数值分析的理论及其应用

G.M.菲利普斯　P.J.泰勒著

数值分析的理论及其应用

〔英〕G.M. 菲利普斯 P.J. 泰勒 著

熊西文 施吉林 张洪庆 战同胜 郜荣春 李盘林 郭秀玲 王希诚 译

上海科学技术出版社

Theory and Applications
of Numerical Analysis

G. M. Phillips, P. J. Taylor

Academic Press, Inc. 1973

数值分析的理论及其应用

〔英〕G.M.菲利普斯
P.J.泰　勒 著

熊西文 施吉林 张洪庆 战同胜
郜荣春 李盘林 郭秀玲 王希诚 译

上海科学技术出版社出版
（上海瑞金二路 450 号）
新华书店上海发行所发行　江苏扬中印刷厂印刷
开本 850×1156 1/32　印张 12.5　字数 412,000
1980 年 6 月第 1 版　1980 年 6 月第 1 次印刷
印数 1—10,000

书号：13119·821　定价：1.40 元

序　言

虽然数值分析这门课的教师和学生为着选择一本好的教材已经伤了很多脑筋，我们仍然相信，要找到一种适合的教材，还是极端困难的．我们这本书是数值分析的引论性教材，是为正在肄业中的数学家、计算机科学家、工程师和其他科学家们写的．整个教程大约分60讲，可放在一年级的微积分课程之后开设．本书的前几章包括部分分析课程的内容，那是为着重新激起读者对这些内容的回忆并提供本课程所必备的基础知识．我们并不假定读者已经有了矩阵代数的知识，因此，简短地介绍了包括矩阵的内容，这些内容对于学过矩阵代数基础的读者也可能有些帮助，它们也可以在本书的基础上配合着其他课程选用．

我们试图一开始就围绕着我们的主题给出符合逻辑的，内容上自给自足的发展，并在实际方法和数学理论上给以同样的强调．因此，不仅精细地陈述各种算法，并且通常给出定理的证明，以求达到融会贯通．我们相信数值分析在数学教学中是非常重要的．数值分析是许多重要的数学概念的源泉和应用场所．我们尽可能在全书中贯穿逼近这个主题并对之进行统一的处理．因此，介绍了不同形态的逼近论思想，并把它们用在非线性代数方程的求解、数值积分和微分方程各章中．

要成为一个好的数值分析家是不容易的，也正如其他数学分枝一样，必需具有高超的技巧和丰富的经验才可能达到从一般到特殊，从抽象到实践的飞跃．书中包括了大量经过加工的例题，就是为着发展读者的这种技能的．每章末尾给出的习题应看做是教材内容的扩充而不仅是检验读者的理解程度．

我们注意到那种不公平地对待没有较多机会去接触计算机的人的情况，因此，几乎全部有关校核例题或解算习题的计算工作，

都尽可能在数学表和台式计算机的帮助下进行．书中搜集了一些框图，用以加深对算法的理解，但却没有收入计算机程序，因为我们认为这样作并不能有效地促进人们之间的交流．当然，我们希望这些内容能鼓励人们去接近计算机并进行更多的进一步的计算．

本书中明显地，但也是故意地，省掉了一些内容．例如有限差分只讲了一点点，因为这些内容用得不多，而且也稍嫌枯燥．由于篇幅所限，书中没有收进对多项式方程的特殊处理，因为这些内容与我们进一步的主题没有太多的直接联系．令人遗憾的是，本书中没有包括特征值和偏微分方程的内容．因为有关内容的任何进一步的研究都要具有更多的关于矩阵代数的知识，而这是超过我们对读者程度的假定的．

（下为致谢部分，译略．）

G. M. Phillips
（安德鲁逊大学
应用数学系）

P. J. Taylor
（斯特林大学计
算机科学系）

1972年12月

目　录

第一章 绪 论

1.1 什么是数值分析?

数值分析是数学的一个分枝，它专门研究数学问题的数值解法，这包括方法的推导，对方法的描述以及对整个求解过程的分析. 我们下面先来看一下数学中的构造性方法，它是指明如何具体去构造数学问题的解. 例如，构造性地证明一个问题的解的存在性将不仅指出解的存在，还将描述其确定的方法. 用反证法证明解的存在性就不是构造性的方法(这种方法先假定解不存在，然后推出矛盾). 考虑下述简单的例子.

例 1.1 证明实系数二次方程

$$x^2+2bx+c=0 \tag{1.1}$$

当 $b^2>c$ 时至少有两个实根.

(1) 构造性证明. 对任意 x、b 和 c,

$$x^2+2bx+c=x^2+2bx+b^2+c-b^2=(x+b)^2+c-b^2,$$

因此当且仅当

$$(x+b)^2+c-b^2=0$$

时，x 是(1.1)的根，亦即

$$(x+b)^2=b^2-c.$$

将两端开方并注意 $b^2-c>0$ 可以推出当且仅当

$$x+b=\pm\sqrt{b^2-c}$$

时，x 是根，从而

$$x=-b\pm\sqrt{b^2-c}. \tag{1.2}$$

这样就证明了方程(1.1)有两个根，并且可根据(1.2)具体算出这两个根.

(2) 非构造性证明. 设

$$q(x)=x^2+2bx+c.$$

首先假设(1.1)没有根,从而 q 恒不为 0. 由于 q 是连续函数(见§2.2),我们可知对所有 x, 或者 $q(x)$ 是恒正的,或者 $q(x)$ 是恒负的. 根据对 b 和 c 所附加的条件,得到

$$q(-b)=b^2-2b^2+c=c-b^2<0,$$

因而 $q(x)$ 必需是恒负的. 但当 $|x|$ 很大时,

$$x^2>|2bx+c|,$$

因此当 $|x|$ 很大时 $q(x)>0$, 我们得到矛盾.

其次假设 q 仅有一个根. 则由 q 的连续性, 当 x 很大时必有 $q(x)>0$, $-x$ 很大时必有 $q(x)<0$. 或者相反,当 x 很大时 $q(x)<0$,当 $-x$ 很大时 $q(x)>0$,这和无论 x 取什么符号, 只要 $|x|$ 很大, 就有 $q(x)>0$ 的事实相矛盾的.

这种证明方法并未指出如何计算根. □

在很长一段时间内, "算法"一词被理解为数学问题的构造性算法. 仅在近代才给出算法的严格的数学定义. 这方面, A. M. Turing(1912—1954)的工作是重要的, 他以计算机的抽象概念为基础给出了算法的定义. 从本书的目的出发,我们将算法定义成数学问题构造性解法的一个完备而确切的描述. 要下形式定义的困难之一是要严格规定方法中容许那些运算. 在这本书中我们规定为加、减、乘、除等基本算术运算. 在一般情况下,在算法中的解答和方法不一定是数值的. 例如早期希腊几何学家研究的算法只使用圆规、直尺和铅笔. 这类算法的一个典型例子就是过给定点作一条直线的垂线. 在本书中我们将仅处理数值算法.

值得注意的是, 直至本世纪初, 数值分析都是以构造性方法为基础的. 早期的著名数学家, 例如 Newton (1642—1727), L. Euler(1707—1783)与 K. F. Gauss(1777—1855)曾提出许多重要的算法和构造性证明. 很有意思的事是, 虽然近年来计算器具的发展使算法的执行更容易了,而非构造性方法却发展了起来.

数字计算机设计的高速发展对数值分析有深远的影响. 在写

这本书的时候，计算机每秒可执行百万次算术运算，这在三十年前是完全不可想象的．甚至十年前，速度也只有每秒一千次．一些冗长而复杂的计算可以处理了，但造成误差的机会也大大增加了．用户对改进数值算法的要求远远超过对计算机改进的要求，因为除非他们掌握某些理论原理，否则他们就很难对自己的问题选择一种最好的算法．

虽然多数数值算法是为使用数字计算机而提出的，但是数值分析的研究对象与计算机程序设计、信息处理（或数据处理）是不同的．而当提出和使用数值算法时，需要有有效的用计算机进行计算的知识．计算机程序设计的问题是将算法编码成适合于计算机的形式(不一定是数字的)．信息处理的问题是组织计算机使其可以处理数据，这些数据不一定是数字形式的．计算机的基本知识可以参见 Phillips 与 Taylor 的书(1969)．

1.2 数值计算法

许多数学问题无法求出解的精确值，而只好寻求它的近似值．例如当结果不是有理数时，由于我们用算术运算只能得到有理数，从而只能求出它的近似值．自然我们希望，按算法提供的结果，其误差能小于我们允许的程度．通常精度越高，其计算量也越大，下一个例子可以表明，要求精度越高，所需要的算术运算次数也越多．

例 1.2 用分半法求平方根（求平方根更有效的算法在§7.6中叙述，这里只是用它来说明精度和运算次数的关系）．给定 $a>0$, $\varepsilon>0$, 我们要求 $\sqrt{a}$ 的近似值，使其误差不超过 ε．我们将假设 $a\geqslant 1$．算法的步骤如图 1.1 的框图所示．在计算过程中，每步都有两个实数 x_0 和 x_1，我们不断改变 x_0 和 x_1，保持 $x_0\leqslant\sqrt{a}\leqslant x_1$，并且重复的将区间 $[x_0, x_1]$ 减半．

因为 $1\leqslant\sqrt{a}\leqslant a$，开始时我们取 $x_0=1$ 与 $x_1=a$，我们计算区

间的中点 $x=\frac{(x_0+x_1)}{2}$ 并且决定 x 是否大于 $\sqrt{a}$，由于我们不知道 $\sqrt{a}$ 的值，因此我们将 x^2 与 a 相比较．如果 $x^2>a$，我们以 x 代替 x_1，否则以 x 代替 x_0．当 $x_1-x_0\leqslant 2\varepsilon$ 时计算停止，这时 $x-\sqrt{a}\leqslant\varepsilon$，其中 ε 是我们要求的精度．框图中的循环叫作迭代．当 ε 减小时，迭代的次数增加．同时 ε 减小时，必需提高算术运算的精度，否则判别 $x^2>a$ 可能会导致不正确的结果，以致 x_0 与 x_1 会由于有效位数不足而重合．如果我们的算术运算能达到任意精度，我们可以将 ε 取成充分小的正数，但不能将 ε 取成 0，因为那将导致无穷多次迭代(除非 $a=1$)．□

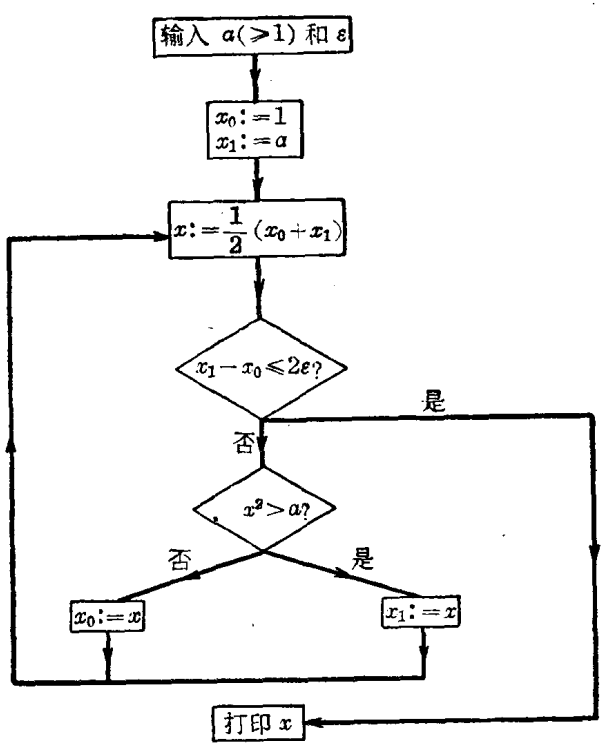

图 1.1　简单的求平方根算法[注]

[注]　符号：=读为赋值，其意思是"变成等于"．

在例 1.2 中我们计算出数列的前几个数，它收敛于$\sqrt{a}$（见§2.3）．由于我们只能计算数列中的前有限项，我们不能得到$\sqrt{a}$的精确值(除非$a=1$)．求得这类数列有许多算法，如果这个数列收敛于我们所要求的解，我们就把这个算法叫作收敛的．

当分析算法时我们将研究近似解的误差．误差是由各种原因产生的，例如算术运算中的舍入误差以及象例 1.2 中那样中断一个无穷过程产生的误差等等．误差常常是不可避免的，但从算法的使用方面看，则应力求避免因为错误而招致失败的那些困扰，这在使用计算机程序时更是如此．对每个算法我们将找出误差的界，即计算解时产生的误差的范围．如果这个界不直接依赖于解本身，可以事前算出，叫作先验误差界．如果这个界直接依赖于解本身，只有当解已基本上算出之后，才能算出这个界，这个界叫作事后误差界．误差的界常常远远的大于实际误差，从而不能对误差作出切合实际的估计．因此对某些问题我们提出各种不同的误差估计方法，误差的界和估计对我们研究误差的渐近特性和改进算法的近似程度是有重大意义的．

衡量误差大小有两种不同的方法，如果a是某个实数，a^*是a的近似值，我们定义$|a-a^*|$为绝对误差，$|a-a^*|/|a|$为相对误差，我们可以看出，如果对a的大小毫无了解，绝对误差的意义是不大的．我们常常给出百分比形式的相对误差．

研究算法的有效性是很重要的．我们常用为达到给定精度所需要的算术运算次数来量测其有效性．有时我们还要考虑计算机存储量等其它因素．

1.3 适定的与良好的问题

在运用数值方法解决具体问题之前，有一个重要的问题即在问题的表述中，当数据变化时，解的变化是否敏感？例如当我们解代数方程组时，方程的系数就是必须预先提供的数据，如果方程组系数只有小的变化但对解却有很大的影响，问题就复杂了．因为

这时即使用没有扰动的系数去进行计算，但仅由舍入而产生的误差也会使我们算不出令人满意的解.

对给定的数据 d 的集合，问题的解设为 $S(d)$，($S(d)$ 与 d 可以是数、函数、矩阵或它们的组合). 现在设 $d+\delta d$ 是数据的扰动集合并且 $S(d+\delta d)$ 是对应的解. 我们用非负数 $\|\delta d\|$ 表示扰动数据的大小，用非负数 $\|S(d+\delta d)-S(d)\|$ 表示相应解的差异的大小. 对给定的数据 d 的集合，如果它满足下述两个条件：

(i) 对数据 d 的集合及接近 d 的数据集合存在唯一解，即有实数 $\varepsilon>0$，对所有满足 $\|\delta d\|<\varepsilon$ 的 δd 存在唯一的解 $S(d+\delta d)$.

(ii) 解 $S(d)$ 连续依赖于数据 d，即当 $\|\delta d\|\to 0$ 时

$$\|S(d+\delta d)-S(d)\|\to 0,$$

则称对给定的数据集合，问题是*适定的*.

如果问题的解多于一个，我们就会不知道用数值方法求出的解是那一个解. 通常我们用补充条件的办法来使解唯一，例如解三次方程时，我们可规定要求最大的根.

如果条件 (ii) 不满足，尽管数据任意接近 d，而对应的解与 $S(d)$ 完全不同，这样用数值算法是很困难的.

如果对给定的数据集合，问题是适定的，并且对数据的每一小的扰动，解仅有小的不同，这个问题叫作*良好的*. 如果解的变化很大，这个问题叫作*病态的*，自然，这些术语也是相对的. 适定的解对其数据往往满足 Lipschitz 条件(见 §2.2). 在这种情况下有常数 $L>0$ 与 $\varepsilon>0$ 使对所有满足 $\|\delta d\|<\varepsilon$ 的 δd 有：

$$\|S(d+\delta d)-S(d)\|\leqslant L\|\delta d\|. \tag{1.3}$$

此时我们可以看出，如果 L 不很大，问题必然是良好的. 限制 $\|\delta d\|<\varepsilon$，表示在这个范围内(1.3)成立，只要 ε 不是很小，这个限制不是很重要的. 如果 (1.3) 中的 L 必须很大，或者这种 L 根本不存在，则数据的小的变化将会引起解的大的变化，这时问题是病态的.

我们前面已经提到，对病态问题，舍入误差对解可以有严重的影响，因此很难用数值方法求解. 这种情况下常常需要用高精度

的算术运算. 如果问题是不适定的,不管算术运算精度如何,舍入误差都可以使数值解变得没有意义. 有时数据本身就有误差,只能达到一定的精度,在这种情况下,结果的可靠程度将与问题的条件有关. 如果问题是病态的,结果中的误差可能变得很大,而如果问题是不适定的,结果就可能是毫无意义的.

下面的例子可以说明适定与良好, 这些性质依赖于问题和数据.

例 1.3 计算二次式

$$q(x)\equiv x^2+x-1150$$

的值. 当 x 接近二次式的根时, 这个问题是病态的. 例如, $x=100/3$ 时, $q(100/3)=-50/9\approx -5.6$. 如果我们取 $x=33$, 它与 $100/3$ 仅差 1%, 我们将求出 $q(33)=-28$, 其值差不多是 $q(100/3)$ 的 5 倍, 当 $x=100/3$ 时, 为使(1.3)成立,必需取 $L\approx 70$ (见问题 1.5). □

假设我们要解的问题是 P, 为了将 P 换成容易计算的问题, 我们用 P_1 近似地代替 P, P_1 是比较容易计算的. 这时会有两类误差. 首先是 P 和 P_1 之间的差, 用 E_P 表示这类误差. 这种误差常常是由于截断一个无穷过程引起的, 例如对无穷级数求和只取有限项就会产生这种误差(见 § 2.3),我们把这种误差叫*截断误差*. 第二类是 P 和 P_1 的解之间的误差, 我们用 E_S 表示, 这类误差有时叫作*整体误差*. E_P 与 E_S 间是有关系的,但是不能推出 E_P 很小时, E_S 也很小. 我们将对各种类型的问题导出 E_P 和 E_S 之间的关系, 对于一个好的方法, 当 E_P 趋近于 0 时, E_S 必须趋于 0. 如果这个性质成立, 我们将说这个算法是*收敛*的. 不幸的是, 在有些问题中,当 E_P 减少时 E_S 不趋于 0.

例 1.4 假定 f 是实变量 x 的实值函数, 我们将对区间 $a\leqslant x\leqslant b$ 中给定的点 x 求 f 的导数 f'. 假定 f 是非常复杂的函数, 我们只能计算 $f(x)$ 而不能精确的算出 $f'(x)$.

我们的算法是用容易微分的函数 g 去替换函数 f,用 $g'(x)$ 来代替 $f'(x)$. 在第五章中我们将指出, 对 $a\leqslant x\leqslant b$ 的所有 x, 只要

f 满足一定的条件，可以选择 g 使误差 $E_P(x)=f(x)-g(x)$ 任意小. 遗憾的是即使 $E_P(x)$ 任意小，$E_S(x)=f'(x)-g'(x)$ 还可能任意大. 例如对 g 的不同选择，令 $E_P(x)=\frac{1}{n}\sin n^2x$，$n=1, 2, 3, \cdots$，则对所有的 x

$$E_P(x)\to 0, \qquad \text{当 } n\to\infty \text{ 时.}$$

但是在 $x=0$ 处由于

$$E_S(x)=n\cos n^2x,$$

故当 $n\to\infty$ 时，$E_S(x)\to\infty$. □

当我们用容易计算的问题 P_1 近似地替换 P 时，我们自然希望 P 和 P_1 是适定的，并且在一定程度上是良好的. 如果 P 不是适定的，一般的说，P_1 的解不能收敛于 P 的解. 作 P_1 逼近时，P 的数据扰动的影响是必然的. 为了保证算法的收敛，我们总要求问题 P 是适定的. 若 P 不是良好的，那么收敛将是慢的.

我们在解 P_1 时也必须用数值计算，P_1 的数据也难免有扰动，因此要求 P_1 是适定的，并且是良好的. 若 P 是良好的，自然希望 P_1 也是良好的，如果不然，我们重新挑选 P_1，使它尽可能好一些.

在某些数值算法中，特别是对微分方程，我们要象例 1.2 那样，计算序列中的每一个元素，这时要考虑每个元素而不仅是它的极限. 在这些问题中我们将利用从前面元素推出后面元素的递推关系. 例如可以用下述递推关系

$$y_{n+1}=F(y_n, y_{n-1}, \cdots, y_{n-m}) \tag{1.4}$$

对 $n=m, m+1, m+2, \cdots$ 计算序列 $\{y_n\}_{n=0}^{\infty}$，其中 F 是 $m+1$ 个变量的已知函数. 我们需要给定起始值 $y_0, y_1, \cdots, y_m$. 在用 (1.4) 时将产生舍入误差，每一步的舍入误差都将影响所有的后继值，所以我们要研究舍入误差传播的影响. 这些影响将是十分重要的，特别是现代计算机能计算序列的很多项，可能会造成很大的舍入误差. 我们常用*稳定*这个词表示适定与良好.

例 1.5 如果 $\{y_n\}_{n=0}^{\infty}$ 满足