n+1} = \frac{1}{2} \times 10^{2-5+1} = 0.005 \tag{2.16}$$

【命题 2】 **有效数字与相对误差限：**

若 x^* 具有 n 位 有效数字，即

$$x^* = \pm 10^m \times (a_1 + a_2 \times 10^{-1} + a_3 \times 10^{-2} + \cdots\cdots + a_n \times 10^{-(n-1)}) \tag{2.7}$$

则相对误差限有上界估计

$$\varepsilon_r^* \leq \frac{1}{2a_1} \times 10^{1-n} \tag{2.8}$$

反之若

$$\varepsilon_r^* \leq \frac{1}{2(a_1+1)} \times 10^{1-n} \tag{2.9}$$

则 x^* 至少具有 n 位 有效数字；

【证明】利用有界性。由于

$$x^* = \pm 10^m \times (a_1 + a_2 \times 10^{-1} + a_3 \times 10^{-2} + \cdots\cdots + a_n \times 10^{-(n-1)}) \tag{2.10}$$

故

$$10^m a_1 \leq | x^* | \leq 10^m (a_1 + 1) \tag{2.11}$$

比如

$$300 = 10^2 3 \leq | 365 | \leq 10^2 (3 + 1) = 400$$

从而（用不等式放大缩小法）

$$\varepsilon_r^* = \frac{\varepsilon^*}{\mid x^* \mid} \leq \frac{\frac{1}{2} \times 10^{m-n+1}}{10^m a_1} = \frac{1}{2a_1} \times 10^{1-n} \tag{2.12}$$

反之，若

$$\varepsilon_r^* \leq \frac{1}{2(a_1+1)} \times 10^{1-n} \tag{2.13}$$

则

$$\varepsilon^* = \mid x^* \mid \varepsilon_r^* \leq 10^m(a_1+1) \times \frac{1}{2(a_1+1)} \times 10^{1-n} = \frac{1}{2} \times 10^{m-n+1} \tag{2.14}$$

故 x^* 至少具有 n 位 有效数字；证毕。

【例 2 】 **有效数字与相对误差限** 欲使 $\sqrt{20}$ 的近似值的相对误差小于 0.001, 至少应当取几位有效数字？

【解】先估计 $\sqrt{20}$ 的首位数字，因 $16 < 20 < 25$, 故 $\sqrt{20} = 4.\cdots\cdots$, 首位数字为 $a_1 = 4$; 由效数字与相对误差限的联络公式，欲使

$$\varepsilon_r^* \leq \frac{1}{2a_1} \times 10^{1-n} \leq \frac{1}{8} \times 10^{1-n} < 10^{-3} \tag{2.17}$$

只须取 $n = 4$.

- 2.3. **误差的传播** Spread of Error

【1 四则运算中的传播】 (Addition, Difference,Multiplication,Division)　设两真值 x_1, x_2 的近似值为 x_1^*, x_2^*, 误差限为 $\varepsilon(x_1^*), \varepsilon(x_2^*)$, 则在四则运算中的误差限为：

和差：

$$\varepsilon(x_1^* \pm x_2^*) = \varepsilon(x_1^*) + \varepsilon(x_2^*) \tag{2.18}$$

积：

$$\varepsilon(x_1^* x_2^*) = \mid x_1^* \mid \varepsilon(x_2^*) + \mid x_2^* \mid \varepsilon(x_1^*) \tag{2.19}$$

商：

$$\varepsilon\left(\frac{x_1^*}{x_2^*}\right) = \frac{\mid x_1^* \mid \varepsilon(x_2^*) + \mid x_2^* \mid \varepsilon(x_1^*)}{\mid x_2^* \mid^2} \tag{2.20}$$

【证明】类似于微分法则的证明，用加减技巧。如

$$\begin{aligned} e(x_1^* x_2^*) = x_1^* x_2^* - x_1 x_2 &= x_1^* x_2^* - x_1 x_2^* + x_1 x_2^* - x_1 x_2 \\ &= (x_1^* - x_1) x_2^* + x_1 (x_2^* - x_2) \\ &\approx (x_1^* - x_1) x_2^* + x_1^* (x_2^* - x_2) \end{aligned}$$

取绝对误差限得

$$\varepsilon(x_1^* x_2^*) = \varepsilon(x_1^*) \mid x_2^* \mid + \varepsilon(x_2^*) \mid x_1^* \mid$$

其余同理可证。

【2 一元可微函数中的传播】 (Differentiation of One-variable) 近似值 x^* 绝对误差限为 $\varepsilon(x^*)$, 则一元可微函数 $f(x)$ 产生的误差限为：

$$\varepsilon(f(x^*)) = \mid f'(x^*) \mid \varepsilon(x^*) \tag{2.21}$$

【证】作 Taylor 展开,

$$f(x) = f(x^*) + f^{'}(x^*)(x - x^*) + \frac{1}{2} f^{''}(\xi)(x - x^*)^2 \tag{2.22}$$

从而

$$\mid f(x) - f(x^*) \mid \leq \mid f^{'}(x^*) \mid\mid (x - x^*) \mid + \frac{1}{2} \mid f^{''}(\xi) \mid (x - x^*)^2 \tag{2.23}$$

两边取上限并忽略高阶无穷小得

$$\varepsilon(f(x^*)) = \mid f'(x^*) \mid \varepsilon(x^*) \tag{2.24}$$

证毕.

【例 1】 **误差在一元可微函数中的传播**

自由落体位移函数 $s=\frac{1}{2}gt^2$，设重力常数 g 精确，而时间 t 有 ±0.1 秒的误差；证明当时间 t 增加时 s 的绝对误差在增加而相对误差在减少。

【解】 由题设 $\varepsilon(t^*)=0.1$ 秒，由误差传播公式

$$\varepsilon(s(t^*)) =\mid s'(t^*)\mid \varepsilon(t^*) = 0.1gt^* \tag{2.25}$$

而相对误差限

$$\varepsilon_r^*(s^*) = \frac{\varepsilon(s^*)}{s^*} = \frac{0.1gt^*}{\frac{1}{2}g(t^*)^2} = \frac{0.2}{t^*} \tag{2.26}$$

所以当时间 t 增加时 s 的绝对误差在增加而相对误差在减少.

第 3 节　数值稳定性与误差病态防治

- 3.1. **病态问题与条件数** (Morbidity and Conditional Number)

初值的微小扰动有可能引发解的巨大误差，比如蝴蝶效应和多米诺骨牌，由此产生病态问题。

【定义 1 病态问题】 (Morbidity Problem) 敏感依赖于舍入误差的问题称为 *病态问题* 。即输入数据的微小扰动将引发输出解的巨大误差。

【例 1 高次多项式】 高次多项式 $f(x)=x^{10}$, 取 $x=1, x^*=1.02$, 则 $f(x)=1, f(x^*)=1.24$, 从而

$$\varepsilon_r(x^*)=0.02, \quad \varepsilon_r(f(x^*))=0.24 \tag{3.1}$$

【例 2 正切函数】 正切函数 $f(x)=tanx$, 取 $x=1.57078, x^*=1.57079$, 则 $f(x)=6.12490\times 10^4, f(x^*)=1.58058\times 10^5$, 从而

$$\varepsilon_r(x^*)=10^{-5}, \quad \varepsilon_r(f(x^*))=10^5 \tag{3.2}$$

【定义 2 可微函数的条件数】 (Condition Number) 可微函数的函数相对误差限与自变量相对误差限的比值称为条件数 (Conditional Number):

$$C_p = \frac{\varepsilon_r(f(x^*))}{\varepsilon_r(x^*)}$$

我们有计算公式:

$$C_p = \left| \frac{xf'(x)}{f(x)} \right| \tag{3.3}$$

【证】利用可微函数误差传播公式

$$\varepsilon(f(x^*)) =| f'(x^*) | \varepsilon(x^*)$$

我们有

$$C_p = \frac{\varepsilon_r(f(x^*))}{\varepsilon_r(x^*)} = \frac{\dfrac{| f'(x^*) | \varepsilon(x^*)}{| f(x^*) |}}{\dfrac{\varepsilon(x^*)}{| x^* |}} =| \frac{x^* f'(x^*)}{f(x^*)} |=| \frac{xf'(x)}{f(x)} | \tag{3.4}$$

从而

$$C_p =| \frac{xf'(x)}{f(x)} | \tag{3.5}$$

【推论　可微函数的函数相对误差限与自变量相对误差限的联络公式】

$$\varepsilon_r(f(x^*)) = C_p \varepsilon_r(x^*) \tag{3.6}$$

【注】条件数是一个无量纲的数；在矩阵论中还有不同定义。当 $C_p > 10$ 时通常认为问题是数值不稳定的。

【例 3 高次多项式的条件数】 高次多项式 $f(x)=x^n, n>0$, 则

$$C_p = \left| \frac{xf'(x)}{f(x)} \right| = \left| \frac{nx^n}{x^n} \right| = n \tag{3.7}$$

如果 x 的相对误差为 0.02 ，则 $f(x)=x^n, n>0$ 的相对误差为 $0.02n$.

【例 4．平方根问题的条件数】

平方根函数 $f(x) = x^{1/2} = \sqrt{x}$, 则

$$C_p = \left| \frac{xf'(x)}{f(x)} \right| = \left| \frac{\frac{x}{2\sqrt{x}}}{\sqrt{x}} \right| = \frac{1}{2} \tag{3.8}$$

如果 x 的相对误差为 0.02，则计算 $f(x) = \sqrt{x}$ 的相对误差为 0.01. 因此平方根问题是非常良态的.

【例 5 几何函数的条件数】

(1) 球体积是关于半径的函数： $V=\frac{4}{3}\pi R^3$, 则

$$C_p = \left| \frac{RV'(R)}{V(R)} \right| = \left| \frac{4\pi R^3}{\frac{4}{3}\pi R^3} \right| = 3 \tag{3.8}$$

如果 R 的相对误差为 δ，则 V 的相对误差为 3δ;

(2) 正方形面积是关于边长的函数： $S=a^2$, 则

$$C_p = \left| \frac{aS'(a)}{S(a)} \right| = \left| \frac{2a^2}{a^2} \right| = 2 \tag{3.9}$$

如果 a 的相对误差为 δ，则 S 的相对误差为 2δ.

- 3.2. **数值稳定性** (Numerical Stability)

由误差传播公式，原始初值的微小误差会在计算过程中传播，犹如标记过的细菌在生物体中运动。如果舍入误差基本上不增长，即输入数据的微小扰动不会引发输出解的巨大误差，或能够得到有效控制，则称此算法是 数值稳定 的。

【例 1 定积分计算】 计算定积分 $I_n = e^{-1}\int_0^1 x^n e^x dx$, 估计误差。

【解】先由分部积分公式导出 $I_n = e^{-1}\int_0^1 x^n e^x dx$

$= e^{-1}(x^n e^x \mid_0^1 - \int_0^1 e^x dx^n)$

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

从而

$$I_{n-1} = \frac{1}{n}(1 - I_n) \tag{3.11}$$

现在我们有两种算法:

(1) 正向迭代：

$$\begin{cases} \tilde{I}_n &= 1 - n\tilde{I}_{n-1}, \\ \tilde{I}_0 &= 1 - \dfrac{1}{e} = 1 - 0.3679 = 0.6321. \end{cases} \tag{3.12}$$

相应第 n 步迭代的误差为

$$E_n = I_n - \tilde{I}_n = -n(I_{n-1} - \tilde{I}_{n-1}) = -nE_{n-1} = \cdots\cdots = (-1)^n n! E_0 \tag{3.13}$$

从而舍入误差以 $n!$ 的速度迅速扩大，造成算法的数值不稳定；

(2) 倒向迭代：先由积分中值定理作值的预估计：

$$e^{-1}\int_0^1 x^n dx < I_n = e^{-1}\int_0^1 x^n e^x dx < e^{-1}\int_0^1 x^n e dx \quad (3.14)$$

即

$$\frac{e^{-1}}{n+1} < I_n < \frac{1}{n+1} \quad (3.15)$$

再取定某一步的近似值作为倒向迭代初值，例如取 $I^*{}_9$；从而有迭代

$$\begin{cases} I^*_{n-1} = \frac{1}{n}(1 - I^*_n), \\ I^*{}_9 = \frac{1}{2}(\frac{e^{-1}}{10} + \frac{1}{10}) = 0.0684. \end{cases} \quad (3.16)$$

相应第 n 步迭代的误差为

$$E_n = \frac{(-1)^n}{n!} E_0 \tag{3.17}$$

从而舍入误差以 $n!$ 的速度迅速降低，使得算法数值稳定。

【练习 1 定积分计算】 给出定积分 $I_n = \int_0^1 \frac{x^n}{x+5} dx$ 的两种迭代算法，比较稳定性。

- 3.3. **误差病害的防治**

数值计算中对于误差有一套“望闻问切”的方法，需要我们掌握几个基本原则，以尽可能避免有效数字的损失。

【原则 1 避免两相近数相减】 若 $x \approx y$, 则 $x - y \approx 0$, 会损失相当多的有效数字；故应尽量避免两相近数相减。处理的方法是采用一些 <u>精细计算变形公式</u>：理论上是等价变形，而计算上却可获得更精细结果。

【例 1】 **二次方程求根问题** 首一实系数二次方程

$$x^2 + 2bx + c = 0 \tag{3.18}$$

有求根公式：

$$x = -b \pm \sqrt{b^2 - c} \tag{3.19}$$

考察判别式 $\Delta = 4(b^2 - c)$, 当 $\Delta > 0$, 存在两个互异实根；当 $b^2 \gg |\, c \,|$ 时， $\sqrt{b^2 - c} \approx |\, b \,|$, 此时直接用求根公式总会产生一个近似于 0 的根： $b < 0$ 时 $-b - \sqrt{b^2 - c} \approx 0$； $b > 0$ 时 $-b + \sqrt{b^2 - c} \approx 0$; 这时通常利用根与系数的关系公式即 Vieta 定理：先求出一个 <u>远离 0 的根</u>$x_1 = -b \pm \sqrt{b^2 - c}$, 再利用公式

$$x_2 = \frac{c}{x_1} \tag{3.20}$$

得到另一个根。

【实例】首一实系数二次方程

$$x^2 - 16x + 1 = 0$$

由求根公式： $x = 8 \pm \sqrt{63}$；若给定 $\sqrt{63} = 7.94$, 则有一根为 $x = 8 - \sqrt{63} = 8 - 7.94 = 0.06$ 近似于 0，只有一位有效数字；若先求出一 远离 0 的根 $x_1 = 8 + \sqrt{63} = 15.94$，再利用公式 $x_2 = \dfrac{c}{x_1} = \dfrac{1}{15.94} \approx 0.0627$ 得到另一个根，至少有 3 位有效数字；显然此种算法优越。

【 例 2 】 **方差计算问题**

在概率论中我们知道，样本 (独立同分布随机变量) 或其数值序列 $\{x_i\}$ 有均值公式：

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

而样本方差可由下式计算：

$$s^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2$$

显然，求和时所有的加和项都是平方形式，必然能够保证方差的非负性．但需要两次遍历样本点数据 – – 计算均值和方差 – – 所以效率相对不高．

此时若用理论上与之等价的另一个方差计算公式 (简单变形得到)：

$$s^2 = \frac{1}{n-1}\sum_{i=1}^{n}({x_i}^2 - n\bar{x}^2)$$

由于只要一次遍历样本点数据，它的效率通常要高于上一个公式．但由于加和项里相减的两个数值 ${x_i}^2$ 与 $n\bar{x}^2$ 相当接近，近似计算时将放大相对误差，更糟糕的是还有可能出现负值．所以有些手算时适用的公式，在电算时恐怕要忍痛割爱．

【常用精细计算变形公式】

(有关证明可自行思考，或参阅郑勋烨《数值分析》讲义)

1、根式函数型： $x \gg 1$ 时,

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

$$\sqrt{x+\frac{1}{x}}-\sqrt{x-\frac{1}{x}}=\frac{2}{x(\sqrt{x+\frac{1}{x}}+\sqrt{x-\frac{1}{x}})}$$

2、指数函数型： $x \approx 0$ 时,

$$\frac{e^{2x}-1}{2}=\frac{e^{x}sh2x}{2chx}$$

3、对数函数型： $x \gg 1$ 时,

$$ln(x+1) - lnx = ln\frac{x+1}{x}$$

$$ln(x-\sqrt{x^2-1}) = -ln(x+\sqrt{x^2-1})$$

4、三角函数型： $x \approx 0$ 时,

$$1 - cos2x = 2sin^2x = \frac{sin^2 2x}{1+cos2x}$$

$$tanx - sinx \approx -\frac{x^3}{2} - \frac{23x^5}{120}$$

5、积分函数型： $x \gg 1$ 时,

$$\int_N^{N+1} \frac{1}{1+x^2}dx = arctan\frac{1}{1+N(N+1)}$$

$$\int_N^{N+1} \frac{1}{1+x}dx = ln\frac{N+2}{N+1}$$

【实例 1 】

$$1 - cos2^{\circ} = 2sin^{2}1^{\circ} = 2 \times 0.0175^{2} \approx 0.0006092$$

【实例 2 】

$$(\sqrt{2}-1)^{6} = [(\sqrt{2}-1)^{2}]^{3} = (3-2\sqrt{2})^{3} = \frac{1}{(3+2\sqrt{2})^{3}}$$

【原则 2 注意运算次序，避免大数吃掉小数】 机器运算时要进行对阶，数量级小的数在与数量级大的数相加减时可能被视为 0，计算时应当先作小数的加和再加到大数上去。若不注意运算次序，四舍五入忽略掉相当多的小数，则可能造成这些小数积少成多得到的大数的丢失，引起巨大误差。犹如一条九斤重的大鱼一天中吞没的小鱼，可能有十九斤。我们却只看到了大鱼。

【例 1 】 **3 位十进制运算** 计算

$$x = 101 + \delta_1 + \delta_2 + \cdots + \delta_{100}, \quad 0.1 \le \delta_i \le 0.4 \tag{3.21}$$

若依四舍五入自左至右逐个相加，则所有小数 δ_i 均被舍去，得到 $x \approx 101$ ，误差甚大；但事实上，我们有估计

$$10 \le \sum_{1}^{100} \delta_i \le 40 \Rightarrow 111 \le x \le 141 \tag{3.22}$$

【例 2】 **8 位十进制浮点数运算** 给定 3 个数 $x \ll y \approx -z$ 如下：

$x = 0.23371258 \times 10^{-4}, y = 0.33671489 \times 10^{2}$

$z = -0.33677811 \times 10^{2}$.

计算其加和．精确值为

$x + y + z =$
$0.23371258 \times 10^{-4} + 0.33671489 \times 10^{2} - 0.33677811 \times 10^{2}$

$= 0.641371258 \times 10^{-3}$.

若依四舍五入自左至右逐个相加，则

$(x + y) + z =$
$(0.23371258 \times 10^{-4} + 0.33671489 \times 10^{2}) - 0.33677811 \times 10^{2}$

$= 0.64100000 \times 10^{-3}$

误差甚大，原因在于大数 y “吃掉” 了小数 x；而

$x + (y + z) =$
$0.23371258 \times 10^{-4} + (0.33671489 \times 10^{2} - 0.33677811 \times 10^{2})$
$= 0.64137126 \times 10^{-3}$.

与精确值非常相近.

【注记】 注意，在这里 $y \approx -z$ 是两个相近的数，本来应该避免二者相减．但由于在此处运算次序的限制是优先原则，两个相近的数相减得到的是数量级非常小的数 $\varepsilon = y + z = y - (-z) \ll x$，为了不让这个小数 ε 被比它大的数 x "吃掉"，我们必须首先获得这个小数，再与数量级大的 x 相加．因此在应用运算原则时，不可生搬硬套，还要通盘考虑．

由此例还可见：**浮点数运算不满足结合律**．这是近似计算与精确计算的重大区别．

【原则 3 采用适当算法，减少运算次数】 采用适当算法，简化计算步骤，减少运算次数，这是数值计算必须遵循的原则。

【算法 1 】 GAUSS **算法** 著名的小学算术问题

$$x = 1 + 2 + 3 + \cdots + 100 \tag{3.23}$$

通常的算法是自左至右逐个相加，要做 99 次加法； Gauss 的算法是

$$x = (1+100)+(2+99)+(3+98)+\cdots+(50+51) = 101\times 50 = 5050 \tag{3.24}$$

【算法 2】 **秦九韶 - 霍纳**（ Chin-Horner ）**算法**

多项式求值

$$p(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_2 x^2 + a_1 x + a_0 \qquad (3.25)$$

通常的算法是自左至右逐个计算 $a_k x^k$ 再相加，要做 $\dfrac{n(n+1)}{2}$ 次乘法和 n 次加法； 秦九韶 - 霍纳（ Chin-Horner ）算法 是利用因式分解的思想将多项式逐次

提取公因式，形成一种“嵌套式结构”：

$$
\begin{aligned}
p(x) &= a_n x^n + a_{n-1}x^{n-1} + \cdots + a_2x^2 + a_1x + a_0 \\
&= (a_nx + a_{n-1})x^{n-1} + \cdots + a_2x^2 + a_1x + a_0 \\
&= ((a_nx + a_{n-1}) + a_{n-2})x^{n-2} + \cdots + a_2x^2 + a_1x + a_0 \\
&= \cdots\cdots \\
&= (((a_nx + a_{n-1}) + a_{n-2})x + \cdots)x + a_0
\end{aligned}
\tag{3.26}
$$

其迭代格式为 倒向形式（ Backward Form ）：

$$\begin{cases} u_n & = a_n, \\ u_k & = u_{k+1}x + a_k, \quad k = n-1, \cdots, 1, 0 \\ u_0 & = p(x) \end{cases} \tag{3.27}$$

只需要做 n 次乘法和 n 次加法；

【注记】 **秦九韶**（ 1202 – 1261 ） 字道古，我国南宋末叶大数学家，为大奸臣贾似道所害。 1247 年著成《数术大略》，发展了数论的“大衍求一术”。

【 例 1 】 **以秦九韶 - 霍纳（ Chin-Horner ）算法进行多项式求值**

$$p(x) = x^5 - 3x^4 + 4x^2 - x + 1 \tag{3.28}$$

【解】

$$\begin{aligned} p(x) &= x^5 - 3x^4 + 4x^2 - x + 1 \\ &= (x-3)x^4 + 4x^2 - x + 1 \\ &= ((x-3)x^2 + 4)x^2 - x + 1 \\ &= (((x-3)x^2 + 4)x - 1)x + 1 \end{aligned} \tag{3.29}$$

比如在 $x = 3$ 处的取值为 $p(3) = 11 \times 3 + 1 = 34.$

【例 2】 以秦九韶 - 霍纳（ Chin-Horner ）算法进行多项式求值

$$p(x) = x^3 - 6.1x^2 + 3.2x + 1.5$$

在 $x = 4.71$ 处的精确值为 -14.263899. 分别以秦九韶 - 霍纳（ Chin-Horner ）算法和通常算法计算并比较结果.

【解】将多项式逐次提取公因式，形成一种“嵌套式结构”(Nested-structure) ：

$$\begin{aligned} p(x) &= x^3 - 6.1x^2 + 3.2x + 1.5 \\ &= (x - 6.1)x^2 + 3.2x + 1.5 \\ &= ((x - 6.1)x + 3.2)x + 1.5 \end{aligned}$$

使用截断法取近似值，（保留 3 位有效数字，3 位之后的数无论是否大于 5 全部舍弃）

$$p(4.71) = ((4.71 - 6.1)4.71 + 3.2)4.71 + 1.5 \approx -14.2$$

相当接近精确值 -14.263899.

但若采用逐个代入原来的多项式并根据同样的截断法取近似值，则有

$$p(4.71) = 4.71^3 - 6.1 \times 4.71^2 + 3.2 \times 4.71 + 1.5 \approx -13.5$$

其精度远不如秦九韶 - 霍纳（ Chin-Horner ）算法的计算结果.

【原则 4 选择主元素，避免小数为分母】

众所周知 0 不可作为分母，若 $\delta \approx 0$，则应避免出现 $\frac{1}{\delta}$ 的形式；若 $|y| \ll |x|$，则应避免出现 $\frac{x}{y}$ 的形式；对于线性方程组，即是选择大数为系数的未知元作为主元素（Pivoting Element），即放在系数矩阵的主对角线上用于消元的元素.

【例 1】 **线性方程组运算** 求解线性方程组

$$\begin{cases} 10^{-5}x + y &= 1, \\ x + y &= 2 \end{cases} \tag{3.30}$$

易由观察得知近似解为 $x = y = 1$. 如果采用不选主元素的消去法，第一个方程乘以 10^5, 方程组变形为

$$\begin{cases} x + 10^5 y &= 10^5, \\ (10^5 - 1)y &= 10^5 - 2 \end{cases} \tag{3.31}$$

得到近似解为 $x \approx 0, y \approx 1$, 严重失真。

而采用选取主元素的消去法，方程组变形为

$$\begin{cases} x + y &= 2, \\ (10^{-5} - 1)y &= 2 \times 10^{-5} - 1 \end{cases} \tag{3.32}$$

得到近似解为 $x \approx 1, y \approx 1$.

- 3.4. **数值实验**

【 实验 1 】 舍入误差造成的多项式图像的不光滑

实验目的：演示舍入误差造成的多项式图像的不光滑；

实验函数：多项式函数

$y = x.^7 - 7 * x.^6 + 21 * x.^5 - 35 * x.^4$

$+35 * x.^3 - 21 * x.^2 + 7 * x - 1$

源程序：

x=0.988:0.0001:1.012;

`y=x.`$^7 - 7 * x.^6 + 21 * x.^5 - 35 * x.^4 + 35 * x.^3$

`-21*x.`$^2 + 7 * x - 1;$

`plot(x,y);`

【实验 2】 舍入误差造成的多项式图像的不光滑

实验目的：演示舍入误差造成的多项式图像的不光滑 (保留函数，更改区间、步长之后的结果)；

实验函数：多项式函数

$y = x.^7 - 7 * x.^6 + 21 * x.^5 - 35 * x.^4 + 35 * x.^3$

$-21 * x.^2 + 7 * x - 1$

源程序：

```
x=0.99:0.001:1.01;
y=x.^7-7*x.^6+21*x.^5-35*x.^4+35*x.^3;
-21*x.^2+7*x-1;
plot(x,y);
```