21 世纪数学系列教材

数值分析简明教程

（修订版）

王能超　编著

华中科技大学出版社
E-mail: hustpp@wuhan.cngb.com

• 本书荣获原国家教委优秀教材二等奖

数值分析简明教程

（修订版）

王能超　编著

本书（初版）荣获原国家教委
优秀教材奖二等奖

华中科技大学出版社
中国·武汉

图书在版编目(CIP)数据

数值分析简明教程(修订版)/王能超　编著
武汉:华中科技大学出版社，　2002 年 9 月
ISBN 7-5609-2703-3

Ⅰ.数…
Ⅱ.王…
Ⅲ.数值分析-高等学校-教材
Ⅳ.O241

数值分析简明教程(修订版)　　王能超　编著

责任编辑:李健萍　　封面设计:刘　卉
责任校对:张兴田　　责任监印:张正林

出版发行:华中科技大学出版社
　　武昌喻家山　邮编:430074　电话:(027)87545012

录　排:华中科技大学出版社照排室
印　刷:湖北新华印务有限公司

开本:787×960　1/16　　印张:15.75　　字数:283 000
版次:2002 年 9 月第 1 版　　印次:2002 年 9 月第 1 次印刷　　印数:1—5 000
ISBN 7-5609-2703-3/O・257　　定价:19.80 元

内 容 提 要

本书是以《数值分析简明教程》(高等教育出版社,1984 年版)一书为基础,经过补充、修改而成的. 原书已发行 20 多万册,深受读者喜爱. 本版继续保持了本书内容精练、深入浅出、通俗易懂的突出特点,在编排上贯穿了计算方法的思想;为方便读者深入掌握有关内容,同时为"数值分析"的习题课提供参考资料,新写了"数值微积分例题选讲"部分,提炼、归纳了数值分析中最重要的一些方法,并对若干例题进行了解析,使本书又增添新的特色.

全书讲授 45～50 学时. 可作为高等院校一般工科专业学生的教材,也可供工程技术人员以及其他科技人员阅读参考.

修订版前言

拙作《数值分析简明教程》(后文简称《简明教程》)自1984年由高等教育出版社出版以来,迄今已过去了18个年头.这期间年年重印,已累计发行20余万册.作者衷心感谢关注支持本书的广大老师和同学们.

本书追求简明.数值分析的基本内容是数值算法的设计与分析.本书坚持这样的观点:对于数值微积分,无论是算法的设计还是算法的分析,其高等数学的基础都是泰勒公式.一些学术界同行评价本书是"泰勒公式包打天下".这种说法是中肯的.

微积分的发明是人类智慧的伟大发展.什么是微积分?华人数学家项武义先生精辟地指出:"俗语常常用'程咬金三斧头'来笑话一个人的招式贫乏,那么微积分可就只有'逼近法'这一斧头了!可是逼近法这一斧头却是无往不利、无坚不摧的!学微积分也就是要学会灵活地运用逼近法去简化和解决实际问题。"(项武义著,微积分大意,人民教育出版社,1978年版)

微积分的精华是逼近法.逼近法的精髓是泰勒公式.作者在编写数值分析教材的过程中始终坚持这一指导思想.

《简明教程》的原型是作者于1978年编写的《工程数学——计算方法》一书.该书是"文革"后受命编写的工科院校的"统编教材".自1978年元月"接受任务"到当年5月在上海通过评审,该书的出版是仓促的.在上海审稿会上,参与审稿的诸位先生协助弥补了书稿中的不少缺陷与不足.西安交大游兆永先生在会上建议增补有关曲线拟合方面的内容,并亲自赶写了一份材料附在书后.后来,作者将这份珍贵的"附录"稍加充实,改写成"曲线拟合的最小二乘法"一节纳入《简明教程》一书的正文,留作永久的纪念。

正如"初版前言"所指出的,《简明教程》一书得以顺利出版,完全仰仗游兆永先生的鼎力支持.令人难以忘怀的是游先生曾对作者透露过的一个"秘密":1988年原国家教委评选优秀教材时,评审会上曾有人提议给予《简明教程》以更高的奖励,游先生婉转地劝阻了这项建议.平平常常才是真.游先生以其崇高的威望和博大的胸怀,无微不至地关怀爱护《简明教程》这本小书的命运.

在《简明教程》即将重版的今天,作者深切地怀念良师挚友游兆永先生.

王能超

2002年6月20日

于华中科技大学

初版前言

人类社会正迈进电子计算机时代.在今天,熟练地运用计算机进行科学计算,已经成为广大科技工作者的一项基本技能,这就需要向高等工科院校的学生普及有关计算方法的知识.本书正是为适应这一形势而编写的.

要提高运用计算机进行科学计算的能力,关键在于加强数学修养.不应当将计算方法片面地理解为各种算法的简单罗列和堆积,同数学分析一样,它也是一门内容丰富、思想方法深刻而有着自身的理论体系的数学学科.本书取名为数值分析正是基于这一认识.

本书是以《工程数学——计算方法》(王能超编,人民教育出版社 1978 年版)一书为基础,经过补充修改编写而成的.

在教育部直属工科院校计算数学教材讨论会上(1983 年,武汉),曾对本书原稿进行了审议.参加审议的有清华大学、浙江大学、西安交通大学、大连工学院、南京工学院、天津大学、重庆大学和华侨大学等院校的老师.由西安交通大学游兆永教授负责主审.参加审议的同志在推荐本书出版的同时,还提出了许多宝贵的意见和建议,编者对此表示深切的谢意.

王能超

1984 年 12 月 25 日

于华中工学院

目　　录

引　论

科学技术的发展提出大量复杂的数值计算问题，这些问题的解决不是人工手算(包括使用算盘以及计算器之类简单的计算工具)所能胜任的，必须依靠电子计算机. 用电子计算机进行这种科学技术计算的工作，称为**科学计算**，或简称**电算**.

科学计算的应用范围非常广泛，国防尖端的一些科研项目，如核武器的研制、导弹的发射等等，始终是科学计算最为活跃的领域. 今天，科学计算在工农业生产的各个部门也正在发挥日益重要的作用.

例如，气象资料的汇总、加工并求得天气图像，这方面工作的计算量大而且时间性强，要求电子计算机作高速或超高速运算，以对天气作出短期及中期预报.

又如，将所设计的船型型体数值表转换成初始数据输入电子计算机，经过计算即可求出外板和肋骨的展开数据. 在造船工业中用这种方法进行数学放样，既节省了人力物力，又缩短了设计周期.

本门课程将着重介绍进行科学计算所必须掌握的一些最基本、最常用的算法.

§1　算　法

1. 研究算法的意义

电子计算机的运算速度高，可以承担大运算量的工作，这是否意味着计算机上的算法我们可以随意选择呢？

我们知道，行列式解法的克莱姆(Cramer)法则原则上可用来求解线性方程组. 用这种方法求解一个 n 阶方程组，要算 $n+1$ 个 n 阶行列式的值，为此总共需要做 $n!(n-1)(n+1)$ 次乘法. 当 n 充分大时，这个计算量是相当惊人的. 譬如一个20阶不算太大的方程组，大约要做 10^{21} 次乘法，这项计算即使用百万次每秒的电子计算机去做，也得要连续工作千百万年才能完成. 当然这是完全没有实际意义的. 其实，解线性方程组有许多实用的算法(参看本书第五章与第六章)，譬如用众所周知的消元法，一个 20 阶的方程组即使用计算器也能很快地解出来. 这个简单的例子说明，能否正确地制定算法是科学计算成败的关键.

2. 什么是算法

针对一个具体的数学问题，可以给出多种解法.

例 1 证明二次方程

$$x^2 + 2bx + c = 0 \tag{1}$$

至多有两个不同的实根.

解 下面提供三种解法.

1)**反证法** 假定方程(1)有三个互异的实根 x_1, x_2 和 x_3，则有

$$x_1^2 + 2bx_1 + c = 0$$

$$x_2^2 + 2bx_2 + c = 0$$

$$x_3^2 + 2bx_3 + c = 0$$

以上式子两两相减得

$$x_2 + x_1 + 2b = 0$$

$$x_3 + x_2 + 2b = 0$$

从而有 $x_1 = x_3$，这与原设矛盾. 证毕.

2)**图解法** 方程(1)配方得

$$(x + b)^2 + c - b^2 = 0 \tag{2}$$

在坐标纸上描出抛物线 $y = (x + b)^2 + c - b^2$，它与 x 轴的交点(横坐标)即为所求的实根，而交点至多只有两个.

3)**公式法** 据式(2)可导出直接的求根公式

$$x_{1,2} = - b \pm \sqrt{b^2 - c} \tag{3}$$

上述三种方法，反证法不是构造性的；作图法虽是构造性的，但不是数值的. 我们所说的"算法"，必须是构造性的数值方法，即不但要论证问题的可解性，而且解的构造是通过数值演算过程来完成的.

同传统意义的近似计算方法不同，我们所要研究的算法是为电子计算机提供的，因此，解题方案当中的每个细节都必须准确地加以定义，并且要完整地描述整个解题过程. 我们所说的"算法"不仅仅是单纯的数学公式，而是指解题方案的准确和完整的描述.

描述算法可以用多种方式，本书常用框图直观地显示算法的全貌.

譬如，设要用公式(3)求解二次方程(1)，则需依判别式 $d = b^2 - c$ 的符号区分下列三种情况：

1° $d < 0$，无实根；

2° $d = 0$，有重根 $x_1 = x_2 = - b$；

3° $d > 0$，可用公式(3) 求得两个互异实根 x_1, x_2.

图 0-1 形象地描述了上面的算法.

这里我们使用了两种形式的框. 一种是矩形框，称**叙述框**. 计算公式就填在这种框内. 另一种是圆边框，称**检查框**，表示算法的判断检查部分. 检查框

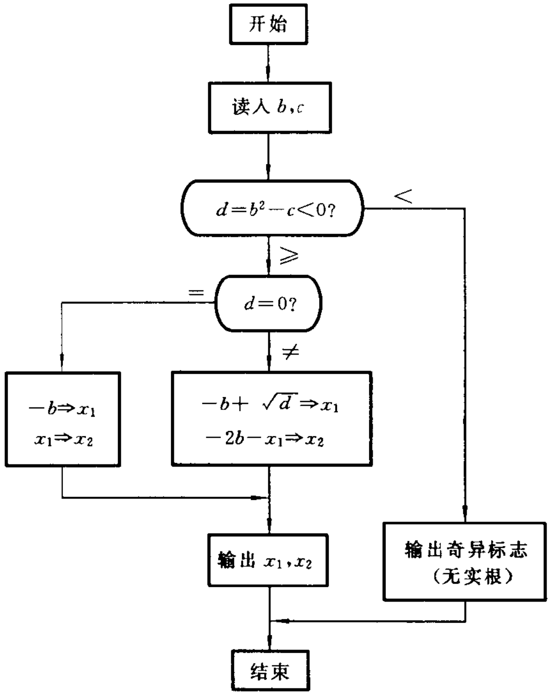

图 0-1

有两个出口,究竟选择哪个出口,要看框内的检查条件是否成立来决定.

今后所有的框图均以$\boxed{\text{开始}}$框标志计算过程开始启动,而用$\boxed{\text{结束}}$框表示计算过程的最终结束.另外,我们将用箭头"→"指明各框执行的顺序.

下面剖析两个常用算法来阐述算法的基本特征.

3. 多项式求值的秦九韶方法

计算公式通常是算法的核心部分.计算机上使用的算法,其计算公式常采取递推化形式.递推化的基本思想是将一个复杂的计算过程归结为简单过程的多次重复,这种重复在算法上表现为循环,描述是容易的.

譬如,设要对给定的 x 求下列多项式的值

$$p(x)=a_0+a_1x+a_2x^2+\cdots+a_nx^n \tag{4}$$

一种看起来很"自然"的算法是直接逐项求和.我们用 t_k 表示 x 的 k 次幂,u_k 表示式(4)右端前 $k+1$ 项的部分和:

$$t_k=x^k$$

$$u_k=a_0+a_1x+\cdots+a_kx^k$$

则

$$\begin{cases} t_k = x \cdot t_{k-1}, \\ u_k = u_{k-1} + a_k t_k, \end{cases} \qquad k = 1,2,\cdots,n \tag{5}$$

作为初值，令

$$\begin{cases} t_0 = 1 \\ u_0 = a_0 \end{cases} \tag{6}$$

利用初值(6)对 $k = 1,2,\cdots$，直到 n 反复执行算式(5)，最终得出的 u_n 就是所求的值 $p(x)$.

统计上述算法的计算量. 加减操作的机器运行时间比乘除操作少得多，在统计计算量时，我们可忽略加减法而只统计乘除法的次数. 递推公式(5) 的每一步需做两次乘法，因此总的计算量为 $2n$ 次乘法.

下面再介绍一种求值方案. 为此首先加工计算公式，设将式(4)按降幂的顺序重排为

$$p(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0$$

从它的前两项提出 x^{n-1}，则有

$$p(x) = (a_n x + a_{n-1}) x^{n-1} + a_{n-2} x^{n-2} + \cdots + a_1 x + a_0$$

经过这个手续，如果算出括号内的值，则问题归结为计算一个 $n-1$ 次多项式(注意降了一次). 再施行同样的手续，则进一步有

$$p(x) = ((a_n x + a_{n-1}) x + a_{n-2}) x^{n-2} + a_{n-3} x^{n-3} + \cdots + a_1 x + a_0$$

这样每做一步，所归结出的多项式就降低一次，最终可将所给计算公式(4)加工成如下嵌套形式

$$p(x) = (\cdots((a_n x + a_{n-1}) x + a_{n-2}) x + \cdots + a_1) x + a_0 \tag{7}$$

我们可利用式(7)结构上的特点，从里往外一层一层地计算. 设用 v_k 表示第 k 层(从里面数起)的值：

$$v_k = (\cdots(a_n x + a_{n-1}) x + \cdots + a_{n-k+1}) x + a_{n-k}$$

那么，第 k 层的结果 v_k 显然等于第 $k-1$ 层的结果 v_{k-1} 乘上 x 再加上系数 a_{n-k}：

$$v_k = x\, v_{k-1} + a_{n-k}, \qquad k = 1,2,\cdots,n \tag{8}$$

作为初值，令

$$v_0 = a_n \tag{9}$$

比较(5)～(6)和(8)～(9)两种算法，后一种不但逻辑结构简单，而且计算量节约了一半.

多项式求值的这种算法称作**秦九韶算法**，它是我国宋代大数学家秦九韶最先

提出的[①].

秦九韶算法的特点在于,它通过一次式的反复计算,逐步得出高次多项式的值.具体地说,它将一个 n 次多项式的求值问题,归结为重复计算 n 个一次式(8)来实现.这种化繁为简的处理方法在数值分析中是屡见不鲜的.

现在考虑秦九韶方法的计算程序.

按式(8)计算,每求出一个"新值"v_k 以后,"老值"v_{k-1} 便失去继续保存的价值,因此可以将新值 v_k 存放在老值 v_{k-1} 所占用的单元内.这样,我们只要设置一个单元 v 进行累算,而将式(8)表为下列动态形式

$$x \cdot v + a_{n-k} \Rightarrow v, \qquad k = 1,2,\cdots,n$$

执行这组算式之前,应先送初值 a_n 到单元 v 中

$$a_n \Rightarrow v$$

图 0-2 描述了秦九韶算法,其中:

[框 1]　**准备部分**.单元 v 中送初值 a_n,单元 k 中送计数值 1.

[框 2]　**计算部分**.每循环一次,单元 v 中的老值 v_{k-1} 为新值 v_k 所替换.

[框 3]　**控制部分**.检查单元 k 中计数值以判断循环应否结束.当计数值为 n 时输出 v 中结果,否则转框 4.

[框 4]　**修改部分**.修改单元 k 中计数值,然后转框 2 再作下一步的计算.

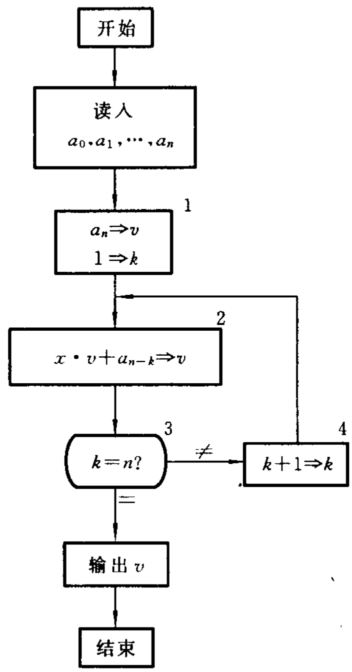

图 0-2

4. 方程求根的二分法

许多实际算法表现为某种无穷递推过程的截断,实现这类算法,不但需要建立计算公式,还需要解决精度控制问题.下述方程求根算法就是这类算法的一个范例.

设函数 $f(x)$ 在 $[a,b]$ 上连续,$f(a)f(b) < 0$,根据连续函数的性质,$f(x)$ 在 $[a,b]$ 内一定有实的零点,即方程 $f(x) = 0$ 在 $[a,b]$ 内一定有实根.这里假定它在 $[a,b]$ 内有唯一的单实根 x^*.

考察有根区间 $[a,b]$,取中点 $x_0 = (a + b)/2$ 将它分为两半,然后进行根的搜索,即检查 $f(x_0)$ 与 $f(a)$ 是否同号:若确系同号,说明所求的根 x^* 在 x_0 的右侧,这

① 外国文献称这一算法为**霍纳**(Horner)**算法**,其实霍纳的工作比秦九韶晚了五百多年.

时令 $a_1=x_0,b_1=b$；否则 x^* 必在 x_0 的左侧，这时令 $a_1=a,b_1=x_0$（图 0-3）. 不管出现哪一种情形，新的有根区间 $[a_1,b_1]$ 的长度仅为 $[a,b]$ 的一半.

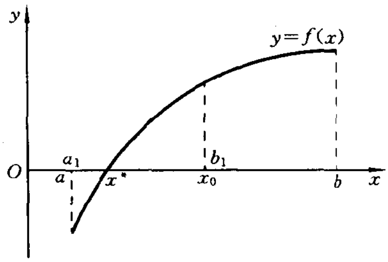

图 0-3

对于压缩了的有根区间 $[a_1,b_1]$ 又可施行同样的手续，即用中点 $x_1=(a_1+b_1)/2$ 将区间 $[a_1,b_1]$ 再分为两半，然后通过根的搜索判定所求的根在 x_1 的哪一侧，从而又确定一个新的有根区间 $[a_2,b_2]$，其长度是 $[a_1,b_1]$ 的一半.

如此反复二分下去，即可得出一系列有根区间

$$[a,b]\supset[a_1,b_1]\supset[a_2,b_2]\supset\cdots$$

其中每个区间都是前一个区间的一半，因此二分 k 次后的有根区间 $[a_k,b_k]$ 的长度

$$b_k-a_k=\frac{1}{2^k}(b-a)$$

可见，如果二分过程无限地继续下去，这些有根区间最终必收缩于一点 x^*. 该点显然就是所求的根.

第 k 次二分后，设取有根区间 $[a_k,b_k]$ 的中点

$$x_k=\frac{1}{2}(a_k+b_k)$$

作为根的近似值，则在二分过程中可以获得一个近似根的序列 $x_0,x_1,x_2,\cdots$，该序列以根 x^* 为极限.

不过在实际计算时，我们不可能也没有必要完成这种无穷过程，因为计算结果允许带有一定的误差. 由于

$$|x^*-x_k|\leqslant\frac{1}{2}(b_k-a_k)=\frac{1}{2^{k+1}}(b-a)$$

只要二分足够多次（即 k 充分大），便有

$$|x^*-x_k|<\varepsilon$$

这里 ε 为预定精度.

上述求根方法称为**二分法**，它是电子计算机上一种常用算法. 我们给出其算法框图（图 0-4），图中 a,b 表示有根区间的

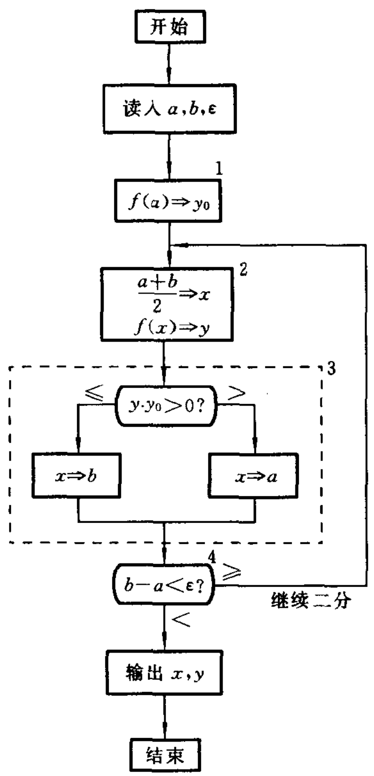

图 0-4

左、右端点；x 表示近似根.

图 0-4 中各框的具体含义如下：

[框 1] 从所给区间 $[a,b]$ 着手二分.

[框 2] 取有根区间 $[a,b]$ 的中点 x 作为近似根.

[框 3] 判定二分后生成的有根区间 $[a,b]$.

[框 4] 检查近似根 x 是否满足精度要求.

例 2 用二分法求方程 $x^3 - x - 1 = 0$ 在区间 $[1, 1.5]$ 内的一个实根，要求误差不超过 0.005 .

解 首先预估所要二分的次数. 按误差估计式

$$|x^* - x_k| \leqslant \frac{1}{2^{k+1}}(b-a)$$

只要二分 6 次，便能达到所要求的精度.

二分法的计算结果如下表：

表 0-1

k	a_k	b_k	x_k
0	1.000 0	1.500 0	1.250 0
1	1.250 0		1.375 0
2		1.375 0	1.312 5
3	1.312 5		1.343 8
4		1.343 8	1.328 1
5		1.328 1	1.320 3
6	1.320 3		1.324 2

§2 误　差

1. 误差分析不容忽视

在研究算法的同时，必须注重误差分析，否则，一个合理的算法也可能得出错误的结果.

例 3 用中心差商公式求 $f(x) = \sqrt{x}$ 在 $x = 2$ 的导数值：

$$f'(2) \approx \frac{\sqrt{2+h} - \sqrt{2-h}}{2h} \tag{10}$$

从理论上说，步长 h 愈小，计算结果愈准确. 上机计算的实际情况将会怎样呢？

解 我们知道，在计算机上数的表示受机器字长的限制，设取五位数字计算，若令 $h = 0.1$，得

$$f'(2) \approx \frac{1.4491 - 1.3784}{0.2} = 0.35350$$

与导数的精确值 $f'(2) = 0.353553\cdots$ 比较，这项计算还是可取的. 但是，如果缩小步长取 $h = 0.0001$，则得

$$f'(2) \approx \frac{1.4142 - 1.4142}{0.0002} = 0$$

算出的结果反而毫无价值.

这个例子告诫我们：两个相近的数相减，会造成有效数字的严重损失. 实际计算中要尽量避免这种情况发生.

例 4 求解方程 $x^2 - (10^5 + 1)x + 10^5 = 0$.

解 仍取五位数字进行计算，并用“$\triangleq$”标记对阶舍入的计算过程. 具体求根利用式(3)，这里 $c = 10^5$，而

$$b = -\frac{1}{2} \times (10^5 + 1) \triangleq -\frac{1}{2} \times 10^5$$

$$\sqrt{b^2 - c} = \sqrt{\left[-\frac{1}{2}(10^5 + 1)\right]^2 - 10^5} \triangleq \frac{1}{2} \times 10^5$$

故有

$$x_1 = -b + \sqrt{b^2 - c} \triangleq 10^5$$

$$x_2 = -b - \sqrt{b^2 - c} \triangleq 0$$

原方程的精确解显然是 $x_1 = 10^5, x_2 = 1$，可见上面求出的结果严重失真.

在计算机上，加减运算之前先要“对阶”，例 4 说明，对阶手续会造成大数“吃掉”小数的后果，因而实际计算时不宜用相差悬殊的两个数作加减运算.

例 5 考察方程组

$$\begin{bmatrix} 1 & \frac{1}{2} & \frac{1}{3} \\ \frac{1}{2} & \frac{1}{3} & \frac{1}{4} \\ \frac{1}{3} & \frac{1}{4} & \frac{1}{5} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} \frac{11}{6} \\ \frac{13}{12} \\ \frac{47}{60} \end{bmatrix}$$

其解为

$$x_1 = x_2 = x_3 = 1$$

解 这类方程组的求解有很大困难. 事实上，如果把系数舍入成三位小数：

$$\begin{bmatrix} 1.000 & 0.500 & 0.333 \\ 0.500 & 0.333 & 0.250 \\ 0.333 & 0.250 & 0.200 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1.830 \\ 1.080 \\ 0.783 \end{bmatrix}$$

则其解变成

$$x_1 = 1.09, \quad x_2 = 0.488, \quad x_3 = 1.49$$

由此看出，尽管系数改变不大，所求出的解却有很大出入，这类问题称作是**病态**的（参看第六章 §4）.

2. 误差的来源

提起数值分析，往往给人以不严格、不准确以至于不够完善的感觉. 其实，数值计算中的近似是正常的，计算误差是不可避免的.

我们知道，为要进行数值计算，首先必须将实际问题归结为数学问题，建立起合适的数学模型. 在建立数学模型时，通常总要加上许多限制，总要忽略一些次要因素，这样建立起的“理想化”的数学模型，虽然具有“精确”而“完美”的外表，其实只是客观现象的一种近似而粗糙的描述. 这种数学描述上的近似必然会产生误差.

另外，在数值计算的过程中不可避免地还会产生其他各种各样的误差，本书主要考察以下两种：

(1) *截断误差*

我们知道，许多数学运算（诸如微分、积分及无穷级数求和等）是通过极限过程来定义的，然而计算机上只能完成有限次的算术运算和逻辑运算，因此需要将解题方案加工成算术运算与逻辑运算的有限序列. 这种加工常常表现为无穷过程的截断，由此产生的误差通常称作**截断误差**.

譬如，指数函数 e^x 可展开为幂级数形式：

$$e^x = 1 + x + \frac{x^2}{2!} + \cdots + \frac{x^n}{n!} + \cdots$$

但用计算机求值时，我们不能得出右端无穷多项的和，而只能截取有限项计算

$$S_n(x) = 1 + x + \frac{x^2}{2!} + \cdots + \frac{x^n}{n!}$$

这样计算部分和 $S_n(x)$ 作为 e^x 的值必然会有误差，据泰勒(Taylor)余项定理，其截断误差为

$$e^x - S_n(x) = \frac{x^{n+1}}{(n+1)!}e^{\theta x}, \qquad 0 < \theta < 1$$

(2) *舍入误差*

计算过程中所用的数据可能位数很多，甚至是无穷小数，然而受机器字长的限制，用机器代码表示的数据必须舍入成一定的位数，这就会引起**舍入误差**. 每一步的舍入误差是微不足道的，但经过计算过程的传播和积累，舍入误差甚至可能会“淹没”所要的真解.

3. 误差限和有效数字

误差虽然不可避免，但人们总是希望计算结果能足够准确，这就需要估计误