

# 数值分析与计算<sup>\*</sup>

潘建瑜

[jypan@math.ecnu.edu.cn](mailto:jypan@math.ecnu.edu.cn)

(2021 年 9 月)

纸上得来终觉浅，绝知此事要躬行。



# 目录

## 第一讲 引论与预备知识

1.1	数值分析引论	1
1.1.1	科学计算	1
1.1.2	数值分析的研究内容	2
1.2	线性代数基础	5
1.2.1	线性空间基本概念	5
1.2.2	范数与赋范线性空间	7
1.2.3	内积与内积空间	7
1.2.4	矩阵的特征值与谱半径	11
1.2.5	向量范数与矩阵范数	13
1.2.6	矩阵标准型	18
1.2.7	对称正定矩阵	20
1.3	数值计算中的误差	22
1.3.1	绝对误差和绝对误差限	22
1.3.2	相对误差和相对误差限	23
1.3.3	有效数字	23
1.3.4	误差估计的基本方法	26
1.3.5	误差分析	28
1.3.6	数值稳定性	28
1.3.7	减小误差危害	30
1.4	课后练习	35

## 第二讲 线性方程组直接解法

2.1	Gauss 消去法	37
2.1.1	Gauss 消去过程	38
2.1.2	Gauss 消去法与 LU 分解	41
2.1.3	列主元 Gauss 消去法与 PLU 分解	45

2.2	矩阵分解法	50
2.2.1	LU 分解与 PLU 分解	50
2.2.2	Cholesky 分解与平方根法	53
2.2.3	三对角线性方程组	56
2.2.4	带状线性方程组	58
2.3	扰动分析	60
2.3.1	矩阵条件数	60
2.3.2	条件数与病态之间的关系	62
2.4	解的改进 *	65
2.4.1	高精度运算	65
2.4.2	矩阵元素缩放 (Scaling)	65
2.4.3	迭代改进法	65
2.5	课后练习	67

### 第三讲 线性最小二乘问题

3.1	问题介绍	70
3.1.1	超定方程组	71
3.1.2	欠定方程组	71
3.2	基本矩阵变换	72
3.2.1	Householder 变换	72
3.2.2	Givens 变换	75
3.2.3	正交变换的舍入误差分析	76
3.3	QR 分解	78
3.3.1	QR 分解的存在性与唯一性	78
3.3.2	基于 MGS 的 QR 分解	81
3.3.3	基于 Householder 变换的 QR 分解	81
3.3.4	基于 Givens 变换的 QR 分解	84
3.3.5	QR 分解的稳定性	85
3.4	奇异值分解	87
3.4.1	奇异值与奇异值分解	87
3.4.2	奇异值的性质	89
3.5	线性最小二乘问题的求解方法	92
3.5.1	正规方程法	92
3.5.2	QR 分解法	93
3.5.3	奇异值分解法	95



3.6	广义逆与最小二乘	96
3.7	最小二乘扰动分析 *	99
3.8	最小二乘问题的推广与应用 *	100
3.9	课后习题	103

## 第四讲 线性方程组迭代方法

4.1	迭代法基本概念	105
4.1.1	向量序列与矩阵序列的收敛性	105
4.1.2	基于矩阵分裂的迭代法	107
4.1.3	迭代法的收敛性	108
4.1.4	迭代方法的收敛速度	110
4.2	Jacobi, Gauss-Seidel 和 SOR	112
4.2.1	Jacobi 迭代方法	112
4.2.2	Gauss-Seidel 迭代方法	113
4.2.3	SOR 迭代方法	113
4.3	收敛性分析	116
4.3.1	不可约与对角占优	116
4.3.2	Jacobi 和 G-S 的收敛性	117
4.3.3	SOR 的收敛性	117
4.4	共轭梯度法	119
4.4.1	最速下降法	119
4.4.2	共轭梯度法	119
4.5	课后练习	120

## 第五讲 矩阵特征值计算

5.1	非对称特征值问题	122
5.1.1	幂迭代方法	122
5.1.2	反迭代方法	123
5.1.3	QR 迭代方法	125
5.1.4	带位移的隐式 QR 迭代	129
5.2	对称特征值问题	139
5.2.1	Jacobi 迭代方法	139
5.2.2	分而治之法	143
5.2.3	对分法	147

5.3	计算奇异值分解 *	150
5.4	扰动分析	157
5.4.1	特征值与 Rayleigh 商	157
5.4.2	对称矩阵特征值的扰动分析	159
5.4.3	Rayleigh 商逼近	160
5.5	应用	162
5.5.1	多项式求根	162
5.5.2	Google 网页排名: PageRank	163
5.6	课后习题	164

## 第六讲 非线性方程

6.1	对分法	168
6.1.1	对分法基本思想	168
6.1.2	对分法的收敛性	169
6.2	不动点迭代法	171
6.2.1	收敛性分析	171
6.2.2	收敛阶	175
6.3	Steffensen 迭代法	177
6.3.1	Aitken 加速技巧	177
6.3.2	Steffensen 迭代法	177
6.4	Newton 法	179
6.4.1	基本思想与迭代格式	179
6.4.2	Newton 法的收敛性	180
6.4.3	简化 Newton 法	181
6.4.4	Newton 下山法	182
6.4.5	重根情形	182
6.5	割线法与抛物线法	184
6.5.1	割线法	184
6.5.2	抛物线法	184
6.6	非线性方程组的数值求解 *	186
6.7	非线性最小二乘问题 *	187
6.8	课后练习	188

## 第七讲 函数插值

7.1	多项式插值	189
-----	-------	-----



7.2	Lagrange 插值	193
7.2.1	Lagrange 基函数	193
7.2.2	插值余项	194
7.2.3	Lagrange 基函数的两个重要性质	196
7.3	Newton 插值	199
7.3.1	差商	200
7.3.2	Newton 插值公式	201
7.3.3	差分	204
7.4	Hermite 插值	207
7.4.1	重节点差商与 Taylor 插值	207
7.4.2	两个典型的 Hermite 插值	207
7.5	分段低次插值	211
7.5.1	分段线性插值	211
7.5.2	分段三次 Hermite 插值	212
7.6	三次样条插值	214
7.6.1	三次样条函数	214
7.6.2	边界条件	214
7.6.3	三次样条函数的计算	215
7.6.4	误差估计	221
7.7	课后练习	222

## 第八讲 函数逼近

8.1	基本概念与预备知识	225
8.2	正交多项式	227
8.2.1	正交函数族与正交多项式	227
8.2.2	Legendre 多项式	230
8.2.3	Chebyshev 多项式	232
8.2.4	Chebyshev 多项式零点插值	233
8.2.5	其他正交多项式	236
8.3	最佳平方逼近	237
8.3.1	怎样求最佳平方逼近	237
8.3.2	用正交函数计算最佳平方逼近	238
8.3.3	最佳平方逼近多项式	239
8.3.4	用正交多项式计算最佳平方逼近多项式	240
8.4	最佳一致逼近	242
8.4.1	最佳一致逼近多项式的存在唯一性	242
8.4.2	$n$ 次多项式的 $n-1$ 次最佳一致逼近多项式	243
8.4.3	Chebyshev 级数与近似最佳一致逼近	243

8.5	最小二乘曲线拟合	245
8.5.1	曲线拟合介绍	245
8.5.2	最小二乘与法方程	246
8.5.3	多项式拟合	249
8.5.4	用正交多项式做最小二乘拟合	249
8.6	有理逼近	252
8.7	课后练习	254

## 第九讲 数值积分与数值微分

9.1	数值积分基本概念	257
9.1.1	机械求积公式	257
9.1.2	代数精度	258
9.1.3	收敛性与稳定性	259
9.1.4	插值型求积公式	260
9.2	Newton-Cotes 公式	261
9.2.1	常用的低次 Newton-Cotes 公式	261
9.2.2	余项公式的推导	262
9.2.3	Newton-Cotes 公式余项的一般形式	266
9.2.4	一般求积公式余项	266
9.3	复合求积公式	268
9.3.1	复合梯形公式	268
9.3.2	复合 Simpson 公式	269
9.4	带导数的求积公式	270
9.4.1	带导数的梯形公式	270
9.4.2	带导数的 Simpson 公式	270
9.5	Romberg 求积公式	271
9.5.1	外推技巧	271
9.5.2	Romberg 算法	272
9.6	自适应求积方法 *	273
9.7	Gauss 求积公式	274
9.7.1	一般 Gauss 求积公式	274
9.7.2	Gauss-Legendre 公式	276
9.7.3	Gauss-Chebyshev 公式	278
9.7.4	复合 Gauss 公式	278





9.8	多重积分	279
9.9	数值微分	280
9.9.1	插值型求导公式	280
9.9.2	一阶导数的差分近似	280
9.9.3	二阶导数的差分近似	281
9.9.4	三次样条求导	281
9.9.5	数值微分的外推算法	281
9.10	课后练习	282

## 第十讲 常微分方程初值问题

10.1	单步法	285
10.1.1	Euler 法	285
10.1.2	梯形法	287
10.1.3	改进的 Euler 法	287
10.1.4	单步法误差分析和收敛性	288
10.2	Runge-Kutta 法	291
10.3	线性多步法	296
10.3.1	显式 Adams 法	296
10.3.2	一般线性多步法	297
10.3.3	预估-校正方法	298
10.4	一阶方程组与高阶方程 *	300
10.4.1	一阶方程组	300
10.4.2	高阶方程	301
10.5	课后练习	303

## 参考文献

请勿外传

*This page intentionally left blank.*

仅供课堂使用，

# 1

## 引论与预备知识

### 1.1 数值分析引论

#### 1.1.1 科学计算

计算机是二十世纪最伟大的科学技术发明之一. 计算机对人类的生产活动和社会活动产生了极其重要的影响. 特别是随着网络的出现, 计算机已经彻底改变了人们的生活, 学习和工作. 它是人类进入信息时代的重要标志之一.

随着计算机技术的飞速发展, 数学方法及计算已成为当今科学研究中不可缺少的手段, 从宇宙飞船到家用电器, 从质量控制到市场营销, 通过建立数学模型, 应用数学理论和方法, 并结合计算机解决实际问题已成为十分普遍的研究模式.

一门科学, 只有当它成功地运用数学时, 才能达到真正完善的地步.

— 马克思 (1818.05.05 – 1883.03.14)

一个国家只有数学蓬勃发展, 才能表现她的国力强大.

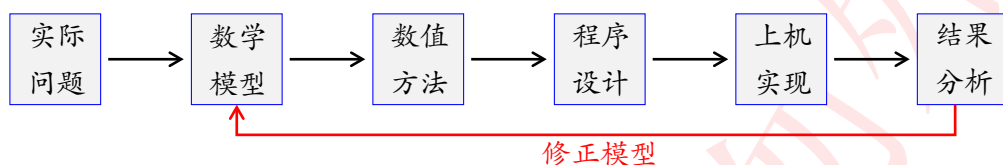
— 陈省身 (1911.10.26 – 2004.12.03)

科学计算就是指借助计算机高速计算的能力, 解决现代科学, 工程, 经济或人文中的复杂问题. 它融建模、算法、软件研制和计算模拟为一体, 是计算机实现其在高科技领域应用的必不可少的纽带和工具. 计算已不仅仅只是作为验证理论模型正确性的手段, 大量的事例表明它已成为重大科学发现的一种重要手段 [64]. 科学计算已经和理论与实验研究一起, 成为当今世界科学技术创新的主要方式 [59], 也是当前公认的从事现代科学研究的第三种方法.

高性能科学计算在国家安全和科技创新等方面的重要作用也日益受到世界各国的重视. 欧美等国家已投入巨资, 大力发展先进的计算方法, 研制大型的实用软件. 2005 年 6 月, 美国总统信息技术咨询委员会 (President's Information Technology Advisory Committee) 在给美国总统提交的报告《计算科学: 确保美国竞争力》(Computational Science: Ensuring America's Competitiveness) 中明确阐述了计算科学的重要性. 报告认为, 虽然计算本身也是一门学科, 但是其有促进其他学科发展的作

用, 21 世纪科学上最重要的、经济上最有前途的前沿研究都有可能通过先进的计算技术和计算科学而得到解决. 报告还认为, 在迅猛发展的高性能计算技术推动下, 计算科学将是 21 世纪确保国家核心竞争能力的战略技术之一, 而科学计算则是计算科学中最主要的内容.

一般来说, 运用科学计算实际问题主要分以下几个步骤:



数值分析的任务主要包括设计数值方法、程序设计、上机实现, 同时还需对数值方法进行理论分析.

2000 年, 由 IEEE 主办的杂志 Computing in Science & Engineering 评选出了 20 世纪对科学与工程的发展影响最大的十个算法, 按时间顺序排列如下 (也可参见 SIAM News, Volume 33, Number 4, 2000):

1. Monte Carlo method (1946)
2. Simplex Method for Linear Programming (1947)
3. Krylov Subspace Iteration Methods (1950)
4. The Decompositional Approach to Matrix Computations (1951)
5. The Fortran Optimizing Compiler (1957)
6. QR Algorithm for Computing Eigenvalues (1959-61)
7. Quicksort Algorithm for Sorting (1962)
8. Fast Fourier Transform (1965)
9. Integer Relation Detection Algorithm (1977)
10. Fast Multipole Method (1987)

其中 2, 3, 4, 6, 8, 10 都与数值分析密切相关.

### 1.1.2 数值分析的研究内容

数值分析主要是研究各类与实际应用相关的数学问题的数值计算方法及其相关的数学理论, 也称为计算方法, 是数学与计算机的有机结合, 对应的数学分支称为计算数学. 1947 年, Von Neumann 和 Goldstine 在 Bulletin of the AMS (美国数学会通报) 上发表了题为 “Numerical inverting of matrices of high order” (高阶矩阵的数值求逆) 的著名论文 [53], 开启了现代计算数学的研究. 半个多世纪以来, 伴随着计算机技术的不断进步, 计算数学得到了蓬勃发展, 并逐渐成为了一个独立和重要的学科.

数值分析研究的核心是算法的设计与分析, 具体研究内容包括数值代数 (线性、非线性), 数值逼近 (数据拟合、函数插值、最佳逼近), 数值积分与数值微分, 微分方程数值解 (常微分方程、偏微分方程), 最优化理论与方法, 误差理论与扰动分析等.



有关计算数学的研究内容和意义可以参阅 *Princeton Companion to Mathematics* 上由 Trefethen 撰写的关于 Numerical Analysis 的介绍 [50].

### 数值分析的主要任务

- 算法设计: 构造求解各种数学问题的高效可靠的数值方法.
- 算法分析: 研究数值方法的收敛性、稳定性、计算复杂性、计算精度等.
- 算法实现: 编程实现与优化、软件开发和维护等.

关于术语“数值方法”或“数值算法”, 一般情况下我们将不加区分地使用.

一个好的数值方法一般需满足以下几点:

- 可靠: 有可靠的理论分析, 即收敛性、稳定性等有数学理论保证.
- 高效: 有良好的计算复杂性 (时间和空间).
- 可用: 可以并易于在计算机上编程实现.
- 实用: 要通过数值试验来证明是行之有效的.

需要指出的是, 数值方法是近似计算, 因此求出的解是有误差的.

对于同一问题, 不同的算法在计算性能上可能相差千百倍或者更多!

#### 例 1.1 线性方程组求解: 克莱姆法则与高斯消去法.

- 克莱姆法则需要计算  $n+1$  个  $n$  阶行列式, 在不计加减运算情况下, 大约需要  $n!(n^2-1)$  次乘除运算;
- 高斯消去法只需约  $2n^3/3$  次乘除和加减运算.

以  $n=20$  为例, 克莱姆法则大概需要  $20!(20^2-1) \approx 9.7 \times 10^{20}$  次运算, 如果用每秒运算 30 亿次 (主频 3.0G) 的计算机求解时, 大约需要 10000 年的时间! 但如果使用高斯消去法, 则 1 秒钟都不要.

#### 学习建议

在学习数值分析时, 大家要注意以下几点:

- 注意掌握数值方法的基本思想和原理 (不能仅仅靠记忆);
- 注意数值方法的设计和分析的一些常用技巧;
- 要重视误差分析、收敛性和稳定性的基本理论;
- 适量的数值计算训练 (包括编程实践和手工推导).

## 参考资料

- [1] R.L. Burden, J. Douglas Faires and A. M. Burden, [Numerical Analysis](#), 10th Edition, Brooks/Cole, Cengage Learning, 2016.
- [2] T. Sauer, [Numerical Analysis](#), 3rd Edition, Pearson, 2018.
- [3] M.T. Heath, [Scientific Computing: An Introductory Survey](#), Revised 2nd Edition, SIAM, 2018.
- [4] J. Stoer and R. Bulirsch, [Introduction to Numerical Analysis](#), 3rd Edition, Springer, 2002.

其中 [1-3] 比较基础, [4] 比较偏理论.

## 本讲义中常用的数学记号

符号	含义
$\mathbb{R}$	实数域
$\mathbb{R}_+$	所有正实数组成的集合
$\mathbb{C}$	复数域
$\mathbb{R}^n$	$n$ 维实向量空间
$\mathbb{C}^n$	$n$ 维复向量空间
$C[a, b]$	$[a, b]$ 上的连续函数空间
$C^p[a, b]$	$[a, b]$ 上的 $p$ 次连续可导函数空间
$\mathbb{H}_n$	所有次数不超过 $n$ 的实系数多项式组成的集合
$\tilde{\mathbb{H}}_n$	所有首项系数为 1 的 $n$ 次实系数多项式组成的集合
$x^\top$	向量的转置
$x^*$	向量的共轭转置
$\text{rank}(A)$	矩阵的秩
$\dim(\mathcal{S})$	线性空间的维数
$\text{span}\{x_1, x_2, \dots, x_k\}$	由 $x_1, x_2, \dots, x_k$ 张成的线性空间



## 1.2 线性代数基础

### 1.2.1 线性空间基本概念

线性空间是线性代数最基本的概念之一,它是定义在某个数域上并满足一定条件的一个集合.我们首先给出数域的概念.

**定义 1.1 (数域)** 设  $\mathbb{F}$  是包含 0 和 1 的一个数集,如果  $\mathbb{F}$  中的任意两个数的和,差,积,商 (除数不为 0) 仍然在  $\mathbb{F}$  中,则称  $\mathbb{F}$  为一个数域.

**例 1.2** 常见的数域有:有理数域  $\mathbb{Q}$ ,实数域  $\mathbb{R}$  和复数域  $\mathbb{C}$ .

(本讲义只考虑实数域和复数域)

**定义 1.2 (线性空间)** 设  $S$  是一个非空集合,  $\mathbb{F}$  是一个数域. 在  $S$  上定义一种代数运算,称为**加法**,记为“+”(即对任意  $x, y \in S$ , 都存在唯一的  $z \in S$ , 使得  $z = x + y$ ), 并定义一个从  $\mathbb{F} \times S$  到  $S$  的代数运算,称为**数乘**,记为“ $\cdot$ ”(即对任意  $\alpha \in \mathbb{F}$  和任意  $x \in S$ , 都存在唯一的  $y \in S$ , 使得  $y = \alpha \cdot x$ ). 如果这两个运算满足下面的规则,则称  $(S, +, \cdot)$  是数域  $\mathbb{F}$  上的一个**线性空间** (通常简称  $S$  是数域  $\mathbb{F}$  上的一个线性空间):

• **加法四条规则**

- (1) **交换律**:  $x + y = y + x, \quad \forall x, y \in S;$
- (2) **结合律**:  $(x + y) + z = x + (y + z), \quad \forall x, y, z \in S;$
- (3) **零元素**: 存在一个元素 0, 使得  $x + 0 = x, \quad \forall x \in S;$
- (4) **逆运算**: 对任意  $x \in S$ , 都存在**负元素**  $y \in S$ , 使得  $x + y = 0$ , 记  $y = -x$ ;

• **数乘四条规则**

- (1) **单位元**:  $1 \cdot x = x, \quad 1 \in \mathbb{F}, \forall x \in S;$
- (2) **结合律**:  $\alpha \cdot (\beta \cdot x) = (\alpha\beta) \cdot x, \quad \forall \alpha, \beta \in \mathbb{F}, x \in S;$
- (3) **分配律**:  $(\alpha + \beta) \cdot x = \alpha \cdot x + \beta \cdot x, \quad \forall \alpha, \beta \in \mathbb{F}, x \in S;$
- (4) **分配律**:  $\alpha \cdot (x + y) = \alpha \cdot x + \alpha \cdot y, \quad \forall \alpha \in \mathbb{F}, x, y \in S.$

为了表示方便,通常省略数乘符号,即将  $\alpha \cdot x$  写成  $\alpha x$ .

**例 1.3** 常见的线性空间有:

- $\mathbb{R}^n \rightarrow$  所有  $n$  维实向量组成的集合,是  $\mathbb{R}$  上的线性空间.
- $\mathbb{C}^n \rightarrow$  所有  $n$  维复向量组成的集合,是  $\mathbb{C}$  上的线性空间.
- $\mathbb{R}^{m \times n} \rightarrow$  所有  $m \times n$  阶实矩阵组成的集合,是  $\mathbb{R}$  上的线性空间.
- $\mathbb{C}^{m \times n} \rightarrow$  所有  $m \times n$  阶复矩阵组成的集合,是  $\mathbb{C}$  上的线性空间.
- $\mathbb{H}_n \rightarrow$  所有次数不超过  $n$  的多项式组成的集合.
- $C[a, b] \rightarrow$  区间  $[a, b]$  上所有连续函数组成的集合.



- $C^p[a, b] \rightarrow$  区间  $[a, b]$  上所有  $p$  次连续可微函数组成的集合.

### 线性无关和线性空间维数

设  $S$  是数域  $\mathbb{F}$  上的一个线性空间,  $x_1, x_2, \dots, x_k$  是  $S$  中的一组向量. 如果存在  $k$  个不全为零的数  $\alpha_1, \alpha_2, \dots, \alpha_k \in \mathbb{F}$ , 使得

$$\alpha_1 x_1 + \alpha_2 x_2 + \dots + \alpha_k x_k = 0,$$

则称  $x_1, x_2, \dots, x_k$  **线性相关**, 否则就是**线性无关**.

设  $x_1, x_2, \dots, x_k$  是  $S$  中的一组向量. 如果  $x \in S$  可以表示为

$$x = \alpha_1 x_1 + \alpha_2 x_2 + \dots + \alpha_k x_k,$$

其中  $\alpha_1, \alpha_2, \dots, \alpha_k \in \mathbb{F}$ , 则称  $x$  可以由  $x_1, x_2, \dots, x_k$  **线性表示**, 或者称  $x$  是  $x_1, x_2, \dots, x_k$  的**线性组合**,  $\alpha_1, \alpha_2, \dots, \alpha_k$  称为 **线性表出系数**.

设向量组  $\{x_1, x_2, \dots, x_m\}$ , 如果存在其中的  $r$  ( $r \leq m$ ) 个线性无关向量  $x_{i_1}, x_{i_2}, \dots, x_{i_r}$ , 使得所有向量都可以由它们线性表示, 则称  $x_{i_1}, x_{i_2}, \dots, x_{i_r}$  为向量组  $\{x_1, x_2, \dots, x_m\}$  的一个**极大线性无关组**, 并称这组向量的**秩**为  $r$ , 记为  $\text{rank}(\{x_1, x_2, \dots, x_m\}) = r$ .

设  $x_1, x_2, \dots, x_n$  是  $S$  中的一组线性无关向量. 如果  $S$  中的任意一个向量都可以由  $x_1, x_2, \dots, x_n$  线性表示, 则称  $x_1, x_2, \dots, x_n$  是  $S$  的一组**基**, 并称  $S$  是  $n$  维的, 即  $S$  的**维数**为  $n$ , 记为  $\dim(S) = n$ . 如果  $S$  中可以找到任意多个线性无关向量, 则称  $S$  是**无限维**的.

### 子空间

设  $S$  是一个线性空间,  $W$  是  $S$  的一个非空子集合. 如果  $W$  关于  $S$  上的加法和数乘也构成一个线性空间, 则称  $W$  为  $S$  的一个**线性子空间**, 有时简称**子空间**.

**例 1.4** 设  $S$  是一个线性空间, 则由零向量组成的子集  $\{0\}$  是  $S$  的一个子空间, 称为零子空间. 另外,  $S$  本身也是  $S$  的子空间. 这两个特殊的子空间称为  $S$  的**平凡子空间**, 其他子空间都是**非平凡子空间**.

下面给出子空间的判别定理.

**定理 1.1** 设  $S$  是数域  $\mathbb{F}$  上的一个线性空间,  $W$  是  $S$  的一个非空子集合. 则  $W$  是  $S$  的一个子空间的充要条件是  $W$  关于加法和数乘封闭, 即

- (1) 对任意  $x, y \in W$ , 有  $x + y \in W$ ;
- (2) 对任意  $\alpha \in \mathbb{F}$  和任意  $x \in W$ , 有  $\alpha x \in W$ .

**例 1.5** 设  $S$  是数域  $\mathbb{F}$  上的一个线性空间,  $x_1, x_2, \dots, x_k \in S$ , 记

$$\text{span}\{x_1, x_2, \dots, x_k\} \triangleq \{ \alpha_1 x_1 + \alpha_2 x_2 + \dots + \alpha_k x_k : \alpha_1, \alpha_2, \dots, \alpha_k \in \mathbb{F} \},$$





即由  $x_1, x_2, \dots, x_k$  的所有线性组合构成的集合, 则  $\text{span}\{x_1, x_2, \dots, x_k\}$  是  $S$  的一个线性子空间, 称为由  $x_1, x_2, \dots, x_k$  **张成的线性空间**.

### 1.2.2 范数与赋范线性空间

**定义 1.3 (范数与赋范线性空间)** 设  $S$  为数域  $F$  ( $F$  可以是  $\mathbb{R}$  或  $\mathbb{C}$ ) 上的线性空间, 若对任意  $x \in S$ , 存在唯一实数与之对应, 记为  $\|x\|$ , 它满足条件:


- (1)  $\|x\| \geq 0$ , 等号当且仅当  $x = 0$  时成立; (正定性)
- (2)  $\|\alpha x\| = |\alpha| \cdot \|x\|, \forall \alpha \in F$ ; (正齐次性)
- (3)  $\|x + y\| \leq \|x\| + \|y\|, \forall x, y \in S$ ; (三角不等式)

则称  $\|\cdot\|$  为线性空间  $S$  上的**范数**, 定义了范数的线性空间称为**赋范线性空间**.

 范数是从  $S$  到  $\mathbb{R}_+ \cup \{0\}$  的**一元函数**, 其中  $\mathbb{R}_+$  表示所有正实数组成的集合.

**例 1.6**  $\mathbb{R}^n / \mathbb{C}^n$  上的常用范数: 设  $x = [x_1, x_2, \dots, x_n]^T \in \mathbb{R}^n / \mathbb{C}^n$

- 1-范数:  $\|x\|_1 = \sum_{i=1}^n |x_i|$ ;
- 2-范数:  $\|x\|_2 = \left( \sum_{i=1}^n |x_i|^2 \right)^{\frac{1}{2}}$ ;
- $p$ -范数:  $\|x\|_p = \left( \sum_{i=1}^n |x_i|^p \right)^{\frac{1}{p}}$ ; ( $p$  为正整数)
- $\infty$ -范数:  $\|x\|_\infty = \max_{1 \leq i \leq n} |x_i|$ .

 1-范数, 2-范数,  $\infty$ -范数可以看作是  $p$ -范数在  $p = 1, 2, \infty$  时的特殊情形.

**例 1.7**  $C[a, b]$  上的常用范数: 设  $f(x) \in C[a, b]$

- 1-范数:  $\|f\|_1 = \int_a^b |f(x)| dx$ ;
- 2-范数:  $\|f\|_2 = \left( \int_a^b |f(x)|^2 dx \right)^{\frac{1}{2}}$ ;
- $p$ -范数:  $\|f\|_p = \left( \int_a^b |f(x)|^p dx \right)^{\frac{1}{p}}$ ; ( $p$  为正整数)
- $\infty$ -范数:  $\|f\|_\infty = \max_{a \leq x \leq b} |f(x)|$ .

### 1.2.3 内积与内积空间

**定义 1.4 (内积与内积空间)** 设  $S$  是数域  $F$  ( $\mathbb{R}$  或  $\mathbb{C}$ ) 上的一个线性空间, 定义一个从  $S \times S$  到  $F$  的代数运算, 记为 “ $(\cdot, \cdot)$ ”, 即对任意  $x, y \in S$ , 都存在唯一的  $f \in F$ , 使得  $f = (x, y)$ . 如果该运算

满足

- (1)  $(x, y) = \overline{(y, x)}, \quad \forall x, y \in \mathbb{S};$
- (2)  $(\alpha x, y) = \alpha(x, y), \quad \forall \alpha \in \mathbb{F}, x, y \in \mathbb{S};$
- (3)  $(x + y, z) = (x, z) + (y, z), \quad \forall x, y, z \in \mathbb{S};$
- (4)  $(x, x) \geq 0$ , 等号当且仅当  $x = 0$  时成立;

则称  $(\cdot, \cdot)$  为  $\mathbb{S}$  上的一个 **内积**, 有时也称为 **标量积**. 定义了内积的线性空间称为 **内积空间**.

定义在实数域  $\mathbb{R}$  上的内积空间称为**欧氏空间**, 定义在复数域  $\mathbb{C}$  上的内积空间称为**酉空间**.

$\overline{(u, v)}$  表示  $(u, v)$  的共轭. 当  $\mathbb{F} = \mathbb{R}$  时, 该条件即为  $(v, u) = (u, v)$ .

内积可以看作是从线性空间  $\mathbb{S}$  到数域  $\mathbb{F}$  的**二元函数**.

由内积定义可以立即得出:

- $(u, \beta v) = \bar{\beta}(u, v), \forall \beta \in \mathbb{F}$
- $(u, v + w) = (u, v) + (u, w)$
- $(u_1 + u_2 + \cdots + u_n, v) = (u_1, v) + (u_2, v) + \cdots + (u_n, v)$

**例 1.8** 考虑  $n$  维线性空间  $\mathbb{R}^n$ , 对任意  $x = [x_1, x_2, \dots, x_n]^T \in \mathbb{R}^n, y = [y_1, y_2, \dots, y_n]^T \in \mathbb{R}^n$ , 定义

$$(x, y) \triangleq y^T x = \sum_{i=1}^n x_i y_i,$$

则  $(x, y)$  是  $\mathbb{R}^n$  上的内积, 因此  $\mathbb{R}^n$  是一个欧氏空间. 这种方式定义的内积称为 **欧拉内积**.

**例 1.9** 考虑  $n$  维线性空间  $\mathbb{C}^n$ , 对任意  $x = [x_1, x_2, \dots, x_n]^T \in \mathbb{C}^n, y = [y_1, y_2, \dots, y_n]^T \in \mathbb{C}^n$ , 定义

$$(x, y) \triangleq y^* x = \sum_{i=1}^n x_i \bar{y}_i,$$

这里  $y^*$  表示  $y$  的共轭转置, 则  $(x, y)$  是  $\mathbb{C}^n$  上的内积, 因此  $\mathbb{C}^n$  是一个酉空间.

以上定义的内积是  $\mathbb{R}^n$  和  $\mathbb{C}^n$  上的标准内积, 若不特别声明,  $\mathbb{R}^n$  和  $\mathbb{C}^n$  上的内积均是指标准内积.

**例 1.10** 考虑实数域  $\mathbb{R}$  上的线性空间  $C[a, b]$ , 对任意  $f(x), g(x) \in C[a, b]$ , 定义


$$(f, g) \triangleq \int_a^b f(x)g(x) dx,$$

则  $(f, g)$  是  $C[a, b]$  上的内积, 此时  $C[a, b]$  是一个欧氏空间.

有了内积以后, 我们就可以定义正交.



**定义 1.5** 设  $\mathbb{S}$  是内积空间,  $u, v \in \mathbb{S}$ , 若  $(u, v) = 0$ , 则称  $u$  与  $v$  是**正交**的.

 正交可以看作是垂直概念的推广.

**定理 1.2** 设  $\mathbb{S}$  是内积空间,  $u_1, u_2, \dots, u_n \in \mathbb{S}$ , 则  $u_1, u_2, \dots, u_n$  线性无关的充要条件是矩阵  $G$  非奇异, 其中

$$G = \begin{bmatrix} (u_1, u_1) & (u_2, u_1) & \cdots & (u_n, u_1) \\ (u_1, u_2) & (u_2, u_2) & \cdots & (u_n, u_2) \\ \vdots & \vdots & & \vdots \\ (u_1, u_n) & (u_2, u_n) & \cdots & (u_n, u_n) \end{bmatrix}.$$

这个矩阵就称为  $u_1, u_2, \dots, u_n$  的 **Gram 矩阵**.

**证明.** 矩阵  $G$  非奇异当且仅当  $Gx = 0$  只有零解. 由  $G$  的定义可知  $Gx = 0$  即为

$$\sum_{i=1}^n (u_i, u_k) x_i = 0, \quad k = 1, 2, \dots, n,$$

即

$$0 = \sum_{i=1}^n (x_i u_i, u_k) = \left( \sum_{i=1}^n x_i u_i, u_k \right), \quad k = 1, 2, \dots, n.$$

所以

$$\left( \sum_{i=1}^n x_i u_i, \sum_{i=1}^n x_i u_i \right) = \sum_{k=1}^n \bar{x}_k \left( \sum_{i=1}^n x_i u_i, u_k \right) = 0.$$

因此  $\sum_{i=1}^n x_i u_i = 0$ . 若  $u_1, u_2, \dots, u_n$  线性无关, 则  $x_i = 0$ , 即  $Gx = 0$  只有零解.

反之, 设  $Gx = 0$  只有零解. 若  $u_1, u_2, \dots, u_n$  线性相关, 则存在一组不全为零的数  $\alpha_i$ , 使得  $\sum_{i=1}^n \alpha_i u_i = 0$ . 于是

$$0 = \left( \sum_{i=1}^n \alpha_i u_i, u_k \right) = \sum_{i=1}^n \alpha_i (u_i, u_k), \quad k = 1, 2, \dots, n.$$


因此  $x = [\alpha_1, \alpha_2, \dots, \alpha_n]^T$  是  $Gx = 0$  的非零解, 与条件矛盾. 故  $u_1, u_2, \dots, u_n$  线性无关.  $\square$

### 导出范数与带权内积

设  $\mathbb{S}$  是内积空间, 对任意  $u \in \mathbb{S}$ , 定义

$$\|u\| \triangleq (u, u)^{\frac{1}{2}},$$

则可以验证,  $\|u\|$  是  $\mathbb{S}$  上的范数. 这就是**由内积导出的范数**.

 任意一个内积都可以导出一个相应的范数.

**例 1.11**  $\mathbb{R}^n$  上由标准内积导出的范数为

$$\|x\| = (x, x)^{\frac{1}{2}} = \left( \sum_{i=1}^n x_i^2 \right)^{\frac{1}{2}}.$$

这就是 2-范数.

设  $\omega_1, \omega_2, \dots, \omega_n$  为任意给定的正实数, 对任意  $x, y \in \mathbb{R}^n$ , 定义

$$(x, y)_{\omega} \triangleq \sum_{i=1}^n \omega_i x_i y_i.$$

可以验证,  $(x, y)_{\omega}$  是  $\mathbb{R}^n$  上的内积, 我们称其为**带权内积**, 其中  $\omega_i$  称为**权系数**.

同样, 可以在  $\mathbb{C}^n$  中定义带权内积.

为了在  $C[a, b]$  上定义带权内积, 我们首先定义权函数.


**定义 1.6** 设  $\omega(x)$  是  $[a, b]$  上的非负函数, 如果  $\omega(x)$  满足

(1) 对任意非负整数  $k$ ,  $\int_a^b x^k \omega(x) dx$  存在且为有限值;

(2) 对  $[a, b]$  上的任意非负连续函数  $g(x)$ , 如果  $\int_a^b g(x) \omega(x) dx = 0$ , 则  $g(x) \equiv 0$ ;

则称  $\omega(x)$  是  $[a, b]$  上的一个**权函数**.

  $[a, b]$  可以是有限区间, 也可以是无限区间, 即  $a$  可以是  $-\infty$ ,  $b$  可以是  $\infty$ ;

 权函数与定义区间相关.

**例 1.12** 常见的权函数有

- $\omega(x) = 1$  是  $[-1, 1]$  上的权函数;
- $\omega(x) = \frac{1}{\sqrt{1-x^2}}$  是  $[-1, 1]$  上的权函数;
- $\omega(x) = e^{-x}$  是  $[0, \infty)$  上的权函数;
- $\omega(x) = e^{-x^2}$  是  $(-\infty, \infty)$  上的权函数.

**例 1.13** 设  $\omega(x)$  是  $[a, b]$  上的权函数, 对任意  $f(x), g(x) \in C[a, b]$ , 定义

$$(f, g)_{\omega} = \int_a^b \omega(x) f(x) g(x) dx,$$

则  $(f, g)_{\omega}$  是  $C[a, b]$  上的带权内积. 对应的带权内积导出范数为

$$\|f\|_{\omega} = (f, f)_{\omega}^{\frac{1}{2}} = \left( \int_a^b \omega(x) f^2(x) dx \right)^{\frac{1}{2}}.$$

特别地, 当  $\omega(x) \equiv 1$  时, 内积导出范数为

$$\|f\| = (f, f)^{\frac{1}{2}} = \left( \int_a^b f^2(x) dx \right)^{\frac{1}{2}},$$



即为  $f(x)$  的 2-范数.

根据定理 1.2, 我们可以得到下面的推论.

**推论 1.3** 设  $\varphi_1(x), \varphi_2(x), \dots, \varphi_n(x) \in C[a, b]$ , 则  $\varphi_1(x), \varphi_2(x), \dots, \varphi_n(x)$  线性无关的充要条件是矩阵  $G$  非奇异, 其中  $G$  是 Gram 矩阵:

$$G = \begin{bmatrix} (\varphi_1, \varphi_1) & (\varphi_2, \varphi_1) & \cdots & (\varphi_n, \varphi_1) \\ (\varphi_1, \varphi_2) & (\varphi_2, \varphi_2) & \cdots & (\varphi_n, \varphi_2) \\ \vdots & \vdots & & \vdots \\ (\varphi_1, \varphi_n) & (\varphi_2, \varphi_n) & \cdots & (\varphi_n, \varphi_n) \end{bmatrix}.$$

### 1.2.4 矩阵的特征值与谱半径

**定义 1.7** 设  $A \in \mathbb{C}^{n \times n}$ , 若存在  $\lambda \in \mathbb{C}$  和非零向量  $x, y \in \mathbb{C}^n$ , 使得

$$Ax = \lambda x, \quad y^* A = \lambda y^*,$$

则  $\lambda$  是  $A$  的**特征值**,  $x$  称为  $A$  对应于  $\lambda$  的(右)**特征向量**,  $y$  称为  $A$  对应于  $\lambda$  的**左特征向量**, 并称  $(\lambda, x)$  为  $A$  的一个**特征对 (eigenpair)**.

#### 关于特征值的几点说明

- 只有当  $A$  是方阵时, 才具有特征值与特征向量;
- 实矩阵的特征值与特征向量有可能是复的;
- $n$  阶矩阵总是存在  $n$  个特征值 (其中可能有相等的);
- 特征值有代数重数和几何重数, 几何重数不超过代数重数;
- 相似变换不改变矩阵的特征值, 合同变换 (congruence transformation) 不改变矩阵的惯性指数.

**思考**

$A^T$  和  $A$ , 它们的特征值和特征向量有什么关系?

$A^{-1}$  和  $A$ , 它们的特征值和特征向量有什么关系?

设  $p(t)$  是一个多项式, 则  $p(A)$  与  $A$  的特征值与特征向量有什么关系?

**定义 1.8 (谱半径)** 设  $A \in \mathbb{C}^{n \times n}$ , 则称

$$\rho(A) \triangleq \max_{\lambda \in \sigma(A)} |\lambda|$$

为  $A$  的**谱半径**, 其中  $\sigma(A)$  为  $A$  谱, 即所有特征值组成的集合:

$$\sigma(A) = \{\lambda_1, \lambda_2, \dots, \lambda_n\}.$$

**定理 1.4** 设  $A \in \mathbb{C}^{n \times n}$ , 则  $\operatorname{tr}(A) \triangleq a_{11} + a_{22} + \cdots + a_{nn}$  称为矩阵  $A$  的迹 (trace). 我们有

$$\det(A) = \lambda_1 \lambda_2 \cdots \lambda_n, \quad \operatorname{tr}(A) = \lambda_1 + \lambda_2 + \cdots + \lambda_n.$$

**定义 1.9** 设  $A \in \mathbb{C}^{n \times n}$ . 若存在一个非奇异矩阵  $X \in \mathbb{C}^{n \times n}$ , 使得

$$X^{-1}AX = \Lambda, \quad (1.1)$$

其中  $\Lambda \in \mathbb{C}^{n \times n}$  是对角矩阵. 则称  $A$  是**可对角化**的, 矩阵  $\Lambda$  的对角线元素即为  $A$  的特征值, 分解 (1.1) 称为矩阵  $A$  的**特征值分解**或**谱分解**.

### 特征值的估计

对任意矩阵  $A \in \mathbb{C}^{n \times n}$ , 我们记

$$\mathcal{H}_A \triangleq \frac{1}{2}(A + A^*), \quad \mathcal{S}_A \triangleq \frac{1}{2}(A - A^*).$$

易知

$$(\mathcal{H}_A)^* = \mathcal{H}_A, \quad (\mathcal{S}_A)^* = -\mathcal{S}_A,$$

即  $\mathcal{H}_A$  是 Hermite 的,  $\mathcal{S}_A$  是 Skew-Hermite (斜 Hermite) 的. 我们分别称它们为  $A$  的 Hermite 部分和 Skew-Hermite 部分.

**定理 1.5 (Bendixson)** 设  $A \in \mathbb{C}^{n \times n}$ , 则

$$\lambda_{\min}(\mathcal{H}_A) \leq \operatorname{Re}(\lambda(A)) \leq \lambda_{\max}(\mathcal{H}_A),$$

$$\lambda_{\min}(i\mathcal{S}_A) \leq \operatorname{Im}(\lambda(A)) \leq \lambda_{\max}(i\mathcal{S}_A),$$

其中  $\operatorname{Re}(\cdot)$  和  $\operatorname{Im}(\cdot)$  分别表示实部和虚部.

这个定理告诉我们, 一个矩阵的特征值的实部的取值范围由其 Hermite 部分确定, 而虚部则由其 Skew-Hermite 部分确定.

根据多项式零点关于多项式系数的连续性, 我们可以得到下面的结论.

**定理 1.6** 矩阵的特征值关于矩阵元素是连续的. 即当矩阵的元素发生变化时, 其特征值的变化是连续的.

设  $A = [a_{ij}] \in \mathbb{C}^{n \times n}$ , 定义集合

$$\mathcal{D}_i \triangleq \left\{ z \in \mathbb{C} : |z - a_{ii}| \leq \sum_{j=1, j \neq i}^n |a_{ij}| \right\}, \quad i = 1, 2, \dots, n. \quad (1.2)$$

这就是  $A$  的  $n$  个 **Gerschgorin 圆盘**.

**定理 1.7 (Gerschgorin 圆盘定理)** 设  $A = [a_{ij}] \in \mathbb{C}^{n \times n}$ , 则  $A$  的所有特征值都包含在  $A$  的 Ger-



schgorin 圆盘的并集中, 即  $\sigma(A) \subset \bigcup_{i=1}^n \mathcal{D}_i$ .

将  $A$  的非对角线元素换成  $\tau a_{ij}$ , 其中  $0 \leq \tau \leq 1$ , 并利用特征值关于矩阵元素的连续性, 我们就可以得到下面的结论.

**定理 1.8** 设  $A = [a_{ij}] \in \mathbb{C}^{n \times n}$ . 如果  $\bigcup_{i=1}^n \mathcal{D}_i$  可分解成两个不相交的子集  $S$  和  $T$ , 即

$$\bigcup_{i=1}^n \mathcal{D}_i = S \cup T \quad \text{且} \quad S \cap T = \emptyset.$$

假定  $S$  由  $k$  个圆盘组成, 而  $T$  由其它  $n - k$  个圆盘组成. 则  $S$  中恰好包含  $A$  的  $k$  个特征值 (重特征值按重数计算), 而  $T$  中则包含  $A$  的其它  $n - k$  个特征值.

如果  $A$  是不可约矩阵 (见定义 4.6), 则我们可以得到下面两个结论.

**定理 1.9** 设  $A \in \mathbb{C}^{n \times n}$  不可约. 如  $A$  的某个特征值  $\lambda$  在  $\bigcup_{i=1}^n \mathcal{D}_i$  的边界上, 则它必定在所有圆盘  $\mathcal{D}_i$  的边界上 (即圆周上).

**推论 1.10** 设  $A \in \mathbb{C}^{n \times n}$  不可约. 如  $\lambda$  在  $\bigcup_{i=1}^n \mathcal{D}_i$  的边界上, 但至少有一个圆盘的圆周不经过  $\lambda$ , 则它必定不是  $A$  的特征值.

## 1.2.5 向量范数与矩阵范数

### 向量范数

定义在向量空间  $\mathbb{R}^n$  或  $\mathbb{C}^n$  上的范数就是**向量范数**. 为简单起见, 我们这里只考虑实数域情形, 复数域情形完全类似.

**定义 1.10 (向量范数)** 若函数  $f(x) : \mathbb{R}^n \rightarrow \mathbb{R}$  满足

- (1)  $f(x) \geq 0, \forall x \in \mathbb{R}^n$  且等号当且仅当  $x = 0$  时成立; (非负性, nonnegativity)
- (2)  $f(\alpha x) = |\alpha| \cdot f(x), \forall x \in \mathbb{R}^n, \alpha \in \mathbb{R}$  (正齐次性, homogeneity)
- (3)  $f(x + y) \leq f(x) + f(y), \forall x, y \in \mathbb{R}^n$ ; (三角不等式, triangular inequality)

则称  $f(x)$  为  $\mathbb{R}^n$  上的**范数 (norm)**, 通常记作  $\|x\|$ .

$\mathbb{R}^n$  上常见的范数有 1-范数, 2-范数,  $p$ -范数和  $\infty$ -范数, 见例 1.6.

下面给出向量范数的一些重要性质.

**定理 1.11 (向量范数的连续性)** 设  $\|\cdot\|$  是  $\mathbb{R}^n$  上的一个向量范数, 则  $f(x) \triangleq \|x\|$  关于  $x$  的每个分量是连续的.

**定义 1.11 (范数的等价性)** 设  $\|\cdot\|_\alpha$  与  $\|\cdot\|_\beta$  是  $\mathbb{R}^n$  空间上的两个向量范数, 若存在正常数  $c_1, c_2$ , 使得

$$c_1\|x\|_\alpha \leq \|x\|_\beta \leq c_2\|x\|_\alpha$$

对任意  $x \in \mathbb{R}^n$  都成立, 则称  $\|\cdot\|_\alpha$  与  $\|\cdot\|_\beta$  是等价的.


**定理 1.12**  $\mathbb{R}^n$  上的所有向量范数都是等价的, 特别地, 有

$$\|x\|_2 \leq \|x\|_1 \leq \sqrt{n} \|x\|_2,$$

$$\|x\|_\infty \leq \|x\|_1 \leq n \|x\|_\infty,$$

$$\|x\|_\infty \leq \|x\|_2 \leq \sqrt{n} \|x\|_\infty.$$

(证明留作练习)

 事实上, 有限维赋范线性空间上的所有范数都是等价的.

**定理 1.13 (Cauchy-Schwartz 不等式)** 设  $(\cdot, \cdot)$  是  $\mathbb{R}^n$  上的内积, 则对任意  $x, y \in \mathbb{R}^n$ , 有

$$|(x, y)| \leq \|x\|_2 \cdot \|y\|_2,$$

且等号成立的充要条件是  $x$  与  $y$  线性相关.

**证明.** 若  $y = 0$ , 则结论显然成立.

假设  $y \neq 0$ , 则对任意  $\alpha \in \mathbb{R}$  有

$$0 \leq (x - \alpha y, x - \alpha y) = (x, x) - 2\alpha(x, y) + \alpha^2(y, y).$$

由于  $y \neq 0$ , 所以  $(y, y) > 0$ . 取  $\alpha = \frac{(x, y)}{(y, y)}$ , 代入上式可得

$$0 \leq (x, x) - \frac{|(x, y)|^2}{(y, y)},$$

即

$$|(x, y)|^2 \leq (x, x) \cdot (y, y) = \|x\|_2^2 \cdot \|y\|_2^2.$$

下面考虑等号成立的条件.

充分性: 如果  $x$  与  $y$  线性相关, 则通过直接验证即可知等号成立.

必要性: 假设等号成立. 如果  $y = 0$ , 则显然  $x$  与  $y$  线性相关. 现假定  $y \neq 0$ . 取  $\alpha = \frac{(x, y)}{(y, y)}$ , 则

$$(x - \alpha y, x - \alpha y) = (x, x) - \frac{|(x, y)|^2}{(y, y)} = 0,$$

即  $x - \alpha y = 0$ . 所以  $x$  与  $y$  线性相关. □

更一般地, 我们有下面的 Holder 不等式.





**定理 1.14 (Holder 不等式)** 设  $(\cdot, \cdot)$  是  $\mathbb{R}^n$  上的内积, 则对任意  $x, y \in \mathbb{R}^n$ , 有

$$|(x, y)| \leq \|x\|_p \cdot \|y\|_q,$$

其中  $p, q > 0$ , 且  $\frac{1}{p} + \frac{1}{q} = 1$ .

## 矩阵范数

**定义 1.12 (矩阵范数)** 若函数  $f(X) : \mathbb{R}^{n \times n} \rightarrow \mathbb{R}$  满足

- (1)  $f(A) \geq 0, \forall A \in \mathbb{R}^{n \times n}$  且等号当且仅当  $A = 0$  时成立;
- (2)  $f(\alpha A) = |\alpha| \cdot f(A), \forall A \in \mathbb{R}^{n \times n}, \alpha \in \mathbb{R}$ ;
- (3)  $f(A + B) \leq f(A) + f(B), \forall A, B \in \mathbb{R}^{n \times n}$ ;
- (4)  $f(AB) \leq f(A)f(B), \forall A, B \in \mathbb{R}^{n \times n}$

则称  $f(X)$  为  $\mathbb{R}^{n \times n}$  上的矩阵范数, 通常记作  $\|X\|$ .

在矩阵范数的定义中, 条件 (4) 称为**相容性条件**. 在有的教材中, 矩阵范数只需满足条件 (1), (2), (3), 并称满足条件 (4) 的矩阵范数为**相容矩阵范数**. 这样的定义与普通范数保持一致. 但在实际使用中, 有用的矩阵范数基本上都要满足相容性, 所以也通常把矩阵范数直接定义成相容矩阵范数. 如果没有特别说明, 本讲义中矩阵范数就是指相容矩阵范数.

类似地, 我们可以定义  $\mathbb{R}^{m \times n}$  和  $\mathbb{C}^{m \times n}$  上的矩阵范数 (相容性条件会复杂一些).

一类常用的矩阵范数就是由向量范数导出的算子范数.

**引理 1.15 (算子范数, 诱导范数, 导出范数)** 设  $\|\cdot\|$  是  $\mathbb{R}^n$  上的向量范数, 则

$$\|A\| \triangleq \sup_{x \in \mathbb{R}^n, x \neq 0} \frac{\|Ax\|}{\|x\|} = \max_{\|x\|=1} \|Ax\|$$

是  $\mathbb{R}^{n \times n}$  上的范数, 称为**算子范数**, 也称为**诱导范数**或**导出范数**.

对于算子范数, 我们有下面的性质: 设  $A \in \mathbb{R}^{n \times n}, x \in \mathbb{R}^n$ , 则

$$\|Ax\| \leq \|A\| \cdot \|x\|.$$

如果没有特别指出哪类范数, 对向量和矩阵使用同样的范数时, 默认是指算子范数.

**例 1.14**  $\mathbb{R}^{n \times n}$  上常见的矩阵范数:

- $F$ -范数

$$\|A\|_F = \sqrt{\sum_{i=1}^n \sum_{j=1}^n |a_{ij}|^2};$$

- $p$ -范数 (算子范数)

$$\|A\|_p = \sup_{x \neq 0} \frac{\|Ax\|_p}{\|x\|_p}, \quad p = 1, 2, \infty.$$

**引理 1.16** 可以证明:

- (1) 1-范数 (也称为 **列范数**):  $\|A\|_1 = \max_{1 \leq j \leq n} \left( \sum_{i=1}^n |a_{ij}| \right);$
- (2)  $\infty$ -范数 (也称为 **行范数**):  $\|A\|_\infty = \max_{1 \leq i \leq n} \left( \sum_{j=1}^n |a_{ij}| \right);$
- (3) 2-范数 (也称为 **谱范数**):  $\|A\|_2 = \sqrt{\rho(A^T A)}.$

**例 1.15** 设  $A = \begin{bmatrix} 1 & -2 \\ -3 & 4 \end{bmatrix}$ , 计算  $\|A\|_1, \|A\|_2, \|A\|_\infty, \|A\|_F$ .

(板书)

计算 2-范数时需要谱半径, 因此通常比计算 1-范数和  $\infty$ -范数更困难. 但在某些情况下可以用下面的范数等价性来估计一个矩阵的 2-范数.

**定理 1.17 (矩阵范数的等价性)**  $\mathbb{R}^{n \times n}$  空间上的所有范数都是等价的, 特别地, 有

$$\begin{aligned} \frac{1}{\sqrt{n}} \|A\|_1 &\leq \|A\|_2 \leq \sqrt{n} \|A\|_1, \\ \frac{1}{\sqrt{n}} \|A\|_\infty &\leq \|A\|_2 \leq \sqrt{n} \|A\|_\infty, \\ \frac{1}{n} \|A\|_\infty &\leq \|A\|_1 \leq n \|A\|_\infty. \end{aligned}$$

(板书)

**定理 1.18** 若  $A$  是对称矩阵, 则  $\|A\|_2 = \rho(A)$ .

(证明留作练习)

**定理 1.19 (矩阵范数的连续性)** 设  $\|\cdot\|$  是  $\mathbb{R}^{n \times n}$  上的一个矩阵范数, 则  $f(A) \triangleq \|A\|$  关于  $A$  的每个分量是连续的.

(证明留作练习)

**定理 1.20** 设  $\|\cdot\|$  是  $\mathbb{R}^{n \times n}$  上的任一算子范数, 若  $\|B\| < 1$ , 则  $I \pm B$  非奇异, 且

$$\|(I \pm B)^{-1}\| \leq \frac{1}{1 - \|B\|}.$$

**证明.** 这里仅证明减号情形, 加号情形的证明类似.



先证明  $I - B$  非奇异. 反证法, 假定  $I - B$  奇异, 则  $(I - B)x = 0$  有非零解, 设为  $\tilde{x}$ , 即  $B\tilde{x} = \tilde{x}$ . 所以  $\|B\| \geq \frac{\|B\tilde{x}\|}{\|\tilde{x}\|} = 1$ , 与条件  $\|B\| < 1$  矛盾. 因此  $I - B$  非奇异.

由  $(I - B)^{-1}(I - B) = I$  可知

$$(I - B)^{-1} = I + (I - B)^{-1}B.$$

从而

$$\|(I - B)^{-1}\| = \|I + (I - B)^{-1}B\| \leq \|I\| + \|(I - B)^{-1}\| \cdot \|B\|.$$

解得

$$\|(I - B)^{-1}\| \leq \frac{\|I\|}{1 - \|B\|} = \frac{1}{1 - \|B\|}.$$

□

### 矩阵范数的其他性质

- (1) 对任意算子范数  $\|\cdot\|$ , 有  $\|A^k\| \leq \|A\|^k$ ;
- (2) 对任意算子范数  $\|\cdot\|$ , 有  $\|Ax\| \leq \|A\| \cdot \|x\|$ ,  $\|AB\| \leq \|A\| \cdot \|B\|$ ;
- (3)  $\|Ax\|_2 \leq \|A\|_F \cdot \|x\|_2$ ,  $\|AB\|_F \leq \|A\|_F \cdot \|B\|_F$ ;
- (4)  $F$ -范数不是算子范数;
- (5)  $\|\cdot\|_2$  和  $\|\cdot\|_F$  是 **酉不变范数**, 即对任意酉矩阵 (或正交矩阵)  $U, V$ , 有

$$\|UA\|_2 = \|AV\|_2 = \|UAV\|_2 = \|A\|_2,$$

$$\|UA\|_F = \|AV\|_F = \|UAV\|_F = \|A\|_F$$

- (6)  $\|A^T\|_2 = \|A\|_2$ ,  $\|A^T\|_1 = \|A\|_\infty$ .


(证明留作课外练习)

### 谱半径与范数之间的关系

**定理 1.21 (谱半径与范数的关系)** 设  $A \in \mathbb{C}^{n \times n}$ , 则

- (1) 对任意算子范数, 有  $\rho(A) \leq \|A\|$ ;
- (2) 反之, 对任意  $\varepsilon > 0$ , 都存在一个算子范数  $\|\cdot\|_\varepsilon$ , 使得  $\|A\|_\varepsilon \leq \rho(A) + \varepsilon$ , 其中范数  $\|\cdot\|_\varepsilon$  依赖于  $A$  和  $\varepsilon$ . 所以, 若  $\rho(A) < 1$ , 则存在算子范数  $\|\cdot\|_\varepsilon$ , 使得  $\|A\|_\varepsilon < 1$ ;

(板书, 只证明结论 (1), 结论 (2) 可查询相关资料)

 事实上, 上述定理中的结论 (1) 对任意矩阵范数都成立.

**推论 1.22** 设  $A \in \mathbb{R}^{n \times n}$ , 则  $\|A\|_2^2 \leq \|A\|_1 \cdot \|A\|_\infty$ , 且

$$\max_{1 \leq i, j \leq n} \{|a_{ij}|\} \leq \|A\|_2 \leq n \max_{1 \leq i, j \leq n} \{|a_{ij}|\}.$$

(证明留作练习)

下面给出谱半径与范数之间的一个非常重要的性质, 有时也用来定义矩阵的谱半径.

**引理 1.23** 设  $G \in \mathbb{R}^{n \times n}$ , 则对任意矩阵范数  $\|\cdot\|$ , 有

$$\rho(G) = \lim_{k \rightarrow \infty} \|G^k\|^{\frac{1}{k}}.$$

## 1.2.6 矩阵标准型

### Jordan 标准型

在计算矩阵的特征值时, 一个基本的思想是通过相似变换, 将其转化成一个形式尽可能简单的矩阵, 使得其特征值更易于计算. 在这里, 我们介绍两个非常重要的矩阵标准型: **Jordan 标准型**和 **Schur 标准型**.


**定理 1.24** 设  $A \in \mathbb{C}^{n \times n}$  有  $p$  个不同的特征值, 则存在非奇异矩阵  $X \in \mathbb{C}^{n \times n}$ , 使得

$$X^{-1}AX = \begin{bmatrix} J_1 & & \\ & J_2 & \\ & & \ddots \\ & & & J_q \end{bmatrix} \triangleq J, \quad (1.3)$$

其中  $J_i$  的维数等于  $\lambda_i$  的代数重数, 且具有下面的结构

$$J_i = \begin{bmatrix} J_{i1} & & \\ & J_{i2} & \\ & & \ddots \\ & & & J_{i\nu_i} \end{bmatrix}, \quad J_{ik} = \begin{bmatrix} \lambda_i & 1 & & \\ & \ddots & \ddots & \\ & & \lambda_i & 1 \\ & & & \lambda_i \end{bmatrix}.$$

这里的  $\nu_i$  为  $\lambda_i$  的几何重数,  $J_{ik}$  为 (对应于  $\lambda_i$  的) **Jordan 块**.

 块对角矩阵  $J$  就称为  $A$  的 Jordan 标准型, 除了其中的 Jordan 块排列次序外是唯一确定的.

易知, 对于每一个 Jordan 块  $J_{ik}$ , 都存在一个列满秩矩阵  $X_{ik}$  使得

$$AX_{ik} = X_{ik}J_{ik}.$$

除此之外, Jordan 标准型还具有以下性质:

- Jordan 块的个数等于  $A$  的线性无关的特征向量的个数;
- $A$  可对角化的充要条件是每个 Jordan 块都是  $1 \times 1$  的, 此时  $X$  的列向量就是  $A$  的特征向量.

**推论 1.25** 所有可对角化矩阵组成的集合在所有矩阵组成的集合中是稠密的.

### Schur 标准型

Jordan 标准型在理论研究中非常有用, 但数值计算比较困难, 目前还没有找到十分稳定有效的数值计算方法. 下面我们介绍一个比较实用的标准型: **Schur 标准型**.



**定理 1.26** 设  $A \in \mathbb{C}^{n \times n}$ , 则存在酉矩阵  $U \in \mathbb{C}^{n \times n}$  使得

$$U^*AU = \begin{bmatrix} \lambda_1 & r_{12} & \cdots & r_{1n} \\ 0 & \lambda_2 & \cdots & r_{2n} \\ \vdots & & \ddots & \vdots \\ 0 & \cdots & 0 & \lambda_n \end{bmatrix} \triangleq R \quad \text{或} \quad A = URU^*, \quad (1.4)$$

其中  $\lambda_1, \lambda_2, \dots, \lambda_n$  是  $A$  的特征值 (可以按任意顺序排列).

**证明.** 我们对  $n$  使用归纳法.

当  $n = 1$  时, 结论显然成立.

假设结论对阶数为  $n - 1$  的矩阵都成立. 考虑  $n$  阶矩阵  $A \in \mathbb{C}^{n \times n}$ . 设  $\lambda$  是  $A$  的一个特征值, 其对应的特征向量为  $x \in \mathbb{C}^n$ , 且  $\|x\|_2 = 1$ . 构造一个以  $x$  为第一列的酉矩阵  $X = [x, \tilde{X}]$ . 于是

$$X^*AX = \begin{bmatrix} x^* \\ \tilde{X}^* \end{bmatrix} A \begin{bmatrix} x & \tilde{X} \end{bmatrix} = \begin{bmatrix} x^*Ax & x^*A\tilde{X} \\ \tilde{X}^*Ax & \tilde{X}^*A\tilde{X} \end{bmatrix}.$$

因为  $x^*Ax = \lambda x^*x = \lambda$ , 且  $\tilde{X}^*Ax = \tilde{X}^*(\lambda x) = \lambda \tilde{X}^*x = 0$ , 故

$$X^*AX = \begin{bmatrix} \lambda & x^*A\tilde{X} \\ 0 & \tilde{X}^*A\tilde{X} \end{bmatrix} \triangleq \begin{bmatrix} \lambda & \tilde{A}_{12} \\ 0 & \tilde{A}_{22} \end{bmatrix},$$

其中  $\tilde{A}_{22} \in \mathbb{C}^{(n-1) \times (n-1)}$ . 根据归纳假设, 存在酉矩阵  $\tilde{U} \in \mathbb{C}^{(n-1) \times (n-1)}$ , 使得  $\tilde{U}^*\tilde{A}_{22}\tilde{U} = \tilde{R} \in \mathbb{C}^{(n-1) \times (n-1)}$  是一个上三角矩阵. 令

$$U = X \begin{bmatrix} 1 & 0 \\ 0 & \tilde{U} \end{bmatrix},$$

则有

$$\begin{aligned} U^*AU &= \begin{bmatrix} 1 & 0 \\ 0 & \tilde{U}^* \end{bmatrix} X^*AX \begin{bmatrix} 1 & 0 \\ 0 & \tilde{U} \end{bmatrix} \\ &= \begin{bmatrix} 1 & 0 \\ 0 & \tilde{U}^* \end{bmatrix} \begin{bmatrix} \lambda & \tilde{A}_{12} \\ 0 & \tilde{A}_{22} \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & \tilde{U} \end{bmatrix} = \begin{bmatrix} \lambda & \tilde{A}_{12}\tilde{U} \\ 0 & \tilde{U}^*\tilde{A}_{22}\tilde{U} \end{bmatrix} = \begin{bmatrix} \lambda & \tilde{A}_{12}\tilde{U} \\ 0 & \tilde{R} \end{bmatrix} \triangleq R. \end{aligned}$$

由于  $\tilde{R}$  是上三角矩阵, 故  $R$  也是一个上三角矩阵, 其对角线元素即为  $A$  的特征值.

由归纳法可知, 定理结论成立. □

### 关于 Schur 标准型的几点说明

- Schur 标准型可以说是酉相似变化下的最简形式;
- 定理中的  $U$  和  $R$  不是唯一的, 其中  $R$  的对角线元素可以按任意顺序排列.

**推论 1.27** 设  $A \in \mathbb{C}^{n \times n}$ , 则

- $A$  是正规矩阵当且仅当 (1.4) 中的  $R$  是对角矩阵, 即  $A$  可酉对角化的充要条件是  $A$  是正规矩阵;

- $A$  是 Hermite 矩阵当且仅当 (1.4) 中的  $R$  是实对角矩阵.

众所周知, 当  $A$  是实矩阵时, 其特征值和特征向量仍可能是复的. 但在实际应用中, 我们在求解一个实矩阵的特征值问题时, 希望尽可能地避免复数运算. 此时, 我们就需要用到下面的 **Schur 标准型** (或拟 **Schur 标准型**).

**定理 1.28** 设  $A \in \mathbb{R}^{n \times n}$ , 则存在正交矩阵  $Q \in \mathbb{R}^{n \times n}$ , 使得

$$Q^T A Q = T, \quad (1.5)$$

其中  $T \in \mathbb{R}^{n \times n}$  是 **拟上三角矩阵**, 即  $T$  是块上三角的, 且对角块为  $1 \times 1$  或  $2 \times 2$  的块矩阵. 若对角块是  $1 \times 1$  的, 则其就是  $A$  的一个特征值, 若对角块是  $2 \times 2$  的, 则其特征值是  $A$  的一对共轭复特征值.

(证明可参见相关资料, 如 [21])

## 1.2.7 对称正定矩阵

**定义 1.13** 设  $A \in \mathbb{C}^{n \times n}$ .

- 若对所有向量  $x \in \mathbb{C}^n$  有  $\operatorname{Re}(x^* A x) \geq 0$ , 则称  $A$  是 **半正定**的;
- 若对所有非零向量  $x \in \mathbb{C}^n$  有  $\operatorname{Re}(x^* A x) > 0$ , 则称  $A$  是 **正定**的;
- 若  $A$  是 Hermite 的且半正定, 则称  $A$  为 **Hermite 半正定**;
- 若  $A$  是 Hermite 的且正定, 则称  $A$  为 **Hermite 正定**;
- 若  $A \in \mathbb{R}^{n \times n}$  是对称的且半正定, 则称  $A$  为 **对称半正定**;
- 若  $A \in \mathbb{R}^{n \times n}$  是对称的且正定, 则称  $A$  为 **对称正定**.

✎ 若对所有向量  $x \in \mathbb{C}^n$  有  $x^* A x \in \mathbb{R}$ , 则  $A^* = A$ .

✎ 本讲义中, 正定和半正定矩阵不要求是对称的或 Hermite 的.

**定理 1.29** 设  $A \in \mathbb{C}^{n \times n}$ . 则  $A$  正定 (或半正定) 的充要条件是矩阵  $H = \frac{1}{2}(A + A^*)$  正定 (或半正定).

(证明留作练习)

**定理 1.30** 设  $A \in \mathbb{R}^{n \times n}$ . 则  $A$  正定 (或半正定) 的充要条件是对任意非零向量  $x \in \mathbb{R}^n$  有  $x^T A x > 0$  (或  $x^T A x \geq 0$ ).

(证明留作练习)

如果  $A$  是 Hermite (半) 正定矩阵, 则可以定义其平方根, 即存在唯一的 Hermite (半) 正定矩阵  $B$ , 使得  $B^2 = A$ . 事实上, 我们有下面更一般的性质.



**定理 1.31** 设  $A \in \mathbb{C}^{n \times n}$  是一个 Hermite 半正定矩阵,  $k$  是一个给定的正整数. 则存在一个唯一的 Hermite 半正定矩阵  $B \in \mathbb{C}^{n \times n}$  使得

$$B^k = A.$$

同时, 我们还有下面的性质:

- (1)  $BA = AB$ , 且存在一个多项式  $p(t)$  使得  $B = p(A)$ ;
- (2)  $\text{rank}(B) = \text{rank}(A)$ , 因此, 若  $A$  是正定的, 则  $B$  也正定;
- (3) 如果  $A$  是实矩阵的, 则  $B$  也是实矩阵.

特别地, 当  $k = 2$  时, 称  $B$  为  $A$  的平方根, 记为  $A^{\frac{1}{2}}$ .

(证明可参见相关资料, 如 [31])

### 1.3 数值计算中的误差

数值方法的特点之一就是所求得解是近似解,总是存在一定的误差.因此,误差分析是数值分析中一个很重要的课题.

**误差**是人们用来描述数值计算中近似解的精确程度,是科学计算中的一个十分重要的概念.科学计算中误差的来源主要有以下几个方面 [63]:

- **模型误差**: 从实际问题中抽象出数学模型,往往是抓住主要因素,忽略次要因素,因此,数学模型与实际之间总会存在一定的误差.
- **观测误差**: 模型中往往包含各种数据或参量,这些数据一般都是通过测量和实验得到的,也会存在一定的误差.
- **截断误差**: 也称**方法误差**,是指对数学模型进行数值求解时产生的误差.
- **舍入误差**: 由于计算机的机器字长有限,做算术运算时存在一定的精度限制,也会产生误差.

在数值分析中,我们总假定数学模型是准确的,因而不考虑模型误差和观测误差,主要研究截断误差和舍入误差对计算结果的影响.

**例 1.16** 近似计算  $\int_0^1 e^{-x^2} dx$  的值.

**解.** 这里我们利用 Taylor 展开, 即

$$\begin{aligned}\int_0^1 e^{-x^2} dx &= \int_0^1 \left( x - x^2 + \frac{x^4}{2!} - \frac{x^6}{3!} + \frac{x^8}{4!} - \cdots \right) dx \\ &= 1 - \frac{1}{3} + \frac{1}{2!} \times \frac{1}{5} - \frac{1}{3!} \times \frac{1}{7} + \frac{1}{4!} \times \frac{1}{9} - \cdots \\ &\triangleq S_4 + R_4,\end{aligned}$$

其中  $S_4$  为前四项的部分和,  $R_4$  为剩余部分. 如果我们以  $S_4$  作为定积分的近似值, 则  $R_4$  就是由此而产生的误差, 这种误差就称为截断误差, 它是由我们的近似方法所造成的.

在计算  $S_4$  的值时, 假定我们保留小数点后 4 位有效数字, 则

$$S_4 = 1 - \frac{1}{3} + \frac{1}{10} - \frac{1}{42} \approx 1.00000 - 0.33333 + 0.10000 - 0.023810 \approx 0.7429$$

这就是我们最后得到的近似值. 这里, 在计算  $S_4$  时所产生的误差就是舍入误差. □

#### 1.3.1 绝对误差和绝对误差限

**定义 1.14** 设  $\tilde{x}$  是  $x$  的近似值, 则称

$$\epsilon \triangleq \tilde{x} - x$$

为近似值  $\tilde{x}$  的**绝对误差**, 简称**误差**.

若存在  $\varepsilon > 0$  使得

$$|\epsilon| = |\tilde{x} - x| \leq \varepsilon,$$

则称  $\varepsilon$  为**绝对误差限**, 简称**误差限**.





在工程中, 通常用  $x = \tilde{x} \pm \varepsilon$  表示  $\tilde{x}$  的误差限为  $\varepsilon$ .

### 关于误差和误差限的几点说明

- 绝对误差不是误差的绝对值, 可能是正的, 也可能是负的.
- 由于精确值通常是不知道的, 因此绝对误差一般也是不可知的.
- 在做误差估计时, 我们所求的通常是误差限;
- 误差限不唯一, 越小越好, 一般是指所能找到的最小上界;
- 近似值的精确程度不能仅仅看绝对误差, 更重要的是看相对误差.

### 1.3.2 相对误差和相对误差限

**定义 1.15** 设  $\tilde{x}$  是  $x$  的近似值, 称

$$\epsilon_r \triangleq \frac{\tilde{x} - x}{x}$$

为近似值  $\tilde{x}$  的 **相对误差**.

若存在  $\varepsilon_r > 0$  使得

$$|\epsilon_r| \leq \varepsilon_r,$$

则称  $\varepsilon_r$  为 **相对误差限**.

由于精确值  $x$  通常难以求出, 因此我们往往采用下面的定义

$$\epsilon_r \triangleq \frac{\tilde{x} - x}{\tilde{x}}.$$

因此有时也经常写成  $x = \tilde{x}(1 - \varepsilon_r)$  或  $\tilde{x} = x(1 + \varepsilon_r)$ .

- 近似值的精确程度通常取决于**相对误差**的大小;
- 实际计算中我们所能得到的通常是相对误差限 (所能找到的最小上界);
- 绝对误差有量纲, 但相对误差没有.

### 1.3.3 有效数字

我们知道, 在取一个浮点数的近似值时, 为了就尽可能地精确, 一个基本原则是“四舍五入”, 这样所得到的数字都是有效数字.

**例 1.17** 已知精确值  $\pi = 3.14159265 \cdots$ , 则近似值  $x_1 = 3.14$  有 3 位有效数字, 近似值  $x_2 = 3.1416$  有 5 位有效数字.

从上面的例子中可以看出, 我们在计算有效数字的个数时, 是从最小的有效数字位开始往前数, 直至第一个非零数字为止, 如果总共有  $n$  个数字, 那么我们就称其有  $n$  位有效数字.

关于有效数字的判断, 我们可以使用下面的方法.

**定理 1.32** 设  $\tilde{x}$  是  $x$  的近似值, 若  $\tilde{x}$  可表示为

$$\tilde{x} = \pm a_1.a_2 \cdots a_n \cdots \times 10^m,$$

其中  $a_i$  是 0 到 9 中的数字, 且  $a_1 \neq 0$ . 若

$$0.5 \times 10^{m-n} < |\tilde{x} - x| \leq 0.5 \times 10^{m-n+1},$$

则  $\tilde{x}$  恰好有  $n$  位有效数字.

☞ 换言之, 若  $0.5 \times 10^{k-1} < |\tilde{x} - x| \leq 0.5 \times 10^k$ , 则  $\tilde{x}$  恰好有  $m - k + 1$  位有效数字.

在上面的例子 1.17 中,  $x_1$  可写为  $x_1 = 3.14 \times 10^0$ , 其与  $\pi$  的误差满足

$$0.5 \times 10^{-3} < |x_1 - \pi| \leq 0.5 \times 10^{-2},$$

所以  $m = 0, k = -2$ , 因此  $x_1$  具有  $m - k + 1 = 3$  位有效数字.

**例 1.18** 根据四舍五入原则写出下列各数的具有 5 位有效数字的近似值:

$$187.9325, \quad 0.03785551, \quad 8.000033.$$

☞ 按四舍五入原则得到的数字都是有效数字.

☞ 一个数末尾的 0 不可以随意添加或省略.

由于在实际计算中, 我们往往只知道误差的某个上限, 根据这个上限, 我们一般只能判断这个近似值所具有的有效数字的最少个数, 即

**推论 1.33** 设  $\tilde{x}$  是  $x$  的近似值, 若  $\tilde{x}$  可表示为

$$\tilde{x} = \pm a_1.a_2 \cdots a_n \cdots \times 10^m,$$

其中  $a_i$  是 0 到 9 中的数字, 且  $a_1 \neq 0$ . 若

$$|\tilde{x} - x| \leq 0.5 \times 10^{m-n+1},$$

则  $\tilde{x}$  至少有  $n$  位有效数字.

**例 1.19** 已知精确值  $x = 2.7182818 \cdots$ , 则近似值  $x_1 = 2.7182$  和  $x_2 = 2.7183$  分别有几位有效数字?

### 有效数字与相对误差

相对误差可以衡量一个近似值的精确程度, 而有效数字也可以评价一个近似值的精确程度, 两者之间有下面的关系.

**定理 1.34 (有效数字与相对误差限)** 设  $\tilde{x}$  是  $x$  的近似值, 若  $\tilde{x}$  可表示为

$$\tilde{x} = \pm a_1.a_2 \cdots a_n \cdots \times 10^m,$$



其中  $a_i$  是 0 到 9 中的数字, 且  $a_1 \neq 0$ . 若  $\tilde{x}$  具有  $n$  位有效数字, 则其相对误差满足

$$|\epsilon_r| \leq \frac{1}{2a_1} \times 10^{-n+1}.$$

反之, 若  $\tilde{x}$  的相对误差满足

$$|\epsilon_r| \leq \frac{1}{2(a_1 + 1)} \times 10^{-n+1}.$$

则  $\tilde{x}$  至少有  $n$  位有效数字.

**证明.** 由  $\tilde{x}$  的表达式可知

$$a_1 \times 10^m \leq |\tilde{x}| \leq (a_1 + 1) \times 10^m.$$

若  $\tilde{x}$  具有  $n$  位有效数字, 则

$$|\epsilon_r| = \frac{|\tilde{x} - x|}{|\tilde{x}|} \leq \frac{0.5 \times 10^{m-n+1}}{a_1 \times 10^m} = \frac{1}{2a_1} \times 10^{-n+1}.$$

反之, 若  $|\epsilon_r| \leq \frac{1}{2(a_1+1)} \times 10^{-n+1}$ , 则

$$|\tilde{x} - x| = |\tilde{x}| \cdot |\epsilon_r| \leq 0.5 \times 10^{m-n+1}.$$

故  $\tilde{x}$  至少有  $n$  位有效数字. □

从这个定理可以看出, 有效数字越多, 相对误差越小. 同样, 如果相对误差越小, 则有效数字越多.

### 拓展阅读

1991 年 2 月 25 日, 在海湾战争中, 伊拉克的飞毛腿导弹打中了美军在沙特的某军事基地, 而当时美军部署的爱国者导弹未能及时拦截, 结果造成 28 名美军士兵阵亡, 这是海湾战争中美军最大的伤亡。造成这起事故的原因是爱国者导弹系统时钟内的一个软件错误, 而这个错误就是由浮点数的舍入误差引起的。

当时, 爱国者导弹系统计算时间的最小单位是  $1/10$  秒。在二进制中,  $1/10$  表示成

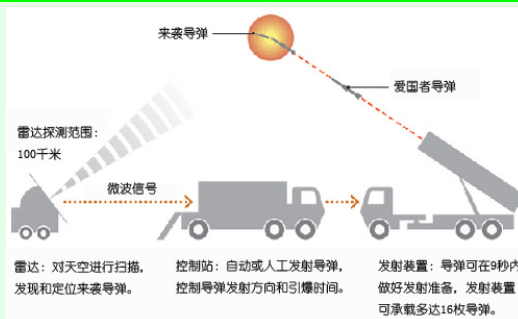
$$0.00011001100110011001100110011001100 \dots$$

因为系统只能支持 24 位, 所以实际使用的是

$$0.000110011001100110011001100$$

因此每 1 秒大约有 0.0001% 的误差 (绝对误差大约为  $10^{-7}$ )。这误差看起来很小, 但累积起来却成了一个致命错误。

美军当时一共部署了两套爱国者系统, 其中一套由于雷达故障正在维修, 而另外一套持续工作了近 100 个小时, 累积的时间误差达到了 0.34 秒。按照飞毛腿导弹的飞行速度 (约  $2000\text{m/s}$ ) 换算, 大概是 600 多米的距离。结果当飞毛腿导弹打过来的时候, 爱国者以为这是一次误报, 没有拦截, 由此造成悲剧。



事实上,以色列方面已经发现了该问题,并于1991年2月11日知会了美军和爱国者导弹软件制造商,同时建议重新启动爱国者系统作为临时解决方案,但美军却不知道每次重启系统的间隔时间是多少。1991年2月16日,制造商更新了软件,包括升级系统到64位,改进计算方法等,但这个软件却在飞毛腿导弹击中军营后一天才运抵部队。

### 拓展阅读

Ariane 系列航天运载火箭是1973年7月由法国提议,并联合西欧11个国家成立的欧洲空间局着手研制的,目前在国际航天市场中占有重要地位。在1996年6月4日的Ariane 5火箭的首飞中,发射后40秒不到就偏离预定轨道而炸毁,原因是软件系统在将64位浮点数转换为16位整数时,浮点数超过16位整数所能表示的范围,发生溢出错误,从而造成这次发射失败。

#### Explosion of Ariane 5

On June 4, 1996 an unmanned Ariane 5 rocket launched by the European Space Agency exploded just forty seconds after lift-off

- The cause was a software error in the inertial reference system
- A 64-bit floating point number was meant to be converted to a 16-bit signed integer
- The number was larger than 32,678, which is the largest storable number for 16-bit, so the conversion failed
- The development of the rocket cost over \$7 billion
- The rocket and its cargo cost \$500 million



### 1.3.4 误差估计的基本方法

再次强调:在做误差估计时,通常指的是估计绝对误差限或相对误差限。

#### 四则运算

设  $\tilde{x}_1$  和  $\tilde{x}_2$  的误差限分别为  $\varepsilon(\tilde{x}_1)$  和  $\varepsilon(\tilde{x}_2)$ , 则

$$\varepsilon(\tilde{x}_1 \pm \tilde{x}_2) \leq \varepsilon(\tilde{x}_1) + \varepsilon(\tilde{x}_2),$$

$$\varepsilon(\tilde{x}_1 \tilde{x}_2) \leq |\tilde{x}_2| \varepsilon(\tilde{x}_1) + |\tilde{x}_1| \varepsilon(\tilde{x}_2) + \varepsilon(\tilde{x}_1) \varepsilon(\tilde{x}_2) \lesssim |\tilde{x}_2| \varepsilon(\tilde{x}_1) + |\tilde{x}_1| \varepsilon(\tilde{x}_2),$$

$$\varepsilon\left(\frac{\tilde{x}_1}{\tilde{x}_2}\right) \lesssim \frac{|\tilde{x}_2| \varepsilon(\tilde{x}_1) + |\tilde{x}_1| \varepsilon(\tilde{x}_2)}{|\tilde{x}_2|^2}.$$

#### 单变量可微函数

一般地, 设  $\tilde{x}$  是  $x$  的近似值, 若  $f(x)$  可导, 则有

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

由于  $\tilde{x} - x$  相对较小, 所以当  $|f''(x)|$  与  $|f'(x)|$  的比值不是很大时, 我们可以忽略二阶项, 即

$$|f(\tilde{x}) - f(x)| \approx |f'(x)| \cdot |\tilde{x} - x|.$$

因此, 可得函数值的误差限

$$\varepsilon(f(\tilde{x})) \approx |f'(x)| \varepsilon(\tilde{x}) \approx |f'(\tilde{x})| \varepsilon(\tilde{x}).$$

#### 多变量可微函数

关于多元函数  $f(x_1, x_2, \dots, x_n)$ , 我们可以得到类似的结论:

$$\varepsilon(f(\tilde{x})) \approx \sum_{k=1}^n \left| \frac{\partial f(\tilde{x})}{\partial x_k} \right| \varepsilon(\tilde{x}_k),$$



其中  $\tilde{x} = [\tilde{x}_1, \tilde{x}_2, \dots, \tilde{x}_n]^\top$  是  $x = [x_1, x_2, \dots, x_n]^\top$  的近似值.

**例 1.20** 测得某场地的长  $x$  和宽  $y$  分别为:  $\tilde{x} = 110\text{m}$ ,  $\tilde{y} = 80\text{m}$ , 其测量误差限分别为  $0.2\text{m}$  和  $0.1\text{m}$ . 试求面积  $S = x \times y$  的绝对误差限和相对误差限.

**解.** 由于  $\varepsilon(\tilde{x}) = 0.2\text{m}$ ,  $\varepsilon(\tilde{y}) = 0.1\text{m}$ , 故

$$\begin{aligned}\varepsilon(\tilde{S}) &\approx \left| \frac{\partial S(\tilde{x}, \tilde{y})}{\partial x} \right| \varepsilon(\tilde{x}) + \left| \frac{\partial S(\tilde{x}, \tilde{y})}{\partial y} \right| \varepsilon(\tilde{y}) \\ &= |\tilde{y}| \cdot \varepsilon(\tilde{x}) + |\tilde{x}| \cdot \varepsilon(\tilde{y}) \\ &= 80 \times 0.2 + 110 \times 0.1 = 27(\text{m}^2).\end{aligned}$$

相对误差限

$$\varepsilon_r(\tilde{S}) = \frac{\varepsilon(\tilde{S})}{|\tilde{S}|} \approx \frac{27}{110 \times 80} \approx 0.0031.$$

□

**例 1.21** 设  $x > 0$ ,  $x$  的相对误差是  $\delta$ , 试估计  $\ln x$  的误差.

**解.** 设  $\tilde{x}$  是  $x$  的近似值. 由题意可知, 相对误差为

$$\left| \frac{\tilde{x} - x}{\tilde{x}} \right| = \delta.$$

所以误差  $|\varepsilon(\tilde{x})| = |\tilde{x} - x| = |\tilde{x}| \cdot \delta$ . 设  $f(x) = \ln(x)$ , 则

$$\varepsilon(f(\tilde{x})) \approx |f'(\tilde{x})| \varepsilon(\tilde{x}) = \left| \frac{1}{\tilde{x}} \right| \cdot |\tilde{x}| \cdot \delta = \delta,$$

即  $\ln x$  的误差约为  $\delta$ .

□

### 1.3.5 误差分析

由于在进行数值计算时, 不仅仅原始数据可能带有误差, 在计算过程中也会出现舍入误差, 而且误差会不断传播和积累, 也会对消. 因此误差分析通常比较复杂. 目前的误差分析方法可分为定量分析和定性分析.

#### 定量分析

- 主要方法有: 向后误差分析法, 向前误差分析法, 区间误差分析法, 概率分析法等.
- 向后误差分析法比较有效, 其它方法仍不完善.
- 实际计算中的运算次数通常都在千万次以上, 如果对每一步运算都做误差估计的话, 工作量会非常大, 而且得到的误差界也往往不太实用.

#### 定性分析

- 目前在数值计算中更关注的是误差的定性分析;
- 定性分析包括数值算法的稳定性, 数学问题与原问题的相容性, 数学问题的适定性, 避免扩大误差的准则等;
- 定性分析的核心是原始数据的误差和计算过程中产生的误差对最终计算结果的影响.

✎ 算法有“优劣”之分, 问题有“好坏”之别, 即使不能定量地估计出最终误差, 但是若能确保计算过程中误差不会被任意放大, 那就能放心地实施计算, 这就是误差定性分析的初衷.

### 1.3.6 数值稳定性

#### 数学问题的适定性

**定义 1.16** 如果数学问题满足

- (1) 对任意满足一定条件的输入数据, 存在一个解,
- (2) 对任意满足一定条件的输入数据, 解是唯一的,
- (3) 问题的解关于输入数据是连续的,

则称该数学问题是**适定的** (well-posed), 否则就称为**不适定的** (ill-posed).

#### 病态问题与条件数

**定义 1.17** 如果输入数据的微小扰动会引起输出数据 (即计算结果) 的很大变化 (误差), 则称该数学问题是**病态**的, 否则就是**良态**的.

**例 1.22** 解线性方程组 
$$\begin{cases} x + \alpha y = 1 \\ \alpha x + y = 0 \end{cases}$$



**解.** 易知当  $\alpha = 1$  时, 方程组无解. 当  $\alpha \neq 1$  时, 解为

$$x = \frac{1}{1 - \alpha^2}, \quad y = \frac{-\alpha}{1 - \alpha^2}.$$

如果  $\alpha \approx 1$ , 则  $\alpha$  的微小误差可能会引起解的很大变化. 比如当  $\alpha = 0.999$  时,  $x \approx 500.25$ . 假定输入数据  $\alpha$  带有 0.0001 的误差, 即实际输入数据为  $\tilde{\alpha} = 0.9991$ , 则此时有  $\tilde{x} \approx 555.81$ , 解的误差约为 55.56, 是输入数据误差的五十多万倍, 因此该问题的病态的.  $\square$

设  $f(x)$  可导,  $\tilde{x}$  是精确值  $x_*$  的近似值, 则由 Taylor 公式可知

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

当  $\tilde{x}$  很接近  $x_*$  时,  $(\tilde{x} - x_*)^2$  非常小. 假定  $f''(x)$  在  $x_*$  附近的邻域内有界且不是很大, 则有

$$\left| \frac{f(\tilde{x}) - f(x_*)}{f(x_*)} \right| \approx \left| \frac{x_* f'(x_*)}{f(x_*)} \right| \times \left| \frac{\tilde{x} - x_*}{x_*} \right|,$$

即函数值的相对误差大约是输入数据相对误差的  $\left| \frac{x_* f'(x_*)}{f(x_*)} \right|$  倍. 这个值就定义为函数  $f(x)$  在  $x_*$  处的 **条件数**, 记为  $C_p(x_*)$ , 其中

$$C_p(x) \triangleq \left| \frac{x f'(x)}{f(x)} \right|.$$

#### 几点说明

- 如果条件数比较大, 则认为问题是病态的, 而且条件数越大问题病态就越严重;
- 病态是问题本身固有的性质, 与数值算法无关;
- 对于病态问题, 选择数值算法时需要更加谨慎.

#### 算法的稳定性

在数值计算过程中, 如果误差不增长或能得到有效控制, 则称该算法是**稳定的**, 否则为**不稳定的**.

#### 例 1.23 近似计算

(Demo11\_Stability.m)

$$S_n = \int_0^1 \frac{x^n}{x+5} dx, \quad n = 1, 2, \dots, 8.$$

**解.** 通过观察可知

$$S_n + 5S_{n-1} = \int_0^1 \frac{x^n + 5x^{n-1}}{x+5} dx = \int_0^1 x^{n-1} = \frac{1}{n},$$

因此,

$$S_n = \frac{1}{n} - 5S_{n-1}. \quad (1.6)$$

易知  $S_0 = \ln 6 - \ln 5 \approx 0.182$  (保留三位有效数字), 利用上面的递推公式可得 (保留三位有效数字)

$$\begin{aligned} S_1 &= 0.0900, & S_2 &= 0.0500, & S_3 &= 0.0833, & S_4 &= -0.166, \\ S_5 &= 1.03, & S_6 &= -4.98, & S_7 &= 25.0, & S_8 &= -125. \end{aligned}$$





另一方面, 我们有

$$\frac{1}{6(n+1)} = \int_0^1 \frac{x^n}{6} dx \leq \int_0^1 \frac{x^n}{x+5} dx \leq \int_0^1 \frac{x^n}{5} dx = \frac{1}{5(n+1)}. \quad (1.7)$$

因此, 上面计算的  $S_4, \dots, S_8$  显然是不对的. 原因是什么呢? 误差!

设  $\tilde{S}_n$  是  $S_n$  的近似值, 则

$$\epsilon(\tilde{S}_n) = \tilde{S}_n - S_n = \left( \frac{1}{n} - 5\tilde{S}_{n-1} \right) - \left( \frac{1}{n} - 5S_{n-1} \right) \approx -5(\tilde{S}_{n-1} - S_{n-1}) = -5\epsilon(\tilde{S}_{n-1}).$$

即误差是以 5 倍速度增长, 这说明计算过程是不稳定的, 因此我们不能使用该算法.

事实上, 递推公式 (1.6) 可以改写为

$$S_{n-1} = \frac{1}{5n} - \frac{1}{5}S_n.$$

因此, 我们可以先估计  $S_8$  的值, 然后通过反向递推, 得到其它值.

我们可以根据 (1.7) 对  $S_8$  做简单的估计, 即

$$S_8 \approx \frac{1}{2} \left( \int_0^1 \frac{x^n}{6} dx + \int_0^1 \frac{x^n}{5} dx \right) \approx 0.0204.$$

于是

$$\begin{aligned} S_7 &= 0.0209, & S_6 &= 0.0244, & S_5 &= 0.0285, & S_4 &= 0.0343, \\ S_3 &= 0.0431, & S_2 &= 0.0580, & S_1 &= 0.0884, & S_0 &= 0.182. \end{aligned}$$

通过误差分析可知, 误差是以  $\frac{1}{5}$  的速度减小, 因此计算过程是稳定的. □

在数值计算中, 误差不可避免, 算法的稳定性是一个非常重要的性质.

在数值计算中, 不要采用不稳定的算法!

用计算机进行整数之间的加减和乘法运算时, 没有误差. (不考虑溢出情况)

### 1.3.7 减小误差危害

为了尽量减小误差给计算结果带来的危害, 在数值计算过程中, 我们应该注意以下几点.

#### (1) 避免相近的数相减

如果两个相近的数相减, 则会损失有效数字, 如  $0.12346 - 0.12345 = 0.00001$ , 两个操作数都有 5 位有效数字, 但计算结果却只有 1 位有效数字.





**例 1.24** 计算  $\sqrt{9.01} - 3$ , 计算过程中保留 3 位有效数字.

**解.** 如果直接计算的话, 可得

$$\sqrt{9.01} = 3.0016662039607 \cdots \approx 3.00.$$

所以  $\sqrt{9.01} - 3 \approx 0.00$ , 一个有效数字都没有!

但如果换一种计算方法, 如

$$\sqrt{9.01} - 3 = \frac{9.01 - 3^2}{\sqrt{9.01} + 3} \approx \frac{0.01}{3.00 + 3} \approx 0.00167.$$

通过精确计算可知  $\sqrt{9.01} - 3 = 0.0016662039607 \cdots$ . 因此第二种计算能得到三位有效数字!  $\square$

通过各种等价公式来计算两个相近的数相减, 是避免有效数字损失的有效手段之一. 下面给出几个常用的等价公式:

$$\begin{aligned}\sqrt{x+\varepsilon} - \sqrt{x} &= \frac{\varepsilon}{\sqrt{x+\varepsilon} + \sqrt{x}} \\ \ln(x+\varepsilon) - \ln(x) &= \ln\left(1 + \frac{\varepsilon}{x}\right) \\ 1 - \cos(x) &= 2 \sin^2 \frac{x}{2}, \quad |x| \ll 1 \\ e^x - 1 &= x \left(1 + \frac{1}{2}x + \frac{1}{6}x^2 + \cdots\right), \quad |x| \ll 1\end{aligned}$$

**例 1.25** 计算  $y = \left(\frac{\sqrt{101} - 10}{\sqrt{101} + 10}\right)^2$ .

**解.** 方法一: 分母有理化

$$y = \left(\frac{\sqrt{101} - 10}{\sqrt{101} + 10}\right)^2 = 80801 - 8040\sqrt{101};$$

方法二: 分子有理化

$$y = \left(\frac{\sqrt{101} - 10}{\sqrt{101} + 10}\right)^2 = \frac{1}{80801 + 8040\sqrt{101}};$$

方法三: 直接将  $\sqrt{101}$  的近似值代入计算.

已知  $\sqrt{101} = 10.0498756 \cdots$ , 分别取近似值 10.04, 10.05, 10.06, 计算结果如下:

$\sqrt{101}$	方法一	方法二	方法三
10.04	79.400	6.1911E-6	3.9840E-6
10.05	-1.0000	6.1880E-6	6.2189E-6
10.06	-81.400	6.1849E-6	8.9462E-6

实际值为  $y = 6.188042227 \cdots \times 10^{-6}$ .  $\square$

**例 1.26** 在 MATLAB 中用双精度数计算

(Demo12\_Significance.m)

$$E_1 = \frac{1 - \cos(x)}{\sin^2(x)} \quad \text{和} \quad E_2 = \frac{1}{1 + \cos(x)}.$$

**解.** 计算结果如下:

$x$	$E_1$	$E_2$
1.0000000000	0.649223205204762	0.649223205204762
0.1000000000	0.501252086288577	0.501252086288571
0.0100000000	0.500012500208481	0.500012500208336
0.0010000000	0.500000124992189	0.5000000125000021
0.0001000000	0.499999998627931	0.5000000001250000
0.0000100000	0.5000000041386852	0.5000000000012500
0.0000010000	0.500044450291337	0.5000000000000125
0.0000001000	0.499600361081322	0.5000000000000001
0.0000000100	0.0000000000000000	0.5000000000000000
0.0000000010	0.0000000000000000	0.5000000000000000
0.0000000001	0.0000000000000000	0.5000000000000000

由此可见, 当  $x$  趋于 0 是,  $E_1$  的计算结果显然是错的, 而  $E_2$  则能很好地计算出近似值.  $\square$

**例 1.27** 计算  $y = \ln 2$ .

(Demo13\_ln.m)

**解.** 方法一: 利用  $f(x) = \ln(1+x)$  的 Taylor 展开

$$\ln(1+x) = x - \frac{1}{2}x^2 + \frac{1}{3}x^3 - \frac{1}{4}x^4 + \cdots + \frac{(-1)^{n+1}}{n}x^n + \cdots.$$

将  $x = 1$  代入后计算结果为

$n$	1	2	3	4	5	10	50	100
$S_n$	1	0.5	0.833	0.583	0.783	0.646	0.683	0.688
$ S_n - \ln 2 $	3.1E-1	1.9E-1	1.4E-1	1.1E-1	9.0E-2	4.8E-2	9.9E-3	5.0E-3

计算到第 100 项, 误差仍有 0.05.

方法二: 利用  $f(x) = \ln\left(\frac{1+x}{1-x}\right)$  的 Taylor 展开

$$\ln\left(\frac{1+x}{1-x}\right) = 2\left(x + \frac{1}{3}x^3 + \frac{1}{5}x^5 + \cdots + \frac{1}{2n-1}x^{2n-1} + \cdots\right).$$

将  $x = 1/3$  代入后计算结果为




$n$	1	2	3	4	5	10
$S_n$	0.667	0.691	0.693	0.693	0.693	0.693
$ S_n - \ln 2 $	2.6E-2	1.8E-3	1.4E-4	1.2E-5	1.1E-6	1.0E-11

计算到第 10 项, 误差已经小于  $10^{-10}$ ! 实际值为  $\ln 2 = 0.693147180559945$ . □

### (2) 避免数量级相差很大的数相除

可能会产生溢出, 即超出计算机所能表示的数的范围. 特别需要注意的是, 尽量不要用很小的数作为除数, 否则为放大分子的误差.

 如果两个数相除, 一般情况下建议把绝对值小的数作为分子, 这在后面的算法中会经常遇到.

### (3) 避免大数吃小数

如  $(10^9 + 10^{-9} - 10^9)/10^{-9}$ , 直接计算的话, 结果为 0.

另外, 在对一组数求和时, 建议按照绝对值从小到大求和.

**例 1.28** 计算  $S = 1 + 2 + 3 + \cdots + 100 + 10^{16}$ .

(Demo14\_Sum.m)

从小到大计算, 结果为

$$S = 1 + 2 + 3 + \cdots + 100 + 10^{16} = 1.0000000000005050 \times 10^{16}.$$

从大到小计算, 结果为

$$S = 10^{16} + 100 + 99 + \cdots + 2 + 1 = 1.0000000000005100 \times 10^{16}.$$

### (4) 简化计算

尽量减少运算次数, 从而减少误差的积累.

**例 1.29** 多项式计算. 设多项式

(Demo15\_Poly.m)

$$p(x) = 5x^5 + 4x^4 + 3x^3 + 2x^2 + 2x + 1.$$

试计算  $p(3)$  的值.

**解.** 如果直接计算的话, 可得

$$p(3) = 5 \times 3^5 + 4 \times 3^4 + 3 \times 3^3 + 2 \times 3^2 + 2 \times 3 + 1.$$

需要做 15 次乘法和 5 次加法.

在实际计算中, 当计算  $x^k$  时, 由于前面已经计算出  $x^{k-1}$ , 因此只需做一次乘法就可以了. 这样整个计算过程可以减少到 9 次乘法和 5 次加法. 但这并不是最佳方案.

事实上, 我们可以将多项式改写为

$$p(x) = (((5x + 4)x + 3)x + 2)x + 1.$$

这样就只需做 5 次乘法和 5 次加法. 显然这是更佳的计算方案.  $\square$

在计算多项式的值时, 我们都是将多项式改写成

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

这里利用了嵌套思想, 如果直接计算的话, 需要  $\frac{n(n+1)}{2}$  次乘法和  $n$  次加法. 但如果采用秦九韶算法的话, 只需做  $n$  次乘法和  $n$  次加法.

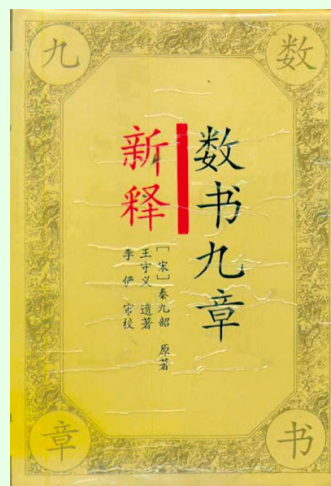
这种计算方法就是著名的 **秦九韶算法** (1247), 五百多年后, 英国数学家 Horner (1819) 重新发现了该公式 [33], 因此西方也称为 **Horner 算法**.

### 拓展阅读

秦九韶 (公元 1208 年–1268 年), 字道古, 南宋末年人, 祖籍鲁郡, 与李冶、杨辉、朱世杰并称宋元数学四大家. 秦九韶是我国古代数学家的杰出代表之一, 他撰写的《数书九章》(1247) 概括了宋元时期中国传统数学的主要成就, 尤其是系统总结和发展了高次方程的数值解法与一次同余问题的解法, 提出了相当完备的“正负开方术”和“大衍求一术”, 对数学发展产生了广泛的影响.

秦九韶既重视理论又重视实践, 既善于继承又勇于创新, 所著《数书九章》共九章十八卷, 每章为一类, 每类 9 题共计 81 个算题, 内容极其丰富, 上至天文、星象、历律、测候, 下至河道、水利、建筑、运输, 许多计算方法和经验常数至今仍有高的参考价值和实践意义, 被称为“算中宝典”.

集合论的创立人, 德国著名数学家康托 (M. Cantor, 1829–1920) 高度评价了秦九韶算法, 他称赞发现这一算法的中国数学家是“最幸运的天才”. 美国著名科学史家萨顿 (G. Sarton, 1884–1956) 则称秦九韶是“他那个民族, 他那个时代, 并且确实也是那个时代最伟大的数学家之一”. 二十世纪最杰出的科学史家李约瑟 (1900–1995) 称赞秦九韶“具有迷人的性格”. 我国数学大师吴文俊把《数书九章》同《九章算术》相提并论为中国传统数学的代表著作“两部《九章》”, 认为《数书九章》是世界中世纪数学的最高水平.



## 1.4 课后练习

**练习 1.1** 证明复数域上的 Cauchy-Schwartz 不等式: 设  $(\cdot, \cdot)$  是  $\mathbb{C}^n$  上的内积, 则有

$$|(x, y)| \leq \|x\|_2 \cdot \|y\|_2, \quad \forall x, y \in \mathbb{C}^n.$$

**练习 1.2** 计算  $f(x)$  关于  $C[0, 1]$  的  $\|f\|_\infty$ ,  $\|f\|_1$  和  $\|f\|_2$ .

$$(1) f(x) = (x-1)^3;$$

$$(2) f(x) = \left|x - \frac{1}{2}\right|;$$

$$(3) f(x) = x^m(1-x)^n, \quad m, n \text{ 均为正整数}.$$

**练习 1.3** 证明定理 1.12 中的三个不等式:

$$(1) \|x\|_2 \leq \|x\|_1 \leq \sqrt{n} \|x\|_2,$$

$$(2) \|x\|_\infty \leq \|x\|_1 \leq n \|x\|_\infty,$$

$$(3) \|x\|_\infty \leq \|x\|_2 \leq \sqrt{n} \|x\|_\infty.$$

**练习 1.4** 证明:

$$(1) \|A\|_1 = \max_{1 \leq j \leq n} \left( \sum_{i=1}^n |a_{ij}| \right); \quad (2) \|A\|_2 = \max_{x \neq 0, y \neq 0} \frac{y^T A x}{\|x\|_2 \|y\|_2}.$$

**练习 1.5** 证明: 定义在  $\mathbb{R}^{n \times n}$  上的算子范数和  $F$ -范数都是相容范数.

**练习 1.6** 设矩阵

$$A = \begin{bmatrix} 5 & -4 & 2 \\ -4 & 5 & -2 \\ 2 & -2 & 1 \end{bmatrix}.$$

计算  $A$  的行范数, 列范数, 2-范数和  $F$ -范数.

**练习 1.7** 证明:

$$(1) \text{对任意的算子范数 } \|\cdot\|, \text{ 有 } \|I\| = 1;$$

$$(2) \text{对任意的相容范数 } \|\cdot\|, \text{ 有 } \|I\| \geq 1.$$

**练习 1.8** 证明:

$$(1) \frac{1}{\sqrt{n}} \|A\|_F \leq \|A\|_1 \leq \sqrt{n} \|A\|_F;$$

$$(2) \|A\|_2 \leq \|A\|_F \leq \sqrt{n} \|A\|_2;$$

$$(3) \|A\|_2^2 \leq \|A\|_1 \cdot \|A\|_\infty.$$

**练习 1.9** 设  $A_k$  是  $A \in \mathbb{R}^{n \times n}$  的  $k$  阶子矩阵 ( $k \leq n$ ). 证明:  $\|A_k\|_p \leq \|A\|_p$ .

**练习 1.10** 设  $\|\cdot\|$  是  $\mathbb{R}^m$  空间上的一个向量范数,  $A \in \mathbb{R}^{m \times n}$ , 且  $\text{rank}(A) = n$ .

证明:  $\|x\| \triangleq \|Ax\|$  是一个向量范数.

**练习 1.11\*** 设  $\|\cdot\|$  是  $\mathbb{R}^n$  空间上的一个向量范数. 证明:  $\|A^{-1}\|^{-1} = \min_{\|x\|=1} \|Ax\|$ .

**练习 1.12** 设  $A \in \mathbb{C}^{n \times n}$ . 试证明对任意矩阵范数都有  $\rho(A) \leq \|A\|$ .

**练习 1.13** 证明定理 1.29:



$A \in \mathbb{C}^{n \times n}$  正定 (半正定) 的充要条件是矩阵  $H = (A + A^*)/2$  正定 (半正定).

练习 1.14 证明定理 1.30:

$A \in \mathbb{R}^{n \times n}$  正定 (半正定) 的充要条件是对任意非零向量  $x \in \mathbb{R}^n$  有  $x^T A x > 0$  ( $x^T A x \geq 0$ ).

练习 1.15 设  $A \in \mathbb{R}^{n \times n}$  正定, 证明:  $A^{-1}$  也正定.

练习 1.16 设  $A = [a_{ij}] \in \mathbb{R}^{n \times n}$  对称正定, 证明: 当  $i \neq j$  时有  $a_{ij}^2 < a_{ii}a_{jj}$ .

练习 1.17 设  $A \in \mathbb{R}^{m \times n}$ ,  $B \in \mathbb{R}^{n \times m}$ , 证明:  $\text{tr}(AB) = \text{tr}(BA)$ .

练习 1.18 设  $a \in \mathbb{R}$ , 证明:

$$\begin{bmatrix} 1 & -a & & & \\ & 1 & -a & & \\ & & \ddots & \ddots & \\ & & & \ddots & -a \\ & & & & 1 \end{bmatrix}^{-1} = \begin{bmatrix} 1 & a & a^2 & \cdots & a^{n-1} \\ & 1 & a & \ddots & \vdots \\ & & \ddots & \ddots & a^2 \\ & & & \ddots & a \\ & & & & 1 \end{bmatrix}.$$

练习 1.19 设  $x > 0$ , 在输入数据时会带有一定的误差, 但相对误差 (绝对值) 不超过  $\delta$ ,  $x^n$  的相对误差. (提示: 计算误差限和相对误差限)

练习 1.20 分别用  $x_1 = \frac{22}{7}$  和  $x_2 = \frac{355}{113}$  作为  $\pi = 3.1415926535 \dots$  的近似值, 各有几位有效数字?

练习 1.21 设  $\tilde{x}_1 = 1.1021$  和  $\tilde{x}_2 = 0.031$  都是四舍五入后得到得近似值, 试估计  $\tilde{x}_1 + \tilde{x}_2$ ,  $\tilde{x}_1 \tilde{x}_2$  和  $\tilde{x}_1/\tilde{x}_2$  的误差.

练习 1.22 计算球体积, 要使相对误差限不超过 1%, 问半径  $R$  所允许的相对误差限是多少?

练习 1.23 求方程  $x^2 + 3x + 1 = 0$  的两个根, 使它至少具有 3 位有效数字. (取  $\sqrt{5} \approx 2.236$ )

练习 1.24 设  $s = \frac{1}{2}gt^2$ , 假定  $g$  是准确的, 而对  $t$  的测量有  $\pm 0.1$  秒的误差. 证明: 当  $t$  增加时  $s$  的绝对误差增加. 相对误差呢?

练习 1.25\* 设  $B \in \mathbb{R}^{m \times n}$  ( $m \leq n$ ) 是满秩矩阵,  $C \in \mathbb{R}^{m \times m}$  是对称半正定矩阵. 试证明:

$$B^T(BB^T)^{-1}B - B^T(C + BB^T)^{-1}B$$

是对称半正定的.

练习 1.26\* 设  $P$  是置换矩阵, 则  $P$  可以表示为一系列初等置换矩阵 (即交换单位矩阵的两行所得到的矩阵) 的乘积. 试问在什么条件下, 有  $P^T = P$ .

练习 1.27\* 设矩阵  $A$  是三对角矩阵, 且两条次对角线元素均非零, 则  $A$  是否可约?

练习 1.28\* 设  $A \in \mathbb{R}^{n \times n}$  是对称正定矩阵, 证明

$$f(x, y) \triangleq y^T A x$$

是  $\mathbb{R}^n$  上的一个内积. 反之, 设  $(\cdot, \cdot)$  是  $\mathbb{R}^n$  上的一个内积, 证明: 存在一个对称正定矩阵  $A \in \mathbb{R}^{n \times n}$  使得

$$(x, y) = y^T A x.$$



# 2

## 线性方程组直接解法

本讲主要介绍如何求解下面的线性方程组

$$Ax = b, \quad A \in \mathbb{R}^{n \times n}, b \in \mathbb{R}^n. \quad (2.1)$$

在自然科学和工程技术的实际应用中, 很多问题的解决最终都归结为求解一个或多个线性方程组. 目前求解线性方程组的方法可以分为两大类: [直接法](#)和[迭代法](#). 本讲主要介绍直接法. 直接法具有良好的稳定性和健壮性, 是当前求解中小规模线性方程组的首选方法, 同时也是求解某些具有特殊结构的大规模线性方程组的主要方法.

在本章中, 我们总是假定系数矩阵  $A$  是非奇异的, 即线性方程组 (2.1) 的解存在且唯一.

### 关于直接法的相关参考文献

- [1] G. H. Golub and C. F. Van Loan, [Matrix Computations](#), 4th, 2013 [21]
- [2] I. S. Duff, A. M. Erisman and J. K. Reid, [Direct Methods for Sparse Matrices](#), 2nd, 2017 [15]
- [3] T. A. Davis, [Direct Methods for Sparse Linear Systems](#), SIAM, 2006 [10]

文献 [2,3] 主要是介绍大规模稀疏线性方程组的直接解法.

### 2.1 Gauss 消去法

Gauss 消去法的基本思想是消元. 早在 2000 年前, 中国古代学者就提出了消元思想 (记载在公元初《九章算术》方程章中), 后来 Newton, Lagrange, Gauss, Jacobi 等都对此做过研究 [22, 50]. 我们目前采用的算法描述方式是十九世纪三十年代后期才形成的.

首先看一个例子.

**例 2.1** 求解下面的线性方程组

$$\begin{cases} x_1 - 2x_2 + 2x_3 = -2 \\ 2x_1 - 3x_2 - 3x_3 = 4 \\ 4x_1 + x_2 + 6x_3 = 3. \end{cases}$$

**解.** 利用 [Gauss 消去法](#)求解: 先写出增广矩阵, 然后通过初等变换将其转换为阶梯形, 最后通过



回代求解. 具体过程可写为

$$\begin{bmatrix} 1 & -2 & 2 & -2 \\ 2 & -3 & -3 & 4 \\ 4 & 1 & 6 & 3 \end{bmatrix} \xrightarrow[\text{③}-\text{①} \times 4]{\text{②}-\text{①} \times 2} \begin{bmatrix} 1 & -2 & 2 & -2 \\ 0 & 1 & -7 & 8 \\ 0 & 9 & -2 & 11 \end{bmatrix} \xrightarrow{\text{③}-\text{②} \times 9} \begin{bmatrix} 1 & -2 & 2 & -2 \\ 0 & 1 & -7 & 8 \\ 0 & 0 & 61 & -61 \end{bmatrix}$$

通过回代求解可得

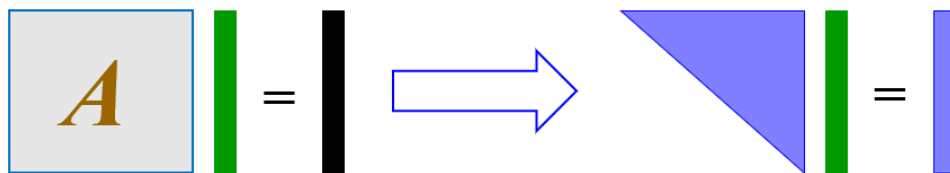
$$\begin{cases} x_3 = -1, \\ x_2 = 8 + 7x_3 = 1, \\ x_1 = -2 + 2x_2 - 2x_3 = 2. \end{cases}$$

□

将以上的做法推广到一般线性方程  $Ax = b$ , 即

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2 \\ \dots\dots\dots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n = b_n \end{cases}$$

高斯消去法的主要思路: 将系数矩阵  $A$  化为上三角矩阵, 然后回代求解:



高斯消去法是求解线性方程组的经典算法, 它在当代数学中有着非常重要的地位和价值, 是线性代数的重要组成部分。高斯消去法除了用于线性方程组求解外, 还用于计算矩阵行列式、求矩阵的秩、计算矩阵的逆等。

### 2.1.1 Gauss 消去过程

本小节给出 Gauss 消去过程的详细执行过程, 写出相应算法, 并编程实现. 记增广矩阵

$$A^{(1)} = \left[ \begin{array}{cccc|c} a_{11}^{(1)} & a_{12}^{(1)} & \cdots & a_{1n}^{(1)} & b_1^{(1)} \\ a_{21}^{(1)} & a_{22}^{(1)} & \cdots & a_{2n}^{(1)} & b_2^{(1)} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{n1}^{(1)} & a_{n2}^{(1)} & \cdots & a_{nn}^{(1)} & b_n^{(1)} \end{array} \right],$$

其中

$$a_{ij}^{(1)} = a_{ij}, \quad b_i^{(1)} = b_i, \quad i, j = 1, 2, \dots, n.$$

**第 1 步:** 消第 1 列.

设  $a_{11}^{(1)} \neq 0$ , 计算  $m_{i1} = \frac{a_{i1}^{(1)}}{a_{11}^{(1)}}, i = 2, 3, \dots, n$ . 对增广矩阵  $A^{(1)}$  进行  $n-1$  次初等变换, 即





依次将  $A^{(1)}$  的第  $i$  行 ( $i > 1$ ) 减去第 1 行的  $m_{i1}$  倍, 将新得到的矩阵记为  $A^{(2)}$ , 即

$$A^{(2)} = \left[ \begin{array}{cccc|c} a_{11}^{(1)} & a_{12}^{(1)} & \cdots & a_{1n}^{(1)} & b_1^{(1)} \\ & a_{22}^{(2)} & \cdots & a_{2n}^{(2)} & b_2^{(2)} \\ & \vdots & \ddots & \vdots & \vdots \\ & a_{n2}^{(2)} & \cdots & a_{nn}^{(2)} & b_n^{(2)} \end{array} \right],$$

其中

$$a_{ij}^{(2)} = a_{ij}^{(1)} - m_{i1}a_{1j}^{(1)}, \quad b_i^{(2)} = b_i^{(1)} - m_{i1}b_1^{(1)}, \quad i, j = 2, 3, \dots, n.$$

第 2 步: 消第 2 列.

设  $a_{22}^{(2)} \neq 0$ , 计算  $m_{i2} = \frac{a_{i2}^{(2)}}{a_{22}^{(2)}}, i = 3, 4, \dots, n$ . 依次将  $A^{(2)}$  的第  $i$  行 ( $i > 2$ ) 减去第 2 行的  $m_{i2}$  倍, 将新得到的矩阵记为  $A^{(3)}$ , 即

$$A^{(3)} = \left[ \begin{array}{ccccc|c} a_{11}^{(1)} & a_{12}^{(1)} & a_{13}^{(1)} & \cdots & a_{1n}^{(1)} & b_1^{(1)} \\ & a_{22}^{(2)} & a_{23}^{(2)} & \cdots & a_{2n}^{(2)} & b_2^{(2)} \\ & & a_{33}^{(3)} & \cdots & a_{3n}^{(3)} & b_3^{(3)} \\ & & \vdots & \ddots & \vdots & \vdots \\ & & a_{n3}^{(3)} & \cdots & a_{nn}^{(3)} & b_n^{(3)} \end{array} \right],$$

其中

$$a_{ij}^{(3)} = a_{ij}^{(2)} - m_{i2}a_{2j}^{(2)}, \quad b_i^{(3)} = b_i^{(2)} - m_{i2}b_2^{(2)}, \quad i, j = 3, 4, \dots, n.$$

依此类推, 经过  $k-1$  步后, 可得新矩阵  $A^{(k)}$ :

$$A^{(k)} = \left[ \begin{array}{cccc|c} a_{11}^{(1)} & \cdots & a_{1k}^{(1)} & \cdots & a_{1n}^{(1)} & b_1^{(1)} \\ & & \vdots & & \vdots & \vdots \\ & & a_{kk}^{(k)} & \cdots & a_{kn}^{(k)} & b_k^{(k)} \\ & & \vdots & \ddots & \vdots & \vdots \\ & & a_{nk}^{(k)} & \cdots & a_{nn}^{(k)} & b_n^{(k)} \end{array} \right], \quad (2.2)$$

第  $k$  步: 消第  $k$  列.

设  $a_{kk}^{(k)} \neq 0$ , 计算  $m_{ik} = \frac{a_{ik}^{(k)}}{a_{kk}^{(k)}}, i = k+1, k+2, \dots, n$ . 依次将  $A^{(k)}$  的第  $i$  行 ( $i > k$ ) 减去第  $k$  行的  $m_{ik}$  倍, 将新得到的矩阵记为  $A^{(k+1)}$ , 矩阵元素的更新公式为

$$a_{ij}^{(k+1)} = a_{ij}^{(k)} - m_{ik}a_{kj}^{(k)}, \quad b_i^{(k+1)} = b_i^{(k)} - m_{ik}b_k^{(k)}, \quad i, j = k+1, k+2, \dots, n. \quad (2.3)$$

这样, 经过  $n-1$  步后, 即可得到一个上三角矩阵  $A^{(n)}$ :

$$A^{(n)} = \left[ \begin{array}{cccc|c} a_{11}^{(1)} & a_{12}^{(1)} & \cdots & a_{1n}^{(1)} & b_1^{(1)} \\ & a_{22}^{(2)} & \cdots & a_{2n}^{(2)} & b_2^{(2)} \\ & & \ddots & \vdots & \vdots \\ & & & a_{nn}^{(n)} & b_n^{(n)} \end{array} \right].$$

最后, 回代求解

$$x_n = \frac{b_n^{(n)}}{a_{nn}^{(n)}}, \quad x_i = \frac{1}{a_{ii}^{(i)}} \left( b_i^{(i)} - \sum_{j=i+1}^n a_{ij}^{(i)} x_j \right), \quad i = n-1, n-2, \dots, 1.$$

以上就是 Gauss 消去法的整个计算过程.

由上面的计算过程可知, Gauss 消去法能顺利进行下去的充要条件是  $a_{kk}^{(k)} \neq 0, k = 1, 2, \dots, n$ , 这些元素被称为 **主元**.

**定理 2.1** 设  $A \in \mathbb{R}^{n \times n}$ , 则所有主元  $a_{kk}^{(k)}$  都不为零的充要条件是  $A$  的所有顺序主子式都不为零, 即

$$D_1 \triangleq a_{11} \neq 0, \quad D_k \triangleq \begin{vmatrix} a_{11} & \cdots & a_{1k} \\ \vdots & \ddots & \vdots \\ a_{k1} & \cdots & a_{kk} \end{vmatrix} \neq 0, \quad k = 2, 3, \dots, n.$$

事实上, 如果  $A$  的所有顺序主子式都不为零, 则主元为

$$a_{11}^{(1)} = D_1, \quad a_{kk}^{(k)} = \frac{D_k}{D_{k-1}}, \quad k = 2, 3, \dots, n.$$

**推论 2.2** Gauss 消去法能顺利完成的充要条件是  $A$  的所有顺序主子式都不为零.

### Gauss 消去法的运算量

下面统计整个 Gauss 消去法的乘除运算的次数.

在第  $k$  步中, 我们需要计算

$$\begin{aligned} m_{ik} &= a_{ik}^{(k)} / a_{kk}^{(k)}, i = k+1, \dots, n \rightarrow n-k \text{ 次} \\ a_{ij}^{(k+1)} &= a_{ij}^{(k)} - m_{ik} a_{kj}^{(k)}, i = k+1, \dots, n \rightarrow (n-k)^2 \text{ 次} \\ b_i^{(k+1)} &= b_i^{(k)} - m_{ik} b_k^{(k)}, i = k+1, \dots, n \rightarrow n-k \text{ 次} \end{aligned}$$

所以整个消去过程的乘除运算为

$$\sum_{k=1}^{n-1} 2(n-k) + (n-k)^2 = \sum_{\ell=1}^{n-1} 2\ell + \ell^2 = n(n-1) + \frac{n(n-1)(2n-3)}{6}.$$

易知, 回代求解过程的乘除运算为

$$1 + \sum_{i=1}^{n-1} n-i+1 = \frac{n(n+1)}{2}.$$

所以整个 Gauss 消去法的乘除运算为

$$\frac{n^3}{3} + n^2 - \frac{n}{3}.$$

同理, 也可统计出, Gauss 消去法中的加减运算次数为

$$\frac{n^3}{3} + \frac{n^2}{2} - \frac{5n}{6}.$$

评价算法的一个重要指标是执行时间, 但这依赖于计算机硬件和编程技巧等, 因此直接给出算法执行时间是不太现实的. 所以我们通常是统计算法中算术运算 (加减乘除) 的次数. 在数值算法中, 大多仅仅涉及加减乘除和开方运算. 一般地, 加减运算次数与乘法运算次数



具有相同的量级, 而除法运算和开方运算次数具有更低的量级.

为了尽可能地减少运算量, 在实际计算中, 数, 向量和矩阵做乘法运算时的先后执行次序为: 先计算数与向量的乘法, 然后计算矩阵与向量的乘法, 最后才计算矩阵与矩阵的乘法. 比如计算  $\alpha ABx$ , 其中  $\alpha$  是数,  $A, B$  是矩阵,  $x$  是向量, 如果按照从左往右计算的话, 则运算量为  $\mathcal{O}(n^3)$ , 但是如果先计算  $\alpha x$ , 然后计算  $B(\alpha x)$ , 最后再计算  $A(B(\alpha x))$  的话, 运算量则为  $\mathcal{O}(n^2)$ , 相差一个量级.

### 2.1.2 Gauss 消去法与 LU 分解

换个角度看 Gauss 消去过程: 每次都是做矩阵初等变换, 因此也可理解为不断地左乘初等矩阵. 将所有这些初等矩阵的乘积记为  $\tilde{L}$ , 则可得  $\tilde{L}A = U$ , 其中  $U$  是一个上三角矩阵. 记  $L \triangleq \tilde{L}^{-1}$ , 则

$$A = LU,$$

这就是著名的矩阵 **LU 分解**.

**矩阵分解**, 即将一个较复杂的矩阵分解成若干具有简单结构的矩阵的乘积, 是矩阵计算中的一个很重要的技术.

假定 Gauss 消去过程能顺利进行, 那么  $U$  一定是一个非奇异上三角矩阵. 下面我们主要研究  $L$  具有什么样的特殊结构或者特殊性质.

我们只需考察第  $k$  步的情形, 即  $A^{(k+1)}$  与  $A^{(k)}$  之间的关系式. 为了讨论方便, 我们这里的记号  $A^{(k)}$  仅表示增广矩阵 (2.2) 的前  $n$  列, 即只包含矩阵部分, 右端项不再包含在里面. 通过观察, 由 Gauss 消去过程 (即更新公式 (2.3)) 可得

$$A^{(k+1)} = L_k A^{(k)},$$

其中

$$L_k = \begin{bmatrix} 1 & & & & \\ & \ddots & & & \\ & & 1 & & \\ & & -m_{k+1,k} & 1 & \\ & & \vdots & & \ddots \\ & & -m_{n,k} & & 1 \end{bmatrix}, \quad m_{ik} = \frac{a_{ik}^{(k)}}{a_{kk}^{(k)}}, \quad i = k+1, k+2, \dots, n. \quad (2.4)$$

令  $k = 1, 2, \dots, n-1$ , 并将所有 Gauss 消去过程结合在一起即可得

$$A^{(n)} = L_{n-1} L_{n-2} \cdots L_1 A^{(1)},$$

即

$$A = A^{(1)} = (L_{n-1} L_{n-2} \cdots L_1)^{-1} A^{(n)}.$$

**引理 2.3** 下面两个等式成立:

$$L_k^{-1} = \begin{bmatrix} 1 & & & & \\ & \ddots & & & \\ & & 1 & & \\ & & m_{k+1,k} & 1 & \\ & & \vdots & & \ddots \\ & & m_{n,k} & & 1 \end{bmatrix}, \quad L_1^{-1} L_2^{-1} \cdots L_{n-1}^{-1} = \begin{bmatrix} 1 & & & & \\ m_{21} & 1 & & & \\ m_{31} & m_{32} & 1 & & \\ m_{41} & m_{42} & m_{43} & 1 & \\ \vdots & \vdots & \vdots & & \ddots \\ m_{n1} & m_{n2} & m_{n3} & \cdots & m_{n,n-1} & 1 \end{bmatrix}.$$

(证明留作练习)

记  $L \triangleq L_1^{-1} L_2^{-1} \cdots L_{n-1}^{-1}$ ,  $U \triangleq A^{(n)}$ , 则

$$A = LU. \quad (2.5)$$

由引理 2.3 可知  $L$  是单位下三角矩阵,  $U$  是非奇异上三角矩阵.

由推论 2.2 可知, 如果  $A$  的所有顺序主子式都不为零, 则 Gauss 消去过程能顺利进行, 因此 LU 分解 (2.5) 也存在. 事实上, 我们有下面的结论.

**定理 2.4 (LU 分解的存在性和唯一性)** 设  $A \in \mathbb{R}^{n \times n}$ . 则存在唯一的单位下三角矩阵  $L$  和非奇异上三角矩阵  $U$ , 使得  $A = LU$  的充要条件是  $A$  的所有顺序主子矩阵  $A_k = A(1:k, 1:k)$  都非奇异,  $k = 1, 2, \dots, n$ .

**证明. 必要性:** 设  $A_{11}$  是  $A$  的  $k$  阶顺序主子矩阵, 将  $A = LU$  写成分块形式

$$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} = \begin{bmatrix} L_{11} & 0 \\ L_{21} & L_{22} \end{bmatrix} \begin{bmatrix} U_{11} & U_{12} \\ 0 & U_{22} \end{bmatrix} = \begin{bmatrix} L_{11}U_{11} & L_{11}U_{12} \\ L_{21}U_{11} & L_{21}U_{12} + L_{22}U_{22} \end{bmatrix}.$$

可得  $A_{11} = L_{11}U_{11}$ . 由于  $L_{11}$  和  $U_{11}$  均非奇异, 所以  $A_{11}$  也非奇异.

**充分性:** 用归纳法.

当  $n = 1$  时, 结论显然成立.

假设结论对  $n - 1$  阶矩阵都成立, 即对任意  $n - 1$  阶矩阵, 如果其所有的顺序主子矩阵都非奇异, 则存在 LU 分解.

考虑  $n$  阶的矩阵  $A$ , 写成分块形式

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix},$$

其中  $A_{11} \in \mathbb{R}^{(n-1) \times (n-1)}$  是  $A$  的  $n - 1$  阶顺序主子矩阵. 由归纳假设可知,  $A_{11}$  存在 LU 分解, 即存在单位下三角矩阵  $L_{11}$  和非奇异上三角矩阵  $U_{11}$  使得

$$A_{11} = L_{11}U_{11}.$$

令

$$L_{21} = A_{21}U_{11}^{-1}, \quad U_{12} = L_{11}^{-1}A_{12}, \quad U_{22} = A_{22} - L_{21}U_{12},$$

则

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} = \begin{bmatrix} L_{11}U_{11} & L_{11}U_{12} \\ L_{21}U_{11} & U_{22} + L_{21}U_{12} \end{bmatrix} = \begin{bmatrix} L_{11} & 0 \\ L_{21} & 1 \end{bmatrix} \begin{bmatrix} U_{11} & U_{12} \\ 0 & U_{22} \end{bmatrix} \triangleq LU.$$



易知  $U$  非奇异, 所以  $A$  存在 LU 分解.

下面证明**唯一性**. 设  $A$  存在两个不同的 LU 分解:

$$A = LU = \tilde{L}\tilde{U},$$

其中  $L$  和  $\tilde{L}$  为单位下三角矩阵,  $U$  和  $\tilde{U}$  为非奇异上三角矩阵. 则有

$$L^{-1}\tilde{L} = U\tilde{U}^{-1},$$

该等式左边为下三角矩阵, 右边为上三角矩阵, 所以只能是对角矩阵. 由于单位下三角矩阵的逆仍然是单位下三角矩阵, 所以  $L^{-1}\tilde{L}$  的对角线元素全是 1, 故

$$L^{-1}\tilde{L} = I,$$

即  $\tilde{L} = L, \tilde{U} = U$ .

由归纳法可知, 结论成立. □

如果  $A$  存在 LU 分解, 则  $Ax = b$  可写为  $LUx = b$ , 因此就等价于求解下面两个方程组

$$\begin{cases} Ly = b, \\ Ux = y. \end{cases}$$

由于  $L$  是单位下三角,  $U$  是非奇异上三角, 因此上面的两个方程组都非常容易求解.

## LU 分解算法

将上面的 LU 分解过程写成算法, 描述如下:

### 算法 2.1. LU 分解

```

1: Set  $L = I, U = 0$    % 将  $L$  设为单位矩阵,  $U$  设为零矩阵
2: for  $k = 1$  to  $n - 1$  do
3:   for  $i = k + 1$  to  $n$  do
4:      $m_{ik} = a_{ik}/a_{kk}$    % 计算  $L$  的第  $k$  列
5:   end for
6:   for  $i = k$  to  $n$  do
7:      $u_{ki} = a_{ki}$    % 计算  $U$  的第  $k$  行
8:   end for
9:   for  $i = k + 1$  to  $n$  do
10:    for  $j = k + 1$  to  $n$  do
11:       $a_{ij} = a_{ij} - m_{ik}u_{kj}$    % 更新  $A(k+1:n, k+1:n)$ 
12:    end for
13:  end for
14: end for
  
```

### 矩阵 $L$ 和 $U$ 的存储

当  $A$  的第  $i$  列被用于计算  $L$  的第  $i$  列后, 在后面的计算中不再被使用. 同样地,  $A$  的第  $i$  行被用于计算  $U$  的第  $i$  行后, 在后面的计算中也不再被使用. 因此, 为了节省存储空间, 我们可以在计算过程中将  $L$  的第  $i$  列存放在  $A$  的第  $i$  列, 将  $U$  的第  $i$  行存放在  $A$  的第  $i$  行, 这样就不需要另外分配空间存储  $L$  和  $U$ . 计算结束后,  $A$  的上三角部分为  $U$ , 其严格下三角部分为  $L$  的严格下三角部分 ( $L$  的对角线全部为 1, 不需要存储). 这样, 算法就更简洁高效, 可以描述为:

#### 算法 2.2. LU 分解

```

1: for  $k = 1$  to  $n - 1$  do
2:   for  $i = k + 1$  to  $n$  do
3:      $a_{ik} = a_{ik} / a_{kk}$ 
4:     for  $j = k + 1$  to  $n$  do
5:        $a_{ij} = a_{ij} - a_{ik}a_{kj}$ 
6:     end for
7:   end for
8: end for

```

根据指标的循环次序, 上述算法也称为 KIJ 型 LU 分解. 在实际计算中, 我们一般不建议使用这个算法. 因为对于指标  $k$  的每次循环, 都需要更新  $A$  的第  $k + 1$  至第  $n$  行. 这种反复读取数据的做法会使得计算效率大大降低.

对于按行存储的数据结构, 如 C/C++, 我们一般采用下面的 IKJ 型 LU 分解.

#### 算法 2.3. LU 分解 (IKJ 型)

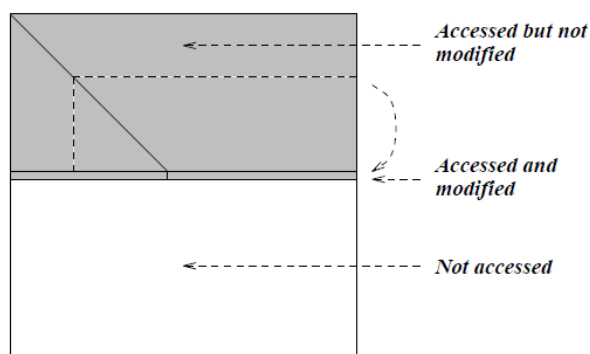
```

1: for  $i = 2$  to  $n$  do
2:   for  $k = 1$  to  $i - 1$  do
3:      $a_{ik} = a_{ik} / a_{kk}$ 
4:     for  $j = k + 1$  to  $n$  do
5:        $a_{ij} = a_{ij} - a_{ik}a_{kj}$ 
6:     end for
7:   end for
8: end for

```

上述算法可以用下图来描述.





如果数据是按列存储的, 如 FORTRAN 或 MATLAB, 则怎样设计算法?

### 2.1.3 列主元 Gauss 消去法与 PLU 分解

我们知道, 只要系数矩阵  $A$  非奇异, 则线性方程组就存在唯一解. 但 Gauss 消去法却不一定有效, 即可能会出现主元为零的情形, 此时算法就进行不下去.

**例 2.2** 求解线性方程组  $Ax = b$ , 其中  $A = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$ ,  $b = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$ .

由于主元  $a_{11} = 0$ , 因此 Gauss 消去法无法顺利进行下去.

在实际计算中, 即使主元都不为零, 但如果主元的值很小, 由于舍入误差的原因, 也可能会给计算结果带来很大的误差.

**例 2.3** 求解线性方程组  $Ax = b$ , 其中  $A = \begin{bmatrix} 0.02 & 61.3 \\ 3.43 & -8.5 \end{bmatrix}$ ,  $b = \begin{bmatrix} 61.5 \\ 25.8 \end{bmatrix}$ , 要求在运算过程中保留 3 位有效数字.

**解.** 根据 LU 分解算法 2.1, 我们可得

$$\begin{aligned} l_{11} &= 1.00, \quad l_{21} = a_{21}/a_{11} = 1.72 \times 10^2, \quad l_{22} = 1.00, \\ u_{11} &= a_{11} = 2.00 \times 10^{-2}, \quad u_{12} = a_{12} = 6.13 \times 10, \\ u_{22} &= a_{22} - l_{21}u_{12} \approx -8.5 - 1.05 \times 10^4 \approx -1.05 \times 10^4, \end{aligned}$$

即

$$A \approx \begin{bmatrix} 1.00 & 0 \\ 1.72 \times 10^2 & 1.00 \end{bmatrix} \begin{bmatrix} 2.00 \times 10^{-2} & 6.12 \times 10 \\ 0 & -1.05 \times 10^4 \end{bmatrix}.$$

解方程组  $Ly = b$  可得

$$y_1 = 6.15 \times 10, \quad y_2 = b_2 - l_{21}y_1 \approx -1.06 \times 10^4.$$

解方程组  $Ux = y$  可得

$$x_2 = y_2/u_{22} \approx 1.01, \quad x_1 = (y_1 - u_{12}x_2)/u_{11} \approx -0.413/u_{11} \approx -20.7$$

易知, 方程的精确解为  $x_1 = 10.0$  和  $x_2 = 1.00$ . 我们发现  $x_1$  的误差非常大, 导致这个问题的原因就是  $|a_{11}|$  太小, 用它做主元时会放大舍入误差.  $\square$

解决上面问题的一个有效方法就是选主元. 具体做法就是, 在执行 Gauss 消去过程的第  $k$  步之前, 插入下面的选主元过程.

① 选取 **列主元**:  $|a_{i_k, k}^{(k)}| = \max_{k \leq i \leq n} \{|a_{i, k}^{(k)}|\}$

② 交换: 如果  $i_k \neq k$ , 则交换第  $k$  行与第  $i_k$  行

也就是说, 在执行第  $k$  步时, 先在  $A^{(k)}$  中第  $k$  列的第  $k$  至  $n$  的元素中选取绝对值最大的元素, 如下图所示

$$\begin{bmatrix} a_{11}^{(1)} & \cdots & a_{1k}^{(1)} & \cdots & a_{1n}^{(1)} \\ & \ddots & \vdots & & \vdots \\ & & a_{kk}^{(k)} & \cdots & a_{kn}^{(k)} \\ & & \vdots & \ddots & \vdots \\ & & a_{nk}^{(k)} & \cdots & a_{nn}^{(k)} \end{bmatrix}$$

假定绝对值最大的元素在第  $i_k$  行 ( $i_k \geq k$ ), 如果  $i_k = k$ , 则  $a_{kk}^{(k)}$  就是主元, 此时不用做任何变动, 否则的话, 就交换第  $k$  行和第  $i_k$  行, 此时主元就是  $a_{i_k, k}^{(k)}$ .

上面选出的  $a_{i_k, k}^{(k)}$  就称为 **列主元**. 易知, 加入这个选主元过程后, 就不会出现主元为零的情形 (除非  $A$  是奇异的). 由此, Gauss 消去法就不会失效. 这种带选主元的 Gauss 消去法就称为 **列主元 Gauss 消去法** 或 **部分选主元 Gauss 消去法** (Gaussian Elimination with Partial Pivoting, GEPP).

下面给出列主元 Gauss 消去法的完整算法, 其中  $L$  存放在  $A$  的下三角部分,  $U$  存放在  $A$  的上三角部分.

#### 算法 2.4. 列主元 Gauss 消去法

```

1: for  $k = 1$  to  $n - 1$  do
2:    $a_{i_k, k} = \max_{k \leq i \leq n} |a_{i, k}|$    % 选列主元
3:   if  $i_k \neq k$  then
4:     for  $j = 1$  to  $n$  do
5:        $a_{tmp} = a_{i_k, j}$ ,  $a_{i_k, j} = a_{k, j}$ ,  $a_{k, j} = a_{tmp}$    % 交换  $A$  的第  $i_k$  行与第  $k$  行
6:     end for
7:      $b_{tmp} = b_{i_k}$ ,  $b_{i_k} = b_k$ ,  $b_k = b_{tmp}$    % 交换  $b$  的第  $i_k$  与第  $k$  个分量
8:   end if
9:   for  $i = k + 1$  to  $n$  do
10:     $a_{ik} = a_{ik} / a_{kk}$    % 计算  $L$  的第  $i$  列
11:    for  $j = k + 1$  to  $n$  do
12:       $a_{ij} = a_{ij} - a_{ik} * a_{kj}$    % 更新  $A(k + 1 : n, k + 1 : n)$ 
13:    end for

```





```

14:       $b_i = b_i - a_{ik}b_k$ 
15:  end for
16: end for
17:  $x_n = b_n/a_{nn}$  % 向后回代求解  $Ux = y$ 
18: for  $i = n - 1$  to 1 do
19:   for  $j = i + 1$  to  $n$  do
20:      $b_i = b_i - a_{ij}x_j$ 
21:   end for
22:    $x_i = b_i/a_{ii}$ 
23: end for

```

列主元 Gauss 消去法对应的矩阵分解称为 **PLU 分解**.

**定理 2.5 (列主元 LU 分解)** 若矩阵  $A$  非奇异, 则存在置换矩阵  $P$ , 使得

$$PA = LU, \quad (2.6)$$

其中  $L$  为单位下三角矩阵,  $U$  为上三角矩阵.

**证明.** 用归纳法.

当  $n = 1$  时, 取  $P = 1$ ,  $U = A$  即可.

假设结论对  $n - 1$  成立.

设  $A \in \mathbb{R}^{n \times n}$  是  $n$  阶非奇异矩阵, 则  $A$  的第一列至少存在一个非零元, 取置换矩阵  $\hat{P}_1$  使得

$$\hat{P}_1 A = \begin{bmatrix} a_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix},$$

其中  $a_{11} \neq 0$ ,  $A_{22} \in \mathbb{R}^{(n-1) \times (n-1)}$ . 记

$$u_{11} = a_{11}, \quad U_{12} = A_{12}, \quad L_{21} = A_{21}/a_{11}, \quad U_{22} = A_{22} - L_{21}U_{12}.$$

则有

$$\begin{bmatrix} 1 & 0 \\ L_{21} & I \end{bmatrix} \begin{bmatrix} u_{11} & U_{12} \\ 0 & U_{22} \end{bmatrix} = \begin{bmatrix} a_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} = \hat{P}_1 A.$$

两边取行列式可得

$$0 \neq \det(P_1 A) = \det \left( \begin{bmatrix} 1 & 0 \\ L_{21} & I \end{bmatrix} \right) \cdot \det \left( \begin{bmatrix} u_{11} & U_{12} \\ 0 & U_{22} \end{bmatrix} \right) = a_{11} \cdot \det(U_{22}).$$

所以  $\det(U_{22}) \neq 0$ , 即  $U_{22} \in \mathbb{R}^{(n-1) \times (n-1)}$  非奇异. 由归纳假设可知, 存在置换矩阵  $\tilde{P}_1$  使得

$$\tilde{P}_1 U_{22} = \tilde{L}_{22} \tilde{U}_{22} \quad \text{或} \quad U_{22} = \tilde{P}_1^T \tilde{L}_{22} \tilde{U}_{22},$$

其中  $\tilde{L}_{22}$  为单位下三角矩阵,  $\tilde{U}_{22}$  为非奇异上三角矩阵. 取

$$P_1 = \begin{bmatrix} 1 & 0 \\ 0 & \tilde{P}_1 \end{bmatrix} \hat{P}_1,$$

则有

$$P_1 A = \begin{bmatrix} 1 & 0 \\ 0 & \tilde{P}_1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ L_{21} & I \end{bmatrix} \begin{bmatrix} u_{11} & U_{12} \\ 0 & U_{22} \end{bmatrix}$$

$$\begin{aligned}
 &= \begin{bmatrix} 1 & 0 \\ \tilde{P}_1 L_{21} & \tilde{P}_1 \end{bmatrix} \begin{bmatrix} u_{11} & U_{12} \\ 0 & \tilde{P}_1^T \tilde{L}_{22} \tilde{U}_{22} \end{bmatrix} \\
 &= \begin{bmatrix} 1 & 0 \\ \tilde{P}_1 L_{21} & \tilde{P}_1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & \tilde{P}_1^T \tilde{L}_{22} \end{bmatrix} \begin{bmatrix} u_{11} & U_{12} \\ 0 & \tilde{U}_{22} \end{bmatrix} \\
 &= \begin{bmatrix} 1 & 0 \\ \tilde{P}_1 L_{21} & \tilde{L}_{22} \end{bmatrix} \begin{bmatrix} u_{11} & U_{12} \\ 0 & \tilde{U}_{22} \end{bmatrix} \\
 &\triangleq LU,
 \end{aligned}$$

其中  $L$  为单位下三角矩阵,  $U$  为非奇异上三角矩阵. 由归纳法可知, 结论成立.  $\square$

得到  $A$  的 PLU 分解 (2.6) 后, 线性方程组  $Ax = b$  就等价于下面两个三角线性方程组:

$$Ly = Pb, \quad Ux = y.$$

**例 2.4** 用部分选主元 LU 分解求解线性方程组  $Ax = b$ , 其中  $A = \begin{bmatrix} 0.02 & 61.3 \\ 3.43 & -8.5 \end{bmatrix}$ ,  $b = \begin{bmatrix} 61.5 \\ 25.8 \end{bmatrix}$ , 要求在运算过程中保留 3 位有效数字.

**解.** 由于  $|a_{21}| > |a_{11}|$ , 根据部分选主元 LU 分解算法, 我们需要将第一行与第二行交换, 即取  $P = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$ , 然后计算  $\tilde{A} = PA = \begin{bmatrix} 3.43 & -8.5 \\ 0.02 & 61.3 \end{bmatrix}$  的 LU 分解, 即设

$$\tilde{A} = LU = \begin{bmatrix} 1 & 0 \\ l_{21} & 1 \end{bmatrix} \begin{bmatrix} u_{11} & u_{12} \\ 0 & u_{22} \end{bmatrix}.$$

直接比较等式两边可得

$$\begin{aligned}
 u_{11} &= \tilde{a}_{11} = 3.43, \quad u_{12} = \tilde{a}_{12} = -8.5, \\
 l_{21} &= \tilde{a}_{21}/u_{11} \approx 5.83 \times 10^{-3}, \\
 u_{22} &= \tilde{a}_{22} - l_{21}u_{12} \approx 61.3 + 0.0496 \approx 61.3,
 \end{aligned}$$

即

$$PA \approx \begin{bmatrix} 1.00 & 0 \\ 5.83 \times 10^{-3} & 1.00 \end{bmatrix} \begin{bmatrix} 3.43 & -8.50 \\ 0 & 61.3 \end{bmatrix}.$$


解方程组  $Ly = Pb$  可得

$$y_1 = 25.8, \quad y_2 \approx 61.2.$$

解方程组  $Ux = y$  可得

$$x_2 = y_2/u_{22} \approx 0.998, \quad x_1 = (y_1 - u_{12}x_2)/u_{11} \approx 10.0.$$

所以, 数值解具有 3 位有效数字. (精确解为  $x_1 = 10, x_2 = 1$ )  $\square$

 列主元 Gauss 消去法比普通 Gauss 消去法要多做一些比较运算, 但列主元 Gauss 消去法 (1) 对系数矩阵要求低, 只需非奇异即可; (2) 比普通 Gauss 消去法更稳定.



列主元 Gauss 消去法是当前求解线性方程组的直接法中的首选算法.

### 全主元 Gauss 消去法

在列主元 Gauss 消去法中, 我们只在某一列中进行选主元, 比如第  $k$  步时的第  $k$  列  $A^{(k)}(k : n, k)$ . 事实上, 我们也可以在剩余的子矩阵中选取主元, 以获得更好的稳定性, 即在第  $k$  步时, 在  $A^{(k)}(k : n, k : n)$  中选取绝对值最大的元素作为主元. 此时可能需要同时做行交换和列交换, 以便将主元移到  $(k, k)$  位置.

- ① 选取 **全主元**:  $|a_{i_k, j_k}^{(k)}| = \max_{k \leq i, j \leq n} \{|a_{i, j}^{(k)}|\}$
- ② 行交换: 如果  $i_k \neq k$ , 则交换第  $k$  行与第  $i_k$  行
- ③ 列交换: 如果  $j_k \neq k$ , 则交换第  $k$  列与第  $j_k$  列

需要指出的是, 如果有列交换, 则会改变  $x_i$  的顺序, 因此需要记录每次的列交换次序.

**定理 2.6 (全主元 LU 分解)** 设矩阵  $A$  非奇异, 则存在置换矩阵  $P_l, P_r$ , 以及单位下三角矩阵  $L$  和非奇异上三角矩阵  $U$ , 使得

$$P_l A P_r = LU.$$

全主元高斯消去法具有更好的稳定性, 但很费时, 在实际计算中一般很少采用.

## 2.2 矩阵分解法

矩阵分解是矩阵计算中的一个非常重要的技术. 通过矩阵分解, 将原矩阵分解成若干个结构简单的矩阵的乘积, 从而将原本复杂的问题转化为若干个相对简单的问题. 这也是我们在实际生活中解决问题的一个基本思想.

### 2.2.1 LU 分解与 PLU 分解

如果存在一个单位下三角矩阵  $L$  和一个非奇异上三角矩阵  $U$ , 使得

$$A = LU,$$

则称  $A$  存在 **LU 分解**. 相对应的, 有 **Crout 分解**, 即存在非奇异下三角矩阵  $\tilde{L}$  和单位上三角矩阵  $\tilde{U}$ , 使得

$$A = \tilde{L}\tilde{U}.$$

另外, 还有 **LDR 分解**, 即存在单位下三角矩阵  $L$ , 单位上三角矩阵  $R$  和对角矩阵  $D$ , 使得

$$A = LDR.$$

显然, 这三种分解在本质上没有任何区别, 在实际计算中可以根据需要选择其中的一种. 这里我们只讨论 LU 分解.

除了利用 Gauss 消去过程来计算 LU 分解外, 还可以通过**待定系数法**来计算. 设  $A$  的 LU 分解为

$$A = LU \quad \text{即} \quad \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} = \begin{bmatrix} 1 & & & \\ l_{21} & 1 & & \\ \vdots & & \ddots & \\ l_{n1} & \cdots & l_{n,n-1} & 1 \end{bmatrix} \begin{bmatrix} u_{11} & u_{12} & \cdots & u_{1n} \\ & u_{22} & & u_{2n} \\ & & \ddots & \vdots \\ & & & u_{nn} \end{bmatrix}.$$

首先观察等式两边的第一行, 可知

$$u_{1j} = a_{1j}, \quad j = 1, 2, \dots, n.$$

即  $U$  的第一行就计算出来了. 再观察等式两边的第一列, 可得

$$l_{i1} = a_{i1}/u_{11}, \quad i = 2, 3, \dots, n.$$

于是  $L$  的第一列就计算出来了. 然后再观察等式两边的第二行, 可得

$$u_{2j} = a_{2j} - l_{21}u_{1j}, \quad j = 2, 3, \dots, n.$$

这就是  $U$  的第二行. 同理, 由等式两边的第二列可得

$$l_{i2} = (a_{i2} - l_{i1}u_{12})/u_{22}, \quad i = 3, 4, \dots, n.$$

这就是  $L$  的第二列. 依此类推, 我们就可以把  $L$  和  $U$  的所有元素都确定下来. 这就是待定系数法.

为了写出具体的算法, 我们先写出一般过程, 即第  $k$  步的计算公式. 此时  $U$  的前  $k-1$  行和  $L$  的前  $k-1$  列已经知道. 比较等式两边的第  $k$  行, 可得

$$u_{kj} = a_{kj} - (l_{k1}u_{1j} + l_{k2}u_{2j} + \cdots + l_{k,k-1}u_{k-1,j}), \quad j = k, k+1, \dots$$



比较等式两边的第  $k$  列, 可得

$$l_{ik} = \frac{1}{u_{kk}}(a_{ik} - l_{i1}u_{1k} - \cdots - l_{i,k-1}u_{k-1,k}), \quad i = k+1, k+2, \dots, n.$$

由此, 我们就可以写出具体的算法. 同样地, 我们将  $L$  存放在  $A$  的下三角部分,  $U$  存放在  $A$  的上三角部分.

#### 算法 2.5. LU 分解 (待定系数法)

```

1: for  $k = 1$  to  $n$  do
2:   for  $j = k$  to  $n$  do
3:     for  $i = 1$  to  $k - 1$  do
4:        $a_{kj} = a_{kj} - a_{ki}a_{ij}$ 
5:     end for
6:   end for
7:   for  $i = k + 1$  to  $n$  do
8:     for  $j = 1$  to  $k - 1$  do
9:        $a_{ik} = a_{ik} - a_{ij}a_{jk}$ 
10:    end for
11:     $a_{ik} = a_{ik} / a_{kk}$ 
12:  end for
13: end for

```

这种通过待定系数法计算 LU 分解的算法也称为 **Doolittle 方法**.

最后, 需要求解两个三角线性方程组, 即  $Ly = b$  和  $Ux = y$ . 完整求解过程可描述如下.

#### 算法 2.6. LU 分解求解线性方程组

```

1: 计算  $A$  的 LU 分解 (此处省略, 可参见算法 2.5)
2:  $y_1 = b_1$    % 向前回代求解  $Ly = b$ 
3: for  $i = 2$  to  $n$  do
4:   for  $k = 1$  to  $i - 1$  do
5:      $b_i = b_i - a_{ik}y_k$ 
6:   end for
7:    $y_i = b_i$ 
8: end for
9:  $x_n = y_n / a_{nn}$    % 向后回代求解  $Ux = y$ 
10: for  $i = n - 1$  to  $1$  do
11:   for  $k = i + 1$  to  $n$  do
12:      $y_i = y_i - a_{ik}x_k$ 
13:   end for

```

```

14:    $x_i = y_i/a_{ii}$ 
15: end for

```

下面考虑列主元 LU 分解 (PLU), 即

$$PA = LU,$$

其中  $P$  是一个置换矩阵. 因此,  $PA$  就相当于对  $A$  的行进行了重新排列. 为了节省存储量, 我们并不存储这个置换矩阵, 而只是用一个向量来表示这个重新排列. 我们将这个向量记为  $p$ , 其元素是  $\{1, 2, \dots, n\}$  的一个重排列. 比如  $p = [1, 3, 2]$  表示交换矩阵的第 2 行和第 3 行, 而  $p = [2, 3, 1]$  则表示将矩阵的第 2 行移到最前面, 将第 3 行移到第 2 行, 将第 1 行移到最后.

易知, 在计算过程中,  $p$  的初始值为  $p = [1, 2, \dots, n]$ . 在选列主元的过程中, 如果出现行交换, 即交换第  $i_k$  行和第  $k$  行, 则需要相应地交换  $p$  的第  $i_k$  位置和第  $k$  位置上的值. 具体算法如下.

#### 算法 2.7. PLU: 列主元 LU 分解或部分选主元 LU 分解

```

1:  $p = [1, 2, \dots, n]$    % 用于记录置换矩阵
2: for  $k = 1$  to  $n - 1$  do
3:    $a_{i_k, k} = \max_{k \leq i \leq n} |a_{i, k}|$    % 选列主元
4:   if  $i_k \neq k$  then
5:     for  $j = 1$  to  $n$  do
6:        $a_{tmp} = a_{i_k, j}, a_{i_k, j} = a_{k, j}, a_{k, j} = a_{tmp}$    % 交换  $A$  的第  $i_k$  行与第  $k$  行
7:     end for
8:      $p_{tmp} = p_{i_k}, p_{i_k} = p_k, p_k = p_{tmp}$    % 更新置换矩阵
9:   end if
10:  for  $i = k + 1$  to  $n$  do
11:     $a_{ik} = a_{ik}/a_{kk}$    % 计算  $L$  的第  $i$  列
12:    for  $j = k + 1$  to  $n$  do
13:       $a_{ij} = a_{ij} - a_{ik} * a_{ij}$    % 更新  $A(k + 1 : n, k + 1 : n)$ 
14:    end for
15:  end for
16: end for

```

将求解两个三角线性方程组结合起来, 就得到完整的求解过程.

#### 算法 2.8. PLU 分解求解线性方程组

```

1: 计算  $A$  的 PLU 分解 (此处省略, 可参见算法 2.7)
2:  $y_1 = b_{p_1}$    % 向前回代求解  $Ly = Pb$ 
3: for  $i = 2$  to  $n$  do
4:   for  $j = 1$  to  $i - 1$  do


```



```

5:       $b_{p_i} = b_{p_i} - a_{ij}y_j$ 
6:  end for
7:       $y_i = b_{p_i}$ 
8:  end for
9:   $x_n = b_{p_n}/a_{nn}$  % 向后回代求解  $Ux = y$ 
10: for  $i = n - 1$  to 1 do
11:     for  $j = i + 1$  to  $n$  do
12:         $y_i = y_i - a_{ij}x_j$ 
13:     end for
14:      $x_i = y_i/a_{ii}$ 
15: end for

```

 算法 2.8 可用于计算矩阵的逆, 也可用于计算矩阵的行列式.

### 2.2.2 Cholesky 分解与平方根法

前面讨论的是一般线性方程组, 并没有考虑系数矩阵的特殊结构. 如果  $A$  是对称正定的, 则可以得到更加简洁高效的方法.

**定理 2.7 (Cholesky 分解)** 设  $A$  对称正定, 则存在唯一的对角线元素全为正的下三角矩阵  $L$ , 使得

$$A = LL^T.$$

该分解称为 Cholesky 分解.

**证明.** 首先证明存在性, 我们用数学归纳法来构造矩阵  $L$ .

当  $n = 1$  时, 由  $A$  的对称正定性可知  $a_{11} > 0$ . 取  $l_{11} = \sqrt{a_{11}}$  即可.

假定结论对所有不超过  $n - 1$  阶的对称正定矩阵都成立. 设  $A \in \mathbb{R}^{n \times n}$  是  $n$  阶对称正定, 则  $A$  可分解为

$$A = \begin{bmatrix} a_{11} & A_{12} \\ A_{12}^T & A_{22} \end{bmatrix} = \begin{bmatrix} \sqrt{a_{11}} & 0 \\ \frac{1}{\sqrt{a_{11}}}A_{12}^T & I \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & \tilde{A}_{22} \end{bmatrix} \begin{bmatrix} \sqrt{a_{11}} & 0 \\ \frac{1}{\sqrt{a_{11}}}A_{12}^T & I \end{bmatrix}^T,$$

其中  $\tilde{A}_{22} = A_{22} - A_{12}^T A_{12} / a_{11}$ . 易知,  $\begin{bmatrix} 1 & 0 \\ 0 & \tilde{A}_{22} \end{bmatrix}$  对称正定, 故  $\tilde{A}_{22}$  是  $n - 1$  阶对称正定矩阵. 根据归纳假设, 存在唯一的对角线元素为正的下三角矩阵  $\tilde{L}$ , 使得  $\tilde{A}_{22} = \tilde{L}\tilde{L}^T$ . 令

$$L = \begin{bmatrix} \sqrt{a_{11}} & 0 \\ \frac{1}{\sqrt{a_{11}}}A_{12}^T & I \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & \tilde{L} \end{bmatrix} = \begin{bmatrix} \sqrt{a_{11}} & 0 \\ \frac{1}{\sqrt{a_{11}}}A_{12}^T & \tilde{L} \end{bmatrix}.$$


易知,  $L$  是对角线元素均为正的下三角矩阵, 且

$$LL^T = \begin{bmatrix} \sqrt{a_{11}} & 0 \\ \frac{1}{\sqrt{a_{11}}}A_{12}^T & I \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & \tilde{L} \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & \tilde{L}^T \end{bmatrix} \begin{bmatrix} \sqrt{a_{11}} & 0 \\ \frac{1}{\sqrt{a_{11}}}A_{12}^T & I \end{bmatrix}^T = A.$$

由归纳法可知, 对任意对称正定实矩阵  $A$ , 都存在一个对角线元素为正的下三角矩阵  $L$ , 使得

$$A = LL^T.$$

唯一性可以采用反证法, 留做练习. □

 Cholesky 分解仅针对对称正定矩阵成立.

**定理 2.8** 若  $A$  对称, 且所有顺序主子式都不为 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.

(证明留作课外练习)

### 如何计算 Cholesky 分解

我们仍然使用待定系数法. 设

$$A = LL^T \quad \text{即} \quad \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} = \begin{bmatrix} l_{11} & & & \\ l_{21} & l_{22} & & \\ \vdots & \vdots & \ddots & \\ l_{n1} & l_{n2} & \cdots & l_{nn} \end{bmatrix} \begin{bmatrix} l_{11} & l_{21} & \cdots & l_{n1} \\ & l_{22} & \cdots & l_{n2} \\ & & \ddots & \vdots \\ & & & l_{nn} \end{bmatrix}.$$

类似于 LU 分解, 直接比较等式两边的元素, 可得一般公式

$$a_{ij} = \sum_{k=1}^n l_{ik}l_{jk} = \sum_{k=1}^{j-1} l_{ik}l_{jk} + l_{jj}l_{ij}, \quad j = 1, 2, \dots, n, \quad i = j, j+1, \dots, n.$$

根据这个计算公式即可得下面的算法 (这里我们将  $L$  存放在  $A$  的下三角部分).

#### 算法 2.9. Cholesky 分解

```

1: for  $j = 1$  to  $n$  do
2:   for  $k = 1$  to  $j - 1$  do
3:      $a_{jj} = a_{jj} - a_{jk}^2$ 
4:   end for
5:    $a_{jj} = \sqrt{a_{jj}}$ 
6:   for  $i = j + 1$  to  $n$  do
7:     for  $k = 1$  to  $j - 1$  do
8:        $a_{ij} = a_{ij} - a_{ik}a_{jk}$ 
9:     end for
10:     $a_{ij} = a_{ij}/a_{jj}$ 
11:  end for
12: end for
  
```





Cholesky 分解算法的乘除运算量为  $\frac{1}{3}n^3 + \mathcal{O}(n^2)$ , 大约为 LU 分解的一半. 另外, Cholesky 分解算法是稳定的 (稳定性与全主元 Gauss 消去法相当), 因此不需要选主元.

利用 Cholesky 分解求解线性方程组的方法就称为 **平方根法**, 具体描述如下:

#### 算法 2.10. Cholesky 分解求解线性方程组

```

1: 计算 Cholesky 分解 (此处省略, 参见算法 2.9)
2:  $y_1 = b_1/a_{11}$    % 向前回代求解  $Ly = b$ 
3: for  $i = 2$  to  $n$  do
4:   for  $j = 1$  to  $i - 1$  do
5:      $b_i = b_i - a_{ij}y_j$ 
6:   end for
7:    $y_i = b_i/a_{ii}$ 
8: end for
9:  $x_n = b_n/a_{nn}$    % 向后回代求解  $L^T x = y$ 
10: for  $i = n - 1$  to  $1$  do
11:   for  $j = i + 1$  to  $n$  do
12:      $y_i = y_i - a_{ji}x_j$ 
13:   end for
14:    $x_i = y_i/a_{ii}$ 
15: end for

```

### LDL<sup>T</sup> 分解

在 Cholesky 分解中, 需要计算平方根. 为了避免计算平方根, 我们可以采用改进的 Cholesky 分解, 即

$$A = LDL^T,$$

其中  $L$  为单位下三角矩阵,  $D$  为对角矩阵. 这个分解也称为 **LDL<sup>T</sup> 分解**. 由待定系数法, 设

$$A = LDL^T = \begin{bmatrix} 1 & & & \\ l_{21} & 1 & & \\ \vdots & \ddots & \ddots & \\ l_{n1} & \cdots & l_{n,n-1} & 1 \end{bmatrix} \begin{bmatrix} d_1 & & & \\ & d_2 & & \\ & & \ddots & \\ & & & d_n \end{bmatrix} \begin{bmatrix} 1 & l_{21} & \cdots & l_{n1} \\ & 1 & \ddots & \vdots \\ & & \ddots & l_{n,n-1} \\ & & & 1 \end{bmatrix}.$$

比较等式两边的元素, 可得

$$a_{ij} = d_j l_{ij} + \sum_{k=1}^{j-1} l_{ik} d_k l_{jk}, \quad j = 1, 2, \dots, n, \quad i = j + 1, j + 2, \dots, n.$$

基于以上分解的求解对称正定线性方程组的算法就称为**改进的平方根法**, 描述如下 (将  $L$  存放在  $A$  的下三角部分,  $D$  存放在  $A$  的对角部分).

**算法 2.11.** 改进的平方根法

```

1: for  $j = 1$  to  $n$  do
2:   for  $k = 1$  to  $j - 1$  do
3:      $a_{jj} = a_{jj} - l_{jk}^2 a_{kk}$ 
4:   end for
5:   for  $i = j + 1$  to  $n$  do
6:     for  $k = 1$  to  $j - 1$  do
7:        $a_{ij} = a_{ij} - a_{ik} a_{kk} a_{jk}$ 
8:     end for
9:      $a_{ij} = a_{ij} / a_{jj}$ 
10:  end for
11: end for
12:  $y_1 = b_1$  % 向前回代求解  $Ly = b$ 
13: for  $i = 2$  to  $n$  do
14:   for  $j = 1$  to  $i - 1$  do
15:      $b_i = b_i - a_{ij} y_j$ 
16:   end for
17:    $y_i = b_i$ 
18: end for
19:  $x_n = b_n / a_{nn}$  % 向后回代求解  $DL^T x = y$ 
20: for  $i = n - 1$  to  $1$  do
21:    $x_i = y_i / a_{ii}$ 
22:   for  $j = i + 1$  to  $n$  do
23:      $x_i = x_i - a_{ji} x_j$ 
24:   end for
25: end for

```

**2.2.3 三对角线性方程组**

这里考虑一类具有简单结构的线性方程组, 即三对角线性方程组. 在计算样条插值时会遇到这类线性方程组. 由于系数矩阵的特殊结构, 我们可以设计更加简洁高效的求解方法.

考虑线性方程组  $Ax = f$ , 其中  $A$  是三对角矩阵:

$$A = \begin{bmatrix} b_1 & c_1 & & & \\ a_1 & \ddots & \ddots & & \\ & \ddots & \ddots & c_{n-1} & \\ & & a_{n-1} & b_n & \end{bmatrix}.$$



我们假定

$$|b_1| > |c_1| > 0, \quad |b_n| > |a_{n-1}| > 0, \quad (2.7)$$

且

$$|b_i| \geq |a_{i-1}| + |c_i|, \quad a_i c_i \neq 0, \quad i = 1, \dots, n-1. \quad (2.8)$$

即  $A$  是**不可约弱对角占优**的 (不可约见定义 4.6, 对角占优见定义 4.7). 此时, 我们可以得到下面的三角分解 (Crout 分解)

$$A = \begin{bmatrix} b_1 & c_1 & & \\ a_1 & \ddots & \ddots & \\ & \ddots & \ddots & c_{n-1} \\ & & a_{n-1} & b_n \end{bmatrix} = \begin{bmatrix} \alpha_1 & & & \\ a_1 & \alpha_2 & & \\ & \ddots & \ddots & \\ & & a_{n-1} & \alpha_n \end{bmatrix} \begin{bmatrix} 1 & \beta_1 & & \\ & 1 & \ddots & \\ & & \ddots & \beta_{n-1} \\ & & & 1 \end{bmatrix} \triangleq LU. \quad (2.9)$$

由待定系数法, 我们可以得到递推公式:

$$\begin{aligned} \alpha_1 &= b_1, \quad \beta_1 = c_1/\alpha_1 = c_1/b_1, \\ \begin{cases} \alpha_i = b_i - a_{i-1}\beta_{i-1}, \\ \beta_i = c_i/\alpha_i = c_i/(b_i - a_{i-1}\beta_{i-1}), \end{cases} & i = 2, 3, \dots, n-1 \\ \alpha_n &= b_n - a_{n-1}\beta_{n-1}. \end{aligned}$$

为了使得算法能够顺利进行下去, 我们需要证明  $\alpha_i \neq 0$ .

**定理 2.9** 设三对角矩阵  $A$  满足条件 (2.7) 和 (2.8). 则  $A$  非奇异, 且

- (1)  $|\alpha_1| = |b_1| > 0$ ;
- (2)  $0 < |\beta_i| < 1, i = 1, 2, \dots, n-1$ ;
- (3)  $0 < |c_i| \leq |b_i| - |a_{i-1}| < |\alpha_i| < |b_i| + |a_{i-1}|, i = 2, 3, \dots, n$ ;

**证明.** 由于  $A$  是不可约且弱对角占优, 所以  $A$  非奇异.

结论 (1) 是显然的.

下面我们证明结论 (2) 和 (3).

由于  $0 < |c_1| < |b_1|$ , 且  $\beta_1 = c_1/b_1$ , 所以  $0 < |\beta_1| < 1$ . 又  $\alpha_2 = b_2 - a_1\beta_1$ , 所以

$$|\alpha_2| \geq |b_2| - |a_1| \cdot |\beta_1| > |b_2| - |a_1| \geq |c_2| > 0, \quad (2.10)$$

$$|\alpha_2| \leq |b_2| + |a_1| \cdot |\beta_1| < |b_2| + |a_1|. \quad (2.11)$$

再由结论 (2.10) 和  $\beta_2$  的计算公式可知  $0 < |\beta_2| < 1$ . 类似于 (2.10) 和 (2.11), 我们可以得到

$$|\alpha_3| \geq |b_3| - |a_2| \cdot |\beta_2| > |b_3| - |a_2| \geq |c_3| > 0,$$

$$|\alpha_3| \leq |b_3| + |a_2| \cdot |\beta_2| < |b_3| + |a_2|.$$

依此类推, 我们就可以证明结论 (2) 和 (3). □

由定理 2.9 可知, 分解 (2.9) 是存在的. 因此, 原方程就转化为求解  $Ly = f$  和  $Ux = y$ . 由此便可得求解三对角线性方程组的**追赶法** 也称为 **Thomas 算法** (1949), 其乘除运算量大约为  $5n$ , 加减运算大约为  $3n$ .

**算法 2.12.** 追赶法

```

1:  $\alpha_1 = b_1$ 
2:  $\beta_1 = c_1/b_1$ 
3:  $y_1 = f_1/b_1$ 
4: for  $i = 2$  to  $n - 1$  do
5:    $\alpha_i = b_i - a_{i-1}\beta_{i-1}$ 
6:    $\beta_i = c_i/\alpha_i$ 
7:    $y_i = (f_i - a_{i-1}y_{i-1})/\alpha_i$ 
8: end for
9:  $\alpha_n = b_n - a_{n-1}\beta_{n-1}$ 
10:  $y_n = (f_n - a_{n-1}y_{n-1})/\alpha_n$ 
11:  $x_n = y_n$ 
12: for  $i = n - 1$  to  $1$  do
13:    $x_i = y_i - \beta_i x_{i+1}$ 
14: end for

```

具体计算时, 由于求解  $Ly = f$  与矩阵 LU 分解是同时进行的, 因此,  $\alpha_i$  可以不用存储. 但  $\beta_i$  需要存储.

由于  $|\beta_i| < 1$ , 因此在回代求解  $x_i$  时, 误差可以得到有效控制.

需要指出的是, 我们也可以考虑下面的分解

$$A = \begin{bmatrix} b_1 & c_1 & & \\ a_1 & \ddots & \ddots & \\ & \ddots & \ddots & c_{n-1} \\ & & a_{n-1} & b_n \end{bmatrix} = \begin{bmatrix} 1 & & & \\ \gamma_1 & 1 & & \\ & \ddots & \ddots & \\ & & \gamma_{n-1} & 1 \end{bmatrix} \begin{bmatrix} \alpha_1 & c_1 & & \\ & \alpha_2 & \ddots & \\ & & \ddots & c_{n-1} \\ & & & \alpha_n \end{bmatrix}. \quad (2.12)$$

但此时  $|\gamma_i|$  可能大于 1. 比如  $\gamma_1 = a_1/b_1$ , 因此当  $|b_1| < |a_1|$  时,  $|\gamma_1| > 1$ . 所以在回代求解时, 误差可能得不到有效控制. 另外一方面, 计算  $\gamma_i$  时也可能会产生较大的舍入误差 (大数除以小数). 但如果  $A$  是列对角占优, 则可以保证  $|\gamma_i| < 1$ .

如果  $A$  是 (行) 对角占优, 则采用分解 (2.9); 如果  $A$  是列对角占优, 则采用分解 (2.12).

**2.2.4 带状线性方程组**

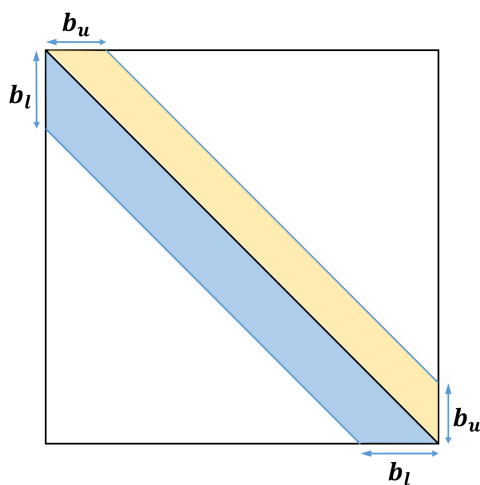
在实际应用中会经常遇到带状线性方程组, 在设计算法时应该充分利用其中的零元素来减少运算量, 从而大幅度地提高计算效率.

设  $A \in \mathbb{R}^{n \times n}$  是带状矩阵, 其下带宽为  $b_l$ , 上带宽为  $b_u$ , 即当  $i > j + b_l$  或  $i < j - b_u$  时有

$$a_{ij} = 0.$$



其形状如下图所示:



对于带状矩阵, 其 LU 分解有如下性质:

**定理 2.10** 设  $A \in \mathbb{R}^{n \times n}$  是带状矩阵, 其下带宽为  $b_l$ , 上带宽为  $b_u$ . 若  $A = LU$  是不选主元的 LU 分解, 则  $L$  为下带宽为  $b_l$  的带状矩阵,  $U$  为上带宽为  $b_u$  的带状矩阵.

若采用部分选主元的 LU 分解, 则有

**定理 2.11** 设  $A \in \mathbb{R}^{n \times n}$  是带状矩阵, 其下带宽为  $b_l$ , 上带宽为  $b_u$ . 若  $PA = LU$  是部分选主元的 LU 分解, 则  $U$  为上带宽不超过  $b_l + b_u$  的带状矩阵,  $L$  为下带宽为  $b_l$  的“基本带状矩阵”, 即  $L$  每列的非零元素不超过  $b_l + 1$  个. (也就是说, 非零元素的个数有限制, 但位置不限)

## 2.3 扰动分析

### 2.3.1 矩阵条件数

在实际应用中, 所给的数据可能是通过实验、测量或观察得来的, 因此通常会带有一定的误差, 这些误差不可避免地会对问题的解产生影响. 考虑线性方程组  $Ax = b$ , 如果  $A$  或  $b$  的微小变化会导致解的巨大变化, 则称此线性方程组是 **病态** 的, 反之则是 **良态** 的.

**例 2.5** 考虑线性方程组  $Ax = b$ , 其中  $A = \begin{bmatrix} 1 & 1 \\ 1 & 1.0001 \end{bmatrix}$ ,  $b = [2, 2]^T$ . 可求得解为  $x = [2, 0]^T$ .

如果  $b$  的第二个元素出现细微误差, 变为  $b = [2, 2.0001]^T$ , 则解变为  $x = [1, 1]^T$ .

由此可见, 当右端项出现细微变化时, 解会出现很大的变化, 因此该线性方程组是病态的.

对于线性方程组而言, 问题是否变态主要取决于系数矩阵是否病态. 怎样来判断一个矩阵是否病态? 目前比较常用的一个重要指标就是 **矩阵条件数**.

**定义 2.1** 设  $A$  非奇异,  $\|\cdot\|$  是任一算子范数, 则称

$$\kappa(A) \triangleq \|A^{-1}\| \cdot \|A\|$$

为  $A$  的 **条件数**.

常用的矩阵条件数有

$$\kappa_2(A) \triangleq \|A^{-1}\|_2 \cdot \|A\|_2, \quad \kappa_1(A) \triangleq \|A^{-1}\|_1 \cdot \|A\|_1, \quad \kappa_\infty(A) \triangleq \|A^{-1}\|_\infty \cdot \|A\|_\infty.$$

$\kappa_2(A)$  也称为 **谱条件数**, 当  $A$  对称时, 有

$$\kappa_2(A) = \frac{\max_{1 \leq i \leq n} |\lambda_i|}{\min_{1 \leq i \leq n} |\lambda_i|}.$$

**例 2.6** 计算例 2.5 中系数矩阵  $A$  的条件数  $\kappa_\infty(A)$  和  $\kappa_2(A)$ .

**解.** 通过直接计算可得

$$A^{-1} = \begin{bmatrix} 10001 & -10000 \\ -10000 & 10000 \end{bmatrix}.$$

所以

$$\kappa_\infty(A) = \|A\|_\infty \cdot \|A^{-1}\|_\infty \approx 4 \times 10^4.$$

由于  $A$  是对称的, 且  $A$  的特征值为

$$\lambda(A) = \frac{2.0001 \pm \sqrt{2.0001^2 - 0.0004}}{2} > 0,$$



所以

$$\kappa_2(A) = \frac{\lambda_{\max}}{\lambda_{\min}} \approx 4 \times 10^4.$$

□

一般情况下, 如果矩阵非对角线元素比较大, 则矩阵很有可能是病态的.

**例 2.7** 已知矩阵

$$A = \begin{bmatrix} 1 & -0.5 & \cdots & -0.5 \\ & 1 & \ddots & \vdots \\ & & \ddots & -0.5 \\ & & & 1 \end{bmatrix} \in \mathbb{R}^{n \times n}.$$

直接计算可知

$$A^{-1} = \begin{bmatrix} 1 & \frac{1}{2} & \frac{1.5}{2} & \frac{1.5^2}{2} & \cdots & \frac{1.5^{n-2}}{2} \\ & 1 & \frac{1}{2} & \frac{1.5}{2} & \ddots & \vdots \\ & & 1 & \ddots & \ddots & \frac{1.5^2}{2} \\ & & & \ddots & \frac{1}{2} & \frac{1.5}{2} \\ & & & & 1 & \frac{1}{2} \\ & & & & & 1 \end{bmatrix}.$$

因此,  $\kappa_1(A)$  和  $\kappa_\infty(A)$  是矩阵维数  $n$  的幂函数. 所以, 随着  $n$  的增大, 条件数增长会非常快. 下表中是  $n = 10, 20, 30, 40, 50$  时  $\kappa_1(A)$  和  $\kappa_2(A)$  的值.

$n$	10	20	30	40	50
$\kappa_1(A)$	$2.1 \times 10^2$	$2.3 \times 10^4$	$2.0 \times 10^6$	$1.5 \times 10^8$	$1.1 \times 10^{10}$
$\kappa_2(A)$	$6.3 \times 10$	$7.6 \times 10^3$	$6.8 \times 10^5$	$5.5 \times 10^7$	$3.9 \times 10^9$

**引理 2.12** 条件数具有以下性质:

- $\kappa(A) \geq 1, \quad \forall A \in \mathbb{R}^{n \times n}.$
- 对任意非零常数  $\alpha \in \mathbb{R}$ , 都有

$$\kappa(\alpha A) = \kappa(A).$$

- 对任意正交矩阵  $Q \in \mathbb{R}^{n \times n}$ , 有  $\kappa_2(Q) = 1$ .
- 设  $Q$  是正交矩阵, 则对任意矩阵  $A$  有

$$\kappa_2(QA) = \kappa_2(A) = \kappa_2(AQ).$$

(证明留作练习)

### 2.3.2 条件数与病态之间的关系

考虑线性方程组  $Ax = b$ . 假定系数矩阵  $A$  是精确的, 而右端项  $b$  有个微小扰动  $\delta b$ . 因此我们实际求解的是线性方程组

$$Ax = b + \delta b.$$

我们称这个方程组为 **扰动方程组**, 其解记为  $\tilde{x}$ .

记  $\delta x \triangleq \tilde{x} - x_*$ , 其中  $x_* = A^{-1}b$  为精确解. 则

$$\delta x = A^{-1}(b + \delta b) - x_* = A^{-1}\delta b.$$

所以

$$\|\delta x\| \leq \|A^{-1}\| \cdot \|\delta b\|. \quad (2.13)$$

又由  $Ax_* = b$  可知

$$\|b\| \leq \|A\| \cdot \|x_*\|, \quad \text{即} \quad \frac{1}{\|x_*\|} \leq \frac{\|A\|}{\|b\|}. \quad (2.14)$$

由 (2.13) 和 (2.14) 两边相乘可得

$$\frac{\|\delta x\|}{\|x_*\|} \leq \|A^{-1}\| \cdot \|A\| \cdot \frac{\|\delta b\|}{\|b\|}$$

**定理 2.13** 设  $\|\cdot\|$  是任一向量范数 (当该范数作用在矩阵上时就是相应的算子范数), 若  $A$  是精确的,  $b$  有个小扰动  $\delta b$ , 则有

$$\frac{\|\delta x\|}{\|x_*\|} \leq \|A^{-1}\| \cdot \|A\| \cdot \frac{\|\delta b\|}{\|b\|}.$$

上述结论说明, 由于右端项的扰动而产生的解的相对误差, 大约被放大了  $\|A^{-1}\| \cdot \|A\|$  倍. 这个倍数正好是系数矩阵的条件数. 另外, 需要指出的是, 这是最坏的情况.

如果  $A$  也有微小的扰动, 设为  $\delta A$ , 则扰动方程组为

$$(A + \delta A)\tilde{x} = b + \delta b.$$

假定  $\|\delta A\|$  很小, 满足  $\|A^{-1}\| \cdot \|\delta A\| < 1$ , 则  $\|A^{-1}\delta A\| \leq \|A^{-1}\| \cdot \|\delta A\| < 1$ . 由定理 1.20 可知  $I + A^{-1}\delta A$  非奇异, 所以  $A + \delta A = A(I + A^{-1}\delta A)$  也非奇异. 于是

$$\begin{aligned} \delta x &= \tilde{x} - x_* = (A + \delta A)^{-1}(b + \delta b - Ax_* - \delta Ax_*) \\ &= (I + A^{-1}\delta A)^{-1}A^{-1}(-\delta Ax_* + \delta b). \end{aligned}$$

由定理 1.20 可知

$$\begin{aligned} \frac{\|\delta x\|}{\|x_*\|} &\leq \|(I + A^{-1}\delta A)^{-1}\| \cdot \|A^{-1}\| \cdot \left( \|\delta A\| + \frac{\|\delta b\|}{\|x_*\|} \right) \\ &\leq \frac{\|A^{-1}\|}{1 - \|A^{-1}\| \cdot \|\delta A\|} \cdot \left( \|\delta A\| + \frac{\|\delta b\|}{\|x_*\|} \right) \\ &= \frac{\|A^{-1}\| \cdot \|A\|}{1 - \|A^{-1}\| \cdot \|A\| \cdot \frac{\|\delta A\|}{\|A\|}} \left( \frac{\|\delta A\|}{\|A\|} + \frac{\|\delta b\|}{\|A\| \cdot \|x_*\|} \right) \end{aligned}$$





$$\leq \frac{\kappa(A)}{1 - \kappa(A) \cdot \frac{\|\delta A\|}{\|A\|}} \left( \frac{\|\delta A\|}{\|A\|} + \frac{\|\delta b\|}{\|b\|} \right)$$

当  $\|\delta A\| \rightarrow 0$  时, 我们可得

$$\frac{\|\delta x\|}{\|x_*\|} \leq \frac{\kappa(A)}{1 - \kappa(A) \cdot \frac{\|\delta A\|}{\|A\|}} \left( \frac{\|\delta A\|}{\|A\|} + \frac{\|\delta b\|}{\|b\|} \right) \rightarrow \kappa(A) \left( \frac{\|\delta A\|}{\|A\|} + \frac{\|\delta b\|}{\|b\|} \right)$$

**定理 2.14** 设  $\|\cdot\|$  是任一向量范数 (当该范数作用在矩阵上时就是相应的算子范数), 假定  $\|A^{-1}\| \cdot \|\delta A\| < 1$ , 则有

$$\frac{\|\delta x\|}{\|x_*\|} \leq \frac{\kappa(A)}{1 - \kappa(A) \cdot \frac{\|\delta A\|}{\|A\|}} \left( \frac{\|\delta A\|}{\|A\|} + \frac{\|\delta b\|}{\|b\|} \right).$$

当  $\delta b = 0$  时, 有

$$\frac{\|\delta x\|}{\|x_*\|} \leq \frac{\kappa(A)}{1 - \kappa(A) \cdot \frac{\|\delta A\|}{\|A\|}} \cdot \frac{\|\delta A\|}{\|A\|}.$$

事实上, 由于  $x_*$  通常是未知的, 因此一个更加实际的情况是考虑  $\delta x$  与  $\tilde{x}$  之间的关系.

**定理 2.15** 设  $\|\cdot\|$  是任一向量范数 (当该范数作用在矩阵上时就是相应的算子范数), 则  $\delta x$  与  $\tilde{x}$  满足下面的关系式

$$\frac{\|\delta x\|}{\|\tilde{x}\|} \leq \|A^{-1}\| \cdot \|A\| \left( \frac{\|\delta A\|}{\|A\|} + \frac{\|\delta b\|}{\|A\| \cdot \|\tilde{x}\|} \right).$$

当  $\delta b = 0$  时, 有

$$\frac{\|\delta x\|}{\|\tilde{x}\|} \leq \kappa(A) \frac{\|\delta A\|}{\|A\|} \quad (2.15)$$

**证明.** 由等式  $(A + \delta A)\tilde{x} = b + \delta b = Ax_* + \delta b$  可知  $A(\tilde{x} - x_*) = -\delta A\tilde{x} + \delta b$ , 即

$$\delta x = A^{-1}(-\delta A\tilde{x} + \delta b).$$

所以

$$\|\delta x\| \leq \|A^{-1}\| \cdot (\|\delta A\| \cdot \|\tilde{x}\| + \|\delta b\|), \quad (2.16)$$

即

$$\frac{\|\delta x\|}{\|\tilde{x}\|} \leq \|A^{-1}\| \cdot \|A\| \left( \frac{\|\delta A\|}{\|A\|} + \frac{\|\delta b\|}{\|A\| \cdot \|\tilde{x}\|} \right).$$

若  $\delta b = 0$ , 则可得

$$\frac{\|\delta x\|}{\|\tilde{x}\|} \leq \kappa(A) \frac{\|\delta A\|}{\|A\|}$$

□

上面的结论具有理论指导作用, 但如果无法获得  $\delta A$  和  $\delta b$  的大小, 则很难用来估计  $\delta x$  的大小. 此时, 我们可以通过残量来估计.

记残量 (残差) 为  $r \triangleq b - A\tilde{x}$ , 则有

$$\delta x = \tilde{x} - x_* = \tilde{x} - A^{-1}b = A^{-1}(A\tilde{x} - b) = -A^{-1}r,$$

所以可得

$$\|\delta x\| \leq \|A^{-1}\| \cdot \|r\|$$

这个估计式的优点是不需要知道  $\delta A$  和  $\delta b$  的大小. 而且在实际计算中,  $r$  通常是可计算的, 因此该估计式比较实用.

**例 2.8** Hilbert 矩阵是一个典型的病态矩阵, 其定义如下

$$H_n = \begin{bmatrix} 1 & \frac{1}{2} & \cdots & \frac{1}{n} \\ \frac{1}{2} & \frac{1}{3} & \cdots & \frac{1}{n+1} \\ \vdots & \vdots & & \vdots \\ \frac{1}{n} & \frac{1}{n+1} & \cdots & \frac{1}{2n-1} \end{bmatrix}, \quad \text{即} \quad H_n = [h_{ij}], \quad h_{ij} = \frac{1}{i+j-1}, \quad i, j = 1, 2, \dots, n.$$

可以验证  $H_n$  是对称正定的. 通过计算可知

$$\kappa_\infty(H_2) = 27, \quad \kappa_\infty(H_3) = 748, \quad \kappa_\infty(H_4) = 28375, \quad \kappa_\infty(H_2) \approx 3.5 \times 10^{13}.$$

由此可知, 当  $n$  较大时, 矩阵条件数非常大.

考虑线性方程组

$$H_n x = b,$$

设精确解为  $x_* = [1, 1, \dots, 1]^T$ , 计算出右端项  $b$ , 然后用 Cholesky 分解求解该线性方程组, 求得的近似解记为  $\tilde{x}$ .

对于不同的  $n$ , 下表中列出了近似解的误差 (见练习 2.21).

$n$	5	10	20	30
$\ \tilde{x} - x_*\ _\infty$	$2.3 \times 10^{-11}$	$6.2 \times 10^{-4}$	$> 7.8$	$> 42.8$
$\kappa_\infty(H_n)$	$9.4 \times 10^5$	$3.5 \times 10^{13}$	$> 10^{19}$	$> 10^{19}$

由此可知, 当  $n \geq 20$  时, Cholesky 分解计算出来的近似解已经没有任何意义了.



## 2.4 解的改进 \*

当矩阵  $A$  是病态时, 即使残量  $r = b - A\tilde{x}$  很小, 所求得的数值解  $\tilde{x}$  仍可能带有较大的误差. 此时需要通过一些方法来提高解的精度.

### 2.4.1 高精度运算

在计算中, 尽可能采用高精度的运算. 比如, 原始数据是单精度的, 但在计算时都采用双精度运算, 或者更高精度的运算. 但更高精度的运算会带来更大的开销.

### 2.4.2 矩阵元素缩放 (Scaling)

如果  $A$  的元素在数量级上相差很大, 则在计算过程中很可能会出现大数与小数的加减运算, 这样就可能会引入更多的舍入误差. 为了避免由于这种情况而导致的舍入误差, 我们可以在求解之前先对矩阵元素进行缩放 (Scaling), 即在矩阵两边同时乘以两个适当的对角矩阵.

**例 2.9** 考虑线性方程组

$$\begin{bmatrix} -4000 & 2000 & 2000 \\ 2000 & 0.78125 & 0 \\ 2000 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 400 \\ 1.3816 \\ 1.9273 \end{bmatrix}.$$

用部分选主元 Gauss 消去法求解, 计算过程中保留 8 位有效数字, 最后求得的数值解为

$$\tilde{x} = [0.00096365, -0.698496, 0.90042329]^T.$$

而精确解为  $x = [1.9273..., -0.698496..., 0.9004233...]^T$ . 数值解的第一个分量存在很大的误差.

我们考虑对矩阵元素进行缩放, 即在方程两边同时乘以一个对角矩阵  $D = \text{diag}(0.00005, 1, 1)$ , 然后求解一个新的方程组

$$DADy = Db.$$

最后令  $\tilde{x} = Dy$ , 即可求得比较精确的数值解.

为了平衡矩阵元素的大小, 一种好的方案是左乘一个对角矩阵  $D^{-1}$  (即对  $A$  的行进行缩放), 其中  $D_{ii} = \sum_{j=1}^n |a_{ij}|$ . 然后再执行部分选主元 LU 分解.

### 2.4.3 迭代改进法

设近似解  $\tilde{x}$ , 残量  $r = b - A\tilde{x}$ . 当  $\tilde{x}$  没达到精度要求时, 可以考虑对其进行改进或修正, 即加上一个修正向量  $z$ , 从而得到新的更好的近似解  $\tilde{x} + z$ . 但问题是: 怎样才能保证  $\tilde{x} + z$  是一个更好的近似解? 最理想的情况是: 构造修正向量  $z$  使得  $\tilde{x} + z$  是原问题的精确解, 即

$$A(\tilde{x} + z) = b,$$

也就是说, 向量  $z$  满足方程组

$$Az = b - A\tilde{x} = r. \quad (2.17)$$

这就是残量方程 (右端项是残量). 但在实际计算中, 我们无法计算出残量方程 (2.17) 的精确解, 所能得到的只能是近似解  $\tilde{z}$ . 但通常  $\|b - A(\tilde{x} + \tilde{z})\| = \|r - A\tilde{z}\|$  应该比较小, 特别地, 比  $\|r\|$  更小. 因此  $\tilde{x} + \tilde{z}$  应该比  $\tilde{x}$  更接近精确解. 如果新的近似解  $\tilde{x} + \tilde{z}$  还不满足精度要求, 则可重复以上过程. 这就是提高解的精度迭代改进法.

#### 算法 2.13. 迭代改进法

- 1: 设  $PA = LU$ ,  $\tilde{x}$  是  $Ax = b$  的近似解
- 2: **while** 近似解  $\tilde{x}$  不满足精度要求, **do**
- 3:     计算  $r = b - A\tilde{x}$
- 4:     求解  $Ly = Pr$ , 即  $y = L^{-1}Pr$
- 5:     求解  $Uz = y$ , 即  $z = U^{-1}y$
- 6:     令  $\tilde{x} = \tilde{x} + z$
- 7: **end while**

由于每次迭代只需计算一次残量和求解两个三角线性方程组, 因此运算量为  $O(n^2)$ . 所以相对来讲还是比较经济的.

为了提高计算精度, 在计算残量  $r$  时最好使用原始数据  $A$ , 而不是  $P^T LU$ , 因此对  $A$  做 PLU 分解时需要保留原矩阵  $A$ , 不能被  $L$  和  $U$  覆盖.

实际计算经验表明, 当  $A$  病态不是很严重时, 即  $\varepsilon_u \kappa_\infty(A) < 1$ , 迭代法可以有效改进解的精度, 最后达到机器精度. 但  $\varepsilon_u \kappa_\infty(A) \geq 1$  时, 一般没什么效果. 这里  $\varepsilon_u$  表示机器精度.



## 2.5 课后练习

练习 2.1 下列矩阵是否存在 LU 分解 ( $L$  单位下三角,  $U$  非奇异上三角)

$$A_1 = \begin{bmatrix} 2 & 1 & 4 \\ 3 & 3 & 1 \\ 3 & 2 & 6 \end{bmatrix}, \quad A_2 = \begin{bmatrix} 1 & 2 & 2 \\ 2 & 3 & 0 \\ 0 & 1 & 4 \end{bmatrix}.$$

练习 2.2 设  $A = [a_{ij}] \in \mathbb{R}^{n \times n}$ , 且  $a_{11} \neq 0$ , 经过一次 Gauss 消去法后得到  $A^{(2)} = \begin{bmatrix} a_{11} & * \\ 0 & A_2 \end{bmatrix}$ .

证明: (1) 若  $A$  对称, 则  $A_2$  也对称;

(2) 若  $A$  对称正定, 则  $A_2$  也对称正定;

(3) 若  $A$  (行) 严格对角占优, 则  $A_2$  也 (行) 严格对角占优.

练习 2.3 设矩阵  $A = \begin{bmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 12 \end{bmatrix}$ , 计算  $A$  的 LU 分解和 PLU 分解.

练习 2.4 计算  $A = \begin{bmatrix} 4 & 2 & 4 \\ 2 & 37 & 8 \\ 4 & 8 & 14 \end{bmatrix}$  的 Cholesky 分解, 并求解  $Ax = b$ , 其中  $b = \begin{bmatrix} 6 \\ -9 \\ 7 \end{bmatrix}$ .

练习 2.5 设矩阵  $A = \begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & a \\ 0 & a & 2 \end{bmatrix}$ , 问: 当  $a$  取何值时,  $A$  存在 Cholesky 分解?

练习 2.6 用追赶法解三对角方程组  $Ax = b$ , 其中

$$A = \begin{bmatrix} 3 & -1 & & \\ -1 & 3 & -1 & \\ & -1 & 3 & -1 \\ & & -1 & 3 \end{bmatrix}, \quad b = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}.$$

练习 2.7 设矩阵  $A = \begin{bmatrix} 1 & 2 \\ 2 & 3 \end{bmatrix}$ , 计算  $\kappa_2(A)$  和  $\kappa_\infty(A)$ .

练习 2.8 设矩阵  $A = \begin{bmatrix} \lambda & 2\lambda \\ 1 & 1 \end{bmatrix}$  ( $\lambda \neq 0$ ), 当  $\lambda$  取何值时,  $\kappa_\infty(A)$  达到最小.

练习 2.9 证明引理 2.12 中的结论:

- $\kappa(A) \geq 1, \quad \forall A \in \mathbb{R}^{n \times n}$ .
- 对任意非零常数  $\alpha \in \mathbb{R}$ , 都有  $\kappa(\alpha A) = \kappa(A)$ .
- 对任意正交矩阵  $Q \in \mathbb{R}^{n \times n}$ , 有  $\kappa_2(Q) = 1$ .
- 设  $Q$  是正交矩阵, 则对任意矩阵  $A$  有  $\kappa_2(QA) = \kappa_2(A) = \kappa_2(AQ)$ .

练习 2.10 设  $A \in \mathbb{R}^{n \times n}$  非奇异, 证明:  $\kappa_2(A^T A) = \left( \kappa_2(A) \right)^2$ .



练习 2.11 验证引理 2.3 中的等式

$$L = L_1 L_2 \cdots L_{n-1} = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ l_{21} & 1 & 0 & \cdots & 0 \\ l_{31} & l_{32} & 1 & \cdots & 0 \\ \vdots & \vdots & & \ddots & \\ l_{n1} & l_{n2} & l_{n3} & \cdots & 1 \end{bmatrix}.$$

练习 2.12 证明 Cholesky 分解的唯一性 (定理 2.7).

练习 2.13 设  $A \in \mathbb{R}^{n \times n}$  对称非奇异, 且存在分解  $A = LDM^T$ , 其中  $L, M \in \mathbb{R}^{n \times n}$  是单位下三角矩阵,  $D \in \mathbb{R}^{n \times n}$  是对角矩阵. 证明:  $L = M$ .

练习 2.14 将  $A \in \mathbb{R}^{n \times n}$  写成分块形式

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix},$$

其中  $A_{11} \in \mathbb{R}^{k \times k}$  ( $1 \leq k \leq n$ ) 非奇异. 我们称矩阵  $S = A_{22} - A_{21}A_{11}^{-1}A_{12}$  为  $A$  中  $A_{11}$  的 Schur 补 (通常简称 Schur 补).

(1) 假设  $A$  存在 LU 分解, 证明: 对于不选主元的 Gauss 消去法, 第  $k$  步后,  $A_{22}$  已被  $S$  覆盖.

(2) 假设  $A_{21} = A_{12}^T$ , 且  $A_{11}$  和  $-A_{22}$  都正定, 证明  $A$  非奇异.

练习 2.15\* 给定一个非奇异矩阵  $A$  和一个向量  $b$ .

试证明: 对充分小的  $\|\delta A\|_2$ , 存在非零的  $\delta A$  和  $\delta b$ , 使得不等式 (2.16) 中的等号成立, 即

$$\|\delta x\|_2 = \|A^{-1}\|_2 (\|\delta A\|_2 \cdot \|\tilde{x}\|_2 + \|\delta b\|_2).$$

练习 2.16\* 设  $A$  是对称对称且非零, 试证明: 存在置换矩阵  $P$  使得

$$PAP^T = \begin{bmatrix} B & E^T \\ E & C \end{bmatrix},$$

其中  $B \in \mathbb{R}$  或  $B \in \mathbb{R}^{2 \times 2}$ , 且非奇异.

练习 2.17 设  $A \in \mathbb{R}^{n \times n}$  是带状矩阵, 其下带宽为  $b_l$ , 上带宽为  $b_u$ . 若  $A$  存在不选主元的 LU 分解, 试统计求解  $Ax = b$  的运算量.

练习 2.18\* 假定已知  $A$  的 LU 分解为  $A = LU$ , 试设计算法计算  $A^{-1}$  的第  $(i, j)$  个元素, 使得运算量尽可能低.

练习 2.19\* 设  $A, B \in \mathbb{R}^{n \times n}$  是两个上三角矩阵,  $\alpha \in \mathbb{R}$  是给定常数, 且  $AB - \alpha I$  非奇异. 试设计求解  $(AB - \alpha I)x = b$  的算法, 使得运算量为  $O(n^2)$ .

## 实践题

练习 2.20 编写函数, 实现 列主元 LU 分解, 并生成一个  $n = 100$  阶的随机矩阵  $A$  进行测试.

要求与提示:

(1) M 文件取名为 `hw21.m` (模板见 `hw21t.m`);



(2) 列主元 LU 分解作为子函数实现, 函数原型为

`function [L,U,p]=PLU(A)`

其中  $L, U$  分别是单位下三角和非奇异上三角矩阵,  $p$  为向量, 代表置换矩阵。

**练习 2.21** 用改进的平方根法求解线性方程组  $H_n x = b$ , 其中  $H_n$  为  $n$  阶 Hilbert 矩阵。

**要求与提示:**

- (1) M 文件取名为 `hw22.m` (模板见课程主页 `hw22t.m`);
- (2) 取  $n = 32$ , MATLAB 中生成  $n$  阶 Hilbert 矩阵的命令是 `H=hilb(n)`;
- (3) 设精确解为  $x_* = [1, 1, \dots, 1]^T$ , 由此计算出右端项  $b$ ;
- (4) 输出相对残量的 2-范数和相对误差的 2-范数, 即

$$\text{relres} \triangleq \frac{\|b - H\tilde{x}\|_2}{\|b\|_2}, \quad \text{relerr} \triangleq \frac{\|\tilde{x} - x_*\|_2}{\|x_*\|_2},$$

其中  $\tilde{x}$  为计算得到的近似解。

**练习 2.22** 带状矩阵的 LU 分解. 设  $A$  是  $n$  阶带状矩阵, 上带宽为  $L < n$ , 下带宽为  $M < n$ , 编写一个函数, 计算  $A$  的 LU 分解 (不带选主元)。

函数原型: `[L,U]=LU_banded(A,L,M)`

# 3

## 线性最小二乘问题

**最小二乘问题** (Least Squares Problem) 包括线性最小二乘问题, 等式约束最小二乘问题, 加权最小二乘问题, 非线性最小二乘问题, 等等. 它在统计学, 最优化问题, 材料与结构力学, 信号与图像处理等方面都有着广泛的应用, 是计算数学的一个重要研究分支, 也是一个活跃的研究领域.

本讲主要介绍求解**线性最小二乘问题**的三种常用方法: 正规方程法 (也称法方程法), QR 分解法和 SVD 分解法. 一般来说, 正规方程法是最快的, 特别是当  $A$  的条件数较小时, 正规方程法几乎与其他方法一样精确. 而 SVD 分解法是最慢的, 但结果最可靠. 综合计算效率和计算精度, 当前的首选方法是 QR 分解法.

为了方便起见, 我们有时将线性最小二乘问题 (3.1) 简称为**最小二乘问题**.

### 最小二乘问题相关参考文献

- [1] A. Björck, [Numerical Methods for Least Squares Problems](#), SIAM, 1996 [5]
- [2] 魏木生等, [广义最小二乘问题的理论和计算](#) (第二版), 2020 [67]


### 3.1 问题介绍

**线性最小二乘问题**就是求解下面问题的最小值点:

$$\min_{x \in \mathbb{R}^n} \|Ax - b\|_2^2 \quad (3.1)$$

其中  $A \in \mathbb{R}^{m \times n}$ ,  $b \in \mathbb{R}^m$ . 问题 (3.1) 的解称为**最小二乘解**.

- 当  $m = n$  且  $A$  非奇异时, 这就是一个线性方程组, 解为  $x_* = A^{-1}b$ ;
- 当  $m > n$  时, 约束个数大于未知量个数, 此时我们称问题 (3.1) 为**超定**的 (overdetermined);
- 当  $m < n$  时, 未知量个数大于约束个数, 此时我们称问题 (3.1) 为**欠定** (或亚定) 的 (underdetermined).

 为了讨论方便, 除非特别说明, 否则本讲总是假定  $A$  是满秩的.



### 3.1.1 超定方程组

当  $m > n$  时, 线性方程组  $Ax = b$  的解可能不存在. 此时一般考虑求解最小二乘问题 (3.1). 记

$$J(x) \triangleq \|Ax - b\|_2^2.$$

易知  $J(x)$  是关于  $x$  的二次函数, 而且是凸函数 (当  $A$  满秩时,  $J(x)$  的 Hesse 阵是正定的). 因此, 由凸函数的性质可知,  $x_*$  是问题 (3.1) 的解当且仅当  $x_*$  是  $J(x)$  的稳定点. 令其一阶导数为零, 可得

$$A^T Ax - A^T b = 0.$$

于是将最小二乘问题转化为一个线性方程组, 这就是后面的正规方程.

如果  $A$  不是满秩, 则  $A^T A$  半正定, 此时解不唯一.

### 3.1.2 欠定方程组

若  $m < n$ , 则线性方程组  $Ax = b$  存在无穷多个解 (假定  $A$  满秩). 这时我们通常寻求最小范数解, 即所有解中范数最小的解. 于是原问题就转化为下面的约束优化问题

$$\min_{Ax=b} \frac{1}{2} \|x\|_2^2 \quad (3.2)$$

对应的 Lagrange 函数为

$$\mathcal{L}(x, \lambda) = \frac{1}{2} \|x\|_2^2 + \lambda^T (Ax - b),$$

其中  $\lambda = [\lambda_1, \lambda_2, \dots, \lambda_m]^T$  是 Lagrange 乘子. 此时优化问题 (3.2) 的解就是  $\mathcal{L}(x, \lambda)$  的鞍点, 即下面方程组的解:

$$\frac{\partial \mathcal{L}}{\partial x} = x + A^T \lambda = 0, \quad \frac{\partial \mathcal{L}}{\partial \lambda} = Ax - b = 0.$$

写成矩阵形式为

$$\begin{bmatrix} I & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} x \\ \lambda \end{bmatrix} = \begin{bmatrix} 0 \\ b \end{bmatrix}.$$

如果  $A$  满秩, 即  $\text{rank}(A) = m$ , 则系数矩阵非奇异 (见练习 ??), 上述方程组存在唯一解.

本讲主要讨论超定线性最小二乘问题的求解.

## 3.2 基本矩阵变换

矩阵计算的一个基本思想就是把复杂的问题转化为等价的且易于求解的问题. 完成这个转化的基本工具就是 **矩阵变换**, 比如高等代数中的三类初等变换. 除此之外, 在矩阵计算中常用的矩阵变换有: Householder 变换和 Givens 变换. 这两类矩阵变换都是正交变换, 可用于求解最小二乘问题、特征值问题、奇异值问题等.

### 3.2.1 Householder 变换

**定义 3.1** 我们称矩阵

$$H = I - \frac{2}{v^*v}vv^* = I - \frac{2}{\|v\|_2^2}vv^*, \quad 0 \neq v \in \mathbb{C}^n, \quad (3.3)$$

为 **Householder 矩阵** (或 **Householder 变换**, 或 **Householder 反射**), 向量  $v$  称为 **Householder 向量**. 我们通常将矩阵 (3.3) 记为  $H(v)$ .

有时也将 Householder 变换定义为

$$H = I - 2vv^*, \quad v \in \mathbb{C}^n \text{ 且 } \|v\|_2 = 1.$$

Householder 变换是 **基本变换矩阵** 的特殊情形, 这里的基本变换矩阵定义为:

$$E(u, v, \tau) = I - \tau uv^*, \quad (3.4)$$

其中  $u, v \in \mathbb{C}^n$  是非零向量,  $\tau$  是一个非零复数. 因此, 基本变换矩阵是单位矩阵的一个秩 1 修正.

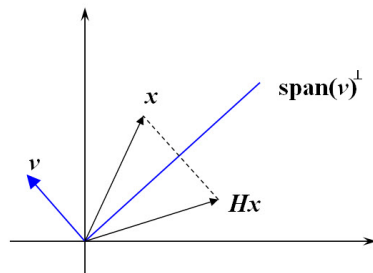
从几何上看, 一个 Householder 变换是一个关于超平面  $\text{span}\{v\}^\perp$  的反射. 由于  $\mathbb{C}^2 = \text{span}\{v\} \oplus \text{span}\{v\}^\perp$ , 因此对任意一个向量  $x \in \mathbb{C}^n$ , 都可写为

$$x = \frac{v^*x}{v^*v}v + y \triangleq \alpha v + y,$$

其中  $\alpha v \in \text{span}\{v\}$ ,  $y \in \text{span}\{v\}^\perp$ . 于是

$$Hx = x - \frac{2}{v^*v}vv^*x = x - 2\alpha v = -\alpha v + y,$$

即  $Hx$  与  $x$  在  $\text{span}\{v\}^\perp$  方向有着相同的分量, 而在  $v$  方向的分量正好相差一个符号. 也就是说,  $Hx$  是  $x$  关于超平面  $\text{span}\{v\}^\perp$  的镜面反射. 因此, Householder 变换也称为 Householder 反射.



下面是关于 Householder 矩阵的几个基本性质.

**定理 3.1** 设  $H \in \mathbb{C}^{n \times n}$  是一个 Householder 矩阵, 则

- (1)  $H^* = H$ , 即  $H$  Hermite 的;
- (2)  $H^*H = I$ , 即  $H$  是酉矩阵;
- (3)  $H^2 = I$ , 所以  $H^{-1} = H$ ;
- (4)  $\det(H) = -1$ ;



(5)  $H$  有两个互异的特征值:  $\lambda = 1$  和  $\lambda = -1$ , 其中  $\lambda = 1$  的代数重数为  $n - 1$ .

Householder 矩阵的一个非常重要的应用就是可以将一个向量除第一个元素以外的所有元素都化为零. 我们首先给出一个引理.

**引理 3.2** 设  $x, y \in \mathbb{C}^n$  为任意两个互异的向量, 则存在一个 Householder 矩阵  $H(v)$  使得  $y = H(v)x$  的充要条件是  $\|x\|_2 = \|y\|_2$  且  $x^*y \in \mathbb{R}$ .

**证明.** 若  $\|x\|_2 = \|y\|_2$  且  $x^*y \in \mathbb{R}$ , 则  $y^*y = x^*x$  且  $x^*y = y^*x$ . 于是

$$\|x - y\|_2^2 = (x - y)^*(x - y) = x^*x - y^*x - x^*y + y^*y = 2(x^*x - y^*x).$$

令  $v = x - y$ , 则有

$$H(v)x = x - \frac{2(x - y)(x - y)^*x}{\|x - y\|_2^2} = x - \frac{2(x - y)(x^*x - y^*x)}{2(x^*x - y^*x)} = y,$$


即存在 Householder 矩阵  $H(v)$  使得  $y = H(v)x$ .

反之, 如果存在 Householder 矩阵  $H$  使得  $y = Hx$ , 由于  $H$  是 Hermite 的, 所以  $x^*y = x^*Hx \in \mathbb{R}$ . 又因为  $H$  是酉矩阵, 所以  $\|y\|_2 = \|Hx\|_2 = \|x\|_2$ .  $\square$

 如果  $x, y$  都是实向量, 则条件  $x^*y \in \mathbb{R}$  自然成立, 此时充要条件就是  $\|x\|_2 = \|y\|_2$ .

由引理 3.2, 我们可以立即得到下面的结论.

**定理 3.3** 设  $x = [x_1, x_2, \dots, x_n]^T \in \mathbb{R}^n$  是一个非零向量, 则存在 Householder 矩阵  $H(v)$  使得  $H(v)x = \alpha e_1$ , 其中  $\alpha = \|x\|_2$  (或  $\alpha = -\|x\|_2$ ),  $e_1 = [1, 0, \dots, 0]^T \in \mathbb{R}^n$ .

 在后面的讨论中, 我们将定理的向量  $v$  称为  $x$  对应的 Householder 向量.

设  $x = [x_1, x_2, \dots, x_n]^T \in \mathbb{R}^n$  是一个实的非零向量, 下面讨论如何计算定理 3.3 中 Householder 矩阵  $H(v)$  所对应的 Householder 向量  $v$ . 由引理 3.2 的证明过程可知

$$v = x - \alpha e_1 = [x_1 - \alpha, x_2, \dots, x_n]^T.$$

在实际计算中, 为了尽可能地减少舍入误差, 我们通常避免两个相近的数做减运算, 否则就会损失有效数字. 因此, 我通常取

$$\alpha = -\text{sign}(x_1) \cdot \|x\|_2. \quad (3.5)$$

事实上, 我们也可以取  $\alpha = \text{sign}(x_1)\|x\|_2$ , 但此时为了减少舍入误差, 我们需要通过下面的公式来计算  $v$  的第一个分量  $v_1$

$$\alpha = \text{sign}(x_1)\|x\|_2, \quad v_1 = x_1 - \alpha = \frac{x_1^2 - \|x\|_2^2}{x_1 + \alpha} = \frac{-(x_2^2 + x_3^2 + \dots + x_n^2)}{x_1 + \alpha}. \quad (3.6)$$

在  $v_1$  的两种计算方法 (3.5) 和 (3.6) 中,  $\alpha$  的取值都与  $x_1$  的符号有关. 但在某些应用中, 我们

需要确保  $\alpha$  非负, 此时我们可以将这两种方法结合起来使用, 即:

$$v_1 = \begin{cases} x_1 - \alpha, & \text{if } \text{sign}(x_1) < 0 \\ \frac{-(x_2^2 + x_3^2 + \cdots + x_n^2)}{x_1 + \alpha}, & \text{otherwise} \end{cases}$$

无论怎样选取  $\alpha$ , 我们都有  $H = I - \beta vv^*$ , 其中

$$\beta = \frac{2}{v^*v} = \frac{2}{(x_1 - \alpha)^2 + x_2^2 + \cdots + x_n^2} = \frac{2}{2\alpha^2 - 2\alpha x_1} = -\frac{1}{\alpha v_1}.$$

**思考** 思考: 如果  $x$  是复向量, 有没有相应的结论?

在实数域中计算 Householder 向量  $v$  的算法如下, 总运算量大约为  $2n$ , 如果加上赋值运算的话则为  $3n$ .

### 算法 3.1. 计算 Householder 向量

```
% Given  $x \in \mathbb{R}^n$ , compute  $v \in \mathbb{R}^n$  such that  $Hx = \|x\|_2 e_1$ , where  $H = I - \beta vv^*$ 
1: function [ $\beta, v$ ] = House( $x$ )
2:  $n = \text{length}(x)$  (here  $\text{length}(x)$  denotes the dimension of  $x$ )
3:  $\sigma = x_2^2 + x_3^2 + \cdots + x_n^2$ 
4:  $v = x$ 
5: if  $\sigma = 0$  then
6:   if  $x_1 < 0$  then
7:      $v_1 = 2x_1, \beta = 2/v_1^2$ 
8:   else
9:      $v_1 = 0, \beta = 0$ 
10:  end if
11: else
12:    $\alpha = \sqrt{x_1^2 + \sigma}$  %  $\alpha = \|x\|_2$ 
13:   if  $x_1 < 0$  then
14:      $v_1 = x_1 - \alpha$ 
15:   else
16:      $v_1 = -\sigma/(x_1 + \alpha)$ 
17:   end if
18:    $\beta = 2/(v_1^2 + \sigma)$ 
19: end if
```

可以证明, 上述算法具有很好的数值稳定性 [55], 即

$$\|\tilde{H} - H\|_2 = \mathcal{O}(\varepsilon_u),$$

其中  $\tilde{H}$  是由上述算法计算得到的近似矩阵,  $\varepsilon_u$  是机器精度.



在实际计算时, 我们可以将向量  $v$  单位化, 使得  $v_1 = 1$ . 这样, 我们就无需为  $v$  另外分配空间, 而是将  $v(2:n)$  存放在  $x(2:n)$  中.

为了避免可能产生的溢出, 我们也可以事先将  $x$  单位化, 即令  $x = x/\|x\|_2$

### Householder 变换与向量和矩阵的乘积

对向量和矩阵做 Householder 变换时, 无需显式写出 Householder 矩阵. 设  $x \in \mathbb{R}^m$ ,  $A \in \mathbb{R}^{m \times n}$ ,  $H = I - \beta vv^* \in \mathbb{R}^m$ , 则

$$Hx = (I - \beta vv^*)x = x - \beta(v^*x)v,$$

$$HA = (I - \beta vv^*)A = A - \beta v(v^*A).$$

运算量大约分别为  $4m$  和  $4mn$ , 远小于一般的矩阵-向量乘积和矩阵-矩阵乘积的运算量.

### 3.2.2 Givens 变换

为简单起见, 我们这里讨论实数域中的 Givens 变换. 设  $\theta \in [0, 2\pi]$ , 我们称矩阵

$$G(i, j, \theta) = \begin{bmatrix} 1 & & & & \\ & \ddots & & & \\ & & c & s & \\ & & -s & c & \\ & & & \ddots & \\ & & & & 1 \end{bmatrix} \in \mathbb{R}^{n \times n}, \quad (i \leq j)$$

为 **Givens 变换** (或 **Givens 旋转**, 或 **Givens 矩阵**), 其中  $c = \cos(\theta)$ ,  $s = \sin(\theta)$ . 即将单位矩阵的  $(i, i)$  和  $(j, j)$  位置上的元素用  $c$  代替, 而  $(i, j)$  和  $(j, i)$  位置上的元素分别用  $s$  和  $-s$  代替, 所得到的矩阵就是  $G(i, j, \theta)$ .

**定理 3.4**  $G(i, j, \theta)$  是正交矩阵, 且  $\det(G(i, j, \theta)) = 1$ .

Givens 变换不是一个基本变换矩阵, 事实上, 它是单位矩阵的一个秩 2 修正, 即

$$G(i, j, \theta) = I + [e_i, e_j] \begin{bmatrix} \cos(\theta) - 1 & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) - 1 \end{bmatrix} [e_i, e_j]^T.$$

我们需要注意的是, 当一个矩阵左乘一个 Givens 矩阵时, 只会影响其第  $i$  行和第  $j$  行的元素. 而当一个矩阵右乘一个 Givens 矩阵时, 只会影响其第  $i$  和第  $j$  列的元素.

**例 3.1** 设  $x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \in \mathbb{R}^2$ , 则存在一个 Givens 变换  $G = \begin{bmatrix} c & s \\ -s & c \end{bmatrix} \in \mathbb{R}^{2 \times 2}$  使得  $Gx = \begin{bmatrix} r \\ 0 \end{bmatrix}$ , 其中  $c, s$  和  $r$  的值如下:

- 若  $x_1 = x_2 = 0$ , 则  $c = 1, s = 0, r = 0$ ;

- 若  $x_1 = 0$  但  $x_2 \neq 0$ , 则  $c = 0, s = x_2/|x_2|, r = |x_2|$ ;
- 若  $x_1 \neq 0$  但  $x_2 = 0$ , 则  $c = \text{sign}(x_1), s = 0, r = |x_1|$ ;
- 若  $x_1 \neq 0$  且  $x_2 \neq 0$ , 则  $c = x_1/r, s = x_2/r, r = \sqrt{x_1^2 + x_2^2}$ .

也就是说, 通过 Givens 变换, 我们可以将向量  $x \in \mathbb{R}^2$  的第二个分量化为 0.

事实上, 对于任意一个向量  $x \in \mathbb{R}^n$ , 我们都可以通过 Givens 变换将其任意一个位置上的分量化为 0. 更进一步, 我们也可以通过若干个 Givens 变换, 将  $x$  中除第一个分量外的所有元素都化为 0.

### 算法 3.2. Givens 变换

```
% Given  $x = [a, b]^T \in \mathbb{R}^2$ , compute  $c$  and  $s$  such that  $Gx = [r, 0]^T$  where  $r = \|x\|_2$ 
1: function  $[c, s] = \text{givens}(a, b)$ 
2: if  $b = 0$  then
3:   if  $a \geq 0$  then
4:      $c = 1, \quad s = 0$ 
5:   else
6:      $c = -1, \quad s = 0$ 
7:   end if
8: else
9:   if  $|b| > |a|$  then
10:     $\tau = \frac{a}{b}, \quad s = \frac{\text{sign}(b)}{\sqrt{1 + \tau^2}}, \quad c = s\tau$ 
11:   else
12:     $\tau = \frac{b}{a}, \quad c = \frac{\text{sign}(a)}{\sqrt{1 + \tau^2}}, \quad s = c\tau$ 
13:   end if
14: end if
```

任何一个正交矩阵都可以写成若干个 Householder 矩阵或 Givens 矩阵的乘积 (见习题 3.4 和 3.5), 所以正交矩阵所对应的线性变换可以看作是反射变换和旋转变换的组合, 因此它不会改变向量的长度与 (不同向量之间的) 角度.

### 3.2.3 正交变换的舍入误差分析

**引理 3.5** 设  $P \in \mathbb{R}^{n \times n}$  是一个精确的 Householder 或 Givens 变换,  $\tilde{P}$  是其浮点运算近似, 则

$$\text{fl}(\tilde{P}A) = P(A + E), \quad \text{fl}(A\tilde{P}) = (A + F)P,$$

其中  $\|E\|_2 = \mathcal{O}(\varepsilon_u) \cdot \|A\|_2, \|F\|_2 = \mathcal{O}(\varepsilon_u) \cdot \|A\|_2$ . (这里  $\varepsilon_u$  表示机器精度)

这说明对一个矩阵做 Householder 变换或 Givens 变换是向后稳定的.



**定理 3.6** 考虑对矩阵  $A$  做一系列的正交变换, 则有

$$\mathfrak{fl}(\tilde{P}_k \cdots \tilde{P}_1 A \tilde{Q}_1 \cdots \tilde{Q}_k) = P_k \cdots P_1 (A + E) Q_1 \cdots Q_k,$$

其中  $\|E\|_2 = \mathcal{O}(\varepsilon_u) \cdot (k\|A\|_2)$ . 这说明整个计算过程是向后稳定的.

一般地, 假设  $X$  是一个非奇异的线性变换,  $\tilde{X}$  是其浮点运算近似. 当  $X$  作用到  $A$  上时, 我们有

$$\mathfrak{fl}(\tilde{X}A) = XA + E = X(A + X^{-1}E) \triangleq X(A + F),$$

其中  $\|E\|_2 = \mathcal{O}(\varepsilon_u) \cdot \|XA\|_2 \leq \mathcal{O}(\varepsilon_u) \cdot \|X\|_2 \cdot \|A\|_2$ , 故

$$\|F\|_2 = \|X^{-1}E\|_2 \leq \mathcal{O}(\varepsilon_u) \cdot \|X^{-1}\|_2 \cdot \|X\|_2 \cdot \|A\|_2 = \mathcal{O}(\varepsilon_u) \cdot \kappa_2(X) \cdot \|A\|_2,$$

因此, 舍入误差将被放大  $\kappa_2(X)$  倍. 当  $X$  是正交变换时,  $\kappa_2(X)$  达到最小值 1, 这就是为什么在浮点运算中尽量使用正交变换的原因.

### 3.3 QR 分解

QR 分解是将一个矩阵分解一个正交矩阵 (酉矩阵) 和一个三角矩阵的乘积. QR 分解被广泛应用于线性最小二乘问题的求解和矩阵特征值的计算.

#### 3.3.1 QR 分解的存在性与唯一性

**定理 3.7 (QR 分解)** [31] 设  $A \in \mathbb{C}^{m \times n}$  ( $m \geq n$ ). 则存在一个单位列正交矩阵  $Q \in \mathbb{C}^{m \times n}$  (即  $Q^*Q = I_{n \times n}$ ) 和一个上三角矩阵  $R \in \mathbb{C}^{n \times n}$ , 使得

$$A = QR. \quad (3.7)$$

若  $A$  列满秩, 则存在一个具有正对角线元素的上三角矩阵  $R$  使得 (3.7) 成立, 且此时 QR 分解唯一, 即  $Q$  和  $R$  都唯一.

**证明.** 设  $A = [a_1, a_2, \dots, a_n] \in \mathbb{C}^{m \times n}$ . 若  $A$  列满秩, 即  $\text{rank}(A) = n$ . 则 QR 分解 (3.7) 就是对  $A$  的列向量组进行 Gram-Schmidt (简称 GS) 正交化过程的矩阵描述 (见算法 3.3).

#### 算法 3.3. Gram-Schmidt 过程

```

1:  $r_{11} = \|a_1\|_2$ 
2:  $q_1 = a_1/r_{11}$ 
3: for  $j = 2$  to  $n$  do
4:    $q_j = a_j$ 
5:   for  $i = 1$  to  $j - 1$  do
6:      $r_{ij} = q_i^* a_j$    %  $q_i^*$  表示共轭转置
7:      $q_j = q_j - r_{ij} q_i$ 
8:   end for
9:    $r_{jj} = \|q_j\|_2$ 
10:   $q_j = q_j/r_{jj}$ 
11: end for

```

由算法 3.3 可知

$$a_1 = r_{11}q_1, \quad a_j = r_{1j}q_1 + r_{2j}q_2 + \cdots + r_{jj}q_j = [q_1, q_2, \dots, q_j] \begin{bmatrix} r_{1j} \\ r_{2j} \\ \vdots \\ r_{jj} \end{bmatrix}, \quad j = 2, 3, \dots, n.$$

记  $Q = [q_1, q_2, \dots, q_n]$ ,  $R = [r_{ij}]_{n \times n}$ , 其中

$$r_{ij} = \begin{cases} q_i^* a_j, & \text{for } i \leq j \\ 0, & \text{for } i > j \end{cases} \quad (3.8)$$





于是 Gram-Schmidt 过程可表示为

$$[a_1, a_2, \dots, a_n] = [q_1, q_2, \dots, q_n] \begin{bmatrix} r_{11} & r_{12} & \cdots & r_{1n} \\ & r_{22} & \cdots & r_{2n} \\ & & \ddots & r_{n-1,n} \\ & & & r_{nn} \end{bmatrix}, \quad \text{即 } A = QR.$$

如果  $A$  不是列满秩, 我们可以通过下面的方式做类似的正交化过程:

- 如果  $a_1 = 0$ , 则令  $q_1 = 0$ ; 否则令  $q_1 = a_1 / \|a_1\|_2$ ;
- 对于  $j = 2, 3, \dots, n$ , 计算  $\tilde{q}_j = a_j - \sum_{i=1}^{j-1} (q_i^* a_j) q_i$ . 如果  $\tilde{q}_j = 0$ , 则表明  $a_j$  可以由  $a_1, a_2, \dots, a_{j-1}$  线性表出, 令  $q_j = 0$ . 否则令  $q_j = \tilde{q}_j / \|\tilde{q}_j\|_2$ .

于是我们有

$$A = QR,$$

其中  $Q = [q_1, q_2, \dots, q_n]$  列正交 (但不是单位列正交), 其列向量要么是单位向量, 要么就是零向量.  $R = [r_{ij}]_{n \times n}$  的定义同 (3.8). 需要注意的是, 如果  $Q$  的某一列  $q_k = 0$ , 那么  $R$  中对应的第  $k$  行就全部为 0.

设  $\text{rank}(A) = l < n$ , 则  $Q$  有  $l$  个非零列, 设为  $q_{i_1}, q_{i_2}, \dots, q_{i_l}$ . 它们形成  $\mathbb{C}^m$  中的一个单位正交向量组, 所以我们可以将其扩展成  $\mathbb{C}^m$  中的一组标准正交基, 即

$$q_{i_1}, q_{i_2}, \dots, q_{i_l}, \tilde{q}_1, \dots, \tilde{q}_{m-l}.$$

然后用  $\tilde{q}_1$  替换  $Q$  中的第一个零列, 用  $\tilde{q}_2$  替换  $Q$  中的第二个零列, 依此类推, 将  $Q$  中的所有零列都替换掉. 将最后得到的矩阵记为  $\tilde{Q}$ , 则  $\tilde{Q} \in \mathbb{C}^{m \times n}$  单位列正交, 且

$$\tilde{Q}R = QR.$$

这是由于  $\tilde{Q}$  中的新添加的列向量正好与  $R$  中的零行相对应. 所以我们有 QR 分解

$$A = \tilde{Q}R.$$

下面证明 **满秩矩阵 QR 分解的存在唯一性**.

存在性: 由于  $A$  列满秩, 由 GS 正交化过程 (算法 3.3) 可知, 存在上三角矩阵  $R = [r_{ij}]_{n \times n}$  满足  $r_{jj} > 0$ , 使得  $A = QR$ , 其中  $Q$  单位列正交.

唯一性: 假设  $A$  存在 QR 分解

$$A = Q_1 R_1 = Q_2 R_2,$$

其中  $Q_1, Q_2 \in \mathbb{C}^{m \times n}$  单位列正交,  $R_1, R_2 \in \mathbb{C}^{n \times n}$  为具有正对角元素的上三角矩阵. 则有

$$Q_1 = Q_2 R_2 R_1^{-1}. \quad (3.9)$$

于是

$$1 = \|Q_1\|_2 = \|Q_2 R_2 R_1^{-1}\|_2 = \|R_2 R_1^{-1}\|_2.$$

又  $R_1, R_2$  均为上三角矩阵, 所以  $R_2 R_1^{-1}$  也是上三角矩阵, 且其对角线元素为  $R_2(i, i) / R_1(i, i)$ ,  $i = 1, 2, \dots, n$ . 因此

$$\frac{R_2(i, i)}{R_1(i, i)} \leq \rho(R_2 R_1^{-1}) \leq \|R_2 R_1^{-1}\|_2 \leq 1, \quad i = 1, 2, \dots, n.$$

同理可证  $R_1(i, i)/R_2(i, i) \leq 1$ . 所以

$$R_1(i, i) = R_2(i, i), \quad i = 1, 2, \dots, n.$$

又  $\|Q_1\|_F^2 = \text{tr}(Q_1^* Q_1) = n$ , 所以由 (3.9) 可知

$$\|R_2 R_1^{-1}\|_F^2 = \|Q_2 R_2 R_1^{-1}\|_F^2 = \|Q_1\|_F^2 = n.$$

由于  $R_2 R_1^{-1}$  的对角线元素都是 1, 所以  $R_2 R_1^{-1}$  只能是单位矩阵, 即  $R_2 = R_1$ . 因此  $Q_2 = A R_2^{-1} = A R_1^{-1} = Q_1$ , 即  $A$  的 QR 分解是唯一的.  $\square$

有时也将 QR 分解定义为: 存在酉矩阵  $Q \in \mathbb{C}^{m \times m}$  使得

$$A = QR,$$

其中  $R = \begin{bmatrix} R_{11} \\ 0 \end{bmatrix} \in \mathbb{C}^{m \times n}$  是上三角矩阵.

由 QR 分解的存在性证明过程可知, 当  $A$  不是满秩矩阵时, 存在一个置换矩阵  $P$ , 使得  $AP$  的前  $l$  列是线性无关的, 其中  $l = \text{rank}(A)$ . 因此我们可以对  $AP$  进行 QR 分解, 于是我们可以得到下面的结论.

**推论 3.8** 设  $A \in \mathbb{C}^{m \times n}$  ( $m \geq n$ ), 且秩为  $l$  ( $0 \leq l \leq n$ ). 则存在一个置换矩阵  $P$ , 使得

$$AP = Q \begin{bmatrix} R_{11} & R_{12} \\ 0 & 0 \end{bmatrix}_{n \times n},$$

其中  $Q \in \mathbb{C}^{m \times n}$  单位列正交,  $R_{11} \in \mathbb{C}^{l \times l}$  是非奇异上三角矩阵.

上述结论也可简化为

$$AP = Q \begin{bmatrix} R_{11} & R_{12} \end{bmatrix},$$

其中  $Q \in \mathbb{C}^{m \times l}$  单位列正交 (推论 3.8 中  $Q$  的前  $l$  列),  $R_{11} \in \mathbb{C}^{l \times l}$  是非奇异上三角矩阵.

通过 GS 正交化过程实现的 QR 分解中,  $Q$  和  $R$  都可能是复矩阵. 如果  $A$  是实矩阵, 则上面证明中的运算都可以在实数下进行, 因此  $Q$  和  $R$  都可以是实矩阵.

如果  $A$  是非奇异的方阵, 则 QR 分解也可以用来求解线性方程组  $Ax = b$ .

基于 GS 正交化的 QR 分解算法 3.3 的运算量大约为  $2mn^2$ . 在后面, 我们会介绍基于 Householder 变换的 QR 分解, 在不需要计算  $Q$  的情况下, 运算量大约为  $2mn^2 - 2n^3/3$ ; 如果需要计算  $Q$ , 则需另外大约  $4m^2n - 4mn^2 + 4n^3/3$  运算量.

**推论 3.9 (满秩分解)** 设  $A \in \mathbb{C}^{m \times n}$  且  $\text{rank}(A) = r \leq \min\{m, n\}$ , 则存在满秩矩阵  $F \in \mathbb{C}^{m \times r}$  和  $G \in \mathbb{C}^{r \times n}$ , 使得

$$A = FG.$$

下面给出 QR 分解的具体实现方法, 分别基于 MGS 过程, Householder 变换和 Givens 变换.



### 3.3.2 基于 MGS 的 QR 分解

在证明 QR 分解的存在性时, 我们利用了 Gram-Schmidt 正交化过程. 但由于数值稳定性方面的原因, 在实际计算中, 我们一般不直接采用 Gram-Schmidt 过程, 取而代之的是 **修正的 Gram-Schmidt 过程** (modified Gram-Schmidt process), 即 **MGS**.

#### 算法 3.4. 基于 MGS 的 QR 分解

```
% Given  $A \in \mathbb{R}^{m \times n}$ , compute  $Q = [q_1, \dots, q_n] \in \mathbb{R}^{m \times n}$  and  $R \in \mathbb{R}^{n \times n}$  such that  $A = QR$ 
1: Set  $R = [r_{ik}] = 0_{n \times n}$  (the  $n \times n$  zero matrix)
2: if  $a_1 = 0$  then
3:    $q_1 = 0$ 
4: else
5:    $r_{11} = \|a_1\|_2$ 
6:    $q_1 = a_1 / \|a_1\|_2$ 
7: end if
8: for  $k = 2$  to  $n$  do
9:    $q_k = a_k$ 
10:  for  $i = 1$  to  $k - 1$  do    % MGS 过程, 注意与 GS 的区别
11:     $r_{ik} = q_i^\top q_k$ 
12:     $q_k = q_k - r_{ik} q_i$ 
13:  end for
14:  if  $q_k \neq 0$  then
15:     $r_{kk} = \|q_k\|_2$ 
16:     $q_k = q_k / r_{kk}$ 
17:  end if
18: end for
```

由 MGS 得到的 QR 分解中,  $Q \in \mathbb{R}^{m \times n}$ ,  $R \in \mathbb{R}^{n \times n}$ . 运算量大约为  $2mn^2$ .

### 3.3.3 基于 Householder 变换的 QR 分解

由定理 3.3 可知, 通过 Householder 变换, 我们可以将任何一个非零变量  $x \in \mathbb{R}^n$  转化成  $\|x\|_2 e_1$ , 即除第一个元素外, 其它都为零. 下面我们就考虑通过 Householder 变换来实现矩阵的 QR 分解.

我们首先考虑  $m = n$  时的情形. 设矩阵  $A \in \mathbb{R}^{n \times n}$ , 令  $H_1 \in \mathbb{R}^{n \times n}$  为一个 Householder 变换, 满足

$$H_1 \begin{bmatrix} a_{11} \\ a_{21} \\ \vdots \\ a_{n1} \end{bmatrix} = \begin{bmatrix} r_1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}.$$

于是

$$H_1 A = \left[ \begin{array}{c|ccc} r_1 & \tilde{a}_{12} & \cdots & \tilde{a}_{1n} \\ \hline 0 & & & \\ \vdots & & \tilde{A}_2 & \\ 0 & & & \end{array} \right],$$

其中  $\tilde{A}_2 \in \mathbb{R}^{(n-1) \times (n-1)}$ . 同样地, 我们可以构造一个 Householder 变换  $\tilde{H}_2 \in \mathbb{R}^{(n-1) \times (n-1)}$ , 将  $\tilde{A}_2$  的第一列中除第一个元素外的所有元素都化为 0, 即

$$\tilde{H}_2 \tilde{A}_2 = \left[ \begin{array}{c|ccc} r_2 & \tilde{a}_{23} & \cdots & \tilde{a}_{2n} \\ \hline 0 & & & \\ \vdots & & \tilde{A}_3 & \\ 0 & & & \end{array} \right].$$

令

$$H_2 = \begin{bmatrix} 1 & 0 \\ 0 & \tilde{H}_2 \end{bmatrix}.$$

则  $H_2 \in \mathbb{R}^{n \times n}$ , 且

$$H_2 H_1 A = \left[ \begin{array}{cc|ccc} r_1 & \tilde{a}_{12} & \tilde{a}_{13} & \cdots & \tilde{a}_{1n} \\ 0 & r_2 & \tilde{a}_{23} & \cdots & \tilde{a}_{2n} \\ \hline 0 & 0 & & & \\ \vdots & \vdots & & \tilde{A}_3 & \\ 0 & 0 & & & \end{array} \right].$$

不断重复上述过程. 这样, 我们就得到一系列的矩阵

$$H_k = \begin{bmatrix} I_{k-1} & 0 \\ 0 & \tilde{H}_k \end{bmatrix}, \quad k = 1, 2, 3, 4, \dots, n-1$$

使得

$$H_{n-1} \cdots H_2 H_1 A = \begin{bmatrix} r_1 & \tilde{a}_{12} & \cdots & \tilde{a}_{1n} \\ 0 & r_2 & \cdots & \tilde{a}_{2n} \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & r_n \end{bmatrix} \triangleq R.$$

由于 Householder 变换都是正交矩阵, 因此  $H_1, H_2, \dots, H_{n-1}$  也都是正交矩阵. 令

$$Q = (H_{n-1} \cdots H_2 H_1)^{-1} = H_1^{-1} H_2^{-1} \cdots H_{n-1}^{-1} = H_1 H_2 \cdots H_{n-1},$$

则  $Q$  也是正交矩阵, 且

$$A = (H_{n-1} \cdots H_2 H_1)^{-1} R = QR.$$

以上就是基于 Householder 变换的 QR 分解的具体实现过程. 最后所得到的上三角矩阵  $R$  就存放在  $A$  的上三角部分. 矩阵  $Q$  可通过下面的算法实现

$$\begin{cases} Q = I_n, \\ Q = QH_k, \quad k = 1, 2, \dots, n-1. \end{cases}$$



如果  $m > n$ , 我们仍然可以通过上面的过程进行 QR 分解, 只是最后我们得到一个正交矩阵  $Q \in \mathbb{R}^{m \times m}$  和一个上三角矩阵  $R \in \mathbb{R}^{m \times n}$ , 使得  $A = QR$ .

如果不需要生成  $Q$ , 则基于 Householder 变换的 QR 分解的总运算量大约为  $2mn^2 - 2n^3/3$ .

如果保留了每一步的 Householder 向量, 则  $Q$  也可以通过下面的向后累积方法实现:

$$Q = I_n,$$

$$Q = H_k Q, \quad k = n-1, n-2, \dots, 1.$$

这样做的好处是一开始  $Q$  会比较稀疏, 随着迭代的进行,  $Q$  才会慢慢变满. 而前面的计算方法, 第一步就将  $Q$  变成了一个满矩阵. 采用这种方法计算  $Q$  的运算量大约为  $4m^2n - 4mn^2 + 4n^3/3$ .

如果将  $Q$  写成下面的形式

$$Q = I + WY^T,$$

则可以采用分块形式来计算  $W$  和  $Y$ , 虽然运算量会稍有增长, 但大多数运算是矩阵乘法, 因此可以尽可能多地采用 3 级 BLAS 运算, 效率可能会更高. 详情可参见 [21].

#### 算法 3.5. 基于 Householder 变换的 QR 分解 (MATLAB)

```
% Given  $A \in \mathbb{R}^{m \times n}$ , compute  $Q$  and  $R$  such that  $A = QR$  where  $Q \in \mathbb{R}^{m \times m}$  and  $R \in \mathbb{R}^{m \times n}$ 
% The upper triangular part of  $R$  is stored in the upper triangular part of  $A$ 
1: Set  $Q = I_{m \times m}$ 
2: for  $k = 1$  to  $n$  do
3:    $x = A(k:m, k)$ 
4:    $[\beta, v_k] = \text{House}(x)$ 
5:    $A(k:m, k:n) = (I_{m-k+1} - \beta v_k v_k^T) A(k:m, k:n)$ 
6:    $= A(k:m, k:n) - \beta v_k (v_k^T A(k:m, k:n))$ 
7:    $Q(:, k:m) = Q(:, k:m) (I_{m-k+1} - \beta v_k v_k^T)$ 
8:    $= Q(:, k:m) - \beta (Q(:, k:m) v_k) v_k^T$ 
9: end for
```

上面的算法只是关于利用 Householder 变换来实现 QR 分解的一个简单描述, 并没有考虑运算量问题. 在实际计算时, 我们通常会保留所有的 Householder 向量. 由于第  $k$  步中  $\tilde{H}_k$  所对应的 Householder 向量  $v_k$  的长度为  $m - k + 1$ , 因此我们先把  $v_k$  单位化, 使得  $v_k$  的第一元素为 1, 这样就只要存储  $v_k(2: \text{end})$ , 共  $m - k$  个元素. 这样, 我们就可以把所有的 Householder 向量存放在  $A$  的严格下三角部分. 而  $A$  的上三角部分仍然存放  $R$ . 在计算  $Q$  时采用向后累积的方法, 所以总的运算量大约为  $4m^2n - 2mn^2 + \frac{2}{3}n^3$ . 若  $m = n$ , 则运算量大约为  $\frac{8}{3}n^3$ .

我们也可以考虑分块 Householder QR 分解, 以便充分利用 3 级 BLAS 运算, 提高计算效率.

### 3.3.4 基于 Givens 变换的 QR 分解

我们同样可以利用 Givens 变换来做 QR 分解.

设  $A \in \mathbb{R}^{n \times n}$ , 首先构造一个 Givens 变换  $G_{21}$ , 作用在  $A$  的最前面的两行上, 使得

$$G_{21} \begin{bmatrix} a_{11} \\ a_{21} \\ a_{31} \\ \vdots \\ a_{n1} \end{bmatrix} = \begin{bmatrix} \tilde{a}_{11} \\ 0 \\ a_{31} \\ \vdots \\ a_{n1} \end{bmatrix}.$$

由于  $G_{21}$  只改变矩阵的第 1 行和第 2 行的值, 所以其它行保存不变. 然后再构造一个 Givens 变换  $G_{31}$ , 作用在  $G_{21}A$  的第 1 行和第 3 行, 将其第一列的第三个元素化为零. 由于  $G_{31}$  只改变矩阵的第 1 行和第 3 行的值, 所以第二行的零元素维持不变. 以此类推, 我们可以构造一系列的 Givens 变换  $G_{41}, G_{51}, \dots, G_{n1}$ , 使得  $G_{n1} \cdots G_{21}A$  的第一列中除第一个元素外, 其它元素都化为零, 即

$$G_{n1} \cdots G_{21}A = \begin{bmatrix} * & * & \cdots & * \\ 0 & * & \cdots & * \\ \vdots & \vdots & & \vdots \\ 0 & * & \cdots & * \end{bmatrix}.$$

下面我们可以对第二列进行类似的处理. 构造 Givens 变换  $G_{32}, G_{42}, \dots, G_{n2}$ , 将第二列的第 3 至第  $n$  个元素全化为零, 同时保持第一列不变.

以此类推, 我们对其他列也做类似的处理. 最后, 通过构造  $\frac{1}{2}n(n-1)$  个 Givens 变换, 将  $A$  化成一个上三角矩阵  $R$ , 即

$$R = G_{n,n-1} \cdots G_{21}A.$$

令  $Q = (G_{n,n-1} \cdots G_{21})^T$ . 由于 Givens 变换是正交矩阵, 所以  $Q$  也是正交矩阵. 于是, 我们就得到矩阵  $A$  的 QR 分解

$$A = QR.$$

- 🔴 与 Householder 变换一样, 在进行 Givens 变换时, 我们不需要显式地写出 Givens 矩阵.
- 🔴 对于稠密矩阵而言, 基于 Givens 变换的 QR 分解的运算量比 Householder 变换要多很多. 因此基于 Givens 变换的 QR 分解主要用于当矩阵的非零下三角元素相对较少时的情形, 比如对上 Hessenberg 矩阵进行 QR 分解.
- 🔴 如果  $A \in \mathbb{R}^{m \times n}$ , 其中  $m > n$ , 我们仍然可以通过 Givens 变换进行 QR 分解.

下面是基于 Givens 变换的 QR 分解的算法描述.

#### 算法 3.6. 基于 Givens 变换的 QR 分解

```
% Given  $A \in \mathbb{R}^{m \times n}$ , compute  $Q$  and  $R$  such that  $A = QR$  where  $Q \in \mathbb{R}^{m \times m}$  and  $R \in \mathbb{R}^{m \times n}$ 
% The upper triangular part of  $R$  is stored in the upper triangular part of  $A$ 
1: Set  $Q = I_{m \times m}$ 
```



```

2: for  $k = 1$  to  $n$  do
3:   for  $i = k + 1$  to  $m$  do
4:      $[c, s] = \text{givens}(a_{kk}, a_{ik})$ 
5:      $\begin{bmatrix} A(k, k:n) \\ A(i, k:n) \end{bmatrix} = G \begin{bmatrix} A(k, k:n) \\ A(i, k:n) \end{bmatrix}$  where  $G = \begin{bmatrix} c & s \\ -s & c \end{bmatrix}$ 
6:      $[Q(1:m, k), Q(1:m, i)] = [Q(1:m, k), Q(1:m, i)]G^T$ 
7:   end for
8: end for

```

### 3.3.5 QR 分解的稳定性

基于 Householder 变换和 Givens 变换的 QR 分解都具有很好的数值稳定性. 详细分析可以参考 [55] 和 [30]. 基于 MGS 的 QR 分解也是向后稳定的, 参见 [40].

当需要计算矩阵  $Q$  时, 基于 MGS 的 QR 分解的运算量相对较少. 因此当  $A$  的列向量具有很好的线性无关性时, 我们可以使用 MGS 来计算 QR 分解. 但需要注意的是, MGS 得到的  $Q$  不是方阵, 除非  $A$  是方阵.

由于舍入误差的原因, 最后得到的矩阵  $Q$  会带有一定的误差, 可能会导致  $Q$  失去正交性. Björck [4] 证明了, 通过 MGS 计算的矩阵  $Q$  满足

$$Q^T Q = I + E_{MGS} \quad \text{其中} \quad \|E_{MGS}\|_2 \approx \varepsilon_u \kappa_2(A).$$

而 Householder 变换计算的矩阵  $Q$  满足

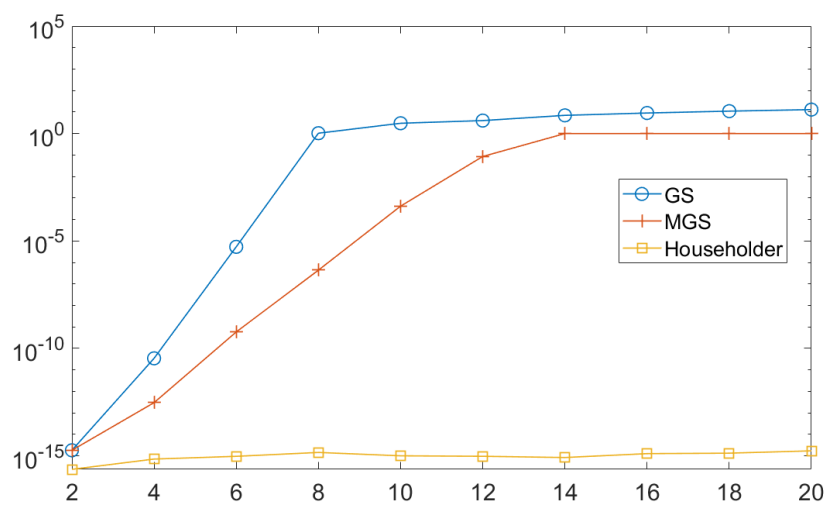
$$Q^T Q = I + E_H \quad \text{其中} \quad \|E_H\|_2 \approx \varepsilon_u.$$

因此, 如果正交性至关重要, 则当  $A$  的列向量接近线性相关时, 最好使用 Householder 变换.

**例 3.2** 编写程序, 分别用 GS, MGS 和 Householder 变换计算  $n$  阶 Hilbert 矩阵  $H$  的 QR 分解, 并比较三种算法的稳定性, 即观察  $\|\tilde{Q}\tilde{R} - H\|_2$  和  $\|\tilde{Q}^T\tilde{Q} - I\|_2$  的值, 其中  $\tilde{Q}$  和  $\tilde{R}$  是计算出来的 QR 分解矩阵因子. (LS\_QR\_stability.m)

试验结果如下:

$n$	GS		MGS		Householder	
	$\ \tilde{Q}\tilde{R} - H\ _2$	$\ \tilde{Q}^T\tilde{Q} - I\ _2$	$\ \tilde{Q}\tilde{R} - H\ _2$	$\ \tilde{Q}^T\tilde{Q} - I\ _2$	$\ \tilde{Q}\tilde{R} - H\ _2$	$\ \tilde{Q}^T\tilde{Q} - I\ _2$
2	0.00e+00	1.81e-15	0.00e+00	1.81e-15	1.24e-16	2.36e-16
4	3.93e-17	3.45e-11	5.55e-17	2.98e-13	2.46e-16	7.08e-16
6	6.21e-17	5.33e-06	2.78e-17	5.81e-10	1.49e-16	9.49e-16
8	7.48e-17	1.03e+00	6.59e-17	4.38e-07	2.57e-16	1.44e-15
10	6.52e-17	3.00e+00	7.49e-17	4.16e-04	6.36e-16	1.00e-15
12	7.87e-17	4.00e+00	7.55e-17	8.52e-02	4.68e-16	9.52e-16
14	6.54e-17	7.00e+00	8.54e-17	9.96e-01	5.71e-16	8.35e-16





### 3.4 奇异值分解

奇异值分解 (SVD) 不仅仅是矩阵计算中非常有用的工具之一, 也是图像处理、数据分析、压缩感知等领域的重要技术.

#### 3.4.1 奇异值与奇异值分解

设  $A \in \mathbb{C}^{m \times n}$  ( $m \geq n$ ), 则  $A^*A \in \mathbb{C}^{n \times n}$  和  $AA^* \in \mathbb{C}^{m \times m}$  都是 Hermite 半正定矩阵, 且它们具有相同的非零特征值 (都是正实数).

**定理 3.10 (SVD)** [21] 设  $A \in \mathbb{C}^{m \times n}$  ( $m \geq n$ ), 则存在酉矩阵  $U \in \mathbb{C}^{m \times m}$  和  $V \in \mathbb{C}^{n \times n}$  使得

$$U^*AV = \begin{bmatrix} \Sigma \\ 0 \end{bmatrix} \quad \text{或} \quad A = U \begin{bmatrix} \Sigma \\ 0 \end{bmatrix} V^*, \quad (3.10)$$

其中  $\Sigma = \text{diag}(\sigma_1, \sigma_2, \dots, \sigma_n) \in \mathbb{R}^{n \times n}$ , 且  $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_n \geq 0$ . 分解 (3.10) 称为  $A$  的**奇异值分解 (SVD)**, 而  $\sigma_1, \sigma_2, \dots, \sigma_n$  则称为  $A$  的**奇异值**.

**证明.** 首先假设  $A \neq 0$ , 否则只需令  $\Sigma = 0$  即可,  $U$  和  $V$  可以是任意酉矩阵.

下面我们对  $m$  和  $n$  用归纳法来证明.

当  $n = 1, m \geq 1$  时, 我们取  $\Sigma = \|A\|_2, V = 1, U \in \mathbb{C}^{m \times m}$  是第一列为  $u_1 = A/\|A\|_2$  的酉矩阵, 于是

$$A = U \begin{bmatrix} \Sigma \\ 0 \end{bmatrix} V^*$$

即为  $A$  的奇异值分解.

假设  $\mathbb{C}^{(m-1) \times (n-1)}$  中的矩阵都存在奇异值分解, 下面证明  $A \in \mathbb{C}^{m \times n}$  也存在有奇异值分解. 由 2-范数的定义可知, 存在向量  $v \in \mathbb{C}^n$  满足  $\|v\|_2 = 1$  使得  $\|A\|_2 = \|Av\|_2$ . 令

$$u = \frac{1}{\sigma} Av \in \mathbb{C}^m, \quad \text{其中 } \sigma = \|A\|_2,$$

则  $\|u\|_2 = 1$ . 我们将  $v$  和  $u$  都扩充成酉矩阵, 即存在  $\tilde{U} \in \mathbb{C}^{m \times (m-1)}$  和  $\tilde{V} \in \mathbb{C}^{n \times (n-1)}$ , 使得  $[u, \tilde{U}] \in \mathbb{C}^{m \times m}$  和  $[v, \tilde{V}] \in \mathbb{C}^{n \times n}$  都是酉矩阵. 于是

$$\tilde{U}^*Av = \tilde{U}^*(\sigma u) = 0, \quad u^*Av = u^*(\sigma u) = \sigma.$$

所以

$$\tilde{A} \triangleq [u, \tilde{U}]^* A [v, \tilde{V}] = \begin{bmatrix} u^*Av & u^*A\tilde{V} \\ \tilde{U}^*Av & \tilde{U}^*A\tilde{V} \end{bmatrix} = \begin{bmatrix} \sigma & u^*A\tilde{V} \\ 0 & \tilde{U}^*A\tilde{V} \end{bmatrix}.$$

又

$$\sigma = \|A\|_2 = \|\tilde{A}\|_2 = \|\tilde{A}^*\|_2 \geq \|\tilde{A}^*e_1\|_2 = \left\| \begin{bmatrix} \sigma & u^*A\tilde{V} \end{bmatrix}^* \right\|_2 = \sqrt{\sigma^2 + \|u^*A\tilde{V}\|_2^2},$$

所以  $\|u^*A\tilde{V}\|_2 = 0$ , 即  $u^*A\tilde{V} = 0$ . 于是

$$[u, \tilde{U}]^* A [v, \tilde{V}] = \begin{bmatrix} \sigma & 0 \\ 0 & A_1 \end{bmatrix},$$

其中  $A_1 = \tilde{U}^* A \tilde{V} \in \mathbb{C}^{(m-1) \times (n-1)}$ . 由归纳假设可知,  $A_1$  存在奇异值分解, 设为

$$A_1 = U_1 \begin{bmatrix} \Sigma_1 \\ 0 \end{bmatrix} V_1^*,$$

其中  $U_1 \in \mathbb{C}^{(m-1) \times (m-1)}$  和  $V_1 \in \mathbb{C}^{(n-1) \times (n-1)}$  都是酉矩阵,  $\Sigma_1 \in \mathbb{R}^{(n-1) \times (n-1)}$  是对角矩阵, 且其对角线元素按降序排列. 令

$$U = \begin{bmatrix} u, \tilde{U} \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & U_1 \end{bmatrix}, \quad V = \begin{bmatrix} v, \tilde{V} \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & V_1 \end{bmatrix},$$

则  $U \in \mathbb{C}^{m \times m}$  和  $V \in \mathbb{C}^{n \times n}$  都是酉矩阵, 且

$$\begin{aligned} U^* A V &= \begin{bmatrix} 1 & 0 \\ 0 & U_1 \end{bmatrix}^* \begin{bmatrix} u, \tilde{U} \end{bmatrix}^* A \begin{bmatrix} v, \tilde{V} \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & V_1 \end{bmatrix} \\ &= \begin{bmatrix} 1 & 0 \\ 0 & U_1 \end{bmatrix}^* \begin{bmatrix} \sigma & 0 \\ 0 & A_1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & V_1 \end{bmatrix} \\ &= \begin{bmatrix} \sigma & 0 \\ 0 & U_1^* A_1 V_1 \end{bmatrix} = \begin{bmatrix} \Sigma \\ 0 \end{bmatrix}, \end{aligned} \quad (3.11)$$

其中  $\Sigma = \begin{bmatrix} \sigma & 0 \\ 0 & \Sigma_1 \end{bmatrix}$ . 又

$$\sigma^2 = \|A\|_2^2 = \|U^* A V\|_2^2 = \left\| \begin{bmatrix} \Sigma \\ 0 \end{bmatrix} \right\|_2^2 = \rho \left( \begin{bmatrix} \sigma^2 & 0 \\ 0 & \Sigma_1^2 \end{bmatrix} \right),$$

所以  $\sigma$  不小于  $\Sigma_1$  中的所有对角线元素, 即  $\Sigma$  的对角线元素也是按降序排列. 因此, (3.11) 这就是  $A$  的奇异值分解.

由归纳法可知, 定理的结论成立. □

该定理也可以通过 Hermite 半正定矩阵的特征值分解来证明.

如果  $A \in \mathbb{R}^{m \times n}$  是实矩阵, 则  $U, V$  也都可以是实矩阵 [31].

由 (3.10) 可知,

$$A^* A = V \Sigma^* \Sigma V^*, \quad A A^* = U \begin{bmatrix} \Sigma \Sigma^* & 0 \\ 0 & 0 \end{bmatrix} U^*.$$

所以  $\sigma_1^2, \sigma_2^2, \dots, \sigma_n^2$  是  $A^* A$  和  $A A^*$  的特征值. 因此,  $A$  的奇异值就是  $A^* A$  的特征值的平方根.

若  $\text{rank}(A) = r \leq n$ , 则有

$$\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_r > \sigma_{r+1} = \dots = \sigma_n = 0.$$

特别地, 如果  $\text{rank}(A) = n$ , 则奇异值都是正的, 此时对角矩阵  $\Sigma$  非奇异.

矩阵  $U = [u_1, u_2, \dots, u_m]$  和  $V = [v_1, v_2, \dots, v_n]$  的列向量分别称为  $A$  的左奇异向量和右奇异向量, 即存在关系式

$$\begin{aligned} A v_i &= \sigma_i u_i, \quad i = 1, 2, \dots, n, \\ A^* u_i &= \sigma_i v_i, \quad i = 1, 2, \dots, n, \end{aligned}$$



$$A^*u_i = 0, \quad i = n+1, n+2, \dots, m.$$

由定理 3.10 可知

$$A = U \begin{bmatrix} \Sigma \\ 0 \end{bmatrix} V^* = \sigma_1 u_1 v_1^* + \sigma_2 u_2 v_2^* + \dots + \sigma_n u_n v_n^* = \sum_{i=1}^n \sigma_i u_i v_i^*.$$

记  $U_n = [u_1, u_2, \dots, u_n] \in \mathbb{C}^{m \times n}$ , 则  $U_n$  是单位列正交矩阵 (即  $U_n^* U_n = I_{n \times n}$ ), 且

$$A = U_n \Sigma V^*. \quad (3.12)$$

这就是所谓的**细 SVD** (**thin SVD**) [21] 或 **降阶 SVD** (**reduced SVD**) [52], 有的文献将 (3.12) 称为奇异值分解.

设  $k < n$ , 我们称

$$A_k = \sigma_1 u_1 v_1^* + \sigma_2 u_2 v_2^* + \dots + \sigma_k u_k v_k^* = \sum_{i=1}^k \sigma_i u_i v_i^*$$

为  $A$  的**截断 SVD** (**truncated SVD**).

#### 奇异值的应用

- 计算矩阵范数
- 计算矩阵条件数
- 计算矩阵的数值秩 (numerical rank)
- 求解最小二乘问题
- 图像处理, 压缩感知
- 矩阵和张量的低秩分解
- 主成分分析, 数据分析
- ... ..

### 3.4.2 奇异值的性质

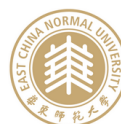
下面是关于奇异值的一些基本性质:

**定理 3.11** 设  $A = U \begin{bmatrix} \Sigma \\ 0 \end{bmatrix} V^*$  是  $A \in \mathbb{C}^{m \times n}$  ( $m \geq n$ ) 的奇异值分解, 则下面结论成立:

- (1)  $A^*A$  的特征值是  $\sigma_i^2$ , 对应的特征向量是  $v_i, i = 1, 2, \dots, n$ ;
- (2)  $AA^*$  的特征值是  $\sigma_i^2$  和  $m - n$  个零, 对应的特征向量是  $u_i, i = 1, 2, \dots, m$ ;
- (3)  $\|A\|_2 = \sigma_1, \|A\|_F = \sqrt{\sigma_1^2 + \sigma_2^2 + \dots + \sigma_n^2}$ ;
- (4) 若  $\text{rank}(A) = r \leq n$ , 则

$$\text{Ran}(A) = \text{span}\{u_1, u_2, \dots, u_r\}, \quad \text{Ker}(A) = \text{span}\{v_{r+1}, v_{r+2}, \dots, v_n\};$$

- (5) 设  $x \in \mathbb{C}^n$  且  $\|x\|_2 = 1$ , 则  $\sigma_n \leq \|Ax\|_2 \leq \sigma_1$ ;
- (6) (酉不变性) 设  $X \in \mathbb{C}^{m \times m}$  和  $Y \in \mathbb{C}^{n \times n}$  是酉矩阵, 则  $\sigma_i(X^*AY) = \sigma_i(A)$ .



(证明留作练习)

**定理 3.12** 设  $A = U\Sigma V^*$  是  $A \in \mathbb{C}^{n \times n}$  的奇异值分解, 则下面结论成立:

- (1)  $|\det(A)| = \sigma_1 \sigma_2 \cdots \sigma_n$ ;
- (2) 若  $A$  非奇异, 则  $\|A^{-1}\|_2 = \sigma_n^{-1}$ ,  $\kappa_2(A) = \sigma_1/\sigma_n$ ;
- (3) 若  $A$  是 Hermite 的, 且  $A = U\Lambda U^*$  是  $A$  的西特征值分解, 即  $U^*U = I$ ,  $\Lambda = \text{diag}(\lambda_1, \lambda_2, \dots, \lambda_n)$ . 设  $|\lambda_1| \geq |\lambda_2| \geq \cdots \geq |\lambda_n|$ , 则  $A = U\Sigma V$  是  $A$  的奇异值分解, 其中  $\sigma_i = |\lambda_i|$ ,  $v_i = \text{sign}(\lambda_i)u_i$ , 若  $\lambda_i = 0$ , 则取  $v_i = u_i$ ;
- (4) 矩阵  $H = \begin{bmatrix} 0 & A^* \\ A & 0 \end{bmatrix}$  的特征值是  $\pm\sigma_i$ , 对应的单位特征向量为  $\frac{1}{\sqrt{2}} \begin{bmatrix} v_i \\ \pm u_i \end{bmatrix}$ . ( $A \in \mathbb{C}^{m \times n}$  时也有类似结论)

(证明留作练习)

**例 3.3** 由奇异值的性质可知, 矩阵的谱条件数取决于最大奇异值和最小奇异值的比值, 如果最大奇异值与最小奇异值比较接近, 则谱条件数就比较小. 那么, 谱条件数与特征值有没有直接关系? 我们知道, 如果矩阵  $A$  对称正定, 则其谱条件数就是最大特征值与最小特征值的比值. 但是, 对于一般的矩阵, 则没有这个性质. 如下面的矩阵:

$$A = \begin{bmatrix} 1 & -\frac{1}{2} & \cdots & -\frac{1}{2} \\ & 1 & \ddots & \vdots \\ & & \ddots & -\frac{1}{2} \\ & & & 1 \end{bmatrix} \in \mathbb{R}^{n \times n},$$

即对角线全是 1, 严格上三角部分全是  $-\frac{1}{2}$ , 其它为 0. 已知  $A$  的所有特征值都是 1, 但其谱条件数却随着矩阵规模的增长而快速变大, 见下表:

$n$	10	20	30	40	50	60	70	80
$\kappa_2(A)$	6.3e+01	7.6e+03	6.8e+05	5.3e+07	3.9e+09	2.7e+11	1.8e+13	1.2e+15

另外, 通过直接计算可知

$$A^{-1} = \begin{bmatrix} 1 & \frac{1}{2} & \frac{1.5}{2} & \frac{1.5^2}{2} & \cdots & \frac{1.5^{n-2}}{2} \\ & 1 & \frac{1}{2} & \frac{1.5}{2} & \ddots & \vdots \\ & & 1 & \ddots & \ddots & \frac{1.5^2}{2} \\ & & & \ddots & \frac{1}{2} & \frac{1.5}{2} \\ & & & & 1 & \frac{1}{2} \\ & & & & & 1 \end{bmatrix}.$$

因此,  $\kappa_1(A)$  和  $\kappa_\infty(A)$  都是矩阵维数  $n$  的幂函数.

SVD 的一个重要应用是计算矩阵的低秩逼近.



**定理 3.13** 设  $A = U_n \Sigma V^*$  是  $A \in \mathbb{C}^{m \times n}$  的细奇异值分解. 令  $A_k = \sum_{i=1}^k \sigma_i u_i v_i^*$ , 则  $A_k$  是

$$\min_{B \in \mathbb{C}^{m \times n}, \text{rank}(B)=k} \|A - B\|_2 \quad (3.13)$$

的一个解, 且

$$\|A - A_k\|_2 = \sigma_{k+1}.$$

此时, 我们称  $A_k$  是  $A$  的一个秩  $k$  逼近.

**证明.** 设  $B \in \mathbb{C}^{m \times n}$  且  $\text{rank}(B) = k$ , 则

$$\dim(\text{Ker}(B)) + \dim(\text{span}\{V_{k+1}\}) = (n - k) + (k + 1) = n + 1 > n,$$

其中  $V_{k+1} = [v_1, v_2, \dots, v_{k+1}] \in \mathbb{C}^{n \times (k+1)}$ . 所以  $\text{Ker}(B)$  与  $\text{span}\{V_{k+1}\}$  有非零公共元素. 令  $0 \neq x \in \text{Ker}(B) \cap \text{span}\{V_{k+1}\}$ , 不失一般性, 我们假设  $\|x\|_2 = 1$ . 故存在  $y \in \mathbb{C}^{k+1}$  满足  $\|y\|_2 = 1$  使得  $x = V_{k+1}y$ . 于是

$$\begin{aligned} \|A - B\|_2^2 &\geq \|(A - B)x\|_2^2 = \|Ax\|_2^2 = \|U_n \Sigma V^* V_{k+1} y\|_2^2 \\ &= \left\| \Sigma \begin{bmatrix} I_{k+1} \\ 0 \end{bmatrix} y \right\|_2^2 = \left\| \begin{bmatrix} \Sigma_1 y \\ 0 \end{bmatrix} \right\|_2^2 = \sum_{i=1}^{k+1} \sigma_i^2 |y_i|^2 \geq \sigma_{k+1}^2, \end{aligned}$$

其中  $\Sigma_1 = \text{diag}(\sigma_1, \sigma_2, \dots, \sigma_{k+1})$ . 这里我们利用了性质  $\sum_{i=1}^{k+1} |y_i|^2 = \|y\|_2^2 = 1$ . 所以

$$\min_{B \in \mathbb{C}^{m \times n}, \text{rank}(B)=k} \|A - B\|_2 \geq \sigma_{k+1}.$$


又  $\text{rank}(A_k) = k$ , 且

$$\|A - A_k\|_2 = \left\| \sum_{i=k+1}^n \sigma_i u_i v_i^* \right\|_2 = \left\| U_n \begin{bmatrix} 0 & & & \\ & \ddots & & \\ & & 0 & \\ & & & \ddots & \\ & & & & \sigma_{k+1} & \\ & & & & & \ddots \\ & & & & & & \sigma_n \end{bmatrix} V^* \right\|_2 = \sigma_{k+1},$$


所以

$$\min_{B \in \mathbb{C}^{m \times n}, \text{rank}(B)=k} \|A - B\|_2 = \sigma_{k+1},$$

且  $A_k$  是问题 (3.13) 的一个解. □

 定理中的 (3.13) 式也可以改写为

$$\min_{B \in \mathbb{C}^{m \times n}, \text{rank}(B) \leq k} \|A - B\|_2 \quad (3.14)$$

 对于 Frobenius 范数, 我们有类似的结论.

如果  $\text{rank}(A) = l \leq n$ , 则  $\sigma_i = 0, i = l + 1, \dots, n$ . 此时

$$A = \sum_{i=1}^l \sigma_i u_i v_i^* = A_l. \quad (3.15)$$

## 3.5 线性最小二乘问题的求解方法

### 3.5.1 正规方程法

这里我们考虑超定线性最小二乘问题的求解.

**定理 3.14** 设  $A \in \mathbb{R}^{m \times n}$  ( $m \geq n$ ). 则  $x_* \in \mathbb{R}^n$  是线性最小二乘问题 (3.1) 的解当且仅当残量  $r = b - Ax_*$  与  $\text{Ran}(A)$  (值域) 正交, 即  $x_*$  是下面的正规方程的解

$$A^T(b - Ax) = 0 \quad \text{或} \quad A^T Ax = A^T b. \quad (3.16)$$

**证明.** 充分性: 设  $x_*$  是正规方程 (8.8) 的解. 对任意向量  $y \in \mathbb{R}^n$ , 由  $(b - Ax_*) \perp \text{Ran}(A)$  可知

$$\begin{aligned} \|Ay - b\|_2^2 &= \|(Ax_* - b) + A(y - x_*)\|_2^2 \\ &= \|Ax_* - b\|_2^2 + \|A(y - x_*)\|_2^2 \\ &\geq \|Ax_* - b\|_2^2. \end{aligned}$$

因此,  $x_*$  是线性最小二乘问题 (3.1) 的解.

必要性: 设  $x_*$  是线性最小二乘问题 (3.1) 的解. 用反证法, 假定  $z \triangleq A^T(b - Ax_*) \neq 0$ . 取  $y = x_* + \alpha z$ , 其中  $\alpha > 0$ , 则有

$$\|Ay - b\|_2^2 = \|Ax_* - b + \alpha Az\|_2^2 = \|Ax_* - b\|_2^2 - 2\alpha \|z\|_2^2 + \alpha^2 \|Az\|_2^2.$$

由于  $\|z\|_2 > 0$ , 当  $\alpha$  充分小时, 有  $2\|z\|_2^2 > \alpha \|Az\|_2^2$ , 即上式右端小于  $\|Ax_* - b\|_2^2$ . 这与  $x_*$  是最小二乘解相矛盾. 所以  $z = 0$ , 即  $A^T(b - Ax_*) = 0$ .  $\square$

由定理 3.14 可知, 求线性最小二乘问题 (3.1) 的解等价于求正规方程 (8.8) 的解. 由于

$$A^T b \in \text{Ran}(A^T) = \text{Ran}(A^T A),$$

因此正规方程  $A^T Ax = A^T b$  是相容 (consistent) 的, 即最小二乘解总是存在的. 当  $A$  非奇异时, 这个解也是唯一的.

**定理 3.15** 设  $A \in \mathbb{R}^{m \times n}$  ( $m > n$ ). 则  $A^T A$  对称正定当且仅当  $A$  是列满秩的, 即  $\text{rank}(A) = n$ . 此时, 线性最小二乘问题 (3.1) 的解是唯一的, 其表达式为

$$x = (A^T A)^{-1} A^T b.$$

当  $A$  列满秩时, 我们就可以使用 Cholesky 分解来求解正规方程, 总的运算量大约为  $mn^2 + \frac{1}{3}n^3 + \mathcal{O}(n^2)$ , 其中大部分的运算量 ( $mn^2$ ) 是用来计算  $A^T A$  (由于  $A^T A$  对称, 因此只需计算其下三角部分).

用正规方程法求解最小二乘问题, 运算量最小, 而且简单直观.

但由于  $A^T A$  的条件数是  $A$  的条件数的平方, 因此对于病态情形 (即  $A$  的条件数比较大), 不建议使用该方法.



**例 3.4** 下面的例子说明, 计算  $A^T A$  可能会损失计算精度: 设

$$A = \begin{bmatrix} 1 & 1 & 1 \\ \varepsilon & & \\ & \varepsilon & \\ & & \varepsilon \end{bmatrix},$$

则

$$A^T A = \begin{bmatrix} 1 + \varepsilon^2 & 1 & 1 \\ 1 & 1 + \varepsilon^2 & 1 \\ 1 & 1 & 1 + \varepsilon^2 \end{bmatrix}.$$

记  $\varepsilon_u$  为机器精度, 则当  $\varepsilon_u < \varepsilon < \sqrt{\varepsilon_u}$  时有  $\varepsilon^2 < \varepsilon_u$ , 由于舍入误差的原因, 通过浮点运算计算得到的  $A^T A$  是奇异的. 但我们注意到  $A$  是满秩的.

### 最小二乘解的几何含义

根据定理 3.14, 我们可以把  $b$  写成

$$b = Ax_* + r, \quad \text{其中 } r \perp \text{Ran}(A). \quad (3.17)$$

所以  $Ax_*$  就是  $b$  在  $\text{Ran}(A)$  上的正交投影, 见图 3.1.

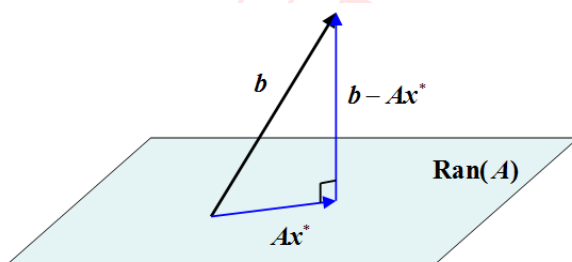


图 3.1. 最小二乘解的几何描述

最小二乘解可能并不唯一, 但分解 (3.17) 总是唯一的.

### 3.5.2 QR 分解法

这里假定  $A \in \mathbb{R}^{m \times n}$  ( $m \geq n$ ) 是满秩的. 设  $A$  的 QR 分解为  $A = QR$ , 我们用三种不同的方法来推导线性最小二乘问题的解.

- 将  $Q$  的扩充成一个正交矩阵, 记为  $[Q, \hat{Q}] \in \mathbb{R}^{m \times m}$ . 于是有

$$\begin{aligned} \|Ax - b\|_2^2 &= \left\| [Q, \hat{Q}]^T (Ax - b) \right\|_2^2 \\ &= \left\| [Q, \hat{Q}]^T (QRx - b) \right\|_2^2 \\ &= \left\| \begin{bmatrix} Rx - Q^T b \\ -\hat{Q}^T b \end{bmatrix} \right\|_2^2 \end{aligned}$$

$$\begin{aligned}
 &= \|Rx - Q^T b\|_2^2 + \|\hat{Q}^T b\|_2^2 \\
 &\geq \|\hat{Q}^T b\|_2^2,
 \end{aligned}$$

等号成立当且仅当  $Rx = Q^T b$ . 所以最小二乘解为

$$x_* = R^{-1}Q^T b.$$

- 将  $b$  写成  $b = (QQ^T + I - QQ^T)b$ , 则

$$\begin{aligned}
 Ax - b &= Ax - (QQ^T + I - QQ^T)b \\
 &= (Ax - QQ^T b) - (I - QQ^T)b.
 \end{aligned}$$

由  $A = QR$  和  $Q^T Q = I$  可知  $A^T(I - QQ^T) = A^T - R^T Q^T Q Q^T = A^T - R^T Q^T = 0$ , 所以

$$(Ax - QQ^T b)^T (I - QQ^T)b = -b^T QQ^T (I - QQ^T)b = 0,$$

即  $Ax - QQ^T b$  与  $(I - QQ^T)b$  正交. 所以

$$\begin{aligned}
 \|Ax - b\|_2^2 &= \|Ax - QQ^T b\|_2^2 + \|(I - QQ^T)b\|_2^2 \\
 &= \|Rx - Q^T b\|_2^2 + \|(I - QQ^T)b\|_2^2 \\
 &\geq \|(I - QQ^T)b\|_2^2,
 \end{aligned}$$

等号成立当且仅当  $Rx = Q^T b$ . 所以最小二乘解为

$$x_* = R^{-1}Q^T b.$$

事实上,  $Q$  的列向量组成  $\text{Ran}(A)$  的一组标准正交基, 因此  $QQ^T$  是  $\text{Ran}(A)$  上的正交投影算子. 由图 3.1 可知,  $x_*$  满足

$$Ax_* = QQ^T b,$$

即  $QRx_* = QQ^T b$ , 由此可得  $x_* = R^{-1}Q^T b$ .

- 解正规方程. 由定理 3.14 可知, 最小二乘解为

$$x_* = (A^T A)^{-1} A^T b = (R^T Q^T Q R)^{-1} R^T Q^T b = (R^T R)^{-1} R^T Q^T b = R^{-1}Q^T b.$$

用 QR 分解来求最小二乘解的运算量大约为  $2mn^2$  (如果采用 Householder 变换的话, 运算量大约为  $2mn^2 - \frac{2}{3}n^3$ ). 当  $m \gg n$  时, 大约为正规方程的两倍. 当  $m = n$  时, 几乎相同.

通常 QR 算法比较稳定, 是求解最小二乘问题的首选方法, 特别是当  $A$  条件数较大 (病态) 时.





## 3.5.3 奇异值分解法

设  $A \in \mathbb{R}^{m \times n}$  列满秩,  $A = U \begin{bmatrix} \Sigma \\ 0 \end{bmatrix} V^T$  是  $A$  的奇异值分解. 令  $U_n$  为  $U$  的前  $n$  列组成的矩阵, 即  $U = [U_n, \tilde{U}]$ , 则

$$\begin{aligned} \|Ax - b\|_2^2 &= \left\| U \begin{bmatrix} \Sigma \\ 0 \end{bmatrix} V^T x - b \right\|_2^2 \\ &= \left\| \begin{bmatrix} \Sigma \\ 0 \end{bmatrix} V^T x - [U_n, \tilde{U}]^T b \right\|_2^2 \\ &= \left\| \begin{bmatrix} \Sigma V^T x - U_n^T b \\ -\tilde{U}^T b \end{bmatrix} \right\|_2^2 \\ &= \|\Sigma V^T x - U_n^T b\|_2^2 + \|\tilde{U}^T b\|_2^2 \\ &\geq \|\tilde{U}^T b\|_2^2, \end{aligned}$$

等号当且仅当  $\Sigma V^T x - U_n^T b = 0$  时成立, 即

$$x = (\Sigma V^T)^{-1} U_n^T b = V \Sigma^{-1} U_n^T b.$$

这就是线性最小二乘问题 (3.1) 的解.

相比于正规方程法和 QR 分解法, 用 SVD 求解最小二乘问题具有更高的健壮性, 但由于需要计算系数矩阵的 SVD, 运算量远超正规方程法和 QR 分解法. 所以只有当系数矩阵秩亏或者接近秩亏时才使用 (此时 QR 分解法可能会失效).

**例 3.5** 分别用三种方法求解最小二乘问题, 比较运算时间. (源代码见 [LS\\_3methods.m](#))

三种方法的运算时间如下 (以秒为单位):

$n$	正规方程法	QR 分解法	奇异值分解法
500	0.0050	0.0220	0.0370
1000	0.0160	0.0340	0.1440
1500	0.0490	0.1330	0.5530
2000	0.0870	0.2070	1.4840
2500	0.1910	0.4430	3.1160
3000	0.2500	0.6950	5.9600
3500	0.4640	1.2130	10.0500
4000	0.4940	1.5700	14.8750
4500	0.6690	2.1680	20.6410
5000	1.0720	2.9350	28.6360

注: 这里结果可能受计算机系统和软件优化影响, 并不一定能准确反映各种方法的实际运算量.

## 3.6 广义逆与最小二乘

### 3.6.1 广义逆

广义逆的概念最早由 Moore [38] 于 1920 年提出, 他给出的定义如下: 设  $A \in \mathbb{C}^{m \times n}$ , 若  $X \in \mathbb{C}^{n \times m}$  满足

$$AX = P_{\text{Ran}(A)}, \quad XA = P_{\text{Ran}(X)}, \quad (3.18)$$

即  $AX$  和  $XA$  分别为  $\text{Ran}(A)$  和  $\text{Ran}(X)$  上的正交投影算子, 则称  $X$  是  $A$  的广义逆.

1955 年, Penrose [43] 利用下面四个矩阵方程给出了广义逆的定义.

**定义 3.2** 设  $A \in \mathbb{C}^{m \times n}$ , 若  $X \in \mathbb{C}^{n \times m}$  满足

$$AXA = A \quad (3.19)$$

$$XAX = X \quad (3.20)$$

$$(AX)^* = AX \quad (3.21)$$

$$(XA)^* = XA. \quad (3.22)$$

则称  $X$  为  $A$  的 **广义逆** (或 **Moore-Penrose 逆**, 简称 **MP 逆**), 记为  $A^\dagger$ .

方程组 (3.18) 和 (3.19)-(3.22) 分别称为 Moore 方程组和 Penrose 方程组.

可以证明, 以上两种定义是等价的.

若  $A \in \mathbb{C}^{n \times n}$  非奇异, 则  $A^\dagger = A^{-1}$ .

**定理 3.16** [67] 设  $A \in \mathbb{C}^{m \times n}$ , 则满足矩阵方程 (3.19)-(3.22) 的矩阵  $X \in \mathbb{C}^{n \times m}$  存在且唯一.

**证明. 存在性:** 可以通过 SVD 构造.

设  $\text{rank}(A) = r > 0$ , 且  $A$  的 SVD 为

$$A = U \begin{bmatrix} \Sigma_1 & 0 \\ 0 & 0 \end{bmatrix} V^*, \quad \Sigma_1 = \text{diag}(\sigma_1, \sigma_2, \dots, \sigma_r) \in \mathbb{R}^{r \times r}.$$

令

$$X = V \begin{bmatrix} \Sigma_1^{-1} & 0 \\ 0 & 0 \end{bmatrix} U^*.$$

容易验证,  $X$  满足矩阵方程 (3.19)-(3.22).

**唯一性:** 假设存在  $X$  和  $Y$  都满足矩阵方程 (3.19)-(3.22). 则

$$X = XAX = X(AX)^* = XX^*A^* = XX^*(AYA)^* = X(AX)^*(AY)^* = (XAX)(AY) = XAY.$$

另一方面,

$$Y = YAY = (YA)^*Y = A^*Y^*Y = (AXA)^*Y^*Y = (XA)^*(YA)^*Y = XAYAY = XAY.$$

所以  $Y = X$ . □



## 3.6.2 广义逆基本性质

**定理 3.17** 设  $A \in \mathbb{C}^{m \times n}$ , 则

- (1)  $(A^\dagger)^\dagger = A$ ;
- (2)  $(A^\top)^\dagger = (A^\dagger)^\top$ ,  $(A^*)^\dagger = (A^\dagger)^*$ ;
- (3)  $\text{rank}(A) = \text{rank}(A^\dagger) = \text{rank}(A^\dagger A)$ ;
- (4)  $(AA^*)^\dagger = (A^*)^\dagger A^\dagger$ ,  $(A^*A)^\dagger = A^\dagger (A^*)^\dagger$ ;
- (5)  $(AA^*)^\dagger AA^* = AA^\dagger$ ,  $(A^*A)^\dagger A^*A = A^\dagger A$ ;
- (6)  $A^\dagger = (A^*A)^\dagger A^* = A^*(AA^*)^\dagger$ ,  
特别地, 若  $A$  列满秩, 则  $A^\dagger = (A^*A)^{-1}A^*$ , 若  $A$  行满秩, 则  $A^\dagger = A^*(AA^*)^{-1}$ ;
- (7) 若  $U, V$  是酉矩阵, 则  $(UAV)^\dagger = V^*A^\dagger U^*$ .

一般来说, 当  $A, B$  是方阵时,

- $(AB)^\dagger \neq B^\dagger A^\dagger$ ;
- $AA^\dagger \neq A^\dagger A$ ;
- $(A^k)^\dagger \neq (A^\dagger)^k$ ;
- $A$  和  $A^\dagger$  的非零特征值并不是互为倒数.

## 3.6.3 广义逆的计算

## 利用 SVD

我们可以利用  $A$  的奇异值分解来计算  $A^\dagger$ , 但运算量较大. 因为奇异值分解通常与  $AA^*$  或  $A^*A$  的特征值分解有关.

## 利用满秩分解

**定理 3.18** 设  $A \in \mathbb{R}^{m \times n}$ .

- (1) 若  $A$  是列满秩矩阵, 则  $A^\dagger = (A^*A)^{-1}A^*$ ;
- (2) 若  $A$  是行满秩矩阵, 则  $A^\dagger = A^*(AA^*)^{-1}$ ;
- (3) 若  $A$  的秩是  $r \leq \min\{m, n\}$ , 且其满秩分解为  $A = FG$ , 其中  $F \in \mathbb{R}^{m \times r}$ ,  $G \in \mathbb{R}^{r \times n}$ , 则  

$$A^\dagger = G^\dagger F^\dagger = G^*(GG^*)^{-1}(F^*F)^{-1}F^*.$$

## 利用 QR 分解

**定理 3.19** 设  $A \in \mathbb{R}^{m \times n}$  ( $m \geq n$ ) 是列满秩矩阵, 其 QR 分解为  $A = QR$ , 其中  $Q \in \mathbb{R}^{m \times n}$ ,

$R \in \mathbb{R}^{n \times n}$ , 则

$$A^\dagger = R^{-1}Q^*.$$

### 其他算法

其他比较重要的算法有: [Greville 递推算法](#), [Cline 算法](#)等.

### 3.6.4 广义逆与线性最小二乘

**定理 3.20** 设  $A \in \mathbb{R}^{m \times n}$  ( $m \geq n$ ), 则线性最小二乘问题 (3.1) 的解为

$$x = A^\dagger b + (I - P_{\text{Ran}(A^\top)})z, \quad \forall z \in \mathbb{R}^n. \quad (3.23)$$

(证明可参见 [67])

通常, 线性最小二乘问题的解 (3.23) 不是唯一的. 但当  $A$  列满秩时,  $P_{\text{Ran}(A^\top)} = I$ , 此时解唯一.

**定理 3.21** 设  $A \in \mathbb{R}^{m \times n}$  ( $m \geq n$ ). 记线性最小二乘问题 (3.1) 的解集为  $S$ , 则

$$\min_{x \in S} \|x\|_2 \quad (3.24)$$

存在唯一解, 即线性最小二乘问题 (3.1) 存在唯一的最小范数解.

(证明可参见 [67])



## 3.7 最小二乘扰动分析 \*

**定理 3.22** 设  $A \in \mathbb{R}^{m \times n}$  ( $m \geq n$ ) 且  $\text{rank}(A) = n$ . 设  $x$  是线性最小二乘问题 (3.1) 的解,  $\tilde{x}$  极小化  $\|(A + \delta A)\tilde{x} - (b + \delta b)\|_2$ , 则

$$\frac{\|\tilde{x} - x\|_2}{\|x\|_2} \leq \varepsilon \cdot \left\{ \frac{2\kappa_2(A)}{\cos \theta} + \kappa_2^2(A) \tan \theta \right\} + O(\varepsilon^2),$$

其中  $\kappa_2(A) = \sigma_1(A)/\sigma_n(A)$ ,  $\theta$  为  $b$  与  $\text{Ran}(A)$  的夹角,

$$\varepsilon \triangleq \max \left\{ \frac{\|\delta A\|_2}{\|A\|_2}, \frac{\|\delta b\|_2}{\|b\|_2} \right\},$$

并假定  $\varepsilon \cdot \kappa_2(A) < 1$  (确保  $A + \delta A$  满秩, 从而  $\tilde{x}$  唯一确定).

我们记

$$\kappa_{LS} \triangleq \frac{2\kappa_2(A)}{\cos \theta} + \kappa_2^2(A) \tan \theta,$$

这就是最小二乘问题的条件数. 当  $\theta = 0$  时,  $b \in \text{Ran}(A)$ , 此时  $\kappa_{LS} = 2\kappa_2(A)$ ; 当  $\theta = \pi/2$  时,  $b \perp \text{Ran}(A)$ , 此时最小二乘解为  $x = 0$ , 而  $\kappa_{LS} = \infty$ ; 当  $0 < \theta < \pi/2$  时,  $\kappa_{LS} = O(\kappa_2^2(A))$ .

定义残量  $r = b - Ax$ ,  $\tilde{r} = (b + \delta b) - (A + \delta A)\tilde{x}$ , 我们有下面的性质 [30]

$$\frac{\|\tilde{r} - r\|_2}{\|r\|_2} \leq \varepsilon \cdot (1 + 2\kappa_2(A)).$$

当我们使用 QR 分解或 SVD 分解求解最小二乘问题时, 由于采用的是正交变换, 它们都是数值稳定的. 而正规方程涉及求解方程组  $A^T A x = A^T b$ , 其精度依赖于条件数  $\kappa_2(A^T A) = \kappa_2^2(A)$ , 因为其误差是以  $\kappa_2^2(A)$  倍数增长. 因此当  $A$  的条件数较大时, 正规方程法的精度会大大降低.

### 3.8 最小二乘问题的推广与应用 \*

#### 3.8.1 正则化

在求解超定线性方程组时, 我们极小化  $\|Ax - b\|_2^2$ . 而对于欠定线性方程组, 由于解不唯一, 我们往往还需要极小化  $\|x\|_2^2$ . 两者合起来就是

$$\min_{x \in \mathbb{R}^n} \frac{1}{2} \|Ax - b\|_2^2 + \frac{\alpha}{2} \|x\|_2^2, \quad (3.25)$$

其中  $\alpha > 0$  是正则化参数. 对应的目标函数记为

$$J(x) = \|Ax - b\|_2^2 + \alpha \|x\|_2^2.$$

当  $\alpha > 0$  时,  $J(x)$  是一个严格凸的二次函数, 因此存在唯一的最小值点. 令  $J(x)$  关于  $x$  的一阶导数为零, 则可得

$$(A^T A + \alpha I)x = A^T b.$$

由于  $A^T A + \alpha I$  对称正定, 故非奇异. 所以问题 (3.25) 的唯一解为

$$x = (A^T A + \alpha I)^{-1} A^T b.$$

#### 3.8.2 加权正则化

一类应用更广泛的问题是下面的加权正则化问题

$$\min_{x \in \mathbb{R}^n} \frac{1}{2} \|Ax - b\|_2^2 + \frac{\alpha}{2} \|Wx\|_2^2, \quad (3.26)$$

其中  $\alpha > 0$  是正则化参数,  $W \in \mathbb{R}^{p \times n}$  是广义加权矩阵, 可以是非负对角矩阵, 对称正定矩阵, 也可以是一般矩阵 (如一阶差分算子, 二阶差分算子等). 需要指出的是,  $W$  不一定要要求是方阵.

经过类似的推导, 可知问题 (3.26) 的解满足

$$(A^T A + \alpha W^T W)x = A^T b.$$

如果  $A^T A + \alpha W^T W$  非奇异, 则存在唯一解

$$x = (A^T A + \alpha W^T W)^{-1} A^T b.$$

#### 3.8.3 约束最小二乘问题

考虑带有约束的最小二乘问题

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & \frac{1}{2} \|Ax - b\|_2^2 \\ \text{s.t.} \quad & Bx = f \end{aligned} \quad (3.27)$$

其中  $Bx = f$  是约束条件,  $B \in \mathbb{R}^{p \times n}$  和  $f \in \mathbb{R}^p$  都是给定的. 对应的 Lagrange 函数为

$$J(x) = \frac{1}{2} \|Ax - b\|_2^2 + \lambda^T (Bx - f).$$

其中  $\lambda$  是 Lagrange 乘子. 分别对  $x$  和  $\lambda$  求一阶导数, 并令其等于零, 可得

$$\begin{bmatrix} A^T A & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} x \\ \lambda \end{bmatrix} = \begin{bmatrix} A^T b \\ f \end{bmatrix}.$$



如果  $A^T A$  非奇异, 且  $B$  行满秩, 则存在唯一解

$$\begin{aligned}\lambda &= [B(A^T A)^{-1} B^T]^{-1} [B(A^T A)^{-1} A^T b - f] \\ x &= (A^T A)^{-1} (A^T b - B^T \lambda).\end{aligned}$$

几类特殊情形

- 如果  $A = I$  且  $f = 0$ , 则有

$$\begin{aligned}\min_{x \in \mathbb{R}^n} \frac{1}{2} \|x - b\|_2^2 \quad \text{s.t.} \quad Bx = 0 \\ \implies x = b - B^T (BB^T)^{-1} Bb.\end{aligned}$$

- 如果  $b = 0$ , 则有

$$\begin{aligned}\min_{x \in \mathbb{R}^n} \frac{1}{2} \|Ax\|_2^2 \quad \text{s.t.} \quad Bx = f \\ \implies x = (A^T A)^{-1} B^T [B(A^T A)^{-1} B^T]^{-1} f.\end{aligned}$$

- 如果  $A = I$  且  $b = 0$ , 则有

$$\begin{aligned}\min_{x \in \mathbb{R}^n} \frac{1}{2} \|x\|_2^2 \quad \text{s.t.} \quad Bx = f \\ \implies x = B^T (BB^T)^{-1} f.\end{aligned}$$

### 3.8.4 多项式数据拟合

最小二乘的一个重要应用就是低次多项式数据拟合: 已知平面上  $n$  个点  $\{(t_i, f_i)\}_{i=1}^n$ , 寻找一个低次多项式来拟合这些数据.

设拟合多项式为

$$p(t) = a_0 + a_1 t + a_2 t^2 + \cdots + a_m t^m.$$

通常  $m \ll n$ . 将上述  $n$  个点  $\{(t_i, f_i)\}_{i=1}^n$  代入可得

$$\begin{bmatrix} 1 & t_1 & t_1^2 & \cdots & t_1^m \\ 1 & t_2 & t_2^2 & \cdots & t_2^m \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & t_n & t_n^2 & \cdots & t_n^m \end{bmatrix}_{n \times (m+1)} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_m \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_n \end{bmatrix} \quad \text{或} \quad Ax = f,$$

其中  $A \in \mathbb{R}^{n \times (m+1)}$ ,  $x = [a_0, a_1, \dots, a_m]^T$ . 由于  $m \ll n$ , 该方程组是超定的, 解通常是不存在的. 因此, 我们寻找一个近似解, 使得残量  $\|f - Ax\|_2$  最小, 即求解最小二乘问题

$$\min_{x \in \mathbb{R}^{m+1}} \|f - Ax\|_2^2$$

### 3.8.5 应线性预测

预测一个时间序列的未来走向, 其中一个常用方法就是线性预测 (linear prediction). 假定一个时间序列在  $t_k$  时刻的值  $f_k$  线性依赖于其前  $m$  个时刻的值  $f_{k-1}, f_{k-2}, \dots, f_{k-m}$ , 即

$$f_k = a_1 f_{k-1} + a_2 f_{k-2} + \cdots + a_m f_{k-m}. \quad (3.28)$$

现在已经测得该时间序列的前  $n$  个值  $f_i$ ,  $i = 0, 1, 2, \dots, n-1$ , 如何预测其未来的取值. 这里  $n \gg m$ .

将现有的数据代入关系式 (3.28) 可得

$$\begin{bmatrix} f_{m-1} & f_{m-2} & f_{m-3} & \cdots & f_0 \\ f_m & f_{m-1} & f_{m-2} & \cdots & f_1 \\ \vdots & \vdots & \vdots & & \vdots \\ f_{n-2} & f_{n-3} & f_{n-4} & \cdots & f_{n-m-1} \end{bmatrix}_{(n-m) \times m} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_m \end{bmatrix} = \begin{bmatrix} f_m \\ f_{m+1} \\ \vdots \\ f_{n-1} \end{bmatrix} \quad \text{或} \quad Ax = f,$$

其中  $A \in \mathbb{R}^{(n-m) \times m}$ ,  $x = [a_1, a_2, \dots, a_m]^T$ . 这也是一个超定问题, 其解也是通过求解下面的最小二乘问题来获取

$$\min_{x \in \mathbb{R}^m} \|f - Ax\|_2^2$$

### 3.8.6 信号恢复与图像处理

在获取数字信号时, 由于各种各样的原因, 最后得到的信号总会带有一定的噪声. 去噪是数字信号和图像处理中的一个基本问题. 其中一个有效方法就是加权最小二乘法.

**例 3.6** 举例: `LS_denoise.m`

### 3.8.7 SVD 与图像压缩

用矩阵  $A$  表示图像, 然后求它的 SVD, 最后保留前  $k$  个奇异值, 即用  $A$  的截断 SVD 来近似  $A$ , 从而达到图像压缩的目的.

**例 3.7** 举例: `LS_SVD_ImageCompress_01.m`, `LS_SVD_ImageCompress_02.m`





### 3.9 课后习题

**练习 3.1** 设  $A \in \mathbb{R}^{m \times n}$ , 其中  $m \leq n$ . 试证明: 矩阵  $\begin{bmatrix} I & A^T \\ A & 0 \end{bmatrix}$  非奇异的充要条件是  $\text{rank}(A) = m$ .

**练习 3.2\*** 设  $B \in \mathbb{R}^{m \times m}$  对称半正定,  $A \in \mathbb{R}^{m \times n}$ , 其中  $m \geq n$ . 试证明: 矩阵  $\begin{bmatrix} B & A \\ A^T & 0 \end{bmatrix}$  非奇异的充要条件是  $A$  列满秩且矩阵  $[B, A]$  行满秩 (即  $A$  列满秩且  $\text{Ker}(B) \cap \text{Ker}(A^T) = \{0\}$ ).

**练习 3.3** 设  $\tau \neq 0$ , 向量  $u, v \in \mathbb{R}^n$  均不为零, 求矩阵  $E(u, v, \tau) = I - \tau uv^T$  的特征值.

**练习 3.4** 设  $A \in \mathbb{R}^{n \times n}$  是正交矩阵, 则  $A$  可表示成至多  $n$  个 Householder 变换的乘积.

**练习 3.5** 设  $A \in \mathbb{R}^{n \times n}$  是正交矩阵, 则  $A$  可表示成至多  $\frac{1}{2}n(n-1)$  个 Givens 变换的乘积.

**练习 3.6** 设  $x = [x_1, x_2, \dots, x_n]^T \in \mathbb{R}^n$  是一个非零向量,  $H$  是 Householder 矩阵, 满足  $Hx = \alpha e_1$ . 证明:  $H$  的第一列与  $x$  平行.

**练习 3.7** 设  $H_k \in \mathbb{R}^{k \times k}$  是 Householder 变换, 其中  $k < n$ .

证明:  $H_n = \begin{bmatrix} I_{n-k} & 0 \\ 0 & H_k \end{bmatrix}$  是  $n$  阶 Householder 变换.

**练习 3.8** 设  $R = \begin{bmatrix} r_{11} & r_{12} \\ 0 & r_{22} \end{bmatrix} \in \mathbb{R}^{2 \times 2}$ , 求一个 Givens 变换  $G$ , 使得  $G^T R G = \begin{bmatrix} r_{22} & r_{12} \\ 0 & r_{11} \end{bmatrix}$ .

**练习 3.9** 设  $R \in \mathbb{R}^{n \times n}$  是一个上三角矩阵, 且对角线元素互不相同. 证明: 存在正交矩阵  $Q$ , 使得  $Q^T R Q$  为上三角矩阵, 且对角线元素为  $R$  的对角线元素的降序排列.

**练习 3.10** 证明定理 3.11 和定理 3.12. (奇异值的相关性质)

**练习 3.11** 设  $A \in \mathbb{R}^{n \times n}$ , 证明  $\rho(A) \leq \sigma_{\max}(A)$ .

**练习 3.12 (极分解)** 设  $A \in \mathbb{C}^{n \times n}$ . 证明:

(1) 存在酉矩阵  $U$  和唯一的 Hermite 半正定矩阵  $P$ , 使得  $A = PU$ .

(2) 进一步, 若  $A$  非奇异, 则  $U$  也唯一.

**练习 3.13\*** 设  $A \in \mathbb{C}^{n \times n}$ .

证明:  $A$  可对角化当且仅当存在 Hermite 正定矩阵  $P$  使得  $P^{-1}AP$  是正规矩阵.

**练习 3.14** 设  $A \in \mathbb{R}^{n \times n}$ , 证明  $\|A\|_F^2 \geq \sum_{i=1}^n |\lambda_i(A)|^2$ .

**练习 3.15** 设  $A \in \mathbb{C}^{n \times n}$  有  $n$  个互不相同的非零特征值, 其 Schur 分解为  $A = URU^*$ , 其中  $U$  为酉矩阵,  $R$  为上三角矩阵. 设矩阵  $B \in \mathbb{C}^{n \times n}$  满足  $AB = BA$ . 证明:  $U^*BU$  是上三角矩阵.

**练习 3.16** 证明下面的结论:

(1) 设  $A, B \in \mathbb{R}^{n \times n}$  可交换, 即  $AB = BA$ . 若  $A$  是对角矩阵且对角线元素互不相等, 则  $B$  也是对角矩阵.

(2) 设  $A, B \in \mathbb{R}^{n \times n}$  可交换, 即  $AB = BA$ . 若  $A$  是块对角矩阵且对角块为标量矩阵, 即  $A = \text{blkdiag}(\lambda_1 I_{n_1}, \lambda_2 I_{n_2}, \dots, \lambda_p I_{n_p})$ , 其中  $n_1 + n_2 + \dots + n_p = n$ , 则  $B$  是具有相应分块

结构的块对角矩阵, 即  $B = \text{blkdiag}(B_1, B_2, \dots, B_p)$ , 其中  $\dim(B_i) = n_i, i = 1, 2, \dots, p$ .

(3) 设  $A, B \in \mathbb{R}^{n \times n}$  都是对称矩阵. 试证明:  $AB = BA$  的充要条件是存在正交矩阵  $Q$  使得

$$Q A Q^T = \Lambda_A, \quad Q B Q^T = \Lambda_B,$$

其中  $\Lambda_A$  和  $\Lambda_B$  分别表示由  $A$  和  $B$  的特征值构成的对角矩阵.

练习 3.17 用 Householder 变换计算矩阵  $A = \begin{bmatrix} 1 & 1 & 1 \\ 2 & -1 & -1 \\ 2 & -4 & 10 \end{bmatrix}$  的 QR 分解.

练习 3.18 设  $A \in \mathbb{R}^{n \times n}$  是一个对角加边矩阵

$$A = \begin{bmatrix} a_1 & b_2 & b_3 & \cdots & b_n \\ c_2 & a_2 & & & \\ c_3 & & a_3 & & \\ \vdots & & & \ddots & \\ c_n & & & & a_n \end{bmatrix}.$$

试给出用 Givens 变换计算  $A$  的 QR 分解的详细算法, 使得运算量为  $O(n^2)$ .

练习 3.19 设  $A \in \mathbb{R}^{m \times n}$  ( $m \geq n$ ) 且  $\text{rank}(A) = n$ . 计算矩阵  $\begin{bmatrix} I & A \\ A^T & 0 \end{bmatrix}$  的谱条件数. (用  $A$  的奇异值表示)

练习 3.20\* 设  $x = [x_1, x_2, \dots, x_n]^T \in \mathbb{C}^n$  是一个非零复向量, 问: 是否存在 Householder 矩阵  $H(v)$  使得  $H(v)x = \alpha e_1$ ? 若存在, 如何计算  $v$ ?

练习 3.21\* 设  $A \in \mathbb{R}^{m \times n}$  ( $m \geq n$ ), 如果  $A$  不是满秩的, 如何求解相应的最小二乘问题?

## 实践题

练习 3.22 编写 Householder 变换函数: 对任意给定的变量  $x \in \mathbb{R}^n$ , 输出  $v$  使得  $H(v)x = \|x\|_2 e_1$ , 其中  $H(v) = I - 2vv^T$ , 函数原型: `v=House2(x)`

练习 3.23 编写 Householder 变换函数: 对任意给定的变量  $x \in \mathbb{R}^n$ , 输出  $v$  使得  $H(v)x = \|x\|_2 e_1$ , 其中  $H(v) = I - \beta vv^T$  且  $v$  的第一个分量为 1. 函数原型: `[beta,v]=House3(x)`

练习 3.24 编写基于 Householder 变换的 QR 分解, 函数原型: `[Q,R]=QR_Householder(A)`

练习 3.25 设  $A = \begin{bmatrix} R \\ S \end{bmatrix}$ , 其中  $R \in \mathbb{R}^{n \times n}$  是上三角矩阵,  $S \in \mathbb{R}^{m \times n}$  是稠密矩阵, 试描述  $A$  的基于 Householder 变换的 QR 算法. 要求算法过程中始终保持  $R$  中的零元.

练习 3.26 构造并实现矩阵的 LQ 分解算法, 即  $A = LQ$ , 其中  $L$  为下三角矩阵,  $Q$  为正交矩阵.



# 4

## 线性方程组迭代方法

直接法的运算量为  $O(n^3)$ , 所以随着矩阵规模的增大, 直接法的运算量也随之快速增长. 对于大规模的线性方程组, 由于运算量太大, 直接法一般不再被采用, 取而代之的是迭代方法.

考虑线性方程组

$$Ax = b,$$

其中  $A \in \mathbb{R}^{n \times n}$  非奇异. 迭代方法的基本思想: 给定一个迭代初始值  $x^{(0)}$ , 通过一定的迭代格式生成一个迭代序列  $\{x^{(k)}\}_{k=0}^{\infty}$ , 使得

$$\lim_{k \rightarrow \infty} x^{(k)} = x_* \triangleq A^{-1}b.$$

目前常用的迭代方法主要有两类, 一类是定常迭代方法, 如 Jacobi, Gauss-Seidel, SOR 等. 另一类是子空间迭代法, 如 CG, GMRES 等.

### 关于迭代法的相关参考文献

- [1] G. H. Golub and C. F. Van Loan, [Matrix Computations](#), 4th, 2013 [21]
- [2] R. S. Varga, [Matrix Iterative Analysis](#), 2nd, 2000 [57]
- [3] D. M. Young, [Iterative Solution of Large Linear Systems](#), 1971 [58]
- [4] Y. Saad, [Iterative Methods for Sparse Linear Systems](#), 2nd, 2003 [47]
- [5] A. Greenbaum, [Iterative Methods for Solving Linear Systems](#), 1997 [23]

## 4.1 迭代法基本概念

### 4.1.1 向量序列与矩阵序列的收敛性

首先给出向量序列收敛的定义.

**定义 4.1 (向量序列的收敛)** 设  $\{x^{(k)}\}_{k=0}^{\infty}$  是  $\mathbb{R}^n$  中的一个向量序列. 如果存在向量  $x = [x_1, x_2, \dots, x_n]^T \in \mathbb{R}^n$  使得

$$\lim_{k \rightarrow \infty} x_i^{(k)} = x_i, \quad i = 1, 2, \dots, n,$$

其中  $x_i^{(k)}$  表示  $x^{(k)}$  的第  $i$  个分量. 则称  $\{x^{(k)}\}$  (按分量) **收敛** 到  $x$ , 记为

$$\lim_{k \rightarrow \infty} x^{(k)} = x.$$

我们称  $x$  为序列  $\{x^{(k)}\}$  的 **极限**.

相类似地, 我们可以给出矩阵序列收敛的定义.

**定义 4.2 (矩阵序列的收敛)** 设  $\{A^{(k)} = [a_{ij}^{(k)}]\}_{k=0}^{\infty}$  是  $\mathbb{R}^{n \times n}$  中的一个矩阵序列. 如果存在矩阵  $A = [a_{ij}] \in \mathbb{R}^{n \times n}$  使得

$$\lim_{k \rightarrow \infty} a_{ij}^{(k)} = a_{ij}, \quad i, j = 1, 2, \dots, n,$$

则称  $A^{(k)}$  收敛到  $A$ , 记为

$$\lim_{k \rightarrow \infty} A^{(k)} = A.$$

我们称  $A$  为  $A^{(k)}$  的 **极限**.

**例 4.1** 设  $0 < |a| < 1$ , 考虑矩阵序列  $\{A^{(k)}\}$ , 其中

$$A^{(k)} = \begin{bmatrix} a & 1 \\ 0 & a \end{bmatrix}^k, \quad k = 1, 2, \dots$$

易知当  $k \rightarrow \infty$  时, 有

$$A^{(k)} = \begin{bmatrix} a & 1 \\ 0 & a \end{bmatrix}^k = \begin{bmatrix} a^k & ka^{k-1} \\ 0 & a^k \end{bmatrix} \rightarrow \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}.$$

关于向量序列和矩阵序列的收敛性, 我们有下面的结论.

**定理 4.1** 设向量序列  $\{x^{(k)}\}_{k=0}^{\infty} \subset \mathbb{R}^n$ , 矩阵序列  $\{A^{(k)} = [a_{ij}^{(k)}]\}_{k=0}^{\infty} \subset \mathbb{R}^{n \times n}$ , 则

- (1)  $\lim_{k \rightarrow \infty} x^{(k)} = x \iff \lim_{k \rightarrow \infty} \|x^{(k)} - x\| = 0$ , 其中  $\|\cdot\|$  为任一向量范数;
- (2)  $\lim_{k \rightarrow \infty} A^{(k)} = A \iff \lim_{k \rightarrow \infty} \|A^{(k)} - A\| = 0$ , 其中  $\|\cdot\|$  为任一矩阵范数;

(板书)

由定理 4.1 可以立即得到下面的两个结论:

$$\lim_{k \rightarrow \infty} A^{(k)} = 0 \iff \lim_{k \rightarrow \infty} \|A^{(k)}\| = 0$$

和

$$\lim_{k \rightarrow \infty} A^k = 0 \iff \lim_{k \rightarrow \infty} \|A^k\| = 0$$



**定理 4.2** 设矩阵序列  $\{A^{(k)} = [a_{ij}^{(k)}]\}_{k=0}^{\infty} \subset \mathbb{R}^{n \times n}$ , 则

$$\lim_{k \rightarrow \infty} A^{(k)} = 0 \iff \lim_{k \rightarrow \infty} A^{(k)}x = 0, \quad \forall x \in \mathbb{R}^n.$$

**证明.** 先证明**必要性** (“ $\implies$ ”). 由条件  $\lim_{k \rightarrow \infty} A^{(k)} = 0$  可知, 对任意算子范数都有  $\lim_{k \rightarrow \infty} \|A^{(k)}\| = 0$ . 因此, 对任意  $x \in \mathbb{R}^n$  有

$$\|A^{(k)}x\| \leq \|A^{(k)}\| \cdot \|x\| \rightarrow 0 \quad (k \rightarrow \infty) \quad \text{即} \quad \lim_{k \rightarrow \infty} A^{(k)}x = 0.$$

下面证明**充分性** (“ $\impliedby$ ”). 取  $x = e_i$ , 即单位矩阵的第  $i$  列, 则由  $\lim_{k \rightarrow \infty} A^{(k)}e_i = 0$  可知,  $A^{(k)}$  的第  $i$  列的极限为 0. 令  $i = 1, 2, \dots, n$ , 则可得  $\lim_{k \rightarrow \infty} A^{(k)} = 0$ .  $\square$

**定理 4.3** 设  $B \in \mathbb{R}^{n \times n}$ , 若存在矩阵范数使得  $\|B\| < 1$ , 则  $\lim_{k \rightarrow \infty} B^k = 0$ .

**证明.** 由条件  $\|B\| < 1$  可知,

$$\|B^k\| \leq \|B\|^k \rightarrow 0 \quad (k \rightarrow \infty).$$

所以  $\lim_{k \rightarrow \infty} B^k = 0$ .  $\square$

**定理 4.4** 设  $B \in \mathbb{R}^{n \times n}$ , 则  $\lim_{k \rightarrow \infty} B^k = 0$  当且仅当  $\rho(B) < 1$ .

**证明.** 先证明**必要性** (“ $\implies$ ”). 反证法. 假设  $\rho(B) \geq 1$ , 则  $B$  存在特征值  $\lambda$ , 满足  $|\lambda| \geq 1$ . 设其对应的特征向量为  $x \neq 0$ , 即  $Bx = \lambda x$ , 则  $B^k x = \lambda^k x$ . 由于  $|\lambda| \geq 1$ , 当  $k \rightarrow \infty$  时  $\lambda^k x$  不可能收敛到 0, 这与条件矛盾. 所以结论成立, 即  $\rho(B) < 1$ .

下面证明**充分性** (“ $\impliedby$ ”). 令  $\varepsilon = \frac{1}{2}(1 - \rho(B))$ , 则  $\varepsilon > 0$ . 因此, 由定理 1.21 可知, 存在某个矩阵范数  $\|\cdot\|_\varepsilon$ , 使得

$$\|B\|_\varepsilon \leq \rho(B) + \varepsilon = \frac{1}{2}(1 + \rho(B)) < 1.$$

所以由定理 4.3 可知  $\lim_{k \rightarrow \infty} B^k = 0$ .  $\square$

**推论 4.5** 设  $B \in \mathbb{R}^{n \times n}$ , 则  $\lim_{k \rightarrow \infty} B^k = 0$  的充要条件是存在某个矩阵范数  $\|\cdot\|$ , 使得  $\|B\| < 1$ .

### 4.1.2 基于矩阵分裂的迭代法

当直接求解  $Ax = b$  比较困难时, 我们可以求解一个近似线性方程组  $Mx = b$ , 其中  $M$  可以看作是  $A$  在某种意义下的近似.

设  $Mx = b$  的解为  $x^{(1)}$ . 易知它与原方程的解  $x_* = A^{-1}b$  之间的误差满足

$$A(x_* - x^{(1)}) = b - Ax^{(1)}.$$

如果  $x^{(1)}$  已经满足精度要求, 即非常接近真解  $x_*$ , 则可以停止计算, 否则需要修正.

设  $\Delta x \triangleq x_* - x^{(1)}$ , 则  $\Delta x$  满足方程

$$A\Delta x = b - Ax^{(1)}.$$

但由于直接求解该方程比较困难, 因此我们还是通过求解近似方程

$$M\Delta x = b - Ax^{(1)},$$

得到一个近似的修正量  $\tilde{\Delta}x$ . 于是修正后的近似解为

$$x^{(2)} = x^{(1)} + \tilde{\Delta}x = x^{(1)} + M^{-1}(b - Ax^{(1)}).$$

如果  $x^{(2)}$  已经满足精度要求, 则停止计算, 否则继续按以上的方式进行修正.

不断重复以上步骤, 于是, 我们就得到一个向量序列

$$x^{(1)}, x^{(2)}, \dots, x^{(k)}, \dots$$

它们都是真解  $x_*$  的近似值, 且满足下面的递推关系

$$x^{(k+1)} = x^{(k)} + M^{-1}(b - Ax^{(k)}), \quad k = 1, 2, \dots$$

这就构成了一个迭代方法. 由于每次迭代的格式是一样的, 因此称为 **定常迭代**.

#### 好的定常迭代需要考虑的两个方面

- (1) 以  $M$  为系数矩阵的线性方程组必须比原线性方程组更容易求解;
- (2)  $M$  应该是  $A$  的一个很好的近似, 且迭代序列  $\{x_k\}$  收敛.

目前一类比较常用的定常迭代方法是基于矩阵分裂的迭代方法, 如 Jacobi 方法, Gauss-Seidel (G-S) 方法, 超松弛 (SOR, Successive Over-Relaxation) 方法等等. 下面给出矩阵分裂的定义.

**定义 4.3 (矩阵分裂 Matrix Splitting)** 设  $A \in \mathbb{R}^{n \times n}$  非奇异, 我们称

$$A = M - N \tag{4.1}$$

为  $A$  的一个**矩阵分裂**, 其中  $M$  非奇异.

给定一个矩阵分裂 (4.1), 则原方程组  $Ax = b$  就等价于  $Mx = Nx + b$ . 于是我们就可以构造出以下的迭代格式

$$Mx^{(k+1)} = Nx^{(k)} + b, \quad k = 0, 1, \dots, \tag{4.2}$$

或

$$x^{(k+1)} = M^{-1}Nx^{(k)} + M^{-1}b \triangleq Bx^{(k)} + f, \quad k = 0, 1, \dots, \tag{4.3}$$

其中  $B \triangleq M^{-1}N$  称为**迭代矩阵**. 这就是基于矩阵分裂 (4.1) 的迭代方法. 易知, 选取不同的  $M$ , 就可以构造出不同的迭代方法.

### 4.1.3 迭代法的收敛性

**定义 4.4** 对任意初始向量  $x^{(0)}$ , 设  $\{x^{(k)}\}$  是由迭代方法 4.3 生成的向量序列, 如果  $\lim_{k \rightarrow \infty} x^{(k)}$  存在, 则称迭代方法 4.3 **收敛**, 否则就称为 **发散**.



**引理 4.6** 设迭代序列  $\{x^{(k)}\}$  收敛, 且  $\lim_{k \rightarrow \infty} x^{(k)} = x_*$ , 则  $x_*$  一定是原方程组的真解.

**证明.** 对迭代格式 4.3 两边取极限可得

$$x_* = \lim_{k \rightarrow \infty} x^{(k+1)} = \lim_{k \rightarrow \infty} (M^{-1}Nx^{(k)} + M^{-1}b) = M^{-1}Nx_* + M^{-1}b.$$

整理后可得  $(M - N)x_* = b$ , 即  $Ax_* = b$ , 结论成立.  $\square$

下面给出迭代方法 4.3 的基本收敛性定理.

**定理 4.7** 对任意初始向量  $x^{(0)}$ , 迭代方法 4.3 收敛的充要条件是  $\rho(B) < 1$ .

**证明.** 先证明**必要性** (“ $\Rightarrow$ ”). 对任意向量  $\tilde{x} \in \mathbb{R}^n$ , 令  $x^{(0)} = \tilde{x} - x_*$ , 则

$$x^{(k)} - x_* = (Bx^{(k-1)} + f) - (Bx_* + f) = B(x^{(k-1)} - x_*) = \cdots = B^k(x^{(0)} - x_*) = B^k\tilde{x}.$$

由迭代方法 4.3 的收敛性可知  $B^k\tilde{x} = x^{(k)} - x_* \rightarrow 0$  ( $k \rightarrow \infty$ ). 根据定理 4.2,  $\lim_{k \rightarrow \infty} B^k = 0$ . 再由定理 4.4 可知  $\rho(B) < 1$ .


下面证明**充分性** (“ $\Leftarrow$ ”). 根据条件  $\rho(B) < 1$ , 可得  $\lim_{k \rightarrow \infty} B^k = 0$ . 所以对任意  $x^{(0)} \in \mathbb{R}^n$ , 当  $k \rightarrow \infty$  时, 有


$$x^{(k)} - x_* = B^k(x^{(0)} - x_*) \rightarrow 0.$$

故  $\lim_{k \rightarrow \infty} x^{(k)} = x_*$ , 即迭代方法 4.3 收敛.  $\square$

由于对任意算子范数都有  $\rho(B) < \|B\|$ , 因此我们可以立即得到下面的结论.

**定理 4.8** 若存在算子范数使得  $\|B\| < 1$ , 则迭代方法 4.3 收敛

 由于计算  $\rho(B)$  通常比较复杂, 而  $\|B\|_1, \|B\|_\infty$  相对比较容易计算, 因此在判别迭代方法收敛性时, 可以先验算一下迭代矩阵的 1-范数或  $\infty$ -范数是否小于 1.

 定理 4.8 是充分条件, 但不是必要条件, 因此判断一个迭代方法不收敛仍然需要使用定理 4.7.

**例 4.2** 讨论迭代方法  $x^{(k+1)} = Bx^{(k)} + f$  的收敛性, 其中  $B = \begin{bmatrix} 0.9 & 0 \\ 0.3 & 0.8 \end{bmatrix}$ .

**解.** 由于  $B$  是下三角矩阵, 因此其特征值分别为  $\lambda_1 = 0.9, \lambda_2 = 0.8$ . 所以  $\rho(B) = 0.9 < 1$ . 故迭代方法收敛.  $\square$

**定理 4.9** 若存在算子范数使得  $q \triangleq \|B\| < 1$ , 则

- (1)  $\|x^{(k)} - x_*\| \leq q^k \|x^{(0)} - x_*\|$ ;
- (2)  $\|x^{(k)} - x_*\| \leq \frac{q}{1-q} \|x^{(k)} - x^{(k-1)}\|$ ;



$$(3) \|x^{(k)} - x_*\| \leq \frac{q^k}{1-q} \|x^{(1)} - x^{(0)}\|.$$

证明.

(1) 由  $x^{(k+1)} = Bx^{(k)} + g$  和  $x_* = Bx_* + g$  可得

$$x^{(k+1)} - x^{(k)} = B(x^{(k)} - x^{(k-1)}) \quad \text{和} \quad x^{(k+1)} - x_* = B(x^{(k)} - x_*).$$

因此有

$$\|x^{(k+1)} - x^{(k)}\| \leq q \|x^{(k)} - x^{(k-1)}\|, \quad (4.4)$$

和

$$\|x^{(k+1)} - x_*\| \leq q \|x^{(k)} - x_*\|. \quad (4.5)$$

反复利用结论 (4.5) 即可得  $\|x^{(k)} - x_*\| \leq q^k \|x^{(0)} - x_*\|$ .

(2) 由 (4.5) 可得

$$\begin{aligned} \|x^{(k+1)} - x^{(k)}\| &= \|(x^{(k+1)} - x_*) - (x^{(k)} - x_*)\| \\ &\geq \|x^{(k)} - x_*\| - \|x^{(k+1)} - x_*\| \\ &\geq (1-q) \|x^{(k)} - x_*\|. \end{aligned}$$

结合 (4.4) 可得

$$\|x^{(k)} - x_*\| \leq \frac{1}{1-q} \|x^{(k+1)} - x^{(k)}\| \leq \frac{q}{1-q} \|x^{(k)} - x^{(k-1)}\|.$$

(3) 反复利用 (4.4) 即可得

$$\|x^{(k+1)} - x^{(k)}\| \leq q^k \|x^{(1)} - x^{(0)}\|.$$

所以

$$\|x^{(k)} - x_*\| \leq \frac{1}{1-q} \|x^{(k+1)} - x^{(k)}\| \leq \frac{q^k}{1-q} \|x^{(1)} - x^{(0)}\|.$$

□

#### 4.1.4 迭代方法的收敛速度

考虑迭代方法 4.3, 第  $k$  步的误差为

$$\varepsilon^{(k)} \triangleq x^{(k)} - x_* = B^k(x^{(0)} - x_*) = B^k \varepsilon^{(0)}.$$

所以

$$\frac{|\varepsilon^{(k)}|}{|\varepsilon^{(0)}|} \leq \|B^k\|.$$

因此平均每次迭代后误差的压缩率为  $\|B^k\|^{1/k}$ .

**定义 4.5** 迭代方法 4.3 的平均收敛速度定义为

$$R_k(B) \triangleq -\ln \|B^k\|^{\frac{1}{k}},$$






**渐进收敛速度** 定义为

$$R(B) \triangleq \lim_{k \rightarrow \infty} R_k(B) = -\ln \rho(B).$$

 如果  $0 < \rho(B) < 1$ , 则迭代方法 (4.2) 线性收敛.

 一般来说,  $\rho(B)$  越小, 迭代方法 4.3 的收敛速度越快.

如果事先给定一个精度要求, 比如要求相对误差满足

$$\frac{\|x^{(k)} - x_*\|}{\|x^{(0)} - x_*\|} < \varepsilon,$$

则可根据下面的公式估计所需迭代步数  $k$ :

$$\|B^k\| < \varepsilon \implies \ln \|B^k\|^{1/k} \leq \frac{1}{k} \ln(\varepsilon) \implies k \geq \frac{-\ln(\varepsilon)}{-\ln \|B^k\|^{1/k}} \approx \frac{-\ln(\varepsilon)}{R(B)}.$$

## 4.2 Jacobi, Gauss-Seidel 和 SOR

### 4.2.1 Jacobi 迭代方法

将矩阵  $A$  分裂为

$$A = D - L - U,$$

其中  $D$  为  $A$  的对角线部分,  $-L$  和  $-U$  分别为  $A$  的严格下三角和严格上三角部分.

在矩阵分裂  $A = M - N$  中取  $M = D, N = L + U$ , 则可得 **Jacobi 迭代** 方法:

$$x^{(k+1)} = D^{-1}(L + U)x^{(k)} + D^{-1}b, \quad k = 0, 1, 2, \dots \quad (4.6)$$

对应的迭代矩阵为

$$J = D^{-1}(L + U).$$

写成分量形式即为

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left( b_i - \sum_{j=1, j \neq i}^n a_{ij} x_j^{(k)} \right), \quad i = 1, 2, \dots, n.$$

由于 Jacobi 迭代中  $x_i^{(k+1)}$  的更新顺序与  $i$  无关, 即可以按顺序  $i = 1, 2, \dots, n$  计算, 也可以按顺序  $i = n, n-1, \dots, 2, 1$  计算, 或者乱序计算. 因此 Jacobi 迭代非常适合并行计算.

#### 算法 4.1. Jacobi 迭代

```

1: Given an initial guess  $x^{(0)}$ 
2: while not converge do    % 停机准则
3:   for  $i = 1$  to  $n$  do
4:      $x_i^{(k+1)} = \left( b_i - \sum_{j=1, j \neq i}^n a_{ij} x_j^{(k)} \right) / a_{ii}$ 
5:   end for
6: end while

```

在编程实现该算法时, “停机准则” 一般是要求相对残量满足一定的精度, 即

$$\frac{\|b - Ax^{(k)}\|}{\|b - Ax^{(0)}\|} < \text{tol},$$

其中  $\text{tol}$  是一个事前给定的精度要求, 如  $10^{-6}$  或  $10^{-8}$ , 等等.

我们有时也将 Jacobi 迭代格式写为

$$x^{(k+1)} = x^{(k)} + D^{-1}(b - Ax^{(k)}) = x^{(k)} + D^{-1}r_k, \quad k = 0, 1, 2, \dots,$$

其中  $r_k \triangleq b - Ax^{(k)}$  是  $k$  次迭代后的残量. 这表明,  $x^{(k+1)}$  是通过对  $x^{(k)}$  做一个修正得到的.



### 4.2.2 Gauss-Seidel 迭代方法

在分裂  $A = M - N$  中取  $M = D - L$ ,  $N = U$ , 即可得 **Gauss-Seidel (G-S) 迭代** 方法:

$$x^{(k+1)} = (D - L)^{-1} U x^{(k)} + (D - L)^{-1} b, \quad k = 0, 1, 2, \dots \quad (4.7)$$

对应的迭代矩阵为

$$G = (D - L)^{-1} U.$$

将 G-S 迭代改写为

$$Dx^{(k+1)} = Lx^{(k+1)} + Ux^{(k)} + b,$$

即可得分量形式


$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left( b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i+1}^n a_{ij} x_j^{(k)} \right), \quad i = 1, 2, \dots, n.$$

#### 算法 4.2. Gauss-Seidel 迭代

```

1: Given an initial guess  $x^{(0)}$ 
2: while not converge do
3:   for  $i = 1$  to  $n$  do
4:      $x_i^{(k+1)} = \frac{1}{a_{ii}} \left( b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i+1}^n a_{ij} x_j^{(k)} \right)$ 
5:   end for
6: end while

```

 G-S 方法的主要优点是充分利用了已经获得的最新数据.

### 4.2.3 SOR 迭代方法

在 G-S 方法的基础上, 我们可以通过引入一个松弛参数  $\omega$  来加快收敛速度. 这就是 SOR (Successive Overrelaxation) 方法. 该方法的基本思想是将 G-S 方法中的第  $k+1$  步近似解与第  $k$  步近似解做一个加权平均, 从而给出一个新的近似解, 即

$$x^{(k+1)} = (1 - \omega)x^{(k)} + \omega D^{-1} (Lx^{(k+1)} + Ux^{(k)} + b). \quad (4.8)$$

整理后即得

$$x^{(k+1)} = (D - \omega L)^{-1} ((1 - \omega)D + \omega U) x^{(k)} + \omega(D - \omega L)^{-1} b, \quad (4.9)$$

其中  $\omega$  称为**松弛参数**.

- 当  $\omega = 1$  时, SOR 即为 G-S 方法,
- 当  $\omega < 1$  时, 称为**低松弛**方法,

- 当  $\omega > 1$  时, 称为**超松弛**方法.

在大多数情况下, 当  $\omega > 1$  时会取得比较好的收敛效果.

✎ SOR 方法曾经在很长一段时间内是科学计算中求解线性方程组的首选方法.

SOR 的迭代矩阵为

$$S_\omega = (D - \omega L)^{-1} ((1 - \omega)D + \omega U),$$

对应的矩阵分裂为

$$M = \frac{1}{\omega}D - L, \quad N = \frac{1 - \omega}{\omega}D + U.$$

由 (4.8) 可知 SOR 迭代的分量形式为

$$\begin{aligned} x_i^{(k+1)} &= (1 - \omega)x_i^{(k)} + \frac{\omega}{a_{ii}} \left( b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(k+1)} - \sum_{j=i+1}^n a_{ij}x_j^{(k)} \right) \\ &= x_i^{(k)} + \frac{\omega}{a_{ii}} \left( b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(k+1)} - \sum_{j=i}^n a_{ij}x_j^{(k)} \right) \end{aligned}$$

#### 算法 4.3. 求解线性方程组的 SOR 迭代方法

```

1: Given an initial guess  $x^{(0)}$  and parameter  $\omega$ 
2: while not converge do
3:   for  $i = 1$  to  $n$  do
4:      $x_i^{(k+1)} = (1 - \omega)x_i^{(k)} + \frac{\omega}{a_{ii}} \left( b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(k+1)} - \sum_{j=i+1}^n a_{ij}x_j^{(k)} \right)$ 
5:   end for
6: end while

```

✎ SOR 方法最大的优点是引入了松弛参数  $\omega$ : 通过选取适当的  $\omega$  就可以大大提高方法的收敛速度.

✎ 如何确定 SOR 的最优松弛因子是一件非常困难的事!

**例 4.3** 分别用 Jacobi, G-S 和 SOR( $\omega = 1.1$ ) 迭代方法求解线性方程组  $Ax = b$ , 其中

$$A = \begin{bmatrix} 2 & -1 & 0 \\ -1 & 3 & -1 \\ 0 & -1 & 2 \end{bmatrix}, \quad b = \begin{bmatrix} 1 \\ 8 \\ -5 \end{bmatrix}.$$

初始向量设为  $x^{(0)} = [0, 0, 0]^T$ , 迭代过程中保留小数点后四位. (Iter\_Jacobi\_GS\_SOR\_01.m)



解. Jacobi 迭代方法: 迭代格式为

$$\begin{cases} x_1^{(k+1)} = \frac{1}{2}(1 + x_2^{(k)}) \\ x_2^{(k+1)} = \frac{1}{3}(8 + x_1^{(k)} + x_3^{(k)}) \\ x_3^{(k+1)} = \frac{1}{2}(-5 + x_2^{(k)}) \end{cases}$$

直接计算可得

$$x^{(1)} = [0.5000, 2.6667, -2.5000]^T, \dots, x^{(21)} = [2.0000, 3.0000, -1.0000]^T.$$

G-S 迭代方法: 迭代格式为

$$\begin{cases} x_1^{(k+1)} = \frac{1}{2}(1 + x_2^{(k)}) \\ x_2^{(k+1)} = \frac{1}{3}(8 + x_1^{(k+1)} + x_3^{(k)}) \\ x_3^{(k+1)} = \frac{1}{2}(-5 + x_2^{(k+1)}) \end{cases}$$

直接计算可得

$$x^{(1)} = [0.5000, 2.8333, -1.0833]^T, \dots, x^{(9)} = [2.0000, 3.0000, -1.0000]^T.$$

SOR 迭代方法: 迭代格式为

$$\begin{cases} x_1^{(k+1)} = x_1^{(k)} + \omega \frac{1}{2}(1 - 2x_1^{(k)} + x_2^{(k)}) \\ x_2^{(k+1)} = x_2^{(k)} + \omega \frac{1}{3}(8 + x_1^{(k+1)} - 3x_2^{(k)} + x_3^{(k)}) \\ x_3^{(k+1)} = x_3^{(k)} + \omega \frac{1}{2}(-5 + x_2^{(k+1)} - 2x_3^{(k)}) \end{cases}$$

直接计算可得

$$x^{(1)} = [0.5500, 3.1350, -1.0257]^T, \dots, x^{(7)} = [2.0000, 3.0000, -1.0000]^T.$$

□

例 4.4 编程实践: 分别用 Jacobi, G-S 和 SOR( $\omega = 1.5$ ) 迭代方法求解线性方程组  $Ax = b$ , 其中

$$A = \begin{bmatrix} 2 & -1 & & \\ -1 & \ddots & \ddots & \\ & \ddots & \ddots & -1 \\ & & -1 & 2 \end{bmatrix}, \quad b = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}.$$

初始向量设为  $x^{(0)} = [0, 0, 0]^T$ .

(Iter\_Jacobi\_GS\_SOR\_02.m)

### 4.3 收敛性分析

根据迭代方法基本收敛性定理 4.7 和定理 4.8, 我们可以直接得到下面的结论:

- Jacobi 迭代收敛的 **充要** 条件:  $\rho(J) < 1$
- G-S 迭代收敛的 **充要** 条件:  $\rho(G) < 1$
- SOR 迭代收敛的 **充要** 条件:  $\rho(S_\omega) < 1$
- Jacobi 迭代收敛的 **充分** 条件:  $\|J\| < 1$
- G-S 迭代收敛的充分条件 **充分** 条件:  $\|G\| < 1$
- SOR 迭代收敛的充分条件 **充分** 条件:  $\|S_\omega\| < 1$

#### 4.3.1 不可约与对角占优

**定义 4.6 (不可约)** 设  $A \in \mathbb{R}^{n \times n}$ , 如果存在置换矩阵  $P$ , 使得  $PAP^T$  为块上三角矩阵, 即

$$PAP^T = \begin{bmatrix} A_{11} & A_{12} \\ 0 & A_{22} \end{bmatrix},$$

其中  $A_{11} \in \mathbb{R}^{k \times k}$  ( $1 \leq k < n$ ), 则称  $A$  为**可约矩阵**, 否则称为**不可约矩阵**.

**例 4.5** 直接验证可知下面两个结论成立:

- (1) 设  $A \in \mathbb{R}^{n \times n}$ , 如果  $A$  的所有元素都非零, 则  $A$  不可约.
- (2) 如果  $A \in \mathbb{R}^{n \times n}$  可约且  $n \geq 2$ , 则  $A$  的零元素个数  $\geq n - 1$ .

**引理 4.10** 设  $A \in \mathbb{R}^{n \times n}$  是三对角矩阵, 且三条对角线上的元素都非零, 则  $A$  不可约.

(证明可参见相关资料)

**思考**

设  $A \in \mathbb{R}^{n \times n}$  是三对角矩阵, 且上, 下次对角线元素均非零, 则  $A$  是否不可约?

**定义 4.7 (对角占优)** 设  $A \in \mathbb{R}^{n \times n}$ , 若

$$|a_{ii}| \geq \sum_{j \neq i} |a_{ij}|$$

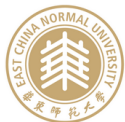
对所有  $i = 1, 2, \dots, n$  都成立, 且至少有一个不等式严格成立, 则称  $A$  为**弱行对角占优**, 简称**行对角占优**. 若对所有  $i = 1, 2, \dots, n$  不等式都严格成立, 则称  $A$  是**严格行对角占优**, 简称**严格对角占优**.

类似地, 可以定义**弱列对角占优**和**严格列对角占优**.

**定理 4.11** 若  $A \in \mathbb{R}^{n \times n}$  是严格对角占优矩阵, 则  $A$  非奇异.

**证明.** 反证法. 假设  $A$  奇异, 即  $Ax = 0$  存在非零解, 不妨设为  $x = [x_1, x_2, \dots, x_n]^T$ . 不失一般性, 设下标  $k$  满足  $\|x\|_\infty = |x_k|$ , 则  $|x_k| > 0$ . 考察  $Ax = 0$  的第  $k$  个方程:

$$a_{k1}x_1 + a_{k2}x_2 + \cdots + a_{kn}x_n = 0.$$



可得

$$|a_{kk}| = \frac{1}{|x_k|} \left| \sum_{j=1, j \neq i}^n a_{kj} x_j \right| \leq \sum_{j=1, j \neq i}^n |a_{kj}| \cdot \frac{|x_j|}{|x_k|} \leq \sum_{j=1, j \neq i}^n |a_{kj}|,$$

这与  $A$  严格对角占优矛盾. 所以  $A$  非奇异.  $\square$

**定理 4.12** 设  $A \in \mathbb{R}^{n \times n}$  是不可约的弱对角占优矩阵, 则  $A$  非奇异.

(证明可参见相关资料)

### 4.3.2 Jacobi 和 G-S 的收敛性

**定理 4.13** 设  $A \in \mathbb{R}^{n \times n}$ , 若  $A$  严格对角占优, 则 Jacobi 和 G-S 迭代方法都收敛.

**证明.** 由于  $A$  严格行对角占优, 故  $\sum_{j \neq i} |a_{ij}|/|a_{ii}| < 1$ . 所以

$$\|J\|_{\infty} = \|D^{-1}(L+U)\|_{\infty} = \max_{1 \leq i \leq n} \sum_{j \neq i} \frac{|a_{ij}|}{|a_{ii}|} < 1.$$

故 Jacobi 迭代方法收敛.

设  $\lambda$  是 G-S 迭代矩阵  $G$  的任意一个特征值, 即  $\det(\lambda I - G) = 0$ . 于是

$$\det(\lambda(D-L) - U) = \det((D-L)^{-1}) \cdot \det(\lambda I - G) = 0.$$

若  $|\lambda| \geq 1$ , 则有

$$|\lambda a_{ii}| > \sum_{j=1, j \neq i}^n |\lambda| \cdot |a_{ij}| \geq \sum_{j=1}^{i-1} |\lambda a_{ij}| + \sum_{j=i+1}^n |a_{ij}|, \quad i = 1, 2, \dots, n.$$

因此  $\lambda(D-L) - U$  是严格对角占优矩阵, 所以非奇异, 与  $\det(\lambda(D-L) - U) = 0$  矛盾. 因此  $|\lambda| < 1$ . 故  $\rho(G) < 1$ , 即 G-S 迭代方法收敛.  $\square$

**定理 4.14** 设  $A \in \mathbb{R}^{n \times n}$ , 若  $A$  是不可约弱对角占优, 则 Jacobi 方法和 G-S 方法都收敛.

(证明可参见相关资料)

**定理 4.15** 设  $A \in \mathbb{R}^{n \times n}$  对称且  $D$  正定, 则 Jacobi 收敛的充要条件是  $2D - A$  正定.

(证明可参见相关资料)

**定理 4.16** 设  $A \in \mathbb{R}^{n \times n}$  对称且  $D$  正定, 则 G-S 收敛的充要条件是  $A$  正定.

(证明可参见相关资料)

### 4.3.3 SOR 的收敛性

我们首先给出 SOR 迭代收敛的一个必要条件.

**定理 4.17** 若 SOR 迭代方法收敛, 则  $0 < \omega < 2$ .

**证明.** SOR 的迭代矩阵为

$$S_\omega = (D - \omega L)^{-1}((1 - \omega)D + \omega U) = (I - \omega \tilde{L})^{-1}((1 - \omega)I + \omega \tilde{U}).$$

所以  $S_\omega$  的行列式为

$$\begin{aligned} \det(S_\omega) &= \det(I - \omega \tilde{L})^{-1} \cdot \det((1 - \omega)I + \omega \tilde{U}) \\ &= (\det(I - \omega \tilde{L}))^{-1} \cdot (1 - \omega)^n \\ &= (1 - \omega)^n. \end{aligned}$$

设  $S_\omega$  的特征为  $\lambda_1, \lambda_2, \dots, \lambda_n$ , 则

$$\lambda_1 \lambda_2 \cdots \lambda_n = \det(S_\omega) = (1 - \omega)^n,$$

故至少有一个特征值的绝对值不小于  $|1 - \omega|$ , 因此  $\rho(S_\omega) \geq |1 - \omega|$ .

若 SOR 收敛, 则  $\rho(S_\omega) < 1$ , 所以  $|1 - \omega| < 1$ , 即  $0 < \omega < 2$ . □

当  $A$  对角占优时, 我们有下面的结论.

**定理 4.18** 设  $A \in \mathbb{R}^{n \times n}$ , 若  $A$  严格对角占优 (或不可约弱对角占优) 且  $0 < \omega \leq 1$ , 则 SOR 收敛.

(证明可参见相关资料)

如果  $A$  是对称正定矩阵, 则有下面的收敛性定理.

**定理 4.19** 设  $A \in \mathbb{R}^{n \times n}$  对称正定, 则 SOR 迭代方法收敛的充要条件是  $0 < \omega < 2$ .

(证明可参见相关资料)

**例 4.6** 考虑线性方程组  $Ax = b$ , 其中  $A = \begin{bmatrix} 1 & a & a \\ a & 1 & a \\ a & a & 1 \end{bmatrix}$ . 试给出 Jacobi, G-S 和 SOR 收敛的充要条件.

(板书)





## 4.4 共轭梯度法

### 4.4.1 最速下降法

见课堂板书 (*To be continued ...*)

### 4.4.2 共轭梯度法

见课堂板书 (*To be continued ...*)

## 4.5 课后练习

练习 4.1 已知线性方程组  $\begin{bmatrix} 3 & 1 \\ 1 & 3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 3 \\ -1 \end{bmatrix}$ , 迭代方法

$$x^{(k+1)} = x^{(k)} + \alpha(Ax^{(k)} - b), \quad k = 0, 1, 2, \dots$$

试问:  $\alpha$  取何值时迭代收敛? 何时收敛最快?

练习 4.2 设线性方程组  $Ax = b$ , 其中  $A$  对称正定且  $0 < \alpha \leq \lambda(A) \leq \beta$ , 迭代方法

$$x^{(k+1)} = x^{(k)} + \omega(b - Ax^{(k)}), \quad k = 0, 1, 2, \dots$$

试证明: 当  $0 < \omega < \frac{2}{\beta}$  时, 迭代方法收敛.

练习 4.3 已知线性方程组

$$\begin{cases} 5x_1 + 2x_2 + x_3 = 3, \\ -2x_1 + 4x_2 + 2x_3 = 7, \\ 3x_1 - 5x_2 + 8x_3 = 2. \end{cases}$$

考察 Jacobi 方法和 Gauss-Seidel 方法的收敛性.

练习 4.4 已知线性方程组

$$(1) \begin{cases} x_1 + 0.4x_2 + 0.5x_3 = 2, \\ 0.4x_1 + x_2 + 0.8x_3 = 1, \\ 0.5x_1 + 0.8x_2 + x_3 = 3. \end{cases} \quad (2) \begin{cases} 2x_1 + 4x_2 - 3x_3 = 3, \\ x_1 + 2x_2 + x_3 = 1, \\ 2x_1 + 2x_2 + 2x_3 = 2. \end{cases}$$

考察 Jacobi 方法和 Gauss-Seidel 方法的收敛性.

练习 4.5 设线性方程组

$$\begin{cases} a_{11}x_1 + a_{12}x_2 = b_1, \\ a_{21}x_1 + a_{22}x_2 = b_2, \end{cases}$$

其中  $a_{11}a_{22} \neq 0$ . 证明: 求解该线性方程组的 Jacobi 迭代和 G-S 迭代具有相同的收敛性, 并求两种方法的收敛速度之比.

练习 4.6 已知线性方程组  $Ax = f$ , 其中

$$A = \begin{bmatrix} 8 & a & 0 \\ b & 8 & b \\ 0 & a & 4 \end{bmatrix}$$

非奇异, 试给出 Jacobi 方法和 Gauss-Seidel 方法收敛的充要条件.

练习 4.7 已知线性方程组:

$$\begin{bmatrix} 5 & 2 & 2 \\ 2 & 5 & 4 \\ 2 & 4 & 5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 2 \\ 6 \\ 7 \end{bmatrix}.$$

写出 Jacobi, Gauss-Seidel 和 SOR( $\omega = 1.5$ ) 方法的分量格式, 并判断这三个方法的收敛性.



# 5

## 矩阵特征值计算

设  $A \in \mathbb{R}^{n \times n}$  是一个非对称的稠密矩阵, 本讲主要讨论如何计算  $A$  的全部特征值和特征向量. 记  $A$  的特征值为  $\lambda_1, \lambda_2, \dots, \lambda_n$ . 本讲中我们总是假定

$$|\lambda_1| \geq |\lambda_2| \geq \dots \geq |\lambda_n| \geq 0,$$

即  $A$  的特征值按绝对值（模）降序排列.

本讲主要介绍以下方法:

- 非对称矩阵特征值计算方法
  - 幂迭代算法
  - 位移策略与反迭代技巧
  - QR 算法与实用的 QR 算法
- 对称矩阵特征值计算方法
  - Jacobi 迭代
  - 分而治之法
  - 对分法

### 关于特征值计算的相关参考资料

- J. H. Wilkinson, The Algebraic Eigenvalue Problem, 1965 [55]
- B. N. Parlett, The Symmetric Eigenvalue Problem, 2nd Eds., 1998 [41]
- G. W. Stewart, Matrix Algorithms, Vol II: Eigensystems, 2001 [48]
- S. Börm and C. Mehl, Numerical Methods for Eigenvalue Problems, 2012 [6]
- G. H. Golub and C. F. Van Loan, Matrix Computations, 2013 [21]
- Y. Saad, Numerical Methods for Large Eigenvalue Problems, 2011 [46]
- Z. Bai, et al, Templates for the Solution of Algebraic Eigenvalue Problems: A Practical Guide, 2000 [2]
- P. Arbenz, Course: Numerical Methods for Solving Large Scale Eigenvalue Problems, 2018.

## 5.1 非对称特征值问题

本讲介绍一般（实）矩阵的特征值计算方法.

### 5.1.1 幂迭代方法

**幂迭代方法** 是计算特征值和特征向量的一种简单易用的算法. 幂迭代虽然简单, 但它却建立了计算特征值和特征向量的算法的一个基本框架.

#### 算法 5.1. 幂迭代算法 (Power Iteration)

```

1: Choose an initial guess  $x^{(0)}$  with  $\|x^{(0)}\|_2 = 1$ 
2: set  $k = 0$ 
3: while not convergence do
4:    $y^{(k+1)} = Ax^{(k)}$ 
5:    $x^{(k+1)} = y^{(k+1)} / \|y^{(k+1)}\|_2$    % 单位化, 防止溢出
6:    $\mu_{k+1} = (x^{(k+1)}, Ax^{(k+1)})$    % 计算内积
7:    $k = k + 1$ 
8: end while
  
```

下面讨论幂迭代的收敛性. 假设

- (1)  $A \in \mathbb{R}^{n \times n}$  是可对角化的, 即  $A = V\Lambda V^{-1}$ , 其中  $\Lambda = \text{diag}(\lambda_1, \lambda_2, \dots, \lambda_n) \in \mathbb{C}^{n \times n}$ ,  $V = [v_1, v_2, \dots, v_n] \in \mathbb{C}^{n \times n}$ , 且  $\|v_i\|_2 = 1, i = 1, 2, \dots, n$ .
- (2) 同时, 我们还假设  $|\lambda_1| > |\lambda_2| \geq |\lambda_3| \geq \dots \geq |\lambda_n|$ .

由于  $V \in \mathbb{C}^{n \times n}$  非奇异, 所以它的列向量构成  $\mathbb{C}^n$  的一组基. 因此迭代初始向量  $x^{(0)}$  可表示为

$$x^{(0)} = \alpha_1 v_1 + \alpha_2 v_2 + \dots + \alpha_n v_n = V[\alpha_1, \alpha_2, \dots, \alpha_n]^T.$$

我们假定  $\alpha_1 \neq 0$ , 即  $x^{(0)}$  不属于  $\text{span}\{v_2, v_3, \dots, v_n\}$  (由于  $x^{(0)}$  是随机选取的, 从概率意义上讲, 这个假设通常是成立的). 于是我们可得

$$A^k x^{(0)} = (V\Lambda V^{-1})^k V \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_n \end{bmatrix} = V \Lambda^k \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_n \end{bmatrix} = V \begin{bmatrix} \alpha_1 \lambda_1^k \\ \alpha_2 \lambda_2^k \\ \vdots \\ \alpha_n \lambda_n^k \end{bmatrix} = \alpha_1 \lambda_1^k V \begin{bmatrix} 1 \\ \frac{\alpha_2}{\alpha_1} \left(\frac{\lambda_2}{\lambda_1}\right)^k \\ \vdots \\ \frac{\alpha_n}{\alpha_1} \left(\frac{\lambda_n}{\lambda_1}\right)^k \end{bmatrix}.$$

又  $|\lambda_i/\lambda_1| < 1, i = 2, 3, \dots, n$ , 所以

$$\lim_{k \rightarrow \infty} \left(\frac{\lambda_i}{\lambda_1}\right)^k = 0, \quad i = 2, 3, \dots, n.$$

故当  $k$  趋向于无穷大时, 向量


$$\left[ 1, \frac{\alpha_2}{\alpha_1} \left(\frac{\lambda_2}{\lambda_1}\right)^k, \dots, \frac{\alpha_n}{\alpha_1} \left(\frac{\lambda_n}{\lambda_1}\right)^k \right]^T, \quad k = 0, 1, 2, \dots$$



收敛到  $e_1 = [1, 0, \dots, 0]^T$ . 所以向量  $x^{(k)} = A^k x^{(0)} / \|A^k x^{(0)}\|_2$  收敛到  $\pm v_1$ , 即  $A$  的对应于 (模) 最大的特征值  $\lambda_1$  的特征向量. 而  $\mu_k = (x^{(k)})^* A x^{(k)}$  则收敛到  $v_1^* A v_1 = \lambda_1$ .

显然, 幂迭代的收敛快慢取决于  $|\lambda_2/\lambda_1|$  的大小,  $|\lambda_2/\lambda_1|$  越小, 收敛越快.

通过上面的分析可知, 幂迭代只能用于计算矩阵的模最大的特征值和其相应的特征向量. 如果  $A$  的模最大的特征值是唯一的, 则称其为 **主特征值**. 当  $|\lambda_2/\lambda_1|$  接近于 1 时, 收敛速度会非常慢. 同时, 如果  $A$  的模最大特征值是一对共轭复数, 则幂迭代就可能就会失效.

 如果需要计算其他特征值, 比如模第二大特征值  $\lambda_2$ , 则可以在模最大特征值  $\lambda_1$  计算出来后, 采用 **收缩 (Deflation)** 技术: 构造酉矩阵  $U$ , 使得

$$U^* A U = \begin{bmatrix} \lambda_1 & A_{12} \\ 0 & A_{22} \end{bmatrix}.$$

然后将幂迭代作用到  $A_{22}$  上, 就可以求出  $\lambda_2$ . 以此类推, 可以依次求出所有特征值 (这里假定特征值互不相同).



思考: 上面的收缩技术中的  $U$  怎么选取?

### 5.1.2 反迭代方法

前面已经提到, 幂迭代算法的收敛速度取决于  $|\lambda_2/\lambda_1|$  的大小. 当它的值接近于 1 时, 收敛速度会非常缓慢. 因此, 为了加快幂迭代算法的收敛速度, 我们希望  $|\lambda_2/\lambda_1|$  的值越小越好.


一个简单易用的方法就是使用 **位移策略**, 即将计算  $A$  的特征值转化为计算  $A - \sigma I$  的特征值, 即对  $A$  做一个移位. 这里  $\sigma$  是一个给定的数, 称  $\sigma$  为 **位移** (shift). 为了使得幂迭代作用到  $A - \sigma I$  时具有更快的收敛速度, 我们要求  $\sigma$  满足下面的两个条件:

- (1)  $\lambda_1 - \sigma$  是  $A - \sigma I$  的模最大的特征值;
- (2)  $\max_{2 \leq i \leq n} \left| \frac{\lambda_i - \sigma}{\lambda_1 - \sigma} \right|$  尽可能地小.

其中第一个条件保证最后所求得特征值是我们所要的, 第二个条件用于加快幂迭代的收敛速度.

显然, 在实际应用中,  $\sigma$  的取值并不是一件容易的事.

**例 5.1** 设  $A = X \Lambda X^{-1}$ , 其中  $\Lambda$  为对角矩阵, 分别用幂迭代方法和带位移的幂迭代方法计算  $A$  的主特征值. (Eig\_Power\_shift.m)

 位移策略在特征值计算中非常重要, 主要是用在反迭代算法和 QR 迭代算法中.

### 反迭代算法

如果我们将幂迭代算法作用在  $A^{-1}$  上, 则可求出  $A$  的模最小的特征值. 事实上, 结合这种思想和位移策略, 我们就可以计算矩阵的任意一个特征值.



**算法 5.2.** 反迭代算法 (Inverse Iteration)

- 1: Choose a scalar  $\sigma$  and an initial vector  $x^{(0)}$  with  $\|x^{(0)}\|_2 = 1$
- 2: set  $k = 0$
- 3: **while** not convergence **do**
- 4:    $y^{(k+1)} = (A - \sigma I)^{-1}x^{(k)}$
- 5:    $x^{(k+1)} = y^{(k+1)} / \|y^{(k+1)}\|_2$
- 6:    $\mu_{k+1} = (x^{(k+1)}, Ax^{(k+1)})$
- 7:    $k = k + 1$
- 8: **end while**

该算法称为**反迭代算法**. 显然, 在反迭代算法中,  $\mu_k$  收敛到距离  $\sigma$  最近的特征值, 而  $x^{(k)}$  则收敛到其对应的特征向量.

设距离  $\sigma$  最近的特征值为  $\lambda_k$ , 则算法的收敛速度取决于

$$\max_{1 \leq i \leq n} \left| \frac{\lambda_k - \sigma}{\lambda_i - \sigma} \right|$$

的大小. 显然,  $\sigma$  约接近于  $\lambda_k$ , 其值越小, 即算法收敛越快. 若  $\sigma \approx \lambda_k$ , 则迭代几步就可以了.

反迭代算法的另一个优点是, 只要选取合适的位移  $\sigma$ , 就可以计算  $A$  的任意一个特征值.

但反迭代算法的缺点也很明显: 每步迭代需要解一个线性方程组  $(A - \sigma I)y^{(k+1)} = x^{(k)}$ , 这就需要对  $A - \sigma I$  做一次 LU 分解. 另外, 与幂迭代一样, 反迭代算法一次只能求一个特征值.

**Rayleigh 商迭代**

在反迭代算法中, 有一个很重要的问题, 就是位移  $\sigma$  的选取.

显然, 选取  $\sigma$  的基本原则是使得其与所求的特征值越靠近越好. 这里需要指出的是, 在每步迭代中, 位移  $\sigma$  可以不一样, 即在迭代过程中可以选取不同的位移  $\sigma$ .

由于在反迭代算法 5.2 中,  $\mu_k$  是收敛到所求的特征值的, 所以我们可以选取  $\mu_k$  作为第  $k$  步的位移. 这时所得到的反迭代算法就称为 **Rayleigh 商迭代** (Rayleigh Quotient Iteration), 简记为 RQI.

**算法 5.3.** Rayleigh 商迭代算法 (Rayleigh Quotient Iteration (RQI))

- 1: Choose an initial vector  $x^{(0)}$  with  $\|x^{(0)}\|_2 = 1$
- 2: set  $k = 0$
- 3: compute  $\sigma = (x^{(0)})^* Ax^{(0)}$
- 4: **while** not converge **do**
- 5:    $y^{(k+1)} = (A - \sigma I)^{-1}x^{(k)}$
- 6:    $x^{(k+1)} = y^{(k+1)} / \|y^{(k+1)}\|_2$
- 7:    $\mu_{k+1} = (x^{(k+1)}, Ax^{(k+1)})$
- 8:    $\sigma = \mu_{k+1}$
- 9:    $k = k + 1$



10: **end while**

一般来说, 如果 Rayleigh 商迭代收敛到  $A$  的一个单特征值, 则至少是二次收敛的, 即具有局部二次收敛性. 如果  $A$  是对称的, 则能达到局部三次收敛.

在 Rayleigh 商迭代中, 由于每次迭代的位移是不同的, 因此每次迭代需要求解一个不同的线性方程组, 这使得运算量大大增加.

**例 5.2** 设  $A = X\Lambda X^{-1}$ , 其中  $\Lambda$  为对角矩阵, 用 Rayleigh 商迭代计算  $A$  的特征值.

(Fig\_Rayleigh.m)

**思考**

幂迭代和反迭代都只能同时计算一个特征对, 如何同时计算多个特征对?

### 5.1.3 QR 迭代方法

**QR 迭代算法**的基本思想是通过不断的正交相似变换, 将  $A$  转化为上三角形式 (或拟上三角形式). 算法形式非常简单, 描述如下:

**算法 5.4.** QR 迭代算法 (QR Iteration)

```
1: Set  $A_1 = A$  and  $k = 1$ 
2: while not convergence do
3:    $A_k = Q_k R_k$     % QR 分解
4:   compute  $A_{k+1} = R_k Q_k$ 
5:    $k = k + 1$ 
6: end while
```

在该算法中, 我们有

$$A_{k+1} = R_k Q_k = (Q_k^T Q_k) R_k Q_k = Q_k^T (Q_k R_k) Q_k = Q_k^T A_k Q_k.$$

由这个递推关系可得

$$A_{k+1} = Q_k^T A_k Q_k = Q_k^T Q_{k-1}^T A_{k-1} Q_{k-1} Q_k = \cdots = Q_k^T Q_{k-1}^T \cdots Q_1^T A Q_1 \cdots Q_{k-1} Q_k.$$

记  $\tilde{Q}_k = Q_1 \cdots Q_{k-1} Q_k = [\tilde{q}_1^{(k)}, \tilde{q}_2^{(k)}, \dots, \tilde{q}_n^{(k)}]$ , 则

$$A_{k+1} = \tilde{Q}_k^T A \tilde{Q}_k, \quad (5.1)$$

即  $A_{k+1}$  与  $A$  正交相似.

### QR 迭代与幂迭代的关系

记  $\tilde{R}_k = R_k R_{k-1} \cdots R_1$ , 则有

$$\tilde{Q}_k \tilde{R}_k = \tilde{Q}_{k-1} (Q_k R_k) \tilde{R}_{k-1} = \tilde{Q}_{k-1} (A_k) \tilde{R}_{k-1} = \tilde{Q}_{k-1} (\tilde{Q}_{k-1}^T A \tilde{Q}_{k-1}) \tilde{R}_{k-1} = A \tilde{Q}_{k-1} \tilde{R}_{k-1},$$



由此递推下去, 即可得

$$\tilde{Q}_k \tilde{R}_k = A^{k-1} \tilde{Q}_1 \tilde{R}_1 = A^{k-1} Q_1 R_1 = A^k. \quad (5.2)$$

故

$$\tilde{Q}_k \tilde{R}_k e_1 = A^k e_1,$$

这说明 QR 迭代与幂迭代有关.

假设  $|\lambda_1| > |\lambda_2| \geq \cdots \geq |\lambda_n|$ , 则当  $k$  充分大时,  $A^k e_1$  收敛到  $A$  的模最大特征值  $\lambda_1$  所对应的特征向量, 故  $\tilde{Q}_k$  的第一列  $\tilde{q}_1^{(k)}$  也收敛到  $\lambda_1$  所对应的特征向量. 因此, 当  $k$  充分大时,  $A \tilde{q}_1^{(k)} \rightarrow \lambda_1 \tilde{q}_1^{(k)}$ , 此时由 (5.1) 可知,  $A_{k+1}$  的第一列

$$A_{k+1}(:, 1) = \tilde{Q}_k^T A \tilde{q}_1^{(k)} \rightarrow \lambda_1 \tilde{Q}_k^T \tilde{q}_1^{(k)} = \lambda_1 e_1,$$

即  $A_{k+1}$  的第一列的第一个元素收敛到  $\lambda_1$ , 而其它元素都趋向于 0, 收敛速度取决于  $|\lambda_2/\lambda_1|$  的大小.

### QR 迭代与反迭代的关系

下面观察  $\tilde{Q}_k$  的最后一列. 由 (5.1) 可知

$$A \tilde{Q}_k = \tilde{Q}_k A_{k+1} = \tilde{Q}_k Q_{k+1} R_{k+1} = \tilde{Q}_{k+1} R_{k+1},$$

所以有

$$\tilde{Q}_{k+1} = A \tilde{Q}_k R_{k+1}^{-1}.$$

由于  $\tilde{Q}_{k+1}$  和  $\tilde{Q}_k$  都是正交矩阵, 上式两边转置后求逆, 可得

$$\tilde{Q}_{k+1} = \left( \tilde{Q}_k^T \right)^{-1} = \left( (R_{k+1}^{-1})^T \tilde{Q}_k^T A^T \right)^{-1} = (A^T)^{-1} \tilde{Q}_k R_{k+1}^T.$$

观察等式两边矩阵的最后一列, 可得

$$\tilde{q}_n^{(k+1)} = c_1 (A^T)^{-1} \tilde{q}_n^{(k)},$$

其中  $c_1$  为某个常数. 依此类推, 可知

$$\tilde{q}_n^{(k+1)} = c (A^T)^{-k} \tilde{q}_n^{(1)},$$

其中  $c$  为某个常数. 假设  $A$  的特征值满足  $|\lambda_1| \geq |\lambda_2| \geq \cdots \geq |\lambda_{n-1}| > |\lambda_n| > 0$ , 则  $\lambda_n^{-1}$  是  $(A^T)^{-1}$  的模最大的特征值. 由幂迭代的收敛性可知,  $\tilde{q}_n^{(k+1)}$  收敛到  $(A^T)^{-1}$  的模最大特征值  $\lambda_n^{-1}$  所对应的特征向量, 即当  $k$  充分大时, 有

$$(A^T)^{-1} \tilde{q}_n^{(k+1)} \rightarrow \lambda_n^{-1} \tilde{q}_n^{(k+1)}.$$

所以

$$A^T \tilde{q}_n^{(k+1)} \rightarrow \lambda_n \tilde{q}_n^{(k+1)}.$$

由 (5.1) 可知,  $A_{k+1}^T$  的最后一列为

$$A_{k+1}^T(:, n) = \tilde{Q}_k^T A^T \tilde{q}_n^{(k)} \rightarrow \lambda_n \tilde{Q}_k^T \tilde{q}_n^{(k)} = \lambda_n e_n,$$

即  $A_{k+1}$  的最后一行的最后一个元素收敛到  $\lambda_n$ , 而其它元素都趋向于 0, 收敛速度取决于  $|\lambda_n/\lambda_{n-1}|$  的大小.





## QR 迭代的收敛性

**定理 5.1** 设  $A = V\Lambda V^{-1} \in \mathbb{R}^{n \times n}$ , 其中  $\Lambda = \text{diag}(\lambda_1, \lambda_2, \dots, \lambda_n)$ , 且  $|\lambda_1| > |\lambda_2| > \dots > |\lambda_n|$ . 若  $V^{-1}$  的所有顺序主子矩阵都非奇异 (即  $V^{-1}$  存在 LU 分解), 则  $A_k$  的对角线以下的元素均收敛到 0.

**证明.** 设  $V = Q_v R_v$  是  $V$  的 QR 分解,  $V^{-1} = L_v U_v$  是  $V^{-1}$  的 LU 分解, 其中  $L_v$  是单位下三角矩阵. 则

$$A^k = V\Lambda^k V^{-1} = Q_v R_v \Lambda^k L_v U_v = Q_v R_v (\Lambda^k L_v \Lambda^{-k}) \Lambda^k U_v.$$

注意到矩阵  $\Lambda^k L_v \Lambda^{-k}$  是一个下三角矩阵, 且其  $(i, j)$  位置上的元素为

$$(\Lambda^k L_v \Lambda^{-k})(i, j) = \begin{cases} 0, & i < j, \\ 1, & i = j, \\ L_v(i, j) \lambda_i^k / \lambda_j^k, & i > j. \end{cases}$$

由于当  $i > j$  时有  $|\lambda_i / \lambda_j| < 1$ , 故当  $k$  充分大时,  $\lambda_i^k / \lambda_j^k$  趋向于 0. 所以我们可以把  $\Lambda^k L_v \Lambda^{-k}$  写成

$$\Lambda^k L_v \Lambda^{-k} = I + E_k,$$

其中  $E_k$  满足  $\lim_{k \rightarrow \infty} E_k = 0$ . 于是

$$A^k = Q_v R_v (I + E_k) \Lambda^k U_v = Q_v (I + R_v E_k R_v^{-1}) R_v \Lambda^k U_v. \quad (5.3)$$

对矩阵  $I + R_v E_k R_v^{-1}$  做 QR 分解:  $I + R_v E_k R_v^{-1} = Q_{E_k} R_{E_k}$ . 由于  $E_k \rightarrow 0$ , 所以  $Q_{E_k} \rightarrow I$ ,  $R_{E_k} \rightarrow I$ . 将其代入 (5.3) 可得

$$A^k = Q_v Q_{E_k} R_{E_k} R_v \Lambda^k U_v = Q_v Q_{E_k} D_k (D_k^{-1} R_{E_k} R_v \Lambda^k U_v), \quad (5.4)$$

其中  $D_k$  是对角矩阵, 其对角线元素的模均为 1, 它使得上三角矩阵  $D_k^{-1} R_{E_k} R_v \Lambda^k U_v$  的对角线元素均为正. 这样, (5.4) 就构成  $A^k$  的 QR 分解. 又由 (5.2) 可知  $A^k = \tilde{Q}_k \tilde{R}_k$ , 根据 QR 分解的唯一性, 我们可得

$$\tilde{Q}_k = Q_v Q_{E_k} D_k, \quad \tilde{R}_k = D_k^{-1} R_{E_k} R_v \Lambda^k U_v.$$

所以由 (5.1) 可知

$$\begin{aligned} A_{k+1} &= \tilde{Q}_k^T A \tilde{Q}_k \\ &= (Q_v Q_{E_k} D_k)^T V \Lambda V^{-1} Q_v Q_{E_k} D_k \\ &= D_k^T Q_{E_k}^T Q_v^T Q_v R_v \Lambda R_v^{-1} Q_v^{-1} Q_v Q_{E_k} D_k \\ &= D_k^T Q_{E_k}^T R_v \Lambda R_v^{-1} Q_{E_k} D_k. \end{aligned}$$

由于  $Q_{E_k} \rightarrow I$ , 所以当  $k \rightarrow \infty$  时,  $A_{k+1}$  收敛到一个上三角矩阵. 收敛速度取决于

$$\max_{1 \leq i < n} \left| \frac{\lambda_{i+1}}{\lambda_i} \right|$$

的大小. □

需要指出的是, 由于  $D_k$  的元素不一定收敛, 故  $A_{k+1}$  对角线以上 (不含对角线) 的元素不一定收敛, 但这不妨碍  $A_{k+1}$  的对角线元素收敛到  $A$  的特征值 (即  $A_{k+1}$  的对角线元素是收敛的).

**例 5.3** QR 迭代算法演示. 设

$$A = X \begin{bmatrix} 9 & & & \\ & 5 & & \\ & & 3 & \\ & & & 1 \end{bmatrix} X^{-1},$$

其中  $X$  是由 MATLAB 随机生成的非奇异矩阵.

在迭代过程中, 对于  $A_k$  的下三角部分中元素, 如果其绝对值小于某个阈值  $tol$ , 则直接将其设为 0, 即

$$a_{ij}^{(k)} = 0 \quad \text{if } i > j \text{ and } |a_{ij}^{(k)}| < tol.$$

这里我们取  $tol = 10^{-6} \max_{1 \leq i, j \leq n} \{|a_{ij}^{(k)}|\}$ .

(Eig\_QR.m)

### 带位移的 QR 迭代

为了加快 QR 迭代的收敛速度, 我们可以采用位移策略和反迭代思想.

**算法 5.5.** 带位移的 QR 迭代算法 (QR Iteration with shift)

- 1: Set  $A_1 = A$  and  $k = 1$
- 2: **while** not convergence **do**
- 3:     Choose a shift  $\sigma_k$
- 4:      $A_k - \sigma_k I = Q_k R_k$      % QR 分解
- 5:     Compute  $A_{k+1} = R_k Q_k + \sigma_k I$
- 6:      $k = k + 1$
- 7: **end while**

与不带位移的 QR 迭代一样, 我们有

$$A_{k+1} = R_k Q_k + \sigma_k I = (Q_k^T Q_k) R_k Q_k + \sigma_k I = Q_k^T (A_k - \sigma_k I) Q_k + \sigma_k I = Q_k^T A_k Q_k$$

所以, 带位移的 QR 算法中所得到的矩阵  $A_k$  仍然与  $A_1 = A$  正交相似.

在带位移的 QR 迭代算法中, 一个很重要的问题就是位移  $\sigma_k$  的选取. 在前面的分析中我们已经知道,  $A_{k+1}(n, n)$  将收敛到  $A$  的模最小的特征值, 且收敛速度取决于模最小特征值与模第二小特征值之间的比值. 显然, 若  $\sigma_k$  就是  $A$  的一个特征值, 则  $A_k - \sigma_k I$  的模最小特征值为 0, 故 QR 算法迭代一步就收敛. 此时

$$A_{k+1} = R_k Q_k + \sigma_k I = \begin{bmatrix} A_{k+1}^{(n-1) \times (n-1)} & * \\ 0 & \sigma_k \end{bmatrix}.$$

如果需要计算  $A$  的其它特征值, 则可对于子矩阵  $A_{k+1}^{(n-1) \times (n-1)}$  使用带位移的 QR 迭代算法.

通常, 如果  $\sigma_k$  与  $A$  的某个特征值非常接近, 则收敛速度通常会很快. 由于  $A_k(n, n)$  收敛到  $A$  的一个特征值, 所以在实际使用中, 一个比较直观的位移选择策略是  $\sigma_k = A_k(n, n)$ . 事实上, 这样的位移选取方法通常会使得 QR 迭代有二次收敛速度.



**例 5.4** 带位移的 QR 迭代算法演示. 所有数据和设置与例 5.3 相同, 在迭代过程中, 取  $\sigma_k = A_k(n, n)$ . 如果  $A_k(n, n)$  已经收敛, 则取  $\sigma_k = A_k(n-1, n-1)$ . (Eig\_QR\_shift.m)

### 5.1.4 带位移的隐式 QR 迭代

QR 迭代算法中需要考虑的另一个重要问题就是运算量: 每一步迭代都需要做一次 QR 分解和矩阵乘积, 运算量为  $O(n^3)$ . 即使每计算一个特征值只需迭代一步, 则计算所有特征值也需要  $O(n^4)$  的运算量. 这是令人无法忍受的. 下面我们就想办法将总运算量从  $O(n^4)$  减小到  $O(n^3)$ .

为了实现这个目标, 我们需要利用 Hessenberg 矩阵. 具体步骤如下: 首先通过相似变化将  $A$  转化成一个上 Hessenberg 矩阵, 然后再对这个 Hessenberg 矩阵实施隐式 QR 迭代. 所谓隐式 QR 迭代, 就是在 QR 迭代中, 我们不需要进行显式的 QR 分解. 这样就可以将 QR 迭代的每一步运算量从  $O(n^3)$  降低到  $O(n^2)$ . 从而将总的运算量降低到  $O(n^3)$ .

#### 上 Hessenberg 矩阵

设  $H = [h_{ij}] \in \mathbb{R}^{n \times n}$ , 若当  $i > j + 1$  时, 有  $h_{ij} = 0$ , 则称  $H$  为上 Hessenberg 矩阵.

**定理 5.2** 设  $A \in \mathbb{R}^{n \times n}$ , 则存在正交矩阵  $Q \in \mathbb{R}^{n \times n}$ , 使得  $Q A Q^T$  是上 Hessenberg 矩阵.

下面我们以一个  $5 \times 5$  的矩阵  $A$  为例, 给出基于 Householder 变换的上 Hessenberg 化过程.

**第一步:** 令  $Q_1 = \text{diag}(I_{1 \times 1}, H_1)$ , 其中  $H_1$  是对应于向量  $A(2:5, 1)$  的 Householder 矩阵. 于是可得

$$Q_1 A = \begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ 0 & * & * & * & * \\ 0 & * & * & * & * \\ 0 & * & * & * & * \end{bmatrix}.$$

由于用  $Q_1^T$  右乘  $Q_1 A$  时, 不会改变  $Q_1 A$  第一列元素的值, 故

$$A_1 \triangleq Q_1 A Q_1^T = \begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ 0 & * & * & * & * \\ 0 & * & * & * & * \\ 0 & * & * & * & * \end{bmatrix}.$$

**第二步:** 令  $Q_2 = \text{diag}(I_{2 \times 2}, H_2)$ , 其中  $H_2$  是对应于向量  $A_1(3:5, 2)$  的 Householder 矩阵, 则用  $Q_2$  左乘  $A_1$  时, 不会改变  $A_1$  的第一列元素的值. 用  $Q_2^T$  右乘  $Q_2 A_1$  时, 不会改变  $Q_2 A_1$  前两列

元素的值. 因此,

$$Q_2 A_1 = \begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ 0 & * & * & * & * \\ 0 & 0 & * & * & * \\ 0 & 0 & * & * & * \end{bmatrix} \quad \text{和} \quad A_2 \triangleq Q_2 A_1 Q_2^T = \begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ 0 & * & * & * & * \\ 0 & 0 & * & * & * \\ 0 & 0 & * & * & * \end{bmatrix}.$$

**第三步:** 令  $Q_3 = \text{diag}(I_{3 \times 3}, H_3)$ , 其中  $H_3$  是对应于向量  $A_2(4:5, 3)$  的 Householder 矩阵, 则有

$$Q_3 A_2 = \begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ 0 & * & * & * & * \\ 0 & 0 & * & * & * \\ 0 & 0 & 0 & * & * \end{bmatrix} \quad \text{和} \quad A_3 \triangleq Q_3 A_2 Q_3^T = \begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ 0 & * & * & * & * \\ 0 & 0 & * & * & * \\ 0 & 0 & 0 & * & * \end{bmatrix}.$$

这时, 我们就将  $A$  转化成成一个上 Hessenberg 矩阵, 即  $Q A Q^T = A_3$  其中  $Q = Q_3 Q_2 Q_1$  是正交矩阵,  $A_3$  是上 Hessenberg 矩阵.

下面是将任意一个矩阵转化成上 Hessenberg 矩阵的算法.

#### 算法 5.6. 上 Hessenberg 化 (Upper Hessenberg Reduction)

- 1: Set  $Q = I$
- 2: **for**  $k = 1$  to  $n - 2$  **do**
- 3:     compute Householder matrix  $H_k = I - \beta_k v_k v_k^T$  with respect to  $A(k+1:n, k)$
- 4:      $A(k+1:n, k:n) = H_k \cdot A(k+1:n, k:n)$   
 $\qquad\qquad\qquad = A(k+1:n, k:n) - \beta_k v_k (v_k^T A(k+1:n, k:n))$
- 5:      $A(1:n, k+1:n) = A(1:n, k+1:n) \cdot H_k^T$   
 $\qquad\qquad\qquad = A(1:n, k+1:n) - \beta_k A(1:n, k+1:n) v_k v_k^T$
- 6:      $Q(k+1:n, :) = H_k \cdot Q(k+1:n, :)$   
 $\qquad\qquad\qquad = Q(k+1:n, :) - \beta_k v_k (v_k^T Q(k+1:n, :))$
- 7: **end for**

在实际计算时, 我们不需要显式地形成 Householder 矩阵  $H_k$ .

上述算法的运算量大约为  $\frac{14}{3}n^3 + \mathcal{O}(n^2)$ . 如果不需要计算特征向量, 则正交矩阵  $Q$  也不用计算, 此时运算量大约为  $\frac{10}{3}n^3 + \mathcal{O}(n^2)$ .

上 Hessenberg 矩阵的一个很重要的性质就是在 QR 迭代中能保持形状不变.

**定理 5.3** 设  $A \in \mathbb{R}^{n \times n}$  是非奇异上 Hessenberg 矩阵, 其 QR 分解为  $A = QR$ , 则  $\tilde{A} \triangleq RQ$  也是上 Hessenberg 矩阵.



**证明.** 设  $A = QR$  是  $A$  的 QR 分解, 则  $Q = AR^{-1}$ . 由于  $R$  是一个上三角矩阵, 所以  $R^{-1}$  也是一个上三角矩阵. 因此  $Q$  的第  $j$  列是  $A$  的前  $j$  列的线性组合. 又  $A$  是上 Hessenberg 矩阵, 所以  $Q$  也是一个上 Hessenberg 矩阵.

相类似地, 我们很容易验证  $RQ$  也是一个上 Hessenberg 矩阵. 所以结论成立.  $\square$

若  $A$  是奇异的, 也可以通过选取适当的  $Q$ , 使得定理 5.3 中的结论成立. 事实上, 由基于 Gram-Schmidt 过程的 QR 分解可知, 如果  $A$  是奇异的上 Hessenberg 矩阵, 则  $Q$  仍可取为上 Hessenberg 矩阵.

由这个性质可知, 如果  $A \in \mathbb{R}^{n \times n}$  是上 Hessenberg 矩阵, 则 QR 迭代中的每一个  $A_k$  都是上 Hessenberg 矩阵. 这样, 在进行 QR 分解时, 运算量可大大降低.

Hessenberg 矩阵还有一个重要性质, 就是在 QR 迭代过程中能保持下次对角线元素非零.

**定理 5.4** 设  $A \in \mathbb{R}^{n \times n}$  是非奇异的上 Hessenberg 矩阵, 且下次对角线元素均非零, 即  $a_{i+1,i} \neq 0$ ,  $i = 1, 2, \dots, n-1$ . 设其 QR 分解为  $A = QR$ , 则  $\tilde{A} \triangleq RQ$  的下次对角线元素也都非零.

(证明留作练习)

易知, 如果上 Hessenberg 矩阵  $A$  存在某个下次对角线元素为零, 则  $A$  一定可约, 此时计算  $A$  的特征值就转化为计算两个更小规模的矩阵的特征值. 因此, 我们只需考虑下次对角线均非零的情形.

需要指出的是, 如果上 Hessenberg 矩阵  $A$  存在某个下次对角线元素为零, 则  $A$  一定可约, 但反之却不成立, 即如果  $A$  是可约的上 Hessenberg 矩阵,  $A$  的下次对角线元素仍可能均非零.

**推论 5.5** 设  $A \in \mathbb{R}^{n \times n}$  是非奇异的上 Hessenberg 矩阵, 且下次对角线元素均非零, 则在带位移的 QR 迭代中, 所有的  $A_k$  的下次对角线元素均非零.

## 隐式 QR 迭代

在 QR 迭代中, 我们需要先做 QR 分解  $A_k = Q_k R_k$ , 然后再计算  $A_{k+1} = R_k Q_k$ . 但事实上, 我们可以将这个过程进行简化, 即在不进行  $A_k$  的 QR 分解的前提下, 直接计算出  $A_{k+1}$ . 这就是**隐式 QR 迭代**.

我们这里考虑不可约的上 Hessenberg 矩阵, 即  $A$  的下次对角线元素都不为 0. 事实上, 若  $A$  是可约的, 则  $A$  就是一个块上三角矩阵, 这时  $A$  的特征值计算问题就转化成计算两个对角块的特征值问题. 隐式 QR 迭代的理论基础就是下面的**隐式 Q 定理**.

**定理 5.6 (Implicit Q Theorem)** 设  $H = Q^T A Q \in \mathbb{R}^{n \times n}$  是一个不可约上 Hessenberg 矩阵, 其中  $Q \in \mathbb{R}^{n \times n}$  是正交矩阵, 则  $Q$  的第 2 至第  $n$  列均由  $Q$  的第一列所唯一确定 (可相差一个符号).

**证明.** 设  $H = Q^T A Q$  和  $G = V^T A V$  都是不可约上 Hessenberg 矩阵, 其中  $Q = [q_1, q_2, \dots, q_n]$ ,  $V = [v_1, v_2, \dots, v_n]$  都是正交矩阵, 且  $q_1 = v_1$ . 下面我们只需证明  $q_i = v_i$  或  $q_i = -v_i$ ,  $i = 2, 3, \dots, n$ .

即证明

$$W \triangleq V^T Q = \text{diag}(1, \pm 1, \dots, \pm 1).$$

记  $W = [w_1, w_2, \dots, w_n]$ ,  $H = [h_{ij}]$ , 则有

$$GW = GV^T Q = (V^T AV)V^T Q = V^T AQ = V^T Q(Q^T AQ) = V^T QH = WH,$$

即

$$Gw_i = \sum_{j=1}^{i+1} h_{ji} w_j, \quad i = 1, 2, \dots, n-1.$$

所以

$$h_{i+1,i} w_{i+1} = Gw_i - \sum_{j=1}^i h_{ji} w_j.$$

因为  $q_1 = v_1$ , 所以  $w_1 = [1, 0, \dots, 0]^T$ . 又  $G$  是上 Hessenberg 矩阵, 利用归纳法, 我们可以证明  $w_i$  的第  $i+1$  到第  $n$  个元素均为 0. 所以  $W$  是一个上三角矩阵. 又  $W$  是正交矩阵, 所以  $W = \text{diag}(1, \pm 1, \dots, \pm 1)$ . 由此, 定理结论成立.  $\square$

由于  $Q_k$  的其它列都是由  $Q_k$  的第一列唯一确定 (至多相差一个符号), 所以我们只要找到一个正交矩阵  $\tilde{Q}_k$  使得其第一列与  $Q_k$  的第一列相等, 且  $\tilde{Q}_k^T A_k \tilde{Q}_k$  为上 Hessenberg 矩阵, 则由隐式 Q 定理可知  $\tilde{Q}_k = W Q_k$ , 其中  $W = \text{diag}(1, \pm 1, \dots, \pm 1)$ . 于是

$$\tilde{Q}_k^T A_k \tilde{Q}_k = W^T Q_k^T A_k Q_k W = W^T A_{k+1} W.$$

又  $W^T A_{k+1} W$  与  $A_{k+1}$  相似, 且对角线元素相等, 而其它元素也至多相差一个符号, 所以不会影响  $A_{k+1}$  的收敛性, 即下三角元素收敛到 0, 对角线元素收敛到  $A$  的特征值. 换句话说, 在 QR 迭代算法中, 如果我们用  $\tilde{Q}_k^T A_k \tilde{Q}_k$  代替  $Q_k A_k Q_k^T$ , 即直接令  $A_{k+1} = \tilde{Q}_k^T A_k \tilde{Q}_k$ , 则其收敛性与原 QR 迭代算法没有任何区别! 这就是隐式 QR 迭代的基本思想.

在实际计算中,  $\tilde{Q}_k$  是一系列 Givens 变换的乘积. 下面我们举一个例子, 具体说明如何利用隐式 Q 定理, 由  $A_1$  得到  $A_2$ .

**例 5.5** 设  $A \in \mathbb{R}^{5 \times 5}$  是一个不可约上 Hessenberg 矩阵, 即

$$A_1 = A = \begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ 0 & * & * & * & * \\ 0 & 0 & * & * & * \\ 0 & 0 & 0 & * & * \end{bmatrix}.$$

**第一步:** 构造一个 Givens 变换  $G_1$ , 其转置如下

$$G_1^T \triangleq G(1, 2, \theta_1) = \begin{bmatrix} c_1 & s_1 & & & \\ -s_1 & c_1 & & & \\ & & 1 & & \\ & & & 1 & \\ & & & & 1 \end{bmatrix}.$$



这里  $G_1$  的第一列  $[c_1, s_1, 0, \dots, 0]^T$  就是  $A_1 - \sigma_1 I$  的第一列  $[a_{11} - \sigma_1, a_{21}, 0, \dots, 0]^T$  的单位化后的列向量, 其中  $\sigma_1$  是位移. 于是有

$$G_1^T A = \begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ 0 & * & * & * & * \\ 0 & 0 & * & * & * \\ 0 & 0 & 0 & * & * \end{bmatrix} \quad \text{和} \quad A^{(1)} \triangleq G_1^T A G_1 = \begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ + & * & * & * & * \\ 0 & 0 & * & * & * \\ 0 & 0 & 0 & * & * \end{bmatrix}.$$

与  $A_1$  相比较,  $A^{(1)}$  在  $(3, 1)$  位置上多出一个非零元, 我们把它记为 “+”, 并称之为 **bulge**. 在下面的计算过程中, 我们的目标就是将其 “赶” 出矩阵, 从而得到一个新的上 Hessenberg 矩阵, 即  $A_2$ .

**第二步:** 为了消去这个 bulge, 我们可以构造 Givens 变换

$$G_2^T \triangleq G(2, 3, \theta_2) = \begin{bmatrix} 1 & & & & \\ & c_2 & s_2 & & \\ & -s_2 & c_2 & & \\ & & & 1 & \\ & & & & 1 \end{bmatrix} \quad \text{使得} \quad G_2^T A^{(1)} = \begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ 0 & * & * & * & * \\ 0 & 0 & * & * & * \\ 0 & 0 & 0 & * & * \end{bmatrix}.$$

为了保持与原矩阵的相似性, 需要再右乘  $G_2$ , 所以

$$A^{(2)} \triangleq G_2^T A^{(1)} G_2 = \begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ 0 & * & * & * & * \\ 0 & + & * & * & * \\ 0 & 0 & 0 & * & * \end{bmatrix}.$$

此时, bulge 从  $(3, 1)$  位置被 “赶” 到  $(4, 2)$  位置.

**第三步:** 与第二步类似, 构造 Givens 变换

$$G_3^T \triangleq G(3, 4, \theta_3) = \begin{bmatrix} 1 & & & & \\ & 1 & & & \\ & & c_3 & s_3 & \\ & & -s_3 & c_3 & \\ & & & & 1 \end{bmatrix} \quad \text{使得} \quad G_3^T A^{(2)} = \begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ 0 & * & * & * & * \\ 0 & 0 & * & * & * \\ 0 & 0 & 0 & * & * \end{bmatrix}.$$

这时

$$A^{(3)} \triangleq G_3^T A^{(2)} G_3 = \begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ 0 & * & * & * & * \\ 0 & 0 & * & * & * \\ 0 & 0 & + & * & * \end{bmatrix}.$$

于是, bulge 又从  $(4, 2)$  位置又被 “赶” 到  $(5, 3)$  位置.



第四步: 再次构造 Givens 变换

$$G_4^T \triangleq G(4, 5, \theta_4) = \begin{bmatrix} 1 & & & & \\ & 1 & & & \\ & & 1 & & \\ & & & c_4 & s_4 \\ & & & -s_4 & c_4 \end{bmatrix} \quad \text{使得} \quad G_4^T A^{(3)} = \begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ 0 & * & * & * & * \\ 0 & 0 & * & * & * \\ 0 & 0 & 0 & * & * \end{bmatrix}$$

于是

$$A^{(4)} \triangleq G_4^T A^{(3)} G_4 = \begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ 0 & * & * & * & * \\ 0 & 0 & * & * & * \\ 0 & 0 & 0 & * & * \end{bmatrix}.$$

现在, bulge 已经被“赶”出矩阵, 且

$$A^{(4)} = G_4^T G_3^T G_2^T G_1^T A_1 G_1 G_2 G_3 G_4 = \tilde{Q}_1^T A_1 \tilde{Q}_1,$$

其中  $\tilde{Q}_1 = G_1 G_2 G_3 G_4$ . 通过直接计算可知,  $\tilde{Q}_1$  的第一列为  $[c_1, s_1, 0, 0, 0]^T$ , 即  $A_1 - \sigma_1 I$  的第一列的单位化. 根据隐式 Q 定理,  $A_2 \triangleq A^{(4)} = \tilde{Q}_1^T A_1 \tilde{Q}_1$  就是我们所需要的矩阵.

如果  $A \in \mathbb{R}^{n \times n}$  是上 Hessenberg 矩阵, 则使用上面的算法, 带位移 QR 迭代中每一步的运算量为  $6n^2 + O(n)$ .

### 位移的选取

在带位移的 QR 迭代算法中, 位移的选取非常重要. 通常, 位移越离某个特征值越近, 收敛速度就越快. 由习题 5.6 可知, 如果位移  $\sigma$  与某个特征值非常接近, 则  $A_k(n, n) - \sigma$  就非常接近于 0. 这说明  $A_k(n, n)$  通常会首先收敛到  $A$  的一个特征值, 所以  $\sigma = A_k(n, n)$  是一个不错的选择. 但是, 如果这个特征值是复数, 这种位移选取方法就可能失效.

下面我们介绍一种针对共轭复特征值的位移选取方法, 即双位移策略.

设  $\sigma \in \mathbb{C}$  是  $A$  的某个复特征值  $\lambda$  的一个很好的近似, 则其共轭  $\bar{\sigma}$  也应该是  $\bar{\lambda}$  的一个很好的近似. 因此我们可以考虑**双位移**策略, 即先以  $\sigma$  为位移迭代一次, 然后再以  $\bar{\sigma}$  为位移迭代一次, 如此不断交替进行迭代. 这样就有

$$\begin{aligned} A_1 - \sigma I &= Q_1 R_1, \\ A_2 &= R_1 Q_1 + \sigma I, \\ A_2 - \bar{\sigma} I &= Q_2 R_2, \\ A_3 &= R_2 Q_2 + \bar{\sigma} I. \end{aligned} \tag{5.5}$$

容易验证

$$A_3 = Q_2^T A_2 Q_2 = Q_2^* Q_1^* A_1 Q_1 Q_2 = Q^* A_1 Q,$$

其中  $Q = Q_1 Q_2$ . 我们注意到  $\sigma$  可能是复的, 所以  $Q_1$  和  $Q_2$  都可能是复矩阵. 但我们却可以选取适当的  $Q_1$  和  $Q_2$ , 使得  $Q = Q_1 Q_2$  是实正交矩阵.





**引理 5.7** 在双位移 QR 迭代 (5.5) 中, 我们可以选取酉矩阵  $Q_1$  和  $Q_2$  使得  $Q = Q_1 Q_2$  是实矩阵.

**证明.** 由于

$$Q_2 R_2 = A_2 - \bar{\sigma} I = R_1 Q_1 + (\sigma - \bar{\sigma}) I,$$

所以

$$\begin{aligned} Q_1 Q_2 R_2 R_1 &= Q_1 (R_1 Q_1 + (\sigma - \bar{\sigma}) I) R_1 \\ &= Q_1 R_1 Q_1 R_1 + (\sigma - \bar{\sigma}) Q_1 R_1 \\ &= (A_1 - \sigma I)^2 + (\sigma - \bar{\sigma})(A_1 - \sigma I) \\ &= A_1^2 - (\sigma + \bar{\sigma}) A_1 + \bar{\sigma} \sigma I, \\ &= A_1^2 - 2\operatorname{Re}(\sigma) A_1 + |\sigma|^2 I \\ &\in \mathbb{R}^{n \times n}. \end{aligned}$$

又  $Q_1 Q_2$  是酉矩阵,  $R_2 R_1$  是上三角矩阵, 故  $(Q_1 Q_2)(R_2 R_1)$  是实矩阵  $A_1^2 - 2\operatorname{Re}(\sigma) A_1 + |\sigma|^2 I = (A_1 - \sigma I)(A_1 - \bar{\sigma} I)$  的 QR 分解. 所以  $Q_1 Q_2$  和  $R_2 R_1$  都可以是实矩阵.  $\square$

由这个引理可知, 存在  $Q_1$  和  $Q_2$ , 使得  $Q = Q_1 Q_2$  是实正交矩阵, 从而  $A_3 = Q^T A_1 Q$  也是实矩阵. 这时, 在迭代过程 (5.5) 中, 我们无需计算  $A_2$ , 可直接由  $A_1$  计算出  $A_3$ .

具体计算过程仍然是根据隐式 Q 定理: 我们只要找到一个实正交矩阵  $Q$ , 使得其第一列与  $A_1^2 - 2\operatorname{Re}(\sigma) A_1 + |\sigma|^2 I$  的第一列平行, 并且  $A_3 = Q^T A_1 Q$  是上 Hessenberg 矩阵即可.

由于  $A_1^2 - 2\operatorname{Re}(\sigma) A_1 + |\sigma|^2 I$  的第一列为

$$\begin{bmatrix} a_{11}^2 + a_{12}a_{21} - 2\operatorname{Re}(\sigma)a_{11} + |\sigma|^2 \\ a_{21}(a_{11} + a_{22} - 2\operatorname{Re}(\sigma)) \\ a_{21}a_{32} \\ 0 \\ \vdots \\ 0 \end{bmatrix}. \quad (5.6)$$

所以  $Q$  的第一列是上述向量的单位化. 其它过程可以通过隐式 QR 迭代来实现. 但此时的“bulge”是一个  $2 \times 2$  的小矩阵. 因此, 在双位移隐式 QR 迭代过程中, 我们需要使用 Householder 变换.

需要指出的是, 双位移 QR 迭代算法中的运算都是实数运算.

下面我们举一个例子, 具体说明如何在实数运算下实现双位移隐式 QR 迭代算法.

**例 5.6** 设  $A \in \mathbb{R}^{6 \times 6}$  是一个不可约上 Hessenberg 矩阵, 即

$$A_1 = A = \begin{bmatrix} * & * & * & * & * & * \\ * & * & * & * & * & * \\ 0 & * & * & * & * & * \\ 0 & 0 & * & * & * & * \\ 0 & 0 & 0 & * & * & * \\ 0 & 0 & 0 & 0 & * & * \end{bmatrix}.$$

**第一步:** 构造一个正交矩阵  $H_1 = \begin{bmatrix} \tilde{H}_1^\top & 0 \\ 0 & I_{3 \times 3} \end{bmatrix}$ , 其中  $\tilde{H}_1 \in \mathbb{R}^{3 \times 3}$ , 使得其第一列与向量 (5.6)

平行. 于是有

$$H_1^\top A = \begin{bmatrix} * & * & * & * & * & * \\ * & * & * & * & * & * \\ + & * & * & * & * & * \\ 0 & 0 & * & * & * & * \\ 0 & 0 & 0 & * & * & * \\ 0 & 0 & 0 & 0 & * & * \end{bmatrix} \quad \text{和} \quad A^{(1)} \triangleq H_1^\top A H_1 = \begin{bmatrix} * & * & * & * & * & * \\ * & * & * & * & * & * \\ + & * & * & * & * & * \\ + & + & * & * & * & * \\ 0 & 0 & 0 & * & * & * \\ 0 & 0 & 0 & 0 & * & * \end{bmatrix}.$$

与  $A_1$  相比较,  $A^{(1)}$  在 (3, 1), (4, 1) 和 (4, 2) 位置上出现 bulge. 在下面的计算过程中, 我们的目标就是要把它们“赶”出矩阵, 从而得到一个新的上 Hessenberg 矩阵.

**第二步:** 令  $H_2 = \begin{bmatrix} I_{1 \times 1} & 0 & 0 \\ 0 & \tilde{H}_2^\top & 0 \\ 0 & 0 & I_{2 \times 2} \end{bmatrix}$ , 其中  $\tilde{H}_2 \in \mathbb{R}^{3 \times 3}$  是对应于  $A(2:4, 1)$  的 Householder

变换, 使得

$$H_2^\top A^{(1)} = \begin{bmatrix} * & * & * & * & * & * \\ * & * & * & * & * & * \\ 0 & * & * & * & * & * \\ 0 & + & * & * & * & * \\ 0 & 0 & 0 & * & * & * \\ 0 & 0 & 0 & 0 & * & * \end{bmatrix} \quad \text{和} \quad A^{(2)} \triangleq H_2^\top A^{(1)} H_2 = \begin{bmatrix} * & * & * & * & * & * \\ * & * & * & * & * & * \\ 0 & * & * & * & * & * \\ 0 & + & * & * & * & * \\ 0 & + & + & * & * & * \\ 0 & 0 & 0 & 0 & * & * \end{bmatrix}.$$

这时, 我们将 bulge 向右下角方向“赶”了一个位置.

**第三步:** 与第二步类似, 令  $H_3 = \begin{bmatrix} I_{2 \times 2} & 0 & 0 \\ 0 & \tilde{H}_3^\top & 0 \\ 0 & 0 & I_{1 \times 1} \end{bmatrix}$ , 其中  $\tilde{H}_3 \in \mathbb{R}^{3 \times 3}$  是对应于  $A(3:5, 2)$

的 Householder 变换, 使得

$$H_3^\top A^{(2)} = \begin{bmatrix} * & * & * & * & * & * \\ * & * & * & * & * & * \\ 0 & * & * & * & * & * \\ 0 & 0 & * & * & * & * \\ 0 & 0 & + & * & * & * \\ 0 & 0 & 0 & 0 & * & * \end{bmatrix} \quad \text{和} \quad A^{(3)} \triangleq H_3^\top A^{(2)} H_3 = \begin{bmatrix} * & * & * & * & * & * \\ * & * & * & * & * & * \\ 0 & * & * & * & * & * \\ 0 & 0 & * & * & * & * \\ 0 & 0 & + & * & * & * \\ 0 & 0 & + & + & * & * \end{bmatrix}.$$

此时, bulge 又被向右下角方向“赶”了一个位置.

**第四步:** 令  $H_4 = \begin{bmatrix} I_{3 \times 3} & 0 \\ 0 & \tilde{H}_4^\top \end{bmatrix}$ , 其中  $\tilde{H}_4 \in \mathbb{R}^{3 \times 3}$  是对应于  $A(4:6, 3)$  的 Householder 变换,



使得

$$H_4^T A^{(3)} = \begin{bmatrix} * & * & * & * & * & * \\ * & * & * & * & * & * \\ 0 & * & * & * & * & * \\ 0 & 0 & * & * & * & * \\ 0 & 0 & 0 & * & * & * \\ 0 & 0 & 0 & + & * & * \end{bmatrix} \quad \text{和} \quad A^{(4)} \triangleq H_4^T A^{(3)} H_4 = \begin{bmatrix} * & * & * & * & * & * \\ * & * & * & * & * & * \\ 0 & * & * & * & * & * \\ 0 & 0 & * & * & * & * \\ 0 & 0 & 0 & * & * & * \\ 0 & 0 & 0 & + & * & * \end{bmatrix}.$$

**第五步:** 此时, 只需构造一个 Givens 变换  $G_5 = \begin{bmatrix} I_{4 \times 4} & 0 \\ 0 & G(4, 5, \theta)^T \end{bmatrix}$ , 使得

$$G_5^T A^{(4)} = \begin{bmatrix} * & * & * & * & * & * \\ * & * & * & * & * & * \\ 0 & * & * & * & * & * \\ 0 & 0 & * & * & * & * \\ 0 & 0 & 0 & * & * & * \\ 0 & 0 & 0 & 0 & * & * \end{bmatrix} \quad \text{和} \quad A^{(5)} \triangleq G_5^T A^{(4)} G_5 = \begin{bmatrix} * & * & * & * & * & * \\ * & * & * & * & * & * \\ 0 & * & * & * & * & * \\ 0 & 0 & * & * & * & * \\ 0 & 0 & 0 & * & * & * \\ 0 & 0 & 0 & 0 & * & * \end{bmatrix}.$$

现在, bulge 已经被全部消除, 且


$$A^{(5)} = Q^T A Q,$$

其中  $Q = H_1 H_2 H_3 H_4 G_5$ . 通过直接计算可知,  $Q$  的第一列即为  $H_1$  的第一列, 也就是向量 (5.6) 的单位化. 根据隐式  $Q$  定理, 可以直接令  $A_3 \triangleq A^{(5)} = Q^T A Q$ .


最后要考虑位移  $\sigma$  的具体选取问题. 在单位 QR 迭代算法中, 如果  $A$  的特征值都是实数的话, 我们可以取  $\sigma_k = A_k(n, n)$ . 推广到复共轭特征值上, 我们可以取  $A_k$  的右下角矩阵

$$\begin{bmatrix} A_k(n-1, n-1) & A_k(n-1, n) \\ A_k(n, n-1) & A_k(n, n) \end{bmatrix}$$

的复共轭特征值作为双位移. 这样选取的位移就是 **Francis 位移**.

 如果上述矩阵的两个特征值都是实的, 则选取其中模较小的特征值做单位 QR 迭代.

一般来说, 采用 Francis 位移的 QR 迭代会使得迭代矩阵的右下角收敛到一个上三角矩阵 (两个实特征值) 或一个  $2 \times 2$  的矩阵 (一对复共轭特征值), 而且通常会有二次渐进收敛性. 在实际计算中, 计算一个特征值一般平均只需迭代两步.

 判断 QR 迭代算法是否收敛主要是看  $A_k(n-1, n-2)$  (或  $A_k(n, n-1)$ ) 是否趋向于 0.

但需要指出的是, QR 迭代并不是对所有矩阵都收敛. 例如:

$$A = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}.$$

对于上面的矩阵, 采用 Francis 位移的 QR 迭代算法无效.

另外, 也可以考虑多重位移策略, 参见 [54].

### 收缩 Deflation

收缩 (deflation) 技术是提高 QR 迭代实用性的一个非常重要的手段.

在隐式 QR 迭代过程中, 当矩阵  $A_{k+1}$  的某个下次对角线元素  $a_{i+1,i}$  很小时, 我们可以将其设为 0. 由于  $A_{k+1}$  是上 Hessenberg 矩阵, 这时  $A_{k+1}$  就可以写成分块上三角形式, 其中两个对角块都是上 Hessenberg 矩阵. 因此我们可以将隐式 QR 迭代作用在这两个规模相对较小的矩阵上, 从而可以大大节约运算量.



## 5.2 对称特征值问题

当  $A \in \mathbb{R}^{n \times n}$  是对称矩阵时, 我们可以充分利用  $A$  的对称结构构造更加快速高效的算法.

关于计算对称矩阵的特征值和特征值向量, 目前常用算法有:

- **Jacobi 迭代方法**: 最古老的方法, 收敛速度较慢, 但精度较高, 且很适合并行计算.
- **Rayleigh 商迭代方法**: 利用 Rayleigh 商作为位移的反迭代算法, 一般具有三次局部收敛性.
- **对称 QR 迭代方法**: 用隐式 QR 算法计算对称矩阵的特征值和特征向量就是对称 QR 迭代方法. 如果只需计算对称三对角矩阵的所有特征值, 则该算法是目前最快的方法, 运算量为  $O(n^2)$ . 如果需要计算所有的特征值和特征向量, 则运算量约为  $6n^3$ .
- **分而治之法 (Divide-and-Conquer)**: 同时计算对称三对角矩阵的特征值和特征向量的一种快速算法. 基本思想是将大矩阵分解成小矩阵, 然后利用递归思想求特征值和特征向量. 在最坏的情形下, 运算量为  $O(n^3)$ , 但在实际应用中, 平均为  $O(n^{2.3})$ . 如果使用快速多极子算法 (FMM) 后, 理论上的运算量可降低到  $O(n \log^p n)$ , 其中  $p$  是一个较小的整数, 这使得分而治之算法成为目前求解对称三对角矩阵的所有特征值和特征向量的最快方法之一.
- **对分法**: 对分法主要用于求解对称三对角矩阵在某个区间中的特征值, 运算量约为  $O(kn)$ , 其中  $k$  为所需计算的特征值的个数. 计算出特征值后再利用反迭代计算特征向量, 在最佳情况下, 即特征值“适当分离”时, 运算量约为  $O(kn)$ , 但在最差情况下, 即特征值成串地紧靠在一起时, 运算量约为  $O(k^2n)$ , 而且不能保证特征向量的精度 (虽然实际上它几乎是精确的).

本节主要介绍 Jacobi 迭代方法, 分而治之法和对分法.

除了 Jacobi 迭代和 Rayleigh 商迭代外, 其余算法都需要先将对称矩阵三对角化. 这个过程大约需花费  $\frac{4}{3}n^3$  的工作量, 如果需要计算特征向量的话, 则总运算量约为  $\frac{8}{3}n^3$ .

### 5.2.1 Jacobi 迭代方法

该算法的基本思想是通过一系列的 **Jacobi 旋转**  $J_k$ , 将  $A$  正交相似于一个对角矩阵, 即

$$A^{(0)} = A, \quad A^{(k+1)} = J_k^T A^{(k)} J_k, \quad k = 0, 1, \dots,$$

且  $A^{(k)}$  收敛到一个对角矩阵, 其中  $J_k$  为 **Jacobi 旋转**, 通常选取  $J_k$  为 Givens 变换, 即

$$J_k = G(i_k, j_k, \theta_k) = \begin{matrix} & i_k & j_k \\ \begin{matrix} i_k \\ j_k \end{matrix} & \begin{bmatrix} 1 & & & & \\ & \ddots & & & \\ & & 1 & & \\ & & & \cos \theta_k & -\sin \theta_k \\ & & & \sin \theta_k & \cos \theta_k \\ & & & & 1 & \\ & & & & & \ddots & \\ & & & & & & 1 \end{bmatrix} \end{matrix}.$$

易知, 在  $A^{(k)}$  两边分别左乘  $J_k^T$  和右乘  $J_k$  时, 只会修改  $A^{(k)}$  的第  $i_k$  和第  $j_k$  行, 以及第  $i_k$  和第  $j_k$  列.

由于  $A^{(k)}$  是对称矩阵, 由下面的引理可知, 通过选取适当的  $\theta_k$ , 可以将  $A^{(k)}(i_k, j_k)$  和  $A^{(k)}(j_k, i_k)$  同时化为 0.

**引理 5.8** 设  $A \in \mathbb{R}^{2 \times 2}$  是对称矩阵, 则存在 Givens 变换  $G \in \mathbb{R}^{2 \times 2}$ , 使得  $G^T A G$  为对角矩阵.

**证明.** 设

$$A = \begin{bmatrix} a & b \\ b & c \end{bmatrix}, \quad G = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix},$$

则

$$\begin{aligned} G^T A G &= \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix}^T \begin{bmatrix} a & b \\ b & c \end{bmatrix} \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix} \\ &= \begin{bmatrix} a \cos^2 \theta + c \sin^2 \theta + b \sin 2\theta & \frac{1}{2}(c-a) \sin 2\theta + b \cos 2\theta \\ \frac{1}{2}(c-a) \sin 2\theta + b \cos 2\theta & a \sin^2 \theta + c \cos^2 \theta - b \sin 2\theta \end{bmatrix} \end{aligned}$$

令  $\frac{1}{2}(c-a) \sin 2\theta + b \cos 2\theta = 0$ , 可得

$$\frac{a-c}{2b} = \cot 2\theta = \frac{1 - \tan^2 \theta}{2 \tan \theta}.$$

解得

$$\tan \theta = \frac{\text{sign}(\tau)}{|\tau| + \sqrt{1 + \tau^2}}, \quad \tau = \frac{a-c}{2b}.$$

故引理结论成立. □

为了使得  $A^{(k)}$  收敛到一个对角矩阵, 其非对角线元素必须趋向于 0. 记  $\text{off}(A)$  为所有非对角线元素的平方和, 即

$$\text{off}(A) = \sum_{i \neq j} a_{ij}^2 = \|A\|_F^2 - \sum_{i=1}^n a_{ii}^2,$$

我们的目标就是使得  $\text{off}(A)$  尽快趋于 0.

**引理 5.9** 设  $A = [a_{ij}] \in \mathbb{R}^{n \times n}$  是对称矩阵,  $\hat{A} = [\hat{a}_{ij}] = J^T A J$ ,  $J = G(i, j, \theta)$ , 其中  $\theta$  的选取使得  $\hat{a}_{ij} = \hat{a}_{ji} = 0$ , 则

$$\text{off}(\hat{A}) = \text{off}(A) - 2a_{ij}^2, \quad i \neq j.$$

**证明.** 记  $A = [a_1, a_2, \dots, a_n]$ . 令  $\tilde{A} = J^T A = [\tilde{a}_{ij}]_{n \times n}$ . 由于  $J$  是正交阵, 故

$$\|J^T a_k\|_2 = \|a_k\|_2, \quad k = 1, 2, \dots, n.$$

又  $J^T$  左乘  $a_k$  时, 只影响其第  $i$  和第  $j$  个元素的值, 故由  $\|J^T a_i\|_2 = \|a_i\|_2$  和  $\|J^T a_j\|_2 = \|a_j\|_2$  可得

$$\tilde{a}_{ii}^2 + \tilde{a}_{ji}^2 = a_{ii}^2 + a_{ji}^2, \quad \tilde{a}_{ij}^2 + \tilde{a}_{jj}^2 = a_{ij}^2 + a_{jj}^2. \quad (5.7)$$

同理, 由  $\hat{A} = \tilde{A} J$  可得

$$\hat{a}_{ii}^2 + \hat{a}_{ij}^2 = \tilde{a}_{ii}^2 + \tilde{a}_{ij}^2, \quad \hat{a}_{ji}^2 + \hat{a}_{jj}^2 = \tilde{a}_{ji}^2 + \tilde{a}_{jj}^2. \quad (5.8)$$



又  $\hat{a}_{ij} = \hat{a}_{ji} = 0$ , 故

$$\hat{a}_{ii}^2 + \hat{a}_{jj}^2 = a_{ii}^2 + a_{jj}^2 + a_{ij}^2 + a_{ji}^2 = a_{ii}^2 + a_{jj}^2 + 2a_{ij}^2.$$

由于  $J^T A J$  只影响  $A$  的第  $i, j$  行和第  $i, j$  列, 故对角线元素中只有  $a_{ii}$  和  $a_{jj}$  受影响. 所以

$$\sum_{k=1}^n \hat{a}_{kk}^2 = \sum_{k=1}^n a_{kk}^2 + 2a_{ij}^2,$$

故

$$\text{off}(\hat{A}) = \|\hat{A}\|_2^2 - \sum_{k=1}^n \hat{a}_{kk}^2 = \|A\|_2^2 - \sum_{k=1}^n a_{kk}^2 - 2a_{ij}^2 = \text{off}(A) - 2a_{ij}^2,$$

即引理结论成立. □

由此可知,  $\text{off}(A^{(k)})$  总是不断减小的. 下面给出 Jacobi 迭代算法.

### 算法 5.7. Jacobi 迭代算法

```

1: Given a symmetric matrix  $A \in \mathbb{R}^{n \times n}$ 
2: if eigenvectors are desired then
3:   set  $J = I$  and  $flag = 1$ 
4: end if
5: while not converge do
6:   choose an index pair  $(i, j)$  such that  $a_{ij} \neq 0$ 
7:    $\tau = (a_{ii} - a_{jj}) / (2a_{ij})$ 
8:    $t = \text{sign}(\tau) / (|\tau| + \sqrt{1 + \tau^2})$   % 计算  $\tan \theta$ 
9:    $c = 1 / \sqrt{1 + t^2}$   % 计算  $\cos \theta$ 
10:   $s = c \cdot t$   % 计算  $\sin \theta$ 
11:   $A = G(i, j, \theta)^T A G(i, j, \theta)$ 
12:  if  $flag = 1$  then
13:     $J = J \cdot G(i, j, \theta)$ 
14:  end if
15: end while

```

该算法涉及到  $a_{ij}$  的选取问题, 一种直观的选取方法就是使得  $a_{ij}$  为所有非对角线元素中绝对值最大的一个, 这就是 **经典 Jacobi 迭代算法**.

### 算法 5.8. 经典 Jacobi 迭代算法

```

1: Given a symmetric matrix  $A \in \mathbb{R}^{n \times n}$ 
2: if eigenvectors are desired then
3:   set  $J = I$  and  $flag = 1$ 
4: end if
5: while  $\text{off}(A) > tol$  do

```

```

6:   choose  $(i, j)$  such that  $|a_{ij}| = \max_{k \neq l} |a_{kl}|$     % 选取绝对值最大的元素
7:    $\tau = (a_{ii} - a_{jj}) / (2a_{ij})$ 
8:    $t = \text{sign}(\tau) / (|\tau| + \sqrt{1 + \tau^2})$ 
9:    $c = 1 / \sqrt{1 + t^2}$ 
10:   $s = c \cdot t$ 
11:   $A = G(i, j, \theta)^T A G(i, j, \theta)$ 
12:  if  $flag = 1$  then
13:     $J = J \cdot G(i, j, \theta)$ 
14:  end if
15: end while

```

可以证明, 经典 Jacobi 算法至少是线性收敛的.

**定理 5.10** 对于经典 Jacobi 算法 5.8, 有

$$\text{off}(A^{(k+1)}) \leq \left(1 - \frac{1}{N}\right) \text{off}(A^{(k)}), \quad N = \frac{n(n-1)}{2}.$$

故  $k$  步迭代后, 有

$$\text{off}(A^{(k)}) \leq \left(1 - \frac{1}{N}\right)^k \text{off}(A^{(0)}) = \left(1 - \frac{1}{N}\right)^k \text{off}(A).$$

**证明.** 由于在经典 Jacobi 算法 5.8 中,  $|a_{ij}| = \max_{k \neq l} |a_{kl}|$ , 故  $\text{off}(A^{(k)}) \leq n(n-1) \left(a_{ij}^{(k)}\right)^2$ , 即

$$2 \left(a_{ij}^{(k)}\right)^2 \geq \frac{1}{N} \text{off}(A^{(k)}), \quad N = \frac{n(n-1)}{2}.$$

所以由引理 5.9 可知

$$\text{off}(A^{(k+1)}) = \text{off}(A^{(k)}) - \left(a_{ij}^{(k)}\right)^2 \leq \left(1 - \frac{1}{N}\right) \text{off}(A^{(k)}).$$

□

事实上, 经典 Jacobi 算法最终是二次局部收敛的.

**定理 5.11** 经典 Jacobi 算法 5.8 是  $N$  步局部二次收敛的, 即对足够大的  $k$ , 有

$$\text{off}(A^{(k+N)}) = O\left(\text{off}^2(A^{(k)})\right).$$

由于在经典 Jacobi 算法中, 每一步都要寻找绝对值最大的非对角元, 比较费时, 因此实用性较差. 我们可以通过逐行扫描来选取  $(i, j)$ , 这就是 **循环 Jacobi 迭代算法**.

**算法 5.9.** 循环 Jacobi 迭代算法 (逐行扫描)

```

1: Given a symmetric matrix  $A \in \mathbb{R}^{n \times n}$ 
2: if eigenvectors are desired then
3:   set  $J = I$  and  $flag = 1$ 
4: end if

```





```

5: while off( $A$ ) >  $tol$  do
6:   for  $i = 1$  to  $n - 1$  do
7:     for  $j = i + 1$  to  $n$  do
8:       if  $a_{ij} \neq 0$  then
9:          $\tau = (a_{ii} - a_{jj}) / (2a_{ij})$ 
10:         $t = \text{sign}(\tau) / (|\tau| + \sqrt{1 + \tau^2})$ 
11:         $c = 1 / \sqrt{1 + t^2}$ 
12:         $s = c \cdot t$ 
13:         $A = G(i, j, \theta)^\top A G(i, j, \theta)$ 
14:        if  $flag = 1$  then
15:           $J = J \cdot G(i, j, \theta)$ 
16:        end if
17:      end if
18:    end for
19:  end for
20: end while

```

循环 Jacobi 也具有局部二次收敛性 [55, page 270].

### 5.2.2 分而治之法

分而治之 (Divide-and-Conquer) 算法是由 Cuppen [9] 于 1981 年首次提出, 但直到 1995 年才出现稳定的实现方式 [26]. 该算法是目前计算维数大于 25 的矩阵的所有特征值和特征向量的最快算法. 下面我们介绍该算法.

考虑不可约对称三对角矩阵

$$T = \left[ \begin{array}{cc|cc} a_1 & b_1 & & \\ & \ddots & \ddots & \\ b_1 & \ddots & a_{m-1} & b_{m-1} \\ & \ddots & b_{m-1} & a_m \\ \hline & & b_m & a_{m+1} & b_{m+1} \\ & & b_m & \ddots & \ddots \\ & & & \ddots & \ddots & b_{n-1} \\ & & & & b_{n-1} & a_n \end{array} \right]$$

$$= \left[ \begin{array}{ccc|ccc} a_1 & b_1 & & & & \\ b_1 & \ddots & \ddots & & & \\ & \ddots & a_{m-1} & b_{m-1} & & \\ & & b_{m-1} & a_m - b_m & & \\ \hline & & & a_{m+1} - b_m & b_{m+1} & \\ & & & b_{m+1} & \ddots & \ddots \\ & & & & \ddots & \ddots & b_{n-1} \\ & & & & & b_{n-1} & a_n \end{array} \right] + \left[ \begin{array}{ccc|ccc} & & & & & \\ & & & & & \\ & & & & & \\ & & & & & \\ \hline & & & b_m & b_m & \\ & & & b_m & b_m & \\ & & & & & \\ & & & & & \\ & & & & & \end{array} \right]$$

$$= \left[ \begin{array}{c|c} T_1 & 0 \\ \hline 0 & T_2 \end{array} \right] + b_m v v^T,$$

其中  $v = [0, \dots, 0, 1, 1, 0, \dots, 0]^T$ . 假定  $T_1$  和  $T_2$  的特征值分解已经计算出来了, 即  $T_1 = Q_1 \Lambda_1 Q_1^T$ ,  $T_2 = Q_2 \Lambda_2 Q_2^T$ , 下面考虑  $T$  的特征值分解.

首先介绍一个引理.

**引理 5.12** 设  $x, y \in \mathbb{R}^n$ , 则  $\det(I + xy^T) = 1 + y^T x$ .

(证明留作练习)

我们首先考虑  $T$  的特征值与  $T_1$  和  $T_2$  的特征值之间的关系.

$$\begin{aligned} T &= \begin{bmatrix} T_1 & 0 \\ 0 & T_2 \end{bmatrix} + b_m v v^T \\ &= \begin{bmatrix} Q_1 \Lambda_1 Q_1^T & 0 \\ 0 & Q_2 \Lambda_2 Q_2^T \end{bmatrix} + b_m v v^T \\ &= \begin{bmatrix} Q_1 & 0 \\ 0 & Q_2 \end{bmatrix} \left( \begin{bmatrix} \Lambda_1 & 0 \\ 0 & \Lambda_2 \end{bmatrix} + b_m u u^T \right) \begin{bmatrix} Q_1 & 0 \\ 0 & Q_2 \end{bmatrix}^T, \end{aligned}$$

其中

$$u = \begin{bmatrix} Q_1 & 0 \\ 0 & Q_2 \end{bmatrix}^T v = \begin{bmatrix} Q_1^T \text{ 的最后一列} \\ Q_2^T \text{ 的第一列} \end{bmatrix}.$$

令  $\alpha = b_m$ ,  $D = \text{diag}(\Lambda_1, \Lambda_2) = \text{diag}(d_1, d_2, \dots, d_n)$ , 并假定  $d_1 \geq d_2 \geq \dots \geq d_n$ . 则  $T$  的特征值与  $D + \alpha u u^T$  的特征值相同.

下面计算  $D + \alpha u u^T$  的特征值. 设  $\lambda$  是  $D + \alpha u u^T$  的一个特征值, 若  $D - \lambda I$  非奇异, 则

$$\det(D + \alpha u u^T - \lambda I) = \det(D - \lambda I) \cdot \det(I + \alpha(D - \lambda I)^{-1} u u^T).$$

故  $\det(I + \alpha(D - \lambda I)^{-1} u u^T) = 0$ . 又由引理 5.12 可知

$$\det(I + \alpha(D - \lambda I)^{-1} u u^T) = 1 + \alpha u^T (D - \lambda I)^{-1} u = 1 + \alpha \sum_{i=1}^n \frac{u_i^2}{d_i - \lambda} \triangleq f(\lambda).$$

故求  $A$  的特征值等价于求**特征方程 (secular equation)**  $f(\lambda) = 0$  的根. 由于

$$f'(\lambda) = \alpha \sum_{i=1}^n \frac{u_i^2}{(d_i - \lambda)^2},$$



当所有的  $d_i$  都互不相同, 且所有的  $u_i$  都不为零时,  $f(\lambda)$  在  $\lambda \neq d_i$  处都是严格单调的 (见下图).

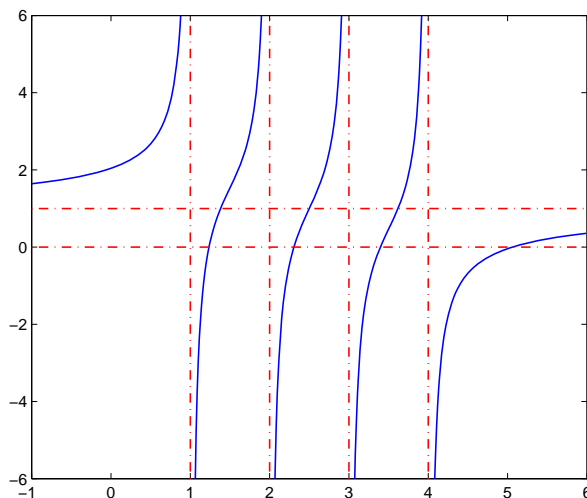


图 5.1.  $f(\lambda) = 1 + 0.5 \left( \frac{1}{4-\lambda} + \frac{1}{3-\lambda} + \frac{1}{2-\lambda} + \frac{1}{1-\lambda} \right)$  的图像

所以  $f(\lambda)$  在每个区间  $(d_{i+1}, d_i)$  内都有一个根, 共  $n-1$  个, 另一个根在  $(d_1, \infty)$  (若  $\alpha > 0$ ) 或  $(-\infty, d_n)$  (若  $\alpha < 0$ ) 中. 由于  $f(\lambda)$  在每个区间  $(d_{i+1}, d_i)$  内光滑且严格单调递增 ( $\alpha > 0$ ) 或递减 ( $\alpha < 0$ ), 所以在实际计算中, 可以使用对分法, 牛顿型方法, 或有理逼近等算法来求解. 通常都能很快收敛, 一般只需迭代几步即可. 因此, 计算一个特征值的运算量约为  $O(n)$ , 计算  $D + \alpha uu^T$  的所有特征值的运算量约为  $O(n^2)$ .

当所有特征值计算出来后, 我们可以利用下面的引理来计算特征向量.

**引理 5.13** 设  $D \in \mathbb{R}^{n \times n}$  为对角矩阵,  $u \in \mathbb{R}^n$ ,  $\alpha \in \mathbb{R}$ , 若  $\lambda$  是  $D + \alpha uu^T$  的特征值, 且  $\lambda \neq d_i$ ,  $i = 1, 2, \dots, n$ , 则  $(D - \lambda I)^{-1}u$  是其对应的特征向量.

**证明.** 由引理 5.12 可知

$$\begin{aligned} 0 &= \det(D + \alpha uu^T - \lambda I) = \det(D - \lambda I) \cdot \det(I + \alpha(D - \lambda I)^{-1}uu^T) \\ &= \det(D - \lambda I) \cdot (1 + \alpha u^T(D - \lambda I)^{-1}u), \end{aligned}$$

故  $1 + \alpha u^T(D - \lambda I)^{-1}u = 0$ , 即  $\alpha u^T(D - \lambda I)^{-1}u = -1$ . 直接计算可得

$$\begin{aligned} (D + \alpha uu^T)((D - \lambda I)^{-1}u) &= (D - \lambda I + \lambda I + \alpha uu^T)(D - \lambda I)^{-1}u \\ &= u + \lambda(D - \lambda I)^{-1}u + (\alpha u^T(D - \lambda I)^{-1}u)u \\ &= u + \lambda(D - \lambda I)^{-1}u - u \\ &= \lambda(D - \lambda I)^{-1}u, \end{aligned}$$

即引理结论成立. □

利用公式  $(D - \lambda I)^{-1}u$  计算所有特征向量的运算量为  $O(n^2)$ . 但这种计算方法的数值稳定性无法保证. 相关讨论和修正方法可参见 [24, 61].

**算法 5.10.** 计算对称三对角矩阵的特征值和特征向量的分而治之法 (函数形式)

```

1: function [Q, Λ] = dc_eig(T)      % T = QΛQT
2: if T is of 1 × 1 then
3:     Q = 1, Λ = T
4:     return
5: end if
6: form T =  $\begin{bmatrix} T_1 & 0 \\ 0 & T_2 \end{bmatrix} + b_m vv^T$ 
7: [Q1, Λ1] = dc_eig(T1)
8: [Q2, Λ2] = dc_eig(T2)
9: form D + αuuT from Λ1, Λ2, Q1, Q2
10: compute the eigenvalues Λ and eigenvectors  $\hat{Q}$  of D + αuuT
11: compute the eigenvectors of T with  $Q = \begin{bmatrix} Q_1 & 0 \\ 0 & Q_2 \end{bmatrix} \cdot \hat{Q}$ 
12: end

```

### 运算量

分而治之算法的计算复杂性分析如下: 用  $t(n)$  表示对  $n$  阶矩阵调用函数 **dc\_eig** 的运算量, 则

$$\begin{aligned}
 t(n) &= 2t(n/2) && \text{递归调用 dc\_eig 两次} \\
 &+ O(n^2) && \text{计算 } D + \alpha uu^T \text{ 的特征值和特征向量} \\
 &+ c \cdot n^3 && \text{计算 } Q.
 \end{aligned}$$

如果计算  $Q$  时使用的是稠密矩阵乘法, 则  $c = 2$ ;

若不计  $O(n^2)$  项, 则由递归公式  $t(n) = 2t(n/2) + c \cdot n^3$  可得  $t(n) \approx c \cdot 4n^3/3$ .

但事实上, 由于**收缩 (deflation)** 现象的存在, 常数  $c$  通常比 1 小得多.

在前面的算法描述过程中, 我们假定  $d_i$  互不相等且  $u_i$  不能为零. 事实上, 容易证明当  $d_i = d_{i+1}$  或  $u_i = 0$  时,  $d_i$  即为  $D + \alpha uu^T$  的特征值, 这种现象我们称为**收缩 (deflation)**. 在实际计算时, 当  $d_i - d_{i+1}$  或  $|u_i|$  小于一个给定的阈值时, 我们就近似认为  $d_i$  为  $D + \alpha uu^T$  的特征值, 即出现收缩现象.

在实际计算中, 收缩现象会经常发生, 而且非常频繁, 所以我们可以而且应该利用这种优点加快分而治之算法的速度 [9, 44].

由于主要的计算量集中在计算  $Q$ , 即算法最后一步的矩阵乘积. 如果  $u_i = 0$ , 则  $d_i$  为特征值, 其对应的特征向量为  $e_i$ , 即  $\hat{Q}$  的第  $i$  列为  $e_i$ , 故计算  $Q$  的第  $i$  列时不需要做任何的计算. 当  $d_i = d_{i+1}$  时, 也存在一个类似的简化.



## 拓展阅读

分而治之的算法于 1981 年被首次提出,但直到 1995 年才由 Gu 和 Eisenstat 给出了一种快速稳定的实现方式,称为**箭型分而治之法** (**Arrowhead Divide-and-Conquer**, ADC). 他们做了大量的数值试验,在试验中,当矩阵规模不超过 6 时,就采用对称 QR 迭代来计算特征值和特征向量.在对特征方程求解时,他们采用的是修正的有理逼近法.数值结果表明,ADC 算法的计算精度可以与其他算法媲美,而计算速度通常比对称 QR 迭代快 5 至 10 倍,比 Cuppen 的分而治之法快 2 倍.详细介绍见 [25, 26].

## 5.2.3 对分法

对分法 (Bisection) 的基本思想是利用惯性定理来计算所需的部分特征值.

**定义 5.1** 设  $A$  为对称矩阵,则其**惯性**定义为

$$\text{Inertia}(A) = (\nu, \zeta, \pi)$$

其中  $\nu, \zeta, \pi$  分别表示  $A$  的负特征值,零特征值和正特征值的个数.

**定理 5.14 (Sylvester 惯性定理)** 设  $A \in \mathbb{R}^{n \times n}$  是对称矩阵,  $X \in \mathbb{R}^{n \times n}$  非奇异,则  $X^T A X$  与  $A$  有相同的惯性.

利用 LU 分解可得  $A - zI = LDL^T$ , 其中  $L$  为非奇异下三角矩阵,  $D$  为对角阵, 则

$$\text{Inertia}(A - zI) = \text{Inertia}(D).$$

由于  $D$  是对角矩阵, 所以  $\text{Inertia}(D)$  很容易计算.

设  $\alpha \in \mathbb{R}$ , 记  $\text{Negcount}(A, \alpha)$  为小于  $\alpha$  的  $A$  的特征值的个数, 即

$$\text{Negcount}(A, \alpha) = \#(\lambda(A) < \alpha).$$

设  $\alpha_1 < \alpha_2$ , 则  $A$  在区间  $[\alpha_1, \alpha_2)$  中的特征值个数为

$$\text{Negcount}(A, \alpha_2) - \text{Negcount}(A, \alpha_1).$$

如果  $\alpha_2 - \alpha_1 < \text{tol}$  (其中  $\text{tol} \ll 1$  为事先给定的阈值), 且  $A$  在  $[\alpha_1, \alpha_2)$  中有特征值, 则我们可将  $[\alpha_1, \alpha_2)$  中的任意一个值作为  $A$  在该区间中的特征值的近似.

由此我们可以给出下面的对分法.

**算法 5.11.** 对分法: 计算  $A$  在  $[a, b)$  中的所有特征值

- 1: Let  $\text{tol}$  be a given threshold
- 2: compute  $n_a = \text{Negcount}(A, a)$
- 3: compute  $n_b = \text{Negcount}(A, b)$
- 4: **if**  $n_a = n_b$  **then**
- 5:     **return**     % 此时  $[a, b)$  中没有  $A$  的特征值

```

6: end if
7: put (a, na, b, nb) onto worklist
8: % worklist 中的元素是“四元素对”, 即由四个数组成的数对
9: while worklist not empty do
10:   remove (low, nlow, up, nup) from the worklist
11:   % (low, nlow, up, nup) 是 worklist 中的任意一个元素
12:   if (up - low) < tol then
13:     print "There are nup - nlow eigenvalues in [low, up]"
14:   else
15:     compute mid = (low + up)/2
16:     compute nmid = Negcount(A, mid)
17:     if (nmid > nlow) then
18:       put (low, nlow, mid, nmid) onto worklist
19:     end if
20:     if (nup > nmid) then
21:       put (mid, nmid, up, nup) onto worklist
22:     end if
23:   end if
24: end while

```

显然, 对分法的主要运算量集中在计算  $\text{Negcount}(A, z)$ . 通常是事先将  $A$  转化成对称三对角矩阵, 这样计算  $A - zI$  的  $\text{LDL}^T$  分解就非常简单:

$$\begin{aligned}
 A - zI &= \begin{bmatrix} a_1 - z & b_1 & & \\ b_1 & \ddots & \ddots & \\ & \ddots & \ddots & b_{n-1} \\ & & b_{n-1} & a_n - z \end{bmatrix} \\
 &= \begin{bmatrix} 1 & & & \\ l_1 & \ddots & & \\ & \ddots & \ddots & \\ & & l_{n-1} & 1 \end{bmatrix} \begin{bmatrix} d_1 & & & \\ & \ddots & & \\ & & \ddots & \\ & & & d_n \end{bmatrix} \begin{bmatrix} 1 & l_1 & & \\ & \ddots & \ddots & \\ & & \ddots & l_{n-1} \\ & & & 1 \end{bmatrix} \triangleq LDL^T.
 \end{aligned}$$

利用待定系数法, 可以得到下面的递推公式

$$d_1 = a_1 - z, \quad d_i = (a_i - z) - \frac{b_{i-1}^2}{d_{i-1}}, \quad i = 2, 3, \dots, n. \quad (5.9)$$

用上面的公式计算  $d_i$  的运算量约为  $4n$ .

注意这里没有选主元, 但针对对称三对角矩阵, 该算法是非常稳定的, 即使当  $d_i$  有可能很小时, 算法依然很稳定 [11].



✎ 由于单独调用一次 Negcount 的运算量为  $4n$ , 故计算  $k$  个特征值的总运算量约为  $O(kn)$ .

当特征值计算出来后, 我们可以使用带位移的逆迭代来计算对应的特征向量. 通常只需迭代 1 至 2 次即可, 由于  $A$  是三对角矩阵, 故计算每个特征向量的运算量为  $O(n)$ . 整个合起来就构成**对分法和逆迭代**.

当特征值紧靠在一起时, 计算出来的特征向量可能会失去正交性, 此时需要进行**再正交化**, 可通过 MGS 的 QR 分解来实现.

### 5.3 计算奇异值分解\*

奇异值分解 (SVD) 具有十分广泛的应用背景, 因此, 如何更好更快地计算一个给定矩阵的 SVD 是科学与工程计算领域中的一个热门研究课题, 吸引了众多专家进行这方面的研究, 也涌现出了许多奇妙的方法.

对任意矩阵  $A \in \mathbb{R}^{m \times n}$ , 其奇异值与对称矩阵  $A^T A$ ,  $AA^T$  和  $\begin{bmatrix} 0 & A^T \\ A & 0 \end{bmatrix}$  的特征值是密切相关的, 故理论上计算对称特征值的算法都可以用于计算奇异值. 但在实际计算中, 我们还可以通过利用 SVD 的特殊结构使得算法更加有效和准确.

与计算对称矩阵的特征值类似, 计算一个矩阵  $A$  的奇异值分解的算法通常分为以下几个步骤 (Jacobi 算法除外):

1. 将  $A$  二对角化:  $B = U_1^T A V_1$ , 其中  $B$  为上二对角矩阵,  $U_1, V_1$  为正交阵;
2. 计算  $B$  的 SVD:  $B = U_2 \Sigma V_2^T$ , 其中  $\Sigma$  为对角阵,  $U_2, V_2$  为正交阵;
3. 合并得到  $A$  的 SVD:  $A = U_1 B V_1^T = (U_1 U_2) B (V_1 V_2)^T$ .

#### 拓展阅读


当前 SVD 在科学与工程计算中有着非常重要的应用, 但 1976 年前却少有人知道和使用 SVD. MATLAB 软件的开发者和 MathWorks 公司的创始人 Cleve Moler 于 1976 年录制一个大约 6 分钟的视频, 主题是 “Matrices and Their Singular Values”, 非常形象地介绍了 SVD 及其应用. 视频地址是 <https://www.youtube.com/watch?v=R9UoFyqJca8>, 相关背景介绍可以参见 1976 Matrix Singular Value Decomposition Film.

#### 5.3.1 二对角化

我们知道, 对称矩阵可以通过一系列 Householder 变换转化为对称三对角矩阵. 对于一般矩阵  $A \in \mathbb{R}^{m \times n}$ , 我们也可以通过 Householder 变换, 将其转化为二对角矩阵, 即计算正交矩阵  $U_1$  和  $V_1$  使得

$$U_1^T A V_1 = B, \quad (5.10)$$

其中  $B$  是一个实 (上) 二对角矩阵. 这个过程就称为**二对角化**.

 需要注意的是, 与对称矩阵的对称三对角化不同,  $A$  与  $B$  是不相似的.

设  $A \in \mathbb{R}^{m \times n}$ , 二对角化过程大致如下:

- (1) 首先确定一个 Householder 矩阵  $H_1 \in \mathbb{R}^{m \times m}$ , 使得  $H_1 A$  的第一列除第一个元素外, 其它分量都为 0, 即

$$H_1 A = \begin{bmatrix} * & * & * & \cdots & * \\ 0 & * & * & \cdots & * \\ 0 & * & * & \cdots & * \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & * & * & \cdots & * \end{bmatrix}.$$





- (2) 再确定一个 Household 矩阵  $\tilde{H}_1 \in \mathbb{R}^{(n-1) \times (n-1)}$ , 把  $H_1 A$  的第一行的第 3 至第  $n$  个元素化为零, 即


$$H_1 A \begin{bmatrix} 1 & 0 \\ 0 & \tilde{H}_1 \end{bmatrix} = \begin{bmatrix} * & * & 0 & \cdots & 0 \\ 0 & * & * & \cdots & * \\ 0 & * & * & \cdots & * \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & * & * & \cdots & * \end{bmatrix}.$$

- (3) 重复上面的过程, 直到把  $A$  最终化为二对角矩阵.

有了分解 (5.10) 以后, 我们可得

$$A^T A = (U_1 B V_1^T)^T U_1 B V_1^T = V_1 B^T B V_1^T,$$

即  $V_1^T A^T A V_1 = B^T B$ . 由于  $B^T B$  是对称三对角的, 所以这就相当于将  $A^T A$  三对角化.

 整个二对角化过程的运算量约为  $4mn^2 + 4m^2n - 4n^3/3$ , 若不需要计算  $U_1$  和  $V_1$ , 则运算量约为  $4mn^2 - 4n^3/3$ .

### 二对角矩阵的奇异值分解

设  $B \in \mathbb{R}^{n \times n}$  是一个二对角矩阵

$$B = \begin{bmatrix} a_1 & b_1 & & & \\ & \ddots & \ddots & & \\ & & \ddots & b_{n-1} & \\ & & & a_n & \end{bmatrix}. \quad (5.11)$$

我们假定二对角矩阵的元素都非负, 即  $a_k, b_k \geq 0$ , 否则可以通过在左右分别乘以一个正交矩阵来实现 (见习题 5.16). 下面三种方法均可将计算  $B$  的 SVD 转化成计算对称三对角矩阵的特征分解:

- (1) 构造**增广矩阵**  $A = \begin{bmatrix} 0 & B^T \\ B & 0 \end{bmatrix}$ , 置换阵  $P = [e_1, e_{n+1}, e_2, e_{n+2}, \dots, e_n, e_{2n}]$ , 则  $T_{ps} = P^T A P$  是对称三对角矩阵, 且  $T_{ps}$  的主对角线元素全为 0, 次对角线元素为  $a_1, b_1, a_2, b_2, \dots, a_{n-1}, b_{n-1}, a_n$ . 若  $(\lambda_i, x_i)$  是  $T_{ps}$  的一个特征对, 则

$$\lambda_i = \pm \sigma_i, \quad P x_i = \frac{1}{\sqrt{2}} \begin{bmatrix} v_i \\ \pm u_i \end{bmatrix},$$

其中  $\sigma_i$  为  $B$  一个奇异值,  $u_i$  和  $v_i$  分别为对应的左和右奇异向量.

- (2) 令  $T_{BB^T} = B B^T$ , 则


$$T_{BB^T} = \begin{bmatrix} a_1^2 + b_1^2 & a_2 b_1 & & & \\ a_2 b_1 & \ddots & & \ddots & \\ & \ddots & a_{n-1}^2 + b_{n-1}^2 & a_n b_{n-1} & \\ & & a_n b_{n-1} & a_n^2 & \end{bmatrix}.$$

$T_{BB^T}$  的特征值为  $B$  的奇异值的平方, 且  $T_{BB^T}$  的特征向量为  $B$  的左奇异向量.

(3) 令  $T_{B^T B} = B^T B$ , 则

$$T_{B^T B} = \begin{bmatrix} a_1^2 & a_1 b_1 & & \\ a_1 b_1 & a_2^2 + b_1^2 & \ddots & \\ & \ddots & \ddots & a_{n-1} b_{n-1} \\ & & a_{n-1} b_{n-1} & a_n^2 + b_{n-1}^2 \end{bmatrix}$$

$T_{B^T B}$  的特征值为  $B$  的奇异值的平方, 且  $T_{B^T B}$  的特征向量为  $B$  的右奇异向量.

 Golub & Kahan (1965) 采用前面两种做法. Reinsch (1970) 采用第三种做法, 其巧妙之处是, 只需对  $B$  做 Givens 变换即可, 类似隐式 QR 迭代.

理论上, 我们可以直接使用 QR 迭代、分而治之法或带反迭代的对分法, 计算三对角矩阵  $T_{ps}$ ,  $T_{BB^T}$  和  $T_{B^T B}$  的特征值和特征向量. 但一般来说, 这种做法并不是最佳的, 原因如下:

- (1) 对  $T_{ps}$  做 QR 迭代并不划算, 因为 QR 迭代计算所有的特征值和特征向量, 而事实上只要计算正的特征值即可;
- (2) 直接构成  $T_{BB^T}$  或  $T_{B^T B}$  是数值不稳定的. 事实上, 这样做可能会使得  $B$  的小奇异值的精度丢失一半.

下面是一些计算奇异值分解的比较实用的算法.

1. **Golub-Kahan SVD 算法**: 由 Golub 和 Kahan [20] 于 1965 年提出, 是一种十分稳定且高效的计算 SVD 的算法, 也是最早的实用 SVD 算法. 主要思想是将带位移的对称 QR 迭代算法隐式地用到  $B^T B$  上, 在该算法中, 并不需要显示地把  $B^T B$  计算出来. 该算法也通常就称为 SVD 算法, 是一个基本且实用的算法, 目前仍然是计算小规模矩阵奇异值分解的常用算法. 关于这个算法的详细描述, 可参见 [21, 68].
2. **dqds 算法**: 由 Fernando 和 Parlett [16] 于 1994 年提出, 是计算二对角矩阵所有奇异值的最快算法, 而且能达到很高的相对精度, 包括奇异值很小的情形. 该算法主要基于对  $B^T B$  的 Cholesky 迭代, 可以看作是 LR 迭代算法的改进. 由于 LR 迭代算法在一定条件下与对称 QR 算法是等价的, 因此该算法也可以看作是 QR 迭代的变形.
3. **分而治之法**: 该算法是计算维数  $n \geq 25$  的矩阵的所有奇异值和奇异向量的最快算法, 但不能保证小奇异值的相对精度, 即  $\sigma_i$  的相对精度为  $O(\varepsilon)\sigma_1$ , 而不是  $O(\varepsilon)\sigma_i$ .
4. **对分法和反迭代**: 主要用于计算某个区间内的奇异值及对应的奇异向量, 能保证较高的相对精度.
5. **Jacobi 迭代**: 可隐式地对  $AA^T$  或  $A^T A$  实施对称 Jacobi 迭代, 能保证较高的相对精度. 最近, Z. Drmač 和 K. Veselić [13, 14] 改进了最初的 Jacobi 算法, 使其变成一个速度快、精度高的实用算法.

在这里, 我们介绍 Golub-Kahan SVD 算法, dqds 算法和 Jacobi 迭代.

### 5.3.2 Golub-Kahan SVD 算法

该算法主要思想是将带位移的对称 QR 迭代算法隐式地用到  $B^T B$  上, 而无需将  $B^T B$  显式地计算出来. Golub-Kahan SVD 算法有时也简称 SVD 算法, 其基本框架是:



- 将矩阵  $A$  二对角化, 得到上二对角矩阵  $B$ ;
- 用隐式 QR 迭代计算  $B^T B$  的特征值分解, 即

$$B^T B = Q \Lambda Q^T, \quad \Lambda = \text{diag}(\sigma_1^2, \sigma_2^2, \dots, \sigma_n^2). \quad (5.12)$$

- 计算  $BQ$  的列主元 QR 分解, 即

$$(BQ)P = UR, \quad (5.13)$$

其中  $P$  是置换矩阵,  $U$  是正交矩阵,  $R$  是上三角矩阵.

由 (5.12) 可知

$$(BQ)^T BQ = \Lambda,$$

因此  $BQ$  是列正交矩阵 (但不是单位列正交). 再由 (5.13) 可知  $R = U^T(BQ)P$  也是列正交矩阵. 又  $R$  是上三角矩阵, 所以  $R$  必定是对角矩阵. 令  $V = QP$ , 则由 (5.13) 可知

$$U^T B V = R.$$

这就是二对角矩阵  $B$  的奇异值分解.

算法的具体实现可参见 [21, 68].

### 5.3.3 dqds 算法

我们首先介绍针对实对称正定矩阵的 LR 算法, 该算法思想与 QR 迭代算法类似, 但提出时间更早.

#### 算法 5.12. 带位移的 LR 算法

```

1: Let  $T_0$  be a given real symmetric positive definite matrix
2: set  $i = 0$ 
3: while not converge do
4:   choose a shift  $\tau_i^2$  satisfying  $\tau_i^2 < \min\{\lambda(T_i)\}$ 
5:   compute  $B_i$  such that  $T_i - \tau_i^2 I = B_i^T B_i$    % Cholesky factorization
6:    $T_{i+1} = B_i B_i^T + \tau_i^2 I$ 
7:    $i = i + 1$ 
8: end while

```

LR 迭代算法在形式上与 QR 迭代算法非常类似. 事实上, 对于不带位移的 LR 迭代算法, 我们可以证明, 两步 LR 迭代等价于一步 QR 迭代.

**引理 5.15** 设  $\tilde{T}$  是不带位移的 LR 算法迭代两步后生成的矩阵,  $\hat{T}$  是不带位移的 QR 算法迭代一步后生成的矩阵, 则  $\tilde{T} = \hat{T}$ .

- 🔥 (1) LR 算法中要求  $T_0$  对称正定, 但并不一定是三对角矩阵;  
 (2) 由该引理可知, QR 算法与 LR 算法有相同的收敛性.

下面我们介绍 dqds (differential quotient difference with shifts) 算法. 该算法是针对三对角的对称正定矩阵  $B^T B$ , 其中  $B$  是二对角矩阵. 在数学上, dqds 算法与 LR 算法是等价的, 但在该算法中, 我们是直接通过  $B_i$  来计算  $B_{i+1}$ , 从而避免计算中间矩阵  $T_{i+1}$ , 这样也就尽可能地避免了由于计算  $B_i B_i^T$  而可能带来的数值不稳定性.

下面推导如何从  $B_i$  直接计算  $B_{i+1}$ . 设

$$B_i = \begin{bmatrix} a_1 & b_1 & & \\ & a_2 & \ddots & \\ & & \ddots & b_{n-1} \\ & & & a_n \end{bmatrix}, \quad B_{i+1} = \begin{bmatrix} \tilde{a}_1 & \tilde{b}_1 & & \\ & \tilde{a}_2 & \ddots & \\ & & \ddots & \tilde{b}_{n-1} \\ & & & \tilde{a}_n \end{bmatrix}.$$

为了书写方便, 我们记  $b_0 = b_n = \tilde{b}_0 = \tilde{b}_n = 0$ . 由 LR 算法 5.12 可知

$$B_{i+1}^T B_{i+1} + \tau_{i+1}^2 I = B_i B_i^T + \tau_i^2 I.$$

比较等式两边矩阵的对角线和上对角线元素, 可得

$$\begin{aligned} \tilde{a}_k^2 + \tilde{b}_{k-1}^2 + \tau_{i+1}^2 &= a_k^2 + b_k^2 + \tau_i^2, \quad k = 1, 2, \dots, n \\ \tilde{a}_k \tilde{b}_k &= a_{k+1} b_k \quad \text{或} \quad \tilde{a}_k^2 \tilde{b}_k^2 = a_{k+1}^2 b_k^2, \quad k = 1, 2, \dots, n-1. \end{aligned}$$

记  $\delta = \tau_{i+1}^2 - \tau_i^2$ ,  $p_k = a_k^2$ ,  $q_k = b_k^2$ ,  $\tilde{p}_k = \tilde{a}_k^2$ ,  $\tilde{q}_k = \tilde{b}_k^2$ , 则可得下面的 **qds 算法**.

#### 算法 5.13. qds 算法的单步 ( $B_i \rightarrow B_{i+1}$ )

- 1:  $\delta = \tau_{i+1}^2 - \tau_i^2$
- 2: **for**  $k = 1$  to  $n - 1$  **do**
- 3:    $\tilde{p}_k = p_k + q_k - \tilde{q}_{k-1} - \delta$
- 4:    $\tilde{q}_k = q_k \cdot (p_{k+1} / \tilde{p}_k)$
- 5: **end for**
- 6:  $\tilde{p}_n = p_n - \tilde{q}_{n-1} - \delta$

qds 算法中的每个循环仅需 5 个浮点运算, 所以运算量较少.

为了提高算法的精确性, 我们引入一个辅助变量  $d_k \triangleq p_k - \tilde{q}_{k-1} - \delta$ , 则

$$\begin{aligned} d_k &= p_k - \tilde{q}_{k-1} - \delta \\ &= p_k - \frac{q_{k-1} p_k}{\tilde{p}_{k-1}} - \delta \\ &= p_k \cdot \frac{\tilde{p}_{k-1} - q_{k-1}}{\tilde{p}_{k-1}} - \delta \\ &= p_k \cdot \frac{p_{k-1} - \tilde{q}_{k-2} - \delta}{\tilde{p}_{k-1}} - \delta \\ &= \frac{p_k}{\tilde{p}_{k-1}} \cdot d_{k-1} - \delta. \end{aligned}$$



于是就可得到 **dqds 算法**:

**算法 5.14.** dqds 算法的单步 ( $B_i \rightarrow B_{i+1}$ )

```

1:  $\delta = \tau_{i+1}^2 - \tau_i^2$ 
2:  $d_1 = p_1 - \delta$ 
3: for  $k = 1$  to  $n - 1$  do
4:    $\tilde{p}_k = d_k + q_k$ 
5:    $t = p_{k+1} / \tilde{p}_k$ 
6:    $\tilde{q}_k = q_k \cdot t$ 
7:    $d_{k+1} = d_k \cdot t - \delta$ 
8: end for
9:  $\tilde{p}_n = d_n$ 

```

dqds 算法的运算量与 dqs 差不多, 但更精确. 关于 dqds 算法中位移的选取, 以及如何判断收敛性, 可以参见 [16].

🔗 dqds 算法是求解二对角矩阵所有奇异值的主要方法, 2000 年就被加入到 LAPACK 中 [42]. 关于 dqds 的最新改进可以参见 [37].

### 5.3.4 Jacobi 算法

本节讨论对矩阵  $M = A^T A$  实施隐式的 Jacobi 算法来计算  $A$  的奇异值.

我们知道, Jacobi 算法的每一步就是对矩阵作 Jacobi 旋转, 即  $A^T A \rightarrow J^T A^T A J$ , 其中  $J$  的选取将某两个非对角元化为 0. 在实际计算中, 我们只需计算  $AJ$ , 故该算法称为**单边 Jacobi 旋转**.

**算法 5.15.** 单边 Jacobi 旋转的单步

```

% 对  $M = A^T A$  作 Jacobi 旋转, 将  $M(i, j), M(j, i)$  化为 0
1: Compute  $m_{ii} = (A^T A)_{ii}$ ,  $m_{ij} = (A^T A)_{ij}$ ,  $m_{jj} = (A^T A)_{jj}$ 
2: if  $m_{ij}$  is not small enough then
3:    $\tau = (m_{ii} - m_{jj}) / (2 \cdot m_{ij})$ 
4:    $t = \text{sign}(\tau) / (|\tau| + \sqrt{1 + \tau^2})$ 
5:    $c = 1 / \sqrt{1 + t^2}$ 
6:    $s = c \cdot t$ 
7:    $A = AG(i, j, \theta)$  %  $G(i, j, \theta)$  为 Givens 变换
8:   if eigenvectors are desired then
9:      $J = J \cdot G(i, j, \theta)$ 

```

```
10:   end if
11: end if
```

在上面算法的基础上, 我们可以给出完整的单边 Jacobi 算法.

**算法 5.16.** 单边 Jacobi: 计算  $A = U\Sigma V^T$

```
1: while  $A^T A$  is not diagonal enough do
2:   for  $i = 1$  to  $n - 1$  do
3:     for  $j = i + 1$  to  $n$  do
4:       调用单边 Jacobi 旋转
5:     end for
6:   end for
7: end while
8: compute  $\sigma_i = \|A(:, i)\|_2, i = 1, 2, \dots, n$ 
9:  $U = [u_1, \dots, u_n]$  with  $u_i = A(:, i)/\sigma_i$ 
10:  $V = J$ 
```

Jacobi 算法的特点:

- 不需要双对角化, 这样可以避免双对角化引入的误差;
- 可达到相对较高的计算精度;
- **速度较慢**. (目前已有快速的改进算法, 参见 [13, 14])



## 5.4 扰动分析

设  $A \in \mathbb{R}^{n \times n}$  是对称矩阵, 则有下面的谱分解.

**定理 5.16** 设  $A \in \mathbb{R}^{n \times n}$  是对称矩阵. 则存在一个正交矩阵  $Q$  使得

$$A = Q\Lambda Q^T$$

其中  $\Lambda = \text{diag}(\lambda_1, \lambda_2, \dots, \lambda_n)$  是一个实对角矩阵.

这里的  $\lambda_i$  就是  $A$  的特征值, 我们假设  $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_n$ . 令  $Q = [q_1, q_2, \dots, q_n]$ , 则  $q_i$  就是  $\lambda_i$  对应的单位正交特征向量.

关于对称矩阵特征值问题的扰动理论, 这里只做一些简单介绍, 若要深入了解这方面的信息, 可以参考 [31, 32, 49, 65].

### 5.4.1 特征值与 Rayleigh 商

**定义 5.2** 设  $A \in \mathbb{R}^{n \times n}$  是对称矩阵, 向量  $x \in \mathbb{R}^n$  非零, 则  $x$  关于  $A$  的 Rayleigh 商定义为:

$$\rho(x, A) = \frac{x^T A x}{x^T x}.$$

有时简记为  $\rho(x)$ .

下面是关于 Rayleigh 商的一些基本性质:

- (1)  $\rho(\alpha x) = \rho(x), \forall \alpha \in \mathbb{R}, \alpha \neq 0$ ;
- (2)  $\rho(q_i) = \lambda_i, i = 1, 2, \dots, n$ ;
- (3) 设  $x = \alpha_1 q_1 + \alpha_2 q_2 + \dots + \alpha_n q_n$ , 则

$$\rho(x) = \frac{\alpha_1^2 \lambda_1 + \alpha_2^2 \lambda_2 + \dots + \alpha_n^2 \lambda_n}{\alpha_1^2 + \alpha_2^2 + \dots + \alpha_n^2};$$

- (4)  $\lambda_n \leq \rho(x) \leq \lambda_1, |\rho(x)| \leq \|A\|_2$ .

实对称矩阵的特征值与 Rayleigh 商之间的一个基本性质是 Courant-Fischer 极小极大定理.

**定理 5.17 (Courant-Fischer)** 设  $A \in \mathbb{R}^{n \times n}$  是对称矩阵, 其特征值为  $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_n$ , 则有

$$\lambda_k = \max_{\mathbb{U} \in \mathbb{S}_k^n} \min_{x \in \mathbb{U}, x \neq 0} \frac{x^T A x}{x^T x} = \min_{\mathbb{V} \in \mathbb{S}_{n-k+1}^n} \max_{x \in \mathbb{V}, x \neq 0} \frac{x^T A x}{x^T x},$$

其中  $\mathbb{S}_i^n$  表示  $\mathbb{R}^n$  中所有  $i$  维子空间构成的集合. 当

$$\mathbb{U} = \text{span}\{q_1, \dots, q_k\}, \quad \mathbb{V} = \text{span}\{q_k, \dots, q_n\}, \quad x = q_k$$

时, 上式中的等号成立.

**证明.** 设  $\mathbb{U} \in \mathbb{S}_k^n$  和  $\mathbb{V} \in \mathbb{S}_{n-k+1}^n$  分别为  $\mathbb{R}^n$  中任意的  $k$  和  $n-k+1$  维子空间. 由于

$$\dim \mathbb{U} + \dim \mathbb{V} = n + 1 > n,$$

可得

$$\mathbb{U} \cap \mathbb{V} \neq \{0\}.$$

故存在非零向量  $\tilde{x} \in \mathbb{U} \cap \mathbb{V}$ . 所以有

$$\min_{x \in \mathbb{U}, x \neq 0} \rho(x) \leq \rho(\tilde{x}) \leq \max_{x \in \mathbb{V}, x \neq 0} \rho(x).$$

由  $\mathbb{U}$  和  $\mathbb{V}$  的任意性可知,

$$\max_{\mathbb{U} \in \mathbb{S}_k^n} \min_{x \in \mathbb{U}, x \neq 0} \rho(x) \leq \min_{\mathbb{V} \in \mathbb{S}_{n-k+1}^n} \max_{x \in \mathbb{V}, x \neq 0} \rho(x). \quad (5.14)$$

取  $\mathbb{U} = \text{span}\{q_1, \dots, q_k\}$ , 则  $\mathbb{U}$  中的任意向量都可写成  $x = \alpha_1 q_1 + \dots + \alpha_k q_k$ , 此时

$$\rho(x) = \frac{x^T A x}{x^T x} = \frac{\alpha_1^2 \lambda_1 + \dots + \alpha_k^2 \lambda_k}{\alpha_1^2 + \dots + \alpha_k^2} \geq \frac{\sum_{i=1}^k \alpha_i^2 \lambda_k}{\sum_{i=1}^k \alpha_i^2} = \lambda_k,$$

即

$$\max_{\mathbb{U} \in \mathbb{S}_k^n} \min_{x \in \mathbb{U}, x \neq 0} \rho(x) \geq \lambda_k. \quad (5.15)$$

同理, 取  $\mathbb{V} = \text{span}\{q_k, \dots, q_n\}$ , 则  $\mathbb{V}$  中的任意向量都可写成  $x = \alpha_k q_k + \dots + \alpha_n q_n$ , 此时

$$\rho(x) = \frac{x^T A x}{x^T x} = \frac{\alpha_k^2 \lambda_k + \dots + \alpha_n^2 \lambda_n}{\alpha_k^2 + \dots + \alpha_n^2} \leq \frac{\sum_{i=k}^n \alpha_i^2 \lambda_k}{\sum_{i=k}^n \alpha_i^2} = \lambda_k,$$

即

$$\min_{\mathbb{V} \in \mathbb{S}_{n-k+1}^n} \max_{x \in \mathbb{V}, x \neq 0} \rho(x) \leq \lambda_k. \quad (5.16)$$

由 (5.14), (5.15), (5.16) 可知, 定理结论成立.  $\square$

当  $k=1$  和  $k=n$  时, 就可以得到下面的定理.

**定理 5.18 (Rayleigh-Ritz)** 设  $A \in \mathbb{R}^{n \times n}$  是对称矩阵, 其特征值为  $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_n$ , 则有

$$\lambda_1 = \max_{x \in \mathbb{R}^n, x \neq 0} \frac{x^T A x}{x^T x}, \quad \lambda_n = \min_{x \in \mathbb{R}^n, x \neq 0} \frac{x^T A x}{x^T x}.$$

由极小极大定理, 我们可以得到下面的特征值分隔定理.

**定理 5.19 (分隔定理)** 设  $A \in \mathbb{R}^{n \times n}$  是对称矩阵,  $B = Q^T A Q$ , 其中  $Q \in \mathbb{R}^{n \times (n-1)}$  满足  $Q^T Q = I_{n-1}$ . 再设  $A$  和  $B$  的特征值分别为

$$\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_n \quad \text{和} \quad \tilde{\lambda}_1 \geq \tilde{\lambda}_2 \geq \dots \geq \tilde{\lambda}_{n-1},$$

则有

$$\lambda_1 \geq \tilde{\lambda}_1 \geq \lambda_2 \geq \tilde{\lambda}_2 \geq \dots \geq \tilde{\lambda}_{n-1} \geq \lambda_n.$$

特别地, 在定理 5.19 中, 取  $Q = [e_1, \dots, e_{i-1}, e_{i+1}, \dots, e_n]$ , 则可以得到下面的结论.

**推论 5.20** 设  $A \in \mathbb{R}^{n \times n}$  是对称矩阵,  $\tilde{A}$  是  $A$  的一个  $n-1$  阶主子矩阵,  $A$  和  $\tilde{A}$  的特征值分别为

$$\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_n \quad \text{和} \quad \tilde{\lambda}_1 \geq \tilde{\lambda}_2 \geq \dots \geq \tilde{\lambda}_{n-1},$$

则有

$$\lambda_1 \geq \tilde{\lambda}_1 \geq \lambda_2 \geq \tilde{\lambda}_2 \geq \dots \geq \tilde{\lambda}_{n-1} \geq \lambda_n.$$

反复应用上面的推论, 即可得到下面的结论.





**推论 5.21** 设  $A \in \mathbb{R}^{n \times n}$  是对称矩阵,  $\tilde{A}$  是  $A$  的一个  $k$  阶主子矩阵 ( $1 \leq k \leq n-1$ ),  $A$  和  $\tilde{A}$  的特征值分别为

$$\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n \quad \text{和} \quad \tilde{\lambda}_1 \geq \tilde{\lambda}_2 \geq \cdots \geq \tilde{\lambda}_k,$$

则有

$$\lambda_i \geq \tilde{\lambda}_i \geq \lambda_{n-k+i}, \quad i = 1, 2, \dots, k.$$

### 5.4.2 对称矩阵特征值的扰动分析

设  $A \in \mathbb{R}^{n \times n}$  是对称矩阵, 扰动矩阵  $E \in \mathbb{R}^{n \times n}$  也是对称矩阵, 下面讨论  $A + E$  的特征值与  $A$  的特征值之间的关系.

由极小极大定理, 我们可以证明下面的性质.

**定理 5.22** 设  $A \in \mathbb{R}^{n \times n}$  和  $B = A + E \in \mathbb{R}^{n \times n}$  都是对称矩阵, 其特征值分别为

$$\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n \quad \text{和} \quad \tilde{\lambda}_1 \geq \tilde{\lambda}_2 \geq \cdots \geq \tilde{\lambda}_n.$$

假定  $E$  的最大和最小特征值分别为  $\mu_1$  和  $\mu_n$ , 则有

$$\lambda_i + \mu_1 \geq \tilde{\lambda}_i \geq \lambda_i + \mu_n, \quad i = 1, 2, \dots, n.$$

**证明.** 由 Courant-Fischer 定理 5.17 和 Rayleigh-Ritz 定理 5.18 可知

$$\begin{aligned} \tilde{\lambda}_i &= \min_{V \in \mathbb{S}_{n-i+1}^n} \max_{x \in V, x \neq 0} \frac{x^T B x}{x^T x} \\ &= \min_{V \in \mathbb{S}_{n-i+1}^n} \max_{x \in V, x \neq 0} \left( \frac{x^T A x}{x^T x} + \frac{x^T E x}{x^T x} \right) \\ &\leq \min_{V \in \mathbb{S}_{n-i+1}^n} \max_{x \in V, x \neq 0} \left( \frac{x^T A x}{x^T x} + \mu_1 \right) \\ &= \min_{V \in \mathbb{S}_{n-i+1}^n} \max_{x \in V, x \neq 0} \frac{x^T A x}{x^T x} + \mu_1 \\ &= \lambda_i + \mu_1. \end{aligned}$$

同理可得

$$\begin{aligned} \tilde{\lambda}_i &= \max_{U \in \mathbb{S}_i^n} \min_{x \in U, x \neq 0} \frac{x^T B x}{x^T x} \\ &= \max_{U \in \mathbb{S}_i^n} \min_{x \in U, x \neq 0} \left( \frac{x^T A x}{x^T x} + \frac{x^T E x}{x^T x} \right) \\ &\geq \max_{U \in \mathbb{S}_i^n} \min_{x \in U, x \neq 0} \left( \frac{x^T A x}{x^T x} + \mu_n \right) \\ &= \max_{U \in \mathbb{S}_i^n} \min_{x \in U, x \neq 0} \frac{x^T A x}{x^T x} + \mu_n \\ &= \lambda_i + \mu_n. \end{aligned}$$

所以定理结论成立. □

根据这个定理, 我们立即可以得到下面的 Weyl 定理.



**定理 5.23 (Weyl)** 设  $A \in \mathbb{R}^{n \times n}$  和  $B = A + E \in \mathbb{R}^{n \times n}$  都是对称矩阵, 其特征值分别为  $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$  和  $\tilde{\lambda}_1 \geq \tilde{\lambda}_2 \geq \cdots \geq \tilde{\lambda}_n$ , 则

$$|\tilde{\lambda}_j - \lambda_j| \leq \|E\|_2, \quad j = 1, 2, \dots, n.$$

该定理的结论可以推广到奇异值情形. 我们首先给出下面的引理.

**引理 5.24** 设  $A \in \mathbb{R}^{m \times n}$  ( $m \geq n$ ) 的奇异值分解为  $A = U\Sigma V$ , 其中  $U = [u_1, \dots, u_n] \in \mathbb{R}^{m \times n}$  为列正交矩阵,  $V = [v_1, \dots, v_n] \in \mathbb{R}^{n \times n}$  为正交矩阵,  $\Sigma = \text{diag}(\sigma_1, \dots, \sigma_n)$ . 将  $U$  扩展成  $n \times n$  的正交矩阵  $[U, \tilde{U}] = [u_1, \dots, u_n, \tilde{u}_1, \dots, \tilde{u}_{m-n}]$ , 令

$$H = \begin{bmatrix} 0 & A^T \\ A & 0 \end{bmatrix} \in \mathbb{R}^{(m+n) \times (m+n)},$$

则  $H$  对称, 且特征值为  $\pm\sigma_i$  和 0 (其中 0 至少为  $m - n$  重特征值), 对应的特征向量分别为  $\frac{\sqrt{2}}{2} \begin{bmatrix} v_i \\ \pm u_i \end{bmatrix}$ ,  $i = 1, 2, \dots, n$ , 和  $\begin{bmatrix} 0 \\ \tilde{u}_j \end{bmatrix}$ ,  $j = 1, 2, \dots, m - n$ . (直接代入验证即可)

由上面的引理和 Weyl 定理 5.23 立即可得

**定理 5.25** 设  $A, B \in \mathbb{R}^{m \times n}$  ( $m \geq n$ ), 它们的奇异值分别为  $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n$  和  $\tilde{\sigma}_1 \geq \tilde{\sigma}_2 \geq \cdots \geq \tilde{\sigma}_n$ . 则

$$|\tilde{\sigma}_j - \sigma_j| \leq \|B - A\|_2, \quad j = 1, 2, \dots, n.$$

最后给出一个基于 F-范数的扰动性质.

**定理 5.26** 设  $A, E \in \mathbb{C}^{n \times n}$  且  $A$  是 Hermite 的,  $A + E$  是正规矩阵. 并设  $A$  的特征值满足

$$\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n,$$

$A + E$  的特征值  $\tilde{\lambda}_1, \tilde{\lambda}_2, \dots, \tilde{\lambda}_n$  满足

$$\text{Re}(\tilde{\lambda}_1) \geq \text{Re}(\tilde{\lambda}_2) \geq \cdots \geq \text{Re}(\tilde{\lambda}_n).$$

则

$$\sum_{i=1}^n |\tilde{\lambda}_i - \lambda_i|^2 \leq \|E\|_F^2.$$

(证明可参见 [31, page 370, Cor. 6.3.8])

### 5.4.3 Rayleigh 商逼近

**定理 5.27** 设对称矩阵  $A \in \mathbb{R}^{n \times n}$  的特征值为  $\lambda_1, \lambda_2, \dots, \lambda_n$ ,

(1) 若  $x \in \mathbb{R}^n$  是单位向量,  $\beta \in \mathbb{R}$ , 则

$$\min_{1 \leq i \leq n} |\lambda_i - \beta| \leq \|Ax - \beta x\|_2; \quad (5.17)$$



(2) 对于给定的非零向量  $x \in \mathbb{R}^n$ , 当  $\beta = \rho(x)$  时,  $\|Ax - \beta x\|_2$  达到最小, 即

$$\min_{\beta \in \mathbb{R}} \|Ax - \beta x\|_2 = \|Ax - \rho(x)x\|_2; \quad (5.18)$$

**证明.** (1) 若  $\beta$  是  $A$  的特征值, 则结论显然成立.

若  $\beta$  不是  $A$  的特征值, 则  $A - \beta I$  非奇异, 故

$$1 = \|x\|_2 = \|(A - \beta I)^{-1}(A - \beta I)x\|_2 \leq \|(A - \beta I)^{-1}\|_2 \cdot \|(A - \beta I)x\|_2. \quad (5.19)$$

由于  $A - \beta I$  对称, 且特征值为  $\lambda_i - \beta$ , 故

$$\|(A - \beta I)^{-1}\|_2 = \frac{1}{\min_{1 \leq i \leq n} |\lambda_i - \beta|}.$$

代入 (5.19) 即可知结论成立.


(2) 由于

$$x^T(Ax - \rho(x)x) = x^T Ax - \frac{x^T Ax}{x^T x} x^T x = 0,$$

即  $x \perp (Ax - \rho(x)x)$ . 所以

$$\begin{aligned} \|Ax - \beta x\|_2^2 &= \|(A - \rho(x))x + (\rho(x) - \beta)x\|_2^2 \\ &= \|Ax - \rho(x)x\|_2^2 + \|(\rho(x) - \beta)x\|_2^2 \\ &\geq \|Ax - \rho(x)x\|_2^2, \end{aligned}$$

所以当  $\beta = \rho(x)$  时,  $\|Ax - \beta x\|_2$  达到最小. □

 由 (5.17) 可知, 在幂迭代和反迭代中可以使用残量  $\|Ax - \tilde{\lambda}x\|_2 < tol$  作为停机准则, 这里  $\tilde{\lambda}$  是迭代过程中计算得到的近似特征值. 等式 (5.18) 则解释了为什么用 Rayleigh 商来近似特征值.

## 5.5 应用

### 5.5.1 多项式求根

考虑  $n$  次多项式

$$p_n(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0. \quad (5.20)$$

这里假定  $a_i$  都是实数, 且  $a_n \neq 0$ . 由代数学基本定理可知,  $p_n(x)$  在复数域中有且仅有  $n$  的零点 (其中重根按重数计).

当  $n = 1$  时, 可直接求解. 当  $n = 2$  时, 可以通过求根公式求解. 当  $n = 3$  时, 也存在相应的求根公式, 但不那么直观, 需要一定的构造技巧才能导出. 当  $n = 4$  时, 可以从三次方程求根公式中导出相应的求根公式.

而当  $n \geq 5$  时, Abel 证明了不存在求根公式, 这意味着只能用数值方法 (迭代法) 求解. 从理论上讲, 任何求解非线性方程的迭代法都可以用来计算多项式的零点, 如著名的 Newton 法. 计算出一个零点后, 通过收缩技术, 将原问题转化为  $n - 1$  多项式零点问题, 然后继续用迭代法求解. 如此不断重复, 直到求出所有的零点. 但由于舍入误差等原因, 对于高次多项式, 随着计算过程的推进, 计算误差会越来越大, 最后的计算结果往往无法令人满意.

在 MATLAB 中, 命令 `roots` 可以计算出多项式的所有零点, 其使用的方法就是计算矩阵特征值的 QR 迭代方法.

首先将多项式 (5.20) 转化为首项系数为 1 的多项式, 记为

$$q_n(x) = x^n + c_{n-1} x^{n-1} + \cdots + c_1 x + c_0.$$

多项式  $q_n(x)$  可以看作是某个  $n$  阶矩阵的特征值多项式, 如:

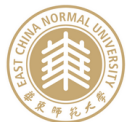
$$A = \begin{bmatrix} 0 & & & -c_0 \\ 1 & 0 & & -c_1 \\ & \ddots & \ddots & \vdots \\ & & 1 & -c_{n-1} \end{bmatrix}. \quad (5.21)$$

我们称矩阵  $A$  为  $q_n(x)$  的友矩阵. 这样, 计算多项式  $q_n(x)$  的零点问题就转化为计算  $A$  的特征值问题.

由于  $A$  已经是上 Hessenberg 矩阵, 因此隐式 QR 迭代方法的第一步 (上 Hessenberg 化) 就不用做了. 虽然  $A$  上三角部分大多是零, 而且分布也很有规律, 但无论是单位移 QR 迭代还是双位移 QR 迭代, 经过一步迭代后, 这些零元素都会消失, 因此总运算量仍然是  $O(n^3)$ .

于是, 如何利用  $A$  的特殊结构, 降低 QR 方法的运算量和存储量, 一直是颇受关注的问题. 最近, 陆续有学者 [1, 3, 8, 56] 提出了快速 QR 方法, 将运算量降为  $O(n^2)$ , 存储量也降为  $O(n)$ . 主要思想是将  $A$  写成一个酉矩阵与秩一矩阵之差:

$$A = \begin{bmatrix} 0 & & & 1 \\ 1 & 0 & & 0 \\ & \ddots & \ddots & \vdots \\ & & 1 & 0 \end{bmatrix} - \begin{bmatrix} c_0 + 1 \\ c_1 \\ \vdots \\ c_{n-1} \end{bmatrix} [0, 0, \dots, 1] \triangleq U - xy^T.$$



易知, 经过一次 QR 迭代后, 仍可以写成一个酉矩阵与秩一矩阵之差. 基于这种观察, 就可以设计出快速的 QR 迭代方法, 详细过程可参见 [1, 3, 8, 56] 或 [68].

### 5.5.2 Goolge 网页排名: PageRank

见课堂板书 ([To be continued ...](#))

## 5.6 课后习题

练习 5.1 设  $J = \begin{bmatrix} \lambda & 1 & & \\ & \ddots & \ddots & \\ & & \ddots & 1 \\ & & & \lambda \end{bmatrix} \in \mathbb{C}^{m \times m}$ , 计算其特征值  $\lambda$  对应的特征向量和左特征向量.

练习 5.2 设  $A \in \mathbb{C}^{n \times n}$ . 证明:

- (1) 若  $A$  是上三角矩阵, 且  $A^*A = AA^*$ , 则  $A$  必定是对角矩阵;
- (2)  $A$  是正规矩阵的充要条件是  $A$  酉相似于一个对角矩阵;
- (3) 设  $\lambda_1, \lambda_2, \dots, \lambda_n$  是  $A$  的特征值, 则  $A$  是正规矩阵的充要条件是  $\sum_{i=1}^n |\lambda_i|^2 = \|A\|_F^2$ .

练习 5.3 设  $\lambda_1, \lambda_2 \in \mathbb{C}$  是  $A \in \mathbb{C}^{n \times n}$  的两个互不相同的特征值,  $x \in \mathbb{C}^n$  是  $\lambda_1$  的特征向量,  $y \in \mathbb{C}^n$  是  $\lambda_2$  的左特征向量. 证明:  $y^*x = 0$ .

练习 5.4 设  $A \in \mathbb{C}^{m \times n}$ ,  $B \in \mathbb{C}^{n \times m}$ , 证明:  $AB$  与  $BA$  具有相同的非零特征值.

练习 5.5 设

$$A = \begin{bmatrix} A_{11} & A_{12} & \cdots & A_{1k} \\ & A_{22} & \cdots & A_{2k} \\ & & \ddots & \vdots \\ & & & A_{kk} \end{bmatrix},$$

其中  $A_{ii}$  都是方阵. 证明:  $A$  的特征值即为对角块  $A_{11}, A_{22}, \dots, A_{kk}$  的特征值的并.

练习 5.6 设  $H = [h_{ij}] \in \mathbb{R}^{n \times n}$  是上 Hessenberg 矩阵, 其 QR 分解为  $H = QR$ , 其中  $R = [r_{ij}] \in \mathbb{R}^{n \times n}$  是上三角矩阵且对角线元素均非负. 证明:

$$r_{kk} \geq |h_{k+1,k}|, \quad k = 1, 2, \dots, n-1.$$

因此, (1) 若  $H$  不可约, 则  $r_{kk} > 0, k = 1, 2, \dots, n-1$ ; (2) 若  $H$  不可约且奇异, 则  $r_{nn} = 0$ .

练习 5.7 证明定理 5.4:

设  $A \in \mathbb{R}^{n \times n}$  是非奇异上 Hessenberg 矩阵且下次对角线元素均非零, 即  $a_{i+1,i} \neq 0, i = 1, 2, \dots, n-1$ . 设其 QR 分解为  $A = QR$ , 则  $\tilde{A} \triangleq RQ$  的下次对角线元素也都非零.

练习 5.8 用 Householder 变换, 将矩阵  $A$  化为上 Hessenberg 型, 其中

$$A = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & -1 & -1 & 3 \\ 2 & -4 & 7 & 3 \\ 1 & 4 & -3 & 6 \end{bmatrix}.$$



## 练习 5.9 证明矩阵

$$A = \begin{bmatrix} 0 & 0 & \cdots & 0 & -c_0 \\ 1 & 0 & \cdots & 0 & -c_1 \\ 0 & 1 & \ddots & \vdots & \vdots \\ \vdots & \ddots & \ddots & 0 & -c_{n-2} \\ 0 & 0 & \cdots & 1 & -c_{n-1} \end{bmatrix}$$

的特征多项式是

$$p(\lambda) = \lambda^n + c_{n-1}\lambda^{n-1} + \cdots + c_1\lambda + c_0.$$

练习 5.10 (收缩 Deflation) 技巧. 设  $A \in \mathbb{R}^{n \times n}$ .

(1) 已知  $A$  的一对实特征值和实特征向量为  $(\lambda, x)$ . 试构造正交矩阵  $Q$ , 使得

$$Q^T A Q = \begin{bmatrix} \lambda_1 & A_{12} \\ 0 & A_{22} \end{bmatrix}, \quad \text{其中 } A_{22} \in \mathbb{R}^{(n-1) \times (n-1)}.$$

(2) 已知  $A$  的一对复共轭特征值  $\lambda$  和  $\bar{\lambda}$ , 以及相应的复特征向量  $x$  和  $\bar{x}$ . 试构造正交矩阵  $Q$ , 使得

$$Q^T A Q = \begin{bmatrix} B & A_{12} \\ 0 & A_{22} \end{bmatrix}, \quad \text{其中 } B \in \mathbb{R}^{2 \times 2}, A_{22} \in \mathbb{R}^{(n-2) \times (n-2)}.$$

练习 5.11 设  $x, y \in \mathbb{R}^n$ , 试证明:  $\det(I + xy^T) = 1 + y^T x$ . (注: 在复数域也成立)练习 5.12 设  $A = D + uu^T$ , 其中  $D = \text{diag}(d_1, d_2, \dots, d_n)$  满足  $d_1 \geq d_2 \geq \cdots \geq d_n$ ,  $u = [u_1, u_2, \dots, u_n]^T$ .

(1) 证明:  $d_i$  是  $A$  的特征值的充要条件是  $d_i = d_{i+1}$  或  $d_i = d_{i-1}$  或  $u_i = 0$ ;

(2) 若  $u_i = 0$ , 则  $e_i$  是与  $d_i$  对应的特征向量;

(3) 若  $d_{i-1} > d_i = d_{i+1} > d_{i+2}$  且  $u_i \neq 0$ , 证明: 对应于  $\lambda = d_i$  的特征向量  $x$  满足  $x_i u_i + x_{i+1} u_{i+1} = 0$ .

思考: 如果  $d_{i-1} > d_i = d_{i+1} = d_{i+2} > d_{i+3}$ , 则结论如何?

练习 5.13 设  $D = \text{diag}(d_1, d_2, \dots, d_n)$ . 若矩阵  $D + \alpha uu^T$  和  $D + \hat{u} \hat{u}^T$  具有相同的特征值, 记为  $\lambda_1, \lambda_2, \dots, \lambda_n$ , 且满足交错性质  $\lambda_1 > d_1 > \lambda_2 > d_2 > \cdots > \lambda_n > d_n$ , 则它们具有相同的特征向量.

思考: 如果没有交错性质, 则结论如何?

练习 5.14 设  $A \in \mathbb{C}^{n \times n}$  是 skew-Hermitian 矩阵, 即  $A^* = -A$ . 证明:

(1)  $A$  的非零特征值是纯虚数;

(2)  $I + A$  非奇异;

(3) 矩阵  $(I + A)^{-1}(I - A)$  是酉矩阵. (该矩阵称为  $A$  的 Cayley 变换)

练习 5.15 设  $B \in \mathbb{R}^{m \times n}$ ,  $m \geq n$  且  $\|B\|_2 < 1$ . 若  $A = \begin{bmatrix} I & B \\ B^T & I \end{bmatrix}$ , 证明:

$$\kappa_2(A) = \frac{1 + \|B\|_2}{1 - \|B\|_2}.$$



**练习 5.16** 设  $B$  是二对角矩阵, 其定义见 (5.11). 试证明: 存在正交矩阵  $Q_1$  和  $Q_2$ , 使得  $Q_1^T B Q_2$  的元素都非负.

**练习 5.17\*** (1) 设  $x \in \mathbb{R}^n$  是一个正向量, 即  $x_i > 0$ . 证明: 由  $x$  定义的 Cauchy 矩阵  $A$

$$a_{ij} = \frac{1}{x_i + x_j}$$

是对称半正定的. 进一步, 若  $x_i$  互不相等, 则  $A$  对称正定. (参见 [69])

(2) 证明: Hilbert 矩阵是对称正定的.

**练习 5.18\*** 设  $x, y \in \mathbb{R}^n$ , 若  $y^T x$  只有零特征值, 证明:  $xy^T$  也只有零特征值.

设  $X, Y \in \mathbb{R}^{n \times 2}$ , 若  $Y^T X$  只有零特征值, 则  $XY^T$  是否也只有零特征值?

**练习 5.19** 考虑用 Householder 变换将矩阵上 Hessenberg 化的算法, 给出具体的乘法运算次数和加减运算次数.

## 实践题

**练习 5.20** 编写 MATLAB 函数, 实现矩阵的上 Hessenberg 化, 即算法 5.6.

```
function [H,Q] = Hess(A)
```

**练习 5.21** 编写计算上 Hessenberg 矩阵  $H$  的带 Francis 位移的隐式 QR 方法, 并上机实现.

```
function D = Eig_QR_Hess(H)
```

**练习 5.22** 编写计算友矩阵 (5.21) 的带 Francis 位移的快速 QR 方法, 并上机实现.

```
function D = Eig_QR_Companion(H)
```

**练习 5.23** 编写程序, 实现对称矩阵  $A \in \mathbb{R}^{n \times n}$  的三对角化.

```
function [T,Q] = Tridiag(A)
```

**练习 5.24** 编写程序, 实现矩阵  $A \in \mathbb{R}^{m \times n}$  ( $m \geq n$ ) 的二对角化.

```
function [T,U,V] = Bidiag(A)
```

**练习 5.25** 编写程序, 实现计算对称三对角矩阵特征值的带 Wilkinson 位移的 QR 算法.

```
function D = Eig_QR(A)
```

**练习 5.26** 编写程序, 实现计算二对角矩阵奇异值的 dqds 算法.

```
function D = dqds(A)
```





# 6

## 非线性方程

考虑下面的方程

$$f(x) = 0.$$

如果  $f(x)$  是一次多项式, 则称为**线性方程**, 否则就称为**非线性方程**. 本章主要介绍求解非线性方程和非线性方程组的常见数值方法.

非线性方程有着非常广泛的实际应用背景, 比如在计算液体在管道中的摩擦系数时, 需要求解下面的方程 [19]


$$\frac{1}{\sqrt{x}} = \frac{1}{k} \ln(\text{Re} \cdot \sqrt{x}) + \left(14 - \frac{5.6}{k}\right),$$

其中  $k$  是某个常数,  $\text{Re}$  代表雷诺 (Reynolds) 数 (一种用来表征流体流动情况的无量纲数, 跟管道的直径, 流体的密度、粘度和流速有关). 显然想要给出  $x$  的解析表达式是不可能的, 只能通过数值求解.

再比如, 一类常见的重要非线性方程就是**代数方程**, 即

$$f(x) = a_0 + a_1x + a_2x^2 + \cdots + a_nx^n = 0,$$

其中  $a_0, a_1, \dots, a_n$  是给定的数. 由代数学基本定理可知, 代数方程在复数域中存在  $n$  个解 (如果有相同的则计算重数). 当  $n = 1, 2, 3, 4$  时, 存在相应的求根公式, 但当  $n \geq 5$  时, 不存在一般的求根公式, 此时只能通过数值方法求解.

 非线性方程 (组) 一般不存在直接解法, 通常需要使用迭代法来求数值解.

### 非线性方程 (组) 相关参考文献

- [1] J. M. Ortega and W. C. Rheinboldt, [Iterative Solution of Nonlinear Equations in Several Variables](#), 1970 [39]
- [2] J. E. Jr Dennis and R. B. Schnabel, [Numerical Methods for Unconstrained Optimization and Nonlinear Equations](#), 1983 [12]
- [3] C. T. Kelley, [Iterative Methods for Linear and Nonlinear Equations](#), 1995 [34]
- [4] C. T. Kelley, [Solving Nonlinear Equations with Newton's Method](#), 2003 [35]
- [5] 范金燕, 袁亚湘, [非线性方程组数值方法](#), 2018 [60]

## 6.1 对分法

首先介绍一些基本概念:

- **解, 根, 零点:** 所有满足  $f(x_*) = 0$  的数  $x_*$  都称为方程的解或根, 也称为  $f(x)$  的零点.
- **根的重数:** 若  $f(x) = (x - x_*)^m g(x)$  且  $g(x_*) \neq 0$ , 则  $x_*$  为  $f(x) = 0$  的  $m$  重根;
- **有解区间:** 若  $[a, b]$  内至少存在  $f(x) = 0$  的一个实数解, 则称  $[a, b]$  为有解区间.

我们研究内容就是在**有解**的前提下求出方程  $f(x) = 0$  的**近似根**.

如果没有特别说明, 本讲我们只考虑实数解, 并且假定  $f(x)$  的表达式中也只包含实数.

非线性方程求解通常比较困难, 而且可能存在多个或无穷多个解, 因此在求解时一般要强调 **求解区域**, 即计算指定区域内的解.

### 6.1.1 对分法基本思想

设  $f(x) = 0$  在区间  $[a, b]$  内至少有一个实数解. 对分法 (Bisection) 的基本思想就是把这个有解区间进行对分, 并找出解所在的小区间, 记为新的有解区间, 然后再对这个小区间进行对分. 依次类推, 直到有解区间的长度足够小为止, 此时有解区间内的任意一点都可以作为  $f(x) = 0$  的近似解. 在实际计算中, 通常取中点.

对分法的数学原理是下面的 **介值定理**.

**定理 6.1** 设  $f(x)$  在  $[a, b]$  内连续, 且  $f(a)f(b) < 0$ , 则在  $(a, b)$  内至少存在一点  $\xi$ , 使得  $f(\xi) = 0$ .

根据对分法的设计思路, 其计算过程可描述如下:

#### 算法 6.1. 对分法

- 1: 给定函数  $f(x)$  和求解区间  $[a, b]$ , 以及精度要求  $\varepsilon > 0$
- 2: 令  $a_1 = a, b_1 = b$
- 3: 计算  $f(a_1)$  和  $f(b_1)$ , 如果  $|f(a_1)| < \varepsilon$ , 则返回数值解  $x = a_1$  并停止计算; 如果  $|f(b_1)| < \varepsilon$ , 则返回数值解  $x = b_1$  并停止计算; 如果  $f(a_1)f(b_1) > 0$ , 则输出算法失败信息并停止计算
- 4: **for**  $k = 1, 2, \dots$ , **do**
- 5:   计算  $x_k = \frac{a_k + b_k}{2}$  和  $f(x_k)$
- 6:   如果  $|f(x_k)| < \varepsilon$  或者  $|b_k - a_k| < \varepsilon$ , 则返回数值解  $x_k$  并停止计算;
- 7:   如果  $f(a_k)f(x_k) < 0$ , 则令  $a_{k+1} = a_k, b_k = x_k$ ; 否则令  $a_{k+1} = x_k, b_{k+1} = b_k$ ;
- 8: **end for**

这里的采用的停机准则是函数值充分小或者求解区间充分小.

用对分法求解时, 可以先画出  $f(x)$  草图, 以确定一个大致的有解区间.



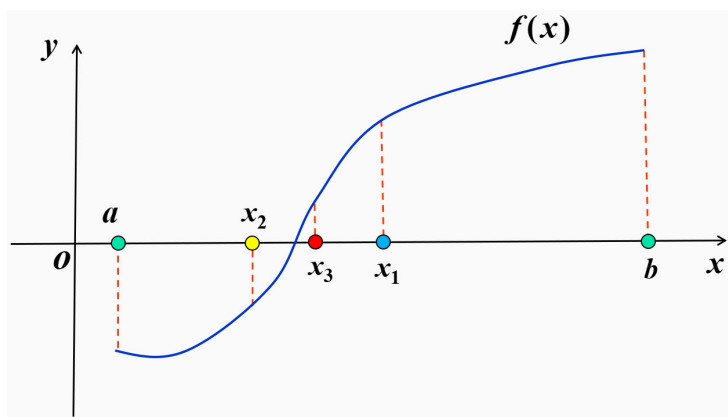


图 6.1. 对分法的几何表示

### 6.1.2 对分法的收敛性

我们对对分法的误差进行估计. 记算法在第  $k$  步时得到的有解区间为  $[a_k, b_k]$ , 中点为  $x_k$ . 则  $a_1 = a, b_1 = b$ , 且

$$|x_k - x_*| = \left| \frac{1}{2}(a_k + b_k) - x_* \right| \leq \frac{1}{2}(b_k - a_k).$$

由于每次都是对分有解区间, 因此有

$$b_k - a_k = \frac{1}{2}(b_{k-1} - a_{k-1}) = \frac{1}{2^2}(b_{k-2} - a_{k-2}) = \cdots = \frac{1}{2^{k-1}}(b_1 - a_1).$$

因此

$$|x_k - x_*| \leq \frac{1}{2^k}(b_1 - a_1) = \frac{1}{2^k}(b - a).$$

所以

$$\lim_{k \rightarrow \infty} |x_k - x_*| = 0,$$

即算法收敛. 因此我们就有下面的收敛性结论.

**定理 6.2** 设  $f(x) \in C[a, b]$ , 且  $f(a)f(b) < 0$ , 则对分法收敛到  $f(x) = 0$  的一个解.

#### 关于对分法的几点注记

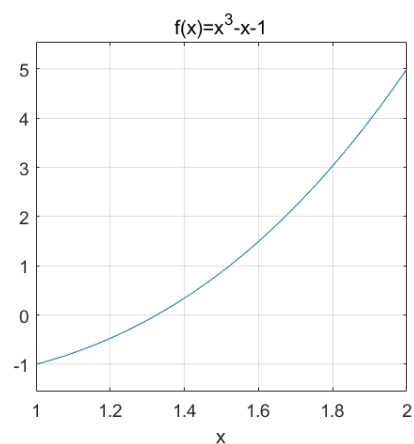
- 适用范围: 只适合求连续函数的 **单重实根**或**奇数重实根**;
- 优点: 简单易用, 只要满足介值定理的条件, 算法总是收敛的;
- 缺点: (1) 收敛速度较缓慢; (2) 不能求复根和偶数重根; (3) 一次只能求一个根;
- 总结: 一般可先用来计算解的一个粗糙估计, 然后再用其他方法进行加速, 如 Newton 法.

**例 6.1** 用对分法求  $f(x) = x^3 - x - 1 = 0$  在  $[1, 2]$  内的根.

(NLS\_bisection.m)

**解.** 易知  $f(1) = -1 < 0, f(2) = 5 > 0$ , 所以  $f(x)$  在  $[1, 2]$  内存在零点 (见下图). 下表中列出了对分法迭代 10 步的数值结果, 其中  $f(a)$  和  $f(b)$  只标出其正负号. 由表中可以看出, 迭代 10 步后得到的近似解为  $x_* \approx 1.3252$ .

$k$	$a/f(a)$	$b/f(b)$	$x$	$f(x)$
1	1.0000/-	2.0000/+	1.5000	0.8750
2	1.0000/-	1.5000/+	1.2500	-0.2969
3	1.2500/-	1.5000/+	1.3750	0.2246
4	1.2500/-	1.3750/+	1.3125	-0.0515
5	1.3125/-	1.3750/+	1.3438	0.0826
6	1.3125/-	1.3438/+	1.3281	0.0146
7	1.3125/-	1.3281/+	1.3203	-0.0187
8	1.3203/-	1.3281/+	1.3242	-0.0021
9	1.3242/-	1.3281/+	1.3262	0.0062
10	1.3242/-	1.3262/+	1.3252	0.0020



□



## 6.2 不动点迭代法

不动点迭代的基本思想是将原方程  $f(x) = 0$  改写成一个等价的方程  $\varphi(x) - x = 0$  或者  $x = \varphi(x)$ , 然后就可以根据这个等价方程构造出一个迭代格式:

$$x_{k+1} = \varphi(x_k), \quad k = 0, 1, 2, \dots, \quad (6.1)$$

其中  $x_0$  为迭代初始值, 可以任意选取. 这就是 **不动点迭代法** (Fixed-Point iteration), 简称**迭代法**,  $\varphi(x)$  称为 **迭代函数**.

由于方程  $f(x) = 0$  和  $x = \varphi(x)$  是等价的, 因此  $x_*$  是  $f(x) = 0$  的解当且仅当  $x_* = \varphi(x_*)$ , 即它是  $\varphi(x)$  的一个不动点.

- 🔍 不动点迭代的一个非常重要的特征是将方程求解转化为函数求值, 后者显然要容易很多.
- 🔍 不动点的一个几何含义是曲线  $y = \varphi(x)$  与直线  $y = x$  的交点, 因此不动点迭代过程可以用几何图像来表示.

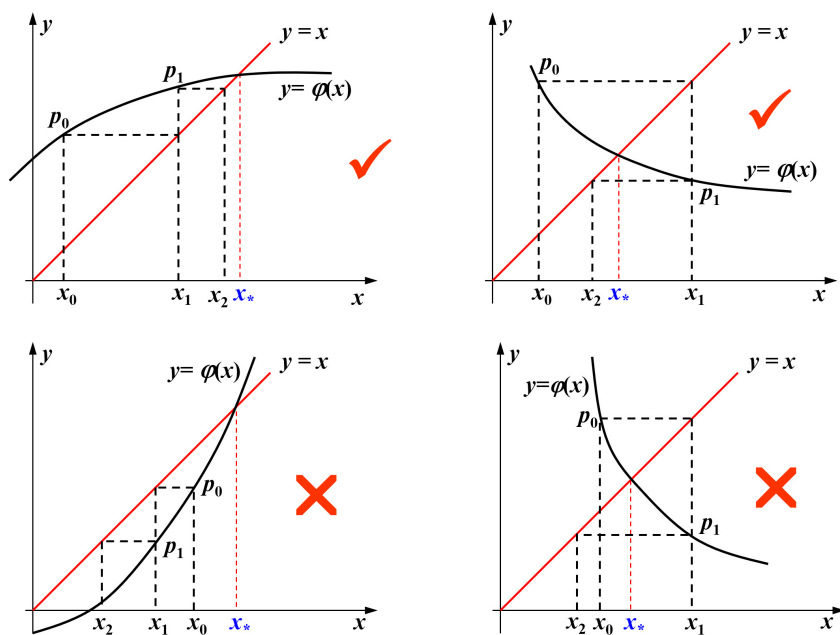


图 6.2. 不同迭代函数的不动点迭代过程几何表示

### 6.2.1 收敛性分析

设  $\varphi(x)$  连续, 迭代 (6.1) 生成的点列为  $x_0, x_1, x_2, \dots, x_k, \dots$ , 如果存在  $x_*$  使得

$$\lim_{k \rightarrow \infty} x_k = x_*,$$

则由连续函数的性质可知

$$x_* = \lim_{k \rightarrow \infty} x_{k+1} = \lim_{k \rightarrow \infty} \varphi(x_k) = \varphi\left(\lim_{k \rightarrow \infty} x_k\right) = \varphi(x_*).$$

因此  $x_*$  是  $\varphi(x)$  的一个不动点, 此时我们称迭代法 **收敛**. 如果点列  $\{x_k\}_{k=0}^{\infty}$  不收敛, 则不动点迭代 (6.1) 是 **发散** 的.

## 全局收敛性

**定理 6.3 (不动点的存在唯一性)** 设  $\varphi(x) \in C[a, b]$  且满足

- (1) 对任意  $x \in [a, b]$ , 都有  $\varphi(x) \in [a, b]$ ,
- (2) 存在常数  $L$ , 满足  $0 < L < 1$ , 使得对任意  $x, y \in [a, b]$  都有

$$|\varphi(x) - \varphi(y)| \leq L|x - y|,$$

则  $\varphi(x)$  在  $[a, b]$  上存在唯一的不动点.

**证明.** 首先证明存在性. 若  $\varphi(a) = a$  或  $\varphi(b) = b$ , 则结论成立. 下面假定  $\varphi(a) \neq a$  且  $\varphi(b) \neq b$ . 则由条件 (1) 可知

$$\varphi(a) > a, \quad \varphi(b) < b.$$

构造函数  $g(x) \triangleq \varphi(x) - x$ , 则

$$g(a)g(b) = (\varphi(a) - a)(\varphi(b) - b) < 0.$$

根据介值定理, 存在  $x_* \in (a, b)$  使得  $g(x_*) = 0$ , 即  $\varphi(x_*) = x_*$ , 所以  $\varphi(x)$  在  $[a, b]$  内存在不动点.

下面证明唯一性. 假设  $\varphi(x)$  在  $[a, b]$  内有两个不动点, 分别设为  $x_1$  和  $x_2$ , 且  $x_1 \neq x_2$ , 则

$$|\varphi(x_1) - \varphi(x_2)| = |x_1 - x_2|.$$

又根据条件 (2) 可知

$$|\varphi(x_1) - \varphi(x_2)| \leq L|x_1 - x_2| < |x_1 - x_2|.$$

由于  $|x_1 - x_2| > 0$ , 所以产生矛盾, 因此假设不成立, 即  $\varphi(x)$  在  $[a, b]$  内的不动点是唯一的.  $\square$

 条件  $|\varphi(x) - \varphi(y)| \leq L|x - y|$  称为 **Lipschitz 条件**, 当  $L < 1$  时, 称  $\varphi(x)$  为 **压缩映射**.

**定理 6.4 (不动点迭代的全局收敛性)** 设  $\varphi(x) \in C[a, b]$  且满足

- (1) 对任意  $x \in [a, b]$ , 都有  $\varphi(x) \in [a, b]$ ,
- (2) 存在常数  $L$ , 满足  $0 < L < 1$ , 使得对任意  $x, y \in [a, b]$  都有

$$|\varphi(x) - \varphi(y)| \leq L|x - y|.$$

则对任意初始值  $x_0 \in [a, b]$ , 不动点迭代 (6.1) 收敛, 且

$$|x_k - x_*| \leq \frac{L}{1-L}|x_k - x_{k-1}| \leq \frac{L^k}{1-L}|x_1 - x_0|,$$

其中  $x_*$  是  $\varphi(x)$  在  $[a, b]$  内存在唯一的不动点.

**证明.** 由条件 (1) 可知,  $x_k \in [a, b]$ ,  $k = 0, 1, 2, \dots$ . 再根据条件 (2), 我们有

$$|x_k - x_*| = |\varphi(x_{k-1}) - \varphi(x_*)| \leq L|x_{k-1} - x_*|. \quad (6.2)$$

依此类推, 可得

$$|x_k - x_*| \leq L^k|x_0 - x_*|.$$

由于  $0 < L < 1$ , 所以

$$\lim_{k \rightarrow \infty} |x_k - x_*| = 0, \quad \text{即} \quad \lim_{k \rightarrow \infty} x_k = x_*.$$



由 (6.2) 直接可得

$$|x_k - x_*| \leq L|x_{k-1} - x_*| \leq L(|x_k - x_{k-1}| + |x_k - x_*|).$$

因为  $L < 1$ , 所以

$$|x_k - x_*| \leq \frac{L}{1-L}|x_k - x_{k-1}|.$$

根据不动点迭代格式和条件 (1), 可得

$$|x_k - x_{k-1}| = |\varphi(x_{k-1}) - \varphi(x_{k-2})| \leq L|x_{k-1} - x_{k-2}|.$$


不断递推下去, 最后可得

$$|x_k - x_{k-1}| \leq L^{k-1}|x_1 - x_0|.$$

所以

$$|x_k - x_*| \leq \frac{L}{1-L}|x_k - x_{k-1}| \leq \frac{L^k}{1-L}|x_1 - x_0|.$$

□

 由定理的结论可知,  $L$  越小, 收敛越快!

如果  $\varphi(x)$  在  $[a, b]$  上连续, 在  $(a, b)$  内可导, 则根据 Lagrange 中值定理可知, 对任意  $x, y \in [a, b]$ , 都存在  $\xi \in (a, b)$  使得

$$\varphi(y) - \varphi(x) = \varphi'(\xi)(y - x),$$

因此

$$|\varphi(y) - \varphi(x)| \leq \max_{\xi \in [a, b]} |\varphi'(\xi)| \cdot |y - x|.$$

于是我们可以立即得到下面的结论.

**推论 6.5** 设  $\varphi(x) \in C^1[a, b]$  且对任意  $x \in [a, b]$ , 都有  $\varphi(x) \in [a, b]$ . 如果存在常数  $L$ , 使得

$$|\varphi'(x)| \leq L < 1, \quad \forall x \in [a, b],$$

则对任意初始值  $x_0 \in [a, b]$ , 不动点迭代 (6.1) 收敛, 且

$$|x_k - x_*| \leq \frac{L}{1-L}|x_k - x_{k-1}| \leq \frac{L^k}{1-L}|x_1 - x_0|.$$

以上两个结论中, 迭代法的收敛性与迭代初值的选取无关, 这种收敛性称为 **全局收敛性**.

**例 6.2** 试构造不动点迭代格式, 计算  $f(x) = x^3 - x - 1$  在  $[1, 2]$  中的零点.

(NLS\_fixpoint\_01.m)

**解.** 构造  $f(x) = 0$  的等价形式  $x = \sqrt[3]{x+1} \triangleq \varphi(x)$ , 则对任意  $x \in [1, 2]$ , 有

$$(1) \quad 1 \leq \varphi(x) \leq 2;$$

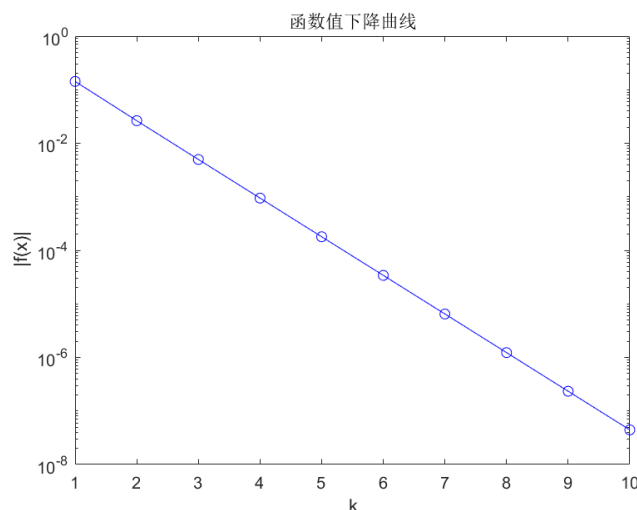
$$(2) \quad |\varphi'(x)| = \left| \frac{1}{3}(x+1)^{-2/3} \right| \leq \frac{1}{3}\sqrt[3]{0.25} < 1.$$

因此, 不动点迭代  $x_{k+1} = \varphi(x_k) = \sqrt[3]{x_k+1}$  全局收敛.

取中点为初始值, 即  $x_0 = 1.5$ , 计算结果见下表. 由表中数据可知, 迭代 10 步后, 函数值已经非

常小, 达到了  $10^{-8}$  量级, 显然比对分法要快很多.

$k$	$x$	$ f(x) $
1	1.3572	1.43e-01
2	1.3309	2.63e-02
3	1.3259	4.98e-03
4	1.3249	9.44e-04
5	1.3248	1.79e-04
6	1.3247	3.41e-05
7	1.3247	6.47e-06
8	1.3247	1.23e-06
9	1.3247	2.33e-07
10	1.3247	4.43e-08



思考: 如果取迭代函数  $\varphi(x) = x^3 - 1$ , 则结果怎样?

□

需要指出的是, 定理 6.4 和推论 6.5 中给出的是充分条件, 不是充要条件.

### 局部收敛性

**定义 6.1** 设  $x_*$  是  $\varphi(x)$  的不动点, 若存在  $x_*$  的某个  $\delta$ -邻域

$$U_\delta(x_*) \triangleq \{x \in \mathbb{R} : |x - x_*| < \delta\},$$

使得对任意  $x_0 \in U_\delta(x_*)$ , 不动点迭代 (6.1) 均收敛, 则称该迭代是**局部收敛**的.

局部收敛意味着只有当初值离真解足够近时, 才能保证收敛. 由于真解是不知道的, 因此如果迭代法只具有局部收敛性, 则初值选取会比较困难, 很有可能无法保证算法的收敛. 这也是局部收敛与全局收敛的最大区别. 在实际计算中, 可以用其他具有全局收敛性的方法 (比如对分法) 获取一个近似解, 然后在进行迭代.

**定理 6.6** 设  $x_*$  是  $\varphi(x)$  的不动点, 若  $\varphi'(x)$  在  $x_*$  的某个邻域内连续且

$$|\varphi'(x_*)| < 1,$$

则不动点迭代 (6.1) 局部收敛.

**证明.** 由条件可知  $\varphi'(x)$  在  $x_*$  的某个邻域内连续且  $|\varphi'(x_*)| < 1$ , 故存在某个领域  $U_\delta(x_*) = [x_* - \delta, x_* + \delta]$  和正常数  $L < 1$  使得

$$|\varphi'(x)| < 1, \quad \forall x \in U_\delta(x_*).$$

此外, 对任意  $x \in U_\delta(x_*)$ , 有

$$|\varphi(x) - x_*| = |\varphi(x) - \varphi(x_*)| \leq L|x - x_*| < \delta,$$





所以  $\varphi(x) \in U_\delta(x_*)$ . 根据推论 6.5, 对任意  $x_0 \in U_\delta(x_*)$ , 迭代法均收敛.  $\square$

## 6.2.2 收敛阶

收敛阶是衡量迭代法收敛速度的一个重要指标.

**定义 6.2** 设迭代  $x_{k+1} = \varphi(x_k)$  收敛到不动点  $x_*$ . 记  $e_k \triangleq x_k - x_*$ , 若

$$\lim_{k \rightarrow \infty} \frac{|e_{k+1}|}{|e_k|^p} = c,$$

其中  $p \geq 1$ , 常数  $c$  与  $k$  无关, 则称该迭代是  $p$  阶收敛的.

- (1) 若  $p = 1$  且  $0 < c < 1$ , 则称 **线性收敛**;
- (2) 若  $p = 2$ , 则称 **二次收敛** 或 **平方收敛**;
- (3) 若  $1 < p < 2$  或  $p = 1$  且  $c = 0$ , 则称 **超线性收敛**.

由定理 6.6 可以立即得到下面的结论.

**定理 6.7** 设  $x_*$  是  $\varphi(x)$  的不动点, 若  $\varphi'(x)$  在  $x_*$  的某个邻域内连续且

$$|\varphi'(x_*)| < 1,$$

则不动点迭代 (6.1) 局部线性收敛.

关于收敛阶的判定, 我们有下面的定理.

**定理 6.8** 设  $x_*$  是  $\varphi(x)$  的不动点且  $p \geq 2$  是正整数. 若  $\varphi^{(p)}(x)$  在  $x_*$  的某个邻域内连续且

$$\varphi'(x_*) = \varphi''(x_*) = \cdots = \varphi^{(p-1)}(x_*) = 0, \quad \varphi^{(p)}(x_*) \neq 0, \quad (6.3)$$

则不动点迭代 (6.1) 是  $p$  阶局部收敛的, 且有

$$\lim_{k \rightarrow \infty} \frac{x_{k+1} - x_*}{x_k - x_*} = \frac{\varphi^{(p)}(x_*)}{p!}.$$

**证明.** 由题意可知  $\varphi'(x_*) = 0$ , 所以迭代法是局部收敛的.

根据 Taylor 展开和条件 (6.3) 可知

$$\begin{aligned} \varphi(x_k) &= \varphi(x_*) + \frac{\varphi^{(1)}(x_*)}{1!}(x_k - x_*) + \cdots + \frac{\varphi^{(p-1)}(x_*)}{(p-1)!}(x_k - x_*)^{p-1} + \frac{\varphi^{(p)}(\xi)}{p!}(x_k - x_*)^p \\ &= \varphi(x_*) + \frac{\varphi^{(p)}(\xi)}{p!}(x_k - x_*)^p, \end{aligned}$$

其中  $\xi$  位于  $x_k$  与  $x_*$  之间. 因此

$$x_{k+1} - x_* = \varphi(x_k) - \varphi(x_*) = \frac{\varphi^{(p)}(\xi)}{p!}(x_k - x_*)^p.$$

由于当  $k \rightarrow \infty$  时  $x_k \rightarrow x_*$ , 所以  $\xi \rightarrow x_*$  ( $k \rightarrow \infty$ ), 于是

$$\lim_{k \rightarrow \infty} \frac{|e_{k+1}|}{|e_k|^p} = \lim_{k \rightarrow \infty} \left| \frac{\varphi^{(p)}(\xi)}{p!} \right| = \left| \frac{\varphi^{(p)}(x_*)}{p!} \right| > 0.$$

根据定义, 迭代法  $p$  阶收敛.  $\square$



**例 6.3** 试构造不同的不动点迭代格式, 计算  $f(x) = x^2 - 3$  的正的零点  $x_* = \sqrt{3}$ .

- (1) 构造  $\varphi(x) = x^2 - 3 + x$ , 则  $\varphi'(x_*) = 2\sqrt{3} + 1 > 1$ , 无法判断其收敛性, 所以该迭代函数不能用.
- (2) 构造  $\varphi(x) = x - \frac{x^2 - 3}{4}$ , 则  $\varphi'(x_*) = 1 - \frac{\sqrt{3}}{2} \approx 0.134 < 1$ , 所以该迭代函数可以采用, 且一阶局部收敛.
- (3) 构造  $\varphi(x) = \frac{1}{2} \left( x + \frac{3}{x} \right)$ , 则  $\varphi'(x_*) = 0$ ,  $\varphi''(x_*) = \frac{2}{\sqrt{3}} \neq 0$ , 所以该迭代函数可以采用, 且二阶局部收敛.

(NLS\_fixpoint\_02.m)

一般来说,  $|\varphi'(x_*)|$  越小, 不动点迭代收敛越快!

在前面的介绍的迭代法中, 计算  $x_{k+1}$  时, 只用到点  $x_k$  的值. 有时, 我们为了利用前面多个点的信息, 比如前面  $\ell \geq 2$  个点, 可以设计下面的迭代法:

$$x_{k+1} = \varphi(x_k, x_{k-1}, \dots, x_{k-\ell+1}), \quad k = \ell - 1, \ell, \ell + 1, \dots \quad (6.4)$$

我们称之为**多点迭代法**. 比如后面会提到的割线法和抛物线法, 都是多点迭代. 显然, 多点迭代法一开始需要给定  $\ell$  个初始值, 即  $x_0, x_1, \dots, x_{\ell-1}$ .



### 6.3 Steffensen 迭代法

不动点迭代的收敛速度取决于迭代函数的选取, 有时收敛会非常缓慢, 这时可以通过适当的方法进行加速. 下面介绍的 Aitken 加速技巧就是一种常用的加速方法.

#### 6.3.1 Aitken 加速技巧

已有不动点迭代

$$x_{k+1} = \varphi(x_k). \quad (6.5)$$

给定初值  $x_0$ , 可得

$$x_1 = \varphi(x_0), \quad x_2 = \varphi(x_1).$$

于是

$$x_1 - x_* = \varphi(x_0) - \varphi(x_*) = \varphi'(\xi_1)(x_0 - x_*),$$

$$x_2 - x_* = \varphi(x_1) - \varphi(x_*) = \varphi'(\xi_2)(x_1 - x_*).$$

如果  $\varphi'(x)$  变化不大, 则可假定  $\varphi'(\xi_1) \approx \varphi'(\xi_2)$ , 即

$$\frac{x_1 - x_*}{x_2 - x_*} \approx \frac{x_0 - x_*}{x_1 - x_*},$$

可得

$$x_* \approx x_0 - \frac{(x_1 - x_0)^2}{x_2 - 2x_1 + x_0}.$$

因此, 我们有理由相信上式右端是  $x_*$  的一个较好的近似. 于是我们可以在原有迭代序列的基础上进行加速, 具体方法为:

$$y_{k+1} = x_k - \frac{(x_{k+1} - x_k)^2}{x_{k+2} - 2x_{k+1} + x_k}. \quad (6.6)$$

这就是 **Aitken 加速方法**. 这时我们就得到两个迭代序列:  $\{x_k\}_{k=0}^{\infty}$  和  $\{y_k\}_{k=0}^{\infty}$ .

**定理 6.9** 假定迭代 6.5 收敛, 且  $\varphi'(x_*) \neq 1$ , 则

$$\lim_{k \rightarrow \infty} \frac{y_{k+1} - x_*}{x_k - x_*} = 0.$$

这意味着  $y_k$  比  $x_k$  更快地收敛到  $x_*$ .

#### 6.3.2 Steffensen 迭代法

将 Aitken 加速技巧与不动点迭代相结合, 就得到 **Steffensen 迭代法**, 具体迭代格式如下:

$$y_k = \varphi(x_k), \quad z_k = \varphi(y_k), \quad x_{k+1} = x_k - \frac{(y_k - x_k)^2}{z_k - 2y_k + x_k}, \quad k = 0, 1, 2, \dots$$

写成不动点迭代形式可得

$$x_{k+1} = \psi(x_k),$$

其中迭代函数  $\psi(x)$  为

$$\psi(x) = x - \frac{(\varphi(x) - x)^2}{\varphi(\varphi(x)) - 2\varphi(x) + x}.$$

**定理 6.10** 设  $x_*$  是  $\psi(x)$  的不动点, 则  $x_*$  是  $\varphi(x)$  的不动点. 反之, 若  $x_*$  是  $\varphi(x)$  的不动点,  $\varphi''(x)$  存在且  $\varphi'(x_*) \neq 1$ , 则  $x_*$  是  $\psi(x)$  的不动点. 另外, 若原不动点迭代是线性收敛的, 则对应的 Steffensen 迭代二阶收敛.

- 🔍 如果原迭代法是  $p$  阶收敛的, 则 Steffensen 迭代  $p+1$  阶收敛.
- 🔍 若原迭代法不收敛, Steffensen 迭代可能收敛.

**例 6.4** 用 Steffensen 迭代法求  $f(x) = x^3 - x - 1$  在区间  $[1, 2]$  内的零点 (取  $\varphi(x) = x^3 - 1$ )  
(NLS\_Steffensen\_01.m)

**例 6.5** 用 Steffensen 迭代法求  $f(x) = 3x^2 - e^x$  在区间  $[3, 4]$  内的零点 (取  $\varphi(x) = 2 \ln(x) + \ln 3$ )  
(NLS\_Steffensen\_02.m)



## 6.4 Newton 法

**Newton 法**是当前求解非线性方程 (组) 的最常用方法, 也是一般情况下的首选方法.

### 6.4.1 基本思想与迭代格式

Newton 法的基本思想是将 **非线性方程线性化**.

设  $x_k$  是  $f(x) = 0$  的一个近似根, 将  $f(x)$  在  $x_k$  处 Taylor 展开可得

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

忽略二次项, 可得  $f(x) \approx P(x)$ , 其中

$$P(x) \triangleq f(x_k) + f'(x_k)(x - x_k).$$

也就是说, 在  $x_k$  附近, 我们用线性函数  $P(x)$  来近似非线性函数  $f(x)$ . 于是, 可以用  $P(x)$  的零点来近似  $f(x)$  的零点, 并将其记为  $x_{k+1}$ , 即

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}. \quad (6.7)$$

这就是 Newton 法的迭代格式, 其几何意义可以用下面的图像表示:

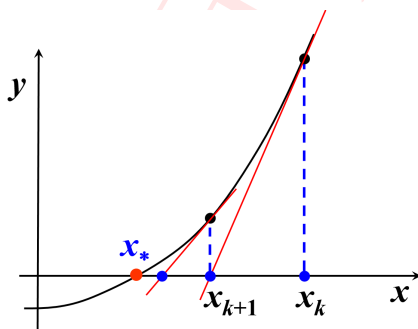


图 6.3. Newton 法的几何含义

为了使得 Newton 法能顺利进行, 一般要求  $f'(x) \neq 0$ .

#### 算法 6.2. Newton 法

- 1: 给定迭代初值  $x_0$ , 精度要求  $\varepsilon$  和最大迭代步数 IterMax
- 2: **if**  $|f(x_0)| < \varepsilon$ , **then**
- 3:     输出近似解  $x_0$ , 停止迭代
- 4: **end if**
- 5: **for**  $k = 1$  to IterMax **do**
- 6:     计算  $x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$
- 7:     **if**  $|x_1 - x_0| < \varepsilon$  或  $|f(x_1)| < \varepsilon$ , **then**
- 8:         输出近似解  $x_1$ , 停止迭代    % 算法收敛

```

9:   end if
10:  x0 = x1
11: end for

```

### 6.4.2 Newton 法的收敛性

由迭代格式 (6.7) 可知, Newton 法也是不动点迭代, 其迭代函数为

$$\varphi(x) = x - \frac{f(x)}{f'(x)}.$$

通过直接计算可得

$$\varphi'(x_*) = 1 - \frac{f'(x_*) \cdot f'(x_*) - f(x_*)f''(x_*)}{(f'(x_*))^2} = 0, \quad \varphi''(x_*) = \frac{f''(x_*)}{f'(x_*)}.$$

根据定理 6.8, 我们可以立即得到下面的结论.

**定理 6.11** 设  $x_*$  是  $f(x)$  的零点, 且  $f'(x_*) \neq 0$ , 则 Newton 法 **至少二阶局部收敛**, 而且有

$$\lim_{k \rightarrow \infty} \frac{x_{k+1} - x_*}{(x_k - x_*)^2} = \frac{\varphi''(x_*)}{2} = \frac{f''(x_*)}{2f'(x_*)}.$$

(板书)

**例 6.6** 编写程序, 用 Newton 法求  $f(x) = xe^x - 1$  的零点.

(NLS\_Newton\_01.m)

**解.** 迭代格式为

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)} = x_k - \frac{x_k e^{x_k} - 1}{e^{x_k}(x_k + 1)}.$$

取初值  $x_0 = 0.5$ , 迭代结果见下表.

$k$	$x$	$ f(x) $
	0.50000000	1.76e-01
1	0.57102044	1.07e-02
2	0.56715557	3.39e-05
3	0.56714329	3.41e-10
4	0.56714329	2.22e-16

从表中数据可以看出, Newton 法迭代 4 步就达到机器精度了, 收敛速度非常快. □

**例 6.7** 用 Newton 法求  $f(x) = x^2 - C = 0$  的正根, 并判断收敛性, 其中  $C > 0$ .

**解.** 易知, Newton 法的迭代格式为

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)} = \frac{1}{2} \left( x_k + \frac{C}{x_k} \right).$$

因此可得

$$x_{k+1} - \sqrt{C} = \frac{1}{2x_k}(x_k - \sqrt{C})^2, \quad x_{k+1} + \sqrt{C} = \frac{1}{2x_k}(x_k + \sqrt{C})^2.$$



所以

$$\frac{x_{k+1} - \sqrt{C}}{x_{k+1} + \sqrt{C}} = \left( \frac{x_k - \sqrt{C}}{x_k + \sqrt{C}} \right)^2.$$

以此类推, 可知

$$\frac{x_{k+1} - \sqrt{C}}{x_{k+1} + \sqrt{C}} = \left( \frac{x_0 - \sqrt{C}}{x_0 + \sqrt{C}} \right)^{2^k} \triangleq q^{2^k},$$

其中  $q = \frac{x_0 - \sqrt{C}}{x_0 + \sqrt{C}}$ . 假定  $x_0 > 0$ , 则  $|q| < 1$ . 直接求解可得

$$x_k - \sqrt{C} = 2\sqrt{C} \frac{q^{2^k}}{1 - q^{2^k}} \rightarrow 0 \quad (k \rightarrow \infty).$$

由此可知, 只要  $x_0 > 0$ , 则 Newton 法总是收敛到正根  $x_* = \sqrt{C}$ , 也就是说, 此时 Newton 法在  $(0, +\infty)$  内是全局收敛的.  $\square$

思考: 如果  $x_0 < 0$ , 则结果会怎样?

一般来说 Newton 法只是局部收敛, 如果初值离真解太远可能就不收敛, 因此初值的选取很重要但也比较困难。幸运得是, 对于计算平方根, Newton 法是全局收敛的, 因此是安全的。

Newton 法的优点是收敛速度快 (至少二阶局部收敛), 特别是当迭代点充分靠近精确解时。但缺点是

- 对重根收敛速度较慢, 只有线性收敛;
- 对初值的选取很敏感, 要求初值相当接近真解, 因此在实际使用时, 可以先用其它方法获取一个近似解, 然后使用 Newton 法加速;
- 每一次迭代都需要计算导数, 工作量大。

### 拓展阅读

#### Newton 法计算平方根与正方形边长的计算.

给定一个正实数  $C$ , 计算其平方根  $\sqrt{C}$  等价于计算面积为  $C$  的正方形的边长. 我们首先任取一个正实数  $L$ , 构造一个长方形, 使得其长为  $L$ , 宽为  $C/L$ , 因此其面积是  $C$ . 下面我们需要做得就是在保持面积不变的情况下, 把这个长方形慢慢变成一个正方形. 怎样才能实现呢?

我们假定  $L > C/L$ , 一个自然的想法就是把长变得短一些, 把宽变得长一些. 因此我们可以用长和宽的平均值作为新的长, 即

$$L_{\text{new}} = \frac{1}{2}(L + C/L),$$

新的宽就是  $C/L_{\text{new}}$ . 这样不断做下去, 长方形会越来越接近正方形, 其长就会越来越接近  $\sqrt{C}$ . 上面的迭代过程事实上就是 Newton 迭代.

### 6.4.3 简化 Newton 法

简化 Newton 法的主要目的是避免每次的求导运算.

基本思想: 用  $f'(x_0)$  替代所有的  $f'(x_k)$ , 这样就只需计算一次导数. 对应的迭代格式为

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_0)}, \quad k = 0, 1, 2, \dots$$

但这样做的代价是收敛速度只有线性收敛.

#### 6.4.4 Newton 下山法

**Newton 下山法** 是为了克服 Newton 法局部收敛的这个缺点.

基本思想: 要求每一步迭代满足下降条件

$$|f(x_{k+1})| < |f(x_k)|, \quad (6.8)$$

即保持函数的绝对值是下降的, 这样就能保证全局收敛性. 具体做法是加入一个 **下山因子**  $\lambda$ , 即

$$x_{k+1} = x_k - \lambda \frac{f(x_k)}{f'(x_k)}, \quad k = 0, 1, 2, \dots$$

下山因子  $\lambda$  的取法: 从  $\lambda = 1$  开始, 逐次减半, 直到满足下降条件 (6.8) 为止.

#### 6.4.5 重根情形

设  $x_*$  是  $f(x) = 0$  的  $m$  ( $m \geq 2$ ) 重根, 即

$$f(x) = (x - x_*)^m g(x),$$

其中  $g(x_*) \neq 0$ .

- **方法一:** 直接使用 Newton 法, 即迭代函数为

$$\varphi(x) = x - \frac{f(x)}{f'(x)},$$

则可得

$$\varphi'(x_*) = 1 - \frac{1}{m} \neq 0,$$

因此, 只有局部线性收敛.

- **方法二:** 用 **改进的 Newton 法**, 即选取迭代函数为

$$\varphi(x) = x - m \frac{f(x)}{f'(x)},$$

则可得

$$\varphi'(x_*) = 0,$$

因此, 至少二阶局部收敛. 但缺点是需要知道  $m$  的值.





- **方法三:** 构造一个等价方程, 使得  $x_*$  是该等价方程的单重根, 然后用 Newton 法求解. 一种简单的构造方法是令

$$\mu(x) \triangleq \frac{f(x)}{f'(x)},$$

则  $x_*$  是  $\mu(x)$  的单重零点. 用 Newton 法求解, 迭代函数为

$$\varphi(x) = x - \frac{\mu(x)}{\mu'(x)} = x - \frac{f(x)f'(x)}{[f'(x)]^2 - f(x)f''(x)}.$$

易知, 该迭代格式至少二阶局部收敛. 但缺点是需要计算二阶导数.

**例 6.8** 编写程序, 分别用以上三种方法计算  $f(x) = x^4 - 4x^2 + 4 = 0$  的二重根  $x_* = \sqrt{2}$ .

(NLS\_Newton\_02.m)

## 6.5 割线法与抛物线法

目的: 避免计算 Newton 法中的导数, 并且尽可能地保持较高的收敛性 (即超线性收敛).

### 6.5.1 割线法

**割线法** (Secant Method) 也称**弦截法**, 主要思想是用**差商代替微商**, 即

$$f'(x_k) \approx f[x_{k-1}, x_k] = \frac{f(x_k) - f(x_{k-1})}{x_k - x_{k-1}}.$$

代入 Newton 法即可得**割线法**迭代格式:

$$x_{k+1} = x_k - \frac{x_k - x_{k-1}}{f(x_k) - f(x_{k-1})} f(x_k), \quad k = 1, 2, \dots$$

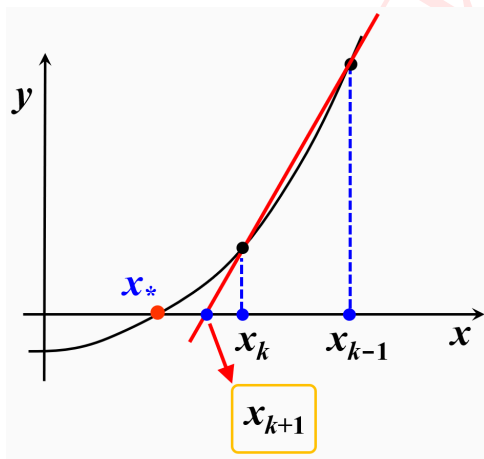


图 6.4. 割线法的几何含义

📌 割线法需要提供两个迭代初始值.

关于割线法的收敛性, 我们有下面的结论.

**定理 6.12** 设  $x_*$  是  $f(x)$  的零点,  $f(x)$  在  $x_*$  的某邻域  $U(x_*, \delta)$  内二阶连续可导, 且  $f'(x) \neq 0$ . 若初值  $x_0, x_1 \in U(x_*, \delta)$ , 则当  $\delta$  充分小时, 割线法具有  $p$  阶收敛性, 其中

$$p = \frac{1 + \sqrt{5}}{2} \approx 1.618,$$

即  $p$  是  $p^2 - p - 1 = 0$  的一个根.

### 6.5.2 抛物线法

在 Newton 法和割线法中, 我们都是用直线来近似  $f(x)$ . Newton 法可以看作是基于单个点的一次 Hermite 插值, 而割线法则是两点线性插值. **抛物线法**的主要思想则是用基于三个点的二次插值多项式来近似  $f(x)$ .



具体做法如下: 假定已知三个相邻的迭代值  $x_{k-2}, x_{k-1}, x_k$ , 构造过点  $(x_{k-2}, f(x_{k-2})), (x_{k-1}, f(x_{k-1})), (x_k, f(x_k))$  的二次曲线  $p_2(x)$ , 然后用  $p_2(x)$  的零点作为下一步的迭代值  $x_{k+1}$ .

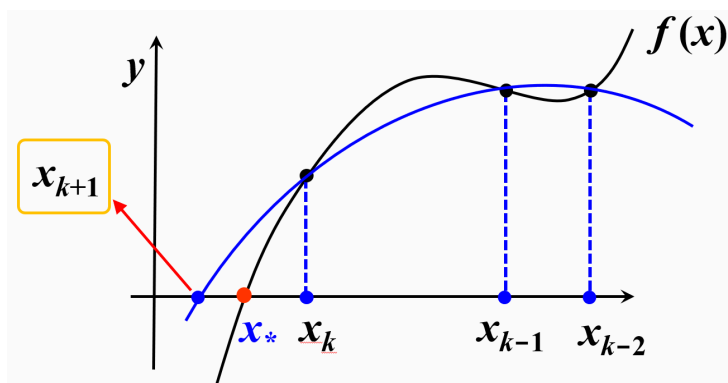


图 6.5. 抛物线法的几何含义

在抛物线法中, 有两个问题需要解决:

(1) 二次曲线  $p_2(x)$  的构造. 可以通过插值方法解决, 比如, 由 Newton 插值公式可得

$$p_2(x) = f(x_k) + f[x_k, x_{k-1}](x - x_k) + f[x_k, x_{k-1}, x_{k-2}](x - x_k)(x - x_{k-1}).$$

(2) 零点的选取. 此时  $p_2(x)$  有两个零点, 即

$$x_k - \frac{2f(x_k)}{\omega \pm \sqrt{\omega^2 - 4f(x_k)f[x_k, x_{k-1}, x_{k-2}]}} ,$$

其中

$$\omega = f[x_k, x_{k-1}] + f[x_k, x_{k-1}, x_{k-2}](x_k - x_{k-1}).$$

取哪个作为  $x_{k+1}$  呢? 通常的做法是取靠近  $x_k$  的那个零点.

在一定条件下可以证明: 抛物线法的局部收敛阶为

$$p \approx 1.840. \quad (p^3 - p^2 - p - 1 = 0)$$

### 几点注记

- (1) 与割线法相比, 抛物线法具有更高的收敛阶.
- (2) 抛物线法可能涉及复数运算, 有时可以用来求复根.
- (3) 抛物线法需提供三个初始值.
- (4) 抛物线法也称为 **Muller 法**.

## 6.6 非线性方程组的数值求解 \*

见课堂板书 (*To be continued ...*)



## 6.7 非线性最小二乘问题 \*

*To be continued ...*

## 6.8 课后练习

**练习 6.1** 用对分法求方程  $f(x) = x^2 + x - 1$  在  $[0, 1]$  中的零点近似值  $\tilde{x}$ , 要求  $|f(\tilde{x})| < 0.5 \times 10^{-2}$ .

**练习 6.2** 计算  $f(x) = x^3 - 2x^2 + 1$  的零点  $x_* = 1$ , 试分析下列迭代公式的 (局部) 收敛性

$$(1) \quad x_{k+1} = 2 - 1/x_k^2$$

$$(2) \quad x_{k+1} = \sqrt[3]{2x_k^2 - 1}$$

$$(3) \quad x_{k+1} = \frac{1}{\sqrt{2 - x_k}}$$

**练习 6.3** 设  $f(x) \in C^1[-\infty, +\infty]$ , 且  $0 < m \leq f'(x) \leq M$ . 试证明: 若  $\alpha \in (0, 2/M)$ , 则对任意初值  $x_0 \in \mathbb{R}$ , 迭代方法

$$x_{k+1} = x_k - \alpha f(x_k), \quad k = 0, 1, 2, \dots$$

收敛到  $f(x)$  的零点  $x_*$ .

**练习 6.4** 设  $\varphi(x) = x - p(x)f(x) - q(x)f^2(x)$ . 试确定  $p(x)$  和  $q(x)$ , 使得以  $\varphi(x)$  为迭代函数的求解  $f(x) = 0$  的不动点迭代至少具有三阶 (局部) 收敛速度.

**练习 6.5** 设  $x_*$  是  $f(x)$  的单重零点, 对于 Newton 法  $x_{k+1} = x_k - f(x_k)/f'(x_k)$ , 证明

$$\lim_{k \rightarrow \infty} \frac{x_{k+1} - x_k}{(x_k - x_{k-1})^2} = -\frac{f''(x_*)}{2f'(x_*)}.$$

(这里假定  $x_k$  是收敛到  $x_*$  的)

**练习 6.6** 应用 Newton 法于  $f(x) = x^3 - a = 0$ , 其中  $a \geq 0$ . 导出计算  $\sqrt[3]{a}$  的迭代公式, 并讨论其全局收敛性, 即对任意给定的  $x_0 \in \mathbb{R}$ , 迭代是否收敛?

**练习 6.7** 设  $a > 0$ , 并假定  $x_0$  充分靠近  $\sqrt{a}$ . 证明迭代公式

$$x_{k+1} = \frac{x_k(x_k^2 + 3a)}{3x_k^2 + a}, \quad k = 0, 1, 2, \dots$$

是计算  $\sqrt{a}$  的三阶方法, 并计算

$$\lim_{k \rightarrow \infty} \frac{x_{k+1} - x_k}{(x_k - x_{k-1})^3}.$$



# 7

## 函数插值

In the mathematical subfield of numerical analysis, interpolation is a method of constructing new data points from a discrete set of known data points.

— Psychology Wiki

许多实际问题都可用函数来表示其某种内在规律的数量关系, 比如气候变化, 但精确的函数表达式通常是无法得到的, 我们能获取的往往只有通过实验或观测得到的数据, 一个很自然的问题就是如何根据这些数据来推测或估计其它点的函数值? 此时就需要用到数值逼近技术, 其中一个主要的手段就是函数插值.

**例 7.1** 已测得在某处海洋不同深度处的水温如下:

深度 (M)	466	741	950	1422	1634
水温 (°C)	7.04	4.28	3.40	2.54	2.13

根据这些数据, 请合理地估计出其它深度 (如 500, 600, 800 米 ...) 处的水温.

### 7.1 多项式插值

什么是插值

**定义 7.1** 已知函数  $y = f(x)$  在区间  $[a, b]$  上有定义, 且已经测得其在点

$$a \leq x_0 < x_1 < \cdots < x_n \leq b \quad (7.1)$$

处的值为  $y_0 = f(x_0), \cdots, y_n = f(x_n)$ . 如果存在一个简单易算的函数  $p(x)$ , 使得

$$p(x_i) = y_i, \quad i = 0, 1, \dots, n, \quad (7.2)$$

则称  $p(x)$  为  $f(x)$  的插值函数. 区间  $[a, b]$  称为插值区间,  $x_i (i = 0, 1, \dots, n)$  称为插值节点, 条件 (7.2) 称为插值条件.

插值节点可以不按递增排列, 但必须确保互不相同!

### 插值法

求插值函数  $p(x)$  的方法就称为**插值法**. 常见的插值法有:

- **多项式插值**:  $p(x)$  为多项式, 多项式是常用的插值函数;
- **分段多项式插值**:  $p(x)$  为分段多项式, 用分段多项式插值是常用的插值法;
- **有理插值**:  $p(x)$  为有理函数;
- **三角插值**:  $p(x)$  为三角函数;
- ... ..

本讲主要介绍多项式插值和分段多项式插值.

### 多项式插值

**定义 7.2 (多项式插值)** 已知函数  $y = f(x)$  在区间  $[a, b]$  上  $n+1$  个点

$$a \leq x_0 < x_1 < \cdots < x_n \leq b$$

处的函数值为  $y_0 = f(x_0), \dots, y_n = f(x_n)$ . **多项式插值** 就是寻找一个次数不超过  $n$  的**多项式**的函数

$$p(x) = c_0 + c_1x + \cdots + c_nx^n, \quad (7.3)$$

使得

$$p(x_i) = y_i, \quad i = 0, 1, \dots, n.$$

需要指出的是,  $p(x)$  的次数有可能小于  $n$ .

**定理 7.1 (多项式插值存在唯一性)** 满足插值条件的多项式  $p(x)$  存在且唯一.

**证明.** 待定系数法. 设  $p(x)$  的表达式为 (7.3). 将插值条件  $p(x_i) = y_i$  代入, 得到一个关于  $c_0, c_1, \dots, c_n$  的线性方程组, 其系数矩阵正好是一个关于  $x_0, x_1, \dots, x_n$  的 Vandermonde 矩阵. 因此当插值节点互不同时, 其行列式不为 0. 所以系数矩阵可逆, 即解存在唯一.  $\square$

该定理的证明过程事实上也给出了一种求  $p(x)$  的方法, 但这个方法比较复杂, 当插值点较多时, 需要解一个很大的线性方程组, 不实用, 后面将给出几个较简单的计算方法.

**例 7.2** 线性插值: 求一个一次多项式  $p(x)$ , 满足:

$$p(x_0) = y_0, \quad p(x_1) = y_1.$$

**解.** 由于  $p(x)$  是一次多项式, 即代表一条直线. 因此由点斜式可知

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





$$\begin{aligned}
 &= y_0 - \frac{y_0(x-x_0)}{x_1-x_0} + y_1 \frac{x-x_0}{x_1-x_0} \\
 &= y_0 \frac{x-x_1}{x_0-x_1} + y_1 \frac{x-x_0}{x_1-x_0}.
 \end{aligned}$$

记  $l_0(x) = \frac{x-x_1}{x_0-x_1}$ ,  $l_1(x) = \frac{x-x_0}{x_1-x_0}$ , 则  $p(x)$  就可以表示成  $l_0(x)$  和  $l_1(x)$  的线性组合, 即

$$p(x) = y_0 l_0(x) + y_1 l_1(x).$$

我们进一步观察可知

$$\begin{aligned}
 l_0(x_0) &= 1, & l_0(x_1) &= 0; \\
 l_1(x_0) &= 0, & l_1(x_1) &= 1.
 \end{aligned}$$

□

**例 7.3** 抛物线插值: 求一个二次多项式  $p(x)$ , 满足:

$$p(x_0) = y_0, \quad p(x_1) = y_1, \quad p(x_2) = y_2.$$

**解.** 借鉴线性插值思想, 如果能构造出三个二次多项式  $l_0(x), l_1(x), l_2(x)$ , 满足

$$\begin{aligned}
 l_0(x_0) &= 1, & l_0(x_1) &= 0, & l_0(x_2) &= 0; \\
 l_1(x_0) &= 0, & l_1(x_1) &= 1, & l_1(x_2) &= 0; \\
 l_2(x_0) &= 0, & l_2(x_1) &= 0, & l_2(x_2) &= 1.
 \end{aligned}$$

则由插值条件可知,  $p(x)$  可以表示成

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

现在的问题是如何构造  $l_0(x), l_1(x), l_2(x)$ . 我们可以使用待定系数法.

由于  $l_0(x)$  是二次多项式, 且满足  $l_0(x_1) = l_0(x_2) = 0$ , 因此  $l_0(x)$  可以写成

$$l_0(x) = \alpha(x-x_1)(x-x_2),$$

其中  $\alpha$  是待定系数 (常数). 将  $l_0(x_0) = y_0$  代入可得  $\alpha = \frac{1}{(x_0-x_1)(x_0-x_2)}$ . 所以

$$l_0(x) = \frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)}.$$

同理可得

$$\begin{aligned}
 l_1(x) &= \frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)}, \\
 l_2(x) &= \frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)}.
 \end{aligned}$$

□

我们注意到,  $p(x)$  之所以可以写成  $l_0(x), l_1(x)$  和  $l_2(x)$  的线性组合, 是因为  $l_0(x), l_1(x), l_2(x)$  组成了线性空间

$$\mathbb{H}_2(x) \triangleq \{\text{次数不超过 2 的多项式的全体}\}$$

的一组基, 而  $p(x) \in \mathbb{H}_2(x)$ , 因此  $p(x)$  可以由  $l_0(x), l_1(x), l_2(x)$  线性表出. 这种利用基函数来计算



插值函数的方法就是**基函数插值法**.

### 基函数插值法

记

$\mathbb{I}_n(x) \triangleq \{\text{所有次数不超过 } n \text{ 的实系数多项式组成的集合}\},$

则  $\mathbb{I}_n$  构成一个  $n+1$  维的线性空间. 易知,  $n$  次多项式插值就是在  $\mathbb{I}_n$  中寻找一个多项式  $p(x)$ , 使得插值条件 (7.2) 成立.

设  $\{\phi_0(x), \phi_1(x), \dots, \phi_n(x)\}$  是  $\mathbb{I}_n$  的一组基, 则  $p(x)$  可以表示成

$$p(x) = a_0\phi_0(x) + a_1\phi_1(x) + \cdots + a_n\phi_n(x),$$

其中  $a_0, a_1, \dots, a_n$  是线性表出系数.

#### 多项式插值的两个关键问题

- (1) 寻找合适的基函数;
- (2) 确定插值多项式在这组基下的线性表出系数.



## 7.2 Lagrange 插值

将线性插值和抛物线插值的思想推广到一般情形, 就得到 **Lagrange 插值法**, 它是基函数插值法的典型代表.

### 7.2.1 Lagrange 基函数

**定义 7.3** 设  $l_k(x)$  是  $n$  次多项式, 且在插值节点  $x_0, x_1, \dots, x_n$  上满足

$$l_k(x_i) = \begin{cases} 1, & i = k \\ 0, & i \neq k \end{cases} \quad i, k = 0, 1, 2, \dots, n. \quad (7.4)$$

则称  $l_k(x)$  为节点  $x_0, x_1, \dots, x_n$  上的  $n$  次 **Lagrange 基函数**.

下面利用构造法计算  $l_k(x)$  的表达式. 由条件 (7.4) 可知  $x_0, \dots, x_{k-1}, x_{k+1}, \dots, x_n$  是  $l_k(x)$  的零点, 又  $l_k(x)$  是  $n$  次多项式, 故可设

$$l_k(x) = \alpha(x - x_0) \cdots (x - x_{k-1})(x - x_{k+1}) \cdots (x - x_n),$$

其中  $\alpha$  是待定系数. 将条件  $l_k(x_k) = 1$  代入可得

$$\alpha = \frac{1}{(x_k - x_0) \cdots (x_k - x_{k-1})(x_k - x_{k+1}) \cdots (x_k - x_n)}.$$

所以

$$\begin{aligned} l_k(x) &= \frac{(x - x_0) \cdots (x - x_{k-1})(x - x_{k+1}) \cdots (x - x_n)}{(x_k - x_0) \cdots (x_k - x_{k-1})(x_k - x_{k+1}) \cdots (x_k - x_n)} \\ &= \prod_{i=0, i \neq k}^n \frac{x - x_i}{x_k - x_i}. \end{aligned}$$

- 容易证明:  $l_0(x), l_1(x), \dots, l_n(x)$  线性无关, 因此它们构成  $\mathbb{H}_n(x)$  的一组基.
- $l_0(x), l_1(x), \dots, l_n(x)$  与插值节点有关, 但与  $f(x)$  无关.

### 如何用 Lagrange 基函数求插值多项式

由于  $l_0(x), l_1(x), \dots, l_n(x)$  构成  $\mathbb{H}_n(x)$  的一组基, 所以插值多项式  $p(x)$  可以写成

$$p(x) = a_0 l_0(x) + a_1 l_1(x) + \cdots + a_n l_n(x).$$

将插值条件 (7.2) 代入可得

$$a_i = y_i, \quad i = 0, 1, 2, \dots, n.$$


所以

$$p(x) = y_0 l_0(x) + y_1 l_1(x) + \cdots + y_n l_n(x).$$

我们将这个多项式记为  $L_n(x)$ , 它就是 **Lagrange 插值多项式**, 即

$$L_n(x) = \sum_{k=0}^n y_k l_k(x) = \sum_{k=0}^n y_k \prod_{i=0, i \neq k}^n \frac{x - x_i}{x_k - x_i}. \quad (7.5)$$

当  $n = 1$  和  $2$  时, 就可以得到线性插值多项式和抛物线插值多项式.

  $L_n(x)$  通常是  $n$  次的, 但也可能会低于  $n$  次. 如: 抛物线插值中, 如果三点共线, 则  $L_2(x)$  是一次多项式.

**例 7.4** 已知函数  $f(x) = \ln(x)$  的函数值如下:

$x$	0.4	0.5	0.6	0.7	0.8
$f(x)$	-0.9163	-0.6931	-0.5108	-0.3567	-0.2231

试分别用线性插值和抛物线插值计算  $\ln(0.54)$  的近似值.

**解.** 为了减小截断误差, 通常选取离插值点  $x$  比较近的点作为插值节点.

**线性插值:** 取插值节点  $x_0 = 0.5, x_1 = 0.6$ . 根据 Lagrange 插值公式, 可得  $f(x)$  在区间  $[0.5, 0.6]$  上的线性插值为


$$L_1(x) = y_0 \frac{x - x_1}{x_0 - x_1} + y_1 \frac{x - x_0}{x_1 - x_0} \approx 0.1823x - 1.6046.$$


将  $x = 0.54$  代入可得  $\ln(0.54) \approx -0.6202$ .

**抛物线插值:** 取插值节点  $x_0 = 0.4, x_1 = 0.5, x_2 = 0.6$ . 根据 Lagrange 插值公式, 可得  $f(x)$  在  $x = 0.54$  上的近似值为

$$\begin{aligned} \ln(0.54) &\approx L_2(0.54) \\ &= y_0 \frac{(0.54 - x_1)(0.54 - x_2)}{(x_0 - x_1)(x_0 - x_2)} + y_1 \frac{(0.54 - x_0)(0.54 - x_2)}{(x_1 - x_0)(x_1 - x_2)} + y_2 \frac{(0.54 - x_0)(0.54 - x_1)}{(x_2 - x_0)(x_2 - x_1)} \\ &\approx -0.6153. \end{aligned}$$

□

 我们将需要插值的点称为 **插值点**, 比如上面例题中的 0.54. 在实际计算中, 一般不需要给出插值多项式的具体表达式, 可以直接将插值点代入进行计算, 得到近似值.

 Lagrange 插值简单方便, 只要给定插值节点就可写出基函数, 易于计算机实现.

## 7.2.2 插值余项

Lagrange 插值多项式的余项记为

$$R_n(x) \triangleq f(x) - L_n(x).$$



**定理 7.2** 设  $f(x) \in C^n[a, b]$  (即存在  $n$  阶连续导数), 且  $f^{(n+1)}(x)$  在  $(a, b)$  内存在. 则对  $\forall x \in [a, b]$ , 都存在  $\xi_x \in (a, b)$  使得

$$R_n(x) = \frac{f^{(n+1)}(\xi_x)}{(n+1)!} \omega_{n+1}(x), \quad (7.6)$$

其中  $\omega_{n+1}(x) = (x - x_0)(x - x_1) \cdots (x - x_n)$ .

**证明.** 只需证明等式 (7.6) 对所有  $x \in [a, b]$  都成立即可.

首先证明等式 (7.6) 在所有插值节点  $x_i$  上都成立. 易知等式左边  $R_n(x_i) = f(x_i) - L_n(x_i) = 0$ . 又由  $\omega_{n+1}(x)$  的表达式可知  $\omega(x_i) = 0$ . 因此等式 (7.6) 成立.

下面证明等式 (7.6) 对其他点也成立. 由  $R_n(x_i) = f(x_i) - L_n(x_i) = 0$  可知  $x_0, x_1, x_2, \dots, x_n$  是  $R_n(x)$  的零点, 因此  $R_n(x)$  可以写成

$$R_n(x) = K(x)(x - x_0)(x - x_1) \cdots (x - x_n) = K(x)\omega_{n+1}(x), \quad (7.7)$$

其中  $K(x)$  待定.

设  $\tilde{x}$  是  $[a, b]$  中的任意一点, 且  $\tilde{x} \neq x_i, i = 0, 1, 2, \dots, n$ . 下面证明等式 (7.6) 在点  $\tilde{x}$  上也成立, 即证明: **存在  $\xi_{\tilde{x}}$  使得  $K(\tilde{x}) = \frac{f^{(n+1)}(\xi_{\tilde{x}})}{(n+1)!}$ .**

构造函数

$$\varphi(x) \triangleq f(x) - L_n(x) - K(\tilde{x})\omega_{n+1}(x). \quad (7.8)$$

由 (7.7) 可知,

$$\varphi(\tilde{x}) = f(\tilde{x}) - L_n(\tilde{x}) - K(\tilde{x})\omega_{n+1}(\tilde{x}) = R_n(\tilde{x}) - K(\tilde{x})\omega_{n+1}(\tilde{x}) = 0.$$

又

$$\varphi(x_i) = f(x_i) - L_n(x_i) - K(\tilde{x})\omega_{n+1}(x_i) = 0 - 0 = 0, \quad i = 0, 1, 2, \dots, n,$$


所以  $\varphi(x)$  在  $[a, b]$  内至少有  $n+2$  个互不相同的零点. 根据条件,  $\varphi(x)$  是  $n$  阶连续可导且存在  $n+1$  阶导数, 因此由罗尔定理可知,  $\varphi'(x)$  在  $(a, b)$  内至少有  $n+1$  个不同的零点. 再根据罗尔定理,  $\varphi''(x)$  在  $(a, b)$  内至少有  $n$  个不同的零点. 以此类推, 可知  $\varphi^{(n+1)}(x)$  在  $(a, b)$  内至少有 1 个零点, 不妨设为  $\xi_{\tilde{x}}$ , 即  $\varphi^{(n+1)}(\xi_{\tilde{x}}) = 0$ . 代入 (7.8) 可得

$$f^{(n+1)}(\xi_{\tilde{x}}) - L_n^{(n+1)}(\xi_{\tilde{x}}) - K(\tilde{x})\omega_{n+1}^{(n+1)}(\xi_{\tilde{x}}) = 0.$$

又  $L_n^{(n+1)}(x) = 0, \omega_{n+1}^{(n+1)}(x) = (n+1)!$ , 因此

$$K(\tilde{x}) = \frac{f^{(n+1)}(\xi_{\tilde{x}})}{(n+1)!}.$$

由此可知, 等式 (7.6) 对  $[a, b]$  内的所有点都成立. □

 余项中的  $\xi_x$  与  $x$  是相关的.

特别地, 当  $n = 1$  时, 线性插值的余项是 (假定  $x_0 < x_1$ )

$$R_1(x) = \frac{1}{2}f''(\xi_x)(x - x_0)(x - x_1), \quad \xi_x \in (x_0, x_1).$$

当  $n = 2$  时, 抛物线插值的余项是 (假定  $x_0 < x_1 < x_2$ )

$$R_2(x) = \frac{1}{6} f'''(\xi_x)(x - x_0)(x - x_1)(x - x_2), \quad \xi_x \in (x_0, x_2).$$



- 余项公式只有当  $f(x)$  的高阶导数存在时才能使用;
- $\xi_x$  与  $x$  有关, 通常无法确定, 因此在实际应用中, 通常是估计其上界, 即

$$\text{如果有 } \max_{a \leq x \leq b} |f^{(n+1)}(x)| = M_{n+1}, \quad \text{则 } |R_n(x)| \leq \frac{M_{n+1}}{(n+1)!} |\omega_{n+1}(x)|.$$

- 在利用插值方法计算插值点  $x$  上的近似值时, 应尽量选取与  $x$  相近的插值节点.

### 7.2.3 Lagrange 基函数的两个重要性质

如果  $f(x)$  是一个次数不超过  $n$  的多项式, 则  $f^{(n+1)}(x) \equiv 0$ , 因此

$$R_n(x) = \frac{f^{(n+1)}(\xi_x)}{(n+1)!} \omega_{n+1}(x) \equiv 0.$$

所以我们有下面的性质.

**性质 7.1** 当  $f(x)$  为一个次数不超过  $n$  的多项式时, 其  $n$  次插值多项式是精确的.

设  $f(x) = x^m$ , 其中  $0 \leq m \leq n$ , 则由性质 7.1 可知  $f(x) = L_n(x)$ , 即

$$\sum_{k=0}^n x_k^m l_k(x) = x^m, \quad 0 \leq m \leq n. \quad (7.9)$$

特别地, 当  $m = 0$  时, 有

$$\sum_{k=0}^n l_k(x) = 1.$$

这是一个很有趣的性质.

**例 7.5** 设  $l_i(x)$  是关于点  $x_0, x_1, \dots, x_5$  的 Lagrange 插值基函数. 证明:

$$\sum_{i=0}^5 (x_i - x)^2 l_i(x) = 0.$$

**证明.** 直接展开可得

$$\begin{aligned} \sum_{i=0}^5 (x_i - x)^2 l_i(x) &= \sum_{i=0}^5 (x_i^2 - 2x_i x + x^2) l_i(x) \\ &= \sum_{i=0}^5 x_i^2 l_i(x) - 2x \sum_{i=0}^5 x_i l_i(x) + x^2 \sum_{i=0}^5 l_i(x) \end{aligned} \quad (7.10)$$



根据等式 (7.9) 可知

$$\sum_{i=0}^5 x_i^2 l_i(x) = x^2, \quad \sum_{i=0}^5 x_i l_i(x) = x, \quad \sum_{i=0}^5 l_i(x) = 1,$$

代入 (7.10), 即可知结论成立.  $\square$

**例 7.6** 设  $f(x) \in C^2[a, b]$ , 证明:

$$\max_{a \leq x \leq b} \left| f(x) - \left[ f(a) + \frac{f(b) - f(a)}{b - a} (x - a) \right] \right| \leq \frac{1}{8} M_2 (b - a)^2,$$

其中  $M_2 = \max_{a \leq x \leq b} |f''(x)|$ .

**证明.** 记

$$L_1(x) = f(a) + \frac{f(b) - f(a)}{b - a} (x - a),$$

则

$$L_1(a) = f(a), \quad L_1(b) = f(b).$$

因此,  $L_1(x)$  是  $f(x)$  关于插值节点  $x_0 = a, x_1 = b$  的线性插值多项式. 由多项式插值的余项公式 (7.6) 可知

$$R_1(x) = f(x) - L_1(x) = \frac{f''(\xi_x)}{2!} (x - a)(x - b), \quad \xi_x \in (a, b).$$

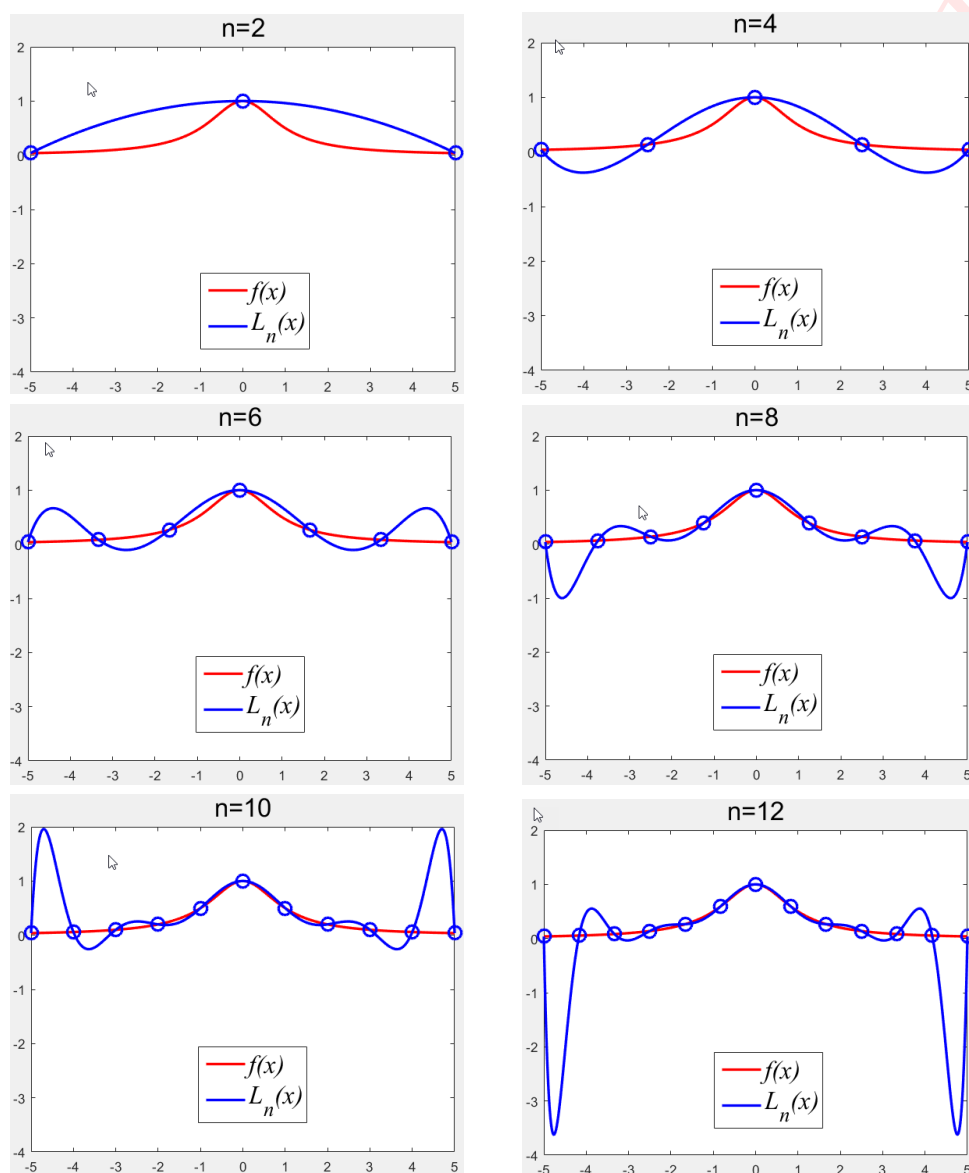
因此当  $x \in [a, b]$  时, 有

$$\begin{aligned} |f(x) - L_1(x)| &= \frac{|f''(\xi_x)|}{2} |(x - a)(x - b)| \\ &\leq \frac{M_2}{2} (x - a)(b - x) \\ &\leq \frac{M_2}{2} \left( \frac{(x - a) + (b - x)}{2} \right)^2 = \frac{1}{8} M_2 (b - a)^2. \end{aligned}$$

所以结论成立.  $\square$

**例 7.7** Runge 现象: 已知函数  $f(x) = \frac{1}{1+x^2}$ , 插值区间  $[-5, 5]$ , 取等距插值节点, 即  $x_i = -5 + \frac{10i}{n}$ ,  $i = 0, 1, 2, \dots, n$ , 画出当  $n = 2, 4, 6, 8, 10, 12$  时, 插值多项式  $L_n$  的图像. 并由此说明, 插值多项式并不一定是次数越高就越好! (Interp\_runge.m)

**解.** 当  $n = 2, 4, 6, 8, 10$  时, 原函数  $f(x)$  和插值多项式  $L_n$  的图像如下:



□

### 拓展阅读

除了 Runge 现象以外, 俄罗斯数学家 C. H. Bernstein 在 1916 年还给出了以下结论: 函数  $f(x) = |x|$  在  $[-1, 1]$  上取  $n+1$  个等距节点进行插值, 其中  $x_0 = -1, x_n = 1$ , 构造出的  $n$  插值多项式记为  $p_n(x)$ . 当  $n$  不断增大时, 除了  $-1, 0, 1$  这三个点外, 在  $[-1, 1]$  中任何点处  $p_n(x)$  都不收敛于  $f(x)$ . 该结论的证明可以参见《函数构造论》(下册, 何旭初和唐述剑翻译, 科学出版社, 1959).





## 7.3 Newton 插值

### 为什么 Newton 插值

Lagrange 插值简单易用, 但若增加插值节点时, 全部基函数  $l_k(x)$  都需重新计算, 很不方便!

解决办法就是寻找新的基函数组, 使得当节点增加时, 只需在原有基函数的基础上再增加一些新的基函数即可. 这样, 原有的基函数仍然可以使用.

基于这种基函数的选取方法, 我们还可能设计一个可以逐次生成插值多项式的算法, 即

$$p_{n+1}(x) = p_n(x) + u_{n+1}(x),$$

其中  $p_{n+1}(x)$  和  $p_n(x)$  分别为  $f(x)$  的  $n+1$  次和  $n$  次插值多项式, 而且  $u_{n+1}(x)$  可以很容易地给出.

### Newton 插值基函数

设插值节点为  $x_0, x_1, \dots, x_n$ , 考虑函数组

$$\begin{aligned}\phi_0(x) &= 1 \\ \phi_1(x) &= x - x_0 \\ \phi_2(x) &= (x - x_0)(x - x_1) \\ &\dots \dots \\ \phi_n(x) &= (x - x_0)(x - x_1) \cdots (x - x_{n-1}).\end{aligned}$$

显然,  $\phi_k(x)$  是  $k$  次多项式, 且  $\phi_0(x), \phi_1(x), \dots, \phi_n(x)$  线性无关, 因此它们组成多项式线性空间  $\mathbb{R}_n(x)$  的一组基.

这组基函数的优点是当增加一个新的插值节点  $x_{n+1}$  时, 只需在原有的基的基础上增加下面的函数即可

$$\phi_{n+1}(x) = (x - x_0)(x - x_1) \cdots (x - x_n).$$

这意味着原来的基函数仍然可以使用. Newton 插值法就是基于这组基函数组的插值方法.

设  $p_n(x)$  是  $f(x)$  在节点  $x_0, x_1, \dots, x_n$  上的  $n$  次插值多项式. 由于  $\phi_0(x), \phi_1(x), \dots, \phi_n(x)$  构成  $\mathbb{R}_n(x)$  上的一组基, 因此  $p_n(x)$  可以写成

$$p_n(x) = a_0\phi_0(x) + a_1\phi_1(x) + \cdots + a_n\phi_n(x).$$

#### 需要解决两个问题

- (1) 怎样确定参数  $a_0, a_1, \dots, a_n$ ?
- (2) 如果得到从  $p_n(x)$  到  $p_{n+1}(x)$  的递推方法?

解决以上问题, 我们需要用到**差商**.

## 7.3.1 差商

**定义 7.4** 设节点  $x_0, x_1, \dots, x_n$ , 我们称

$$f[x_i, x_j] = \frac{f(x_j) - f(x_i)}{x_j - x_i}$$


为  $f(x)$  关于点  $x_i, x_j$  的**一阶差商**; 称

$$f[x_i, x_j, x_k] = \frac{f[x_j, x_k] - f[x_i, x_j]}{x_k - x_i}$$

为  $f(x)$  关于点  $x_i, x_j, x_k$  的**二阶差商**; 一般地, 称

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

为  $f(x)$  关于点  $x_0, x_1, \dots, x_k$  的 **$k$  阶差商**.

 差商有时也称为**均差**.

## 差商基本性质

- 差商可以表示为函数值的线性组合, 即 (可以用归纳法证明)

$$f[x_0, x_1, \dots, x_k] = \sum_{j=0}^k \frac{f(x_j)}{\prod_{i=0, i \neq j}^k (x_j - x_i)} = \sum_{j=0}^k \frac{f(x_j)}{\omega'_{k+1}(x_j)}, \quad (7.11)$$

其中  $\omega_{k+1}(x) = (x - x_0)(x - x_1) \cdots (x - x_k)$ . 由此可见, 差商与节点的排序无关, 即差商具有下面的**对称性**:

$$f[x_0, x_1, \dots, x_k] = f[x_{i_0}, x_{i_1}, \dots, x_{i_k}]$$

其中  $i_0, i_1, \dots, i_k$  是  $0, 1, \dots, k$  的一个任意排列.

- 根据差商的对称性, 我们可以给出差商的等价定义, 如:

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

- 若  $h(x) = \alpha f(x)$ , 其中  $\alpha$  是常数, 则

$$h[x_0, x_1, \dots, x_k] = \alpha f[x_0, x_1, \dots, x_k].$$

- 若  $h(x) = f(x) + g(x)$ , 则

$$h[x_0, x_1, \dots, x_k] = f[x_0, x_1, \dots, x_k] + g[x_0, x_1, \dots, x_k].$$

- $k$  阶差商与  $k$  阶导数之间的关系: 若  $f(x)$  在  $[a, b]$  上有  $k$  阶导数, 则至少存在一点  $\xi \in (a, b)$ , 使得

$$f[x_0, x_1, \dots, x_k] = \frac{f^{(k)}(\xi)}{k!}. \quad (7.12)$$

## 差商的计算

利用差商的递推定义, 我们可以构造下面的**差商表**来计算差商.



$x_i$	$f(x_i)$	一阶差商	二阶差商	三阶差商	...	$n$ 阶差商
$x_0$	$f(x_0)$					
$x_1$	$f(x_1)$	$f[x_0, x_1]$				
$x_2$	$f(x_2)$	$f[x_1, x_2]$	$f[x_0, x_1, x_2]$			
$x_3$	$f(x_3)$	$f[x_2, x_3]$	$f[x_1, x_2, x_3]$	$f[x_0, x_1, x_2, x_3]$		
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\cdots$	
$x_n$	$f(x_n)$	$f[x_{n-1}, x_n]$	$f[x_{n-2}, x_{n-1}, x_n]$	$f[x_{n-3}, x_{n-2}, x_{n-1}, x_n]$	$\cdots$	$f[x_0, x_1, \dots, x_n]$

**例 7.8** 已知  $y = f(x)$  的函数值表如下, 试计算其各阶差商

(Interp\_diff\_01.m)

$i$	0	1	2	3
$x_i$	-2	-1	1	2
$f(x_i)$	5	3	17	21

**解.** 通过计算可得差商表如下:

$x_i$	$f(x_i)$	一阶差商	二阶差商	三阶差商
-2	5			
-1	3	-2		
1	17	7	3	
2	21	4	-1	-1

□

### 7.3.2 Newton 插值公式

下面我们开始推导 Newton 插值公式. 由差商的定义可知

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

所以

$$f(x) = f(x_0) + f[x, x_0](x - x_0). \quad (7.13)$$

同理, 由

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

可得

$$f[x, x_0] = f[x_0, x_1] + f[x, x_0, x_1](x - x_1). \quad (7.14)$$

以此类推, 我们有

$$f[x, x_0, x_1] = f[x_0, x_1, x_2] + f[x, x_0, x_1, x_2](x - x_2) \quad (7.15)$$

$\vdots$

$$f[x, x_0, \dots, x_{n-1}] = f[x_0, x_1, \dots, x_n] + f[x, x_0, x_1, \dots, x_n](x - x_n). \quad (7.16)$$

将等式 (7.13)-(7.16) 联立可得 (依次将后面一式代入前面一式)

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

所以

$$f(x) = N_n(x) + R_n(x),$$

其中

$$N_n(x) = a_0 + a_1(x - x_0) + \cdots + a_n(x - x_0) \cdots (x - x_{n-1}), \quad (7.17)$$

$$a_0 = f(x_0), \quad a_i = f[x_0, x_1, \dots, x_i], \quad i = 1, 2, \dots, n.$$

$$R_n(x) = f[x, x_0, \dots, x_n](x - x_0) \cdots (x - x_{n-1})(x - x_n).$$

我们注意到,  $N_n(x)$  是一个  $n$  次多项式. 而且通过直接验证可知

$$R_n(x_i) = 0, \quad i = 0, 1, 2, \dots, n.$$

所以

$$f(x_i) = N_n(x_i) + R_n(x_i) = N_n(x_i), \quad i = 0, 1, 2, \dots, n.$$

即  $N_n(x)$  是满足插值条件 (7.2) 的  $n$  次插值多项式. 我们称之为 **Newton 插值多项式**.

由  $N_n(x)$  的表达式, 我们可以立即得到下面的递推公式:

$$N_{n+1}(x) = N_n(x) + f[x_0, x_1, \dots, x_{n+1}] \prod_{i=0}^n (x - x_i).$$

由插值多项式的存在唯一性可知, Newton 插值多项式与 Lagrange 插值多项式是一样的, 即

$$N_n(x) \equiv L_n(x),$$

所以, 它们的插值余项也一样, 即

$$f[x, x_0, \dots, x_n] \prod_{i=0}^n (x - x_i) \equiv \frac{f^{(n+1)}(\xi_x)}{(n+1)!} \prod_{i=0}^n (x - x_i).$$

由此, 我们立即可以得到下面的结论.

**性质 7.2** 设  $f(x) \in C^n[a, b]$  ( $n$  阶连续可导), 且  $f^{(n+1)}(x)$  在  $(a, b)$  内存在. 则对  $\forall x \in [a, b]$ , 存



在  $\xi_x \in (a, b)$  使得

$$f[x, x_0, \dots, x_n] = \frac{f^{(n+1)}(\xi_x)}{(n+1)!}.$$

这就是我们前面提到的差商与导数之间的关系 (7.12), 即

$$f[x_0, x_1, \dots, x_k] = \frac{f^{(k)}(\xi)}{k!}.$$

🔗 Newton 插值余项更具实用性, 因为它仅涉及插值点与插值节点的差商, 而不需要计算导数, 因此在导数不存在的情况下仍然可以使用. 但在计算差商  $f[x, x_0, \dots, x_n]$  时, 由于  $f(x)$  未知, 只能使用插值得到的近似值, 因此得到的差商可能具有一定的偏差.

**例 7.9** 已知函数  $f(x) = \ln(x)$  的函数值如下:

(Interp\_newton\_01.m)

$x$	0.4	0.5	0.6	0.7	0.8
$f(x)$	-0.9163	-0.6931	-0.5108	-0.3567	-0.2231

试分别用 Newton 线性插值和抛物线插值计算  $\ln(0.54)$  的近似值.

**解.** 取插值节点  $x_0 = 0.5, x_1 = 0.6, x_2 = 0.4$ , 做差商表:

$x_i$	$f(x_i)$	一阶差商	二阶差商
0.5	-0.6931		
0.6	-0.5108	1.8230	
0.4	-0.9163	2.0275	-2.0450

于是可得, Newton 线性插值在  $x = 0.54$  上的近似值为

$$N_1(x) = f(x_0) + f[x_0, x_1](x - x_0) \approx -0.6931 + 1.8230(x - 0.5) \approx -0.6202.$$

Newton 抛物线插值在  $x = 0.54$  上的近似值为

$$N_2(x) = N_1(x) + f[x_0, x_1, x_2](x - x_0)(x - x_1) \approx -0.6153.$$

□

**思考**

思考: 这里插值节点顺序为什么这么取?

🔗 在 Newton 插值法中, 我们只需要使用差商表中对角线部分的值.

🔗 增加插值节点时, 新增的插值点必须排在已有插值节点的后面 (不是按值的大小排序).

可以看出, 当增加一个节点时, 牛顿插值公式只需在原来的基础上增加一项, 前面的计算结果仍然可以使用. 与拉格朗日插值相比, 牛顿插值具有灵活增加插值节点的优点!

### 7.3.3 差分

在实际应用中, 为了计算方便, 我们通常采用等距节点, 即:

$$x_i = x_0 + i \times h, \quad i = 0, 1, 2, \dots, n, \quad (7.18)$$

这里  $h > 0$  为步长. 此时, 我们可以用差分简化 Newton 插值公式.

**定义 7.5 (向前差分)** 设  $\{x_i\}$  是由 (7.18) 定义的等距节点, 则函数  $f(x)$  在  $x_i$  处以  $h$  为步长的一阶 (向前) 差分定义为

$$\Delta f_i = f(x_{i+1}) - f(x_i) = f(x_i + h) - f(x_i).$$

类似地, 称

$$\Delta^2 f_i = \Delta(\Delta f_i) = \Delta f_{i+1} - \Delta f_i$$

为  $x_i$  处的二阶差分. 一般地, 称

$$\Delta^n f_i = \Delta(\Delta^{n-1} f_i) = \Delta^{n-1} f_{i+1} - \Delta^{n-1} f_i$$

为  $x_i$  处的  $n$  阶差分.

为了表示方便, 我们引入两个算子:

$$\mathbf{I}f_i = f_i, \quad \mathbf{E}f_i = f_{i+1},$$

分别称为不变算子和位移算子. 于是

$$\Delta f_i = f_{i+1} - f_i = \mathbf{E}f_i - \mathbf{I}f_i = (\mathbf{E} - \mathbf{I})f_i.$$

所以我们可以得到下面的差分与函数值之间的关系:

$$\begin{aligned} \Delta^n f_i &= (\mathbf{E} - \mathbf{I})^n f_i = \left[ \sum_{k=0}^n (-1)^k \binom{n}{k} \mathbf{E}^{n-k} \right] f_i \\ &= \sum_{k=0}^n (-1)^k \frac{n(n-1) \cdots (n-k+1)}{k!} f_{n-k+i}. \end{aligned}$$

反之, 有

$$f_{n+i} = \mathbf{E}^n f_i = (\mathbf{I} + \Delta)^n f_i = \sum_{k=0}^n \binom{n}{k} \Delta^k f_i.$$

### 差分与差商之间的关系

由差商定义可知

$$f[x_0, x_1] = \frac{f_1 - f_0}{x_1 - x_0} = \frac{1}{h} \Delta f_0.$$

事实上, 对于任意两个相邻的节点, 都有上面的等式, 即

$$f[x_k, x_{k+1}] = \frac{f_{k+1} - f_k}{x_{k+1} - x_k} = \frac{1}{h} \Delta f_k.$$

对于任意三个相邻的节点, 有

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



$$\begin{aligned}
 &= \frac{1}{2h} \left( \frac{1}{h} \Delta f_{k+1} - \frac{1}{h} \Delta f_k \right) \\
 &= \frac{1}{2} \frac{1}{h^2} \Delta^2 f_k.
 \end{aligned}$$

一般地, 对于任意  $m+1$  个相邻的节点, 有

$$f[x_k, x_{k+1}, \dots, x_{k+m}] = \frac{1}{m!} \frac{1}{h^m} \Delta^m f_k.$$


所以由差商与导数之间的关系 (7.12) 可知

$$\Delta^m f_k = h^m m! \times f[x_k, x_{k+1}, \dots, x_{k+m}] = h^m f^{(m)}(\xi), \quad \xi \in (x_k, x_{k+m}).$$

### 差分的计算

与差商表类似, 我们也可以通过下面的差分表来计算差分.

$x_i$	$f(x_i)$	一阶差分	二阶差分	三阶差分	$\dots$	$n$ 阶差分
$x_0$	$f(x_0)$	$\Delta f_0$	$\Delta^2 f_0$	$\Delta^3 f_0$	$\dots$	$\Delta^n f_0$
$x_1$	$f(x_1)$	$\Delta f_1$	$\Delta^2 f_1$	$\Delta^3 f_1$		
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$		
$x_{n-2}$	$f(x_{n-2})$	$\Delta f_{n-2}$	$\Delta^2 f_{n-2}$			
$x_{n-1}$	$f(x_{n-1})$	$\Delta f_{n-1}$				
$x_n$	$f(x_n)$					

 MATLAB 中计算差分的函数为: **diff(x)**, 可以计算高阶差分, 如: **diff(x,2)**.

### Newton 向前插值公式

如果采用等距插值节点  $x_i = x_0 + ih$ , 其中  $h > 0$  为步长, 则我们可以用差分来简化 Newton 插值公式. 设  $x = x_0 + th$ , 则

$$\begin{aligned}
 N_n(x) &= N_n(x_0 + th) \\
 &= f(x_0) + t\Delta f_0 + \frac{t(t-1)}{2!} \Delta^2 f_0 + \frac{t(t-1)(t-2)}{3!} \Delta^3 f_0 + \dots \\
 &\quad + \frac{t(t-1)\dots(t-n+1)}{n!} \Delta^n f_0.
 \end{aligned} \tag{7.19}$$

这就是 **Newton 向前插值公式**. 插值余项为

$$R_n(x) = \frac{f^{(n+1)}(\xi_x)}{(n+1)!} t(t-1)\dots(t-n)h^{n+1}.$$

**例 7.10** 给定  $f(x) = \cos(x)$  在等距节点  $0 : 0.1 : 0.5$  处的函数值, 试用 4 次 Newton 向前插值公式计算  $f(0.048)$  的近似值, 并估计误差. (Interp\_newton\_02.m)

**解.** 取等距节点  $x = 0 : 0.1 : 0.4$ , 做差分表

$x_i$	$f(x_i)$	一阶差分	二阶差分	三阶差分	四阶差分
0.0	1.00000	-0.00500	-0.00993	-0.00013	-0.00012
0.1	0.99500	-0.01493	-0.00980	-0.00025	
0.2	0.98007	-0.02473	-0.00955		
0.3	0.95534	-0.03428			
0.4	0.92106				


插值点  $x = 0.048$ , 则  $t = (x - x_0)/h = 0.48$ . 所以由 Newton 向前插值公式 (7.19) 可知,  $f(0.048)$  的近似值为

$$\begin{aligned}
 N_4(0.048) &= 1.00000 + 0.48 \times (-0.00500) \\
 &\quad + \frac{1}{2!} \times 0.48(0.48 - 1)(-0.00993) \\
 &\quad + \frac{1}{3!} \times 0.48(0.48 - 1)(0.48 - 2)(-0.00013) \\
 &\quad + \frac{1}{4!} \times 0.48(0.48 - 1)(0.48 - 2)(0.48 - 3)(-0.00012) \\
 &\approx 0.99884.
 \end{aligned}$$

余项为

$$\begin{aligned}
 |R_4(0.048)| &= \left| \frac{f^{(5)}(\xi)}{5!} t(t-1)(t-2)(t-3)(t-4)h^5 \right| \\
 &\leq \frac{h^5}{5!} \cdot |t(t-1)(t-2)(t-3)(t-4)| \cdot \max_{0 \leq x \leq 0.4} |f^{(5)}(x)| \\
 &\approx 1.09212 \times 10^{-7}.
 \end{aligned}$$

% 四次 Newton 向前插值公式只需用到前面 5 个等距插值节点. □

 Newton 向前插值公式只需用到差分表的第一行.

## 向后差分与中心差分

与向前差分类似, 我们还可以定义**向后差分**:

$$\begin{aligned}
 \nabla f_i &= f(x_i) - f(x_{i-1}), \\
 \nabla^k f_i &= \nabla(\nabla^{k-1} f_i) = \nabla^{k-1} f_i - \nabla^{k-1} f_{i-1}, \quad k = 2, 3, \dots
 \end{aligned}$$

和**中心差分**:

$$\begin{aligned}
 \delta f_i &= f(x_{i+\frac{1}{2}}) - f(x_{i-\frac{1}{2}}), \\
 \delta^k f_i &= \delta(\delta^{k-1} f_i) = \delta^{k-1} f_{i+\frac{1}{2}} - \delta^{k-1} f_{i-\frac{1}{2}}, \quad k = 2, 3, \dots
 \end{aligned}$$





## 7.4 Hermite 插值

### 为什么 Hermite 插值

在许多实际应用中, 不仅要求函数值相等, 而且还要求若干阶导数也相等, 如机翼设计等.

设插值节点为  $x_0, x_1, \dots, x_n$ , 如果要求插值多项式满足

$$p(x_i) = f(x_i), p'(x_i) = f'(x_i), p''(x_i) = f''(x_i), \dots, p^{(m)}(x_i) = f^{(m)}(x_i).$$

则计算这类插值多项式的方法就称为 **Hermite 插值**.

✎ 在 Hermite 插值中, 并不一定需要在所有插值节点上的导数都相等, 在有些情况下, 可能只需要在部分插值点上的导数值相等即可.

### 7.4.1 重节点差商与 Taylor 插值

首先介绍差商的一个重要性质.

**定理 7.3** 设  $x_0, x_1, \dots, x_n$  为  $[a, b]$  上的互异节点,  $f(x) \in C^n[a, b]$ , 则  $f[x_0, x_1, \dots, x_n]$  是其变量的连续函数.

例如,  $f[x, y] = \frac{f(y) - f(x)}{y - x}$  关于  $x$  和  $y$  都连续, 且当  $y \rightarrow x$  时有

$$f[x, x] \triangleq \lim_{y \rightarrow x} f[x, y] = f'(x).$$

这就是  $f$  在  $x$  点的一阶重节点差商.

一般地,  $f$  在点  $x$  的  $n$  阶 **重节点差商** 定义为

$$f[\underbrace{x, x, \dots, x}_{n+1 \text{ 个}}] \triangleq \lim_{x_i \rightarrow x} f[x, x_1, x_2, \dots, x_n] = \frac{1}{n!} f^{(n)}(x).$$

在 Newton 插值公式 (7.17) 中, 令  $x_i \rightarrow x_0, i = 1, 2, \dots, n$ , 则

$$\begin{aligned} N_n(x) &= f(x_0) + f[x_0, x_1](x - x_0) + f[x_0, x_1, x_2](x - x_0)(x - x_1) + \dots \\ &\quad + f[x_0, x_1, \dots, x_n](x - x_0) \cdots (x - x_{n-1}) \\ &= f(x_0) + f'(x_0)(x - x_0) + \frac{1}{2!} f''(x_0)(x - x_0)^2 + \dots + \frac{1}{n!} f^{(n)}(x_0)(x - x_0)^n. \end{aligned}$$

这就是 **Taylor 插值**, 也即是  $f(x)$  在  $x_0$  点的 Taylor 展开式的前  $n+1$  项之和. 余项为

$$R_n(x) = \frac{1}{(n+1)!} f^{(n+1)}(\xi)(x - x_0)^{n+1}.$$

✎ Taylor 插值就是在一个插值点  $x_0$  的  $n$  次 Hermite 插值.

### 7.4.2 两个典型的 Hermite 插值

一般来说, 给定  $m+1$  个插值条件, 就可以构造出一个  $m$  次 Hermite 插值多项式. 这里介绍两个典型的 Hermite 插值: **三点三次 Hermite 插值** 和 **两点三次 Hermite 插值**.

## (1) 三点三次 Hermite 插值

设插值节点为  $x_0, x_1, x_2$ , 则满足插值条件

$$p(x_0) = f(x_0), \quad p(x_1) = f(x_1), \quad p(x_2) = f(x_2), \quad p'(x_1) = f'(x_1),$$

的多项式  $p(x)$  就称为**三点三次 Hermite 插值**多项式.

由于  $p(x_i) = f(x_i)$ , 仿照 Newton 插值多项式, 我们可以设

$$p(x) = f(x_0) + f[x_0, x_1](x - x_0) + f[x_0, x_1, x_2](x - x_0)(x - x_1) + a(x - x_0)(x - x_1)(x - x_2). \quad (7.20)$$

其中  $a$  是待定系数. 将  $p'(x_1) = f'(x_1)$  代入, 可得

$$a = \frac{f'(x_1) - f[x_0, x_1] - f[x_0, x_1, x_2](x_1 - x_0)}{(x_1 - x_0)(x_1 - x_2)}.$$

根据插值条件, 我们可以将插值余项写成

$$R_3(x) = f(x) - p(x) = K(x)(x - x_0)(x - x_1)^2(x - x_2),$$

其中  $K(x)$  待定. 与 Lagrange 插值余项公式的推导过程类似, 可得

$$R_3(x) = \frac{f^{(4)}(\xi_x)}{4!}(x - x_0)(x - x_1)^2(x - x_2),$$

其中  $\xi_x$  位于由  $x_0, x_1, x_2$  和  $x$  所界定的区间内.

**例 7.11** 已知函数  $f(x) = x^{\frac{3}{2}}$ , 插值条件为

$$f\left(\frac{1}{4}\right) = \frac{1}{8}, \quad f(1) = 1, \quad f\left(\frac{9}{4}\right) = \frac{27}{8}, \quad f'(1) = \frac{3}{2},$$

是给出三次 Hermite 插值多项式, 并写出余项. (计算过程中不要做近似计算)

**解.** 做差商表

$x_i$	$f(x_i)$	一阶差商	二阶差商
1/4	1/8		
1	1	7/6	
9/4	27/8	19/10	11/30

所以三次插值多项式可设为

$$p(x) = \frac{1}{8} + \frac{7}{6}\left(x - \frac{1}{4}\right) + \frac{11}{30}\left(x - \frac{1}{4}\right)(x - 1) + \alpha\left(x - \frac{1}{4}\right)(x - 1)\left(x - \frac{9}{4}\right).$$

将  $p'(1) = f'(1) = \frac{3}{2}$  代入, 可得  $\alpha = -\frac{14}{225}$ , 所以三次 Hermite 插值多项式为

$$\begin{aligned} p(x) &= \frac{1}{8} + \frac{7}{6}\left(x - \frac{1}{4}\right) + \frac{11}{30}\left(x - \frac{1}{4}\right)(x - 1) - \frac{14}{225}\left(x - \frac{1}{4}\right)(x - 1)\left(x - \frac{9}{4}\right) \\ &= -\frac{14}{225}x^3 + \frac{263}{450}x^2 + \frac{233}{450}x - \frac{1}{25}. \end{aligned}$$



余项为

$$\begin{aligned} R(x) &= f(x) - p(x) = \frac{f^{(4)}(\xi_x)}{4!} \left(x - \frac{1}{4}\right) (x-1)^2 \left(x - \frac{9}{4}\right) \\ &= \frac{9\xi_x^{-5/2}}{16 \times 4!} \left(x - \frac{1}{4}\right) (x-1)^2 \left(x - \frac{9}{4}\right). \end{aligned}$$

□

## (2) 两点三次 Hermite 插值

设插值节点为  $x_0, x_1$ , 则满足插值条件

$$p(x_0) = f(x_0), \quad p(x_1) = f(x_1), \quad p'(x_0) = f'(x_0), \quad p'(x_1) = f'(x_1)$$

的多项式  $p(x)$  就称为**两点三次 Hermite 插值**多项式, 记为  $H_3(x)$ .

仿照 Lagrange 多项式的思想, 可设

$$H_3(x) = a_0\alpha_0(x) + a_1\alpha_1(x) + b_0\beta_0(x) + b_1\beta_1(x),$$

其中  $\alpha_0(x), \alpha_1(x), \beta_0(x), \beta_1(x)$  均为三次多项式, 且满足

$$\alpha_0(x_0) = 1, \alpha_0(x_1) = 0, \alpha'_0(x_0) = 0, \alpha'_0(x_1) = 0;$$

$$\alpha_1(x_0) = 0, \alpha_1(x_1) = 1, \alpha'_1(x_0) = 0, \alpha'_1(x_1) = 0;$$

$$\beta_0(x_0) = 0, \beta_0(x_1) = 0, \beta'_0(x_0) = 1, \beta'_0(x_1) = 0;$$

$$\beta_1(x_0) = 0, \beta_1(x_1) = 0, \beta'_1(x_0) = 0, \beta'_1(x_1) = 1.$$

根据插值条件可得

$$H_3(x) = f(x_0)\alpha_0(x) + f(x_1)\alpha_1(x) + f'(x_0)\beta_0(x) + f'(x_1)\beta_1(x).$$

剩下的问题就是如何确定  $\alpha_0(x), \alpha_1(x), \beta_0(x), \beta_1(x)$  的表达式.

我们首先考虑  $\alpha_0(x)$ . 由于  $\alpha_0(x)$  是三次多项式, 且  $\alpha_0(x_1) = 0, \alpha'_0(x_1) = 0$ , 所以可设

$$\alpha_0(x) = (ax + b) \left( \frac{x - x_1}{x_0 - x_1} \right)^2.$$

将  $\alpha_0(x_0) = 1, \alpha'_0(x_0) = 0$  代入可得

$$a = \frac{2}{x_1 - x_0}, \quad b = \frac{x_1 - 3x_0}{x_1 - x_0} = 1 - \frac{2x_0}{x_1 - x_0}.$$

所以

$$\alpha_0(x) = \left( 1 + 2 \frac{x - x_0}{x_1 - x_0} \right) \left( \frac{x - x_1}{x_0 - x_1} \right)^2.$$

同理可得

$$\alpha_1(x) = \left( 1 + 2 \frac{x - x_1}{x_0 - x_1} \right) \left( \frac{x - x_0}{x_1 - x_0} \right)^2.$$

下面考虑  $\beta_0(x)$ . 根据插值条件  $\beta_0(x_0) = 0, \beta_0(x_1) = 0, \beta'_0(x_1) = 0$ , 可设

$$\beta_0(x) = a(x - x_0) \left( \frac{x - x_1}{x_0 - x_1} \right)^2.$$



将  $\beta'_0(x_0) = 1$  代入可得  $a = 1$ , 所以

$$\beta_0(x) = (x - x_0) \left( \frac{x - x_1}{x_0 - x_1} \right)^2.$$

同理可得

$$\beta_1(x) = (x - x_1) \left( \frac{x - x_0}{x_1 - x_0} \right)^2.$$

所以

$$\begin{aligned} H_3(x) = & f(x_0) \left( 1 + 2 \frac{x - x_0}{x_1 - x_0} \right) \left( \frac{x - x_1}{x_0 - x_1} \right)^2 + f(x_1) \left( 1 + 2 \frac{x - x_1}{x_0 - x_1} \right) \left( \frac{x - x_0}{x_1 - x_0} \right)^2 \\ & + f'(x_0)(x - x_0) \left( \frac{x - x_1}{x_0 - x_1} \right)^2 + f'(x_1)(x - x_1) \left( \frac{x - x_0}{x_1 - x_0} \right)^2. \end{aligned}$$

(7.21)

插值余项为

$$R_3(x) = \frac{f^{(4)}(\xi_x)}{4!} (x - x_0)^2 (x - x_1)^2.$$



## 7.5 分段低次插值

### 为什么分段插值

由前面的例 7.7 (即 Runge 现象) 可知, 当  $n \rightarrow \infty$  时, 插值多项式  $L_n(x)$  并不一定收敛于  $f(x)$ . 事实上, 对于 7.7, 可以证明, 存在常数  $c \approx 3.63$ , 当  $|x| \leq c$  时,  $\lim_{n \rightarrow \infty} L_n(x) = f(x)$ , 而当  $|x| > c$  时,  $\{L_n(x)\}$  发散.

怎样才能构造出收敛的插值方法呢 (即可以使得插值误差趋于零)? 这就需要从插值余项入手. 由插值余项公式可知通过观察发现, 插值余项通常有两部分组成:  $f(x)$  的导数和  $\omega_{n+1}(x)$ , 即

$$\max_{a \leq x \leq b} |R_n(x)| = \max_{a \leq x \leq b} \left| \frac{f^{(n+1)}(\xi_x)}{(n+1)!} \omega_{n+1}(x) \right| \leq \max_{a \leq x \leq b} \left| \frac{f^{(n+1)}(\xi_x)}{(n+1)!} \right| \cdot \max_{a \leq x \leq b} |\omega_{n+1}(x)|.$$

由于  $f(x)$  是给定的, 因此上式右端的第一项也是相对确定的. 所以我们只能想办法尽量降低  $\max_{a \leq x \leq b} |\omega_{n+1}(x)|$  的大小. 显然该值与插值区间  $[a, b]$  的长度有关: 长度越小则值也越小!

于是, 人们提出了一个切实可行的方法: **分段插值方法**, 即将插值区间分割成若干小区间, 然后在每个小区间上进行插值, 这样就可以降低每个小区间上的插值误差, 从而减小在整个区间上插值的误差.

下面我们介绍两个常用的分段插值方法: **分段线性插值**和**分段三次 Hermite 插值**.

### 7.5.1 分段线性插值

**定义 7.6** 设  $a = x_0 < x_1 < \cdots < x_n = b$  为  $[a, b]$  上的互异节点, 已知  $f(x)$  在这些节点上的函数值为  $f_0, f_1, \dots, f_n$ . 求分段函数  $I_h(x)$  满足

- (1)  $I_h(x) \in C[a, b]$ ;
- (2)  $I_h(x_k) = f_k, \quad k = 0, 1, 2, \dots, n$ ;
- (3)  $I_h(x)$  在每个小区间  $[x_k, x_{k+1}]$  上是线性多项式.

这就是**分段线性插值**,  $I_h(x)$  就称为  $f(x)$  在  $[a, b]$  上的**分段线性插值函数**.

由定义直接可知  $I_h(x)$  在小区间  $[x_k, x_{k+1}]$  上的表达式为

$$I_h(x) = \frac{x - x_{k+1}}{x_k - x_{k+1}} f_k + \frac{x - x_k}{x_{k+1} - x_k} f_{k+1}, \quad x \in [x_k, x_{k+1}], \quad (7.22)$$

且在  $[x_k, x_{k+1}]$  上余项满足

$$\max_{x_k \leq x \leq x_{k+1}} |f(x) - I_h(x)| \leq \frac{1}{2!} \max_{x_k \leq x \leq x_{k+1}} |f''(x)| \cdot \max_{x_k \leq x \leq x_{k+1}} |(x - x_k)(x - x_{k+1})| \quad (7.23)$$

$$\leq \frac{h_k^2}{8} \max_{x_k \leq x \leq x_{k+1}} |f''(x)|, \quad (7.24)$$

其中  $h_k = x_{k+1} - x_k$ . 令  $h = \max_{0 \leq k \leq n-1} \{h_k\}$ , 我们有下面的结论.

**定理 7.4** 若  $f(x) \in C^2[a, b]$ , 则分段线性插值函数  $I_h(x)$  满足

$$\max_{a \leq x \leq b} |f(x) - I_h(x)| \leq \frac{M_2}{8} h^2,$$

其中  $M_2 = \max_{a \leq x \leq b} |f''(x)|$ . 所以

$$\lim_{h \rightarrow 0} I_h(x) = f(x)$$

在  $[a, b]$  上一致成立, 即  $I_h(x)$  在  $[a, b]$  上一致收敛到  $f(x)$ .


**证明.** 由 (7.24) 可知

$$\max_{x_k \leq x \leq x_{k+1}} |f(x) - I_h(x)| \leq \frac{h_k^2 M_2}{8} \leq \frac{h^2 M_2}{8}.$$

所以

$$\max_{a \leq x \leq b} |f(x) - I_h(x)| \leq \max_{0 \leq k \leq n-1} \max_{x_k \leq x \leq x_{k+1}} |f(x) - I_h(x)| \leq \max_{0 \leq k \leq n-1} \frac{h^2 M_2}{8} = \frac{h^2 M_2}{8}.$$

□

 分段线性插值简单易用, 但插值函数在插值节点不可导.

**思考**

如果是分段抛物线插值, 则结论是怎样的?

## 7.5.2 分段三次 Hermite 插值

**定义 7.7** 设  $a = x_0 < x_1 < \cdots < x_n = b$  为  $[a, b]$  上的互异节点, 已知  $f(x)$  在这些节点上的函数值和导数分别为  $f_0, f_1, \dots, f_n$  和  $f'_0, f'_1, \dots, f'_n$ . 求分段函数  $I_h(x)$  满足

- (1)  $I_h(x) \in C^1[a, b]$ ;
- (2)  $I_h(x_k) = f_k, I'_h(x_k) = f'_k, \quad k = 0, 1, 2, \dots, n$ ;
- (3)  $I_h(x)$  在每个小区间  $[x_k, x_{k+1}]$  上是三次多项式.

这就是**分段三次 Hermite 插值**,  $I_h(x)$  就称为  $f(x)$  在  $[a, b]$  上的**分段三次 Hermite 插值函数**.

由两点三次 Hermite 插值公式 (7.21) 可知,  $I_h(x)$  在小区间  $[x_k, x_{k+1}]$  上的表达式为

$$\begin{aligned} I_h(x) = & \left(1 + 2 \frac{x - x_k}{x_{k+1} - x_k}\right) \left(\frac{x - x_{k+1}}{x_k - x_{k+1}}\right)^2 f_k \\ & + \left(1 + 2 \frac{x - x_{k+1}}{x_k - x_{k+1}}\right) \left(\frac{x - x_k}{x_{k+1} - x_k}\right)^2 f_{k+1} \\ & + (x - x_k) \left(\frac{x - x_{k+1}}{x_k - x_{k+1}}\right)^2 f'_k + (x - x_{k+1}) \left(\frac{x - x_k}{x_{k+1} - x_k}\right)^2 f'_{k+1}, \end{aligned} \quad (7.25)$$

$$x \in [x_k, x_{k+1}].$$

由两点三次 Hermite 插值法的余项公式, 可知

$$\max_{x_k \leq x \leq x_{k+1}} |f(x) - I_h(x)| \leq \frac{h_k^4}{384} \max_{x_k \leq x \leq x_{k+1}} |f^{(4)}(x)|,$$

其中  $h_k = x_{k+1} - x_k$ . 令  $h = \max_{0 \leq k \leq n-1} \{h_k\}$ , 我们有下面的结论.



**定理 7.5** 若  $f(x) \in C^4[a, b]$ , 则分段三次 Hermite 插值函数  $I_h(x)$  满足

$$\max_{a \leq x \leq b} |f(x) - I_h(x)| \leq \frac{M_4}{384} h^4,$$

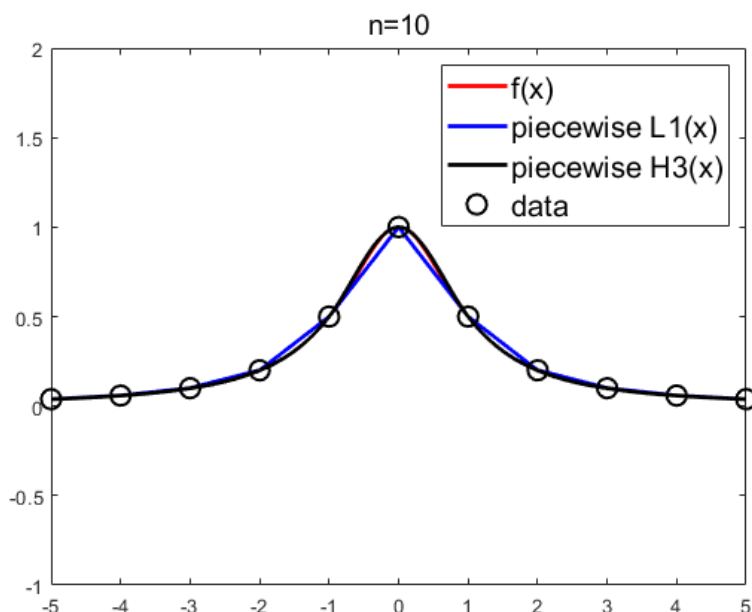
其中  $M_4 = \max_{a \leq x \leq b} |f^{(4)}(x)|$ . 所以,  $I_h(x)$  在区间  $[a, b]$  上一致收敛到  $f(x)$ .

由余项公式可知, 当  $h$  比较小时, 分段三次 Hermite 插值比分段线性插值具有更高的精度, 且  $h$  越小, 误差下降也越快.

分段三次 Hermite 插值的缺点: 需要知道  $f(x)$  在插值节点的导数值, 而且插值函数只有一阶导数, 光滑度不高.

**例 7.12** 设  $f(x) = \frac{1}{1+x^2}$ ,  $x \in [-5, 5]$ , 取等距插值节点  $x_k = -5 + k$  (即 10 等分插值区间). 试分别用分段线性插值和分段三次 Hermite 插值画出  $f(x)$  的近似图像.

(Interp\_pp\_01.m)



## 7.6 三次样条插值

为了增加分段插值函数的光滑性, 我们可以使用样条函数进行插值. 目前常用的为三次样条函数, 它具有二阶连续导数.

**定义 7.8** 设  $a = x_0 < x_1 < \cdots < x_n = b$  为  $[a, b]$  上的互异节点, 已知  $f(x)$  在这些节点上的函数值为  $f(x_k) = f_k, k = 0, 1, \dots, n$ . 求插值函数  $S(x)$  满足

- (1)  $S(x) \in C^2[a, b]$ , 即二阶连续可导;
- (2)  $S(x_k) = y_k, k = 0, 1, 2, \dots, n$ ;
- (3)  $S(x)$  是分段三次函数, 即在每个小区间  $[x_k, x_{k+1}]$  上是三次多项式.

这就是**三次样条插值**,  $S(x)$  就称为  $f(x)$  在  $[a, b]$  上的**三次样条插值函数**.

### 7.6.1 三次样条函数

**定义 7.9 (三次样条函数)** 设  $a = x_0 < x_1 < \cdots < x_n = b$  为  $[a, b]$  上的互异节点, 若函数  $S(x) \in C^2[a, b]$ , 且在每个小区间  $[x_k, x_{k+1}]$  上是三次多项式, 则称其为**三次样条函数**.

我们可以将  $S(x)$  在小区间  $[x_k, x_{k+1}]$  上的表达式记为  $s_k(x)$ , 即

$$S(x) = s_k(x), \quad x \in [x_k, x_{k+1}], \quad k = 0, 1, 2, \dots, n-1,$$

其中  $s_k(x)$  是三次多项式, 且满足

$$s_k(x_k) = f_k, \quad s_k(x_{k+1}) = f_{k+1}. \quad (7.26)$$

于是

$$S(x) = \begin{cases} s_0(x), & x \in [x_0, x_1] \\ s_1(x), & x \in [x_1, x_2] \\ \vdots \\ s_{n-1}(x), & x \in [x_{n-1}, x_n]. \end{cases} \quad (7.27)$$

由于  $S(x) \in C^2[a, b]$ , 所以  $S'(x_k^-) = S'(x_k^+)$ ,  $S''(x_k^-) = S''(x_k^+)$ , 即

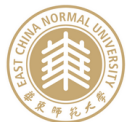
$$s'_{k-1}(x_k^-) = s'_k(x_k^+), \quad s''_{k-1}(x_k^-) = s''_k(x_k^+), \quad k = 1, 2, \dots, n-1. \quad (7.28)$$

每个  $s_k(x)$  均为三次多项式, 有 4 个待定系数, 所以共有  $4n$  个待定系数, 故需  $4n$  个方程. 由 (7.26) 和 (7.28) 可以得到  $2n + 2(n-1) = 4n - 2$  个方程, 还缺 2 个方程!

实际问题中, 通常会对样条函数  $S(x)$  在两个端点  $x = a$  和  $x = b$  处的状态有一定的要求, 这就是**边界条件**.

### 7.6.2 边界条件

我们这里介绍三类常用的边界条件.





(1) **第一类边界条件**: 指定函数在两端点处的一阶导数, 即

$$S'(x_0) = f'_0, \quad S'(x_n) = f'_n$$

(2) **第二类边界条件**: 指定函数在端点处的二阶导数, 即

$$S''(x_0) = f''_0, \quad S''(x_n) = f''_n.$$

如果  $f''_0 = f''_n = 0$ , 则称为**自然边界条件**, 此时  $S(x)$  称为**自然样条函数**.

(3) **第三类边界条件**: 假定  $f(x)$  是周期函数, 并设  $x_n - x_0$  是一个周期, 于是要求  $S(x)$  也是周期函数, 即

$$S(x_0) = S(x_n), \quad S'(x_0^+) = S'(x_n^-), \quad S''(x_0^+) = S''(x_n^-).$$

此时  $S(x)$  称为**周期样条函数**.

由于  $S(x_0) = f_0$  和  $S(x_n) = f_n$  是已知的, 所以第三类边界中只有后面两个才是新增加的约束.

### 7.6.3 三次样条函数的计算

由于  $S(x)$  二阶可导, 所以可设

$$S''(x_k) = M_k, \quad k = 0, 1, 2, \dots, n,$$

下面我们用  $M_k$  来表示  $S(x)$ . 考虑  $S(x)$  在区间  $[x_k, x_{k+1}]$  上的表达式  $s_k(x)$ , 满足

$$s''_k(x_k) = M_k, \quad s''_k(x_{k+1}) = M_{k+1}.$$

由于  $s_k(x)$  是三次多项式, 故  $s''_k(x)$  为线性函数. 所以由线性插值公式可知

$$s''_k(x) = \frac{x_{k+1} - x}{h_k} M_k + \frac{x - x_k}{h_k} M_{k+1},$$

其中  $h_k = x_{k+1} - x_k$ . 两边在  $[x_k, x_{k+1}]$  上积分两次后可得

$$s_k(x) = \frac{(x_{k+1} - x)^3}{6h_k} M_k + \frac{(x - x_k)^3}{6h_k} M_{k+1} + c_1 x + c_2, \quad (7.29)$$

其中  $c_1, c_2$  为积分常数. 将  $s_k(x_k) = f_k, s_k(x_{k+1}) = f_{k+1}$  代入后可得

$$c_1 = \frac{1}{h_k}(f_{k+1} - f_k) - \frac{h_k}{6}(M_{k+1} - M_k) = \frac{1}{h_k} \left[ \left( f_{k+1} - \frac{M_{k+1}h_k^2}{6} \right) - \left( f_k - \frac{M_k h_k^2}{6} \right) \right],$$

$$c_2 = f_k - \frac{M_k h_k^2}{6} - c_1 x_k = \frac{x_{k+1}}{h_k} \left( f_k - \frac{M_k h_k^2}{6} \right) - \frac{x_k}{h_k} \left( f_{k+1} - \frac{M_{k+1} h_k^2}{6} \right).$$

代入 (7.29), 整理后可得

$$s_k(x) = \frac{(x_{k+1} - x)^3}{6h_k} M_k + \frac{(x - x_k)^3}{6h_k} M_{k+1} + \frac{x_{k+1} - x}{h_k} \left( f_k - \frac{M_k h_k^2}{6} \right) + \frac{x - x_k}{h_k} \left( f_{k+1} - \frac{M_{k+1} h_k^2}{6} \right). \quad (7.30)$$

即  $s_k(x)$  可表示成  $x_{k+1} - x$  和  $x - x_k$  的奇次项的线性组合.

将  $x_{k+1} = x_k + h_k$  代入 (7.30), 整理后可得

$$s_k(x) = \frac{M_{k+1} - M_k}{6h_k}(x - x_k)^3 + \frac{M_k}{2}(x - x_k)^2 + \left( \frac{f_{k+1} - f_k}{h_k} - \frac{h_k(M_{k+1} + 2M_k)}{6} \right)(x - x_k) + f_k. \quad (7.31)$$

现在, 问题转化为如何确定  $M_0, M_1, \dots, M_n$  的值?

由于  $S(x) \in C^2[a, b]$ , 所以在节点处的一阶导数也存在, 故

$$S'(x_k^-) = S'(x_k^+), \quad k = 1, 2, \dots, n-1,$$

也即

$$s'_{k-1}(x_k^-) = s'_k(x_k^+).$$

所以可得方程

$$\frac{h_{k-1}}{6}M_{k-1} + \frac{h_{k-1} + h_k}{3}M_k + \frac{h_k}{6}M_{k+1} = \frac{f_{k+1} - f_k}{h_k} - \frac{f_k - f_{k-1}}{h_{k-1}}.$$

为了书写方便, 我们记

$$\mu_k = \frac{h_{k-1}}{h_{k-1} + h_k}, \quad \lambda_k = \frac{h_k}{h_{k-1} + h_k},$$

$$d_k = \frac{6(f[x_k, x_{k+1}] - f[x_{k-1}, x_k])}{h_{k-1} + h_k} = 6f[x_{k-1}, x_k, x_{k+1}],$$

则上面的方程可写为

$$\mu_k M_{k-1} + 2M_k + \lambda_k M_{k+1} = d_k, \quad k = 1, 2, \dots, n-1. \quad (7.32)$$

上述方程中的系数有个重要性质:  $\mu_k + \lambda_k = 1$

方程 (7.32) 也可以写成

$$h_{k-1}M_{k-1} + 2(h_{k-1} + h_k)M_k + h_kM_{k+1} = 6(f[x_k, x_{k+1}] - f[x_{k-1}, x_k])$$

这里有  $n+1$  个变量, 但只有  $n-1$  个方程. 此时需要通过边界条件增加两个方程. 下面对三种边界条件分别讨论.

### (1) 第一类边界条件

给出函数在两端点处的一阶导数:  $S'(x_0) = f'_0$  和  $S'(x_n) = f'_n$ , 即

$$s'_0(x_0^+) = f'_0, \quad s'_n(x_n^-) = f'_n.$$

因此可得

$$2M_0 + M_1 = \frac{6}{h_0}(f[x_0, x_1] - f'_0)$$

$$M_{n-1} + 2M_n = \frac{6}{h_{n-1}}(f'_n - f[x_{n-1}, x_n]).$$



令  $d_0 = \frac{6}{h_0}(f[x_0, x_1] - f'_0)$ ,  $d_n = \frac{6}{h_{n-1}}(f'_n - f[x_{n-1}, x_n])$ , 则上式与 (7.32) 联立可得方程组

$$\begin{bmatrix} 2 & 1 & & & \\ \mu_1 & 2 & \lambda_1 & & \\ & \ddots & \ddots & \ddots & \\ & & \mu_{n-1} & 2 & \lambda_{n-1} \\ & & & 1 & 2 \end{bmatrix} \begin{bmatrix} M_0 \\ M_1 \\ \vdots \\ M_{n-1} \\ M_n \end{bmatrix} = \begin{bmatrix} d_0 \\ d_1 \\ \vdots \\ d_{n-1} \\ d_n \end{bmatrix}. \quad (7.33)$$

这是一个  $(n+1) \times (n+1)$  的线性方程组, 且系数矩阵严格对角占优, 因此存在唯一解. 我们可以使用 Gauss 消去法或追赶法来求解 (具体过程在后面的章节中描述).

## (2) 第二类边界条件

给出函数在端点处的二阶导数:  $S''(x_0) = f''_0$  和  $S''(x_n) = f''_n$ , 即

$$M_0 = f''_0, \quad M_n = f''_n,$$

可得方程组

$$\begin{bmatrix} 2 & \lambda_1 & & & \\ \mu_2 & 2 & \lambda_2 & & \\ & \ddots & \ddots & \ddots & \\ & & \mu_{n-2} & 2 & \lambda_{n-2} \\ & & & \mu_{n-1} & 2 \end{bmatrix} \begin{bmatrix} M_1 \\ M_2 \\ \vdots \\ M_{n-2} \\ M_{n-1} \end{bmatrix} = \begin{bmatrix} d_1 - \mu_1 f''_0 \\ d_2 \\ \vdots \\ d_{n-2} \\ d_{n-1} - \lambda_{n-1} f''_n \end{bmatrix}. \quad (7.34)$$

这是一个  $(n-1) \times (n-1)$  的线性方程组, 系数矩阵也严格对角占优, 因此存在唯一解.

## (3) 第三类边界条件

要求  $S(x)$  是周期函数, 满足

$$S'(x_0^+) = S'(x_n^-), \quad S''(x_0^+) = S''(x_n^-),$$

即

$$s'_0(x_0^+) = s'_n(x_n^-), \quad s''(x_0^+) = s''(x_n^-).$$

可得

$$\lambda_n M_1 + \mu_n M_{n-1} + 2M_n = d_n, \quad M_0 = M_n,$$

其中

$$\lambda_n = \frac{h_0}{h_0 + h_{n-1}}, \quad \mu_n = \frac{h_{n-1}}{h_0 + h_{n-1}}, \quad d_n = \frac{6(f[x_0, x_1] - f[x_{n-1}, x_n])}{h_0 + h_{n-1}}.$$



与 (7.32) 联立可得方程组

$$\begin{bmatrix} 2 & \lambda_1 & & \mu_1 \\ \mu_2 & 2 & \lambda_2 & \\ & \ddots & \ddots & \ddots \\ & & \mu_{n-1} & 2 & \lambda_{n-1} \\ \lambda_n & & & \mu_n & 2 \end{bmatrix} \begin{bmatrix} M_1 \\ M_2 \\ \vdots \\ M_{n-1} \\ M_n \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_{n-1} \\ d_n \end{bmatrix}. \quad (7.35)$$

这是一个  $n \times n$  的线性方程组, 系数矩阵也严格对角占优, 因此存在唯一解.

由于  $M_k$  在力学中解释为细梁在  $x_k$  截面处的弯矩, 因此方程组 (7.33), (7.34) 和 (7.35) 在工程中称为**三弯矩方程**.

### 具体计算过程

由上面的分析可知, 三次样条插值的具体计算过程如下:

- (1) 根据给定的插值条件和边界条件写出关于  $M_0, M_1, \dots, M_n$  的线性方程组;
- (2) 解线性方程组, 求得  $M_k$ ;
- (3) 将  $M_k$  代入  $s_k(x)$  的表达式 (7.31), 得到  $S(x)$  在插值区间  $[a, b]$  上的分段表达式.

MATLAB 提供了计算三次样条插值的函数: **spline**, 其输出结果为  $[a_3, a_2, a_1, a_0]$ , 表示

$$s_k(x) = a_3(x - x_k)^3 + a_2(x - x_k)^2 + a_1(x - x_k) + a_0,$$

因此, 我们在计算时也可以将  $s_k(x)$  写成上述形式, 即 (7.31) 式.

三次样条插值函数具有二阶可导, 但仅利用插值节点上的函数值, 再加上边界条件.

**例 7.13** 函数  $f(x)$  定义在  $[27.7, 30]$  上, 插值节点及相应函数值下表, 试求三次样条插值多项式  $S(x)$ , 满足边界条件  $S'(27.7) = 3.0, S'(30) = -4.0$ . (Interp\_spline\_01.m)

$x$	27.7	28	29	30
$f(x)$	4.1	4.3	4.1	3.0

**解.** 做差商表

$x_i$	$f(x_i)$	一阶差商	二阶差商
27.7	4.1		
28	4.3	2/3	
29	4.1	-0.2	-2/3
30	3.0	-1.1	-9/20



由题意可知  $h_0 = 0.3, h_1 = 1.0, h_2 = 1.0$ , 所以

$$\begin{aligned}\mu_1 &= \frac{h_0}{h_0 + h_1} = \frac{3}{13}, \quad \lambda_1 = 1 - \mu_1 = \frac{10}{13}, \\ \mu_2 &= \frac{h_1}{h_1 + h_2} = \frac{1}{2}, \quad \lambda_2 = 1 - \mu_2 = \frac{1}{2}, \\ d_0 &= \frac{6}{h_0}(f[x_0, x_1] - f'_0) = 20 \left( \frac{2}{3} - 3.0 \right) = -\frac{140}{3}, \\ d_1 &= 6f[x_0, x_1, x_2] = -4, \\ d_2 &= 6f[x_1, x_2, x_3] = -2.7, \\ d_3 &= \frac{6}{h_2}(f'_3 - f[x_2, x_3]) = 6(-4 + 1.1) = -17.4.\end{aligned}$$

因此可得线性方程组

$$\begin{bmatrix} 2 & 1 & & \\ \frac{3}{13} & 2 & \frac{10}{13} & \\ & \frac{1}{2} & 2 & \frac{1}{2} \\ & & 1 & 2 \end{bmatrix} \begin{bmatrix} M_0 \\ M_1 \\ M_2 \\ M_3 \end{bmatrix} = \begin{bmatrix} -\frac{140}{3} \\ 3 \\ -4 \\ -2.7 \\ -17.4 \end{bmatrix},$$

解得 (追赶法)

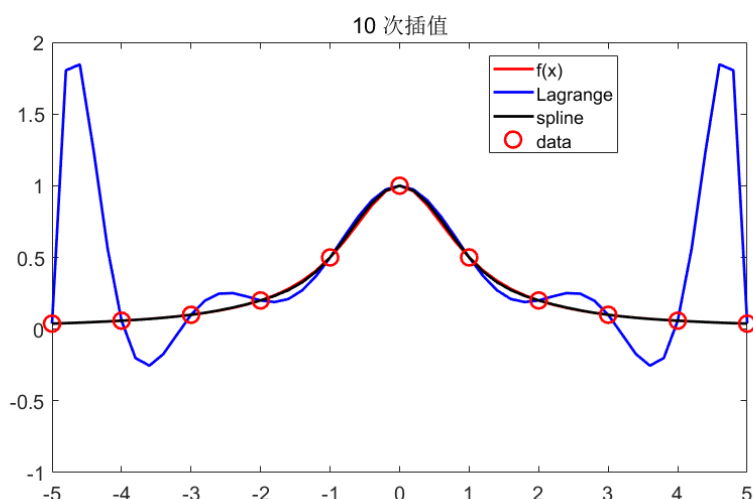
$$\begin{aligned}M_3 &= -\frac{4603}{505} \approx -9.115, \quad M_2 = \frac{419}{505} \approx 0.830, \\ M_1 &= -\frac{40}{101} \approx -0.396, \quad M_0 = -\frac{7130}{303} \approx -23.531.\end{aligned}$$

代入  $s_k(x)$  的表达式 (7.30) 可得

$$S(x) = \begin{cases} 13.293(x-27.7)^3 - 11.766(x-27.7)^2 + 3.000(x-27.7) + 4.1, & x \in [27.7, 28] \\ 0.072(x-28)^3 + 0.198(x-28)^2 - 0.470(x-28) + 4.3, & x \in [28, 29] \\ -1.657(x-29)^3 + 0.415(x-29)^2 + 0.143(x-29) + 4.1, & x \in [29, 30]. \end{cases}$$

□

**例 7.14** 函数  $f(x) = \frac{1}{1+x^2}$ , 插值区间  $[-5, 5]$ , 取 11 个等距节点 (10 等分), 试画出 10 次插值多项式  $L_{10}(x)$  与三次样条插值多项式  $S(x)$  的函数图形. (Interp\_spline\_02.m)

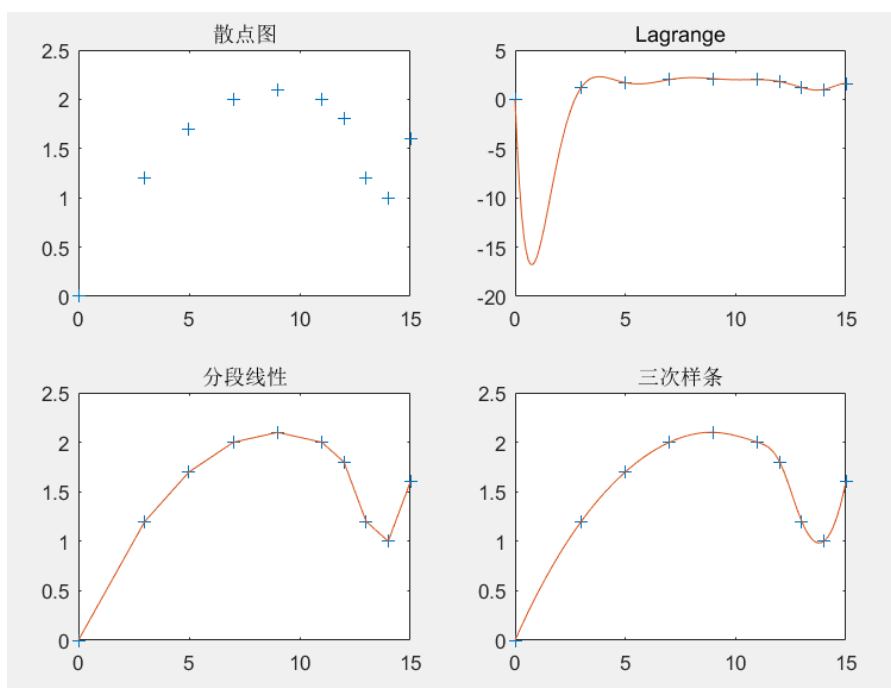


**例 7.15** 机床加工. 待加工零件的外形根据工艺要求由一组数据  $(x, y)$  给出, 用程控铣床加工时每一刀只能沿  $x$  方向和  $y$  方向走非常小的一步, 这就需要从已知数据得到加工所要求的步长很小的  $(x, y)$  坐标. 下表中给出的  $x, y$  数据位于机翼断面的下轮廓线上, 假设需要得到  $x$  坐标每改变 0.1 时的  $y$  坐标. 试完成加工所需数据, 画出曲线. 要求用 Lagrange, 分段线性和三次样条三种插值方法计算.

(Interp\_spline\_03.m)

$x$	0	3	5	7	9	11	12	13	14	15
$y$	0	1.2	1.7	2.0	2.1	2.0	1.8	1.2	1.0	1.6

**解.** 图形为



可以看出, Lagrange 插值的结果根本不能用. 分段线性插值的光滑性较差 (特别是在  $x = 14$  附近弯曲处), 建议选用三次样条插值的结果.  $\square$

### 7.6.4 误差估计

**定理 7.6** 设  $f(x) \in C^4[a, b]$ ,  $S(x)$  为满足第一类或第二类边界条件的三次样条函数, 则

$$\max_{a \leq x \leq b} |f(x) - S(x)| \leq \frac{5}{384} \max_{a \leq x \leq b} |f^{(4)}(x)| h^4,$$

$$\max_{a \leq x \leq b} |f'(x) - S'(x)| \leq \frac{1}{24} \max_{a \leq x \leq b} |f^{(4)}(x)| h^3,$$

$$\max_{a \leq x \leq b} |f''(x) - S''(x)| \leq \frac{3}{8} \max_{a \leq x \leq b} |f^{(4)}(x)| h^2,$$

其中  $h = \max_{0 \leq k \leq n-1} \{h_k\}$ .

(证明可参见相关资料)

该定理说明, 当  $h \rightarrow 0$  时,  $S(x)$  及其一阶导数  $S'(x)$  和二阶导数  $S''(x)$  均收敛到  $f(x)$  及其一阶导数  $f'(x)$  和二阶导数  $f''(x)$ .

## 7.7 课后练习

**练习 7.1** 已知  $f(-1) = -3$ ,  $f(1) = 0$ ,  $f(2) = 4$ , 分别用以下三种方法计算  $f(x)$  的二次插值多项式:

- (1) 用单项式基函数, 即  $\{1, x, x^2\}$ ,
- (2) 用 Lagrange 基函数,
- (3) 用 Newton 基函数.

**练习 7.2** 已知函数  $f(x) = \cos(x)$  在  $[0, \pi/2]$  上的等距节点函数值, 步长为  $h = \pi/10800$ . 如果所给的函数值具有 5 位有效数字, 试计算用线性插值求  $\cos(x)$  近似值时的总误差界.  
(提示: 总误差界要考虑插值误差和所给数据的舍入误差)

**练习 7.3** 设  $x_0, x_1, \dots, x_n$  为互异节点, 求证:

- (1)  $\sum_{j=0}^n x_j^k l_j(x) \equiv x^k \quad (k = 0, 1, \dots, n);$
- (2)  $\sum_{j=0}^n (x_j - x)^k l_j(x) \equiv 0 \quad (k = 0, 1, \dots, n).$

**练习 7.4** 设  $f(x) \in C^2[a, b]$  且  $f(a) = f(b) = 0$ , 求证:

$$\max_{a \leq x \leq b} |f(x)| \leq \frac{1}{8}(b-a)^2 \max_{a \leq x \leq b} |f''(x)|.$$

**练习 7.5** 已知  $f(x) = e^x$  在  $[-4, 4]$  上的等距节点函数值表, 若用二次插值求  $e^x$  的近似值 ( $x \in [-4, 4]$ ), 要使截断误差不超过  $10^{-6}$ , 问步长  $h$  应取多少?

**练习 7.6** 证明  $n$  阶差商的性质:

- (1) 若  $h(x) = cf(x)$ , 则

$$h[x_0, x_1, \dots, x_n] = cf[x_0, x_1, \dots, x_n];$$

- (2) 若  $h(x) = f(x) + g(x)$ , 则

$$h[x_0, x_1, \dots, x_n] = f[x_0, x_1, \dots, x_n] + g[x_0, x_1, \dots, x_n].$$

**练习 7.7** 已知  $f(x) = x^7 + x^4 + 3x + 1$ , 求  $f[2^0, 2^1, \dots, 2^7]$  和  $f[2^0, 2^1, \dots, 2^8]$ .

**练习 7.8** 证明以下结论

- (1)  $\Delta(f_k g_k) = f_k \Delta g_k + g_{k+1} \Delta f_k,$
- (2)  $\sum_{k=0}^{n-1} f_k \Delta g_k = f_n g_n - f_0 g_0 - \sum_{k=0}^{n-1} g_{k+1} \Delta f_k,$
- (3)  $\sum_{k=0}^{n-1} \Delta^2 y_k = \Delta y_n - \Delta y_0.$

**练习 7.9** 若  $f(x) = a_0 + a_1 x + a_2 x^2 + \dots + a_n x^n$  有  $n$  个不同的零点  $x_1, x_2, \dots, x_n$ , 证明

$$\sum_{j=1}^n \frac{x_j^k}{f'(x_j)} = \begin{cases} 0, & 0 \leq k \leq n-2; \\ a_n^{-1}, & k = n-1. \end{cases}$$

**练习 7.10** 求次数不超过 3 的多项式  $p(x)$ , 满足

$$p(x_0) = f(x_0), \quad p(x_1) = f(x_1), \quad p'(x_0) = f'(x_0), \quad p''(x_0) = f''(x_0).$$





练习 7.11 求次数不超过 3 的多项式  $p(x)$ , 满足

$$p(0) = 0, p'(0) = 1, p(1) = 1, p'(1) = 2.$$

练习 7.12 证明两点三次 Hermite 插值余项为

$$R_3(x) = \frac{f^{(4)}(\xi_x)}{4!}(x-x_0)^2(x-x_1)^2,$$

其中  $\xi_x \in (x_0, x_1)$  且与  $x$  相关. 并由此给出分段三次 Hermite 插值的误差限.

练习 7.13 求一个次数不超过 4 的多项式  $p(x)$ , 满足

$$p(0) = p'(0) = 0, p(1) = p'(1) = 1, p(2) = 1.$$

(提示: 这里需要使用非标准的插值计算方法, 可以采用一些技巧, 不要死算.)

练习 7.14 设  $f(x) = \frac{1}{1+x^2}$ , 在  $[-5, 5]$  上取  $n$  等分点做分段线性插值, 计算  $n = 10$  时插值函数  $I_h(x)$  在各区间中点处的值, 并估计误差.

练习 7.15 求  $f(x) = x^2$  在  $[a, b]$  上分段线性插值函数  $I_h(x)$ , 并估计误差.

练习 7.16 求  $f(x) = x^4$  在  $[a, b]$  上分段三次 Hermite 插值函数  $I_h(x)$ , 并估计误差.

练习 7.17 给定数据表如下:

$x_k$	0.25	0.30	0.39	0.45	0.53
$y_k$	0.5000	0.5477	0.6245	0.6708	0.7280

试求三次样条插值函数  $S(x)$ , 满足

$$(1) S'(0.25) = 1.0000, S'(0.53) = 0.6868;$$

$$(2) S''(0.25) = S''(0.53) = 0.$$

练习 7.18 设  $f(x) \in C^2[a, b]$ ,  $S(x)$  是三次样条函数

(1) 证明:

$$\begin{aligned} & \int_a^b [f''(x)]^2 dx - \int_a^b [S''(x)]^2 dx \\ &= \int_a^b [f''(x) - S''(x)]^2 dx + 2 \int_a^b S''(x)[f''(x) - S''(x)] dx; \end{aligned}$$

(2) 设插值节点  $a = x_0 < x_1 < \cdots < x_{n-1} < x_n = b$ , 若  $S(x_k) = f(x_k)$ , 证明:

$$\int_a^b S''(x)[f''(x) - S''(x)] dx = S''(b)[f'(b) - S'(b)] - S''(a)[f'(a) - S'(a)].$$

练习 7.19\* 设  $f(x) \in C^2(a, b)$  且  $f^{(3)}(x)$  存在, 试证明

$$\lim_{x_1 \rightarrow x_0, x_2 \rightarrow x_0} f[x_0, x_1, x_2] = \frac{1}{2} f''(x_0).$$

练习 7.20\* 设  $a = x_0 < x_1 < \cdots < x_{2n} = b$  为  $[a, b]$  上的  $2n$  个等距插值节点, 定义分段等距抛物线插值函数  $I_h(x)$  如下:

$$(1) I_h(x) \in C[a, b];$$

$$(2) I_h(x_k) = f(x_k), \quad k = 0, 1, 2, \dots, 2n;$$

$$(3) I_h(x) \text{ 在每个小区间 } [x_{2k}, x_{2(k+1)}] \text{ 上是二次多项式, } k = 0, 1, 2, \dots, n.$$



类似于定理 7.4 中的结论, 证明: 若  $f(x) \in C^3[a, b]$ , 则

$$\max_{a \leq x \leq b} |f(x) - I_h(x)| \leq \frac{\sqrt{3}}{27} M_3 h^3,$$

其中  $h = \frac{b-a}{2n}$ ,  $M_3 = \max_{a \leq x \leq b} |f^{(3)}(x)|$ .



# 8

## 函数逼近

函数逼近的基本思想就是使用简单易算的函数去近似表达式复杂的函数. 最简单的函数莫过于多项式函数, 因此, 人们首先考虑用多项式函数去做函数逼近. 对于闭区间上任意连续函数  $f(x)$ , 由 Weierstrass 定理可知, 存在一个多项式序列一致收敛到  $f(x)$ . 这就是用多项式函数逼近一般连续函数的理论基础.

### 函数逼近相关参考文献

- [1] 王仁宏, [数值逼近](#) (第二版), 2012 [66]
- [2] 蒋尔雄, 赵风光, 苏仰峰, [数值逼近](#) (第二版), 复旦大学出版社, 2008 [62]
- [3] L. N. Trefethen, [Approximation Theory and Approximation Practice](#), Extended Edition, 2019 [51]

### 8.1 基本概念与预备知识

#### 什么是函数逼近

对于一个给定的复杂函数  $f(x)$ , 在某个表达式较简单的函数类  $\Phi$  中寻找一个函数  $p^*(x)$ , 使其在某种度量下距离  $f(x)$  最近, 即[最佳逼近](#). 这就是[函数逼近](#).

- 🔪 函数  $f(x)$  通常较复杂, 但一般是连续的. 我们这里主要考虑  $[a, b]$  上的连续函数, 即  $f(x) \in C[a, b]$ ;
- 🔪 函数类  $\Phi$  通常由简单函数构成, 比如多项式, 分段多项式, 有理函数, 或者三角函数等;
- 🔪 在不同的度量下,  $f(x)$  的最佳逼近可能不一样;
- 🔪 函数逼近通常采用基函数法.

### 曲线拟合

如果只知道函数在部分节点上的值, 且这些数值带有一定的误差, 需要在函数类  $\Phi$  中寻找一个函数  $p(x)$ , 使其在某种度量下是这些数据的[最佳逼近](#), 这就是[曲线拟合](#), 也称为[数据拟合](#), 可以看作是离散情况下的函数逼近.

## 多项式逼近的理论基础

**定理 8.1 (Weierstrass 逼近定理)** 设  $f \in C[a, b]$ , 则对任意的  $\varepsilon > 0$  存在一个多项式  $p(x)$ , 使得

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

在  $[a, b]$  上一致成立.

**证明.** 该定理有多种证明方法, 其中 Bernstein 方法是一种构造性证明, 不仅证明了多项式的存在性, 而且也给出了构造方法. 详细的证明方法可参见相关数值逼近的文献, 比如 [66].  $\square$

该定理也称为 Weierstrass 第一定理. 该定理表明, 任意一个闭区间上的连续函数都可以用多项式来一致逼近, 即实系数多项式构成的集合在  $C[a, b]$  内是处处稠密的.

## 最佳逼近多项式

**定义 8.1** 设  $\Phi$  为某个函数空间, 给定函数  $f(x) \in C[a, b]$ , 若存在  $g^*(x) \in \Phi$ , 使得

$$\|f(x) - g^*(x)\| = \min_{g(x) \in \Phi} \|f(x) - g(x)\|,$$

则称  $g^*(x)$  为  $f(x)$  在  $\Phi$  中的  $[a, b]$  上的**最佳逼近函数**.

$g^*(x)$  与函数空间  $\Phi$ , 范数  $\|\cdot\|$  和区间  $[a, b]$  有关.

**定义 8.2** 设  $\mathbb{H}_n$  为所有次数不超过  $n$  的多项式组成的函数空间, 给定函数  $f(x) \in C[a, b]$ , 若存在  $p^*(x) \in \mathbb{H}_n$ , 使得

$$\|f(x) - p^*(x)\| = \min_{p(x) \in \mathbb{H}_n} \|f(x) - p(x)\|,$$

则称  $p^*(x)$  为  $f(x)$  在  $[a, b]$  上的  **$n$  次最佳逼近多项式**. 若使用的范数为  $\|\cdot\|_\infty$ , 则称  $p^*(x)$  为  **$n$  次最佳一致逼近多项式**; 若使用的范数为  $\|\cdot\|_2$ , 则称  $p^*(x)$  为  **$n$  次最佳平方逼近多项式**.

## 最小二乘拟合

如果只知道  $f(x)$  在某些节点上的函数值  $f(x_i) = y_i$  ( $i = 0, 1, 2, \dots, m$ ), 在某个函数空间  $\Phi$  中寻找  $g^*(x)$  使得

$$\sum_{i=1}^m |y_i - g^*(x_i)|^2 = \min_{g(x) \in \Phi} \sum_{i=1}^m |y_i - g(x_i)|^2,$$

则称  $g^*(x)$  为  $f(x)$  的**最小二乘拟合**. 若  $\Phi$  取为  $\mathbb{H}_n$ , 则称  $g^*(x)$  为  $f(x)$  的  **$n$  次最小二乘拟合多项式**. 这里一般有  $m > n$ .

求解最佳逼近多项式需要用到一类重要的多项式: 正交多项式.




## 8.2 正交多项式

### 8.2.1 正交函数族与正交多项式

**定义 8.3 (正交函数)** 设  $f(x), g(x) \in C[a, b]$ ,  $\rho(x)$  是  $[a, b]$  上的权函数, 若

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


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

 正交与所使用的内积和权函数有关.

**定义 8.4 (正交函数族)** 设  $\varphi_0(x), \varphi_1(x), \dots, \varphi_n(x), \dots \in C[a, b]$ ,  $\rho(x)$  是  $[a, b]$  上的权函数, 若

$$(\varphi_i, \varphi_j) = \int_a^b \rho(x) \varphi_i(x) \varphi_j(x) dx = \begin{cases} 0, & i \neq j \\ A_i > 0, & i = j \end{cases} \quad i, j = 0, 1, 2, \dots,$$

则称  $\{\varphi_n(x)\}_{n=0}^{\infty}$  是  $[a, b]$  上带权  $\rho(x)$  的正交函数族.

 如果所有的  $A_i$  都等于 1, 则称为标准正交函数族.

#### 例 8.1 三角函数系

$$1, \quad \cos x, \quad \sin x, \quad \cos 2x, \quad \sin 2x, \dots$$

在  $[-\pi, \pi]$  上是带权  $\rho(x) = 1$  的正交函数族.

**证明.** 由三角函数的“积化和差”公式可知 (也可以利用被积函数的奇偶性)

$$(1, 1) = \int_{-\pi}^{\pi} dx = 2\pi;$$

$$(\sin nx, \sin mx) = \int_{-\pi}^{\pi} \sin nx \sin mx dx = \begin{cases} \pi, & m = n \\ 0, & m \neq n \end{cases} \quad m, n = 1, 2, \dots,$$

$$(\cos nx, \cos mx) = \int_{-\pi}^{\pi} \cos nx \cos mx dx = \begin{cases} \pi, & m = n \\ 0, & m \neq n \end{cases} \quad m, n = 1, 2, \dots,$$

$$(\cos nx, \sin mx) = \int_{-\pi}^{\pi} \cos nx \sin mx dx = 0, \quad m, n = 0, 1, 2, \dots$$

□

**定义 8.5 (正交多项式)** 设  $p_n(x)$  是首项系数不为零的  $n$  次多项式,  $n = 0, 1, 2, \dots$ ,  $\rho(x)$  是  $[a, b]$  上的权函数, 若对  $i, j = 0, 1, 2, \dots$  有

$$(p_i, p_j) = \int_a^b \rho(x) p_i(x) p_j(x) dx = \begin{cases} 0, & i \neq j, \\ A_i > 0, & i = j, \end{cases}$$

则称  $\{p_n(x)\}_{n=0}^{\infty}$  为  $[a, b]$  上带权  $\rho(x)$  正交, 并称  $p_n(x)$  为  $n$  次正交多项式.

设  $\{p_n(x)\}_{n=0}^{\infty}$  为  $[a, b]$  上带权  $\rho(x)$  正交多项式, 则显然  $p_0(x), p_1(x), p_2(x), \dots, p_n(x)$  线性无关, 所以它们构成  $\mathbb{R}_n$  的一组正交基.

由于  $p_n(x)$  与  $p_0(x), p_1(x), \dots, p_{n-1}(x)$  都正交, 故  $p_n(x)$  与  $H_{n-1}$  中的任意多项式都正交, 即

$$(p_n(x), p(x)) = \int_a^b \rho(x) p_n(x) p(x) dx = 0, \quad \forall p(x) \in H_{n-1}. \quad (8.1)$$

下面我们给出一个正交多项式的三项递推公式.

**定理 8.2** 设  $\{p_k(x)\}_{k=0}^{\infty}$  为  $[a, b]$  上带权  $\rho(x)$  正交多项式, 且首项系数均为 1, 则有

$$p_{n+1}(x) = (x - \alpha_n)p_n(x) - \beta_n p_{n-1}(x), \quad n = 1, 2, \dots,$$

其中  $p_0(x) = 1, p_1(x) = x - \alpha_0$ ,

$$\alpha_n = \frac{(xp_n, p_n)}{(p_n, p_n)}, \quad n = 0, 1, 2, \dots, \quad \beta_n = \frac{(p_n, p_n)}{(p_{n-1}, p_{n-1})}, \quad n = 1, 2, \dots$$

**证明.** 首先验证  $\alpha_0 = (xp_0, p_0)/(p_0, p_0) = (x, 1)/(1, 1)$ . 因为  $p_1(x)$  与  $p_0(x)$  正交, 所以

$$0 = (p_1, p_0) = (x - \alpha_0, 1) = (x, 1) - \alpha_0(1, 1),$$

即  $\alpha_0 = (x, 1)/(1, 1)$ .

下面考虑  $n \geq 1$  时的情形. 由于  $\{p_0(x), p_1(x), \dots, p_{n+1}(x)\}$  的首项系数都为 1, 且构成  $H_{n+1}$  的一组基, 而  $x p_n(x) \in H_{n+1}$ , 且首项系数为 1, 故  $x p_n(x)$  可以表示为

$$x p_n(x) = \alpha_0 p_0(x) + \alpha_1 p_1(x) + \dots + \alpha_n p_n(x) + p_{n+1}(x).$$

两边用  $p_k(x)$  做内积, 利用  $\{p_k(x)\}_{k=0}^{\infty}$  的正交性可得

$$(x p_n, p_k) = (p_{n+1}, p_k) + \sum_{i=0}^n \alpha_i (p_i, p_k) = \alpha_k (p_k, p_k).$$

又

$$(x p_n, p_k) = \int_a^b \rho(x) x p_n(x) p_k(x) dx = (p_n, x p_k).$$

因此, 当  $0 \leq k \leq n-2$  时, 由正交多项式的性质 (8.1) 可知,

$$\alpha_k (p_k, p_k) = (x p_n, p_k) = (p_n, x p_k) = 0,$$

即  $\alpha_0 = \alpha_1 = \dots = \alpha_{n-2} = 0$ . 于是  $x p_n(x) = p_{n+1}(x) + \alpha_n p_n(x) + \alpha_{n-1} p_{n-1}(x)$ . 为了书写方便, 我们用  $\beta_n$  来表示  $\alpha_{n-1}$ , 即

$$x p_n(x) = p_{n+1}(x) + \alpha_n p_n(x) + \beta_n p_{n-1}(x).$$

两边用  $p_n(x)$  做内积, 可得

$$\alpha_n = \frac{(x p_n, p_n)}{(p_n, p_n)}.$$

同理, 两边用  $p_{n-1}(x)$  做内积, 可得

$$\beta_n = \frac{(x p_n, p_{n-1})}{(p_{n-1}, p_{n-1})}.$$



又

$$(xp_n, p_{n-1}) = (p_n, xp_{n-1}) = (p_n, p_n) + (p_n, xp_{n-1} - p_n) = (p_n, p_n),$$

其中  $(p_n, xp_{n-1} - p_n) = 0$  是因为  $xp_{n-1} - p_n \in H_{n-1}$ . 因此

$$\beta_n = \frac{(p_n, p_n)}{(p_{n-1}, p_{n-1})}.$$

□

所有首项系数为 1 的正交多项式族都满足这个公式, 该公式也给出了正交多项式的一个递推计算方法.

**定理 8.3** 设  $\{p_n(x)\}_{n=0}^{\infty}$  是  $[a, b]$  上带权  $\rho(x)$  的正交多项式, 则当  $n \geq 1$  时,  $p_n(x)$  在  $(a, b)$  内有  $n$  个不同零点.

**证明.** 假设  $p_n(x)$  在  $(a, b)$  内没有零点, 则  $p_n(x)$  在  $(a, b)$  内不变号, 故

$$\left| \int_a^b \rho(x) p_n(x) dx \right| > 0.$$

另一方面, 由 (8.1) 可知

$$0 = (p_n, 1) = \int_a^b \rho(x) p_n(x) dx.$$

矛盾. 因此  $p_n(x)$  在  $(a, b)$  内至少有一个零点.

假设  $p_n(x)$  在  $(a, b)$  内没有奇数重零点, 则  $p_n(x)$  在  $(a, b)$  内不变号, 同样可以导出矛盾. 因此  $p_n(x)$  在  $(a, b)$  内至少有一个奇数重零点.

设  $p_n(x)$  在  $(a, b)$  的所有奇数重零点为  $x_1, x_2, \dots, x_l$  ( $1 \leq l \leq n$ ). 构造多项式  $p_l(x) = (x - x_1)(x - x_2) \cdots (x - x_l)$ . 则  $p_l(x)p_n(x)$  在  $(a, b)$  内只有偶数重零点, 因此在  $(a, b)$  内不变号. 于是

$$|(p_n, p_l)| = \left| \int_a^b \rho(x) p_n(x) p_l(x) dx \right| > 0.$$

如果  $l < n$ , 则由 (8.1) 可知,  $(p_n, p_l) = 0$ , 矛盾. 因此  $l = n$ .

□

### Gram-Schmidt 正交化

事实上, 任意一组线性无关的向量组 (或函数族), 均可通过 Gram-Schmidt 正交化过程产生一组正交的线性无关组.

#### Gram-Schmidt 正交化过程

易知  $\{1, x, x^2, \dots, x^n, \dots\}$  是线性无关的. 相应的 Gram-Schmidt 正交化过程可描述为:

$$\begin{aligned} p_0(x) &= 1, \\ p_k(x) &= x^k - \sum_{j=0}^{k-1} c_{kj} p_j(x), \quad c_{kj} = \frac{(x^k, p_j)}{(p_j, p_j)}, \quad k = 1, 2, 3, \dots \end{aligned}$$

### 8.2.2 Legendre 多项式

设  $[a, b] = [-1, 1]$ , 权函数  $\rho(x) = 1$ , 将线性无关函数组  $\{1, x, x^2, \dots, x^n, \dots\}$  正交化后得到的正交多项式就是 **Legendre 多项式** (勒让德多项式), 记为

$$P_0(x), P_1(x), P_2(x), \dots$$

Legendre 多项式的一般形式为

$$P_0(x) = 1, \quad P_n(x) = \frac{1}{2^n n!} \frac{d^n}{dx^n} (x^2 - 1)^n, \quad x \in [-1, 1], \quad n = 1, 2, \dots \quad (8.2)$$

- $P_n(x)$  的首项系数为  $\frac{(2n)!}{2^n (n!)^2}$ ;
- 若令  $\tilde{P}_n(x) = \frac{2^n (n!)^2}{(2n)!} P_n(x)$ , 则称  $\tilde{P}_n(x)$  为**首项系数为 1 的 Legendre 多项式**.

#### 定理 8.4 (正交性)

$$(P_n, P_m) = \int_a^b P_n(x) P_m(x) dx = \begin{cases} 0, & m \neq n \\ \frac{2}{2n+1}, & m = n \end{cases} \quad (8.3)$$

**证明.** 令  $\varphi(x) = (x^2 - 1)^n$ , 则

$$\varphi^{(k)}(\pm 1) = 0, \quad k = 0, 1, \dots, n-1.$$

设  $Q(x) \in C^n[-1, 1]$ , 则由分部积分法可知

$$\begin{aligned} \int_{-1}^1 P_n(x) Q(x) dx &= \frac{1}{2^n n!} \int_{-1}^1 Q(x) \varphi^{(n)}(x) dx \\ &= \frac{1}{2^n n!} \left( Q(x) \varphi(x) \Big|_{-1}^1 - \int_{-1}^1 Q'(x) \varphi^{(n-1)}(x) dx \right) \\ &= \frac{-1}{2^n n!} \int_{-1}^1 Q'(x) \varphi^{(n-1)}(x) dx \\ &= \dots \\ &= \frac{(-1)^n}{2^n n!} \int_{-1}^1 Q^{(n)}(x) \varphi(x) dx. \end{aligned}$$

取  $Q(x) = P_m(x)$ , 则当  $0 \leq m \leq n-1$  时有  $Q^{(n)}(x) = 0$ , 所以

$$(P_n, P_m) = 0.$$

当  $m = n$  时, 则有

$$Q^{(n)}(x) = \frac{1}{2^n n!} \varphi^{(2n)}(x) = \frac{(2n)!}{2^n n!},$$

所以

$$(P_n, P_m) = \frac{(-1)^n (2n)!}{2^n n! 2^n n!} \int_{-1}^1 (x^2 - 1)^n dx = \frac{(2n)!}{2^{2n} (n!)^2} \int_{-1}^1 (1 - x^2)^n dx.$$

令  $x = \sin \theta$ , 则

$$\int_{-1}^1 (1 - x^2)^n dx = \int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} \cos^{2n} \theta d \sin \theta = \int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} \cos^{2n+1} \theta d \theta$$





$$\begin{aligned}
 &= \cos^{2n} \theta \sin \theta \Big|_{-\frac{\pi}{2}}^{\frac{\pi}{2}} - \int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} 2n \cos^{2n-1} \theta (-\sin \theta) \sin \theta \, d\theta \\
 &= 2n \int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} \cos^{2n-1} \theta \, d\theta - 2n \int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} \cos^{2n+1} \theta \, d\theta.
 \end{aligned}$$

所以

$$\int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} \cos^{2n+1} \theta \, d\theta = \frac{2n}{2n+1} \int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} \cos^{2n-1} \theta \, d\theta.$$

以此类推, 我们可得

$$\int_{-1}^1 (1-x^2)^n \, dx = \int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} \cos^{2n+1} \theta \, d\theta = \frac{2n}{2n+1} \cdot \frac{2(n-1)}{2(n-1)+1} \cdots \frac{2}{3} \int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} \cos \theta \, d\theta = \frac{2^n n! 2^n n! 2}{(2n+1)!}.$$

所以

$$(P_n, P_m) = \frac{(2n)!}{2^{2n}(n!)^2} \int_{-1}^1 (1-x^2)^n \, dx = \frac{2}{2n+1}.$$

□

**定理 8.5 (奇偶性)**  $P_{2k}(x)$  只含偶次幂,  $P_{2k+1}(x)$  只含奇次幂, 故

$$P_n(-x) = (-1)^n P_n(x).$$

**证明.** 由于  $(x^2-1)^n$  至包含偶数次项, 因此由表达式 (8.2) 可知, 当  $n$  是奇数时,  $P_n(x)$  只包含奇数次幂; 当  $n$  是偶数时,  $P_n(x)$  只包含偶数次幂. □

**定理 8.6 (递推公式)**

$$(n+1)P_{n+1}(x) = (2n+1)xP_n(x) - nP_{n-1}(x), \quad n = 1, 2, \dots,$$

其中  $P_0(x) = 1, P_1(x) = x$ .

(证明与定理 8.2 类似)

**定理 8.7 (零点)**  $P_n(x)$  在  $(-1, 1)$  内有  $n$  个不同的零点.

(直接由定理 8.3 可得)

**例 8.2** 给出 5 次 Legendre 多项式  $P_5(x)$  的表达式.

(Approx\_i\_Legendre.m)

**解.** 由递推公式可知

$$P_0(x) = 1$$

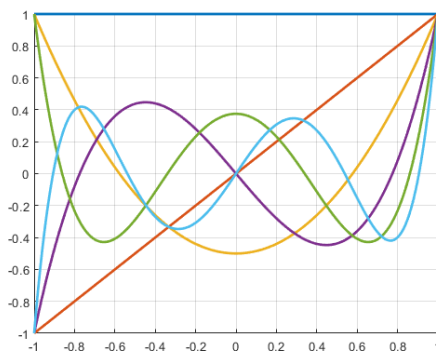
$$P_1(x) = x$$

$$P_2(x) = \frac{1}{2}(3xP_1(x) - P_0(x)) = \frac{1}{2}(3x^2 - 1)$$

$$P_3(x) = \frac{1}{3}(5xP_2(x) - 2P_1(x)) = \frac{1}{2}(5x^3 - 3x)$$

$$P_4(x) = \frac{1}{4}(7xP_3(x) - 3P_2(x)) = \frac{1}{8}(35x^4 - 30x^2 + 3)$$

$$P_5(x) = \frac{1}{5}(9xP_4(x) - 4P_3(x)) = \frac{1}{8}(63x^5 - 70x^3 + 15x)$$



□

### 8.2.3 Chebyshev 多项式

设  $[a, b] = [-1, 1]$ , 权函数  $\rho(x) = \frac{1}{\sqrt{1-x^2}}$ , 将线性无关函数组  $\{1, x, x^2, \dots, x^n, \dots\}$  正交化后得到的正交多项式就是 **Chebyshev 多项式**, 记为  $T_0(x), T_1(x), T_2(x), \dots$

Chebyshev 多项式的一般形式为

$$T_n(x) = \cos(n \arccos x), \quad x \in [-1, 1], \quad n = 1, 2, \dots$$

- 显然,  $|T_n(x)| \leq 1$ .
- 令  $x = \cos \theta, \theta \in [0, \pi]$ , 则  $\theta = \arccos x$ , 所以

$$\begin{aligned} T_n(x) &= \cos(n\theta) = \cos^n \theta - C_n^2 \cos^{n-2} \theta \sin^2 \theta + C_n^4 \cos^{n-4} \theta \sin^4 \theta + \dots \\ &= x^n - C_n^2 x^{n-2} (1-x^2) + C_n^4 x^{n-4} (1-x^2)^2 + \dots \end{aligned}$$

故  $T_n(x)$  是  $n$  次多项式.

#### 定理 8.8 (正交性)

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

(证明留作练习)

#### 定理 8.9 (递推公式)

$$T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x), \quad n = 1, 2, \dots$$

其中  $T_0(x) = 1, T_1(x) = x$ .

**证明.** 令  $x = \cos \theta$ , 则由三角恒等式

$$\cos(n+1)\theta + \cos(n-1)\theta = 2 \cos \theta \cos n\theta$$

可得

$$T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x).$$

□

**定理 8.10 (奇偶性)**  $T_{2n}(x)$  只含偶次幂,  $T_{2n+1}(x)$  只含奇次幂, 故

$$T_n(-x) = (-1)^n T_n(x).$$

(该性质由递推公式直接可得)



**定理 8.11 (零点)**  $T_n(x)$  在  $(-1, 1)$  内有  $n$  个不同的零点:

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

**证明.** 由  $T_n(x) = 0$  可得  $n \arccos x = \left(k - \frac{1}{2}\right)\pi$ , 即

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

□

**定理 8.12 (极值点)**  $T_n(x)$  在  $[-1, 1]$  内有  $n+1$  个极值点 (含两个端点):

$$\tilde{x}_k = \cos \frac{k\pi}{n}, \quad k = 0, 1, 2, \dots, n.$$

**证明.** 直接求导可得

$$T'_n(x) = \frac{n \sin(n \arccos x)}{\sqrt{1-x^2}}.$$

令  $T'_n(x) = 0$  可得极值点

$$\tilde{x}_k = \cos \frac{k\pi}{n}, \quad k = 1, 2, \dots, n-1.$$

再加上两个端点  $\tilde{x}_0 = 1$  和  $\tilde{x}_n = -1$ .

□

**定理 8.13**  $T_n(x)$  的首项系数为  $2^{n-1}$ .

(直接由递推公式可得)

令  $\tilde{T}_n(x) = \frac{1}{2^{n-1}}T_n(x)$ , 则称  $\tilde{T}_n(x)$  为首项系数为 1 的 Chebyshev 多项式.

**例 8.3** 给出 5 次 Chebyshev 多项式  $T_5(x)$  的表达式.

(Approx\_i\_Chebyshev.m)

**解.** 由递推公式可知

$$T_0(x) = 1$$

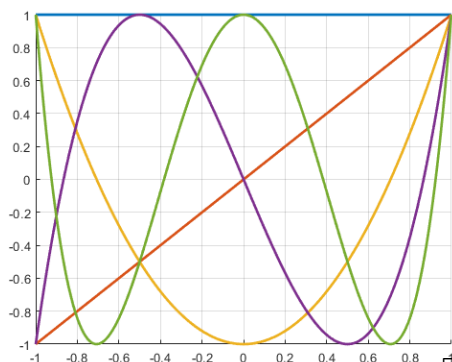
$$T_1(x) = x$$

$$T_2(x) = 2xT_1(x) - T_0(x) = 2x^2 - 1$$

$$T_3(x) = 2xT_2(x) - T_1(x) = 4x^3 - 3x$$

$$T_4(x) = 2xT_3(x) - T_2(x) = 8x^4 - 8x^2 + 1$$

$$T_5(x) = 2xT_4(x) - T_3(x) = 16x^5 - 20x^3 + 5x$$



□

### 8.2.4 Chebyshev 多项式零点插值

我们首先介绍 Chebyshev 多项式的一个重要性质.

**定理 8.14** 设  $\tilde{T}_n(x)$  是首项系数为 1 的 Chebyshev 多项式, 即  $\tilde{T}_n(x) = \frac{1}{2^{n-1}}T_n(x)$ , 则

$$\max_{-1 \leq x \leq 1} |\tilde{T}_n(x)| \leq \max_{-1 \leq x \leq 1} |p(x)|, \quad \forall p(x) \in \tilde{\mathbb{H}}_n,$$

其中  $\tilde{\mathbb{H}}_n$  表示次数不超过  $n$  的所有首项系数为 1 的多项式组成的集合. 且

$$\max_{-1 \leq x \leq 1} |\tilde{T}_n(x)| = \frac{1}{2^{n-1}}.$$

(证明可参见相关文献)

这个性质表明, 在次数不超过  $n$  的所有首项系数为 1 的多项式中,  $\tilde{T}_n(x)$  在  $[-1, 1]$  上与零的偏差是最小的 (在无穷范数意义下).

该性质等价形式为

$$\|\tilde{T}_n(x)\|_\infty = \min_{p(x) \in \tilde{\mathbb{H}}_n} \|p(x)\|_\infty,$$

即  $\tilde{T}_n(x)$  是  $\tilde{\mathbb{H}}_n$  中无穷范数最小的. (注: 这里的  $\|\cdot\|_\infty$  是指  $C[-1, 1]$  上的无穷范数)

利用定理 8.14, 我们可以采用 Chebyshev 多项式的零点作为节点进行多项式插值, 以使得插值的总体误差达到最小化.

设  $L_n(x)$  是  $f(x)$  在  $[-1, 1]$  上的  $n$  次插值多项式, 插值节点为  $x_0, x_1, \dots, x_n$ , 则插值余项为

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

所以总体插值误差为

$$\max_{-1 \leq x \leq 1} |f(x) - L_n(x)| \leq \frac{M_{n+1}}{(n+1)!} \max_{-1 \leq x \leq 1} |\omega_{n+1}(x)|,$$

其中  $M_{n+1} = \max_{-1 \leq x \leq 1} |f^{(n+1)}(x)|$ . 因此, 要使得插值误差最小化, 我们就需要

$$\max_{-1 \leq x \leq 1} |\omega_{n+1}(x)| = \|\omega_{n+1}(x)\|_\infty$$

尽可能地小. 由定理 8.14 可知, 当  $\omega_{n+1}(x) = \tilde{T}_{n+1}(x)$  时,  $\|\omega_{n+1}(x)\|_\infty$  达到最小, 且最小值为  $\frac{1}{2^n}$ . 相应的插值节点即为  $\tilde{T}_{n+1}(x)$  的零点, 也就是  $T_{n+1}(x)$  的零点.

**定理 8.15** 设  $f(x) \in C^{n+1}[-1, 1]$ , 插值节点为  $T_{n+1}(x)$  的零点, 即

$$x_k = \cos \frac{2k+1}{2(n+1)}\pi, \quad k = 0, 1, 2, \dots, n.$$

令  $L_n(x)$  是  $f(x)$  在  $[-1, 1]$  上的  $n$  次插值多项式, 则插值误差满足

$$\|f(x) - L_n(x)\|_\infty \leq \frac{1}{2^n(n+1)!} \|f^{(n+1)}(x)\|_\infty. \quad (8.4)$$

上面的定理表明: 若  $f(x) \in C^{n+1}[-1, 1]$ , 则当  $n \rightarrow \infty$  时,  $L_n(x)$  一致收敛到  $f(x)$ .

如果插值区间是  $[a, b]$ , 则需要做变量替换

$$x(t) = \frac{b-a}{2}t + \frac{b+a}{2}, \quad t \in [-1, 1].$$



令  $t_k$  为 Chebyshev 多项式  $T_{n+1}$  的零点, 则插值节点为

$$x_k = \frac{b-a}{2}t_k + \frac{b+a}{2} = \frac{b-a}{2}\cos\frac{2k+1}{2(n+1)}\pi + \frac{b+a}{2}, \quad k=0, 1, 2, \dots, n, \quad (8.5)$$

总体插值误差

$$\begin{aligned} \max_{a \leq x \leq b} |f(x) - L_n(x)| &\leq \frac{1}{2^n(n+1)!} \max_{-1 \leq t \leq 1} \left| \frac{d^{n+1}f}{dt^{n+1}} \right| \\ &= \frac{1}{2^n(n+1)!} \frac{(b-a)^{n+1}}{2^{n+1}} \max_{-1 \leq t \leq 1} |f^{(n+1)}(x(t))| \\ &= \frac{(b-a)^{n+1}}{2^{2n+1}(n+1)!} \max_{a \leq x \leq b} |f^{(n+1)}(x)|. \end{aligned}$$

上面的总体误差也可以直接将插值点 (8.5) 代入  $\omega_{n+1}(x)$  后获得. 事实上, 由于

$$x - x_k = \frac{b-a}{2}(t - t_k), \quad k=0, 1, 2, \dots, n,$$

故

$$\omega_{n+1}(x) = \prod_{k=0}^n (x - x_k) = \prod_{k=0}^n \frac{b-a}{2}(t - t_k) = \frac{(b-a)^{n+1}}{2^{n+1}} \tilde{T}_{n+1}(t).$$

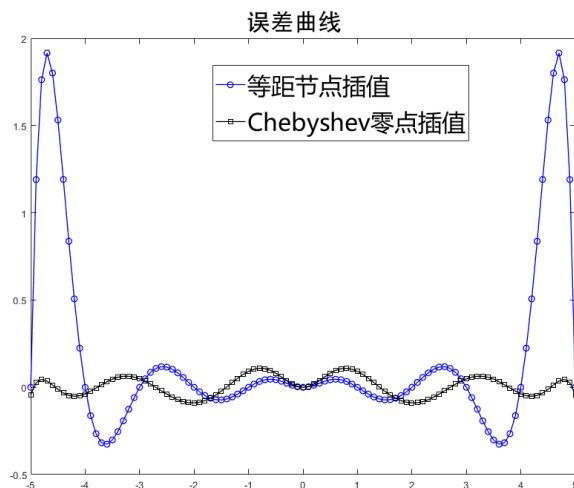
因此

$$\max_{a \leq x \leq b} |\omega_{n+1}(x)| = \frac{(b-a)^{n+1}}{2^{n+1}} \max_{-1 \leq t \leq 1} \tilde{T}_{n+1}(t) = \frac{(b-a)^{n+1}}{2^{n+1}} \cdot \frac{1}{2^n}$$

为了尽可能地减小插值误差, 在可以自由选取插值节点时, 我们尽量使用 Chebyshev 多项式零点.

**例 8.4** 求  $f(x) = e^x$  在  $[0, 1]$  上的四次插值多项式  $L_4(x)$ , 插值节点为  $T_5(x)$  的零点, 并估计总体误差. (板书)

**例 8.5** 设  $f(x) = \frac{1}{1+x^2}$ , 在  $[-5, 5]$  上分别用等距节点和 Chebyshev 多项式零点做 10 次多项式插值, 绘图比较两种插值的数值效果. (Approximate\_Chebyshev\_interp.m)



## 8.2.5 其他正交多项式

### 第二类 Chebyshev 多项式

在区间  $[-1, 1]$  上, 带权  $\rho(x) = \sqrt{1-x^2}$  正交的多项式就称为**第二类 Chebyshev 多项式**, 其一般表达式为

$$U_n(x) = \frac{\sin((n+1)\arccos x)}{\sqrt{1-x^2}}, \quad x \in [-1, 1], \quad n = 0, 1, 2, \dots \quad (8.6)$$

• **正交性:**

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

• **递推公式:**

$$U_{n+1}(x) = 2xU_n(x) - U_{n-1}(x), \quad n = 1, 2, \dots,$$

其中  $U_0(x) = 1, U_1(x) = 2x$ .

### Laguerre 多项式

在区间  $[0, \infty]$  上, 带权  $\rho(x) = e^{-x}$  正交的多项式就称为**Laguerre 多项式**, 其一般表达式为

$$L_n(x) = e^x \frac{d^n}{dx^n} (x^n e^{-x}), \quad x \in [0, \infty], \quad n = 0, 1, 2, \dots$$

• **正交性:**

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

• **递推公式:**

$$L_{n+1}(x) = (2n+1-x)L_n(x) - n^2 L_{n-1}(x), \quad n = 1, 2, \dots,$$

其中  $L_0(x) = 1, L_1(x) = 1-x$ .

### Hermite 多项式

在区间  $[-\infty, \infty]$  上, 带权  $\rho(x) = e^{-x^2}$  正交的多项式就称为**Hermite 多项式**, 其一般表达式为

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

• **正交性:**

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

• **递推公式:**

$$H_{n+1}(x) = 2xH_n(x) - 2nH_{n-1}(x), \quad n = 1, 2, \dots,$$

其中  $H_0(x) = 1, H_1(x) = 2x$ .



## 8.3 最佳平方逼近

### 什么是最佳平方逼近

设  $f(x) \in C[a, b]$ , 在某个给定的简单易算的函数集  $\Phi \subset C[a, b]$  中寻找  $S^*(x)$ , 使得

$$\|f(x) - S^*(x)\|_2^2 = \min_{S(x) \in \Phi} \|f(x) - S(x)\|_2^2.$$

我们称  $S^*(x)$  为  $f(x)$  在  $\Phi$  中的**最佳平方逼近函数**. 这里的范数  $\|\cdot\|_2$  是  $C[a, b]$  上的带权内积导出的范数, 即

$$\|f(x) - S(x)\|_2^2 = \int_a^b \rho(x)(f(x) - S(x))^2 dx.$$

### 8.3.1 怎样求最佳平方逼近

设  $\varphi_0(x), \varphi_1(x), \dots, \varphi_n(x)$  是  $\Phi$  的一组基, 则对任意  $S(x) \in \Phi$ ,  $S(x)$  可表示为

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

因此

$$\|f(x) - S(x)\|_2^2 = \int_a^b \rho(x) \left( f(x) - \sum_{i=0}^n a_i \varphi_i(x) \right)^2 dx \triangleq I(a_0, a_1, \dots, a_n).$$

这是一个关于  $a_0, a_1, \dots, a_n$  的多元函数. 于是, 求最佳逼近函数  $S^*(x)$  就转化为求多元函数  $I(a_0, a_1, \dots, a_n)$  的最小值问题. 易知  $I(a_0, a_1, \dots, a_n)$  是一个正定二次型, 所以  $I(a_0, a_1, \dots, a_n)$  取最小值的充要条件是

$$\frac{\partial I(a_0, a_1, \dots, a_n)}{\partial a_k} = 0, \quad k = 0, 1, 2, \dots, n.$$

通过求导, 上式可化为

$$2 \int_a^b \rho(x) \left( f(x) - \sum_{i=0}^n a_i \varphi_i(x) \right) \varphi_k(x) dx = 0, \quad (8.7)$$

也即

$$\sum_{i=0}^n a_i \int_a^b \rho(x) \varphi_i(x) \varphi_k(x) dx = \int_a^b \rho(x) f(x) \varphi_k(x) dx.$$

写成内积形式即为

$$\sum_{i=0}^n (\varphi_k, \varphi_i) a_i = (\varphi_k, f), \quad k = 0, 1, 2, \dots, n.$$

写成矩阵形式可得

$$\begin{bmatrix} (\varphi_0, \varphi_0) & (\varphi_0, \varphi_1) & \cdots & (\varphi_0, \varphi_n) \\ (\varphi_1, \varphi_0) & (\varphi_1, \varphi_1) & \cdots & (\varphi_1, \varphi_n) \\ \vdots & \vdots & & \vdots \\ (\varphi_n, \varphi_0) & (\varphi_n, \varphi_1) & \cdots & (\varphi_n, \varphi_n) \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} (\varphi_0, f) \\ (\varphi_1, f) \\ \vdots \\ (\varphi_n, f) \end{bmatrix}. \quad (8.8)$$

我们称这个方程为**法方程**, 系数矩阵记为  $G$ , 右端项记为  $d$ . 由于  $\varphi_0(x), \varphi_1(x), \dots, \varphi_n(x)$  线性无关, 由推论 1.3 可知, 法方程 (8.8) 的系数矩阵  $G$  非奇异, 因此法方程存在唯一解.



- 求  $S^*(x) \iff$  解法方程  $Ga = d$ .
- 存在唯一解  $\iff G$  非奇异  $\iff \varphi_0, \varphi_1, \dots, \varphi_n$  线性无关.
- $S^*(x)$  是  $f(x)$  在  $\Phi$  中的最佳逼近  $\iff (f - S^*, \varphi_k) = 0, k = 0, 1, \dots, n \iff (f - S^*, S) = 0, \forall S \in \Phi$ .

**定理 8.16** 设  $a_0^*, a_1^*, \dots, a_n^*$  是法方程 (8.8) 的解, 则  $S^*(x)$  是  $f(x)$  在  $\Phi$  中的最佳平方逼近函数, 其中

$$S^*(x) = a_0^* \varphi_0(x) + a_1^* \varphi_1(x) + \dots + a_n^* \varphi_n(x).$$

**证明.** 对任意  $S(x) \in \Phi$ , 有  $S(x) - S^*(x) \in \Phi$ . 由于  $a_0^*, a_1^*, \dots, a_n^*$  是法方程 (8.8) 的解, 故

$$(f - S^*, \varphi_k) = 0, \quad k = 0, 1, 2, \dots, n.$$

所以有

$$(f - S^*, S(x) - S^*(x)) = 0.$$

于是

$$\begin{aligned} \|f - S\|_2^2 - \|f - S^*\|_2^2 &= \int_a^b [(f - S)^2 - (f - S^*)^2] dx \\ &= \int_a^b [(S - S^*)^2 - 2(f - S^*)(S - S^*)] dx \\ &= \int_a^b (S - S^*)^2 dx \geq 0, \end{aligned}$$

其中等号当且仅当  $S - S^* = 0$ , 即  $S = S^*$  时成立. □



该定理给出了计算最佳平方逼近函数的一个方法.

记  $\delta(x) = f(x) - S^*(x)$  为平方逼近误差. 由 (8.7) 可知  $(f - S^*, \varphi_k) = 0$ , 因此

$$\|\delta(x)\|_2^2 = (f - S^*, f - S^*) = (f - S^*, f) = \|f\|_2^2 - \sum_{i=0}^n a_i^* (\varphi_i, f).$$

### 8.3.2 用正交函数计算最佳平方逼近

通过法方程计算最佳平方逼近时, 需要解一个方程组, 当  $n$  较大时, 会带来一定的困难. 因此我们考虑用正交函数族来计算最佳平方逼近.

设  $\varphi_0(x), \varphi_1(x), \dots, \varphi_n(x)$  是  $\Phi$  的一组正交基, 则法方程的系数矩阵就为一个对角矩阵, 即

$$G = \begin{bmatrix} (\varphi_0, \varphi_0) & & & \\ & (\varphi_1, \varphi_1) & & \\ & & \ddots & \\ & & & (\varphi_n, \varphi_n) \end{bmatrix},$$

故法方程的解为

$$a_k^* = \frac{(\varphi_k, f)}{(\varphi_k, \varphi_k)}, \quad k = 0, 1, 2, \dots, n.$$





于是  $f(x)$  在  $\Phi$  中的最佳平方逼近函数为

$$S^*(x) = \sum_{k=0}^n \frac{(\varphi_k, f)}{(\varphi_k, \varphi_k)} \varphi_k(x), \quad (8.9)$$

误差

$$\|\delta(x)\|_2^2 = \|f(x) - S^*(x)\|_2^2 = (f, f) - (S^*, f) = \|f\|_2^2 - \sum_{k=0}^n \frac{(\varphi_k, f)^2}{(\varphi_k, \varphi_k)}.$$

由于  $\|\delta(x)\|_2^2 \geq 0$ , 所以有

$$\sum_{k=0}^n \frac{(\varphi_k, f)^2}{(\varphi_k, \varphi_k)} \leq \|f\|_2^2, \quad \forall f \in C[a, b].$$

上述不等式称为 **Bessel 不等式**.

### 广义 Fourier 级数

设  $\{\varphi_n(x)\}_{n=0}^\infty$  是正交函数族, 对  $f(x) \in C[a, b]$ , 构造级数

$$a_0^* \varphi_0(x) + a_1^* \varphi_1(x) + \cdots + a_n^* \varphi_n(x) + \cdots \quad (8.10)$$

其中  $a_k^* = \frac{(\varphi_k, f)}{(\varphi_k, \varphi_k)}$ . 这就是关于  $f(x)$  的**广义 Fourier 级数**, 它是 Fourier 级数的推广, 其中  $a_k^*$  称为**广义 Fourier 系数**. 若正交函数族取为

$$1, \cos x, \sin x, \cos 2x, \sin 2x, \dots,$$

则级数 (8.10) 就是 Fourier 级数.

### 8.3.3 最佳平方逼近多项式

设  $\Phi = \mathbb{I}_n$  (次数不超过  $n$  的所有多项式组成的集合), 则  $f(x)$  在  $\Phi$  中的最佳平方逼近就称为  $f(x)$  的  **$n$  次最佳平方逼近多项式**, 记为  $S_n^*(x)$ .

**例 8.6** 设  $[a, b] = [0, 1]$ , 权函数  $\rho(x) \equiv 1$ , 取  $\mathbb{I}_n$  的一组基  $\{1, x, x^2, \dots, x^n\}$ . 求  $f(x) \in C[0, 1]$  的  $n$  次最佳平方逼近多项式.

**解.** 由于  $\varphi_i(x) = x^i$ , 直接计算可得

$$(\varphi_i, \varphi_j) = \int_0^1 x^{i+j} dx = \frac{1}{i+j+1}.$$

所以法方程的系数矩阵为

$$G = \begin{bmatrix} 1 & \frac{1}{2} & \frac{1}{3} & \cdots & \frac{1}{n+1} \\ \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \cdots & \frac{1}{n+2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \frac{1}{n+1} & \frac{1}{n+2} & \frac{1}{n+3} & \cdots & \frac{1}{2n+1} \end{bmatrix} \triangleq H,$$

这就是著名的 **Hilbert 矩阵**. 右端项  $d = [d_0, d_1, \dots, d_n]^T$ , 其中

$$d_k = (\varphi_k, f) = \int_0^1 x^k f(x) dx, \quad k = 0, 1, 2, \dots, n.$$

解法方程可得  $a_k^*$ , 于是  $f(x)$  的  $n$  次最佳平方逼近多项式为

$$S_n^*(x) = \sum_{k=0}^n a_k^* x^k.$$

□

**例 8.7** 设  $f(x) = \sqrt{1+x^2}$ , 求  $f(x)$  在  $[0, 1]$  上的一次最佳平方逼近多项式.

(板书)

▮ Hilbert 矩阵对称正定, 但高度病态, 当维数较大时, 会给数值求解带来很大的困难. 因此, 我们很少用这种方法来求最佳平方逼近多项式.

### 8.3.4 用正交多项式计算最佳平方逼近多项式

**定理 8.17** 设  $f(x) \in C[a, b]$ ,  $\{\varphi_n(x)\}_{n=0}^\infty$  是正交多项式族,  $S_n^*(x)$  是由 (8.9) 给出的  $n$  次最佳平方逼近多项式, 则

$$\lim_{n \rightarrow \infty} \|f(x) - S_n^*(x)\|_2 = 0,$$

即  $S_n^*(x)$  一致收敛到  $f(x)$ .

(证明可参见相关资料)

下面考虑用 Legendre 多项式来计算最佳平方逼近多项式.

**定理 8.18** 设  $f(x) \in C[-1, 1]$ , 权函数  $\rho(x) \equiv 1$ , 则  $f(x)$  在  $[-1, 1]$  上的  $n$  次最佳平方逼近多项式为

$$S_n^*(x) = a_0^* P_0(x) + a_1^* P_1(x) + \dots + a_n^* P_n(x),$$

其中  $P_k(x)$  为  $k$  次 Legendre 多项式,

$$a_k^* = \frac{(P_k, f)}{(P_k, P_k)} = \frac{2k+1}{2} \int_{-1}^1 P_k(x) f(x) dx.$$

误差

$$\|\delta_n(x)\|_2^2 = \|f(x) - S_n^*(x)\|_2^2 = \|f(x)\|_2^2 - \sum_{k=0}^n a_k^* (P_k, f) = \int_{-1}^1 f^2(x) dx - \sum_{k=0}^n \frac{2(a_k^*)^2}{2k+1}.$$

▮ 该定理给出了求解最佳平方逼近多项式的计算公式和误差表达式, 通常都使用这种方法计算最佳平方逼近多项式.



**例 8.8** 设  $f(x) = e^x$ , 求  $f(x)$  在  $[-1, 1]$  上的三次最佳平方逼近多项式.

(板书)

**定理 8.19** 设  $f(x) \in C^2[-1, 1]$ , 则对  $\forall x \in [-1, 1]$  和  $\forall \varepsilon > 0$ , 当  $n$  充分大时, 有

$$|f(x) - S_n^*(x)| \leq \frac{\varepsilon}{\sqrt{n}}.$$

(证明可参见相关资料)

**定理 8.20** 在所有首项系数为 1 的  $n$  次多项式中,  $\tilde{P}_n(x)$  在  $[-1, 1]$  上与零的平方逼近误差最小, 即

$$\|\tilde{P}_n(x)\|_2 = \min_{p(x) \in \tilde{\mathbb{H}}_n} \|p(x)\|_2 = \min_{p(x) \in \tilde{\mathbb{H}}_n} \left( \int_{-1}^1 p^2(x) dx \right)^{\frac{1}{2}},$$

其中  $\tilde{P}_n(x)$  是首项系数为 1 的 Legendre 多项式,  $\tilde{\mathbb{H}}_n$  表示所有首项系数为 1 的  $n$  次多项式组成的集合.

(板书)

这是 Legendre 多项式的一个重要性质, 与  $\tilde{T}_n(x)$  的“无穷范数最小”性质 (见定理 8.14) 相类似.

#### 一般区间上的最佳平方逼近多项式的计算方法

计算过程如下:

- (1) 做变换替换  $x(t) = \frac{b-a}{2}t + \frac{b+a}{2}$ , 将  $f(x)$  转化为  $g(t) = f(x(t))$ ,  $t \in [-1, 1]$ ;
- (2) 通过 Legendre 多项式计算出  $g(t)$  在  $[-1, 1]$  上的最佳平方逼近多项式  $S^*(t)$ ;
- (3) 将  $t = \frac{2x-b-a}{b-a}$  代入  $S^*(t)$ , 给出  $f(x)$  在  $[a, b]$  上的最佳平方逼近多项式

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

## 8.4 最佳一致逼近

### 什么是最佳一致逼近

设  $f(x) \in C[a, b]$ , 在某个给定的函数集  $\Phi \subset C[a, b]$  中寻找  $S^*(x)$ , 使得

$$\|f(x) - S^*(x)\|_\infty = \min_{S(x) \in \Phi} \|f(x) - S(x)\|_\infty.$$

我们称  $S^*(x)$  为  $[a, b]$  上  $f(x)$  在  $\Phi$  中的**最佳一致逼近函数**. 若  $\Phi = \mathbb{I}_n$ , 则称  $S^*(x)$  为  $f(x)$  在  $[a, b]$  上的  $n$  次**最佳一致逼近多项式**. 这里的范数  $\|\cdot\|_\infty$  是  $C[a, b]$  上的无穷范数, 即

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

### 8.4.1 最佳一致逼近多项式的存在唯一性

**定理 8.21 (Chebyshev 定理)** 设  $f(x) \in C[a, b]$ , 则  $f(x)$  在  $[a, b]$  上存在唯一的  $n$  次最佳一致逼近多项式, 且  $p_n^*(x)$  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  $[a, b]$  内至少存在  $n+2$  个交错偏差点  $x_0, x_1, x_2, \dots, x_{n+1}$ , 即

$$f(x_i) - p_n^*(x_i) = (-1)^i \max_{a \leq x \leq b} |p_n^*(x) - f(x)|, \quad i = 0, 1, 2, \dots, n+1.$$

(证明可参见相关资料)

该定理揭示了最佳一致逼近多项式的特征, 并描述了最佳一致逼近多项式误差曲线的性态, 是构造最佳一致逼近多项式的主要理论依据.

### 零次与一次最佳一致逼近多项式

作为例子, 我们考虑  $n=0$  和  $n=1$  时的情形.

**例 8.9** 设  $f(x) \in C[a, b]$ , 则  $f(x)$  的零次最佳一致逼近多项式为

$$p_0^*(x) = \frac{1}{2} \left( \min_{a \leq x \leq b} f(x) + \max_{a \leq x \leq b} f(x) \right).$$

**例 8.10** 设  $f(x) \in C^2[a, b]$  且  $f''(x) > 0, x \in [a, b]$ , 求  $f(x)$  的一次最佳一致逼近多项式.

**解.** 设  $f(x)$  的一次最佳一致逼近多项式为

$$p_1^*(x) = \alpha x + \beta, \quad \alpha, \beta \in \mathbb{R}.$$

由 Chebyshev 定理 8.21 可知,  $f(x) - p_1^*(x)$  在  $[a, b]$  内至少存在 3 个偏差点, 不妨设为  $x_0 < x_1 < x_2$ . 则  $x_1$  必定在  $(a, b)$  内, 且  $x_1$  为  $p_1^*(x) - f(x)$  的驻点, 即

$$f'(x_1) - (p_1^*)'(x_1) = 0.$$

所以  $f'(x_1) = \alpha$ . 又  $f''(x) > 0$ , 即  $f'(x)$  严格单调递增. 因此

$$(f(x) - p_1^*(x))' = f'(x) - \alpha$$



在  $[a, b]$  内不能再有其它零点. 因此  $f(x) - p_1^*(x)$  至多只有 3 个偏差点, 且其它两个偏差点为端点. 所以  $f(x) - p_1^*(x)$  只有 3 个偏差点  $x_0, x_1, x_2$ , 且  $x_0 = a, x_2 = b$ . 从而

$$f(a) - p_1^*(a) = -(f(x_1) - p_1^*(x_1)) = f(b) - p_1^*(b).$$

代入后解得

$$\alpha = \frac{f(b) - f(a)}{b - a}, \quad \beta = \frac{f(a) + f(x_1)}{2} - \frac{(a + x_1)(f(b) - f(a))}{2(b - a)},$$

其中  $x_1$  由下面的等式确定:

$$f'(x_1) = \frac{f(b) - f(a)}{b - a}.$$

□

当  $n \geq 2$  时, 求最佳一致逼近多项式是非常复杂和困难的. 但也存在某些特殊情形, 最佳一致逼近多项式比较容易计算.

### 8.4.2 $n$ 次多项式的 $n - 1$ 次最佳一致逼近多项式

如果  $f(x)$  是一个  $n$  次多项式, 则我们可以利用首项系数为 1 的 Chebyshev 多项式与零偏差最小的性质 (见定理 8.14), 构造其  $n - 1$  次的最佳一致逼近多项式.

**定理 8.22** 设  $f(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0$ , 其中  $a_n \neq 0$ , 则

$$p_{n-1}^*(x) = f(x) - a_n \tilde{T}_n(x)$$

是  $f(x)$  在  $[-1, 1]$  上的  $n - 1$  次最佳一致逼近多项式.

(证明留作练习)

**例 8.11** 设  $f(x) = 2x^3 + x^2 + 2x - 1$ , 求  $f(x)$  在  $[-1, 1]$  上的 2 次最佳一致逼近多项式.

**证明.** 由定理 8.22 可知,  $f(x)$  在  $[-1, 1]$  上的 2 次最佳一致逼近多项式为

$$p_2^*(x) = f(x) - 2\tilde{T}_3(x) = x^2 + \frac{7}{2}x - 1.$$

□

**思考**

思考: 如何计算首项系数非零的  $n$  次多项式  $f(x)$  在  $[a, b]$  上的  $n - 1$  次最佳一致逼近多项式?

### 8.4.3 Chebyshev 级数与近似最佳一致逼近

对于任意一个  $f(x) \in C[a, b]$ , 计算其最佳一致逼近多项式是非常困难的, 目前还没有一个可以经过有限步计算出  $f(x)$  的  $n$  次最佳一致逼近多项式的方法.

关于  $n$  次最佳一致逼近多项式的构造, 可以采用 Remes 算法, 但这是一种迭代近似算法, 而且比较复杂, 运算量也较大, 详细介绍可以参考相关文献. 在实际应用中, 人们往往更倾向于寻找近似的最佳一致逼近多项式.

Chebyshev 展开就是一种很有效的计算近似最佳一致逼近多项式的方法. 设  $f \in C[-1, 1]$ , 权函数  $\rho(x) = \frac{1}{\sqrt{1-x^2}}$ , 在  $f(x)$  的广义 Fourier 级数 (8.10) 中取  $\varphi_k = T_k(x)$ , 则可得

$$\frac{a_0^*}{2} + \sum_{k=1}^{\infty} a_k^* T_k(x),$$

这就是  $f(x)$  在  $[-1, 1]$  上的 **Chebyshev 级数**, 其中

$$a_k^* = \frac{2}{\pi} \int_{-1}^1 \frac{T_k(x)f(x)}{\sqrt{1-x^2}} dx = \frac{2}{\pi} \int_0^\pi f(\cos \theta) \cos k\theta d\theta, \quad k = 0, 1, 2, \dots$$

注意: 由于  $(T_0, T_0) = \pi$ , 因此这里的  $a_0^* = 2 \frac{(T_0, f)}{(T_0, T_0)}$ .

**定理 8.23** 若  $f''(x)$  在  $[-1, 1]$  上分段连续, 则 Chebyshev 级数一致收敛, 即

$$f(x) = \frac{a_0^*}{2} + \sum_{k=1}^{\infty} a_k^* T_k(x).$$

记部分和

$$C_n^*(x) = \frac{a_0^*}{2} + \sum_{k=1}^n a_k^* T_k(x),$$

则余项为 (级数一致收敛, 余项收敛到 0)

$$f(x) - C_n^*(x) = \sum_{k=n+1}^{\infty} a_k^* T_k(x) \approx a_{n+1}^* T_{n+1}(x).$$

由于  $\|\tilde{T}_{n+1}(x)\|_\infty$  在  $\tilde{\mathbb{T}}_{n+1}$  中最小, 故  $C_n^*(x)$  可看作是  $f(x)$  在  $[-1, 1]$  上的近似最佳一致逼近多项式.

**例 8.12** 求  $f(x) = e^x$  在  $[-1, 1]$  上的 Chebyshev 级数部分和  $C_3^*(x)$ .

(板书)



## 8.5 最小二乘曲线拟合

### 8.5.1 曲线拟合介绍

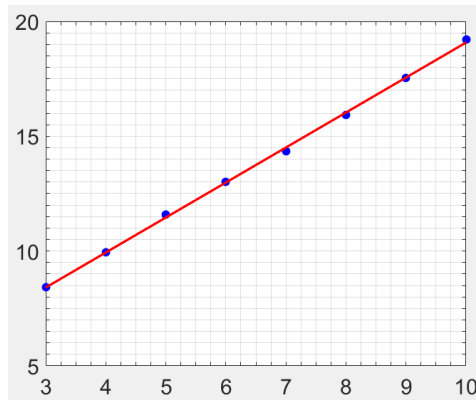
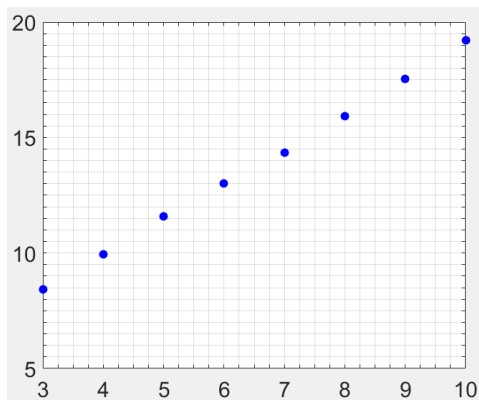
曲线拟合 (curve fitting) 是指选择适当的曲线来拟合通过观测或实验所获得的数据. 科学和工程中遇到的很多问题, 往往只能通过诸如采样、实验等方法获得若干离散的数据. 根据这些数据, 如果能够找到一个连续的函数 (即曲线) 或者更加密集的离散方程, 使得实验数据与方程的曲线能够在最大程度上近似吻合, 就可以根据曲线方程对数据进行理论分析和数值预测, 对某些不具备测量条件的位置的结果进行估算.

我们首先看一个简单的线性数据拟合问题.

**例 8.13** 回想一下中学物理课的“速度与加速度”实验: 假设某物体正在做加速运动, 加速度未知, 实验人员从时间  $t_0 = 3$  秒时刻开始, 每隔 1 秒时间对这个物体进行测速, 得到一组速度和时间的离散数据 (见下表). 请根据实验数据推算该物体的加速度.

时间 $t$ (秒)	3	4	5	6	7	8	9	10
速度 $v$ (米/秒)	8.41	9.94	11.58	13.02	14.33	15.92	17.54	19.22

**解. 实验法:** 在坐标纸中画出这些点, 如下图 (左图) 所示.



可以看出, 测量结果呈现典型的线性特征. 沿着该线性特征画一条直线, 使尽量多的测量点能够位于直线上, 或者与直线的偏差尽量小, 见上图 (右图). 这条直线就是我们根据测量结果拟合的速度与时间的函数关系. 最后测量出直线的斜率  $k$ , 它就被测物体的加速度. 经过测量, 我们实验测到的物体加速度值约为  $1.52$  米/秒<sup>2</sup>.

**数学方法:** 设加速度为  $a$  (米/秒<sup>2</sup>), 则速度  $v$  与时间  $t$  之间的关系式为

$$v = v_0 + a(t - t_0).$$

其中  $v_0 = 8.41$ ,  $t_0 = 3$ . 将实验数据  $(t_i, v_i)$  依次代入可得

$$\begin{cases} 9.94 = 8.41 + a \\ 11.58 = 8.41 + 2a \\ 13.02 = 8.41 + 3a \\ 14.33 = 8.41 + 4a \\ 15.92 = 8.41 + 5a \\ 17.54 = 8.41 + 6a \\ 19.22 = 8.41 + 7a \end{cases}$$

显然, 这个方程组是无解的.

事实上, 由于实验存在误差, 上面的每个方程并不需要严格成立, 因此我们只要求偏差尽可能地小即可. 也就是说, 使得偏差平方和尽可能地小, 即转化为下面的最小化问题

$$\min_{a \in \mathbb{R}} \sum_{i=1}^7 |v_i - v_0 - a(t_i - t_0)|^2.$$

这是一个最小二乘问题, 这就是曲线拟合 (数据拟合) 的最小二乘法.  $\square$

### 8.5.2 最小二乘与法方程

给定数据

$x$	$x_0$	$x_1$	$x_2$	$\cdots$	$x_m$
$y$	$y_0$	$y_1$	$y_2$	$\cdots$	$y_m$

在函数族  $\Phi \triangleq \text{span}\{\varphi_0(x), \varphi_1(x), \dots, \varphi_n(x)\}$  中寻找函数  $S^*(x)$ , 使得它与这组数据的偏差平方和最小, 即

$$\sum_{i=0}^m |S^*(x_i) - y_i|^2 = \min_{S(x) \in \Phi} \sum_{i=0}^m |S(x_i) - y_i|^2.$$

这就是**曲线拟合的最小二乘法**. 这里的  $n$  通常远远小于  $m$ , 即  $n \ll m$ .

在进行数据拟合时, 也可以使用其他标准 (拟合方法), 如**极小化偏差的最大值**, 即

$$\max_{0 \leq i \leq m} |S^*(x_i) - y_i| = \min_{S(x) \in \Phi} \max_{0 \leq i \leq m} |S(x_i) - y_i|.$$

但上述极小化问题求解很复杂.

另一种拟合方法是**极小化偏差之和**, 即

$$\sum_{i=0}^m |S^*(x_i) - y_i| = \min_{S(x) \in \Phi} \sum_{i=0}^m |S(x_i) - y_i|.$$

但由于目标函数不可导, 求解也很困难.





## 带权最小二乘

在某些应用中, 在各个点上的权重可能不一样, 因此我们需要带权的数据拟合问题, 即

$$\min_{S(x) \in \Phi} \sum_{i=0}^m \omega_i |S(x_i) - y_i|^2, \quad (8.11)$$

其中  $\omega_i$  都是正实数, 代表在各个点处的权重.

离散数据拟合的最小二乘问题实质上可以看作是最佳平方逼近问题的离散形式, 因此, 可以将求连续函数的最佳平方逼近函数的方法直接用于求解该问题.

下面考虑问题 (8.11) 的求解. 对任意  $S(x) \in \Phi = \text{span}\{\varphi_0(x), \varphi_1(x), \dots, \varphi_n(x)\}$ , 可设

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

则原问题就转化为求下面的多元函数的最小值点:

$$I(a_0, a_1, \dots, a_n) \triangleq \sum_{i=0}^m \omega_i |S(x_i) - y_i|^2 = \sum_{i=0}^m \omega_i \left[ \sum_{j=0}^n a_j \varphi_j(x_i) - y_i \right]^2.$$

由于  $I(a_0, a_1, \dots, a_n)$  是正定的, 因此其最小值点就是其驻点. 令偏导数为零, 可得

$$0 = \frac{\partial I(a_0, a_1, \dots, a_n)}{\partial a_k} = 2 \sum_{i=0}^m \omega_i \varphi_k(x_i) \left[ \sum_{j=0}^n a_j \varphi_j(x_i) - y_i \right], \quad k = 0, 1, 2, \dots, n.$$

整理后可写为

$$\sum_{j=0}^n \left[ \sum_{i=0}^m \omega_i \varphi_k(x_i) \varphi_j(x_i) - y_i \right] a_j = \sum_{i=0}^m \omega_i y_i \varphi_k(x_i), \quad k = 0, 1, 2, \dots, n.$$

引入记号

$$(\varphi_j, \varphi_k) \triangleq \sum_{i=0}^m \omega_i \varphi_j(x_i) \varphi_k(x_i), \quad (y, \varphi_k) \triangleq \sum_{i=0}^m \omega_i y_i \varphi_k(x_i), \quad (8.12)$$

则上面的方程可简写为

$$\sum_{j=0}^n (\varphi_j, \varphi_k) a_j = (y, \varphi_k), \quad k = 0, 1, 2, \dots, n.$$

写成矩阵形式即可得法方程:

$$Ga = d, \quad (8.13)$$

其中

$$G = \begin{bmatrix} (\varphi_0, \varphi_0) & (\varphi_0, \varphi_1) & \cdots & (\varphi_0, \varphi_n) \\ (\varphi_1, \varphi_0) & (\varphi_1, \varphi_1) & \cdots & (\varphi_1, \varphi_n) \\ \vdots & \vdots & & \vdots \\ (\varphi_n, \varphi_0) & (\varphi_n, \varphi_1) & \cdots & (\varphi_n, \varphi_n) \end{bmatrix}, \quad a = \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{bmatrix}, \quad d = \begin{bmatrix} (y, \varphi_0) \\ (y, \varphi_1) \\ \vdots \\ (y, \varphi_n) \end{bmatrix}.$$

将法方程的解记为  $a_0^*, a_1^*, a_2^*, \dots, a_n^*$ , 则最佳平方逼近函数为

$$S^*(x) = a_0^* \varphi_0(x) + a_1^* \varphi_1(x) + \dots + a_n^* \varphi_n(x).$$

为了确保法方程的解存在唯一, 我们要求系数矩阵  $G$  非奇异.

需要指出的是, 我们在 (8.12) 中引入的记号  $(\varphi_j, \varphi_k)$  并不构成  $C[a, b]$  或  $\Phi$  中的内积, 因此仅仅凭借  $\varphi_0(x), \varphi_1(x), \dots, \varphi_n(x)$  的线性无关并不能推出  $G$  是非奇异的.

**定理 8.24** 设  $\varphi_0(x), \varphi_1(x), \dots, \varphi_n(x) \in C[a, b]$  线性无关. 如果  $\varphi_0(x), \varphi_1(x), \dots, \varphi_n(x)$  的任意 (非零) 线性组合在点集  $\{x_0, x_1, \dots, x_m\}$  上至多只有  $n$  个不同的零点, 则  $G$  非奇异, 此时法方程 (8.13) 存在唯一解.

上述定理中的条件称为 **Haar 条件**, 显然, 如果取  $\varphi_i(x) = x^i$ , 则 Haar 条件成立.

**例 8.14** 已知数据表如下, 求  $f(x)$  的最小二乘拟合函数  $S^*(x)$ .

$x_i$	0.24	0.65	0.95	1.24	1.73	2.01	2.23	2.52	2.77	2.99
$y_i$	0.23	-0.26	-1.10	-0.45	0.27	0.10	-0.29	0.24	0.56	1.00

**证明.** 分两步走: 先确定一组基, 然后通过解法方程求出表出系数.

在坐标平面上描出数据点, 根据点的分布情况, 选取基函数

$$\varphi_0(x) = \ln x, \quad \varphi_1(x) = \cos x, \quad \varphi_2(x) = e^x.$$

由直接计算可得法方程

$$\begin{bmatrix} 6.7941 & -5.3475 & 63.259 \\ -5.3475 & 5.1084 & -49.009 \\ 63.259 & -49.009 & 1002.5 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} 1.6131 \\ -2.3821 \\ 26.773 \end{bmatrix}.$$

解得

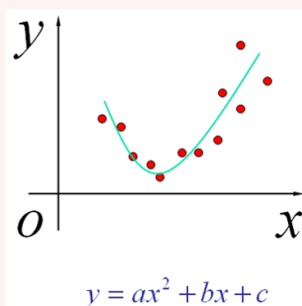
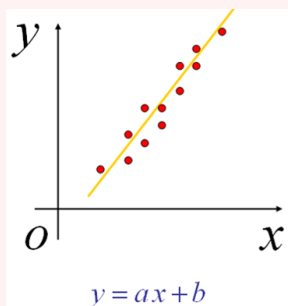
$$a_0 = -1.0410, \quad a_1 = -1.2613, \quad a_2 = 0.03073.$$

所以最小二乘拟合函数为

$$S^*(x) = -1.0410 \ln x - 1.2613 \cos x + 0.03073e^x.$$

□

对于曲线拟合问题, 如何选择数学模型很重要 (即函数空间  $\Phi$ , 也即基函数  $\varphi_0(x), \varphi_1(x), \dots, \varphi_n(x)$ ), 通常需要根据物理意义或所给数据的分布情况来确定.



### 8.5.3 多项式拟合

在数据拟合时, 如果取  $\varphi_0(x) = 1, \varphi_1(x) = x, \dots, \varphi_n(x) = x^n$ , 即  $\Phi = \mathbb{I}_n$ , 则相应的法方程为

$$\begin{bmatrix} \sum_{i=0}^m \omega_i & \sum_{i=0}^m \omega_i x_i & \cdots & \sum_{i=0}^m \omega_i x_i^n \\ \sum_{i=0}^m \omega_i x_i & \sum_{i=0}^m \omega_i x_i^2 & \cdots & \sum_{i=0}^m \omega_i x_i^{n+1} \\ \vdots & \vdots & & \vdots \\ \sum_{i=0}^m \omega_i x_i^n & \sum_{i=0}^m \omega_i x_i^{n+1} & \cdots & \sum_{i=0}^m \omega_i x_i^{2n} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} \sum_{i=0}^m \omega_i y_i \\ \sum_{i=0}^m \omega_i x_i y_i \\ \cdots \\ \sum_{i=0}^m \omega_i x_i^n y_i \end{bmatrix}.$$

设解为  $[a_0^*, a_1^*, \dots, a_n^*]^\top$ , 则

$$S^*(x) = a_0^* + a_1^* x + \cdots + a_n^* x^n$$

即为  $f(x)$  的  $n$  次最小二乘拟合多项式.

**例 8.15** 已知数据表如下, 求 2 次最小二乘拟合多项式.

$x_i$	0	0.25	0.50	0.75	1.00
$y_i$	1.0000	1.2840	1.6487	2.1170	2.7183

**证明.** 设 2 次最小二乘拟合多项式为  $p_2(x) = a_0 + a_1 x + a_2 x^2$ . 直接计算得法方程

$$\begin{bmatrix} 5 & 2.5 & 1.875 \\ 2.5 & 1.875 & 1.5625 \\ 1.875 & 1.5625 & 1.3825 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} 8.7680 \\ 5.4514 \\ 4.4015 \end{bmatrix}.$$


解得

$$a_0 = 1.0052, \quad a_1 = 0.8641, \quad a_2 = 0.8437.$$

所以此组数据的 2 次最小二乘拟合多项式为

$$S_2^*(x) = 1.0052 + 0.8641x + 0.8437x^2.$$

□

 如果没有给出权系数  $\omega_i$ , 则默认所有权系数都是 1.

由于当  $n$  较大时, 直接求解法方程的计算成本较高, 而且系数矩阵是病态的 (与 Hilbert 矩阵类似), 因此该方法不适合计算高次最小二乘拟合多项式, 此时可以借助正交多项式来进行求解.

### 8.5.4 用正交多项式做最小二乘拟合

带权正交 (离散情形)

**定义 8.6** 给定点集  $\{x_i\}_{i=0}^m$  以及各点处的权系数  $\{\omega_i\}_{i=0}^m$ , 如果函数族  $\{\varphi_k(x)\}_{k=0}^n$  满足

$$(\varphi_k, \varphi_j) \triangleq \sum_{i=0}^m \omega_i \varphi_k(x_i) \varphi_j(x_i) = \begin{cases} 0, & k \neq j, \\ A_k \neq 0, & k = j, \end{cases} \quad (8.14)$$

则称  $\{\varphi_k(x)\}_{k=0}^n$  关于点集  $\{x_i\}_{i=0}^m$  带权  $\{\omega_i\}_{i=0}^m$  正交.

如果  $\varphi_k(x)$  为首项系数不为零的  $k$  次多项式, 则  $\{\varphi_k(x)\}_{k=0}^n$  就是关于点集  $\{x_i\}_{i=0}^m$  的带权正交多项式.

**思考** 由 (8.14) 所定义的关系式是否构成内积?

**定理 8.25** 设多项式  $p_0(x), p_1(x), \dots, p_n(x)$  是关于点集  $x_0, x_1, \dots, x_m$  的带权  $\omega_0, \omega_1, \dots, \omega_m$  正交, 则  $f(x)$  的  $n$  次最小二乘拟合多项式为

$$S_n^*(x) = a_0^* p_0(x) + a_1^* p_1(x) + a_2^* p_2(x) + \dots + a_n^* p_n(x),$$

其中

$$a_k^* = \frac{(p_k, f)}{(p_k, p_k)}, \quad k = 0, 1, 2, \dots, n.$$

上述结论中的  $(\cdot, \cdot)$  是由 (8.14) 所定义的.

法方程问题不存在了, 但另外还有一个问题需要解决, 即带权正交多项式如何计算?

### 带权正交 (离散情形) 多项式的递推计算方法

与正常的正交多项式类似, 我们也可以构造出在离散带权正交情形下的三项递推公式:

$$\begin{aligned} p_0(x) &= 1, \quad p_1(x) = x - \alpha_0, \\ p_{k+1}(x) &= (x - \alpha_k) p_k(x) - \beta_k p_{k-1}(x), \quad k = 1, 2, \dots, \end{aligned} \quad (8.16)$$

其中

$$\alpha_k = \frac{(x p_k, p_k)}{(p_k, p_k)}, \quad k = 0, 1, \dots, \quad \beta_k = \frac{(p_k, p_k)}{(p_{k-1}, p_{k-1})}, \quad k = 1, 2, \dots$$

可以证明, 由上述递推方法构造出来的  $\{p_k(x)\}_{k=0}^n$  关于点集  $\{x_i\}_{i=0}^m$  是带权  $\{\omega_i\}_{i=0}^m$  正交的.

在实际计算时,  $n$  可以事先给定, 或在计算过程中根据计算精度来确定.

在计算过程中可以将构造正交多项式族、解法方程、形成拟合多项式穿插进行.

该方法非常适合编程实现, 是目前多项式最小二乘拟合最好的计算方法, 有通用程序.

**例 8.16** 已知数据表如下, 求 2 次最小二乘拟合多项式 (计算过程中小数点后保留 2 位).

$x_i$	0	0.5	0.6	0.7	0.8	0.9	1.00
$y_i$	1.00	1.75	1.96	2.19	2.44	2.71	3.00

(Approximate\_datafit\_orth\_poly\_01.m)



**证明.** 由题意可知,  $m = 6, n = 2$ . 直接计算可得

$$(p_0, p_0) = \sum_{i=0}^m 1 = 7, \quad (f, p_0) = \sum_{i=0}^m y_i = 15.05, \quad (xp_0, p_0) = \sum_{i=0}^m x_i = 4.5.$$

所以

$$a_0 = \frac{(f, p_0)}{(p_0, p_0)} \approx 2.15, \quad \alpha_0 = \frac{(xp_0, p_0)}{(p_0, p_0)} \approx 0.64.$$

于是可得

$$p_1(x) = x - 0.64.$$

不断利用三项递推公式, 可求得

$$a_1 \approx 1.98, \quad \alpha_1 \approx 0.36, \quad \beta_0 \approx 0.094;$$

$$p_2(x) = x^2 - 0.98x + 0.12;$$

$$a_2 \approx 1.00.$$

所以此组数据的 2 次最小二乘拟合多项式为

$$S_2^*(x) = a_0 p_0(x) + a_1 p_1(x) + a_2 p_2(x) = x^2 + x + 1.$$

□

🔗 MATLAB 中计算最小二乘拟合多项式的函数是 **polyfit**. 另外, MATLAB 还提供了可视化的曲线拟合工具, 支持多种拟合方式, 也可以自定义基函数, 启动命令是 **cftool** (Fit curves and surfaces to data).

## 8.6 有理逼近

### 为什么有理逼近?

- 多项式逼近的优点: 计算简便.
- 多项式逼近的缺点: 若原函数  $f(x)$  在某点附近无解 (可能存在奇点), 则用多项式逼近效果较差.

### 有理逼近

用有理函数来做逼近

$$R_{nm}(x) \triangleq \frac{p_n(x)}{q_m(x)} = \frac{\sum_{k=0}^n a_k x^k}{\sum_{k=0}^m b_k x^k}.$$

- 最佳有理一直逼近:

$$\min \|R_{nm} - f(x)\|_{\infty}$$

- 最佳有理平方逼近:

$$\min \|R_{nm} - f(x)\|_2$$

### Pade 逼近与 Taylor 展开

Pade 逼近基本思想: 以尽可能快的速度与 Taylor 展开式相匹配.

**定义 8.7** 设  $f(x) \in C^{N+1}(-a, a)$ ,  $N = m + n$ , 如果有理函数

$$R_{nm}(x) \triangleq \frac{a_0 + a_1 x + \cdots + a_n x^n}{1 + b_1 x + \cdots + b_m x^m} = \frac{p_n(x)}{q_m(x)}$$

满足 (1)  $p_n(x)$  与  $q_m(x)$  无公共因式; (2)  $R_{nm}(x)^{(k)}(0) = f^{(k)}(0)$ ,  $k = 0, 1, \dots, N$ ; 则称  $R_{nm}(x)$  为  $f(x)$  在  $x = 0$  处的  $(n, m)$  阶 **Pade 逼近**, 记作  $R(n, m)$ .

#### 几点说明

- Pade 逼近是一类特殊的有理逼近.
- $R_{nm}(x)$  和  $f(x)$  的 Taylor 展开式的前  $m + n$  项是一样的.
- $q_m(x)$  的常数项为 1 (标准化处理,  $b_0 \neq 0$ )

### $R(n, m)$ 的计算

记  $N = m + n$ , 设  $f(x) \in C^{N+1}(-a, a)$ ,  $f(x)$  在  $x = 0$  处的 Taylor 展开为

$$f(x) = f(0) + f^{(1)}(0)x + \frac{1}{2!}f^{(2)}(0)x^2 + \cdots + \frac{1}{N!}f^{(N)}(0)x^N + \frac{1}{(N+1)!}f^{(N+1)}(\xi)x^{N+1},$$

前  $N + 1$  项部分和为

$$p(x) = c_0 + c_1 x + c_2 x^2 + \cdots + c_N x^N, \quad c_k = \frac{1}{k!}f^{(k)}(0).$$



设  $h(x) = p(x)q_m(x) - p_n(x)$ , 则条件  $R_{nm}(x)^{(k)}(0) = f^{(k)}(0)$  等价于

$$h^{(k)}(0) = 0, \quad k = 0, 1, \dots, N$$

又通过计算可知

$$(p(x)q_m(x) - p_n(x))^{(k)} \Big|_{x=0} = k! \sum_{j=0}^k c_j b_{k-j} - k! a_k,$$


所以可得

$$\begin{cases} a_k = \sum_{j=0}^{k-1} c_j b_{k-j} + c_k & k = 0, 1, 2, \dots, n \\ 0 = \sum_{j=0}^{k-1} b_{k-j}, & k = n+1, n+2, \dots, N. \end{cases} \quad (8.17)$$

为了书写方便, 当  $j > m$  时令  $b_j = 0$ .

$N+1$  个未知量:  $a_0, a_1, \dots, a_n, b_1, \dots, b_m$ ,  $N+1$  个方程, 因此存在唯一解的充要条件是系数矩阵非奇异.

**定理 8.26** 设  $f(x) \in C^{N+1}(-a, a)$ ,  $N = m + n$ , 则  $R_{nm}(x)$  (其中  $b_0 = 1$ ) 是  $f(x)$  的  $(n, m)$  阶 Pade 逼近的充要条件是  $p_n$  和  $q_n$  的系数满足方程 (8.17).

  $b_1, b_2, \dots, b_m$  通过解方程得到,  $a_0, a_1, \dots, a_n$  可直接计算.

## 8.7 课后练习

练习 8.1 证明函数组  $\{1, x, x^2, \dots, x^n\}$  线性无关.

练习 8.2 设  $f(x), g(x) \in C^1[a, b]$ , 定义

$$(1) (f, g) = \int_a^b f'(x)g'(x) dx;$$

$$(2) (f, g) = \int_a^b f'(x)g'(x) dx + f(a)g(a);$$

问它们是否构成内积.

练习 8.3 令  $T_n^*(x) = T_n(2x - 1)$ ,  $x \in [0, 1]$ , 试证  $\{T_n^*\}$  是在  $[0, 1]$  上的带权  $\rho(x) = \frac{1}{\sqrt{x-x^2}}$  的正交多项式, 并求  $T_0^*(x), T_1^*(x), T_2^*(x), T_3^*(x)$ .

练习 8.4 设权函数  $\rho(x) = 1+x^2$ ,  $x \in [-1, 1]$ . 试求首项系数为 1 的关于  $\rho(x)$  正交的多项式  $\varphi_n(x)$ ,  $n = 0, 1, 2, 3$ .

练习 8.5 试证明由 (8.6) 给出的第二类 Chebyshev 多项式是  $[-1, 1]$  上带权

$$\rho(x) = \sqrt{1-x^2}$$

的正交多项式.

练习 8.6 设  $n \geq 1$ , 证明: 对每一个 Chebyshev 多项式  $T_n(x)$ , 有

$$\int_{-1}^1 \frac{(T_n(x))^2}{\sqrt{1-x^2}} dx = \frac{\pi}{2}.$$

练习 8.7 用  $T_3(x)$  的零点做插值节点, 求  $f(x) = e^x$  在区间  $[-1, 1]$  上的二次插值多项式, 并估计误差.

练习 8.8 设  $f(x) = x^2 + 3x + 2$ ,  $x \in [0, 1]$ , 试求  $f(x)$  在  $[0, 1]$  上关于  $\rho(x) = 1$ ,  $\Phi = \text{span}\{1, x\}$  的最佳平方逼近多项式. 若取  $\Phi = \text{span}\{1, x, x^2\}$ , 则最佳平方逼近多项式是什么?

练习 8.9 求  $f(x) = x^3$  在  $[-1, 1]$  上的二次最佳平方逼近多项式.

练习 8.10 求  $f(x)$  在指定区间上对于  $\Phi = \text{span}\{1, x\}$  的最佳平方逼近多项式:

$$(1) f(x) = \frac{1}{x}, x \in [1, 3];$$

$$(2) f(x) = e^x, x \in [0, 1];$$

$$(3) f(x) = \cos \pi x, x \in [0, 1];$$

$$(4) f(x) = \ln x, x \in [1, 2].$$

练习 8.11 设  $f(x) = \sin \frac{\pi x}{2}$ , 利用 Legendre 多项式求  $f(x)$  在  $[-1, 1]$  上的三次最佳平方逼近多项式.

练习 8.12 已知实验数据如下:

$x_i$	19	25	31	38	44
$y_i$	19.0	32.3	49.9	73.3	97.8





用最小二乘法求形如  $y = a + bx^2$  的经验公式, 并计算均方误差.

**练习 8.13** 证明: 由递推公式 (8.16) 所定义的多项式是关于点集  $\{x_i\}_{i=0}^m$  带权  $\{\omega_i\}_{i=0}^m$  正交的.

**练习 8.14** 设  $\omega_1, \omega_2, \dots, \omega_n$  为给定的正实数, 对任意  $x, y \in \mathbb{R}^n$ , 定义

$$(x, y)_\omega = \sum_{i=1}^n \omega_i x_i y_i,$$

证明:  $(x, y)_\omega$  是  $\mathbb{R}^n$  上的内积.

**练习 8.15** 证明定理 8.22 中的结论, 即:

$p_{n-1}^*(x) = f(x) - a_n \tilde{T}_n(x)$  是  $n$  次多项式  $f(x) = a_n x^n + \dots + a_1 x + a_0$  在  $[-1, 1]$  上的  $n-1$  次最佳一致逼近多项式, 其中  $a_n \neq 0$ .

# 9

## 数值积分与数值微分

考虑定积分

$$I = \int_a^b f(x) \, dx. \quad (9.1)$$

在微积分中, 我们可以使用 Newton-Leibnitz 公式

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

其中  $F(x)$  是被积函数  $f(x)$  的一个原函数. 但是

- 在很多情况下, 被积函数的原函数很难求出, 或者原函数很复杂, 如  $f(x) = \frac{1}{1+x^6}$  的原函数为


$$F(x) = \frac{1}{3} \arctan x + \frac{1}{6} \arctan \left( x - \frac{1}{x} \right) + \frac{1}{4\sqrt{3}} \ln \frac{x^2 + x\sqrt{3} + 1}{x^2 - x\sqrt{3} + 1} + C.$$

- 原函数无法用初等函数表示, 如

$$f(x) = \frac{\sin x}{x}, \quad f(x) = e^{-x^2}, \quad f(x) = \sqrt{1 + k^2 \sin^2 x}.$$

- 在某些实际应用中, 被积函数  $f(x)$  的表达式是未知的, 只是通过实验或测量等手段给出了某些离散点上的值.

在这些情况下, 我们就需要考虑通过近似方法来计算定积分的近似值, 即**数值积分**.

 数值积分的基本思想是用函数值的线性组合来近似定积分, 有时也会用到导数值.

### 数值积分主要研究的问题

数值积分主要考虑以下问题:

- (1) 求积公式的构造;
- (2) 精确程度的衡量;
- (3) 余项估计/误差估计.

## 9.1 数值积分基本概念

### 9.1.1 机械求积公式

设  $f(x) \in C[a, b]$ , 取节点  $a = x_0 < x_1 < x_2 < \cdots < x_n < x_{n+1} = b$ , 根据定积分的定义, 有

$$\int_a^b f(x) dx = \lim_{h \rightarrow 0} \sum_{i=0}^n h_i f(\xi_i), \quad \xi_i \in [x_i, x_{i+1}],$$


其中  $h_i = x_{i+1} - x_i$ ,  $h = \max_i \{h_i\}$ . 当  $h$  充分小,  $n$  充分大时, 我们就有下面的近似公式

$$\int_a^b f(x) dx \approx \sum_{i=0}^n h_i f(\xi_i). \quad (9.2)$$

这就是目前常用的求积公式. 为了方便起见, 我们将上述公式改写为 (将记号  $h_i$  和  $\xi_i$  更换为  $A_i$  和  $x_i$ )

$$\int_a^b f(x) dx \approx \sum_{i=0}^n A_i f(x_i) \triangleq I_n(f). \quad (9.3)$$

这里  $x_i$  称为**求积节点**, 满足  $a \leq x_0 < x_1 < \cdots < x_n \leq b$ , 系数  $A_i$  称为**求积系数**, 与函数  $f(x)$  无关. 该求积公式就称为**机械求积公式**.

 机械求积公式只包含函数值, 但求积公式并不局限于机械求积公式, 有些求积公式可能会包含其它信息, 如导数值等.

**例 9.1** 设  $f(x) \in C[a, b]$ , 则由积分中值定理可知, 存在  $\xi \in [a, b]$  使得

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

不幸的是,  $\xi$  的取值往往是不可知的. 但我们可以考虑用  $[a, b]$  中的某个点上的函数值来近似  $f(\xi)$ :

- 如果用左端点的函数值  $f(a)$  来近似  $f(\xi)$ , 则可得**左矩形公式**:

$$\int_a^b f(x) dx \approx f(a)(b-a);$$

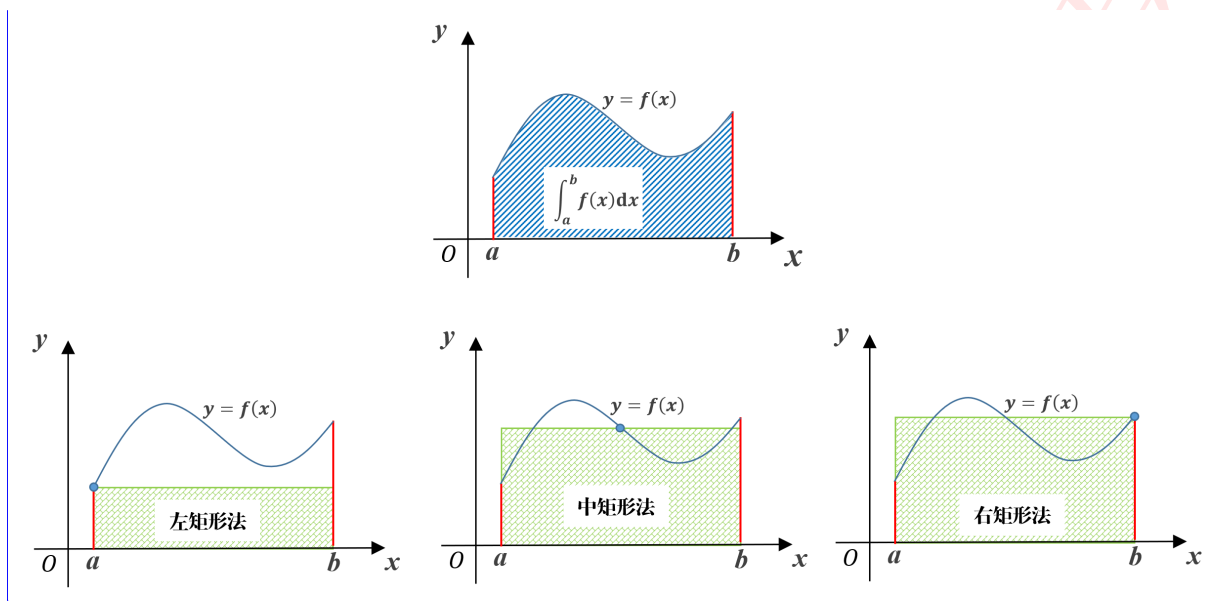
- 如果用右端点的函数值  $f(b)$  来近似  $f(\xi)$ , 则可得**右矩形公式**:

$$\int_a^b f(x) dx \approx f(b)(b-a);$$

- 如果用中点的函数值来近似  $f(\xi)$ , 则可得**中矩形公式**:

$$\int_a^b f(x) dx \approx f\left(\frac{a+b}{2}\right)(b-a).$$

这类公式称为**矩形公式**, 其几何意义就是用矩形的面积来近似曲边梯形的面积, 见下图.



### 9.1.2 代数精度

根据 Weierstrass 逼近定理 8.1 可知, 任意一个连续函数都可以通过多项式来一致逼近, 因此, 如果一个求积公式能对次数较高的多项式精确成立, 那么我们就认为该求积公式具有较高的精度. 基于这样的想法, 我们给出下面的代数精度概念.

**定义 9.1** 如果一个求积公式对所有次数不超过  $m$  的多项式精度成立, 但对  $m+1$  次多项式不精确成立, 则称该求积公式具有  $m$  次**代数精度**.

#### 代数精度的计算方法

由定义可知, 一个求积公式具有  $m$  次代数精度当且仅当求积公式

- (1) 对  $f(x) = 1, x, x^2, \dots, x^m$  精确成立;
- (2) 对  $f(x) = x^{m+1}$  不精确成立.

这给出了计算一个求积公式的代数精度的方法.

**例 9.2** 试确定系数  $A_i$ , 使得下面的求积公式具有尽可能高的代数精度, 并求出此求积公式的代数精度.

$$\int_{-1}^1 f(x) dx \approx A_0 f(-1) + A_1 f(0) + A_2 f(1).$$

**解.** 分别取  $f(x) = 1, x, x^2$ , 令求积公式精确成立, 可得

$$\begin{cases} A_0 + A_1 + A_2 = b - a \\ -A_0 + A_2 = 0 \\ A_0 + A_2 = \frac{2}{3} \end{cases}$$



求解该方程组, 可得  $A_0 = \frac{1}{3}$ ,  $A_1 = \frac{4}{3}$ ,  $A_2 = \frac{1}{3}$ . 因此求积公式为

$$\int_{-1}^1 f(x) dx \approx \frac{1}{3}f(-1) + \frac{4}{3}f(0) + \frac{1}{3}f(1).$$

将  $f(x) = x^3$  代入可得, 公式左边为

□

由于求积公式 (9.3) 中含有  $n+1$  个参数, 因此可以选取合适的  $A_i$  的值, 使得求积公式 (9.3) 至少具有  $n$  次代数精度.

**例 9.3 (非机械求积公式)** 试确定下面求积公式中的系数, 使其具有尽可能高的代数精度:

$$\int_0^1 f(x) dx \approx A_0 f(0) + A_1 f(1) + B_0 f'(0).$$

(板书)

**引理 9.1** 设机械求积公式 (9.3) 具有  $m(\geq 0)$  次代数精度, 则有

$$A_0 + A_1 + \cdots + A_n = b - a. \quad (9.4)$$

**证明.** 将  $f(x) = 1$  代入求积公式 (9.3), 令等式精确成立即可. (也可以直接从 (9.2) 中得出) □

上面的结论是机械求积公式的一个基本性质.

### 9.1.3 收敛性与稳定性

**定义 9.2** 设求积公式的余项为  $R[f]$ , 若

$$\lim_{h \rightarrow 0} R[f] = 0,$$

则称求积公式是收敛的, 其中  $h = \max_{0 \leq i \leq n-1} \{x_{i+1} - x_i\}$ .

由定义可以, 求积公式的收敛性与定积分的存在性是类似的, 即当分割足够细时极限存在, 而且两者的值相等.

在利用机械求积公式计算定积分时, 需要计算函数值. 由于存在一定的舍入误差, 因此最后的结果也会带有一定的误差. 求积公式的稳定性就是用来表示这些舍入误差对计算结果的影响.

**定义 9.3** 考虑机械求积公式 (9.3). 设  $\tilde{f}_k$  是计算  $f(x_k)$  时得到的近似值. 如果对任给的  $\varepsilon > 0$ , 都存在  $\delta > 0$ , 使得当  $|f(x_k) - \tilde{f}_k| < \delta$  对  $k = 0, 1, 2, \dots, n$  都成立时, 有

$$\left| \sum_{k=0}^n A_k f(x_k) - \sum_{k=0}^n A_k \tilde{f}_k \right| < \varepsilon,$$

则称求积公式 (9.3) 是稳定的.

下面给出一个判别机械求积公式稳定性的充分条件.

**定理 9.2** 若机械求积公式 (9.3) 中的求积系数  $A_i$  都是正数, 则求积公式是稳定的.

(板书)

### 9.1.4 插值型求积公式

一个构造求积公式的常用方法就是使用插值多项式. 设  $L_n(x)$  是  $f(x)$  关于节点  $a \leq x_0 < x_1 < \cdots < x_n \leq b$  的  $n$  次插值多项式, 则

$$\begin{aligned} \int_a^b f(x) dx &\approx \int_a^b L_n(x) dx = \int_a^b \sum_{k=0}^n l_k(x) f(x_k) dx \\ &= \sum_{k=0}^n \left( \int_a^b l_k(x) dx \right) f(x_k) \triangleq \sum_{k=0}^n A_k f(x_k). \end{aligned} \quad (9.5)$$

这就是**插值型求积公式**, 其中  $l_k(x)$  是  $n$  次 Lagrange 基函数,  $A_k = \int_a^b l_k(x) dx$ .

由插值余项公式可知, 插值型求积公式 (9.5) 的余项为

$$R[f] = \int_a^b (f(x) - L_n(x)) dx = \int_a^b R_n(x) dx = \int_a^b \frac{f^{(n+1)}(\xi_x)}{(n+1)!} \omega_{n+1}(x) dx, \quad (9.6)$$

其中

$$\omega_{n+1}(x) = (x - x_0)(x - x_1) \cdots (x - x_n).$$

由于  $\xi_x$  是关于  $x$  未知函数, 上面的误差值是无法得到的, 因此我们通常用下面的方法来估计误差

$$|R[f]| = \left| \int_a^b \frac{f^{(n+1)}(\xi_x)}{(n+1)!} \omega_{n+1}(x) dx \right| \leq \frac{M_{n+1}}{(n+1)!} \int_a^b |\omega_{n+1}(x)| dx,$$

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

**引理 9.3** 插值型求积公式 (9.5) 至少具有  $n$  次代数精度.

**证明.** 依次将  $f(x) = 1, x, x^2, \dots, x^n$  代入, 可得  $R_n(x) = 0$ . 故求积公式对  $f(x) = 1, x, x^2, \dots, x^n$  精确成立, 所以代数精度至少为  $n$ .  $\square$

事实上, 我们有下面的性质.

**定理 9.4** 机械求积公式 (9.3) 至少具有  $n$  次代数精度的充要条件是该公式是插值型的.

(板书)

当机械求积公式具有尽可能高的代数精度时, 它总是插值型的.



## 9.2 Newton-Cotes 公式

**定义 9.4** 如果插值型求积公式 (9.5) 中的节点为等距节点, 即

$$x_k = a + kh, \quad h = \frac{b-a}{n}, \quad k = 0, 1, 2, \dots, n,$$

则该求积公式就称为 **Newton-Cotes 公式**, 记为

$$I_n(f) = (b-a) \sum_{k=0}^n C_k^{(n)} f(x_k), \quad (9.7)$$

其中  $C_k^{(n)}$  称为 **Cotes 系数**, 其值为

$$C_k^{(n)} = \frac{1}{b-a} \int_a^b l_k(x) dx = \frac{h}{b-a} \int_0^n \prod_{i=0, i \neq k}^n \frac{t-i}{k-i} dt = \frac{(-1)^{n-k}}{n k! (n-k)!} \int_0^n \prod_{i=0, i \neq k}^n (t-i) dt.$$

### Cotes 系数的两个简单性质

$$(1) \sum_{k=0}^n C_k^{(n)} = 1; \quad (2) C_k^{(n)} = C_{n-k}^{(n)}, \quad k = 0, 1, 2, \dots, n.$$

### 9.2.1 常用的低次 Newton-Cotes 公式

下面给出几个常用的低次 Newton-Cotes 公式:

- 当  $n = 1$  时, 可得  $C_0^{(1)} = C_1^{(1)} = \frac{1}{2}$ , 此时的 Newton-Cotes 公式为

$$I_1(f) = \frac{b-a}{2} (f(a) + f(b)). \quad (9.8)$$

这就是 **梯形公式**, 通常记作为  $T(f)$ .

- 当  $n = 2$  时, 可得  $C_0^{(2)} = \frac{1}{6}$ ,  $C_1^{(2)} = \frac{4}{6}$ ,  $C_2^{(2)} = \frac{1}{6}$ , 此时的 Newton-Cotes 公式为

$$I_2(f) = \frac{b-a}{6} \left( f(a) + 4f\left(\frac{a+b}{2}\right) + f(b) \right). \quad (9.9)$$

这就是 **Simpson 公式** 或 **抛物线公式**, 通常记作为  $S(f)$ .

- 当  $n = 3$  时, 可得  $C_0^{(3)} = \frac{1}{8}$ ,  $C_1^{(3)} = \frac{3}{8}$ ,  $C_2^{(3)} = \frac{3}{8}$ ,  $C_3^{(3)} = \frac{1}{8}$ , 此时的 Newton-Cotes 公式为

$$I_3(f) = \frac{b-a}{8} (f(x_0) + 3f(x_1) + 3f(x_2) + f(x_3)). \quad (9.10)$$

该公式称为 **Simpson 3/8 公式** 或 **Boole 公式** 或 **Milne 公式**.

- 当  $n = 4$  时, 可得  $C_0^{(4)} = \frac{7}{90}$ ,  $C_1^{(4)} = \frac{32}{90}$ ,  $C_2^{(4)} = \frac{12}{90}$ ,  $C_3^{(4)} = \frac{32}{90}$ ,  $C_4^{(4)} = \frac{7}{90}$ , 此时的 Newton-Cotes 公式为

$$I_4(f) = \frac{b-a}{90} (7f(x_0) + 32f(x_1) + 12f(x_2) + 32f(x_3) + 7f(x_4)). \quad (9.11)$$

这公式就是 **Cotes 公式**.



当  $n > 7$  时, Cotes 系数中会出现负数, 会导致算法的不稳定, 因此我们不考虑  $n > 7$  时的 Newton-Cotes 公式.

梯形公式和抛物线公式简单易用, 因此很受欢迎.

**定理 9.5** 当  $n$  是奇数时, Newton-Cotes 公式至少具有  $n$  次代数精度. 当  $n$  是偶数时, Newton-Cotes 公式至少具有  $n+1$  次代数精度.

**证明.** 由于 Newton-Cotes 公式是插值型的, 因此它至少具有  $n$  次代数精度.

当  $n$  是偶数时, 将  $f(x) = x^{n+1}$  代入余项公式 (9.6) 可得

$$R[f] = \int_a^b \frac{f^{(n+1)}(\xi_x)}{(n+1)!} \omega_{n+1}(x) dx = \int_a^b \omega_{n+1}(x) dx$$

做变量代换  $x = a + th$ . 由于  $x_k = a + kh$ , 所以

$$R[f] = \int_a^b \omega_{n+1}(x) dx = h^{n+2} \int_0^n \prod_{k=0}^n (t-k) dt$$

由于  $n$  是偶数, 再做变量代换  $t = n-s$ , 可得

$$R[f] = h^{n+2} \int_n^0 (-1)^{n+2} \prod_{k=0}^n (s-(n-k)) ds = (-1)^{n+3} \int_0^n \prod_{i=0}^n (s-i) ds = -R[f].$$

所以  $R[f] = 0$ , 故 Newton-Cotes 公式对  $f(x) = x^{n+1}$  精确成立, 即具有  $n+1$  次代数精度.  $\square$

## 9.2.2 余项公式的推导

首先给出几个引理.

**引理 9.6** 设基于节点  $x_0, x_1, x_2, \dots, x_n$  的插值型求积公式 (9.5) 具有  $m$  ( $m \geq n$ ) 次代数精度, 则对任意  $f(x) \in C[a, b]$ , 有

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

其中  $p_m(x)$  是  $f(x)$  关于节点  $x_0, x_1, x_2, \dots, x_n$  的  $m$  次插值多项式, 即  $p_m(x)$  满足

$$p_m(x_k) = f(x_k), \quad k = 0, 1, 2, \dots, n.$$

(注: 当  $m > n$  时, 其它  $m-n$  个插值条件可以任取, 如要求导数值相等).

**证明.** 由于求积公式具有  $m$  次代数精度, 即对  $m$  次多项式精确成立, 所以

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

$\square$

上述结论也可以推广到包含导数的求积公式.

**引理 9.7** 设基于节点  $x_0, x_1, x_2, \dots, x_n$  的求积公式为

$$I_n(f) = \sum_{k=0}^n A_k f(x_k) + \sum_{j \in \mathbb{Z}_1} B_j f'(x_j). \quad (9.12)$$

其中  $\mathbb{Z}_1 \subset \{0, 1, 2, \dots, n\}$ , 即可以只包含部分节点上的导数值. 若该求积公式具有  $m$  ( $m \geq n$ )





次代数精度, 则对任意  $f(x) \in C[a, b]$ , 有

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

其中  $p_m(x)$  是  $f(x)$  关于节点  $x_0, x_1, x_2, \dots, x_n$  的  $m$  次 Hermite 插值多项式, 即  $p_m(x)$  满足

$$\begin{cases} p_m(x_k) = f(x_k), & k = 0, 1, 2, \dots, n \\ p'_m(x_j) = f'(x_j), & j \in \mathbb{Z}_1. \end{cases}$$

(注: 如果插值条件个数小于  $m+1$ , 则其它插值条件可以任取)

假设  $\mathbb{Z}_1$  中的元素个数为  $r$  ( $r \leq n+1$ ), 则一般总是有  $m \geq n+r$ .

如果求积公式中包含二阶以上的导数 (如后面会提到的修正 Simpson 公式 (9.19)), 我们仍可以得到相类似的结论.

下面是关于差商的连续性.

**引理 9.8** 设  $a = x_0 < x_1 < x_2 < \dots < x_{n-1} < x_n = b$ , 函数  $f(x) \in C^1[a, b]$ , 则差商

$$g(x) \triangleq f[x, x_0, x_1, \dots, x_n]$$

关于  $x$  在  $[a, b]$  上连续. 这里  $g(x)$  在节点  $x_k$  上的值是通过重节点差商来定义的. 进一步, 若  $f(x) \in C^2[a, b]$ , 则有

$$g'(x) = f[x, x, x_0, x_1, \dots, x_n],$$

且  $g'(x)$  也关于  $x$  在  $[a, b]$  上连续.

**证明.** 可参见: E. Isaacson and H. Keller, Analysis of Numerical Methods, John Wiley and Sons, London-New York, 1966. 第 255 页.  $\square$

上述结论可以推广到重节点情形, 即允许节点  $x_0, x_1, x_2, \dots, x_n$  有重合的.

**引理 9.9** 设  $a = x_0 \leq x_1 \leq x_2 \leq \dots \leq x_{n-1} \leq x_n = b$ , 函数  $f(x) \in C^{n+1}[a, b]$ , 则差商

$$g(x) \triangleq f[x, x_0, x_1, \dots, x_n]$$

关于  $x$  在  $[a, b]$  上连续. 进一步, 若  $f(x) \in C^{n+2}[a, b]$ , 则有

$$g'(x) = f[x, x, x_0, x_1, \dots, x_n],$$

且  $g'(x)$  也关于  $x$  在  $[a, b]$  上连续.

## 梯形公式的余项

**定理 9.10** 设  $f(x) \in C^2[a, b]$ , 则梯形求积公式的余项为

$$R[f] = -\frac{(b-a)^3}{12} f''(\eta), \quad \eta \in (a, b),$$

所以,带余项的梯形公式可写为

$$\int_a^b f(x) dx = \frac{b-a}{2} (f(a) + f(b)) - \frac{(b-a)^3}{12} f''(\eta), \quad \eta \in (a, b). \quad (9.13)$$

**证明.** 设  $p_1(x)$  是  $f(x)$  关于节点  $x_0 = a, x_1 = b$  的一次插值多项式. 由 Newton 插值的余项公式可知

$$f(x) - p_1(x) = f[x, x_0, x_1] \omega_2(x),$$

其中  $\omega_2(x) = (x - x_0)(x - x_1)$ . 由于梯形公式的代数精度为 1, 故

$$\int_a^b p_1(x) dx = \frac{b-a}{2} (p_1(a) + p_1(b)).$$

所以

$$\begin{aligned} R[f] &= \int_a^b f(x) dx - \frac{b-a}{2} (f(a) + f(b)) \\ &= \int_a^b f(x) dx - \frac{b-a}{2} (p_1(a) + p_1(b)) \quad (\text{插值条件: } p_1(a) = f(a), p_1(b) = f(b)) \\ &= \int_a^b (f(x) - p_1(x)) dx \\ &= \int_a^b f[x, x_0, x_1] \omega_2(x) dx. \end{aligned}$$

由于  $f[x, x_0, x_1]$  关于  $x$  在  $[a, b]$  上连续,  $\omega_2(x)$  在  $[a, b]$  内不变号, 所以由积分中值定理可知, 存在  $z \in (a, b)$ , 使得

$$R[f] = f[z, x_0, x_1] \int_a^b \omega_2(x) dx = -\frac{(b-a)^3}{6} f[z, x_0, x_1].$$

由差商与导数之间的关系可知, 存在  $\eta \in (a, b)$ , 使得  $f[z, x_0, x_1] = \frac{f''(\eta)}{2!}$ . 这里  $\eta$  与  $z$  有关. 所以

$$R[f] = -\frac{(b-a)^3}{12} f''(\eta), \quad \eta \in (a, b).$$

□

 如果使用 Lagrange 插值余项公式, 则可得

$$R[f] = \int_a^b \frac{1}{2!} f''(\xi_x) (x-a)(x-b) dx, \quad \xi_x \in (a, b).$$

易知  $(x-a)(x-b)$  在  $[a, b]$  内不变号, 如果  $f''(\xi_x)$  关于  $x$  连续, 则由积分中值定理可知, 存在  $\eta \in (a, b)$  使得

$$R[f] = \frac{1}{2!} f''(\eta) \int_a^b (x-a)(x-b) dx = -\frac{(b-a)^3}{12} f''(\eta).$$

需要指出的是, 这里要证明  $f''(\xi_x)$  关于  $x$  是连续的. 事实上, 由 Lagrange 插值余项公式可知

$$f(x) - p_1(x) = \frac{1}{2!} f''(\xi_x) (x-a)(x-b),$$

即

$$f''(\xi_x) = \frac{2!(f(x) - p_1(x))}{(x-a)(x-b)} \triangleq g(x).$$

如果  $f \in C^2[a, b]$ , 则上式右端 (即  $g(x)$ ) 显然在除插值节点以外的所有点都连续. 应用



L'Hôpital 法则, 我们可以求得  $g(x)$  在插值节点处的极限 (在两端点处为右极限或左极限). 而根据余项公式, 在插值节点处,  $f''(\xi_x)$  可以取任意值. 因此, 我们可以将  $g(x)$  的极限设为  $f''(\xi_x)$  在节点处的值, 这样  $f''(\xi_x)$  就在整个区间上连续. 以上结论可推广到一般情形, 即: 设  $f \in C^{n+1}[a, b]$ , 可得  $n$  次多项式插值余项

$$f(x) - p_n(x) = \frac{1}{(n+1)!} f^{(n+1)}(\xi_x)(x-x_0)(x-x_1)\cdots(x-x_n), \quad \xi_x \in [a, b],$$

通过定义  $f^{(n+1)}(\xi_x)$  在插值节点处的值, 可使得  $f^{(n+1)}(\xi_x)$  关于  $x$  是连续的.

### Simpson 公式的余项

**定理 9.11** 设  $f(x) \in C^4[a, b]$ , 则 Simpson 求积公式的余项为

$$R[f] = -\frac{(b-a)^5}{2880} f^{(4)}(\eta), \quad \eta \in (a, b),$$

所以, 带余项的 Simpson 公式可写为

$$\int_a^b f(x) dx = \frac{b-a}{6} \left( f(a) + 4f\left(\frac{a+b}{2}\right) + f(b) \right) - \frac{(b-a)^5}{2880} f^{(4)}(\eta), \quad \eta \in (a, b). \quad (9.14)$$

**证明.** 基本思想与梯形求积公式余项类似, 但具体做法有所不同.

由于 Simpson 公式具有 3 次代数精度, 构造  $f(x)$  关于点  $x_0 = a, x_1 = \frac{1}{2}(a+b), x_2 = b$  的三点三次 Hermite 插值多项式  $H_3(x)$ , 满足

$$H_3(x_k) = f(x_k), \quad k = 0, 1, 2, \quad H'_3(x_1) = f'(x_1).$$

于是有

$$\int_a^b H_3(x) dx = \frac{b-a}{6} (H_3(x_0) + 4H_3(x_1) + H_3(x_2)), \quad (9.15)$$

且插值余项为

$$R(x) = \frac{f^{(4)}(\xi_x)}{4!} (x-x_0)(x-x_1)^2(x-x_2).$$

所以

$$\begin{aligned} R[f] &= \int_a^b f(x) dx - \frac{b-a}{6} (f(x_0) + 4f(x_1) + f(x_2)) \\ &= \int_a^b f(x) dx - \frac{b-a}{6} (H_3(x_0) + 4H_3(x_1) + H_3(x_2)) \quad (\text{插值条件}) \\ &= \int_a^b f(x) dx - \int_a^b H_3(x) dx \quad (\text{由(9.15)}) \\ &= \int_a^b (f(x) - H_3(x)) dx \\ &= \int_a^b \frac{f^{(4)}(\xi_x)}{4!} (x-x_0)(x-x_1)^2(x-x_2) dx. \end{aligned}$$

由于  $f^{(4)}(\xi_x)$  是  $x$  的连续函数, 且  $(x-x_0)(x-x_1)^2(x-x_2)$  在  $[a, b]$  内不变号, 所以由积分中值定理可知, 存在  $\eta \in (a, b)$ , 使得

$$R[f] = \int_a^b \frac{f^{(4)}(\xi_x)}{4!} (x-x_0)(x-x_1)^2(x-x_2) dx = \frac{f^{(4)}(\eta)}{4!} \int_a^b (x-x_0)(x-x_1)^2(x-x_2) dx.$$

将  $x_0 = a, x_1 = \frac{1}{2}(a+b), x_2 = b$  代入, 可求得

$$\int_a^b (x-x_0)(x-x_1)^2(x-x_2) dx = -\frac{(b-a)^5}{120}.$$

因此, Simpson 公式的余项为

$$R[f] = -\frac{f^{(4)}(\eta)}{4!} \cdot \frac{(b-a)^5}{120} = -\frac{(b-a)^5}{2880} f^{(4)}(\eta), \quad \eta \in (a, b).$$

□

事实上, 我们可以将  $H_3(x)$  写成 Newton 插值形式, 即

$$\begin{aligned} H_3(x) &= f(x_0) + f[x_0, x_1](x-x_0) + f[x_0, x_1, x_1](x-x_0)(x-x_1) \\ &\quad + f[x_0, x_1, x_1, x_2](x-x_0)(x-x_1)^2. \end{aligned}$$

这可以看作是 **重节点 Newton 插值**. 因此, 插值余项可表示为

$$R_3(x) = f[x, x_0, x_1, x_1, x_2](x-x_0)(x-x_1)^2(x-x_2).$$

请读者自行验证.

### 计算求积公式余项小结 (三步曲)

- (1) 计算求积公式的代数精度, 设为  $m$ ;
- (2) 构造  $m$  次插值多项式, 写出插值条件和插值余项 (除导数部分外, 要确保不变号);
- (3) 利用积分中值定理, 计算出求积公式的余项.

### 9.2.3 Newton-Cotes 公式余项的一般形式

**定理 9.12** 当  $n$  是奇数时, 若  $f(x) \in C^{n+1}[a, b]$ , 则 Newton-Cotes 公式的余项为

$$R[f] = \frac{h^{n+2} f^{(n+1)}(\eta)}{(n+1)!} \int_0^n t(t-1)(t-2) \cdots (t-n) dt, \quad \eta \in (a, b).$$

当  $n$  是偶数时, 若  $f(x) \in C^{n+2}[a, b]$ , 则 Newton-Cotes 公式的余项为

$$R[f] = \frac{h^{n+3} f^{(n+2)}(\eta)}{(n+2)!} \int_0^n t^2(t-1)(t-2) \cdots (t-n) dt, \quad \eta \in (a, b).$$

**证明.** 可参见: J. Stoer and R. Bulirsch, Introduction to Numerical Analysis, 3rd Edition, 2002. □

### 9.2.4 一般求积公式余项

考虑更一般的求积公式 (带导数信息)

$$\int_a^b f(x) dx = \sum_{k=0}^n A_k f(x_k) + \sum_{i_1 \in \mathbb{Z}_1} A_{i_1} f'(x_{i_1}) + \sum_{i_2 \in \mathbb{Z}_2} A_{i_2} f''(x_{i_2}) + \cdots + \sum_{i_r \in \mathbb{Z}_r} A_{i_r} f^{(r)}(x_{i_r}). \quad (9.16)$$

其中  $\mathbb{Z}_j \subset \{0, 1, 2, \dots, n\}$ , 即求积公式中可以包含全部或部分节点上的导数信息.

设求积公式 (9.16) 的代数精度为  $m$ , 若  $f(x) \in C^{m+1}[a, b]$ , 则其余项可表示为

$$R[f] = \int_a^b f^{(m+1)}(t) K(t) dt,$$



其中  $K(t)$  称为 **Peano 核**, 具体表达式可参见 “Introduction to Numerical Analysis” (Stoer and Bulirsch, 3rd Edition, 2002).

在大多数求积公式中,  $K(t)$  在  $[a, b]$  内不变号, 此时, 根据积分中值定理可知, 存在  $\eta \in (a, b)$  使得

$$R[f] = f^{(m+1)}(\eta) \int_a^b K(t) dt. \quad (9.17)$$

需要注意的是, 余项公式 (9.17) 并不是对所有求积公式 (9.16) 都成立.

**例 9.4** 给出下面求积公式的余项

$$\int_0^1 f(x) dx \approx \frac{2}{3}f(0) + \frac{1}{3}f(1) + \frac{1}{6}f'(0).$$

**解.** 首先求出代数精度, 然后构造插值多项式  $p(x)$ , 满足

$$p(0) = f(0), \quad p(1) = f(1), \quad p'(0) = f'(0),$$

并写出插值余项  $R(x)$ . 代入求积公式后, 利用积分中值定理给出求积公式的余项表达式. 具体过程请读者自行完成.  $\square$

### 9.3 复合求积公式

与分段插值的想法类似, 为了提高计算精度, 我们也可以将积分区间分割成若干小区间, 然后再在每个小区间使用低次求积公式, 这就是 **复合求积公式**, 也称 **复化求积公式**.

为了简单起见, 我们通常等分积分区间. 本节主要介绍两类常用的复合求积公式: 复合梯形公式和复合 Simpson 公式.

#### 9.3.1 复合梯形公式

将  $[a, b]$  划分为  $n$  等份, 即取节点

$$x_k = a + kh, \quad h = \frac{b-a}{n}, \quad k = 0, 1, 2, \dots, n.$$

在每个小区间  $[x_k, x_{k+1}]$  上采用梯形公式, 可得

$$\int_{x_k}^{x_{k+1}} f(x) dx \approx \frac{h}{2} (f(x_k) + f(x_{k+1})).$$

所以

$$\begin{aligned} \int_a^b f(x) dx &= \sum_{k=0}^{n-1} \int_{x_k}^{x_{k+1}} f(x) dx \approx \sum_{k=0}^{n-1} \frac{h}{2} (f(x_k) + f(x_{k+1})) \\ &= h \left( \frac{f(a) + f(b)}{2} + \sum_{k=1}^{n-1} f(x_k) \right). \end{aligned}$$

这就是 **复合梯形公式** (Composite Trapezoidal rule), 通常记为  $T_n$ , 即

$$T_n = h \left( \frac{f(a) + f(b)}{2} + \sum_{k=1}^{n-1} f(x_k) \right).$$

设  $f(x) \in C^2[a, b]$ , 则在每个小区间  $[x_k, x_{k+1}]$  上的余项为  $-\frac{h^3}{12} f''(\eta_k)$ , 所以整体余项为

$$R_n[f] = \int_a^b f(x) dx - T_n = -\frac{h^3}{12} \sum_{k=0}^{n-1} f''(\eta_k).$$

由于  $f''(x)$  在  $[a, b]$  上连续, 且

$$\min_{a \leq x \leq b} f''(x) \leq \frac{1}{n} \sum_{k=0}^{n-1} f''(\eta_k) \leq \max_{a \leq x \leq b} f''(x),$$

所以由介值定理可知, 存在  $\eta \in (a, b)$ , 使得  $f''(\eta) = \frac{1}{n} \sum_{k=0}^{n-1} f''(\eta_k)$ . 故

$$R_n[f] = -\frac{nh^3}{12} f''(\eta) = -\frac{b-a}{12} h^2 f''(\eta), \quad \eta \in (a, b).$$

由此可知, 当  $n \rightarrow \infty$  时,  $R_n[f] \rightarrow 0$ , 所以复合梯形公式是收敛性的. 易知公式中的求积系数都是正的, 因此复合梯形公式是稳定的.



复合梯形公式看似精度不高,但如果被积函数是以  $b-a$  为周期的周期函数,则其具有  $n$  阶三角多项式精度:

$$\int_a^b f(x) dx - h \sum_{k=0}^{n-1} f(a+kh) = \begin{cases} -(b-a), & \text{若 } m \neq 0 \text{ 被 } n \text{ 整除,} \\ 0, & \text{其他,} \end{cases}$$

其中  $f(x) = \exp\left(\frac{2\pi i m x}{b-a}\right)$ ,  $i$  表示虚部单位,  $h = \frac{b-a}{n}$ .

### 9.3.2 复合 Simpson 公式

相类似地,我们可以得到**复合 Simpson 公式** (Composite Simpson's Rule), 通常记为  $S_n$ :

$$\begin{aligned} \int_a^b f(x) dx &\approx \frac{h}{6} \sum_{k=0}^{n-1} \left( f(x_k) + 4f(x_{k+\frac{1}{2}}) + f(x_{k+1}) \right) \\ &= \frac{h}{6} \left( f(a) + f(b) + 2 \sum_{k=1}^{n-1} f(x_k) + 4 \sum_{k=0}^{n-1} f(x_{k+\frac{1}{2}}) \right). \end{aligned}$$

设  $f(x) \in C^4[a, b]$ , 则复合 Simpson 公式的余项为

$$R_n[f] = \int_a^b f(x) dx - S_n = -\frac{b-a}{2880} h^4 f^{(4)}(\eta).$$

易知, 复合 Simpson 公式是收敛的, 也是稳定的.

**例 9.5** 已知  $f(x) = \frac{\sin x}{x}$  的取值如下表, 试分别用复合梯形公式和复合 Simpson 公式计算  $\int_0^1 f(x) dx$  的近似值, 并估计误差. (Quad\_Trap\_Simpson.m)

$x$	0	1/8	2/8	3/8	4/8	5/8	6/8	7/8	1
$f(x)$	1.0000	0.9974	0.9896	0.9767	0.9589	0.9362	0.9089	0.8772	0.8415

(板书)

## 9.4 带导数的求积公式

如果知道被积函数在积分区间的两个端点处的导数值, 则可将它们用来提高求积公式的精度.

### 9.4.1 带导数的梯形公式

带余项的复合梯形求积公式为

$$\int_a^b f(x) dx = T_n - \frac{h^3}{12} \sum_{k=0}^{n-1} f''(\eta_k),$$

其中  $\eta_k \in [x_k, x_{k+1}]$ . 根据定积分的定义, 当  $h$  充分小时, 有

$$h \sum_{k=0}^{n-1} f''(\eta_k) \approx \int_a^b f''(x) dx = f'(b) - f'(a).$$

于是, 我们可以得到下面的求积公式

$$\int_a^b f(x) dx \approx T_n - \frac{h^2}{12} (f'(b) - f'(a)).$$

这就是带端点导数值的复合梯形公式.

当  $n = 1$  时,

$$\int_a^b f(x) dx \approx \frac{b-a}{2} (f(a) + f(b)) - \frac{(b-a)^2}{12} (f'(b) - f'(a)). \quad (9.18)$$

这就是**修正的梯形公式** (Corrected Trapezoidal Rule). 可以验证, 求积公式 (9.18) 的余项为

$$R[f] = \frac{(b-a)^5}{720} f^{(4)}(\eta), \quad \eta \in (a, b).$$

### 9.4.2 带导数的 Simpson 公式

相类似地, 我们可以给出带端点导数的复合 Simpson 公式

$$\int_a^b f(x) dx \approx S_n - \frac{h^4}{2880} (f'''(b) - f'''(a)).$$

当  $n = 1$  时,

$$\int_a^b f(x) dx \approx \frac{b-a}{6} \left( f(a) + 4f\left(\frac{a+b}{2}\right) + f(b) \right) - \frac{h^4}{2880} (f'''(b) - f'''(a)). \quad (9.19)$$

这就是**修正的 Simpson 公式** (Corrected Simpson's Rule).

#### 提高算法精度的一种重要手段

由上面的推导过程可知, 两类方法都是通过将余项中的部分加入到求积公式中, 从而达到进一步降低误差的目的. 这也是提高算法精度的一种重要手段.





## 9.5 Romberg 求积公式

### 9.5.1 外推技巧

在使用复合求积公式时, 由于一开始并不清楚  $n$  该取多大. 如果  $n$  太小的话就达不到计算精度要求, 反之, 如果  $n$  太大, 则会增加额外的工作量. 因此我们可以采用动态的方法, 即将区间不断对分, 直到所得到的计算结果满足指定的精度为止. 下面以梯形公式为例来说明这个递推过程.

将积分区间  $n$  等分, 可得复合求积公式

$$T_n = \frac{h}{2} \sum_{i=0}^{n-1} (f(x_i) + f(x_{i+1})), \quad h = \frac{b-a}{n}.$$

如果再将每个小区间  $[x_i, x_{i+1}]$  二等分, 则新的复合梯形公式为

$$T_{2n} = \frac{h}{4} \sum_{i=0}^{n-1} (f(x_i) + 2f(x_{i+\frac{1}{2}}) + f(x_{i+1})) = \frac{1}{2}T_n + \frac{h}{2} \sum_{i=0}^{n-1} f\left(x_i + \frac{1}{2}h\right). \quad (9.20)$$

这就是复合梯形法的递推公式.

下面我们给出具体的计算过程.

- (1) 记  $h^{(0)} = b - a$ , 计算: (梯形公式)

$$T^{(0)} = \frac{h^{(0)}}{2} (f(x_0) + f(x_1)), \quad x_i = a + ih^{(0)}, i = 0, 1.$$

- (2) 将积分区间二等分, 记  $h^{(1)} = \frac{b-a}{2} = \frac{1}{2}h^{(0)}$ , 计算: (复合梯形公式)

$$T^{(1)} = \frac{h^{(1)}}{2} \sum_{i=0}^{2-1} (f(x_i) + f(x_{i+1})) = \frac{1}{2}T^{(0)} + h^{(1)} \sum_{i=0}^{2^0-1} f(x_{2i+1}),$$

其中  $x_i = a + ih^{(1)}, i = 0, 1, 2$ .

- (3) 再将每个小区间二等分, 记  $h^{(2)} = \frac{b-a}{2^2} = \frac{1}{2}h^{(1)}$ , 计算: (复合梯形公式)

$$T^{(2)} = \frac{h^{(2)}}{2} \sum_{i=0}^{2^2-1} (f(x_i) + f(x_{i+1})) = \frac{1}{2}T^{(1)} + h^{(2)} \sum_{i=0}^{2^1-1} f(x_{2i+1}),$$

其中  $x_i = a + ih^{(2)}, i = 0, 1, \dots, 2^2$ .

- (4) 再将每个小区间二等分, 记  $h^{(3)} = \frac{b-a}{2^3} = \frac{1}{2}h^{(2)}$ , 计算: (复合梯形公式)

$$T^{(3)} = \frac{h^{(3)}}{2} \sum_{i=0}^{2^3-1} (f(x_i) + f(x_{i+1})) = \frac{1}{2}T^{(2)} + h^{(3)} \sum_{i=0}^{2^2-1} f(x_{2i+1}),$$

其中  $x_i = a + ih^{(3)}, i = 0, 1, \dots, 2^3$ .

- (5) 依此类推, 对于  $k = 4, 5, 6, \dots$ , 记  $h^{(k)} = \frac{b-a}{2^k} = \frac{1}{2}h^{(k-1)}$ , 计算: (复合梯形公式)

$$T^{(k)} = \frac{h^{(k)}}{2} \sum_{i=0}^{2^k-1} (f(x_i) + f(x_{i+1})) = \frac{1}{2}T^{(k-1)} + h^{(k)} \sum_{i=0}^{2^{k-1}-1} f(x_{2i+1}),$$

其中  $x_i = a + ih^{(k)}, i = 0, 1, \dots, 2^k$ .



**例 9.6** 用梯形法的递推公式计算定积分  $\int_0^1 \frac{\sin x}{x} dx$ , 要求计算精度满足  $|T_{2n} - T_n| < \varepsilon = 10^{-7}$ .

(Quad\_Trap\_recursion.m)

## 9.5.2 Romberg 算法

梯形递推公式算法简单, 易于计算机实现, 但收敛速度较慢. 下面我们介绍一个加速技巧, 即 Romberg 算法, 可以大大地提升收敛速度.

易知  $T_n$  的值与步长  $h$  有关, 为了讨论方便, 我们记  $T_n \triangleq T(h)$ . 由复合梯形公式余项可知

$$I(f) \triangleq \int_a^b f(x) dx = T(h) - \frac{b-a}{12} h^2 f''(\eta).$$

**定理 9.13** 设  $f(x) \in C^\infty[a, b]$ , 则有

$$T(h) = I(f) + \alpha_1 h^2 + \alpha_2 h^4 + \alpha_3 h^6 + \cdots$$

其中  $\alpha_k$  与  $f(x)$  有关, 但与  $h$  无关.

(感兴趣的同学可以参考相关资料)

由定理 9.13 可知  $T(h) - I(f) = O(h^2)$ , 即复合梯形公式的误差阶为  $O(h^2)$ . 由于系数  $\alpha_k$  与  $h$  无关, 将积分区间再次二等分后可得

$$T\left(\frac{h}{2}\right) = I(f) + \alpha_1 \frac{h^2}{4} + \alpha_2 \frac{h^4}{16} + \alpha_3 \frac{h^6}{32} + \cdots$$

因此

$$S(h) \triangleq \frac{4T\left(\frac{h}{2}\right) - T(h)}{3} = I(f) + \beta_1 h^4 + \beta_2 h^6 + \cdots \quad (9.21)$$

如果用  $S(h)$  来近似  $I(f)$ , 则误差阶提高到了  $O(h^4)$ . 事实上,  $S(h)$  就是复合 Simpson 公式. 上面的这种通过线性组合提高误差阶的方法就是 **外推算法**, 或 **Richardson 外推**, 这是一个提高计算精度的非常重要的技巧.

类似地, 我们有

$$C(h) \triangleq \frac{16S\left(\frac{h}{2}\right) - S(h)}{15} = I(f) + \gamma_1 h^6 + \gamma_2 h^8 + \cdots \quad (9.22)$$

这时, 误差阶提高到了  $O(h^6)$ . 事实上,  $C(h)$  就是复合 Cotes 公式.

这样不断利用外推技巧, 我们就可以不断提高计算精度. 这个外推过程就是 **Romberg 算法**.

## Romberg 算法计算过程

Romberg 算法的计算过程如下.

### 算法 9.1. Romberg Algorithm

- 1: 令  $k = 0, h = b - a$
- 2: 计算  $T_0^{(0)} = \frac{h}{2}(f(a) + f(b))$
- 3: 令  $k = 1$



- 4: 利用梯形法的递推公式 (9.20) 计算  $T_0^{(k)} = T\left(\frac{h}{2^k}\right)$
- 5: **for**  $i = 1, 2, \dots, k$  **do**
- 6:     计算  $T_i^{(k-i)} = \frac{4^i T_{i-1}^{(k-i+1)} - T_{i-1}^{(k-i)}}{4^i - 1}$
- 7: **end for**
- 8: 若  $|T_k^{(0)} - T_{k-1}^{(0)}| < \varepsilon$ , 则终止计算, 取  $T_k^{(0)}$  为定积分近似值.
- 9: 令  $k = k + 1$ , 转到第 4 步

Romberg 算法的计算过程也可以用下面的表格来描述.

$k$	$h^{(k)}$	$T_0^{(k)}$	$T_1^{(k)}$	$T_2^{(k)}$	$T_3^{(k)}$	$T_4^{(k)}$	$\dots$
0	$b - a$	$T_0^{(0)}$					
1	$\frac{b-a}{2}$	$T_0^{(1)}$	$T_1^{(0)}$				
2	$\frac{b-a}{2^2}$	$T_0^{(2)}$	$T_1^{(1)}$	$T_2^{(0)}$			
3	$\frac{b-a}{2^3}$	$T_0^{(3)}$	$T_1^{(2)}$	$T_2^{(1)}$	$T_3^{(0)}$		
4	$\frac{b-a}{2^4}$	$T_0^{(4)}$	$T_1^{(3)}$	$T_2^{(2)}$	$T_3^{(1)}$	$T_4^{(0)}$	
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

**例 9.7** 用 Romberg 算法计算定积分  $\int_0^1 \frac{\sin x}{x} dx$ , 要求计算精度满足  $|T_{2n} - T_n| < \varepsilon = 10^{-7}$ .

(Quad\_Romberg.m)

## 9.6 自适应求积方法 \*

见课堂板书 (*To be continued ...*)

## 9.7 Gauss 求积公式

### 为什么 Gauss 求积

在 Newton-Cotes 公式中, 我们选取的是等距节点, 这样做的好处就是计算方便. 但等距节点不一定是最好的选择, 事实上, 我们可以更好地选取节点, 使得求积公式具有更高的代数精度.

**例 9.8** 试确定  $A_i$  和  $x_i$ , 使得下面的求积公式具有尽可能高的代数精度, 并求出该求积公式的代数精度.

$$\int_{-1}^1 f(x) dx \approx A_0 f(x_0) + A_1 f(x_1). \quad (9.23)$$

(板书)


由此可见, 采用不等距节点的两点求积公式 (9.23) 比采用等距节点的求积公式 (即梯形公式) 具有更高的代数精度.

### 9.7.1 一般 Gauss 求积公式


**定义 9.5** 设  $\rho(x)$  是  $[a, b]$  上的权函数, 若求积公式

$$\int_a^b \rho(x) f(x) dx \approx \sum_{i=0}^n A_i f(x_i), \quad (9.24)$$

具有  $2n+1$  次代数精度, 则称该公式为 **Gauss 求积公式**, 节点  $x_i$  称为 **Gauss 点**,  $A_i$  称为 **Gauss 系数**.

 需要指出的是, 求积公式 (9.24) 的右端只包含  $f(x)$  的函数值, 与权函数无关.

求积公式 (9.24) 中含有  $2n+2$  的待定参数, 即  $A_i$  和  $x_i, i=0, 1, 2, \dots, n$ . 我们可以将  $f(x) = 1, x, x^2, \dots, x^{2n+1}$  代入, 并令求积公式 (9.24) 精确成立, 然后解出  $A_i$  和  $x_i$ . 这样就可以使得求积公式至少具有  $2n+1$  次代数精度, 所以, Gauss 求积公式总是存在的.

 事实上, 求积公式 (9.24) 的代数精度不可能超过  $2n+1$ . 取  $2n+2$  次多项式  $f(x) = (x - x_0)(x - x_1)^2 \cdots (x - x_n)^2$ , 则  $\sum_{i=0}^n A_i f(x_i) = 0$ , 但显然

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

即求积公式 (9.24) 对  $2n+2$  次多项式  $f(x)$  不精确成立, 所以它的代数精度小于  $2n+2$ .

**定理 9.14** Gauss 求积公式是具有最高代数精度的插值型求积公式.

我们所关心的是如何构造 Gauss 求积公式. 从例 9.8 可以看出, 我们可以将  $f(x) = 1, x, x^2, \dots, x^{2n+1}$  代入, 并令求积公式 (9.24) 精确成立, 这样就能解出  $A_i$  和  $x_i$ . 但这时需要解一个非线性方程组, 而一般情况下, 求解非线性方程组是非常困难的. 因此, 当  $n > 2$  时, 这种方法是不可行的.



一个比较可行的方法是将  $x_i$  和  $A_i$  分开计算, 即先通过特殊的方法求出 Gauss 点  $x_i$ , 然后再用待定系数法解出  $A_i$ . 这也是目前构造 Gauss 公式的通用方法. 下面我们就介绍如何计算 Gauss 点.

### Gauss 点的计算

**定理 9.15** 设节点  $a \leq x_0 < x_1 < \cdots < x_n \leq b$ , 则插值型求积公式 (9.24) 是 Gauss 公式的充要条件是多项式

$$\omega_{n+1}(x) = (x - x_0)(x - x_1) \cdots (x - x_n)$$

与所有次数不超过  $n$  的多项式正交, 即

$$\int_a^b \rho(x) \omega_{n+1}(x) p(x) dx = 0, \quad \forall p(x) \in H_n.$$

(板书)

### 计算 Gauss 点的一般方法

- (1) 设  $\omega_{n+1}(x) = x^{n+1} + a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0$ ;
- (2) 利用  $\omega_{n+1}(x)$  与  $p(x) = 1, x, x^2, \dots, x^n$  正交 (带权) 的性质, 得到  $n+1$  个线性方程, 解出  $a_k, k = 0, 1, 2, \dots, n$ , 这样就能确定多项式  $\omega_{n+1}(x)$ ;
- (3) 求出多项式  $\omega_{n+1}(x)$  的  $n+1$  个零点, 这就是 Gauss 点.

**例 9.9** 试确定  $A_i$  和  $x_i$ , 使得下面的求积公式具有尽可能高的代数精度

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

(板书)

### Gauss 求积公式的余项

设  $p_{2n+1}(x)$  是  $f(x)$  关于点  $x_0, x_1, x_2, \dots, x_n$  的  $2n+1$  次 Hermite 插值多项式, 满足

$$p_{2n+1}(x_i) = f(x_i), \quad p'_{2n+1}(x_i) = f'(x_i), \quad i = 0, 1, 2, \dots, n.$$

若  $f(x) \in C^{2n+2}[a, b]$ , 则可以验证, 插值余项为

$$R_n(x) \triangleq f(x) - p_{2n+1}(x) = \frac{f^{(2n+2)}(\xi_x)}{(2n+2)!} \omega_{n+1}^2.$$

由于求积公式 (9.24) 具有  $2n+1$  次代数精度, 故

$$\int_a^b \rho(x) p_{2n+1}(x) dx = \sum_{i=0}^n A_i p_{2n+1}(x_i).$$

所以, Gauss 求积公式的余项为

$$\begin{aligned} R_n[f] &\triangleq \int_a^b \rho(x) f(x) dx - \sum_{i=0}^n A_i f(x_i) \\ &= \int_a^b \rho(x) f(x) dx - \sum_{i=0}^n A_i p_{2n+1}(x_i) \end{aligned}$$



$$\begin{aligned}
 &= \int_a^b \rho(x) (f(x) - p_{2n+1}(x)) dx \\
 &= \int_a^b \rho(x) \frac{f^{(2n+2)}(\xi_x)}{(2n+2)!} \omega_{n+1}^2 dx.
 \end{aligned}$$

假定  $f^{(2n+2)}(\xi_x)$  在  $[a, b]$  上关于  $x$  是连续的, 则由积分中值定理可知, 存在  $\eta \in (a, b)$ , 使得

$$R_n[f] = \frac{f^{(2n+2)}(\eta)}{(2n+2)!} \int_a^b \rho(x) \omega_{n+1}^2 dx. \quad (9.25)$$

### Gauss 公式的收敛性与稳定性

**定理 9.16** 设  $f(x) \in C[a, b]$ , 则 Gauss 求积公式是收敛的, 即

$$\lim_{n \rightarrow \infty} \sum_{i=0}^n A_i f(x_i) = \int_a^b \rho(x) f(x) dx.$$

(证明可参见相关参考文献)

**定理 9.17** Gauss 求积公式中的系数  $A_i$  全是正数, 因此 Gauss 求积公式是稳定的.

**证明.** 令  $f(x) = l_k^2(x) = \left( \prod_{i=0, i \neq k}^n \frac{x - x_i}{x_k - x_i} \right)^2 \in H_{2n}$ . 由于 Gauss 公式具有  $2n+1$  次代数精度, 所以

$$\int_a^b \rho(x) l_k^2(x) dx = \sum_{i=0}^n A_i l_k^2(x_i) = A_k.$$

上式左边显然大于 0, 故  $A_k > 0$ , 所以结论成立.  $\square$

### 9.7.2 Gauss-Legendre 公式

根据定理 9.15, 如果  $\{p_n(x)\}_{n=0}^\infty$  是一组在  $[a, b]$  上带权  $\rho(x)$  正交的多项式族, 则  $p_{n+1}(x)$  的零点就是 Gauss 点. 因此, 我们可以使用已知的正交多项式, 如 Legendre 多项式和 Chebyshev 多项式等.

设  $[a, b] = [-1, 1]$ , 权函数  $\rho(x) = 1$ , 则 Gauss 点即为 Legendre 多项式  $P_{n+1}(x)$  的零点, 此时的 Gauss 公式就称为 **Gauss-Legendre 公式**, 简称 **G-L 公式**.

下面介绍几个低次 G-L 公式:

- 当  $n = 0$  时,  $P_{n+1}(x) = x$ , 因此 Gauss 点为  $x_0 = 0$ . 将  $f(x) = 1$  代入公式, 令等式精确成立, 即可解出  $A_0 = 2$ . 所以 G-L 公式为

$$\int_{-1}^1 f(x) dx \approx 2f(0).$$

- 当  $n = 1$  时,  $P_{n+1}(x) = \frac{1}{2}(3x^2 - 1)$ , 因此 Gauss 点为  $x_0 = -\frac{\sqrt{3}}{3}$ ,  $x_1 = \frac{\sqrt{3}}{3}$ . 将  $f(x) = 1, x$



代入公式, 令等式精确成立, 即可解出  $A_0 = 1, A_1 = 1$ . 所以 G-L 公式为

$$\int_{-1}^1 f(x) dx \approx f\left(-\frac{\sqrt{3}}{3}\right) + f\left(\frac{\sqrt{3}}{3}\right) \approx f(-0.5774) + f(0.5774).$$

- 当  $n = 2$  时, 类似地, 可以得到 G-L 公式为

$$\begin{aligned} \int_{-1}^1 f(x) dx &\approx \frac{5}{9} f\left(-\frac{\sqrt{15}}{5}\right) + \frac{8}{9} f(0) + \frac{5}{9} f\left(\frac{\sqrt{15}}{5}\right) \\ &\approx 0.5556 f(-0.7746) + 0.8889 f(0) + 0.5556 f(0.7746). \end{aligned}$$

- 当  $n = 3$  时, 可得 G-L 求积公式为

$$\begin{aligned} \int_{-1}^1 f(x) dx &\approx \frac{90 - 5\sqrt{30}}{180} f\left(-\sqrt{\frac{15 + 2\sqrt{30}}{35}}\right) + \frac{90 + 5\sqrt{30}}{180} f\left(-\sqrt{\frac{15 - 2\sqrt{30}}{35}}\right) \\ &\quad + \frac{90 + 5\sqrt{30}}{180} f\left(\sqrt{\frac{15 - 2\sqrt{30}}{35}}\right) + \frac{90 - 5\sqrt{30}}{180} f\left(\sqrt{\frac{15 + 2\sqrt{30}}{35}}\right) \end{aligned}$$

当  $n \geq 4$  时, 我们可以使用数值方法计算  $P_{n+1}(x)$  的零点.

**例 9.10** 用三点 G-L 公式 ( $n = 2$ ) 计算定积分  $\int_0^{\frac{\pi}{2}} x^2 \cos x dx$ .

(板书)

### G-L 公式的余项

由于此时  $\omega_{n+1}(x) = \tilde{P}_{n+1}(x)$ . 由余项公式 (9.25) 和 Legendre 多项式性质 (8.3) 可知, G-L 公式的余项为

$$R_n[f] = \frac{f^{(2n+2)}(\eta)}{(2n+2)!} \int_{-1}^1 \tilde{P}_{n+1}^2 dx = \frac{2^{2n+3}[(n+1)!]^4}{(2n+3)[(2n+2)!]^3} f^{(2n+2)}(\eta), \quad \eta \in (-1, 1).$$

这表明 G-L 公式具有很高的精度, 比如

$$R_1[f] = \frac{1}{135} f^{(4)}(\eta), \quad R_2[f] = \frac{1}{15750} f^{(6)}(\eta).$$

### 一般区间上的 Gauss-Legendre 公式

当积分区间是  $[a, b]$  时, 我们可以做一个变量代换

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

于是可得

$$\int_a^b f(x) dx = \frac{b-a}{2} \int_{-1}^1 f(x(t)) dt = \frac{b-a}{2} \int_{-1}^1 f\left(\frac{b-a}{2}t + \frac{b+a}{2}\right) dt,$$

然后对上式右端的定积分使用 G-L 公式即可.

### 9.7.3 Gauss-Chebyshev 公式

设  $[a, b] = [-1, 1]$ , 权函数  $\rho(x) = \frac{1}{\sqrt{1-x^2}}$ , 则 Gauss 点即为 Chebyshev 多项式  $T_{n+1}(x)$  的零点, 此时的 Gauss 公式就称为 **Gauss-Chebyshev 公式**, 简称 **G-C 公式**.

易知  $T_{n+1}(x)$  的零点为

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

利用待定系数法可以求得

$$A_i = \frac{\pi}{n+1}.$$

所以 G-C 公式为

$$\int_a^b \frac{1}{\sqrt{1-x^2}} f(x) dx \approx \frac{\pi}{n+1} \sum_{i=0}^n f\left(\cos \frac{2(n+1)}{2i+1}\pi\right). \quad (9.26)$$

由公式 (9.25) 可知, G-C 公式的余项为

$$R[f] = \frac{f^{(2n+2)}(\eta)}{(2n+2)!} \int_a^b \frac{1}{\sqrt{1-x^2}} \tilde{T}_{n+1}^2 dx = \frac{2\pi}{2^{2n+2}(2n+2)!} f^{(2n+2)}(\eta), \quad \eta \in (-1, 1).$$

**例 9.11** 用五点 G-C 公式 ( $n=4$ ) 计算奇异积分  $\int_{-1}^1 \frac{e^x}{\sqrt{1-x^2}} dx$ .

(板书)

### 9.7.4 复合 Gauss 公式

Gauss 公式最突出的优点就是具有最高的代数精度. 但缺点是 Gauss 点比较难计算, 而且当节点增加时, 需要重新计算 Gauss 点. 因此我们在实际应用中, 通常是将积分区间分割成若干小区间, 然后在每个小区间上使用低次的 Gauss 公式, 这就是 **复合 Gauss 公式**.





## 9.8 多重积分

计算多重积分的基本思想是化为累次积分, 然后逐个进行数值积分.

对于二重积分, 如果积分区域  $\Omega$  为矩形区域, 则

$$\iint_{\Omega} f(x, y) \, dx \, dy = \int_a^b \left( \int_c^d f(x, y) \, dy \right) \, dx.$$

如果积分区域  $\Omega$  是  $x$  型区域, 则


$$\iint_{\Omega} f(x, y) \, dx \, dy = \int_a^b \left( \int_{y_1(x)}^{y_2(x)} f(x, y) \, dy \right) \, dx.$$

如果积分区域  $\Omega$  是  $y$  型区域, 则

$$\iint_{\Omega} f(x, y) \, dx \, dy = \int_c^d \left( \int_{x_1(y)}^{x_2(y)} f(x, y) \, dx \right) \, dy.$$

**例 9.12** 利用两点 Gauss 公式计算二重积分  $\int_{\Omega} (x^2 + 2y^2) \, dx$ , 其中  $\Omega = [-1, 1] \times [-1, 1]$ .

(留作课外练习)

 为了提高计算精度, 通常也使用复合求积公式.

## 9.9 数值微分

基本想法与数值积分类似, 即用函数值的线性组合来近似某点的导数值.

### 9.9.1 插值型求导公式

设  $p_n(x)$  是  $f(x)$  基于节点  $x_0, x_1, x_2, \dots, x_n$  的插值多项式, 则可以用  $p_n(x)$  的导数来近似  $f(x)$  的导数, 即


$$f'(x) \approx p'_n(x).$$

这就是**插值型求导公式**. 由插值余项公式可知

$$\begin{aligned} f'(x) - p'_n(x) &= \frac{d}{dx} \left( \frac{f^{(n+1)}(\xi_x)}{(n+1)!} \omega_{n+1}(x) \right) \\ &= \frac{f^{(n+1)}(\xi_x)}{(n+1)!} \omega'_{n+1}(x) + \omega_{n+1}(x) \frac{d}{dx} \left( \frac{f^{(n+1)}(\xi_x)}{(n+1)!} \right). \end{aligned}$$

这里假定  $f^{(n+1)}(\xi_x)$  关于  $x$  可导. 由于  $\xi_x$  是关于  $x$  的未知函数, 因此右端第二项是不可求的. 但当  $x$  是节点时, 即  $x = x_i$ , 有

$$f'(x_i) - p'_n(x_i) = \frac{f^{(n+1)}(\xi_x)}{(n+1)!} \omega'_{n+1}(x_i) = \frac{f^{(n+1)}(\xi_x)}{(n+1)!} \prod_{k=0, k \neq i}^n (x_i - x_k). \quad (9.27)$$

 一般情况下, 我们只考虑函数在节点处的导数值. 其他点的导数可以通过插值等手段获得.

### 9.9.2 一阶导数的差分近似

#### 两点公式

设节点  $x_0, x_1$ , 记  $h = x_1 - x_0$ , 则可得一次插值多项式

$$p_1(x) = \frac{x - x_1}{x_0 - x_1} f(x_0) + \frac{x - x_0}{x_1 - x_0} f(x_1).$$

因此

$$\begin{aligned} f'(x_0) &= p'_1(x_0) + \frac{f''(\xi_0)}{2} (x_0 - x_1) = \frac{1}{h} (f(x_1) - f(x_0)) - \frac{h}{2} f''(\xi_0), \\ f'(x_1) &= p'_1(x_1) + \frac{f''(\xi_1)}{2} (x_1 - x_0) = \frac{1}{h} (f(x_1) - f(x_0)) + \frac{h}{2} f''(\xi_1). \end{aligned}$$

#### 三点公式

考虑等距节点  $x_0, x_1 = x_0 + h, x_2 = x_0 + 2h$ , 对应的二次 Lagrange 插值多项式为

$$p_2(x) = \frac{(x - x_1)(x - x_2)}{(x_0 - x_1)(x_0 - x_2)} f(x_0) + \frac{(x - x_0)(x - x_2)}{(x_1 - x_0)(x_1 - x_2)} f(x_1) + \frac{(x - x_0)(x - x_1)}{(x_2 - x_0)(x_2 - x_1)} f(x_2).$$

做变量代换:  $x = x_0 + th$ , 则可得

$$p_2(t) = \frac{1}{2} (t-1)(t-2) f(x_0) + t(t-2) f(x_1) + \frac{1}{2} t(t-1) f(x_2).$$



于是

$$\frac{dp_2}{dt} = \frac{1}{2}((2t-3)f(x_0) - 4(t-1)f(x_1) + (2t-1)f(x_2)).$$

所以

$$\frac{dp_2}{dx} = \frac{dp_2}{dt} \bigg/ \frac{dx}{dt} = \frac{1}{2h}((2t-3)f(x_0) - 4(t-1)f(x_1) + (2t-1)f(x_2)).$$

分别令  $x = x_0, x_1, x_2$ , 即  $t = 0, 1, 2$ , 可得

$$p_2'(x_0) = \frac{1}{2h}(-3f(x_0) + 4f(x_1) - f(x_2)),$$

$$p_2'(x_1) = \frac{1}{2h}(f(x_2) - f(x_0)),$$

$$p_2'(x_2) = \frac{1}{2h}(f(x_0) - 4f(x_1) + 3f(x_2)).$$

由公式 (9.27) 可知

$$\begin{aligned} f'(x_0) &= \frac{1}{2h}(-3f(x_0) + 4f(x_1) - f(x_2)) + \frac{h^2}{3}f^{(3)}(\xi_0), \\ f'(x_1) &= \frac{1}{2h}(f(x_2) - f(x_0)) - \frac{h^2}{6}f^{(3)}(\xi_1), \\ f'(x_2) &= \frac{1}{2h}(f(x_0) - 4f(x_1) + 3f(x_2)) + \frac{h^2}{3}f^{(3)}(\xi_2). \end{aligned} \quad (9.28)$$

这就是三点求导公式, 特别公式 (9.28), 是常用的计算一阶导数的 **中心差分公式**.

### 9.9.3 二阶导数的差分近似


计算  $p_2(x)$  关于  $x$  的二阶导数可得

$$\frac{d^2p_2}{dx^2} = \frac{1}{h^2}(f(x_0) - 2f(x_1) + f(x_2)).$$

结合公式 (9.27) 可知

$$f''(x_1) = \frac{1}{h^2}(f(x_0) - 2f(x_1) + f(x_2)) - \frac{h^2}{12}f^{(4)}(\xi).$$

这就是计算二阶导数常用的 **中心差分公式**.

 事实上, 以上计算一阶导数和二阶导数的中心差分公式都可以从 Taylor 展开式得到.

### 9.9.4 三次样条求导

见课堂板书 (*To be continued ...*)

### 9.9.5 数值微分的外推算法

见课堂板书 (*To be continued ...*)

## 9.10 课后练习

**练习 9.1** 确定下列求积公式中的待定参数, 使其具有尽可能高的代数精度, 并指出所构造的求积公式的代数精度.

$$(1) \int_{-1}^1 f(x) dx \approx A_{-1}f(-1) + A_0f(0) + A_1f(1)$$

$$(2) \int_{-2}^2 f(x) dx \approx A_{-1}f(-1) + A_0f(0) + A_1f(1)$$

$$(3) \int_{-1}^1 f(x) dx \approx \frac{1}{3}(f(-1) + 2f(x_1) + 3f(x_2))$$

$$(4) \int_0^h f(x) dx \approx \frac{h}{2}(f(0) + f(h)) + ah^2(f'(0) - f'(h))$$

**练习 9.2** 分别用复合梯形法和复合 Simpson 方法计算下列定积分:

$$(1) \int_0^2 \frac{x}{4+x^2} dx, \quad n=8$$

$$(2) \int_1^7 \sqrt{x} dx, \quad n=4$$

**练习 9.3** 用 Simpson 公式计算定积分  $\int_1^2 e^{-x} dx$ , 并估计误差.

(提示: 用余项公式估计误差)

**练习 9.4** 推导下列求积公式的余项公式:

$$(1) \int_a^b f(x) dx \approx (b-a)f(a)$$

$$(2) \int_a^b f(x) dx \approx (b-a)f\left(\frac{a+b}{2}\right)$$

(提示: 用标准的三步法)

**练习 9.5** 设  $f''(x) > 0, x \in [a, b]$ , 证明用梯形公式计算定积分  $I = \int_a^b f(x) dx$  所得结果比准确值大, 并说明其几何意义.

**练习 9.6** 已知  $\rho(x) = \frac{1}{\sqrt{x}}$  是  $[0, 2]$  上的权函数, 试构造 Gauss 求积公式

$$\int_0^2 \frac{1}{\sqrt{x}} f(x) dx \approx A_0 f(x_0) + A_1 f(x_1).$$

计算过程中保留小数点后两位数字. (提示: 采用标准方法, 即先求 Gauss 点, 然后求 Gauss 系数.)

**练习 9.7** 用  $n=2$  的 Gauss-Legendre 求积公式计算定积分:

$$\int_0^2 e^x \sin(x) dx,$$

计算过程中保留小数点后两位数字.

**练习 9.8** 试确定下面数值微分公式的截断误差表达式

$$f'(x_0) \approx \frac{1}{2h}(3f(x_0+h) - f(x_0) - 2f(x_0+2h)).$$

(提示: 利用插值多项式)



练习 9.9 用三点公式计算  $f(x)$  在  $x = 1.2$  处的一阶导数和二阶导数. 函数值如下:  $f(1.1) = 0.227$ ,  $f(1.2) = 0.207$ ,  $f(1.3) = 0.189$ .

练习 9.10\* 给出复合两点 Gauss-Legendre 求积公式的余项公式.

# 10

## 常微分方程初值问题

微分方程 (包括常微分方程和偏微分方程) 主要用来描述宏观和微观世界的基本运动规律, 在物理、化学、生物、经济等多个领域发挥着广泛的作用.

本讲主要介绍常微分方程初值问题

$$\begin{cases} \frac{dy}{dx} = f(x, y), & x \in [a, b], \\ y(a) = y_0, \end{cases} \quad (10.1a)$$

$$(10.1b)$$

的数值求解方法, 即计算  $y(x)$  在  $x = b$  点的近似值 (或者  $y(x)$  在  $[a, b]$  内任意一点上的近似值).

为了确保解的存在唯一, 我们假定  $f(x, y)$  关于  $y$  满足 **Lipschitz 条件**, 即存在实数  $L > 0$ , 使得

$$|f(x, y_1) - f(x, y_2)| \leq L|y_1 - y_2|. \quad (10.2)$$

这里  $L$  称为 **Lipschitz 常数**.

**定理 10.1 (解的存在唯一性, Picard-Lindelof 定理)** 设  $f(x, y)$  在区域  $D \triangleq \{(x, y) \mid x \in [a, b], y \in \mathbb{R}\}$  上连续, 而且关于  $y$  满足 Lipschitz 条件, 则初值问题 (10.1) 存在唯一的连续可微解.

我们这里主要讨论在解存在唯一的前提下如何通过数值方法进行求解.

### 关于常微分方程数值解的相关文献

- [1] E. Hairer, S. P. Nørsett and G. Wanner, [Solving Ordinary Differential Equations I: Nonstiff Problems](#), 2nd, 2009 [28]
- [2] E. Hairer and G. Wanner, [Solving Ordinary Differential Equations II: Stiff and Differential-Algebraic Problems](#), 2nd, 1996 [27]
- [3] J. C. Butcher, [Numerical Methods for Ordinary Differential Equations](#), 3rd, 2016 [7]
- [4] C. Gear, [Numerical Initial Value Problems in Ordinary Differential Equations](#), 1971 [17]
- [5] D. F. Griffiths and D. J. Higham, [Numerical Methods for Ordinary Differential Equations: Initial Value Problems](#), 2010 [18]

## 10.1 单步法

对初值问题 (10.1a) 的两边求定积分可得

$$\int_a^b \frac{dy}{dx} dx = \int_a^b f(x, y) dx,$$

即

$$y(b) - y(a) = \int_a^b f(x, y) dx.$$

于是可得

$$y(b) = y(a) + \int_a^b f(x, y) dx.$$

对右端的定积分进行数值计算, 就可得到求解初值问题的数值方法. 不同的数值积分方法将导出不同的初值问题求解方法.

### 10.1.1 Euler 法

**Euler 公式**

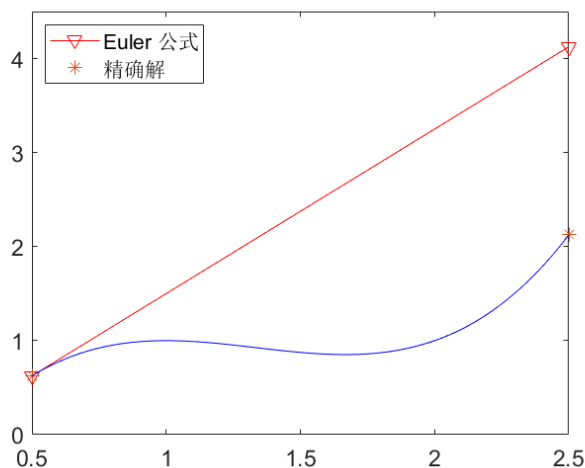
用左点矩形公式近似  $\int_a^b f(x, y) dx$ , 则可得

$$y(b) \approx y(a) + (b - a)f(a, y(a)) = y(a) + (b - a)f(a, y_0) \quad (10.3)$$

这就是 **Euler 公式**.

#### Euler 公式的几何意义

Euler 公式的基本思想也是“以直代曲”, 即用“过初值点  $(a, y_0)$ , 以  $f(a, y_0)$  为斜率的直线  $L_1(x)$ ”来近似曲线  $y(x)$ , 然后用  $L_1(b)$  来近似  $y(b)$ .



**Euler 法**

显然, 在整个区间上用 Euler 公式来计算  $y(b)$  的近似值的话, 误差较大. 于是我们很自然地想到将求解区间分割成若干小区间, 然后在每个小区间上使用 Euler 公式.

将  $[a, b]$  分割成  $n$  个小区间, 即

$$a = x_0 < x_1 < \cdots < x_n = b,$$


在每个小区间  $[x_i, x_{i+1}]$  上使用 Euler 公式可得

$$\begin{aligned} y(x_1) &\approx y_0 + (x_1 - x_0)f(x_0, y_0) \triangleq y_1, \\ y(x_2) &\approx y_1 + (x_2 - x_1)f(x_1, y_1) \triangleq y_2, \\ &\vdots \\ y(x_n) &\approx y_{n-1} + (x_n - x_{n-1})f(x_{n-1}, y_{n-1}) \triangleq y_n. \end{aligned}$$

记  $h_k = x_{k+1} - x_k$ , 则上式可写成一般形式:

$$y_{k+1} = y_k + h_k f(x_k, y_k), \quad k = 0, 1, 2, \dots, n-1. \quad (10.4)$$

这就是 **Euler 方法**, 也称为 **Euler 折线法**, 其中  $h_k$  称为步长.

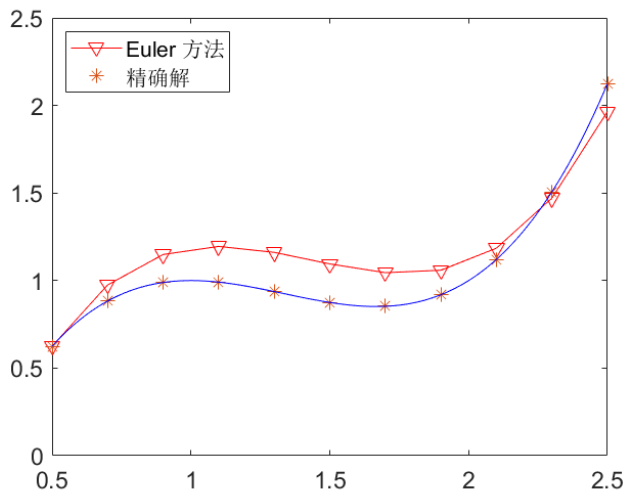
 为了方便计算, 我们通常对求解区间进行等分, 即每个小区间长度  $h_k$  都相等, 此时我们就称该方法为**等步长方法**.

**例 10.1** 用等步长 Euler 法求解初值问题

(ODE\_Euler\_01.m)

$$\begin{cases} \frac{dy}{dx} = f(x, y) = 3x^2 - 8x + 5, & x \in [0.5, 2.5], \\ y(a) = y_0 = 0.625, \end{cases}$$

下图是以  $n = 10$  为例的结果. 可见, 计算效果比在整个区间上直接使用 Euler 公式要好很多.





### 10.1.2 梯形法

为了进一步提高计算精度, 在近似定积分时, 可以用梯形公式替代左矩形公式, 即

$$\begin{aligned} y(x_{k+1}) &= y(x_k) + \int_{x_k}^{x_{k+1}} f(x, y) \, dx \\ &\approx y(x_k) + \frac{x_{k+1} - x_k}{2} [f(x_k, y(x_k)) + f(x_{k+1}, y(x_{k+1}))], \end{aligned}$$

于是可得**梯形法**:

$$y_{k+1} = y_k + \frac{h_k}{2} [f(x_k, y_k) + f(x_{k+1}, y_{k+1})]. \quad (10.5)$$

由于梯形公式 (10.5) 的右端项  $f(x_{k+1}, y_{k+1})$  含有  $y_{k+1}$ , 因此每次计算  $y_{k+1}$  时需要求解一个方程, 而且往往是非线性的, 这很可能会给求解带来很大的困难.

#### 显式方法与隐式方法

如果每个迭代步不需要解方程, 则称为**显式方法**, 反之, 如果每个迭代步需要解方程, 则称为**隐式方法**. Euler 法是显式的, 而梯形法是隐式的.

### 10.1.3 改进的 Euler 法

为了避免解方程, 每次使用梯形公式进行数值求解时, 可以先用 Euler 法得到一个近似的  $y_{k+1}$ , 然后再代入梯形公式的右端, 即

$$\begin{cases} \tilde{y}_{k+1} = y_k + (x_{k+1} - x_k)f(x_k, y_k), \\ y_{k+1} = y_k + \frac{x_{k+1} - x_k}{2} [f(x_k, y_k) + f(x_{k+1}, \tilde{y}_{k+1})], \quad k = 0, 1, 2, \dots, n-1. \end{cases}$$

这样就不用解方程, 相应的方法称为**改进的 Euler 法**.

#### 预估与校正

改进的 Euler 法中的第一步称为**预估** (Predictor), 第二步称为**校正** (corrector). **预估-校正**是常微分方程数值求解的重要手段之一.

改进的 Euler 方法也通常写为

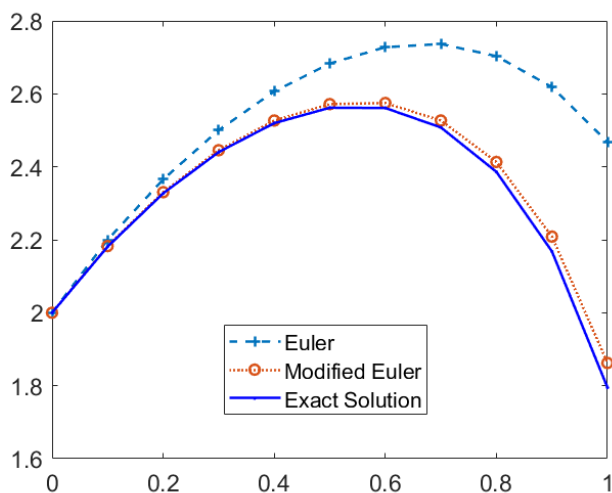
$$\begin{aligned} y_{k+1} &= y_k + \frac{1}{2} h_k (K_1 + K_2), \quad h_k = x_{k+1} - x_k, \\ K_1 &= f(x_k, y_k), \quad K_2 = f(x_k + h_k, y_k + h_k K_1). \end{aligned} \quad (10.6)$$

**例 10.2** 用 Euler 法和改进的 Euler 法求解初值问题

(ODE\_Euler\_02.m)

$$\begin{cases} \frac{dy}{dx} = f(x, y) = y - \frac{12x}{y}, & x \in [0, 1], \\ y(0) = y_0 = 2. \end{cases}$$

下图是以  $n = 10$  为例的结果.



### $\theta$ -方法

更一般地, 我们可以构造如下的  $\theta$ -方法:

$$y_{k+1} = y_k + h_k [\theta f(x_k, y_k) + (1 - \theta) f(x_{k+1}, y_{k+1})],$$

其中  $\theta \in \mathbb{R}$  是加权系数.

- 若  $\theta = 1$ , 则是 Euler 法;
- 若  $\theta = 1/2$ , 则是梯形法;
- 若  $\theta = 0$ , 则是**向后 Euler 法**.

当  $\theta \neq 1$  时, 算法都是隐式的.

#### 10.1.4 单步法误差分析和收敛性

在计算  $y_{k+1}$  时, 如果只需用到  $y_k$  的值 (即前一步的值), 则称为**单步法**, 如果需用到  $y_k, y_{k-1}, \dots, y_{k-r+1}$  的值 (即需要用到前  $r \geq 2$  步的值), 则称为**多步法**.

**例 10.3** Euler 法, 梯形法, 改进的 Euler 法都属于单步法.

求解初值问题 (10.1) 的等步长单步法的一般形式可写为

$$y_{k+1} = y_k + h\Phi(x_k, y_k, y_{k+1}, h), \quad (10.7)$$

这里的  $\Phi$  称为**增量函数**. 如果  $\Phi$  中包含  $y_{k+1}$ , 则方法是**隐式** (implicit) 的, 否则就是**显式** (explicit)




的. 所以显式等步长单步法可表示为


$$y_{k+1} = y_k + h\Phi(x_k, y_k, h), \quad (10.8)$$

**定义 10.1** 设  $y(x)$  是初值问题 (10.1) 的精确解, 则称

$$R_{k+1} \triangleq y(x_{k+1}) - y(k_k) - h\Phi(x_k, y(x_k), h)$$

为显式单步法 (10.8) 的**局部截断误差**.

 注意  $y_k$  与  $y(x_k)$  的区别和两者之间的关系.

 定义中假定  $(x_k, y_k)$  是精确的, 然后迭代一步后得到的数值解的误差, 所以称为是局部的.

**例 10.4** 计算等步长 Euler 法和梯形法的局部截断误差.

**证明.** 利用 Taylor 展开, 可得等步长 Euler 法的局部截断误差为

$$R_{k+1} = \frac{h^2}{2}y^{(2)}(x_k) + O(h^3).$$

等步长梯形法的局部截断误差为

$$R_{k+1} = -\frac{h^3}{12}y^{(3)}(x_k) + O(h^4).$$

□

## 相容性与阶

**定义 10.2 (等步长显式单步法的相容性)** 若增量函数  $\Phi(x, y, h)$  在  $h = 0$  处连续且满足

$$\Phi(x, y, 0) = f(x, y),$$

则称显式单步法 (10.8) 是**相容的** (consistent).

对于显式单步法 (10.8), 当  $h \rightarrow 0$  时我们有

$$\frac{y_{k+1} - y_k}{h} = \Phi(x_k, y_k, h) \rightarrow \Phi(x_k, y_k, 0).$$

又

$$\lim_{h \rightarrow 0} \frac{y(x_{k+1}) - y(x_k)}{h} = y'(x_k) = f(x_k, y_k).$$

所以要使得数值解收敛到精确解,  $\Phi(x_k, y_k, 0)$  必须等于  $f(x_k, y_k)$ , 即算法必须是相容的.

**定义 10.3 (等步长显式单步法的阶)** 设  $p$  是使得下列等式成立的最大正整数

$$R_{k+1} = O(h^{p+1}),$$

则称显式单步法 (10.8) 是  $p$  **阶**的.

**例 10.5** 等步长 Euler 法是 1 阶的, 等步长梯形法是 2 阶的.

**定理 10.2** 等步长显式单步法 (10.8) 相容的充要条件是阶数  $p \geq 1$ .

### 收敛性

**定义 10.4 (收敛性)** 设  $y(x)$  是初值问题 (10.1) 的精确解,  $y_k$  是  $x_k$  处的数值解, 如果对所有  $k = 1, 2, \dots, n$  都有

$$\lim_{h \rightarrow 0} |y_k - y(x_k)| = 0,$$

则称算法是**收敛**的.

这里  $y_k - y(x_k)$  称为**整体截断误差**, 注意与前面的局部截断误差之间的区别.

如果不是等步长方法, 则极限中的  $h$  是指所有小区间长度的最大值.

**定理 10.3** 设显式单步法 (10.8) 是  $p$  阶的, 且增量函数  $\Phi$  关于  $y$  满足 Lipschitz 条件, 并假定初值是精确的, 即  $y_0 = y(x_0)$ , 则算法的整体截断误差满足

$$y_k - y(x_k) = O(h^p),$$

即算法收敛且收敛阶是  $p$ .

**定理 10.4** 如果增量函数  $\Phi(x, y, h)$  关于  $x, y, h$  均满足 Lipschitz 条件, 则等步长显式单步法的收敛性与相容性等价.

**推论 10.5** 设  $f(x, y)$  关于  $y$  满足 Lipschitz 条件, 则

- (1) Euler 法收敛;
- (2) 改进的 Euler 法收敛.



## 10.2 Runge-Kutta 法

在单步法中, 我们需要近似计算

$$y(x_{k+1}) = y(x_k) + \int_{x_k}^{x_{k+1}} f(x, y) \, dx, \quad (10.9)$$

易知算法的精度取决于数值积分的精度.

为了得到更高精度的算法, 我们可以考虑使用下面具有更高代数精度的求积公式:

$$\int_{x_k}^{x_{k+1}} f(x, y) \, dx \approx h_k \sum_{i=1}^r \alpha_i f(x_k + \lambda_i h_k, y(x_k + \lambda_i h_k)), \quad (10.10)$$

其中  $\alpha_i$  为求积系数,

$$h_k = x_{k+1} - x_k, \quad 0 \leq \lambda_1 < \lambda_2 < \cdots < \lambda_r \leq 1,$$

即在  $[x_k, x_{k+1}]$  内取  $r$  个节点做数值积分.

- 为了利用初值信息, 在后面的讨论中我们都要求  $\lambda_1 = 0$ , 即第一个节点为  $(x_k, y_k)$ .
- 一般来说, 积分点越多, 精度可能越高 (但不宜太多, 否则计算复杂).
- 由于  $y(x_k + \lambda_i h_k)$  无法获得, 因此需要用近似方法计算, 以避免解方程.

### Runge-Kutta 法的构造

利用改进的 Euler 法相类似的思想, 我们给出一类实用求解格式的构造方法:

- (1) 首先取  $\lambda_1 = 0$ , 则 (10.10) 右端第一项就是  $f(x_k, y_k)$ , 将其记作为  $K_1$ , 即

$$K_1 \triangleq f(x_k, y_k).$$

- (2) 考虑第二项  $f(x_k + \lambda_2 h_k, y(x_k + \lambda_2 h_k))$ , 需要先计算  $y(x_k + \lambda_2 h_k)$  的近似值, 我们可以在区间  $[x_k, x_k + \lambda_2 h_k]$  进行数值积分 (为了避免解方程, 我们还是使用左矩形法), 于是可得

$$y(x_k + \lambda_2 h_k) = y(x_k) + \int_{x_k}^{x_k + \lambda_2 h_k} f(x, y) \, dx \approx y_k + \mu_{21} h_k f(x_k, y_k) = y_k + \mu_{21} h_k K_1,$$

其中  $\mu_{21}$  为某个常数 (事实上就是  $\lambda_2$ ). 所以

$$f(x_k + \lambda_2 h_k, y(x_k + \lambda_2 h_k)) \approx f(x_k + \lambda_2 h_k, y_k + \mu_{21} h_k K_1) \triangleq K_2,$$

即  $K_2$  可以作为  $f(x_k + \lambda_2 h_k, y(x_k + \lambda_2 h_k))$  的近似值.

- (3) 类似地, 第三项需要先计算  $y(x_k + \lambda_3 h_k)$  的近似值, 我们用复合左矩形法, 即

$$\begin{aligned} y(x_k + \lambda_3 h_k) &= y(x_k) + \int_{x_k}^{x_k + \lambda_3 h_k} f(x, y) \, dx \\ &= y(x_k) + \int_{x_k}^{x_k + \lambda_2 h_k} f(x, y) \, dx + \int_{x_k + \lambda_2 h_k}^{x_k + \lambda_3 h_k} f(x, y) \, dx \\ &\approx y_k + \mu_{31} h_k K_1 + \mu_{32} h_k K_2, \end{aligned}$$

其中  $\mu_{31}$  和  $\mu_{32}$  为常数 (具体表达式后面再给出). 于是

$$f(x_k + \lambda_3 h_k, y(x_k + \lambda_3 h_k)) \approx f(x_k + \lambda_3 h_k, y_k + \mu_{31} h_k K_1 + \mu_{32} h_k K_2) \triangleq K_3,$$

即  $K_3$  可以作为  $f(x_k + \lambda_3 h_k, y(x_k + \lambda_3 h_k))$  的近似值.

(4) 依此类推, 我们可以求得  $f(x_k + \lambda_i h_k, y(x_k + \lambda_i h_k))$  的近似值:

$$K_i \triangleq f\left(x_k + \lambda_i h_k, y_k + h_k \sum_{j=1}^{i-1} \mu_{ij} K_j\right), \quad i = 4, 5, \dots, r.$$

最后, 将  $K_i$  代入公式 (10.10) 可得

$$y(x_{k+1}) \approx y_{k+1} = y_k + h_k \sum_{i=1}^r \alpha_i f\left(x_k + \lambda_i h_k, y_k + h_k \sum_{j=1}^{i-1} \mu_{ij} K_j\right),$$

这就是 **Runge-Kutta 法** 的一般格式. 整理后改写为

$$y_{k+1} = y_k + h_k \sum_{i=1}^r \alpha_i K_i, \quad k = 0, 1, 2, \dots, n-1, \quad (10.11)$$

其中

$$K_1 = f(x_k, y_k), \quad K_i = f\left(x_k + \lambda_i h_k, y_k + h_k \sum_{j=1}^{i-1} \mu_{ij} K_j\right).$$

虽然在上面的推导过程中可以直接计算出  $\mu_{ij}$  的值, 但这样构造出来的方法不一定具有最高精度. 因此我们可以将该方法进行推广, 通过待定系数法来这些参数 (包括  $\mu_{ij}$  和  $\alpha_i$ ) 的值, 以使得新计算方法具有更高的计算精度.

Runge-Kutta 法是一类算法, 而不仅仅是一个算法.

### Runge-Kutta 法的几何意义

由积分中值定理可知, 存在  $\xi \in (x_k, x_{k+1})$  使得

$$y(x_{k+1}) = y(x_k) + \int_{x_k}^{x_{k+1}} f(x, y) dx = y(x_k) + h_k f(\xi, y(\xi)).$$

所以 Runge-Kutta 法 (10.11) 就是用  $\sum_{i=1}^r \alpha_i K_i$  来近似  $f(\xi, y(\xi))$ .

### 如何确定 Runge-Kutta 法中的参数

- 首先确定  $r$  的大小和  $\lambda_i$ , 即数值积分的节点.



- 用待定系数法计算参数  $\alpha_i$  和  $\mu_{ij}$ , 选取准则: 使得计算格式具有尽可能高的精度.
- 需要用到的工具: Taylor 展开.

为了讨论方便, 我们这里只考虑等步长的 Runge-Kutta 法, 即  $h_k = h$ . 关于变步长的 Runge-Kutta 法, 可以参考相关文献.

### $r = 1$ 时的 Runge-Kutta 法

如果  $r = 1$ , 即只取一个点, 也就是  $\lambda_1 = 0$ , 相应的 Runge-Kutta 格式为

$$y_{k+1} = y_k + \alpha_1 h K_1 = y_k + \alpha_1 h f(x_k, y_k).$$

利用 Taylor 展开, 可得局部截断误差为

$$\begin{aligned} R_{k+1} &= y(x_{k+1}) - y(x_k) - \alpha_1 h f(x_k, y(x_k)) \\ &= y'(x_k)h + \frac{1}{2}y''(x_k)h^2 + O(h^3) - \alpha_1 h y'(x_k) \\ &= (1 - \alpha_1)y'(x_k)h + \frac{1}{2}y''(x_k)h^2 + O(h^3). \end{aligned}$$

显然当  $\alpha_1 = 1$  时, 达到最高阶 = 1.

事实上, 此时的 Runge-Kutta 法就是 Euler 法.

### $r = 2$ 时的 Runge-Kutta 法

如果  $r = 2$ , 则相应的 Runge-Kutta 格式为

$$y_{k+1} = y_k + h(\alpha_1 h K_1 + \alpha_2 h K_2), \quad K_1 = f(x_k, y_k), \quad K_2 = f(x_k + \lambda_2 h, y_k + \mu_{21} h K_1).$$

类似地, 利用 Taylor 展开, 可得局部截断误差为

$$\begin{aligned} R_{k+1} &= (1 - \alpha_1 - \alpha_2)y'(x_k)h \\ &\quad + \left(\frac{1}{2} - \alpha_2\lambda_2\right)f'_x(x_k, y(x_k))h^2 \\ &\quad + \left(\frac{1}{2} - \alpha_2\mu_{21}\right)f'_y(x_k, y(x_k))f(x_k, y(x_k))h^2 + O(h^3). \end{aligned}$$

令

$$(1 - \alpha_1 - \alpha_2) = 0, \quad \frac{1}{2} - \alpha_2\lambda_2 = 0, \quad \frac{1}{2} - \alpha_2\mu_{21} = 0, \quad (10.12)$$

则可得到二阶的 Runge-Kutta 法.

需要注意的是, 方程组 (10.12) 是非线性的, 因此要考虑解的存在性和唯一性. 通过分析可知, 非线性方程组 (10.12) 的解是存在的, 但不唯一.

记  $\alpha_2 = a$ , 则可求得

$$\alpha_1 = 1 - a, \quad \lambda_2 = \frac{1}{2a}, \quad \mu_{21} = \frac{1}{2a}.$$

下面给出两个典型的迭代格式:

(1) 取  $a = \frac{1}{2}$ , 则可得迭代格式

$$y_{k+1} = y_k + \frac{1}{2}h(K_1 + K_2), \quad K_1 = f(x_k, y_k), \quad K_2 = f(x_k + h, y_k + hK_1).$$

这其实就是改进的 Euler 法.

(2) 取  $a = 1$ , 则可得迭代格式

$$y_{k+1} = y_k + hK_2, \quad K_1 = f(x_k, y_k), \quad K_2 = f\left(x_k + \frac{1}{2}h, y_k + \frac{1}{2}hK_1\right).$$

称为中点公式 Euler 法 (除了左端点外, 另一个数值积分节点为中点).

**思考**

能否得到更高阶的迭代格式?

( $r = 2$  时, 显式 Runge-Kutta 法的阶至多只能达到二阶)

### $r = 3$ 时的 Runge-Kutta 法

如果  $r = 3$ , 通过类似的讨论, 经过复杂的计算, 我们可以得到三阶 Runge-Kutta 法, 其中一个常用迭代格式为

$$y_{k+1} = y_k + \frac{h}{6}(K_1 + 4K_2 + K_3),$$

$$K_1 = f(x_k, y_k), \quad K_2 = f\left(x_k + \frac{1}{2}h, y_k + \frac{1}{2}hK_1\right), \quad K_3 = f(x_k + h, y_k - hK_1 + 2hK_2).$$

### $r = 4$ 时的 Runge-Kutta 法

如果  $r = 4$ , 通过类似的讨论, 我们可以得到四阶 Runge-Kutta 法, 其中一个典型代表是

$$y_{k+1} = y_k + \frac{h}{6}(K_1 + 2K_2 + 2K_3 + K_4), \quad (10.13)$$

$$K_1 = f(x_k, y_k),$$

$$K_2 = f\left(x_k + \frac{1}{2}h, y_k + \frac{1}{2}hK_1\right),$$

$$K_3 = f\left(x_k + \frac{1}{2}h, y_k + \frac{1}{2}hK_2\right),$$

$$K_4 = f(x_k + h, y_k + hK_3).$$

这就是有名的经典 Runge-Kutta 法, 是当前比较流行的 Runge-Kutta 法, 形式简单且具有较高的收敛速度.

**定理 10.6** 设  $f(x, y)$  关于  $y$  满足 Lipschitz 条件, Runge-Kutta 法 (10.13) 收敛且具有 4 阶收敛阶.





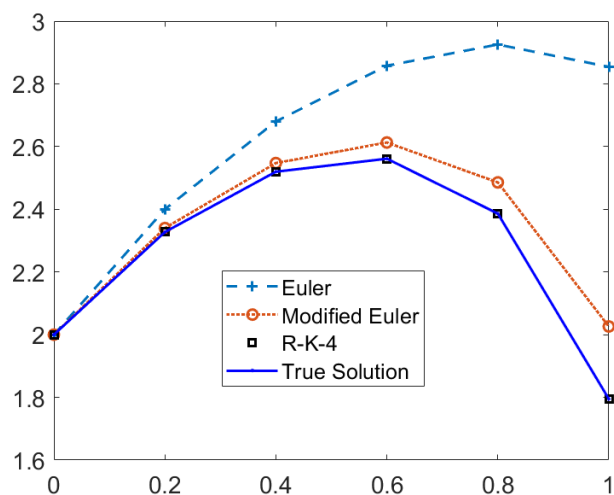
MATLAB 中提供了初值问题求解器 `ode45`, 采用的是变步长的 Runge-Kutta 方法, 属于中阶方法, 也是求解初值问题的首选方法. 另外还有低阶方法 `ode23` 和变阶方法 `ode113`.

**例 10.6** 用经典 Runge-Kutta 法求解初值问题

(ODE\_RK4.m)

$$\begin{cases} \frac{dy}{dx} = f(x, y) = y - \frac{12x}{y}, & x \in [0, 1], \\ y(0) = y_0 = 2. \end{cases}$$

下图是以  $n = 5$  为例的结果. 用经典 Runge-Kutta 法计算出来的数值解几乎与精确解一致.



## 10.3 线性多步法

### 10.3.1 显式 Adams 法

设等步长网格点  $a = x_0 < x_1 < x_2 < \cdots < x_n = b$ , 步长为  $h$ . 假定数值解  $y_1, y_2, \dots, y_k$  已经得到, 下面考虑计算  $y_{k+1}$ . 我们构造  $f(x, y)$  关于  $s$  个节点  $x_{k-s+1}, x_{k-s+1}, \dots, x_k$  的  $s-1$  次插值多项式:

$$L_{s-1}(x, y) = f(x_{k-s+1}, y(x_{k-s+1}))l_0(x) + f(x_{k-s+2}, y(x_{k-s+2}))l_1(x) + \cdots + f(x_k, y(x_k))l_s(x),$$

其中  $l_j(x)$  为  $s-1$  次 Lagrange 基函数, 即

$$l_j(x) = \prod_{i=0, i \neq j}^{s-1} \frac{x - x_{k-s+1+i}}{x_{k-s+1+j} - x_{k-s+1+i}}, \quad j = 0, 1, 2, \dots, s.$$

代入 (10.9) 可得

$$\begin{aligned} y(x_{k+1}) &= y(x_k) + \sum_{j=0}^{s-1} f(x_{k-s+1+j}, y(x_{k-s+1+j})) \int_{x_k}^{x_{k+1}} l_j(x) dx \\ &\approx y_k + h \sum_{j=0}^s b_j f(x_{k-s+1+j}, y(x_{k-s+1+j})), \end{aligned}$$

其中

$$b_j = \frac{1}{h} \int_{x_k}^{x_{k+1}} l_j(x) dx. \quad (10.14)$$

可以证明  $b_j$  的值与  $h$  和  $k$  无关.

于是, 我们就得到下面的线性多步法

$$y_{k+1} = y_k + h \sum_{j=0}^{s-1} b_j f(x_{k-s+1+j}, y(x_{k-s+1+j})). \quad (10.15)$$

具有上面这种形式的迭代格式就称为**显式 Adams 法**, 也称为 **Adams-Bashforth 法**.

下面是几个典型的显式 Adams 法:

- 取  $s = 1$ , 则可得**单步显式 Adams 法**:

$$y_{k+1} = y_k + hf(x_k, y_k).$$

- 取  $s = 2$ , 则可得**两步显式 Adams 法**:

$$y_{k+1} = y_k + h \left( \frac{3}{2} f(x_k, y_k) - \frac{1}{2} f(x_{k-1}, y_{k-1}) \right).$$

- 取  $s = 3$ , 则可得**三步显式 Adams 法**:

$$y_{k+1} = y_k + h \left( \frac{23}{12} f(x_k, y_k) - \frac{4}{3} f(x_{k-1}, y_{k-1}) + \frac{5}{12} f(x_{k-2}, y_{k-2}) \right).$$



## 10.3.2 一般线性多步法

Adams 法只是线性多步法中的一类方法. 一般地, 线性  $s$  步法可表示为

$$y_{k+1} = \sum_{j=0}^{s-1} a_j y_{k-s+1+j} + h \sum_{j=0}^s b_j f(x_{k-s+1+j}, y_{k-s+1+j}), \quad k = s-1, s, \dots, \quad (10.16)$$

其中  $a_j, b_j$  为参数. 如果  $b_s \neq 0$ , 则右端项的  $f$  中含有  $y_{k+1}$ , 称为**隐式线性多步法**, 否则就称为**显式线性多步法**.

如果  $a_0 = a_1 = \dots = a_{s-2} = 0, a_{s-1} = 1$ , 则线性多步法 (10.16) 就称为 **Adams 法**. 对于 Adams 法, 如果  $b_j \neq 0$ , 则称为**隐式 Adams 法** (也称为 **Adams-Moulton 法**), 否则就是**显式 Adams 方法**.

线性  $s$  步法 (10.16) 中的  $y_1, \dots, y_{s-1}$  需要通过其他方法获得.

## 系数的确定

我们使用待定系数法来确定多步法中的参数  $a_j, b_j$ , 基本原则就是使得方法具有尽可能高的阶数. 通过 Taylor 展开 (在  $(x_{k-s+1}, y(x_{k-s+1}))$  点处, 并注意到  $f(x, y) = y'$ ), 我们可得多步法 (10.16) 的局部截断误差为

$$\begin{aligned} R_{k+1} &= y(x_{k+1}) - \sum_{j=0}^{s-1} a_j y(x_{k-s+1+j}) - h \sum_{j=0}^s b_j f(x_{k-s+1+j}, y(x_{k-s+1+j})) \\ &= \sum_{i=0}^{\infty} \frac{1}{i!} y^{(i)}(x_{k-s+1}) s^i h^i - \sum_{j=0}^{s-1} a_j \sum_{i=0}^{\infty} \frac{1}{i!} y^{(i)}(x_{k-s+1}) j^i h^i + h \sum_{j=0}^s b_j \sum_{i=0}^{\infty} \frac{1}{i!} y^{(i+1)}(x_{k-s+1}) j^i h^i \\ &= \left( 1 - \sum_{j=0}^{s-1} a_j \right) y(x_k) + \sum_{i=1}^{\infty} \frac{1}{i!} \left( s^i - \sum_{j=0}^{s-1} a_j j^i - i \sum_{j=0}^s b_j j^{i-1} \right) y^{(i)}(x_k) h^i \end{aligned}$$

所以要使得多步法是  $p$  阶的, 即  $R_{k+1} = O(h^{p+1})$ , 必须要求

$$\begin{cases} 1 - \sum_{j=0}^{s-1} a_j = 0, \\ s^i - \sum_{j=0}^{s-1} a_j j^i - i \sum_{j=0}^s b_j j^{i-1} = 0, \quad i = 1, 2, \dots, p. \end{cases} \quad (10.17)$$

为了书写方便, 这里假定  $0^0 = 1$ .


线性方程组 (10.17) 有  $2s+1$  个未知量, 因此理论上可以构造出最高具有  $2s$  阶的计算方法. 但当  $s \geq 3$  时, 这样构造出来的  $2s$  阶计算方法是不收敛的.

**定理 10.7** 线性多步法 (10.16) 是  $p$  ( $p \geq 1$ ) 阶的当且仅当存在常数  $c \neq 0$ , 使得

$$q(x) - \tilde{q}(x) \ln x = c(x-1)^{p+1} + O(|x-1|^{p+2}), \quad x \rightarrow 1,$$

其中  $q(x) \triangleq 1 - \sum_{j=0}^{s-1} a_j x^j, \tilde{q}(x) \triangleq \sum_{j=0}^s b_j x^j$ .

**证明.** 做变量代换  $x = e^z$ , 则  $x \rightarrow 1$  等价于  $z \rightarrow 0$ . 在  $z = 0$  处做 Taylor 展开即可.  $\square$

 该结论给出了多步法的构造方法, 同时也给出了验证多步法的阶的一个有效方法. 需要注意的是, 方程组 (10.17) 是非线性的, 如果存在解的话, 解不一定唯一.

### 常见的多步法

下面介绍几个常见的多步法, 为了书写方便, 我们记  $f_k \triangleq f(x_k, y_k)$ .

- **单步 Adams-Moulton 法** (隐式 2 阶, 也就是前面介绍的**梯形法**):

$$y_{k+1} = y_k + \frac{h}{2}(f_{k+1} + f_k).$$

- **两步 Adams-Moulton 法** (隐式 3 阶):

$$y_{k+1} = y_k + \frac{h}{12}(5f_{k+1} + 8f_k - f_{k-1}).$$

- **三步 Adams-Moulton 法** (隐式 4 阶):

$$y_{k+1} = y_k + \frac{h}{24}(9f_{k+1} + 19f_k - 5f_{k-1} + f_{k-2}).$$

- **四步 Adams-Moulton 法** (隐式 5 阶):

$$y_{k+1} = y_k + \frac{h}{720}(251f_{k+1} + 646f_k - 264f_{k-1} + 106f_{k-2} - 19f_{k-3}).$$

- **Simpson 法** (隐式 4 阶):

$$y_{k+1} = y_{k-1} + \frac{h}{3}(f_{k+1} + 4f_k + f_{k-1}).$$

- **Hamming 法** (隐式 4 阶):

$$y_{k+1} = \frac{1}{8}(9y_k - y_{k-2}) + \frac{3h}{8}(f_{k+1} + 2f_k - f_{k-1}).$$

- **Dahlquist 法** (显式 3 阶):

$$y_{k+1} = -4y_k + 5y_{k-1} + h(4f_k + 2f_{k-1}).$$

- **Milne 法** (显式 4 阶):

$$y_{k+1} = y_{k-3} + \frac{4h}{3}(2f_k - f_{k-1} + 2f_{k-2}).$$

### 10.3.3 预估-校正方法

对于隐式的线性多步法 (10.16), 每次迭代都需要求解一个方程 (通常是非线性的), 因此需要借助迭代法求解  $y_{k+1}$ . 构造不动点迭代法:

$$y_{k+1}^{[i+1]} = \sum_{j=0}^{s-1} a_j y_{k-s+1+j} + h \sum_{j=0}^{s-1} b_j f_{k-s+1+j} + h b_s f(x_{k+1}, y_{k+1}^{[i]}), \quad i = 0, 1, 2, \dots, \quad (10.18)$$

其中  $y_{k+1}^{[0]}$  为迭代初值. 由不动点迭代的收敛性定理 (6.4) 和 Lipschitz 条件 (10.2) 可知, 迭代格式 (10.18) 收敛的充分条件是

$$Lh|b_s| < 1,$$

其中  $L$  是 Lipschitz 常数.



我们知道, 对于非线性方程的迭代求解, 一个好的初值往往能有效减少迭代步数. 对于迭代法 (10.18), 我们可以考虑先用显式多步法得到一个近似值, 然后将其作为迭代初值, 比如取

$$y_{k+1}^{[0]} = \sum_{j=0}^{s-1} \tilde{a}_j y_{k-s+1+j} + h \sum_{j=0}^{s-1} \tilde{b}_j f_{k-s+1+j}.$$

这就是**预估-校正**算法的基本思想.

 理论上, 任何一个显式多步法和隐式多步法都可以组合成一个**预估-校正**算法.

## 10.4 一阶方程组与高阶方程 \*

### 10.4.1 一阶方程组

考虑一阶常微分方程组

$$\begin{cases} \frac{dY}{dx} = F(x, Y) \\ Y(x_0) = Y_0 \end{cases} \quad x \in [a, b], \quad (x_0 = a)$$

其中

$$Y = \begin{bmatrix} y_1(x) \\ y_2(x) \\ \vdots \\ y_m(x) \end{bmatrix}, \quad F = \begin{bmatrix} f_1(x, Y) \\ f_2(x, Y) \\ \vdots \\ f_m(x, Y) \end{bmatrix}, \quad Y_0 = \begin{bmatrix} y_1^0 \\ y_2^0 \\ \vdots \\ y_m^0 \end{bmatrix}.$$

对应的四阶经典 Runge-Kutta 法迭代格式

$$\begin{aligned} Y_{k+1} &= Y_k + \frac{h}{6}(K_1 + 2K_2 + 2K_3 + K_4), \quad k = 0, 1, 2, \dots, n, \\ K_1 &= F(x_k, Y_k), \\ K_2 &= F\left(x_k + \frac{1}{2}h, Y_k + \frac{1}{2}hK_1\right), \\ K_3 &= F\left(x_k + \frac{1}{2}h, Y_k + \frac{1}{2}hK_2\right), \\ K_4 &= F(x_k + h, Y_k + hK_3). \end{aligned}$$

**例 10.7** 用经典 Runge-Kutta 法求解初值问题

$$\begin{cases} y' = f(x, y, z), \\ z' = g(x, y, z), \\ y(x_0) = y_0, \quad z(x_0) = z_0. \end{cases}$$

**证明.** 四阶经典 Runge-Kutta 法的迭代格式为

$$\begin{aligned} y_{k+1} &= y_k + \frac{h}{6}(K_1 + 2K_2 + 2K_3 + K_4), \\ z_{k+1} &= z_k + \frac{h}{6}(L_1 + 2L_2 + 2L_3 + L_4), \end{aligned}$$

其中

$$\begin{aligned} K_1 &= f(x_k, y_k, z_k), \quad L_1 = g(x_k, y_k, z_k), \\ K_2 &= f\left(x_k + \frac{1}{2}h, y_k + \frac{1}{2}hK_1, z_k + \frac{1}{2}hL_1\right), \quad L_2 = g\left(x_k + \frac{1}{2}h, y_k + \frac{1}{2}hK_1, z_k + \frac{1}{2}hL_1\right), \\ K_3 &= f\left(x_k + \frac{1}{2}h, y_k + \frac{1}{2}hK_2, z_k + \frac{1}{2}hL_2\right), \quad L_3 = g\left(x_k + \frac{1}{2}h, y_k + \frac{1}{2}hK_2, z_k + \frac{1}{2}hL_2\right), \\ K_4 &= f(x_k + h, y_k + hK_3, z_k + hL_3), \quad L_4 = g(x_k + h, y_k + hK_3, z_k + hL_3). \end{aligned}$$



$$K_4 = f(x_k + h, y_k + hK_3, z_k + hL_3), \quad L_4 = g(x_k + h, y_k + hK_3, z_k + hL_3).$$

□

### 10.4.2 高阶方程

考虑  $n$  阶常微分方程

$$\begin{cases} \frac{d^m y}{dx^m} = f(x, y, y^{(1)}, \dots, y^{(m-1)}) \\ y(x_0) = y_0, y^{(1)}(x_0) = y_0^{(1)}, \dots, y^{(m-1)}(x_0) = y_0^{(m-1)}, \end{cases} \quad x \in [a, \leq b], \quad (x_0 = a)$$

引入变量

$$z_1 = y, \quad z_2 = y^{(1)}, \dots, z_m = y^{(m-1)},$$

则原方程就等价于下面的一阶方程组

$$\begin{cases} z_1' = z_2 \\ z_2' = z_3 \\ \vdots \\ z_m' = f(x, z_1, z_2, \dots, z_m) \\ z_1(x_0) = y_0, z_2(x_0) = y_0^{(1)}, \dots, z_m(x_0) = y_0^{(m-1)}. \end{cases}$$

如何就可以用四阶经典 Runge-Kutta 法进行数值求解.

**例 10.8** 求解 Van der Pol 方程

(ODE\_van\_der\_Pol.m)

$$\begin{cases} \frac{d^2 y}{dx^2} + \mu \left( \frac{1}{3} \left( \frac{dy}{dx} \right)^3 - \frac{dy}{dx} \right) + y = 0, \\ y(0) = 1, y'(0) = 1. \end{cases}$$

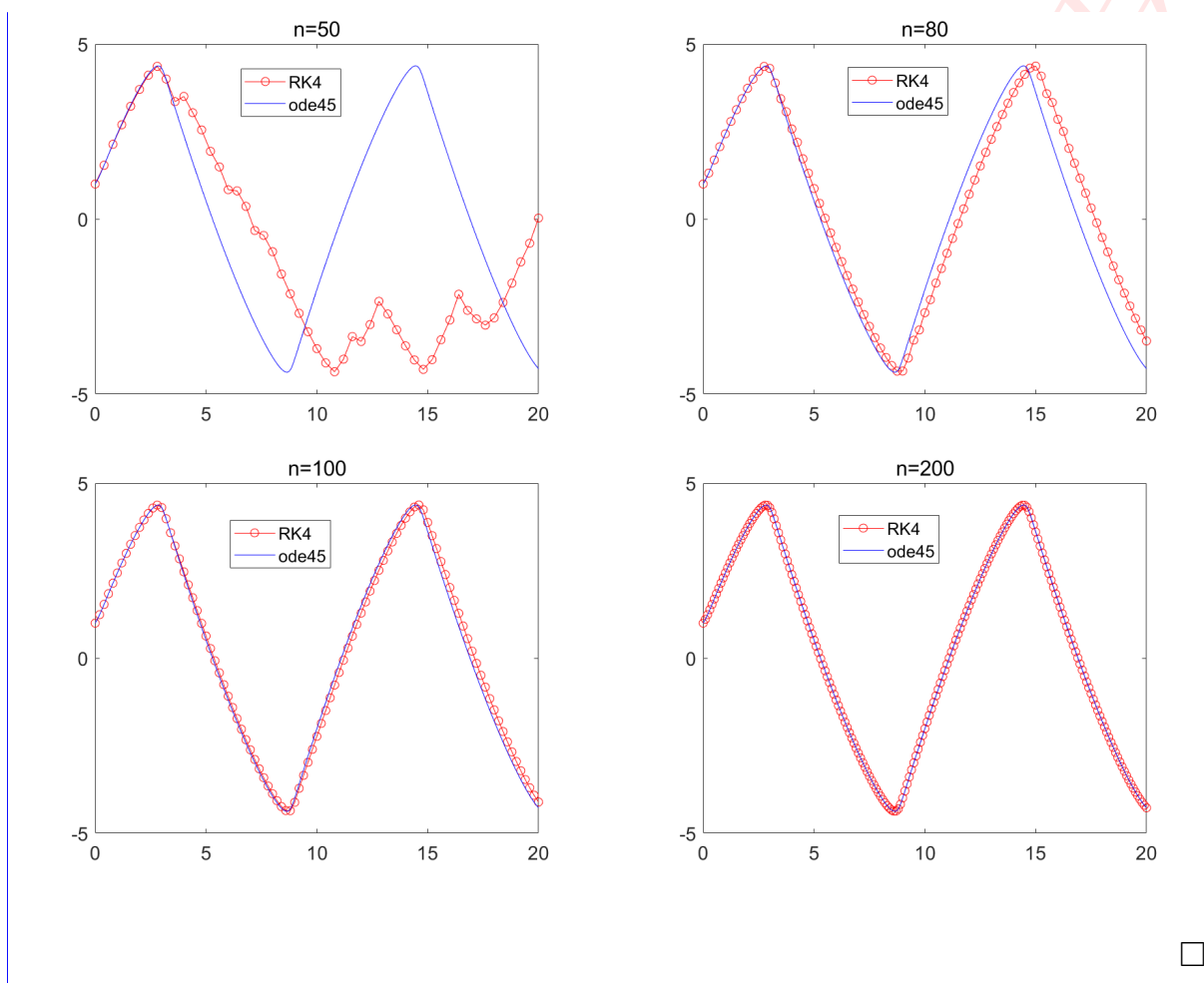
求解区间为  $[0, 20]$ .

**证明.** 首先转化为一阶方程组

$$\begin{cases} z_1' = z_2, \\ z_2' = -z_1 - \mu \left( \frac{1}{3} z_2^3 - z_2 \right), \end{cases}$$

然后使用 Runge-Kutta 求解.

下图是以  $\mu = 5$  为例, 分别用四阶经典 Runge-Kutta 法和 MATLAB 自带 `ode45` 函数求解后的结果, 其中  $n$  为区间等分个数. 由于 `ode45` 是自适应变步长方法, 节点个数是由函数自动根据需要确定的, 在本例中大约是 329.

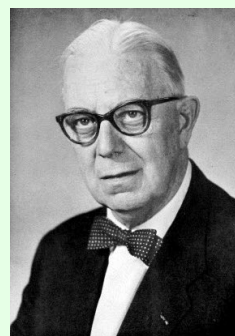


### 拓展阅读

Balthasar van der Pol (1889 – 1959) 是荷兰的一位著名电子工程师, 是现代实验动力学的开创者. 1927 年, 为了描述真空管放大器的极限环振荡现象, 他首次提出了著名的 van der Pol 方程

$$\ddot{y} - \mu(1 - y^2)\dot{y} + y = 0.$$

也称为 van der Pol 振荡器方程. 此后, 人们对方程进行了大量研究和改进, 并应用到其他领域, 包括物理学、生物学、神经学、经济学等. van der Pol 方程已经成为描述震荡系统一种基础模型, 在更加复杂的动力系统的建模中广泛使用.





## 10.5 课后练习

**练习 10.1** 试利用 Taylor 展开构造 3 阶的等步长显式单步法.  
(事实上, 可以通过 Taylor 展开构造任意阶的等步长单步法)

**练习 10.2** 证明由 (10.14) 所定义的  $b_j$  的值, 即

$$b_j = \frac{1}{h} \int_{x_{k-s}}^{x_{k+1}} l_j(x) \, dx$$

与  $h$  和  $k$  无关.

# 参考文献

- [1] J. L. Aurentz, T. Mach, R. Vandebril and D. S. Watkins, Fast and backward stable computation of roots of polynomials, *SIAM Journal on Matrix Analysis and Applications*, 36 (2015), 942–973. [162](#), [163](#)
- [2] Z. Bai, J. Demmel, J. Dongarra, A. Ruhe and H. van der Vorst, *Templates for the Solution of Algebraic Eigenvalue Problems: A Practical Guide*, SIAM, Philadelphia, 2000. [121](#)
- [3] D.A. Bini, P. Boito, Y. Eidelman, L. Gemignani and I. Gohberg, A fast implicit QR eigenvalue algorithm for companion matrices, *Linear Algebra and its Applications*, 432 (2010), 2006–2031. [162](#), [163](#)
- [4] Åke Björck, Solving linear least square problems by Gram-Schmidt orthogonalization, *BIT*, 7 (1967), 1–21. [85](#)
- [5] Åke Björck, *Numerical Methods for Least Squares Problems*, SIAM, Philadelphia, PA, 1996. [70](#)
- [6] S. Börm and C. Mehl, *Numerical Methods for Eigenvalue Problems*, De Gruyter, 2012. [121](#)
- [7] J. C. Butcher, *Numerical Methods for Ordinary Differential Equations*, 3rd Edition, John Wiley & Sons, 2016. [284](#)
- [8] S. Chandrasekaran, M. Gu, J. Xia and J. Zhu, A fast QR algorithm for companion matrices, In: J. A. Ball, Y. Eidelman, J.W. Helton, V. Olshevsky, J. Rovnyak (eds) *Recent Advances in Matrix and Operator Theory*, 111–143, 2007. *Operator Theory: Advances and Applications* book series, vol 179, [162](#), [163](#)
- [9] J.J.M. Cuppen, A Divide and Conquer Method for the Symmetric Tridiagonal Eigenproblem, *Numerische Mathematik*, 36 (1981), 177–195. [143](#), [146](#)
- [10] T. A. Davis, *Direct Methods for Sparse Linear Systems*, SIAM, 2006. [37](#)
- [11] J. W. Demmel, *Applied Numerical Linear Algebra*, SIAM, Philadelphia, PA, 1997. [148](#)
- [12] J. E. Jr Dennis and R. B. Schnabel, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, Prentice Hall, Englewood Cliffs, 1983. [167](#)
- [13] Z. Drmač and K. Veselić, New fast and accurate jacobi SVD algorithm. I *SIAM Journal on Matrix Analysis and Applications*, 29 (2008), 1322–1342. [152](#), [156](#)
- [14] Z. Drmač and K. Veselić, New fast and accurate jacobi SVD algorithm. II *SIAM Journal on Matrix Analysis and Applications*, 29 (2008), 1343–1362. [152](#), [156](#)
- [15] I. S. Duff, A. M. Erisman and J. K. Reid, *Direct Methods for Sparse Matrices*, 2nd edition, Oxford, 2017. [37](#)
- [16] K. Fernando and B. Parlett, Accurate singular values and differential qr algorithms, *Numerische Mathematik*, 67 (1994), 191–229. [152](#), [155](#)

- [17] C. Gear, *Numerical Initial Value Problems in Ordinary Differential Equations*, Prentice-Hall, Englewood Cliffs, 1971. [284](#)
- [18] D. F. Griffiths and D. J. Higham, *Numerical Methods for Ordinary Differential Equations: Initial Value Problems*, Springer, 2010. [284](#)
- [19] C. F. Gerald and P. O. Wheatley, *Applied Numerical Analysis*, 7th edition, 白峰杉改编, 高等教育出版社, 2006. [167](#)
- [20] G. H. Golub and W. Kahan, Calculating the singular values and pseudo-inverse of a matrix, *SIAM Journal on Numerical Analysis*, Series B, 2 (1965), 205–224. [152](#)
- [21] G. H. Golub and C. F. Van Loan, *Matrix Computations*, The 4th Edition, The Johns Hopkins University Press, Baltimore, MD, 2013. [20](#), [37](#), [83](#), [87](#), [89](#), [105](#), [121](#), [152](#), [153](#)
- [22] J. F. Grcar, Mathematicians of Gaussian Elimination, *Notices of the AMS*, Number 6, 58 (2011), 782–792. [37](#)
- [23] A. Greenbaum, *Iterative Methods for Solving Linear Systems*, SIAM, Philadelphia, 1997. [105](#)
- [24] M. Gu and S. C. Eisenstat, A stable algorithm for the rank-1 modification of the symmetric eigenproblem, *SIAM Journal on Matrix Analysis and Applications*, 15 (1994), 1266–1276. [145](#)
- [25] M. Gu and S. C. Eisenstat, A Divide-and-Conquer algorithm for the bidiagonal SVD, *SIAM Journal on Matrix Analysis and Applications*, 16 (1995), 79–92. [147](#)
- [26] M. Gu and S. C. Eisenstat, A Divide-and-Conquer algorithm for the symmetric tridiagonal eigenproblem, *SIAM Journal on Matrix Analysis and Applications*, 16 (1995), 172–191. [143](#), [147](#)
- [27] E. Hairer and G. Wanner, *Solving Ordinary Differential Equations II: Stiff and Differential-Algebraic Problems*, 2nd Edition, Springer, 1996. [284](#)
- [28] E. Hairer, S. P. Nørsett and G. Wanner, *Solving Ordinary Differential Equations I: Nonstiff Problems*, 2nd Edition, Springer, 2009. [284](#)
- [29] M. R. Hestenes and E. L. Stiefel, Methods of conjugate gradients for solving linear systems, *Journal of research of the National Bureau of Standards*, 49 (1952), 2379.
- [30] N. J. Higham, *Accuracy and Stability of Numerical Algorithms*, Second Edition, SIAM, Philadelphia, 2002. [85](#), [99](#)
- [31] R. A. Horn and C. R. Johnson, *Matrix Analysis*, Cambridge University Press, New York, 1985. 2nd edition, 2013. [21](#), [78](#), [88](#), [157](#), [160](#)
- [32] R. A. Horn and C. R. Johnson, *Topics in Matrix Analysis*, Cambridge University Press, New York, 1991. [157](#)
- [33] W. G. Horner, A New Method of Solving Numerical Equations of All Orders, by Continuous Approximation, *Philosophical Transactions of the Royal Society of London*, 109 (1819), 308–335. [34](#)
- [34] C. T. Kelley, *Iterative Methods for Linear and Nonlinear Equations*, SIAM, Philadelphia, 1995. [167](#)
- [35] C. T. Kelley, *Solving Nonlinear Equations with Newton's Method*, SIAM, Philadelphia, 2003. [167](#)
- [36] C. Lanczos Solutions of systems of linear equations by minimized iterations, *Journal of research of the*

- National Bureau of Standards*, 49 (1952), 2341.
- [37] S. Li, M. Gu and B. N. Parlett, An Improved DQDS Algorithm, *SIAM Journal on Scientific Computing*, 36 (2014), C290–C308. [155](#)
- [38] E. H. Moore, On the reciprocal of the general algebraic matrix, *Bulletin of the American Mathematical Society*, 26 (1920), 394–395. [96](#)
- [39] J. M. Ortega and W. C. Rheinboldt, *Iterative Solution of Nonlinear Equations in Several Variables*, Academic Press, New York-London, 1970. [167](#)
- [40] C. C. Paige, M. Rozložník and Z. Strakoš, Modified Gram–Schmidt (MGS), least squares, and backward stability of MGS-GMRES, *SIAM Journal on Matrix Analysis and Applications*, (28) 2006, 264–284. [85](#)
- [41] B. N. Parlett, *The Symmetric Eigenvalue Problem*, The 2nd Edition, SIAM, Philadelphia, PA, 1998. [121](#)
- [42] B. N. Parlett and O. Marques, An implementation of the dqds algorithm (positive case), *Linear Algebra and its Applications*, 309 (2000), 217–259. [155](#)
- [43] R. Penrose, A generalized inverse for matrices, *Mathematical Proceedings of the Cambridge Philosophical Society*, 51 (1955), 406–413. [96](#)
- [44] J. Rutter, *A Serial Implementation of Cuppen's Divide and Conquer Algorithm for the Symmetric Eigenvalue Problem*, Master's Thesis, University of California, 1994. [146](#)
- [45] Y. Saad and M. H. Schultz, GMRES: A generalized minimal residual method for solving nonsymmetric linear systems, *SIAM Journal on Scientific & Statistical Computing*, 7 (1986), 856–869.
- [46] Y. Saad, *Numerical Methods for Large Eigenvalue Problems: Theory and Algorithms*, Manchester University Press, Manchester, UK, 1992. 2nd revised edition, SIAM, Philadelphia, 2011. [121](#)
- [47] Y. Saad, *Iterative Methods for Sparse Linear Systems*, 2nd edition, SIAM, Philadelphia, 2003. [105](#)
- [48] G. W. Stewart, *Matrix Algorithms, Vol II: Eigensystems*, SIAM, Philadelphia, PA, 2001. [121](#)
- [49] G. W. Stewart and J. G. Sun, *Matrix Perturbation Theory*, Academic Press, New York, 1990. [157](#)
- [50] L. N. Trefethen, Numerical Analysis, in *Princeton Companion to Mathematics*, Edited by T. Gowers, J. Barrow-Green and I. Leader, Princeton University Press, 2008. [3](#), [37](#)
- [51] L. N. Trefethen, *Approximation Theory and Approximation Practice*, Extended Edition, SIAM, Philadelphia, 2019. [225](#)
- [52] L. N. Trefethen and D. Bau, *Numerical Linear Algebra*, SIAM, Philadelphia, PA, 1997. [89](#)
- [53] John Von Neumann and H. H. Goldstine, Numerical inverting of matrices of high order, *Bulletin of the American Mathematical Society*, 53 (1947), 1021–1099. [2](#)
- [54] D. S. Watkins, *The Matrix Eigenvalue Problem: GR and Krylov Subspace Methods*, SIAM, Philadelphia, 2007. [138](#)
- [55] J. H. Wilkinson, *The Algebraic Eigenvalue Problem*, Clarendon Press, Oxford University, Oxford, 1965. [74](#), [85](#), [121](#), [143](#)
- [56] M. Van Barel, R. Vandebril and P. Van Dooren, Computing the eigenvalues of a companion matrix,



2008. [162](#), [163](#)
- [57] R. S. Varga, *Matrix Iterative Analysis*, Prentice-Hall, Englewood Cliffs, NJ, 1962. 2nd edition, Springer-Verlag, Berlin, 2000. [105](#)
- [58] D. M. Young, *Iterative Solution of Large Linear Systems*, Academic Press, New York, 1971. [105](#)
- [59] 陈志明, 科学计算: 科技创新的第三种方法, 中国科学院院刊, 27 (2012), 161–166. [1](#)
- [60] 范金燕, 袁亚湘, 非线性方程组数值方法, 科学出版社, 2018. [167](#)
- [61] 蒋尔雄, 矩阵计算, 科学出版社, 2008. [145](#)
- [62] 蒋尔雄, 赵风光, 苏仰峰, 数值逼近 (第二版), 复旦大学出版社, 2008. [225](#)
- [63] 李庆扬, 王能超, 易大义, 数值分析, 第五版, 清华大学出版社, 2008. [22](#)
- [64] 石钟慈, 第三种科学方法 — 计算机时代的科学计算, 清华大学出版社, 暨南大学出版社, 2000. [1](#)
- [65] 孙继广, 矩阵扰动分析, 科学出版社, 北京, 2001. [157](#)
- [66] 王仁宏, 数值逼近 (第二版), 高等教育出版社, 1999. [225](#), [226](#)
- [67] 魏木生, 李莹, 赵建立, 广义最小二乘问题的理论和计算, 第二版, 科学出版社, 北京, 2020. [70](#), [96](#), [98](#)
- [68] 徐树方, 钱江, 矩阵计算六讲, 高等教育出版社, 北京, 2011. [152](#), [153](#), [163](#)
- [69] 詹兴致, 矩阵论, 高等教育出版社, 北京, 2008. [166](#)