科学计算及其软件教学丛书/石钟慈主编

数值逼近

吴宗敏　苏仰峰◎编著

科学出版社
www.sciencep.com

科学计算及其软件教学丛书

数值逼近

吴宗敏　苏仰峰　编著

科学出版社
北京

内 容 简 介

本书是“科学计算及其软件教学丛书”之一，介绍数值逼近的基本理论、方法和应用. 主要内容包括：数值运算与误差、函数空间、插值与逼近、样条表示与插值、数值积分和非线性方程的求解等. 全书在一般理论讨论的基础上，尽可能给出可实现的 Matlab 程序，以适用于计算及实际问题的应用. 章后附有习题，可供练习.

本书可作为研究教学型大学、教学型大学计算数学与应用数学本科生的基础课程教材和参考书，也可供科学与工程计算的科技人员学习参考.

图书在版编目(CIP)数据

数值逼近/吴宗敏，苏仰峰编著. —北京：科学出版社，2008
(科学计算及其软件教学丛书)
ISBN 978-7-03-020179-9
I. 数… II. ①吴… ②苏… III. 数值逼近 IV. O174.41
中国版本图书馆 CIP 数据核字(2008) 第 006338 号

责任编辑：李鹏奇 李晓鹏 / 责任校对：陈玉凤
责任印制：张克忠 / 封面设计：耕者设计工作室

科学出版社 出版
北京东黄城根北街 16 号
邮政编码：100717
http://www.sciencep.com

北京市文林印务有限公司 印刷

科学出版社发行 各地新华书店经销
*
2008 年 2 月第 一 版 开本：B5(720×1000)
2008 年 2 月第一次印刷 印张：9
印数：1—4 000 字数：162 000

定价：18.00 元

(如有印装质量问题，我社负责调换〈文林〉)

《科学计算及其软件教学丛书》序

随着国民经济的快速发展，科学和技术研究中提出的计算问题越来越多，越来越复杂. 计算机及其应用软件的迅猛发展为这些计算问题的解决创造了良好的条件，而培养一大批以数学和计算机为主要工具，研究各类问题在计算机上求解的数学方法及计算机应用软件的专业人才也越来越迫切.

1998 年前后，教育部着手对大学数学专业进行调整，将计算数学及其应用软件、信息科学、运筹与控制专业合并，成立了 “信息与计算科学专业”. 该专业成立之初，在培养目标、指导思想、课程设置、教学规范等方面存在不少争议，教材建设也众说纷纭. 科学出版社的编辑曾多次找我，就该专业的教材建设问题与我有过多次的讨论. 2005 年 11 月在大连理工大学召开的第九届全国高校计算数学年会上，还专门讨论了教材编写工作，并成立了编委会. 在会上，编委会就教材编写的定位和特色等问题进行了讨论并达成了共识. 按照教育部数学与统计学教学指导委员会起草的 “信息与计算科学专业教学规范” 的要求，决定邀请部分高校教学经验丰富的教师编写一套教材，定名为 “科学计算及其软件教学丛书”. 该丛书涵盖信息与计算科学专业的大部分核心课程，偏重计算数学及应用软件. 丛书主要面向研究与教学型、教学型大学信息与计算科学专业的本科生和研究生. 为此，科学出版社曾调研了国内不同层次的上百所学校，听取了广大教师的意见和建议. 这套丛书将于今年秋季问世，第一批包括《小波分析》、《数值逼近》等十余本教材. 选材上强调科学性、系统性，内容力求深入浅出，简明扼要.

丛书的编委和各位作者为丛书的出版做了大量的工作，在此表示衷心的感谢. 我们诚挚地希望这套丛书能为信息与计算科学专业教学的发展起到积极的推动作用，也相信丛书在各方面的支持与帮助下会愈出愈好.

石钟慈

2007 年 7 月

前　言

数学经常将被研究真实世界的对象抽象成函数. 譬如, 天气预报中的温度分布、金融领域的股票指数、价格、GDP, 等等. 可以这么说, 函数是数学研究真实世界的最主要、最普遍的对象.

数值简单地说就是函数的值, 也可以是导数值或高阶导数值, 更为精确的数学定义应该是函数的泛函值, 用工程的语言就是利用某种测量手段获得的信息值.

数值逼近就是利用已经获得的信息来计算、描述被研究函数的函数值、导数值等 (需要了解的函数的其他泛函值). 数值逼近是计算数学的重要基础. 例如, 实际问题中的微分方程解, 人们往往只能获得其右端函数的某些测量值, 所以这是一个典型的数值逼近问题.

在数值逼近中, 由于已知信息的不完全性, 计算可能不是精确的. 譬如, 人们拿函数相近两点的差商近似地模拟函数的导数. 提请读者注意的是, 逼近 Approximation 这个词非常重要. 逼近不完全等同于近似, 逼近是一个过程, 数值逼近不仅要给出需要计算对象的一个近似算法, 而且当已知信息越来越多、越来越完备时, 将得到越来越精确的结果.

在数值分析中还有一对矛盾是计算方法的精确性与复杂性. 要达到高精度的结果往往需要较为复杂的计算, 由于计算机只能处理有限位的二进制数, 复杂的计算却会带来较大的累积误差. 本书的第 1 章就是解释这个现象, 引入计算误差的概念. 第 2 章是数值逼近的数学基础, 主要是函数空间、正交基等内容. 数值逼近很多是建立在函数逼近基础上的, 也就是利用已知信息, 尽可能地描述函数本身. 第 3 章介绍的是函数的插值与逼近, 主要是多项式插值与逼近. 第 4 章是样条函数的插值与逼近. 第 5 章讨论数值积分. 第 6 章是非线性方程的求解方法. 第 7、8 章可以作为选读内容, 分别讨论样条逼近的理论与推广的样条函数逼近及小波等有关内容, 这些都是现代及近年来出现的数值逼近新方法与新理论以及可能的发展方向. 所以, 建议应该尽可能地选读.

数值逼近是数值分析的一个重要内容, 有别于其他数学课程, 本课程将重点放在计算方法上, 而将较为理论的部分作为选读内容. 也就是要求学生在学完一段内容后, 应该自己编制程序来实现这个算法, 这对理解课程内容将会有较大的助益, 我们也建议教师在教授本课程时, 应该增加编程的内容. 本书的最后也附有一些

Matlab 的程序作为样本. 读者以及授课的教师也可以在 Matlab 的 Tool-box 中找到相关程序. 需要强调的是, 我们还是建议学生有自己亲手编程的经历, 所以在习题中也提出了这样的要求.

要再次强调的是, 希望尽可能地将选读内容作为授课内容, 从而提高读者数值逼近的理论基础.

作　者

2007 年 8 月

目　录

第 1 章　数值运算与误差……1
1.1　数值运算……1
1.2　误差及其来源……2
1.3　科学计算中应该考虑的问题……3
第 2 章　函数空间……5
2.1　多项式, Taylor 展开, Rolle 引理……6
2.2　正交基, 对偶正交基……9
习题 2……12
第 3 章　插值与逼近……13
3.1　多项式插值 (Euler、Lagrange)……13
3.2　差分与差商……16
3.3　多项式插值 (Newton、Neville-Aitken)……19
3.4　Hermitian 插值……23
3.5　多项式最小二乘逼近……25
3.6　Shepard 插值与运动最小二乘……27
习题 3……29
第 4 章　样条表示与插值……30
4.1　Bernstein-Bezier 多项式表示……30
4.2　分段多项式插值……33
4.3　样条表示与插值……36
习题 4……41
第 5 章　数值积分……43
5.1　Newton-Cotes 公式……43
5.2　复化 Newton-Cotes 公式……47
5.3　Gauss 型求积公式……51
5.4　特殊函数的数值积分……57
5.5　高维空间中的数值积分……59

习题 5 ····· 61
第 6 章　非线性方程的求解 ····· 63
6.1　二分法 ····· 63
6.2　不动点迭代 ····· 65
6.3　牛顿法及割线法 ····· 66
6.4　非线性方程组的求解 ····· 70
习题 6 ····· 71
*** 第 7 章　样条逼近的进一步讨论** ····· 73
7.1　B 样条函数基 ····· 73
7.2　等距节点上的样条 ····· 77
7.3　样条函数插值及奇次样条函数插值的最优性质 ····· 81
*** 第 8 章　推广的样条表示与插值** ····· 84
8.1　Tschebycheffian 系与推广的 Taylor 展开、Rolle 引理 ····· 84
8.2　推广的样条与推广的 B 样条 ····· 90
8.3　推广 B 样条的插值、递推算法 ····· 94
8.4　多项式再生, 逼近阶 ····· 97
8.5　等距节点、细分算法、小波、主样条 ····· 103
习题 8 ····· 118
第 9 章　程序代码 (例) ····· 119
参考文献 ····· 132

第 1 章　数值运算与误差

1.1　数 值 运 算

1.1.1　数的浮点表示及浮点运算

一个数在计算机中是以二进制的形式表示的. 在本书中我们不准备深入展开这方面的讨论. 为了简单起见, 假定数在计算机中是以我们熟悉的十进制表示的. 我们来看 $-\frac{10}{3}$ 在计算机中是如何表示的.

首先计算机不能用分数或者无限位小数来表示这个数, 只能用有限位来表示, 如 8 位十进制, 这时 $-\frac{10}{3}$ 表示为 -3.3333333. 其次为了能存储更大或者更小的数, 我们将数用科学计数法来表示, 这样 $-\frac{10}{3}$ 就表示为

$$-0.33333333 \times 10^{+1}.$$

在计算机中存储下面的信息: 一个负号, 8 个 3, 一个 1, 以及 1 前面的 + 号. 注意, 小数点以及小数点前面的 0 是不用存储的, 表示十进制的 10 也不用存储 (在计算机里是二进制的 2). 这样的表示方式称为浮点表示. 这里要强调的是, 由于数总是用有限位来表示的, 几乎所有的数在用浮点数来表示时都有误差. 这里的误差为

$$-\frac{10}{3} - (-0.33333333 \times 10^{+1}) \neq 0,$$

这个误差称为舍入误差.

我们再来看两个数在计算机中是如何运算的. 我们考虑 $\frac{1}{3} - \frac{1}{30} = 0.3$. 首先将两个数表示为浮点数 $fl\left(\frac{1}{3}\right) = 0.33333333 \times 10^{0}$, $fl\left(\frac{1}{30}\right) = 0.33333333 \times 10^{-1}$. 然后进行运算, 计算得到的结果是 $0.299999997 \times 10^{0}$, 这时需要进行舍入运算, 得到的结果是 0.30000000×10^{0}, 正好是我们要得到的结果.

1.1.2　舍入误差的影响

在实际进行科学计算时, 我们要对很多的浮点数进行操作, 这时舍入误差会积累. 有时积累的效果会使计算结果完全错误, 有时计算结果和实际的结果又是相符的. 我们以两个例子来说明舍入误差对计算结果的影响.

例 1.1　$10^{10} + 1 - 10^{10} = 1$. 在计算机中运算时, 前面两个数进行运算, 然后进行舍入, 得到的结果是 10^{10}, 再进行减法运算, 得到的结果是 0.

例 1.2　考虑两个函数：第一个函数是

$$f(x) = (x-2)^9,$$

第二个函数是将 $f(x)$ 进行二项展开

$$g(x) = x^9 - \mathrm{C}_9^1 2x^8 + \mathrm{C}_9^2 2^2 x^7 - \mathrm{C}_9^3 2^3 x^6 + \mathrm{C}_9^4 2^4 x^5 - \mathrm{C}_9^5 2^5 x^4 + \cdots.$$

这两个函数在理论上应该是相等的. 在图 1.1 中, 我们对 $[1.93, 2.07]$ 中的 1000 个点进行求值. 我们可以看到, 用 $f(x)$ 计算得到的值很准确, 而用 $g(x)$ 计算得到的值连正负号都不能保证.

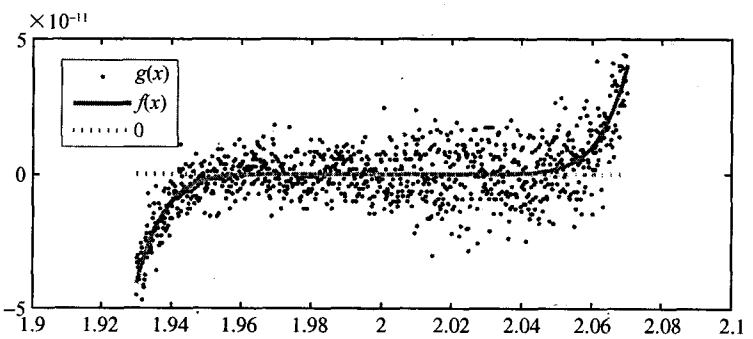

图 1.1　多项式计算中舍入误差的影响

1.2　误差及其来源

对于一个数 x, 假定 $\hat{x}$ 是其计算得到的值, 其误差为

$$e = \hat{x} - x.$$

误差越小说明 $\hat{x}$ 越精确. 在几乎所有的情况下, x 是没有办法得到的 (否则我们还要 $\hat{x}$ 干什么?), 从而 e 也没有办法得到. 但是我们通常感兴趣的是 $|e|$ 不会超过多少, 也就是误差的界; 有时我们也对误差的符号感兴趣, 而其真实值则是不可能得到的. 这个误差界通常可以通过理论分析得到.

对于两个值 $x_1 = 10^{10}, x_2 = 1$, 如果经过估计它们计算得到的值 $\hat{x}_1, \hat{x}_2$ 的误差均不超过 10, 很显然这个估计对于 x_1 有效, 而对于 x_2 无效. 我们定义相对误差为

$$\eta = \frac{|x - \hat{x}|}{x}.$$

对于 x_1, $\eta \leqslant 10^{-9}$, 而对于 x_2, $\eta \leqslant 10$, 所以 $\hat{x}_1$ 和 $\hat{x}_2$ 虽然绝对误差界相同, 但是相对误差相差很多.

同样, 由于得不到 x, 我们用

$$\hat{\eta} = \frac{|x - \hat{x}|}{|\hat{x}|}$$

来替代 η, 在实践中我们能够得到的也只能是 $\hat{\eta}$ 的一个上界.

例如, 对于区间 $[a, b]$ 上的连续函数 $f(x)$, 满足 $f(a) \cdot f(b) < 0$. 如果用 $c = \dfrac{a+b}{2}$ 来近似代替 f 在 $[a, b]$ 上的零点 x^*, 则误差界为 $|c - x^*| < \dfrac{b-a}{2}$, 若 $c \neq 0$, 则相对误差界为 $\hat{\eta} < \dfrac{b-a}{|b+a|}$.

完整地求解一个问题, 其误差有多个来源. 我们考虑一个实际而简单的问题: 一个弹性系数为 k 的弹簧, 其上端固定, 下端悬挂一个质量为 m 的物体, 忽略空气阻力, 则其运动方程为

$$m\frac{\mathrm{d}^2x}{\mathrm{d}t^2} + kx = f(t), \quad x(0) = x_0, \quad \left.\frac{\mathrm{d}x}{\mathrm{d}t}\right|_{t=0} = v_0,$$

其中, x 为物体距离平衡位置的偏移量, $\dfrac{\mathrm{d}x}{\mathrm{d}t}$ 为速度, $\dfrac{\mathrm{d}^2x}{\mathrm{d}t^2}$ 为加速度, $f(t)$ 为作用在物体上的外力. 则用计算机求解得到的 $\hat{x}(t)$ 与真正的偏移量之间一定会有误差. 误差的来源包括:

(1) 测量误差. 包括质量 m 和弹性系数 k 中的误差.

(2) 模型误差. 我们忽略了空气阻力, 而事实上阻力总是存在的; 即使将空气阻力考虑进模型, 如何考虑阻力项也是一个问题.

(3) 离散误差. 对二阶常微分方程可以采用多种方法求解. 例如, Euler 方法, 这时可能的误差有数值导数的误差、数值积分的误差等.

(4) 计算误差. 在计算机的浮点运算中产生的舍入误差及其累积.

1.3　科学计算中应该考虑的问题

当我们决定用数值方法来求解一个问题时, 我们将一个完整的问题分成好几个小的问题. 每个小问题都有好几种算法可供选择. 我们究竟选取哪些算法呢? 或者说评价一个算法好坏的标准是什么? 我们可以通过下面几个因素来考虑:

(1) 精确性. 一个算法能够被你使用, 首先它必须能够使计算结果达到你要求的精度, 也就是结果的误差要在你允许的范围内.

(2) 快速性. 如果精确性能够达到要求, 当然算法越快越好; 求解一个小问题的时间是 0.1 秒或者 0.01 秒可能不会产生太大的区别, 但是如果在求解一个大问题的过程中需要反复求解这个小问题, 这两个算法的差别就是巨大的.

(3) 存储量. 算法的存储量是指计算过程中变量所占计算机内存的大小. 在当代的科学计算中, 有的问题规模非常巨大, 存储一个向量就需要数十兆字节的内存.

一般说来, 不存在某个算法在这三个方面同时达到最优. 我们有时可以用快速性和存储量进行互换. 例如, 我们可以适当增加冗余运算来减少存储量, 或者增加计算量来提高精度等.

另外, 对于一个给定的算法, 编程的不同会造成程序效率不同, 有时会相差数十倍. 所以, 我们希望读者能够使用合适的计算机程序语言, 如 Matlab, FORTRAN, 或者 C (C++) 等将本书中的算法及例子加以实现, 一方面可以通过实践来提高编程水平, 另一方面可加深对书中算法的理解.

第2章 函数空间

定义 2.1 一组函数 X, 在加、减、数乘运算下封闭, 就称这组函数构成一个线性的函数空间. 为了简化, 通常简单地称作函数空间.

显然, 零函数一定是函数线性空间的元素, 因为 $f(x)-f(x)$ 或 $0*f(x)$ 是这个函数空间中的元素.

定义 2.2 如果函数空间 X 中存在函数 $\{b_1(x),\cdots,b_n(x)\}$, 函数空间 X 中任何函数都可以表示成它们的线性组合

$$X=\{f(x)|f(x)=\sum\lambda_j b_j(x)\},$$

那么称 $\{b_1(x),\cdots,b_n(x)\}$ 是函数空间 X 的一个框架. 进一步地, 如果 $\{b_1(x),\cdots,b_n(x)\}$ 还线性无关, 即当且仅当 $\{\lambda_j\}$ 都是 0 时, 才有 $\sum\lambda_j b_j(x)=0$, 那么称这个框架还是一组基 (或者称基底). 基的个数称为这个空间的维数. 这时可以证明 X 的任何一组基底的个数都是一样的, 即维数是唯一确定的.

注 2.1 基底一般不是唯一确定的, 也就是说, 基底有多种选取方法. 所以, 选一个什么样的函数空间来表示我们要研究的对象函数是一个非常重要的问题, 如何在一个空间中寻找一组好的基底也是一个非常有意义的问题. 它可能影响计算的稳定性与复杂性.

定义 2.3 我们知道, 函数就是将数映照到数的一种变换, 称将函数映照到数的变换为泛函. 如果泛函 L 对任何的函数 $f(x)$,$g(x)$ 及数 α,β 满足

$$L(\alpha f(x)+\beta g(x))=\alpha Lf(x)+\beta Lg(x),$$

那么称之为线性泛函.

例 2.1 函数在某点取值, 函数在某点取导数值或高阶导数值、函数在某区域求积分等都是线性泛函. 譬如, $Lf(x)=f(0)$, $Lf(x)=f'(1)$, $Lf(x)=\int_0^1 f(x)\mathrm{d}x$, $Lf(x)=f(0)-3f'(1)+\int_0^1 \mathrm{e}^x f(x)\mathrm{d}x$ 都是线性泛函. 读者可以自行验证这些泛函的线性性质.

数值逼近的根本问题是: 如果已知某个函数的一些泛函值 (通过某种、某几种测量手段获得的一些测量值), 要计算 (或者估计) 这个函数的其他泛函值 (无法测量或者还没有测量到的值).

数值逼近的基本思想是：针对目标函数，根据已知的函数信息，在一个简单的函数空间中，用一个简单的函数作为逼近或近似，然后用这个逼近或近似的较容易计算的泛函值来模拟、近似需要求得的目标函数本身的泛函值.

一个简单的例子是：如果已知 $\{f(x_1), f(x_2)\}$,$\{x_1, x_2\}$ 靠得很近，那么可以用一次多项式或直线 (割线) 来逼近函数

$$f(x) \sim f(x_1)\frac{x_2 - x}{x_2 - x_1} + f(x_2)\frac{x - x_1}{x_2 - x_1}.$$

对这个逼近函数求导推得，在 $x \in (x_1, x_2)$,

$$f'(x) \sim \frac{f(x_2) - f(x_1)}{x_2 - x_1},$$

从而上述公式的右端可以作为函数 $f(x)$ 的导数在 $x \in [x_1, x_2]$ 的一个数值逼近.

数值逼近首先要解决的问题是：应该采用什么样的函数空间来作为逼近空间？或者说什么是简单的函数空间？然后，当然是应该在这个函数空间选择一组什么样的基？

2.1　多项式, Taylor 展开, Rolle 引理

我们最熟悉也可以说是最简单的函数空间应该是多项式函数空间.　称由 $\{1, x, \cdots, x^n\}$ 张成的函数线性空间为 n 次多项式函数空间，记为 $\mathcal{P}_n$. 由代数基本定理，容易验证，$\{1, x, \cdots, x^n\}$ 线性无关；否则，如果有不全为零的系数 $\{\lambda_j\}_{j=0}^n$,

$$\sum_{j=0}^{n} \lambda_j x^j = 0,$$

那么在两两不同的 $n+1$ 个点 $\{x_k\}_{k=0}^n$ 上

$$\sum_{j=0}^{n} \lambda_j x_k^j = 0,$$

这个线性方程组的系数行列式是一个 Vandermonde 行列式. 从而非奇异，奇次方程不可能存在非零解，因此导得所有的 λ_j 都应该为零，这与反证假设矛盾. 同时我们得到，$\mathcal{P}_n$ 的维数是 $n+1$. 在这个函数空间中，函数 x 非常重要. 事实上，多项式函数空间可以由 x 的幂次及线性组合得到，这在计算机发展的时代特别重要，因为事实上计算机的基本运算只有加、减、乘及判别运算. 其他引导出来的运算都可能带来较大的数值误差.

在多项式函数空间中还有一个函数 $g(x) = x^n$ 具有特殊的地位. 可以用下面的两个引理来说明.

引理 2.1　$\{g(x), g'(x), \cdots, g^{(n)}(x)\}$ 张成 $\mathcal{P}_n$.

证明　这个引理的证明是平凡的. □

引理 2.2　如果 $\{x_0, x_1, \cdots, x_n\}$ 两两不同, 那么 $g(x)$ 的平移

$$\{g(x-x_0), g(x-x_1), \cdots, g(x-x_n)\}$$

也张成 $\mathcal{P}_n$.

证明　用反证法. 如果 $\sum_{j=0}^{n} c_j g(x-x_j) = 0$, 两项展开 $g(x-x_j)$, 分析 x^k 前的系数, 有 $\sum_{j=0}^{n} c_j x_j^k = 0, k = 0, \cdots, n$. 这个线性方程组的系数行列式是我们熟知的 Vandermonde 行列式, 从而只有零解. 这与反证假设矛盾, 所以 $\{g(x-x_0), g(x-x_1), \cdots, g(x-x_n)\}$ 线性无关, 张成空间的维数是 $n+1$, 也就是说, 张成的就是 $\mathcal{P}_n$. □

定义 2.4　如果函数 $g(x)$ 满足上述两个引理, 那么称 $g(x)$ 是 $\mathcal{P}_n$ 的一个生成子.

可以这么认为, 具备生成子的函数空间是比较简单的, 这样的函数空间事实上由一个函数生成 (通过平移及线性组合或求导及线性组合). 一般地, 不加证明可以给出如下定理, 上述两个引理是该定理的两个特殊情形.

定理 2.1　如果 $\{x_1, \cdots, x_l\}$ 两两不同, 对于 x_j 定义一个重数 γ_j, 满足 $\sum \gamma_j = n+1$, 那么 $g(x)$ 是 $\mathcal{P}_n$ 生成子的充分必要条件是

$$\{g(x-x_1), g'(x-x_1), \cdots, g^{(\gamma_1-1)}(x-x_1), \cdots, g(x-x_l), \cdots, g^{(\gamma_l-1)}(x-x_l)\}$$

张成 $\mathcal{P}_n$.

注 2.2　上述定理证明的关键是: 重节点可以看成是两两不同点的极限. 譬如, $\Big\{g(x-x_0), g(x-x_1)\Big\}$ 与 $\Big\{g(x-x_0), \dfrac{g(x-x_1)-g(x-x_0)}{x_1-x_0}\Big\}$ 张成相同的空间. 取极限就得到空间 $\{g(x-x_0), g'(x-x_0)\}$. 也可以这么认为, 导函数是函数平移的线性组合的极限形式.

例 2.2　在空间 $\mathcal{P}_n$, 事实上只要 $g(x)$ 是一个非退化的 n 次多项式, 那么 $g(x)$ 就是 $\mathcal{P}_n$ 的一个生成子. 特别地

$$(x-x_0)^n, \quad \prod_{j=0}^{n}(x-x_j), \quad \prod_{j=1}^{l}(x-x_j)^{\gamma_j}$$

都是 $\mathcal{P}_n$ 的一些特殊的生成子.

讨论生成子的目的是: 人们可以通过对一个函数 (生成子) 的平移、求导、线性组合等运算张成整个多项式函数空间. 这样的函数空间有计算机存储及表示简单

的优点, 人们只要存储一个函数, 然后平移获得空间的一组基, 从而张成整个函数空间. 在计算机快速发展的时代, 这样的性质尤为重要.

定理 2.2 (Taylor)　如果 $f(x)$ 的 $n+1$ 阶导数连续, 且 $f^{(n+1)}(x)$ 有界, 那么

$$f(x)=\sum_{j=0}^{n} f^{(j)}(x_0)(x-x_0)^j/j!+\mathcal{O}(x-x_0)^{n+1}.$$

这是熟知的 Taylor 展开式. Taylor 定理告诉我们: 已知 $\{f^{(j)}(x_0)\}_{j=0}^n$, 那么在点 x_0 的附近, 我们可以用 Taylor 展开式 $\sum_{j=0}^{n} f^{(j)}(x_0)(x-x_0)^j/j!$ 模拟、逼近函数 $f(x)$.

Taylor 展开的另一种表示是: 对于生成子 $g(x)=x^n/n!$, 在 x_0 附近, 函数

$$f(x)\sim\sum f^{(j)}(x_0)g^{(n-j)}(x-x_0).$$

引理 2.3 (Rolle)　如果 $f(x)\in C^n$, $\{f(x_j)=0\}_{j=0}^n$, 那么存在 $\xi\in(x_0,x_n)$, 使得 $f^n(\xi)=0$.

这个定理已经在数学分析中出现过, 它将在第 3 章被用来估计多项式插值的误差.

定理 2.3　如果 $\{x_j\}_{j=0}^n\in[a,b]$ 两两不同, $f(x)$ 在 $[a,b]$ 上 $n+1$ 次可导且 $\{f(x_j)=0\}_{j=0}^n$, 那么在 $[a,b]$ 上

$$\|f(x)\|_\infty<(b-a)^{n+1}\|f^{(n+1)}(x)\|_\infty.$$

证明　逐次利用 Rolle 引理, 在区间 (a,b), 存在 $\{\xi_{jk}\}_{k=0}^{n-j}$, 使得 $f^{(j)}(\xi_{jk})=0$. 对于误差估计利用数学归纳法, 有

$$|f^{(n+1)}(x)|\leqslant\|f^{(n+1)}(x)\|_\infty,$$

如果在 $[a,b]$ 上

$$\|f^{(j+1)}(x)\|_\infty<(b-a)^{n-j}\|f^{(n+1)}(x)\|_\infty,$$

那么可以在 x 的附近找到 ξ_{jk},

$$|f^{(j)}(x)|=\left|\int_{\xi_{jk}}^{x} f^{(j+1)}(t)\mathrm{d}t\right|<(b-a)^{n-j+1}\|f^{(n+1)}(x)\|_\infty,$$

所以

$$\|f^{(j)}(x)\|_\infty<(b-a)^{n-j+1}\|f^{(n+1)}(x)\|_\infty.$$　□

注 2.3　当 $\{x_j\}$ 有重节点时, 上述定理也是对的. 特别地, 当所有的节点都重合时, 就是 Taylor 展开式的余项公式.