微分方程数值解

陈文斌　程　晋　吴新明　李立康　编著

博学·数学系列

复旦大学出版社
www.fudanpress.com.cn

图书在版编目(CIP)数据

微分方程数值解/陈文斌、程晋、吴新明、李立康编著. —上海:复旦大学出版社,2014.8
(复旦博学·数学系列)
ISBN 978-7-309-10786-9

Ⅰ. 微… Ⅱ. ①陈…②程…③吴…④李… Ⅲ. 微分方程-数值计算-高等学校-教材
Ⅳ. 0241.8

中国版本图书馆 CIP 数据核字(2014)第 139876 号

微分方程数值解
陈文斌 程 晋 吴新明 李立康 编著
责任编辑/范仁梅

复旦大学出版社有限公司出版发行
上海市国权路 579 号 邮编:200433
网址:fupnet@fudanpress.com http://www.fudanpress.com
门市零售:86-21-65642857 团体订购:86-21-65118853
外埠邮购:86-21-65109143
上海肖华印务有限公司

开本 787×960 1/16 印张 19.75 字数 327 千
2014 年 8 月第 1 版第 1 次印刷

ISBN 978-7-309-10786-9/O·540
定价:38.00 元

目　录

第一章　数值分析基础

这一章将从简单的递推格式开始, 介绍数值分析的一些基础知识, 说明计算什么时候是可靠的和可信的, 什么时候是会出问题的. 函数逼近和数值积分作为两个最基本的离散方法, 这里也将简单介绍. 如果前期已经选修过"数值逼近"的读者, 可以跳过本章, 或作为选读.

1.1　一个简单的递推格式

回忆一下在高中, 曾经做过的非常简单的练习:

例1.1.1　设序列$u(n)$满足

$$u(n+2)-2u(n+1)+u(n)=0. \tag{1.1.1}$$

当$u(1)=u(2)=0.1$时, 求$u(n)$.

显然当$u(1)=u(2)=0.1$的时候, 容易求得这个序列的每一项都等于0.1. 下面用MATLAB程序来验证这个简单的结果.

```
function u = Ch1Exam1(u0,u1,a,b,N)                                  1
% 用三项递推公式求序列: u(n+2) = a*u(n+1) + b*u(n)                  2
% 输入:  u0 u1: 初始值                                              3
%         a,b: 系数                                                 4
%         N: 递推步数                                               5
% 输出:  u(1:N)                                                     6
% 例如:  u=Ch1Exam1(0.1,0.1,2,-1,100);                              7
%         u=Ch1Exam1(0.1,0.1,3,-2,100);                             8
u = [u0; u1];                                                       9
for i=1:N-2                                                         10
    u2 = a*u1 + b*u0;                                               11
    u0 = u1; u1 = u2;                                               12
    u = [u; u2];                                                    13
end                                                                 14
```

如果在MATLAB中执行这个程序

```
>>u=Ch1Exam1(0.1, 0.1, 2, -1, 100); plot(u);
```

那么就会形成一个100维的向量$\boldsymbol{u}$, 其元素全部是0.1(见图1.1).

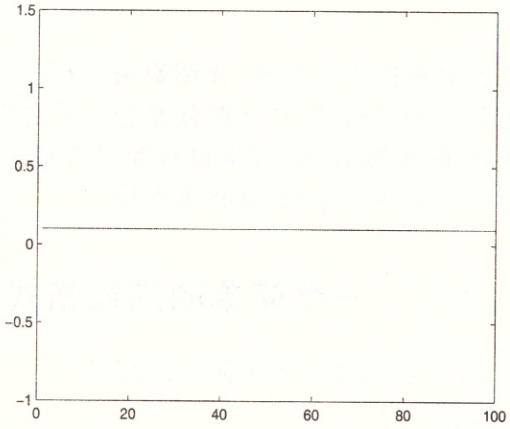


图 1.1: 100维的向量$\boldsymbol{u}$

注意到利用上面的程序, 可以计算如下的三项递推公式:

$$u(n+2) = au(n+1) + bu(n). \tag{1.1.2}$$

下面再做个试验, 取$a = 3, b = -2$, 也就是说求如下的问题.

例1.1.2 用程序Ch1Exam1.m计算$u(n)$, 这里$u(n)$满足

$$u(n+2) - 3u(n+1) + 2u(n) = 0, \tag{1.1.3}$$

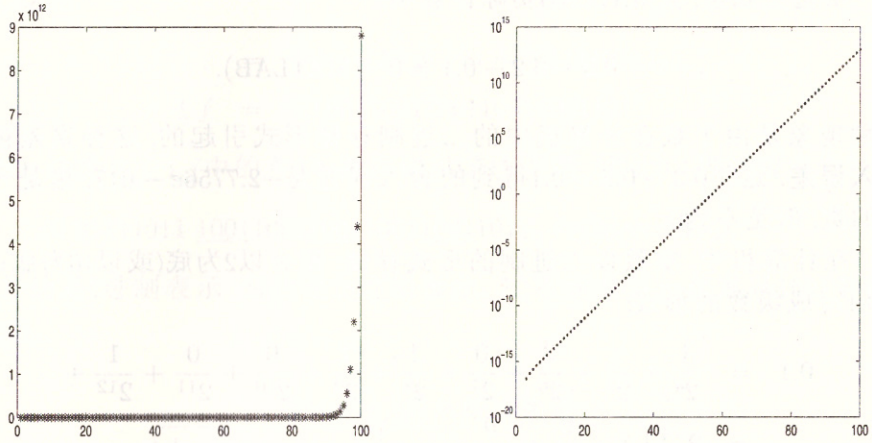
$$u(1) = 0.1, \quad u(2) = 0.1. \tag{1.1.4}$$

显然如果把初始值代到递推公式, 容易知道这里$u(n)$也总是等于0.1. 现在通过给定初始值, 可以调用如下程序Ch1Exam1:

```
>>[u]=Ch1Exam1(0.1,0.1,3,-2,100);figure(1);plot(u,'*')
```

图1.2(a)的计算结果与我们的想象差得实在太远, 这里算出来的$u(n)$并不是常数0.1, 而是随着迭代步数n的增大, 增长得非常快(指数增长)!

现在来观察误差的半对数图, 也就是用半对数图来画计算得到的$u(n)$和精确解的差:



(a) 计算解$u(n)$　　(b) 计算解与精确解误差的半对数图

图 1.2: 例子1.1.2

```
>>e=u-0.1; semilogy(e,'.')
```

从误差的半对数图图1.2(b)可以看到: 随着求解的进行, 在半对数图里误差近似于一条直线, 也就是说误差随迭代步数指数增长. 如果看$n = 100$的误差, 有$e(100) \approx 8.7961 \times 10^{12}$, 精确解完全被误差淹没.

这个例子告诉我们: 哪怕简单的计算格式都可能产生问题. 那么在例子1.1.2中, 问题到底是如何引起的呢? 误差为什么会指数增长呢? 这要从计算机中数的表示和存储说起.

1.1.1 0.1不能被双精度精确表示

下面做一个简单的练习.

例1.1.3 在MATLAB中检查一下0.3 - 0.2和0.3 - 0.2 - 0.1.

```
>>0.3 - 0.2
        0.1000
>>0.3 - 0.2 - 0.1
        -2.7756e-017
```

从这里知道, 在MATLAB实际计算中:

$$0.3 - 0.2 - 0.1 \neq 0 \quad \text{(MATLAB)}.$$

这种现象是由于数在计算机中的二进制存储形式引起的, 这种误差被称为**舍入误差**. 这里$0.3 - 0.2 - 0.1$得到的舍入误差是$-2.7756e-017$, 这是一个很小的数, 但是不为0.

在计算机中, 实数以二进制的形式存储, 那么以2为底(或以16为底), 可以把0.1写成级数的形式:

$$\begin{aligned} 0.1 &= \frac{1}{2^4} + \frac{1}{2^5} + \frac{0}{2^6} + \frac{0}{2^7} + \frac{1}{2^8} + \frac{1}{2^9} + \frac{0}{2^{10}} + \frac{0}{2^{11}} + \frac{1}{2^{12}} + \cdots \\ &= 2^{-4}\left(1 + \frac{9}{16} + \frac{9}{16^2} + \cdots\right). \end{aligned}$$

由于这个级数不能展开成有限项, 这就意味着:

0.1不能用有限位的寄存器精确存储.

在工业上, IEEE754-1985标准(新的标准为IEEE754-2008, 等价的国际标准为ISO/ IEC/IEEE 60559:2011)用来规范浮点数的存储和运算. 在存储之前, 0.1首先要按照IEEE754-1985标准进行规范化:

$$0.1 = +2^e \cdot (1+f)_2, \quad e = -4, \quad f = 10011001\cdots.$$

在实际存储的时候, 需要存3个信息(见表1.1):

表 1.1: 数的存储

符号位	指数部分	尾数部分
s	e	f

- 符号位s: $s=0$ 表示正数, $s=1$表示负数.
- 指数部分e: 如果是单精度, 用8位表示, 并存储成$e+127$; 如果是双精度, 用11位表示, 并存储成$e+1023$.
- 尾数部分f: 如果是单精度, 则存储23位; 如果是双精度, 则存储52位.

这样可以得到数0.1的两种存储格式:

- 单精度存储(single precision). 0.1表示成规范的(normalized)单精度数为

$$0.1 = (-1)^s \cdot (2^{e-127}) \cdot (1.f)_2.$$