夏省祥　于正文　著

常用数值算法及其MATLAB实现

清华大学出版社

夏省祥　于正文　著

常用数值算法及其MATLAB实现

清华大学出版社

内 容 简 介

本书详细介绍了求解数值问题的常用算法的算法原理及其 MATLAB 实现，偏重于算法的实现，强调例题的分析和应用。主要内容包括：线性方程组的直接解法和迭代解法、插值和函数逼近、数值积分、数值优化、矩阵的特征值问题、解非线性方程和方程组的数值方法及常微分方程和偏微分方程的数值解法。

本书可作为高等院校数学与应用数学专业、信息与计算科学专业和计算机应用等专业的本科生及工科硕士研究生的教材或参考书，也可供从事科学与工程计算的技术人员参考。

图书在版编目（CIP）数据

常用数值算法及其 MATLAB 实现/夏省祥，于正文著. --北京：清华大学出版社，2014
ISBN 978-7-302-35334-8

Ⅰ. ①常…　Ⅱ. ①夏… ②于…　Ⅲ. ①数值计算－Matlab 软件　Ⅳ. ①O245

中国版本图书馆 CIP 数据核字(2014)第 021112 号

责任编辑：汪　操
封面设计：陈国熙
责任校对：赵丽敏
责任印制：何　芊

出版发行：清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦 A 座　　**邮　　编**：100084
社 总 机：010-62770175　　**邮　　购**：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质量反馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
课件下载：http://www.tup.com.cn,010-62770175-4113
印 装 者：北京鑫海金澳胶印有限公司
经　　销：全国新华书店
开　　本：185mm×260mm　　**印　　张**：23.25　　**字　　数**：563 千字
版　　次：2014 年 4 月第 1 版　　**印　　次**：2014 年 4 月第1 次印刷
印　　数：1～3000
定　　价：45.00 元

产品编号：052657-01

前言

FOREWORD

随着社会的发展和科学技术的进步，需要解决的问题越来越多，也越来越复杂，计算机与计算数学的关系也越来越密切，古老的计算数学发展成了一门现代意义下的新学科——科学计算。科学计算在国防、经济、天气预报、工程、航空航天工业、自然科学等领域有着广泛的应用，科学计算已和理论计算、实验并列为三大科学方法。科学计算离不开计算机，但它更离不开计算方法。美国著名的计算数学家 Babuska 曾说过："没有好的计算方法，超级计算机就是超级废铁。"人类的计算能力等于计算工具的效率与计算方法的效率的乘积，这一形象化的公式表达了硬件与计算方法对于计算能力的同等重要性。现代意义下的计算数学要研究的是在计算机上进行大规模计算的有效算法及其相应的数学理论，它是科学计算的核心。

本书详细、系统地阐述了常用的数值算法和一些现代算法的原理，并用目前最流行的三大数学软件 MATLAB，Maple 和 Mathematica 之一的 MATLAB 全部实现了这些数值算法，本书偏重于算法的实现，强调例题的分析和应用，引导读者轻松入门，深刻理解、掌握算法原理，并迅速应用。

在结构体系方面，先介绍数值算法的详细计算方法（公式）和相关概念，其次给出实现算法的 MATLAB 程序，最后给出范例。力求把最实用、最重要的知识讲清楚，把最有效的算法和最实用的程序展现给读者。每个算法后都列举了典型范例，对大多数例题采用多种数值解法（包括 MATLAB 程序包中的数值算法），并尽量用图形显示计算结果，以便直观观察和比较不同方法的计算效果。对有精确解（解析解）的问题，将数值算法求出的数值解与精确解比较，客观地评价数值算法的优劣，以便选择精度高的最佳数值算法。在编程过程中采用高效的计算方式，减少不必要的重复计算，尽量少调用函数且注重误差的传播等编程细节，并对一些算法的适用范围、优劣和误差以及参数和初始值对计算结果的影响进行了分析。帮助读者理解、掌握、改进数值算法，提高数值分析的技能和编程能力。

本书从二十多本国内外教材和十几篇国内外公开发表的论文中精选了 170 多个典型例题，并通过大量的数据结果和 150 多幅图表详细地介绍了常用的经典数值算法和一些现代算法的算法原理及其应用。所有源程序完全开放，程序全部用形式参数书写，读者只需输入参数、函数和数据等就可方便地使用它们，当然也可以根据自己的需求更改这些程序。书中的所有算法程序都在 MATLAB 7.1 中验证通过，并通过不同的算法或精确解检验了程序的正确性。

国家自然科学基金项目(项目编号：51078225)和山东省高等教育名校建设工程—山东建筑大学特色专业建设项目对本书的出版给予了资助，在此表示衷心的感谢。

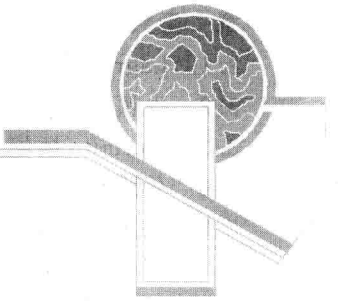
由于作者水平所限，书中不妥或错误之处在所难免，恳请读者批评指正。

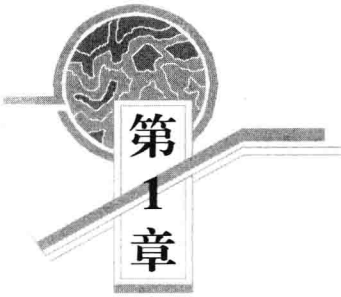
作　者

2014 年 2 月

目 录

CONTENTS





第1章 引 论

1.1 误差的来源

在解决工程和科学问题时，会由不同的原因产生误差。首先，误差可能来自数学模型，一般情况下，数学模型无法确切地表达实际问题，这种由数学模型与实际问题之间产生的误差称为模型误差。其次，数学模型中常包含一些通过观察所得到的数据，由观测值而产生的误差称为观测误差。由于这两种误差不是科学计算过程能够避免的，因此，在科学计算中，我们重点关注如下两种误差：舍入误差和截断误差。

1.1.1 舍入误差

由于计算机硬件只支持有限位机器数的运算，因此有时不能确切地表示实数的真实值，这种误差称为舍入误差。

例 1.1 考察积分 $T(n)=\int_0^1 \frac{x^n}{x+9}\mathrm{d}x$，利用 MATLAB，求 $n=30$ 的积分值。

解 在计算之前，先估计一下积分值：$0\leqslant\int_0^1 \frac{x^n}{x+9}\mathrm{d}x\leqslant\int_0^1 x^n\mathrm{d}x=\frac{1}{n+1}$。在 MATLAB 命令窗口输入：

```
>>fun='x^30/(x+9)';
>>T30=int(fun,0,1)
```

则屏幕显示结果：

```
T30 =42391158275216203514294433201 * ln(2)
     +42391158275216203514294433201 * ln(5)
     -84782316550432407028588866402 * ln(3)
     -16511966940855481709335615784931213253l/36969675600
```

(1.1)

利用 vpa 函数，用 16 位精度进行计算则其浮点值为

$$\int_0^1 \frac{x^n}{x+9}\mathrm{d}x = \mathrm{vpa(T30,16)} = -0.60\times 10^{14}。$$

这显然是个错误的结果，下面分析产生错误的原因。注意到

$$T(n)+9T(n-1)=\int_0^1\frac{x^n+9x^{n-1}}{x+9}\mathrm{d}x=\int_0^1 x^{n-1}\mathrm{d}x=\frac{1}{n},\quad T(0)=\int_0^1\frac{1}{x+9}\mathrm{d}x=\ln\frac{10}{9}。$$

求解此递推关系，则有

$$T(n)=(\ln(2)+\ln(5)-2\ln(3))(-9)^n+\sum_{n_0=1}^{n}\frac{(-9)^{n-n_0}}{n_0}。\tag{1.2}$$

如果我们直接用递推方法求 $T(30)$，则得到如下结果。

```
>>T0=log(10/9);
for n=1:30
    T0=vpa(-9*T0+1/n);
end
T30=T0
T30=2034160460084.18287
```

是什么原因导致的错误呢？也许你已经猜到，是舍入误差。在运算时用到的初值是 ln(10/9)(在 MATLAB 输出中它被写成了：ln(2)+ln(5)−2ln(3))，此数在计算机中无法精确表达，如果用 32 位的精度，则计算机表示数 ln(10/9)的舍入误差大约是 2^{-32}。当我们对(1.1)式、(1.2)式进行计算时，误差将变为

```
>>err=2^(-32)*9^30
err=9.869960666451651e+018
```

利用 vpa 函数，若用 50 位精度进行计算则其浮点值为

$$\int_0^1\frac{x^n}{x+9}\mathrm{d}x=\mathrm{vpa(T30,50)}=0.0032359487369。$$

1.1.2 截断误差

用一个基本的表达式替换一个比较复杂的表达式时所产生的误差，称为截断误差。这一术语是从用截断泰勒级数替换一个复杂表达式的技术中衍生的。

例 1.2 对无穷泰勒级数 $e^x=1+x+\frac{x^2}{2!}+\frac{x^3}{3!}+\cdots+\frac{x^n}{n!}+\cdots$，用有限项近似表达 e^x 时的图像比较。

解 建立如下脚本文件。

```
fplot('exp(x)',[-5 5 -3 20],'b:+');
hold on
fplot('1+x+x.^2/2',[-5 5 -3 20],'r:*');
fplot('1+x+x.^2/2+x.^3/6',[-5 5 -3 20],'k:.');
fplot('1+x+x.^2/2+x.^3/6+x.^4/24+x.^5/120',[-5 5 -3 20],'g:o');
legend('e^x','TL(2,x)','TL(3,x)','TL(5,x)');
hold off
```

存为 ex1_2.m。在 MATLAB 命令窗口调用 ex1_2，则得到下面的结果(图像为图 1.1)。

用部分和 $S_N=\sum_{n=0}^{N}\frac{x^n}{n!}$ 近似代替无限和 $e^x=\sum_{n=0}^{\infty}\frac{x^n}{n!}$，所产生的理论误差为 $E_N=\sum_{n=N+1}^{\infty}\frac{x^n}{n!}$。

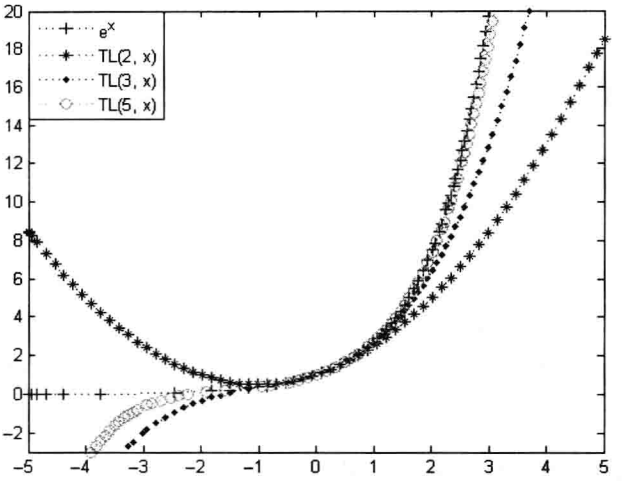


图 1.1 e^x 及其近似表达式的图形

当 x 接近于 0 时，级数收敛速度较快，此时舍入误差的实质部分包含在舍入部分的第一项中。这样就可一直求和，直到第 N 项满足

$$\frac{|\ \text{term}_N\ |}{|\ S_N\ |}<\varepsilon \quad \text{或} \quad |\ \text{term}_N\ |<\varepsilon\,|\ S_N\ |$$

为止，其中 ε 是我们要求的相对误差。例如，计算 e^7。

```
>>vpa(exp(7),15)
ans=1096.63315842846。
```

如果要求相对误差为 10^{-9}，则 e^7 的近似值和所要计算的项数分别为 sum＝1096.633158 和 k＝30。

```
>>term=1;
sum=term;
k=1;
while term/sum>=10^(-9)
    term=term*7/k;
    sum=sum+term;
    k=k+1;
end
sum
k
```

在 MATLAB 命令窗口执行上述代码，则有

```
sum=1.096633158318981e+003
k=30
```

当我们计算到第 30 项时，截断误差为 $\sum_{n=31}^{\infty}\frac{7^n}{n!}$，约为 2.45×10^{-8}，上述级数的第一项为$\frac{7^{31}}{31!}$，

约为 1.92×10^{-8}，是截断误差 2.45×10^{-8} 的实质部分。

1.2 误差的传播

在进行计算时，如果参与运算的量有误差，则计算结果也会有误差。参与运算的量的误差可以通过一系列的运算进行传播，如四则运算、函数运算等。

尽管在某种程度上我们无法避免误差的影响，但是要对计算误差负责的是我们自己，而不是计算机。当我们的计算机对无意的谎言坚持自己是清白的时候，我们编程者和使用者必须对所用算法而产生的误差负责，而且还要为自己的粗心付出被机器欺骗的代价。因此，我们应尽力减少误差量并使误差量对最终结果的影响最小化，下面是避免误差增大的几条原则。

1.2.1 尽量避免两个相近的数相减

如果对两个相近的数进行减法运算，将造成有效数字的严重损失，相对误差迅速增加。

设 $f(x_1,x_2,\cdots,x_n)$ 是 n 元函数，则 f 最大可能的误差为

$$|\Delta f|\approx\left|\frac{\partial f}{\partial x_1}\Delta x_1\right|+\left|\frac{\partial f}{\partial x_2}\Delta x_2\right|+\cdots+\left|\frac{\partial f}{\partial x_n}\Delta x_n\right|。\tag{1.3}$$

例 1.3 若 $z=f(x,y)=x-y$，则由(1.3)式得 $|\Delta z|=\left|\frac{\partial z}{\partial x}\Delta x\right|+\left|\frac{\partial z}{\partial y}\Delta y\right|=|\Delta x|+|\Delta y|$，所以 Δz 的相对误差为

$$\left|\frac{\Delta z}{z}\right|=\frac{|\Delta x|+|\Delta y|}{|x-y|}。$$

如果 $x=3\pm0.001$，$y=3.003\pm0.001$，则 $\left|\frac{\Delta z}{z}\right|=\frac{|0.001|+|0.001|}{|3-3.003|}\approx0.6667$。

例 1.4 设 $f(x)=x(\sqrt{x+1}-\sqrt{x})$，$g(x)=\frac{x}{\sqrt{x+1}+\sqrt{x}}$，用 15 位精度计算 $f(5000)$，$g(5000)$。

解

```
>>f=inline('x*(sqrt(x+1)-sqrt(x))','x');
>>g=inline('x/(sqrt(x+1)+sqrt(x))','x');
>>vpa(f(5000),15)
ans=35.3535714690878
>>vpa(g(5000),15)
ans=35.3535714691290
```

理论上讲，$f(x)=g(x)$，由于两个相近的数 $\sqrt{5001}$ 和 $\sqrt{5000}$ 相减导致 $f(5000)$ 的精度损失。

例 1.5 设 $f(x)=\frac{1-\cos(x)}{x^2}$。

(1) 画出 $f(x)$ 在 $-5\pi\leqslant x\leqslant5\pi$ 上的图像。

(2) 验证 $\lim\limits_{x\to0}f(x)=\frac{1}{2}$。

(3) 取 $x_0=11\times10^{-6}$。

(a) 用 30 位精度,求 $f(x)$在 x_0 处的值;

(b) 用 10 位精度,求 $f(x)$在 x_0 处的值。

解 (1)

```
>>ff=inline('(1-cos(x))/x^2');
>>fplot(ff,[-5*pi,5*pi])
```

图像为图 1.2。

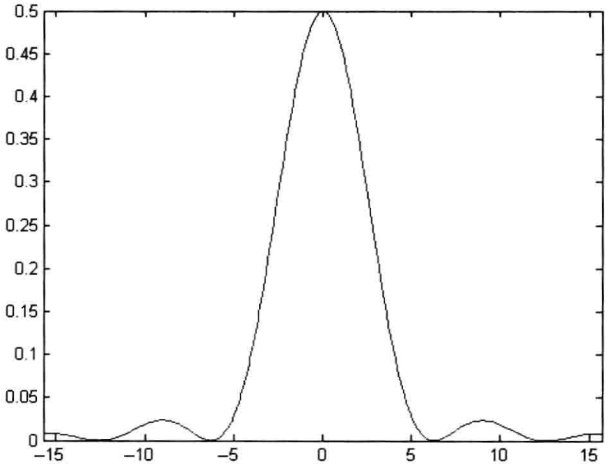


图 1.2 $f(x)$在$[-5\pi,5\pi]$上的图像

(2) 将 $f(x)$定义为符号函数。

```
function y=exa1_5(x)
syms x
y=(1-cos(x))/x^2;
```

存为 exa1_5.m。

```
>>limit(exa1_5,0)
ans=1/2
```

(3) (a)

```
>>x0=vpa(11/1000000,10);
>>(1-vpa(cos(x0),30))/x0^2          %用 30 位精度计算 cos(x0)
ans=0.49999999999496
```

(b)

```
>>(1-vpa(cos(x0),10))/x0^2          %用 10 位精度计算 cos(x0)
ans=0.826446280992
```

当 x 接近于 0 时,$1-\cos(x)\approx\frac{x^2}{2}$,由于两个近似相等的数 1 和 $\cos(x)$相减,导致误差增大。

1.2.2 防止接近零的数做除数

分母接近零的数会产生大的误差或溢出，可以用数学公式化简后再计算。

例 1.6 当 $x \gg 1$ 时，计算 $f(x)=\dfrac{58}{\sqrt{x+1}-\sqrt{x}}$，可改为 $g(x)=58(\sqrt{x+1}+\sqrt{x})$ 进行计算。理论上，$f(x)=g(x)$。

解 取 $x=10^9, 10^{16}$，建立脚本文件计算 $f(x), g(x)$，得到如下结果。

```
fx=inline('58/(sqrt(x+1)-sqrt(x))');
gx=inline('58*(sqrt(x+1)+sqrt(x))');
x1=10^9;
fx1=fx(x1)
gx1=gx(x1)
x2=10^16;
fx2=fx(x2)
gx2=gx(x2)
```

存为 ex1_6.m。调用 ex1_6，有

```
>>ex1_6
fx1=3.668241513567938e+006
gx1=3.668242086712380e+006
fx2=Inf
gx2=1.160000000000000e+010
```

1.2.3 防止大数吃小数

当两个绝对值相差很大的数进行加法或减法运算时，绝对值小的数有可能被绝对值大的数“吃掉”，从而引起错误的结果。

例 1.7 用 5 位浮点数，计算

$$S = 12345 + 0.3 + 0.3 + 0.4。 \tag{1.4}$$

解 $S=0.12345\times10^5+0.000003\times10^5+0.000003\times10^5+0.000004\times10^5=12345$。在(1.4)式中，重新排序计算得

$$S = 0.000003\times10^5 + 0.000003\times10^5 + 0.000004\times10^5 + 0.12345\times10^5 = 12346。$$

1.2.4 简化计算步骤，减少运算次数

简化计算步骤是提高程序执行速度的关键，它不仅可以节省时间，还能减少舍入误差。例如，计算 n 次多项式 $p_n(x)=a_nx^n+a_{n-1}x^{n-1}+\cdots+a_1x+a_0$ 的值时，如果先求 a_kx^k 再求和，需要 $n(n+1)/2$ 次乘法，n 次加法。如果按下式计算

$$p_n(x) = x(\cdots x(x(a_nx + a_{n-1}) + a_{n-2}) + \cdots + a_1) + a_0,$$

则只需要 n 次乘法，n 次加法，这就是计算多项式著名的秦九韶算法。

例 1.8 用上述两种方法计算多项式 $p_{n-1}(x)=a_nx^{n-1}+a_{n-1}x^{n-2}+\cdots+a_2x+a_1$，其中 $n=10^6$，$[a_1,a_2,\cdots,a_n]=[1,2,\cdots,10^6]$，在 $x=1.0012$ 时的值。

解 建立如下的脚本文件，并存为 ex1_8.m。

```
x0=1.00012;
t=1;
a=[1:10^6];
tic                                    %开始计时
pn=a(10^6);
for k=(10^6-1):-1:1
    pn=pn*x0+a(k);                     %按秦九韶算法计算
end
pn
ts1=toc                                %计算从上次计时到现在所用时间
tic
summ=a(1);
for k=2:10^6
    t=t*x0;
    summ=a(k)*t+summ;                  %按计算 ak*x^k 方式计算
end
summ
ts2=toc
```

在 MATLAB 命令窗口执行 ex1_8.m，则有

```
>>ex1_8
pn=1.0700e+062
ts1=0.0159
summ=1.0700e+062
ts2=0.0261
```

1.3 数值算法的稳定性

对于一个算法，如果初始数据的较小误差不会对最终结果产生较大的影响，则称此算法是数值稳定的，否则称此算法为不稳定的。衡量算法好坏的两个重要指标是稳定性和计算复杂性。计算复杂性包括时间复杂性和空间复杂性。时间复杂性即计算量：一个算法所需四则运算总次数，在实际中通常以乘、除法的次数作为算法的计算量，单位是 flop。空间复杂性即存储量。

例 1.9 设 $\boldsymbol{A},\boldsymbol{B},\boldsymbol{C},\boldsymbol{D}$ 分别是 $10\times20,20\times50,50\times1,1\times100$ 的矩阵，试按不同的算法求矩阵乘积 $\boldsymbol{E}=\boldsymbol{ABCD}$。

解 由矩阵乘法的结合律，可有如下算法：

(1) $\boldsymbol{E}=((\boldsymbol{AB})\boldsymbol{C})\boldsymbol{D}$，计算量 $N=11500$flop。

(2) $\boldsymbol{E}=\boldsymbol{A}(\boldsymbol{B}(\boldsymbol{CD}))$，计算量 $N=125000$flop。

(3) $\boldsymbol{E}=(\boldsymbol{A}(\boldsymbol{BC}))\boldsymbol{D}$，计算量 $N=2200$flop。

例 1.10 用无限精度算法结合如下 2 个方案，可递归生成序列 $\{a_n\}=\left\{\frac{1}{3^n}\right\}_{n=0}^{\infty}$。