数值分析与算法*

潘建瑜

jypan@math.ecnu.edu.cn

(2024 年 2 月)

*本讲义仅供《数值分析》课堂教学使用

纸上得来终觉浅，绝知此事要躬行。

目 录

第九讲 矩阵特征值计算

第十讲 常微分方程初值问题

1 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, 对人类的生产活动和社会活动产生了极其重要的影响. 特别是随着互联网的出现, 计算机已经彻底改变了人们的生活、学习和工作方式.

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.

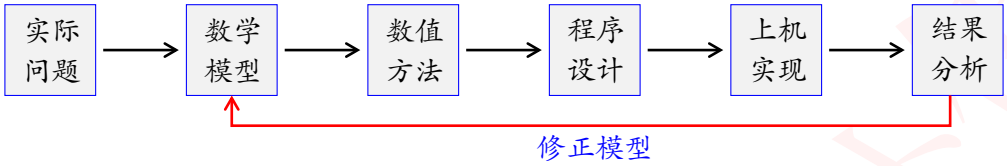
> 一门科学, 只有当它成功地运用数学时, 才能达到真正完善的地步.
>
> — 马克思 (1818 – 1883)

科学计算是指借助计算机高速计算的能力, 运用数学方法解决现代科学、工程、经济或人文中的复杂问题. 它融数学建模、算法设计、软件研发和计算模拟为一体. 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 [72]. 科学计算已经和理论研究与实验研究一起, 成为当今世界科学技术创新的主要方式 [65], 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.

> The purpose of computing is insight, not numbers.
>
> — Richard Wesley Hamming (1915 – 1998, 1968 年图灵奖获得者)

一般来说, 运用数学方法解决实际问题主要分以下几个步骤:



数值分析的任务主要包括其中的数值方法、程序设计和上机实现.

1.1.2 数值分析

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, 对应的数学分支为计算数学. 1947 年, Von Neumann 和 Goldstine 在美国数学会通报 (Bulletin of the AMS) 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 [60], 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.

数值分析的核心是算法的设计与分析, 研究内容包括数值代数 (线性、非线性), 数值逼近 (函数插值、最佳逼近、数据拟合), 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.

✍ 有关数值分析的研究内容和意义, 可以参阅普林斯顿数学手册 (*Princeton Companion to Mathematics*) 中由 Trefethen 撰写的关于 Numerical Analysis 的介绍 [54].

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 (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.

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.
- 数学模型分析: 数学问题的结构特点, 数学模型的评估 (合理性、准确性).

✍ 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).
- 易用: 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, 因此求出的解通常是带有误差的.
✍ 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, 大约需要 10000 年! 但如果使用高斯消去法, 则 0.1 秒钟都不到.

学习建议

在学习数值分析时, 大家要注意以下几点:

- 掌握数值方法的基本思想和原理 (不能仅仅靠记忆);
- 掌握数值方法的设计和分析的一些常用技术和技巧;
- 重视误差分析、收敛性和稳定性分析等基本理论;
- 适量的数值计算训练 (包括编程实践和手工推导, 有助于加深对算法的理解).

主要参考资料

[1] 李庆扬, 王能超, 易大义, **数值分析**, 第五版, 2008.
[2] R.L. Burden, J. Douglas Faires and A. M. Burden, *Numerical Analysis*, 10th Edition, 2016.
[3] T. Sauer, *Numerical Analysis*, 3rd Edition, 2018.
[4] M.T. Heath, *Scientific Computing: An Introductory Survey*, Revised 2nd Edition, 2018.
[5] J. Stoer and R. Bulirsch, *Introduction to Numerical Analysis*, 3rd Edition, 2002.

其中 [2-4] 比较基础, [5] 比较偏理论.

1.2 线性代数基础

本小节简要介绍讲义中所涉及的线性代数基础知识, 特别是线性空间和矩阵的相关基础知识, 主要参考 [30, 31, 64, 66].

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

- 加法四条规则
  - (1) 交换律: $x + y = y + x, \quad \forall\, x, y \in \mathbb{S}$;
  - (2) 结合律: $(x + y) + z = x + (y + z), \quad \forall\, x, y, z \in \mathbb{S}$;
  - (3) 零元素: 存在一个元素 0, 使得 $x + 0 = x, \quad \forall\, x \in \mathbb{S}$;
  - (4) 逆运算: 对任意 $x \in \mathbb{S}$, 都存在负元素 $y \in \mathbb{S}$, 使得 $x + y = 0$, 记 $y = -x$;
- 数乘四条规则
  - (1) 单位元: $1 \cdot x = x, \quad 1 \in \mathbb{F}, \forall\, x \in \mathbb{S}$;
  - (2) 结合律: $\alpha \cdot (\beta \cdot x) = (\alpha\beta) \cdot x, \quad \forall\, \alpha, \beta \in \mathbb{F},\ x \in \mathbb{S}$;
  - (3) 分配律: $(\alpha + \beta) \cdot x = \alpha \cdot x + \beta \cdot x, \quad \forall\, \alpha, \beta \in \mathbb{F},\ x \in \mathbb{S}$;
  - (4) 分配律: $\alpha \cdot (x + y) = \alpha \cdot x + \alpha \cdot y, \quad \forall\, \alpha \in \mathbb{F},\ x, y \in \mathbb{S}$.

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

例 1.3 常见的线性空间有:

- $\mathbb{R}^n \to$ 所有 n 维实向量组成的集合, 是 $\mathbb{R}$ 上的线性空间.
- $\mathbb{C}^n \to$ 所有 n 维复向量组成的集合, 是 $\mathbb{C}$ 上的线性空间.
- $\mathbb{R}^{m\times n} \to$ 所有 $m \times n$ 阶实矩阵组成的集合, 是 $\mathbb{R}$ 上的线性空间.

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

线性无关、秩、基和维数

设 $\mathbb{S}$ 是数域 $\mathbb{F}$ 上的一个线性空间, $x_1,x_2,\dots,x_k$ 是 $\mathbb{S}$ 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

$$\alpha_1x_1+\alpha_2x_2+\cdots+\alpha_kx_k=0,$$

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 $\mathbb{S}$ 中的一组向量. 如果 $x\in\mathbb{S}$ 可以表示为

$$x=\alpha_1x_1+\alpha_2x_2+\cdots+\alpha_kx_k,$$

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 ($r\le m$) 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 $\operatorname{rank}(\{x_1,x_2,\dots,x_m\})=r$.

设 $x_1,x_2,\dots,x_n$ 是 $\mathbb{S}$ 中的一组线性无关向量. 如果 $\mathbb{S}$ 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 $\mathbb{S}$ 的一组**基**, 并称 $\mathbb{S}$ 是 n 维的, 即 $\mathbb{S}$ 的**维数**为 n, 记为 $\dim(\mathbb{S})=n$. 如果 $\mathbb{S}$ 中可以找到任意多个线性无关向量, 则称 $\mathbb{S}$ 是**无限维**的.

子空间

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

例 1.4 设 $\mathbb{S}$ 是数域 $\mathbb{F}$ 上的一个线性空间, $x_1,x_2,\dots,x_k\in\mathbb{S}$, 记

$$\operatorname{span}\{x_1,x_2,\dots,x_k\}\triangleq\left\{\alpha_1x_1+\alpha_2x_2+\cdots+\alpha_kx_k\ :\ \alpha_1,\alpha_2,\dots,\alpha_k\in\mathbb{F}\right\},$$

即由 $x_1,x_2,\dots,x_k$ 的所有线性组合构成的集合. 可以验证, $\operatorname{span}\{x_1,x_2,\dots,x_k\}$ 是 $\mathbb{S}$ 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 矩阵的特征值与谱半径

定义 1.3 设 $A\in\mathbb{R}^{n\times n}$ (或 $\mathbb{C}^{n\times n}$), 若存在 $\lambda\in\mathbb{C}$ 和非零向量 $x,y\in\mathbb{C}^n$, 使得

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

这里 y^* 表示 y 的共轭转置, 则 λ 是 A 的**特征值**, x 称为 A 对应于 λ 的 (右) **特征向量**, y 称为 A 对应于 λ 的**左特征向量**, 并称 (λ,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也有可能是复的;
- n 阶矩阵总是存在 n 个特征值 (其中可能有相等的), 通常记为 $\lambda_1, \lambda_2, \ldots, \lambda_n$;
- 特征值也可以通过特征多项式的零点来定义;
- 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, 合同变换不改变矩阵的惯性指数.

思考: 设 $A \in \mathbb{R}^{n\times n}$, 则 A^{T} 和 A 的特征值和特征向量是什么关系? A^{-1} 和 A 的特征值和特征向量是什么关系? 更一般地, 设 $p(t)$ 是一个多项式, 则 $p(A)$ 和 A 的特征值与特征向量是什么关系? 如果 $f(t)$ 是有理函数呢?

定义 1.4 (谱半径) 设 $A \in \mathbb{R}^{n\times n}$ (或 $\mathbb{C}^{n\times n}$), 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:

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

根据高次多项式的韦达定理 [59], 我们可以得到下面的结论.

定理 1.1 设 $A \in \mathbb{R}^{n\times n}$ (或 $\mathbb{C}^{n\times n}$), 称 $\mathrm{tr}(A) \triangleq a_{11} + a_{22} + \cdots + a_{nn}$ 为矩阵 A 的迹 (trace), 有

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

定义 1.5 (特征值分解) 设 $A \in \mathbb{R}^{n\times n}$ (或 $\mathbb{C}^{n\times n}$). 若存在非奇异矩阵 $X \in \mathbb{C}^{n\times n}$, 使得

$$X^{-1}AX = \Lambda, \tag{1.1}$$

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 Λ 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.

需要指出的是, 并不是所有特征值都可以对角化, 如果 2 阶以上的 Jordan 块矩阵.

定理 1.2 设 $A \in \mathbb{R}^{n\times n}$ (或 $\mathbb{C}^{n\times n}$), 则

(1) A 可对角化当且仅当 A 有 n 个线性无关的特征向量;

(2) A 可对角化当且仅当 A 的所有特征值的代数重数与几何重数都相等.

特别地, 若 A 有 n 个互不相等的特征值, 则 A 可对角化.

如果 A 是对称的, 则 A 的所有特征值都是实的, 而且可以正交对角化.

定理 1.3 设 $A \in \mathbb{R}^{n\times n}$ 对称, 则 A 的特征值 $\lambda_1, \lambda_2, \ldots, \lambda_n$ 都是实数, 且存在正交矩阵 $Q \in \mathbb{R}^{n\times n}$, 使得

$$Q^{\mathsf{T}}AQ = \Lambda \quad 或 \quad A = Q\Lambda Q^{\mathsf{T}},$$

其中 $\Lambda = \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n)$ 是实对角矩阵, Q 的第 i 列为 A 的对应于 λ_i 的特征向量.

✍ 该结论对复矩阵也成立, 此时 Q 是复的酉矩阵, 但 Λ 仍然是实对角矩阵.

1.2.3 对称正定矩阵

定义 1.6 设 $A \in \mathbb{C}^{n\times n}$.

- 若对所有向量 $x \in \mathbb{C}^n$ 有 $\mathrm{Re}(x^*Ax) \geq 0$, 则称 A 是半正定的;
- 若对所有非零向量 $x \in \mathbb{C}^n$ 有 $\mathrm{Re}(x^*Ax) > 0$, 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 $x^*Ax \in \mathbb{R}$, 则 $A^* = A$.

定理 1.4 设 $A \in \mathbb{C}^{n\times n}$, 则 A 正定 (或半正定) 的充要条件是 $H = \frac{1}{2}(A + A^*)$ 正定 (或半正定).

(留作课外自习)

定理 1.5 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 $x^{\mathsf{T}}Ax > 0$ (或 $x^{\mathsf{T}}Ax \geq 0$). (留作课外自习)

定理 1.6 (Hermite 矩阵正定的充分条件) $A \in \mathbb{C}^{n\times n}$ 是 Hermite 正定矩阵的充要条件是 A 是 Hermite 的, 且所有特征值都大于零或者所有顺序主子式都大于零. (留作课外自习)

对称正定矩阵的更多性质

设 $A \in \mathbb{R}^{n\times n}$ 是对称正定矩阵, 则

(1) A 非奇异, 且 A^{-1} 也对称正定;
(2) A 的所有特征值都是正实数;
(3) A 的所有顺序主子阵都对称正定的;
(4) A 的所有顺序主子式都大于零.

✍ 对于 Hermite 正定矩阵, 也有类似的结论.

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 1.7 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) $\operatorname{rank}(B) = \operatorname{rank}(A)$, 因此, 若 A 是正定的, 则 B 也正定;

(2) 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 $A^{\frac{1}{2}}$.

(留作课外自习, 证明可参见相关资料, 如 [30])

1.2.4 向量范数与矩阵范数

我们首先给出一般线性空间上的范数的定义.

定义 1.7 (范数与赋范线性空间) 设 $\mathbb{S}$ 为数域 $\mathbb{F}$ ($\mathbb{F}$ 可以是 $\mathbb{R}$ 或 $\mathbb{C}$) 上的线性空间, 若对任意 $x \in \mathbb{S}$, 存在唯一实数与之对应, 记为 $\|x\|$, 满足:

(1) $\|x\| \geq 0$, 等号当且仅当 $x = 0$ 时成立; (正定性)

(2) $\|\alpha x\| = |\alpha| \cdot \|x\|, \forall\, \alpha \in \mathbb{F}$; (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 $\mathbb{S}$ 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 $\mathbb{S}$ 到 $\mathbb{R}_+ \bigcup \{0\}$ 的一元函数, 其中 $\mathbb{R}_+$ 表示所有正实数组成的集合.

例 1.5 $C[a,b]$ 上的常用范数: 设 $f(x) \in C[a,b]$,

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

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.

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

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

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

下面给出向量范数的一些性质. 为简单起见, 我们这里只考虑实数情形, 复数情形类似.

定理 1.8 (向量范数的连续性) 设 $\|\cdot\|$ 是 $\mathbb{R}^n$ 上的向量范数, 则 $f(x)\triangleq\|x\|$ 关于 x 的每个分量是连续的.

(证明留作课外练习, 利用范数的三角不等式)

定义 1.8 (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

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

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

定理 1.9 (向量范数的等价性) $\mathbb{R}^n$ 上的所有向量范数都是等价的, 特别地, 有

$$\|x\|_2\le\|x\|_1\le\sqrt{n}\,\|x\|_2,$$
$$\|x\|_\infty\le\|x\|_1\le n\,\|x\|_\infty,$$
$$\|x\|_\infty\le\|x\|_2\le\sqrt{n}\,\|x\|_\infty.$$

(板书, 只证明等价性, 三个不等式留作练习)

✍ 事实上, 有限维赋范线性空间上的所有范数都是等价的 [80].

矩阵范数

矩阵范数除了满足一般范数的要求外, 通常还要求满足相容性条件.

定义 1.9 (矩阵范数) 若函数 $f(X):\mathbb{R}^{n\times n}\to\mathbb{R}$ 满足

(1) $f(A)\ge0,\forall\,A\in\mathbb{R}^{n\times n}$, 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) $f(A+B)\le f(A)+f(B),\forall\,A,B\in\mathbb{R}^{n\times n}$;

(4) $f(AB) \le f(A)f(B), \forall\, A, B \in \mathbb{R}^{n\times n}$;

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 1.10 (算子范数) 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. (板书)

✍ 对于算子范数, 我们可以立即得到下面的性质: 设 $A \in \mathbb{R}^{n\times n}$, $x \in \mathbb{R}^n$, 则

$$\|Ax\| \le \|A\| \cdot \|x\|. \tag{1.2}$$

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.

✍ 如果存在矩阵范数和向量范数满足 (1.2), 则称它们是**相容**的.

例 1.7 $\mathbb{R}^{n\times n}$ 上常见的矩阵范数:

- p-范数 (算子范数)

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

- Frobenius 范数, 简称 F-范数

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

定理 1.11 可以证明:

(1) 1-范数 (也称为 列范数): $\|A\|_1 = \max\limits_{1\le j\le n} \left(\sum\limits_{i=1}^{n} |a_{ij}|\right)$;

(2) ∞-范数 (也称为 行范数): $\|A\|_\infty = \max\limits_{1\le i\le n} \left(\sum\limits_{j=1}^{n} |a_{ij}|\right)$;

(3) 2-范数 (也称为 谱范数): $\|A\|_2 = \sqrt{\rho(A^{\mathsf{T}}A)}$;

(4) F-范数: $\|A\|_F = \sqrt{\mathrm{tr}(A^{\mathsf{T}}A)}$.

(板书, 以 ∞-范数和 2-范数为例, 1-范数, F-范数留作练习)

矩阵范数的一些基本性质

(1) 对任意范数 $\|\cdot\|$, 有 $\|I\| \geq 1$, 对任意的算子范数 $\|\cdot\|$, 有 $\|I\| = 1$;
(2) 对任意范数 $\|\cdot\|$, 有 $\|A^k\| \leq \|A\|^k$;
(3) $\|A^{\mathsf{T}}\|_2 = \|A\|_2$, $\|A^{\mathsf{T}}\|_1 = \|A\|_\infty$;
(4) 若 A 是正规矩阵, 则 $\|A\|_2 = \rho(A)$, 特别地, 若 A 是对称矩阵, 则 $\|A\|_2 = \rho(A)$;
(5) F-范数不是算子范数;
(6) $\|\cdot\|_2$ 和 $\|\cdot\|_F$ 是 **酉不变范数**, 即对任意正交矩阵 (或酉矩阵) U, V, 有

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

(留作课外自习)

例 1.8 设 $A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$, 计算 $\|A\|_1, \|A\|_2, \|A\|_\infty, \|A\|_F$.

(板书, $6, \sqrt{15+\sqrt{221}}, 7, \sqrt{30}$)

✍ 计算 2-范数时需要求谱半径, 因此通常比计算 1-范数和 ∞-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 1.12 (矩阵范数的等价性) $\mathbb{R}^{n\times n}$ 上的所有范数都是等价的, 特别地, 有

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

(板书, 等价性证明略, 只证明三个不等式, 利用矩阵特征值与迹的关系式)

定理 1.13 (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. (留作课外自习)

定理 1.14 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

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

(板书)

谱半径与范数之间的关系

定理 1.15 (谱半径与范数的关系) 设 $A \in \mathbb{R}^{n\times n}$, 则

(1) 对任意算子范数, 有 $\rho(A) \le \|A\|$;

(2) 反之, 对任意 $\varepsilon > 0$, 都存在一个算子范数 $\|\cdot\|_\varepsilon$, 使得 $\|A\|_\varepsilon \le \rho(A) + \varepsilon$, 其中范数 $\|\cdot\|_\varepsilon$ 依赖于 A 和 ε. 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) 可参考 [71]))

✍ 事实上, 上述定理中的结论 (1) 对任意矩阵范数都成立, 见练习 **??**.

推论 1.16 设 $A \in \mathbb{R}^{n\times n}$, 则 $\|A\|_2^2 \le \|A\|_1 \cdot \|A\|_\infty$, 且

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

(留作练习)

推论 1.17 设 $A \in \mathbb{R}^{n\times n}$, 则对任意矩阵范数 $\|\cdot\|$, 有

$$\rho(A) = \lim_{k\to\infty} \left\|A^k\right\|^{\frac{1}{k}}. \tag{1.3}$$

等式 (1.3) 有时也用来定义矩阵的谱半径.

1.2.5 内积与内积空间

定义 1.10 (内积与内积空间) 设 $\mathbb{S}$ 是数域 $\mathbb{F}$ ($\mathbb{R}$ 或 $\mathbb{C}$) 上的一个线性空间, 定义一个从 $\mathbb{S}\times\mathbb{S}$ 到 $\mathbb{F}$ 的代数运算, 记为 "$(\cdot,\cdot)$", 即对任意 $x, y \in \mathbb{S}$, 都存在唯一的 $f \in \mathbb{F}$, 使得 $f = (x, y)$. 如果该运算满足

(1) $(y, x) = \overline{(x, y)}$, $\quad \forall\, x, y \in \mathbb{S}$;

(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) $(x, x) \ge 0$, 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**.

✍ $\overline{(x, y)}$ 表示 (x, y) 的共轭. 当 $\mathbb{F} = \mathbb{R}$ 时, 条件 (1) 即为 $(y, x) = (x, y)$.

✍ 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:

- $(x, \beta y) = \bar{\beta}(x, y), \forall\, \beta \in \mathbb{F}$
- $(x, y + z) = (x, y) + (x, z)$

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

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

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**欧氏内积**.

例 1.10 考虑线性空间 $\mathbb{C}^n$, 对任意 $x = [x_1, x_2, \dots, x_n]^{\mathsf{T}} \in \mathbb{C}^n$, $y = [y_1, y_2, \dots, y_n]^{\mathsf{T}} \in \mathbb{C}^n$, 定义

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

则 (x, y) 是 $\mathbb{C}^n$ 上的内积, 因此 $\mathbb{C}^n$ 是一个酉空间.

✍ 需要指出的是, 同一个线性空间中可以定义多个内积, 参见练习 ⁇.

✍ 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, 我们一般采用 $\mathbb{R}^n$ 和 $\mathbb{C}^n$ 上标准内积.

例 1.11 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

$$(f, g) \triangleq \int_a^b f(x) g(x)\, \mathrm{d}x,$$

容易验证, (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 1.11 设 $\mathbb{S}$ 是内积空间, $x, y \in \mathbb{S}$, 若 $(x, y) = 0$, 则称 x 与 y 是**正交**的.

✍ 正交可以看作是垂直概念的推广.

定理 1.18 (Cauchy-Schwarz 不等式) 设 $\mathbb{S}$ 是实数域上的内积空间, 则对任意 $u, v \in \mathbb{S}$, 有

$$|(u, v)|^2 \leq (u, u) \cdot (v, v). \tag{1.4}$$

(板书)

☕ 思考: Cauchy-Schwarz 不等式 (1.4) 中, 等号成立的充要条件是什么?

✍ Cauchy-Schwarz 不等式在复数域中也成立.

定理 1.19 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

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

称为 $u_1, u_2, \dots, u_n$ 的 **Gram 矩阵**. (板书)

内积导出范数

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

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

则可以验证, $\|x\|$ 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 1.12 $\mathbb{R}^n$ 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.

带权内积

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 ω_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 1.12 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, $\int_a^b x^k \omega(x)\,\mathrm{d}x$ 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 $\int_a^b g(x)\omega(x)\,\mathrm{d}x = 0$, 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 ∞;

✍ 权函数与定义区间 $[a,b]$ 相关.

例 1.13 常见的权函数有

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

定义 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

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

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

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

特别地, 当 $\omega(x)\equiv 1$ 时, 内积导出范数为

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

即为 $f(x)$ 的 2-范数.

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 1.20 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

$$X^{-1}AX=\begin{bmatrix} J_1 & & & \\ & J_2 & & \\ & & \ddots & \\ & & & J_q \end{bmatrix}\triangleq J, \tag{1.5}$$

其中 J_i 的维数等于 λ_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 ν_i 为 λ_i 的几何重数, J_{ik} 为 (对应于 λ_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.

推论 1.21 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 1.22 设 $A \in \mathbb{C}^{n\times n}$, 则存在酉矩阵 $U \in \mathbb{C}^{n\times n}$ 使得

$$U^*AU = \begin{bmatrix} \lambda_1 & r_{12} & \cdots & r_{1n} \\ 0 & \lambda_2 & \cdots & r_{2n} \\ \vdots & & \ddots & \vdots \\ 0 & \cdots & 0 & \lambda_n \end{bmatrix} \triangleq R \quad \text{或} \quad A = URU^*, \tag{1.6}$$

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

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 1.23 设 $A \in \mathbb{C}^{n\times n}$, 则

- A 是正规矩阵当且仅当 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 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, 我们在计算一个实矩阵的特征值时, 希望尽可能地避免复数运算. 此时, 我们就需要用到下面的**实 Schur 标准型** (或**拟 Schur 标准型**).

定理 1.24 设 $A \in \mathbb{R}^{n\times n}$, 则存在正交矩阵 $Q \in \mathbb{R}^{n\times n}$, 使得

$$Q^{\mathsf{T}}AQ = T, \tag{1.7}$$

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 1×1 或 2×2 的块矩阵. 若对角块是 1×1 的, 则其就是 A 的一个特征值, 若对角块是 2×2 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])

1.3 数值计算中的误差

数值方法的特点之一就是所求得的解是近似解, 通常总是存在一定的误差 (error). 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 [70]:

- **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, 数学模型与实际问题之间总会存在一定的误差.
- **数据误差**: 模型中往往包含各种数据或参量, 这些数据一般都是通过观测、测量、实验或计算得到的, 也会存在一定的误差.
- **截断误差** (truncation error): 也称**方法误差**, 是指对数学模型进行数值近似时产生的误差.
- **舍入误差** (roundoff error / rounding error): 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数据误差, 主要研究截断误差和舍入误差对计算结果的影响.

例 1.14 近似计算 $\int_0^1 e^{-x^2}\,\mathrm{d}x$ 的值.

解. 这里我们利用 Taylor 展开, 即

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

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

精度是数值计算中的一个非常重要的概念, 衡量精度通常依靠两个指标, 一个是绝对误差, 一个是相对误差.

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 (absolute error), 简称误差. 若存在 $\varepsilon > 0$ 使得

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

则称 ε 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 ε.

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**.

1.3.2 相对误差和相对误差限

当精确值比较大时, 绝对误差能较好地表示近似值的精确程度, 比如 2021.38 ± 0.01 表示有 5 位有效数字; 但当精确值比较小时, 情况就不一样了, 比如 0.004 ± 0.01, 没有任何有效数字. 因此, 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.

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 (relative error). 若存在 $\varepsilon_r > 0$ 使得

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

则称 ε_r 为相对误差限.

✍ 由于精确值 x 通常是未知的, 因此我们也可以采用下面的方式来定义相对误差:

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

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

关于相对误差和相对误差限的几点说明

- 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”, 这样所的得到的数字都是有效数字 (significant digits).

例 1.15 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, 近似值 $x_3 = 3.1415$ 有 4 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, 我们有下面的结论.

定理 1.25 *设 $\tilde{x}$ 是 x 的近似值, 若 $\tilde{x}$ 可表示为*

$$\tilde{x} = \pm 0.a_1a_2\cdots a_n\cdots \times 10^m,$$

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

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

则 $\tilde{x}$ 恰好有 n 位有效数字.

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

在上面的例 1.15 中, x_1 可写为 $x_1 = 0.314 \times 10^1$, 其与 π 的误差满足

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

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

例 1.16 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:

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

187.93, 0.037856, 8.0000

✍ 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.

推论 1.26 *设 $\tilde{x}$ 是 x 的近似值, 若 $\tilde{x}$ 可表示为*

$$\tilde{x} = \pm 0.a_1a_2\cdots a_n\cdots \times 10^m,$$

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

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

则 $\tilde{x}$ 至少有 n 位有效数字.

例 1.17 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?

解. 直接计算可得

$$0.5\times10^{-4}<|x_1-x|=0.818...\times10^{-4}\le0.5\times10^{-3},$$
$$0.5\times10^{-5}<|x_2-x|=0.181...\times10^{-4}\le0.5\times10^{-4}.$$

又 $x_1=0.27182\times10^1$ 和 $x_2=0.27183\times10^1$, 因此, x_1 有 $1-(-3)=4$ 位有效数字, x_2 有 $1-(-4)=5$ 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 1.27 (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*

$$\tilde{x}=\pm0.a_1a_2\dots a_n\dots\times10^m,$$

其中 a_i 是 0 到 9 中的数字, 且 $a_1\neq0$. 若 $\tilde{x}$ 具有 n 位有效数字, 则其相对误差满足

$$|\epsilon_r|\le\frac{1}{2a_1}\times10^{-n+1}.$$

反之, 若 $\tilde{x}$ 的相对误差满足

$$|\epsilon_r|\le\frac{1}{2(a_1+1)}\times10^{-n+1}.$$

则 $\tilde{x}$ 至少有 n 位有效数字. (板书)

✍ 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.

机器精度 (unit roundoff)

在进行数值计算时, 舍入误差不可避免. 对于不同的数据类型, 相对误差也是不一样的. 目前科学计算中常用的浮点数是双精度数和单精度数, 相对误差分别大约是 $\varepsilon_u\approx1.1\times10^{-16}$ 和 $\varepsilon_u\approx6.0\times10^{-8}$, 即两个浮点数做数值计算 (加减乘除等), 实际计算值 $\tilde{f}$ 与精确值 f 满足 $|\tilde{f}-f|\le\varepsilon_u|f|$, 工程中也记为 $f=\tilde{f}(1\pm\varepsilon_u)$.

1.3.4 误差估计的基本方法

✍ 这里需要再次强调的是, 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.

设 $\tilde{x}$ 是近似值, 为了表述方便, 我们用 $\varepsilon(\tilde{x})$ 表示其误差限.

四则运算

设 $\tilde{x}_1$ 和 $\tilde{x}_2$ 的误差限分别为 $\varepsilon(\tilde{x}_1)$ 和 $\varepsilon(\tilde{x}_2)$, 则可以验证下面的结论成立:

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

$$\varepsilon(\tilde{x}_1\tilde{x}_2) \leq |\tilde{x}_2|\varepsilon(\tilde{x}_1) + |\tilde{x}_1|\varepsilon(\tilde{x}_2) + \varepsilon(\tilde{x}_1)\varepsilon(\tilde{x}_2) \approx |\tilde{x}_2|\varepsilon(\tilde{x}_1) + |\tilde{x}_1|\varepsilon(\tilde{x}_2),$$
$$\varepsilon\left(\frac{\tilde{x}_1}{\tilde{x}_2}\right) \lessapprox \frac{|\tilde{x}_2|\varepsilon(\tilde{x}_1) + |\tilde{x}_1|\varepsilon(\tilde{x}_2)}{|\tilde{x}_2|^2}.$$

由此可知, 数据的误差在运算过程中会累积和传递, 同时也可能会相消.

单变量可微函数

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

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

由于 $|x - \tilde{x}|$ 相对较小, 所以当 $|f''(x)|$ 与 $|f'(x)|$ 的比值不是很大时, 我们可以忽略二次项, 即

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

因此, 可得函数值的误差限

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

例 1.18 设 $x > 0$, x 的相对误差是 δ, 试估计计算 $\ln x$ 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

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

即计算 $\ln x$ 可能产生的误差约为 δ. □

多变量可微函数

关于多元函数 $f(x_1, x_2, \ldots, x_n)$, 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 $\tilde{x} = [\tilde{x}_1, \tilde{x}_2, \ldots, \tilde{x}_n]^\intercal$ 是 $x = [x_1, x_2, \ldots, x_n]^\intercal$ 的近似值.

例 1.19 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 0.2m 和 0.1m. 试求面积 $S = xy$ 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

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

□

1.3.5 问题的适定性和算法的稳定性

数学问题的适定性

定义 1.16 如果一个数学问题满足

(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).

✍ 有时也称不适定的问题为不稳定 (unstable) 的 [44].

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 1.20 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 α 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 α 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. □

可微函数的条件数

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 $\left|\dfrac{x_* f'(x_*)}{f(x_*)}\right|$ 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

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

显然, 条件数越大, 计算所得到的函数值的相对误差就可能越大. 因此, 如果函数的条件数比较大, 那么我们就认为问题是病态的, 而且条件数越大问题病态就越严重.

关于病态问题的几点说明

- 病态是问题本身固有的性质, 与数值算法无关.
- 条件数是衡量问题是否病态的一个重要指标, 不同问题的条件数有着不同的定义, 例如矩阵的条件数见定义 2.1.
- 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, 或者能得到有效控制, 则称该算法是 **稳定** 的, 否则为 **不稳定**的.

例 1.21 近似计算 (Demo11_Stablity.m)

$$S_n = \int_0^1 \frac{x^n}{x+5}\,\mathrm{d}x, \quad n = 1, 2, \ldots, 8.$$

✍ 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)

1.3.6 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 1.22 计算 $\sqrt{9.01} - 3$, 计算过程中保留 3 位有效数字.

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:

$$\sqrt{x+\varepsilon} - \sqrt{x} = \frac{\varepsilon}{\sqrt{x+\varepsilon} + \sqrt{x}}$$

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

例 1.23 计算 $y=\left(\dfrac{\sqrt{101}-10}{\sqrt{101}+10}\right)^2$.

例 1.24 在 MATLAB 中用双精度数计算 (Demo12_Significance.m)

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

例 1.25 计算 $y=\ln 2$. (Demo13_ln.m)

(2) 避免数量级相差很大的数相除

当两个数量级相差很大的数进行除法运算时, 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, 否则会放大分子的误差.

✍ 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.

(3) 避免大数吃小数

比如 $(10^9+10^{-9}-10^9)/10^{-9}$, 在双精度环境下直接计算的话, 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 1.26 计算 $S=1+2+3+\cdots+100+10^{16}$. (Demo14_Sum.m)

(4) 简化计算

尽量减少运算次数, 从而减少误差的积累.

例 1.27 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.

✍ 在计算多项式的值时, 我们都是将多项式改写成

$$\begin{aligned}p(x)&=a_nx^n+a_{n-1}x^{n-1}+a_{n-2}x^{n-2}+\cdots+a_1x+a_0\\&=((\cdots((a_nx+a_{n-1})x+a_{n-2})x+\cdots)x+a_1)x+a_0.\end{aligned}$$

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 [32], 因此西方也称为 **Horner 算法**.

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

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

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, 54]. 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.

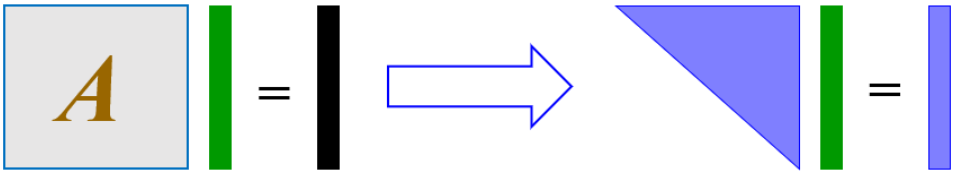
例 2.1 求解下面的线性方程组

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

将以上的做法推广到一般线性方程 $Ax = b$, 即

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

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, 也是当前求解中小规模线性方程组的首选方法, 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, 还可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

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

第 1 步: 消第 1 列.

设 $a_{11}^{(1)} \neq 0$, 计算 $l_{i1} = \dfrac{a_{i1}^{(1)}}{a_{11}^{(1)}}$, $i = 2, 3, \ldots, n$. 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 l_{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

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

第 2 步: 消第 2 列.

设 $a_{22}^{(2)} \neq 0$, 计算 $l_{i2} = \dfrac{a_{i2}^{(2)}}{a_{22}^{(2)}}$, $i = 3, 4, \ldots, n$. 依次将 $A^{(2)}$ 的第 i 行 $(i > 2)$ 减去第 2 行的

l_{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

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

依此类推, 经过 $k-1$ 步后, 可得新矩阵 $A^{(k)}$:

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

第 k 步: 消第 k 列.

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

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

依此类推, **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**

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

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 $a_{kk}^{(k)} \neq 0, k = 1,2,\ldots,n$, 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

$$D_1 \triangleq a_{11} \neq 0, \quad D_k \triangleq \det\left(\begin{bmatrix} a_{11} & \cdots & a_{1k} \\ \vdots & \ddots & \vdots \\ a_{k1} & \cdots & a_{kk} \end{bmatrix}\right) \neq 0, \quad k = 2, 3, \ldots, n.$$

✍ 事实上, 如果 A 的所有顺序主子式都不为零, 则

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

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.

2.1.2 运算量统计

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

$$\begin{aligned} l_{ik} &= a_{ik}^{(k)} / a_{kk}^{(k)}, \quad i = k+1, \ldots, n \quad \to \quad n-k \text{ 次} \\ a_{ij}^{(k+1)} &= a_{ij}^{(k)} - l_{ik} a_{kj}^{(k)}, \quad i, j = k+1, \ldots, n \quad \to \quad (n-k)^2 \text{ 次} \\ b_i^{(k+1)} &= b_i^{(k)} - l_{ik} b_k^{(k)}, \quad i = k+1, \ldots, n \quad \to \quad n-k \text{ 次} \end{aligned}$$

所以整个消去过程的乘除运算为

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

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

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

把加减乘除运算合在一起, 则为

$$\frac{2n^3}{3} + \frac{3n^2}{2} - \frac{7n}{6} = \frac{2n^3}{3} + \mathcal{O}(n^2).$$

✍ 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. 一般情况下, 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 αABx, 其中 α 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 αx, 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.

2.2 矩阵分解法

2.2.1 矩阵 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 是上三角矩阵. 记 $L \triangleq \tilde{L}^{-1}$, 则

$$A = LU,$$

这就是有名的矩阵 **LU 分解**.

矩阵分解

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. 通过矩阵分解, 可以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.

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. 下面我们研究 L 具有什么样的特殊结构或者特殊性质.

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 消去过程更新公式 (2.3) 可得

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

其中

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

令 $k = 1, 2, \ldots, n-1$, 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:

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

(留作练习)

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

$$A=LU. \tag{2.5}$$

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, $k=1,2,\ldots,n$. (板书)

用 LU 分解求解线性方程组

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 分解的算法实现

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** $i=k$ to n **do**
4: $u_{ki}=a_{ki}$ % 计算 U 的第 k 行
5: **end for**
6: **for** $i=k+1$ to n **do**
7: $l_{ik}=a_{ik}/a_{kk}$ % 计算 L 的第 k 列

```
        for j = k + 1 to n do
            a_ij = a_ij − l_ik u_kj    % 更新 A(k+1:n, k+1:n)
        end for
    end for
end for
```

✍ LU 分解有时也称为 **Doolittle 分解**, 除了上面的实现方式外, 也可以通过待定系数法计算.

矩阵 L 和 U 的存储

当 A 的第 i 列 (严格下三角部分) 被用于计算 L 的第 i 列后, 在后面的计算中不再被使用. 而 A 的第 i 行 (上三角部分) 更新后就是 U 的第 i 行. 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 (严格下三角部分, L 的对角线全部为 1, 不需要存储), 将 U 的第 i 行存放在 A 的第 i 行 (上三角部分), 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 的绝对下三角部分.

这样, 算法就更简洁高效, 描述如下.

算法 2.2. LU 分解 (用 A 存储 L 和 U)

```
for k = 1 to n − 1 do
    for i = k + 1 to n do
        a_ik = a_ik / a_kk
        for j = k + 1 to n do
            a_ij = a_ij − a_ik a_kj
        end for
    end for
end for
```

2.2.2 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. 但在 Gauss 消去法的计算过程中, 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 LU 分解的第 k 步之前, 插入下面的选主元过程.

① 选取 **列主元**: $\left|a_{i_k,k}^{(k)}\right|=\max\limits_{k\le i\le n}\left\{\left|a_{i,k}^{(k)}\right|\right\}$

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 ($i_k\ge k$), 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**. 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 2.3. 列主元 Gauss 消去法

for $k=1$ to $n-1$ **do**
  $a_{i_k,k}=\max_{k\le i\le n}|a_{i,k}|$ % 选列主元
  if $i_k\neq k$ **then**
    for $j=1$ to n **do**
      $a_{tmp}=a_{i_k,j}, a_{i_k,j}=a_{k,j}, a_{k,j}=a_{tmp}$ % 交换 A 的第 i_k 行与第 k 行
    end for
    $b_{tmp}=b_{i_k}, b_{i_k}=b_k, b_k=b_{tmp}$ % 交换 b 的第 i_k 与第 k 个分量
  end if
  for $i=k+1$ to n **do**
    $a_{ik}=a_{ik}/a_{kk}$ % 计算 L 的第 i 列
    for $j=k+1$ to n **do**

```
        a_ij = a_ij − a_ik a_kj    % 更新 A(k+1:n, k+1:n)
      end for
      b_i = b_i − a_ik b_k
    end for
  end for
  x_n = b_n / a_nn    % 向后回代求解 Ux = y
  for i = n − 1 to 1 do
    for j = i + 1 to n do
      b_i = b_i − a_ij x_j
    end for
    x_i = b_i / a_ii
  end for
```

列主元 Gauss 消去法对应的矩阵分解称为**列主元 LU 分解**, 记为 **PLU** .

定理 2.5 (列主元 LU 分解, PLU) *若矩阵 $A \in \mathbb{R}^{n\times n}$ 非奇异, 则存在置换矩阵 P, 使得*

$$PA = LU, \tag{2.6}$$

其中 L 为单位下三角矩阵, U 为上三角矩阵. (板书)

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. (板书)

✍ 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.

列主元 LU 分解的算法实现

设

$$PA = LU,$$

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 $\{1, 2, \ldots, n\}$ 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 $p=[1,2,\ldots,n]$. 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 2.4. PLU: 列主元 LU 分解或部分选主元 LU 分解

$p=[1,2,\ldots,n]$ % 用于记录置换矩阵
for $k=1$ to $n-1$ **do**
  $a_{i_k,k}=\max_{k\le i\le n}|a_{i,k}|$ % 选列主元
  if $i_k\neq k$ **then**
    for $j=1$ to n **do**
      $a_{tmp}=a_{i_k,j}, a_{i_k,j}=a_{k,j}, a_{k,j}=a_{tmp}$ % 交换 A 的第 i_k 行与第 k 行
    end for
    $p_{tmp}=p_{i_k}, p_{i_k}=p_k, p_k=p_{tmp}$ % 更新置换矩阵
  end if
  for $i=k+1$ to n **do**
    $a_{ik}=a_{ik}/a_{kk}$ % 计算 L 的第 i 列
    for $j=k+1$ to n **do**
      $a_{ij}=a_{ij}-a_{ik}a_{ij}$ % 更新 $A(k+1:n,k+1:n)$
    end for
  end for
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 2.5. PLU 分解求解线性方程组

计算 A 的 PLU 分解 (此处省略, 可参见算法 2.4)
$y_1=b_{p_1}$ % 向前回代求解 $Ly=Pb$
for $i=2$ to n **do**
  for $j=1$ to $i-1$ **do**
    $b_{p_i}=b_{p_i}-a_{ij}y_j$
  end for
  $y_i=b_{p_i}$
end for
$x_n=b_{p_n}/a_{nn}$ % 向后回代求解 $Ux=y$
for $i=n-1$ to 1 **do**
  for $j=i+1$ to n **do**
    $y_i=y_i-a_{ij}x_j$
  end for
  $x_i=y_i/a_{ii}$

15: **end for**

✍ 算法 2.5 可用于计算矩阵的逆, 也可用于计算矩阵的行列式.

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**: $|a_{i_k,j_k}^{(k)}| = \max\limits_{k\le i,j\le n}\left\{|a_{i,j}^{(k)}|\right\}$

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

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.

其他类似分解

除了 LU 分解外, 常用的还有 **Crout 分解**, 即

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

其中 $\tilde{L}$ 为非奇异下三角矩阵, $\tilde{U}$ 为单位上三角矩阵.

另外还有 **LDR 分解**, 即

$$A = LDR,$$

其中 L 为单位下三角矩阵, R 为单位上三角矩阵, D 为对角矩阵.

以上两种分解与 LU 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.

2.2.3 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 2.6 (Cholesky 分解) 设 $A \in \mathbb{R}^{n\times n}$ 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

$$A = LL^{\mathsf{T}}.$$

该分解称为 **Cholesky 分解**. (板书)

✍ Cholesky 分解仅针对对称正定矩阵成立.

如何计算 Cholesky 分解

我们使用待定系数法. 设

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

直接比较等式两边的元素, 可得一般公式

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

由此可得计算公式

$$\begin{cases} l_{jj} = \left(a_{jj} - \sum\limits_{k=1}^{j-1} l_{jk}^2 \right)^{\frac{1}{2}}, \\ l_{ij} = \dfrac{1}{l_{jj}} \left(a_{ij} - \sum\limits_{k=1}^{j-1} l_{ik}l_{jk} \right), \quad j = 1, 2, \ldots, n,\ i = j, j+1, \ldots, n. \end{cases}$$

根据这个公式即可得下面的算法描述.

1: **for** $j = 1$ to n **do**
2: $\quad l_{jj} = \left(a_{jj} - \sum\limits_{k=1}^{j-1} l_{jk}^2 \right)^{\frac{1}{2}}$ % 先计算对角线元素
3: $\quad$ **for** $i = j+1$ to n **do**
4: $\quad\quad l_{ij} = \left(a_{ij} - \sum\limits_{k=1}^{j-1} l_{ik}l_{jk} \right) / l_{jj}$ % 计算 L 的第 j 列
5: $\quad$ **end for**
6: **end for**

实际计算时我们可以将 L 存放在 A 的下三角部分, 具体算法如下.

算法 2.6. Cholesky 分解

1: **for** $j = 1$ to n **do**
2: $\quad$ **for** $k = 1$ to $j-1$ **do**
3: $\quad\quad a_{jj} = a_{jj} - a_{jk}^2$
4: $\quad$ **end for**
5: $\quad a_{jj} = \sqrt{a_{jj}}$
6: $\quad$ **for** $i = j+1$ to n **do**
7: $\quad\quad$ **for** $k = 1$ to $j-1$ **do**

```
        a_ij = a_ij − a_ik a_jk
    end for
    a_ij = a_ij / a_jj
  end for
end for
```

✍ Cholesky 分解算法的加减乘除运算量为 $\frac{1}{3}n^3+\mathcal{O}(n^2)$, 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 2.7. 平方根法: Cholesky 分解求解线性方程组

```
计算 Cholesky 分解 (此处省略, 参见算法 2.6)
y_1 = b_1 / a_11    % 向前回代求解 Ly = b
for i = 2 to n do
  for j = 1 to i − 1 do
    b_i = b_i − a_ij y_j
  end for
  y_i = b_i / a_ii
end for
x_n = b_n / a_nn    % 向后回代求解 L^T x = y
for i = n − 1 to 1 do
  for j = i + 1 to n do
    y_i = y_i − a_ji x_j
  end for
  x_i = y_i / a_ii
end for
```

LDL$^\intercal$ 分解与改进的平方根法

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

$$A = LDL^\intercal,$$

其中 L 是单位下三角矩阵, D 是对角线元素全为正的对角矩阵. 这个分解也称为 **LDL$^\intercal$ 分解**. 由待定系数法, 设

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

比较等式两边的元素, 可得

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

由此可得计算公式

---

for $j = 1$ to n **do**
  $d_j = a_{jj} - \sum_{k=1}^{j-1} d_k l_{jk}^2$ % 先计算 D 的对角线元素
  for $i = j+1$ to n **do**
    $l_{ij} = \left(a_{ij} - \sum_{k=1}^{j-1} d_k l_{ik} l_{jk} \right) / d_j$ % 计算 L 的第 j 列
  end for
end for

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 2.8. 改进的平方根法

for $j = 1$ to n **do**
  for $k = 1$ to $j-1$ **do**
    $a_{jj} = a_{jj} - a_{jk}^2 a_{kk}$
  end for
  for $i = j+1$ to n **do**
    for $k = 1$ to $j-1$ **do**
      $a_{ij} = a_{ij} - a_{ik} a_{kk} a_{jk}$
    end for
    $a_{ij} = a_{ij} / a_{jj}$
  end for

```
end for
y_1 = b_1    % 向前回代求解 Ly = b
for i = 2 to n do
    for j = 1 to i − 1 do
        b_i = b_i − a_ij y_j
    end for
    y_i = b_i
end for
x_n = b_n/a_nn    % 向后回代求解 DL^T x = y
for i = n − 1 to 1 do
    x_i = y_i/a_ii
    for j = i + 1 to n do
        x_i = x_i − a_ji x_j
    end for
end for
```

对于对称正定矩阵, 其对应的 LDL$^\intercal$ 分解中的 D 的对角线元素都是正的, 如果允许其对角线元素出现负数, 则可得下面的结论.

定理 2.7 *若 $A \in \mathbb{R}^{n\times n}$ 对称, 且所有顺序主子式都不为 0, 则 A 可唯一分解为*

$$A = LDL^\intercal,$$

其中 L 为单位下三角矩阵, D 为对角矩阵. (留作课外自习)

思考: 若 $A \in \mathbb{R}^{n\times n}$ 存在 LDL$^\intercal$ 分解, 则 A 一定是对称的. 反之, 如果 A 对称, 则 A 一定存在 LDL$^\intercal$ 分解吗?

2.2.4 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

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

且

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

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)

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

由待定系数法, 我们可以得到递推公式:

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

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 2.8 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) $0 < |c_i| \leq |b_i| - |a_{i-1}| < |\alpha_i| < |b_i| + |a_{i-1}|, \quad i = 2,3,\dots,n.$ (板书)

由定理 2.8 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 2.9. 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: $\quad \alpha_i = b_i - a_{i-1}\beta_{i-1}$
6: $\quad \beta_i = c_i/\alpha_i$
7: $\quad y_i = (f_i - a_{i-1}y_{i-1})/\alpha_i$
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, α_i 可以不用存储. 但 β_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

$$A = \begin{bmatrix} b_1 & c_1 & & \\ a_1 & \ddots & \ddots & \\ & \ddots & \ddots & c_{n-1} \\ & & a_{n-1} & b_n \end{bmatrix} = \begin{bmatrix} 1 & & & \\ \gamma_1 & 1 & & \\ & \ddots & \ddots & \\ & & \gamma_{n-1} & 1 \end{bmatrix} \begin{bmatrix} \alpha_1 & c_1 & & \\ & \alpha_2 & \ddots & \\ & & \ddots & c_{n-1} \\ & & & \alpha_n \end{bmatrix}. \tag{2.10}$$

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 γ_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 (2.10).

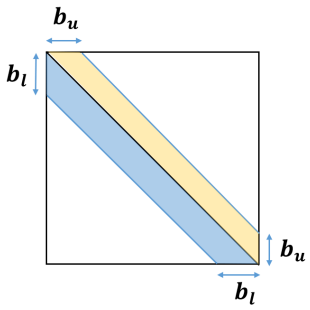
2.2.5 带状线性方程组

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

$$a_{ij} = 0.$$

其形状如右图所示:



对于带状矩阵, 其 LU 分解有如下性质:

定理 2.9 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 2.10 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 $PA = LU$ 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 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

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, 因此我们需要进行误差分析. 除了数据误差和截断误差外, 在用计算机进行数值计算时, 每次运算 (加减乘除等) 还会产生舍入误差. 对于大规模问题, 实际运算次数往往非常巨大, 因此直接分析舍入误差比较复杂, 而且得到的结果往往不一定能反映实际计算情况. 当前一种比较实用的误差分析方法是将舍入误差看作是对原始数据的扰动, 即将其归结到数据误差中, 这样就可以在很大程度上简化误差分析过程. 这种误差分析方法称为**向后误差分析**.

2.3.1 矩阵条件数

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 $A = \begin{bmatrix} 1 & 1 \\ 1 & 1.0001 \end{bmatrix}$, $b = [2, 2]^\mathsf{T}$. 可求得解为 $x = [2, 0]^\mathsf{T}$.

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

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

$$\kappa_2(A) = \frac{\max\limits_{1\le i\le n} |\lambda_i|}{\min\limits_{1\le i\le n} |\lambda_i|}.$$

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

✍ 一般情况下, 如果矩阵非对角线元素相对比较大, 则矩阵很有可能是病态的.

如果 A 对称, 则其谱条件数由特征值决定, 但对于一般矩阵, 该结论并不成立.

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 的指数函数. 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
$\kappa_1(A)$	2.1×10^2	2.3×10^4	2.0×10^6	1.5×10^8	1.1×10^{10}
$\kappa_2(A)$	6.3×10	7.6×10^3	6.8×10^5	5.5×10^7	3.9×10^9

引理 2.11 条件数具有以下性质:

- $\kappa(A)\geq1,\quad\forall\,A\in\mathbb{R}^{n\times n}$.
- 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
$$\kappa_2(QA)=\kappa_2(A)=\kappa_2(AQ).$$

(留作练习)

2.3.2 条件数与扰动分析

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 δb. 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

$$\|\delta x\| \leq \|A^{-1}\| \cdot \|\delta b\|. \tag{2.11}$$

又由 $Ax_* = b$ 可知

$$\|b\| \leq \|A\| \cdot \|x_*\|, \quad \text{即} \quad \frac{1}{\|x_*\|} \leq \frac{\|A\|}{\|b\|}. \tag{2.12}$$

由 (2.11) 和 (2.12) 两边相乘可得

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

定理 2.12 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 δb, 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 δA, 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 1.14 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

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

由定理 1.14 可知

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

当 $\|\delta A\| \to 0$ 时, 我们可得

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

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），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 δx 与 $\tilde{x}$ 之间的关系.

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），则 δx 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

$$\frac{\|\delta x\|}{\|\tilde{x}\|}\leq\kappa(A)\frac{\|\delta A\|}{\|A\|}\tag{2.13}$$

(板书)

✍ 上面的结论具有理论指导作用, 但如果无法获得 δA 和 δb 的大小, 则很难用来估计 δx 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 δA 和 δb 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

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

可以验证 H_n 是对称正定的. 通过计算可知

$$\kappa_\infty(H_2)=27,\quad\kappa_\infty(H_3)=748,\quad\kappa_\infty(H_4)=28375,\quad\kappa_\infty(H_{10})\approx3.5\times10^{13}.$$

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 $x_* = [1, 1, \ldots, 1]^\mathsf{T}$, 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.

n	5	10	20	30
$\|\tilde{x} - x_*\|_\infty$	2.3×10^{-11}	6.2×10^{-4}	> 7.8	> 42.8
$\kappa_\infty(H_n)$	9.4×10^{5}	3.5×10^{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 矩阵元素缩放或平衡 (Scaling or equilibration)

在求解线性方程组时, 先对系数矩阵元素进行适当的缩放是一种很常用的技术手段.

一方面, 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. 在对矩阵元素进行缩放时, 大约需要 $\mathcal{O}(n^2)$ 运算量, 通常不会产生较大的舍入误差.

另一方面, 由前面的扰动分析可知, 矩阵条件数对数值计算的误差有着很大的影响, 是衡量问题是否病态的一个重要指标. 在实际计算中, 如果遇到病态问题, 我们希望能够通过一些简单的方法降低其条件数. 其中一类简单有效的方法就是在矩阵两边同时乘以一个对角矩阵, 即寻找对角矩阵 D_r 和 D_c, 使得 $D_r^{-1}AD_c^{-1}$ 的条件数达到最小 (或者尽可能小). 显然, 寻找这样的对角矩阵是非常困难的, 目前仍然是一个开放问题 [51].

一种比较可行的实用方案是使得缩放后的矩阵的每行或每列具有相同的 p-范数. 比如取 $D_r = \mathrm{diag}(d_1, d_2, \ldots, d_n)$, 其中 $d_i = \sum_{j=1}^{n} |a_{ij}|$, 这样 $D_r^{-1}A$ 的每行的 1-范数都一样, 但没法使得所有列也具有相同的范数.

如果矩阵

$$D_r^{-1}AD_c^{-1}$$

的所有行和所有列都具有相同 (或近似相同) 的范数, 则称为 A 的**均衡化** (equilibration). 有学者提出了一些迭代方法对 A 进行均衡化. 更多信息可参见相关资料.

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

$$Az = b - A\tilde{x} = r. \tag{2.14}$$

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 (2.14) 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的迭代改进法.

算法 2.10. 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 $\mathcal{O}(n^2)$. 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 $P^{\mathsf{T}}LU$, 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 ε_u 表示机器精度.

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, 材料与结构力学, 信号与图像处理, 机器学习和数据科学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**线性最小二乘问题**的三种常用求解方法 (直接法, 矩阵分解法): 正规方程 Cholesky 分解法, QR 分解法和 SVD 分解法. 一般来说, Cholesky 分解法是最快的, 特别是当 A 的条件数较小时, 该方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, 本讲义中我们将线性最小二乘问题简称为**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 [75]

3.1 问题介绍

线性最小二乘问题就是求解下面的最值问题:

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

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 Hessen 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

$$A^\mathsf{T}Ax - A^\mathsf{T}b = 0.$$

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 $A^\mathsf{T}A$ 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). 这时我们通常寻求满足一定条件的解. 比如常见的最小范数解, 即所有解中范数最小的解, 此时原问题就转化为下面的**约束优化问题**

$$\min_{Ax=b} \frac{1}{2}\|x\|_2^2, \tag{3.2}$$

对应的 **Lagrange 函数**为

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

其中 $\lambda = [\lambda_1, \lambda_2, \dots, \lambda_m]^\mathsf{T}$ 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:

$$\frac{\partial \mathcal{L}}{\partial x} = x + A^\mathsf{T}\lambda = 0, \quad \frac{\partial \mathcal{L}}{\partial \lambda} = Ax - b = 0.$$

写成矩阵形式为

$$\begin{bmatrix} I & A^\mathsf{T} \\ A & 0 \end{bmatrix} \begin{bmatrix} x \\ \lambda \end{bmatrix} = \begin{bmatrix} 0 \\ b \end{bmatrix}.$$

如果 A 满秩, 即 $\operatorname{rank}(A) = m$, 则系数矩阵非奇异 (见练习 **??**), 上述方程组存在唯一解.

本讲义主要讨论超定线性最小二乘问题的求解.

3.2 Householder 变换与 Givens 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. 完成这个转化的一个基本工具就是**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

$$H = I - \frac{2}{v^*v}vv^* = I - \frac{2}{\|v\|_2^2}vv^*, \quad 0 \neq v \in \mathbb{C}^n, \tag{3.3}$$

为 **Householder 变换** (或 **Householder 矩阵**, 或 **Householder 反射**), 向量 v 称为 **Householder 向量**. 我们通常将矩阵 (3.3) 记为 $H(v)$.

- Householder 矩阵是单位矩阵的秩-1 修正.
- Householder 变换也可以定义为

$$H = I - 2vv^*, \quad v \in \mathbb{C}^n \text{ 且 } \|v\|_2 = 1.$$

- Householder 矩阵由 Householder 向量唯一确定.

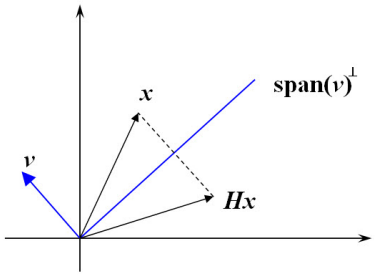
从几何上看, 一个 Householder 变换是一个关于超平面 $\mathrm{span}\{v\}^\perp$ 的反射. 由于 $\mathbb{C}^2 = \mathrm{span}\{v\} \oplus \mathrm{span}\{v\}^\perp$, 因此对任意一个向量 $x \in \mathbb{C}^n$, 都可写成

$$x = \frac{v^*x}{v^*v}v + y \triangleq \alpha v + y,$$

其中 $\alpha v \in \mathrm{span}\{v\}$, $y \in \mathrm{span}\{v\}^\perp$. 于是

$$Hx = x - \frac{2}{v^*v}vv^*x = x - 2\alpha v = -\alpha v + y,$$

即 Hx 与 x 在 $\mathrm{span}\{v\}^\perp$ 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 $\mathrm{span}\{v\}^\perp$ 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 Hermitian 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 $x^*y \in \mathbb{R}$. (板书)

✍ 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 $x = [x_1, x_2, \dots, x_n]^\mathsf{T} \in \mathbb{R}^n$ 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 $\alpha = -\|x\|_2$), $e_1 = [1, 0, \dots, 0]^\mathsf{T} \in \mathbb{R}^n$.

✍ 在后面的讨论中, 我们将定理中的向量 v 称为 x 对应的 Householder 向量.

设 $x = [x_1, x_2, \dots, x_n]^\mathsf{T} \in \mathbb{R}^n$ 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

$$v = x - \alpha e_1 = [x_1 - \alpha, x_2, \dots, x_n]^\mathsf{T}.$$

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

$$\alpha = -\operatorname{sign}(x_1) \cdot \|x\|_2. \tag{3.4}$$

事实上, 我们也可以取 $\alpha = \operatorname{sign}(x_1)\|x\|_2$, 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

$$\alpha = \operatorname{sign}(x_1)\|x\|_2, \quad v_1 = x_1 - \alpha = \frac{x_1^2 - \|x\|_2^2}{x_1 + \alpha} = \frac{-(x_2^2 + x_3^2 + \cdots + x_n^2)}{x_1 + \alpha}. \tag{3.5}$$

在 v_1 的两种计算方法 (3.4) 和 (3.5) 中, α 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 α 非负, 此时我们可以将这两种方法结合起来使用, 即:

$$v_1 = \begin{cases} x_1 - \alpha, & \text{if } \operatorname{sign}(x_1) < 0 \\ \dfrac{-(x_2^2 + x_3^2 + \cdots + x_n^2)}{x_1 + \alpha}, & \text{otherwise} \end{cases}$$

无论怎样计算 v, 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: 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 $2n$.

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)

$n = \texttt{length}(x)$ (here $\texttt{length}(x)$ denotes the dimension of x)
$\sigma = x_2^2 + x_3^2 + \cdots + x_n^2$
$v = x$
if $\sigma = 0$ **then**
    if $x_1 < 0$ **then**
        $v_1 = 2x_1, \beta = 2/v_1^2$
    else
        $v_1 = 0, \beta = 0$
    end if
else
    $\alpha = \sqrt{x_1^2 + \sigma}$ % $\alpha = \|x\|_2$
    if $x_1 < 0$ **then**
        $v_1 = x_1 - \alpha$
    else
        $v_1 = -\sigma/(x_1 + \alpha)$
    end if
    $\beta = 2/(v_1^2 + \sigma)$
end if

可以证明, 上述算法具有很好的数值稳定性 [62], 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, ε_u 是机器精度.

✍ 在实际计算时, 我们可以将向量 v 规范化, 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 $x \leftarrow x/\|x\|_2$

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, 我们这里这讨论实数域中的 Givens 变换. 设 $\theta \in [0, 2\pi]$, 我们称矩阵

$$G(i,j,\theta) = \begin{bmatrix} 1 & & & & & & \\ & \ddots & & & & & \\ & & c & & s & & \\ & & & \ddots & & & \\ & & -s & & c & & \\ & & & & & \ddots & \\ & & & & & & 1 \end{bmatrix} \in \mathbb{R}^{n\times n}, \qquad (i<j)$$

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.

定理 3.4 $G(i,j,\theta)$ 是正交矩阵, 且 $\det(G(i,j,\theta))=1$.

✍ Givens 变换可以看作是单位矩阵的一个秩-2 修正, 即

$$G(i,j,\theta) = I + [e_i, e_j] \begin{bmatrix} \cos(\theta)-1 & \sin(\theta) \\ -\sin(\theta) & \cos(\theta)-1 \end{bmatrix} [e_i, e_j]^{\mathsf{T}}.$$

✍ 因此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 (Givens 变换化零) 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 $c=\operatorname{sign}(x_1), s=0, r=|x_1|$;
- 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 $x=[a,b]^{\mathsf{T}} \in \mathbb{R}^2$, compute c and s such that $Gx=[r,0]^{\mathsf{T}}$ where $r=\|x\|_2$

1: **function** $[c,s]$ = **givens**(a,b)

2: **if** $b=0$ **then**

3: **if** $a \geq 0$ **then**

```
    c = 1,   s = 0
  else
    c = −1,   s = 0
  end if
else
  if |b| > |a| then   % 绝对值大的数作为分母
    τ = a/b,   s = sign(b)/√(1+τ²),   c = sτ
  else
    τ = b/a,   c = sign(a)/√(1+τ²),   s = cτ
  end if
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 ?? 和 ??), 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

$$\mathrm{fl}(\tilde{P}A) = P(A+E), \quad \mathrm{fl}(A\tilde{P}) = (A+F)P,$$

其中 $\|E\|_2 = \mathcal{O}(\varepsilon_u)\|A\|_2$, $\|F\|_2 = \mathcal{O}(\varepsilon_u)\|A\|_2$. (这里 ε_u 表示机器精度)

这表明对一个矩阵做 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 $P_1, P_2, \ldots, P_k$ 和 $Q_1, Q_2, \ldots, Q_k$, 则有

$$\mathrm{fl}(\tilde{P}_k \cdots \tilde{P}_1 A \tilde{Q}_1 \cdots \tilde{Q}_k) = P_k \cdots P_1 (A+E) Q_1 \cdots Q_k,$$

其中 $\|E\|_2 = \mathcal{O}(\varepsilon_u)(k\|A\|_2)$. 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

$$\mathrm{fl}(\tilde{X}A) = XA + E = X(A + X^{-1}E) \triangleq X(A+F),$$

其中 $\|E\|_2 = \mathcal{O}(\varepsilon_u)\cdot\|XA\|_2 \le \mathcal{O}(\varepsilon_u)\cdot\|X\|_2\cdot\|A\|_2$, 故

$$\|F\|_2 = \|X^{-1}E\|_2 \le \mathcal{O}(\varepsilon_u)\cdot\|X^{-1}\|_2\cdot\|X\|_2\cdot\|A\|_2 = \mathcal{O}(\varepsilon_u)\cdot\kappa_2(X)\cdot\|A\|_2,$$

因此, 舍入误差可能会被放大 $\kappa_2(X)$ 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) [30] 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

$$A = QR. \tag{3.6}$$

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 (3.6) 成立, 且此时 QR 分解唯一, 即 Q 和 R 都唯一.

证明. 设 $A = [a_1, a_2, \ldots, a_n] \in \mathbb{C}^{m\times n}$. 若 A 列满秩, 即 $\operatorname{rank}(A) = n$. 则 QR 分解 (3.6) 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

```
r_11 = ||a_1||_2
q_1 = a_1 / r_11
for j = 2 to n do
    q_j = a_j
    for i = 1 to j − 1 do
        r_ij = q_i^* a_j    % q_i^* 表示共轭转置
        q_j = q_j − r_ij q_i
    end for
    r_jj = ||q_j||_2
    q_j = q_j / r_jj
end for
```

由算法 3.3 可知: $a_1 = r_{11}q_1$,

$$a_j = r_{1j}q_1 + r_{2j}q_2 + \cdots + r_{jj}q_j = [q_1, q_2, \ldots, q_j]\begin{bmatrix} r_{1j} \\ r_{rj} \\ \vdots \\ r_{jj} \end{bmatrix}, \quad j = 2, 3, \ldots, n.$$

记 $Q = [q_1, q_2, \ldots, q_n]$, $R = [r_{ij}]_{n\times n}$, 其中

$$r_{ij} = \begin{cases} q_i^* a_j, & \text{for } i \leq j \\ 0, & \text{for } i > j \end{cases} \tag{3.7}$$

于是 Gram-Schmidt 过程可表示为

$$[a_1,a_2,\dots,a_n]=[q_1,q_2,\dots,q_n]\begin{bmatrix} r_{11} & r_{12} & \cdots & r_{1n} \\ & r_{22} & \cdots & r_{2n} \\ & & \ddots & r_{n-1,n} \\ & & & r_{nn} \end{bmatrix},\quad \text{即}\quad A=QR.$$

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 (3.7). 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 $\mathrm{rank}(A)=l<n$, 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

$$q_{i_1},q_{i_2},\dots,q_{i_l},\tilde{q}_1,\dots,\tilde{q}_{m-l}.$$

然后我们用 $\tilde{q}_1$ 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

$$A=Q_1R_1=Q_2R_2,$$

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

$$Q_1=Q_2R_2R_1^{-1}. \tag{3.8}$$

于是

$$1=\|Q_1\|_2=\|Q_2R_2R_1^{-1}\|_2=\|R_2R_1^{-1}\|_2.$$

又 R_1,R_2 均为上三角矩阵, 所以 $R_2R_1^{-1}$ 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

$$\frac{R_2(i,i)}{R_1(i,i)}\le\rho(R_2R_1^{-1})\le\|R_2R_1^{-1}\|_2\le 1,\quad i=1,2,\dots,n.$$

同理可证 $R_1(i,i)/R_2(i,i) \le 1$. 所以

$$R_1(i,i) = R_2(i,i), \quad i = 1, 2, \dots, n.$$

又 $\|Q_1\|_F^2 = \mathrm{tr}(Q_1^* Q_1) = n$, 所以由 (3.8) 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. □

✍ 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 $l = \mathrm{rank}(A)$. 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 $A \in \mathbb{C}^{m\times n}$ $(m \ge n)$, 且秩为 l $(0 \le l \le n)$, 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.

✍ QR 分解中的 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 $2mn^2$.

推论 3.9 (满秩分解) 设 $A \in \mathbb{C}^{m\times n}$ 且 $\mathrm{rank}(A) = r \le \min\{m, n\}$, 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 $Q = [q_1, \ldots, q_n] \in \mathbb{R}^{m\times n}$ and $R \in \mathbb{R}^{n\times n}$ such that $A = QR$

Set $R = [r_{ik}] = 0_{n\times n}$ (the $n \times n$ zero matrix)
if $a_1 = 0$ **then**
    $q_1 = 0$
else
    $r_{11} = \|a_1\|_2$
    $q_1 = a_1/\|a_1\|_2$
end if
for $k = 2$ to n **do**
    $q_k = a_k$
    for $i = 1$ to $k-1$ **do** % MGS 过程, 注意与 GS 的区别
        $r_{ik} = q_i^{\mathsf{T}} q_k$
        $q_k = q_k - r_{ik} q_i$
    end for
    if $q_k \neq 0$ **then**
        $r_{kk} = \|q_k\|_2$
        $q_k = q_k/r_{kk}$
    end if
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

$$H_1A=\left[\begin{array}{c|ccc} r_1 & \tilde{a}_{12} & \cdots & \tilde{a}_{1n} \\ \hline 0 & & & \\ \vdots & & \tilde{A}_2 & \\ 0 & & & \end{array}\right],$$

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

$$H_2H_1A=\left[\begin{array}{cc|ccc} r_1 & \tilde{a}_{12} & \tilde{a}_{13} & \cdots & \tilde{a}_{1n} \\ 0 & r_2 & \tilde{a}_{23} & \cdots & \tilde{a}_{2n} \\ \hline 0 & 0 & & & \\ \vdots & \vdots & & \tilde{A}_3 & \\ 0 & 0 & & & \end{array}\right].$$

不断重复上述过程. 这样, 我们就得到一系列的矩阵

$$H_k=\begin{bmatrix}I_{k-1} & 0\\ 0 & \tilde{H}_k\end{bmatrix},\quad k=1,2,3,4,\ldots,n-1$$

使得

$$H_{n-1}\cdots H_2H_1A=\begin{bmatrix} r_1 & \tilde{a}_{12} & \cdots & \tilde{a}_{1n} \\ 0 & r_2 & \cdots & \tilde{a}_{2n} \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & r_n \end{bmatrix}\triangleq R.$$

由于 Householder 变换都是正交矩阵, 因此 $H_1,H_2,\ldots,H_{n-1}$ 也都是正交矩阵. 令

$$Q=(H_{n-1}\cdots H_2H_1)^{-1}=H_1^{-1}H_2^{-1}\cdots H_{n-1}^{-1}=H_1H_2\cdots H_{n-1},$$

则 Q 也是正交矩阵, 且

$$A=(H_{n-1}\cdots H_2H_1)^{-1}R=QR.$$

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

$$\begin{cases} Q = I_n, \\ Q = QH_k, \quad k = 1, 2, \ldots, n-1. \end{cases}$$

✍ 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:

$$\begin{cases} Q = I_n, \\ Q = H_kQ, \quad k = n-1, n-2, \ldots, 1. \end{cases}$$

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. 计算 Q 的运算量大约为 $4m^2n - 4mn^2 + 4n^3/3$. 如果只需要计算 Q 的前 n 列, 则运算量大约为 $2mn^2 - 2n^3/3$, 此时 QR 分解的总运算量为 $4mn^2 - 4n^3/3$. 若 $m = n$, 则为 $8n^3/3$.

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

Set $Q = I_{m\times m}$
for $k = 1$ to n **do**
  $x = A(k:m, k)$
  $[\beta, v_k]$ = **House**(x)
  $A(k:m, k:n) = (I_{m-k+1} - \beta v_k v_k^{\mathsf{T}})A(k:m, k:n)$
  $= A(k:m, k:n) - \beta v_k\big(v_k^{\mathsf{T}}A(k:m, k:n)\big)$
  $Q(:, k:m) = Q(:, k:m)(I_{m-k+1} - \beta v_k v_k^{\mathsf{T}})$
  $= Q(:, k:m) - \beta\big(Q(:, k:m)v_k\big)v_k^{\mathsf{T}}$
end for

✍ 上面只是对基于 Householder 变换的 QR 分解算法的一个简单描述, 并没有考虑运算的优化. 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 $m - k + 1$, 因此我们可以先把 v_k 归一化: 使得 v_k 的第一元素为 1. 这样就只要存储 $v_k(k+1:m)$, 共 $m - k$ 个元素, 可以存放在 A 的第 k 列的严格下三角部分. 因此, 就可以用 A 的上三角部分存放 R, 严格下三角部分存放 Householder 向量.

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 $G_{41}, G_{51}, \ldots, G_{n1}$, 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 $G_{32}, G_{42}, \ldots, G_{n2}$, 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 转化成一个上三角矩阵 R, 即

$$R = G_{n,n-1}\cdots G_{21}A.$$

令 $Q = (G_{n,n-1}\cdots G_{21})^{\mathsf{T}}$. 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.
- ✍ 如果矩阵的非零下三角元素相对较少 (比如上 Hessenberg 矩阵), 则可以采用 Givens 变换.

3.3.5 QR 分解的稳定性

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. 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 [62] 和 [29]. 基于 MGS 的 QR 分解也是向后稳定的, 参见 [40].

Björck [4] 证明了, 通过 MGS 计算的矩阵 Q 满足

$$Q^{\mathsf{T}}Q = I + E_{MGS} \quad \text{其中} \quad \|E_{MGS}\|_2 \approx \varepsilon_u \kappa_2(A).$$

而 Householder 变换计算的矩阵 Q 满足

$$Q^{\mathsf{T}}Q = I + E_H \quad \text{其中} \quad \|E_H\|_2 \approx \varepsilon_u.$$

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 $\|\tilde{Q}^{\mathsf{T}}\tilde{Q}-I\|_2$ 的值, 其中 $\tilde{Q}$ 和 $\tilde{R}$ 是计算出来的 QR 分解矩阵因子. 试验结果如下: (LS_QR_stability.m)

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 Hermitian 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

$$U^*AV = \begin{bmatrix} \Sigma \\ 0 \end{bmatrix} \quad \text{或} \quad A = U \begin{bmatrix} \Sigma \\ 0 \end{bmatrix} V^*, \tag{3.9}$$

其中 $\Sigma = \operatorname{diag}(\sigma_1, \sigma_2, \ldots, \sigma_n) \in \mathbb{R}^{n\times n}$, 且 $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_n \geq 0$. 分解 (3.9) 称为 A 的奇异值分解 (SVD), 而 $\sigma_1, \sigma_2, \ldots, \sigma_n$ 则称为 A 的奇异值. (板书)

✍ 该定理也可以通过 Hermitian 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 [30].

由 (3.9) 可知,

$$A^*A = V\Sigma^*\Sigma V^*, \quad AA^* = U \begin{bmatrix} \Sigma\Sigma^* & 0 \\ 0 & 0 \end{bmatrix} U^*.$$

所以 $\sigma_1^2, \sigma_2^2, \ldots, \sigma_n^2$ 是 A^*A 和 AA^* 的特征值. 因此, A 的奇异值就是 A^*A 的特征值的平方根.

若 $\operatorname{rank}(A) = r \leq n$, 则有

$$\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > \sigma_{r+1} = \cdots = \sigma_n = 0.$$

特别地, 如果 $\operatorname{rank}(A) = n$, 则奇异值都是正的, 此时对角矩阵 Σ 非奇异.

矩阵 $U = [u_1, u_2, \ldots, u_m]$ 和 $V = [v_1, v_2, \ldots, v_n]$ 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

$$\begin{aligned} Av_i &= \sigma_i u_i, \quad i = 1, 2, \ldots, n, \\ A^*u_i &= \sigma_i v_i, \quad i = 1, 2, \ldots, n, \\ A^*u_i &= 0, \quad i = n+1, n+2, \ldots, m. \end{aligned}$$

由定理 3.10 可知

$$A = U \begin{bmatrix} \Sigma \\ 0 \end{bmatrix} V^* = \sigma_1 u_1 v_1^* + \sigma_2 u_2 v_2^* + \cdots + \sigma_n u_n v_n^* = \sum_{i=1}^{n} \sigma_i u_i v_i^*.$$

记 $U_n = [u_1, u_2, \ldots, u_n] \in \mathbb{C}^{m\times n}$, 则 U_n 是单位列正交矩阵 (即 $U_n^*U_n = I_{n\times n}$), 且

$$A = U_n \Sigma V^*. \tag{3.10}$$

这就是所谓的**细 SVD** (**thin SVD**) [21] 或 **降阶 SVD** (**reduced SVD**) [56], 有的文献将 (3.10) 称为奇异值分解.

设 $k < n$, 我们称

$$A_k = \sigma_1 u_1 v_1^* + \sigma_2 u_2 v_2^* + \cdots + \sigma_k u_k v_k^* = \sum_{i=1}^{k} \sigma_i u_i v_i^*$$

为 A 的**截断 SVD** (**truncated SVD**). 若 $\mathrm{rank}(A) = r < n$, 则有

$$A = A_r = \sum_{i=1}^{r} \sigma_i u_i v_i^*.$$

奇异值的应用

- 计算矩阵范数和条件数
- 计算矩阵的数值秩 (numerical rank)
- 求解最小二乘问题
- 矩阵和张量的低秩分解
- 图像处理, 压缩感知, 主成分分析, 数据降维, ...

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 σ_i^2, 对应的特征向量是 v_i, $i = 1, 2, \ldots, n$;

(2) AA^* 的特征值是 σ_i^2 和 $m - n$ 个零, 对应的特征向量是 u_i, $i = 1, 2, \ldots, m$;

(3) $\|A\|_2 = \sigma_1$, $\|A\|_F = \sqrt{\sigma_1^2 + \sigma_2^2 + \cdots + \sigma_n^2}$;

(4) 若 $\mathrm{rank}(A) = r \leq n$, 则

$$\mathrm{Ran}(A) = \mathrm{span}\{u_1, u_2, \ldots, u_r\}, \quad \mathrm{Ker}(A) = \mathrm{span}\{v_{r+1}, v_{r+2}, \ldots, v_n\};$$

(5) (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 $\sigma_i(X^*AY) = \sigma_i(A)$.

(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 $\|A^{-1}\|_2 = \sigma_n^{-1}$, $\kappa_2(A) = \sigma_1/\sigma_n$; (留作练习)

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 $A \in \mathbb{C}^{m\times n}$ 的降阶奇异值分解. 令 $A_k = \sum_{i=1}^{k} \sigma_i u_i v_i^*$, 则 A_k 是

$$\min_{B\in\mathbb{C}^{m\times n},\ \mathrm{rank}(B)=k} \|A - B\|_2 \tag{3.11}$$

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. (板书)

✍ 定理中的 (3.11) 式也可以改写为

$$\min_{B\in\mathbb{C}^{m\times n},\ \mathrm{rank}(B)\le k} \|A-B\|_2 \tag{3.12}$$

3.5 线性最小二乘问题的求解方法

3.5.1 正规方程

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 $\mathrm{Ran}(A)$ (值域) 正交, 即 x_* 是下面的**正规方程** (或**法方程**) 的解

$$A^\mathsf{T}(b - Ax) = 0 \quad \text{或} \quad A^\mathsf{T}Ax = A^\mathsf{T}b. \tag{3.13}$$

(板书)

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 (7.8) 的解. 由于

$$A^\mathsf{T}b \in \mathrm{Ran}(A^\mathsf{T}) = \mathrm{Ran}(A^\mathsf{T}A),$$

因此正规方程 $A^\mathsf{T}Ax = A^\mathsf{T}b$ 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 $A \in \mathbb{R}^{m\times n}$ $(m \geq n)$, 则 $A^\mathsf{T}A$ 对称正定当且仅当 A 是列满秩的, 即 $\mathrm{rank}(A) = n$. 此时, 线性最小二乘问题 (3.1) 的解是唯一的, 其表达式为

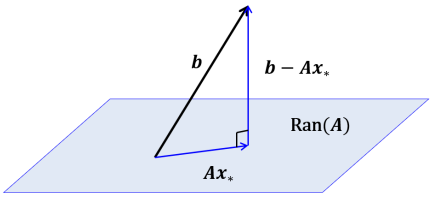
$$x_* = (A^\mathsf{T}A)^{-1}A^\mathsf{T}b.$$

最小二乘解的几何含义

根据定理 3.14, 我们可以把 b 写成

$$b = Ax_* + r, \quad \text{其中} \quad r \perp \mathrm{Ran}(A).$$

所以 Ax_* 就是 b 在 $\mathrm{Ran}(A)$ 上的正交投影, 见右图.



需要指出的是, 最小二乘解可能并不唯一, 但上述分解是唯一的.

3.5.2 Cholesky 分解法

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是用来计算 $A^\mathsf{T}A$ (由于 $A^\mathsf{T}A$ 对称, 因此只需计算其下三角部分).

- ✍ 用 Cholesky 分解求解正规方程, 运算量最小, 而且简单直观.
- ✍ 但由于 $A^\mathsf{T}A$ 的条件数是 A 的条件数的平方, 因此对于病态问题 (即 A 的条件数比较大), 不建议使用该方法.

例 3.3 下面的例子说明, 计算 $A^{\mathsf{T}}A$ 可能会损失计算精度: 设

$$A = \begin{bmatrix} 1 & 1 & 1 \\ \varepsilon & & \\ & \varepsilon & \\ & & \varepsilon \end{bmatrix},$$

则

$$A^{\mathsf{T}}A = \begin{bmatrix} 1+\varepsilon^2 & 1 & 1 \\ 1 & 1+\varepsilon^2 & 1 \\ 1 & 1 & 1+\varepsilon^2 \end{bmatrix}.$$

记 ε_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 $A^{\mathsf{T}}A$ 是奇异的. 但我们注意到 A 是满秩的.

3.5.3 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 $A = QR$, 由定理 3.14 可知, 最小二乘解为

$$x_* = (A^{\mathsf{T}}A)^{-1}A^{\mathsf{T}}b = (R^{\mathsf{T}}Q^{\mathsf{T}}QR)^{-1}R^{\mathsf{T}}Q^{\mathsf{T}}b = (R^{\mathsf{T}}R)^{-1}R^{\mathsf{T}}Q^{\mathsf{T}}b = R^{-1}Q^{\mathsf{T}}b.$$

✍ QR 分解法的运算量大约为 $2mn^2$ (如果采用 Householder 变换的话, 运算量大约为 $4mn^2 - 4n^3/3$). 当 $m \gg n$ 时, 大约为正规方程的两倍.

✍ QR 分解法比较稳定, 是当前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.

3.5.4 奇异值分解法

设 $A \in \mathbb{R}^{m\times n}$ 列满秩, $A = U\begin{bmatrix}\Sigma \\ 0\end{bmatrix}V^{\mathsf{T}}$ 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

$$\begin{aligned} \|Ax-b\|_2^2 &= \left\| U\begin{bmatrix}\Sigma \\ 0\end{bmatrix}V^{\mathsf{T}}x - b\right\|_2^2 \\ &= \left\| \begin{bmatrix}\Sigma \\ 0\end{bmatrix}V^{\mathsf{T}}x - [U_n, \tilde{U}]^{\mathsf{T}}b\right\|_2^2 \\ &= \left\| \begin{bmatrix}\Sigma V^{\mathsf{T}}x - U_n^{\mathsf{T}}b \\ -\tilde{U}^{\mathsf{T}}b\end{bmatrix}\right\|_2^2 \\ &= \|\Sigma V^{\mathsf{T}}x - U_n^{\mathsf{T}}b\|_2^2 + \|\tilde{U}^{\mathsf{T}}b\|_2^2 \geq \|\tilde{U}^{\mathsf{T}}b\|_2^2, \end{aligned}$$

等号当且仅当 $\Sigma V^{\mathsf{T}}x - U_n^{\mathsf{T}}b = 0$ 时成立, 即

$$x = (\Sigma V^{\mathsf{T}})^{-1}U_n^{\mathsf{T}}b = V\Sigma^{-1}U_n^{\mathsf{T}}b.$$

这就是线性最小二乘问题 (3.1) 的解.

✍ 相比于 Cholesky 分解法和 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 Cholesky 分解法和 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 3.4 分别用三种方法求解最小二乘问题, 比较运算时间. (LS_3methods.m)

3.6 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, 当 A 满秩时, 解是不唯一的, 因此问题是不适定的. 此时我们需要进行**正则化** (regularization). 一个简单的正则化方法就是极小化 $\|x\|_2^2$, 即

$$\min_{x\in\mathbb{R}^n} \frac{1}{2}\|Ax-b\|_2^2+\frac{\alpha}{2}\|x\|_2^2, \tag{3.14}$$

其中第二项就是**正则项**, $\alpha>0$ 是正则化参数. 对应的目标函数记为

$$J(x)=\frac{1}{2}\|Ax-b\|_2^2+\frac{\alpha}{2}\|x\|_2^2.$$

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

$$(A^{\mathsf{T}}A+\alpha I)x=A^{\mathsf{T}}b.$$

由于 $A^{\mathsf{T}}A+\alpha I$ 对称正定, 故非奇异. 所以问题 (3.14) 的唯一解为

$$x=(A^{\mathsf{T}}A+\alpha I)^{-1}A^{\mathsf{T}}b.$$

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

$$\min_{x\in\mathbb{R}^n} \frac{1}{2}\|Ax-b\|_2^2+\frac{\alpha}{2}\|Wx\|_2^2, \tag{3.15}$$

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 不一定要求是方阵.

经过类似的推导, 可知问题 (3.15) 的解满足

$$(A^{\mathsf{T}}A+\alpha W^{\mathsf{T}}W)x=A^{\mathsf{T}}b.$$

如果 $A^{\mathsf{T}}A+\alpha W^{\mathsf{T}}W$ 非奇异, 则存在唯一解

$$x=(A^{\mathsf{T}}A+\alpha W^{\mathsf{T}}W)^{-1}A^{\mathsf{T}}b.$$

3.8.3 约束最小二乘问题

考虑带有约束的最小二乘问题

$$\begin{aligned}&\min_{x\in\mathbb{R}^n} \frac{1}{2}\|Ax-b\|_2^2 \\ &\text{s.t.}\quad Bx=f\end{aligned} \tag{3.16}$$

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

$$J(x)=\frac{1}{2}\|Ax-b\|_2^2+\lambda^{\mathsf{T}}(Bx-f).$$

其中 λ 是 Lagrange 乘子. 分别对 x 和 λ 求一阶导数, 并令其等于零, 可得

$$\begin{bmatrix}A^{\mathsf{T}}A & B^{\mathsf{T}}\\ B & 0\end{bmatrix}\begin{bmatrix}x\\ \lambda\end{bmatrix}=\begin{bmatrix}A^{\mathsf{T}}b\\ f\end{bmatrix}.$$

如果 $A^\mathsf{T}A$ 非奇异, 且 B 行满秩, 则存在唯一解

$$\begin{aligned}\lambda &= \left[B(A^\mathsf{T}A)^{-1}B^\mathsf{T}\right]^{-1}\left[B(A^\mathsf{T}A)^{-1}A^\mathsf{T}b - f\right]\\ x &= (A^\mathsf{T}A)^{-1}(A^\mathsf{T}b - B^\mathsf{T}\lambda).\end{aligned}$$

几类特殊情形

- 如果 $A = I$ 且 $f = 0$, 则有
$$\begin{aligned}&\min_{x\in\mathbb{R}^n}\frac{1}{2}\|x-b\|_2^2 \quad \text{s.t.} \quad Bx = 0\\ &\qquad\Longrightarrow x = b - B^\mathsf{T}(BB^\mathsf{T})^{-1}Bb.\end{aligned}$$
- 如果 $b = 0$, 则有
$$\begin{aligned}&\min_{x\in\mathbb{R}^n}\frac{1}{2}\|Ax\|_2^2 \quad \text{s.t.} \quad Bx = f\\ &\qquad\Longrightarrow x = (A^\mathsf{T}A)^{-1}B^\mathsf{T}\left[B(A^\mathsf{T}A)^{-1}B^\mathsf{T}\right]^{-1}f.\end{aligned}$$
- 如果 $A = I$ 且 $b = 0$, 则有
$$\begin{aligned}&\min_{x\in\mathbb{R}^n}\frac{1}{2}\|x\|_2^2 \quad \text{s.t.} \quad Bx = f\\ &\qquad\Longrightarrow x = B^\mathsf{T}(BB^\mathsf{T})^{-1}f.\end{aligned}$$

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

$$p(t) = a_0 + a_1t + a_2t^2 + \cdots + a_mt^m.$$

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 $A \in \mathbb{R}^{n\times(m+1)}$, $x = [a_0, a_1, \ldots, a_m]^\mathsf{T}$. 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 $f_{k-1}, f_{k-2}, \ldots, f_{k-m}$, 即

$$f_k = a_1f_{k-1} + a_2f_{k-2} + \cdots + a_mf_{k-m}. \tag{3.17}$$

现在已经测得该时间序列的前 n 个值 f_i, $i = 0, 1, 2, \ldots, n-1$, 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 (3.17) 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 $A \in \mathbb{R}^{(n-m)\times m}$, $x = [a_1, a_2, \ldots, a_m]^\intercal$. 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 3.5 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 3.6 举例: `LS_SVD_ImageCompress_01.m`, `LS_SVD_ImageCompress_02.m`

4 线性方程组迭代法

直接法的运算量为 $\mathcal{O}(n^3)$, 所以随着未知量个数的增大, 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, 除了具有特殊结构的线性方程组外, 直接法一般不再被采用, 取而代之的是迭代法.

考虑线性方程组

$$Ax = b,$$

其中 $A \in \mathbb{R}^{n\times n}$ 非奇异. 迭代法的基本思想: 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

$$\lim_{k\to\infty} x^{(k)} = x_* \triangleq A^{-1}b.$$

目前常用的迭代法主要有两类, 一类是定常迭代法, 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 [58]

[3] D. M. Young, *Iterative Solution of Large Linear Systems*, 1971 [63]

[4] Y. Saad, *Iterative Methods for Sparse Linear Systems*, 2nd, 2003 [49]

4.1 定常迭代法

4.1.1 迭代法基本概念

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 是 A 在某种意义下的近似, 其对应的线性方程组比较容易求解. 然后我们用 $Mx = b$ 的解来近似 $Ax = b$ 的解. 如果近似解满足精度要求, 则停止计算, 否则就通过某种修正方法来提升近似解的精度, 直到满足要求为止. 下面给出具体的描述.

记 $Mx = b$ 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

$$A\left(x_* - x^{(1)}\right) = b - Ax^{(1)}.$$

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.

记 $\Delta x \triangleq x_* - x^{(1)}$, 则 Δx 满足方程

$$A\Delta x = b - Ax^{(1)}.$$

如果能解出 Δx, 则 $x^{(1)} + \Delta x$ 就是精确解. 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

$$x^{(2)} \triangleq x^{(1)} + \tilde{\Delta} x = x^{(1)} + M^{-1}(b - Ax^{(1)}).$$

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近似解组成的向量序列

$$x^{(1)}, x^{(2)}, \dots, x^{(k)}, \dots.$$

它们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法. 由于每次迭代的格式是一样的, 因此称为 **定常迭代法**.

好的定常迭代法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: 迭代序列 $\{x^{(k)}\}$ **快速收敛** 到真解 x_*.

目前一类常用的定常迭代法是基于矩阵分裂的迭代法, 如 Jacobi 迭代法, G-S (Gauss-Seidel) 迭代法, SOR (Successive Overrelaxation, 超松弛) 迭代法等.

定义 4.1 (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

$$x^{(k+1)} = M^{-1}Nx^{(k)} + M^{-1}b \triangleq Gx^{(k)} + g, \quad k = 0, 1, \dots, \tag{4.3}$$

其中 $G \triangleq M^{-1}N$ 称为**迭代矩阵**, $x^{(0)}$ 为迭代初值, 可以任意给定 (如零向量), 或者通过其他方法获取. 这就是基于矩阵分裂 (4.1) 的定常迭代法, 其中 $x^{(k)}$ 称为第 k 步迭代解.

显然, 选取不同的 M, 就可以构造出不同的迭代法. 下面介绍三个经典的迭代法.

4.1.2 Jacobi 迭代法

将矩阵 A 写成

$$A = D - L - U,$$

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 迭代法**:

$$x^{(k+1)} = D^{-1}(L+U)x^{(k)} + D^{-1}b, \quad k = 0, 1, 2, \ldots. \tag{4.4}$$

对应的迭代矩阵为

$$G_{\mathrm{J}} \triangleq D^{-1}(L+U).$$

写成分量形式即为

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

✍ 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 $i = 1, 2, \ldots, n$ 计算, 也可以按顺序 $i = n, n-1, \ldots, 2, 1$ 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. 求解线性方程组的 Jacobi 迭代法

1: Given an initial guess $x^{(0)}$
2: **while** not converge **do** % 停机准则
3: **for** $i = 1$ to n **do**
4: $x_i^{(k+1)} = \left(b_i - \sum_{j=1, j\neq i}^{n} a_{ij}x_j^{(k)}\right) \Big/ a_{ii}$
5: **end for**
6: **end while**

✍ 在对线性方程组进行数值求解时, "停机准则" 一般是指要求相对残量满足一定的精度, 即

$$\frac{\|b - Ax^{(k)}\|}{\|b - Ax^{(0)}\|} < \mathrm{tol},$$

其中 tol 是一个事前给定的精度要求, 如 10^{-6} 或 10^{-8} 等.

我们也可以将 Jacobi 迭代格式写为

$$x^{(k+1)} = x^{(k)} + D^{-1}(b - Ax^{(k)}) = x^{(k)} + D^{-1}r_k, \quad k = 0, 1, 2, \ldots,$$

其中 $r_k \triangleq b - Ax^{(k)}$ 是第 k 次迭代后的**残量**. 这表明, $x^{(k+1)}$ 是通过对 $x^{(k)}$ 做修正得到的.

4.1.3 Gauss-Seidel 迭代法

在分裂 $A = M - N$ 中取 $M = D - L$, $N = U$, 即可得 **Gauss-Seidel (G-S) 迭代法**:

$$x^{(k+1)} = (D-L)^{-1}Ux^{(k)} + (D-L)^{-1}b, \quad k = 0, 1, 2, \ldots. \tag{4.5}$$

对应的迭代矩阵为

$$G_{\mathrm{GS}} \triangleq (D-L)^{-1}U.$$

将 G-S 迭代格式 (4.5) 改写为

$$Dx^{(k+1)} = Lx^{(k+1)} + Ux^{(k)} + b,$$

即可得分量形式

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

算法 4.2. 求解线性方程组的 G-S 迭代法

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 迭代法的主要优点是充分利用了已经获得的最新数据, 因此可能会更快收敛.

✍ G-S 迭代法对 $x^{(k+1)}$ 的分量的更新有顺序要求, 即必须按 $x_1^{(k+1)}, x_2^{(k+1)}, \ldots, x_n^{(k+1)}$ 的顺序进行更新, 因此不适合并行计算.

4.1.4 SOR 迭代法

在 G-S 迭代法的基础上, 我们可以通过引入一个松弛参数 ω 来加快收敛速度. 这就是 SOR (Successive Overrelaxation) 迭代法的基本思想. SOR 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

$$x^{(k+1)} = (1-\omega)x^{(k)} + \omega D^{-1}\left(Lx^{(k+1)} + Ux^{(k)} + b\right). \tag{4.6}$$

整理后即为

$$x^{(k+1)} = (D-\omega L)^{-1}\big((1-\omega)D + \omega U\big)x^{(k)} + \omega(D-\omega L)^{-1}b, \tag{4.7}$$

其中 ω 称为**松弛参数**.

- 当 $\omega = 1$ 时, SOR 即为 G-S 迭代法,

- 当 $\omega < 1$ 时, 称为**低松弛**迭代法,
- 当 $\omega > 1$ 时, 称为**超松弛**迭代法.

✍ SOR 方法曾经在很长一段时间内是求解线性方程组的首选方法.
✍ 在大多数情况下, 当 $\omega > 1$ 时会取得比较好的收敛效果.

SOR 的迭代矩阵为

$$G_{\mathrm{SOR}} = (D - \omega L)^{-1}\big((1-\omega)D + \omega U\big),$$

对应的矩阵分裂为

$$M = \frac{1}{\omega}D - L, \quad N = \frac{1-\omega}{\omega}D + U.$$

由 (4.6) 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 迭代法*

1: Given an initial guess $x^{(0)}$ and parameter ω
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 ω, 通过选取适当的 ω 可以大大提高方法的收敛速度. 但如何确定 SOR 的最优松弛参数是一件非常困难的事!

例 4.1 分别用 Jacobi, G-S 和 SOR($\omega = 1.1$) 迭代法求解线性方程组 $Ax = b$, 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 $x^{(0)} = [0,0,0]^{\mathsf{T}}$, 迭代过程中保留小数点后四位. `(Iter_Jacobi_GS_SOR_01.m)`

例 4.2 编程实践: 分别用 Jacobi, G-S 和 SOR($\omega = 1.5$) 迭代法求解线性方程组 $Ax = b$, 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 $x^{(0)} = [0, 0, \ldots, 0]^{\mathsf{T}}$. (Iter_Jacobi_GS_SOR_02.m)

关于定常迭代法的注记

定常迭代法主要取决于 M 的选取: 一方面要使得以 M 为系数矩阵的线性方程组更容易求解, 另一方面也要使得 M 是 A 在某种意义下的很好近似. 一般来说, 要构造出好的定常迭代法, 必须充分利用原问题本身的结构特点.

4.2 收敛性分析

4.2.1 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 4.2 (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 $x = [x_1, x_2, \dots, x_n]^{\mathsf{T}} \in \mathbb{R}^n$, 使得

$$\lim_{k\to\infty} x_i^{(k)} = x_i, \quad i = 1, 2, \dots, n,$$

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, 即 x 为序列 $\{x^{(k)}\}$ 的极限, 记为

$$\lim_{k\to\infty} x^{(k)} = x.$$

相类似地, 我们可以给出矩阵序列收敛的定义.

定义 4.3 (矩阵序列的收敛) 设 $\left\{A^{(k)} = \left[a_{ij}^{(k)}\right]\right\}_{k=0}^{\infty}$ 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

$$\lim_{k\to\infty} a_{ij}^{(k)} = a_{ij}, \quad i, j = 1, 2, \dots, n,$$

则称 $A^{(k)}$ (按分量) 收敛到 A, 即 A 为 $A^{(k)}$ 的极限, 记为

$$\lim_{k\to\infty} A^{(k)} = A.$$

例 4.3 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 $k \to \infty$ 时, 有

$$A^{(k)} = \begin{bmatrix} a & 1 \\ 0 & a \end{bmatrix}^k = \begin{bmatrix} a^k & ka^{k-1} \\ 0 & a^k \end{bmatrix} \to \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}.$$

关于向量序列和矩阵序列的收敛性, 我们有下面的一般判别方法.

定理 4.1 设向量序列 $\{x^{(k)}\}_{k=0}^{\infty} \subset \mathbb{R}^n$, 矩阵序列 $\left\{A^{(k)} = \left[a_{ij}^{(k)}\right]\right\}_{k=0}^{\infty} \subset \mathbb{R}^{n\times n}$, 则

(1) $\lim\limits_{k\to\infty} x^{(k)} = x$ 当且仅当 $\lim\limits_{k\to\infty} \|x^{(k)} - x\| = 0$, 其中 $\|\cdot\|$ 为任一向量范数;

(2) $\lim\limits_{k\to\infty} A^{(k)} = A$ 当且仅当 $\lim\limits_{k\to\infty} \|A^{(k)} - A\| = 0$, 其中 $\|\cdot\|$ 为任一矩阵范数;

(板书, 根据范数的等价性, 只需证明对无穷范数成立即可)

例 4.4 由定理 4.1 可以立即得到下面的两个结论:

$$\lim_{k\to\infty} A^{(k)} = 0 \Longleftrightarrow \lim_{k\to\infty} \|A^{(k)}\| = 0$$

和

$$\lim_{k\to\infty} A^{k} = 0 \Longleftrightarrow \lim_{k\to\infty} \|A^{k}\| = 0$$

定理 4.2 设矩阵序列 $\left\{A^{(k)} = \left[a_{ij}^{(k)}\right]\right\}_{k=0}^{\infty} \subset \mathbb{R}^{n\times n}$, 则

$$\lim_{k\to\infty} A^{(k)} = 0 \Longleftrightarrow \lim_{k\to\infty} A^{(k)}x = 0, \quad \forall\, x \in \mathbb{R}^n.$$

(板书)

定理 4.3 设 $A \in \mathbb{R}^{n\times n}$, 若存在矩阵范数使得 $\|A\| < 1$, 则 $\lim\limits_{k\to\infty} A^k = 0$. (板书)

定理 4.4 设 $A \in \mathbb{R}^{n\times n}$, 则 $\lim\limits_{k\to\infty} A^k = 0$ 当且仅当 $\rho(A) < 1$. (板书)

推论 4.5 设 $A \in \mathbb{R}^{n\times n}$, 则 $\lim\limits_{k\to\infty} A^k = 0$ 的充要条件是存在某个矩阵范数 $\|\cdot\|$, 使得 $\|A\| < 1$.

4.2.2 迭代法的收敛性

定义 4.4 对任意初始向量 $x^{(0)}$, 设 $\{x^{(k)}\}$ 是由迭代法 (4.3) 生成的向量序列, 如果 $\lim\limits_{k\to\infty} x^{(k)}$ 存在, 则称迭代法 (4.3) **收敛**, 否则就称为 **发散**.

✍ 需要注意的是, 算法收敛是指对任意初值. 也就是说, 如果存在一个初值, 使得迭代序列不收敛, 则算法就不收敛.

引理 4.6 设由迭代法 (4.3) 生成的迭代序列 $\{x^{(k)}\}$ 收敛, 且 $\lim\limits_{k\to\infty} x^{(k)} = x_*$, 则 x_* 是原方程组的解. (板书)

下面是定常迭代法 (4.3) 的基本收敛性定理.

定理 4.7 (基本收敛性定理) 对任意初始向量 $x^{(0)}$, 迭代法 (4.3) 收敛的充要条件是 $\rho(G) < 1$. (板书)

由于对任意范数都有 $\rho(G) \le \|G\|$, 因此我们可以立即得到下面的结论.

定理 4.8 若存在矩阵范数使得 $\|G\| < 1$, 则迭代法 (4.3) 收敛

✍ 由于计算 $\rho(G)$ 通常比较复杂, 而 $\|G\|_1$, $\|G\|_\infty$ 相对比较容易计算, 因此在判别迭代法收敛性时, 可以先验算一下迭代矩阵的 1-范数或 ∞-范数是否小于 1.

✍ 定理 4.8 是充分条件, 但不是必要条件. 判断一个定常迭代法不收敛仍然需要使用定理 4.7.

例 4.5 讨论迭代法 $x^{(k+1)} = Gx^{(k)} + g$ 的收敛性, 其中 $G = \begin{bmatrix} 0.9 & 0 \\ 0.3 & 0.8 \end{bmatrix}$.

解. 由于 G 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 $\rho(G) = 0.9 < 1$. 故迭代法收敛. □

当迭代矩阵满足 $\|G\| < 1$ 时, 迭代法收敛. 下面的结论告诉我们, 经过 k 步迭代后, 近似解的误差大致会下降多少, 即迭代解的近似程度如何.

定理 4.9 若存在算子范数使得 $q \triangleq \|G\| < 1$, 则

(1) $\|x^{(k)} - x_*\| \le q^k \|x^{(0)} - x_*\|$;

(2) $\|x^{(k)} - x_*\| \le \dfrac{q}{1-q} \|x^{(k)} - x^{(k-1)}\|$;

(3) $\|x^{(k)} - x_*\| \le \dfrac{q^k}{1-q} \|x^{(1)} - x^{(0)}\|$.

(板书)

✍ 定理 4.9 中的结论 (2) 和 (3) 都可以用来估计近似解 $x^{(k)}$ 的误差. 通常结论 (2) 会更准确一些. 因此, 实际应用中有时也以相邻两次迭代解之间的相对误差作为算法终止的条件, 即

$$\frac{\|x^{(k)}-x^{(k-1)}\|}{\|x^{(k)}\|}<\text{tol},$$

其中 tol 是一个事前给定的精度要求, 比如 10^{-6} 或 10^{-8} 等.

✍ 定理 4.9 的条件中要求是算子范数, 这是为了有相应的向量范数. 事实上也可以改为任意矩阵范数, 因为任意矩阵范数都存在相容的向量范数, 见练习 ??.

迭代法的收敛速度

考虑迭代法 (4.3), 第 k 步的误差为

$$\varepsilon^{(k)}\triangleq x^{(k)}-x_*=G^k(x^{(0)}-x_*)=G^k\varepsilon^{(0)}.$$

所以

$$\frac{\|\varepsilon^{(k)}\|}{\|\varepsilon^{(0)}\|}\le\|G^k\|.$$

因此平均每次迭代后误差的压缩率 (下降率) 大约为 $\|G^k\|^{1/k}$.

定义 4.5 迭代法 (4.3) 的**平均收敛速度**定义为

$$R_k(G)\triangleq-\ln\|G^k\|^{\frac{1}{k}},$$

渐进收敛速度定义为

$$R(G)\triangleq\lim_{k\to\infty}R_k(G)=-\ln\rho(G).$$

✍ 平均收敛速度与迭代步数 k 和矩阵范数有关, 渐进收敛速度则与 k 和矩阵范数无关.

✍ 如果 $0<\rho(G)<1$, 则迭代法 (4.3) 线性收敛.

✍ 一般来说, $\rho(G)$ 越小, 迭代法 (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:

$$\|G^k\|<\varepsilon\Longrightarrow\ln\|G^k\|^{1/k}\le\frac{1}{k}\ln(\varepsilon)\Longrightarrow k\ge\frac{-\ln(\varepsilon)}{-\ln\|G^k\|^{1/k}}\approx\frac{-\ln(\varepsilon)}{R(G)}.$$

4.3 经典迭代法的收敛性

这里考虑三个经典迭代法 (Jacobi、G-S 和 SOR) 的收敛性. 首先, 根据迭代法基本收敛性定理 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: $\rho(G_{\mathrm{J}}) < 1$
- G-S 迭代收敛的 **充要** 条件: $\rho(G_{\mathrm{GS}}) < 1$
- SOR 迭代收敛的 **充要** 条件: $\rho(G_{\mathrm{SOR}}) < 1$
- Jacobi 迭代收敛的 **充分** 条件: $\|G_{\mathrm{J}}\| < 1$
- G-S 迭代收敛的充分条件 **充分** 条件: $\|G_{\mathrm{GS}}\| < 1$
- SOR 迭代收敛的充分条件 **充分** 条件: $\|G_{\mathrm{SOR}}\| < 1$

如果系数矩阵具有某些特殊结构, 则存在一些更加简洁的判别方法. 这里考虑对角占优和对称正定两种情形, 更多情形请参见相关参考资料.

4.3.1 不可约与对角占优情形

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

$$PAP^{\mathsf{T}} = \begin{bmatrix} A_{11} & A_{12} \\ 0 & A_{22} \end{bmatrix},$$

其中 $A_{11} \in \mathbb{R}^{k\times k}$ $(1 \le k < n)$, 则称 A 为可约矩阵, 否则称为 不可约矩阵.

例 4.6 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 $n \ge 2$, 则 A 的零元素个数 $\ge n-1$.

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、下两条次对角线上的元素都非零, 则 A 是否不可约?

在后面的讨论中, 我们只考虑不可约的情形.

为什么只需考虑不可约情形

若 A 可约, 即存在置换矩阵 P, 使得

$$PAP^{\mathsf{T}} = \begin{bmatrix} A_{11} & A_{12} \\ 0 & A_{22} \end{bmatrix},$$

则线性方程组 $Ax = b$ 等价于 $PAP^{\mathsf{T}}Px = Pb$. 记 $y \triangleq Px$, $f \triangleq Pb$, 则

$$\begin{bmatrix} A_{11} & A_{12} \\ 0 & A_{22} \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \end{bmatrix}.$$

因此, 原方程组就转化为下面两个更小规模的子方程组

$$A_{22}y_2 = f_2,$$

$$A_{11}y_1 = f_1 - A_{12}y_2.$$

显然, 求解这两个方程组比求解原方程组所需的运算量更少.

以此类推, 如果 A_{22} 或 A_{11} 仍然是可约的, 则可以转化为更小规模的子问题.

对于特征值问题, 也是如此.

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 $i = 1, 2, \ldots, n$ 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**, 有时简称**行对角占优**或**对角占优**. 若对所有 $i = 1, 2, \ldots, n$, 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**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. (板书)

定理 4.12 设 $A \in \mathbb{R}^{n\times n}$ 是不可约的对角占优矩阵, 则 A 非奇异.

(留作课外自习, 证明可参见相关资料)

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 迭代法都收敛. (板书)

✍ 事实上, 当 A 严格对角占优时, 有

$$\|G_{\mathrm{GS}}\|_\infty \leq \|G_{\mathrm{J}}\|_\infty < 1.$$

该结论的证明留作练习.

定理 4.14 设 $A \in \mathbb{R}^{n\times n}$, 若 A 是不可约对角占优, 则 Jacobi 和 G-S 迭代法都收敛.

(留作课外自习, 证明可参见 [58])

SOR 的收敛性

我们首先给出 SOR 迭代收敛的一个必要条件. 记 $\tilde{L} = D^{-1}L, \tilde{U} = D^{-1}U$.

定理 4.15 若 SOR 迭代法收敛, 则 $0 < \omega < 2$. (板书)

当 A 对角占优时, 我们有下面的结论.

定理 4.16 设 $A \in \mathbb{R}^{n\times n}$, 若 A 严格对角占优 (或不可约对角占优) 且 $0 < \omega \leq 1$, 则 SOR 收敛.
(留作课外自习, 证明可参见 [71])

4.3.2 对称正定情形

引理 4.17 设 $A \in \mathbb{R}^{n\times n}$ 对称, $A = M - N$, 其中 M 非奇异.

(1) 如果 A 和 $M^{\mathsf{T}} + N$ 都是正定的, 则 $\rho(M^{-1}N) < 1$;

(2) 如果 $\rho(M^{-1}N) < 1$ 且 $M^{\mathsf{T}} + N$ 正定, 则 A 是正定的.

(板书)

对于 Jacobi 迭代法, 我们有 $M^{\mathsf{T}} + N = D + L + U = 2D - A$. 根据引理 4.17, 我们立刻可以得到下面的结论.

定理 4.18 设 $A \in \mathbb{R}^{n\times n}$ 对称且 $2D - A$ 正定, 则 Jacobi 迭代法收敛的充要条件是 A 正定.

更进一步, 我们有下面的结论 [63, page 111].

定理 4.19 设 $A \in \mathbb{R}^{n\times n}$ 对称且对角线元素都为正, 则 Jacobi 迭代法收敛的充要条件是 A 和 $2D - A$ 都正定. (板书)

对于 G-S 迭代法, 当 A 对称时有

$$M^{\mathsf{T}} + N = (D - L)^{\mathsf{T}} + U = D - L^{\mathsf{T}} + U = D,$$

于是下面的结论成立.

定理 4.20 设 $A \in \mathbb{R}^{n\times n}$ 对称且对角线元素都为正, 则 G-S 迭代法收敛的充要条件是 A 正定.

对于 SOR 迭代法, 当 A 对称时有

$$M^{\mathsf{T}} + N = \left(\frac{1}{\omega}D - L\right)^{\mathsf{T}} + \frac{1-\omega}{\omega}D + U = \frac{2-\omega}{\omega}D,$$

所以我们有下面的结论.

定理 4.21 考虑 SOR 迭代法.

(1) 设 $A \in \mathbb{R}^{n\times n}$ 对称正定, 则 SOR 迭代法收敛的充要条件是 $0 < \omega < 2$;

(2) 设 $A \in \mathbb{R}^{n\times n}$ 对称且对角线元素都为正, 若存在 ω 使得 SOR 迭代法收敛, 则 A 正定.

推论 4.22 设 $A \in \mathbb{R}^{n\times n}$ 对称且对角线元素都为正, 则 SOR 迭代法收敛的充要条件是 A 正定且 $0 < \omega < 2$.

例 4.7 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

共轭梯度法是子空间迭代法的典型代表, 最早由 Hestenes 和 Stiefel [28] 于 1952 年提出, 但当时并没有引起大家的广泛关注, 直到 1971 年才逐渐流行起来 [45], 目前已成为求解大规模 (特别是稀疏的) 对称正定线性方程组的首选方法 [50].

设 $A \in \mathbb{R}^{n\times n}$ 对称正定, 共轭梯度法的基本思想是将计算线性方程组的解转为计算下面二次函数的最小值点:

$$\Phi(x) \triangleq \frac{1}{2}x^{\mathsf{T}}Ax - b^{\mathsf{T}}x \quad .$$

定理 4.23 设 $A \in \mathbb{R}^{n\times n}$ 对称正定, 则 x_* 是 $Ax = b$ 的解当且仅当 x_* 是 $\Phi(x)$ 的最小值点, 即 x_* 是最小化问题

$$\min_{x\in\mathbb{R}^n} \frac{1}{2}x^{\mathsf{T}}Ax - b^{\mathsf{T}}x \tag{4.8}$$

的解. (板书)

✍ 定理也可以利用凸函数极值点与导数 (梯度) 的关系得到, 即一阶导数为零.

4.4.1 最速下降法

求解最小化问题 (4.8) 的一类常用方法是**线搜索方法**. 给定迭代初始值 $x^{(0)}$, 我们通过下面的方法来计算 $x^{(1)}$: 首先确定一个使得 $\Phi(x)$ 变小的方向 $p_1 \in \mathbb{R}^n$, 即**下降方向**, 满足 $\|p_1\| = 1$, 然后计算步长 $\alpha_1 \in \mathbb{R}_+$, 使得 $\Phi(x)$ 沿该下降方向达到最小, 即

$$x^{(1)} = x^{(0)} + \alpha_1 p_1, \quad \text{其中} \quad \alpha_1 = \operatorname*{argmin}_{\alpha>0} \Phi\big(x^{(0)} + \alpha p_1\big) \quad .$$

由于 A 对称正定, 直接计算可得

$$\begin{aligned}\Phi\big(x^{(0)} + \alpha p_1\big) &= \frac{1}{2}\big(x^{(0)} + \alpha p_1\big)^{\mathsf{T}} A\big(x^{(0)} + \alpha p_1\big) - b^{\mathsf{T}}\big(x^{(0)} + \alpha p_1\big)\\ &= \frac{1}{2}\alpha^2 p_1^{\mathsf{T}}Ap_1 + \alpha p_1^{\mathsf{T}}Ax^{(0)} + \frac{1}{2}\big(x^{(0)}\big)^{\mathsf{T}}Ax^{(0)} - b^{\mathsf{T}}x^{(0)} - \alpha b^{\mathsf{T}}p_1\\ &= \frac{1}{2}\alpha^2 p_1^{\mathsf{T}}Ap_1 - \alpha p_1^{\mathsf{T}}r_0 + \Phi\big(x^{(0)}\big),\end{aligned} \tag{4.9}$$

其中 $r_0 = b - Ax^{(0)}$. 因此当 p_1 确定后, $\Phi\big(x^{(0)} + \alpha p_1\big)$ 是关于 α 的一元二次函数, 且二次项系数为正, 所以

$$\alpha_1 = \operatorname*{argmin}_{\alpha>0} \Phi\big(x^{(0)} + \alpha p_1\big) = \frac{p_1^{\mathsf{T}}r_0}{p_1^{\mathsf{T}}Ap_1}.$$

下面讨论如何选取下降方向 p_1. 根据多元函数的 Taylor 展开公式, 我们有

$$\Phi(x) = \Phi\big(x^{(0)}\big) + \big(x - x^{(0)}\big)^{\mathsf{T}}\nabla\Phi\big(x^{(0)}\big) + o\big(\|x - x^{(0)}\|\big).$$

记 p 为 $x - x^{(0)}$ 所在的方向, 即 $p = \dfrac{x - x^{(0)}}{\|x - x^{(0)}\|}$, 则

$$\big(x - x^{(0)}\big)^{\mathsf{T}}\nabla\Phi\big(x^{(0)}\big) = \|x - x^{(0)}\| \cdot p^{\mathsf{T}}\nabla\Phi\big(x^{(0)}\big).$$

因此, 只要满足 $p^{\mathsf{T}}\nabla\Phi\big(x^{(0)}\big) < 0$, 则 p 就是下降方向.

为了使得下降速度尽可能地快一些, 我们希望 $p^{\mathsf{T}}\nabla\Phi\big(x^{(0)}\big)$ 越小越好. 由 Cauchy-Schwarz 不等式可知

$$\big|p^{\mathsf{T}}\nabla\Phi\big(x^{(0)}\big)\big| \leq \|p\| \cdot \|\nabla\Phi\big(x^{(0)}\big)\|,$$

等号当且仅当 p 与 $\nabla\Phi\big(x^{(0)}\big)$ 共线时成立, 因此当 $p = -\dfrac{\nabla\Phi\big(x^{(0)}\big)}{\nabla\Phi\big(x^{(0)}\big)}$ 时, $p^{\mathsf{T}}\nabla\Phi\big(x^{(0)}\big)$ 达到最小. 我们称该下降方向为**最速下降方向**, 相应的线搜索方法为**最速下降法**. 由于最速下降方向就是 $\Phi(x)$ 在当前迭代点处的负梯度, 因此也称为**负梯度方向法**.

✍ 需要指出的是, 我们在讨论下降方向时舍弃了 Taylor 展开式中的高阶项, 因此这里得到的最速下降方向是局部最优的.

最速下降法的一般格式可表示为

$$x^{(k)} = x^{(k-1)} + \alpha_k p_k, \quad k = 1, 2, \ldots,$$
$$\text{其中 } p_k = -\nabla\Phi\big(x^{(k-1)}\big) = b - Ax^{(k-1)} = r_{k-1}, \quad \alpha_k = \frac{p_k^{\mathsf{T}} r_{k-1}}{p_k^{\mathsf{T}} A p_k} = \frac{r_{k-1}^{\mathsf{T}} r_{k-1}}{r_{k-1}^{\mathsf{T}} A r_{k-1}}$$

✍ 实际计算时, 我们无需对下降方向 p_k 进行单位化.

易知, 最速下降法的迭代解具有下面的局部最优性

$$x^{(k)} = \operatorname*{argmin}_{x \in x^{(k-1)} + \operatorname{span}\{p_k\}} \Phi(x). \tag{4.10}$$

算法 4.4. 最速下降法 (Steepest Descent Algorithm)

1: 给定初值 $x^{(0)}$, 令 $k = 1$
2: **while** not converge **do**
3: 计算负梯度方向 (残量) $r_{k-1} = b - Ax^{(k-1)}$ 和步长 $\alpha_k = \dfrac{r_{k-1}^{\mathsf{T}} r_{k-1}}{r_{k-1}^{\mathsf{T}} A r_{k-1}}$
4: 计算 $x^{(k)} = x^{(k-1)} + \alpha_k p_k$, 其中 $p_k = r_{k-1}$
5: 令 $k = k + 1$
6: **end while**

下面给出最速下降法的收敛性.

定理 4.24 设 $A \in \mathbb{R}^{n\times n}$ 对称正定, 则对任意初值 $x^{(0)}$, 最速下降法都收敛, 且

$$\frac{\big\|x^{(k+1)} - x_*\big\|_A}{\big\|x^{(k)} - x_*\big\|_A} \leq \frac{\kappa - 1}{\kappa + 1},$$

其中 κ 为 A 的谱条件数, 范数 $\|x\|_A \triangleq \sqrt{x^{\mathsf{T}} A x}$.

(留作课外自习, 可以参见矩阵计算或最优化方法的相关教材, 如 [37, 49])

例 4.8 用最速下降法求解 $Ax=b$, 其中 $A=\begin{bmatrix}15 & 2\\ 2 & 15\end{bmatrix}$, $b=\begin{bmatrix}17\\17\end{bmatrix}$, 初值 $x^{(0)}=\begin{bmatrix}-0.5\\0\end{bmatrix}$. (板书)

4.4.2 共轭梯度法

最速下降法的主要缺点是每次选取下降方向时只考虑局部最优, 无法保证下降方向 $p_1, p_2, \dots$ 线性无关, 这意味着迭代过程可能会出现 "之" 字型, 导致收敛速度变得非常缓慢.

例 4.9 用最速下降法求解 $Ax=b$, 其中 $A=\begin{bmatrix}2 & 1\\ 1 & 3\end{bmatrix}$, $b=\begin{bmatrix}3\\4\end{bmatrix}$, 初值 $x^{(0)}=\begin{bmatrix}-3\\0.5\end{bmatrix}$. (板书)

为了改善最速下降法的收敛性质, 需要对下降方向的选取方法进行改进. 由此, **共轭方向法**应运而生.

定义 4.8 设 $A\in\mathbb{R}^{n\times n}$ 对称正定, 若非零向量 $x, y\in\mathbb{R}^n$ 满足

$$y^{\mathsf{T}}Ax=0,$$

则称 x 和 y 是 A-共轭的, 也称 A-正交的.

✍ 若 $A=I$, 则 A-共轭就是通常意义下的正交.

下面我们说明用相互 A-共轭的向量作为下降方向的优点. 设 $p_1, p_2, \dots, p_n$ 相互 A-共轭, 则容易证明 $p_1, p_2, \dots, p_n$ 线性无关, 因此构成 $\mathbb{R}^n$ 的一组基. 对任意给定的初值 $x^{(0)}$, 我们可以设

$$x_*-x^{(0)}=\alpha_1p_1+\alpha_2p_2+\cdots+\alpha_np_n. \tag{4.11}$$

两边分别左乘 $p_k^{\mathsf{T}}A$, 可得

$$\alpha_k=\frac{p_k^{\mathsf{T}}A\left(x_*-x^{(0)}\right)}{p_k^{\mathsf{T}}Ap_k}=\frac{p_k^{\mathsf{T}}\left(b-Ax^{(0)}\right)}{p_k^{\mathsf{T}}Ap_k},\quad k=1,2,\dots,n. \tag{4.12}$$

✍ 如果采用一组正交基作为下降方向, 则可以得到类似的结论, 但此时的线性表出系数 α_i 的表达式中会含有 x_*. 这也是为什么要采用 A-共轭向量作为下降方向.

这意味着我们可以通过下降方向 $p_1, p_2, \dots, p_n$ 和右端项 b, 以及初值 $x^{(0)}$ 将方程组的解表示出来.

实际计算时, 由于 n 通常很大, 一般情况下不需要把所有 α_k 都计算出来, 只要计算前面部分, 得到一个满足精度要求的近似解即可. 因此, 我们通常把 (4.11) 的计算看作是一个迭代过程, 即

$$x^{(k)}=x^{(k-1)}+\alpha_kp_k,\quad k=1,2,\dots.$$

我们把 α_k 的计算公式 (4.12) 也改写一下. 由于

$$x^{(k-1)}-x^{(0)}=\alpha_1p_1+\alpha_2p_2+\cdots+\alpha_{k-1}p_{k-1},$$

所以 $p_k^{\mathsf{T}}A\left(x^{(k-1)}-x^{(0)}\right)=0$, 即 $p_k^{\mathsf{T}}Ax^{(0)}=p_k^{\mathsf{T}}Ax^{(k-1)}$, 因此

$$\alpha_k=\frac{p_k^{\mathsf{T}}\left(b-Ax^{(0)}\right)}{p_k^{\mathsf{T}}Ap_k}=\frac{p_k^{\mathsf{T}}\left(b-Ax^{(k-1)}\right)}{p_k^{\mathsf{T}}Ap_k}=\frac{p_k^{\mathsf{T}}r_{k-1}}{p_k^{\mathsf{T}}Ap_k}.$$

以相互 A-共轭的向量作为下降方向的线搜索方法就称为**共轭方向法**. 相比于最速下降法的局部最优性 (4.10), 共轭方向法具有全局最优性.

定理 4.25 设 $x^{(k)}$ 是由共轭方向法得到的迭代解, 则

$$x^{(k)} = \underset{x\in x^{(0)}+\mathrm{span}\{p_1,p_2,\ldots,p_k\}}{\operatorname{argmin}} \Phi(x).$$

(留作课外自习, 可以参见最优化方法的相关教材, 如 [37])

显然, 由 (4.11) 可知, 在不考虑舍入误差的情况下, 共轭方向法具有有限步终止性质.

定理 4.26 设 $A \in \mathbb{R}^{n\times n}$ 对称正定, 如果不考虑舍入误差, 则共轭方向法至多经过 n 迭代后, 近似解 $x^{(k)}$ 就是精确解 x_*.

✍ 在共轭方向法中, 虽然我们仍然称 p_k 为下降方向或搜索方向, 但我们不要求 p_k 是单位向量, 即不需要进行单位化, 因为这对算法的构造和实施没有任何影响.

现在剩下的问题就是如何构造相互 A-共轭的下降方向. 首先需要指出的是, 这样的向量组并不唯一. 事实上, 与 Gram-Schmidt 正交化过程类似, 任何一组线性无关的向量组都可以构造出一组相互 A-共轭的向量组. 目前最成功的是共轭梯度 (Conjugate Gradient) 方向. 它是通过递推方法来构造的, 可以看作是将最速下降方向进行 A-共轭化, 下面给出具体描述.

给定迭代初值 $x^{(0)}$, 令 p_1 为 $\Phi(x)$ 在 $x^{(0)}$ 处的负梯度方向

$$p_1 = -\nabla\Phi\big(x^{(0)}\big) = b - Ax^{(0)} = r_0,$$

首先以 p_1 为下降方向, 计算近似解 $x^{(1)}$, 即

$$x^{(1)} = x^{(0)} + \alpha_1 p_1, \quad \text{其中 } \alpha_1 = \frac{p_1^{\mathsf{T}} r_0}{p_1^{\mathsf{T}} A p_1}.$$

然后计算 $\Phi(x)$ 在 $x^{(1)}$ 处的负梯度方向, 即

$$-\nabla\Phi\big(x^{(1)}\big) = r_1 = b - Ax^{(1)} = r_0 - \alpha_1 A p_1.$$

将其与 p_1 进行 A-共轭化, 得到下降方向 p_2

$$p_2 = r_1 + \beta_1 p_1, \quad \text{其中 } \beta_1 = -\frac{r_1^{\mathsf{T}} A p_1}{p_1^{\mathsf{T}} A p_1}.$$

以此类推, 我们就可以得到相互 A-共轭的下降方向 $p_1, p_2, \ldots, p_k, \ldots$. 该过程同时也把近似解 $x^{(k)}$ 和残量 r_k 计算出来了.

在计算 p_{k+1} 时, 需要将 r_k 与所有 $p_1, p_2, \ldots, p_k$ 进行 A-共轭化. 事实上, 由于 A 对称正定, 我们只需将 r_k 与 p_k 进行 A-共轭化即可, 即

$$p_{k+1} = r_k + \beta_k p_k, \quad \text{其中} \quad \beta_k = -\frac{r_k^{\mathsf{T}} A p_k}{p_k^{\mathsf{T}} A p_k}.$$

这样不仅可以简化计算, 同时可以证明, p_{k+1} 与 $p_1, p_2, \ldots, p_{k-1}$ 都 A-共轭 (见定理 4.27).

于是共轭梯度法可描述为

$$\begin{cases} x^{(k)} = x^{(k-1)} + \alpha_k p_k, \quad \text{其中} \quad \alpha_k = \dfrac{p_k^{\mathsf{T}} r_{k-1}}{p_k^{\mathsf{T}} A p_k}, \\ r_k = b - A x^{(k)} = r_{k-1} - \alpha_k A p_k, \\ p_{k+1} = r_k + \beta_k p_k, \quad \text{其中} \quad \beta_k = -\dfrac{r_k^{\mathsf{T}} A p_k}{p_k^{\mathsf{T}} A p_k}, \end{cases} \qquad k = 1, 2, \ldots, \tag{4.13}$$

其中 $p_1 = r_0$.

定理 4.27 设 $A \in \mathbb{R}^{n\times n}$ 对称正定, 共轭梯度法 (4.13) 迭代 m 步后 ($m < n$), 对任意 k ($1 \le k \le m$) 有

(1) $r_k^{\mathsf{T}} r_i = 0, \quad i = 0, 1, 2, \ldots, k-1$;
(2) $r_k^{\mathsf{T}} p_i = 0, \quad i = 1, 2, \ldots, k$;
(3) $p_{k+1}^{\mathsf{T}} A p_i = 0, \quad i = 1, 2, \ldots, k$;
(4) $\operatorname{span}\{r_0, r_1, \ldots, r_k\} = \operatorname{span}\{p_1, p_2, \ldots, p_{k+1}\} = \operatorname{span}\{r_0, Ar_0, \ldots, A^k r_0\}$.

(留作课外自习, 可以参见矩阵计算或最优化方法的相关教材, 如 [76, 78])

利用上述定理中的性质, 我们还可以进一步简化计算, 提高计算效率.

推论 4.28 共轭梯度法 (4.13) 中的 α_k 和 β_k 可以通过下面的公式计算

$$\alpha_k = \frac{r_{k-1}^{\mathsf{T}} r_{k-1}}{p_k^{\mathsf{T}} A p_k}, \quad \beta_k = \frac{r_k^{\mathsf{T}} r_k}{r_{k-1}^{\mathsf{T}} r_{k-1}}.$$

(留作练习)

于是我们得到完整的共轭梯度法.

算法 4.5. 共轭梯度法 (Conjugate Gradient Algorithm)

1: 给定初值 $x^{(0)}$
2: 计算 $r_0 = b - Ax^{(0)}$, 并令 $p_1 = r_0$, $k = 1$
3: **while** not converge **do**
4: 计算 $x^{(k)} = x^{(k-1)} + \alpha_k p_k, \quad \text{其中} \quad \alpha_k = \dfrac{r_{k-1}^{\mathsf{T}} r_{k-1}}{p_k^{\mathsf{T}} A p_k}$
5: 计算 $r_k = r_{k-1} - \alpha_k A p_k$
6: 计算 $p_{k+1} = r_k + \beta_k p_k, \quad \text{其中} \quad \beta_k = \dfrac{r_k^{\mathsf{T}} r_k}{r_{k-1}^{\mathsf{T}} r_{k-1}}$
7: **end while**

对于共轭梯度法, 我们有下面的收敛性质.

定理 4.29 设 $A \in \mathbb{R}^{n\times n}$ 对称正定, 则对任意初值 $x^{(0)}$, 共轭梯度法都收敛, 且

$$\frac{\|x^{(k)} - x_*\|_A}{\|x^{(0)} - x_*\|_A} \le 2\left(\frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1}\right)^k,$$

其中 κ 为 A 的谱条件数, 范数 $\|x\|_A \triangleq \sqrt{x^{\mathsf{T}} A x}$.

(留作课外自习, 可以参见矩阵计算的相关教材, 如 [49, 76]))

由此可知, 共轭梯度法每次迭代的误差下降率为 $\dfrac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}$.

例 4.10 用共轭梯度法求解 $Ax=b$, 其中 $A=\begin{bmatrix}2 & 1\\ 1 & 3\end{bmatrix}$, $b=\begin{bmatrix}3\\ 4\end{bmatrix}$, 初值 $x^{(0)}=\begin{bmatrix}-3\\ 0.5\end{bmatrix}$. (板书)

例 4.11 用共轭梯度法求解离散二维 Poission 方程.

(Iter_CG_gallery.m)

5 非线性方程数值解法

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]

$$\frac{1}{\sqrt{x}}=\frac{1}{k}\ln\left(\mathrm{Re}\cdot\sqrt{x}\right)+\left(14-\frac{5.6}{k}\right),$$

其中 k 是某个常数, 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, 大家所熟悉的**代数方程**, 即

$$f(x)=a_0+a_1x+a_2x^2+\cdots+a_nx^n=0,$$

其中 $a_0,a_1,\ldots,a_n$ 是给定的系数. 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.

✍ 非线性方程一般不存在直接解法, 通常需要使用迭代法来求数值解.

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.

我们本章所研究内容就是在**有解**的前提下求出方程 $f(x)=0$ 的**近似解**.

✍ 如果没有特别说明, 我们这里只考虑实数解, 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.

非线性方程相关参考文献

[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 [33]

[4] C. T. Kelley, *Solving Nonlinear Equations with Newton's Method*, 2003 [34]

[5] 范金燕, 袁亚湘, **非线性方程组数值方法**, 2018 [67]

本章主要内容包括:

- 对分法
- 不动点迭代法
- Aitken 加速技巧与 Steffensen 迭代法
- Newton 法及其各种变形
- 割线法与抛物线法

5.1 对分法

5.1.1 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) 的基本思想就是将这个有解区间进行对分, 即分成两个小区间, 然后找出解所在的小区间, 将其记为新的有解区间, 其长度为原来有解区间的一半. 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**根的存在定理** (**介值定理**的推论).

定理 5.1 *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 ξ, 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 5.1. 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** $k=1,2,\ldots,$ **do**
5: 计算 $x_k=\dfrac{a_k+b_k}{2}$ 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 $a_{k+1}=a_k, b_{k+1}=x_k$; 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.

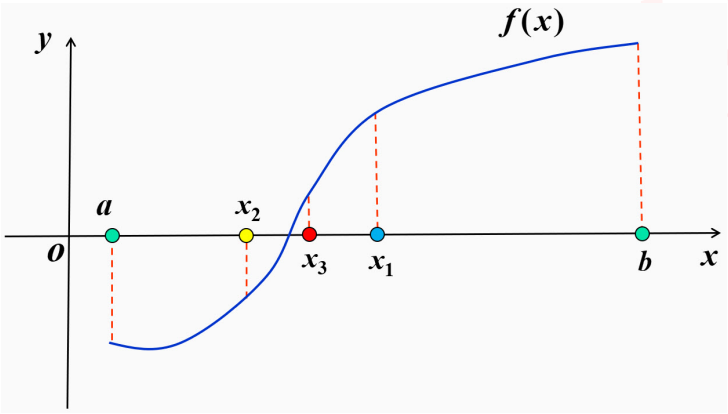


图 5.1. 对分法的几何表示

5.1.2 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

$$|x_k-x_*|=\left|\frac{1}{2}(a_k+b_k)-x_*\right|\le\frac{1}{2}(b_k-a_k).$$

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

$$|x_k-x_*|\le\frac{1}{2^k}(b_1-a_1)=\frac{1}{2^k}(b-a).$$

所以

$$\lim_{k\to\infty}|x_k-x_*|=0,$$

即算法收敛. 因此我们就有下面的收敛性结论.

定理 5.2 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 5.1 用对分法求 $f(x)=x^3-x-1=0$ 在 $[1,2]$ 内的根. (NLS_bisection.m)

5.2 不动点迭代法

5.2.1 基本思想

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:

$$x_{k+1} = \varphi(x_k), \quad k = 0, 1, 2, \ldots, \tag{5.1}$$

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.

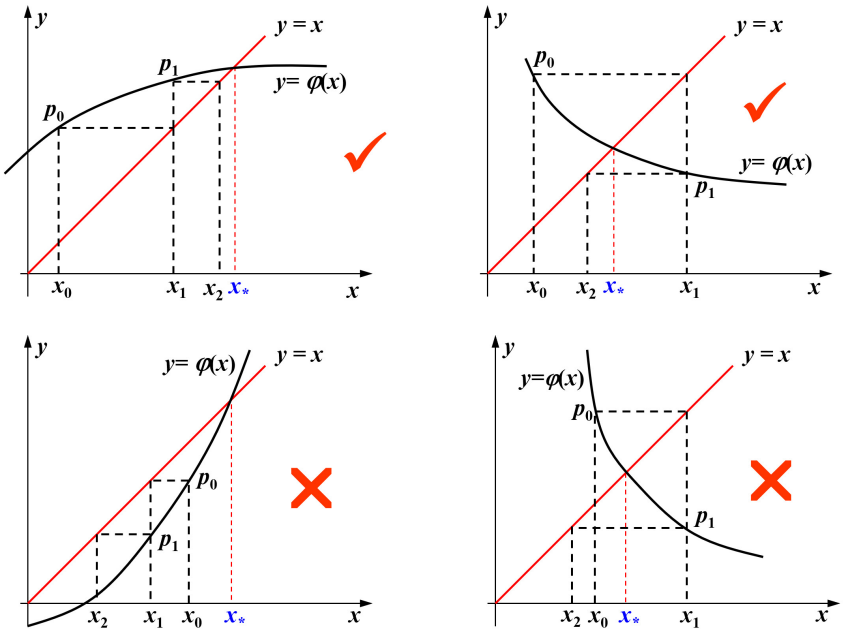


图 5.2. 不同迭代函数的不动点迭代过程几何表示

5.2.2 收敛性分析

设 $\varphi(x)$ 连续, 迭代 (5.1) 生成的点列为 $x_0, x_1, x_2, \ldots, x_k, \ldots$, 如果存在 x_* 使得

$$\lim_{k\to\infty} x_k = x_*,$$

则由连续函数的性质可知

$$x_* = \lim_{k\to\infty} x_{k+1} = \lim_{k\to\infty} \varphi(x_k) = \varphi\left(\lim_{k\to\infty} x_k\right) = \varphi(x_*).$$

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 (5.1) 是 **发散** 的.

全局收敛性

定理 5.3 (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. (板书)

✍ 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 5.4 (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 (5.1) 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 $[a,b]$ 内的唯一不动点. (板书)

✍ 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 5.5 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 (5.1) 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 5.2 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)

✍ 需要指出的是, 定理 5.4 和推论 5.5 中给出的是充分条件, 不是充要条件.

局部收敛性

定义 5.1 设 x_* 是 $\varphi(x)$ 的不动点, 若存在 x_* 的某个 δ-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 (5.1) 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, 然后再进行迭代.

定理 5.6 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 (5.1) 局部收敛. (板书)

5.2.3 收敛阶

收敛阶是衡量迭代法收敛速度快慢的一个重要指标.

定义 5.2 设迭代 $x_{k+1}=\varphi(x_k)$ 收敛到不动点 x_*. 记 $e_k \triangleq x_k - x_*$, 若

$$\lim_{k\to\infty} \frac{|e_{k+1}|}{|e_k|^p} = c,$$

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 5.6 可以立即得到下面的结论.

定理 5.7 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 (5.1) 局部线性收敛.

关于收敛阶的判定, 我们有下面的定理.

定理 5.8 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

$$\varphi'(x_*) = \varphi''(x_*) = \cdots = \varphi^{(p-1)}(x_*) = 0, \quad \varphi^{(p)}(x_*) \neq 0, \tag{5.2}$$

则不动点迭代 (5.1) 是 p 阶局部收敛的, 且有

$$\lim_{k\to\infty} \frac{x_{k+1} - x_*}{x_k - x_*} = \frac{\varphi^{(p)}(x_*)}{p!}.$$

(板书)

例 5.3 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 $\varphi(x) = x - \dfrac{x^2 - 3}{4}$, 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 $\varphi(x) = \dfrac{1}{2}\left(x + \dfrac{3}{x}\right)$, 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:

$$x_{k+1} = \varphi(x_k, x_{k-1}, ..., x_{k-\ell+1}), \quad k = \ell - 1, \ell, \ell + 1, \ldots. \tag{5.3}$$

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 ℓ 个初始值, 即 $x_0, x_1, \ldots, x_{\ell-1}$.

5.3 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.

5.3.1 Aitken 加速技巧

已有不动点迭代

$$x_{k+1} = \varphi(x_k). \tag{5.4}$$

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:

$$y_{k+1} = x_k - \frac{(x_{k+1} - x_k)^2}{x_{k+2} - 2x_{k+1} + x_k}. \tag{5.5}$$

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 5.9 假定迭代 5.4 收敛, 且 $\varphi'(x_*) \neq 1$, 则

$$\lim_{k\to\infty} \frac{y_{k+1} - x_*}{x_k - x_*} = 0.$$

这意味着 y_k 比 x_k 更快地收敛到 x_*.

5.3.2 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:

$$y_k = \varphi(x_k), \quad z_k = \varphi(y_k), \quad x_{k+1} = x_k - \frac{(y_k - x_k)^2}{z_k - 2y_k + x_k}, \quad k = 0, 1, 2, \ldots.$$

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 5.10 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 5.4 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 5.5 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)

5.4 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.

5.4.1 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

$$x_{k+1}=x_k-\frac{f(x_k)}{f'(x_k)}. \tag{5.6}$$

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:

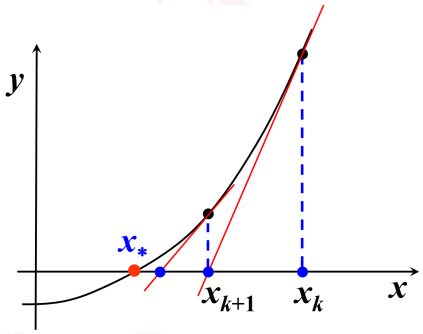


图 5.3. Newton 法的几何含义

✍ 为了使得 Newton 法能顺利进行, 一般要求 $f'(x)\neq 0$.

算法 5.2. Newton 法

给定迭代初值 x_0, 精度要求 ε 和最大迭代步数 IterMax
if $|f(x_0)|<\varepsilon$, **then**
    输出近似解 x_0, 停止迭代
end if
for $k=1$ to IterMax **do**
    计算 $x_1=x_0-\dfrac{f(x_0)}{f'(x_0)}$
    if $|x_1-x_0|<\varepsilon$ 或 $|f(x_1)|<\varepsilon$, **then**
        输出近似解 x_1, 停止迭代 % 算法收敛

9: **end if**
10: $x_0 = x_1$
11: **end for**

5.4.2 Newton 法的收敛性

由迭代格式 (5.6) 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 5.8, 我们可以立即得到下面的结论.

定理 5.11 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

$$\lim_{k\to\infty} \frac{x_{k+1} - x_*}{(x_k - x_*)^2} = \frac{\varphi''(x_*)}{2} = \frac{f''(x_*)}{2f'(x_*)}.$$

(板书)

例 5.6 编写程序, 用 Newton 法求 $f(x) = xe^x - 1$ 的零点. (NLS_Newton_01.m)

例 5.7 用 Newton 法求 $f(x) = x^2 - C = 0$ 的正根, 并判断收敛性, 其中 $C > 0$.

✍ 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, 难度和工作量都可能会比较大.

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

$$L_{new} = \frac{1}{2}(L + C/L),$$

新的宽就是 C/L_{new}. 这样不断做下去, 长方形就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.

5.4.3 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

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_0)}, \quad k = 0, 1, 2, \ldots.$$

但这样做的代价是收敛速度只有线性收敛.

5.4.4 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

$$|f(x_{k+1})| < |f(x_k)|, \tag{5.7}$$

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** λ, 即

$$x_{k+1} = x_k - \lambda\frac{f(x_k)}{f'(x_k)}, \quad k = 0, 1, 2, \ldots.$$

下山因子 λ 的取法: 从 $\lambda = 1$ 开始, 逐次减半, 直到满足下降条件 (5.7) 为止.

5.4.5 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 5.8 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)

5.5 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).

5.5.1 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:

$$x_{k+1} = x_k - \frac{x_k - x_{k-1}}{f(x_k) - f(x_{k-1})} f(x_k), \quad k = 1, 2, \ldots.$$

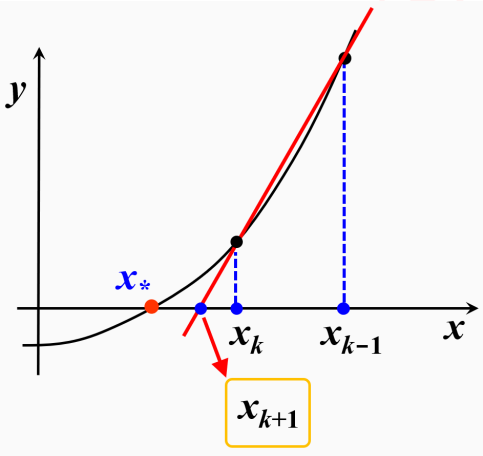


图 5.4. 割线法的几何含义

✍ 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 5.12 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 δ 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.

5.5.2 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 Hermitian 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}.

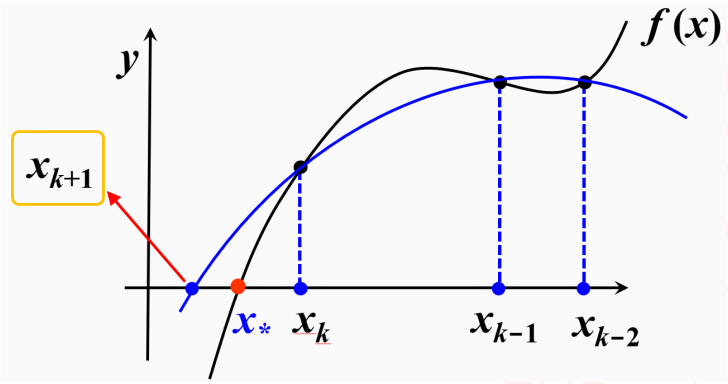


图 5.5. 抛物线法的几何含义

✍ 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

$$p \approx 1.840. \qquad (p^3 - p^2 - p - 1 = 0)$$

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 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 6.1 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.

6.1 多项式插值

什么是插值

定义 6.1 已知函数 $y = f(x)$ 在区间 $[a, b]$ 上有定义, 且已经测得其在点

$$a \leq x_0 < x_1 < \cdots < x_n \leq b \tag{6.1}$$

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

$$p(x_i) = y_i, \quad i = 0, 1, \ldots, n, \tag{6.2}$$

则称 $p(x)$ 为 $f(x)$ 的插值函数. 区间 $[a, b]$ 称为插值区间, x_i $(i = 0, 1, \ldots, n)$ 称为插值节点, 条件 (6.2) 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 6.2 (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 $y_0 = f(x_0), \cdots, y_n = f(x_n)$. 多项式插值就是寻找一个次数不超过 n 的多项式的函数

$$p(x) = c_0 + c_1 + \cdots + c_n x^n, \tag{6.3}$$

使得

$$p(x_i) = y_i, \quad i = 0, 1, \ldots, n.$$

✍ 需要指出的是, $p(x)$ 的次数有可能小于 n.

定理 6.1 (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 (6.3). 将插值条件 $p(x_i) = y_i$ 代入, 得到一个关于 $c_0, c_1, \ldots, c_n$ 的线性方程组, 其系数矩阵正好是一个关于 $x_0, x_1, \ldots, x_n$ 的 Vandermonde 矩阵. 因此当插值节点互不相同时, 其行列式不为 0. 所以系数矩阵可逆, 即解存在唯一. □

✍ 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 6.2 线性插值: 求一个一次多项式 $p(x)$, 满足:

$$p(x_0) = y_0, \quad p(x_1) = y_1.$$

例 6.3 抛物线插值: 求一个二次多项式 $p(x)$, 满足:

$$p(x_0) = y_0, \quad p(x_1) = y_1, \quad p(x_2) = y_2.$$

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

$$\mathbb{H}_2(x) \triangleq \{\text{次数不超过 } 2 \text{ 的多项式的全体}\}$$

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

$$\mathbb{H}_n(x) \triangleq \{\text{所有次数不超过 } n \text{ 的实系数多项式组成的集合}\},$$

则 $\mathbb{H}_n$ 构成一个 $n+1$ 维的线性空间. 易知, n 次多项式插值就是在 $\mathbb{H}_n$ 中寻找一个多项式 $p(x)$, 使得插值条件 (6.2) 成立.

设 $\{\phi_0(x), \phi_1(x), \ldots, \phi_n(x)\}$ 是 $\mathbb{H}_n$ 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 $a_0, a_1, \ldots, a_n$ 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.

6.2 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.

6.2.1 Lagrange 基函数

定义 6.3 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

$$l_k(x_i) = \begin{cases} 1, & i = k \\ 0, & i \neq k \end{cases} \qquad i, k = 0, 1, 2, \dots, n. \tag{6.4}$$

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 (6.4) 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 α 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 (6.2) 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

$$L_n(x) = \sum_{k=0}^{n} y_k l_k(x) = \sum_{k=0}^{n} y_k \prod_{i=0,\, i\neq k}^{n} \frac{x - x_i}{x_k - x_i}. \tag{6.5}$$

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 6.4 已知函数 $f(x) = \ln(x)$ 的函数值如下: (Interp_Lagrange_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

试分别用线性插值和抛物线插值计算 $\ln(0.54)$ 的近似值.

✍ 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.

6.2.2 插值余项

首先复习一下数学分析中的罗尔 (Roll) 定理.

定理 6.2 (罗尔 (Roll) 中值定理) 设 $f(x)$ 满足: (1) 在 $[a, b]$ 上连续, (2) 在 (a, b) 内可导, (3) $f(a) = f(b)$, 则在 (a, b) 内至少存在一点 ξ, 使得

$$f'(\xi) = 0.$$

记 Lagrange 插值多项式的余项为

$$R_n(x) \triangleq f(x) - L_n(x).$$

定理 6.3 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

$$R_n(x) = \frac{f^{(n+1)}(\xi_x)}{(n+1)!}\, \omega_{n+1}(x), \tag{6.6}$$

其中 $\omega_{n+1}(x) = (x - x_0)(x - x_1)\cdots(x - x_n)$. (板书)

✍ 余项中的 ξ_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;
- ξ_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

  如果有 $\max\limits_{a\le x\le b}|f^{(n+1)}(x)|=M_{n+1}$, 则 $|R_n(x)|\le\dfrac{M_{n+1}}{(n+1)!}|\omega_{n+1}(x)|$.
- 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.

例 6.5 数据同例 6.5, 试估计用线性插值和抛物线插值计算 $\ln(0.54)$ 时的误差.

思考: 由上例可知, 抛物线插值要优于线性插值, 是不是插值多项式次数越高, 插值精度就越好?

例 6.6 Runge 现象: 已知函数 $f(x)=\dfrac{1}{1+x^2}$, 插值区间 $[-5,5]$, 取等距插值节点, 即 $x_i=-5+\dfrac{10i}{n}$, $i=0,1,2,\dots,n$, 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_Lagrange_Runge.m)

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).

例 6.7 设 $f(x)\in C^2[a,b]$, 证明:

$$\max_{a\le x\le b}\left|f(x)-\left[f(a)+\frac{f(b)-f(a)}{b-a}(x-a)\right]\right|\le\frac{1}{8}M_2(b-a)^2,$$

其中 $M_2=\max\limits_{a\le x\le b}|f''(x)|$.

6.2.3 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 6.1 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 $0 \le m \le n$, 则由性质 6.1 可知 $f(x) = L_n(x)$, 即

$$\sum_{k=0}^{n} x_k^m l_k(x) = x^m, \quad 0 \le m \le n. \tag{6.7}$$

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 6.8 设 $l_i(x)$ 是关于点 $x_0, x_1, \ldots, x_5$ 的 Lagrange 插值基函数. 证明:

$$\sum_{i=0}^{5}(x_i - x)^2 l_i(x) = 0.$$

6.3 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

$$\begin{aligned}
\phi_0(x) &= 1 \\
\phi_1(x) &= x - x_0 \\
\phi_2(x) &= (x - x_0)(x - x_1) \\
&\cdots\cdots \\
\phi_n(x) &= (x - x_0)(x - x_1)\cdots(x - x_{n-1}).
\end{aligned}$$

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 $\mathbb{H}_n(x)$ 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

$$\phi_{n+1}(x) = (x - x_0)(x - x_1)\cdots(x - x_{n-1})(x - x_n).$$

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 $\mathbb{H}_n(x)$ 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**.

6.3.1 差商

定义 6.4 设节点 $x_0, x_1, \ldots, x_n$, 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

$$f[x_0, x_1, \ldots, x_k] = \frac{f[x_1, x_2, \ldots, x_k] - f[x_0, x_1, \ldots, x_{k-1}]}{x_k - x_0}$$

为 $f(x)$ 关于点 $x_0, x_1, \ldots, x_k$ 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)

$$f[x_0, x_1, \ldots, x_k] = \sum_{j=0}^{k} \frac{f(x_j)}{\prod\limits_{i=0, i\neq j}^{k} (x_j - x_i)} = \sum_{j=0}^{k} \frac{f(x_j)}{\omega'_{k+1}(x_j)}, \tag{6.8}$$

  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**:

$$f[x_0, x_1, \ldots, x_k] = f[x_{i_0}, x_{i_1}, \ldots, x_{i_k}]$$

  其中 $i_0, i_1, \ldots, i_k$ 是 $0, 1, \ldots, k$ 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:

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

- 若 $h(x) = \alpha f(x)$, 其中 α 是常数, 则

$$h[x_0, x_1, \ldots, x_k] = \alpha f[x_0, x_1, \ldots, x_k].$$

- 若 $h(x) = f(x) + g(x)$, 则

$$h[x_0, x_1, \ldots, x_k] = f[x_0, x_1, \ldots, x_k] + g[x_0, x_1, \ldots, x_k].$$

- 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

$$f[x_0, x_1, \ldots, x_k] = \frac{f^{(k)}(\xi)}{k!}. \tag{6.9}$$

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	$\cdots$	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	$f(x_n)$	$f[x_{n-1},x_n]$	$f[x_{n-2},x_{n-1},x_n]$	$f[x_{n-3},x_{n-2},x_{n-1},x_n]$	$\cdots$	$f[x_0,x_1,\ldots,x_n]$

例 6.9 已知 $y=f(x)$ 的函数值表如下, 试计算其各阶差商 (Interp_newton_dq.m)

i	0	1	2	3
x_i	-2	-1	1	2
$f(x_i)$	5	3	17	21

6.3.2 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

$$f(x)=f(x_0)+f[x,x_0](x-x_0). \tag{6.10}$$

同理, 由

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

可得

$$f[x,x_0]=f[x_0,x_1]+f[x,x_0,x_1](x-x_1). \tag{6.11}$$

以此类推, 我们有

$$f[x,x_0,x_1]=f[x_0,x_1,x_2]+f[x,x_0,x_1,x_2](x-x_2) \tag{6.12}$$

$$\vdots$$

$$f[x,x_0,\ldots,x_{n-1}]=f[x_0,x_1,\ldots,x_n]+f[x,x_0,x_1,\ldots,x_n](x-x_n). \tag{6.13}$$

将等式 (6.10)-(6.13) 联立可得 (依次将后面一式代入前面一式)

$$\begin{aligned}f(x)=f(x_0)&+f[x_0,x_1](x-x_0)\\&+f[x_0,x_1,x_2](x-x_0)(x-x_1)\\&+\cdots\\&+f[x_0,x_1,\ldots,x_n](x-x_0)\cdots(x-x_{n-1})\\&+f[x,x_0,\ldots,x_n](x-x_0)\cdots(x-x_{n-1})(x-x_n).\end{aligned}$$

所以

$$f(x) = N_n(x) + R_n(x),$$

其中

$$N_n(x) = a_0 + a_1(x - x_0) + \cdots + a_n(x - x_0)\cdots(x - x_{n-1}), \tag{6.14}$$

$$a_0 = f(x_0), \quad a_i = f[x_0, x_1, \ldots, x_i], \quad i = 1, 2, \ldots, n.$$

$$R_n(x) = f[x, x_0, \ldots, x_n](x - x_0)\cdots(x - x_{n-1})(x - x_n).$$

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

$$R_n(x_i) = 0, \quad i = 0, 1, 2, \ldots, n.$$

所以

$$f(x_i) = N_n(x_i) + R_n(x_i) = N_n(x_i), \quad i = 0, 1, 2, \ldots, n.$$

即 $N_n(x)$ 是满足插值条件 (6.2) 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:

$$N_{n+1}(x) = N_n(x) + f[x_0, x_1, \ldots, x_{n+1}]\prod_{i=0}^{n}(x - x_i).$$

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

$$f[x, x_0, \ldots, x_n]\prod_{i=0}^{n}(x - x_i) \equiv \frac{f^{(n+1)}(\xi_x)}{(n+1)!}\prod_{i=0}^{n}(x - x_i).$$

由此, 我们立即可以得到下面的结论.

性质 6.2 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

$$f[x, x_0, \ldots, x_n] = \frac{f^{(n+1)}(\xi_x)}{(n+1)!}.$$

这就是我们前面提到的差商与导数之间的关系 (6.9), 即

$$f[x_0, x_1, \ldots, x_k] = \frac{f^{(k)}(\xi)}{k!}.$$

✍ 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 $f[x, x_0, \ldots, x_n]$ 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 6.10 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.

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!

6.3.3 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:

$$x_i = x_0 + i \times h, \quad i = 0, 1, 2, \ldots, n, \tag{6.15}$$

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 6.5 (向前差分) 设 $\{x_i\}$ 是由 (6.15) 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

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

一般地, 对于任意 $m+1$ 个相邻的节点, 有

$$f[x_k, x_{k+1}, \ldots, x_{k+m}] = \frac{1}{m!}\frac{1}{h^m}\Delta^m f_k.$$

所以由差商与导数之间的关系 (6.9) 可知

$$\Delta^m f_k = h^m m! \times f[x_k, x_{k+1}, \ldots, x_{k+m}] = h^m f^{(m)}(\xi), \quad \xi \in (x_k, x_{k+m}).$$

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	$\cdots$	n阶差分
x_0	$f(x_0)$	Δf_0	$\Delta^2 f_0$	$\Delta^3 f_0$	$\cdots$	$\Delta^n f_0$
x_1	$f(x_1)$	Δf_1	$\Delta^2 f_1$	$\Delta^3 f_1$		
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$		
x_{n-2}	$f(x_{n-2})$	Δf_{n-2}	$\Delta^2 f_{n-2}$			
x_{n-1}	$f(x_{n-1})$	Δf_{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 $x_i = x_0 + i\,h$, 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

$$\begin{aligned} N_n(x) &= N_n(x_0 + th) \\ &= f(x_0) + t\Delta f_0 + \frac{t(t-1)}{2!}\Delta^2 f_0 + \frac{t(t-1)(t-2)}{3!}\Delta^3 f_0 + \cdots \\ &\quad + \frac{t(t-1)\cdots(t-n+1)}{n!}\Delta^n f_0. \end{aligned} \tag{6.16}$$

这就是**Newton 向前插值公式**. 插值余项为

$$R_n(x) = \frac{f^{(n+1)}(\xi_x)}{(n+1)!}t(t-1)\cdots(t-n)h^{n+1}.$$

例 6.11 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`)

✍ 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**:

$$\begin{aligned} \nabla f_i &= f(x_i) - f(x_{i-1}), \\ \nabla^k f_i &= \nabla(\nabla^{k-1} f_i) = \nabla^{k-1} f_i - \nabla^{k-1} f_{i-1}, \quad k = 2, 3, \ldots \end{aligned}$$

和**中心差分**:

$$\begin{aligned} \delta f_i &= f(x_{i+\frac{1}{2}}) - f(x_{i-\frac{1}{2}}), \\ \delta^k f_i &= \delta(\delta^{k-1} f_i) = \delta^{k-1} f_{i+\frac{1}{2}} - \delta^{k-1} f_{i-\frac{1}{2}}, \quad k = 2, 3, \ldots. \end{aligned}$$

6.4 Hermitian 插值

为什么 Hermitian 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

$$p(x_i) = f(x_i),\ p'(x_i) = f'(x_i),\ p''(x_i) = f''(x_i),\ \cdots, p^{(m)}(x_i) = f^{(m)}(x_i).$$

则计算这类插值多项式的方法就称为**Hermitian 插值**.

✍ 在 Hermitian 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.

6.4.1 重节点差商与 Taylor 插值

首先介绍差商的一个重要性质.

定理 6.4 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, $f[x,y] = \dfrac{f(y)-f(x)}{y-x}$ 关于 x 和 y 都连续, 且当 $y \to x$ 时有

$$f[x,x] \triangleq \lim_{y\to x} f[x,y] = f'(x).$$

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

$$f[\underbrace{x,x,\dots,x}_{n+1\text{ 个}}] \triangleq \lim_{x_i\to x} f[x,x_1,x_2,\dots,x_n] = \frac{1}{n!}f^{(n)}(x).$$

在 Newton 插值公式 (6.14) 中, 令 $x_i \to x_0$, $i = 1,2,\dots,n$, 则

$$\begin{aligned} N_n(x) &= f(x_0) + f[x_0,x_1](x-x_0) + f[x_0,x_1,x_2](x-x_0)(x-x_1) + \cdots \\ &\quad + f[x_0,x_1,\dots,x_n](x-x_0)\cdots(x-x_{n-1}) \\ &= f(x_0) + f'(x_0)(x-x_0) + \frac{1}{2!}f''(x_0)(x-x_0)^2 + \cdots + \frac{1}{n!}f^{(n)}(x_0)(x-x_0)^n. \end{aligned}$$

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 Hermitian 插值.

6.4.2 两个典型的 Hermitian 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 Hermitian 插值多项式. 这里介绍两个典型的 Hermitian 插值: **三点三次 Hermitian 插值** 和 **两点三次 Hermitian 插值**.

(1) 三点三次 Hermitian 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 Hermitian 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

$$\begin{aligned} p(x) = {} & f(x_0) + f[x_0, x_1](x - x_0) + f[x_0, x_1, x_2](x - x_0)(x - x_1) \\ & + a(x - x_0)(x - x_1)(x - x_2). \end{aligned} \tag{6.17}$$

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 ξ_x 位于由 x_0, x_1, x_2 和 x 所界定的区间内.

例 6.12 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 Hermitian 插值多项式, 并写出余项. (计算过程中不要做近似计算)

(2) 两点三次 Hermitian 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 Hermitian 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

$$\begin{aligned} &\alpha_0(x_0) = 1,\ \alpha_0(x_1) = 0,\ \alpha_0'(x_0) = 0,\ \alpha_0'(x_1) = 0; \\ &\alpha_1(x_0) = 0,\ \alpha_1(x_1) = 1,\ \alpha_1'(x_0) = 0,\ \alpha_1'(x_1) = 0; \\ &\beta_0(x_0) = 0,\ \beta_0(x_1) = 0,\ \beta_0'(x_0) = 1,\ \beta_0'(x_1) = 0; \\ &\beta_1(x_0) = 0,\ \beta_1(x_1) = 0,\ \beta_1'(x_0) = 0,\ \beta_1'(x_1) = 1. \end{aligned}$$

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

$$\begin{aligned} H_3(x) = {} & f(x_0)\left(1 + 2\frac{x - x_0}{x_1 - x_0}\right)\left(\frac{x - x_1}{x_0 - x_1}\right)^2 + f(x_1)\left(1 + 2\frac{x - x_1}{x_0 - x_1}\right)\left(\frac{x - x_0}{x_1 - x_0}\right)^2 \\ & + f'(x_0)(x - x_0)\left(\frac{x - x_1}{x_0 - x_1}\right)^2 + f'(x_1)(x - x_1)\left(\frac{x - x_0}{x_1 - x_0}\right)^2. \end{aligned} \tag{6.18}$$

插值余项为

$$R_3(x) = \frac{f^{(4)}(\xi_x)}{4!}(x - x_0)^2(x - x_1)^2.$$

6.5 分段低次插值

为什么分段插值

由前面的例 6.6 (即 Runge 现象) 可知, 当 $n \to \infty$ 时, 插值多项式 $L_n(x)$ 并不一定收敛于 $f(x)$. 事实上, 对于例 6.6, 可以证明, 存在常数 $c \approx 3.63$, 当 $|x| \le c$ 时, $\lim\limits_{n\to\infty} L_n(x) = f(x)$, 而当 $|x| > c$ 时, $\{L_n(x)\}$ 发散.

我们自然会想到的问题是, 怎样才能构造出收敛的插值方法呢 (即插值误差趋于零)? 这就需要从插值余项入手. 通过观察插值余项公式, 我们发现插值余项有两部分组成: $f(x)$ 的 $n+1$ 阶导数和 $\omega_{n+1}(x)$, 即

$$\max_{a\le x\le b} |R_n(x)| = \max_{a\le x\le b} \left| \frac{f^{(n+1)}(\xi_x)}{(n+1)!}\,\omega_{n+1}(x) \right| \le \max_{a\le x\le b} \left| \frac{f^{(n+1)}(\xi_x)}{(n+1)!} \right| \cdot \max_{a\le x\le b} |\omega_{n+1}(x)|\,.$$

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 $\max\limits_{a\le x\le b} |\omega_{n+1}(x)|$ 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 Hermitian 插值**.

6.5.1 分段线性插值

定义 6.6 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 $f_0, f_1, \ldots, f_n$. 求分段函数 $I_h(x)$ 满足

(1) $I_h(x) \in C[a,b]$;

(2) $I_h(x_k) = f_k, \quad k = 0,1,2,\ldots,n$;

(3) $I_h(x)$ 在每个小区间 $[x_{k-1}, x_k]$ 上是线性多项式, $k = 1,2,\ldots,n$.

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 $[x_{k-1}, x_k]$ 上的表达式为

$$I_h(x) = \frac{x - x_k}{x_{k-1} - x_k} f_{k-1} + \frac{x - x_{k-1}}{x_k - x_{k-1}} f_k, \quad x \in [x_{k-1}, x_k], \qquad k = 1,2,\ldots,n, \tag{6.19}$$

且在 $[x_{k-1}, x_k]$ 上余项满足

$$\begin{aligned} \max_{x_{k-1}\le x\le x_k} |f(x) - I_h(x)| &\le \frac{1}{2!} \max_{x_{k-1}\le x\le x_k} |f''(x)| \cdot \max_{x_{k-1}\le x\le x_k} |(x - x_{k-1})(x - x_k)| \\ &\le \frac{h_k^2}{8} \max_{x_{k-1}\le x\le x_k} |f''(x)|, \end{aligned} \tag{6.20}$$

其中 $h_k = x_k - x_{k-1}$. 令 $h = \max\limits_{1\le k\le n} \{h_k\}$, 我们有下面的结论.

定理 6.5 若 $f(x)\in C^2[a,b]$, 则分段线性插值函数 $I_h(x)$ 满足

$$\max_{a\le x\le b}|f(x)-I_h(x)|\le\frac{M_2}{8}h^2,$$

其中 $M_2=\max\limits_{a\le x\le b}|f''(x)|$. 所以

$$\lim_{h\to0}I_h(x)=f(x)$$

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 (6.20) 可知

$$\max_{x_{k-1}\le x\le x_k}|f(x)-I_h(x)|\le\frac{h_k^2M_2}{8}\le\frac{M_2}{8}h^2.$$

所以

$$\max_{a\le x\le b}|f(x)-I_h(x)|\le\max_{1\le k\le n}\max_{x_{k-1}\le x\le x_k}|f(x)-I_h(x)|\le\max_{1\le k\le n}\frac{h^2M_2}{8}=\frac{M_2}{8}h^2.$$

□

✍ 分段线性插值的优点: 简单易用; 缺点: 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?

6.5.2 分段三次 Hermitian 插值

定义 6.7 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 $[x_{k-1},x_k]$ 上是三次多项式.

这就是**分段三次 Hermitian 插值**, $I_h(x)$ 就称为 $f(x)$ 在 $[a,b]$ 上的**分段三次 Hermitian 插值函数**.

由两点三次 Hermitian 插值公式 (6.18) 可知, $I_h(x)$ 在小区间 $[x_{k-1},x_k]$ 上的表达式为

$$\begin{aligned}I_h(x)=&\left(1+2\frac{x-x_{k-1}}{x_k-x_{k-1}}\right)\left(\frac{x-x_k}{x_{k-1}-x_k}\right)^2f_{k-1}\\&+\left(1+2\frac{x-x_k}{x_{k-1}-x_k}\right)\left(\frac{x-x_{k-1}}{x_k-x_{k-1}}\right)^2f_k\\&+(x-x_{k-1})\left(\frac{x-x_k}{x_{k-1}-x_k}\right)^2f_{k-1}'+(x-x_k)\left(\frac{x-x_{k-1}}{x_k-x_{k-1}}\right)^2f_k',\\&\qquad x\in[x_{k-1},x_k].\end{aligned}\tag{6.21}$$

由两点三次 Hermitian 插值法的余项公式, 可知

$$\max_{x_{k-1}\le x\le x_k}|f(x)-I_h(x)|\le\frac{h_k^4}{384}\max_{x_{k-1}\le x\le x_k}|f^{(4)}(x)|,$$

其中 $h_k=x_k-x_{k-1}$. 令 $h=\max\limits_{1\le k\le n}\{h_k\}$, 我们有下面的结论.

定理 6.6 若 $f(x) \in C^4[a,b]$, 则分段三次 Hermitian 插值函数 $I_h(x)$ 满足

$$\max_{a\le x\le b} |f(x) - I_h(x)| \le \frac{M_4}{384} h^4,$$

其中 $M_4 = \max\limits_{a\le x\le b} |f^{(4)}(x)|$. 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 Hermitian 插值比分段线性插值具有更高的精度, 且 h 越小, 误差越小.

✍ 分段三次 Hermitian 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 6.13 设 $f(x) = \dfrac{1}{1+x^2}$, $x \in [-5,5]$, 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 Hermitian 插值画出 $f(x)$ 的近似图像.

(Interp_piecewise_poly.m)

6.6 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 6.8 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 $f(x_k) = f_k, k = 0,1,\ldots,n$. 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) $S(x_k) = y_k, \quad k = 0,1,2,\ldots,n$;

(3) $S(x)$ 是分段三次函数, 即在每个小区间 $[x_{k-1}, x_k]$ 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.

6.6.1 三次样条函数

定义 6.9 (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 $[x_{k-1}, x_k]$ 上的表达式记为 $s_k(x)$, 即

$$S(x) = s_k(x), \quad x \in [x_{k-1}, x_k], \quad k = 1,2,\ldots,n,$$

其中 $s_k(x)$ 是三次多项式, 且满足

$$s_k(x_{k-1}) = f_{k-1}, \quad s_k(x_k) = f_k. \tag{6.22}$$

于是

$$S(x) = \begin{cases} s_1(x), & x \in [x_0, x_1] \\ s_2(x), & x \in [x_1, x_2] \\ \quad\vdots & \\ s_n(x), & x \in [x_{n-1}, x_n] \end{cases}. \tag{6.23}$$

由于 $S(x) \in C^2[a,b]$, 所以 $S'(x_k^-) = S'(x_k^+)$, $S''(x_k^-) = S''(x_k^+)$, 即

$$s_k'(x_k^-) = s_{k+1}'(x_k^+), \quad s_k''(x_k^-) = s_{k+1}''(x_k^+), \quad k = 1,2,\ldots,n-1. \tag{6.24}$$

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 (6.22) 和 (6.24) 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**.

6.6.2 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.

6.6.3 三次样条函数的计算

由于 $S(x)$ 二阶可导, 所以可设

$$S''(x_k) = M_k, \quad k = 0, 1, 2, \ldots, n,$$

下面我们用 M_k 来表示 $S(x)$. 考虑 $S(x)$ 在区间 $[x_{k-1}, x_k]$ 上的表达式 $s_k(x)$, 满足

$$s''_k(x_{k-1}) = M_{k-1}, \quad s''_k(x_k) = M_k.$$

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

$$s''_k(x) = \frac{x_k - x}{h_k} M_{k-1} + \frac{x - x_{k-1}}{h_k} M_k,$$

其中 $h_k = x_k - x_{k-1}$. 两边在 $[x_{k-1}, x_k]$ 上积分两次后可得

$$s_k(x) = \frac{(x_k - x)^3}{6h_k} M_{k-1} + \frac{(x - x_{k-1})^3}{6h_k} M_k + c_1 x + c_2, \tag{6.25}$$

其中 c_1, c_2 为积分常数. 将 $s_k(x_{k-1}) = f_{k-1}$, $s_k(x_k) = f_k$ 代入后可得

$$c_1 = \frac{1}{h_k}(f_k - f_{k-1}) - \frac{h_k}{6}(M_k - M_{k-1}) = \frac{1}{h_k}\left[\left(f_k - \frac{M_k h_k^2}{6}\right) - \left(f_{k-1} - \frac{M_{k-1} h_k^2}{6}\right)\right],$$

$$c_2 = f_{k-1} - \frac{M_{k-1} h_k^2}{6} - c_1 x_{k-1} = \frac{x_k}{h_k}\left(f_{k-1} - \frac{M_{k-1} h_k^2}{6}\right) - \frac{x_{k-1}}{h_k}\left(f_k - \frac{M_k h_k^2}{6}\right).$$

代入 (6.25), 整理后可得

$$\begin{aligned} s_k(x) = {} & \frac{(x_k - x)^3}{6h_k} M_{k-1} + \frac{(x - x_{k-1})^3}{6h_k} M_k \\ & + \frac{x_k - x}{h_k}\left(f_{k-1} - \frac{M_{k-1} h_k^2}{6}\right) + \frac{x - x_{k-1}}{h_k}\left(f_k - \frac{M_k h_k^2}{6}\right). \end{aligned} \tag{6.26}$$

即 $s_k(x)$ 可表示成 $x_k - x$ 和 $x - x_{k-1}$ 的奇次项的线性组合.

将 $x_k = x_{k-1} + h_k$ 代入 (6.26), 整理后可得

$$\begin{aligned} s_k(x) = &\frac{M_k - M_{k-1}}{6h_k}(x - x_{k-1})^3 + \frac{M_{k-1}}{2}(x - x_{k-1})^2 \\ &+ \left(\frac{f_k - f_{k-1}}{h_k} - \frac{h_k(M_k + 2M_{k-1})}{6}\right)(x - x_{k-1}) + f_{k-1}. \end{aligned} \tag{6.27}$$

✍ 现在, 问题转化为如何确定 $M_0, M_1, \ldots, M_n$ 的值?

由于 $S(x) \in C^2[a,b]$, 所以在节点处的一阶导数存在, 故

$$S'(x_k^-) = S'(x_k^+), \quad k = 1, 2, \ldots, n-1,$$

也即

$$s_k'(x_k^-) = s_{k+1}'(x_k^+).$$

所以可得方程

$$\frac{h_k}{6}M_{k-1} + \frac{h_k + h_{k+1}}{3}M_k + \frac{h_{k+1}}{6}M_{k+1} = \frac{f_{k+1} - f_k}{h_{k+1}} - \frac{f_k - f_{k-1}}{h_k}.$$

为了书写方便, 我们记

$$\begin{aligned} &\mu_k = \frac{h_k}{h_k + h_{k+1}}, \quad \lambda_k = \frac{h_{k+1}}{h_k + h_{k+1}}, \\ &d_k = \frac{6(f[x_k, x_{k+1}] - f[x_{k-1}, x_k])}{h_k + h_{k+1}} = 6f[x_{k-1}, x_k, x_{k+1}], \end{aligned}$$

则上面的方程可写为

$$\mu_k M_{k-1} + 2M_k + \lambda_k M_{k+1} = d_k, \quad k = 1, 2, \ldots, n-1. \tag{6.28}$$

✍ 上述方程中的系数有个重要性质: $\mu_k + \lambda_k = 1$

✍ 方程 (6.28) 也可以写成

$$h_k M_{k-1} + 2(h_k + h_{k+1})M_k + h_{k+1}M_{k+1} = 6(f[x_k, x_{k+1}] - f[x_{k-1}, x_k])$$

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

$$s_1'(x_0^+) = f_0', \quad s_n'(x_n^-) = f_n'.$$

因此可得

$$2M_0 + M_1 = \frac{6}{h_1}(f[x_0, x_1] - f_0')$$

$$M_{n-1}+2M_n=\frac{6}{h_n}(f'_n-f[x_{n-1},x_n]).$$

令 $d_0=\frac{6}{h_1}(f[x_0,x_1]-f'_0)$, $d_n=\frac{6}{h_n}(f'_n-f[x_{n-1},x_n])$, 则上式与 (6.28) 联立可得方程组

$$\begin{bmatrix} 2 & 1 & & & \\ \mu_1 & 2 & \lambda_1 & & \\ & \ddots & \ddots & \ddots & \\ & & \mu_{n-1} & 2 & \lambda_{n-1} \\ & & & 1 & 2 \end{bmatrix} \begin{bmatrix} M_0 \\ M_1 \\ \vdots \\ M_{n-1} \\ M_n \end{bmatrix} = \begin{bmatrix} d_0 \\ d_1 \\ \vdots \\ d_{n-1} \\ d_n \end{bmatrix}. \tag{6.29}$$

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.

(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

$$\begin{bmatrix} 2 & \lambda_1 & & & \\ \mu_2 & 2 & \lambda_2 & & \\ & \ddots & \ddots & \ddots & \\ & & \mu_{n-2} & 2 & \lambda_{n-2} \\ & & & \mu_{n-1} & 2 \end{bmatrix} \begin{bmatrix} M_1 \\ M_2 \\ \vdots \\ M_{n-2} \\ M_{n-1} \end{bmatrix} = \begin{bmatrix} d_1-\mu_1 f''_0 \\ d_2 \\ \vdots \\ d_{n-2} \\ d_{n-1}-\lambda_{n-1}f''_n \end{bmatrix}. \tag{6.30}$$

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

$$s'_1(x_0^+)=s'_n(x_n^-),\quad s''_1(x_0^+)=s''_n(x_n^-).$$

可得

$$\lambda_n M_1+\mu_n M_{n-1}+2M_n=d_n,\quad M_0=M_n,$$

其中

$$\lambda_n=\frac{h_0}{h_0+h_{n-1}},\quad \mu_n=\frac{h_{n-1}}{h_0+h_{n-1}},\quad d_n=\frac{6(f[x_0,x_1]-f[x_{n-1},x_n]}{h_1+h_n}.$$

与 (6.28) 联立可得方程组

$$\begin{bmatrix} 2 & \lambda_1 & & & \mu_1 \\ \mu_2 & 2 & \lambda_2 & & \\ & \ddots & \ddots & \ddots & \\ & & \mu_{n-1} & 2 & \lambda_{n-1} \\ \lambda_n & & & \mu_n & 2 \end{bmatrix} \begin{bmatrix} M_1 \\ M_2 \\ \vdots \\ M_{n-1} \\ M_n \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_{n-1} \\ d_n \end{bmatrix}. \tag{6.31}$$

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 (6.29), (6.30) 和 (6.31) 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 $M_0, M_1, \ldots, M_n$ 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 (6.27), 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

$$s_k(x) = a_3(x - x_{k-1})^3 + a_2(x - x_{k-1})^2 + a_1(x - x_{k-1}) + a_0,$$

因此, 我们在计算时也可以将 $s_k(x)$ 写成上述形式, 即 (6.27) 式.

✍ 三次样条插值的优点: 二阶连续可导, 且只需利用插值节点上的函数值, 加上边界条件.

例 6.14 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

例 6.15 函数 $f(x) = \dfrac{1}{1+x^2}$, 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 6.16 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

6.6.4 误差估计

定理 6.7 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

$$\max_{a\le x\le b}|f(x)-S(x)|\le\frac{5}{384}\max_{a\le x\le b}|f^{(4)}(x)|h^4,$$

$$\max_{a\le x\le b}|f'(x)-S'(x)|\le\frac{1}{24}\max_{a\le x\le b}|f^{(4)}(x)|h^3,$$

$$\max_{a\le x\le b}|f''(x)-S''(x)|\le\frac{3}{8}\max_{a\le x\le b}|f^{(4)}(x)|h^2,$$

其中 $h=\max\limits_{0\le k\le n-1}\{h_k\}$.

(证明可参见相关资料)

✍ 该定理说明, 当 $h\to 0$ 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 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 [74]

[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 [69]

[3] L. N. Trefethen, *Approximation Theory and Approximation Practice*, 2019 [55]

7.1 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 Φ 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 Φ 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 Φ 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 7.1 (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

$$\max_{a \le x \le b} |f(x) - p(x)| < \varepsilon$$

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 [74]. □

✍ 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 7.1 设 Φ 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 Φ 中的 $[a,b]$ 上的**最佳逼近函数**.

✍ $g^*(x)$ 与函数空间 Φ, 范数 $\|\cdot\|$ 和区间 $[a,b]$ 有关.

定义 7.2 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 $f(x_i) = y_i$ $(i = 0, 1, 2, \ldots, m)$, 在某个函数空间 Φ 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 Φ 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.

7.2 正交多项式

7.2.1 正交函数族与正交多项式

定义 7.3 (正交函数) 设 $f(x), g(x) \in C[a,b]$, $\rho(x)$ 是 $[a,b]$ 上的权函数, 若

$$(f,g) = \int_a^b \rho(x) f(x) g(x)\,\mathrm{d}x = 0,$$

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 7.4 (正交函数族) 设 $\varphi_0(x), \varphi_1(x), \ldots, \varphi_n(x), \ldots \in C[a,b]$, $\rho(x)$ 是 $[a,b]$ 上的权函数, 若

$$(\varphi_i, \varphi_j) = \int_a^b \rho(x)\varphi_i(x)\varphi_j(x)\,\mathrm{d}x = \begin{cases} 0, & i \neq j \\ A_i > 0, & i = j \end{cases} \quad i,j = 0,1,2,\ldots,$$

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 7.1 三角函数系

$$1, \quad \cos x, \quad \sin x, \quad \cos 2x, \quad \sin 2x, \ldots$$

在 $[-\pi, \pi]$ 上是带权 $\rho(x) = 1$ 的正交函数族.

定义 7.5 (正交多项式) 设 $p_n(x)$ 是首项系数不为零的 n 次多项式, $n = 0,1,2,\ldots$, $\rho(x)$ 是 $[a,b]$ 上的权函数, 若对 $i,j = 0,1,2,\ldots$ 有

$$(p_i, p_j) = \int_a^b \rho(x) p_i(x) p_j(x)\,\mathrm{d}x = \begin{cases} 0, & i \neq j, \\ A_i > 0, & i = j, \end{cases}$$

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 $p_0(x), p_1(x), p_2(x), \ldots, p_n(x)$ 线性无关, 所以它们构成 $\mathbb{H}_n$ 的一组**正交基**.

由于 $p_n(x)$ 与 $p_0(x), p_1(x), \ldots, p_{n-1}(x)$ 都正交, 故 $p_n(x)$ 与 H_{n-1} 中的任意多项式都正交, 即

$$(p_n(x), p(x)) = \int_a^b \rho(x) p_n(x) p(x)\,\mathrm{d}x = 0, \quad \forall\, p(x) \in H_{n-1}. \tag{7.1}$$

下面我们给出一个正交多项式的三项递推公式.

定理 7.2 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 $p_0(x)=1, p_1(x)=x-\alpha_0$,

$$\alpha_n=\frac{(xp_n,p_n)}{(p_n,p_n)},\ n=0,1,2,\dots,\quad \beta_n=\frac{(p_n,p_n)}{(p_{n-1},p_{n-1})},\ n=1,2,\dots.$$

(板书)

✍ 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 7.3 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. (板书)

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:

$$p_0(x)=1,$$
$$p_k(x)=x^k-\sum_{j=0}^{k-1}c_{kj}p_j(x),\quad c_{kj}=\frac{(x^k,p_j)}{(p_j,p_j)},\quad k=1,2,3,\dots.$$

7.2.2 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

$$P_0(x)=1,\quad P_n(x)=\frac{1}{2^n n!}\frac{\mathrm{d}^n}{\mathrm{d}x^n}(x^2-1)^n,\quad x\in[-1,1],\quad n=1,2,\dots. \tag{7.2}$$

- $P_n(x)$ 的首项系数为 $\dfrac{(2n)!}{2^n(n!)^2}$;
- 若令 $\tilde{P}_n(x)=\dfrac{2^n(n!)^2}{(2n)!}P_n(x)$, 则称 $\tilde{P}_n(x)$ 为**首项系数为 1 的 Legendre 多项式**.

定理 7.4 (正交性)

$$(P_n, P_m) = \int_a^b P_n(x)P_m(x)\,\mathrm{d}x = \begin{cases} 0, & m \neq n \\ \dfrac{2}{2n+1}, & m = n \end{cases} \tag{7.3}$$

(板书)

定理 7.5 (奇偶性) $P_{2k}(x)$ 只含偶次幂, $P_{2k+1}(x)$ 只含奇次幂, 故

$$P_n(-x) = (-1)^n P_n(x).$$

(板书)

定理 7.6 (递推公式)

$$(n+1)P_{n+1}(x) = (2n+1)xP_n(x) - nP_{n-1}(x), \quad n = 1, 2, \ldots,$$

其中 $P_0(x) = 1, P_1(x) = x$. (证明与定理 7.2 类似)

定理 7.7 (零点) $P_n(x)$ 在 $(-1, 1)$ 内有 n 个不同的零点.

(直接由定理 7.3 可得)

例 7.2 给出 5 次 Legendre 多项式 $P_5(x)$ 的表达式. (Approxi_Legendre.m)

7.2.3 Chebyshev 多项式

设 $[a, b] = [-1, 1]$, 权函数 $\rho(x) = \dfrac{1}{\sqrt{1-x^2}}$, 将线性无关函数组 $\{1, x, x^2, \ldots, x^n, \ldots\}$ 正交化后得到的正交多项式就是 **Chebyshev 多项式**, 记为 $T_0(x), T_1(x), T_2(x), \ldots$.

Chebyshev 多项式的一般形式为

$$T_n(x) = \cos(n \arccos x), \quad x \in [-1, 1], \quad n = 1, 2, \ldots.$$

- 显然, $|T_n(x)| \leq 1$.
- 令 $x = \cos\theta$, $\theta \in [0, \pi]$, 则 $\theta = \arccos x$, 所以

$$\begin{aligned} T_n(x) = \cos(n\theta) &= \cos^n\theta - C_n^2 \cos^{n-2}\theta \sin^2\theta + C_n^4 \cos^{n-4}\theta \sin^4\theta + \cdots \\ &= x^n - C_n^2 x^{n-2}(1-x^2) + C_n^4 x^{n-4}(1-x^2)^2 + \cdots. \end{aligned}$$

故 $T_n(x)$ 是 n 次多项式.

定理 7.8 (正交性)

$$(T_n, T_m) = \int_{-1}^1 \rho(x)T_n(x)T_m(x)\,\mathrm{d}x = \begin{cases} 0, & m \neq n \\ \dfrac{\pi}{2}, & m = n \neq 0, \\ \pi, & m = n = 0. \end{cases}$$

(留作练习)

定理 7.9 (递推公式)

$$T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x), \quad n = 1, 2, \ldots$$

其中 $T_0(x) = 1, T_1(x) = x$. (板书)

定理 7.10 (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 7.11 (零点) $T_n(x)$ 在 $(-1, 1)$ 内有 n 个不同的零点:

$$x_k = \cos\frac{2k-1}{2n}\pi, \quad k = 1, 2, \ldots, n.$$

(板书)

定理 7.12 (极值点) $T_n(x)$ 在 $[-1, 1]$ 内有 $n+1$ 个极值点 (含两个端点):

$$\tilde{x}_k = \cos\frac{k\pi}{n}, \quad k = 0, 1, 2, \ldots, n.$$

(板书)

定理 7.13 $T_n(x)$ 的首项系数为 2^{n-1}.

(直接由递推公式可得)

✍ 令 $\tilde{T}_n(x) = \dfrac{1}{2^{n-1}}T_n(x)$, 则称 $\tilde{T}_n(x)$ 为**首项系数为 1 的 Chebyshev 多项式**.

例 7.3 给出 5 次 Chebyshev 多项式 $T_5(x)$ 的表达式. (Approxi_Chebyshev.m)

7.2.4 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 7.14 设 $\tilde{T}_n(x)$ 是首项系数为 1 的 Chebyshev 多项式, 即 $\tilde{T}_n(x) = \dfrac{1}{2^{n-1}}T_n(x)$, 则

$$\max_{-1\le x\le 1}|\tilde{T}_n(x)| \le \max_{-1\le x\le 1}|p(x)|, \quad \forall\, p(x) \in \tilde{\mathbb{H}}_n,$$

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

$$\max_{-1\le x\le 1}|\tilde{T}_n(x)| = \frac{1}{2^{n-1}}.$$

(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 7.14, 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

$$\max_{-1\le x\le 1} |f(x) - L_n(x)| \le \frac{M_{n+1}}{(n+1)!} \max_{-1\le x\le 1} |\omega_{n+1}(x)|,$$

其中 $M_{n+1} = \max\limits_{-1\le x\le 1} |f^{(n+1)}(x)|$. 因此, 要使得插值误差最小化, 我们就需要

$$\max_{-1\le x\le 1} |\omega_{n+1}(x)| = \|\omega_{n+1}(x)\|_\infty$$

尽可能地小. 由定理 7.14 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 $\dfrac{1}{2^n}$. 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 7.15 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

$$\|f(x) - L_n(x)\|_\infty \le \frac{1}{2^n(n+1)!}\|f^{(n+1)}(x)\|_\infty. \tag{7.4}$$

✍ 上面的定理表明: 若 $f(x) \in C^{n+1}[-1,1]$, 则当 $n \to \infty$ 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

$$x_k = \frac{b-a}{2}t_k + \frac{b+a}{2} = \frac{b-a}{2}\cos\frac{2k+1}{2(n+1)}\pi + \frac{b+a}{2}, \quad k = 0, 1, 2, \dots, n, \tag{7.5}$$

总体插值误差

$$\begin{aligned}\max_{a\le x\le b} |f(x) - L_n(x)| &\le \frac{1}{2^n(n+1)!} \max_{-1\le t\le 1} \left|\frac{\mathrm{d}^{n+1} f}{\mathrm{d}t^{n+1}}\right| \\ &= \frac{1}{2^n(n+1)!}\frac{(b-a)^{n+1}}{2^{n+1}} \max_{-1\le t\le 1} |f^{(n+1)}(x(t))|\end{aligned}$$

$$= \frac{(b-a)^{n+1}}{2^{2n+1}(n+1)!} \max_{a\le x\le b} |f^{(n+1)}(x)|.$$

✍ 上面的总体误差也可以直接将插值点 (7.5) 代入 $\omega_{n+1}(x)$ 后获得. 事实上, 由于

$$x - x_k = \frac{b-a}{2}(t - t_k), \quad k = 0, 1, 2, \ldots, n,$$

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

$$\max_{a\le x\le b} |\omega_{n+1}(x)| = \frac{(b-a)^{n+1}}{2^{n+1}} \max_{-1\le t\le 1} \tilde{T}_{n+1}(t) = \frac{(b-a)^{n+1}}{2^{n+1}} \cdot \frac{1}{2^n}$$

✍ 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 7.4 求 $f(x) = \mathrm{e}^x$ 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 7.5 设 $f(x) = \dfrac{1}{1+x^2}$, 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. (Approxi_Chebyshev_interp.m)

7.2.5 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

$$U_n(x) = \frac{\sin((n+1)\arccos x)}{\sqrt{1-x^2}}, \quad x \in [-1, 1], \quad n = 0, 1, 2, \ldots. \tag{7.6}$$

- **正交性**:

$$(U_n, U_m) = \int_{-1}^{1} \rho(x) U_n(x) U_m(x)\,\mathrm{d}x = \begin{cases} 0, & m \ne n \\ \dfrac{\pi}{2}, & m = n \end{cases}$$

- **递推公式**:

$$U_{n+1}(x) = 2xU_n(x) - U_{n-1}(x), \quad n = 1, 2, \ldots,$$

其中 $U_0(x) = 1, U_1(x) = 2x$.

Laguerre 多项式

在区间 $[0, \infty]$ 上, 带权 $\rho(x) = \mathrm{e}^{-x}$ 正交的多项式就称为 **Laguerre 多项式**, 其一般表达式为

$$L_n(x) = \mathrm{e}^x \frac{\mathrm{d}^n}{\mathrm{d}x^n}(x^n \mathrm{e}^{-x}), \quad x \in [0, \infty], \quad n = 0, 1, 2, \ldots.$$

- **正交性**:

$$(L_n, L_m) = \int_0^\infty \rho(x) L_n(x) L_m(x)\,\mathrm{d}x = \begin{cases} 0, & m \neq n \\ (n!)^2, & m = n \end{cases}$$

- **递推公式**:

$$L_{n+1}(x) = (2n + 1 - x)L_n(x) - n^2 L_{n-1}(x), \quad n = 1, 2, \ldots,$$

其中 $L_0(x) = 1, L_1(x) = 1 - x$.

Hermitian 多项式

在区间 $[-\infty, \infty]$ 上, 带权 $\rho(x) = \mathrm{e}^{-x^2}$ 正交的多项式就称为 **Hermitian 多项式**, 其一般表达式为

$$H_n(x) = (-1)^n \mathrm{e}^{x^2} \frac{\mathrm{d}^n}{\mathrm{d}x^n}(\mathrm{e}^{-x^2}), \quad x \in [-\infty, \infty], \quad n = 0, 1, 2, \ldots.$$

- **正交性**:

$$(H_n, H_m) = \int_{-\infty}^\infty \rho(x) H_n(x) H_m(x)\,\mathrm{d}x = \begin{cases} 0, & m \neq n \\ 2^n n! \sqrt{n}, & m = n \end{cases}$$

- **递推公式**:

$$H_{n+1}(x) = 2xH_n(x) - 2nH_{n-1}(x), \quad n = 1, 2, \ldots,$$

其中 $H_0(x) = 1, H_1(x) = 2x$.

7.3 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 Φ 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

$$\|f(x)-S(x)\|_2^2=\int_a^b\rho(x)(f(x)-S(x))^2\,\mathrm{d}x.$$

7.3.1 怎样求最佳平方逼近

设 $\varphi_0(x),\varphi_1(x),\dots,\varphi_n(x)$ 是 Φ 的一组基, 则对任意 $S(x)\in\Phi$, $S(x)$ 可表示为

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

因此

$$\|f(x)-S(x)\|_2^2=\int_a^b\rho(x)\left(f(x)-\sum_{i=0}^n a_i\varphi_i(x)\right)^2\mathrm{d}x\triangleq I(a_0,a_1,\dots,a_n).$$

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

$$2\int_a^b\rho(x)\left(f(x)-\sum_{i=0}^n a_i\varphi_i(x)\right)\varphi_k(x)\,\mathrm{d}x=0,\tag{7.7}$$

也即

$$\sum_{i=0}^n a_i\int_a^b\rho(x)\varphi_i(x)\varphi_k(x)\,\mathrm{d}x=\int_a^b\rho(x)f(x)\varphi_k(x)\,\mathrm{d}x.$$

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

$$\begin{bmatrix}(\varphi_0,\varphi_0)&(\varphi_0,\varphi_1)&\cdots&(\varphi_0,\varphi_n)\\(\varphi_1,\varphi_0)&(\varphi_1,\varphi_1)&\cdots&(\varphi_1,\varphi_n)\\\vdots&\vdots&&\vdots\\(\varphi_n,\varphi_0)&(\varphi_n,\varphi_1)&\cdots&(\varphi_n,\varphi_n)\end{bmatrix}\begin{bmatrix}a_0\\a_1\\\vdots\\a_n\end{bmatrix}=\begin{bmatrix}(\varphi_0,f)\\(\varphi_1,f)\\\vdots\\(\varphi_n,f)\end{bmatrix}.\tag{7.8}$$

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 ?? 可知, 法方程 (7.8) 的系数矩阵 G 非奇异, 因此法方程存在唯一解.

- 求 $S^*(x) \Longleftrightarrow$ 解法方程 $Ga = d$.
- 存在唯一解 $\Longleftrightarrow G$ 非奇异 $\Longleftrightarrow \varphi_0, \varphi_1, \ldots, \varphi_n$ 线性无关.
- $S^*(x)$ 是 $f(x)$ 在 Φ 中的最佳逼近 $\Longleftrightarrow (f - S^*, \varphi_k) = 0, k = 0, 1, \ldots, n \Longleftrightarrow (f - S^*, S) = 0, \forall\, S \in \Phi$.

定理 7.16 设 $a_0^*, a_1^*, \ldots, a_n^*$ 是法方程 (7.8) 的解, 则 $S^*(x)$ 是 $f(x)$ 在 Φ 中的最佳平方逼近函数, 其中

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

(板书)

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 (7.7) 可知 $(f - S^*, \varphi_k) = 0$, 因此

$$\|\delta(x)\|_2^2 = (f - S^*, f - S^*) = (f - S^*, f) = \|f\|_2^2 - \sum_{i=0}^{n} a_i^*(\varphi_i, f).$$

7.3.2 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 $\varphi_0(x), \varphi_1(x), \ldots, \varphi_n(x)$ 是 Φ 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

$$a_k^* = \frac{(\varphi_k, f)}{(\varphi_k, \varphi_k)}, \quad k = 0, 1, 2, \ldots, n.$$

于是 $f(x)$ 在 Φ 中的最佳平方逼近函数为

$$S^*(x) = \sum_{k=0}^{n} \frac{(\varphi_k, f)}{(\varphi_k, \varphi_k)}\varphi_k(x), \tag{7.9}$$

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

$$a_0^*\varphi_0(x)+a_1^*\varphi_1(x)+\cdots+a_n^*\varphi_n(x)+\cdots \tag{7.10}$$

其中 $a_k^*=\dfrac{(\varphi_k,f)}{(\varphi_k,\varphi_k)}$. 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

$$1,\cos x,\sin x,\cos 2x,\sin 2x,\ldots,$$

则级数 (7.10) 就是 Fourier 级数.

7.3.3 最佳平方逼近多项式

设 $\Phi=\mathbb{H}_n$ (次数不超过 n 的所有多项式组成的集合), 则 $f(x)$ 在 Φ 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 7.6 设 $[a,b]=[0,1]$, 权函数 $\rho(x)\equiv 1$, 取 $\mathbb{H}_n$ 的一组基 $\{1,x,x^2,\ldots,x^n\}$. 求 $f(x)\in C[0,1]$ 的 n 次最佳平方逼近多项式.

例 7.7 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.

7.3.4 用正交多项式计算最佳平方逼近多项式

定理 7.17 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 (7.9) 给出的 n 次最佳平方逼近多项式, 则

$$\lim_{n\to\infty}\|f(x)-S_n^*(x)\|_2=0,$$

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 7.18 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

$$S_n^*(x)=a_0^*P_0(x)+a_1^*P_1(x)+\cdots+a_n^*P_n(x),$$

其中 $P_k(x)$ 为 k 次 Legendre 多项式,

$$a_k^*=\frac{(P_k,f)}{(P_k,P_k)}=\frac{2k+1}{2}\int_{-1}^{1}P_k(x)f(x)\,\mathrm{d}x.$$

误差

$$\|\delta_n(x)\|_2^2 = \|f(x) - S_n^*(x)\|_2^2 = \|f(x)\|_2^2 - \sum_{k=0}^{n} a_k^*(P_k, f) = \int_{-1}^{1} f^2(x)\,\mathrm{d}x - \sum_{k=0}^{n} \frac{2\,(a_k^*)^2}{2k+1}.$$

✍ 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 7.8 设 $f(x) = \mathrm{e}^x$, 求 $f(x)$ 在 $[-1, 1]$ 上的三次最佳平方逼近多项式.

(板书)

定理 7.19 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 7.20 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

$$\|\tilde{P}_n(x)\|_2 = \min_{p(x) \in \tilde{\mathbb{H}}_n} \|p(x)\|_2 = \min_{p(x) \in \tilde{\mathbb{H}}_n} \left(\int_{-1}^{1} p^2(x)\,\mathrm{d}x \right)^{\frac{1}{2}},$$

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 7.14) 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 $x(t) = \dfrac{b-a}{2} t + \dfrac{b+a}{2}$, 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 $t = \dfrac{2x - b - a}{b - a}$ 代入 $S^*(t)$, 给出 $f(x)$ 在 $[a, b]$ 上的最佳平方逼近多项式

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

例 7.9 举个例子. *To be continued ...*

7.4 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 Φ 中的**最佳一致逼近函数**. 若 $\Phi=\mathbb{H}_n$, 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

$$\|f(x)-S(x)\|_\infty=\max_{a\le x\le b}|f(x)-S(x)|.$$

7.4.1 最佳一致逼近多项式的存在唯一性

定理 7.21 (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

$$f(x_i)-p_n^*(x_i)=(-1)^i\max_{a\le x\le b}|p_n^*(x)-f(x)|,\quad i=0,1,2,\dots,n+1.$$

(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 7.10 设 $f(x)\in C[a,b]$, 则 $f(x)$ 的零次最佳一致逼近多项式为

$$p_0^*(x)=\frac{1}{2}\left(\min_{a\le x\le b}f(x)+\max_{a\le x\le b}f(x)\right).$$

例 7.11 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.

当 $n\ge 2$ 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.

7.4.2 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 7.14), 构造其 $n-1$ 次的最佳一致逼近多项式.

定理 7.22 设 $f(x)=a_nx^n+a_{n-1}x^{n-1}+\cdots+a_1x+a_0$, 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. (留作练习)

例 7.12 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 7.22 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: 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?

7.4.3 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 $\rho(x)=\dfrac{1}{\sqrt{1-x^2}}$, 在 $f(x)$ 的广义 Fourier 级数 (7.10) 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

$$a_k^*=\frac{2}{\pi}\int_{-1}^{1}\frac{T_k(x)f(x)}{\sqrt{1-x^2}}\,\mathrm{d}x=\frac{2}{\pi}\int_0^{\pi}f(\cos\theta)\cos k\theta\,\mathrm{d}\theta,\quad k=0,1,2,\ldots.$$

注意: 由于 $(T_0,T_0)=\pi$, 因此这里的 $a_0^*=2\dfrac{(T_0,f)}{(T_0,T_0)}$.

定理 7.23 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 $\tilde{\mathbb{H}}_{n+1}$ 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 7.13 求 $f(x)=\mathrm{e}^x$ 在 $[-1,1]$ 上的 Chebyshev 级数部分和 $C_3^*(x)$.

(板书)

7.5 最小二乘曲线拟合

7.5.1 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 7.14 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

7.5.2 最小二乘与法方程

给定数据

x	x_0	x_1	x_2	$\cdots$	x_{m-1}	x_m
y	y_0	y_1	y_2	$\cdots$	y_{m-1}	y_m

在函数族 $\Phi \triangleq \operatorname{span}\{\varphi_0(x), \varphi_1(x), \ldots, \varphi_n(x)\}$ 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

$$\max_{0\le i\le m} |S^*(x_i) - y_i| = \min_{S(x)\in\Phi} \max_{0\le i\le m} |S(x_i) - y_i|.$$

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

$$\min_{S(x)\in\Phi}\sum_{i=0}^{m}\omega_i|S(x_i)-y_i|^2, \tag{7.11}$$

其中 ω_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 (7.11) 的求解. 对任意 $S(x)\in\Phi=\operatorname{span}\{\varphi_0(x),\varphi_1(x),\dots,\varphi_n(x)\}$, 可设

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

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

$$\sum_{j=0}^{n}\left[\sum_{i=0}^{m}\omega_i\varphi_k(x_i)\varphi_j(x_i)\right]a_j=\sum_{i=0}^{m}\omega_i y_i\varphi_k(x_i),\quad k=0,1,2,\dots,n.$$

引入记号

$$(\varphi_j,\varphi_k)\triangleq\sum_{i=0}^{m}\omega_i\varphi_j(x_i)\varphi_k(x_i),\quad (y,\varphi_k)\triangleq\sum_{i=0}^{m}\omega_i y_i\varphi_k(x_i), \tag{7.12}$$

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**:

$$Ga=d, \tag{7.13}$$

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

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

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 (7.12) 中引入的记号 (φ_j, φ_k) 并不构成 $C[a,b]$ 或 Φ 中的内积, 因此仅仅凭借 $\varphi_0(x), \varphi_1(x), \ldots, \varphi_n(x)$ 的线性无关并不能推出 G 是非奇异的.

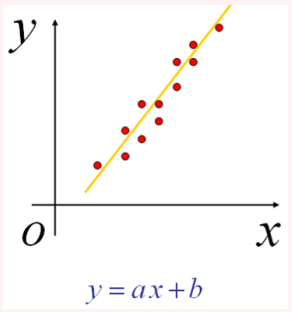
定理 7.24 设 $\varphi_0(x), \varphi_1(x), \ldots, \varphi_n(x) \in C[a,b]$ 线性无关. 如果 $\varphi_0(x), \varphi_1(x), \ldots, \varphi_n(x)$ 的任意 (非零) 线性组合在点集 $\{x_0, x_1, \ldots, x_m\}$ 上至多只有 n 个不同的零点, 则 G 非奇异, 此时法方程 (7.13) 存在唯一解.

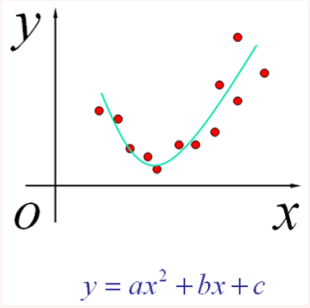
✍ 上述定理中的条件称为 **Haar 条件**, 显然, 如果取 $\varphi_i(x) = x^i$, 则 Haar 条件成立.

例 7.15 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

✍ 对于曲线拟合问题, 如何选择数学模型很重要 (即函数空间 Φ, 也即基函数 $\varphi_0(x), \varphi_1(x), \ldots, \varphi_n(x)$), 通常需要根据物理意义或所给数据的分布情况来确定.





7.5.3 多项式拟合

在数据拟合时, 如果取 $\varphi_0(x) = 1, \varphi_1(x) = x, \ldots, \varphi_n(x) = x^n$, 即 $\Phi = \mathbb{H}_n$, 则相应的法方程为

$$\begin{bmatrix} \sum\limits_{i=0}^{m} \omega_i & \sum\limits_{i=0}^{m} \omega_i x_i & \cdots & \sum\limits_{i=0}^{m} \omega_i x_i^n \\ \sum\limits_{i=0}^{m} \omega_i x_i & \sum\limits_{i=0}^{m} \omega_i x_i^2 & \cdots & \sum\limits_{i=0}^{m} \omega_i x_i^{n+1} \\ \vdots & \vdots & & \vdots \\ \sum\limits_{i=0}^{m} \omega_i x_i^n & \sum\limits_{i=0}^{m} \omega_i x_i^{n+1} & \cdots & \sum\limits_{i=0}^{m} \omega_i x_i^{2n} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} \sum\limits_{i=0}^{m} \omega_i y_i \\ \sum\limits_{i=0}^{m} \omega_i x_i y_i \\ \cdots \\ \sum\limits_{i=0}^{m} \omega_i x_i^n y_i \end{bmatrix}.$$

设解为 $[a_0^*, a_1^*, \ldots, a_n^*]^{\mathsf{T}}$, 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 7.16 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

✍ 如果没有给出权系数 ω_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.

7.5.4 用正交多项式做最小二乘拟合

带权正交 (离散情形)

定义 7.6 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

$$(\varphi_k,\varphi_j)\triangleq\sum_{i=0}^m\omega_i\varphi_k(x_i)\varphi_j(x_i)=\begin{cases}0, & k\neq j,\\ A_k\neq 0, & k=j,\end{cases}\tag{7.14}$$

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 (7.14) 所定义的关系式是否构成内积?

定理 7.25 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

$$S_n^*(x)=a_0^*p_0(x)+a_1^*p_1(x)+a_2^*p_2(x)+\cdots+a_n^*p_n(x),$$

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 (7.14) 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:

$$\begin{aligned}&p_0(x)=1,\quad p_1(x)=x-\alpha_0,\\ &p_{k+1}(x)=(x-\alpha_k)p_k(x)-\beta_kp_{k-1}(x),\quad k=1,2,\dots,\end{aligned}\tag{7.16}$$

其中

$$\alpha_k = \frac{(xp_k, p_k)}{(p_k, p_k)}, \quad k = 0, 1, \ldots, \qquad \beta_k = \frac{(p_k, p_k)}{(p_{k-1}, p_{k-1})}, \quad k = 1, 2, \ldots.$$

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 7.17 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

(Approxi_datafit_orth_poly_01.m)

- 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).

7.6 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

$$R_{nm}(x) \triangleq \frac{p_n(x)}{q_m(x)} = \sum_{k=0}^{n} a_k x^k / \sum_{k=0}^{m} b_k x^k.$$

- 最佳有理一致逼近:

$$\min \|R_{nm} - f(x)\|_\infty$$

- 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 7.7 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) $R_{nm}(x)^{(k)}(0) = f^{(k)}(0)$, $k = 0, 1, \ldots, N$; 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

$$(p(x)q_m(x)-p_n(x))^{(k)}\Big|_{x=0}=k!\sum_{j=0}^{k}c_jb_{k-j}-k!a_k,$$

所以可得

$$\begin{cases} a_k=\sum_{j=0}^{k-1}c_jb_{k-j}+c_k & k=0,1,2,\dots,n\\ 0=\sum_{j=0}^{k-1}b_{k-j}, & k=n+1,n+2,\dots,N. \end{cases}\tag{7.17}$$

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 7.26 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 (7.17).

✍ $b_1,b_2,\dots,b_m$ 通过解方程得到, $a_0,a_1,\dots,a_n$ 可直接计算.

8 数值积分与数值微分

考虑定积分

$$I = \int_a^b f(x)\,\mathrm{d}x. \tag{8.1}$$

在微积分中, 我们可以使用 Newton-Leibnitz 公式

$$\int_a^b f(x)\,\mathrm{d}x = F(b) - F(a),$$

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 $f(x) = \dfrac{1}{1+x^6}$ 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
$$f(x) = \frac{\sin x}{x}, \quad f(x) = \mathrm{e}^{-x^2}, \quad f(x) = \sqrt{1 + k^2\sin^2 x}.$$
- 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.

8.1 数值积分基本概念

8.1.1 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

$$\int_a^b f(x)\,\mathrm{d}x=\lim_{h\to 0}\sum_{i=0}^{n}h_i f(\xi_i),\quad \xi_i\in[x_i,x_{i+1}],$$

其中 $h_i=x_{i+1}-x_i$, $h=\max\limits_i\{h_i\}$. 当 h 充分小, n 充分大时, 我们就有下面的近似公式

$$\int_a^b f(x)\,\mathrm{d}x\approx\sum_{i=0}^{n}h_i f(\xi_i). \tag{8.2}$$

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 ξ_i 更换为 A_i 和 x_i)

$$\int_a^b f(x)\,\mathrm{d}x\approx\sum_{i=0}^{n}A_i f(x_i)\triangleq I_n(f). \tag{8.3}$$

这里 x_i 称为**求积节点**, 满足 $a\le x_0<x_1<\cdots<x_n\le b$, 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 8.1 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

$$\int_a^b f(x)\,\mathrm{d}x=f(\xi)(b-a).$$

不幸的是, ξ 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**:
$$\int_a^b f(x)\,\mathrm{d}x\approx f(a)(b-a);$$
- 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**:
$$\int_a^b f(x)\,\mathrm{d}x\approx f(b)(b-a);$$
- 如果用中点的函数值来近似 $f(\xi)$, 则可得**中矩形公式**:
$$\int_a^b f(x)\,\mathrm{d}x\approx f\left(\frac{a+b}{2}\right)(b-a).$$

8.1.2 代数精度

根据 Weierstrass 逼近定理 7.1 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 8.1 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 8.2 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.

$$\int_{-1}^{1} f(x)\,\mathrm{d}x \approx A_0 f(-1)+A_1 f(0)+A_2 f(1).$$

✍ 由于求积公式 (8.3) 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 (8.3) 至少具有 n 次代数精度.

例 8.3 (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:

$$\int_{0}^{1} f(x)\,\mathrm{d}x \approx A_0 f(0)+A_1 f(1)+B_0 f'(0).$$

(板书)

引理 8.1 设机械求积公式 (8.3) 具有 $m(\geq 0)$ 次代数精度, 则有

$$A_0+A_1+\cdots A_n=b-a. \tag{8.4}$$

(板书)

✍ 上面的结论是机械求积公式的一个基本性质.

8.1.3 收敛性与稳定性

定义 8.2 设求积公式的余项为 $R[f]$, 若

$$\lim_{h\to 0} R[f]=0,$$

则称求积公式是收敛的, 其中 $h=\max\limits_{0\leq i\leq n-1}\{x_{i+1}-x_i\}$.

✍ 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 8.3 考虑机械求积公式 (8.3). 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 (8.3) 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 8.2 若机械求积公式 (8.3) 中的求积系数 A_i 都是正数, 则求积公式是稳定的.

(板书)

8.2 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

$$\begin{aligned} \int_a^b f(x)\,\mathrm{d}x \approx \int_a^b L_n(x)\,\mathrm{d}x &= \int_a^b \sum_{k=0}^{n} l_k(x) f(x_k)\,\mathrm{d}x \\ &= \sum_{k=0}^{n} \left(\int_a^b l_k(x)\,\mathrm{d}x \right) f(x_k) \triangleq \sum_{k=0}^{n} A_k f(x_k). \end{aligned} \tag{8.5}$$

这就是**插值型求积公式**, 其中 $l_k(x)$ 是 n 次 Lagrange 基函数, $A_k = \int_a^b l_k(x)\,\mathrm{d}x$.

由多项式插值余项公式可知, 插值型求积公式 (8.5) 的余项为

$$R[f] = \int_a^b \big(f(x) - L_n(x)\big)\,\mathrm{d}x = \int_a^b \frac{f^{(n+1)}(\xi_x)}{(n+1)!} \omega_{n+1}(x)\,\mathrm{d}x, \tag{8.6}$$

其中

$$\omega_{n+1}(x) = (x - x_0)(x - x_1) \cdots (x - x_n).$$

✍ 由于 ξ_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

$$|R[f]| = \left| \int_a^b \frac{f^{(n+1)}(\xi_x)}{(n+1)!} \omega_{n+1}(x)\,\mathrm{d}x \right| \leq \frac{M_{n+1}}{(n+1)!} \int_a^b |\omega_{n+1}(x)|\,\mathrm{d}x,$$

其中 $M_{n+1} = \max\limits_{a\leq x\leq b} |f^{(n+1)}(x)|$.

引理 8.3 插值型求积公式 (8.5) 至少具有 n 次代数精度. (板书)

事实上, 我们有下面的性质.

定理 8.4 机械求积公式 (8.3) 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)

✍ 由该定理可知, 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.

8.3 Newton-Cotes 公式

定义 8.4 如果插值型求积公式 (8.5) 中的节点为等距节点, 即

$$x_k = a + kh, \quad h = \frac{b-a}{n}, \quad k = 0, 1, 2, \ldots, n,$$

则该求积公式就称为 **Newton-Cotes 公式**, 记为

$$I_n(f) = (b-a) \sum_{k=0}^{n} C_k^{(n)} f(x_k), \tag{8.7}$$

其中 $C_k^{(n)}$ 称为 **Cotes 系数**, 其值为

$$C_k^{(n)} = \frac{1}{b-a} \int_a^b l_k(x)\,\mathrm{d}x = \frac{h}{b-a} \int_0^n \prod_{i=0, i\neq k}^{n} \frac{t-i}{k-i}\,\mathrm{d}t = \frac{(-1)^{n-k}}{n\,k!(n-k)!} \int_0^n \prod_{i=0, i\neq k} (t-i)\,\mathrm{d}t.$$

Cotes 系数的两个简单性质

(1) $\sum\limits_{k=0}^{n} C_k^{(n)} = 1$; (2) $C_k^{(n)} = C_{n-k}^{(n)}$, $k = 0, 1, 2, \ldots, n$.

8.3.1 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 $C_0^{(1)} = C_1^{(1)} = \dfrac{1}{2}$, 此时的 Newton-Cotes 公式为

$$I_1(f) = \frac{b-a}{2}\,(f(a) + f(b)). \tag{8.8}$$

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

$$I_2(f)=\frac{b-a}{6}\left(f(a)+4f\left(\frac{a+b}{2}\right)+f(b)\right). \tag{8.9}$$

这就是**抛物线公式**或 **Simpson 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

$$I_3(f)=\frac{b-a}{8}\big(f(x_0)+3f(x_1)+3f(x_2)+f(x_3)\big). \tag{8.10}$$

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

$$I_4(f)=\frac{b-a}{90}\big(7f(x_0)+32f(x_1)+12f(x_2)+32f(x_3)+7f(x_4)\big). \tag{8.11}$$

该公式称为 **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 8.5 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.

(板书)

8.3.2 余项公式的推导

首先给出几个引理.

引理 8.6 设基于节点 $x_0, x_1, x_2, \ldots, x_n$ 的插值型求积公式 (8.5) 具有 m $(m \geq n)$ 次代数精度, 则对任意 $f(x) \in C[a,b]$, 有

$$\sum_{k=0}^{n} A_k f(x_k) = \int_a^b p_m(x)\,\mathrm{d}x,$$

其中 $p_m(x)$ 是 $f(x)$ 关于节点 $x_0, x_1, x_2, \ldots, x_n$ 的 m 次插值多项式, 即 $p_m(x)$ 满足

$$p_m(x_k)=f(x_k), \quad k=0,1,2,\ldots,n.$$

(注: 当 $m>n$ 时, 其它 $m-n$ 个插值条件可以任取, 比如要求在某些节点的导数值相等). (板书)

上述结论也可以推广到包含导数的求积公式.

引理 8.7 设基于节点 $x_0, x_1, x_2, \dots, x_n$ 的求积公式为

$$I_n(f) = \sum_{k=0}^{n} A_k f(x_k) + \sum_{j\in\mathbb{Z}_1} B_j f'(x_j). \tag{8.12}$$

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

$$I_n(f) = \int_a^b p_m(x)\,\mathrm{d}x,$$

其中 $p_m(x)$ 是 $f(x)$ 关于节点 $x_0, x_1, x_2, \dots, x_n$ 的 m 次 Hermitian 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 (8.18), 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 8.8 设 $a = x_0 < x_1 < x_2 < \cdots < x_{n-1} < x_n = b$, 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. □

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 8.9 设 $a = x_0 \leq x_1 \leq x_2 \leq \cdots \leq x_{n-1} \leq x_n = b$, 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 8.10 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

$$\int_a^b f(x)\,\mathrm{d}x=\frac{b-a}{2}\Big(f(a)+f(b)\Big)-\frac{(b-a)^3}{12}f''(\eta),\quad \eta\in(a,b). \tag{8.13}$$

(板书)

✍ 如果使用 Lagrange 插值余项公式, 则可得

$$R[f]=\int_a^b \frac{1}{2!}f''(\xi_x)(x-a)(x-b)\,\mathrm{d}x,\quad \xi_x\in(a,b).$$

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

$$R[f]=\frac{1}{2!}f''(\eta)\int_a^b (x-a)(x-b)\,\mathrm{d}x=-\frac{(b-a)^3}{12}f''(\eta).$$

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 8.11 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

$$\int_a^b f(x)\,\mathrm{d}x=\frac{b-a}{6}\left(f(a)+4f\left(\frac{a+b}{2}\right)+f(b)\right)-\frac{(b-a)^5}{2880}f^{(4)}(\eta),\quad \eta\in(a,b). \tag{8.14}$$

(板书)

✍ 我们也可以将 $H_3(x)$ 写成 Newton 插值形式, 即

$$H_3(x) = f(x_0) + f[x_0, x_1](x - x_0) + f[x_0, x_1, x_1](x - x_0)(x - x_1) + f[x_0, x_1, x_1, x_2](x - x_0)(x - x_1)^2.$$

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.

8.3.3 Newton-Cotes 公式余项的一般形式

定理 8.12 *当 n 是奇数时, 若 $f(x) \in C^{n+1}[a, b]$, 则 Newton-Cotes 公式的余项为*

$$R[f] = \frac{h^{n+2} f^{(n+1)}(\eta)}{(n+1)!} \int_0^n t(t-1)(t-2)\cdots(t-n)\,\mathrm{d}t, \quad \eta \in (a, b).$$

当 n 是偶数时, 若 $f(x) \in C^{n+2}[a, b]$, 则 Newton-Cotes 公式的余项为

$$R[f] = \frac{h^{n+3} f^{(n+2)}(\eta)}{(n+2)!} \int_0^n t^2(t-1)(t-2)\cdots(t-n)\,\mathrm{d}t, \quad \eta \in (a, b).$$

证明. 可参见: J. Stoer and R. Bulirsch, Introduction to Numerical Analysis, 3rd Edition, 2002. □

8.3.4 一般求积公式余项

考虑更一般的求积公式 (带导数信息)

$$\int_a^b f(x)\,\mathrm{d}x = \sum_{k=0}^{n} A_k f(x_k) + \sum_{i_1 \in \mathbb{Z}_1} A_{i_1} f'(x_{i_1}) + \sum_{i_2 \in \mathbb{Z}_2} A_{i_2} f''(x_{i_1}) + \cdots + \sum_{i_r \in \mathbb{Z}_r} A_{i_r} f^{(r)}(x_{i_r}). \tag{8.15}$$

其中 $\mathbb{Z}_j \subset \{0, 1, 2, \ldots, n\}$, 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 (8.15) 的代数精度为 m, 若 $f(x) \in C^{m+1}[a, b]$, 则其余项可表示为

$$R[f] = \int_a^b f^{(m+1)}(t) K(t)\,\mathrm{d}t,$$

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

$$R[f] = f^{(m+1)}(\eta) \int_a^b K(t)\,\mathrm{d}t. \tag{8.16}$$

需要注意的是, 余项公式 (8.16) 并不是对所有求积公式 (8.15) 都成立.

例 8.4 给出下面求积公式的余项

$$\int_0^1 f(x)\,\mathrm{d}x \approx \frac{2}{3}f(0)+\frac{1}{3}f(1)+\frac{1}{6}f'(0).$$

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. □

8.4 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.

8.4.1 复合梯形公式

将 $[a,b]$ 划分为 n 等份, 即取节点

$$x_k=a+kh,\quad h=\frac{b-a}{n},\quad k=0,1,2,\ldots,n.$$

在每个小区间 $[x_k,x_{k+1}]$ 上采用梯形公式, 可得

$$\int_{x_k}^{x_{k+1}} f(x)\,\mathrm{d}x \approx \frac{h}{2}(f(x_k)+f(x_{k+1})).$$

所以

$$\begin{aligned}\int_a^b f(x)\,\mathrm{d}x &= \sum_{k=0}^{n-1}\int_{x_k}^{x_{k+1}} f(x)\,\mathrm{d}x \approx \sum_{k=0}^{n-1}\frac{h}{2}(f(x_k)+f(x_{k+1}))\\ &= h\left(\frac{f(a)+f(b)}{2}+\sum_{k=1}^{n-1}f(x_k)\right).\end{aligned}$$

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 $-\dfrac{h^3}{12}f''(\eta_k)$, 所以整体余项为

$$R_n[f]=\int_a^b f(x)\,\mathrm{d}x - T_n = -\frac{h^3}{12}\sum_{k=0}^{n-1}f''(\eta_k).$$

由于 $f''(x)$ 在 $[a,b]$ 上连续, 且

$$\min_{a\le x\le b} f''(x) \le \frac{1}{n}\sum_{k=0}^{n-1} f''(\eta_k) \le \max_{a\le x\le b} f''(x),$$

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 $n\to\infty$ 时, $R_n[f]\to 0$, 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:

$$\int_a^b f(x)\,\mathrm{d}x - h\sum_{k=0}^{n-1} f(a+kh) = \begin{cases} -(b-a), & \text{若 } m\neq 0 \text{ 被 } n \text{ 整除},\\ 0, & \text{其他}, \end{cases}$$

其中 $f(x)=\exp\left(\dfrac{2\pi \mathbf{i}\, mx}{b-a}\right)$, $\mathbf{i}$ 表示虚部单位, $h=\dfrac{b-a}{n}$.

8.4.2 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:

$$\begin{aligned}\int_a^b f(x)\,\mathrm{d}x &\approx \frac{h}{6}\sum_{k=0}^{n-1}\left(f(x_k)+4f(x_{k+\frac12})+f(x_{k+1})\right)\\ &= \frac{h}{6}\left(f(a)+f(b)+2\sum_{k=1}^{n-1}f(x_k)+4\sum_{k=0}^{n-1}f(x_{k+\frac12})\right).\end{aligned}$$

设 $f(x)\in C^4[a,b]$, 则复合 Simpson 公式的余项为

$$R_n[f]=\int_a^b f(x)\,\mathrm{d}x - S_n = -\frac{b-a}{2880}h^4 f^{(4)}(\eta).$$

易知, 复合 Simpson 公式是收敛的, 也是稳定的.

例 8.5 已知 $f(x)=\dfrac{\sin x}{x}$ 的取值如下表, 试分别用复合梯形公式和复合 Simpson 公式计算 $\displaystyle\int_0^1 f(x)\,\mathrm{d}x$ 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)

8.5 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.

8.5.1 带导数的梯形公式

带余项的复合梯形求积公式为

$$\int_a^b f(x)\,\mathrm{d}x = T_n - \frac{h^3}{12}\sum_{k=0}^{n-1} f''(\eta_k),$$

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

$$h\sum_{k=0}^{n-1} f''(\eta_k) \approx \int_a^b f''(x)\,\mathrm{d}x = f'(b) - f'(a).$$

于是, 我们可以得到下面的求积公式

$$\int_a^b f(x)\,\mathrm{d}x \approx T_n - \frac{h^2}{12}\Big(f'(b) - f'(a)\Big).$$

这就是带端点导数值的复合梯形公式.

当 $n = 1$ 时,

$$\int_a^b f(x)\,\mathrm{d}x \approx \frac{b-a}{2}\Big(f(a) + f(b)\Big) - \frac{(b-a)^2}{12}\Big(f'(b) - f'(a)\Big). \tag{8.17}$$

这就是**修正的梯形公式** (Corrected Trapezoidal Rule). 可以验证, 求积公式 (8.17) 的余项为

$$R[f] = \frac{(b-a)^5}{720} f^{(4)}(\eta), \quad \eta \in (a, b).$$

8.5.2 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

$$\int_a^b f(x)\,\mathrm{d}x \approx S_n - \frac{h^4}{2880}\Big(f'''(b) - f'''(a)\Big).$$

当 $n = 1$ 时,

$$\int_a^b f(x)\,\mathrm{d}x \approx \frac{b-a}{6}\left(f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)\right) - \frac{h^4}{2880}\Big(f'''(b) - f'''(a)\Big). \tag{8.18}$$

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.

8.6 Romberg 求积公式

8.6.1 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

$$T_{2n} = \frac{h}{4}\sum_{i=0}^{n-1}(f(x_i)+2f(x_{i+\frac{1}{2}})+f(x_{i+1})) = \frac{1}{2}T_n + \frac{h}{2}\sum_{i=0}^{n-1} f\left(x_i+\frac{1}{2}h\right). \tag{8.19}$$

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 $x_i = a+ih^{(2)}, i=0,1,\ldots,2^2.$

(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 $x_i = a+ih^{(3)}, i=0,1,\ldots,2^3.$

(5) 依此类推, 对于 $k=4,5,6,\ldots$, 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 $x_i = a+ih^{(k)}, i=0,1,\ldots,2^k.$

例 8.6 用梯形法的递推公式计算定积分 $\int_0^1 \frac{\sin x}{x}\,\mathrm{d}x$, 要求计算精度满足 $|T_{2n}-T_n|<\varepsilon=10^{-7}$.

(Quad_Trap_recursion.m)

8.6.2 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

$$I(f) \triangleq \int_a^b f(x)\,\mathrm{d}x = T(h) - \frac{b-a}{12}h^2 f''(\eta).$$

定理 8.13 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 α_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 8.13 可知 $T(h)-I(f)=\mathcal{O}(h^2)$, 即复合梯形公式的误差阶为 $\mathcal{O}(h^2)$. 由于系数 α_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

$$S(h) \triangleq \frac{4T\left(\frac{h}{2}\right) - T(h)}{3} = I(f) + \beta_1 h^4 + \beta_2 h^6 + \cdots. \tag{8.20}$$

如果用 $S(h)$ 来近似 $I(f)$, 则误差阶提高到了 $\mathcal{O}(h^4)$. 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

$$C(h) \triangleq \frac{16S\left(\frac{h}{2}\right) - S(h)}{15} = I(f) + \gamma_1 h^6 + \gamma_2 h^8 + \cdots. \tag{8.21}$$

这时, 误差阶提高到了 $\mathcal{O}(h^6)$. 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 8.1. 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 (8.19) 计算 $T_0^{(k)} = T\left(\frac{h}{2^k}\right)$

5: **for** $i = 1, 2, \ldots, k$ **do**

6: 计算 $T_i^{(k-i)} = \dfrac{4^i T_{i-1}^{(k-i+1)} - T_{i-1}^{(k-i)}}{4^i - 1}$

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	$h^{(k)}$	$T_0^{(k)}$	$T_1^{(k)}$	$T_2^{(k)}$	$T_3^{(k)}$	$T_4^{(k)}$	$\cdots$
0	$b-a$	$T_0^{(0)}$					
1	$\dfrac{b-a}{2}$	$T_0^{(1)}$	$T_1^{(0)}$				
2	$\dfrac{b-a}{2^2}$	$T_0^{(2)}$	$T_1^{(1)}$	$T_2^{(0)}$			
3	$\dfrac{b-a}{2^3}$	$T_0^{(3)}$	$T_1^{(2)}$	$T_2^{(1)}$	$T_3^{(0)}$		
4	$\dfrac{b-a}{2^4}$	$T_0^{(4)}$	$T_1^{(3)}$	$T_2^{(2)}$	$T_3^{(1)}$	$T_4^{(0)}$	
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

例 8.7 用 Romberg 算法计算定积分 $\displaystyle\int_0^1 \frac{\sin x}{x}\,\mathrm{d}x$, 要求计算精度满足 $|T_{2n} - T_n| < \varepsilon = 10^{-7}$.

(Quad_Romberg.m)

8.7 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 8.8 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.

$$\int_{-1}^{1} f(x)\,\mathrm{d}x \approx A_0 f(x_0) + A_1 f(x_1). \tag{8.22}$$

(板书)

由此可见, 采用不等距节点的两点求积公式 (8.22) 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.

8.7.1 一般 Gauss 求积公式

定义 8.5 设 $\rho(x)$ 是 $[a,b]$ 上的权函数, 若求积公式

$$\int_a^b \rho(x) f(x)\,\mathrm{d}x \approx \sum_{i=0}^{n} A_i f(x_i), \tag{8.23}$$

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 (8.23) 的右端只包含 $f(x)$ 的函数值, 与权函数无关.

求积公式 (8.23) 中含有 $2n+2$ 的待定参数, 即 A_i 和 x_i, $i=0,1,2,\ldots,n$. 我们可以将 $f(x)=1,x,x^2,\ldots,x^{2n+1}$ 代入, 并令求积公式 (8.23) 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 (8.23) 的代数精度不可能超过 $2n+1$. 取 $2n+2$ 次多项式 $f(x)=(x-x_0)^2(x-x_1)^2\cdots(x-x_n)^2$, 则 $\sum\limits_{i=0}^{n} A_i f(x_i)=0$, 但显然

$$\int_a^b \rho(x) f(x)\,\mathrm{d}x > 0,$$

即求积公式 (8.23) 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 8.14 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 8.8 可以看出, 我们可以将 $f(x)=1,x,x^2,\ldots,x^{2n+1}$ 代入, 并令求积公式 (8.23) 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 8.15 设节点 $a \leq x_0 < x_1 < \cdots < x_n \leq b$, 则插值型求积公式 (8.23) 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

$$\int_a^b \rho(x)\omega_{n+1}(x)p(x)\,\mathrm{d}x = 0, \quad \forall\, p(x) \in H_n.$$

(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 $p(x) = 1, x, x^2, \ldots, x^n$ 正交 (带权) 的性质, 得到 $n+1$ 个线性方程, 解出 a_k, $k = 0, 1, 2, \ldots, n$, 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 8.9 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

$$\int_0^1 \sqrt{x} f(x)\,\mathrm{d}x \approx A_0 f(x_0) + A_1 f(x_1).$$

(板书)

Gauss 求积公式的余项

设 $p_{2n+1}(x)$ 是 $f(x)$ 关于点 $x_0, x_1, x_2, \ldots, x_n$ 的 $2n+1$ 次 Hermitian 插值多项式, 满足

$$p_{2n+1}(x_i) = f(x_i), \quad p'_{2n+1}(x_i) = f'(x_i), \quad i = 0, 1, 2, \ldots, n.$$

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 (8.23) 具有 $2n+1$ 次代数精度, 故

$$\int_a^b \rho(x)p_{2n+1}(x)\,\mathrm{d}x = \sum_{i=0}^n A_i p_{2n+1}(x_i).$$

所以, Gauss 求积公式的余项为

$$\begin{aligned} R_n[f] &\triangleq \int_a^b \rho(x)f(x)\,\mathrm{d}x - \sum_{i=0}^n A_i f(x_i) \\ &= \int_a^b \rho(x)f(x)\,\mathrm{d}x - \sum_{i=0}^n A_i p_{2n+1}(x_i) \end{aligned}$$

$$= \int_a^b \rho(x)\Big(f(x) - p_{2n+1}(x)\Big)\,\mathrm{d}x$$
$$= \int_a^b \rho(x)\frac{f^{(2n+2)}(\xi_x)}{(2n+2)!}\omega_{n+1}^2\,\mathrm{d}x.$$

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

$$R_n[f] = \frac{f^{(2n+2)}(\eta)}{(2n+2)!}\int_a^b \rho(x)\omega_{n+1}^2\,\mathrm{d}x. \tag{8.24}$$

Gauss 公式的收敛性与稳定性

定理 8.16 设 $f(x) \in C[a,b]$, 则 Gauss 求积公式是收敛的, 即

$$\lim_{n\to\infty}\sum_{i=0}^{n} A_i f(x_i) = \int_a^b \rho(x)f(x)\,\mathrm{d}x.$$

(证明可参见相关参考文献)

定理 8.17 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. (板书)

8.7.2 Gauss-Legendre 公式

根据定理 8.15, 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

$$\int_{-1}^{1} f(x)\,\mathrm{d}x \approx 2f(0).$$

- 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

$$\int_{-1}^{1} f(x)\,\mathrm{d}x \approx f\left(-\frac{\sqrt{3}}{3}\right) + f\left(\frac{\sqrt{3}}{3}\right) \approx f(-0.5774) + f(0.5774).$$

- 当 $n=2$ 时, 类似地, 可以得到 G-L 公式为

$$\begin{aligned}\int_{-1}^{1} f(x)\,\mathrm{d}x &\approx \frac{5}{9}f\left(-\frac{\sqrt{15}}{5}\right)+\frac{8}{9}f(0)+\frac{5}{9}f\left(\frac{\sqrt{15}}{5}\right)\\ &\approx 0.5556f(-0.7746)+0.8889f(0)+0.5556f(0.7746).\end{aligned}$$

- 当 $n=3$ 时, 可得 G-L 求积公式为

$$\begin{aligned}\int_{-1}^{1} f(x)\,\mathrm{d}x \approx{}& \frac{90-5\sqrt{30}}{180}f\left(-\sqrt{\frac{15+2\sqrt{30}}{35}}\right)+\frac{90+5\sqrt{30}}{180}f\left(-\sqrt{\frac{15-2\sqrt{30}}{35}}\right)\\ &+\frac{90+5\sqrt{30}}{180}f\left(\sqrt{\frac{15-2\sqrt{30}}{35}}\right)+\frac{90-5\sqrt{30}}{180}f\left(\sqrt{\frac{15+2\sqrt{30}}{35}}\right)\end{aligned}$$

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 8.10 用三点 G-L 公式 ($n=2$) 计算定积分 $\displaystyle\int_0^{\frac{\pi}{2}} x^2\cos x\,\mathrm{d}x$.

(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 (8.24) 和 Legendre 多项式性质 (7.3) 可知, G-L 公式的余项为

$$R_n[f]=\frac{f^{(2n+2)}(\eta)}{(2n+2)!}\int_{-1}^{1}\tilde{P}_{n+1}^2\,\mathrm{d}x=\frac{2^{2n+3}[(n+1)!]^4}{(2n+3)[(2n+2)!]^3}f^{(2n+2)}(\eta),\quad \eta\in(-1,1).$$

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

$$\int_a^b f(x)\,\mathrm{d}x=\frac{b-a}{2}\int_{-1}^{1}f(x(t))\,\mathrm{d}t=\frac{b-a}{2}\int_{-1}^{1}f\left(\frac{b-a}{2}t+\frac{b+a}{2}\right)\,\mathrm{d}t,$$

然后对上式右端的定积分使用 G-L 公式即可.

8.7.3 Gauss-Chebyshev 公式

设 $[a,b]=[-1,1]$, 权函数 $\rho(x)=\dfrac{1}{\sqrt{1-x^2}}$, 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

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

利用待定系数法可以求得

$$A_i=\frac{\pi}{n+1}.$$

所以 G-C 公式为

$$\int_a^b\frac{1}{\sqrt{1-x^2}}f(x)\,\mathrm{d}x\approx\frac{\pi}{n+1}\sum_{i=0}^{n}f\left(\cos\frac{2i+1}{2(n+1)}\pi\right). \tag{8.25}$$

由公式 (8.24) 可知, G-C 公式的余项为

$$R[f]=\frac{f^{(2n+2)}(\eta)}{(2n+2)!}\int_a^b\frac{1}{\sqrt{1-x^2}}\tilde{T}_{n+1}^2\,\mathrm{d}x=\frac{2\pi}{2^{2n+2}(2n+2)!}f^{(2n+2)}(\eta),\quad \eta\in(-1,1).$$

例 8.11 用五点 G-C 公式 ($n=4$) 计算奇异积分 $\displaystyle\int_{-1}^{1}\frac{\mathrm{e}^x}{\sqrt{1-x^2}}\,\mathrm{d}x$.

(板书)

8.7.4 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**.

8.8 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 Ω 为矩形区域, 则

$$\iint_\Omega f(x,y)\,\mathrm{d}x\,\mathrm{d}y = \int_a^b \left(\int_c^d f(x,y)\,\mathrm{d}y\right)\,\mathrm{d}x.$$

如果积分区域 Ω 是 x 型区域, 则

$$\iint_\Omega f(x,y)\,\mathrm{d}x\,\mathrm{d}y = \int_a^b \left(\int_{y_1(x)}^{y_2(x)} f(x,y)\,\mathrm{d}y\right)\,\mathrm{d}x.$$

如果积分区域 Ω 是 y 型区域, 则

$$\iint_\Omega f(x,y)\,\mathrm{d}x\,\mathrm{d}y = \int_c^d \left(\int_{x_1(y)}^{x_2(y)} f(x,y)\,\mathrm{d}x\right)\,\mathrm{d}y.$$

例 8.12 利用两点 Gauss 公式计算二重积分 $\int_\Omega (x^2+2y^2)\,\mathrm{d}x$, 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.

8.9 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.

8.9.1 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

$$\begin{aligned} f'(x) - p_n'(x) &= \frac{\mathrm{d}}{\mathrm{d}x}\left(\frac{f^{(n+1)}(\xi_x)}{(n+1)!}\omega_{n+1}(x)\right) \\ &= \frac{f^{(n+1)}(\xi_x)}{(n+1)!}\omega_{n+1}'(x) + \omega_{n+1}(x)\frac{\mathrm{d}}{\mathrm{d}x}\left(\frac{f^{(n+1)}(\xi_x)}{(n+1)!}\right). \end{aligned}$$

这里假定 $f^{(n+1)}(\xi_x)$ 关于 x 可导. 由于 ξ_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

$$f'(x_i) - p_n'(x_i) = \frac{f^{(n+1)}(\xi_x)}{(n+1)!}\omega_{n+1}'(x_i) = \frac{f^{(n+1)}(\xi_x)}{(n+1)!}\prod_{k=0,k\neq i}^{n}(x_i - x_k). \tag{8.26}$$

✍ 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.

8.9.2 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

$$\frac{\mathrm{d}p_2}{\mathrm{d}t} = \frac{1}{2}\big((2t-3)f(x_0) - 4(t-1)f(x_1) + (2t-1)f(x_2)\big).$$

所以

$$\frac{\mathrm{d}p_2}{\mathrm{d}x} = \frac{\mathrm{d}p_2}{\mathrm{d}t}\Big/\frac{\mathrm{d}x}{\mathrm{d}t} = \frac{1}{2h}\big((2t-3)f(x_0) - 4(t-1)f(x_1) + (2t-1)f(x_2)\big).$$

分别令 $x = x_0, x_1, x_2$, 即 $t = 0, 1, 2$, 可得

$$\begin{aligned} p_2'(x_0) &= \frac{1}{2h}\big(-3f(x_0) + 4f(x_1) - f(x_2)\big), \\ p_2'(x_1) &= \frac{1}{2h}\big(f(x_2) - f(x_0)\big), \\ p_2'(x_2) &= \frac{1}{2h}\big(f(x_0) - 4f(x_1) + 3f(x_2)\big). \end{aligned}$$

由公式 (8.26) 可知

$$\begin{aligned} f'(x_0) &= \frac{1}{2h}\big(-3f(x_0) + 4f(x_1) - f(x_2)\big) + \frac{h^2}{3}f^{(3)}(\xi_0), \\ f'(x_1) &= \frac{1}{2h}\big(f(x_2) - f(x_0)\big) - \frac{h^2}{6}f^{(3)}(\xi_1), \\ f'(x_2) &= \frac{1}{2h}\big(f(x_0) - 4f(x_1) + 3f(x_2)\big) + \frac{h^2}{3}f^{(3)}(\xi_2). \end{aligned} \tag{8.27}$$

这就是三点求导公式, 特别公式 (8.27), 是常用的计算一阶导数的 **中心差分公式**.

8.9.3 二阶导数的差分近似

计算 $p_2(x)$ 关于 x 的二阶导数可得

$$\frac{\mathrm{d}^2p_2}{\mathrm{d}x^2} = \frac{1}{h^2}\big(f(x_0) - 2f(x_1) + f(x_2)\big).$$

结合公式 (8.26) 可知

$$f''(x_1) = \frac{1}{h^2}\big(f(x_0) - 2f(x_1) + f(x_2)\big) - \frac{h^2}{12}f^{(4)}(\xi).$$

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.

8.9.4 三次样条求导

见课堂板书 (*To be continued ...*)

8.9.5 数值微分的外推算法

见课堂板书 (*To be continued ...*)

9 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 $\lambda_1, \lambda_2, \ldots, \lambda_n$. 本讲中我们总是假定

$$|\lambda_1| \geq |\lambda_2| \geq \cdots \geq |\lambda_n| \geq 0,$$

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 [62]
- B. N. Parlett, *The Symmetric Eigenvalue Problem*, 2nd Eds., 1998 [41]
- G. W. Stewart, *Matrix Algorithms, Vol II: Eigensystems*, 2001 [52]
- 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 [48]
- 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.

9.1 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.

9.1.1 幂迭代法

幂迭代法 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 9.1. 幂迭代法 (Power Iteration)

1: Choose an initial guess $x^{(0)}$ with $\|x^{(0)}\|_2 = 1$
2: set $k = 0$
3: **while** not convergence **do**
4: $\quad y^{(k+1)} = Ax^{(k)}$
5: $\quad x^{(k+1)} = y^{(k+1)}/\|y^{(k+1)}\|_2$ % 单位化, 防止溢出
6: $\quad \mu_{k+1} = \left(x^{(k+1)}, Ax^{(k+1)}\right)$ % 计算内积
7: $\quad k = k + 1$
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 $\Lambda = \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n) \in \mathbb{C}^{n\times n}$, $V = [v_1, v_2, \ldots, v_n] \in \mathbb{C}^{n\times n}$, 且 $\|v_i\|_2 = 1, i = 1, 2, \ldots, n$.

(2) 同时, 我们还假设 $|\lambda_1| > |\lambda_2| \geq |\lambda_3| \geq \cdots \geq |\lambda_n|$.

由于 $V \in \mathbb{C}^{n\times n}$ 非奇异, 所以它的列向量组构成 $\mathbb{C}^n$ 的一组基. 因此迭代初始向量 $x^{(0)}$ 可表示为

$$x^{(0)} = \alpha_1 v_1 + \alpha_2 v_2 + \cdots + \alpha_n v_n = V[\alpha_1, \alpha_2, \ldots, \alpha_n]^{\mathsf{T}}.$$

我们假定 $\alpha_1 \neq 0$, 即 $x^{(0)}$ 不属于 $\mathrm{span}\{v_2, v_3, \ldots, v_n\}$ (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 $|\lambda_i/\lambda_1| < 1, i = 2, 3, \ldots, n$, 所以

$$\lim_{k\to\infty} \left(\frac{\lambda_i}{\lambda_1}\right)^k = 0, \quad i = 2, 3, \ldots, n.$$

故当 k 趋向于无穷大时, 向量

$$\left[1, \frac{\alpha_2}{\alpha_1}\left(\frac{\lambda_2}{\lambda_1}\right)^k, \ldots, \frac{\alpha_n}{\alpha_1}\left(\frac{\lambda_n}{\lambda_1}\right)^k\right]^{\mathsf{T}}, \quad k = 0, 1, 2, \ldots$$

收敛到 $e_1 = [1, 0, \ldots, 0]^{\mathsf{T}}$. 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 λ_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 λ_2, 则可以在模最大特征值 λ_1 计算出来后, 采用 **收缩 (Deflation)** 技术: 构造酉矩阵 U, 使得

$$U^*AU = \begin{bmatrix} \lambda_1 & A_{12} \\ 0 & A_{22} \end{bmatrix}.$$

然后将幂迭代作用到 A_{22} 上, 就可以求出 λ_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).

位移策略

前面已经提到, 幂迭代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 σ 是一个给定的数, 称 σ 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 σ 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) $\max\limits_{2\le i\le n}\left|\dfrac{\lambda_i-\sigma}{\lambda_1-\sigma}\right|$ 尽可能地小.

其中第一个条件保证最后所求得的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, σ 的取值并不是一件容易的事.

例 9.1 设 $A = X\Lambda X^{-1}$, 其中 Λ 为对角矩阵, 分别用幂迭代法和带位移的幂迭代法计算 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, 主要是用在反迭代法和 QR 迭代法中.

9.1.2 反迭代法

如果我们将幂迭代法作用在 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 9.2. *带位移的反迭代法* (Inverse Iteration)

1: Choose a scalar σ and an initial vector $x^{(0)}$ with $\|x^{(0)}\|_2 = 1$
2: set $k = 0$
3: **while** not convergence **do**
4: $\quad y^{(k+1)} = (A-\sigma I)^{-1}x^{(k)}$

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**反迭代法**. 显然, 在反迭代法中, μ_k 收敛到距离 σ 最近的特征值, 而 $x^{(k)}$ 则收敛到其对应的特征向量.

设距离 σ 最近的特征值为 λ_k, 则算法的收敛速度取决于

$$\max_{1\le i\le n}\left|\frac{\lambda_k-\sigma}{\lambda_i-\sigma}\right|$$

的大小. 显然, σ 约接近于 λ_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.

反迭代法的另一个优点是, 只要选取合适的位移 σ, 就可以计算 A 的任意一个特征值.

但反迭代法的缺点也很明显: 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, 反迭代法一次只能求一个特征值.

Rayleigh 商迭代

在反迭代法中, 有一个很重要的问题, 就是位移 σ 的选取.

显然, 选取 σ 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 σ 可以不一样, 即在迭代过程中可以选取不同的位移 σ.

由于在反迭代法中, μ_k 是收敛到所求的特征值的, 所以我们可以选取 μ_k 作为第 k 步的位移. 这时所得到的反迭代法就称为 **Rayleigh 商迭代** (Rayleigh Quotient Iteration), 简记为 RQI.

算法 9.3. Rayleigh 商迭代法 (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: $\mu_{k+1} = (x^{(k+1)}, Ax^{(k+1)})$ % Rayleigh Quotient
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 9.2 设 $A = X\Lambda X^{-1}$, 其中 Λ 为对角矩阵, 用 Rayleigh 商迭代计算 A 的特征值.

(Eig_Rayleigh.m)

9.1.3 QR 迭代法

QR 迭代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 9.4. QR 迭代法 (QR Iteration)

1: Set $A_1 = A$ and $k = 1$
2: **while** not convergence **do**
3: $[Q_k, R_k] = \mathtt{QR}(A_k)$ % 计算 A_k 的 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

$$A_{k+1} = R_k Q_k = (Q_k^{\mathsf{T}} Q_k) R_k Q_k = Q_k^{\mathsf{T}} (Q_k R_k) Q_k = Q_k^{\mathsf{T}} A_k Q_k.$$

由这个递推关系可得

$$A_{k+1} = Q_k^{\mathsf{T}} A_k Q_k = Q_k^{\mathsf{T}} Q_{k-1}^{\mathsf{T}} A_{k-1} Q_{k-1} Q_k = \cdots = Q_k^{\mathsf{T}} Q_{k-1}^{\mathsf{T}} \cdots Q_1^{\mathsf{T}} A Q_1 \cdots Q_{k-1} Q_k.$$

记 $\tilde{Q}_k = Q_1 \cdots Q_{k-1} Q_k$, 则

$$A_{k+1} = \tilde{Q}_k^{\mathsf{T}} A \tilde{Q}_k, \tag{9.1}$$

即 A_{k+1} 与 A 正交相似.

定理 9.1 (收敛性) 设 $A = V\Lambda V^{-1} \in \mathbb{R}^{n\times n}$, 其中 $\Lambda = \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n)$, 且 $|\lambda_1| > |\lambda_2| > \cdots > |\lambda_n|$. 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.

例 9.3 QR 迭代法演示. 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

$$a_{ij}^{(k)} = 0 \quad \text{if} \quad i > j \text{ and } |a_{ij}^{(k)}| < \mathrm{tol}\,.$$

这里我们取 $\mathrm{tol} = 10^{-6} \max\limits_{1\le i,j\le n}\{|a_{ij}^{(k)}|\}$. (Eig_QR.m)

注记 (详情可参见相关资料)

- 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**.
- QR 迭代法中需要考虑的另一个重要问题就是运算量, 实际应用时需要先将 A 转化为上 Hessenberg 矩阵, 然后构造**带位移隐式 QR 迭代法**.
- **带位移隐式 QR 迭代法**是当前计算中小规模矩阵所有特征值的首选方法.

9.2 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. 目前常用算法有: Jacobi 迭代法, Rayleigh 商迭代法, 对称 QR 迭代法, 分而治之法, 对分法等. 这里主要介绍分而治之法, 这是当前计算对称三对角矩阵所有特征值和特征向量的首选方法.

9.2.1 Hessenberg 矩阵

为了减少运算量, 我们首先需要先对 A 进行简化, 即通过正交相似变换将其转化为**上 Hessenberg 矩阵** (非对称矩阵情形) 或 **对称三对角矩阵** (对称矩阵情形).

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 9.2 设 $A \in \mathbb{R}^{n\times n}$, 则存在正交矩阵 $Q \in \mathbb{R}^{n\times n}$, 使得 QAQ^{T} 是上 Hessenberg 矩阵.

下面我们以一个 5×5 的矩阵 A 为例, 给出基于 Householder 变换的**上 Hessenberg 化**过程.

第一步: 令 $Q_1 = \mathrm{diag}(I_{1\times 1}, H_1)$, 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

$$A_1 \triangleq Q_1 A Q_1^{\mathsf{T}} = \begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ 0 & * & * & * & * \\ 0 & * & * & * & * \\ 0 & * & * & * & * \end{bmatrix}.$$

第二步: 令 $Q_2 = \mathrm{diag}(I_{2\times 2}, H_2)$, 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,

$$Q_2 A_1 = \begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ 0 & * & * & * & * \\ 0 & 0 & * & * & * \\ 0 & 0 & * & * & * \end{bmatrix} \quad \text{和} \quad A_2 \triangleq Q_2 A_1 Q_2^{\mathsf{T}} = \begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ 0 & * & * & * & * \\ 0 & 0 & * & * & * \\ 0 & 0 & * & * & * \end{bmatrix}.$$

第三步: 令 $Q_3 = \mathrm{diag}(I_{3\times 3}, H_3)$, 其中 H_3 是对应于向量 $A_2(4:5,3)$ 的 Householder 矩阵, 则有

$$Q_3 A_2 = \begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ 0 & * & * & * & * \\ 0 & 0 & * & * & * \\ 0 & 0 & 0 & * & * \end{bmatrix} \quad \text{和} \quad A_3 \triangleq Q_3 A_2 Q_3^{\mathsf{T}} = \begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ 0 & * & * & * & * \\ 0 & 0 & * & * & * \\ 0 & 0 & 0 & * & * \end{bmatrix}.$$

这时, 我们就将 A 转化成一个上 Hessenberg 矩阵, 即 $QAQ^{\mathsf{T}} = A_3$ 其中 $Q = Q_3Q_2Q_1$ 是正交矩阵, A_3 是上 Hessenberg 矩阵.

✍ 在实际计算时, 我们不需要显式地形成 Householder 矩阵 H_k.

推论 9.3 设 $A \in \mathbb{R}^{n\times n}$ 对称, 则存在正交矩阵 $Q \in \mathbb{R}^{n\times n}$, 使得 QAQ^{T} 是对称三对角矩阵.

9.2.2 分而治之法

分而治之法 (Divide-and-Conquer) 是由 Cuppen [9] 于 1981 年首次提出. 该算法是目前计算大规模对称三对角矩阵的特征值和特征向量的首选方法.

考虑不可约对称三对角矩阵

$$
\begin{aligned}
T &= \left[\begin{array}{ccccc|ccccc}
a_1 & b_1 & & & & & & & & \\
b_1 & \ddots & \ddots & & & & & & & \\
 & \ddots & a_{m-1} & b_{m-1} & & & & & & \\
 & & b_{m-1} & a_m & b_m & & & & & \\
\hline
 & & & b_m & a_{m+1} & b_{m+1} & & & & \\
 & & & & b_{m+1} & \ddots & \ddots & & & \\
 & & & & & \ddots & \ddots & b_{n-1} & & \\
 & & & & & & b_{n-1} & a_n &
\end{array}\right] \\
&= \left[\begin{array}{ccccc|ccccc}
a_1 & b_1 & & & & & & & & \\
b_1 & \ddots & \ddots & & & & & & & \\
 & \ddots & a_{m-1} & b_{m-1} & & & & & & \\
 & & b_{m-1} & a_m - b_m & & & & & & \\
\hline
 & & & & a_{m+1} - b_m & b_{m+1} & & & & \\
 & & & & b_{m+1} & \ddots & \ddots & & & \\
 & & & & & \ddots & \ddots & b_{n-1} & & \\
 & & & & & & b_{n-1} & a_n &
\end{array}\right] + \left[\begin{array}{ccccc|ccccc}
 & & & & & & & & & \\
 & & & & & & & & & \\
 & & & & & & & & & \\
 & & & & & b_m & b_m & & & \\
\hline
 & & & & & b_m & b_m & & & \\
 & & & & & & & & & \\
 & & & & & & & & & \\
 & & & & & & & & &
\end{array}\right] \\
&= \left[\begin{array}{c|c} T_1 & 0 \\ \hline 0 & T_2 \end{array}\right] + b_m v v^{\mathsf{T}},
\end{aligned}
$$

其中 $v = [0, \ldots, 0, 1, 1, 0, \ldots, 0]^{\mathsf{T}}$. 假定 T_1 和 T_2 的特征值分解已经计算出来了, 即 $T_1 = Q_1\Lambda_1Q_1^{\mathsf{T}}$, $T_2 = Q_2\Lambda_2Q_2^{\mathsf{T}}$, 下面考虑 T 的特征值分解.

首先介绍一个引理.

引理 9.4 设 $x, y \in \mathbb{R}^n$, 则 $\det(I + xy^{\mathsf{T}}) = 1 + y^{\mathsf{T}}x$.

(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.

$$\begin{aligned} T &= \begin{bmatrix} T_1 & 0 \\ 0 & T_2 \end{bmatrix} + b_m vv^{\mathsf{T}} \\ &= \begin{bmatrix} Q_1\Lambda_1 Q_1^{\mathsf{T}} & 0 \\ 0 & Q_2\Lambda_2 Q_2^{\mathsf{T}} \end{bmatrix} + b_m vv^{\mathsf{T}} \\ &= \begin{bmatrix} Q_1 & 0 \\ 0 & Q_2 \end{bmatrix} \left(\begin{bmatrix} \Lambda_1 & 0 \\ 0 & \Lambda_2 \end{bmatrix} + b_m uu^{\mathsf{T}} \right) \begin{bmatrix} Q_1 & 0 \\ 0 & Q_2 \end{bmatrix}^{\mathsf{T}}, \end{aligned}$$

其中

$$u = \begin{bmatrix} Q_1 & 0 \\ 0 & Q_2 \end{bmatrix}^{\mathsf{T}} v = \begin{bmatrix} Q_1^{\mathsf{T}} \text{ 的最后一列} \\ Q_2^{\mathsf{T}} \text{ 的第一列} \end{bmatrix}.$$

令 $\alpha = b_m$, $D = \operatorname{diag}(\Lambda_1, \Lambda_2) = \operatorname{diag}(d_1, d_2, \ldots, d_n)$, 并假定 $d_1 \geq d_2 \geq \cdots \geq d_n$. 则 T 的特征值与 $D + \alpha uu^{\mathsf{T}}$ 的特征值相同.

下面计算 $D + \alpha uu^{\mathsf{T}}$ 的特征值. 设 λ 是 $D + \alpha uu^{\mathsf{T}}$ 的一个特征值, 若 $D - \lambda I$ 非奇异, 则

$$\det(D + \alpha uu^{\mathsf{T}} - \lambda I) = \det(D - \lambda I) \cdot \det(I + \alpha(D - \lambda I)^{-1} uu^{\mathsf{T}}).$$

故 $\det(I + \alpha(D - \lambda I)^{-1} uu^{\mathsf{T}}) = 0$. 又由引理 9.4 可知

$$\det(I + \alpha(D - \lambda I)^{-1} uu^{\mathsf{T}}) = 1 + \alpha u^{\mathsf{T}}(D - \lambda I)^{-1} u = 1 + \alpha \sum_{i=1}^{n} \frac{u_i^2}{d_i - \lambda} \triangleq f(\lambda).$$

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).

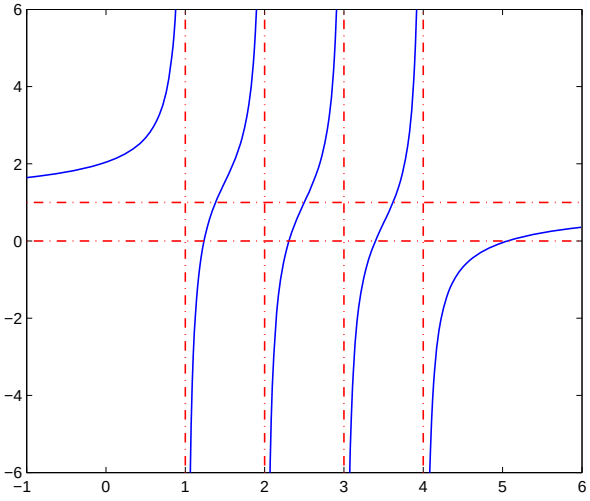


图 9.1. $f(\lambda) = 1 + 0.5\left(\frac{1}{4-\lambda} + \frac{1}{3-\lambda} + \frac{1}{2-\lambda} + \frac{1}{1-\lambda}\right)$ 的图像

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, ∞) (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 $\mathcal{O}(n)$, 计算 $D + \alpha uu^{\mathsf{T}}$ 的所有特征值的运算量约为 $\mathcal{O}(n^2)$.

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 9.5 设 $D \in \mathbb{R}^{n\times n}$ 为对角矩阵, $u \in \mathbb{R}^n$, $\alpha \in \mathbb{R}$, 若 λ 是 $D + \alpha uu^{\mathsf{T}}$ 的特征值, 且 $\lambda \neq d_i$, $i = 1, 2, \ldots, n$, 则 $(D - \lambda I)^{-1}u$ 是其对应的特征向量. (板书)

算法 9.5. 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 $[Q, \Lambda]$ = dc_eig(T) % $T = Q\Lambda Q^{\mathsf{T}}$
2: **if** T is of 1×1 **then**
3: $Q = 1, \Lambda = T$
4: return
5: **end if**
6: form $T = \begin{bmatrix} T_1 & 0 \\ 0 & T_2 \end{bmatrix} + b_m vv^{\mathsf{T}}$
7: $[Q_1, \Lambda_1]$ = dc_eig(T_1)
8: $[Q_2, \Lambda_2]$ = dc_eig(T_2)
9: form $D + \alpha uu^{\mathsf{T}}$ from $\Lambda_1, \Lambda_2, Q_1, Q_2$
10: compute the eigenvalues Λ and eigenvectors $\hat{Q}$ of $D + \alpha uu^{\mathsf{T}}$
11: compute the eigenvectors of T with $Q = \begin{bmatrix} Q_1 & 0 \\ 0 & Q_2 \end{bmatrix} \cdot \hat{Q}$
12: end

注记

在实际使用时, 分而治之法涉及非线性方程的求解和特征向量的稳定计算方法, 以及如果运用收缩现象大幅减少运算量等问题, 因此直到 1995 年才出现稳定高效的实现方式 [25].

9.3 应用

9.3.1 多项式求根

考虑 n 次多项式

$$p_n(x) = a_n x^n + a_{n-1}x^{n-1} + \cdots + a_1 x + a_0. \tag{9.2}$$

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 迭代法.

首先将多项式 (9.2) 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:

$$A = \begin{bmatrix} 0 & & & -c_0 \\ 1 & 0 & & -c_1 \\ & \ddots & \ddots & \vdots \\ & & 1 & -c_{n-1} \end{bmatrix}. \tag{9.3}$$

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 $\mathcal{O}(n^3)$.

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, 57] 提出了快速 QR 方法, 将运算量降为 $\mathcal{O}(n^2)$, 存储量也降为 $\mathcal{O}(n)$. 主要思想是将 A 写成一个酉矩阵与秩一矩阵之差:

$$A = \begin{bmatrix} 0 & & & 1 \\ 1 & 0 & & 0 \\ & \ddots & \ddots & \vdots \\ & & 1 & 0 \end{bmatrix} - \begin{bmatrix} c_0 + 1 \\ c_1 \\ \vdots \\ c_{n-1} \end{bmatrix} [0, 0, \ldots, 1] \triangleq U - xy^{\mathsf{T}}.$$

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 迭代法, 详细过程可参见 [1, 3, 8, 57] 或 [77].

9.3.2 Goolge 网页排名: PageRank

见课堂板书 (*To be continued ...*)

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**

$$\frac{\mathrm{d}y}{\mathrm{d}x} = f(x,y), \quad x \in [a,b], \tag{10.1a}$$

$$y(a) = y_0, \tag{10.1b}$$

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

$$|f(x,y_1) - f(x,y_2)| \le L|y_1 - y_2|. \tag{10.2}$$

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 $D \triangleq \{(x,y)\,|\,x\in[a,b], y\in\mathbb{R}\}$ 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 [27]

[2] E. Hairer and G. Wanner, *olving Ordinary Differential Equations II: Stiff and Differential-Algebraic Problems*, 2nd, 1996 [26]

[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

$$\int_a^b \frac{\mathrm{d}y}{\mathrm{d}x}\,\mathrm{d}x = \int_a^b f(x,y)\,\mathrm{d}x,$$

即

$$y(b) - y(a) = \int_a^b f(x,y)\,\mathrm{d}x.$$

于是可得

$$y(b) = y(a) + \int_a^b f(x,y)\,\mathrm{d}x.$$

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

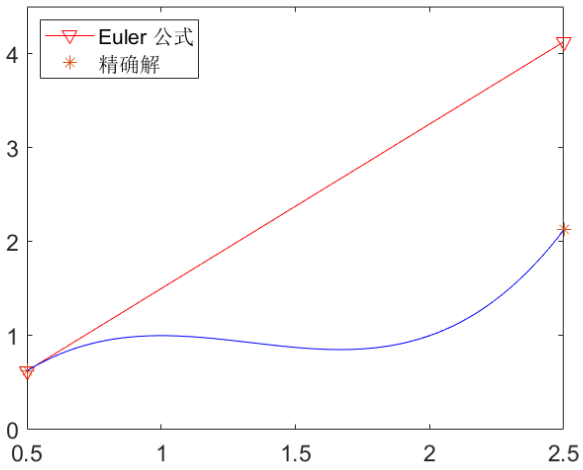
用左点矩形公式近似 $\int_a^b f(x,y)\,\mathrm{d}x$, 则可得

$$y(b) \approx y(a) + (b-a)f(a,y(a)) = y(a) + (b-a)f(a,y_0). \tag{10.3}$$

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:

$$y_{k+1} = y_k + h_k f(x_k, y_k), \quad k = 0, 1, 2, \ldots, n-1 \tag{10.4}$$

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)

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

$$\begin{aligned} y(x_{k+1}) &= y(x_k) + \int_{x_k}^{x_{k+1}} f(x, y)\,\mathrm{d}x \\ &\approx y(x_k) + \frac{x_{k+1} - x_k}{2}\big[f(x_k, y(x_k)) + f(x_{k+1}, y(x_{k+1}))\big], \end{aligned}$$

于是可得**梯形法**:

$$y_{k+1} = y_k + \frac{h_k}{2}\big[f(x_k, y_k) + f(x_{k+1}, y_{k+1})\big] \tag{10.5}$$

✍ 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

$$\begin{cases} \tilde{y}_{k+1} = y_k + (x_{k+1} - x_k) f(x_k, y_k), \\ y_{k+1} = y_k + \dfrac{x_{k+1} - x_k}{2} \left[f(x_k, y_k) + f(x_{k+1}, \tilde{y}_{k+1}) \right], \end{cases} \quad k = 0, 1, 2, \ldots, n-1.$$

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

$$\begin{aligned} y_{k+1} &= y_k + \frac{1}{2} h_k (K_1 + K_2), \quad h_k = x_{k+1} - x_k, \\ K_1 &= f(x_k, y_k), \quad K_2 = f(x_k + h_k, y_k + h_k K_1). \end{aligned} \tag{10.6}$$

例 10.2 用 Euler 法和改进的 Euler 法求解初值问题 (ODE_Euler_02.m)

θ-方法

更一般地, 我们可以构造如下的 θ-方法:

$$y_{k+1} = y_k + h_k\big[\theta f(x_k, y_k) + (1-\theta) f(x_{k+1}, y_{k+1})\big],$$

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 $y_k, y_{k-1}, \ldots, y_{k-r+1}$ 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

$$y_{k+1} = y_k + h\Phi(x_k, y_k, y_{k+1}, h), \tag{10.7}$$

这里的 Φ 称为**增量函数**. 如果 Φ 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

$$y_{k+1} = y_k + h\Phi(x_k, y_k, h), \tag{10.8}$$

定义 10.1 设 $y(x)$ 是初值问题 (10.1) 的精确解, 则称

$$R_{k+1} \triangleq y(x_{k+1}) - y(x_k) - h\Phi(x_k, y(x_k), h)$$

为显示单步法 (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.

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** (consisten).

对于显式单步法 (10.8), 当 $h\to 0$ 时我们有

$$\frac{y_{k+1}-y_k}{h}=\Phi(x_k,y_k,h)\to\Phi(x_k,y_k,0).$$

又

$$\lim_{h\to 0}\frac{y(x_{k+1})-y(x_k)}{h}=y'(x_k)=f(x_k,y_k).$$

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

$$R_{k+1}=\mathcal{O}(h^{p+1}),$$

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

$$\lim_{h\to 0}|y_k-y(x_k)|=0,$$

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 Φ 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

$$y_k-y(x_k)=\mathcal{O}(h^p),$$

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

$$y(x_{k+1}) = y(x_k) + \int_{x_k}^{x_{k+1}} f(x,y)\,\mathrm{d}x, \tag{10.9}$$

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:

$$\int_{x_k}^{x_{k+1}} f(x,y)\,\mathrm{d}x \approx h_k \sum_{i=1}^{r} \alpha_i f\big(x_k+\lambda_i h_k, y(x_k+\lambda_i h_k)\big), \tag{10.10}$$

其中 α_i 为求积系数,

$$h_k = x_{k+1} - x_k, \quad 0 \le \lambda_1 < \lambda_2 < \cdots < \lambda_r \le 1,$$

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 $f\big(x_k+\lambda_2 h_k, y(x_k+\lambda_2 h_k)\big)$, 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

$$y(x_k+\lambda_2 h_k) = y(x_k) + \int_{x_k}^{x_k+\lambda_2 h_k} f(x,y)\,\mathrm{d}x \approx y_k + \mu_{21} h_k f(x_k, y_k) = y_k + \mu_{21} h_k K_1,$$

其中 μ_{21} 为某个常数 (事实上就是 λ_2). 所以

$$f\big(x_k+\lambda_2 h_k, y(x_k+\lambda_2 h_k)\big) \approx f\big(x_k+\lambda_2 h_k, y_k+\mu_{21} h_k K_1\big) \triangleq K_2,$$

即 K_2 可以作为 $f\big(x_k+\lambda_2 h_k, y(x_k+\lambda_2 h_k)\big)$ 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

$$\begin{aligned} y(x_k+\lambda_3 h_k) &= y(x_k) + \int_{x_k}^{x_k+\lambda_3 h_k} f(x,y)\,\mathrm{d}x \\ &= y(x_k) + \int_{x_k}^{x_k+\lambda_2 h_k} f(x,y)\,\mathrm{d}x + \int_{x_k+\lambda_2 h_k}^{x_k+\lambda_3 h_k} f(x,y)\,\mathrm{d}x \\ &\approx y_k + \mu_{31} h_k K_1 + \mu_{32} h_k K_2, \end{aligned}$$

其中 μ_{31} 和 μ_{32} 为常数 (具体表达式后面再给出). 于是

$$f\big(x_k+\lambda_3 h_k, y(x_k+\lambda_3 h_k)\big) \approx f\big(x_k+\lambda_3 h_k, y_k+\mu_{31}h_k K_1+\mu_{32}h_k K_2\big) \triangleq K_3,$$

即 K_3 可以作为 $f\big(x_k+\lambda_3 h_k, y(x_k+\lambda_3 h_k)\big)$ 的近似值.

(4) 依此类推, 我们可以求得 $f\big(x_k+\lambda_i h_k, y(x_k+\lambda_i h_k)\big)$ 的近似值:

$$K_i \triangleq f\Big(x_k+\lambda_i h_k, y_k+h_k\sum_{j=1}^{i-1}\mu_{ij}K_j\Big), \quad i=4,5,\dots,r.$$

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

$$y_{k+1} = y_k + h_k\sum_{i=1}^{r}\alpha_i K_i, \quad k=0,1,2,\dots,n-1, \tag{10.11}$$

其中

$$K_1 = f(x_k,y_k), \quad K_i = f\Big(x_k+\lambda_i h_k, y_k+h_k\sum_{j=1}^{i-1}\mu_{ij}K_j\Big).$$

✍ 虽然在上面的推导过程中可以直接计算出 μ_{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 μ_{ij} 和 α_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

$$y(x_{k+1}) = y(x_k) + \int_{x_k}^{x_{k+1}} f(x,y)\,\mathrm{d}x = y(x_k) + h_k f\big(\xi, y(\xi)\big).$$

所以 Runge-Kutta 法 (10.11) 就是用 $\sum_{i=1}^{r}\alpha_i K_i$ 来近似 $f\big(\xi,y(\xi)\big)$.

如何确定 Runge-Kutta 法中的参数

- 首先确定 r 的大小和 λ_i, 即数值积分的节点.

- 用待定系数法计算参数 α_i 和 μ_{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

$$\begin{aligned} R_{k+1} &= y(x_{k+1}) - y(x_k) - \alpha_1 h f(x_k, y(x_k)) \\ &= y'(x_k)h + \frac{1}{2}y''(x_k)h^2 + \mathcal{O}(h^3) - \alpha_1 h y'(x_k) \\ &= (1-\alpha_1)y'(x_k)h + \frac{1}{2}y''(x_k)h^2 + \mathcal{O}(h^3). \end{aligned}$$

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

$$y_{k+1} = y_k + h(\alpha_1 h K_1 + \alpha_2 h K_2), \quad K_1 = f(x_k, y_k),\ K_2 = f\big(x_k + \lambda_2 h, y_k + \mu_{21} h K_1\big).$$

类似地, 利用 Taylor 展开, 可得局部截断误差为

$$\begin{aligned} R_{k+1} = {} & (1-\alpha_1-\alpha_2)y'(x_k)h \\ & + \left(\frac{1}{2} - \alpha_2\lambda_2\right) f'_x(x_k, y(x_k))h^2 \\ & + \left(\frac{1}{2} - \alpha_2\mu_{21}\right) f'_y(x_k, y(x_k))f(x_k, y(x_k))h^2 + \mathcal{O}(h^3). \end{aligned}$$

令

$$(1-\alpha_1-\alpha_2) = 0, \quad \frac{1}{2} - \alpha_2\lambda_2 = 0, \quad \frac{1}{2} - \alpha_2\mu_{21} = 0, \tag{10.12}$$

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

$$y_{k+1}=y_k+\frac{1}{2}h(K_1+K_2),\quad K_1=f(x_k,y_k),\ K_2=f\big(x_k+h,y_k+hK_1\big).$$

这其实就是改进的 Euler 法.

(2) 取 $a=1$, 则可得迭代格式

$$y_{k+1}=y_k+hK_2,\quad K_1=f(x_k,y_k),\ K_2=f\left(x_k+\frac{1}{2}h,y_k+\frac{1}{2}hK_1\right).$$

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

$$\begin{aligned}&y_{k+1}=y_k+\frac{h}{6}(K_1+4K_2+K_3),\\&K_1=f(x_k,y_k),\ K_2=f\left(x_k+\frac{1}{2}h,y_k+\frac{1}{2}hK_1\right),\ K_3=f\left(x_k+h,y_k-hK_1+2hK_2\right).\end{aligned}$$

$r=4$ 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

$$\begin{aligned}y_{k+1}&=y_k+\frac{h}{6}(K_1+2K_2+2K_3+K_4),\\K_1&=f(x_k,y_k),\\K_2&=f\left(x_k+\frac{1}{2}h,y_k+\frac{1}{2}hK_1\right),\\K_3&=f\left(x_k+\frac{1}{2}h,y_k+\frac{1}{2}hK_2\right),\\K_4&=f\left(x_k+h,y_k+hK_3\right).\end{aligned}\tag{10.13}$$

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)

$$\begin{cases} \dfrac{\mathrm{d}y}{\mathrm{d}x} = f(x,y) = y - \frac{12x}{y}, \quad x \in [0,1], \\ y(0) = y_0 = 2. \end{cases}$$

10.3 单步法的稳定性

稳定性是微分方程 (包括 ODE 和 PDE) 数值求解的一个重要概念.

问题的稳定性

首先考虑初值问题 (10.1) 本身的稳定性, 即问题的解对初值的敏感性. 我们知道, 不同的初值所对应的解是不一样的, 如果初值带有一定的误差或扰动, 则不可避免地会引起解的偏差.

定义 10.5 对于初值问题 (10.1), 假定解在 $[a,\infty]$ 内存在唯一. 考虑由于初值 y_0 的扰动而使得解 $y(x)$ 产生偏差, 如果 $x\to\infty$ 时 $y(x)$ 的偏差能控制在某个有界范围内, 则称初值问题是稳定的, 否则就是不稳定的. 特别地, 如果 $x\to\infty$ 时, $y(x)$ 的偏差收敛到 0, 则称该初值问题是渐进稳定的 (asympotically stable).

例 10.7 讨论下面的初值问题的稳定性:

$$\begin{cases}\dfrac{\mathrm{d}y}{\mathrm{d}x}=f(x,y)=\lambda y, & x\in[a,\infty],\\ y(a)=y_0,\end{cases}$$

✍ 一般来说, 常微分方程初值问题的稳定性讨论比较复杂, 特别是当 $f(t,y)$ 是非线性时, 这时可以考虑用 Taylor 展开用线性常微分方程做近似, 从而研究其局部稳定性.

✍ 实际遇到的大多数初值问题是稳定的, 我们这里假定需求解的初值问题是稳定性.

10.4 线性多步法

10.4.1 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 $y_1, y_2, \ldots, y_k$ 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 $x_{k-s+1}, x_{k-s+1}, \ldots, x_k$ 的 $s-1$ 次插值多项式:

$$L_{s-1}(x,y) = f\big(x_{k-s+1}, y(x_{k-s+1})\big)l_0(x) + f\big(x_{k-s+2}, y(x_{k-s+2})\big)l_1(x) + \cdots + f\big(x_k, y(x_k)\big)l_s(x),$$

其中 $l_j(x)$ 为 $s-1$ 次 Lagrange 基函数, 即

$$l_j(x) = \prod_{i=0,\ i\neq j}^{s-1} \frac{x - x_{k-s+1+i}}{x_{k-s+1+j} - x_{k-s+1+i}}, \quad j = 0, 1, 2, \ldots, s.$$

代入 (10.9) 可得

$$\begin{aligned} y(x_{k+1}) &= y(x_k) + \sum_{j=0}^{s-1} f\big(x_{k-s+1+j}, y(x_{k-s+1+j})\big) \int_{x_k}^{x_{k+1}} l_j(x)\,\mathrm{d}x \\ &\approx y_k + h\sum_{j=0}^{s} b_j f\big(x_{k-s+1+j}, y(x_{k-s+1+j})\big), \end{aligned}$$

其中

$$b_j = \frac{1}{h}\int_{x_k}^{x_{k+1}} l_j(x)\,\mathrm{d}x. \tag{10.14}$$

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

$$\boxed{y_{k+1} = y_k + h\sum_{j=0}^{s-1} b_j f\big(x_{k-s+1+j}, y(x_{k-s+1+j})\big)}. \tag{10.15}$$

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**:

$$y_{k+1} = y_k + hf\big(x_k, y_k\big).$$

- 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**:

$$y_{k+1} = y_k + h\left(\frac{23}{12}f(x_k, y_k) - \frac{4}{3}f(x_{k-1}, y_{k-1}) + \frac{5}{12}f(x_{k-2}, y_{k-2})\right).$$

10.4.2 一般线性多步法

Adams 法只是线性多步法中的一类方法法. 一般地, 线性 s 步法可表示为

$$y_{k+1}=\sum_{j=0}^{s-1}a_j y_{k-s+1+j}+h\sum_{j=0}^{s}b_j f\left(x_{k-s+1+j},y_{k-s+1+j}\right),\quad k=s-1,s,\dots, \tag{10.16}$$

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 $a_0=a_1=\cdots=a_{s-2}=0,a_{s-1}=1$, 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-Monlton 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

$$\begin{aligned}R_{k+1}&=y(x_{k+1})-\sum_{j=0}^{s-1}a_j y(x_{k-s+1+j})-h\sum_{j=0}^{s}b_j f\left(x_{k-s+1+j},y(x_{k-s+1+j})\right)\\&=\sum_{i=0}^{\infty}\frac{1}{i!}y^{(i)}(x_{k-s+1})s^ih^i-\sum_{j=0}^{s-1}a_j\sum_{i=0}^{\infty}\frac{1}{i!}y^{(i)}(x_{k-s+1})j^ih^i+h\sum_{j=0}^{s}b_j\sum_{i=0}^{\infty}\frac{1}{i!}y^{(i+1)}(x_{k-s+1})j^ih^i\\&=\left(1-\sum_{j=0}^{s-1}a_j\right)y(x_k)+\sum_{i=1}^{\infty}\frac{1}{i!}\left(s^i-\sum_{j=0}^{s-1}a_jj^i-i\sum_{j=0}^{s}b_jj^{i-1}\right)y^{(i)}(x_k)h^i\end{aligned}$$

所以要使得多步法是 p 阶的, 即 $R_{k+1}=\mathcal{O}(h^{p+1})$, 必须要求

$$\begin{cases}1-\sum\limits_{j=0}^{s-1}a_j=0,\\ s^i-\sum\limits_{j=0}^{s-1}a_jj^i-i\sum\limits_{j=0}^{s}b_jj^{i-1}=0,\quad i=1,2,\dots,p.\end{cases} \tag{10.17}$$

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

$$q(x)-\tilde{q}(x)\ln x=c(x-1)^{p+1}+\mathcal{O}(|x-1|^{p+2}),\quad x\to 1,$$

其中 $q(x)\triangleq 1-\sum\limits_{j=0}^{s-1}a_jx^j,\ \tilde{q}(x)\triangleq\sum\limits_{j=0}^{s}b_jx^j$.

证明. 做变量代换 $x = e^z$, 则 $x \to 1$ 等价于 $z \to 0$. 在 $z = 0$ 处做 Taylor 展开即可. □

✍ 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

下面列举几个常见的多步法, 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):
$$y_{k+1} = y_{k-3} + \frac{4h}{3}(2f_k - f_{k-1} + 2f_{k-2}).$$

10.4.3 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:

$$y_{k+1}^{[i+1]} = \sum_{j=0}^{s-1} a_j y_{k-s+1+j} + h\sum_{j=0}^{s-1} b_j f_{k-s+1+j} + hb_s f\left(x_{k+1}, y_{k+1}^{[i]}\right), \quad i = 0, 1, 2, \ldots, \tag{10.18}$$

其中 $y_{k+1}^{[0]}$ 为迭代初值. 由不动点迭代的收敛性定理 (5.4) 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.

10.4.4 多步法的相容性和收敛性

针对线性多步法 (10.16), 构造对应的第一和第二特征多项式:

$$\rho(\lambda) = \lambda^s - \sum_{j=0}^{s-1} a_j \lambda^j, \quad \sigma(\lambda) = \sum_{j=0}^{s} b_j \lambda^j.$$

与单步法一样, 我们称多步法 (10.16) 是**相容**的, 当且仅当多步法至少是 1 阶的, 即局部截断误差 $R_{k+1} = \mathcal{O}(h^2)$. 由 (10.17) 可直接得到下面的结论.

定理 10.8 线性多步法 (10.16) 是相容的当且仅当

$$\rho(1) = 0, \quad \rho'(1) = \sigma(1).$$

10.5 一阶方程组与高阶方程 *

10.5.1 一阶方程组

考虑一阶常微分方程组

$$\begin{cases} \dfrac{\mathrm{d}Y}{\mathrm{d}x} = F(x,Y) \\ Y(x_0) = Y_0 \end{cases} \qquad x \in [a, \leq b], \quad (x_0 = a)$$

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

$$\begin{aligned} Y_{k+1} &= Y_k + \frac{h}{6}(K_1 + 2K_2 + 2K_3 + K_4), \quad k = 0,1,2,\ldots,n, \\ K_1 &= F(x_k, Y_k), \\ K_2 &= F\left(x_k + \frac{1}{2}h, Y_k + \frac{1}{2}hK_1\right), \\ K_3 &= F\left(x_k + \frac{1}{2}h, Y_k + \frac{1}{2}hK_2\right), \\ K_3 &= F\left(x_k + h, Y_k + hK_3\right). \end{aligned}$$

例 10.8 用经典 Runge-Kutta 法求解初值问题

$$\begin{cases} y' = f(x,y,z), \\ z' = g(x,y,z), \\ y(x_0) = y_0,\ z(x_0) = z_0. \end{cases}$$

10.5.2 高阶方程

考虑 n 阶常微分方程

$$\begin{cases} \dfrac{\mathrm{d}^m y}{\mathrm{d}x^m} = f\left(x, y, y^{(1)}, \ldots, y^{(m-1)}\right) \\ y(x_0) = y_0,\ y^{(1)}(x_0) = y_0^{(1)}, \ldots, y^{(m-1)}(x_0) = y_0^{(m-1)}, \end{cases} \qquad x \in [a, \leq b]. \quad (x_0 = a)$$

引入变量

$$z_1 = y, \quad z_2 = y^{(1)}, \ldots, z_m = y^{(m-1)},$$

则原方程就等价于下面的一阶方程组

$$\begin{cases} z_1' = z_2 \\ z_2' = z_3 \\ \vdots \\ z_m' = f\left(x, z_1, z_2, \dots, z_m\right) \\ z_1(x_0) = y_0,\ z_2(x_0) = y_0^{(1)}, \dots, z_m(x_0) = y_0^{(m-1)}. \end{cases}$$

如何就可以用四阶经典 Runge-Kutta 法进行数值求解.

例 10.9 求解 Van der Pol 方程 (ODE_van_der_Pol.m)

$$\begin{cases} \dfrac{\mathrm{d}^2 y}{\mathrm{d}x^2} + \mu\left(\dfrac{1}{3}\left(\dfrac{\mathrm{d}y}{\mathrm{d}x}\right)^3 - \dfrac{\mathrm{d}y}{\mathrm{d}x}\right) + y = 0, \\ y(0) = 1,\ y'(0) = 1. \end{cases}$$

求解区间为 $[0, 20]$.

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. 194, 195

[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. 184

[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. 194, 195

[4] Åke Björck, Solving linear least square problems by Gram-Schmidt orthogonalization, BIT, 7 (1967), 1–21. 65

[5] Åke Björck, *Numerical Methods for Least Squares Problems*, SIAM, Philadelphia, PA, 1996. 52

[6] S. Börm and C. Mehl, *Numerical Methods for Eigenvalue Problems*, De Gruyter, 2012. 184

[7] J. C. Butcher, *Numerical Methods for Ordinary Differential Equations*, 3rd Edition, John Wiley & Sons, 2016. 196

[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, 194, 195

[9] J.J.M. Cuppen, A Divide and Conquer Method for the Symmetric Tridiagonal Eigenproblem, *Numerische Mathematik*, 36 (1981), 177–195. 191

[10] T. A. Davis, *Direct Methods for Sparse Linear Systems*, SIAM, 2006. 27

[11] J. W. Demmel, *Applied Numerical Linear Algebra*, SIAM, Philadelphia, PA, 1997.

[12] J. E. Jr Dennis and R. B. Schnabel, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, Prentice Hall, Englewood Cliffs, 1983. 97

[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.

[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.

[15] I. S. Duff, A. M. Erisman and J. K. Reid, *Direct Methods for Sparse Matrices*, 2nd edition, Oxford, 2017. 27

[16] K. Fernando and B. Parlett, Accurate singular values and differential qd algorithms, *Numerische Mathematik*, 67 (1994), 191–229.

[17] C. Gear, *Numerical Initial Value Problems in Ordinary Differential Equations*, Prentice-Hall, Englewood

Cliffs, 1971. 196

[18] D. F. Griffiths and D. J. Higham, *Numerical Methods for Ordinary Differential Equations: Initial Value Problems*, Springer, 2010. 196

[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. 96

[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.

[21] G. H. Golub and C. F. Van Loan, *Matrix Computations*, The 4th Editon, The Johns Hopkins University Press, Baltimore, MD, 2013. 17, 27, 67, 76, 184

[22] J. F. Grcar, Mathematicians of Gaussian Elimination, *Notices of the AMS*, Number 6, 58 (2011), 782–792. 27

[23] 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.

[24] 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.

[25] 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. 193

[26] E. Hairer and G. Wanner, *Solving Ordinary Differential Equations II: Stiff and Differential-Algebraic Problems*, 2nd Edition, Springer, 1996. 196

[27] E. Hairer, S. P. Nørsett and G. Wanner, *Solving Ordinary Differential Equations I: Nonstiff Problems*, 2nd Edition, Springer, 2009. 196

[28] 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. 90

[29] N. J. Higham, *Accuracy and Stability of Numerical Algorithms*, Second Edition, SIAM, Philadelphia, 2002. 65

[30] R. A. Horn and C. R. Johnson, *Matrix Analysis*, Cambridge University Press, New York, 1985. 2nd edition, 2013. 4, 8, 59, 67

[31] R. A. Horn and C. R. Johnson, *Topics in Matrix Analysis*, Cambridge University Press, New York, 1991. 4

[32] 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. 26

[33] C. T. Kelley, *Iterative Methods for Linear and Nonlinear Equations*, SIAM, Philadelphia, 1995. 97

[34] C. T. Kelley, *Solving Nonlinear Equations with Newton's Method*, SIAM, Philadelphia, 2003. 97

[35] C. Lanczos Solutions of systems of linear equations by minimized iterations, *Journal of research of the National Bureau of Standards*, 49 (1952), 2341.

[36] S. Li, M. Gu and B. N. Parlett, An Improved DQDS Algorithm, *SIAM Journal on Scientific Computing*,

36 (2014), C290–C308.

[37] D. G. Luenberger and Y. Y. Ye, *Linear and Nonlinear Programming*, 4th edition, Springer, 2016. 91, 93

[38] E. H. Moore, On the reciprocal of the general algebraic matrix, *Bulletin of the American Mathematical Society*, 26 (1920), 394–395.

[39] J. M. Ortega and W. C. Rheinboldt, *Iterative Solution of Nonlinear Equations in Several Variables*, Academic Press, New York-London, 1970. 97

[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. 65

[41] B. N. Parlett, *The Symmetric Eigenvalue Problem*, The 2nd Edition, SIAM, Philadelphia, PA, 1998. 184

[42] B. N. Parlett and O. Marques, An implementation of the dqds algorithm (positive case), *Linear Algebra and its Applications*, 309 (2000), 217–259.

[43] R. Penrose, A generalized inverse for matrices, *Mathematical Proceedings of the Cambridge Philosophical Society*, 51 (1955), 406–413.

[44] A. Quarteroni, R. Sacco and F. Saleri, *Numerical Mathematics*, 2nd Edition, Springer, 2007. 23

[45] J. K. Reid, On the method of conjugate gradients for the solution of large sparse systems of linear equations, in *Large Sparse Sets of Linear Equations*, edited by J. K. Reid, p. 231, Academic Press, New York, 1971. 90

[46] J. Rutter, *A Serial Implementation of Cuppen'S Divide and Conquer Algorithm for the Symmetric Eigenvalue Problem*, Master's Thesis, University of California, 1994.

[47] 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.

[48] Y. Saad, *Numerical Methods for Large Eigenvalue Problems: Theory and Algorithms*, Manchester University Press, Manchester, UK, 1992. 2nd revised edition, SIAM, Philadelphia, 2011. 184

[49] Y. Saad, *Iterative Methods for Sparse Linear Systems*, 2nd edition, SIAM, Philadelphia, 2003. 76, 91, 95

[50] Y. Saad and H. A. van der Vorst, Iterative solution of linear systems in the 20th century, *Journal of Computational and Applied Mathematics*, 123 (2000), 1–33. 90

[51] J. Scott, Scaling and pivoting in an out-of-core sparse direct solver, *ACM Transactions on Mathematical Software*, 37 (2010), Issue 2, Article No. 19. 50

[52] G. W. Stewart, *Matrix Algorithms, Vol II: Eigensystems*, SIAM, Philadelphia, PA, 2001. 184

[53] G. W. Stewart and J. G. Sun, *Matrix Perturbation Theory*, Academic Press, New York, 1990.

[54] L. N. Trefethen, Numerical Analysis, in *Princeton Companion to Mathematics*, Edited by T. Gowers, J. Barrow-Green and I. Leader, Princeton University Press, 2008. 2, 27

[55] L. N. Trefethen, *Approximation Theory and Approximation Practice*, Extended Edition, SIAM, Philadelphia, 2019. 138

[56] L. N. Trefethen and D. Bau, *Numerical Linear Algebra*, SIAM, Philadelphia, PA, 1997. 67

[57] M. Van Barel, R. Vandebril and P. Van Dooren, Computing the eigenvalues of a companion matrix, 2008. http://bezout.dm.unipi.it/cortona08/slides/VanBarel.pdf 194, 195

[58] R. S. Varga, *Matrix Iterative Analysis*, Prentice-Hall, Englewood Cliffs, NJ, 1962. 2nd edition, Springer-Verlag, Berlin, 2000. 76, 87

[59] Vieta's Formulas, From MathWorld–A Wolfram Web Resource, https://mathworld.wolfram.com/VietasFormulas.html, access February 2024. 6

[60] 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

[61] D. S. Watkins, *The Matrix Eigenvalue Problem: GR and Krylov Subspace Methods*, SIAM, Philadelphia, 2007.

[62] J. H. Wilkinson, *The Algebraic Eigenvalue Problem*, Clarendon Press, Oxford University, Oxford, 1965. 56, 65, 184

[63] D. M. Young, *Iterative Solution of Large Linear Systems*, Academic Press, New York, 1971. 76, 88

[64] 北京大学数学系, 高等代数 (第三版), 高等教育出版社, 2003. 4

[65] 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

[66] 戴华, 矩阵论, 科学出版社, 2001. 4

[67] 范金燕, 袁亚湘, 非线性方程组数值方法, 科学出版社, 2018. 97

[68] 蒋尔雄, 矩阵计算, 科学出版社, 2008.

[69] 蒋尔雄, 赵风光, 苏仰峰, 数值逼近 (第二版), 复旦大学出版社, 2008. 138

[70] 李庆扬, 王能超, 易大义, 数值分析, 第五版, 清华大学出版社, 2008. 18

[71] 潘建瑜, 矩阵计算讲义. 12, 88

[72] 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

[73] 孙继广, 矩阵扰动分析, 科学出版社, 北京, 2001.

[74] 王仁宏, 数值逼近 (第二版), 高等教育出版社, 1999. 138, 139

[75] 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. 52

[76] 徐树方, 矩阵计算的理论与方法, 北京大学出版社, 北京, 1995. 94, 95

[77] 徐树方, 钱江, 矩阵计算六讲, 高等教育出版社, 北京, 2011. 195

[78] 袁亚湘, 孙文瑜, 最优化理论与方法, 科学出版社, 北京, 1997. 94

[79] 詹兴致, 矩阵论, 高等教育出版社, 北京, 2008.

[80] 张恭庆, 林源渠, 泛函分析讲义, 上册, 北京大学出版社, 北京, 1987. 9