

数值分析笔记

Python version

Jiaqi Z.¹

2023 年 11 月 25 日

¹Copyright ©2023 Jiaqi Z. All rights reserved.

目录

说明	iii
0.1 关于本笔记的版权与使用说明	iii
0.2 创作者名单	iii
1 绪论	1
1.1 误差	1
1.1.1 误差来源与分类	1
1.1.2 误差概念	2
1.1.3 相对误差限和有效数字的关系	5
1.2 数值运算的误差估计	6
1.2.1 四则运算误差估计	6
1.2.2 函数值误差估计	7
1.3 算法数值稳定性	8
1.4 数值计算中应该注意的一些原则	11
1.4.1 避免两相近数相减	11
1.4.2 避免除数绝对值远小于被除数绝对值	11
1.4.3 避免大数“吃”小数	11
1.4.4 简化计算步骤, 避免误差积累	13
2 插值法	17
2.1 引言	17
2.2 Lagrange 插值法	19
2.2.1 线性插值	19
2.2.2 抛物插值	19

2.2.3	Lagrange 插值多项式	20
2.2.4	插值余项	22
2.2.5	Lagrange 插值优缺点	24
2.3	Newton 插值	25
2.3.1	Newton 插值	25
2.3.2	差商	26
2.3.3	Newton 插值余项	28
2.4	等距节点差商公式	28
2.4.1	差分定义	28
2.4.2	差分的性质	29
2.4.3	等距节点插值公式	30
2.5	Hermite 插值	31
2.5.1	引入, Hermite 插值多项式的存在唯一性	31
2.5.2	Hermite 插值多项式构造	31
2.6	分段低次插值	35
2.6.1	多项式插值的 Runge 现象	35
2.6.2	分段线性插值	36
2.6.3	分段三次 Hermite 插值	39
2.7	三次样条插值	41
2.7.1	三次样条函数	41
2.7.2	三次样条插值函数构造	41
2.7.3	三转角方程	43
2.7.4	三弯矩方程	45
2.7.5	三次样条插值函数的收敛性 *	48
3	函数逼近与计算	53
3.1	引言	53
3.1.1	函数逼近的问题的一般提法	53
3.1.2	常用的度量标准	53
3.2	最佳一致逼近	54
3.2.1	最佳一致逼近概念	54
3.2.2	最佳一致逼近多项式的存在性	54
3.2.3	$C[a, b]$ 上最佳一致逼近	54
3.2.4	相关概念	55

3.2.5	$C[a, b]$ 上的最佳一致逼近特征	56
3.2.6	一次最佳逼近多项式 ($n = 1$)	57
3.2.7	Chebyshev 多项式及其应用	59
3.3	最佳平方逼近	63
3.3.1	内积空间	63
3.3.2	相关概念	64
3.3.3	内积空间上的最佳平方逼近	65
3.3.4	连续函数的最佳平方逼近	68
3.4	正交多项式	71
3.4.1	正交化手续	71
3.4.2	正交多项式的性质	71
3.4.3	常用的正交多项式	72
3.5	函数按正交多项式展开	74
3.6	曲线拟合的最小二乘法	77
3.6.1	问题提出	77
3.6.2	曲线拟合的步骤	77
3.6.3	2-范数度量下的曲线拟合 (最小二乘法)	77
3.6.4	用正交函数作最小二乘拟合	80
4	数值积分	85
4.1	引言	85
4.1.1	数值积分的必要性	85
4.1.2	数值积分的基本思想	86
4.1.3	求积公式的代数精度	87
4.2	插值型求积公式	88
4.2.1	定义	88
4.2.2	截断误差与代数精度	89
4.3	Newton-Cotes 公式	89
4.3.1	Cotes 系数	89
4.3.2	Newton-Cotes 公式	90
4.3.3	几种常见的低阶求积公式	92
4.3.4	复化求积公式	93
4.4	Romberg 算法	95
4.4.1	Romberg 求积公式	95

4.4.2	理查德森外推加速法	96
4.4.3	复化梯形公式的渐进展开式	96
4.5	Gauss 型求积公式	97
4.5.1	高斯型求积公式的构造	98
4.5.2	几种常见的 Gauss 型求积公式	100
4.5.3	Gauss 公式的余项	102
4.5.4	Hermite 多项式的余项	103
4.5.5	Gauss 型求积公式的收敛性	103
4.5.6	Gauss 型求积公式的数值稳定性	103
4.5.7	复化两点 Gauss-Legendre 求积公式	104
4.5.8	复化三点 Gauss-Legendre 求积公式	104
5	线性方程组的直接解法	105
5.1	Gauss 消去法	105
5.1.1	三角形方程组回代法	105
5.1.2	顺序 Gauss 消去法	106
5.1.3	Gauss 主元素消去法	108
5.2	解三对角方程组的追赶法	114
5.3	矩阵的三角分解法	114
5.3.1	Gauss 消元法矩阵形式	114
5.3.2	矩阵三角分解的定义	116
5.3.3	矩阵三角分解的存在性	116
5.4	Gauss 消去法变形	122
5.4.1	矩阵 LDR 分解	122
5.4.2	平方根法	123
5.4.3	改进平方根法	124
5.5	线性方程组的性态和解的误差估计	125
6	线性方程组的迭代解法	127
6.1	一般迭代法	127
6.1.1	迭代格式构造	127
6.1.2	迭代法收敛条件	128
6.1.3	迭代法的误差估计	129
6.2	Jacobi 迭代法	129

6.3	Gauss-Seidel 迭代法	131
6.4	Jacobi 迭代法和 Gauss-Seidel 迭代法收敛性	132
6.5	超松弛法	135
7	常微分方程数值解法	139
7.1	引言	139
7.1.1	初值问题的数值解法	139
7.1.2	初值问题解存在唯一性	140
7.1.3	初值问题的离散化方法	140
7.2	Euler 方法	141
7.2.1	Euler 公式	141
7.2.2	隐式 Euler 法 (向后 Euler 法)	142
7.2.3	梯形公式	142
7.2.4	中点 Euler 公式	143
7.2.5	预测校正法	143
7.3	Runge-Kutta 方法	144
7.3.1	二阶 Runge-Kutta 格式精度	145
A	矩阵分析基础	149
A.1	向量范数	149
A.1.1	向量范数的定义	149
A.1.2	常用的向量范数	149
A.1.3	向量范数性质	150
A.2	矩阵范数	151
A.2.1	矩阵范数的定义	151
A.2.2	常用的矩阵范数	152
A.2.3	矩阵范数与特征值之间的关系	153
A.2.4	矩阵的条件数	155
A.3	初等矩阵	156
A.3.1	初等矩阵	156
A.3.2	初等下三角矩阵	156
B	常见 Runge-Kutta 公式	159
B.1	二阶 Runge-Kutta 方法	159

B.1.1	二阶中点方法	159
B.1.2	二阶 Hune 方法	160
B.2	三阶 Runge-Kutta 方法	160
B.2.1	三阶 Kutta 方法	160
B.2.2	三阶 Hune 方法	161
B.3	四阶 Runge-Kutta 方法	161
B.3.1	四阶经典 Runge-Kutta 方法	162

说明

0.1 关于本笔记的版权与使用说明

- 本笔记可免费用于学习, 科研等非商业活动;
- 可以以非商业目的进行传播, 但在传播过程中必须保证笔记内容的完整性 (截止到 GitHub 仓库¹最新发布时, ”笔记” 包括但不限于仓库内笔记 Latex 源码, pdf 文件, 演示程序代码等. 下同), 需保证作者信息完整, 不得进行修改;
- 本笔记不可用于任何商业用途 (如确有需要, 需联系作者);
- 除在 GitHub 仓库以 pull request 形式进行编辑修改外, 不允许对笔记进行修改并公开传播私自修改版本 (以 GitHub 仓库版本为标准版本);
- 本笔记著作权归作者 (Jiaqi Z.) 所有, 对本笔记进行创作的人员也可获得著作权, 其他著作权所有者不得违反上述版权说明;
- 本笔记如因违反上述说明传播而造成不良影响, 与作者和其他创作者无关, 特此声明;

以上说明解释权归 Jiaqi Z. 所有, 且如有后续更新, 以 GitHub 仓库最新版说明为准.

0.2 创作者名单

本笔记除 Jiaqi Z. 参与主要整理之外, 以下人员也参与创作:

¹GitHub 仓库地址:<https://github.com/JackyZhang00/numerical-analysis-notes>

- Jiakang L. 负责”数值积分”章节的整理

Chapter 1

绪论

1.1 误差

1.1.1 误差来源与分类

1. (模型误差): 从实际模型中抽象出数学模型;

例如, 一个质量为 m 的小球做自由落体运动, 则位置 s 与时间 t 的关系式满足:

$$m \frac{d^2 s}{dt^2} = mg$$

不难想见, 该式仅在不考虑阻力时成立.

2. (观测误差): 通过测量得到模型中参数的值;
3. (方法误差 (或称截断误差)): 求近似解时所引入的误差;

例 1.1.1. 考虑函数 $f(x)$ 做 *Taylor* 多项式展开所导致的截断误差.

解. 对函数 $f(x)$ 计算 *Taylor* 多项式, 有

$$P_n(x) = f(0) + \frac{f'(0)}{1!}x + \frac{f''(0)}{2!}x^2 + \cdots + \frac{f^{(n)}(0)}{n!}x^n$$

由于有限项, 因此多项式有截断误差

$$R_n(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!}x^{n+1}$$

其中, $\xi \in (x, 0)$

□

4. (舍入误差): 机器字长有限所引起的误差

其中, 方法误差和舍入误差是数值分析所重点考虑的误差, 同时, 方法误差是可以避免的.

1.1.2 误差概念

绝对误差与绝对误差限

定义 1.1.1 (绝对误差与绝对误差限). 设 x 是准确值, x^* 是 x 的一个近似值, 则称

$$e(x^*) = x^* - x$$

为 x^* 的绝对误差, 简称误差.

同时, 误差的绝对值的上限 $\varepsilon(x^*)$, 即有

$$|e(x^*)| = |x^* - x| \leq \varepsilon(x^*)$$

$\varepsilon(x^*)$ 称为绝对误差限.

注意: 误差有正有负, 而误差限恒为正值.

习惯上, 我们把精确值和测量值的关系表示为

$$x = x^* \pm \varepsilon$$

相对误差与相对误差限

定义 1.1.2 (相对误差与相对误差限). 设 x 为准确值, x^* 为近似值, 称

$$e_r^* = e_r^*(x^*) = \frac{e(x^*)}{x} = \frac{x^* - x}{x}$$

为近似值 x^* 的相对误差.

同时, 其绝对值的上限 ε_r^* , 即有

$$\left| \frac{x - x^*}{x} \right| \leq \varepsilon_r^*$$

ε_r^* 称为相对误差限.

可以证明, 当 e_r^* 较小时, 有

$$e_r^* \approx \frac{x^* - x}{x^*}$$

同时易得

$$\varepsilon_r^* = \frac{\varepsilon^*}{|x^*|}$$

有效数字

定义 1.1.3 (有效数字, 有效位数, 有效数). 若近似值 x^* 误差满足

$$|x - x^*| \leq \frac{1}{2} \times 10^{-n}$$

则称 x^* 近似表示 x 准确到小数点后第 n 位, 并从第 n 位起一直到最左边非零数字之间的一切数字称为有效数字, 位数为有效位数.

若所有数字均为有效数字, 则称为有效数

例 1.1.2. 考虑圆周率 π , 且有近似值 $\pi_1 = 3.14, \pi_2 = 3.1415, \pi_3 = 3.1416, \pi_4 = 3.14159$. 考虑它们的有效数字, 且判断是否为有效数.

解. 对于 $\pi_1 = 3.14$, 有 $|\pi - \pi_1| \approx 0.00159 \leq 0.5 \times 10^{-2}$, 即 π_1 精确到小数点后 2 位, 有效数字是 3 位, 是有效数.

同理, 有 $|\pi - \pi_2| \approx 0.0000926 \leq 0.5 \times 10^{-3}$, 即 π_2 精确到小数点后 3 位, 有效数字是 4 位, 不是有效数.

$|\pi - \pi_3| \approx 0.0000073 \leq 0.5 \times 10^{-4}$, 即 π_3 精确到小数点后 4 位, 有效数字是 5 位, 是有效数.

$|\pi - \pi_4| \approx 0.0000026 \leq 0.5 \times 10^{-5}$, 即 π_4 精确到小数点后 5 位, 有效数字是 6 位, 是有效数. \square

从上例中不难看出, 有效数通常是采取四舍五入所得到的近似值.

扩展: 我们可以简单给出关于四舍五入的证明.

证明. 设准确值为 x , 其近似值为 x^* , 考虑近似值精确到小数点后 n 位, 即

$$|x - x^*| \leq 5 \times 10^{-(n+1)}$$

若其为有效数, 则 x^* 为小数点后 n 位, 不妨设

$$x^* = a + b \cdot 10^{-n}$$

其中 $b \in [1, 10)$

特别地, 分两种情况讨论.

若 $x > x^*$, 即真实值大于近似值, 此时有

$$x \leq x^* + 5 \times 10^{-(n+1)} = a + b \cdot 10^{-n} + 5 \times 10^{-(n+1)}$$

即当小数点后第 $n+1$ 位小于等于 5 时, 舍去后面的数字可以得到有效数.

若 $x < x^*$, 即真实值小于近似值, 此时有

$$\begin{aligned} x &\geq x^* - 5 \times 10^{-(n+1)} = a + b \cdot 10^{-n} - 5 \times 10^{-(n+1)} \\ &= a + (b-1) \cdot 10^{-n} + 5 \times 10^{-(n+1)} \end{aligned}$$

即当小数点后第 $n+1$ 位大于等于 5 时, 进位可以得到有效数. \square

十进制浮点表示法

定义 1.1.4. 设 x^* 为任一十进制数, 则 x^* 可表示为

$$x^* = \pm 0.a_1 a_2 \cdots a_n \cdots \times 10^m$$

其中, a_1 为 1 到 9 之间的一个数字, $a_2 \cdots a_n$ 为 0 到 9 之间的一个数字, m 为整数. 这样表示的 x^* 称为十进制浮点数 (规格化浮点数).

有效数字的等价定义 (基于浮点表示法)

定义 1.1.5. 若近似值 $x^* = \pm 0.a_1 a_2 \cdots a_n a_{n+1} \cdots a_{n+p} \times 10^m (a_1 \neq 0)$ 的误差限是某一位上的半个单位, 即

$$|x - x^*| \leq \frac{1}{2} \times 10^{m-n} \quad (1.1)$$

则称 x^* 有 n 位有效数字.

例 1.1.3. 设 $x_1^* = 0.0051, x_2^* = 5.100$, 两数均为四舍五入得到, 求两个数字的有效位数.

解. 由于有

$$\begin{aligned} \varepsilon(x_1^*) &= 0.5 \times 10^{-4}, x_1^* = 0.51 \times 10^{-2} \\ \varepsilon(x_2^*) &= 0.5 \times 10^{-3}, x_2^* = 0.51 \times 10^1 \end{aligned}$$

可得

$$\begin{aligned} \varepsilon(x_1^*) &= 0.5 \times 10^{-2-2} \\ \varepsilon(x_2^*) &= 0.5 \times 10^{1-4} \end{aligned}$$

即, x_1^* 有两位有效数字, x_2^* 有四位有效数字. \square

例 1.1.4. 设 $x_1^* = 2.180, x_2^* = 10.210$, 均具有四位有效数字, 求绝对误差限和相对误差限.

解. 对 x_1^* , 有

$$x_1^* = 0.2180 \times 10^1$$

即 $m = 1$, 且具有四位有效数字, 即 $n = 4$, 则根据公式 (1.1), 有

$$\varepsilon(x_1^*) = 0.5 \times 10^{1-4} = 0.5 \times 10^{-3}$$

其相对误差限为

$$\varepsilon_r(x_1^*) = \frac{\varepsilon(x_1^*)}{|x_1^*|} = 0.023\%$$

同理可得, 对于 x_2^* , 有

$$\varepsilon(x_2^*) = 0.5 \times 10^{-2}, \varepsilon_r(x_2^*) = 0.049\%$$

□

1.1.3 相对误差限和有效数字的关系

关于有效数字和相对误差限之间的关系, 有如下定理.

定理 1.1.1. 对于用式 (1.1.4) 表示的近似数 x^* , 若 x^* 具有 n 位有效数字, 则其相对误差限为

$$\varepsilon_r^* \leq \frac{1}{2a_1} \times 10^{-n}$$

证明. 由式 1.1.4 可得

$$a_1 \times 10^m \leq |x^*| \leq (a_1 + 1) \times 10^m$$

当 x^* 有 n 位有效数字时, 有

$$|x - x^*| = |x^*| \varepsilon_r^* \leq (a_1 + 1) \times 10^m \times \frac{1}{2(a_1 + 1)} \times 10^{-n} = 0.5 \times 10^{m-n}$$

故 x^* 有 n 位有效数字. □

上述定理表明: 有效位数越多, 相对误差限越小.

例 1.1.5. 令 $\sqrt{20}$ 的近似值相对误差限小于 0.1% , 则需要取多少位有效数字?

解. 由定理 1.1.1 可知

$$\varepsilon_r^* \leq \frac{1}{2a_1} \times 10^{-n}$$

由于 $\sqrt{20} \approx 4.4$, 故 $a_1 = 4$, 只需要取 $n = 4$, 有

$$\varepsilon_r^* \leq 0.125 \times 10^{-3} < 10^{-3} = 0.1\%$$

即只需要对 $\sqrt{20}$ 的近似值取 4 位有效数字, 其相对误差限就可以小于 0.1%, 此时有

$$\sqrt{20} \approx 4.472.$$

□

1.2 数值运算的误差估计

1.2.1 四则运算误差估计

两个近似数分别为 x_1^* 和 x_2^* , 误差限分别为 $\varepsilon(x_1^*), \varepsilon(x_2^*)$, 进行四则运算的误差限分别为:

$$\begin{aligned}\varepsilon(x_1^* \pm x_2^*) &= \varepsilon(x_1^*) + \varepsilon(x_2^*) \\ \varepsilon(x_1^* x_2^*) &\approx |x_1^*| \varepsilon(x_2^*) + |x_2^*| \varepsilon(x_1^*) \\ \varepsilon(x_1^* / x_2^*) &\approx \frac{|x_1^*| \varepsilon(x_2^*) + |x_2^*| \varepsilon(x_1^*)}{|x_2^*|^2}\end{aligned}$$

下面试着给出加减法误差的证明, 对于乘法和除法的证明, 将在后面给出.

证明.

$$\begin{aligned}|e(x_1^* \pm x_2^*)| &= |(x_1^* \pm x_2^*) - (x_1 \pm x_2)| \\ &= |(x_1^* - x_1) \pm (x_2^* - x_2)| \\ &\leq |x_1^* - x_1| + |x_2^* - x_2| \\ &\leq \varepsilon(x_1^*) + \varepsilon(x_2^*)\end{aligned}$$

□

1.2.2 函数值误差估计

一元函数误差估计

设 $f(x)$ 是一元函数, x 的近似值为 x^* , 以 $f(x^*)$ 近似 $f(x)$, 其误差限记作 $\varepsilon(f(x^*))$, 可用 Taylor 展开

$$f(x) - f(x^*) = f'(x^*)(x - x^*) + \frac{f''(\xi)}{2}\varepsilon^2(x^*)$$

其中, ξ 介于 x, x^* 之间, 取绝对值有

$$|f(x) - f(x^*)| \leq |f'(x^*)|\varepsilon(x^*) + \frac{|f''(\xi)|}{2}\varepsilon^2(x^*)$$

假定 $f'(x^*)$ 与 $f''(x^*)$ 的比值不大, 可忽略 $\varepsilon(x^*)$ 的高阶项, 于是可得误差限为

$$\varepsilon(f(x^*)) \approx |f'(x^*)|\varepsilon(x^*)$$

相对误差限为

$$\varepsilon_r(f(x^*)) \approx \frac{|f'(x^*)|\varepsilon(x^*)}{|f(x^*)|} = C_p(f, x^*)\varepsilon_r(x^*)$$

其中,

$$C_p(f, x^*) = \frac{|x^* f'(x^*)|}{|f(x^*)|}$$

称为 $f(x^*)$ 的条件数.

多元函数误差估计

当 f 为多元函数时计算 $A = f(x_1, x_2, \dots, x_n)$, 如果 x_1, x_2, \dots, x_n 的近似值为 $x_1^*, x_2^*, \dots, x_n^*$, 则 A 的近似值为 $A^* = f(x_1^*, x_2^*, \dots, x_n^*)$, 于是函数值 A^* 的误差 $e(A^*)$ 由 Taylor 展开, 得

$$\begin{aligned} e(A^*) &= A^* - A = f(x_1^*, x_2^*, \dots, x_n^*) - f(x_1, x_2, \dots, x_n) \\ &\approx \sum_{k=1}^n \left(\frac{\partial f(x_1^*, x_2^*, \dots, x_n^*)}{\partial x_k} \right) (x_k^* - x_k) = \sum_{k=1}^n \left(\frac{\partial f}{\partial x_k} \right)^* e_k^* \end{aligned}$$

于是误差限为

$$\varepsilon(A^*) \approx \sum_{k=1}^n \left| \left(\frac{\partial f}{\partial x_k} \right)^* \right| \varepsilon(x_k^*) \quad (1.2)$$

而 A^* 的相对误差限为

$$\varepsilon_r^* = \varepsilon_r(A^*) = \frac{\varepsilon(A^*)}{|A^*|} \approx \sum_{k=1}^n \left| \left(\frac{\partial f}{\partial x_k} \right)^* \right| \frac{\varepsilon(x_k^*)}{|A^*|}$$

例 1.2.1. 已测得某场地长 l 的值为 $l^* = 110 \text{ m}$, 宽 d 的值为 $d^* = 80 \text{ m}$, 已知 $|l - l^*| \leq 0.2 \text{ m}$, $|d - d^*| \leq 0.1 \text{ m}$, 试求面积 $S = ld$ 的绝对误差限与相对误差限.

解. 因为 $S = ld$, $\frac{\partial S}{\partial l} = d$, $\frac{\partial S}{\partial d} = l$, 由式 1.2 可知

$$\varepsilon(S^*) \approx \left| \left(\frac{\partial S}{\partial l} \right)^* \right| \varepsilon(l^*) + \left| \left(\frac{\partial S}{\partial d} \right)^* \right| \varepsilon(d^*)$$

其中,

$$\left(\frac{\partial S}{\partial l} \right)^* = d^* = 80 \text{ m}, \left(\frac{\partial S}{\partial d} \right)^* = l^* = 110 \text{ m}$$

而

$$\varepsilon(l^*) = 0.2 \text{ m}, \varepsilon(d^*) = 0.1 \text{ m}$$

于是绝对误差限为

$$\varepsilon(S^*) \approx (80 \times 0.2 + 110 \times 0.1) \text{ m}^2 = 27 \text{ m}^2$$

相对误差限为

$$\varepsilon_r(S^*) = \frac{\varepsilon(S^*)}{|S^*|} = \frac{\varepsilon(S^*)}{l^* d^*} \approx \frac{27}{8800} = 0.31\%$$

□

注意: 绝对误差限有量纲, 而相对误差限没有量纲.

1.3 算法数值稳定性

定义 1.3.1 (数值稳定). 一个算法如果初始数值有微小扰动 (即有误差), 而计算过程中舍入误差不增长, 使得结果产生微小误差. 则称该算法为数值稳定的. 反之称为数值不稳定.

例 1.3.1. 计算定积分

$$I_n = \int_0^1 \frac{x^n}{n+5} dx, n = 0, 1, 2, \dots, 8$$

解. 对被积函数变形, 得

$$\begin{aligned} I_n &= \int_0^1 \frac{(x+5)-5}{x+5} x^{n-1} dx \\ &= \int_0^1 x^{n-1} dx - 5 \int_0^1 \frac{x^{n-1}}{x+5} dx \\ &= \frac{1}{n} - 5I_{n-1} \end{aligned}$$

其中, $n = 1, 2, \dots, 8$.

易知, $I_0 = \ln 6 - \ln 5 = \ln 1.2$, 由于机器只能计算小数, 取三位有效数字, 即 $\ln 1.2 \approx 0.182$.

分析上述积分, 可知, $0 < I_n < 0.2$, 且随着 n 增大, I_n 逐渐减小, 当 $n \rightarrow \infty$ 时, $I_n \rightarrow 0$.

迭代计算上述积分, 可得结果为:

$$I_0 = 0.182, I_1 = 0.09, I_2 = 0.05, I_3 = 0.083, I_4 = -0.17$$

$$I_5 = 1.03, I_6 = -5.0, I_7 = 25.14, I_8 = -125.59$$

可以发现, 该算法数值不稳定.

若对上述积分递推公式进行变形, 可得

$$I_{n-1} = \frac{1}{5n} - \frac{1}{5}I_n, n = 9, 8, \dots, 1$$

由于当 $n \rightarrow \infty$ 时, $I_n \rightarrow 0$, 因此当 n 充分大时, 可近似认为 $I_n = I_{n+1}$, 故有 $I_9 \approx I_{10}$, 将其代入并求解方程, 可得 $I_9 \approx 0.017$.

迭代计算, 可得结果为

$$I_0 = 0.182, I_1 = 0.088, I_2 = 0.058, I_3 = 0.043, I_4 = 0.034$$

$$I_5 = 0.028, I_6 = 0.024, I_7 = 0.021, I_8 = 0.019$$

该算法为数值稳定的.

分析二者的误差, 可得对于第一个算法, 其误差为

$$e_n = |I_n - I_n^*| = 5|e_{n-1}| = 5^n|e_1|$$

而对于第二个算法, 其误差为

$$|e_{n-1}| = |I_{n-1} - I_{n-1}^*| = \frac{1}{5}|e_n| = \left(\frac{1}{5}\right)^n |e_9|$$

□

通过上述例子, 可以看到对于同一个问题, 使用不同算法, 得到的误差结果可能有很大不同.

扩展: 考虑到数值分析需要结合计算机使用, 故在笔记的适当地方, 将给出代码以供参考 (注: 代码不唯一. 且考虑到算法的设计原则, 如无必要, 不会引入相应的库函数).

本例的运行代码如下所示:

```
1 # 验证数值稳定性(例题) Exercise1-1.py
2 # 方法1(数值不稳定)
3 def I1(n):
4     if n==0:
5         return 0.182
6     else:
7         return 1/n-5*I1(n-1)
8 # 方法2(数值稳定)
9 def I2(n):
10    if n==9:
11        return 0.017
12    else:
13        return 1/(5*(n+1))-(1/5)*I2(n+1)
14
15 for n in range(0,9):
16     print(f"I1_{n}_={I1(n)}")
17
18 for n in range(0,9):
19     print(f"I2_{n}_={I2(n)}")
```

定义 1.3.2 (良态与病态). 对于一个数学问题, 若初始数据有微小扰动 (即误差), 导致计算结果产生较小误差, 则称此问题是良态的, 否则称其为病态的.

注意: 良态和病态是针对数学问题本身的, 与算法无关.

1.4 数值计算中应该注意的一些原则

1.4.1 避免两相近数相减

使用两相近数相减, 将会导致有效数字损失. 下面的例子将有效说明这一点:

例 1.4.1. 计算函数 $y = \sqrt{x+1} - \sqrt{x}$ 在 $x = 1000$ 处的取值.

已知 y 的四位有效数字为 0.01580

解. 若选择直接相减, 则有 $y = \sqrt{1001} - \sqrt{1000} \approx 31.64 - 31.62 = 0.02$, 只有两位有效数字.

若选择对其进行变形, 令

$$y = \frac{1}{\sqrt{x+1} + \sqrt{x}}$$

则可得

$$y = \frac{1}{\sqrt{1001} + \sqrt{1000}} \approx \frac{1}{31.64 + 31.62} = 0.01581$$

有三位有效数字. □

注意: 在本例中, 使用第二种方法得到的只有三位有效数字, 这是因为第四位有效数字是 0 而不是 1.

1.4.2 避免除数绝对值远小于被除数绝对值

例如, $\frac{42}{0.01}$ 和 $\frac{42}{0.011}$ 的结果分别是 4200 和 3818.18, 明显可以发现, 除数只变化了 0.001, 结果变化了 381.82

1.4.3 避免大数”吃”小数

由于计算机字长是有限的, 在计算过程中需要考虑到对阶, 例如, 计算下面的式子:

$$10^9 + 1$$

在计算前首先需要将其规格化, 即将上式化为

$$0.1 \times 10^{10} + 0.1 \times 10^1$$

在计算机计算过程中, 需要进行对阶, 即将指数部分化为相同的. 在这里, 计算机将会做如下处理:

$$0.1 \times 10^{10} + 0.0000000001 \times 10^{10} = 0.1000000001 \times 10^{10}$$

同时, 考虑到计算机内部的小数存储是有长度限制的, 假设以 8 位为例, 则上式中的小数最后一位的 1 将被舍去, 从而得到结果为 0.1×10^{10} , 显然与结果不符.

下面的例子将详细说明这一点:

例 1.4.2. 用单精度 (浮点数保留 8 位小数) 计算

$$10^9 + 40 + 39 + \cdots + 1$$

解. 假设从左到右计算, 由于

$$10^9 + 40 = 0.1 \times 10^{10} + 0.4 \times 10^2 \approx 0.1 \times 10^{10}$$

会出现大数”吃”小数的情况.

假设从右到左计算, 则首先计算 $1 + 2 + \cdots + 40$, 不难得结果为 820, 即有

$$\begin{aligned} \text{原式} &= 820 + 10^9 = 0.82 \times 10^3 + 0.1 \times 10^{10} \\ &= 0.000000082 \times 10^{10} + 0.1 \times 10^{10} \\ &= 0.100000082 \times 10^{10} \approx 0.10000008 \times 10^{10} = 1000000800 \end{aligned}$$

显然误差较小. □

下面的代码将演示这一点

注意: 由于计算机内部的存储方式和实际计算有些许误差 (计算机采用二进制存储), 因此运行结果可能与理论分析不一样.

```

1 # 演示大数"吃"小数 Exercise1-2.py
2 import numpy as np
3 # 使用从左到右的计算方式, 会有很大误差
4 def example1():
5     result = np.float32(0)
6     result = result + np.float32(1e9)

```

```

7     for i in range(1,41):
8         result = result + np.float32(i)
9         print(f"从左到右计算结果为{result}")
10    # 使用从右到左的计算方式, 误差较小
11    def example2():
12        result = np.float32(0)
13        for i in range(1,41):
14            result = result + np.float32(i)
15            result = result + np.float32(1e9)
16            print(f"从右到左计算结果为{result}")
17    # 运行结果
18    example1()
19    example2()

```

1.4.4 简化计算步骤, 避免误差积累

例 1.4.3. 多项式求值: 给定 x , 求下列 n 次多项式的值:

$$P(x) = a_0 + a_1x + \cdots + a_nx^n$$

解. 若采用直接求和的方法, 则有

$$P_n(x) = a_0 + a_1x + a_2x \cdot x + \cdots + a_n \underbrace{x \cdot x \cdot x \cdots x}_{n \uparrow x}$$

一共需要 $\frac{n(n+1)}{2}$ 次乘法, n 次加法

若使用逐项求和, 即令

$$x^2 = x \cdot x, x^3 = x^2 \cdot x, \cdots x^n = x^{n-1} \cdot x$$

一共需要 $(2n-1)$ 次乘法, n 次加法

若采用秦九韶算法 (Horner 算法), 则可以将上式整理为

$$P_n(x) = a_0 + x(a_1 + x(a_2 + x(\cdots + x(a_{n-1} + a_nx) \cdots)))$$

一共需要 n 次乘法, n 次加法

□

可以明显发现, 使用秦九韶算法的效率明显优于其他两个算法.
对于秦九韶算法, 可以使用如下递推公式:

$$\begin{cases} S_n = a_n \\ S_k = xS_{k+1} + a_k, k = n-1, n-2, \dots, 0 \\ P_n(x) = S_0 \end{cases}$$

实际上机运行可以参考如下代码:

```

1  # 演示100000次多项式不同算法时间差异(假设每一项系数a_n都是2)
   Exercise1-3.py
2  import time
3  POWER = 100000
4  # 直接求和
5  def method1():
6      result = 0
7      x = 2
8      a = []
9      for i in range(0, POWER+1):
10         a.append(2)
11         start_time = time.time()
12         for i in range(0, POWER+1):
13             result = result + a[i]*x**i
14         end_time = time.time()
15         # print(result)
16         print(end_time-start_time)
17  # 使用逐项求和
18  def method2():
19      result = 0
20      x = [1, 2]
21      for i in range(2, POWER+1):
22         x.append(x[i-1]*2)
23      a = []
24      for i in range(0, POWER+1):
25         a.append(2)

```



```
26     start_time = time.time()
27     for i in range(0,POWER+1):
28         result = result + a[i]*x[i]
29     end_time = time.time()
30     # print(result)
31     print(end_time-start_time)
32 # 秦九韶算法
33 def method3():
34     s = 2
35     x = 2
36     start_time = time.time()
37     for i in range(1,POWER+1):
38         s = s*x+2
39         # s.append(s[i-1]*x+2)
40     end_time = time.time()
41     # print(s)
42     print(end_time-start_time)
43
44 method1()
45 method2()
46 method3()
```

注意： 在本地测试时，得到运行结果分别为 10.107, 0.264, 0.249(单位“秒”)。这个时间可能在不同计算机上会有所差距，但一般情况下，随着多项式次数的增加，时间差距会逐渐拉大。同时，三种算法的时间增长速率也会有明显差距。

Chapter 2

插值法

2.1 引言

定义 2.1.1 (插值法). 设函数 $y = f(x)$ 在区间 $[a, b]$ 上有定义, 且已知在点 $a \leq x_0 \leq x_1 < \cdots < x_n \leq b$ 上的值 y_0, y_1, \cdots, y_n , 若存在一简单函数 $P(x)$, 使

$$P(x_i) = y_i, i = 0, 1, \cdots, n \quad (2.1)$$

成立, 则称 $P(x)$ 为函数 $f(x)$ 的插值函数, 点 x_0, x_1, \cdots, x_n 为插值节点, 包括插值节点的区间 $[a, b]$ 称为插值区间, 求插值函数 $P(x)$ 的方法称为插值法.

定义 2.1.2 (多项式插值). 若 $P(x)$ 是次数不超过 n 的代数多项式, 即

$$P(x) = a_0 + a_1x + \cdots + a_nx^n \quad (2.2)$$

其中 a_i 为实数, 则称 $P(x)$ 为插值多项式, 相应的插值法称为多项式插值.

本章所讨论的主要内容是多项式插值.

在寻找插值多项式之前, 需要对其存在性与唯一性进行讨论¹. 给出如下定理:

定理 2.1.1. 对于给定互异节点 x_0, x_1, \cdots, x_n , 满足插值条件式 (2.1) 的 n 阶插值多项式 (2.2) 存在且唯一.

¹ 存在性表明插值多项式存在, 唯一性表明无论采用哪种插值方法, 得到的结果是唯一的.

证明. 设所要构造的插值多项式为

$$P_n(x) = a_0 + a_1x + \cdots + a_nx^n$$

由插值条件

$$P_n(x_i) = y_i, i = 0, 1, \cdots, n$$

得如下线性方程组

$$\begin{cases} 1 \cdot a_0 + x_0a_1 + \cdots + x_0^na_n = y_0 \\ 1 \cdot a_0 + x_1a_1 + \cdots + x_1^na_n = y_1 \\ \vdots \\ 1 \cdot a_0 + x_na_1 + \cdots + x_n^na_n = y_n \end{cases}$$

求解 a_0, a_1, \cdots, a_n , 计算系数行列式

$$D = \begin{vmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^n \\ 1 & x_1 & x_1^2 & \cdots & x_1^n \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^n \end{vmatrix}$$

该行列式为 Vandermonde 行列式, 其值为

$$D = \prod_{0 \leq j < i \leq n} (x_i - x_j)$$

当 $x_i \neq x_j$ 时, $D \neq 0$, 即 $P_n(x)$ 由 a_0, a_1, \cdots, a_n 唯一确定 □

在实际计算过程中, 直接求解方程组往往计算量较大, 且方程组可能具有病态性. 例如, 对于 x_1, x_2, x_3, x_4 , 若值分别为 0.1, 0.2, 0.3, 0.4, 则行列式 $D = 1.2 \times 10^{-6} \approx 0$.

因此, 通常的做法是在 n 次多项式空间中寻找一组基函数

$$\varphi_0(x), \varphi_1(x), \cdots, \varphi_n(x)$$

使得

$$P_n(x) = a_0\varphi_0(x) + a_1\varphi_1(x) + \cdots + a_n\varphi_n(x)$$

不同的基函数对应不同的插值法. 本章重点讨论 Lagrange 插值法与 Newton 插值法.

2.2 Lagrange 插值法

2.2.1 线性插值

例 2.2.1. 对于节点 $(x_0, y_0), (x_1, y_1)$, 求一次多项式

解. 利用直线的两点式, 不难得到其插值多项式为

$$\begin{aligned} P_1 &= \left(\frac{x - x_1}{x_0 - x_1} \right) y_0 + \left(\frac{x - x_0}{x_1 - x_0} \right) y_1 \\ &= l_0(x)y_0 + l_1(x)y_1 = \sum_{i=0}^1 l_i(x)y_i \end{aligned}$$

□

在这里, 称

$$l_0(x) = \frac{x - x_1}{x_0 - x_1}, l_1(x) = \frac{x - x_0}{x_1 - x_0}$$

为一次 Lagrange 插值基函数.

不难验证, 对于一次 Lagrange 插值基函数而言, 存在如下性质

- $l_0(x), l_1(x)$ 均为一次多项式
- $l_0(x_0) = 1, l_1(x_0) = 0, l_0(x_1) = 0, l_1(x_1) = 1$

2.2.2 抛物插值

与线性插值类似, 对于抛物插值, 设有三个插值点, 分别为 $(x_0, y_0), (x_1, y_1), (x_2, y_2)$, 可得其插值多项式为

$$P_2(x) = y_0 l_0(x) + y_1 l_1(x) + y_2 l_2(x)$$

其中 $l_0(x), l_1(x), l_2(x)$ 均为二次多项式, 且有

$$l_0(x_0) = 1, l_1(x_0) = 0, l_2(x_0) = 0$$

$$l_0(x_1) = 0, l_1(x_1) = 1, l_2(x_1) = 0$$

$$l_0(x_2) = 0, l_1(x_2) = 0, l_2(x_2) = 1$$

2.2.3 Lagrange 插值多项式

将上述结论推广至 n 阶情况. 即假设有 $n+1$ 个节点 x_0, x_1, \dots, x_n 的 n 阶插值多项式 $L_n(x)$, 且满足条件

$$L_n(x_i) = y_i, i = 1, 2, \dots, n \quad (2.3)$$

类似于线性插值和抛物插值, 我们首先需要定义出基函数.

定义 2.2.1. 若 n 次多项式 $l_j(x), j = 0, 1, \dots, n$ 在 $n+1$ 个节点 $x_0 < x_1 < \dots < x_n$ 上满足条件

$$l_j(x_k) = \begin{cases} 1, & k = j \\ 0, & k \neq j \end{cases}, j, k = 0, 1, \dots, n$$

则称这 $n+1$ 个 n 次多项式 $l_0(x), l_1(x), \dots, l_n(x)$ 为节点 x_0, x_1, \dots, x_n 上的 n 次插值基函数.

利用其性质, 可以得到基函数形式为

$$l_k(x) = \frac{(x-x_0) \cdots (x-x_{k-1})(x-x_{k+1}) \cdots (x-x_n)}{(x_k-x_0) \cdots (x_k-x_{k-1})(x_k-x_{k+1}) \cdots (x_k-x_n)}, k = 0, 1, \dots, n \quad (2.4)$$

扩展: 下面将说明如何计算基函数的形式.

利用性质, 可知对于 $l_k(x), k = 0, 1, \dots, n$, 当 $x \neq x_k$ 时, 其函数值为 0. 则可以将其分解为若干因式 $(x-x_j), j = 0, 1, \dots, n$ 且 $j \neq k$, 即

$$l_k(x) = \lambda(x-x_0)(x-x_1) \cdots (x-x_{k-1})(x-x_{k+1}) \cdots (x-x_n), k = 0, 1, \dots, n$$

同时, 由于当 $x = x_k$ 时, $l_k(x_k) = 1$, 可得待定系数 λ 为

$$\lambda = \frac{1}{(x_k-x_0)(x_k-x_1) \cdots (x_k-x_{k-1})(x_k-x_{k+1}) \cdots (x_k-x_n)}, k = 0, 1, \dots, n$$

代入并整理, 可得基函数的具体形式为

$$l_k(x) = \frac{(x-x_0) \cdots (x-x_{k-1})(x-x_{k+1}) \cdots (x-x_n)}{(x_k-x_0) \cdots (x_k-x_{k-1})(x_k-x_{k+1}) \cdots (x_k-x_n)}, k = 0, 1, \dots, n$$

上式因此得证.

下面将试着给出基于 Lagrange 多项式插值的一个程序代码, 仅供参考.

```
1  # 使用拉格朗日多项式插值法的实例 Exercise2-1.py
2  # 假设四个插值点分别为 (1,2), (2,3), (3,6), (4,7)
3  # 实际运行时这些数据可以自行修改, 从而观察插值的实际作用.
4
5  import numpy as np
6  import matplotlib.pyplot as plt
7
8  def lagrange_interpolation(x, points):
9      n = len(points)
10     result = 0.0
11     for i in range(n):
12         xi, yi = points[i]
13         term = yi
14         for j in range(n):
15             if i != j:
16                 xj, yj = points[j]
17                 term *= (x - xj) / (xi - xj)
18         result += term
19     return result
20
21 x = [1,2,3,4]
22 y = [2,3,6,7]
23 plt.scatter(x,y,color="red")
24 points = list(zip(x,y))
25 x = np.arange(1,5,0.01)
26 result = lagrange_interpolation(x, points)
27 plt.plot(x,result)
28 plt.show()
```

使用这段代码运行的结果如图 2.1所示.

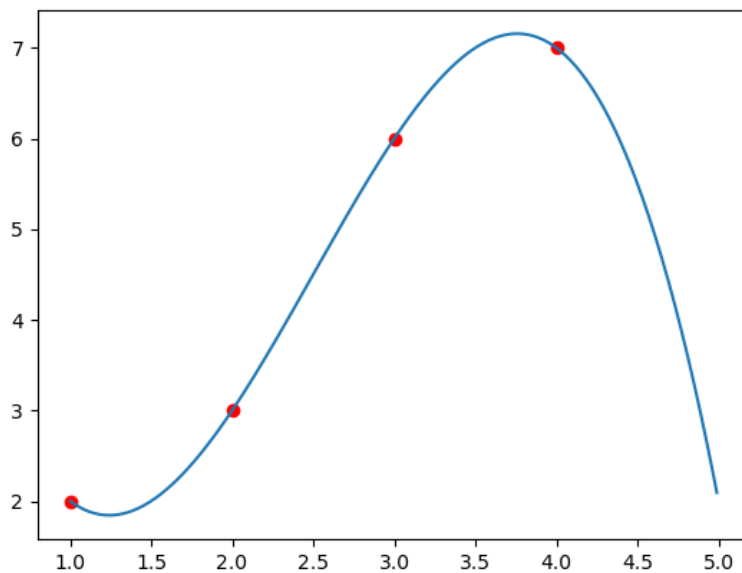


图 2.1: Lagrange 多项式插值 (使用上述代码生成)

2.2.4 插值余项

使用 $L_n(x)$ 近似表示 $f(x)$, 则会引起截断误差. 其截断误差为 $R_n(x) = f(x) - L_n(x)$. 通常会称 $R_n(x)$ 为插值多项式的余项或简称为插值余项.

为估计插值余项, 有如下定理:

定理 2.2.1. 设 $f^{(n)}(x)$ 在 $[a, b]$ 上连续, $f^{(n+1)}(x)$ 在 (a, b) 存在, 节点 $a \leq x_0 < x_1 < \cdots < x_n \leq b$, $L_n(x)$ 是满足条件式 (2.3) 的插值多项式, 则对于任何 $x \in [a, b]$, 插值余项

$$R_n(x) = f(x) - L_n(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!} \prod_{i=0}^n (x - x_i) \quad (2.5)$$

其中, $\xi \in (a, b)$ 且依赖于 x

证明. 由插值条件可知, $R_n(x)$ 在节点 $x_k, k = 0, 1, \dots, n$ 上为 0, 即可

以做因式分解

$$R_n(x) = K(x)(x - x_0)(x - x_1) \cdots (x - x_n) = K(x) \prod_{i=0}^n (x - x_i)$$

其中, $K(x)$ 是与 x 有关的待定系数.

把 x 视作区间 $[a, b]$ 上的一个固定点, 作函数

$$\varphi(t) = f(t) - L_n(t) - K(t)(t - x_0)(t - x_1) \cdots (t - x_n)$$

因此, $\varphi(t)$ 在 x_0, x_1, \dots, x_n 和 x 处均为 0, 即在区间 $[a, b]$ 上有 $n+2$ 个零点. 根据 Roll 定理可知, $\varphi'(t)$ 在两个零点间至少有一个零点, 即在区间 $[a, b]$ 上, $\varphi'(t)$ 至少有 $n+1$ 个零点. 以此类推, 不难得出 $\varphi^{(n+1)}$ 在区间 (a, b) 内至少有一个零点, 将其记为 $\xi \in (a, b)$, 使得

$$\varphi^{(n+1)}(\xi) = f^{(n+1)}(\xi) - (n+1)!K(x) = 0$$

整理可得

$$K(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!}, \xi \in (a, b)$$

将其代入上式, 可得余项表达式 (2.5)

□

通常, ξ 无法具体确定, 从而实际操作中, 经常估计误差上限. 由

$$|f^{(n+1)}(x)| \leq M_{n+1}$$

对于任意的 $x \in (a, b)$, 将

$$\frac{M_{n+1}}{(n+1)!} \prod_{i=0}^n |x - x_i|$$

作为误差估计上限. 通常取

$$M_{n+1} = \max_{a \leq x \leq b} |f^{(n+1)}(x)|$$

特别的, 若 $f(x)$ 为任一次数小于等于 n 的多项式, 即 $f(x) \in H_n \in \text{span}\{1, x, x^2, \dots, x^n\}$, 此时 $f^{(n+1)} \equiv 0$, 即 $R_n(x) \equiv 0$. 因此, 插值多项式对于次数小于等于 n 的多项式总是精确的.

例 2.2.2. 设有 $x_0 \neq x_1 \neq x_2 \neq x_3 \neq x_4$, 且 $l_i(x), i = 0, 1, 2, 3, 4$ 为 Lagrange 插值基函数. 求

$$\sum_{i=0}^4 (3x_i^4 + 4x_i^2 + 2x_i + 1) l_i(x)$$

解. 设函数 $f(x) = 3x^4 + 4x^2 + 2x + 1$, 则

$$\text{原式} = \sum_{i=0}^4 f(x_i)l_i(x) = f(x)$$

□

例 2.2.3. 已知 $\sin \frac{\pi}{6} = \frac{1}{2}$, $\sin \frac{\pi}{4} = \frac{\sqrt{2}}{2}$, $\sin \frac{\pi}{3} = \frac{\sqrt{3}}{2}$, 利用 $\sin x$ 的一次, 二次 Lagrange 插值, 计算 $\sin 50^\circ$ 并估计误差

解. 当 $n = 1$ 时, 利用 x_0, x_1 , 有

$$\begin{aligned}\Rightarrow L_1(x) &= \frac{x - \frac{\pi}{4}}{\frac{\pi}{6} - \frac{\pi}{4}} \cdot \frac{1}{2} + \frac{x - \frac{\pi}{6}}{\frac{\pi}{4} - \frac{\pi}{6}} \cdot \frac{\sqrt{2}}{2} \\ \Rightarrow L_1\left(\frac{5}{18}\pi\right) &\approx 0.77614\end{aligned}$$

其误差

$$R_1(x) = \frac{f''(\xi)}{2!} \left| x - \frac{\pi}{6} \right| \left| x - \frac{\pi}{4} \right|$$

其中 $|\sin \xi| \leq \frac{\sqrt{2}}{2}$, 因此有

$$\left| R_1\left(\frac{5}{18}\pi\right) \right| < 0.01319$$

类似地, 利用 x_1, x_2 , 可得 $L_1(x) \approx 0.76008$, 估计误差, 由于 $|\sin \xi| \leq \frac{\sqrt{3}}{2}$

$$\left| R_1\left(\frac{5}{18}\pi\right) \right| < 0.00660$$

当 $n = 2$ 时, 有

□

2.2.5 Lagrange 插值优缺点

优点: 具有严格的规律性, 便于记忆与理论推导;

缺点: 计算量大, 每次添加 (或删除) 节点都需要重新计算基函数, 不具有承袭性.

为解决上述缺点, 将引出新的插值法, 即 Newton 插值.

2.3 Newton 插值

2.3.1 Newton 插值

与 Lagrange 插值类似, 首先考虑 $n = 1$

例 2.3.1. 已知两个节点 x_0, x_1 , 其函数值分别为 y_0, y_1 , 试求一次多项式 $P_1(x) = A_0 + A_1x$, 使得 $P_1(x_0) = y_0, P_1(x_1) = y_1$

解. 利用直线方程点斜式, 可知

$$P_1(x) = y_0 + \frac{y_1 - y_0}{x_1 - x_0}(x - x_0)$$

此时, 插值节点为 x_0, x_1 , 基函数设为 $1, (x - x_0)$, 组合系数为 $A_0 = y_0, A_1 = \frac{y_1 - y_0}{x_1 - x_0}$ □

考虑 $n = 2$ 的情况, 要求具有承袭性. 设 $g(x) = P_2(x) - P_1(x)$, 则 $g(x)$ 为次数不超过 2 的多项式, 且有

$$g(x_i) = P_2(x_i) - P_1(x_i) = y_i - y_i = 0, i = 0, 1$$

因此可对其进行因式分解, 有

$$g(x) = A_2(x - x_0)(x - x_1)$$

$$\Rightarrow P_2(x) = P_1(x) + A_2(x - x_0)(x - x_1)$$

故, 对于 $n = 2$ 而言, 插值节点为 x_0, x_1, x_2 , 基函数为 $1, (x - x_0), (x - x_0)(x - x_1)$, 组合系数为 A_0, A_1, A_2 . 插值多项式为

$$P_2(x) = A_0 + A_1(x - x_0) + A_2(x - x_0)(x - x_1)$$

类似地, 给定插值条件

$$N_n(x_i) = f(x_i), i = 0, 1, \dots, n$$

, 插值节点为 $x_i, i = 0, 1, \dots, n$, 基函数为 $1, (x - x_0), (x - x_0)(x - x_1), \dots, (x - x_0)(x - x_1) \cdots (x - x_{n-1})$, 组合系数为 $A_i, i = 0, 1, \dots, n$

下面需要讨论如何求解组合系数.

设 $A_n(x) = A_0 + A_1(x - x_0) + A_2(x - x_0)(x - x_1) + \cdots + A_n(x - x_0)(x - x_1) \cdots (x - x_{n-1})$, 利用插值条件

$$N_n(x_i) = f(x_i), i = 0, 1, \dots, n$$

代入, 得线性方程组

$$\begin{pmatrix} 1 & 0 & \cdots & 0 \\ 1 & x_1 - x_0 & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ 1 & x_n - x_0 & \cdots & \prod_{i=0}^{n-1} (x_n - x_i) \end{pmatrix} \begin{pmatrix} A_0 \\ A_1 \\ \vdots \\ A_n \end{pmatrix} = \begin{pmatrix} f(x_0) \\ f(x_1) \\ \vdots \\ f(x_n) \end{pmatrix}$$

求解方程组, 可得

$$\begin{aligned} A_0 &= f(x_0) \\ A_1 &= \frac{f(x_1) - f(x_0)}{x_1 - x_0} \\ A_2 &= \frac{\frac{f(x_2) - f(x_0)}{x_2 - x_0} - \frac{f(x_1) - f(x_0)}{x_1 - x_0}}{x_2 - x_1} \\ &\vdots \end{aligned}$$

为简化计算, 总结上述规律, 给出下面关于差商的定义

2.3.2 差商

差商的定义

定义 2.3.1 (差商). 称

$$f[x_0, x_k] = \frac{f(x_k) - f(x_0)}{x_k - x_0}$$

为函数 $f(x)$ 关于点 x_0, x_k 的一阶差商, 称

$$f[x_0, x_1, x_k] = \frac{f[x_0, x_k] - f[x_0, x_1]}{x_k - x_1}$$

为 $f(x)$ 关于点 x_0, x_1, x_k 的二阶差商. 一般地, 称

$$f[x_0, x_1, \dots, x_k] = \frac{f[x_0, x_1, \dots, x_{k-2}, x_k] - f[x_0, x_1, \dots, x_{k-1}]}{x_k - x_{k-1}}$$

为 $f(x)$ 的 k 阶差商.

由差商定义可知: 高阶差商是两个低一阶差商的差商.

利用差商定义, 可知组合系数为:

$$\begin{aligned} A_0 &= f(x_0) = f[x_0] \\ A_1 &= f[x_0, x_1] \\ &\vdots \\ A_n &= f[x_0, x_1, \dots, x_n] \end{aligned}$$

$$\begin{aligned} \Rightarrow N_n(x) &= f(x_0) + f[x_0, x_1](x - x_0) + f[x_0, x_1, x_2](x - x_0)(x - x_1) \\ &\quad + \dots + f[x_0, x_1, \dots, x_n](x - x_0)(x - x_1) \dots (x - x_{n-1}) \end{aligned}$$

差商性质

差商与函数值有如下关系:

定理 2.3.1. 记 $\omega(x) = (x - x_0)(x - x_1) \dots (x - x_n)$, 则

$$f[x_0, x_1, \dots, x_n] = \sum_{i=0}^n \frac{f(x_i)}{\omega'(x_i)}$$

证明. 对于 $f(x)$, 使用 Lagrange 插值法, 可得:

$$L_n(x) = \sum_{i=0}^n f(x_i) l_i(x)$$

使用 Newton 插值法, 可得:

$$N_n(x) = f(x_0) + f[x_0, x_1](x - x_0) + \dots + f[x_0, x_1, \dots, x_n](x - x_0)(x - x_1) \dots (x - x_{n-1}) \quad (2.6)$$

由于插值多项式具有唯一性, 因此两种插值方法得到的结果相同, 即同次系数相等. 整理可得

$$\sum_{i=0}^n f(x_i) = f[x_0, x_1, \dots, x_n](x - x_0)(x - x_1) \dots (x - x_{n-1})$$

对 $\omega(x)$ 求导后, 原式得证. □

差商的值与节点的排列顺序无关, 即差商具有对称性. 用公式表示为:

$$f[x_0, \dots, x_i, \dots, x_j, \dots, x_n] = f[x_0, \dots, x_j, \dots, x_i, \dots, x_n]$$

设 $f(x)$ 在 $[a, b]$ 有 n 阶导数, 且 $x_0, x_1, \dots, x_n \in [a, b]$, 则存在 $\xi \in (a, b)$, 使得

$$f[x_0, x_1, \dots, x_n] = \frac{f^{(n)}(\xi)}{n!}$$

例 2.3.2. 若 $f(x) = 3x^4 + 4x^2 + 2x + 1$, 计算

$$f[2^0, 2^1, 2^2, 2^3, 2^4], f[2^0, 2^1, 2^2, 2^3, 2^4, 2^5]$$

解.

$$\begin{aligned} f[2^0, 2^1, 2^2, 2^3, 2^4] &= \frac{f^{(4)}(\xi)}{4!} = 3 \\ f[2^0, 2^1, 2^2, 2^3, 2^4, 2^5] &= \frac{f^{(5)}(\xi)}{5!} = 0 \end{aligned}$$

□

由前面的性质, 不难得到, 对于差商而言, 有

$$f[x_0, x_1, \dots, x_n] = \frac{f[x_1, x_2, \dots, x_n] - f[x_0, x_1, \dots, x_{n-1}]}{x_n - x_0}$$

该性质表明在实际计算差商过程中, 可以使用列表法计算.

2.3.3 Newton 插值余项

定理 2.3.2. *Newton* 插值多项式的余项为:

$$R_n(x) = f[x_0, x_1, \dots, x_n] \omega_{n+1}(x)$$

其中,

$$\omega_{n+1}(x) = (x - x_0)(x - x_1) \cdots (x - x_n) = \prod_{i=0}^n (x - x_i)$$

2.4 等距节点差商公式

2.4.1 差分定义

定义 2.4.1 (差分). 设函数 $y = f(x)$ 在等距节点 $x_k = x_0 + kh$ ($k = 0, 1, \dots, n$) 上的值 $f_k = f(x_k)$ 已知, 其中 h 为常数, 称为步长, 称偏差

$$\begin{aligned}\Delta f_k &= f_{k+1} - f_k \\ \nabla f_k &= f_k - f_{k-1} \\ \delta f_k &= f(x_k + h/2) - f(x_k - h/2) = f_{k+\frac{1}{2}} - f_{k-\frac{1}{2}}\end{aligned}$$

分别称为 $f(x)$ 在 x_k 处以 h 为步长的向前差分, 向后差分, 中心差分. 符号 Δ, ∇, δ 分别称为向前差分算子, 向后差分算子, 中心差分算子.

利用一阶差分, 类似可定义二阶差分为

$$\Delta^2 f_k = \Delta f_{k+1} - \Delta f_k = f_{k+2} - 2f_{k+1} + f_k$$

, 一般地, m 阶差分为

$$\Delta^m f_k = \Delta^{m-1} f_{k+1} - \Delta^{m-1} f_k$$

2.4.2 差分的性质

定义不变算子 I 和移位算子 E , 即有

$$If_k = f_k, Ef_k = f_{k+1}$$

由差分定义, 可得 $\Delta = E - I, \nabla = I - E^{-1}, \delta = E^{\frac{1}{2}} - E^{-\frac{1}{2}}$

由不变算子和移位算子, 可得如下性质:

各阶差分均可用函数值表示, 即

$$\begin{aligned}\Delta^n f_k &= (E - I)^n f_k = \sum_{j=0}^n (-1)^j \binom{n}{j} E^{n-j} f_k = \sum_{j=0}^n (-1)^j \binom{n}{j} f_{k+n-j} \\ \nabla^n f_k &= (1 - E^{-1})^n f_k = \sum_{j=0}^n (-1)^{n-j} \binom{n}{j} E^{j-n} f_k = \sum_{j=0}^n (-1)^{n-j} \binom{n}{j} f_{k+j-n}\end{aligned}$$

其中 $\binom{n}{j} = \frac{n(n-1)\cdots(n-j+1)}{j!}$ 为二项式展开系数

差商与差分之间满足下面的关系. 例如, 对于向前差分, 由定义

$$\begin{aligned} f[x_k, x_{k+1}] &= \frac{f_{k+1} - f_k}{x_{k+1} - x_k} = \frac{\Delta f_k}{h} \\ f[x_k, x_{k+1}, x_{k+2}] &= \frac{f[x_{k+1}, x_{k+2}] - f[x_k, x_{k+1}]}{x_{k+2} - x_k} = \frac{1}{2h^2} \Delta^2 f_k \end{aligned}$$

一般有

$$f[x_k, x_{k+1}, \dots, x_{k+m}] = \frac{1}{m!} \frac{1}{h^m} \Delta^m f_k, m = 1, 2, \dots, n \quad (2.7)$$

同理, 对于向后差分, 有

$$f[x_k, x_{k-1}, \dots, x_{k-m}] = \frac{1}{m!} \frac{1}{h^m} \nabla^m f_k \quad (2.8)$$

利用差商的导数关系, 可得差分与导数的关系为

$$\Delta^n f_k = j^n f^{(n)}(\xi)$$

2.4.3 等距节点插值公式

利用 Newton 插值公式, 将差商用相应的差分替代, 即可得到等距节点插值公式.

若节点 $x_k = x_0 + kh, k = 0, 1, \dots, n$, 要计算 x_0 附近点 x 的函数 $f(x)$ 的值, 令 $x = x_0 + th$, 其中 $0 \leq t \leq 1$, 于是有

$$\omega_{k+1}(x) = \prod_{j=0}^k (x - x_j) = t(t-1) \cdots (t-k) h^{k+1}$$

将该式与式 (2.7) 代入公式 (2.6), 可得

$$N_n(x_0 + th) = f_0 + t\Delta f_0 + \frac{t(t-1)}{2!} \Delta^2 f_0 + \cdots + \frac{t(t-1) \cdots (t-n+1)}{n!} \Delta^n f_0$$

上式称为 Newton 前插公式. 同时, 由于其使用 x_0 附近的点, 故又称为表初公式.

不难得到, 上式的余项可由式 (2.5) 得到

$$R_n(x) = \frac{t(t-1) \cdots (t-n)}{(n+1)!} h^{n+1} f^{(n+1)}(\xi), \xi \in (x_0, x_n)$$

类似地, 若要计算 x_n 附近的值 $f(x)$, 则可将 Newton 插值公式 (??) 按照倒序排列, 即 x_n, x_{n-1}, \dots, x_0 的次序. 此时有

$$N_n(x) = f(x_n) + f[x_n, x_{n-1}](x - x_n) + f[x_n, x_{n-1}, x_{n-2}](x - x_n)(x - x_{n-1}) \\ + \cdots + f[x_n, x_{n-1}, \cdots, x_0](x - x_n) \cdots (x - x_1)$$

变换 $x = x_n - th$ ($-1 \leq t \leq 0$), 并利用公式 (2.8), 代入可得

$$N_n(x_n + th) = f_n + t\nabla f_n + \frac{t(t+1)}{2!}\nabla^2 f_n + \cdots + \frac{t(t+1)\cdots(t+n-1)}{n!}\nabla^n f_n$$

上式称为 *Newton* 后插公式或表末公式. 其余项为

$$R_n(x) = f(x) - N_n(x_n + th) = \frac{t(t+1)\cdots(t+n)h^{n+1}f^{(n+1)}(\xi)}{(n+1)!}, \xi \in (x_0, x_n)$$

2.5 Hermite 插值

2.5.1 引入, Hermite 插值多项式的存在唯一性

在有些情况下, 不仅要求节点上函数值相等, 同时要求节点上导数值相等, 甚至要求高阶导数值相等. 满足该要求的插值多项式就是 *Hermite* 插值多项式.

仅讨论函数值与导数值个数相等情况. 设在节点 $a \leq x_0 < x_1 < \cdots < x_n \leq b$ 上, $y_i = f(x_i)$, $m_i = f'(x_i)$ ($i = 0, 1, \cdots, n$), 要求插值多项式 $H(x)$ 满足条件

$$H(x_i) = y_i, H'(x_i) = m_i, (i = 0, 1, \cdots, n)$$

给出 $(2n+2)$ 个条件, 可唯一确定一个次数不超过 $2n+1$ 的多项式 H_{2n+1} , 其形式为

$$H_{2n+1}(x) = a_0 + a_1x + \cdots + a_{2n+1}x^{2n+1}$$

2.5.2 Hermite 插值多项式构造

Lagrange 型 Hermite 插值多项式

先求插值基函数 $\alpha_i(x)$ 和 $\beta_i(x)$, $i = 0, 1, \cdots, n$, 共有 $2n+2$ 个, 每一个都是 $2n+1$ 次多项式, 且满足条件:

$$\begin{cases} \alpha_i(x_k) = \delta_{ik} = \begin{cases} 0, j \neq k, \\ 1, j = k, \end{cases} & , \alpha'_i(x_k) = 0 \\ \beta_i(x_k) = 0, \beta'_i(x_k) = \delta_{ik} \end{cases}$$

因此, 满足插值条件的插值多项式 $H(x) = H_{2n+1}(x)$ 可用插值基函数表示为

$$H_{2n+1}(x) = \sum_{i=0}^n [y_i \alpha_i(x) + m_i \beta_i(x)]$$

显然有

$$H_{2n+1}(x_k) = y_k, H'_{2n+1}(x_k) = m_k, k = 0, 1, \dots, n$$

为构造基函数, 使用 Lagrange 插值基函数 $l_i(x)$, 令

$$\alpha_i(x) = (ax + b)l_i^2(x)$$

其中 $l_i(x)$ 是式 (??) 所表示的基函数. 由上述条件, 有

$$\alpha_i(x_i) = (ax_i + b)l_i^2(x_i) = 1,$$

$$\alpha'_i(x_i) = l_i(x_i) [al_i(x_i) + 2(ax_i + b)l'_i(x_i)] = 0$$

整理可得

$$\begin{cases} ax_i + b = 1 \\ a + 2l'_i(x_i) = 0 \end{cases}$$

解得

$$a = -2l'_i(x_i), b = 1 + 2x_i l'_i(x_i)$$

由于

$$l_i(x) = \frac{(x - x_0) \cdots (x - x_{i-1})(x - x_{i+1}) \cdots (x - x_n)}{(x_i - x_0) \cdots (x_i - x_{i-1})(x_i - x_{i+1}) \cdots (x_i - x_n)}$$

取对数后求导, 得

$$l'_i(x_i) = \sum_{\substack{k=0 \\ k \neq i}}^n \frac{1}{x_i - x_k}$$

于是可得

$$\alpha_i(x) = \left[1 - 2(x - x_i) \sum_{\substack{k=0 \\ k \neq i}}^n \frac{1}{x_i - x_k} \right] l_i^2(x) \quad (2.9)$$

同理可得

$$\beta_i(x) = (x - x_i)l_i^2(x) \quad (2.10)$$

扩展: 下面证明满足插值条件得插值多项式是唯一的.

证明. 使用反证法, 设 $H_{2n+1}(x)$ 和 $\overline{H}_{2n+1}(x)$ 均满足条件, 则有

$$\varphi(x) = H_{2n+1}(x) - \overline{H}_{2n+1}(x)$$

其在每个节点上均有二重根, 故 $\varphi(x)$ 有 $2n+2$ 重根, 但 $\varphi(x)$ 是不高于 $2n+1$ 次的多项式, 故 $\varphi \equiv 0$.

唯一性得证. \square

利用 Lagrange 插值余项证明方法, 可得若 $f(x)$ 在 (a, b) 内的 $2n+2$ 阶导数存在, 则插值余项为

$$R(x) = f(x) - H_{2n+1}(x) = \frac{f^{(2n+2)}(\xi)}{(2n+2)!} \omega_{n+1}^2(x) \quad (2.11)$$

其中 $\xi \in (a, b)$ 且与 x 有关.

考虑特殊情况 $n = 1$, 取节点为 x_k, x_{k+1} , 插值多项式为 $H_3(x)$, 满足条件

$$\begin{cases} H_3(x_k) = y_k, H_3(x_{k+1}) = y_{k+1} \\ H'_3(x_k) = m_k, H'_3(x_{k+1}) = m_{k+1} \end{cases} \quad (2.12)$$

相应的插值基函数为 $\alpha_k(x), \alpha_{k+1}(x), \beta_k(x), \beta_{k+1}(x)$, 且满足条件

$$\begin{aligned} \alpha_k(x_k) &= 1, \alpha_k(x_{k+1}) = 0, \alpha'_k(x_k) = \alpha'_k(x_{k+1}) = 0, \\ \alpha_{k+1}(x_k) &= 0, \alpha_{k+1}(x_{k+1}) = 1, \\ \alpha'_{k+1}(x_k) &= \alpha'_{k+1}(x_{k+1}) = 0, \\ \beta_k(x_k) &= \beta_k(x_{k+1}) = 0, \\ \beta'_k(x_k) &= 1, \beta'_k(x_{k+1}) = 0, \\ \beta_{k+1}(x_k) &= \beta_{k+1}(x_{k+1}) = 0, \\ \beta'_{k+1}(x_k) &= 0, \beta'_{k+1}(x_{k+1}) = 1 \end{aligned}$$

根据式 (2.9) 和 (2.10), 可得

$$\begin{cases} \alpha_k(x) = \left(1 + 2 \frac{x-x_k}{x_{k+1}-x_k}\right) \left(\frac{x-x_{k+1}}{x_k-x_{k+1}}\right)^2, \\ \alpha_{k+1}(x) = \left(1 + 2 \frac{x-x_{k+1}}{x_k-x_{k+1}}\right) \left(\frac{x-x_k}{x_{k+1}-x_k}\right)^2; \\ \beta_k(x) = (x-x_k) \left(\frac{x-x_{k+1}}{x_k-x_{k+1}}\right)^2, \\ \beta_{k+1}(x) = (x-x_{k+1}) \left(\frac{x-x_k}{x_{k+1}-x_k}\right)^2 \end{cases}$$

满足式 (2.12) 的插值多项式是

$$H_3(x) = y_k \alpha_k(x) + y_{k+1} \alpha_{k+1}(x) + m_k \beta_k(x) + m_{k+1} \beta_{k+1}(x)$$

由式 (2.11) 可得其余项为

$$R_3(x) = \frac{1}{4!} f^{(4)}(\xi)(x - x_k)^2(x - x_{k+1})^2$$

Newton 型 Hermite 插值多项式

若给定插值条件

$$H(x_0) = f(x_0), H'(x_0) = f'(x_0), \dots, H^{(m)}(x_0) = f^{(m)}(x_0)$$

利用 Newton 插值法, 其节点为 x_0, x_0, \dots, x_0 , 基函数分别为 $1, (x - x_0), (x - x_0)^2, \dots, (x - x_0)^m$, 组合系数为 $f(x_0), f[x_0, x_0], \dots, f[x_0, x_0, \dots, x_0]$ 考虑差商与导数的性质, 有

$$f[x_0, x_0, \dots, x_0] = \frac{f^{(n)}}{n!}$$

故容易得到, Newton 型 Hermite 插值多项式为

$$H(x) = f(x_0) + f'(x_0)(x - x_0) + \dots + \frac{f^{(m)}}{m!}(x - x_0)^m$$

例 2.5.1. 给定函数表 求插值多项式 $P_4(x)$

x_i	3	4	6
y_i	6	0	2
y'_i	1		-1

解. 利用 Newton 法, 列差商表如下所示 将差商表计算可得

$$P_4 = 6 + 1(x-3) + (-7)(x-3)^2 + \frac{28}{9}(x-3)^2(x-4) + \left(-\frac{38}{27}\right)(x-3)^2(x-4)(x-6)$$

其插值余项为

$$R_4(x) = f[3, 3, 4, 6, 6, x](x-3)^2(x-4)(x-6)^2$$

□

表 2.1: 差商表

x_k	$f(x_k)$	一阶差商	二阶差商	三阶差商	四阶差商
3	6				
3	6	$f[3, 3] = 1$			
4	0	$f[3, 4]$	$f[3, 3, 4]$		
6	2	$f[4, 6]$	$f[3, 4, 6]$	$f[3, 3, 4, 6]$	
6	2	$f[6, 6] = -1$	$f[4, 6, 6]$	$f[3, 4, 6, 6]$	$f[3, 3, 4, 6, 6]$

2.6 分段低次插值

2.6.1 多项式插值的 Runge 现象

例 2.6.1. 在 $[-5, 5]$ 考察函数

$$f(x) = \frac{1}{1+x^2}$$

的 $L_n(x)$

使用下面的代码:

```

1  # Runge现象演示 Exercise2-3.py
2
3  import numpy as np
4  import matplotlib.pyplot as plt
5  from scipy.interpolate import lagrange
6
7  # 定义函数
8  def f(x):
9      return 1 / (1 + x ** 2)
10
11 # 生成插值节点
12 n = 11
13 x = np.linspace(-5, 5, n)
14 y = f(x)
15
16 # 生成插值多项式

```

```

17 poly = lagrange(x, y)
18
19 # 生成等距节点
20 x_interp = np.linspace(-5, 5, 1000)
21 y_interp = f(x_interp)
22
23 # 生成插值函数
24 y_poly = poly(x_interp)
25
26 # 绘图
27 plt.plot(x_interp, y_interp)
28 plt.plot(x_interp, y_poly)
29 plt.plot(x, y, 'o')
30 plt.show()

```

生成图像如图 (2.2) 所示

可以看到, 随着节点的增多, 插值多项式在节点处出现振荡. 这就是多项式插值的 *Runge* 现象. 为解决这一问题, 采用分段插值的方法.

2.6.2 分段线性插值

在每个区间 $[x_i, x_{i+1}]$ 上, 用一阶多项式逼近 $f(x)$, 即

$$f(x) = P_1(x) = \frac{x - x_{i+1}}{x_i - x_{i+1}}y_i + \frac{x - x_i}{x_{i+1} - x_i}y_{i+1}, x \in [x_i, x_{i+1}]$$

若用插值基函数表示, 若整个节点在 $[a, b]$ 之间, 则在整个区间的 $P_1(x)$ 可以表示为

$$P_1(x) = \sum_{i=0}^n y_i l_i(x)$$

其中 $l_i(x)$ 满足条件 $l_i(x_k) = \delta_{ik}, i, k = 0, 1, \dots, n$, 其形式为

$$l_i(x) = \begin{cases} \frac{x - x_{i-1}}{x_i - x_{i-1}}, & x_{i-1} \leq x \leq x_i (i = 0 \text{ 略去}), \\ \frac{x_i - x_{i+1}}{x - x_{i+1}}, & x_i \leq x \leq x_{i+1} (i = n \text{ 略去}), \\ 0, & x \in [a, b], x \notin [x_{i-1}, x_{i+1}]. \end{cases}$$

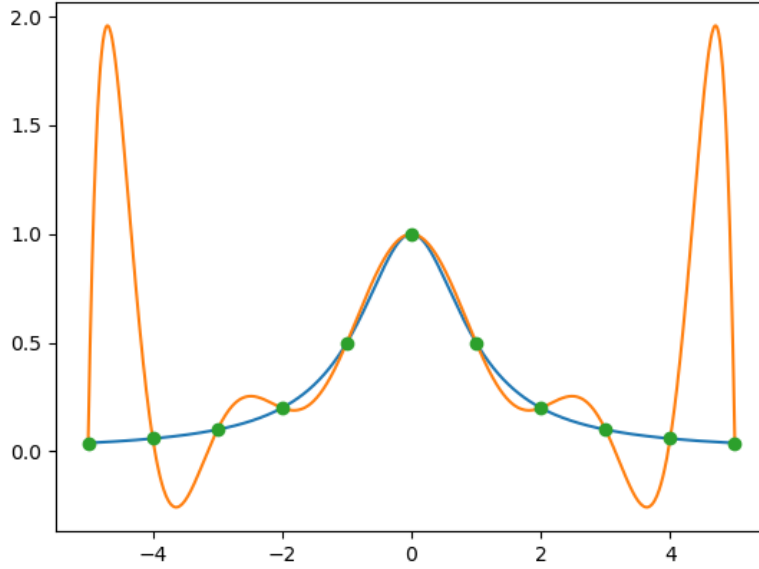


图 2.2: Runge 现象

不难发现, 基函数 $l_i(x)$ 在 x_i 附近不为 0, 其他地方均为零, 这种性质称为局部非零性质.

关于分段线性插值的误差估计, 给出如下定理 (暂不做证明)

定理 2.6.1. 若 $f(x)$ 在 $[a, b]$ 上二阶连续可微, 则分段线性插值 $P(x)$ 余项有如下估计:

$$|R(x)| = |f(x) - P(x)| \leq \frac{h^2}{8} M$$

其中,

$$h = \max_{0 \leq i \leq n-1} (x_{i+1} - x_i)$$

$$M = \max_{a \leq x \leq b} |f''(x)|$$

下面的程序, 演示了分段线性插值 (图像如图 2.3所示). 可以看到, 分段线性插值在节点处不光滑. 为了改善该问题, 我们可以借助于 Hermite 插值, 即下面的分段三次 Hermite 插值.

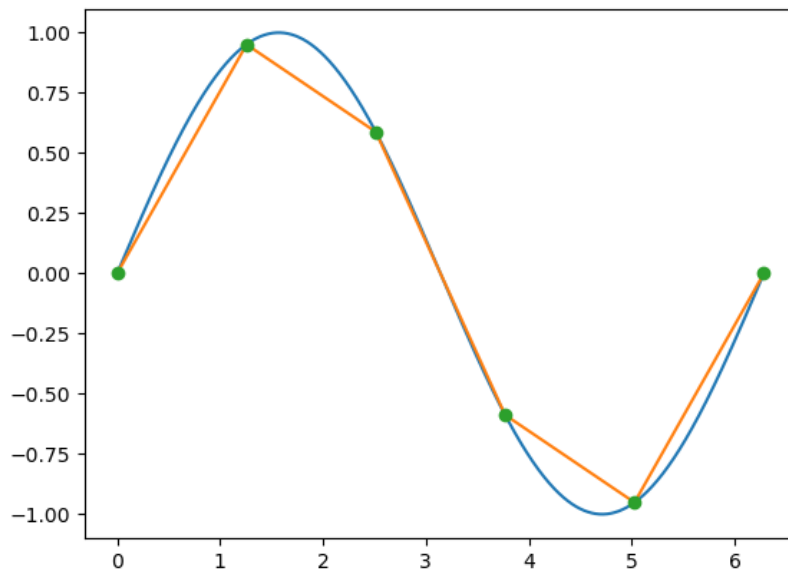
```
1  # 分段线性插值演示 Exercise2-4.py
2
3  import numpy as np
4  import matplotlib.pyplot as plt
5
6  # 定义函数
7  def f(x):
8      return np.sin(x)
9
10 # 生成插值节点
11 n = 6
12 x = np.linspace(0, 2*np.pi, n)
13 y = f(x)
14
15 # 生成分段线性插值函数
16 x_interp = np.linspace(0, 2*np.pi, 1000)
17 y_interp = np.zeros_like(x_interp)
18 for i in range(len(x_interp)):
19     if x_interp[i] <= x[0]:
20         y_interp[i] = y[0]
21     elif x_interp[i] >= x[-1]:
22         y_interp[i] = y[-1]
23     else:
24         j = np.searchsorted(x, x_interp[i])
25         x_left = x[j-1]
26         x_right = x[j]
27         y_left = y[j-1]
28         y_right = y[j]
29         slope = (y_right - y_left) / (x_right - x_left)
30         y_interp[i] = y_left + slope * (x_interp[i] - x_left)
31
32 # 绘图
33 plt.plot(x_interp, f(x_interp))
```



```

34 plt.plot(x_interp, y_interp)
35 plt.plot(x, y, 'o')
36 plt.show()

```

图 2.3: $\sin x$ 的分段线性插值

2.6.3 分段三次 Hermite 插值

给定节点 $a \leq x_0 < x_1 < \cdots \leq b$, $f(x)$ 在节点 x_i 上的函数值及导数值分别为 y_i, m_i . 在每个子区间 $[x_i, x_{i+1}]$ 上作两点三次 Hermite 插值, 插值函数为

$$H(x) = \sum_{i=0}^n [y_i \varphi_i(x) + m_i \psi_i(x)] \quad (2.13)$$

其中基函数为

$$\varphi_i(x) = \begin{cases} \left(\frac{x-x_{i-1}}{x_i-x_{i-1}}\right)^2 \left(1 + 2\frac{x-x_i}{x_{i-1}-x_i}\right), & x_{i-1} \leq x \leq x_i, \\ \left(\frac{x-x_{i+1}}{x_i-x_{i+1}}\right)^2 \left(1 + 2\frac{x-x_i}{x_{i+1}-x_i}\right), & x_i \leq x \leq x_{i+1}, \\ 0, & x \notin [x_{i-1}, x_{i+1}] \end{cases}, (i = 1, 2, \dots, n-1).$$

$$\psi_i(x) = \begin{cases} (x-x_i) \left(\frac{x-x_{i-1}}{x_i-x_{i-1}}\right)^2, & x_{i-1} \leq x \leq x_i, \\ (x-x_i) \left(\frac{x-x_{i+1}}{x_i-x_{i+1}}\right)^2, & x_i \leq x \leq x_{i+1}, \\ 0, & x \notin [x_{i-1}, x_{i+1}] \end{cases}$$

由三次 Hermite 插值的余项可以估计分段 Hermite 插值的余项. 设 $H(x)$ 是给定节点

$$a = x_0 < x_1 < \dots < x_n = b$$

上的分段三次 Hermite 插值函数, $f(x) \in C^4[a, b]$, 则 $f(x)$ 与 $H(x)$ 的误差限为

$$|R(x)| \leq |f(x) - H(x)| \leq \frac{h^4}{384} M_4$$

其中,

$$h = \max_{0 \leq i \leq n-1} |x_{i+1} - x_i|,$$

$$M = \max_{a \leq x \leq b} |f^{(4)}(x)|$$

下面的代码演示了分段三次 Hermite 插值, 并绘制了图 2.4. 可以看到, 与线性插值 (图 2.3) 相比, 三次 Hermite 插值具有光滑性.

扩展: 在该代码中使用了 *Python* 的库函数 *PchipInterpolator()*, 这个函数会在内部自动计算导数值, 因此与前面所讨论的理论在形式上可能有些许不同.

```

1  # 分段三次Hermite插值演示 Exercise2-5.py
2
3  import numpy as np
4  import matplotlib.pyplot as plt
5  from scipy.interpolate import PchipInterpolator
6
7  # 定义函数

```

```
8 def f(x):
9     return np.sin(x)
10
11 # 生成插值节点
12 n = 5
13 x = np.linspace(0, 2*np.pi, n)
14 y = f(x)
15
16 # 生成分段三次Hermite插值函数
17 x_interp = np.linspace(0, 2*np.pi, 1000)
18 y_interp = PchipInterpolator(x, y)(x_interp)
19
20 # 绘图
21 plt.plot(x_interp, f(x_interp))
22 plt.plot(x_interp, y_interp)
23 plt.plot(x, y, 'o')
24 plt.show()
```

2.7 三次样条插值

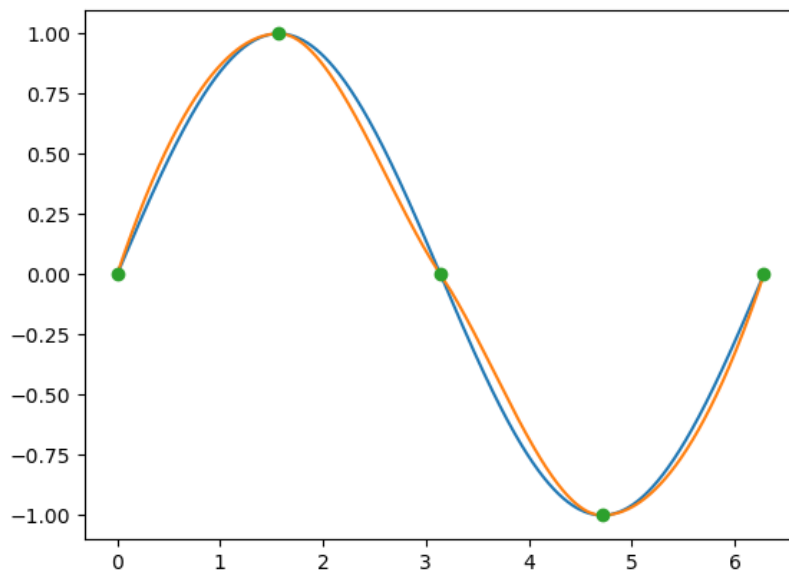
2.7.1 三次样条函数

设对 $y = f(x)$ 在区间 $[a, b]$ 上给定一组节点 $a = x_0 < x_1 < \cdots < x_n = b$ 和相应的函数值 y_0, y_1, \cdots, y_n , 如果 $s(x)$ 具有如下性质:

1. 在每个子区间 $[x_{i-1}, x_i], i = 1, 2, \cdots, n$ 上 $s(x)$ 是不高于三次的多项式;
2. $s(x), s'(x), s''(x)$ 在 $[a, b]$ 上连续, 则称 $s(x)$ 是三次样条函数. 若再有
3. $s(x_i) = f(x_i), i = 0, 1, \cdots, n$, 则称 $s(x)$ 为 $y = f(x)$ 的三次样条插值函数.

2.7.2 三次样条插值函数构造

首先对三次样条插值的存在性进行证明

图 2.4: $\sin x$ 的三次 Hermite 插值

证明. 设有 n 个区间, 对于每个区间 S_i , 插值函数为

$$S_i(x) = a_i + b_i x + c_i x^2 + d_i x^3, i = 0, 1, \dots, n-1$$

对于每一个小区间, 两个端点给定, 总区间共有 $2n$ 个条件.

对于节点 x_i , 其一阶导数连续, 即

$$S'_{i-1}(x_i) = S'_i(x_i)$$

共 $(n-1)$ 个条件.

同理, 其二阶导数连续, 有 $(n-1)$ 个条件.

因此, 给定条件共 $(4n-2)$ 个, 小于 $4n$, 因此解存在但不唯一. \square

为保证解的唯一性, 需要补充边界条件. 常见的边界条件有如下四种:

固支边界条件 (D-1 样条):

$$\begin{cases} S'(x_0) = f'(x_0) = m_0 \\ S'(x_n) = f'(x_n) = m_n \end{cases}$$

弯矩边界条件 (D-2 样条):

$$\begin{cases} S'''(x_0) = f''(x_0) = M_0 \\ S'''(x_n) = f''(x_n) = M_n \end{cases}$$

自然边界条件 (自然样条):

$$\begin{cases} S''(x_0) = 0 \\ S''(x_n) = 0 \end{cases}$$

周期边界条件 (周期样条):

$$\begin{cases} S(x_0) = S(x_n) \\ S'(x_0) = S'(x_n) \\ S''(x_0) = S''(x_n) \end{cases}$$

2.7.3 三转角方程

假定 $S'(x)$ 在节点 x_i 的值为 $S'(x_i) = m_i, i = 0, 1, \dots, n$, 由式 (2.13) 可得

$$S(x) = \sum_{i=0}^n [y_i \varphi_i(x) + m_i \psi_i(x)]$$

显然, 该表达式在 $[a, b]$ 区间连续, 且满足插值条件. 而式中的导数值 m_i 未知, 因此可利用导数的连续条件:

$$S''(x_i - 0) = S''(x_i + 0), i = 1, 2, \dots, n-1$$

以及边界条件确定.

下面考虑 $S(x)$ 在 $[x_i, x_{i+1}]$ 的表达式

$$\begin{aligned} S(x) = & \frac{(x - x_{i+1})^2 [h_i + 2(x - x_i)]}{h_i^3} y_i + \frac{(x - x_i)^2 [h_i + 2(x_{i+1} - x)]}{h_i^3} y_{i+1} \\ & + \frac{(x - x_{i+1})^2 (x - x_i)}{h_i^2} m_i + \frac{(x - x_i)^2 (x - x_{i+1})}{h_i^2} m_{i+1} \end{aligned}$$

这里 $h_i = x_{i+1} - x_i$, 对 $S(x)$ 求二阶导数, 得

$$\begin{aligned} S''(x) = & \frac{6x - 2x_i - 4x_{i+1}}{h_i^2} m_i + \frac{6x - 4x_i - 2x_{i+1}}{h_i^2} m_{i+1} \\ & + \frac{6(x_i + x_{i+1} - 2x)}{h_i^3} (y_{i+1} - y_i) \end{aligned}$$

于是

$$S''(x_i + 0) = -\frac{4}{h_i}m_i - \frac{2}{h_i}m_{i+1} + \frac{6}{h_i^2}(y_{i+1} - y_i)$$

同理, 可得 $S''(x)$ 在区间 $[x_{i-1}, x_i]$ 得表达式为

$$\begin{aligned} S''(x) &= \frac{6x - 2x_{i-1} - 4x_i}{h_{i-1}^2}m_{i-1} + \frac{6x - 4x_{i-1} - 2x_i}{h_{i-1}^2}m_i \\ &\quad + \frac{6(x_{i-1} + x_i - 2x)}{h_{i-1}^2}(y_i - y_{i-1}) \end{aligned}$$

及

$$S''(x_i - 0) = \frac{2}{h_{i-1}}m_{i-1} + \frac{4}{h_{i-1}}m_i - \frac{6}{h_{i-1}^2}(y_i - y_{i-1})$$

利用二阶导数连续性, 化简可得结果为

$$\lambda_i m_{i-1} + 2m_i + \mu_i m_{i+1} = g_i, i = 1, 2, \dots, n-1 \quad (2.14)$$

其中,

$$\begin{aligned} \lambda_i &= \frac{h_i}{h_{i-1} + h_i}, \mu_i = \frac{h_{i-1}}{h_{i-1} + h_i} \\ g_i &= 3(\lambda_i f[x_{i-1}, x_i] + \mu_i f[x_i, x_{i+1}]) \end{aligned}$$

方程是关于未知数 m_0, m_1, \dots, m_n 的 $n-1$ 个方程, 若加上固支边界条件 $m_0 = f'_0, m_n = f'_n$, 则方程为含有 m_1, \dots, m_{n-1} 的 $(n-1)$ 个方程, 使用矩阵形式可表示为

$$\begin{pmatrix} 2 & \mu_1 & 0 & \cdots & \cdots & 0 \\ \lambda_2 & 2 & \mu_2 & \ddots & & \vdots \\ 0 & \lambda_3 & 2 & \mu_3 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & & \ddots & \lambda_{n-2} & 2 & \mu_{n-2} \\ 0 & \cdots & \cdots & 0 & \lambda_{n-1} & 2 \end{pmatrix} \begin{pmatrix} m_1 \\ m_2 \\ m_3 \\ \vdots \\ m_{n-2} \\ m_{n-1} \end{pmatrix} = \begin{pmatrix} g_1 - \lambda_1 f'_0 \\ g_2 \\ \vdots \\ g_{n-2} \\ g_{n-1} - \mu_{n-1} f'_n \end{pmatrix}$$

如果边界条件为弯矩边界条件 (或特殊情况下的自然边界条件), 则方程

组为

$$\begin{pmatrix} 2 & 1 & 0 & \cdots & \cdots & 0 \\ \lambda_1 & 2 & \mu_1 & \ddots & & \vdots \\ 0 & \lambda_2 & 2 & \mu_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & & \ddots & \lambda_{n-1} & 2 & \mu_{n-1} \\ 0 & \cdots & \cdots & 0 & 1 & 2 \end{pmatrix} \begin{pmatrix} m_0 \\ m_1 \\ m_2 \\ \vdots \\ m_{n-1} \\ m_n \end{pmatrix} = \begin{pmatrix} g_0 \\ g_1 \\ g_2 \\ \vdots \\ g_{n-1} \\ g_n \end{pmatrix}$$

若采用周期性边界条件, 则

$$m_0 = m_n$$

从而整理得方程组为

$$\begin{pmatrix} 2 & \mu_1 & 0 & \cdots & 0 \\ \lambda_2 & 2 & \mu_2 & \ddots & \vdots \\ 0 & \lambda_3 & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & 2 & \mu_{n-1} \\ 0 & \cdots & 0 & \lambda_n & 2 \end{pmatrix} \begin{pmatrix} m_1 \\ m_2 \\ \vdots \\ m_{n-1} \\ m_n \end{pmatrix} = \begin{pmatrix} g_1 \\ g_2 \\ \vdots \\ g_{n-1} \\ g_n \end{pmatrix}$$

上述方程组当中, 每个方程都联系三个 m_i , 故称为三转角方程. 同时, 三个方程组的系数矩阵都为严格对角占优矩阵, 故方程有唯一解.

扩展:

定义 2.7.1 (严格对角占优矩阵). 若方阵 $A = (a_{ij}) \in R^{n \times n}$, 满足

$$|a_{ii}| > \sum_{j=1, j \neq i}^n |a_{ij}|, i = 1, 2, \cdots, n$$

称 A 为严格对角占优矩阵

2.7.4 三弯矩方程

设函数 $f(x)$ 是定义在区间 $[a, b]$ 上的一个二次连续可微函数, 节点 $a = x_0 < x_1 < \cdots < x_n = b$, 令

$$M_i = S''(x_i), i = 0, 1, \cdots, n$$

其中 $S(x)$ 在每一个小区间 $[x_i, x_{i+1}]$, $i = 0, 1, \dots, n-1$ 上都是三次多项式, 故 $S''(x)$ 在 $[x_i, x_{i+1}]$ 的表达式为:

$$S''_i(x) = M_i \frac{x_{i+1} - x}{h_{i+1}} + M_{i+1} \frac{x - x_i}{h_{i+1}}$$

其中, $h_{i+1} = x_{i+1} - x_i$, 将上式做两次积分, 得

$$\begin{aligned} S'_i(x) &= -M_i \frac{(x_{i+1} - x)^2}{2h_{i+1}} + M_{i+1} \frac{(x - x_i)^2}{2h_{i+1}} + A_i \\ S_i(x) &= M_i \frac{(x_{i+1} - x)^3}{6h_{i+1}} + M_{i+1} \frac{(x - x_i)^3}{6h_{i+1}} + A_i(x - x_{i+1}) + B_i \end{aligned}$$

其中, A_i, B_i 为积分常数. 由插值条件并整理可得

$$\begin{cases} A_i = \frac{y_{i+1} - y_i}{h_{i+1}} - \frac{h_{i+1}}{6}(M_{i+1} - M_i) \\ B_i = y_{i+1} - \frac{M_{i+1}}{6}h_{i+1}^2 \end{cases}$$

将其代入, 可得

$$\begin{aligned} S_i(x) &= M_i \frac{(x_{i+1} - x)^3}{6h_{i+1}} + M_{i+1} \frac{(x - x_i)^3}{6h_{i+1}} \\ &\quad + \left(y_i - \frac{M_i}{6}h_{i+1}^2 \right) \frac{x_{i+1} - x}{h_{i+1}} + \left(y_{i+1} - \frac{M_{i+1}}{6}h_{i+1}^2 \right) \frac{x - x_i}{h_{i+1}}, i = 0, 1, \dots, n-1 \end{aligned}$$

对上式进行微分, 有

$$\begin{aligned} S'_i(x) &= -M_i \frac{(x_{i+1} - x)^2}{2h_{i+1}} + M_{i+1} \frac{(x - x_i)^2}{2h_{i+1}} + \frac{y_{i+1} - y_i}{h_{i+1}} - \frac{h_{i+1}}{6}(M_{i+1} - M_i) \\ S'_{i-1}(x) &= -M_{i-1} \frac{(x_i - x)^2}{2h_i} + M_i \frac{(x - x_{i-1})^2}{2h_i} + \frac{y_i - y_{i-1}}{h_i} - \frac{h_i}{6}(M_i - M_{i-1}) \\ i &= 1, 2, \dots, n-1 \end{aligned}$$

因此有

$$\begin{aligned} S'_{i-1}(x_i) &= \frac{h_i}{3}M_i + \frac{y_i - y_{i-1}}{h_i} + \frac{h_i}{6}M_{i-1} \\ S'_i(x_i) &= \frac{h_{i+1}}{3}M_i + \frac{y_{i+1} - y_i}{h_{i+1}} + \frac{h_{i+1}}{6}M_{i+1} \end{aligned}$$

由于 $S'_{i-1}(x_i) = S'_i(x_i)$, 从而有

$$h_i M_{i-1} + 2(h_i + h_{i+1})M_i + h_{i+1}M_{i+1} = 6 \left(\frac{y_{i+1} - y_i}{h_{i+1}} - \frac{y_i - y_{i-1}}{h_i} \right), i = 1, 2, \dots, n-1$$

记

$$\begin{aligned}\mu_i &= \frac{h_{i+1}}{h_i + h_{i+1}} \\ \lambda_i &= 1 - \mu_i \\ d_i &= 6 \left[\frac{y_{i+1} - y_i}{h_{i+1}} - \frac{y_i - y_{i-1}}{h_i} \right] / (h_i + h_{i+1}) = 6f[x_{i-1}, x_i, x_{i+1}]\end{aligned}$$

可将上式改写为

$$\lambda_i M_{i-1} + 2M_i + \mu_i M_{i+1} = d_i, i = 1, 2, \dots, n-1$$

可以发现, 上式与式 (2.14) 具有相同的形式, 因此所得到的线性方程组的形式也与上面相同.

对于固支边界条件, 给定两端点导数值 $S'(x_0) = y'_0, S'(x_n) = y'_n$, 有

$$\begin{aligned}S'_0(x_0) &= \frac{y_1 - y_0}{h_1} - \frac{h_1}{3}M_0 - \frac{h_1}{6}M_1 = y'_0 = f[x_0, x_0] \\ S'_{n-1}(x_n) &= \frac{y_n - y_{n-1}}{h_n} + \frac{h_n}{6}M_{n-1} + \frac{h_n}{3}M_n = y'_n = f[x_n, x_n]\end{aligned}$$

于是可得

$$\begin{aligned}2M_0 + M_1 &= \frac{6}{h_1} \left(\frac{y_1 - y_0}{h_1} - y'_0 \right) = 6f[x_0, x_0, x_1] \\ M_{n-1} + 2M_n &= \frac{6}{h_n} \left(y'_n - \frac{y_n - y_{n-1}}{h_n} \right) = 6f[x_{n-1}, x_n, x_n]\end{aligned}$$

将其补充为方程组第一个和最后一个方程, 即可得到三弯矩方程为

$$\begin{pmatrix} 2 & 1 & & & & \\ \lambda_1 & 2 & \mu_1 & & & \\ & \lambda_2 & 2 & \mu_2 & & \\ & & & \ddots & & \\ & & & & \lambda_{n-1} & 2 & \mu_{n-1} \\ & & & & & 1 & 2 \end{pmatrix} \begin{pmatrix} M_0 \\ M_1 \\ M_2 \\ \vdots \\ M_{n-1} \\ M_n \end{pmatrix} = \begin{pmatrix} d_0 \\ d_1 \\ d_2 \\ \vdots \\ d_{n-1} \\ d_n \end{pmatrix}$$

其中,

$$\begin{aligned}d_0 &= 6f[x_0, x_0, x_1] \\ d_n &= 6f[x_{n-1}, x_n, x_n] \\ d_i &= 6f[x_{i-1}, x_i, x_{i+1}], i = 1, 2, \dots, n-1\end{aligned}$$

对于弯矩边界条件, 给定 $M_0 = y_0'', M_n = y_n''$, 方程组可整理为

$$\begin{pmatrix} 2 & \mu_1 & & & \\ \lambda_2 & 2 & \mu_2 & & \\ & \lambda_3 & 2 & \mu_3 & \\ & & & \ddots & \\ & & & & \lambda_{n-2} & 2 & \mu_{n-2} \\ & & & & & \lambda_{n-1} & 2 \end{pmatrix} \begin{pmatrix} M_1 \\ M_2 \\ M_3 \\ \vdots \\ M_{n-2} \\ M_{n-1} \end{pmatrix} = \begin{pmatrix} d_1 \\ d_2 \\ d_3 \\ \vdots \\ d_{n-2} \\ d_{n-1} \end{pmatrix}$$

其中,

$$\begin{aligned} d_1 &= 6f[x_0, x_1, x_2] - \lambda_1 f''(x_0) \\ d_{n-1} &= 6f[x_{n-2}, x_{n-1}, x_n] - \mu_{n-1} f''(x_n) \\ d_i &= 6f[x_{i-1}, x_i, x_{i+1}], i = 2, \dots, n-2 \end{aligned}$$

将上述方程组当中的 $M_n = M_0 = 0$, 可得自然边界条件, 不再赘述.

对于周期样条, 使用类似的方法, 可得方程组

$$\begin{pmatrix} 2 & \mu_1 & & & \lambda_1 \\ \lambda_2 & 2 & \mu_2 & & \\ & \lambda_3 & 2 & \mu_3 & \\ & & & \ddots & \\ & & & & \lambda_{n-2} & 2 & \mu_{n-2} \\ \mu_n & & & & & \lambda_n & 2 \end{pmatrix} \begin{pmatrix} M_1 \\ M_2 \\ M_3 \\ \vdots \\ M_{n-1} \\ M_n \end{pmatrix} = \begin{pmatrix} d_1 \\ d_2 \\ d_3 \\ \vdots \\ d_{n-1} \\ d_n \end{pmatrix}$$

其中,

$$d_i = 6f[x_{i-1}, x_i, x_{i+1}], i = 1, 2, \dots, n$$

可以验证, 上述系数矩阵也满足严格对角占优的定义.

2.7.5 三次样条插值函数的收敛性 *

本部分不做证明, 只做介绍.

定义 2.7.2 (一致范数). 设函数 $f(x)$ 是区间 $[a, b]$ 上的连续函数, 则记

$$\|f\|_\infty = \max_{a \leq x \leq b} |f(x)|$$

为函数 $f(x)$ 的一致范数

定理 2.7.1. 设被插值函数 $f(x) \in C^4[a, b]$, $S(x)$ 为满足自然边界条件的三次样条插值函数, 则在插值区间 $[a, b]$ 上成立余项估计式

$$\|f^{(k)} - S^{(k)}\|_{\infty} \leq c_k h^{4-k} \|f^{(4)}\|_{\infty}, k = 0, 1, 2$$

其中,

$$h = \max_{0 \leq i \leq n-1} h_i, h_i = x_{i+1} - x_i,$$

$$c_0 = \frac{1}{16}, c_1 = c_2 = \frac{1}{2}$$

最后, 通过一个三次样条插值的程序结束这一章. 对于 Python 程序而言, 我们调用 `CubicSpline()` 函数进行样条插值. 函数默认是采用自然边界条件, 即得到图 2.5所示:

```

1  # 三次样条插值演示(自然边界条件) Exercise2-6.py
2
3  import numpy as np
4  from scipy.interpolate import CubicSpline
5  import matplotlib.pyplot as plt
6
7  # 创建一些数据点
8  x = np.array([0, 1, 2, 3, 4, 5])
9  y = np.array([0, 2, 1, 3, 2, 0])
10
11 # 使用 CubicSpline 函数进行样条插值
12 cs = CubicSpline(x, y)
13
14 # 生成插值点
15 x_interp = np.linspace(0, 5, 100)
16 y_interp = cs(x_interp)
17
18 # 绘制原始数据和插值曲线
19 plt.plot(x, y, 'o')
20 plt.plot(x_interp, y_interp)
21 plt.show()

```

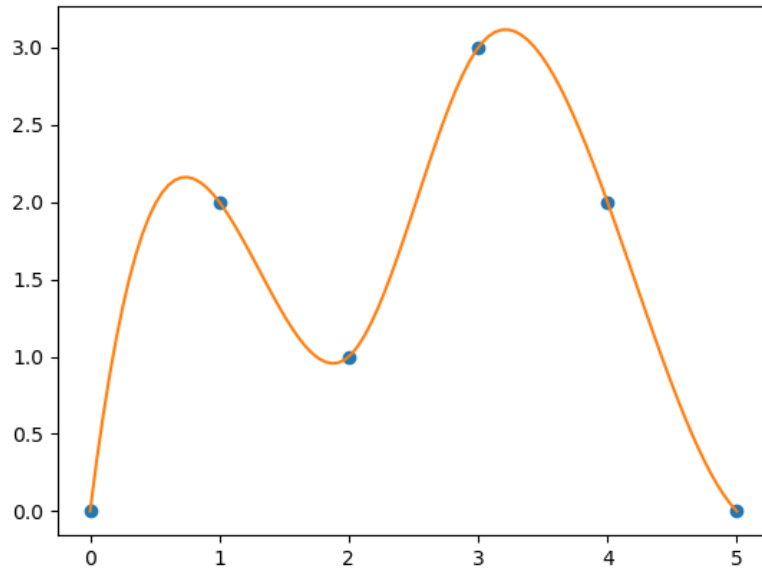


图 2.5: 自然边界条件的三次样条插值

若想采用其他边界条件, 如希望采用固支边界条件, 并令端点一阶导数为 0, 则可以在 Python 当中使用如下代码:

```
1  # 三次样条插值演示(固支边界条件) Exercise2-7.py
2
3  import numpy as np
4  from scipy.interpolate import CubicSpline
5  import matplotlib.pyplot as plt
6
7  # 创建一些数据点
8  x = np.array([0, 1, 2, 3, 4, 5])
9  y = np.array([0, 2, 1, 3, 2, 0])
10
11 # 使用CubicSpline函数进行样条插值, 且端点一阶导数值为0
12 cs = CubicSpline(x, y, bc_type=((1,0),(1,0)))
```

```
13  
14 # 生成插值点  
15 x_interp = np.linspace(0, 5, 100)  
16 y_interp = cs(x_interp)  
17  
18 # 绘制原始数据和插值曲线  
19 plt.plot(x, y, 'o')  
20 plt.plot(x_interp, y_interp)  
21 plt.show()
```

所得图像如图 2.6所示

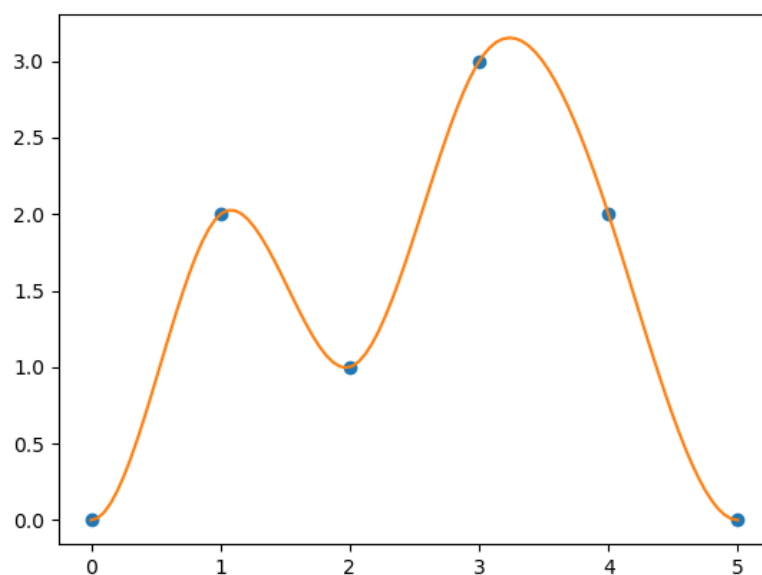


图 2.6: 固支边界条件的三次样条插值

Chapter 3

函数逼近与计算

3.1 引言

3.1.1 函数逼近的问题的一般提法

对于函数类 A 中给定的函数 $f(x)$, 要求在另一类较简单且便于计算的函数类 $B(\subset A)$ 中寻找一个函数 $P(x)$, 使得函数 $P(x)$ 与 $f(x)$ 之差在某种度量意义下最小.

本章所研究的函数类 A 通常为区间 $[a, b]$ 上的连续函数, 记作 $C[a, b]$; 函数类 B 通常为代数多项式或三角多项式.

3.1.2 常用的度量标准

最佳一致逼近

定义 3.1.1 (最佳一致逼近). 若以函数 $f(x)$ 和 $P(x)$ 的最大误差

$$\max_{x \in [a, b]} |f(x) - P(x)| = \|f(x) - P(x)\|_{\infty}$$

作为度量误差 $f(x) - P(x)$ ”大小”的标准, 在这种意义下的函数逼近称为最佳一致逼近或均匀逼近

最佳平方逼近

定义 3.1.2 (最佳平方逼近). 采用

$$\sqrt{\int_a^b [f(x) - P(x)]^2 dx} = \|f(x) - P(x)\|_2$$

作为度量误差“大小”标准的函数逼近称为最佳平方逼近或均方逼近.

3.2 最佳一致逼近

3.2.1 最佳一致逼近概念

定义 3.2.1. 设函数 $f(x)$ 是区间 $[a, b]$ 上的连续函数, 对于任意给定的 $\varepsilon > 0$, 如果存在多项式 $P(x)$, 使不等式

$$\max_{a \leq x \leq b} |f(x) - P(x)| < \varepsilon$$

成立, 则称多项式 $P(x)$ 在区间 $[a, b]$ 上一致逼近 (或均匀逼近) 于函数 $f(x)$.

3.2.2 最佳一致逼近多项式的存在性

定理 3.2.1 (Weierstrass 定理). 若 $f(x)$ 是区间 $[a, b]$ 上的连续函数, 则对于任意 $\varepsilon > 0$, 总存在多项式 $P(x)$, 使对一切 $a \leq x \leq b$, 有

$$\|f(x) - P(x)\|_\infty < \varepsilon$$

定理证明略.

可以证明的是, 多项式 $P(x)$ 形式为:

$$P(x) = \sum_{i=0}^{n(\varepsilon)} f\left(\frac{i}{n}\right) C_n^i x^i (1-x)^{n-i}$$

且

$$\lim_{n \rightarrow \infty} P_n(x) \rightarrow f(x)$$

3.2.3 $C[a, b]$ 上最佳一致逼近

在次数不超过 n 的多项式当中, 找到一个 $p_n^*(x)$, 使得

$$\|f(x) - p_n^*(x)\|_\infty = \min_{p_n(x) \in H_n} \|f(x) - p_n(x)\|_\infty$$

其中, H_n 表示由所有次数不超过 n 的代数多项式构成的线性空间. 上述问题为 $C[a, b]$ 空间中最佳一致逼近问题.

下面的定理说明了该最佳一致逼近多项式的存在性.

定理 3.2.2 (Borel 定理). 对任意的 $f(x) \in C[a, b]$, 在 H_n 中都存在对 $f(x)$ 的最佳一致逼近多项式, 记为 $p_n^*(x)$, 使得

$$\|f(x) - p_n^*(x)\|_\infty = \min_{p_n(x) \in H_n} \|f(x) - p_n(x)\|_\infty$$

称 $p_n^*(x)$ 为 $f(x)$ 的 n 次最佳一致逼近多项式. 或简称为最佳逼近多项式.

3.2.4 相关概念

偏差

定义 3.2.2 (偏差). 若 $P_n(x) \in H_n, f(x) \in C[a, b]$, 则称

$$\Delta(f, P_n) = \|f - P_n\|_\infty = \max_{a \leq x \leq b} |f(x) - P_n(x)|$$

为 $f(x)$ 与 $P_n(x)$ 在 $[a, b]$ 上的偏差

显然, $\Delta(f, P_n) \geq 0$, $\Delta(f, P_n)$ 的全体组成一个集合, 记作 $\Delta(f, H_n)$, 下界为 0.

最小偏差

定义 3.2.3. 若记集合的下确界为

$$E_n = \inf_{P_n \in H_n} \{\Delta(f, P_n)\} = \inf_{P_n \in H_n} \max_{a \leq x \leq b} |f(x) - P_n(x)|$$

则称 E_n 为 $f(x)$ 在 $[a, b]$ 上的最小偏差

偏差点

定义 3.2.4 (偏差点). 设 $f(x) \in C[a, b], P(x) \in H_n$, 若在 $x = x_0$ 上有

$$|P(x_0) - f(x_0)| = \max_{a \leq x \leq b} |P(x) - f(x)| = \mu$$

则称 x_0 是 $P(x) - f(x)$ 的偏差点.

若 $P(x_0) - f(x_0) = \mu$, 则称 x_0 为正偏差点;

若 $P(x_0) - f(x_0) = -\mu$, 则称 x_0 为负偏差点;

交错点组

定义 3.2.5. 若函数 $f(x)$ 在其定义域内的某一区间 $[a, b]$ 上存在 n 个点 $x_k, k = 1, 2, \dots, n$, 使得

$$\begin{aligned} |f(x_k)| &= \max |f(x)| = \|f(x)\|_{\infty}, k = 1, 2, \dots, n \\ -f(x_k) &= f(x_{k+1}), k = 1, 2, \dots, n-1 \end{aligned}$$

则称点集 $x_k, k = 1, 2, \dots, n$ 为函数 $f(x)$ 在区间 $[a, b]$ 上的一个交错点组, 点 x_k 为交错点.

3.2.5 $C[a, b]$ 上的最佳一致逼近特征

引理 3.2.3. 设 $f(x)$ 是区间 $[a, b]$ 上的连续函数, $P_n^*(x)$ 是 $f(x)$ 的 n 次最佳一致逼近多项式, 则 $f(x) - P_n^*(x)$ 在区间 $[a, b]$ 上必同时存在正负偏差点.

定理 3.2.4 (Chebyshev 定理). 设 $f(x)$ 是区间 $[a, b]$ 上的连续函数, 则 $P_n^*(x)$ 是 $f(x)$ 的 n 次最佳一致逼近多项式的充要条件是: $f(x) - P_n^*(x)$ 在区间 $[a, b]$ 上存在一个至少由 $n+2$ 个点组成的交错点组.

例 3.2.1. 求函数 $f(x) = \sin 4x$ 在 $[0, 2\pi]$ 上的 6 次最佳一致逼近 $P_6^*(x)$

解. 令

$$R(x) = f(x) - P_6^*(x)$$

考虑函数 $f(x) = \sin 4x$ 在区间 $[0, 2\pi]$ 存在由 8 个点组成的交错点组.

令 $R(x) = \sin 4x$, 则有

$$P_6^*(x) = f(x) - R(x) = 0$$

□

推论 3.2.5. 在 $P_n[a, b]$ 中, 若存在对函数 $f(x) \in C[a, b]$ 的最佳一致逼近元, 则唯一.

推论 3.2.6. 设 $f(x)$ 是区间 $[a, b]$ 上的连续函数, 则 $f(x)$ 的 n 次最佳一致逼近多项式是 $f(x)$ 的某个 n 次插值多项式.

推论 3.2.7. 设 $f(x)$ 是区间 $[a, b]$ 上的连续函数, $P_n^*(x)$ 是 $f(x)$ 的 n 次最佳一致逼近多项式, 若 $f^{(n+1)}(x)$ 在 (a, b) 内存在且保号, 则 $f(x) - P_n^*(x)$ 在区间 $[a, b]$ 上恰好存在一个由 $n+2$ 个点组成的交错点组, 且两个端点 a, b 都在交错点组中.

3.2.6 一次最佳逼近多项式 ($n = 1$)

推导过程

设 $f(x) \in C^2[a, b]$, 且 $f''(x)$ 在 (a, b) 内不变号, 要求 $f(x)$ 在 $[a, b]$ 上的一次最佳一致逼近多项式 $P_1(x) = a_0 + a_1x$. 由上述推论, $f(x) - P_1(x)$ 在 $[a, b]$ 上恰好由 3 个点构成的交错点组, 且区间端点 a, b 属于交错点组. 设另一交错点组为 x_2 . 则有

$$\begin{cases} f'(x_2) - P_1'(x_2) = 0 \\ f(a) - P_1(a) = f(b) - P_1(b) \\ f(a) - P_1(a) = -[f(x_2) - P_1(x_2)] \end{cases}$$

即

$$\begin{cases} f'(x_2) = a_1 \\ a_0 + a_1a - f(a) = f(x_2) - [a_0 + a_1x_2] \\ a_0 + a_1a - f(a) = a_0 + a_1b - f(b) \end{cases}$$

解得

$$\begin{aligned} a_1 &= \frac{f(b) - f(a)}{b - a} = f'(x_2) \\ a_0 &= \frac{f(a) + f(x_2)}{2} - \frac{f(b) - f(a)}{b - a} \frac{a + x_2}{2} \end{aligned}$$

即

$$P_1(x) = \frac{f(x_2) + f(a)}{2} + \frac{f(b) - f(a)}{b - a} \left(x - \frac{a + x_2}{2} \right)$$

例 3.2.2. 求 $Rf(x) = \sqrt{1+x^2}$ 在 $[0, 1]$ 上的一次最佳一致逼近多项式.

解. 因为

$$\begin{aligned} f'(x) &= \frac{x}{\sqrt{1+x^2}} \\ f''(x) &= \frac{1}{\sqrt[3]{(1+x^2)^2}} > 0 \end{aligned}$$

设 $f(x)$ 在 $[0, 1]$ 上的一次最佳一致逼近多项式为 $P_1(x) = a_0 + a_1x$, 则由

$$a_1 = \frac{f(b) - f(a)}{b - a} = f'(x_2)$$

可得

$$a_1 = \sqrt{2} - 1 \approx 0.414$$

再由

$$f'(x_2) = a_1$$

可得

$$\frac{x_2}{\sqrt{1+x_2^2}} = \sqrt{2} - 1$$

解得

$$x_2 = \sqrt{\frac{\sqrt{2}-1}{2}} \approx 0.4551, f(x_2) = \sqrt{1+x_2^2} \approx 1.0986$$

由

$$a_0 = \frac{f(a) + f(x_2)}{2} - \frac{f(b) - f(a)}{b - a} \frac{a + x_2}{2}$$

得

$$a_0 = \frac{1 + \sqrt{1+x_2^2}}{2} - a_1 \frac{x_2}{2} \approx 0.955$$

于是得 $\sqrt{1+x^2}$ 得最佳一次逼近多项式为

$$P_1(x) = 0.955 + 0.414x$$

最小偏差为

$$\|\sqrt{1+x^2} - P_1(x)\|_\infty = \max_{0 \leq x \leq 1} |\sqrt{1+x^2} - P_1(x)| = |\sqrt{1+0^2} - P_1(0)| \leq 0.045$$

□

n 次最佳一致逼近多项式 (推广)

设 $f(x) \in C^{n+1}[a, b]$, 且 $f^{(n+1)}(x)$ 在 (a, b) 内不变号吗要求 $f(x)$ 在 $[a, b]$ 上得 n 次最佳一致逼近多项式

$$P_n(x) = a_0 + a_1x + \cdots + a_nx^n$$

由推论可知, $f(x) - P_n(x)$ 在 $[a, b]$ 上恰好有 $n + 2$ 个点构成得交错点组, 且 a, b 属于交错点组. 设另外 n 个交错点为 x_1, x_2, \dots, x_n , 则有

$$\begin{cases} f'(x_i) - P'_n(x_i) = 0, i = 1, 2, \dots, n \\ f(a) - P_n(a) = (-1)^i (f(x_i) - P_n(x_i)), i = 1, 2, \dots, n \\ f(a) - P_n(a) = (-1)^{n+1} [f(b) - P_n(b)] \end{cases}$$

3.2.7 Chebyshev 多项式及其应用

定义

定义 3.2.6 (Chebyshev 多项式). 称

$$T_n = \cos(n \arccos x), |x| \leq 1$$

为 Chebyshev 多项式

注: 令 $\theta = \arccos x$, 则 $\cos \theta = x$. 而

$$\cos n\theta = \cos^n \theta - C_n^2 \cos^{n-2} \theta \sin^2 \theta + C_n^4 \cos^{n-4} \theta \sin^4 \theta - \dots$$

故 T_n 是关于 x 得 n 次代数多项式.

性质

正交性: 由 $T_n(x)$ 所组成的序列 $T_n(x)$ 在区间 $[-1, 1]$ 上带权 $\rho(x) = \frac{1}{\sqrt{1-x^2}}$ 的正交多项式序列, 且

$$\int_{-1}^1 \frac{1}{\sqrt{1-x^2}} T_m(x) T_n(x) dx = \begin{cases} 0, & m \neq n \\ \frac{\pi}{2}, & m = n \neq 0 \\ \pi, & m = n = 0 \end{cases}$$

递推关系: 相邻的三个 Chebyshev 多项式具有如下递推关系式:

$$\begin{cases} T_0(x) = 1, T_1(x) = x \\ T_{n+1}(x) = 2x \cdot T_n(x) - T_{n-1}(x), n = 1, 2, \dots \end{cases}$$

奇偶性: Chebyshev 多项式 T_n , 当 n 为奇数时为奇函数, 当 n 为偶数时为偶函数. 即

$$\begin{aligned} T_n(-x) &= \cos[n \arccos(-x)] = \cos(n\pi - n \arccos x) \\ &= (-1)^n \cos(n \arccos x) = (-1)^n T_n(x) \end{aligned}$$

T_n 在区间 $[-1, 1]$ 上有 n 个不同的零点

$$x_k = \cos \frac{(2k-1)\pi}{2n}, k = 1, 2, \dots, n$$

$T_n(x)$ 在 $[-1, 1]$ 上有 $n+1$ 个不同的极值点

$$x'_k = \cos k \frac{\pi}{n}, k = 0, 1, \dots, n$$

使 $T_n(x)$ 轮流取得最大值 1 和最小值 -1.

$T_n(x)$ 的最高次项系数为 $2^{n-1}, n = 1, 2, \dots$

在区间 $[-1, 1]$ 上, 在所有首项系数为 1 的 n 次多项式 $p_n(x)$ 中. $\tilde{T}_n(x) = \frac{1}{2^{n-1}} T_n(x)$ 与零的偏差最小, 且偏差为 $\frac{1}{2^{n-1}}$, 即对于任何 $p(x) \in H_n(x)$, 有

$$\frac{1}{2^{n-1}} = \|\tilde{T}_n(x) - 0\|_{\infty} = \min_{p(x) \in H_n} \|p(x) - 0\|_{\infty}$$

上述性质又被称为 Chebyshev 多项式的最小模性质.

应用

设 $f(x) = b_0 + b_1 x + \dots + b_{n+1} x^{n+1} (b_{n+1} \neq 0)$ 为 $[-1, 1]$ 上的 $n+1$ 次多项式, 要求 $f(x)$ 在 $[-1, 1]$ 上的 n 次最佳一致逼近 $P_n^*(x)$.

由于首项系数为 1 的 $n+1$ 次 Chebyshev 多项式 $\tilde{T}_{n+1}(x)$ 无穷范数最小, 有

$$\frac{f(x) - P_n^*(x)}{b_{n+1}} = \tilde{T}_{n+1}(x) = \frac{1}{2^n} T_{n+1}(x)$$

于是

$$P_n^*(x) = f(x) - b_{n+1} \tilde{T}_{n+1}(x)$$

其偏差为

$$\|f(x) - P_n^*(x)\|_{\infty} = \left\| \frac{b_{n+1}}{2^n} T_{n+1}(x) \right\|_{\infty} = \frac{|b_{n+1}|}{2^n}$$

例 3.2.3. 设 $f(x) = 4x^4 + 2x^3 - 5x^2 + 8x - \frac{5}{2}, |x| \leq 1$, 求 $f(x)$ 在 $[-1, 1]$ 上的 3 次最佳一致逼近元 $P_3^*(x)$

解. 由 $f(x)$ 的表达式可知 $b_4 = 4$, 首项系数为 1 的 4 次 Chebyshev 多项式为

$$\tilde{T}_4(x) = x^4 - x^2 + \frac{1}{8}$$

设 $f(x)$ 在 $[-1, 1]$ 上的 3 次最佳一致逼近多项式为 $P_3^*(x)$, 则

$$\frac{f(x) - P_3^*(x)}{4} = \tilde{T}_4(x)$$

从而

$$P_3^*(x) = f(x) - 4\tilde{T}_4(x) = 2x^3 - x^2 + 8x - 3$$

其最小偏差为

$$\|f(x) - P_3^*(x)\|_{\infty} = \left\| \frac{b_4}{2^3} T_4(x) \right\|_{\infty} = \frac{|4|}{2^3} = \frac{1}{2}$$

□

例 3.2.4. 若 $f(x) = 4x^4 - 5x^2 + 8x - \frac{5}{2}$, $|x| \leq 1$, 重求上题

解. 重复上述步骤, 可得 3 次最佳一致逼近多项式为

$$P_3^*(x) = -x^2 + 8x - 3$$

(注意: 这里的 3 次多项式是次数不超过 3)

□

考虑一般区间 $[a, b]$ 上的多项式降阶. 设 $f(x) = b_0 + b_1x + \cdots + b_{n+1}x^{n+1}$, $b_{n+1} \neq 0$ 为 $[a, b]$ 上的 $n+1$ 次多项式, 要求 $f(x)$ 在 $[a, b]$ 上的 n 次最佳一致逼近 $P_n^*(x)$.

首先做区间变换, 令

$$x = \frac{b-a}{2}t + \frac{b+a}{2}$$

其中 $t \in [-1, 1]$, 则

$$f(x) = f\left(\frac{b-a}{2}t + \frac{b+a}{2}\right) = g(t) \in C[-1, 1]$$

设 $g(t)$ 在 $[-1, 1]$ 的 n 次最佳一致逼近多项式为 $P_n(t)$, 则

$$P_n(t) = g(t) - c_{n+1}\tilde{T}_{n+1}(t)$$

其中, c_{n+1} 为 $g(t)$ 的最高此项系数.

$$c_{n+1} = b_{n+1} \left(\frac{b-a}{2} \right)^{n+1}$$

于是, $f(x)$ 在 $[a, b]$ 上的 n 次最佳一致逼近多项式为

$$P_n^*(x) = P_n \left(\frac{2x - b - a}{b - a} \right)$$

最小偏差为

$$\|f(x) - P_n(x)\|_\infty = \|g(t) - P_n(t)\|_\infty = \frac{|c_{n+1}|}{2^n}$$

下面再考虑插值多项式的情况. 设 $f(x) \in C[-1, 1]$, 且存在 $n+1$ 阶连续导数 $f^{(n+1)}(x)$. 如何在 $[-1, 1]$ 上确定互异节点 x_0, x_1, \dots, x_n , 使得 $f(x)$ 的 n 次插值多项式的余项最小.

由插值余项定理, 可知 n 次插值多项式 $L_n(x)$ 的余项为

$$R_n(x) = f(x) - L_n(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!} \omega_{n+1}(x)$$

其中,

$$\omega_{n+1}(x) = \prod_{i=0}^n (x - x_i), \xi \in (-1, 1)$$

其误差上界为

$$\begin{aligned} |f(x) - L_n(x)| &\leq \frac{1}{(n+1)!} \max_{-1 \leq x \leq 1} |f^{(n+1)}(x)| \max_{-1 \leq x \leq 1} |\omega_{n+1}(x)| \\ &= \frac{1}{(n+1)!} \|f^{(n+1)}\|_\infty \|\omega_{n+1}(x)\|_\infty \end{aligned}$$

要使余项达到最小, 即令 $\|\omega_{n+1}(x)\|_\infty$ 尽可能小. 由于是一个首项系数为 1 的 $n+1$ 次多项式. 故有 Chebyshev 多项式, 取

$$\omega_{n+1}(x) = \tilde{T}_{n+1}(x)$$

同时有

$$\omega_{n+1}(x) = \prod_{i=0}^n (x - x_i)$$

故只需令 $x_i, i = 0, 1, \dots, n$ 为 $n+1$ 次 Chebyshev 多项式的零点, 即

$$x_i = \cos \frac{(2i+1)\pi}{2(n+1)}, i = 0, 1, \dots, n$$

3.3 最佳平方逼近

3.3.1 内积空间

内积空间定义

定义 3.3.1 (内积空间). 设 X 为 (实) 线性空间, 在 X 上定义了内积是指对 X 中每一对元素 x, y , 都有一实数, 记为 (x, y) 与之对应, 且这个对应满足:

1.

$$(x, x) \geq 0, (x, x) = 0 \Leftrightarrow x = 0$$

;

2.

$$(x, y) = (y, x), x, y \in X$$

;

3.

$$(\lambda x, y) = \lambda(x, y), x, y \in X; \lambda \in R$$

;

4.

$$(x + y, z) = (x, z) + (y, z), x, y, z \in X$$

则称 X 为内积空间, 称二元函数 (x, y) 为内积

2-范数定义及其性质

定义 3.3.2. 设 X 为内积空间, 则对任意的 $x \in X$, 称

$$\|x\|_2 = \sqrt{(x, x)}$$

为 x 的 2-范数或欧式范数.

特别的, 对 $f(x) \in C[a, b]$, 称

$$\|f\|_2 = \sqrt{\int_a^b \rho(x) f^2(x) dx} = \sqrt{(f, f)}$$

为函数 $f(x)$ 的 Euclid 范数

设 X 是一内积空间, 则对任意的 $x, y \in X$, 有

1. Cauchy-Schwarz 不等式:

$$(x, y)^2 \leq (x, x)(y, y)$$

2. 三角不等式:

$$\|x + y\|_2 \leq \|x\|_2 + \|y\|_2$$

讨论特殊的两种内积空间,

- n 维欧氏空间 R^n , 内积就是两向量数量积, 即

$$(x, y) = x^T y = \sum x_i y_i$$

- 连续函数空间 $C[a, b]$, 内积定义为积分运算或带权函数积分运算, 即

$$(f(x), g(x)) = \int_a^b f(x)g(x) dx, f(x), g(x) \in C[a, b]$$

或者

$$(f(x), g(x)) = \int_a^b \rho(x)f(x)g(x) dx, f(x), g(x) \in C[a, b]$$

定义 3.3.3 (权函数). 设 $\rho(x)$ 定义在有限或无限区间 $[a, b]$ 上, 如果具有下列性质:

1. 对任意 $x \in [a, b], \rho(x) \geq 0$;
2. 积分 $\int_a^b |x|^n \rho(x) dx$ 存在, $n = 0, 1, 2, \dots$;
3. 对非负连续函数 $g(x)$, 若 $\int_a^b g(x)\rho(x) dx = 0$, 则在 (a, b) 上有 $g(x) \equiv 0$

称满足上述条件的 $\rho(x)$ 为 $[a, b]$ 上的权函数

3.3.2 相关概念

距离

R^n 中, 两个向量 x, y 之间的距离定义为

$$\text{dist}(x, y) = \|x - y\|_2 = \sqrt{(x - y, x - y)} = \sqrt{\sum_i (x_i - y_i)^2}$$

连续函数空间 $C[a, b]$ 中, 两个函数 $f(x)$ 与 $g(x)$ 的欧式距离为

$$\text{dist}(f(x), g(x)) = \|f(x) - g(x)\|_2 = \sqrt{\int_a^b \rho(x) [f(x) - g(x)]^2 dx}$$

正交

在 R^n 中, 若两个向量 x, y 的内积为零, 即

$$(x, y) = 0$$

则称向量 x, y 正交.

在连续函数空间 $C[a, b]$ 中, 若函数 $f(x)$ 与 $g(x)$ 满足

$$(f, g) = \int_a^b \rho(x) f(x) g(x) dx = 0$$

则称 $f(x)$ 与 $g(x)$ 在 $[a, b]$ 上带权 $\rho(x)$ 正交.

进一步, 设在 $C[a, b]$ 上给定函数系 $\varphi_k(x)$, 若满足条件

$$(\varphi_j(x), \varphi_k(x)) = \begin{cases} 0, & j \neq k \\ \text{常数 } A_k > 0, & j = k, j, k = 0, 1, \dots \end{cases}$$

则称函数系 $\varphi_k(x)$ 是 $[a, b]$ 上带权 $\rho(x)$ 的正交坐标系.

特别地, 若 $A_k = 1$, 则该函数系称为标准正交坐标系.

若上述定义中函数系为多项式系, 则称之为 $C[a, b]$ 上带权 $\rho(x)$ 的正交多项式系, 并称 $\varphi_n(x)$ 为 $[a, b]$ 上带权 $\rho(x)$ 的 n 次正交多项式.

3.3.3 内积空间上的最佳平方逼近

函数系的线性关系

定义 3.3.4. 设 $\varphi_0(x), \varphi_1(x), \dots, \varphi_n(x)$ 为区间 $[a, b]$ 上的连续函数, 若关系式

$$a_0 \varphi_0(x) + a_1 \varphi_1(x) + \dots + a_n \varphi_n(x) = 0$$

当且仅当 $a_0 = a_1 = \dots = a_n = 0$ 时成立, 则称函数在 $[a, b]$ 上是线性无关的, 否则称线性相关.

函数系线性无关的判别条件

定理 3.3.1. 连续函数 $\varphi_0(x), \varphi_1(x), \dots, \varphi_n(x)$ 在 $[a, b]$ 上线性无关的充分必要条件是它们的 Gram 行列式

$$G_n = \begin{vmatrix} (\varphi_0, \varphi_0) & (\varphi_0, \varphi_1) & \cdots & (\varphi_0, \varphi_n) \\ (\varphi_1, \varphi_0) & (\varphi_1, \varphi_1) & \cdots & (\varphi_1, \varphi_n) \\ \vdots & \vdots & \cdots & \vdots \\ (\varphi_n, \varphi_0) & (\varphi_n, \varphi_1) & \cdots & (\varphi_n, \varphi_n) \end{vmatrix} \neq 0$$

设 $\varphi_0(x), \varphi_1(x), \dots, \varphi_n(x)$ 是 $[a, b]$ 上线性无关的连续函数, a_0, a_1, \dots, a_n 是任意实数, 则

$$S(x) = a_0\varphi_0(x) + a_1\varphi_1(x) + \cdots + a_n\varphi_n(x)$$

的全体是 $C[a, b]$ 的一个子集, 记作

$$\Phi = \text{span}\{\varphi_0, \varphi_1, \dots, \varphi_n\}$$

并称 $\varphi_0(x), \varphi_1(x), \dots, \varphi_n(x)$ 是生成集合 Φ 的一个基底.

最佳平方逼近元的定义

设 X 为线性内积空间, $\varphi_0, \varphi_1, \dots, \varphi_n$ 为 X 上 $n+1$ 个线性无关元, 即

$$\Phi = \text{span}\{\varphi_0, \varphi_1, \dots, \varphi_n\}$$

定义 3.3.5 (最佳平方逼近元). 对任意的 $X \in X$, 在 X 的子空间 Φ 中, 求 g 的在 2-范数意义下的最佳逼近元 S^* , 即求 $S^* \in \Phi$, 使得

$$\|S^* - g\|_2 = \min_{S \in \Phi} \|S - g\|_2$$

对任意 $S \in \Phi$ 成立. 若满足上式的 $S^* \in \Phi$ 存在, 则称 S^* 为 $g \in X$ 的最佳平方逼近元

最佳平方逼近元的存在性

定理 3.3.2. 设 X 为线性内积空间, 由线性无关组 $\varphi_0, \varphi_1, \dots, \varphi_n$ 张成的线性空间 Φ 为 X 的子空间, 则对任意的 $g \in X$, 存在 $S^* \in \Phi$ 为 g 的最佳平方逼近元.

最佳平方逼近元的特征

定理 3.3.3. $S^* \in \Phi = \text{span}\{\varphi_0, \varphi_1, \dots, \varphi_n\}$ 为 $g \in X$ (线性内积空间) 的最佳平方逼近元的充要条件是: $g - S^*$ 与一切 $\varphi_j, j = 0, 1, \dots, n$ 正交. 其中, $\varphi_0, \varphi_1, \dots, \varphi_n$ 为 X 的 $n+1$ 个线性无关元.

该定理所说的 $g - S^*$ 与一切 φ_j 正交, 指 $g - S^*$ 与一切 φ_j 的内积为零, 即

$$(g - S^*, \varphi_j) = 0, j = 0, 1, \dots, n$$

最佳平方逼近元的唯一性

定理 3.3.4. 线性内积空间 X 的子空间 Φ 中若存在对 $g \in X$ 的最佳平方逼近元, 则唯一.

最佳平方逼近元的求解

由最佳平方逼近元充要条件, 若假定

$$S^* = \sum_{j=0}^n c_j^* \varphi_j(x)$$

则可得出

$$\left(g - \sum_{j=0}^n c_j^* \varphi_j, \varphi_i \right) = 0, i = 0, 1, \dots, n$$

移项变形为

$$\left(\sum_{j=0}^n c_j^* \varphi_j, \varphi_i \right) = (\varphi_i, g), i = 0, 1, \dots, n$$

其中, $c_j^*, j = 0, 1, \dots, n$ 为待定系数. 用矩阵表示方程组为:

$$\begin{pmatrix} (\varphi_0, \varphi_0) & (\varphi_0, \varphi_1) & \cdots & (\varphi_0, \varphi_n) \\ (\varphi_1, \varphi_0) & (\varphi_1, \varphi_1) & \cdots & (\varphi_1, \varphi_n) \\ \vdots & \vdots & \cdots & \vdots \\ (\varphi_n, \varphi_0) & (\varphi_n, \varphi_1) & \cdots & (\varphi_n, \varphi_n) \end{pmatrix} \begin{pmatrix} c_0^* \\ c_1^* \\ \vdots \\ c_n^* \end{pmatrix} = \begin{pmatrix} (\varphi_0, g) \\ (\varphi_1, g) \\ \vdots \\ (\varphi_n, g) \end{pmatrix}$$

此方程称为法方程组.

若选取的一组基底 $\varphi_0(x), \varphi_1(x), \dots, \varphi_n(x)$ 满足

$$(\varphi_i, \varphi_j) = \begin{cases} 0, & i \neq j \\ A_i > 0, & i = j \end{cases}$$

则称其为正交基, 此时

$$c_j^* = \frac{(\varphi_j, g)}{(\varphi_j, \varphi_j)}, j = 0, 1, \dots, n$$

最佳平方逼近元的误差估计

设 $S^* = \sum_{j=0}^n c_j^* \varphi_j \in \Phi = \text{span}\{\varphi_0, \varphi_1, \dots, \varphi_n\}$ 为 $g \in X$ 的最佳平方逼近元, 则其平方误差为

$$\begin{aligned} \|g - S^*\|_2^2 &= (g - S^*, g - S^*) = (g - S^*, g) - (g - S^*, S^*) \\ &= (g - S^*, g) - (g - S^*, \sum_{j=0}^n c_j^* \varphi_j) \\ &= (g, g) - (S^*, g) - \sum_{j=0}^n c_j^* (g - S^*, \varphi_j) \\ &= (g, g) - (S^*, g) = (g, g) - \sum_{j=1}^n c_j^* (g - \varphi_j) \end{aligned}$$

均方误差为

$$\|g - S^*\|_2 = \sqrt{(g, g) - \sum_{j=0}^n c_j^* (g, \varphi_j)}$$

3.3.4 连续函数的最佳平方逼近

对于给定的函数 $f(x) \in C[a, b]$, 要求函数

$$S^* \in \Phi = \text{span}\{\varphi_0, \varphi_1, \dots, \varphi_n\}$$

使

$$\int_a^b \rho(x) [f(x) - S^*(x)]^2 dx = \min_{S(x) \in \Phi} \int_a^b \rho(x) [f(x) - S(x)]^2 dx$$

若这样的 $S^*(x)$ 存在, 则称为 $f(x)$ 在区间 $[a, b]$ 上的最佳平方逼近函数.

特别地, 若取 $\Phi = H_n = \text{span}\{1, x, \dots, x^n\}$, 则称

$$S^*(x) = a_0 \cdot 1 + a_1 x + \dots + a_n x^n$$

为 $f(x)$ 在 $[a, b]$ 上的 n 次最佳平方逼近多项式.

计算 $C[a, b]$ 上函数 $f(x)$ 的 n 次最佳平方逼近多项式. 取

$$\varphi_0 = 1, \varphi_1 = x, \dots, \varphi_n = x^n$$

设 $f(x)$ 的 n 次最佳平方逼近多项式为

$$S^*(x) = \sum_{j=0}^n a_j \cdot \varphi_j(x) = \sum_{j=0}^n a_j \cdot x^j$$

构造法方程组:

$$\begin{pmatrix} (\varphi_0, \varphi_0) & (\varphi_0, \varphi_1) & \cdots & (\varphi_0, \varphi_n) \\ (\varphi_1, \varphi_0) & (\varphi_1, \varphi_1) & \cdots & (\varphi_1, \varphi_n) \\ \vdots & \vdots & & \vdots \\ (\varphi_n, \varphi_0) & (\varphi_n, \varphi_1) & \cdots & (\varphi_n, \varphi_n) \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{pmatrix} = \begin{pmatrix} (f, \varphi_0) \\ (f, \varphi_1) \\ \vdots \\ (f, \varphi_n) \end{pmatrix}$$

其中,

$$\begin{aligned} (\varphi_i, \varphi_j) &= \int_a^b \varphi_i \cdot \varphi_j \, dx = \int_a^b x^i \cdot x^j \, dx \\ &= \frac{b^{i+j+1} - a^{i+j+1}}{i+j+1}, i, j = 0, 1, \dots, n \end{aligned}$$

$$(f, \varphi_j) = \int_a^b f(x) \cdot \varphi_j \, dx = \int_a^b f(x) \cdot x^j \, dx, j = 0, 1, \dots, n$$

由此可得函数 $f(x)$ 在 H_n 中的 n 次最佳平方逼近多项式:

$$S^*(x) = \sum_{j=0}^n a_j^* x^j$$

平方误差为:

$$\|f(x) - S^*(x)\|_2^2 = (f, f) - \sum_{j=0}^n a_j^* (f, \varphi_j) = \int_a^b [f(x)]^2 \, dx - \sum_{j=0}^n a_j^* (f, \varphi_j)$$

由于 $\varphi_0, \varphi_1, \dots, \varphi_n$ 线性无关, 故 $G_n \neq 0$, 上述方程组存在唯一解 $a_k = a_k^*, k = 0, 1, \dots, n$

特别地, 当 $a = 0, b = 1$ 时, 法方程组系数矩阵为

$$H = \begin{pmatrix} 1 & \frac{1}{2} & \frac{1}{3} & \cdots & \frac{1}{n+1} \\ \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \cdots & \frac{1}{n+2} \\ \frac{1}{3} & \frac{1}{4} & \frac{1}{5} & \cdots & \frac{1}{n+3} \\ \vdots & \vdots & \vdots & & \vdots \\ \frac{1}{n+1} & \frac{1}{n+2} & \frac{1}{n+3} & \cdots & \frac{1}{2n+1} \end{pmatrix}$$

该矩阵为 Hilbert 矩阵, 为病态矩阵.

例 3.3.1. 求 $g(x) = \sqrt{x}$ 在 $H_1[0, 1]$ 中的最佳平方逼近元, 并估计误差

解. 取 $\varphi_0 = 1, \varphi_1 = x, H_1[0, 1] = \text{span}\{1, x\}$, 记

$$S_1^*(x) = a_0 + a_1 x$$

由

$$\begin{aligned} (\varphi_0, \varphi_0) &= \int_0^1 1 \cdot 1 \, dx = 1 \\ (\varphi_0, \varphi_1) &= \int_0^1 1 \cdot x \, dx = \frac{1}{2} \\ (\varphi_1, \varphi_1) &= \int_0^1 x \cdot x \, dx = \frac{1}{3} \end{aligned}$$

同样可求得

$$\begin{aligned} (\varphi_0, g) &= \int_0^1 \sqrt{x} \, dx = \frac{2}{3} \\ (\varphi_1, g) &= \int_0^1 x \cdot \sqrt{x} \, dx = \frac{2}{5} \end{aligned}$$

所以, 关于 a_0, a_1 的法方程组为

$$\begin{cases} a_0 + \frac{1}{2}a_1 = \frac{2}{3} \\ \frac{1}{2}a_0 + \frac{1}{3}a_1 = \frac{2}{5} \end{cases}$$

解得

$$a_0 = \frac{4}{15}, a_1 = \frac{4}{5}$$

即 $S_1^*(x) = \frac{4}{5}x + \frac{4}{15}$ 为 $H_1[0, 1]$ 对 $g(x)$ 的最佳平方逼近元. 且误差为

$$\begin{aligned} \|\delta_n\|_2^2 &= \|\sqrt{x} - S_1^*(x)\|_2^2 = \int_0^1 (\sqrt{x})^2 \, dx - \sum_{k=0}^1 a_k (\sqrt{x}, \varphi_k) \\ &= \frac{1}{2} - \frac{4}{15} \times \frac{2}{3} - \frac{4}{5} \times \frac{2}{5} = \frac{1}{255} \approx 0.0044 \end{aligned}$$

□

3.4 正交多项式

3.4.1 正交化手续

当权函数 $\rho(x)$ 及区间 $[a, b]$ 给定后, 可由幂函数系 $1, x, x^2, \dots, x^n, \dots$ 利用 Schmidt 正交化方法构造出和正交多项式系

$$g_0(x) = 1, \\ g_k(x) = x^k - \sum_{i=0}^{k-1} \frac{(x^k, g_i)}{(g_i, g_i)} \cdot g_i, k = 1, 2, \dots$$

3.4.2 正交多项式的性质

1. $g_n(x)$ 是最高次项系数为 1 的 n 次多项式.
2. 任一 n 次多项式 $P_n(x) \in H_n$ 均可表示为 $g_0(x), g_1(x), \dots, g_n(x)$ 的线性组合.
3. 当 $n \neq m$ 时, $(g_n, g_m) = 0$, 且 $g_n(x)$ 与任一次数小于 n 的多项式正交.
4. 递推性,

$$g_{n+1}(x) = (x - \alpha_n)g_n(x) - \beta_n g_{n-1}(x), n = 0, 1, \dots$$

其中,

$$g_0(x) = 1, g_{-1}(x) = 0, \\ \alpha_n = \frac{(xg_n, g_n)}{(g_n, g_n)}, n = 0, 1, \dots, \\ \beta_n = \frac{(g_n, g_n)}{(g_{n-1}, g_{n-1})}, n = 1, 2, \dots$$

这里 $(xg_n, g_n) = \int_a^b xg_n^2(x)\rho(x) dx$

5. 设 $g_0(x), g_1(x), \dots$ 是在 $[a, b]$ 上带权 $\rho(x)$ 的正交多项式序列, 则 $g_n(x) (n \geq 1)$ 的 n 个根都是单重实根, 且都在区间 (a, b) 内.

3.4.3 常用的正交多项式

第一类 Chebyshev 多项式

定义:

$$T_n = \cos(n \arccos x), |x| \leq 1$$

性质:

1. 递推关系

$$\begin{cases} T_0(x) = 1, T_1(x) = x \\ T_{n+1}(x) = 2x \cdot T_n(x) - T_{n-1}(x), n = 1, 2, \dots \end{cases}$$

2. 正交性

由 $T_n(x)$ 所组成的序列 $T_n(x)$ 在区间 $[-1, 1]$ 上带权 $\rho(x) = \frac{1}{\sqrt{1-x^2}}$ 的正交多项式序列, 且

$$(T_m(x), T_n(x)) = \int_{-1}^1 \frac{1}{\sqrt{1-x^2}} T_m(x) T_n(x) dx = \begin{cases} 0, & m \neq n \\ \frac{\pi}{2}, & m = n \neq 0 \\ \pi, & m = n = 0 \end{cases}$$

Legendre 多项式

定义 3.4.1 (n 次 Legendre 多项式). 多项式

$$P_n(x) = \frac{1}{2^n \cdot n!} \frac{d^n}{dx^n} [(x^2 - 1)^n], n = 0, 1, 2, \dots$$

称为 n 次 Legendre 多项式

正交性: Legendre 多项式序列 $P_n(x)$ 是 $[-1, 1]$ 上带权 $\rho = 1$ 的正交多项式序列, 且

$$(P_m(x), P_n(x)) = \int_{-1}^1 P_m(x) P_n(x) dx = \begin{cases} 0, & m \neq n \\ \frac{2}{2n+1}, & m = n \end{cases}$$

相邻的三个 Legendre 多项式具有如下递推关系:

$$\left\{ P_0(x) = 1, P_1(x) = x, P_{n+1}(x) = \frac{2x+1}{n+1} x P_n(x) - \frac{n}{n+1} P_{n-1}(x), n = 1, 2, \dots \right.$$

奇偶性: 当 n 为偶数时, $P_n(x)$ 为偶函数; 当 n 为奇数时, $P_n(x)$ 为奇函数.

$P_n(x)$ 在区间 $[-1, 1]$ 内存在 n 个互异实零点. 且最高项系数为 $\frac{(2n)!}{2^n(n!)^2}$

在所有首项系数为 1 的 n 次多项式中, Legendre 多项式 $\tilde{P}_n(x)$ 在 $[-1, 1]$ 上与零的平方误差最小. 即对于任何 $p(x) \in H_n(x)$, 有

$$\|\tilde{P}_n(x) - 0\|_2 = \min_{p(x) \in H_n} \|p(x) - 0\|_2$$

扩展:

$$\|\tilde{T}_n(x) - 0\|_\infty = \min_{p(x) \in H_n} \|p(x) - 0\|_\infty$$

其他常用的正交多项式

定义 3.4.2 (第二类 Chebyshev 多项式). 称

$$U_n(x) = \frac{\sin[(n+1)\arccos x]}{\sqrt{1-x^2}}$$

为第二类 Chebyshev 多项式

其中, $U_n(x)$ 是区间 $[-1, 1]$ 带权 $\rho(x) = \sqrt{1-x^2}$ 的正交多项式序列,

$$\int_{-1}^1 \sqrt{1-x^2} U_m(x) U_n(x) dx = \begin{cases} 0, & m \neq n \\ \frac{\pi}{2}, & m = n \end{cases}$$

且具有如下递推关系式:

$$\begin{cases} U_0(x) = 1, U_1(x) = 2x \\ U_{n+1}(x) = 2xU_n(x) - U_{n-1}(x), n = 1, 2, \dots \end{cases}$$

定义 3.4.3 (Laguerre 多项式). 称多项式

$$L_n(x) = e^x \frac{d^n}{dx^n} (x^n e^{-x}), 0 \leq x < +\infty, n = 0, 1, 2, \dots$$

为 Laguerre 多项式

$L_n(x)$ 是定义在区间 $[0, +\infty)$ 上带权 $\rho(x) = e^{-x}$ 的正交多项式序列.

$$\int_0^\infty e^{-x} \cdot L_m(x) L_n(x) dx = \begin{cases} 0, & m \neq n \\ (n!)^2, & m = n \end{cases}$$

递推关系式:

$$\begin{cases} L_0(x) = 1, L_1(x) = 1 - x \\ L_{n+1}(x) = (1 + 2n - x)L_n(x) - n^2 \cdot L_{n-1}(x), n = 1, 2, \dots \end{cases}$$

定义 3.4.4 (Hermite 多项式). 称多项式

$$H_n(x) = (-1)^n \cdot e^{x^2} \cdot \frac{d^n}{dx^n} (e^{-x^2}), x \in (-\infty, \infty), n = 0, 1, 2, \dots$$

为 Hermite 多项式

$H_n(x)$ 是区间 $(-\infty, \infty)$ 上带权 $\rho(x) = e^{-x^2}$ 的正交多项式序列.

$$\int_{-\infty}^{\infty} e^{-x^2} H_m(x) \cdot H_n(x) dx = \begin{cases} 0, & m \neq n \\ 2^n \cdot n! \cdot \sqrt{\pi}, & m = n \end{cases}$$

递推关系式

$$\begin{cases} H_0(x) = 1, H_1(x) = 2x \\ H_{n+1}(x) = 2xH_n(x) - 2nH_{n-1}(x), n = 1, 2, \dots \end{cases}$$

3.5 函数按正交多项式展开

设 $\Phi = \text{span}\{\varphi_0, \varphi_1, \dots, \varphi_n\}$, 其中 $\varphi_i (i = 0, 1, \dots, n)$ 是 $[a, b]$ 上带权 $\rho(x)$ 的正交多项式系, 给定 $f(x) \in C[a, b]$, 若 $S_n^*(x) = a_0\varphi_0(x) + \dots + a_n\varphi_n(x)$ 为 $f(x)$ 在 $[a, b]$ 上的 n 次最佳平方逼近多项式, 则由正交多项式的性质

$$a_k = \frac{(f, \varphi_k)}{(\varphi_k, \varphi_k)}, k = 0, 1, \dots, n$$

于是

$$S_n^*(x) = \sum_{k=0}^n \frac{(f, \varphi_k)}{(\varphi_k, \varphi_k)} \varphi_k(x)$$

其均方误差为

$$\|\delta_n\|_2 = \|f(x) - S_n^*(x)\|_2 = \sqrt{(f, f) - \sum_{k=0}^n a_k^2 (\varphi_k, \varphi_k)}$$

例 3.5.1. 求 $f(x) = e^x$ 在 $[-1, 1]$ 上的三次最佳平方逼近多项式.

解. 考虑使用 Legendre 多项式, 计算 $(f, P_k), k = 0, 1, 2, 3$, 即

$$(f, P_0) = \int_{-1}^1 e^x dx = e - \frac{1}{e} \approx 2.3504$$

$$(f, P_1) = \int_{-1}^1 x e^x dx = 2e^{-1} \approx 0.7358$$

$$(f, P_2) = \int_{-1}^1 \left(\frac{3}{2}x^2 - \frac{1}{2}\right) e^x dx = e - \frac{7}{e} \approx 0.1431$$

$$(f, P_3) = \int_{-1}^1 \left(\frac{5}{2}x^3 - \frac{3}{2}x\right) e^x dx = \frac{37}{e} - 5e \approx 0.2013$$

所以得

$$\begin{aligned} a_0^* &= \frac{(f, P_0)}{2} = 1.1752 \\ a_1^* &= 3 \frac{(f, P_1)}{2} = 1.1036 \\ a_2^* &= 5 \frac{(f, P_2)}{2} = 0.3578 \\ a_3^* &= 7 \frac{(f, P_3)}{2} = 0.07046 \end{aligned}$$

所以

$$\begin{aligned} S_3^*(x) &= 1.1752 + 1.1036x + 0.3578 \left(\frac{3}{2}x^2 - \frac{1}{2}\right) + 0.07046 \left(\frac{5}{2}x^3 - \frac{3}{2}x\right) \\ &= 0.9963 + 0.9979x + 0.5367x^2 + 0.1761x^3 \end{aligned}$$

均方误差为

$$\|\delta_n\|_2 = \|e^x - S_3^*(x)\|_2 = \sqrt{\int_{-1}^1 e^{2x} dx - \sum_{k=0}^3 \frac{2}{2k+1} a_k^{*2}} \leq 0.0084$$

最大误差为

$$\|\delta_n\|_\infty = \|e^x - S_3^*(x)\|_\infty \leq 0.0112$$

□

下面使用 Python 代码演示这道题. 注意, 代码当中的系数为 Legendre 多项式的系数.

```
1  # 使用Legendre多项式进行平方逼近演示(例题解答) Exercise3-1.py
2  import numpy as np
3  from scipy.special import legendre
4  from scipy.optimize import least_squares
5
6  # 计算Legendre多项式的系数
7  def legendre_coefficients(x, y, degree):
8      A = np.zeros((len(x), degree + 1))
9      for i in range(degree + 1):
10         A[:, i] = legendre(i)(x)
11     return np.linalg.lstsq(A, y, rcond=None)[0]
12 # 定义逼近函数
13 def approximation_func(x, coefficients):
14     result = np.zeros_like(x)
15     for i, coeff in enumerate(coefficients):
16         result += coeff * legendre(i)(x)
17     return result
18 def legendre_approximation(x, y, degree):
19     # 计算逼近函数的系数
20     coefficients = legendre_coefficients(x, y, degree)
21     # 使用最小二乘法优化逼近函数的系数
22     result = least_squares(lambda coefficients:
23         approximation_func(x, coefficients) - y, coefficients)
24     return result.x
25
26 x = np.linspace(-1, 1, 100)
27 y = np.exp(x)
28 # 最佳平方逼近
29 degree = 3
30 coefficients = legendre_approximation(x, y, degree)
31 # 计算逼近函数的值
32 approximation = approximation_func(x, coefficients)
33 # 打印结果
```

```
33 print("系数:", coefficients)
```

输出结果为:[1.17530022 1.10367036 0.35831431 0.07053157], 与计算结果一致.

3.6 曲线拟合的最小二乘法

3.6.1 问题提出

已知一系列测量数据 (x_i, y_i) , 要求简单函数 $f(x)$ 使得 $\rho_i = y_i - f(x_i)$ 在总体上尽可能小. 这里 $y_i \neq f(x_i)$, 称 ρ_i 为残差, 这种构造近似函数的方法称为曲线拟合, $f(x)$ 为拟合函数.

通常, 使 ρ_i 尽可能小的度量准则包括:

- 使 $\max_{1 \leq i \leq m} |P(x_i) - y_i|$ 尽可能小;
- 使 $\sum_{i=1}^m |P(x_i) - y_i|$ 尽可能小;
- 使 $\sum_{i=1}^m |P(x_i) - y_i|^2$ 尽可能小

3.6.2 曲线拟合的步骤

1. 根据条件画出散点图;
2. 观察分布, 选择适当的函数类构造拟合函数;
3. 根据某一逼近准则确定拟合函数的未知参数.

3.6.3 2-范数度量下的曲线拟合 (最小二乘法)

给定一组数据 $(x_i, y_i), i = 1, 2, \dots, m$, 确定拟合函数

$$f(x) = c_1\varphi_1(x) + c_2\varphi_2(x) + \dots + c_n\varphi_n(x)$$

使得

$$\|r\|_2^2 = \sum_{i=1}^m \rho_i^2 = \sum_{i=1}^m [y_i - f(x_i)]^2 = \sum_{i=1}^m \left[y_i - \sum_{j=1}^n c_j \varphi_j(x_i) \right]^2$$

达到极小, 这里 $n \leq m$.

若记

$$E(c_1, c_2, \dots, c_n) = \sum_{i=1}^m \left[y_i - \sum_{j=1}^n c_j \varphi_j(x_i) \right]^2$$

则 E 是 c_1, \dots, c_n 的多元函数, 于是问题转化为求多元函数 $E(c_1, c_2, \dots, c_n)$ 的极小值问题.

由多元函数极值必要条件, 在 E 的极值点处, 有

$$\frac{\partial E}{\partial c_k} = \sum_{i=1}^m -2 \left(y_i - \sum_{j=1}^n c_j \varphi_j(x_i) \right) \varphi_k(x_i) = 0, k = 1, 2, \dots, n$$

即

$$\sum_{j=1}^n c_j \sum_{i=1}^m \varphi_j(x_i) \varphi_k(x_i) = \sum_{i=1}^m y_i \varphi_k(x_i), k = 1, 2, \dots, n$$

若记

$$\Phi_i = \begin{pmatrix} \varphi_i(x_1) \\ \varphi_i(x_2) \\ \vdots \\ \varphi_i(x_m) \end{pmatrix}, i = 1, 2, \dots, n, b = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{pmatrix}$$

则

$$\sum_{i=1}^m \varphi_j(x_i) \varphi_k(x_i) = (\Phi_j, \Phi_k) \sum_{i=1}^m y_i \varphi_k(x_i) = (\Phi_k, b)$$

于是上式可写为

$$\sum_{j=1}^n c_j (\Phi_j, \Phi_k) = (\Phi_k, b), k = 1, 2, \dots, n$$

于是有关于 c_1, c_2, \dots, c_n 的线性方程组

$$\begin{pmatrix} (\Phi_1, \Phi_1) & (\Phi_2, \Phi_1) & \cdots & (\Phi_n, \Phi_1) \\ (\Phi_1, \Phi_2) & (\Phi_2, \Phi_2) & \cdots & (\Phi_n, \Phi_2) \\ \vdots & \vdots & & \vdots \\ (\Phi_1, \Phi_n) & (\Phi_2, \Phi_n) & \cdots & (\Phi_n, \Phi_n) \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{pmatrix} = \begin{pmatrix} (\Phi_1, b) \\ (\Phi_2, b) \\ \vdots \\ (\Phi_n, b) \end{pmatrix}$$

或等价写作:

$$\begin{pmatrix} \Phi_1^T \Phi_1 & \Phi_2^T \Phi_1 & \cdots & \Phi_n^T \Phi_1 \\ \Phi_1^T \Phi_2 & \Phi_2^T \Phi_2 & \cdots & \Phi_n^T \Phi_2 \\ \vdots & \vdots & & \vdots \\ \Phi_1^T \Phi_n & \Phi_2^T \Phi_n & \cdots & \Phi_n^T \Phi_n \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{pmatrix} = \begin{pmatrix} \Phi_1^T b \\ \Phi_2^T b \\ \vdots \\ \Phi_n^T b \end{pmatrix}$$

若记

$$A = (\Phi_1, \Phi_2, \dots, \Phi_n) = \begin{pmatrix} \varphi_1(x_1) & \varphi_2(x_1) & \cdots & \varphi_n(x_1) \\ \varphi_1(x_2) & \varphi_2(x_2) & \cdots & \varphi_n(x_2) \\ \vdots & \vdots & & \vdots \\ \varphi_1(x_m) & \varphi_2(x_m) & \cdots & \varphi_n(x_m) \end{pmatrix}$$

$$x = \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{pmatrix}$$

则上述方程组可表示为

$$A^T A x = A^T b$$

由于 A 是列满秩矩阵 (列向量线性无关), 故系数矩阵对称正定, 方程组的解存在唯一, 故可得拟合参数 c_1, c_2, \dots, c_n

扩展: 若矩阵 $A \in \mathbb{R}^{m \times n}$, 则 $A^T A \in \mathbb{R}^{n \times n}$, $AA^T \in \mathbb{R}^{m \times m}$, 它们具有如下共同特点:

- 对称半正定 ($\forall x \in \mathbb{R}^n, x^T(A^T A)x \geq 0$);
- 特征值非负;
- $\text{rank}(A^T A) = \text{rank}(AA^T) = \text{rank}(A)$;
- 非零特征值个数相同, 且非零特征值相同;
- $\text{tr}(AA^T) = \text{tr}(A^T A)$

若取 $\varphi_0 = 1, \varphi_1(x) = x, \varphi_2(x) = x^2, \dots, \varphi_n(x) = x^n$, 则拟合函数

$$f(x) = c_0 + c_1 x + c_2 x^2 + \cdots + c_n x^n$$

称为多项式拟合, 此时

$$A = \begin{pmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^n \\ 1 & x_2 & x_2^2 & \cdots & x_2^n \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x_m & x_m^2 & \cdots & x_m^n \end{pmatrix}$$

$$b = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{pmatrix}$$

$$x = \begin{pmatrix} c_0 \\ c_1 \\ \vdots \\ c_n \end{pmatrix}$$

于是

$$A^T A x = A^T b$$

一些特殊情况下的非线性拟合, 可通过变量代换, 转换为线性拟合. 例如:

1. 双曲拟合

$$\frac{1}{y} = a + b \frac{1}{x} \Rightarrow u = a + bv (u = \frac{1}{y}, v = \frac{1}{x})$$

2. 对数拟合

$$y = a + b \ln x \Rightarrow y = a + bu (u = \ln x)$$

3. 指数拟合

$$y = ae^{bx} \Rightarrow \ln y = \ln a + bx \Rightarrow u = c + bx (u = \ln y, c = \ln a)$$

3.6.4 用正交函数作最小二乘拟合

给定

$$\begin{cases} x_0, \cdots, x_m \\ y_0, \cdots, y_m \end{cases}$$

与正交函数系 $\varphi_0, \varphi_1, \dots, \varphi_n$, 则最小二乘拟合系数为:

$$P_n(x) = a_0\varphi_0(x) + a_1\varphi_1(x) + \dots + a_n\varphi_n(x)$$

其中,

$$a_k = \frac{(f, \varphi_k)}{(\varphi_k, \varphi_k)} = \frac{\sum_{i=0}^m \rho(x_i) y_i \varphi_k(x_i)}{\sum_{i=1}^m \rho(x_i) \varphi_k^2(x_i)}$$

最后以一个最小二乘法的例题结束这一章.

例 3.6.1. 给定 (t_i, f_i) 的一组数据

t_i	f_i
0	81.4
20	77.7
40	74.2
60	72.4
80	70.3
100	68.8

求拟合函数 $f(x)$

解. 作图可知点分布在直线附近, 故使用线性函数拟合. 设

$$\varphi_0(t) = 1, \varphi_1(t) = t$$

将数据代入, 可得

$$A = \begin{pmatrix} 1 & 0 \\ 1 & 20 \\ 1 & 40 \\ 1 & 60 \\ 1 & 80 \\ 1 & 100 \end{pmatrix}, x = \begin{pmatrix} c_1 \\ c_2 \end{pmatrix}, b = \begin{pmatrix} 81.4 \\ 77.7 \\ 74.2 \\ 72.4 \\ 70.3 \\ 68.8 \end{pmatrix}$$

代入公式

$$A^T A x = A^T b$$

可得拟合函数为

$$f(t) = 80.3 - 0.1t$$

□

下面的代码实现了上面的计算功能:

```
1  # 演示线性拟合的最小二乘法 Exercise3-2.py
2  import numpy as np
3  import matplotlib.pyplot as plt
4  # 数据
5  x = np.array([0,20,40,60,80,100])
6  y = np.array([81.4,77.7,74.2,72.4,70.3,68.8])
7  # 进行线性拟合
8  coefficients = np.polyfit(x, y, 1)
9  slope = coefficients[0]
10 intercept = coefficients[1]
11 # 绘制拟合曲线
12 x_fit= np.arange(0,100,0.01)
13 y_fit = slope*x_fit+intercept
14 plt.scatter(x,y)
15 plt.plot(x_fit,y_fit,c="red")
16 plt.show()
17 # 打印拟合结果
18 print(f"表达式为{slope:.1f}t+{intercept:.1f}")
```

拟合结果图像如图 3.1所示.

最后作为扩展演示, 下面是一个使用 Legendre 多项式进行三次拟合的程序:

```
1  # 使用Legendre多项式进行三次拟合 Exercise3-3.py
2  import numpy as np
3  import matplotlib.pyplot as plt
4  from scipy.special import legendre
5  # 示例数据
6  x = np.array([3.73,5.63,8.67,10.49,13.26,16.74,
7  19.55,22.19,25.36,28.31,30.83])
8  y = np.array([21.53,21.43,21.43,21.33,21.22,
9  20.71,20.20,19.69,18.78,17.76,16.84])
```

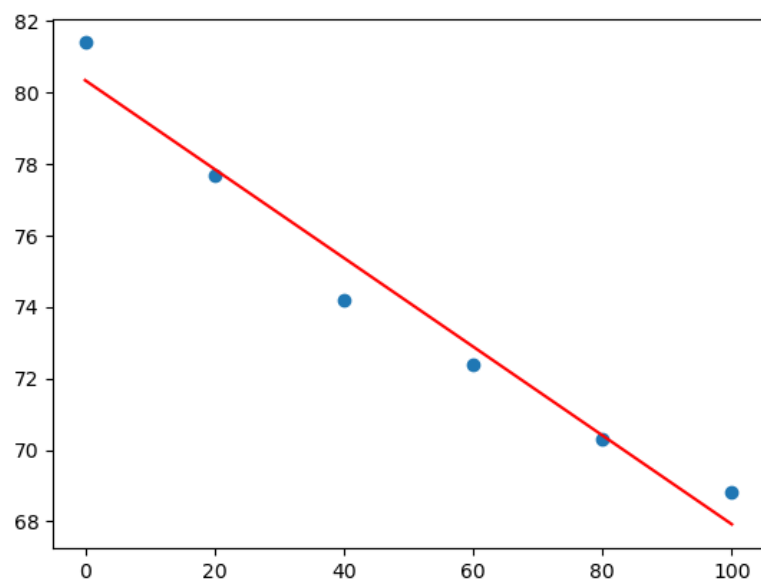


图 3.1: 使用线性拟合结果

```
10 # 进行Legendre拟合
11 coefficients = np.polynomial.legendre.legfit(x, y, 3)
12 # 绘制拟合曲线
13 x_fit = np.arange(3.73,30.83,0.01)
14 y_fit = np.polynomial.legendre.legval(x_fit, coefficients)
15 plt.scatter(x,y)
16 plt.plot(x_fit,y_fit,c="red")
17 plt.show()
```

得到图像如图 3.2所示.

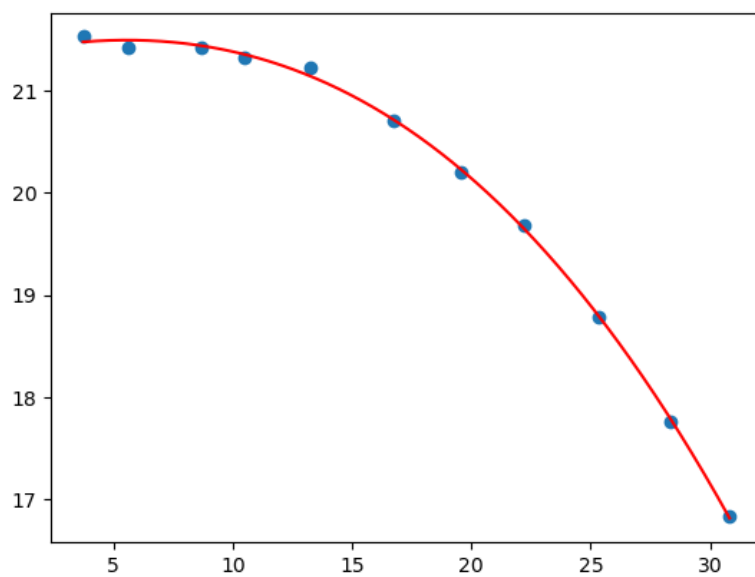


图 3.2: 使用 Legendre 多项式对一组数据进行三次拟合

Chapter 4

数值积分

4.1 引言

4.1.1 数值积分的必要性

本章主要讨论如下形式的一元函数积分

$$I(f) = \int_a^b f(x) \, dx$$

在微积分里, 按 Newton-Leibniz 公式求定积分

$$I(f) = \int_a^b f(x) \, dx = F(b) - F(a)$$

要求 $f(x)$ 的原函数 $F(x)$:

- 有解析表达式;
- 为初等函数.

实际上,

1. $f(x)$ 的原函数 $F(x)$ 不能用初等函数表示, 如 $\sin x^2, \cos x^2, \frac{\sin x}{x}, \frac{1}{\ln x}$ 等;
2. 被积函数的原函数可以用初等函数表示, 但其表达式相当复杂, 计算极不方便, 如 $x^2\sqrt{2x^2+3}$;

3. $f(x)$ 没有表达式, 只有数表形式。

这时就需要积分的数值方法来帮忙了。

4.1.2 数值积分的基本思想

数值积分的理论依据

依据积分中值定理, 对于连续函数 $f(x)$, 在 $[a, b]$ 内存在一点 ξ , 使得

$$I(f) = \int_a^b f(x) dx = (b-a)f(\xi)$$

称 $f(\xi)$ 为 $f(x)$ 在区间 $[a, b]$ 上的平均高度。

求积公式的构造

若简单选取区间端点或中点的函数值作为平均高度, 则求积公式 (左矩形公式, 中矩形公式, 右矩形公式) 如下:

$$I(f) \approx f(a)(b-a)$$

$$I(f) \approx f\left(\frac{a+b}{2}\right)(b-a)$$

$$I(f) \approx f(b)(b-a)$$

定义 4.1.1 (两点求积公式). 若取 a, b 两点, 并令 $f(\xi) = \frac{f(a)+f(b)}{2}$, 则可得梯形公式 (两点求积公式)

$$I(f) \approx \frac{f(a)+f(b)}{2}(b-a)$$

定义 4.1.2 (三点求积公式/Simpson 公式). 若取三点 $a, b, c = \frac{a+b}{2}$ 并令 $f(\xi) = \frac{[f(a)+4f(c)+f(b)]}{6}$, 则可得 Simpson 公式 (三点求积公式)

$$I(f) \approx (b-a) \frac{[f(a)+4f(c)+f(b)]}{6}$$

定义 4.1.3 (机械求积). 一般地, 取区间 $[a, b]$ 内 $n+1$ 个点 $x_i, (i = 0, 1, 2, \dots, n)$ 处的高度 $f(x_i), (i = 0, 1, 2, \dots, n)$, 通过加权平均的方法近似地得出平均高度 $f(\xi)$ 。这类求积方法称为机械求积。

$$\int_a^b f(x) dx \approx (b-a) \sum_{i=0}^n \lambda_i f(x_i)$$

或写成：

$$\int_a^b f(x) dx \approx \sum_{k=0}^n A_k f(x_k)$$

其中 A_k 称为求积系数, x_k 称为求积节点。

记：

数值求积公式：

$$i_n(f) = \sum_{k=0}^n A_k f(x_k)$$

求积公式余项 (误差)：

$$R(f) = I(f) - I_n(f) = \int_a^b f(x) dx - \sum_{k=0}^n A_k f(x_k)$$

构造或确定一个求积公式, 要解决的问题包括：

1. 确定求积系数 A_k 和求积节点 x_k ;
2. 确定衡量求积公式好坏的标准;
3. 求积公式的误差估计和收敛性分析.

4.1.3 求积公式的代数精度

定义 4.1.4. 称求积公式 $I_n(f) = \sum_{k=0}^n A_k f(x_k)$ 具有 m 次代数精度, 如果它满足如下两个条件：

1. 对所有 m 次多项式 $P_m(x)$, 有

$$R(P_m) = I(P_m) - I_n(P_m) = 0$$

2. 存在 $m+1$ 次多项式 $P_{m+1}(x)$, 使得

$$R(P_{m+1}) = I(P_{m+1}) - I_n(P_{m+1}) \neq 0$$

上述定义中的条件等价于：

$$R(x^k) = I(x^k) - I_n(x^k) = 0, (0 \leq k \leq m)$$

$$R(x_{m+1}) \neq 0$$

注意： 梯形公式与中矩形公式都只具有 1 次代数精度。

一般的, 若要使求积公式 (1) 具有 m 次代数精度, 则只要使求积公式对 $f(x) = 1, x, x^2, \dots, x^m$ 都准确成立, 即

$$\begin{cases} \sum_{k=0}^n A_k = b - a \\ \sum_{k=0}^n A_k x_k = \frac{1}{2}(b^2 - a^2) \\ \sum_{k=0}^n A_k x_k^m = \frac{1}{m+1}(b^{m+1} - a^{m+1}) \end{cases}$$

4.2 插值型求积公式

4.2.1 定义

定义 4.2.1 (拉格朗日插值公式). 在积分区间 $[a, b]$ 上, 取 $n+1$ 个节点 $x_i, i = 0, 1, 2, \dots, n$ 作 $f(x)$ 的 n 次代数插值多项式 (拉格朗日插值公式):

$$L_n(x) = \sum_{k=0}^n l_k(x) f(x_k)$$

则有 $f(x) = L_n(x) + R_n(x)$ 其中, $R_n(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!} w_{n+1}(x)$ 为插值余项。
 $w_{n+1}(x) = \prod_{j=0}^n (x - x_j)$

于是有:

$$\begin{aligned} \int_a^b f(x) dx &= \int_a^b L_n(x) dx + \int_a^b R_n(x) dx \\ &= \sum_{j=0}^n \left[\int_a^b l_j(x) dx \right] f(x_j) + \int_a^b R_n(x) dx \end{aligned}$$

取

$$\int_a^b f(x) dx \approx \sum_{j=0}^n f(x_j) \int_a^b l_j(x) dx$$

其中 $\int_a^b l_j(x) dx$ 为 A_j 。

$A_j = \int_a^b \prod_{i=0, i \neq j}^n \frac{(x - x_i)}{(x_j - x_i)} dx$ 由节点决定, 与 $f(x)$ 无关。

4.2.2 截断误差与代数精度

截断误差

$$\begin{aligned} R(f) &= \int_a^b f(x) dx - \sum_{k=0}^n A_k f(x_k) \\ &= \int_a^b [f(x) - L_n(x)] dx \\ &= \int_a^b \frac{f^{(n+1)}(\xi_x)}{(n+1)!} \prod_{k=0}^n (x - x_k) dx \end{aligned}$$

代数精度

定理 4.2.1. 形如 $\sum_{k=0}^n A_k f(x)$ 的求积公式至少有 n 次代数精度 \Leftrightarrow 该公式为插值型 (即: $A_k = \int_a^b l_k(x) dx$)

推论 4.2.2. 求积系数 A_k 满足:

$$\sum_{k=0}^n A_k = b - a$$

4.3 Newton-Cotes 公式

4.3.1 Cotes 系数

取节点为等距分布: $x_i = a + ih, h = \frac{b-a}{n}, i = 0, 1, \dots, n$ 由此构造的插值型求积公式称为 Newton-Cotes 公式, 此时求积系数:

$$\begin{aligned} A_i &= \int_{x_0}^{x_n} \prod_{j \neq i} \frac{(x - x_j)}{x_i - x_j} dx \\ &= \int_0^n \prod_{i \neq j} \frac{(t - j)h}{(i - j)h} \times h dt \\ &= \frac{(b-a)(-1)^{n-i}}{ni!(n-i)!} \int_0^n \prod_{j \neq i} (t - j) dt \\ &= C_i^{(n)}(b-a) \end{aligned}$$

其中, $C_i^{(n)}$ 为 Cotes 系数。

4.3.2 Newton-Cotes 公式

定义

定义 4.3.1 (n 阶闭型 Newton-Cotes 求积公式). 记:

$$C_i^{(n)} = \frac{(-1)^{n-i}}{i!(n-i)!n} \int_0^n \left[\prod_{k=0, k \neq i}^n (t-k) \right] dt$$

则 $A_i = (b-a)C_i^{(n)}, i = 0, 1, 2, \dots, n$ 求积公式变为

$$\int_a^b f(x) dx \approx (b-a) \sum_{i=0}^n C_i^{(n)} f(x_i)$$

称上式为 n 阶闭型 Newton-Cotes 求积公式。

注意: 由式 $C_i^{(n)} = \frac{(-1)^{n-i}}{i!(n-i)!n} \int_0^n [\prod_{k=0, k \neq i}^n (t-k)] dt$ 确定的 Cotes 系数只与 i 和 n 有关, 与 $f(x)$ 和积分区间 $[a, b]$ 无关, 且满足: (1) $C_i^{(n)} = C_{n-i}^{(n)}$
(2) $\sum_{k=0}^n C_k^{(n)} = 1$

截断误差

Newton-Cotes 公式的截断误差为:

$$\begin{aligned} R(f) &= \int_a^b \frac{f^{(n+1)}(\xi)}{(n+1)!} w_{(n+1)}(x) dx \\ &= \frac{h^{n+2}}{(n+1)!} \int_0^n f^{(n+1)}(\xi) \left[\prod_{j=0}^n (t-j) \right] dt, \xi \in (a, b) \end{aligned}$$

代数精度

定理 4.3.1. 当阶数 n 为偶数时, Newton-Cotes 公式至少具有 $n+1$ 次代数精度。

证明. 只需验证当 n 为偶数时, Newton-Cotes 公式对 $f(x) = x^{n+1}$ 的余项为零。由于 $f(x) = x^{n+1}$, 所以 $f^{(n+1)}(x) = (n+1)!$ 即得 $R(f) = h^{n+2} \int_0^n \prod_{j=0}^n (t-j) dt$ 引进变换 $t = u + \frac{n}{2}$, 因为 n 为偶数, 故 $\frac{n}{2}$ 为整数, 于是有

$$R(f) = h^{n+2} \int_{-\frac{n}{2}}^{\frac{n}{2}} \prod_{j=0}^n (u + \frac{n}{2} - j) du$$

据此可得 $R(f) = 0$, 因为上述被积函数是个奇函数。 \square

数值稳定性

现在讨论舍入误差对计算结果产生的影响。设用公式

$$I_n(f) = (b-a) \sum_{j=0}^n C_j^{(n)} f(x_j)$$

近似计算积分

$$I(f) = \int_a^b f(x) dx$$

时, 其中函数值 $f(x_j)$ 有误差 $\varepsilon_j (j=0, 1, 2, \dots, n)$, 而计算 $C_j^{(n)}$ 没有误差, 中间计算过程中的舍入误差也不考虑, 则在 $I_n(f)$ 的计算中, 由 ε_j 引起的误差为:

$$\begin{aligned} e_n &= (b-a) \sum_{j=0}^n C_j^{(n)} f(x_j) - (b-a) \sum_{j=0}^n C_j^{(n)} (f(x_j) + \varepsilon_j) \\ &= -(b-a) \sum_{j=0}^n C_j^{(n)} \varepsilon_j \end{aligned}$$

如果 $C_j^{(n)}$ 都是正数, 并设 $\varepsilon = \max_{0 \leq j \leq n} |\varepsilon_j|$ 则有

$$|e_n| \leq \varepsilon (b-a) \sum_{j=0}^n C_j^{(n)} = \varepsilon (b-a)$$

故 e_n 是有界的, 即由 ε_j 引起的受到控制, 不超过 ε 的 $(b-a)$ 倍, 保证了数值计算的稳定性。而当 $n > 7$ 时, $C_j^{(n)}$ 将出现负数, $\sum_{j=0}^n |C_j^{(n)}|$ 将随 n 增大, 因而不能保证数值稳定性。故高阶公式不宜采用, 有实用价值的仅仅是几种低阶的求积公式。

4.3.3 几种常见的低阶求积公式

梯形公式: (代数精度 =1)

$$\begin{aligned}
 n &= 1 : \\
 C_0^{(1)} &= \frac{1}{2}, C_1^{(1)} = \frac{1}{2} \\
 \int_a^b f(x) dx &\approx \frac{b-a}{2} [f(a) + f(b)] \\
 R[f] &= \int_a^b \frac{f''(\xi_x)}{2!} (x-a)(x-b) dx \\
 &= -\frac{(b-a)^3}{12} f''(\xi), \xi \in [a, b]
 \end{aligned}$$

Simpson 公式: (代数精度 =3)

$$\begin{aligned}
 n &= 2 : \\
 C_0^{(2)} &= \frac{1}{6}, C_1^{(2)} = \frac{2}{3}, C_2^{(2)} = \frac{1}{6} \\
 \int_a^b f(x) dx &\approx \frac{b-a}{6} [f(a) + 4f(\frac{a+b}{2}) + f(b)] \\
 R[f] &= -\frac{(b-a)^5}{2880} f^{(4)}(\xi), \xi \in (a, b)
 \end{aligned}$$

Cotes 公式: (代数精度 =5)

$$\begin{aligned}
 n &= 4 : \\
 \int_a^b f(x) dx &\approx \frac{b-a}{90} [7f(x_0) + 32f(x_1) + 12f(x_2) + 32f(x_3) + 7f(x_4)] \\
 x_k &= a + kh, h = \frac{b-a}{4}, k = 0, 1, 2, 3, 4 \\
 R[f] &= -\frac{8}{945} \left(\frac{b-a}{4}\right)^7 f^{(6)}(\xi) \\
 &= -\frac{2(b-a)}{945} \left(\frac{b-a}{4}\right)^6 f^{(6)}(\xi), \xi \in [a, b]
 \end{aligned}$$

4.3.4 复化求积公式

定义 4.3.2 (复化梯形公式). $h = \frac{b-a}{n}$, $x_k = a + kh$ ($k = 0, \dots, n$) 在每个 $[x_k, x_{k+1}]$ 上用梯形公式:

$$\begin{aligned}\int_{x_k}^{x_{k+1}} f(x) dx &\approx \frac{x_{k+1} - x_k}{2} [f(x_k) + f(x_{k+1})], k = 0, \dots, n-1 \\ \int_a^b f(x) dx &\approx \sum_{k=0}^{n-1} \frac{h}{2} [f(x_k) + f(x_{k+1})] = \frac{h}{2} [f(a) + 2 \sum_{k=1}^{n-1} f(x_k) + f(b)] = T_n \\ R[f] &= \sum_{k=0}^{n-1} [-\frac{h^3}{12} f''(\eta_k)] = -\frac{h^2}{12} (b-a) \frac{\sum_{k=0}^{n-1} f''(\eta_k)}{n} = -\frac{h^2}{12} (b-a) f''(\eta), \eta \in (a, b)\end{aligned}$$

定义 4.3.3 (复化 Simpson 公式). $h = \frac{b-a}{n}$, $x_k = a + kh$ ($k = 0, \dots, n$) 在每个 $[x_k, x_{k+1}]$ 上使用 Simpson 公式:

$$\begin{aligned}\int_{x_k}^{x_{k+1}} f(x) dx &\approx \frac{h}{6} [f(x_k) + 4f(x_{k+\frac{1}{2}}) + f(x_{k+1})] \\ \int_a^b f(x) dx &\approx \sum_{k=0}^{n-1} \int_{x_k}^{x_{k+1}} f(x) dx \approx \frac{h}{6} [f(a) + 4 \sum_{k=0}^{n-1} f(x_{k+\frac{1}{2}}) + 2 \sum_{k=1}^{n-1} f(x_k) + f(b)] = S_n \\ R[f] &= -\frac{b-a}{2880} h^4 f^{(4)}(\xi) = -\frac{b-a}{180} (\frac{h}{2})^4 f^{(4)}(\xi), \xi \in (a, b)\end{aligned}$$

定义 4.3.4 (复化 Cotes 公式). $h = \frac{b-a}{n}$, $x_k = a + kh$ ($k = 0, \dots, n$)

$$\begin{aligned}\int_{x_k}^{x_{k+1}} f(x) dx &\approx \frac{h}{90} [7f(x_k) + 32f(x_{k+\frac{1}{4}}) + 12f(x_{k+\frac{1}{2}}) + 32f(x_{k+\frac{3}{4}}) + 7f(x_{k+1})] \\ \int_a^b f(x) dx &\approx \sum_{k=0}^{n-1} [7f(x_k) + 32f(x_{k+\frac{1}{4}}) + 12f(x_{k+\frac{1}{2}}) + 32f(x_{k+\frac{3}{4}}) + 7f(x_{k+1})] = C_n \\ R[f] &= -\frac{2(b-a)}{945} (\frac{h}{4})^6 f^{(6)}(\xi), \xi \in (a, b)\end{aligned}$$

例 4.3.1. 利用数据表

复化梯形公式的误差估计

误差先验估计式: 给定精度 ε , 如何求 n ?

例 4.3.2. 要求 $|I - T_n| < \varepsilon$, 如何判断 $n = ?$ 由

$$R[f] = I(f) - T_n(f) = -\frac{h^2}{12} (b-a) f''(\xi)$$

若记 $M_2 = \max_{a \leq x \leq b} |f''(x)|$ 则可令

$$\left| \frac{h^2}{12}(b-a)f''(\xi) \right| \leq \frac{h^2}{12}(b-a)M_2 \leq \varepsilon$$

其中, $h = \frac{b-a}{n}$, 于是, $n \geq \sqrt{\frac{(b-a)^3 M_2}{12\varepsilon}}$

$$\begin{aligned} R[f] &= -\frac{h^2}{12}(b-a)f''(\varepsilon) \\ &= -\frac{h^2}{12} \sum_{k=1}^n [f''(\xi_k) \cdot h] \approx -\frac{h^2}{12} \int_a^b f''(x) dx \\ &= -\frac{h^2}{12} [f'(b) - f'(a)] \end{aligned}$$

上例中若要求 $|I - T_n| < 10^{-6}$, 则

$$|R_n[f]| \approx \frac{h^2}{12} |f'(1) - f'(0)| = \frac{h^2}{6} < 10^{-6} \Rightarrow h < 0.00244949$$

即取 $n = 409$ 通常采取将区间不断对分的方法, 即取 $n = 2^k$ 上例中 $2^k \geq 409 \Rightarrow k = 9$ 时, $T_{512} = 3.141592502$

误差后验估计式: 注意到区间再次对分时

$$\begin{aligned} R_{2n}[f] &\approx -\frac{1}{12} [f'(b) - f'(a)] \left(\frac{h}{2}\right)^2 \\ &\approx \frac{1}{4} R_n[f] \\ \Rightarrow \frac{I - T_{2n}}{I - T_n} &\approx \frac{1}{4} \\ \Rightarrow I - T_{2n} &\approx \frac{1}{3} (T_{2n} - T_n) = \frac{1}{4-1} (T_{2n} - T_n) \end{aligned}$$

可用来判断迭代是否停止。

复化 Simpson 公式的误差估计

误差先验估计式:

$$\begin{aligned} R[f] &= I - S_n = -\frac{b-a}{180} \left(\frac{h}{4}\right)^4 f^{(4)}(\xi) \\ &\approx -\frac{1}{180} [f^{(3)}(b) - f^{(3)}(a)] \left(\frac{h}{2}\right)^4 \end{aligned}$$

误差后验估计式:

$$I - S_{2n} \approx \frac{1}{15} (S_{2n} - S_n) = \frac{1}{4^2 - 1} (S_{2n} - S_n)$$

复化 Cotes 公式的误差估计

误差先验估计式:

$$\begin{aligned} R[f] &= I - C_n = -\frac{2(b-a)}{945} \left(\frac{h}{4}\right)^6 f^{(6)}(\xi), \xi \in (a, b) \\ &\approx -\frac{2}{945} [f^{(5)}(b) - f^{(5)}(a)] \left(\frac{h}{4}\right)^6 \end{aligned}$$

误差后验估计式:

$$I - C_{2n} \approx \frac{1}{63} (C_{2n} - C_n) = \frac{1}{4^3 - 1} (C_{2n} - C_n)$$

复化求积公式的收敛速度 (阶)

定义 4.3.5 (p 阶收敛). 若一个复化求积公式的误差满足 $\lim_{h \rightarrow 0} \frac{R[f]}{h^p} = C < \infty$ 且 $C \neq 0$, 则称该公式是 p 阶收敛的。

注意: 根据上述定义不难验证

$$T_n \sim O(h^2), S_n \sim O(h^4), C_n \sim O(h^6) \quad (4.1)$$

4.4 Romberg 算法

4.4.1 Romberg 求积公式

例 4.4.1. 计算

$$\pi = \int_0^1 \frac{4}{1+x^2} dx$$

若取 $\varepsilon = 10^{-6}$, 则利用复化求积公式进行计算时, 须将区间对分 9 次, 得到 $T_{512} = 3.141592502$

考察 $\frac{I-T_{2n}}{I-T_n} \approx \frac{1}{4}$, 若由 $I \approx \frac{4T_{2n}-T_n}{4-1} = \frac{4}{3}T_{2n} - \frac{1}{3}T_n$ 来计算 I 效果是否好些?

$$\frac{4}{3}T_8 - \frac{1}{3}T_4 = 3.141592502 = S_4$$

一般有:

- $\frac{4T_{2n}}{4-1} = S_n$
- $\frac{4^2 S_{2n} - S_n}{4^2 - 1} = C_n$
- $\frac{4^3 C_{2n} - C_n}{4^3 - 1} = R_n$ (Romberg 求积公式)

4.4.2 理查德森外推加速法

利用低阶公式产生高精度的结果。设对于某一 $h \neq 0$, 有公式 $T_0(h)$ 近似计算某一未知值 I 。由 Taylor 展开得到:

$$T_0(h) - I = \alpha_1 h + \alpha_2 h^2 + \alpha_3 h^3 + \cdots$$

现将 h 对分, 得:

$$T_0\left(\frac{h}{2}\right) - I = \alpha_1\left(\frac{h}{2}\right) + \alpha_2\left(\frac{h}{2}\right)^2 + \alpha_3\left(\frac{h}{2}\right)^3 + \cdots$$

如何将公式精度由 $O(h)$ 提高到 $O(h^2)$?

$$\frac{2T_0\left(\frac{h}{2}\right) - T_0(h)}{2 - 1} - I = -\frac{1}{2}\alpha_2 h^2 - \frac{3}{4}\alpha_3 h^3 - \cdots$$

即:

$$\begin{aligned} T_1(h) &= \frac{2T_0\left(\frac{h}{2}\right) - T_0(h)}{2 - 1} = I + \beta_1 h^2 + \beta_2 h^3 + \cdots \\ T_2(h) &= \frac{2^2 T_1\left(\frac{h}{2}\right) - T_1(h)}{2^2 - 1} = I + \gamma_1 h^3 + \gamma_2 h^4 + \cdots \\ \Rightarrow T_m(h) &= \frac{2^m T_{m-1}\left(\frac{h}{2}\right) - T_{m-1}(h)}{2^m - 1} = I + \sigma_1 h^{m+1} + \sigma_2 h^{m+2} + \cdots \end{aligned}$$

4.4.3 复化梯形公式的渐进展开式

定理 4.4.1. 设 $f(x) \in C^\infty[a, b]$, 则成立

$$T_n(f) = I + a_1 h^2 + a_2 h^4 + \cdots + a_k h^{2k} + \cdots$$

其中, $h = \frac{b-a}{n}$, 系数 $a_k (k=1, 2, \cdots)$ 与 h 无关。

加速公式:

$$T_m^{(k)} = \frac{4^m T_{m-1}^{(k+1)} - T_{m-1}^{(k)}}{4^m - 1}, k=1, 2, \cdots; m=1, 2, \cdots$$

例 4.4.2. 求形如 $\int_{-1}^1 f(x)dx \approx A_0 f(x_0) + A_1 f(x_1)$ 的两点求积公式。

(1) 求梯形公式 (即以 $x_0 = -1, x_1 = 1$ 为节点的插值型求积公式) 立即可得。

$$\int_{-1}^1 f(x)dx \approx f(-1) + f(1)$$

(只具有 1 次代数精度) (2) 若对求积公式中的四个待定系数 A_0, A_1, x_0, x_1 适当选取, 使求积公式对 $f(x) = 1, x, x^2, x^3$ 都准确成立, 则 A_0, A_1, x_0, x_1 需满足如下方程组:

$$\begin{cases} A_0 + A_1 = b - a \\ A_0x_0 + A_1x_1 = \frac{1}{2}(b^2 - a^2) \\ A_0x_0^2 + A_1x_1^2 = \frac{1}{3}(b^3 - a^3) \\ A_0x_0^3 + A_1x_1^3 = \frac{1}{4}(b^4 - a^4) \end{cases}$$

4.5 Gauss 型求积公式

定义 4.5.1 (Gauss 型求积公式). 具有 $2n + 1$ 次代数精度的插值型求积公式

$$\int_a^b \rho(x)f(x) dx \approx \sum_{k=0}^n A_k f(x_k)$$

称为 Gauss 型求积公式。

注意: Gauss 型求积公式是代数精度最高的插值型求积公式。

事实上, 对于插值型求积公式

$$\int_a^b \rho(x)f(x) dx \approx \sum_{k=0}^n nA_k f(x_k)$$

其代数精度最高可达 $2n + 1$ 次 (Gauss 型求积公式)。考虑 $2n + 2$ 次多项式 $w_{n+1}^2(x)$, 其中 $w_{n+1}(x) = \prod_{j=0}^n (x - x_j)$

$$\int_a^b \rho(x)w_{n+1}^2(x) dx > 0$$

而 $\sum_{k=0}^n A_k w_{n+1}^2(x_k) = 0$ 故

$$\int_a^b \rho(x)w_{n+1}^2(x) dx \neq \sum_{k=0}^n A_k w_{n+1}^2(x_k) = 0$$

4.5.1 高斯型求积公式的构造

1. 待定系数法将节点 x_0, \dots, x_n 以及系数 A_0, \dots, A_n 都作为待定系数。并令求积公式对 $f(x) = 1, x, x^2, \dots, x^{2n-1}$ 精确成立可得非线性方程组:

$$\begin{cases} \sum_{k=0}^n A_k = b - a \\ \sum_{k=0}^n A_k x_k = \frac{1}{2(b^2 - a^2)} \\ \vdots \\ \sum_{k=0}^n A_k x_k^{2n+1} = \frac{1}{2n+2} (b^{2n+2} - a^{2n+2}) \end{cases}$$

求解该方程组即可得相应的求积节点与求积系数。

例 4.5.1. 求 $\int_0^1 \sqrt{x} f(x) dx$ 的 2 点 Gauss 公式。

解. 设 $\int_0^1 \sqrt{x} f(x) dx \approx A_0 f(x_0) + A_1 f(x_1)$, 应有 3 次代数精度。令上述公式 $f(x) = 1, x, x^2, x^3$ 精确成立可得

$$\begin{cases} \frac{2}{3} = A_0 + A_1 \\ \frac{2}{5} = A_0 x_0 + A_1 x_1 \\ \frac{2}{7} = A_0 x_0^2 + A_1 x_1^2 \\ \frac{2}{9} = A_0 x_0^3 + A_1 x_1^3 \end{cases}$$

(不是线性方程组, 不易求解。) 解得:

$$\begin{cases} x_0 \approx 0.8212 \\ x_1 \approx 0.2899 \\ A_0 \approx 0.3891 \\ A_1 \approx 0.2776 \end{cases}$$

□

2. 正交多项式法

定理 4.5.1. x_0, \dots, x_n 为 Gauss 点 $\Leftrightarrow w(x) = \prod_{k=0}^n (x - x_k)$ 与任意次数不大于 n 的多项式 $P(x)$ (带权) 正交。

证明. " \Rightarrow ": x_0, \dots, x_n 为 Gauss 点, 则公式

$$\int_a^b \rho(x) f(x) dx \approx \sum_{k=0}^n A_k f(x_k)$$

具有 $2n+1$ 次代数精度。

对任意次数不大于 n 的多项式 $P_n(x)$, $P_n(x)w(x)$ 的次数不大于 $2n+1$, 则代入公式应精确成立:

$$\int_a^b \rho(x)P_m(x)w(x) dx = \sum_{k=0}^n A_k P_m(x_k)w(x_k) = 0$$

” \Leftarrow ”: 要证明 x_0, \dots, x_n 为 Gauss 点, 即要证公式对任意次数不大于 $2n+1$ 的多项式 $P_{2n+1}(x)$ 精确成立, 即证明:

$$\int_a^b \rho(x)P_m(x) dx = \sum_{k=0}^n A_k P_m(x_k)$$

设 $P_{2n+1}(x) = w(x)q(x) + r(x)$

$$\begin{aligned} \int_a^b \rho(x)P_m(x) dx &= \int_a^b \rho(x)w(x)q(x) dx + \int_a^b \rho(x)r(x) dx \\ &= \sum_{k=0}^n A_k r(x_k) \\ &= \sum_{k=0}^n A_k P_m(x_k) \end{aligned}$$

□

正交多项式族 $\varphi_0, \varphi_1, \dots, \varphi_n, \dots$ 有性质: 任意次数不大于 n 的多项式 $P(x)$ 必与 φ_{n+1} 正交。 \Rightarrow 若取 $w(x)$ 为其中的 φ_{n+1} , 则 φ_{n+1} 的根就是 Gauss 点。

例 4.5.2. 使用正交多项式重解上例

解.

$$\int_0^1 \sqrt{x}f(x) dx \approx A_0 f(x_0) + A_1 f(x_1)$$

Step1: 构造正交多项式 φ_2 设 $\varphi_0(x) = 1, \varphi_1(x) = x + a, \varphi_2(x) = x^2 + bx + c$

1. $(\varphi_0, \varphi_1) = 0 \Rightarrow \int_0^1 \sqrt{x}(x+a) dx = 0 \Rightarrow a = -\frac{3}{5}$
2. $(\varphi_0, \varphi_2) = 0 \Rightarrow \int_0^1 \sqrt{x}(x^2 + bx + c) dx = 0 \Rightarrow b = -\frac{10}{9}$
3. $(\varphi_1, \varphi_2) = 0 \Rightarrow \int_0^1 \sqrt{x}(x - \frac{3}{5})(x + bx + c) dx = 0 \Rightarrow c = \frac{5}{21}$

即 $\varphi_2(x) = x^2 - \frac{10}{9}x + \frac{5}{21}$

Step2: 求 $\varphi_2 = 0$ 的 2 个根, 即为 Gauss 点 x_0, x_1

$$x_{0;1} = \frac{\frac{10}{9} \pm \sqrt{(\frac{10}{9})^2 - \frac{20}{21}}}{2}$$

Step3: 代入 $f(x) = 1, x$ 以求解 A_0, A_1

结果与前一方法相同: $x_0 \approx 0.8212, x_1 \approx 0.2899, A_0 \approx 0.3891, A_1 \approx 0.2776$ 利用此公式计算 $\int_0^1 \sqrt{x}e^x dx$ 的值

$$\begin{aligned} \int_0^1 \sqrt{x}e^x dx &\approx A_0 e^{x_0} + A_1 e^{x_1} \\ &= 0.3891 \times e^{0.8212} + 0.2776 \times e^{0.2899} \approx 1.2555 \end{aligned}$$

□

注意: 构造正交多项式也可以利用 $L-S$ 拟合中介绍过的递推式进行。

4.5.2 几种常见的 Gauss 型求积公式

Gauss-Legendre 求积公式

Legendre 多项式: 定义在 $[-1, 1]$, 上 $\rho(x) \equiv 1$

$$P_k(x) = \frac{1}{2^k k!} \frac{d^k}{dx^k} (x^2 - 1)^k$$

满足:

$$(P_k, P_l) = \begin{cases} 0 & k \neq l \\ \frac{2}{2k+1} & k = l \end{cases}$$

$P_0 = 1, P_1 = x$, 递推公式: $(k+1)P_{k+1} = (2k+1)xP_k - kP_{k-1}$

Gauss-Legendre 求积公式:

$$\int_{-1}^1 f(x) dx \approx \sum_{k=0}^n A_k f(x_k)$$

其中, 求积节点为 P_{n+1} 的根 (求积系数通过解线性方程组得到)。

区间 $[a, b]$ 上的 Gauss-Legendre 公式:

$$\begin{aligned}\int_a^b f(x) dx &= \int_{-1}^1 f\left(\frac{b-a}{2}t + \frac{b+a}{2}\right) \cdot \frac{b-a}{2} dt \\ &\approx \sum_{k=0}^n A_k \frac{b-a}{2} f\left(\frac{b-a}{2}t_k + \frac{b+a}{2}\right)\end{aligned}$$

其中, t_k 为 $n+1$ 次 Legendre 多项式 P_{n+1} 的根。

Gauss-Chebyshev 求积公式:

Chebyshev 多项式: 定义在 $[-1, 1]$ 上, $\rho(x) = \frac{1}{\sqrt{1-x^2}}$

$$T_k(x) = \cos(k \times \arccos x)$$

T_{n+1} 的根为:

$$x_k = \cos\left(\frac{2k+1}{2n+2}\pi\right), k = 0, \dots, n$$

以此为节点构造公式

$$\int_{-1}^1 \frac{f(x)}{\sqrt{1-x^2}} dx \approx \sum_{k=0}^n A_k f(x_k) = \frac{\pi}{n+1} \sum_{k=0}^n f(x_k)$$

称为 Gauss-Chebyshev 多项式。**注意:** 到积分端点 ± 1 可能是积分的奇点, 用普通 Newton-Cotes 公式在端点会出问题。而 Gauss 公式可能避免此问题的发生。

第二类 Gauss-Chebyshev 求积公式

第二类 Chebyshev 多项式: 区间 $[-1, 1]$ 上, 带权 $\rho(x) = \sqrt{1-x^2}$

$$U_n(x) = \frac{\sin[(n+1)\arccos x]}{\sqrt{1-x^2}} \quad (n = 0, 1, 2, \dots)$$

以 U_{n+1} 的零点作为求积节点构造的公式

$$\begin{aligned}\int_{-1}^1 \sqrt{1-x^2} f(x) dx &\approx \sum_{k=0}^n A_k f(x_k) \\ &= \frac{\pi}{n+1} \sum_{k=0}^n \sin^2 \frac{k\pi}{n+1} f\left(\cos \frac{k\pi}{n+1}\right)\end{aligned}$$

称为第二类 Gauss-Chebyshev 公式。

Gauss-Laguerre 求积公式

Laguerre 多项式:

$$L_n(x) = e^x \frac{d^n}{dx^n} (x^n e^{-x}), x \in [0, +\infty], n = 0, 1, 2, \dots$$

$$\rho(x) = e^{-x}$$

以 $n+1$ 次 Laguerre 多项式的零点为求积节点构造的公式

$$\int_0^{+\infty} e^{-x} f(x) dx \approx \sum_{k=0}^n A_k f(x_k)$$

称为 Gauss-Laguerre 求积公式。

Gauss-Hermite 求积公式

Hermite 多项式:

$$H_n(x) = (-1)^n \cdot e^{x^2} \cdot \frac{d^n}{dx^n} (e^{-x^2}), x \in (-\infty, +\infty), n = 0, 1, 2, \dots$$

$$\rho(x) = e^{-x^2}$$

以 $n+1$ 次 Hermite 多项式的零点为求积节点构造的公式

$$\int_{-\infty}^{+\infty} e^{-x^2} f(x) dx \approx \sum_{k=0}^n A_k f(x_k)$$

称为 Gauss-Hermite 求积公式。

4.5.3 Gauss 公式的余项

$$\begin{aligned} R[f] &= \int_a^b f(x) dx - \sum_{k=0}^n A_k f(x_k) \\ &= \int_a^b f(x) dx - \sum_{k=0}^n A_k P(x_k) \\ &= \int_a^b f(x) dx - \int_a^b P(x) dx \\ &= \int_a^b [f(x) - P(x)] dx \end{aligned}$$

4.5.4 Hermite 多项式的余项

Hermite 多项式的插值条件为: $H(x_k) = f(x_k), H'(x_k) = f'(x_k)$ Hermite 插值余项为

$$R(x) = f(x) - H(x) = \frac{f^{(2n+2)}(\xi_x)}{(2n+2)!} \omega^2(x)$$

其中, $\omega(x) = (x - x_0) \cdots (x - x_n)$, ξ_x 与 x 有关。

$$\begin{aligned} \Rightarrow R[f] &= \int_a^b R(x) dx \\ &= \int_a^b \frac{f^{(2n+2)}(\xi_x)}{(2n+2)!} \omega^2(x) dx \\ &= \frac{f^{(2n+2)}(\xi_x)}{(2n+2)!} \int_a^b \omega^2(x) dx \end{aligned}$$

其中, $\xi \in (a, b)$

4.5.5 Gauss 型求积公式的收敛性

定理 4.5.2. 若 $f(x) \in C[a, b]$, 则 Gauss 型求积公式所求积分值序列 $I_n(f) = \sum_{k=1}^n A_k f(x_k)$ 收敛于积分值 $I(f)$, 即

$$\lim_{n \rightarrow \infty} \sum_{k=1}^n A_k f(x_k) = \int_a^b \rho(x) f(x) dx$$

4.5.6 Gauss 型求积公式的数值稳定性

定理 4.5.3. Gauss 型求积公式的求积系数都大于零, 从而 Gauss 型求积公式是数值稳定的。

复化 Gauss-Legendre 求积公式

$$h = \frac{b-a}{n}, x_k = a + kh \quad (k = 0, \cdots, n)$$

$$\begin{aligned} \int_{x_k}^{x_{k+1}} f(x) dx &= \frac{h}{2} \int_{-1}^1 f\left(\frac{h}{2}t + a + \left(k + \frac{1}{2}\right)h\right) dt \\ &\approx \frac{h}{2} \sum_{k=0}^n A_k f\left(\frac{h}{2}t_k + a + \left(k + \frac{1}{2}\right)h\right) \end{aligned}$$

其中, t_k 为 $n+1$ 次 Legendre 多项式 P_{n+1} 的根。从而

$$\int_a^b f(x) dx \approx \frac{b-a}{2n} \sum_{k=0}^{n-1} \left[\sum_{j=0}^n A_j^{(k)} f\left(\frac{h}{2} t_j^{(k)} + a + \left(k + \frac{1}{2}\right)h\right) \right]$$

4.5.7 复化两点 Gauss-Legendre 求积公式

$$\int_a^b f(x) dx \approx \frac{b-a}{2n} \sum_{k=0}^{n-1} \left[F_k\left(-\frac{1}{\sqrt{3}}\right) + F_k\left(\frac{1}{\sqrt{3}}\right) \right]$$

其中, $F_k(t) = f\left(\frac{h}{2}t + a + \left(k + \frac{1}{2}\right)h\right)$

4.5.8 复化三点 Gauss-Legendre 求积公式

$$\int_a^b f(x) dx \approx \frac{b-a}{2n} \sum_{k=0}^{n-1} \left[\frac{5}{9} F_k\left(-\sqrt{\frac{3}{5}}\right) + \frac{8}{9} F_k(0) + \frac{5}{9} F_k\left(\sqrt{\frac{3}{5}}\right) \right]$$

Chapter 5

线性方程组的直接解法

5.1 Gauss 消去法

5.1.1 三角形方程组回代法

如:

$$\begin{cases} a_{11}x_1 & = b_1 \\ a_{21}x_1 + a_{22}x_2 & = b_2 \\ \cdots & \cdots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n & = b_n \end{cases}$$

或上三角方程组:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1 \\ a_{22}x_2 + \cdots + a_{2n}x_n = b_2 \\ \cdots \\ a_{nn}x_n = b_n \end{cases}$$

5.1.2 顺序 Gauss 消去法

对于一般的线性方程组

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \cdots + a_{1n}x_n = a_{1,n+1} \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + \cdots + a_{2n}x_n = a_{2,n+1} \\ \cdots \\ a_{n1}x_1 + a_{n2}x_2 + a_{n3}x_3 + \cdots + a_{nn}x_n = a_{n,n+1} \end{cases}$$

记

$$\begin{aligned} \overline{A}^{(1)} &= (A, b) \\ &= \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} & a_{1,n+1} \\ a_{21} & a_{22} & \cdots & a_{2n} & a_{2,n+1} \\ \vdots & \vdots & & \vdots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} & a_{n,n+1} \end{pmatrix} = \begin{pmatrix} a_{11}^{(1)} & a_{12}^{(1)} & \cdots & a_{1n}^{(1)} & a_{1,n+1}^{(1)} \\ a_{21}^{(1)} & a_{22}^{(1)} & \cdots & a_{2n}^{(1)} & a_{2,n+1}^{(1)} \\ \vdots & \vdots & & \vdots & \vdots \\ a_{n1}^{(1)} & a_{n2}^{(1)} & \cdots & a_{nn}^{(1)} & a_{n,n+1}^{(1)} \end{pmatrix} \end{aligned}$$

解 n 阶方程组的顺序 Gauss 消去一般步骤

第一次消元, 设 $a_{11}^{(1)} \neq 0$, 令

$$m_{i1} = \frac{a_{i1}^{(1)}}{a_{11}^{(1)}}$$

称 m_{i1} 为消元因子. 令 $a_{ij}^{(2)} = a_{ij}^{(1)} - m_{i1}a_{1j}^{(1)}, i = 2, \cdots, n, j = 2, \cdots, n+1$, 则得矩阵

$$\begin{pmatrix} a_{11}^{(1)} & a_{12}^{(1)} & \cdots & a_{1n}^{(1)} \\ 0 & a_{22}^{(2)} & \cdots & a_{2n}^{(2)} \\ \vdots & \vdots & & \vdots \\ 0 & a_{n2}^{(2)} & \cdots & a_{nn}^{(2)} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} a_{1,n+1}^{(1)} \\ a_{2,n+1}^{(2)} \\ \vdots \\ a_{n,n+1}^{(2)} \end{pmatrix}$$

即 $A^{(2)}x = b^{(2)}$

以此类推, 对于第 k 次消元, 若 $a_{kk}^{(k)} \neq 0$, 令

$$m_{ik} = \frac{a_{ik}^{(k)}}{a_{kk}^{(k)}}$$

令 $a_{ik}^{(k+1)} = a_{ij}^{(k)} - m_{ik}a_{kj}^{(k)}, i = k+1, \dots, n; j = k+1, \dots, n+1$ 则

$$A^{(k)} = \begin{pmatrix} a_{11}^{(1)} & a_{12}^{(1)} & \cdots & \cdots & \cdots & a_{1n}^{(1)} \\ & a_{22}^{(2)} & \cdots & \cdots & \cdots & a_{2n}^{(2)} \\ & & \ddots & \vdots & \vdots & \vdots \\ & & & a_{kk}^{(k)} & \cdots & a_{kn}^{(k)} \\ & & & \vdots & & \vdots \\ & & & a_{nk}^{(k)} & \cdots & a_{nn}^{(k)} \end{pmatrix}, b^{(k)} = \begin{pmatrix} a_{1,n+1}^{(1)} \\ a_{2,n+1}^{(2)} \\ \vdots \\ a_{k-1,n+1}^{(k-1)} \\ a_{k,n+1}^{(k)} \\ \vdots \\ a_{n,n+1}^{(k)} \end{pmatrix}$$

即 $A^{(k)}x = b^{(k)}$

共进行 $n-1$ 步, 即可得到三角方程组

$$\begin{pmatrix} a_{11}^{(1)} & a_{12}^{(1)} & \cdots & a_{1n}^{(1)} \\ & a_{22}^{(2)} & \cdots & a_{2n}^{(2)} \\ & & \ddots & \vdots \\ & & & a_{nn}^{(n)} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} a_{1,n+1}^{(1)} \\ a_{2,n+1}^{(2)} \\ \vdots \\ a_{n,n+1}^{(n)} \end{pmatrix}$$

得

$$A^{(n)}x = b^{(n)}$$

利用回代过程, 求解得

$$x_n = \frac{a_{n,n+1}^{(n)}}{a_{nn}^{(n)}}$$

$$x_k = \frac{a_{k,n+1}^{(k)} - \sum_{j=k+1}^n a_{kj}^{(k)}x_j}{a_{kk}^{(k)}}, k = n-1, n-2, \dots, 2, 1$$

称 $a_{kk}^{(k)}$ 为主元素 (或主元).

注: 若第一个主元素为 0, 则可把第一列得非零元素调整至第一行, 此时 Gauss 消去法仍然可用.

一般地, 关于顺序 Gauss 消去法可执行的前提, 有如下引理:

引理 5.1.1. 给定线性方程组 $Ax = b$, 按顺序 Gauss 消去法所形成的各主元素 $a_{kk}^{(k)} (k = 1, 2, \dots, n)$ 均不为零, 从而 Gauss 消去法可顺利执行的

充要条件是 n 阶方阵 A 的所有顺序主子式都不为零, 即

$$D_1 = a_{11} \neq 0, \dots, D_k = \begin{vmatrix} a_{11} & \cdots & a_{1k} \\ \vdots & \ddots & \vdots \\ a_{k1} & \cdots & a_{kk} \end{vmatrix} \neq 0, k = 2, 3, \dots, n$$

当线性方程组的系数矩阵为对称正定或严格对角占优时, 按顺序 Gauss 消去法计算是稳定的.

定理 5.1.2. 若矩阵 A 非奇异, 即 A^{-1} 存在, 则可通过逐次消元以及行交换, 将方程组转化为三角形方程组并求出唯一解.

定理 5.1.3. 若 A 的所有顺序主子式均不为 0, 则 Gauss 消元无需换行即可进行到底并求出唯一解.

5.1.3 Gauss 主元素消去法

考虑一个方程组:

$$\begin{pmatrix} 0.001 & 2.000 & 3.000 \\ -1.000 & 3.712 & 4.623 \\ -2.000 & 1.072 & 5.643 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 1.000 \\ 2.000 \\ 3.000 \end{pmatrix}$$

可以发现, 该方程组在使用四位浮点数运算, 得到的计算解 (近似解) 为 $(-0.4000, -0.09980, 0.4000)^T$, 显然与精确解不符. 出现该问题的原因是, 在消元计算时, 使用了小主元 0.001, 从顺序 Gauss 消去法的角度看, 不为零的主元可以使用, 但从误差的角度看, 使用小主元 (做除数) 会导致数量级增大, 从而引起舍入误差. 因此, 我们需要通过矩阵变换, 以获得合适的主元.

完全主元素法

考虑在每一次消元过程中, 在增广矩阵中选择绝对值最大的一项作为主元素, 并将其移动至对角主元位置, 然后进行消元. 经过一系列的消元, 可将方程组化为

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ & a_{22} & \cdots & a_{2n} \\ & & \ddots & \vdots \\ & & & a_{nn} \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix}$$

其中, y_1, y_2, \dots, y_n 为未知数 x_1, x_2, \dots, x_n 经过选择主元调换后的次序. 这里未知数次序的调换主要是由于列变换导致的 (行变换不会交换未知数的位置)

关于完全主元素法, 有如下特点:

1. 在选择主元时需要花费较多机器时间;
2. 需要实时记录 x 的顺序变化;

列主元消去法

在选择主元时仅考虑按列选取, 然后按行使之变到主元位置上, 再进行消元计算.

为了简单起见, 将通过一个例子实际演示用列主元 Gauss 消去法求解过程.

例 5.1.1. 使用列主元消去法求解方程组

$$\begin{cases} x_1 + x_2 + x_3 = 6 \\ 2x_1 + 2x_2 - x_3 = 3 \\ 3x_1 + x_3 = 6 \end{cases}$$

解. 线性方程组增广矩阵为

$$\begin{pmatrix} 1 & 1 & 1 & 6 \\ 2 & 2 & -1 & 3 \\ 3 & 0 & 1 & 6 \end{pmatrix}$$

对第一列而言, 主元为 3, 故将其作为第一行, 作行变换, 得

$$\begin{pmatrix} 3 & 0 & 1 & 6 \\ 2 & 2 & -1 & 3 \\ 1 & 1 & 1 & 6 \end{pmatrix} \rightarrow \begin{pmatrix} 3 & 0 & 1 & 6 \\ 0 & 2 & -5/3 & -1 \\ 0 & 1 & 2/3 & 4 \end{pmatrix}$$

对第二列而言, 2 为主元, 故保持不变, 得

$$\begin{pmatrix} 3 & 0 & 1 & 6 \\ 0 & 2 & -5/3 & -1 \\ 0 & 1 & 2/3 & 4 \end{pmatrix} \rightarrow \begin{pmatrix} 3 & 0 & 1 & 6 \\ 0 & 2 & -5/3 & -1 \\ 0 & 0 & 9/6 & 9/2 \end{pmatrix}$$

利用回代法, 得方程组解为 $(1, 2, 3)^T$

□

上述解答用程序语言可以实现如下:

```
1  # 使用列主元Gauss消去法求解方程组 Exercise5-1.py
2
3  import numpy as np
4
5  #使用列主元Gauss消去法求解方程组
6  def gauss_elimination(A, b):
7      n = A.shape[0]
8      # 消元
9      for j in range(n):
10         # 选择最大值作为主元
11         max_idx = j + np.argmax(np.abs(A[j:, j]))
12         if j != max_idx:
13             A[[j, max_idx]] = A[[max_idx, j]]
14             b[[j, max_idx]] = b[[max_idx, j]]
15         # 计算主元下面的元素
16         pivot = A[j, j]
17         for i in range(j+1, n):
18             factor = A[i, j] / pivot
19             A[i, :] -= factor * A[j, :]
20             b[i] -= factor * b[j]
21         # 回代
22         x = np.zeros(n)
23         for j in range(n-1, -1, -1):
24             x[j] = (b[j] - np.dot(A[j, j+1:], x[j+1:])) / A[j, j]
25         return x
26
27  # 计算方程组
28  A = np.array([[1,1,1],[2,2,-1],[3,0,1]], dtype=float)
29  b = np.array([6,3,6], dtype=float)
30  x = gauss_elimination(A, b)
31  print("x=", x)
```


程序计算结果与我们计算结果是一致的.

Gauss-Jordan 消去法

基本思想: 同时消去对角线上方和下方的元素.

特点:

1. 消元和回代同时进行;
2. 乘除法次数比 Gauss 消去法大

下面以一个线性方程组为例, 介绍如何实现 Gauss-Jordan 消去法:

例 5.1.2. 使用 *Gauss-Jordan* 消去法求解方程组

$$\begin{cases} x_1 + x_2 + x_3 = 6 \\ 2x_1 + x_2 - x_3 = 1 \\ 3x_1 + x_3 = 6 \end{cases}$$

解. 对于 Gauss-Jordan 消去法, 我们不考虑主元. 增广矩阵为

$$\begin{pmatrix} 1 & 1 & 1 & 6 \\ 2 & 1 & -1 & 1 \\ 3 & 0 & 1 & 6 \end{pmatrix}$$

首先进行第一次消元, 得到结果为

$$\begin{pmatrix} 1 & 1 & 1 & 6 \\ 2 & 1 & -1 & 1 \\ 3 & 0 & 1 & 6 \end{pmatrix} \rightarrow \begin{pmatrix} 1 & 1 & 1 & 6 \\ 0 & -1 & -3 & -11 \\ 0 & -3 & -2 & -12 \end{pmatrix}$$

对第二行进行消元, 对于 Gauss-Jordan 消去法, 我们需要同时考虑对角线上下的元素. 因此, 消元后结果如下:

$$\begin{pmatrix} 1 & 1 & 1 & 6 \\ 0 & -1 & -3 & -11 \\ 0 & -3 & -2 & -12 \end{pmatrix} \rightarrow \begin{pmatrix} 1 & 0 & -2 & -5 \\ 0 & 1 & 3 & 11 \\ 0 & 0 & 7 & 21 \end{pmatrix}$$

可以看到, 经过第二步消元后, 第二列除主元素外所有元素都为 0. 同理, 第三步消元得到结果为

$$\begin{pmatrix} 1 & 0 & -2 & -5 \\ 0 & 1 & 3 & 11 \\ 0 & 0 & 7 & 21 \end{pmatrix} \rightarrow \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 1 & 3 \end{pmatrix}$$

因此解得方程组解为 $(1, 2, 3)^T$

□

列主元 Gauss-Jordan 消去法

与列主元 Gauss 消去法类似, 列主元 Gauss-Jordan 消去法步骤为:

第一步, 选择第一列的主元, 并将其换至第一行, 将第一个方程的系数变为 1, 同时从其余 $n-1$ 个方程中消去 x_1 ;

第二步, 再第二列后 $n-1$ 个元素中选择主元, 将第二个方程 x_2 系数变为 1, 并从其他 $n-1$ 个方程中消去 x_2

...

第 k 步, 在第 k 列后 $n-k$ 个元素中选主元并换行, 将第 k 个方程的 x_k 系数变为 1, 并从其他 $n-1$ 个方程中消去变量 x_k

消元公式为

$$\begin{cases} a_{kj}^{(k)} = \frac{a_{kj}^{(k-1)}}{a_{kk}^{(k-1)}}, j = k, k+1, \dots, n+1 \\ a_{ij}^{(k)} = a_{ij}^{(k-1)} - a_{ik}^{(k-1)} a_{kj}^{(k)} \\ j = k, k+1, \dots, n+1 \\ i = 1, 2, \dots, k-1, k+1, \dots, n \end{cases}$$

对 $k = 1, 2, \dots$ 进行上述步骤至第 n 步后, 方程组可变为

$$\begin{cases} x_1 = a_{1,n+1}^{(n)} \\ x_2 = a_{2,n+1}^{(n)} \\ \vdots \\ x_n = a_{n,n+1}^{(n)} \end{cases}$$

即为所求的解.

Gauss-Jordan 消去法应用

对于线性方程组系, 如

$$AX = b_1, AX = b_2, \cdots, AX = b_m$$

其中,

$$A = \begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{pmatrix}, X = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}, b_i = \begin{pmatrix} a_{1,n+1}^{(i)} \\ a_{2,n+1}^{(i)} \\ \vdots \\ a_{n,n+1}^{(i)} \end{pmatrix}, i = 1, 2, \cdots, n$$

因此上述方程组系可写作

$$AX = B = (b_1, \cdots, b_n)$$

其解为

$$X = A^{-1}B$$

特殊的, 设 $A = (a_{ij})_{n \times n}$ 是非奇异矩阵, $|A| \neq 0$, 令

$$X = A^{-1} = (x_{ij})_{n \times n}$$

由于 $AA^{-1} = AX = I$, 因此求解 X 的过程等价于当 $m = n, B = I$ 的方程组求解, 此时得到 $X = A^{-1}$

例 5.1.3. 用 Gauss-Jordan 消去法求解方程组 $Ax = b$, 并求出 A^{-1} , 其中,

$$A = \begin{pmatrix} 1 & 3 & 1 \\ 1 & 2 & 4 \\ 5 & 1 & 2 \end{pmatrix}, b = \begin{pmatrix} 10 \\ 17 \\ 13 \end{pmatrix}$$

解. 将系数矩阵, 单位矩阵和 b 组成增广矩阵, 并对其进行 Gauss-Jordan 消元过程, 可得过程为

$$\begin{pmatrix} 1 & 3 & 1 & 1 & 10 \\ 1 & 2 & 4 & 1 & 17 \\ 5 & 1 & 2 & 1 & 13 \end{pmatrix} \rightarrow \begin{pmatrix} 1 & 3 & 1 & 1 & 10 \\ 0 & -1 & 3 & -1 & 7 \\ 0 & -14 & -3 & -5 & -37 \end{pmatrix} \\ \rightarrow \begin{pmatrix} 1 & 10 & -2 & 3 & 31 \\ 1 & -3 & 1 & -1 & 0 & -7 \\ -45 & 9 & -14 & 1 & -135 \end{pmatrix} \rightarrow \begin{pmatrix} 1 & 0 & -0.11 & 0.22 & 1 \\ 1 & 0.4 & -0.07 & -0.07 & 2 \\ 1 & -0.2 & 0.31 & -0.02 & 3 \end{pmatrix}$$

故方程组的解为 $(1, 2, 3)^T$, 系数矩阵的逆为

$$A^{-1} = \begin{pmatrix} 0 & -0.11 & 0.22 \\ 0.4 & -0.07 & -0.07 \\ -0.2 & 0.31 & -0.02 \end{pmatrix}$$

□

5.2 解三对角方程组的追赶法

求解线性方程组

$$\begin{pmatrix} b_1 & c_1 & & & \\ a_2 & b_2 & c_2 & & \\ & \ddots & \ddots & \ddots & \\ & & a_{n-1} & b_{n-1} & c_{n-1} \\ & & & a_n & b_n \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n-1} \\ x_n \end{pmatrix} = \begin{pmatrix} d_1 \\ d_2 \\ \vdots \\ d_{n-1} \\ d_n \end{pmatrix}$$

且满足:

$$\begin{cases} |b_1| > |c_1| \\ |b_i| \geq |a_i| + |c_i|, i = 2, \dots, n-1 \\ |b_n| > |a_n| \end{cases}$$

消元公式为

$$\begin{cases} r_1 = \frac{c_1}{b_1}, y_1 = \frac{d_1}{b_1} \\ r_k = \frac{c_k}{b_k - r_{k-1}a_k}, \quad k = 2, 3, \dots, n-1 \\ y_k = \frac{d_k - y_{k-1}a_k}{b_k - r_{k-1}a_k}, \quad k = 2, 3, \dots, n \end{cases}$$

回代公式

$$\begin{cases} x_n = y_n \\ x_k = y_k - r_k x_{k+1}, \quad k = n-1, n-2, \dots, 1 \end{cases}$$

5.3 矩阵的三角分解法

5.3.1 Gauss 消元法矩阵形式

对于 Gauss 消元法, 每一步消元过程相当于左乘下三角矩阵 L_k

记

$$A^{(2)} = L_1^{-1}A^{(1)}, b^{(2)} = L_1^{-1}b^{(1)}$$

其中,

$$L_1^{-1} = \begin{pmatrix} 1 & & & & \\ -l_{21} & 1 & & & \\ -l_{31} & 0 & 1 & & \\ \vdots & \vdots & \vdots & \ddots & \\ -l_{n1} & 0 & 0 & \cdots & 1 \end{pmatrix}, l_{i1} = \frac{a_{i1}^{(1)}}{a_{11}^{(1)}}, i = 2, 3, \cdots, n$$

同理, 有

$$A^{(3)} = L_2^{-1}A^{(2)} = L_2^{-1}L_1^{-1}A^{(1)}, b^{(3)} = L_2^{-1}L_1^{-1}b^{(1)}$$

其中,

$$L_2^{-1} = \begin{pmatrix} 1 & & & & \\ 0 & 1 & & & \\ 0 & -l_{32} & 1 & & \\ \vdots & \vdots & \vdots & \ddots & \\ 0 & -l_{n2} & 0 & \cdots & 1 \end{pmatrix}, l_{i2} = \frac{a_{i2}^{(2)}}{a_{22}^{(2)}}, i = 3, 4, \cdots, n$$

一般地, 对 $i = 1, 2, \cdots, n-1$, 令

$$L_i^{-1} = \begin{pmatrix} 1 & & & & & \\ 0 & 1 & & & & \\ \vdots & & 1 & & & \\ 0 & \cdots & -l_{i+1,i} & 1 & & \\ \vdots & & \vdots & \vdots & \ddots & \\ 0 & & -l_{ni} & & & 1 \end{pmatrix}, L_i = \begin{pmatrix} 1 & & & & & \\ 0 & 1 & & & & \\ \vdots & & \ddots & & & \\ \vdots & & & 1 & & \\ 0 & & & l_{i+1,i} & 1 & \\ \vdots & & & \vdots & & \ddots \\ 0 & & & l_{ni} & & & 1 \end{pmatrix}$$

$$l_{ki} = \frac{a_{ki}^{(i)}}{a_{ii}^{(i)}}, k = i+1, \cdots, n$$

则

$$A^{(n)} = L_{n-1}^{-1}L_{n-2}^{-1} \cdots L_1^{-1}A^{(1)}$$

$$b^{(n)} = L_{n-1}^{-1}L_{n-2}^{-1} \cdots L_1^{-1}b^{(1)}$$

于是有

$$A^{(1)} = L_1 L_2 \cdots L_{n-1} A^{(n)} = L A^{(n)} = LU$$

将上式分解方式, 称为矩阵 A 的 LU 分解.

5.3.2 矩阵三角分解的定义

定义 5.3.1. 设 A 是 $n \times n$ 实矩阵, 如果存在下三角矩阵 L 与上三角矩阵 U , 使得 $A = LU$, 则称为矩阵 A 的三角分解, 若存在单位下三角矩阵 L , 对角矩阵 D 以及单位上三角矩阵 R , 使得 $A = LDR$, 则称为矩阵 A 的 LDR 分解.

若 L 为单位下三角矩阵而 U 是一般上三角矩阵, 则称该分解为 *Doolittle* 分解; 若 L 为一般下三角矩阵而 U 是单位上三角矩阵, 则称该分解为 *Crout* 分解.

5.3.3 矩阵三角分解的存在性

定理 5.3.1. 设 A 为 $n \times n$ 实矩阵, 若求解 $AX = b$ 用顺序 *Gauss* 消去法能够完成, 即 $a_{kk}^{(k)} \neq 0, k = 1, 2, \cdots, n$, 则矩阵 A 可分解为单位下三角矩阵 L 与上三角矩阵 U 的乘积, 即

$$A = LU$$

且这种分解唯一.

若 A 已经实现了 LU 分解, 则将方程组 $Ax = b$ 转化为 $(LU)x = b$, 从而计算得

$$Ly = b$$

$$Ux = y$$

若直接从矩阵 A 的元素得到计算 L, U 的递推公式, 而不需要中间步骤, 则称这种分解方法为直接三角分解法

直接三角分解法

例 5.3.1. 使用三角分解法解方程组

$$\begin{cases} x_1 + x_2 + x_3 = 6 \\ 4x_2 - x_3 = 5 \\ 2x_1 - 2x_2 + x_3 = 1 \end{cases}$$

解. 系数矩阵为

$$A = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 4 & -1 \\ 2 & -2 & 1 \end{pmatrix}$$

计算消元因子, 有

$$m_{21} = 0/1 = 0$$

$$m_{31} = 2/1 = 2$$

得行变换后矩阵为

$$A \rightarrow \begin{pmatrix} 1 & 1 & 1 \\ 0 & 4 & -1 \\ 0 & -4 & -1 \end{pmatrix}$$

类似地, 消元因子

$$m_{32} = -4/4 = -1$$

矩阵变换

$$\begin{pmatrix} 1 & 1 & 1 \\ 0 & 4 & -1 \\ 0 & 0 & -2 \end{pmatrix}$$

因此, 由 Gauss 消去法得矩阵分解

$$A = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 2 & -1 & 1 \end{pmatrix} \begin{pmatrix} 1 & 1 & 1 \\ 0 & 4 & -1 \\ 0 & 0 & -2 \end{pmatrix}$$

解方程组

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 2 & -1 & 1 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} = \begin{pmatrix} 6 \\ 5 \\ 1 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 1 & 1 \\ 0 & 4 & -1 \\ 0 & 0 & -2 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix}$$

解得

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$$

□

Doolittle 三角分解法

对于矩阵三角分解, 也可以使用待定系数法, 称为 Doolittle 分解法.

例 5.3.2. 利用 Doolittle 三角分解法重解上题

解. 待定系数法, 设

$$L = \begin{pmatrix} 1 & 0 & 0 \\ l_{21} & 1 & 0 \\ l_{31} & l_{32} & 1 \end{pmatrix}, U = \begin{pmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{pmatrix}$$

利用矩阵分解表达式, 有

$$\begin{pmatrix} 1 & 1 & 1 \\ 0 & 4 & -1 \\ 2 & -2 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ l_{21} & 1 & 0 \\ l_{31} & l_{32} & 1 \end{pmatrix} \begin{pmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{pmatrix}$$

得

$$u_{11} = 1, u_{12} = 1, u_{13} = 1$$

$$l_{21} = 0, u_{22} = 4, u_{23} = -1$$

$$l_{31} = 2, l_{32} = -1, u_{33} = -2$$

故将矩阵分解为

$$A = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 2 & -1 & 1 \end{pmatrix} \begin{pmatrix} 1 & 1 & 1 \\ 0 & 4 & -1 \\ 0 & 0 & -2 \end{pmatrix}$$

□

例 5.3.3. 用矩阵三角分解法求解方程组

$$\begin{pmatrix} 1 & 2 & 3 \\ 2 & 5 & 2 \\ 3 & 1 & 5 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 14 \\ 18 \\ 20 \end{pmatrix}$$

解. 待定系数法,

$$\begin{pmatrix} 1 & 2 & 3 \\ 2 & 5 & 2 \\ 3 & 1 & 5 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ l_{21} & 1 & 0 \\ l_{31} & l_{32} & 1 \end{pmatrix} \begin{pmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{pmatrix}$$

解得

$$\begin{aligned} u_{11} &= 1, u_{12} = 2, u_{13} = 3 \\ l_{21} &= 2, u_{22} = 1, u_{23} = -4 \\ l_{31} &= 3, l_{32} = -5, u_{33} = -24 \end{aligned}$$

即

$$L = \begin{pmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 3 & -5 & 1 \end{pmatrix}, U = \begin{pmatrix} 1 & 2 & 3 \\ 0 & 1 & -4 \\ 0 & 0 & -24 \end{pmatrix}$$

解方程组

$$Ly = b$$

$$Ux = y$$

可得

$$x = (1, 2, 3)^T$$

□

紧凑格式的 Doolittle 三角分解法

考虑增广矩阵 $(A; b)$, 将消元因子作为矩阵变换过程中的元素, 从而在变换过程中一次性得到 L, U, y , 将这种过程称为紧凑格式的 Doolittle 三角分解法.

例 5.3.4. 使用紧凑格式的 Doolittle 三角分解法重解上题

解. 考虑增广矩阵

$$\begin{pmatrix} 1 & 2 & 3 & 14 \\ 2 & 5 & 2 & 18 \\ 3 & 1 & 5 & 20 \end{pmatrix}$$

消元因子 $m_{21} = 2, m_{31} = 3$, 变换得

$$\begin{pmatrix} 1 & 2 & 3 & 14 \\ 2 & 1 & -4 & -10 \\ 3 & -5 & -4 & -22 \end{pmatrix}$$

类似地, 消元因子 $m_{32} = -5$, 故得变换后矩阵为

$$\begin{pmatrix} 1 & 2 & 3 & 14 \\ 2 & 1 & -4 & -10 \\ 3 & -5 & -24 & -72 \end{pmatrix}$$

因此直接解得

$$L = \begin{pmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 3 & -5 & 1 \end{pmatrix}, U = \begin{pmatrix} 1 & 2 & 3 \\ 0 & 1 & -4 \\ 0 & 0 & -24 \end{pmatrix}, y = \begin{pmatrix} 14 \\ -10 \\ -72 \end{pmatrix}$$

解方程组 $Ux = y$, 可得

$$x = (1, 2, 3)^T$$

□

上述例题可使用下述程序代码求解:

```
1 # 使用紧凑格式的Doolittle三角分解法求解线性方程组 Exercise5-2.py
2
```

```
3 import numpy as np
4
5 def doolittle_triangular_decomposition(A):
6     """
7     Doolittle三角分解法
8     """
9     n = A.shape[0]
10    L = np.zeros((n, n))
11    U = np.zeros((n, n))
12    for i in range(n):
13        L[i, i] = 1
14        for j in range(i, n):
15            U[i, j] = A[i, j] - sum(L[i, k] * U[k, j] for k in
16                                   range(i))
17            if i != j:
18                L[j, i] = (A[j, i] - sum(L[j, k] * U[k, i] for k
19                                       in range(i))) / U[i, i]
20    return L, U
21
22 def forward_substitution(L, b):
23     """
24     前向替代
25     """
26     n = L.shape[0]
27     x = np.zeros(n)
28     for i in range(n):
29         x[i] = b[i] / L[i, i]
30         for j in range(i):
31             x[i] -= x[j] * L[i, j] / L[i, i]
32    return x
33
34 def backward_substitution(U, b):
35     """
```

```

34     回代
35     """
36     n = U.shape[0]
37     x = np.zeros(n)
38     for i in reversed(range(n)):
39         x[i] = (b[i] - sum(U[i, j] * x[j] for j in range(i+1, n
40             ))) / U[i, i]
41     return x
42
43 def doolittle_solver(A, b):
44     """
45     使用Doolittle三角分解法求解线性方程组
46     """
47     L, U = doolittle_triangular_decomposition(A)
48     y = forward_substitution(L, b)
49     x = backward_substitution(U, y)
50     return x
51
52 A = np.array([[1,2,3],[2,5,2],[3,1,5]], dtype=float)
53 b = np.array([14,18,20], dtype=float)
54 x = doolittle_solver(A, b)
55 print("x=", x)

```

类似地, 还有紧凑格式的列主元 Doolittle 三角分解法, 不做详细说明. 需要特别注意的是, 在变换矩阵过程中所得到的消元因子, 不参与行变换的计算, 但需要参与行交换.

5.4 Gauss 消去法变形

5.4.1 矩阵 LDR 分解

定理 5.4.1. 若 n 阶矩阵 A 的所有顺序主子式均不为 0, 则矩阵 A 存在唯一分解式 $A = LDR$, 其中 L, R 分别为 n 阶单位下三角矩阵和单位上三角矩阵, D 为对角元素不为 0 的 n 阶对角矩阵. 上述分解方法称为 A 的

LDR 分解.

推论 5.4.2. 设 A 为 n 阶对称矩阵, 且 A 的所有顺序主子式均不为 0, 则 A 可唯一分解为

$$A = LDL^T$$

其中, L 为单位下三角矩阵, D 为对角矩阵

特殊地, 考虑对称正定矩阵, 其定义为

定义 5.4.1. 设 $A \in R^{n \times n}$, 若 A 满足

1.

$$A^T = A$$

2. $\forall x \in R^n$, 且 $x \neq 0$, 有

$$x^T Ax > 0$$

则称 A 为对称正定矩阵.

关于对称正定矩阵, 有如下性质

1. A 是非奇异矩阵, 且 A^{-1} 是对称正定矩阵;
2. A 地顺序主子式均大于 0, 即

$$\det(A_k) > 0, k = 1, 2, \dots, n$$

3. A 的特征值

$$\lambda_i(A) > 0, i = 1, 2, \dots, n$$

对于对称正定矩阵, 矩阵有特殊的分解, 即平方根法

5.4.2 平方根法

定理 5.4.3. 若 A 为对称正定矩阵, 则存在一个实的非奇异下三角矩阵, 使得 $A = LL^T$, 且当限定的对角元素为正时, 这种分解是唯一的, 称为矩阵 A 的 *Cholesky* 分解.

对于平方根法, 其数值是稳定的, 即在计算过程中不需要选择主元, 但在计算过程中会引入平方根计算, 因此计算难度会较大 (但计算量较一般的 LU 分解小)

5.4.3 改进平方根法

为避免开方运算, 我们改进平方根法, 即考虑将矩阵 A 分解为 LDL^T , 其中 L 为单位下三角矩阵, D 为对角矩阵.

通过分解, 将方程组转换为 $LDL^T x = b$, 求解下列方程组

$$Ly = bL^T x = D^{-1}y$$

下面将通过一个例子实际演示使用改进的平方根法求解方程组

例 5.4.1. 用改进平方根法求解方程组

$$\begin{pmatrix} 1 & 2 & 1 \\ 2 & 3 & 1 \\ 1 & 1 & 2 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 4 \\ 6 \\ 4 \end{pmatrix}$$

解. 系数矩阵为对称矩阵, 则可以将其分解为

$$\begin{pmatrix} 1 & 2 & 1 \\ 2 & 3 & 1 \\ 1 & 1 & 2 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ l_{21} & 1 & 0 \\ l_{31} & l_{32} & 1 \end{pmatrix} \begin{pmatrix} d_1 & & \\ & d_2 & \\ & & d_3 \end{pmatrix} \begin{pmatrix} 1 & l_{21} & l_{31} \\ 0 & 1 & l_{32} \\ 0 & 0 & 1 \end{pmatrix}$$

解得

$$L = \begin{pmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 1 & 1 & 1 \end{pmatrix}, D = \begin{pmatrix} 1 & & \\ & -1 & \\ & & 2 \end{pmatrix}$$

解方程

$$Ly = b$$

得

$$y = (4, -2, 2)^T$$

解方程

$$L^T x = D^{-1}y$$

其中

$$D^{-1} = \begin{pmatrix} 1 & & \\ & -1 & \\ & & \frac{1}{2} \end{pmatrix}$$

得

$$x = (1, 1, 1)^T$$

□

使用改进平方根法有如下优点:

1. 计算量小, 是目前计算对称正定矩阵方程组得有效方法;
2. 计算简单, 没有开方计算;
3. 精度较高;
4. 可以推广至非正定方程组的求解

5.5 线性方程组的性态和解的误差估计

在求解线性方程 $Ax = b$ 时, 系数 A 和 b 的误差对解 x 有影响.

当 A 精确时, 设 b 有误差 δb , 设得到的解为 $x + \delta x$, 即

$$A(x + \delta x) = b + \delta b$$

有

$$\delta x = A^{-1} \delta b$$

由范数的性质可知,

$$\|\delta x\| \leq \|A^{-1}\| \|\delta b\|$$

又因为

$$\|b\| = \|Ax\| \leq \|A\| \|x\| \Rightarrow \frac{1}{\|x\|} \leq \frac{\|A\|}{\|b\|}$$

即

$$\frac{\|\delta x\|}{\|x\|} \leq \|A\| \|A^{-1}\| \frac{\|\delta b\|}{\|b\|}$$

同理, 设 b 精确, A 有误差 δA , 解为 $x + \delta x$, 即

$$(A + \delta A)(x + \delta x) = b$$

可得

$$\frac{\|\delta x\|}{\|x\|} \leq \frac{\|A\| \|A^{-1}\| \frac{\|\delta A\|}{\|A\|}}{1 - \|A\| \|A^{-1}\| \frac{\|\delta A\|}{\|A\|}}$$

可以发现, $\|A\| \|A^{-1}\|$ 是衡量误差放大的重要因素, 称为条件数, 记作 $\text{cond}(A)$.

定义 5.5.1. 设线性方程组的系数矩阵是非奇异的, 若 $\text{cond}(A)$ 越大, 则称这个方程组越病态, 反之, 若 $\text{cond}(A)$ 越小, 则称方程组越良态.

通常来说, 判断一个矩阵是否病态, 可采用如下方法:

1. 行列式很大或很小;
2. 元素之间相差较大数量级;
3. 主元消去过程中出现小主元;
4. 特征值相差较大数量级

Chapter 6

线性方程组的迭代解法

6.1 一般迭代法

将线性方程组 $Ax = b$ 改写为等价形式 $x = Bx + g$, 建立某一种迭代格式

$$x_{k+1} = BX_k + g$$

从而从初始值 x_0 出发, 得到序列 x_k

6.1.1 迭代格式构造

将矩阵 A 分裂为

$$A = Q - C, |Q| \neq 0$$

则

$$\begin{aligned} Ax = b &\Leftrightarrow (Q - C)x = b \\ &\Leftrightarrow (I - Q^{-1}C)x = Q^{-1}b \\ &\Leftrightarrow x = Bx + g \end{aligned}$$

其中,

$$\begin{aligned} B &= Q^{-1}C \\ g &= Q^{-1}b \end{aligned}$$

将上式改写为迭代过程

$$x_{k+1} = BX_k + g$$

称这种迭代过程为逐次逼近法, B 为迭代矩阵. 对于给定的初值 x_0 , 可得到向量序列

$$x_0, x_1, \dots, x_n, \dots$$

定义 6.1.1. 若

$$\lim_{n \rightarrow \infty} x_n = x^*$$

则称逐次逼近法收敛, 否则称逐次逼近法不收敛或发散

下面的定理保证了, 收敛值 x^* 为方程组的解

定理 6.1.1. 对于任意给定的初始向量 x_0 , 如果经过逐次逼近法产生的向量序列收敛于向量 x^* , 则 x^* 是方程组 $x = Bx + g$ 的解

证明.

$$\lim_{k \rightarrow \infty} x_{k+1} = \lim_{k \rightarrow \infty} (Bx_k + g)$$

由极限的收敛性可知,

$$x^* = Bx^* + g$$

□

6.1.2 迭代法收敛条件

引理 6.1.2. 当 $k \rightarrow \infty$ 时, $B^k \rightarrow 0$ 的充要条件是:

$$\rho(B) < 1$$

定理 6.1.3. 设线性方程组 $x = Bx + g$ 有唯一解, 那么逐次逼近法对任意初始向量 x_0 收敛的充要条件是迭代矩阵 B 的谱半径 $\rho(B) < 1$

证明.

$$\begin{cases} x^* = Bx^* + g \\ x_{k+1} = Bx_k + g \end{cases}$$

$$\Rightarrow x_{k+1} - x^* = B(x_k - x^*) = \dots = B^{k+1}(x_0 - x^*)$$

因此,

$$\lim_{k \rightarrow \infty} (x_{k+1} - x^*) = 0 \Leftrightarrow \lim_{k \rightarrow \infty} B^{k+1} = 0$$

由上述引理可知, 其充要条件为 $\rho(B) < 1$

□

事实上, 在实际验证收敛性的时候, 我们可以使用谱半径, 但更简单的, 可以使用下面的定理 (这个定理是充分条件而不是充要条件)

定理 6.1.4. 若逐次逼近法的迭代矩阵满足 $\|B\| < 1$, 则逐次逼近法收敛. 其中, $\|\cdot\|$ 为矩阵 B 的某种范数.

由于我们在计算范数的时候, 矩阵的 1-范数与 ∞ -范数都可以直接用矩阵元素计算, 因此更容易判断收敛性.

6.1.3 迭代法的误差估计

定理 6.1.5. 若存在一个矩阵范数使得 $\|B\| < 1$, 则迭代收敛, 且有误差估计

$$\begin{aligned} \|x^* - x_{k+1}\| &\leq \frac{\|B\|^{k+1}}{1 - \|B\|} \|x_1 - x_0\| \\ \|x^* - x_{k+1}\| &\leq \frac{\|B\|}{1 - \|B\|} \|x_{k+1} - x_k\| \end{aligned}$$

其中, 第一个式子用于判断误差表达式与收敛速度, 第二个式子用于判断终止条件 (即停机准则).

6.2 Jacobi 迭代法

设有 n 阶方程组

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2 \\ \vdots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n = b_n \end{cases}$$

若系数矩阵非奇异, 且 $a_{ii} \neq 0 (i = 1, 2, \dots, n)$, 将方程组改写为

$$\begin{cases} x_1 = \frac{1}{a_{11}}(b_1 - a_{12}x_2 - a_{13}x_3 - \dots - a_{1n}x_n) \\ x_2 = \frac{1}{a_{22}}(b_2 - a_{21}x_1 - a_{23}x_3 - \dots - a_{2n}x_n) \\ \vdots \\ x_n = \frac{1}{a_{nn}}(b_n - a_{n1}x_1 - a_{n2}x_2 - \dots - a_{n,n-1}x_{n-1}) \end{cases}$$

写成迭代格式

$$\begin{cases} x_1^{(k+1)} = \frac{1}{a_{11}}(b_1 - a_{12}x_2^{(k)} - a_{13}x_3^{(k)} - \dots - a_{1n}x_n^{(k)}) \\ x_2^{(k+1)} = \frac{1}{a_{22}}(b_2 - a_{21}x_1^{(k)} - a_{23}x_3^{(k)} - \dots - a_{2n}x_n^{(k)}) \\ \vdots \\ x_n^{(k+1)} = \frac{1}{a_{nn}}(b_n - a_{n1}x_1^{(k)} - a_{n2}x_2^{(k)} - \dots - a_{n,n-1}x_{n-1}^{(k)}) \end{cases}$$

也可以简单写作

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left(b_i - \sum_{\substack{j=1 \\ j \neq i}}^n a_{ij}x_j^{(k)} \right), i = 1, 2, \dots, n$$

写成矩阵形式,

$$\begin{aligned} Ax = b &\Leftrightarrow (D - L - U)x = b \\ &\Leftrightarrow Dx = (L + U)x + b \\ &\Leftrightarrow x = D^{-1}(L + U)x + D^{-1}b \end{aligned}$$

定义

$$B_J = D^{-1}(L + U), f = D^{-1}b$$

其中 B_J 为 Jacobi 迭代阵, 则迭代格式可以写作

$$x^{(k+1)} = B_J x^{(k)} + f$$

注意: 矩阵 D 是对角矩阵, L 和 U 分别为严格下三角矩阵和严格上三角矩阵, 且 L 和 U 的元素为 A 的相反数

不难验证, B_J 满足下面的形式

$$B_J = \begin{pmatrix} 0 & -\frac{a_{12}}{a_{11}} & -\frac{a_{13}}{a_{11}} & \cdots & -\frac{a_{1n}}{a_{11}} \\ -\frac{a_{21}}{a_{22}} & 0 & -\frac{a_{23}}{a_{22}} & \cdots & -\frac{a_{2n}}{a_{22}} \\ -\frac{a_{31}}{a_{33}} & -\frac{a_{32}}{a_{33}} & 0 & \cdots & -\frac{a_{3n}}{a_{33}} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ -\frac{a_{n1}}{a_{nn}} & -\frac{a_{n2}}{a_{nn}} & -\frac{a_{n3}}{a_{nn}} & \cdots & 0 \end{pmatrix}$$

1. 在迭代过程中, 矩阵 A 的元素不会改变, 故可以事先调整好 A 的值, 使得对角元素 $a_{ii} \neq 0$, 否则 A 不可逆;
2. 在计算的时候, 需要计算完 $x^{(k)}$ 之后才能继续计算 $x^{(k+1)}$, 因此需要两组向量存储. 从计算机的角度看, 占用了内存空间.

为了解决空间占用问题, 需要给出另一种迭代方法

6.3 Gauss-Seidel 迭代法

若使用下面的形式计算

$$\begin{aligned} x_1^{(k+1)} &= \frac{1}{a_{11}}(-a_{12}x_2^{(k)} - a_{13}x_3^{(k)} - a_{14}x_4^{(k)} - \cdots - a_{1n}x_n^{(k)} + b_1) \\ x_2^{(k+1)} &= \frac{1}{a_{22}}(-a_{21}x_1^{(k+1)} - a_{23}x_3^{(k)} - a_{24}x_4^{(k)} - \cdots - a_{2n}x_n^{(k)} + b_2) \\ x_3^{(k+1)} &= \frac{1}{a_{33}}(-a_{31}x_1^{(k+1)} - a_{32}x_2^{(k+1)} - a_{34}x_4^{(k)} - \cdots - a_{3n}x_n^{(k)} + b_3) \\ &\vdots \\ x_n^{(k+1)} &= \frac{1}{a_{nn}}(-a_{n1}x_1^{(k+1)} - a_{n2}x_2^{(k+1)} - a_{n3}x_3^{(k+1)} - \cdots - a_{n,n-1}x_{n-1}^{(k+1)} + b_n) \end{aligned}$$

用矩阵形式写作

$$\begin{aligned} x^{(k+1)} &= D^{-1}(Lx^{(k+1)} + Ux^{(k)}) + D^{-1}b \\ \Leftrightarrow (D - L)x^{(k+1)} &= Ux^{(k)} + b \\ \Leftrightarrow x^{(k+1)} &= (D - L)^{-1}Ux^{(k)} + (D - L)^{-1}b \end{aligned}$$

定义

$$B_{G-S} = (D - L)^{-1}U, g = (D - L)^{-1}b$$

称 B_{G-S} 为 Gauss-Seidel 迭代阵.

上式可以理解为将系数矩阵 A 作另一个分裂

$$A = (D - L) - U$$

此时有

$$\begin{aligned} Ax = b &\Leftrightarrow ((D - L) - U)x = b \\ &\Leftrightarrow (D - L)x = Ux + b \\ &\Leftrightarrow x = (D - L)^{-1}Ux + (D - L)^{-1}b \\ &\Leftrightarrow x_{k+1} = (D - L)^{-1}Ux_k + (D - L)^{-1}b \\ &\Leftrightarrow x_{k+1} = B_{G-S}x_k + g \end{aligned}$$

6.4 Jacobi 迭代法和 Gauss-Seidel 迭代法收敛性

定理 6.4.1. n 阶矩阵 A 是按行严格对角占优矩阵的充分必要条件是 $Jacobi$ 迭代法的迭代矩阵满足 $\|B_J\|_\infty < 1$

按行严格对角占优矩阵, 指的是对于矩阵 $A = (a_{ij})_{n \times n}$, 有

$$|a_{ii}| > \sum_{j=1, j \neq i}^n |a_{ij}|$$

定理 6.4.2. 如果 A 是严格对角占优矩阵, 则 $Jacobi$ 和 $Gauss-Seidel$ 迭代法都收敛

定理 6.4.3. 若 A 是不可约对角占优矩阵, 则 $Jacobi$ 迭代法与 $Gauss-Seidel$ 迭代法都收敛

定义 6.4.1 (可约矩阵与不可约矩阵). 设 $A = (a_{ij})_{n \times n}$, 当 $n \geq 2$ 时, 如果存在 n 阶置换阵 P , 使得

$$P^T A P = \begin{pmatrix} A_{11} & A_{12} \\ 0 & A_{22} \end{pmatrix}$$

其中 A_{11} 为 r 阶子矩阵, A_{22} 为 $n - r$ 阶子矩阵, 且 $1 \leq r \leq n$, 则称矩阵 A 为可约矩阵, 否则, 若不存在置换阵 P 使上式成立, 则称矩阵 A 为不可约矩阵.

定理 6.4.4. 若 A 使 n 阶正定矩阵, 则 *Gauss-Seidel* 迭代法收敛.

定理 6.4.5. 若 A 使有正对角元的 n 阶对称矩阵, 则 *Jacobi* 迭代法收敛的充要条件使 A 和 $2D - A$ 同为正定矩阵

对于 *Jacobi* 迭代, 其特征方程为

$$\begin{aligned} |\lambda I - B_J| = 0 &\Leftrightarrow |\lambda I - D^{-1}(L + U)| = 0 \\ &\Leftrightarrow |\lambda D - (L + U)| = 0 \\ &\Leftrightarrow \begin{vmatrix} \lambda a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & \lambda a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & \lambda a_{nn} \end{vmatrix} = 0 \end{aligned}$$

同理, 对于 *Gauss-Seidel* 迭代

$$\begin{aligned} |\lambda I - B_{G-S}| = 0 &\Leftrightarrow |\lambda I - (D - L)^{-1}U| = 0 \\ &\Leftrightarrow |\lambda(D - L) - U| = 0 \\ &\Leftrightarrow \begin{vmatrix} \lambda a_{11} & a_{12} & \cdots & a_{1n} \\ \lambda a_{21} & \lambda a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \lambda a_{n1} & \lambda a_{n2} & \cdots & \lambda a_{nn} \end{vmatrix} = 0 \end{aligned}$$

例 6.4.1. 分析线性方程组

$$\begin{cases} 2x_1 - x_2 + x_3 = 1 \\ x_1 + x_2 + x_3 = 1 \\ x_1 + x_2 - 2x_3 = 1 \end{cases}$$

使用 *Jacobi* 迭代和 *Gauss-Seidel* 迭代的收敛性

解. 对于 *Jacobi* 迭代, 其特征方程为

$$\begin{vmatrix} 2\lambda & -1 & 1 \\ 1 & \lambda & 1 \\ 1 & 1 & -2\lambda \end{vmatrix} = 0$$

解得其特征值为 $0, \pm \frac{\sqrt{5}}{2}i$, 因此谱半径大于 1, *Jacobi* 迭代法不收敛.

对于 Gauss-Seidel 迭代, 特征方程为

$$\begin{vmatrix} 2\lambda & -1 & 1 \\ \lambda & \lambda & 1 \\ \lambda & \lambda & -2\lambda \end{vmatrix} = 0$$

其特征值为 $0, -\frac{1}{2}$, 因此谱半径小于 1, Gauss-Seidel 迭代法收敛. \square

例 6.4.2. 分析线性方程组

$$\begin{cases} x_1 + x_2 = 1 \\ x_1 + 2x_2 - 2x_3 = 1 \\ -2x_2 + 5x_3 = 1 \end{cases}$$

的 Jacobi 迭代和 Gauss-Seidel 迭代收敛性

解. 系数矩阵 A 是正定矩阵, 因此 Gauss-Seidel 迭代法收敛. 又因为矩阵

$$2D - A = \begin{pmatrix} 1 & -1 & 0 \\ -1 & 2 & 2 \\ 0 & 2 & 5 \end{pmatrix}$$

也是正定矩阵, 因此 Jacobi 迭代法也收敛. \square

例 6.4.3. 分析线性方程组

$$\begin{cases} 10x_1 + x_3 - 5x_4 = -7 \\ x_1 + 8x_2 - 3x_3 = 11 \\ 3x_1 + 2x_2 - 8x_3 + x_4 = 23 \\ x_1 - 2x_2 + 2x_3 + 7x_4 = 17 \end{cases}$$

的 Jacobi 迭代和 Gauss-Seidel 迭代的收敛性

解. 方程系数矩阵为

$$A = \begin{pmatrix} 10 & 0 & 1 & -5 \\ 1 & 8 & -3 & 0 \\ 3 & 2 & -8 & 1 \\ 1 & -2 & 2 & 7 \end{pmatrix}$$

为严格对角占优矩阵, 因此 Jacobi 迭代和 Gauss-Seidel 迭代均收敛. \square

6.5 超松弛法

对于超松弛法而言, 每一次迭代分为如下两个步骤:

1. 迭代, 即

$$\tilde{x}_i^{(k+1)} = \frac{1}{a_{ii}} \left(b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i+1}^n a_{ij} x_j^{(k)} \right)$$

2. 加速, 令

$$x_i^{(k+1)} = (1 - \omega) x_i^{(k)} + \omega \tilde{x}_i^{(k+1)}, i = 1, 2, \dots, n$$

即

$$x_i^{(k+1)} = (1 - \omega) x_i^{(k)} + \frac{\omega}{a_{ii}} \left(b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i+1}^n a_{ij} x_j^{(k)} \right)$$

用矩阵形式表示为

$$\begin{aligned} x^{(k+1)} &= (1 - \omega) x^{(k)} + \omega D^{-1} (b - Lx^{(k+1)} + Ux^{(k)}) \\ \Leftrightarrow (D - \omega L)x^{(k+1)} &= [(1 - \omega)D + \omega U]x^{(k)} + \omega b \\ \Leftrightarrow x^{(k+1)} &= (D - \omega L)^{-1} [(1 - \omega)D + \omega U]x^{(k)} + \omega (D - \omega L)^{-1} b \end{aligned}$$

对于超松弛法, 其收敛的充要条件为 $\rho(L_\omega) < 1$, 其中,

$$L_\omega = (D - \omega L)^{-1} [(1 - \omega)D + \omega U]$$

定理 6.5.1. 超松弛法收敛的必要条件是松弛因子

$$0 < \omega < 2$$

定理 6.5.2. 给定线性方程组 $Ax = b$, 如果 A 是对称正定矩阵, 且 $0 < \omega < 2$, 则超松弛法收敛.

其中, 当 $\omega = 1$ 时, 超松弛法过渡为 Gauss-Seidel 迭代.

最后以课本的一道课后练习题结束

例 6.5.1. 设方程组

$$\begin{cases} x_1 - \frac{1}{4}x_2 - \frac{1}{4}x_4 = \frac{1}{2} \\ x_2 - \frac{1}{4}x_3 - \frac{1}{4}x_4 = \frac{1}{2} \\ -\frac{1}{4}x_1 - \frac{1}{4}x_2 + x_3 = \frac{1}{2} \\ -\frac{1}{4}x_1 - \frac{1}{4}x_2 + x_4 = \frac{1}{2} \end{cases}$$

求解该方程组对 *Jacobi* 迭代和 *Gauss-Seidel* 迭代法的迭代矩阵谱半径, 并判断其收敛性.

解. 由题意可知, 其系数矩阵为

$$A = \begin{pmatrix} 1 & 0 & -\frac{1}{4} & -\frac{1}{4} \\ 0 & 1 & -\frac{1}{4} & -\frac{1}{4} \\ -\frac{1}{4} & -\frac{1}{4} & 1 & 0 \\ -\frac{1}{4} & -\frac{1}{4} & 0 & 1 \end{pmatrix}$$

将其分解为对角矩阵 D , 严格下三角矩阵 L 和严格上三角矩阵 U , 使其满足 $A = D - L - U$, 得

$$D = \begin{pmatrix} 1 & & & \\ & 1 & & \\ & & 1 & \\ & & & 1 \end{pmatrix}$$

$$L = \begin{pmatrix} 0 & & & \\ 0 & 0 & & \\ \frac{1}{4} & \frac{1}{4} & 0 & \\ \frac{1}{4} & \frac{1}{4} & 0 & 0 \end{pmatrix}$$

$$U = \begin{pmatrix} 0 & 0 & \frac{1}{4} & \frac{1}{4} \\ & 0 & \frac{1}{4} & \frac{1}{4} \\ & & 0 & 0 \\ & & & 0 \end{pmatrix}$$

对于 *Jacobi* 迭代, 其迭代矩阵为

$$B_J = D^{-1}(L + U) = \begin{pmatrix} 0 & 0 & \frac{1}{4} & \frac{1}{4} \\ 0 & 0 & \frac{1}{4} & \frac{1}{4} \\ \frac{1}{4} & \frac{1}{4} & 0 & 0 \\ \frac{1}{4} & \frac{1}{4} & 0 & 0 \end{pmatrix}$$

其谱半径为特征值模得最大值, 计算可得

$$\rho(B_J) = 0.5$$

同理, 对于 Gauss-Seidel 迭代, 其迭代矩阵为

$$B_{G-S} = (D - L)^{-1}U = \begin{pmatrix} 0 & 0 & \frac{1}{4} & \frac{1}{4} \\ 0 & 0 & \frac{1}{4} & \frac{1}{4} \\ 0 & 0 & \frac{1}{8} & \frac{1}{8} \\ 0 & 0 & \frac{1}{8} & \frac{1}{8} \end{pmatrix}$$

谱半径

$$\rho(B_{G-S}) = 0.25$$

由于其谱半径均小于 1, 因此 Jacobi 迭代和 Gauss-Seidel 迭代法均收敛. \square

关于收敛性, 也可以使用系数矩阵的正定性来判断, 即由于系数矩阵 A 为正定矩阵, 则 Gauss-Seidel 迭代法收敛. 又由于系数矩阵为对称矩阵, 且 $2D - A$ 为正定矩阵, 因此 Jacobi 迭代法收敛.

可以发现, 使用两种方法得到的收敛性结果是一致的.

Chapter 7

常微分方程数值解法

7.1 引言

常微分方程的数值解法分为

1. 初值问题的数值解法
2. 边值问题的数值解法

7.1.1 初值问题的数值解法

一阶常微分方程初值问题一般形式

一阶常微分方程初值问题的一般形式为

$$\begin{cases} y'(x) = f(x, y(x)), a \leq x \leq b \\ y(a) = y_0 \end{cases}$$

求解方法主要是构造迭代格式, 即将连续的微分方程及初值条件离散化, 并使用不同的数值方法进行求解. 常见的构造方法为离散点函数值的集合 + 线性组合结构 = 近似公式.

进行微分方程数值解法时, 主要解决如下问题:

1. 如何将方程离散化, 并建立迭代公式;
2. 如何估计迭代公式局部截断误差与整体误差;
3. 如何保证迭代公式稳定性与收敛性.

相关定义

记

$$D = (x, y) | a \leq x \leq b, y(x) - \delta \leq y \leq y(x) + \delta$$

为 $f(x, y)$ 在 D 上对 y 满足 Lipschitz 条件指: 存在 $L > 0$, 使得对于任意的 $x \in [a, b], y_1, y_2 \in [y(x) - \delta, y(x) + \delta]$, 有

$$|f(x, y_1) - f(x, y_2)| \leq L|y_1 - y_2|$$

7.1.2 初值问题解存在唯一性

对于一阶常微分方程初值问题

$$\begin{cases} \frac{dy}{dx} = f(x, y), x \in [a, b] \\ y(a) = y_0 \end{cases}$$

只要 $f(x, y)$ 在 $[a, b] \times R^1$ 上连续, 且关于 y 满足 Lipschitz 条件, 即存在与 x, y 无关常数 L , 使得

$$|f(x, y_1) - f(x, y_2)| \leq L|y_1 - y_2|$$

对任意定义在 $[a, b]$ 上的 $y_1(x), y_2(x)$ 成立, 则上述微分方程初值问题存在唯一解.

求函数 $y(x)$ 在一系列节点处的近似值

$$y_i \approx y(x_i), i = 1, 2, \dots, n$$

的方法称为微分方程的数值解法, y_1, \dots, y_n 为微分方程的数值解.

称节点间距 $h_i = x_{i+1} - x_i, i = 0, 1, \dots, n-1$ 为步长, 通常采用等距节点, 即 $h_i = h = \text{常数}$

7.1.3 初值问题的离散化方法

按照某一递推公式, 按节点从左至右顺序依次求出 $y(x_i)$ 的近似值 $y_i, i = 1, 2, \dots, n$, 取 $y_0 = \eta$, 若计算 y_{i+1} 只用到前一步的值 y_i , 则称该方法为单步方法; 若用到前 r 步的值 $y_i, y_{i-1}, \dots, y_{i-r+1}$, 则称该方法为 r 步方法.

7.2 Euler 方法

7.2.1 Euler 公式

采用向前差商

$$y'(x_0) \approx \frac{y(x_1) - y(x_0)}{h}$$

近似导数, 则

$$y(x_1) \approx y(x_0) + hy'(x_0) = y_0 + hf(x_0, y_0)$$

并将该结果记为 y_1

类似地, 可以推导得到迭代格式为

$$y_{i+1} = y_i + hf(x_i, y_i), i = 0, 1, \dots, n-1$$

定义 7.2.1 (局部截断误差). 在假设 $y_i = y(x_i)$, 即第 i 步计算精确前提下, 考虑的截断误差

$$R_i = y(x_{i+1}) - y_{i+1}$$

称为局部截断误差

定义 7.2.2. 若某算法局部截断误差为 $O(h^{p+1})$, 则称该算法有 p 阶精度.

对于 Euler 法, 可以分析

$$\begin{aligned} R_i &= y(x_{i+1}) - y_{i+1} \\ &= \left[y(x_i) + hy'(x_i) + \frac{h^2}{2}y''(x_i) + O(h^3) \right] - [y_i + hf(x_i, y_i)] \\ &= \frac{h^2}{2}y''(x_i) + O(h^3) \\ &= O(h^2) \end{aligned}$$

称

$$\frac{h^2}{2}y''(x_i)$$

为 R_i 的主项. 由定义可知, Euler 法具有一阶精度.

例 7.2.1. 用 Euler 公式求解初值问题

$$\begin{cases} y' = -2xy^2, 0 \leq x \leq 1.2 \\ y(0) = 1 \end{cases}$$

要求取步长为 $h = 0.1$

解. 应用 Euler 公式

$$\begin{cases} y_{i+1} = y_i - 2hx_i y_i^2, i = 0, 1, \dots, 11 \\ y(0) = 1 \end{cases}$$

其中 $x_i = 0.1i$, 可直接进行迭代计算. 在此不再赘述. □

7.2.2 隐式 Euler 法 (向后 Euler 法)

使用向后差商近似导数, 即

$$y'(x_1) \approx \frac{y(x_1) - y(x_0)}{h}$$

有

$$y(x_1) \approx y_0 + hf(x_1, y(x_1))$$

以此可得迭代格式为

$$y_{i+1} = y_i + hf(x_{i+1}, y_{i+1}), i = 0, 1, \dots, n-1$$

注意到未知数 y_{i+1} 同时出现在等式两边, 因此在实际应用时不能直接得到, 称为隐式 Euler 公式.

与显式 Euler 公式类似, 隐式 Euler 公式的局部截断误差为

$$R_i = y(x_{i+1}) - y_{i+1} = -\frac{h^2}{2}y''(x_i) + O(h^3)$$

即具有 1 阶精度.

7.2.3 梯形公式

计算显式 Euler 公式和隐式 Euler 公式的平均值, 可得

$$y_{i+1} = y_i + \frac{h}{2}[f(x_i, y_i) + f(x_{i+1}, y_{i+1})], i = 0, 1, \dots, n-1$$

注意： 梯形公式的局部截断误差为

$$R_i = O(h^3)$$

即具有 2 阶精度. 但该公式仍然为隐式公式.

7.2.4 中点 Euler 公式

使用中心差商近似导数

$$y'(x_1) \approx \frac{y(x_2) - y(x_0)}{2h}$$

即

$$y(x_2) \approx y(x_0) + 2hf(x_1, y(x_1))$$

可得迭代格式为

$$y_{i+1} = y_{i-1} + 2hf(x_i, y_i), i = 1, 2, \dots, n-1$$

假设 $y_{i-1} = y(x_{i-1}), y_i = y(x_i)$, 则可以导出

$$R_i = O(h^3)$$

即中点 Euler 公式具有 2 阶精度

注意： 中点 Euler 公式是显式公式, 且精度相较显式 Euler 公式有所提高. 但中点 Euler 公式多一个初值, 可能在此影响精度.

为解决上述问题, 我们尝试改进梯形公式, 给出下面的方法

7.2.5 预测校正法

先使用显式 Euler 公式进行预测, 算出

$$\overline{y_{i+1}} = y_i + hf(x_i, y_i)$$

并将预测值 $\overline{y_{i+1}}$ 代入梯形公式右侧进行校正, 使得

$$y_{i+1} = y_i + \frac{h}{2} [f(x_i, y_i) + f(x_{i+1}, \overline{y_{i+1}})]$$

迭代格式为

$$y_{i+1} = y_i + \frac{h}{2} [f(x_i, y_i) + f(x_{i+1}, y_i + hf(x_i, y_i))], i = 0, 1, \dots, n-1$$

可以验证的是, 该算法具有 2 阶精度. 且是单步递推, 求解过程较简单.

注意: 实际计算中, 可先计算向前 Euler 公式, 再将计算结果代入向后 Euler 公式, 再将两个结果取算术平均值.

例 7.2.2. 用改进 Euler 公式重解上述初值问题, 步长 $h = 0.1$

解. 其迭代形式为

$$y_0 = 1$$

$$\begin{cases} y_{i+1}^{(p)} = y_i + 2hx_i y_i^2 \\ y_{i+1}^{(c)} = y_i - 2hx_{i+1} (y_{i+1}^{(p)})^2 \\ y_{i+1} = \frac{1}{2} (y_{i+1}^{(p)} + y_{i+1}^{(c)}) \end{cases}$$

□

7.3 Runge-Kutta 方法

对于改进的 Euler 法, 将其改写为

$$\begin{cases} y_{i+1} = y_i + h [\frac{1}{2}K_1 + \frac{1}{2}K_2] \\ K_1 = f(x_i, y_i) \\ K_2 = f(x_i + h, y_i + hK_1) \end{cases}$$

考察上式中 K_1, K_2 的系数和步长 h , 试着调整它们的值, 将改进 Euler 法进行推广:

$$\begin{cases} y_{i+1} = y_i + h[\lambda K_1 + \lambda_2 K_2] \\ K_1 = f(x_i, y_i) \\ K_2 = f(x_i + ph, y_i + phK_1) \end{cases}$$

首先考虑确定系数 λ_1, λ_2, p 使其具有 2 阶精度, 即

$$R_i = y(x_{i+1}) - y_{i+1} = O(h^3)$$

将 K_2 在 (x_i, y_i) 处作 Taylor 展开

$$\begin{aligned} K_2 &= f(x_i + ph, y_i + phK_1) \\ &= f(x_i, y_i) + phf_x(x_i, y_i) + phK_1f_y(x_i, y_i) + O(h^2) \\ &= y'(x_i) + phy''(x_i) + O(h^2) \end{aligned}$$

将其代入推广 Euler 公式, 得

$$\begin{aligned} y_{i+1} &= y_i + h \{ \lambda_1 y'(x_i) + \lambda_2 [y'(x_i) + phy''(x_i) + O(h^2)] \} \\ &= y_i + (\lambda_1 + \lambda_2)hy'(x_i) + \lambda_2 ph^2 y''(x_i) + O(h^3) \end{aligned}$$

将二者进行比较, 若要求 $R_i = O(h^3)$, 则必有

$$\begin{aligned} \lambda_1 + \lambda_2 &= 1 \\ \lambda_2 p &= \frac{1}{2} \end{aligned}$$

上式存在无穷多解, 所有满足上式得格式统称为 2 阶 Runge-Kutta 格式. 特别地, 当 $p = 1, \lambda_1 = \lambda_2 = 1/2$ 时, 上式化为改进的 Euler 法.

除此之外, 还有若干方法 (如二阶中点方法, 二阶 Heun 方法等), 在此不赘述.

7.3.1 二阶 Runge-Kutta 格式精度

令

$$\begin{aligned} F &= f_x + f_y f \\ G &= f_{xx} + 2f_{xy}f + f_{yy}f^2 \end{aligned}$$

则

$$k_2 = f + phF + \frac{1}{2}p^2 h^2 G + O(h^3)$$

即

$$\begin{aligned} y_{i+1} &= y_i + (\lambda_1 + \lambda_2)hf + \lambda_2 ph^2 F + \frac{1}{2}\lambda_2 p^2 h^3 G + O(h^4) \\ &= y_i + hf + \frac{1}{2}h^2 F + \frac{1}{4}ph^3 G + O(h^4) \end{aligned}$$

同时得

$$\begin{aligned} y(x_{i+1}) &= y(x_i) + hy'(x_i) + \frac{h^2}{2!}y''(x_i) + \frac{h^3}{3!}y'''(x_i) + O(h^4) \\ &= y_i + hf + \frac{h^2}{2}F + \frac{h^3}{6}(G + f_y F) + O(h^4) \end{aligned}$$

因此对于二阶 Runge-Kutta 方法而言, 其精度不超过二阶.

对于高精度而言, 可以取更多的点, 即做如下推广:

$$\begin{cases} y_{i+1} = y_i + h[\lambda_1 K_1 + \lambda_2 K_2 + \cdots + \lambda_m K_m] \\ K_1 = f(x_i, y_i) \\ K_2 = f(x_i + \alpha_2 h, y_i + \beta_{21} h K_1) \\ \vdots \\ K_m = f(x_i + \alpha_m h, y_i + \beta_{m1} h K_1 + \beta_{m2} h K_2 + \cdots + \beta_{m, m-1} h K_{m-1}) \end{cases}$$

其中 $\lambda_i (i = 1, 2, \cdots, m), \alpha_i (i = 2, \cdots, m), \beta_{ij} (i = 1, 2, \cdots, m; j = 1, 2, \cdots, i-1)$ 均为待定系数.

对于 n 阶显式 Runge-Kutta 公式:

$$\begin{cases} y_{i+1} = y_i + h \sum_{j=1}^n \lambda_j k_j \\ k_1 = f(x_i, y_i) \\ k_j = f\left(x_i + c_j h, y_i + h \sum_{m=1}^{j-1} a_{jm} k_m\right), j = 2, 3, \cdots, n \end{cases}$$

其中

$$\sum_{m=1}^{j-1} a_{jm} = c_j$$

当 $n = 3$ 时, 得

$$\begin{cases} y_{i+1} = y_i + h(\lambda_1 k_1 + \lambda_2 k_2 + \lambda_3 k_3) \\ k_1 = f(x_i, y_i) \\ k_2 = f(x_i + c_2 h, y_i + c_2 h k_1) \\ k_3 = f(x_i + c_3 h, y_i + a_{31} h k_1 + a_{32} h k_2) \\ c_3 = a_{31} + a_{32} \end{cases}$$

记

$$F = f_x + f_y x, G = f_{xx} + 2f f_{xy} + f^2 f_{yy}$$

则可得

$$\begin{aligned} & a_{31} k_1 + a_{32} k_2 \\ &= c_3 f + c_2 a_{32} h F + \frac{1}{2} c_2^2 a_{32} h^2 G + O(h^3) \\ & k_3 = f + c_3 h F + h^2 (c_2 a_{32} F f_y + \frac{1}{2} c_3^2 G) + O(h^3) \end{aligned}$$

整理, 若要使局部截断误差为 $O(h^4)$, 则必须有

$$\begin{cases} \lambda_1 + \lambda_2 + \lambda_3 = 1 \\ \lambda_2 c_2 + \lambda_3 c_3 = \frac{1}{2} \\ \lambda_2 c_2^2 + \lambda_3 c_3^2 = \frac{1}{3} \\ \lambda_3 c_2 a_{32} = \frac{1}{6} \end{cases}$$

对于不同的取值得到不同的方法. 具体公式详见书后附录.

类似地, 也可以得到四阶 Runge-Kutta 方法, 这里以四阶经典 Runge-Kutta 方法为例 (公式见书后附录), 演示一道例题.

例 7.3.1. 用四阶 Runge-Kutta 方法求初值问题

$$f(x, y) = -\frac{1}{2+y}, y(0) = 1$$

取步长 $h = 1$, 求 $y(1)$ 的值

解. 取 $x_0 = 0, y_0 = 1, x_1 = x_0 + h = 1$, 计算 k_1, k_2, k_3, k_4 有

$$\begin{cases} k_1 = f(x_n, y_n) \\ k_2 = f(x_n + \frac{h}{2}, y_n + \frac{h}{2}k_1) \\ k_3 = f(x_n + \frac{h}{2}, y_n + \frac{h}{2}k_2) \\ k_4 = f(x_n + h, y_n + hk_3) \end{cases}$$

将其代入

$$y_{n+1} = y_n + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

得

$$y_1 = 1 + \frac{1}{6} \times (-0.3333 - 2 \times 0.3529 - 2 \times 0.3542 - 0.3780) \approx 0.645744444$$

即使用四阶 Runge-Kutta 方法在 $x = 1$ 处得估计值为 0.645744444. \square

对于上例而言, 不难验证该问题有精确解析解

$$y(x) = \sqrt{9-2x} - 2$$

即对于 $x = 1$ 而言, 可得其精确值为

$$y(1) = \sqrt{7} - 2 \approx 0.645751311$$

截断误差为

$$R_1 = y(1) - y_1 = 0.0000068666$$

注意: 对于二阶 Runge-Kutta 方法, 其局部截断误差为 $O(h^3)$, 对于三阶 Runge-Kutta 方法, 局部截断误差为 $O(h^4)$, 对于四阶 Runge-Kutta 方法, 局部截断误差为 $O(h^5)$.

可以看出, 随着计算 K_i 的值越多, 其精度越高. 但根据 Butcher 给出得计算量与可达最高精度阶数关系, 当 K_i 的个数达到 $n(n \geq 8)$ 时, 可达到最高精度反而降低至 $O(h^{n-2})$. 因此在实际应用时, 常采用低阶算法而将步长 h 取小.

附录 A

矩阵分析基础

A.1 向量范数

A.1.1 向量范数的定义

定义 A.1.1. 设 $X \in R^n$, $\|X\|$ 表示定义在 R^n 上的一个实函数, 称之为 X 的范数, 其具有如下性质:

1. 非负性, 即对一切 $X \in R^n, X \neq 0, \|X\| > 0$;
2. 齐次性, 即对任何实数 $a \in R, x \in R^n$, 有 $\|aX\| = \|a\| \cdot \|X\|$;
3. 三角不等式, 即对任意两个向量 $X, Y \in R^n$, 有 $\|X + Y\| \leq \|X\| + \|Y\|$

A.1.2 常用的向量范数

设向量 $X = (x_1, x_2, \dots, x_n)^T \in R^n$, 则定义三种常用范数为

$$\begin{aligned}\|X\|_{\infty} &= \max_{1 \leq i \leq n} |x_i| \\ \|X\|_1 &= \sum_{i=1}^n |x_i| \\ \|X\|_2 &= \left(\sum_{i=1}^n |x_i|^2 \right)^{1/2}\end{aligned}$$

分别称为 ∞ -范数, 1-范数和 2-范数

例 A.1.1. 计算向量 $X = (1, 2, -3)^T$ 的范数

解.

$$\begin{aligned}\|X\|_1 &= \sum_{i=1}^n |x_i| = 6 \\ \|X\|_2 &= \left(\sum_{i=1}^n x_i^2 \right)^{1/2} = \sqrt{14} \\ \|X\|_\infty &= \max_{1 \leq i \leq n} |x_i| = 3\end{aligned}$$

□

A.1.3 向量范数性质

定义 A.1.2. 如果 R^n 中有两个范数 $\|X\|_s$ 与 $\|X\|_t$, 存在常数 $m, M > 0$, 使得对任意 n 维向量 X , 有

$$m\|X\|_s \leq \|X\|_t \leq M\|X\|_s$$

则称这两个范数等价

可以验证, 向量范数有如下不等式关系:

$$\begin{aligned}\frac{1}{n}\|X\|_1 &\leq \|X\|_\infty \leq \|X\|_1 \\ \|X\|_\infty &\leq \|X\|_1 \leq n\|X\|_\infty \\ \|X\|_\infty &\leq \|X\|_2 \leq \sqrt{n}\|X\|_\infty\end{aligned}$$

定义 A.1.3. 设给定 R^n 中的向量序列 X_k , 即

$$X_0, X_1, \dots, X_k, \dots$$

若对任何 $i (i = 1, 2, \dots, n)$ 都有

$$\lim_{k \rightarrow \infty} x_i^{(k)} = x_i^*$$

则向量

$$X^* = (x_1^*, x_2^*, \dots, x_n^*)^T$$

称为向量序列 X_k 的极限, 或者说向量序列 X_k 依坐标收敛于向量 X^* , 记为

$$\lim_{k \rightarrow \infty} X_k = X^*$$

定理 A.1.1. 向量序列 X_k 依坐标收敛于 X^* 的充要条件是

$$\lim_{k \rightarrow \infty} \|X_k - X^*\| = 0$$

注: 若某向量序列在某一范数意义下收敛时, 根据向量范数的等价性, 则在其他范数意义下也收敛.

A.2 矩阵范数

A.2.1 矩阵范数的定义

定义 A.2.1. 设对任意矩阵 $A \in R^{n \times n}$, 按一定的规则有一实数与之对应, 记作 $\|A\|$, 若 $\|A\|$ 满足

1. 正定性: $\|A\| \geq 0$, 当且仅当 $A = 0$ 时有 $\|A\| = 0$;
2. 齐次性: $\|cA\| = |c| \|A\|, \forall c \in R$;
3. 三角不等式: $\|A + B\| \leq \|A\| + \|B\|$;
4. 相容性: $\|AB\| \leq \|A\| \|B\|$

则称 $\|A\|$ 为矩阵 A 的范数

定义 A.2.2 (矩阵的算子范数). 设 $x \in R^n, A \in R^{n \times n}, \|X\|_v$ 是向量范数 ($v = 1, 2, \infty$), 则

$$\|A\|_v = \sup_{X \neq \theta} \frac{\|AX\|_v}{\|X\|_v}$$

是矩阵的非负函数, 称为矩阵 A 的算子范数

注:

$$\|A\|_v = \sup_{X \neq \theta} \frac{\|AX\|_v}{\|X\|_v} = \sup_{X \neq \theta} \|A \frac{X}{\|X\|_v}\|$$

令

$$y = \frac{X}{\|X\|_v}$$

则

$$\|A\|_v = \max_{\|y\|_v=1} \|Ay\|_v$$

定理 A.2.1. 设 $\|\cdot\|_v$ 是 R^n 中的向量范数, 则 $\|A\|_v$ 为 $R^{n \times n}$ 上的矩阵范数, 且满足

$$\|Ax\|_v \leq \|A\|_v \|x\|_v$$

A.2.2 常用的矩阵范数

定理 A.2.2. 设 n 阶方阵 $A = (a_{ij})_{n \times n}$, 则

1. 与 $\|x\|_1$ 相容的矩阵范数是

$$\|A\|_1 = \max_j \sum_{i=1}^n |a_{ij}|$$

2. 与 $\|x\|_\infty$ 相容的矩阵范数是

$$\|A\|_\infty = \max_i \sum_{j=1}^n |a_{ij}|$$

3. 与 $\|x\|_2$ 相容的矩阵范数是

$$\|A\|_2 = \sqrt{\lambda_{\max}(A^T A)}$$

其中 $\lambda_{\max}(A^T A)$ 为矩阵 $A^T A$ 的最大特征值.

上述三种范数又分别称为矩阵的 1-范数, ∞ -范数, 2-范数. 根据求和方式, 又分别称为列和范数, 行和范数, 谱范数

特殊的, 定义 Frobenius 范数为:

$$\|A\|_F = \sqrt{\sum_{i=1}^n \sum_{j=1}^n |a_{ij}|^2}$$

可将该范数看作对向量范数 $\|\cdot\|_2$ 的直接推广.

可以证明, 对于方阵 $A \in R^{n \times n}$ 和 $x \in R^n$, 有

$$\|Ax\|_2 \leq \|A\|_F \cdot \|x\|_2$$

注意, F-范数不是算子范数, 但有如下性质:

$$\|A\|_F = \sqrt{\text{tr}(A^T A)} = \sqrt{\text{tr}(A A^T)}$$

例 A.2.1. 计算矩阵

$$A = \begin{pmatrix} 1 & -2 \\ -3 & 4 \end{pmatrix}$$

的各种范数

解.

$$\|A\|_{\infty} = \max_{1 \leq i \leq n} \sum_{j=1}^n |a_{ij}| = \max\{1+2, 3+4\} = 7$$

$$\|A\|_1 = \max_{1 \leq j \leq n} \sum_{i=1}^n |a_{ij}| = \max\{1+3, 2+4\} = 6$$

$$\|A\|_F = \left(\sum_{i,j=1}^n a_{ij}^2 \right)^{1/2} = \sqrt{1+2+9+16} \approx 5.477$$

下面计算 2-范数

$$\|A\|_2 = \sqrt{\lambda_{\max}(A^T A)}$$

$$A^T A = \begin{pmatrix} 1 & -3 \\ -2 & 4 \end{pmatrix} \begin{pmatrix} 1 & -2 \\ -3 & 4 \end{pmatrix} = \begin{pmatrix} 10 & -14 \\ -14 & 20 \end{pmatrix}$$

令

$$\begin{vmatrix} \lambda - 10 & -14 \\ -14 & \lambda - 20 \end{vmatrix} = 0$$

解得

$$\lambda_{1,2} = 15 \pm \sqrt{221}$$

故最大特征值为

$$\lambda_{\max} = 15 + \sqrt{221}$$

所以得 2-范数

$$\|A\|_2 = \sqrt{\lambda_{\max}(A^T A)} = \sqrt{15 + \sqrt{221}}$$

□

A.2.3 矩阵范数与特征值之间的关系

定义 A.2.3. 矩阵 A 的所有特征值的最大模称为 A 的谱半径, 记作

$$\rho(A) = \max_{1 \leq i \leq n} |\lambda_i|$$

定理 A.2.3. 矩阵 A 谱半径不超过 A 的任一矩阵范数, 即

$$\rho(A) = \max_{1 \leq i \leq n} |\lambda_i| \leq \|A\|$$

证明. 设 λ 是矩阵 A 的任一特征值, x 为对应特征向量, 则特征方程

$$Ax = \lambda x$$

由矩阵范数的相容性可知,

$$|\lambda| \|x\| = \|\lambda x\| = \|Ax\| \leq \|A\| \|x\|$$

即

$$|\lambda| \leq \|A\|$$

□

推论 A.2.4. 若 A 为对称矩阵, 则

$$\rho(A) = \|A\|_2$$

注: $R^{n \times n}$ 中任意两个矩阵范数也是等价的.

定义 A.2.4. 设 $\|\cdot\|$ 为 $R^{n \times n}$ 上的矩阵范数, $A, B \in R^{n \times n}$, 称 $\|A - B\|$ 为 A 与 B 之间的距离

定义 A.2.5. 设给定 $R^{n \times n}$ 中的矩阵序列 A_k , 若

$$\lim_{k \rightarrow \infty} \|A_k - A\| = 0$$

则称矩阵序列 A_k 收敛于矩阵 A , 记作

$$\lim_{k \rightarrow \infty} A_k = A$$

定理 A.2.5. 若 $\|B\| < 1$, 则 $I \pm B$ 为非奇异矩阵, 且

$$\|(I \pm B)^{-1}\| \leq \frac{1}{1 - \|B\|}$$

其中, $\|\cdot\|$ 为矩阵的算子范数.

证明. 反证法, 假设 $\det\{(I \pm B)\} = 0$, 则线性方程组 $(I \pm B)x = 0$ 有非零解, 即存在 $x_0 \neq 0$ 使得

$$Bx_0 = x_0, \frac{\|Bx_0\|}{\|x_0\|} = 1$$

因此有

$$\|B\| \geq 1$$

与假设矛盾.

又因为

$$(I \pm B)(I \pm B)^{-1} = I$$

有

$$(I \pm B)^{-1} = I \mp B(I \pm B)^{-1}$$

从而

$$\|(I \pm B)^{-1}\| \leq \|I\| + \|B\| \|(I \pm B)^{-1}\|$$

即

$$\|(I \pm B)^{-1}\| \leq \frac{1}{1 - \|B\|}$$

□

定理 A.2.6. 设 $B \in R^{n \times n}$, 则由 B 的各幂次得到的矩阵序列 $B^k, k = 0, 1, 2, \dots$ 收敛于零矩阵, 即

$$\lim_{k \rightarrow \infty} B^k = 0$$

的充要条件是

$$\rho(B) < 1$$

A.2.4 矩阵的条件数

定义 A.2.6. 设矩阵 A 为非奇异矩阵, 则称

$$\text{cond}(A) = \|A^{-1}\| \|A\|$$

为矩阵 A 的条件数, 其中 $\|\cdot\|$ 为矩阵的算子范数.

对矩阵 A 的任意一个算子范数 $\|\cdot\|$, 有

1. $\text{cond}(A) = \|A^{-1}\| \|A\| \geq \|A^{-1} \cdot A\| = \|I\| = 1$;
2. $\text{cond}(kA) = \text{cond}(A)$, k 为非零常数;
3. 若 $\|A\| = 1$, 则 $\text{cond}(A) = \|A^{-1}\|$

注: $\text{cond}(A)$ 与所取范数有关. 常用的条件数有:

$$\begin{aligned}\text{cond}(A)_1 &= \|A^{-1}\|_1 \|A\|_1 \\ \text{cond}(A)_\infty &= \|A^{-1}\|_\infty \|A\|_\infty \\ \text{cond}(A)_2 &= \sqrt{\lambda_{\max}(A^T A) / \lambda_{\min}(A^T A)}\end{aligned}$$

特别地, 当 A 对称时, 则

$$\text{cond}(A)_2 = \frac{\max |\lambda_i|}{\min |\lambda_i|}$$

A.3 初等矩阵

A.3.1 初等矩阵

定义 A.3.1. 设向量 $u, v \in R^n, \sigma \in R$, 则形如

$$E(u, v, \sigma) = I - \sigma uv^T$$

的矩阵叫做实初等矩阵, 其中 I 为 n 阶单位矩阵

A.3.2 初等下三角矩阵

定义 A.3.2. 令向量 $u = l_i = (0, \dots, 0, l_{i+1,i}, \dots, l_{ni})^T$, 向量 $v = e_i, \sigma = 1$, 则称矩阵

$$L_i = L_i(l_i) = E(l_i, e_i; 1) = I - l_i e_i^T = \begin{pmatrix} 1 & & & & \\ & \ddots & & & \\ & & 1 & & \\ & & -l_{i+1,i} & 1 & \\ & & \vdots & & \ddots \\ & & -l_{ni} & & & 1 \end{pmatrix}$$

为初等下三角阵

定理 A.3.1. 初等下三角阵 L_i 具有如下性质:

1. $L_i^{-1}(l_i) = L_i(-l_i), |L_i| = 1$;

2.

$$L = L_1(l_1)L_2(l_2)\cdots L_{n-1}(l_{n-1}) = \begin{pmatrix} 1 & & & \\ -l_{21} & 1 & & \\ \vdots & \vdots & \ddots & \\ -l_{n1} & -l_{n2} & \cdots & 1 \end{pmatrix}$$

为单位下三角阵;

3. 任何一个单位下三角阵 $L \in R^n$ 都可分裂成

$$L = I - l_1 e_1^T - l_2 e_2^T - \cdots - l_{n-1} e_{n-1}^T$$

因此, 对任一非奇异下三角阵 L , 都可分裂成一个非奇异对角阵和若干个下三角阵的乘积;

4. L_i 左乘矩阵 A 的结果是从 A 的各行中减去第 i 行乘一个因子.

附录 B

常见 Runge-Kutta 公式

正如前文所说, Runge-Kutta 公式具有无穷多种格式, 这里给出一些常见的 Runge-Kutta 公式

B.1 二阶 Runge-Kutta 方法

$$\begin{cases} y_{i+1} = y_i + h[\lambda_1 K_1 + \lambda_2 K_2] \\ K_1 = f(x_i, y_i) \\ K_2 = f(x_i + ph, y_i + phK_1) \end{cases}$$

其中

$$\lambda_1 + \lambda_2 = 1, \lambda_2 p = \frac{1}{2}$$

B.1.1 二阶中点方法

取 $\lambda_1 = 0, \lambda_2 = 1, p = 1/2$

$$\begin{cases} y_{i+1} = y_i + hk_2 \\ k_1 = f(x_i, y_i) \\ k_2 = f(x_i + \frac{h}{2}, y_i + \frac{h}{2}k_1) \end{cases}$$

B.1.2 二阶 Hune 方法

取 $\lambda_1 = 1/4, \lambda_2 = 3/4, p = 2/3$

$$\begin{cases} y_{i+1} = y_i + h \left(\frac{1}{4}k_1 + \frac{3}{4}k_2 \right) \\ k_1 = f(x_i, y_i) \\ k_2 = f(x_i + \frac{2h}{3}, y_i + \frac{2h}{3}k_1) \end{cases}$$

B.2 三阶 Runge-Kutta 方法

$$\begin{cases} y_{i+1} = y_i + h(\lambda_1 k_1 + \lambda_2 k_2 + \lambda_3 k_3) \\ k_1 = f(x_i, y_i) \\ k_2 = f(x_i + c_2 h, y_i + c_2 h k_1) \\ k_3 = f(x_i + c_3 h, y_i + a_{31} h k_1 + a_{32} h k_2) \\ c_3 = a_{31} + a_{32} \end{cases}$$

其中

$$\begin{cases} \lambda_1 + \lambda_2 + \lambda_3 = 1 \\ \lambda_2 c_2 + \lambda_3 c_3 = \frac{1}{2} \\ \lambda_2 c_2^2 + \lambda_3 c_3^2 = \frac{1}{3} \\ \lambda_3 c_2 a_{32} = \frac{1}{6} \end{cases}$$

B.2.1 三阶 Kutta 方法

取 $\lambda_1 = 1/6, \lambda_2 = 2/3, \lambda_3 = 1/6, c_2 = 1/2, c_3 = 1, a_{32} = 2, a_{31} = -1$

$$\begin{cases} y_{i+1} = y_i + h \left(\frac{1}{6}k_1 + \frac{2}{3}k_2 + \frac{1}{6}k_3 \right) \\ k_1 = f(x_i, y_i) \\ k_2 = f(x_i + \frac{1}{2}h, y_i + \frac{1}{2}h k_1) \\ k_3 = f(x_i + h, y_i - h k_1 + 2h k_2) \end{cases}$$

B.2.2 三阶 Hune 方法

取 $\lambda_1 = 1/4, \lambda_2 = 0, \lambda_3 = 3/4, c_2 = 1/3, c_3 = 2/3, a_{32} = 2/3, a_{31} = 0$

$$\begin{cases} y_{i+1} = y_i + h \left(\frac{1}{4}k_1 + \frac{3}{4}k_3 \right) \\ k_1 = f(x_i, y_i) \\ k_2 = f\left(x_i + \frac{1}{3}h, y_i + \frac{1}{3}hk_1\right) \\ k_3 = f\left(x_i + \frac{2}{3}h, y_i + \frac{2}{3}hk_2\right) \end{cases}$$

B.3 四阶 Runge-Kutta 方法

$$\begin{cases} y_{i+1} = y_i + h(\lambda_1 k_1 + \lambda_2 k_2 + \lambda_3 k_3 + \lambda_4 k_4) \\ k_1 = f(x_i, y_i) \\ k_2 = f(x_i + c_2 h, y_i + c_2 h k_1) \\ k_3 = f(x_i + c_3 h, y_i + a_{31} h k_1 + a_{32} h k_2) \\ k_4 = f(x_i + c_4 h, y_i + a_{41} h k_1 + a_{42} h k_2 + a_{43} h k_3) \\ c_3 = a_{31} + a_{32}, c_4 = a_{41} + a_{42} + a_{43} \end{cases}$$

其中

$$\begin{cases} a_{31} + a_{32} = c_3 \\ a_{41} + a_{42} + a_{43} = c_4 \\ \lambda_1 + \lambda_2 + \lambda_3 + \lambda_4 = 1 \\ \lambda_2 c_2 + \lambda_3 c_3 + \lambda_4 c_4 = \frac{1}{2} \\ \lambda_2 c_2^2 + \lambda_3 c_3^2 + \lambda_4 c_4^2 = \frac{1}{3} \\ \lambda_2 c_2^3 + \lambda_3 c_3^3 + \lambda_4 c_4^3 = \frac{1}{4} \\ \lambda_3 c_2 a_{32} + \lambda_4 (c_2 a_{42} + c_3 a_{43}) = \frac{1}{6} \\ \lambda_3 c_2^2 a_{32} + \lambda_4 (c_2^2 a_{42} + c_3^2 a_{43}) = \frac{1}{12} \\ \lambda_3 c_2 c_3 a_{32} + \lambda_4 c_4 (c_2 a_{42} + c_3 a_{43}) = \frac{1}{8} \\ \lambda_4 c_2 a_{32} a_{43} = \frac{1}{24} \end{cases}$$

B.3.1 四阶经典 Runge-Kutta 方法

$$\begin{cases} y_{i+1} = y_i + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4) \\ k_1 = f(x_i, y_i) \\ k_2 = f(x_i + \frac{h}{2}, y_i + \frac{h}{2}k_1) \\ k_3 = f(x_i + \frac{h}{2}, y_i + \frac{h}{2}k_2) \\ k_4 = f(x_i + h, y_i + hk_3) \end{cases}$$