第 1 章　数值分析与科学计算引论

1.1　数值分析的对象、作用与特点

1.1.1　数学科学与数值分析

数学是科学之母，科学技术离不开数学，它通过建立数学模型与数学产生紧密联系，数学又以各种形式应用于科学技术各领域. 数值分析也称计算数学，是数学科学的一个分支，它研究用计算机求解各种数学问题的数值计算方法及其理论与软件实现，用计算机求解科学技术问题通常经历以下步骤：

① 根据实际问题建立数学模型.

② 由数学模型给出数值计算方法.

③ 根据计算方法编制算法程序(数学软件)在计算机上算出结果.

第①步建立数学模型通常是应用数学的任务，而第②，③步就是计算数学的任务，也就是数值分析研究的对象，它涉及数学的各个分支，内容十分广泛. 作为“数值分析”课程，只介绍其中最基本、最常用的数值计算方法及其理论，它包括插值与数据逼近，数值微分与积分，线性方程组的数值求解，非线性方程与方程组求解，特征值计算，常微分方程数值解等. 它们都是以数学问题为研究对象，只是不像纯数学那样只研究数学本身的理论，而是把理论与计算紧密结合，着重研究数学问题的数值算法及其理论. 与其他数学课程一样，数值分析也是一门内容丰富，研究方法深刻，有自身理论体系的课程，既有纯数学高度抽象性与严密科学性的特点，又有应用广泛性与实际试验高度技术性的特点，是一门与计算机使用密切结合，实用性很强的数学课程.

1.1.2　计算数学与科学计算

几十年来由于计算机及科学技术的快速发展，求解各种数学问题的数值方法也愈来愈多地应用于科学技术各领域，新的计算性交叉学科分支不断涌现，如计算力学，计算物理，计算化学，计算生物学，计算经济学等，统称科学计算，它涉及数学的各个分支，研究它们适合于计算机编程的算法就是计算数学的研究范畴. 计算数学是各种计算性学科的共性基础，兼有基础性、应用性和边缘性的数学学科. 科学计算是一门工具性、方法性、边缘性的学科，发展迅速. 它与理论研究和科学实验成为现代科学发展的三种主要手段，它们相辅相成又互相独立. 在实际应用中导出的数学模型其完备形式往往不能方便地求出精确解，于是只能转化为简化模型求其数值解，如将复杂的非线性模型忽略一些因素而简化为可以求出精确解

的线性模型，但这样做往往不能满足近似程度的要求. 因此使用数值方法直接求解做较少简化的模型，可以得到满足近似程度要求的结果，使科学计算发挥更大的作用，这正是得益于计算机与计算数学的快速发展.

1.1.3 计算方法与计算机

数值分析也称计算方法，它与计算工具发展密切相关. 在电子计算机出现以前，计算工具只有算盘，算图，算表，算尺和手摇及电动计算机，计算方法只能计算规模较小的问题. 计算方法是数学的一个组成部分，很多方法都与当时的数学家名字相联系，如牛顿(Newton)插值公式，方程求根的牛顿法，解线性方程组的高斯(Gauss)消去法，多项式求值的秦九韶算法，计算积分的辛普森(Simpson)公式等. 这表明计算方法就是数学的一部分，它没有形成单独的学科分支. 只是在计算机出现以后，才使计算方法迅速发展并形成数学科学的一个独立分支——计算数学.

当代计算能力的大幅度提高既来自计算机的进步，也来自计算方法的进步，两者发展相辅相成又互相促进. 例如，1955 年至 1975 年的 20 年间计算机的运算速度提高数千倍，而同一时期解决一定规模的椭圆形偏微分方程计算方法的效率提高约一百万倍，说明计算方法的进步对提高计算能力的贡献更为重要，由于计算规模的不断扩大和计算方法的发展启发了新的计算机体系结构，诞生并发展了并行计算机. 而计算机的更新换代也对计算方法提出了新的标准和要求. 自计算机诞生以来，经典的计算方法业已经历了一个重新评价、筛选、改造和创新的过程，与此同时涌现了许多新概念、新课题和能发挥计算机解题潜力的新方法，这就构成了现代意义的计算数学.

1.1.4 数值问题与算法

能用计算机计算的“数值问题”是指输入数据(即问题中的自变量与原始数据)与输出数据(结果)之间函数关系的一个确定而无歧义的描述，输入输出数据可用有限维向量表示. 根据这种定义，“数学问题”有的是“数值问题”，如线性方程组求解. 也有不是“数值问题”的“数学问题”，如常微分方程 $\frac{dy}{dx}=x^2+y^2, y(0)=0$，它不是数值问题，因为输出不是数据而是连续函数 $y=y(x)$. 但只要将连续问题离散化，使输出数据是 $y(x)$ 在求解区间 $[a,b]$ 上的离散点 $x_i=a+ih(i=1,2,\cdots,n)$ 上的近似值，就是“数值问题”. 数值问题可用各种数值方法求解，这些数值方法就是算法. 计算方法就是研究各种“数值问题”的算法.

计算的基本单位称为算法元，它由算子、输入元和输出元组成. 算子可以是简单操作，如算术运算(+，−，×，/)、逻辑运算，也可以是宏操作，如向量运算、数组传输、基本初等函数求值等；输入元和输出元可分别视为若干变量或向量. 由一个或多个算法元组成一个进程，它是算法元的有限序列，一个数值问题的算法是指按规定顺序执行一个或多个完整的进程. 通过它们将输入元变换成一个输出元. 面向计算机的算法可分为串行算法和并行算法

两类，只有一个进程的算法适合于串行计算机，称为串行算法. 有两个以上进程的算法适合于并行计算机，称为并行算法. 对于一个给定的数值问题可以有许多不同的算法，它们都能给出近似答案，但所需的计算量和得到的精确程度可能相差很大. 一个面向计算机，有可靠理论分析且计算复杂性好的算法就是一个好算法. 理论分析主要是连续系统的离散化及离散型方程的数值问题求解，它包括误差分析、稳定性、收敛性等基本概念，它刻画了算法的可靠性、准确性. 计算复杂性包含计算时间复杂性与存储空间复杂性两个方面. 在同一规模、同一精度条件下，计算时间少的算法为计算时间复杂性好，而占用内存空间少的算法为存储空间复杂性好，它实际上就是算法中计算量与存储量的分析. 对解同一问题的不同算法其计算复杂性可能差别很大，例如解 n 阶的线性方程组，若依照克拉默(Cramer)法则用行列式解法要算 $n+1$ 个 n 阶行列式的值，对 $n=20$ 的线性方程组就需要 9.7×10^{21} 次乘除法运算，若用每秒亿次的计算机也要算 30 万年，这是无法实现的，若用高斯列主元消去法(见第 5 章)则只需做 3060 次乘除运算. 且 n 愈大相差就愈大，这表明算法研究的重要性，也说明只提高计算机速度，而不改进和选用好的算法是不行的.

综上所述，数值分析是研究数值问题的算法，概括起来有四点：

第一，面向计算机，要根据计算机的特点提供切实可行的有效算法. 即算法只能包括加、减、乘、除运算和逻辑运算，这些运算是计算机能直接处理的运算.

第二，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. 这些都建立在相应数学理论的基础上.

第三，要有好的计算复杂性，时间复杂性好是指节省计算时间，空间复杂性好是指节省存储空间，这也是建立算法要研究的问题，它关系到算法能否在计算机上实现.

第四，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.

根据“数值分析”课程的特点，学习时我们首先要注意掌握方法的基本原理和思想，要注意方法处理的技巧及其与计算机的结合，要重视误差分析、收敛性及稳定性的基本理论；其次，要通过例子，学习使用各种数值方法解决实际计算问题；最后，为了掌握本课的内容，还应做一定数量的理论分析与计算练习. 由于本课内容包括了微积分、线性代数、常微分方程的数值方法，读者必须掌握这几门课中与数值分析相关的基本内容，才能学好这门课程.

1.2 数值计算的误差

1.2.1 误差来源与分类

用计算机解决科学计算问题首先要建立数学模型，它是对被描述的实际问题进行抽象、简化而得到的，因而是近似的. 我们把数学模型与实际问题之间出现的这种误差称为**模型误差**. 只有实际问题提法正确，建立数学模型时又抽象、简化得合理，才能得到好的结果.

由于这种误差难于用数量表示，通常都假定数学模型是合理的，这种误差可忽略不计，在“数值分析”中不予讨论. 在数学模型中往往还有一些根据观测得到的物理量，如温度、长度、电压等，这些参量显然也包含误差. 这种由观测产生的误差称为**观测误差**，在“数值分析”中也不讨论这种误差. 数值分析只研究用数值方法求解数学模型产生的误差.

当数学模型不能得到精确解时，通常要用数值方法求它的近似解，其近似解与精确解之间的误差称为**截断误差**或**方法误差**. 例如，可微函数 $f(x)$ 用泰勒(Taylor)多项式

$$P_n(x) = f(0) + \frac{f'(0)}{1!}x + \frac{f''(0)}{2!}x^2 + \cdots + \frac{f^{(n)}(0)}{n!}x^n$$

近似代替，则数值方法的截断误差是

$$R_n(x) = f(x) - P_n(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!}x^{n+1}, \quad \xi \text{ 在 } 0 \text{ 与 } x \text{ 之间.}$$

有了求解数学问题的计算公式以后，用计算机做数值计算时，由于计算机的字长有限，原始数据在计算机上表示时会产生误差，计算过程又可能产生新的误差，这种误差称为**舍入误差**. 例如，用 3.141 59 近似代替 π，产生的误差

$$R = \pi - 3.141\ 59 = 0.000\ 002\ 6\cdots$$

就是舍入误差.

此外由原始数据或机器中的十进制数转化为二进制数产生的初始误差对数值计算也将造成影响，分析初始数据的误差通常也归结为舍入误差.

研究计算结果的误差是否满足精度要求就是误差估计问题，本书主要讨论算法的截断误差与舍入误差，而截断误差将结合具体算法讨论. 为分析数值运算的舍入误差，先要对误差基本概念做简单介绍.

1.2.2 误差与有效数字

定义 1 设 x 为准确值，x^* 为 x 的一个近似值，称 $e^* = x^* - x$ 为近似值的**绝对误差**，简称**误差**.

通常我们不能算出准确值 x，也不能算出误差 e^* 的准确值，只能根据测量工具或计算情况估计出误差的绝对值不超过某正数 ε^*，也就是误差绝对值的一个上界. ε^* 叫做近似值的**误差限**，它总是正数. 例如，用毫米刻度的米尺测量一长度 x，读出和该长度接近的刻度 x^*，x^* 是 x 的近似值，它的误差限是 0.5 mm，于是 $|x^* - x| \leqslant 0.5$ mm；如读出的长度为 765 mm，则有 $|765 - x| \leqslant 0.5$. 从这个不等式我们仍不知道准确的 x 是多少，但知道 $764.5 \leqslant x \leqslant 765.5$，即 x 在区间 $[764.5, 765.5]$ 内.

对于一般情形，$|x^* - x| \leqslant \varepsilon^*$，即

$$x^* - \varepsilon^* \leqslant x \leqslant x^* + \varepsilon^*,$$

这个不等式有时也表示为 $x = x^* \pm \varepsilon^*$.

误差限的大小还不能完全表示近似值的好坏. 例如，有两个量 $x = 10 \pm 1$，$y = 1000 \pm 5$，

则

$$x^* = 10,\quad \varepsilon_x^* = 1;\quad y^* = 1000,\quad \varepsilon_y^* = 5.$$

虽然 ε_y^* 是 ε_x^* 的 5 倍，但 $\varepsilon_y^*/y^* = \dfrac{5}{1000} = 0.5\%$ 比 $\varepsilon_x^*/x^* = \dfrac{1}{10} = 10\%$ 要小得多，这说明 y^* 近似 y 的程度比 x^* 近似 x 的程度要好得多. 所以，除考虑误差的大小外，还应考虑准确值 x 本身的大小. 我们把近似值的误差 e^* 与准确值 x 的比值

$$\frac{e^*}{x} = \frac{x^* - x}{x}$$

称为近似值 x^* 的**相对误差**，记作 e_r^*.

在实际计算中，由于真值 x 总是不知道的，通常取

$$e_r^* = \frac{e^*}{x^*} = \frac{x^* - x}{x^*}$$

作为 x^* 的相对误差，条件是 $e_r^* = \dfrac{e^*}{x^*}$ 较小，此时

$$\frac{e^*}{x} - \frac{e^*}{x^*} = \frac{e^*(x^* - x)}{x^* x} = \frac{(e^*)^2}{x^*(x^* - e^*)} = \frac{(e^*/x^*)^2}{1 - (e^*/x^*)}$$

是 e_r^* 的平方项级，故可忽略不计.

相对误差也可正可负，它的绝对值上界叫做**相对误差限**，记作 ε_r^*，即 $\varepsilon_r^* = \dfrac{\varepsilon^*}{|x^*|}$.

根据定义，上例中 $\dfrac{\varepsilon_x^*}{|x^*|} = 10\%$ 与 $\dfrac{\varepsilon_y^*}{|y^*|} = 0.5\%$ 分别为 x 与 y 的相对误差限，可见 y^* 近似 y 的程度比 x^* 近似 x 的程度好.

当准确值 x 有多位数时，常常按四舍五入的原则得到 x 的前几位近似值 x^*，例如

$$x = \pi = 3.141\,592\,65\cdots,$$

取 3 位　$x_3^* = 3.14$，$\varepsilon_3^* \leqslant 0.002$，

取 5 位　$x_5^* = 3.1416$，$\varepsilon_5^* \leqslant 0.000\,008$，

它们的误差都不超过末位数字的半个单位，即

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

定义 2　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，就说 x^* 有 n 位**有效数字**. 它可表示为

$$x^* = \pm 10^m \times (a_1 + a_2 \times 10^{-1} + \cdots + a_n \times 10^{-(n-1)}), \tag{2.1}$$

其中 $a_i (i = 1, 2, \cdots, n)$ 是 0 到 9 中的一个数字，$a_1 \neq 0$，m 为整数，且

$$|x - x^*| \leqslant \frac{1}{2} \times 10^{m-n+1}. \tag{2.2}$$

例如，取 $x^* = 3.14$ 作 π 的近似值，x^* 就有 3 位有效数字，取 $x^* = 3.1416 \approx \pi$，x^* 就有 5 位有效数字.

例 1 按四舍五入原则写出下列各数的具有 5 位有效数字的近似数：187.9325，0.037 855 51，8.000 033，2.718 281 8.

按定义，上述各数的具有 5 位有效数字的近似数分别是

$$187.93,\ 0.037\,856,\ 8.0000,\ 2.7183.$$

注意 $x=8.000\,033$ 的 5 位有效数字近似数是 8.0000 而不是 8，因为 8 只有 1 位有效数字.

例 2 如果以 $\mathrm{m/s^2}$ 为单位，重力常数 $g\approx 9.80\ \mathrm{m/s^2}$；若以 $\mathrm{km/s^2}$ 为单位，$g\approx 0.009\,80\ \mathrm{km/s^2}$，它们都具有 3 位有效数字，因为按第一种写法

$$|\,g-9.80\,|\leqslant \frac{1}{2}\times 10^{-2},$$

根据(2.1)式，这里 $m=0$，$n=3$；按第二种写法

$$|\,g-0.009\,80\,|\leqslant \frac{1}{2}\times 10^{-5},$$

这里 $m=-3$，$n=3$. 它们虽然写法不同，但都具有 3 位有效数字. 至于绝对误差限，由于单位不同结果也不同，$\varepsilon_1^*=\frac{1}{2}\times 10^{-2}\mathrm{m/s^2}$，$\varepsilon_2^*=\frac{1}{2}\times 10^{-5}\mathrm{km/s^2}$. 而相对误差相同，因为

$$\varepsilon_r^*=0.005/9.80=0.000\,005/0.009\,80.$$

注意相对误差与相对误差限是无量纲的，而绝对误差与误差限是有量纲的.

例 2 说明有效位数与小数点后有多少位数无关. 然而，从(2.2)式可得到具有 n 位有效数字的近似数 x^*，其绝对误差限为

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

在 m 相同的情况下，n 越大则 10^{m-n+1} 越小，故有效位数越多，绝对误差限越小.

至于有效数字与相对误差限的关系，有下面的定理.

定理 1 设近似数 x^* 表示为

$$x^*=\pm 10^m\times(a_1+a_2\times 10^{-1}+\cdots+a_l\times 10^{-(l-1)}),\qquad (2.1)'$$

其中 $a_i(i=1,2,\cdots,l)$ 是 0 到 9 中的一个数字，$a_1\neq 0$，m 为整数. 若 x^* 具有 n 位有效数字，则其相对误差限

$$\varepsilon_r^*\leqslant\frac{1}{2a_1}\times 10^{-(n-1)};$$

反之，若 x^* 的相对误差限 $\varepsilon_r^*\leqslant\frac{1}{2(a_1+1)}\times 10^{-(n-1)}$，则 x^* 至少具有 n 位有效数字.

证明 由(2.1)′式可得

$$a_1\times 10^m\leqslant|\,x^*\,|<(a_1+1)\times 10^m,$$

当 x^* 具有 n 位有效数字时

$$\varepsilon_r^*=\frac{|\,x-x^*\,|}{|\,x^*\,|}\leqslant\frac{0.5\times 10^{m-n+1}}{a_1\times 10^m}=\frac{1}{2a_1}\times 10^{-n+1};$$

反之,由

$$|x-x^*|=|x^*|\varepsilon_r^* < (a_1+1)\times 10^m \times \frac{1}{2(a_1+1)}\times 10^{-n+1}$$

$$=0.5\times 10^{m-n+1},$$

故 x^* 至少具有 n 位有效数字. 证毕.

定理 1 说明,有效位数越多,相对误差限越小.

例 3 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?

设取 n 位有效数字,由定理 1,$\varepsilon_r^* \leqslant \frac{1}{2a_1}\times 10^{-n+1}$. 由于$\sqrt{20}=4.4,\cdots$,知 $a_1=4$,故只要取 $n=4$,就有

$$\varepsilon_r^* \leqslant 0.125\times 10^{-3} < 10^{-3} = 0.1\%,$$

即只要对$\sqrt{20}$的近似值取 4 位有效数字,其相对误差限就小于0.1%.此时由开方表得 $\sqrt{20}\approx 4.472$.

1.2.3 数值运算的误差估计

设两个近似数 x_1^* 与 x_2^* 的误差限分别为 $\varepsilon(x_1^*)$及 $\varepsilon(x_2^*)$,则它们进行加、减、乘、除运算得到的误差限分别满足不等式

$$\varepsilon(x_1^* \pm x_2^*) \leqslant \varepsilon(x_1^*)+\varepsilon(x_2^*);$$

$$\varepsilon(x_1^* x_2^*) \leqslant |x_1^*|\varepsilon(x_2^*)+|x_2^*|\varepsilon(x_1^*);$$

$$\varepsilon(x_1^*/x_2^*) \leqslant \frac{|x_1^*|\varepsilon(x_2^*)+|x_2^*|\varepsilon(x_1^*)}{|x_2^*|^2}, \quad x_2^* \neq 0.$$

更一般的情况是,当自变量有误差时计算函数值也产生误差,其误差限可利用函数的泰勒展开式进行估计. 设 $f(x)$是一元可微函数,x 的近似值为 x^*,以 $f(x^*)$近似 $f(x)$,其误差界记作 $\varepsilon(f(x^*))$,由泰勒展开

$$f(x)-f(x^*) = f'(x^*)(x-x^*)+\frac{f''(\xi)}{2}(x-x^*)^2, \quad \xi\text{介于}x,x^*\text{之间},$$

取绝对值得

$$|f(x)-f(x^*)| \leqslant |f'(x^*)|\varepsilon(x^*)+\frac{|f''(\xi)|}{2}\varepsilon^2(x^*).$$

假定 $f'(x^*)$与 $f''(x^*)$的比值不太大,可忽略 $\varepsilon(x^*)$的高阶项,于是可得计算函数的误差限

$$\varepsilon(f(x^*)) \approx |f'(x^*)|\varepsilon(x^*).$$

当 f 为多元函数时,例如计算 $A=f(x_1,x_2,\cdots,x_n)$. 如果$x_1,x_2,\cdots,x_n$ 的近似值为 $x_1^*,x_2^*,\cdots,x_n^*$,则 A 的近似值为 $A^*=f(x_1^*,x_2^*,\cdots,x_n^*)$,于是由泰勒展开得函数值 A^* 的误差 $e(A^*)$为

$$e(A^*)=A^*-A=f(x_1^*,x_2^*,\cdots,x_n^*)-f(x_1,x_2,\cdots,x_n)$$
$$\approx\sum_{k=1}^{n}\left(\frac{\partial f(x_1^*,x_2^*,\cdots,x_n^*)}{\partial x_k}\right)(x_k^*-x_k)=\sum_{k=1}^{n}\left(\frac{\partial f}{\partial x_k}\right)^* e_k^*,$$

于是误差限

$$\varepsilon(A^*)\approx\sum_{k=1}^{n}\left|\left(\frac{\partial f}{\partial x_k}\right)^*\right|\varepsilon(x_k^*); \tag{2.3}$$

而 A^* 的相对误差限为

$$\varepsilon_r^*=\varepsilon_r(A^*)=\frac{\varepsilon(A^*)}{|A^*|}\approx\sum_{k=1}^{n}\left|\left(\frac{\partial f}{\partial x_k}\right)^*\right|\frac{\varepsilon(x_k^*)}{|A^*|}. \tag{2.4}$$

例 4 已测得某场地长 l 的值为 $l^*=110$ m,宽 d 的值为 $d^*=80$ m,已知 $|l-l^*|\leqslant 0.2$ m,$|d-d^*|\leqslant 0.1$ m. 试求面积 $s=ld$ 的绝对误差限与相对误差限.

解 因 $s=ld$, $\frac{\partial s}{\partial l}=d$, $\frac{\partial s}{\partial d}=l$,由(2.3)式知

$$\varepsilon(s^*)\approx\left|\left(\frac{\partial s}{\partial l}\right)^*\right|\varepsilon(l^*)+\left|\left(\frac{\partial s}{\partial d}\right)^*\right|\varepsilon(d^*),$$

其中

$$\left(\frac{\partial s}{\partial l}\right)^*=d^*=80\text{ m},\quad\left(\frac{\partial s}{\partial d}\right)^*=l^*=110\text{ m},$$

而 $\varepsilon(l^*)=0.2$ m,$\varepsilon(d^*)=0.1$ m,于是绝对误差限

$$\varepsilon(s^*)\approx 80\times(0.2)+110\times(0.1)=27(\text{m}^2);$$

相对误差限

$$\varepsilon_r(s^*)=\frac{\varepsilon(s^*)}{|s^*|}=\frac{\varepsilon(s^*)}{l^*d^*}\approx\frac{27}{8800}=0.31\%.$$

1.3 误差定性分析与避免误差危害

数值运算中的误差分析是个很重要而复杂的问题,1.2 节讨论了不精确数据运算结果的误差限,它只适用于简单情形,然而一个工程或科学计算问题往往要运算千万次,由于每步运算都有误差,如果每步都做误差分析是不可能的,也不科学,因为误差积累有正有负,绝对值有大有小,都按最坏情况估计误差限得到的结果比实际误差大得多,这种保守的误差估计不反映实际误差积累. 考虑到误差分布的随机性,有人用概率统计方法,将数据和运算中的舍入误差视为适合某种分布的随机变量,然后确定计算结果的误差分布,这样得到的误差估计更接近实际,这种方法称为**概率分析法**.

20 世纪 60 年代以后对舍入误差估计提出了一些新方法,较重要的是威尔金森(Wilkinson)的向后误差分析法和穆尔(Moore)的区间分析法. 但都不是十分有效,到目前为止舍入误差的定量估计尚无有效的分析方法,为确保数值计算的正确性通常只进行定性

分析.

1.3.1 算法的数值稳定性

用一个算法进行计算,由于初始数据误差在计算中传播使计算结果误差增长很快,就是数值不稳定的,先看下例.

例 5 计算 $I_n = e^{-1}\int_0^1 x^n e^x dx (n = 0,1,\cdots)$ 并估计误差.

由分部积分可得计算 I_n 的递推公式

$$\begin{cases} I_n = 1 - nI_{n-1}, & n = 1,2,\cdots, \\ I_0 = e^{-1}\int_0^1 e^x dx = 1 - e^{-1}. \end{cases} \tag{3.1}$$

若计算出 I_0,代入(3.1)式,可逐次求出 $I_1, I_2, \cdots$ 的值. 要算出 I_0 就要先计算 e^{-1},若用泰勒多项式展开部分和

$$e^{-1} \approx 1 + (-1) + \frac{(-1)^2}{2!} + \cdots + \frac{(-1)^k}{k!},$$

并取 $k=7$,用 4 位小数计算,则得 $e^{-1} \approx 0.3679$,截断误差 $R_7 = |e^{-1} - 0.3679| \leqslant \frac{1}{8!} < \frac{1}{4} \times 10^{-4}$. 计算过程中小数点后第 5 位的数字按四舍五入原则舍入,由此产生的舍入误差这里先不讨论. 当初值取为 $I_0 \approx 0.6321 = \tilde{I}_0$ 时,用(3.1)式递推的计算公式为

$$\text{(A)} \begin{cases} \tilde{I}_0 = 0.6321; \\ \tilde{I}_n = 1 - n\tilde{I}_{n-1}, & n = 1,2,\cdots. \end{cases}$$

计算结果见表 1-1 的 $\tilde{I}_n$ 列. 用 $\tilde{I}_0$ 近似 I_0 产生的误差 $E_0 = I_0 - \tilde{I}_0$ 就是初值误差,它对后面计算结果是有影响的.

表 1-1 计算结果

n	$\tilde{I}_n$(用(A)算)	I_n^*(用(B)算)	n	$\tilde{I}_n$(用(A)算)	I_n^*(用(B)算)
0	0.6321 ↓	0.6321	5	0.1480 ↓	0.1455
1	0.3679 ↓	0.3679	6	0.1120 ↓	0.1268
2	0.2642	0.2643	7	0.2160	0.1121
3	0.2074	0.2073 ↑	8	−0.7280	0.1035 ↑
4	0.1704	0.1708 ↑	9	7.552	0.0684 ↑

从表 1-1 中看到 $\tilde{I}_8$ 出现负值,这与一切 $I_n > 0$ 相矛盾. 实际上,由积分估值得

$$\frac{e^{-1}}{n+1} = e^{-1}(\min_{0\leqslant x\leqslant 1} e^x)\int_0^1 x^n dx < I_n < e^{-1}(\max_{0\leqslant x\leqslant 1} e^x)\int_0^1 x^n dx = \frac{1}{n+1}. \tag{3.2}$$

因此,当 n 较大时,用 $\tilde{I}_n$ 近似 I_n 显然是不正确的. 这里计算公式与每步计算都是正确的,那

么是什么原因使计算结果出现错误呢？主要就是初值 $\tilde{I}_0$ 有误差 $E_0=I_0-\tilde{I}_0$，由此引起以后各步计算的误差 $E_n=I_n-\tilde{I}_n$ 满足关系

$$E_n=-nE_{n-1},\quad n=1,2,\cdots.$$

由此容易推得

$$E_n=(-1)^n n!E_0,$$

这说明 $\tilde{I}_0$ 有误差 E_0，则 $\tilde{I}_n$ 就是 E_0 的 $n!$ 倍误差. 例如，$n=8$，若 $|E_0|=\frac{1}{2}\times10^{-4}$，则 $|E_8|=8!\times|E_0|>2$. 这就说明 $\tilde{I}_8$ 完全不能近似 I_8 了. 它表明计算公式(A)是数值不稳定的.

我们现在换一种计算方案. 由(3.2)式取 $n=9$，得

$$\frac{e^{-1}}{10}<I_9<\frac{1}{10},$$

我们粗略取 $I_9\approx\frac{1}{2}\left(\frac{1}{10}+\frac{e^{-1}}{10}\right)=0.0684=I_9^*$，然后将公式(3.1)倒过来算，即由 I_9^* 算出 I_8^*，I_7^*，$\cdots$，I_0^*，公式为

$$\text{(B)}\begin{cases}I_9^*=0.0684,\\ I_{n-1}^*=\dfrac{1}{n}(1-I_n^*),\quad n=9,8,\cdots,1;\end{cases}$$

计算结果见表 1-1 的 I_n^* 列. 我们发现 I_0^* 与 I_0 的误差不超过 10^{-4}. 记 $E_n^*=I_n-I_n^*$，则 $|E_0^*|=\frac{1}{n!}|E_n^*|$，$E_0^*$ 比 E_n^* 缩小了 $n!$ 倍，因此，尽管 E_9^* 较大，但由于误差逐步缩小，故可用 I_n^* 近似 I_n. 反之，当用方案(A)计算时，尽管初值 $\tilde{I}_0$ 相当准确，由于误差传播是逐步扩大的，因而计算结果不可靠. 此例说明，数值不稳定的算法是不能使用的.

定义 3 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.

在例 5 中算法(B)是数值稳定的，而算法(A)是不稳定的.

1.3.2 病态问题与条件数

对一个数值问题本身如果输入数据有微小扰动(即误差)，引起输出数据(即问题解)相对误差很大，这就是病态问题. 例如，计算函数值 $f(x)$ 时，若 x 有扰动 $\Delta x=x-x^*$，其相对误差为 $\frac{\Delta x}{x}$，函数值 $f(x^*)$ 的相对误差为 $\frac{f(x)-f(x^*)}{f(x)}$. 相对误差比值

$$\left|\frac{f(x)-f(x^*)}{f(x)}\right|\Big/\left|\frac{\Delta x}{x}\right|\approx\left|\frac{xf'(x)}{f(x)}\right|=C_p,\tag{3.3}$$

C_p 称为计算函数值问题的条件数. 自变量相对误差一般不会太大，如果条件数 C_p 很大，将引起函数值相对误差很大，出现这种情况的问题就是病态问题.

例如，取 $f(x)=x^n$，则有 $C_p=n$，它表示相对误差可能放大 n 倍. 如 $n=10$，有 $f(1)=1$，$f(1.02)\approx 1.24$，若取 $x=1$，$x^*=1.02$ 自变量相对误差为 2%，函数值相对误差为 24%，这时问题可以认为是病态的. 一般情况下，条件数 $C_p\geqslant 10$ 就认为是病态，C_p 越大病态越严重.

例 6 求解线性方程组

$$\begin{cases} x+\alpha y=1, \\ \alpha x+y=0. \end{cases} \tag{3.4}$$

解 当 $\alpha=1$ 时，系数行列式为零，方程无解，但当 $\alpha\neq 1$ 时解为 $x=\dfrac{1}{1-\alpha^2}$，$y=-\dfrac{\alpha}{1-\alpha^2}$. 当 $\alpha\approx 1$ 时，若输入数据 α 有微小扰动(误差)，则解的误差很大. 例如，取 $\alpha=0.99$，则解 $x\approx 50.25$；如果 α 有误差 0.001，取 $\alpha^*=0.991$，则解 $x^*\approx 55.81$，误差 $|x^*-x|\approx 5.56$ 很大，表明此时线性方程组(3.4)是病态的. 实际上，由 $x=\dfrac{1}{1-\alpha^2}$ 是 α 的函数，利用(3.3)式可求得

$$C_p=\left|\frac{\alpha x'(\alpha)}{x(\alpha)}\right|=\left|\frac{2\alpha^2}{1-\alpha^2}\right|.$$

当 $\alpha=0.99$ 时 $C_p\approx 100$，表明条件数很大，故问题是病态的.

注意病态问题不是计算方法引起的，是数值问题自身固有的，因此，对数值问题首先要分清问题是否病态，对病态问题就必须采取相应的特殊方法以减少误差危害.

1.3.3 避免误差危害

数值计算中通常不采用数值不稳定算法，在设计算法时还应尽量避免误差危害，防止有效数字损失，通常要避免两相近数相减和用绝对值很小的数做除数，还要注意运算次序和减少运算次数. 下面举例说明.

例 7 求 $x^2-16x+1=0$ 的小正根.

解 $x_1=8+\sqrt{63}$，$x_2=8-\sqrt{63}\approx 8-7.94=0.06=x_2^*$，$x_2^*$ 只有一位有效数字. 若改用

$$x_2=8-\sqrt{63}=\frac{1}{8+\sqrt{63}}\approx\frac{1}{15.94}\approx 0.0627,$$

具有三位有效数字.

例 8 计算 $A=10^7(1-\cos 2^\circ)$(用四位数学用表).

由于 $\cos 2^\circ=0.9994$，直接计算

$$A=10^7(1-\cos 2^\circ)=10^7(1-0.9994)=6\times 10^3.$$

只有一位有效数字. 若利用 $1-\cos x=2\sin^2\dfrac{x}{2}$，则

$$A=10^7(1-\cos 2^\circ)=2\times(\sin 1^\circ)^2\times 10^7=6.13\times 10^3,$$

具有三位有效数字(这里 $\sin 1^\circ=0.0175$).

此例说明,可通过改变计算公式避免或减少有效数字的损失. 类似地,如果 x_1 和 x_2 很接近时,则

$$\lg x_1 - \lg x_2 = \lg \frac{x_1}{x_2}.$$

用右边算式有效数字就不损失. 当 x 很大时,

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

都用右端算式代替左端. 一般情况,当 $f(x) \approx f(x^*)$ 时,可用泰勒展开

$$f(x) - f(x^*) = f'(x^*)(x - x^*) + \frac{f''(x^*)}{2}(x - x^*)^2 + \cdots$$

取右端的有限项近似左端. 如果无法改变算式,则采用增加有效位数进行运算;在计算机上则采用双倍字长运算,但这要增加机器计算时间且多占内存单元.

例 9 在五位十进制计算机上,计算

$$A = 52\,492 + \sum_{i=1}^{1000} \delta_i,$$

其中 $0.1 \leqslant \delta_i \leqslant 0.9$.

把运算的数写成规格化形式

$$A = 0.524\,92 \times 10^5 + \sum_{i=1}^{1000} \delta_i.$$

由于在计算机内计算时要对阶,若取 $\delta_i = 0.9$,对阶时 $\delta_i = 0.000\,009 \times 10^5$,在五位的计算机中表示为机器 0,因此

$$A = 0.524\,92 \times 10^5 + 0.000\,009 \times 10^5 + \cdots + 0.000\,009 \times 10^5$$
$$\triangleq 0.524\,92 \times 10^5 \text{(符号 } \triangleq \text{ 表示机器中相等)},$$

结果显然不可靠,这是由于运算中出现了大数 52 492"吃掉"小数 δ_i 造成的. 如果计算时先把数量级相同的一千个 δ_i 相加,最后再加 52 492,就不会出现大数"吃"小数现象,这时

$$0.1 \times 10^3 \leqslant \sum_{i=1}^{1000} \delta_i \leqslant 0.9 \times 10^3,$$

于是

$$0.001 \times 10^5 + 0.524\,92 \times 10^5 \leqslant A \leqslant 0.009 \times 10^5 + 0.524\,92 \times 10^5,$$
$$52\,592 \leqslant A \leqslant 53\,392.$$

例 10 利用公式

$$\ln(1+x) = \sum_{n=1}^{\infty} (-1)^{n+1} \frac{x^n}{n}$$

的前 N 项和,可计算 $\ln 2$ 的近似值(令 $x=1$). 若要精确到 10^{-5},需要对 $N=100\,000$ 项求和,不但计算量很大,其舍入误差积累也很严重. 但若改用

$$\ln\frac{1+x}{1-x}=2\left(x+\frac{x^3}{3}+\frac{x^5}{5}+\cdots+\frac{x^{2n+1}}{2n+1}+\cdots\right),$$

取 $x=1/3$，只要计算前 10 项之和，其截断误差便小于 10^{-10}.

1.4 数值计算中算法设计的技术

在数值计算中算法设计好坏不但影响计算结果的精度，还可大量节省计算时间，下面给出几个具有代表性的算法，其基本原则是数值分析中常用的，应引起读者关注.

1.4.1 多项式求值的秦九韶算法

一个计算问题如果能减少运算次数，不但可节省计算量还可减少舍入误差，这是算法设计中一个重要原则，以多项式求值为例，设给定 n 次多项式

$$p(x)=a_0x^n+a_1x^{n-1}+\cdots+a_{n-1}x+a_n,\quad a_0\neq 0,$$

求 x^* 处的值 $p(x^*)$. 若直接计算每一项 a_ix^{n-i} 再相加，共需求

$$\sum_{i=0}^{n}(n-i)=1+2+\cdots+n=\frac{n(n+1)}{2}=O(n^2)$$

次乘法，n 次加法. 若采用

$$p(x)=(\cdots(a_0x+a_1)x+\cdots+a_{n-1})x+a_n,$$

它可表示为

$$\begin{cases}b_0=a_0,\\ b_i=b_{i-1}x^*+a_i,\quad i=1,2,\cdots,n,\end{cases}\tag{4.1}$$

则 $b_n=p(x^*)$ 即为所求. 此算法称为秦九韶算法，用它计算 n 次多项式 $p(x)$ 的值只用 n 次乘法和 n 次加法，乘法次数由 $O(n^2)$ 降为 $O(n)$，且只用 $n+2$ 个存储单元，这是计算多项式值最好的算法，它是我国南宋数学家秦九韶于 1247 年提出的，国外称此算法为 Hernor 算法，是 1819 年给出的，比秦九韶算法晚 500 多年.

秦九韶算法还有另一个好处是求 $p'(x)$ 在 x^* 点的值. 实际上由(4.1)式有

$$\begin{aligned}p(x)&=(x-x^*)(b_0x^{n-1}+\cdots+b_{n-2}x+b_{n-1})+b_n\\&=(x-x^*)q(x)+b_n,\\p(x^*)&=b_n,\end{aligned}$$

其中

$$q(x)=b_0x^{n-1}+b_1x^{n-2}+\cdots+b_{n-2}x+b_{n-1}.$$

对 x 求导得

$$p'(x)=q(x)+(x-x^*)q'(x),$$

故 $p'(x^*)=q(x^*)$. 从而得用秦九韶算法(4.1)计算 $p'(x^*)$ 的算法如下：

$$\begin{cases} c_0 = b_0, \\ c_i = c_{i-1}x^* + b_i, \quad i = 1,2,\cdots,n-1, \end{cases} \tag{4.2}$$

则 $c_{n-1}=q(x^*)=p'(x^*)$. 具体计算可见下例.

例 11 设 $p(x)=2x^4-3x^2+3x-4$,用秦九韶算法求 $p(-2)$及 $p'(-2)$的值.

解 用(4.1)式及(4.2)式构造出下列计算表格(表 1-2):

表 1-2 系数表

	x^4 系数	x^3 系数	x^2 系数	x^1 系数	常数项
	$a_0=2$	$a_1=0$	$a_2=-3$	$a_3=3$	$a_4=-4$
		$b_0x^*=-4$	$b_1x^*=8$	$b_2x^*=-10$	$b_3x^*=14$
$x^*=-2$	$b_0=2$	$b_1=-4$	$b_2=5$	$b_3=-7$	$b_4=10$
$x^*=-2$		$c_0x^*=-4$	$c_1x^*=16$	$c_2x^*=-42$	
	$c_0=2$	$c_1=-8$	$c_2=21$	$c_3=-49=p'(-2)$	

此处 $b_4=p(-2)=10$,$q(x)=2x^3-4x^2+5x-7$,$c_3=q(-2)=p'(-2)=-49$.

减少乘除法运算次数是算法设计中十分重要的一个原则,另一典型例子是离散傅里叶(Fourier)变换(DFT),如点数太多其计算量太大,即使高速计算机也难于广泛使用,直至 20 世纪 60 年代提出 DFT 的快速算法 FFT 才使它得以广泛使用. FFT 算法就是快速算法的一个典范.

1.4.2 迭代法与开方求值

迭代法是一种按同一公式重复计算逐次逼近真值的算法,是数值计算普遍使用的重要方法,以开方运算为例,它不是四则运算,因此在计算机上求开方值就要转化为四则运算,使用的就是迭代法.

假定 $a>0$,求$\sqrt{a}$等价于解方程

$$x^2 - a = 0. \tag{4.3}$$

这是方程求根问题,可用迭代法求解(见第 7 章). 现在用简单方法构造迭代法,先给一个初始近似 $x_0>0$,令 $x=x_0+\Delta x$,Δx 是一个校正量,称为增量,于是(4.3)式化为

$$(x_0+\Delta x)^2 = a, \quad \text{即} \quad x_0^2 + 2x_0\Delta x + (\Delta x)^2 = a^2.$$

由于 Δx 是小量,若省略高阶项$(\Delta x)^2$,则得

$$x_0^2 + 2x_0\Delta x \approx a, \quad \text{即} \quad \Delta x \approx \frac{1}{2}\left(\frac{a}{x_0} - x_0\right).$$

于是

$$x = x_0 + \Delta x \approx \frac{1}{2}\left(x_0 + \frac{a}{x_0}\right) = x_1.$$

这里 x_1 不是$\sqrt{a}$的真值,但它是真值 $x=\sqrt{a}$的进一步近似,重复以上过程可得到迭代公式

$$x_{k+1}=\frac{1}{2}\left(x_k+\frac{a}{x_k}\right),\quad k=0,1,2,\cdots, \tag{4.4}$$

它可逐次求得 $x_1,x_2,\cdots$,若

$$\lim_{k\to\infty}x_k=x^*,$$

则 $x^*=\sqrt{a}$,容易证明序列$\{x_k\}$对任何 $x_0>0$ 均收敛,且收敛很快.

例 12 用迭代法(4.4)求$\sqrt{3}$,取 $x_0=2$.

解 若计算精确到 10^{-6},由(4.4)式可求得

$$x_1=1.75,\quad x_2=1.732\,14,$$
$$x_3=1.732\,051,\quad x_4=1.732\,051,$$

计算停止. 由于$\sqrt{3}=1.732\,050\,8\cdots$,可知只要迭代 3 次误差即小于$\frac{1}{2}\times10^{-6}$.

迭代法(4.4)每次迭代只做一次除法一次加法与一次移位(右移一位就是除以 2),计算量很小. 计算机中求$\sqrt{a}$一般只要精度达到 10^{-8}即可,只需 4～5 次迭代就能达到精度要求,计算量很少,计算机(含计算器)中计算$\sqrt{a}$用的就是迭代法(4.4).

无论在实用上或理论上,求解线性或非线性方程,迭代法都是重要的方法,本书将多处论及.

1.4.3 以直代曲与化整为“零”

在数值计算中将非线性问题线性化(在几何上体现为在局部范围内用直线近似曲线)是常用方法.

圆周率 π 的计算是古代数学的一个光辉成就,早在公元前 3 世纪阿基米德用内接正 96 边形与外切正 96 边形的周长近似圆周,求得 $\pi\approx3.14$. 圆是曲边图形,圆面积的计算是数学方法上以直代曲的典范,公元 3 世纪我国魏晋时期大数学家刘徽(早祖冲之二百多年)用“割圆术”求得 $\pi\approx3.1416$. 他不是将正多边形的边数固定在一个确定数目上,而是从正六边形(即 6 等分圆周)做起,逐次二分各弧段,做 k 次后将圆周分割为 6×2^k 个小扇形,化整为“零”,然后以弦代弧,即用弦所在的直线段代替其上小扇形的曲边,用小三角形面积代替曲边小扇形面积(如图 1-1 所示),再求和就得圆面积 $S=\pi r^2$ 的近似值 $\overline{S}$,从而可求得 $\pi\approx\frac{\overline{S}}{r^2}$. 显然,分割次数越多,结果越准确. “割圆术”中提出“割之又割”,直至无穷,最终以内接正 6×2^k 边形面积的极限求得圆面积 S,这与 17 世纪发明微积分的思想极其相似! 但数值计算不取极限,只是采用以直代曲与化整为“零”求和的思想. 通常将非线性问题线性化,在几何图形上就是以直代曲. 例如求函数方程 $f(x)=0$ 的根,在几何上 $y=f(x)$表现为平面上的一条曲线,它与 x 轴交点的横坐标即为方

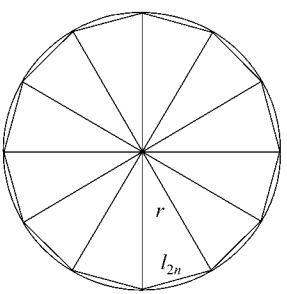

图 1-1

程的根 x^*，假如已给出一个近似根 x_k，用该点$(x_k, f(x_k))$处的切线逼近该曲线，令 x_{k+1}为该切线与 x 轴交点的横坐标，一般情况下，x_{k+1}近似方程的根 x^* 的程度比 x_k 近似 x^* 的程度要好(如图 1-2). 上述以直代曲相当于用切线方程

$$y = f(x_k) + f'(x_k)(x - x_k) = 0$$

的根 x_{k+1}近似 x^*，从而

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}, \quad k = 0,1,2,\cdots. \tag{4.5}$$

这就是方程求根的牛顿迭代法(见第 7 章)，它是以直代曲建立迭代序列的典型例子.

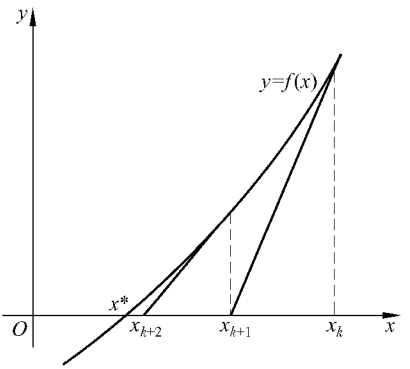

图 1-2

在微积分中计算定积分

$$I(f) = \int_a^b f(x)\mathrm{d}x$$

的梯形公式

$$I(f) \approx \frac{b-a}{2}[f(a) + f(b)] = T_1. \tag{4.6}$$

它是用通过曲线 $y=f(x)$上两点 $A(a,f(a))$及 $B(b,f(b))$的直线(弦)近似曲线的弧，用梯形面积近似曲边梯形面积(如图 1-3)，这也是以直代曲. 为提高计算精度仍然采用化整为“零”，将$[a,b]$分割为小区间 $a=x_0<x_1<\cdots<x_n=b$，其中

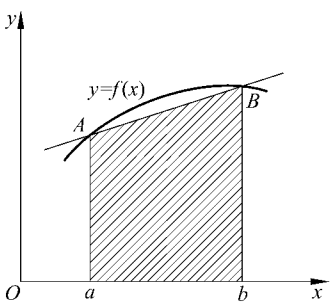

图 1-3

$$x_i = a + ih, \quad h = \frac{b-a}{n}.$$

在每个小区间$[x_{i-1}, x_i]$$(i=1,2,\cdots,n)$上用梯形公式计算，再求和得到

$$I(f) \approx \sum_{i=1}^{n} \frac{h}{2}[f(x_{i-1}) + f(x_i)] = T_n, \tag{4.7}$$

称它为复合梯形公式，显然$\lim\limits_{n\to\infty} T_n = I(f)$. 只要取足够大的 n 就可得到满足精度要求的积分值 $I(f)$.

1.4.4 加权平均的松弛技术

刘徽用“割圆术”求得 $\pi=3.1416$，如果单纯用“割圆”计算相当于割到 3072 边形，计算量是惊人的！在古代没有计算工具只用手算是十分困难的. 但他不是单纯采用“割圆”计算，而是利用了现代计算方法中的松弛技术，令内接正 n 边形面积 S_n 近似圆面积 S，取半径 $r=10$，计算出

$$S_{96} = 313\frac{584}{625}, \quad S_{192} = 314\frac{64}{625},$$

用松弛法，令 $\overline{S}=S_{192}+\omega(S_{192}-S_{96})$，$\omega$ 为松弛参数.

若取 $\omega=\dfrac{36}{105}$，则得

$$\overline{S}=314\frac{64}{625}+\frac{36}{105}\left(314\frac{64}{625}-313\frac{584}{625}\right)=314\frac{4}{25}=314.16,$$

于是

$$\pi=\frac{\overline{S}}{r^2}=\frac{314.16}{100}=3.1416.$$

$\overline{S}$ 与 $S_{3072}=314.1590$ 近似，但计算量却大大节省. 松弛技术是计算方法中一种提高收敛速度的有效方法，设量 $x=x^*$ 为精确值，x_0 与 x_1 为 x^* 的两个近似值，其加权平均为

$$\overline{x}=x_1+\omega(x_1-x_0)=(1+\omega)x_1-\omega x_0,$$

其中 ω 称为松弛因子. 通常 x_1 是比 x_0 更接近真值 x^*，要求 $\overline{x}$ 比 x_1 更接近 x^*，可选 $\omega>0$. 若增量 $\omega\Delta x_0=\omega(x_1-x_0)$ 选得适当，$\overline{x}$ 就可最好地逼近真值 x^*，当然选最优的 ω 很困难，但在“割圆术”中刘徽找到了 $\omega=\dfrac{36}{105}$，使

$$\overline{S}=S_{192}+\frac{36}{105}(S_{192}-S_{96})$$

是一个很接近真值 S 的近似.

在数值计算中利用松弛技术的方法称为**松弛法**，它也是“数值分析”中常用的方法，如在积分近似计算的梯形求积公式(4.7)中，取 $n=1,2$ 可分别得

$$T_1=\frac{b-a}{2}[f(a)+f(b)],$$

$$T_2=\frac{b-a}{4}[f(a)+2f(c)+f(b)],\quad c=\frac{a+b}{2}.$$

为了得到 $I(f)$ 更精确的近似同样可用松弛法，令

$$S_1=T_2+\omega(T_2-T_1)\approx(1+\omega)T_2-\omega T_1,$$

若取 $\omega=1/3$，则得

$$S_1=\frac{4}{3}T_2-\frac{1}{3}T_1=\frac{b-a}{6}[f(a)+4f(c)+f(b)].\tag{4.8}$$

这就是计算积分 $I(f)$ 的辛普森公式(见第 4 章)，比梯形公式精度高，在迭代法中使用松弛技术同样可加速收敛.

1.5 数学软件

本书主要介绍科学计算中最常用的数值方法，对算法不做详细描述，具体算法细节通常可利用现有的数学软件，书中虽对个别算法给出计算步骤，这也只是为了让读者加深对算法的理解，提高选择合适算法的能力，并能广泛地使用算法. 我们的目标是使读者能在已有的

数学软件库中选择有关数值方法的软件并算出结果，为此需对本课涉及的数学软件包做简单介绍.

在计算机上进行科学计算传统的算法语言是 Fortran 语言，C 语言是一种比 Fortran 更灵活和更具表现力的语言，是目前教学中普遍使用的语言，但很多软件包是用 Fortran 语言编写的，最早开发的软件包 EISPACK 是第一个大型数值软件包，LINPACK 是求解线性方程组和最小二乘问题的 Fortran 子程序包，1992 年问世的 LAPACK 是将上述两个软件中的算法集整合成一个统一的更新软件包而代替了 LINPACK 和 EISPACK. 这些软件包是高效的、精确的和可靠的，易于维护和移植，可直接从有关网站或文献中获得.

商业软件包也代表了数值方法当前的技术水平，它们的内容往往以公共域软件包为基础，但在函数库中包括了每一种问题的求解方法. 其中 IMSL(International Mathematical and Statistical Libraries，国际数学与统计学库)，分别由数值数学，统计学和特殊函数的 MATH，STAT，SFUN 程序库组成，包含 900 多个子程序，解决了大部分常见的数值分析问题. NAG(Numerical Algorithms Group，数值算法集)是一个综合数学软件库，整个程序含 1000 多个由 FORTRAN 编写的子程序，约 400 个 C 子程序和 200 多个 FORTRAN 90 子程序. NAG 程序库包含有大部分数值分析标准算法的子程序.

MATLAB 是 MATrix LABoratory 的缩写，即矩阵实验室，它整合了非线性方程组、数值积分、三次样条函数、曲线拟合、最优化、常微分方程和绘图工具等功能，但它主要是以 EISPACK 和 LINPACK 子程序为基础. MATLAB 目前是由 C 和汇编语言编写的，它的基本结构是执行矩阵运算，是一个对求解线性方程特别有用的功能强大的自包容系统. MATLAB 的基本数据单元是不需要指定维数和特殊说明的矩阵，可把它看成一种计算机语言，它比其他高级语言简单方便，但绝不能取代高级语言.

评　注

1.1 节对“数值分析”所做的介绍，目的是使读者对它的内容、特点、作用、历史等有概括的了解，进一步学习可参看《中国大百科全书·数学》中有关计算数学的条目，数值分析的历史可参见文献[10]，关于科学计算在计算机时代的最新进展可见文献[11]. 误差分析的一般讨论是“数值分析”教材包含的基本内容，但有关舍入误差的定量分析是一个困难的问题，本章未做讨论，文中提到的威尔金森的向后误差估计可参看文献[12]，他还提出了大量计算反例，说明了某些算法的不稳定性，这方面较新的研究成果可见文献[13]. 关于利用区间运算进行误差分析，除了穆尔的最早著作[14]，还有一些文献[15,16]，但实际应用仍较困难. 因此本书更看重有关算法稳定性和问题的病态性，关于病态问题的一般概念可参看文献[17]. 1.4 节是关于数值计算的几个典型例子，它代表了数值计算算法设计的一些共同思想和方法. 在本书后面章节均可见到. 1.5 节所介绍的数学软件是本书各章都要用到的. 更详细的内容可见文献[7,8].

复习与思考题

1. 什么是数值分析？它与数学科学和计算机的关系如何？

2. 何谓算法？如何判断数值算法的优劣？

3. 列出科学计算中误差的三个来源，并说出截断误差与舍入误差的区别.

4. 什么是绝对误差与相对误差？什么是近似数的有效数字？它与绝对误差和相对误差有何关系？

5. 什么是算法的稳定性？如何判断算法稳定？为什么不稳定算法不能使用？

6. 什么是问题的病态性？它是否受所用算法的影响？

7. 什么是迭代法？试利用 $x^3-a=0$ 构造计算$\sqrt[3]{a}$的迭代公式.

8. 直接利用以直代曲的原则构造求方程 $x^2-a=0$ 的根 $x^*=\sqrt{a}$的迭代法.

9. 举例说明什么是松弛技术.

10. 考虑无穷级数 $\sum\limits_{n=1}^{\infty}\dfrac{1}{n}$，它是发散的，在计算机上计算它的部分和，会得到什么结果？为什么？

11. 判断下列命题的正确性：

(1) 解对数据的微小变化高度敏感是病态的.

(2) 高精度运算可以改善问题的病态性.

(3) 无论问题是否病态，只要算法稳定都能得到好的近似值.

(4) 用一个稳定的算法计算良态问题一定会得到好的近似值.

(5) 用一个收敛的迭代法计算良态问题一定会得到好的近似值.

(6) 两个相近数相减必然会使有效数字损失.

(7) 计算机上将 1000 个数量级不同的数相加，不管次序如何结果都是一样的.

习　题

1. 设 $x>0$，x 的相对误差为 δ，求 $\ln x$ 的误差.

2. 设 x 的相对误差为 2%，求 x^n 的相对误差.

3. 下列各数都是经过四舍五入得到的近似数，即误差限不超过最后一位的半个单位，试指出它们是几位有效数字：

$$x_1^*=1.1021,\quad x_2^*=0.031,\quad x_3^*=385.6,$$
$$x_4^*=56.430,\quad x_5^*=7\times 1.0.$$

4. 利用公式(2.3)求下列各近似值的误差限：

(1) $x_1^*+x_2^*+x_4^*$；

(2) $x_1^* x_2^* x_3^*$；

(3) x_2^* / x_4^*.

其中 $x_1^*, x_2^*, x_3^*, x_4^*$ 均为第 3 题所给的数.

5. 计算球体积要使相对误差限为 1%，问度量半径 R 时允许的相对误差限是多少？

6. 设 $Y_0 = 28$，按递推公式

$$Y_n = Y_{n-1} - \frac{1}{100}\sqrt{783}, \quad n = 1,2,\cdots$$

计算到 Y_{100}. 若取 $\sqrt{783} \approx 27.982$(5 位有效数字)，试问计算 Y_{100} 将有多大误差？

7. 求方程 $x^2 - 56x + 1 = 0$ 的两个根，使它至少具有 4 位有效数字($\sqrt{783} \approx 27.982$).

8. 当 $x \approx y$ 时计算 $\ln x - \ln y$ 有效位数会损失. 改用 $\ln x - \ln y = \ln \frac{x}{y}$ 是否就能减少舍入误差？(提示：考虑对数函数何时出现病态).

9. 正方形的边长大约为 100 cm，应怎样测量才能使其面积误差不超过 1 cm^2？

10. 设 $S = \frac{1}{2}gt^2$，假定 g 是准确的，而对 t 的测量有 ± 0.1 秒的误差，证明当 t 增加时 S 的绝对误差增加，而相对误差却减少.

11. 序列 $\{y_n\}$ 满足递推关系

$$y_n = 10y_{n-1} - 1, \quad n = 1,2,\cdots,$$

若 $y_0 = \sqrt{2} \approx 1.41$(三位有效数字)，计算到 y_{10} 时误差有多大？这个计算过程稳定吗？

12. 计算 $f = (\sqrt{2} - 1)^6$，取 $\sqrt{2} \approx 1.4$，利用下列等式计算，哪一个得到的结果最好？

$$\frac{1}{(\sqrt{2}+1)^6}, \quad (3 - 2\sqrt{2})^3,$$

$$\frac{1}{(3+2\sqrt{2})^3}, \quad 99 - 70\sqrt{2}.$$

13. $f(x) = \ln(x - \sqrt{x^2 - 1})$，求 $f(30)$ 的值. 若开平方用 6 位函数表，问求对数时误差有多大？若改用另一等价公式

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

计算，求对数时误差有多大？

14. 用秦九韶算法求多项式 $p(x) = 3x^5 - 2x^3 + x + 7$ 在 $x = 3$ 处的值.

15. 用迭代法 $x_{k+1} = \frac{1}{1+x_k}$ $(k = 0,1,\cdots)$ 求方程 $x^2 + x - 1 = 0$ 的正根 $x^* = \frac{-1+\sqrt{5}}{2}$，取 $x_0 = 1$，计算到 x_5，问 x_5 有几位有效数字.

16. 用不同的方法计算积分 $\int_0^{1/2} e^x dx$：

(1) 用原函数计算到 6 位小数.