数值逼近

杨　畅　史晓冉　编著

科学出版社

数值逼近

杨　畅　史晓冉　编著

科学出版社
北京

内 容 简 介

本书是根据理科数值逼近教学大纲要求及学科发展需要编写的, 全书共 6 章, 包括绪论、多项式插值、曲线曲面的拟合、正交多项式与函数逼近、数值积分、有理逼近介绍. 本书以浅显的方法讲解理论, 并配以大量的图例进行说明, 力求做到让数值逼近的理论知识变得通俗易懂.

本书可作为数学与应用数学专业和信息与计算科学专业本科生的教材, 也可以作为工科各专业研究生和科研及工程技术人员的参考书.

图书在版编目(CIP)数据

数值逼近/杨畅, 史晓冉编著. —北京: 科学出版社, 2017.11

ISBN 978-7-03-055045-3

Ⅰ. ①数… Ⅱ. ①杨… ②史… Ⅲ. ①数值逼近 Ⅳ. ①O174.41

中国版本图书馆 CIP 数据核字(2017) 第 265189 号

责任编辑: 张中兴 梁 清 / 责任校对: 张凤琴

责任印制: 吴兆东 / 封面设计: 迷底书装

科学出版社 出版

北京东黄城根北街 16 号

邮政编码: 100717

http://www.sciencep.com

北京教图印刷有限公司 印刷

科学出版社发行 各地新华书店经销

*

2017 年 11 月第 一 版 开本: 720×1000 B5

2017 年 11 月第一次印刷 印张: 15

字数: 302 000

定价: 49.00 元

(如有印装质量问题, 我社负责调换)

前　言

数值逼近是指各种逼近问题的数值分析. 该课程是计算数学学科的核心课程, 也是各学科数值计算的基础. 作者多年来担任哈尔滨工业大学数学系数值逼近课程的任课教师, 将所讲授内容与心得整理成书. 全书所述内容适合数学与应用数学专业和信息与计算科学专业高年级本科生学习, 也可以作为工科各专业研究生和科研人员及工程技术人员的参考材料.

数值逼近课程的首要目标是介绍有哪些逼近问题, 如函数逼近、数据逼近、积分逼近等问题. 其次, 数值逼近课程要介绍针对每种逼近问题有哪些逼近方法, 以及每种方法是如何构造的. 接下来, 也是数值逼近课程的重点, 是对每种逼近方法的逼近效果的分析. 最后, 通过大量的实例来检验数值逼近方法的优劣. 数值逼近课程的教学目标是具有独立的认识逼近问题本质的能力; 针对逼近问题能构造并分析逼近方法; 能用计算机实现逼近问题的数值逼近计算.

本书编写的内容正是呼应数值逼近课程的学习目标. 全书共 6 章, 第 1，2，4，5 章由杨畅编写, 第 3，6 章由史晓冉编写. 各章节的主要内容和特点如下.

- 第 1 章是绪论, 首先，主要说明什么是数值逼近; 其次, 给出度量逼近程度的工具, 即误差; 最后，介绍数值计算中避免和减小误差的若干原则.
- 第 2 章介绍多项式插值问题, 它可以针对已知函数, 也可以针对离散数据. 首先对一般的插值问题给出了多种等价的构造方法, 以及各方法的优劣比较; 接着给出了含有导数插值条件的 Hermite 插值方法; 然后介绍了 Runge 现象以及分段插值插值方法, 其中包括重要的样条插值方法; 最后介绍了几种常用的多元多项式插值方法.
- 第 3 章介绍曲线曲面的拟合方法. 通过控制点设计曲线曲面的形状, 其中数据点不需要被完全匹配而避免了插值的 Runge 现象. 首先介绍了 Bezier 曲线的定义、特殊的几何性质, 以及曲线的求值、分割、升阶及光滑拼接等算法. 并引入了有理 Bezier 曲线、张量积 Bezier 曲面的定义与性质. 接下来介绍 B 样条曲线, 既能保证一定的光滑性, 又克服了 Bezier 曲线控制点的全局控制性. 最后说明 B 样条曲线的求值分割、节点插入等算法, 并简单介绍了最常用的一般有理 B 样条 (NURBS) 方法.
- 第 4 章的目的是介绍函数逼近理论, 即在无穷范数或二范数意义下用多项式对连续函数或离散数据的逼近. 无穷范数意义下的逼近即为最佳一致逼近, 介绍的内容包括最佳一致逼近的存在性、特征、收敛速度、近似方法. 二

范数意义下即为最佳平方逼近, 其中针对离散数据的最佳平方逼近又称为最小二乘法. 介绍的内容包括最佳平方逼近的存在唯一性、构造方法及其与最佳一致逼近多项式的比较. 值得说明的是, 由于函数逼近中使用了大量的正交多项式, 所以特别地将正交多项式单独作为一节来介绍.

- 第 5 章的目的是介绍数值积分的基础知识. 首先是插值型求积公式, 它在等距节点情形时就是 Newton-Cotes 求积公式. 接着由于高阶 Newton-Cotes 公式不稳定, 所以引出了复合求积公式. 为了进一步提高复合求积公式的效率, 于是通过外插值法引出了复合求积公式的加速方法, 即 Romberg 方法. 最后, 在非等距节点情形下, 介绍了能达到最高逼近精度的 Gauss 型求积公式及其变形形式.
- 第 6 章的目的是简单介绍非线性问题中常用的有理逼近方法. 对于多项式逼近问题, 求解参数是线性的, 有关计算简单且有很好的光滑性. 但由于多项式不能很好地逼近无界函数等问题, 进而引入有更好逼近效果的有理函数作为复杂函数的近似, 介绍了最佳一致有理逼近的定义、有理函数插值算法以及 Pade 逼近方法.

为了能更好地学习数值逼近课程, 建议读者的学习过程分为以下几个步骤. 首先应该充分掌握各种逼近方法的构造思想, 并能复述出逼近方法; 其次理解各种逼近方法的误差估计, 并应用误差估计去辨别逼近方法的优劣; 最后应用逼近方法练习每章节的习题, 并通过计算机验证数值结果.

本书用通俗简明的语言来描述数值逼近理论知识, 通过大量的算例和图示说明逼近方法的逼近效果. 每章的结尾配以一定数量的习题, 便于读者练习. 为了方便读者上机实践, 本书还给出了部分逼近方法的算法. 本书最后列出了书中提及或相关的参考文献, 供读者进一步的参考学习.

由于作者水平所限, 不妥之处在所难免, 敬请读者予以批评指正.

作　者

2017 年 6 月于哈尔滨工业大学

目　录

第1章 绪 论

1.1 数值分析简介

随着数学在工程中应用的不断发展, 许多的工程难题被抽象为各种数学模型. 要实际解决这些工程难题, 就必须找到其对应数学模型的解. 然而, 解决这些数学模型面临着以下困难:

(1) 某些数学问题在理论上没有解决的方法, 解决此类问题必须用数值计算方法求其近似值. 例如求多项式的根, 若多项式的次数大于或等于五次, 则无求根公式.

(2) 某些数学问题在理论上有解决方法, 但在实际中并不实用, 必须寻找新的行之有效的计算方法. 例如克拉默 (Cramer) 法则, 当线性方程组的系数矩阵非奇异时, 可以给出线性方程组的解. 但在实际计算中, 当线性方程组的规模大于 3 时, 应用克拉默法则求解线性方程组是不实用的. 因为克拉默法则中需要计算多个行列式, 而行列式的计算常常要消耗大量的计算资源. 因此克拉默法则的计算效率非常低.

(3) 某些数学问题即使在实践中有解决的方法, 仍然需要进行误差分析. 这里涉及数值稳定性, 会在 1.3 节中详细讲解.

由此可见, 数值求解数学问题显得十分重要, 而研究数值求解的方法和理论称为数值分析. 数值分析中所研究的内容众多, 主要分为三大类: 数值逼近, 即各种逼近问题的数值分析; 数值代数, 即代数问题的数值计算方法及其有关理论; 微分方程数值求解, 即常微分方程和偏微分方程的数值解法.

本书着重介绍数值逼近, 包括函数插值、样条理论、函数及离散数据在特定意义下的逼近及数值积分等. 这里提及的内容会在后续章节中逐一地讲解.

1.2 误 差 分 析

1.2.1 误差的来源

在解决实际问题时, 如果想要系统地研究问题, 就必须将其抽象为数学模型. 例如, 在投射测试中, 希望知道投射物的飞行距离. 设投射物的实际飞行距离为 f_p. 又假设投射物的飞行轨迹可以由以下光滑曲线来逼近:

$$\begin{cases} x(t, d_i, i = 1, 2, \cdots), \\ y(t, d_i, i = 1, 2, \cdots), \end{cases}$$

其中 $d_i, i = 1, 2, \cdots$ 代表模型参数, 例如投射角度、风速等. 于是, 投射物的飞行距离可以表示为

$$f(d_i, i = 1, 2, \cdots) = \int_0^T \sqrt{x'(t, d_i, i = 1, 2, \cdots)^2 + y'(t, d_i, i = 1, 2, \cdots)^2}\mathrm{d}t.$$

因此, 由数学建模所产生的误差为 $e_m = |f_p - f(d_i, i = 1, 2, \cdots)|$. 注意到数学模型中的参数是通过测量得到的, 而在测量过程中可能产生参数的误差 $\delta d_i, i = 1, 2, \cdots$. 则由测量误差导致的误差为 $e_s = |f(d_i, i = 1, 2, \cdots) - f(d_i + \delta d_i, i = 1, 2, \cdots)|$. 接着数学模型的求解过程中需要用到数值计算方法, 得到的数值解为 $f_n(d_i + \delta d_i, i = 1, 2, \cdots)$, 而误差为 $e_n = |f(d_i + \delta d_i, i = 1, 2, \cdots) - f_n(d_i + \delta d_i, i = 1, 2, \cdots)|$. 最后, 在计算过程中, 计算机数位长度的限制, 也会对最终的计算结果产生影响. 记最终得到的计算结果为 $\hat{f}_n(d_i + \delta d_i, i = 1, 2, \cdots)$, 则相应的误差为 $e_a = |f_n(d_i + \delta d_i, i = 1, 2, \cdots) - \hat{f}_n(d_i + \delta d_i, i = 1, 2, \cdots)|$. 因此, 由计算所产生的误差为 $e_c = e_n + e_a$, 而全局误差为 $e = e_m + e_s + e_c$.

综上所述, 误差的来源分为以下四种情况.

(1) 模型误差, 即实际问题解与数学模型解之间的误差.

(2) 测量误差, 即测量具体数据时产生的误差.

(3) 截断误差 (也称方法误差), 即数学模型的准确解与数值计算方法的解之间的误差.

(4) 舍入误差, 即由计算机字长限制而产生的误差.

各误差之间的联系总结在图 1.1 中.

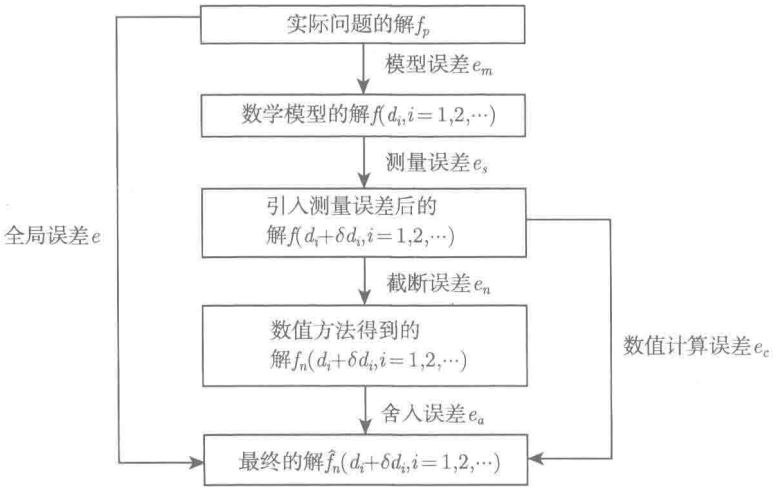

图 1.1　误差的来源

1.2.2 误差的度量

对于同一个数学问题, 采用不同的度量方法会得出不同的结果. 一般度量误差的标准有以下三种形式.

1. 绝对误差与绝对误差限

定义 1.2.1 设 x 为某个量的精确值, x^* 是它的一个近似值, 则称 $E(x^*) = |x - x^*|$ 为近似值 x^* 的绝对误差, 简称误差. 一般情况下, 精确值 x 是未知的, 因此 x^* 的绝对误差 $E(x^*)$ 也就求不出来. 但是如果能求出 x^* 误差一个范围 $E(x^*) = |x - x^*| \leqslant \delta(x^*)$, 则称 $\delta(x^*)$ 为近似值 x^* 的绝对误差限, 简称误差限.

例 1.2.1 设 $x = \pi = 3.1415926535\cdots$. 若取 x 的一个近似值 $x^* = 3.14159$, 则有

$$E(x^*) = |x - x^*| \leqslant 0.5 \times 10^{-5},$$

称 x^* 的误差限为 $\delta(x^*) = 0.5 \times 10^{-5}$.

一个近似数的误差限并不唯一, 通常取满足 $E(x^*) \leqslant 0.5 \times 10^n (n$ 为整数) 的最小值.

2. 相对误差与相对误差限

绝对误差有时不能完全刻画一个近似数的精确程度. 例如测量一个书桌和一个体育场的面积, 误差都是 1cm^2. 显然后者的测量更精确. 因此, 决定某量的近似值的精度, 除了考虑绝对误差的大小, 还要考虑该量自身的大小.

定义 1.2.2 设 x 为某个量的精确值, x^* 是它的一个近似值, 则称 $E_r(x^*) = \dfrac{|x - x^*|}{|x|}$ 为近似值 x^* 的相对误差. 一般情况下, 精确值 x 是未知的, 因此在实际计算中常取$E_r(x^*) = \dfrac{|x - x^*|}{|x^*|}$ 作为 x^* 的相对误差. 若 $E_r(x^*)$ 小于某个已知正数 $\delta_r(x^*)$, 即 $E_r(x^*) = \dfrac{|x - x^*|}{x^*} \leqslant \delta_r(x^*)$, 则称 $\delta_r(x^*)$ 为近似值 x^* 的相对误差限.

例 1.2.2 取 x 和 x^* 同例 1.2.1, 则有

$$E_r(x^*) = \frac{|x - x^*|}{|x^*|} \leqslant 0.8 \times 10^{-6},$$

称 x^* 的相对误差限为 $\delta_r(x^*) = 0.8 \times 10^{-6}$.

3. 有效数字

当某量的精确值的位数较多时, 通常采用四舍五入的方法取 x 的前面若干位, 作为 x 的近似值.

定义 1.2.3　如果近似值 x^* 的误差不超过 x^* 最后一位数字的半个单位，若该位数字到 x^* 的第一位非零数字共有 n 位，那么这 n 位数字称为 x^* 的有效数字，并称 x^* 具有 n 位有效数字. 如果用数学语言表示，设

$$x^* = \pm 10^k \times 0.a_1 a_2 \cdots a_n \cdots,$$

其中 $a_i, (i = 1, 2, \cdots)$ 是 $0 \sim 9$ 的整数，且 $a_1 \neq 0$，k 为整数. 如果

$$|x - x^*| \leqslant \frac{1}{2} \times 10^{k-n},$$

则称 x^* 为 x 的具有 n 位有效数字的近似值.

例 1.2.3　取 x 和 x^* 同例 1.2.1，且有

$$E(x^*) \approx 0.000003 < 0.000005.$$

故 x^* 具有 6 位有效数字.

例 1.2.4　设 $z = \mathrm{e}^{-5} = 0.0067379\cdots$，近似值为 $z^* = 0.00673$. 容易验证

$$E(z^*) \approx 0.000008 < 0.00005.$$

故 6, 7 是 z^* 的有效数字，而 3 不是有效数字.

4. 三种度量间的关系

绝对误差、相对误差和有效数字都是用来度量近似数的误差的，它们之间必然存在一定的联系. 实际上，由相对误差的定义可知，相对误差与绝对误差的关系是：$E_r(x^*) = E(x^*)/|x^*|$. 由有效数字的定义可知，有效数字与绝对误差的关系如下：若 $E(x^*) = |x - x^*| \leqslant \frac{1}{2} \times 10^{k-n}$，则 x^* 具有 n 位有效数字. 而有效数字与相对误差的关系可由以下定理得到.

定理 1.2.1　设 x 的近似值为 x^*，

(1) 若 x^* 有 n 位有效数字，则 $\dfrac{|x - x^*|}{|x^*|} \leqslant \dfrac{1}{2a_1} \times 10^{1-n}$；

(2) 若 $\dfrac{|x - x^*|}{|x^*|} \leqslant \dfrac{1}{2(a_1 + 1)} \times 10^{1-n}$，则 x^* 至少具有 n 位有效数字.

证明　(1) 由 x^* 的定义有

$$a_1 \times 10^{k-1} \leqslant |x^*|.$$

故当 x^* 有 n 位有效数字时，有

$$\frac{|x - x^*|}{|x^*|} \leqslant \frac{0.5 \times 10^{k-n}}{a_1 \times 10^{k-1}} = \frac{1}{2a_1} \times 10^{1-n}.$$

即命题 (1) 得证.

(2) 又由 x^* 的定义有

$$|x^*| \leqslant (a_1+1)\times 10^{k-1}.$$

根据命题 (2) 的条件有

$$\begin{aligned}|x-x^*| &\leqslant \frac{1}{2(a_1+1)}\times 10^{1-n}|x^*| \\ &\leqslant \frac{1}{2(a_1+1)}\times 10^{1-n}\times(a_1+1)\times 10^{k-1} \\ &\leqslant \frac{1}{2}\times 10^{k-n}.\end{aligned}$$

故 x^* 具有 n 位有效数字. 即命题 (2) 得证. □

1.2.3 先验估计和后验估计

数值方法的稳定性可以从不同角度来分析, 其中主要分为先验估计和后验估计两类.

先验估计就是度量计算所得结果 f_n 与真实结果 f 之间的误差. 先验估计又分为向前误差分析和向后误差分析两种.

(1) 向前误差分析: 即找到数值结果误差 $\delta f = |f_n - f|$ 的一个上界. 而这个误差可能是由初始数值的误差或者数值算法的截断误差造成的.

(2) 向后误差分析: 假设不考虑计算误差的前提下, 给出初始数据误差的估计. 即将计算结果表示为 $f_n = f(d_i + \delta f_i, i = 1, 2, \cdots)$, 再将 $|\delta d_i|, i = 1, 2, \cdots$ 的上界估计出来.

先验误差估计可以被用在数值方法稳定性的分析中, 也可以用在收敛性分析中.

与先验误差估计不同, 后验误差估计是基于数值方法已计算出来的数值结果, 来给出误差估计.

例 1.2.5 用分段梯形求积公式计算积分. 设定积分准确值为 I, 当区间被划分为 n 段时, 用分段梯形求积公式求得的结果为 I_n. 当区间被划分为 $2n$ 段时, 用分段梯形求积公式求得的结果为 I_{2n}. 则三个量之间有如下关系 (具体分析过程参见 5.5 节):

$$I - I_n \approx \frac{1}{1-(1/2)^4}(I_{2n} - I_n).$$

可见后验误差估计旨在将误差 $\delta f = f_n - f$ 表示为如下形式:

$$|\delta f| \leqslant C|f_n - f_{2n}|,$$

其中 C 为与 n 无关的常数.

后验误差分析在适应性方法中发挥着重要的作用. 实际上, 通过变化离散变量(例如空间步长), 后验误差分析可以使得误差不超过某个固定的上限. 这种应用适应性方式控制误差大小的方法称为适应性数值方法. 误差是否超过某个固定的上限可以看作一种收敛测试. 在实际计算中, 先选择一个数值较大的离散变量, 并计算相应的数值解. 若数值解通过了收敛测试, 则此数值解为所需要的解; 如若不然, 按照一定的方式缩小离散变量, 再重新计算数值解. 重复以上过程, 直到得到的数值解通过收敛测试为止.

1.3　避免和减小误差的若干原则

根据数值计算的经验发现, 舍入误差是不可避免的, 而精确的误差估计通常是不可能的. 所以在计算过程中要尽可能避免误差的危害, 防止有效数字的损失. 通过长期对误差产生原因的分析及误差传播规律的分析, 人们总结出了以下若干计算原则.

1. 避免两个相近数相减

例 1.3.1　设 $x=1000$, 计算

$$y=\sqrt{x+1}-\sqrt{x}.$$

y 的准确值为 0.015807437428958. 假设在计算中取 4 位有效数字计算, 则 $\sqrt{x+1}\approx 31.64$, $\sqrt{x}\approx 31.62$, 故 $y\approx y^*=0.02$. 此时, 结果只剩 1 位有效数字, 相对误差为 $E_r(y^*)=\dfrac{|y-y^*|}{|y^*|}=\dfrac{|0.015807437428958-0.02|}{0.02}=20.96\%$. 可见相对误差超过了 20%, 严重影响了计算精度.

为了解释产生严重误差的原因, 下面给出相对误差的一个估计:

$$\begin{aligned}E_r(y^*)&=\frac{|(x_1-x_2)-(x_1^*-x_2^*)|}{|x_1^*-x_2^*|}=\frac{|(x_1-x_1^*)-(x_2-x_2^*)|}{|x_1^*-x_2^*|}\\&\leqslant\frac{|x_1-x_1^*|+|x_2-x_2^*|}{|x_1^*-x_2^*|}\leqslant\frac{\delta(x_1^*)+\delta(x_2^*)}{|x_1^*-x_2^*|}\leqslant\frac{\delta_r(x_1^*)|x_1^*|+\delta_r(x_2^*)|x_2^*|}{|x_1^*-x_2^*|}.\end{aligned}$$

容易发现, 当两个相近的数相减时, $|x_1^*|/|x_1^*-x_2^*|\gg 1, |x_2^*|/|x_1^*-x_2^*|\gg 1$. 可见两数相减后的相对误差被放大了大约 $|x_1^*|/|x_1^*-x_2^*|$ 倍.

对例 1.3.1 改进的算法如下:

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

改进后的算法和原算法在数学上是等价的, 但计算的结果却不一样. 例如 $y^{**}=\dfrac{1}{x_1^*+x_2^*}\approx 0.01581$. 通过与准确值的对比, 发现 y^{**} 有 4 位有效数字. 对改进后算法的相对误差的分析有

$$E_r(y^{**})=\frac{\left|\dfrac{1}{x_1^*+x_2^*}-\dfrac{1}{x_1+x_2}\right|}{\left|\dfrac{1}{x_1^*+x_2^*}\right|}=\left|1-\frac{x_1^*+x_2^*}{x_1+x_2}\right|=\frac{|(x_1-x_1^*)+(x_2-x_2^*)|}{|x_1+x_2|}$$
$$\leqslant\frac{|x_1-x_1^*|+|x_2-x_2^*|}{|x_1+x_2|}\leqslant\frac{\delta(x_1^*)+\delta(x_2^*)}{|x_1+x_2|}\leqslant\frac{\delta_r(x_1^*)|x_1^*|+\delta_r(x_2^*)|x_2^*|}{|x_1+x_2|}.$$

可见, 改进后算法不会放大相对误差.

2. 避免用绝对值过小的数作除数

例 1.3.2 计算

$$z=\frac{x}{y}.$$

设 $x=2.7182, y=0.001$, 则准确解为 2718.2. 若分母变为 $y^*=0.0011$, 分子不变, 即 $x^*=x$. 则

$$z^*=\frac{x^*}{y^*}=\frac{2.7182}{0.0011}\approx 2471.1.$$

绝对误差为 $E(z^*)=247.1$, 相对误差为 $E_r(z^*)=10\%$. 可见误差变化巨大.

对例 1.3.2 进行绝对误差分析有

$$E(z^*)=\left|\frac{x^*}{y^*}-\frac{x}{y}\right|=\frac{1}{|y|}\left|x-\frac{x^*y}{y^*}\right|=\frac{1}{|y|}\left|x-x^*+x^*-\frac{x^*y}{y^*}\right|$$
$$\leqslant\frac{1}{|y|}\left(\delta(x^*)+|x^*|\frac{\delta(y^*)}{|y^*|}\right).$$

进而有相对误差估计为

$$E_r(z^*)\leqslant\frac{|y^*|}{|y|}(\delta_r(x^*)+\delta_r(y^*)).$$

从以上估计可以看到, 除数很小, 造成了绝对误差被放大. 从相对误差的角度也不难看出, 分母 y^* 的相对误差变化很大, 造成了 $\dfrac{x^*}{y^*}$ 的相对误差变大.

3. 要防止大数“吃掉”小数

大数“吃掉”小数是由计算机数位限制造成的. 例如计算机中最广泛使用的浮点数运算标准是 IEEE754—1985(IEEE Standard for Floating-Point Arithmetic). 其中双精度浮点数定义方式如下:

$$x=s\cdot f\cdot 2^e,$$

其中

- s 表示符号 (sign)：0 代表正数, 1 代表负数.
- f 表示分数 (fraction)：

$$f = 2^{-1} + f_2 2^{-2} + \cdots + f_{53} 2^{-53}, \quad f_i \in \{0,1\}, \quad i \in \{2, \cdots, 53\}.$$

- e 表示指数 (exponent)：

$$e = e_0 2^0 + e_1 2^1 + \cdots + e_{10} 2^{10} - 1022, \quad e_i \in \{0,1\}, \quad i \in \{0, \cdots, 10\}.$$

IEEE754—1985 标准中规定 e 的取值范围是 $e \in [-1022, 1025]$, 其中 $e = -1022$ 时表示 0, $e = 1025$ 时表示无穷大. 双精度浮点数的存储方式如图 1.2 所示.

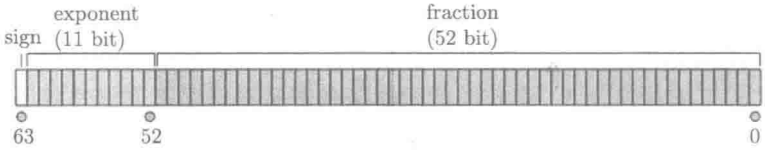

图 1.2　IEEE754—1985 标准中 64bits 存储格式

记由上述方式定义的集合为 $\mathbb{F}$, 则集合 $\mathbb{F}$ 包含有限个数字, 其中最大值为

$$x_{\max} = (2^{-1} + 2^{-2} + \cdots + 2^{-53}) \times 2^{1024} \approx 1.79769 \times 10^{308},$$

最小的正数为

$$x_{\text{posmin}} = 2^{-1} \times 2^{-1021} \approx 2.22507 \times 10^{-308}.$$

设 x 满足 $x_{\text{posmin}} < |x| < x_{\max}$, 则舍入方式为

$$rd(x) = \text{sign}(x) \begin{cases} (0.1 f_2 \cdots f_{53})_2 \cdot 2^e, & f_{54} = 0, \\ ((0.1 f_2 \cdots f_{53})_2 + 2^{-53}) \cdot 2^e, & f_{54} = 1. \end{cases}$$

当两个数做加法运算时, 首先要对齐指数位, 然后再进行分数位的加法. 根据这个规则, 计算 $1+10^{-16}$ 时, 对齐指数位有 $1 = (0.1)_2 \times 2^1, 10^{-16} = (2^{-55} + \cdots) \times 2^1$, 再做分数相加有 $1+10^{-16} = ((0.1)_2 + 2^{-55} + \cdots) \times 2^1$. 经过舍入后得到 $rd(1+10^{-16}) = 1$, 即舍入算法使大数“吃掉”了小数. 如果仅仅是计算 $1 + 10^{-16}$ 则影响不大. 但从下面的例子中, 可见大数“吃掉”小数会对最终的结果产生巨大的误差.

例 1.3.3　*求解一元二次方程 $x^2 - (1+10^{-16})x + 10^{-16} = 0$ 的根. 易知方程的两个实根为 $x_1 = 1, x_2 = 10^{-16}$. 若用求根公式计算有*

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

由于 $rd(1+10^{-16})=1$, $rd(\sqrt{(1+10^{-16})^2-4\times 10^{-16}})=1$, 从而得到 $x_1=1$, $x_2=0$. 显然 x_2 的结果是错误的. 要防止出现这样的错误, 应将计算 x_2 的公式改为

$$x_2=\frac{10^{-16}}{x_1}=10^{-16}.$$

从而得到正确的结果.

4. 简化计算步骤, 减少重复运算

例 1.3.4 用乘法计算 x^{255}, 有以下两种方法.

解 **方法一** 直接计算;

方法二 将原式改写为

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

首先, 分析两种方法的舍入误差. 假设乘法的舍入误差可以表示为如下形式:

$$rd(ab)=ab(1+\delta).$$

对以上两种方法分别做舍入误差分析有

$$rd(x^{255})=x^{255}\prod_{i=1}^{254}(1+\delta_i),$$

$$rd(x\cdot x^2\cdot x^4\cdot x^8\cdot x^{16}\cdot x^{32}\cdot x^{64}\cdot x^{128})=x^{255}\prod_{i=1}^{254}(1+\tilde{\delta}_i).$$

可以看到, 两种方法的舍入误差具有相似的形式. 若取 $x=\sqrt{2}$, 两种方法的相对误差分别为 1.74×10^{-14} 和 2.84×10^{-14}. 可见两种方法的相对误差在同一数量级.

其次, 从计算成本的角度考虑. 方法一共需要 254 次乘法运算. 而在方法二中除 x 外, 每个因子计算需要 1 次乘法运算, 共 7 次乘法运算, 接着连乘计算 7 次, 总共 14 次乘法运算. 所以方法二相对于方法一大大减少了计算成本.

从例 1.3.4 可以看出, 尽管两种方法在舍入误差上没有太大区别, 但是方法二在计算成本上有着巨大的优势. 所以, 在计算中应该尽量化简计算步骤, 以达到最小的计算成本的目的.

这里还要特别介绍多项式计算中非常重要的一种方法: Horner 方法, 也称为嵌套乘法. 考虑一个标准的多项式

$$a_0+a_1x+a_2x^2+a_3x^3+a_4x^4,$$

将其写成以下嵌套形式

$$a_0+x(a_1+x(a_2+x(a_3+xa_4))).$$