面向21世纪课程教材
Textbook Series for 21st Century

实用数值计算方法

电子科技大学应用数学系 编

高等教育出版社
HIGHER EDUCATION PRESS

内容提要

本书是教育部“高等教育面向21世纪教学内容和课程体系改革计划”的研究成果.是面向21世纪课程教材和教育部工科数学学科“九五”规划教材.本书介绍了用计算机求解数学问题的常用方法和算法,按照“低学时、重实际、模块化”的要求组织教材内容,介绍了一些新的方法或算法;联系数学模型,结合使用MATLAB软件,介绍算法的构造和使用;计算实例取材丰富;每章配有习题和实验题;对一些离散优化问题的算法也作了适当介绍.本书可作高等工科院校的教材或参考书,也可供工程技术人员及自学者参考.

图书在版编目(CIP)数据

实用数值计算方法/电子科技大学应用数学系编.
—北京:高等教育出版社,2001

ISBN 7-04-008830-4

Ⅰ.实… Ⅱ.电… Ⅲ.数值计算-计算方法
Ⅳ.0241

中国版本图书馆CIP数据核字(2000)第62109号

实用数值计算方法
电子科技大学应用数学系 编

出版发行	高等教育出版社		
社　　址	北京市东城区沙滩后街55号	邮政编码	100009
电　　话	010-64054588	传　　真	010-64014048
网　　址	http://www.hep.edu.cn		
经　　销	新华书店北京发行所		
印　　刷	北京外文印刷厂		
开　　本	787×960　1/16	版　　次	2001年1月第1版
印　　张	12	印　　次	2001年1月第1次印刷
字　　数	210 000	定　　价	10.70元

本书如有缺页、倒页、脱页等质量问题,请到所购图书销售部门联系调换。

前　言

面向21世纪，现代社会正经历着由工业经济向知识经济的转变，以计算机为主体的信息技术逐渐改变人们的生活方式、思维方式和工作方式.计算机最明显的功能就是高速度地进行大量的数值计算，因此，科学计算现已与理论研究、科学实验并列为科学研究的三大支柱.

精确定量思维需要人们从实际问题中提炼数学问题，抽象化为数学模型，用科学计算求出此模型的解或近似解，然后回到现实中进行检验，必要时还要修改模型使之更加切合实际，最后编制解决问题的软件包，以便得到更广泛的应用.在科学计算过程中需要设计算法在计算机上实现，我们用下面流程图说明：

实际问题 → 数学模型 → 算法选择 → 数据结果 → 结果分析与检验

计算方法是研究数学模型求数值解的算法与理论的一门学科，它随计算工具的发展而发展，在计算机技术飞速发展的今天，计算方法的研究对象就是用计算机进行数值计算的算法与理论.

本教材是根据教育部“九五”重点立项课题《面向21世纪工科数学教学内容和课程体系改革的研究与实践》的要求与原国家教委批准印发的《高等工业学校计算方法教学基本要求》编写的一本改革教材，与传统教材相比，有以下几个特点：

一、着重介绍方法，适当淡化理论.由于本书的使用对象是工科大学生，计算方法课程学时较少，因此，本书侧重实用性，主要介绍数值计算的各种方法(或算法)，力求揭示方法的实质，使学生掌握方法的基本思想，而不追求理论的系统性和完整性.

二、介绍了一些新的方法或算法.传统的计算方法课程内容，主要介绍将连续问题转化为离散问题，并求数值解的方法.在当今计算机技术迅速发展的时代，更多的离散问题被提出，在大量的信息处理问题中，非数值型算法越来越重要.因此，本书介绍了最近十几年科技人员使用较多的方法，如分形算法、神经网络算法、Bezier曲线方法、蒙特卡罗算法等.此外在附录1中还专门介绍了解决离散问题的算法，如穷举法、聚类算法、贪婪算法等.

三、与实际问题相结合，激发学生学习兴趣.本书力图从一些实际问题引出常用方法和算法，引导学生去解决一些数学模型问题，使学生看到数学应用的价值；同时，这些富有趣味的实际问题，能吸引学生思考，启迪学生思维，开阔学生

眼界,激发学生学习兴趣.

四、与实现算法的数学软件相结合,提高编程效率.使用计算机进行数值计算就要根据算法编写程序.早在70年代后期,专家们就利用已有的算法开发了计算机语言的标准程序库.在这些程序库的基础上,80年代中期开始,世界上一些软件公司相继推出了一些功能十分强大的数学软件,如MATLAB、MAPLE、MATHEMATICA等,它们实际上是具有一定数据处理能力与计算功能的集成系统.这些数学软件为我们提供了高效的编程环境.因此,本书在附录2中针对各章内容介绍了MATLAB软件的相关语句.

五、加强了上机实验教学环节.由于解决实际问题时需要综合应用各种算法并与建立数学模型相结合,进行数学实验.因此,本书每章后面编写了相应的实验课题以供选用.

本书是在我校使用的《数值计算方法》讲义基础上修改而成,根据"低学时、重实际、模块化"的要求增删了一些内容.全书共分八章,内容包括引论、解线性方程组的直接法、插值方法、数据拟合方法、数值积分方法、常微分方程数值解、非线性方程求根方法以及解线性方程组的迭代法.每章配有上机实验题与习题.

使用本书约需34学时,其中讲课22学时,上机12学时.带*号的内容可根据学时多少酌情取舍.本书可作为高等工科院校数值计算方法课教材或参考书,也可供工程技术人员及自学者参考.

本书由钟尔杰、谢云荪编写,由西安交通大学葛仁杰教授主审.参加审稿的还有上海交通大学向隆万教授,大连理工大学施光燕教授、施吉林教授,同济大学骆承钦教授,电子科技大学赵善中教授,清华大学白峰杉教授,合肥工业大学苏化明教授,西北工业大学叶正麟教授.他们认真审阅了原稿,并提出不少宝贵意见,我们对此表示衷心的感谢!

本书编写过程中,得到全国工科数学课程教学指导委员会主任、西安交通大学马知恩教授,高等教育出版社文小西编审,电子科技大学校长助理、教务处长兰家隆教授,应用数学系系主任钟守铭教授的大力支持.我们借此机会向他们表示衷心的感谢!

编写实用的具有新意的数值计算方法改革教材需要不断地探索.我们虽经努力,但由于缺乏经验,水平有限,疏漏之处,恳请同行专家及读者指正.

编 者

1999年12月

第一章　引　　论

数值计算方法(或计算方法)是研究数学问题求数值解的算法和有关理论的一门学科,它的理论与方法随计算工具的发展而发展.在古代,人类研究的数学问题几乎总与计算有关,而计算工具的简陋,使求解问题受到很大限制.现代科学技术日新月异,尤其是计算机技术飞速发展,人类可以用计算机做复杂的数值计算、数据处理(包括图形,图象,声音,文字)以及数据传输(通过计算机局域网或国际互联网),计算机不仅是现代计算工具,而且已成了我们工作环境的一部分.计算方法所研究的内容是用计算机解决数学问题的方法(包括算法)和理论.用计算机进行数值计算的核心是算法设计.算法是对解题方案的准确描述,是"解题的操作序列".在研究算法时,离不开误差分析,同时还需要考虑算法的稳定性、收敛性、计算工作量等问题.

§1.1　算 法 简 介

纵观数学科学的发展,我们可以发现,数学本身可分为两大体系:演绎体系,算法体系.历史上的阿基米德、牛顿都是研究算法很有成效的数学家.计算方法所研究的算法主要是数值型算法(与非数值型算法区别),这类算法是对某些给定的数据,按照一定的运算秩序进行运算的运算序列.

算法的概念源于数学,计算机科学的发展使它的内涵有了很大变化.为了解决某一类特定的问题,算法应给出一系列操作,而且每一操作都能在计算机的环境中被执行;算法应包含有限次运算;一个算法可以有多个输入量,即问题的初始数据;算法实现后,它应该有一个或多个输出量,这是算法对初始数据运算的结果,即问题的解答.

一、算法的结构

为了能设计出优良的算法,同时方便交流和阅读,计算机专家们提出了算法结构的思想.算法的基本结构被分为三种:即**顺序结构**、**分支结构**和**循环结构**.三种基本结构可以组合成其它各种各样结构的算法.下面分别举例说明.

例 1.1　顺序结构(又称简单结构)算法.

已知三角形$\triangle ABC$三条边的长度:a、b、c.用海伦公式计算三角形面积S.

算法 1.1

第一步：输入三角形三条边的长度：a、b、c；

第二步：计算三角形的半周长：

$$p = (a + b + c)/2;$$

第三步：计算三角形面积：

$$S = \sqrt{p(p-a)(p-b)(p-c)};$$

第四步：输出计算结果 S，结束.

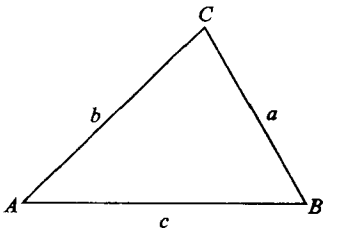

图 1-1

例 1.2　分支结构算法.

对给定的一元二次方程：$ax^2 + bx + c = 0$，用求根公式求根.

算法 1.2

第一步：输入二次项、一次项和常数项的系数：a、b、c；

第二步：计算判别式的值：$p = b^2 - 4ac$；

第三步：判断：如果 $p \geqslant 0$，则 $x_1 = (-b - \sqrt{p})/2a$，$x_2 = (-b + \sqrt{p})/2a$；

否则，$x_1 = (-b - \mathrm{i}\sqrt{-p})/2a$，$x_2 = (-b + \mathrm{i}\sqrt{-p})/2a$；

第四步：输出方程的两个根 x_1、x_2，结束.

例 1.3　循环结构算法.

裴波纳奇数列$\{F_n\}$的通项满足递推公式：$F_n = F_{n-1} + F_{n-2}$（$F_1 = 1$，$F_2 = 1$）.求裴波纳奇数列的前十项.

算法 1.3

第一步：置初值 $k = 2$，$F_1 = 1$，$F_2 = 1$；

第二步：$k \leftarrow k + 1$，$F_k = F_{k-1} + F_{k-2}$；

第三步：判断：若 $k < 10$ 则转第二步，否则输出 F_k（$k = 1, 2, \cdots, 10$），结束.

第二步中“$\leftarrow$”表示赋值，它的作用是将其右边表达式的值赋值给左边的变量.

由上面的例子可知，顺序结构算法按顺序执行每一步操作；分支结构算法带有逻辑判断条件，当条件成立则执行某些操作，条件不成立则执行另一些操作；而循环结构算法带控制条件，当条件成立时重复执行循环体内的操作，条件不成立时退出循环体.

> 一个解决实际问题的算法往往是由计算机可执行的基本操作（加、减、乘、除、判断、置数以及调用内部函数等）按照顺序、分支、循环这三种基本结构组成.

二、算法的描述方式

算法可以用解题操作步骤来描述，也可以用计算机程序来表示．而人们在解决实际问题时，常用算法框图来描述算法．算法框图为大多数人所使用，主要原因是这种方式直观、方便、便于交流，特别是在设计算法时能较好地考虑算法执行时的动态性．

例 1.4 设计算法用以计算一个 n 次多项式

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

在自变量 x 处的值．

先考虑一种比较直观的累加算法，即从低次项开始逐步累加各项．

记 $S_0 = a_0, S_k = S_{k-1} + a_kx^k (k = 1,2,\cdots,n)$，$S_n$ 即所求．在设计算法时，为了节约计算量，引入一个工作变量 u，用以保存自变量的方幂 $x^k (k = 1,2,\cdots,n)$．算法框图见图 1－2．所用乘法次数为 $2n$．

现考虑另一种著名的秦九韶方法，以 4 次多项式为例说明算法思想．

$$\begin{aligned} P_4(x) &= a_0 + a_1x + a_2x^2 + a_3x^3 + a_4x^4 \\ &= a_0 + x(a_1 + x(a_2 + x(a_3 + xa_4))) \end{aligned}$$

上式右端有三层括号，规定内层括号的计算优先．即最先计算 $a_3 + xa_4$ 的值．求一个四次多项式只需四次乘法．同理，求 n 次多项式只需 n 次乘法．记 $S_n = a_n$，$S_{k-1} = a_{k-1} + xS_k, (k = n, n-1,\cdots,1)$，$S_0$ 即所求函数值算法框图见图 1－3．

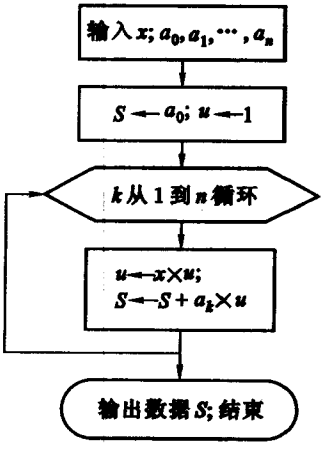

图 1－2 多项式累加算法

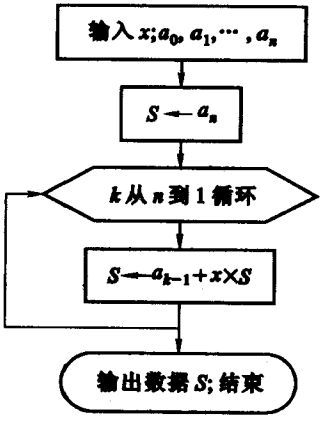

图 1－3 秦九韶算法

也可以按解题步骤方式，描述如下：

算法 1.4(秦九韶算法)

第一步：输入多项式的系数 $a_0, a_1, \cdots, a_n$ 及自变量的值 x；

第二步：置 $k \leftarrow n$；$S \leftarrow a_n$；

第三步：计算 $S \leftarrow a_{k-1} + xS$；

第四步：判断：若 $k > 1$，则置 $k \leftarrow k-1$，转第三步；否则，输出 S，结束.

显然，秦九韶算法结构简单，所用乘法次数少，优于前一种算法.

§1.2 误差与有效数字

算法的实现必须在计算机上进行，计算机并不是像一般人想像哪样可以解决一切问题而不出差错. 半个世纪以来计算机带给我们这个世界的诸多烦恼中，误差问题最为突出. 小到银行利率的错算，大到导弹的错误发射，除了操作人员的疏忽、机器的故障引起过失误差外，计算机在处理数据过程中还存在计算误差. 这主要是计算机机器数所引起的，机器数系的特点是**有限、离散**，且具有一定分布规律；这和数学上常用的实数系**无限、稠密、连续**的特点完全不同. 在科学计算中实数通常被表示成如下形式

$$\pm 0.a_1a_2\cdots a_n \times 10^m$$

这就是实数的十进制浮点数表示. 其中，$a_1 \neq 0$ 且 $a_1, a_2, \cdots, a_n$ 都是 0～9 中的任一整数，$0.a_1a_2\cdots a_n$ 称为尾数. 10^m 用于确定小数点位置，指数 m 称为阶码. 因此，一个浮点数包含尾数和阶码两部分.

在实数的二进制表示中，每位数不是 0 就是 1，因而用两种不同的物理信息就可以表示出来. 所以计算机机器数采用二进制浮点数表示，即

$$\pm 0.b_1b_2\cdots b_l \times 2^t$$

其中，$b_1, \cdots, b_l$ 取 0 或 1，且 $b_1 = 1$. 二进制浮点数和十进制的情形一样，也是由尾数和阶码两个部分组成，$0.b_1b_2\cdots b_l$ 是尾数，t 是阶码. 计算机的特点决定了尾数的位数 l 是有限正整数，阶码 t 是有界的整数. 所以，机器数中有最大的数，也有最小的数. 由于机器数的位数有限，用机器数表示实数时，很多情况下都带有误差.

在 2400 多年前，古希腊人提出了被称为几何三大问题的古典难题. 这说明在历史上，人类就常被**误差**所困扰. 下面问题就是三大难题之一.

例 1.5 立方倍积问题. 做一个立方体，使其体积为已知立方体的二倍.

解 不妨设已知立方体体积为 1. 要做的立方体体积为 2，则所做立方体高度应该为 $h = \sqrt[3]{2}$. 用计算机计算出 $\sqrt[3]{2} \approx 1.259\ 921\ 049\ 894\ 87$(15 位数). 尽管精确度相当高，但仍是近似值. 下面的表 1－1 列出了对 h 取前有限位数时，计算所得体积的误差.

表 1-1 立方倍积问题的计算

位 数	高 度	体 积	误 差
2	1.2	1.728	$2.720\,0\times10^{-1}$
3	1.25	1.953 125	$4.687\,5\times10^{-2}$
4	1.259	1.995 616 979	$4.383\,0\times10^{-3}$
5	1.259 9	1.999 899 757 799	$1.002\,4\times10^{-4}$
6	1.259 92	1.999 995 000 191 49	$4.999\,8\times10^{-6}$
7	1.259 921	1.999 999 762 390 49	$2.376\,1\times10^{-7}$
8	1.259 921 0	1.999 999 762 390 49	$2.376\,1\times10^{-7}$
9	1.259 921 04	1.999 999 952 878 60	$4.712\,1\times10^{-8}$

由表 1-1 可知,计算机机器数的有限位特点使这一问题只能在满足一定的精度条件下解决.误差是无法消除的.

一、误差的分类

误差是表示某个量值的精确数与近似数之间差异的度量.在科学计算中误差来源一般有以下四个方面:

(1) 模型误差:由实际问题抽象、简化为数学问题(建立数学模型)时所引起的误差;

(2) 观测误差:测量工具的限制或在数据的获取时随机因素所引起的物理量的误差;

(3) 截断误差:又称方法误差.用数值法求解数学模型时,得到的正确解和模型准确解之间的误差,通常是用有限过程代替无限过程所引起;

(4) 舍入误差:由于计算机所表示的数的位数有限,通常用四舍五入的办法取近似值,由此引起的误差.

根据误差来源,可以将误差分为两类:一类是固有误差,包括模型误差和观测误差;另一类是在计算过程中引起的计算误差,包括截断误差和舍入误差.计算方法所考虑的主要是计算误差.

有时,带有误差的数据也被人们频繁使用.例如,在某次人口普查中,经统计我国某省的人口数为 7 123 万,这就是一个近似数,其舍入误差不超过 0.5 万.

用 3.141 592 6 来代替圆周率 π,其舍入误差为 $R=\pi-3.141\,592\,6$.

在计算收敛的无穷级数时,取有限项代替无限项.如对于正弦函数

$$\sin x = x - \frac{1}{3!}x^3 + \frac{1}{5!}x^5 - \frac{1}{7!}x^7 + \cdots,$$

取 $S=x-\frac{1}{3!}x^3+\frac{1}{5!}x^5$，作近似计算，其截断误差为 $R=\sin x-S$.

二、绝对误差与相对误差

假设某一数据的准确值为 x^*，其近似值为 x，称 $E(x)=x^*-x$ 为 x 的绝对误差(简称误差). $|E(x)|$ 的大小标志着 x 的精确程度. $|E(x)|$ 越小说明 x 的精确程度越高. 由于准确值一般无法得到，从而误差也无法求得. 在实际测量计算时，可以根据具体情况估计出 $|E(x)|$ 的大小范围，即指定一个适当小的正数 ε，使得，

$$|E(x)|=|x^*-x|\leqslant\varepsilon$$

ε 称为绝对误差限. 显然

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

故通常用 $x^*=x\pm\varepsilon$ 表示近似值的精确度(准确值的所在范围).

在实际问题中，判断一个近似值的精确度高低不仅要观察绝对误差大小，还应考虑数据本身的大小. 所以人们也常用相对误差来衡量一个数据的精确程度. 相对误差定义为绝对误差与准确值之比，即

$$E_r(x)=\frac{E(x)}{x^*}=\frac{x^*-x}{x^*}\qquad(x^*\neq 0)$$

与绝对误差的情形类似，人们只能估计相对误差的范围，通常指定一个适当小的正数 η，表示相对误差限，即

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

在处理具体问题时，由于准确值不能求得，常用的不等式为：

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

例1.6 月球到地球的平均距离 $d_1=384\ 401\pm 0.5(\mathrm{km})$；由某一公共汽车站到下一站的距离 $d_2=1.4\pm 0.4(\mathrm{km})$. 第一个数据的绝对误差限是 $0.5(\mathrm{km})$，第二个数据的绝对误差限为 $0.4(\mathrm{km})$. 根据绝对误差概念，第一个数据精确度较差；但由于

$$|E_r(d_1)|\leqslant\left|\frac{0.5}{384\ 401}\right|\leqslant\frac{1}{100\ 000},\quad |E_r(d_2)|\leqslant\left|\frac{0.4}{1.4}\right|\leqslant\frac{4}{100}$$

根据相对误差概念，第一个数据精确度较高. 这比较合理，与人们的常识一致.

注意：绝对误差是有量纲数，而相对误差是无量纲数.

三、有效数字与机器数精度

当某一个数据有很多位数时，人们常常按照“四舍五入”的原则取它的前几位数学作为近似值. 由此抽象出有效数字的概念.

例 1.7 我们知道 $\pi=3.141\,592\,6\cdots$ 是一个无理数，由于无理数不能用有限位数表示，考虑 π 的不同近似值：

取一位数：$x_1=3$，　　有　$|\pi-x_1|\leqslant 0.5=\frac{1}{2}$；

取三位数：$x_3=3.14$，　　有　$|\pi-x_3|\leqslant 0.005=\frac{1}{2}\times 10^{-2}$；

取五位数：$x_5=3.141\,6$　　有　$|\pi-x_5|\leqslant 0.000\,05=\frac{1}{2}\times 10^{-4}$.

分析上面近似值的误差，它们都没有超过末位数的半个单位，由此引出下面定义.

定义 1.1 设 x^* 的近似数为 $x=\pm 0.a_1a_2\cdots a_n\times 10^m$，其中，$a_1,a_2,\cdots,a_n$ 都是 0～9 中的任一整数，且 $a_1\neq 0$. 若 x 的绝对误差限满足：

$$E(x)=|x-x^*|\leqslant\frac{1}{2}\times 10^{m-n}$$

则称近似数 x 具有 n 位有效数字.

> 若近似值 x 的绝对误差限是某一位上的半个单位，该位到 x 的第一位非零数字一共有 n 位，则称近似值 x 有 n 位有效数字.

上面的例子中，3.14 具有 3 位有效数字，3.141 592 6 具有 8 位有效数字.

由定义 1.1 可知，如果用浮点数 x 表示某个数据的近似数，当阶码 m 一定时，x 的有效数字位数越多，绝对误差越小.

由定义 1.1 还可以知道，对于有效数字位数为 n 的近似数

$$x=\pm 0.a_1a_2\cdots a_n\times 10^m$$

其相对误差满足：

$$|E_r(x)|=\left|\frac{x-x^*}{x^*}\right|\leqslant\frac{1}{2a_1}\times 10^{-n+1}\leqslant 5\times 10^{-n}$$

这说明 x 的有效数字位数越多，相对误差越小.

对于二进制数系统，舍入规则与“四舍五入”相仿. 设 x^* 是一个 $l+1$ 位的二进制数

$$x^*=0.b_1b_2\cdots b_lb_{l+1}\times 2^t$$

其中，$b_1,\cdots,b_l,b_{l+1}$ 取 0 或 1，且 $b_1=1$. 它的 l 位近似数舍入规则为

$$x = \begin{cases} 0.b_1b_2\cdots b_l \times 2^t, & b_{l+1} = 0 \\ (0.b_1b_2\cdots b_l + 2^{-l}) \times 2^t, & b_{l+1} = 1 \end{cases}$$

由于 x 的相对误差上限可表示为

$$|E_r(x)| = \left|\frac{x - x^*}{x^*}\right| \leqslant 2^{-l}$$

所以，在有效数字为 l 位的机器数系统中，我们称机器数精度为

$$\text{eps} = 2^{-l}$$

尽管机器数采用二进制浮点数表示，但人们在使用计算机进行数值计算时仍然和十进制浮点数打交道. 由于每一个二进制数对应于一个十进制数，机器数的位数有限，从而用计算机表示十进制数的位数也有限. 一台计算机可以用多少位有效数字表示实数，这一问题与计算机硬件和软件环境有密切关系. 例如，一台 586 微机在数学软件 MATLAB 环境下，用 15 位有效数字表示实数；在 FORTRAN77 语言环境下用 8 位或 16 位(分单精度或双精度)有效数字表示实数.

§1.3 递推算法的数值稳定性问题

递推算法是解决实际问题时使用相当普遍的一种算法，它的数学描述是带初值的递推关系式.

例 1.8 小猴吃桃问题. 有一天小猴摘下了若干个桃子，当即吃掉了一半，还觉得不过瘾，又多吃了一个. 第二天接着吃了剩下的一半，又多吃了一个. 以后每天都是吃掉尚存的桃子的一半零一个. 到第十天早上，小猴准备吃桃子时，看到只剩下 1 个桃子了. 问小猴第一天共摘下了多少个桃子?

解 设第 k 天的桃子数为 p_k，则桃子数目变化规律为

$$p_k = \frac{1}{2}p_{k-1} - 1$$

这是正向递推的关系式，解之，可得逆向递推关系式

$$p_{k-1} = 2(p_k + 1) \quad (k = 10, 9, \cdots, 2)$$

由初值 $p_{10} = 1$，根据上式设计循环算法计算出 $p_1 = 1\,534$. 即第一天的桃子数为 1 534.

类似上例的很多问题仅涉及整数序列递推，可以根据初值条件选择正向递推公式或者逆向递推公式用以解决实际问题. 尽管正向递推和逆向递推公式在数学上完全等价，却导致两种不同的算法. 对于实数序列的递推，由于初始误差的存在，可能一种方向的递推会使误差逐步增大，而另一方向的递推会使得误差逐步减小.

例 1.9 利用递推式计算定积分 $I_n = e^{-1}\int_0^1 x^n e^x dx \quad (n = 0,1,2,\cdots,20)$ 的值.

$$I_0 = e^{-1}\int_0^1 e^x dx = e^{-1}(e-1) = 1 - e^{-1}$$

$$I_n = e^{-1}\int_0^1 x^n e^x dx = e^{-1}\left(x^n e^x\Big|_0^1 - n\int_0^1 x^{n-1}e^x dx\right) = 1 - nI_{n-1}$$

由此可得带初值的递推关系式

$$\begin{cases} I_0 = 1 - e^{-1} \\ I_n = 1 - nI_{n-1} \quad (n = 1,2,\cdots) \end{cases}$$

直接用 $I_0 = 1 - e^{-1}$ 在 15 位有效数的计算机上计算得

$$I_0 \approx 0.632\ 120\ 558\ 828\ 56.$$

利用递推式逐次计算可得 20 个数据如下:(表 1-2)

表 1-2

S_1	0.367 879 441 171 44	S_{11}	0.077 352 229 358 78
S_2	0.264 241 117 657 12	S_{12}	0.071 773 247 694 64
S_3	0.207 276 647 028 65	S_{13}	0.066 947 779 969 72
S_4	0.170 893 411 885 38	S_{14}	0.062 731 080 423 87
S_5	0.145 532 940 573 08	S_{15}	0.059 033 793 641 90
S_6	0.126 802 356 561 52	S_{16}	0.055 459 301 729 57
S_7	0.112 383 504 069 36	S_{17}	0.057 191 870 597 31
S_8	0.100 931 967 445 09	S_{18}	-0.029 453 670 751 54
S_9	0.091 612 292 994 17	S_{19}	1.559 619 744 279 19
S_{10}	0.083 877 070 058 29	S_{20}	-30.192 394 885 583 78

表中 $S_1,\cdots,S_{20}$ 是积分值 $I_1,\cdots,I_{20}$ 的计算近似值. 由于被积函数非负,对正整数 n,有 $I_n \geqslant 0$ 成立. 但是表 1-2 中有两个负数 S_{18} 和 S_{20},这显然是错误的. 导致这一错误的直接原因是初始数据的误差,正向递推公式使得误差在计算过程中增大.

考虑另一种算法,由递推公式 $I_n = 1 - nI_{n-1}$ 解得 $I_{n-1} = \frac{1}{n}(1 - I_n)$,这是逆向递推公式. 对 I_n 的值作估计,有

$$I_n = e^{-1}\int_0^1 x^n e^x dx \leqslant e^{-1}e^1\int_0^1 x^n dx = \frac{1}{n+1}$$

取 $I_{30} \approx S_{30} = \frac{1}{31}$,利用递推公式

$$S_{n-1} = \frac{1}{n}(1 - S_n),\ n = 30,29,28,\cdots,2$$

计算出 $S_{29}, S_{28}, \cdots, S_1$，并选取前 20 个数据，列表如下：(表 1－3)

表 1－3

S_1	0.367 879 441 171 44	S_{11}	0.077 352 228 862 66
S_2	0.264 241 117 657 12	S_{12}	0.071 773 253 648 03
S_3	0.207 276 647 028 65	S_{13}	0.066 947 702 575 62
S_4	0.170 893 411 885 38	S_{14}	0.062 732 163 941 38
S_5	0.145 532 940 573 08	S_{15}	0.059 017 540 879 30
S_6	0.126 802 356 561 53	S_{16}	0.055 719 345 931 24
S_7	0.112 383 504 069 30	S_{17}	0.052 771 119 168 99
S_8	0.100 931 967 445 59	S_{18}	0.050 119 854 958 09
S_9	0.091 612 292 989 66	S_{19}	0.047 722 755 796 21
S_{10}	0.083 877 070 103 39	S_{20}	0.045 544 884 075 82

表 1－3 中没有出现负数，数据变化表明实数序列 $\{S_n\}$ 随 n 的增加单调下降趋于零，这与积分值数列 $\{I_n\}$ 的变化规律相吻合.

前一种算法用正向递推，尽管初值的近似值精确度很高，但是在计算过程中造成了误差扩散，这是数值不稳定的算法.

> 在算法执行过程中，舍入误差对计算结果影响大的一类算法被称为数值不稳定的算法.

后一种算法用逆向递推，虽然初始数据 S_{30} 带有明显误差，但计算结果却比较可靠. 由误差传播规律

$$E_{n-1} = I_{n-1} - S_{n-1} = \frac{1}{n}(1 - I_n) - \frac{1}{n}(1 - S_n) = -\frac{1}{n}E_n \quad (n = 30,29,\cdots,2)$$

可知，逆向递推计算过程中初始数据误差 E_{30} 的传播规律为

$$|E_{n-1}| = \frac{1}{n(n+1)\cdots 29 \cdot 30}|E_{30}| \quad (n = 29,28,\cdots,2)$$

所以，逆向递推使误差绝对值逐次减小，故这是一种数值稳定的算法.

§1.4 算法设计与选择时的几个原则

一、尽量减少计算工作量

计算机执行一个算法所花费的时间代价除了与问题的规模大小有关外，主要依赖于解决问题过程中所用乘除法次数的多少，也就是计算工作量的大小. 历史上著名的例子是快速傅氏变换(FFT)算法，它将离散傅氏变换的乘法次数

$n \times n$减少到$n \times \log_2 n$，使一个原来用计算机需几百年才能完成的工作现在只需几个小时就可以完成.计算工作量小的算法不仅节约计算机运行时间，而且使误差积累小.

在例 1.4 中两种求 n 次多项式 $P_n(x)=a_0+a_1x+a_2x^2+\cdots+a_nx^n$ 值的算法分别为累加算法和秦九韶算法.它们所用的乘法次数分别为 $2n$ 和 n.显然，秦九韶算法优于累加算法.

二、选用数值稳定性好的算法

对同一个数学问题，即使在数学公式已经确定的情况下，仍然可以设计出多种不同的算法.而不同的算法在执行过程中对数据误差的影响是不一样的，舍入误差对计算结果影响不大的算法被称为数值稳定的算法.

在例 1.9 中，根据误差分析可知，由逆向递推公式所设计的算法是数值稳定的算法，而另一种正向递推公式所设计的算法是数值不稳定的算法.

三、避免两个相近的数据相减

如果 $x_1 \approx x_2$，现分析两个数的近似数作减法所得结果的误差.设 $y=x_1-x_2$，则利用二元函数的全微分近似 $\mathrm{d}y=\mathrm{d}x_1-\mathrm{d}x_2$，得绝对误差限的估计

$$|E(y)| \leqslant |E(x_1)|+|E(x_2)|$$

从而得相对误差限的估计

$$|E_r(y)| \leqslant \frac{|x_1|}{|y|}|E_r(x_1)|+\frac{|x_2|}{|y|}|E_r(x_2)|$$

这表明当 $x_1 \approx x_2$(即 $y \approx 0$)时，$|E_r(x_1-x_2)|$ 可能很大，导致数值计算结果的有效数字位数减少.所以应避免两相近的数据直接作减法运算，具体处理方案随数学表达式不同而不同.常用一些恒等式来将其变形，如当 x 为充分大正数时，

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

当 x 的绝对值很小时，

$$1-\cos x=2\sin^2(x/2)$$

$$\mathrm{e}^x-1 \approx x+\frac{1}{2!}x^2+\frac{1}{3!}x^3+\cdots+\frac{1}{n!}x^n$$

四、绝对值相对小的数不宜作除数

如果 x_1 是绝对值很小的数，现分析用 x_2 的近似数除以 x_1 的近似数计算结果的误差.设 $y=x_2/x_1$，由于

$$dy = \frac{x_1 dx_2 - x_2 dx_1}{x_1^2}$$

所以有绝对误差限的估计

$$|E(y)| \leqslant \frac{1}{|x_1|}|E(x_2)| + \frac{|x_2|}{|x_1^2|}|E(x_1)|$$

这表明：当 x_1 的绝对值很小时，用 x_2 除以 x_1 所得结果的绝对误差可能很大. 因此从误差分析的角度来看，用绝对值很小的数作除数是不好的算法.

五、避免大数和很小的数直接相加

一个绝对值很大的数和一个绝对值很小的数直接相加时，很可能发生所谓“大数吃小数”的现象，从而影响计算结果的可靠性. 这主要是计算机表示的数位数有限这一客观现实引起的.

例如，$a = 10^{13}$，$b = 4$，设想这两个数在具有 12 位浮点数计算机系统(12 位有效位数系)中相加，在机器数系统中相加的原则是先对阶，后相加. 对阶时，

$$\begin{aligned} a + b &= 10^{13} + 4 \\ &= 1.000\ 000\ 000\ 00 \times 10^{13} + 0.000\ 000\ 000\ 00\ \boxed{0\ 4} \times 10^{13} \end{aligned}$$

由于系统只保留前 12 位作为有效数，方框中数据被舍去，实际加法操作如下：

$$1.000\ 000\ 000\ 00 \times 10^{13} + 0.000\ 000\ 000\ 00 \times 10^{13}$$

最后 $a + b$ 计算结果是 $1.000\ 000\ 000\ 00 \times 10^{13}$，即 a 的值作为计算结果赋给 $a + b$. 这显然是很不合理的.

由于“大数吃小数现象”，考虑当绝对值悬殊的一系列数相加时，如果有

$$|x_1| > |x_2| > |x_3| > \cdots > |x_n|$$

成立，则应该按绝对值由小到大的秩序确定累加的先后次序.

实 验 一

1.1 水手、猴子和椰子问题：五个水手带了一只猴子来到南太平洋的一个荒岛上，发现那里有一大堆椰子. 由于旅途的颠簸，大家都很疲倦，很快就入睡了. 第一个水手醒来后，把椰子平分成五堆，将多余的一只给了猴子，他私藏了一堆后便又去睡了. 第二、第三、第四、第五个水手也陆续起来，和第一个水手一样，把椰子分成五堆，恰多一只给猴子，私藏一堆，再去入睡，天亮以后，大家把余下的椰子重新等分成五堆，每人分一堆，正好余一只再给猴子. 试问原先共有几只椰子？

试分析椰子数目的变化规律，利用逆向递推的方法求解这一问题(15621).

1.2 设，$I_n=\int_0^1 \frac{x^n}{5+x}\mathrm{d}x$

(1) 从 I_0 尽可能精确的近似值出发，利用递推公式：

$$I_n=-5I_{n-1}+\frac{1}{n} \quad (n=1,2,\cdots 20)$$

计算从 I_1 到 I_{20} 的近似值；

(2) 从 I_{30} 较粗糙的估计值出发，用递推公式：

$$I_{n-1}=-\frac{1}{5}I_n+\frac{1}{5n} \quad (n=30,29,\cdots,3,2)$$

计算从 I_1 到 I_{20} 的近似值；

(3) 分析所得结果的可靠性以及出现这种现象的原因.

1.3 绘制 Koch 分形曲线

问题描述：从一条直线段开始，将线段中间的三分之一部分用一个等边三角形的另两条边代替，形成具有 5 个结点的新的图形(图 1－4)；在新的图形中，又将图中每一直线段中间的三分之一部分都用一个等边三角形的另两条边代替，再次形成新的图形(图 1－5)，这时，图形中共有 17 个结点. 这种迭代继续进行下去可以形成 Koch 分形曲线. 在迭代过程中，图形中的结点将越来越多，而曲线最终显示细节的多少将取决于所进行的迭代次数和显示系统的分辨率. Koch 分形曲线的绘制与算法设计和计算机实现相关.

图 1－4 第一次迭代

图 1－5 第二次迭代

问题分析：考虑由直线段(2 个点)产生第一个图形(5 个点)的过程. 设 P_1 和 P_5 分别为原始直线段的两个端点. 现在需要在直线段的中间依次插入三个点 P_2,P_3,P_4 产生第一次迭代的图形(图 1－4). 显然，P_2 位于 P_1 点右端直线段的三分之一处，P_4 位于 P_1 点右端直线段的三分之二处；而 P_3 点的位置可以看成是由 P_4 点绕 P_2 旋转 60 度(逆时针方向)而得到的，故可以处理为向量 P_2P_4 经正交变换而得到向量 P_2P_3. 形成算法如下：

(1) $P_2=P_1+(P_5-P_1)/3$；

(2) $P_4=P_1+2(P_5-P_1)/3$；

(3) $P_3=P_2+(P_4-P_2)\times A^{\mathrm{T}}$；

在算法的第三步中，A 为正交矩阵：