科学计算简明讲义*

潘建瑜

jypan@math.ecnu.edu.cn

(2025 年 9 月)

*本讲义仅供 “智能算法与大数据分析” 微专业《科学计算》课程教学使用

纸上得来终觉浅，绝知此事要躬行。

目 录

第一讲 科学计算引论

第二讲 非线性方程数值解法

1 科学计算引论

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.

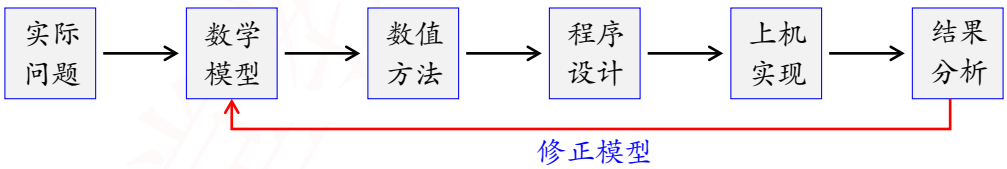
随着计算机技术的飞速发展, 数学方法及计算已成为当今科学研究中不可缺少的手段, 从宇宙飞船到家用电器, 从质量控制到市场营销, 通过建立数学模型, 应用数学理论和方法, 并结合计算机解决实际问题已成为十分普遍的研究模式.

> 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, 是重大科学发现的一种重要手段和科学技术创新的主要方式, 已成为从事现代科学研究的第三种方法.

> 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:



✍ 本课程主要关注其中的数值方法.

数值计算基本思想与方法

数值计算的基本思想是构造解的一个近似, 即 **数值逼近**, 比如用形式简单的函数及其组合来近似复杂函数 (如神经网络), 通过迭代法构造近似解等. 常用方法有

- 等价变换: 将原问题转化为更容易求解的一个或多个等价问题.
- 分而治之: 将原问题分割成若干子问题.
- 外推: 提升算法精度和收敛速度的有效手段之一.

- 递归: 实现高效算法的重要技术, 如 FFT.

数值算法

数值计算的主要研究内容包括

- 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.

一个好的数值算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, 数值算法是近似计算, 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.

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] 喻文健, **数值分析与算法**, 第三版, 清华大学出版社, 2020.

[3] T. Sauer, *Numerical Analysis*, 3rd Edition, 2018.

[4] J. J. Leader, *Numerical Analysis and Scientific Computation*, 2nd Edition, 2022.

[5] A. J. Salgado and S. M. Wise, *Classical Numerical Analysis: A Comprehensive Course*, 2023.

1.1 线性代数基础

本节简要介绍讲义中所涉及的线性代数基础知识.

1.1.1 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.

例 1.2 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, $x_1, x_2, \ldots, x_k$ 是 $\mathbb{S}$ 中的一组向量. 如果存在 k 个不全为零的数 $\alpha_1, \alpha_2, \ldots, \alpha_k \in \mathbb{F}$, 使得

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

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

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

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

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

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

设 $x_1, x_2, \ldots, x_n$ 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 $x_1, x_2, \ldots, x_n$ 线性表示, 则称 $x_1, x_2, \ldots, x_n$ 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 1.3 设 $\mathbb{S}$ 是数域 $\mathbb{F}$ 上的一个线性空间, $x_1, x_2, \ldots, x_k \in \mathbb{S}$, 记

$$\mathrm{span}\{x_1, x_2, \ldots, x_k\} \triangleq \left\{ \alpha_1 x_1 + \alpha_2 x_2 + \cdots + \alpha_k x_k \ : \ \alpha_1, \alpha_2, \ldots, \alpha_k \in \mathbb{F} \right\},$$

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

1.1.2 矩阵的特征值与谱半径

定义 1.1 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.

定义 1.2 (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, 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 1.3 (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, 比如 2 阶以上的 Jordan 块矩阵. 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 1.2 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

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

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.

1.1.3 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 1.4 (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 1.4 $C[a,b]$ 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 1.5 $\mathbb{R}^n/\mathbb{C}^n$ 上的常用范数: 设 $x = [x_1, x_2, \ldots, x_n]^{\mathsf{T}} \in \mathbb{R}^n$ 或 $\mathbb{C}^n$,

- 1-范数: $\|x\|_1 = \displaystyle\sum_{i=1}^n |x_i|$;

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

定义 1.5 (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 1.3 (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

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

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

(1) $f(A) \ge 0, \forall\, A \in \mathbb{R}^{n\times n}$, 且等号当且仅当 $A = 0$ 时成立;
(2) $f(\alpha A) = |\alpha| \cdot f(A), \forall\, A \in \mathbb{R}^{n\times n}, \alpha \in \mathbb{R}$;
(3) $f(A + B) \le f(A) + f(B), \forall\, A, B \in \mathbb{R}^{n\times n}$;
(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 1.4 (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**.

✍ 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

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

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 1.6 $\mathbb{R}^{n\times n}$ 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 1.5 可以证明:

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

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

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

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

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

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

例 1.7 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 1.6 (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

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

谱半径与范数之间的关系

定理 1.7 (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 $\|A\|_\varepsilon<1$;

证明. 设 λ_1 是 A 的模最大特征值, 即 $\rho(A)=|\lambda_1|$. 令 x 为 A 的对应于 λ_1 的满足 $\|x\|=1$ 的特征向量, 即 $Ax=\lambda_1 x$, 则

$$\rho(A)=|\lambda_1|=|\lambda_1|\cdot\|x\|=\|\lambda_1 x\|=\|Ax\|\le\|A\|\|x\|=\|A\|.$$

□

推论 1.8 设 $A\in\mathbb{R}^{n\times n}$, 则 $\|A\|_2^2\le\|A\|_1\cdot\|A\|_\infty$.

1.1.4 内积与内积空间

定义 1.7 (内积与内积空间) 设 $\mathbb{S}$ 是 $\mathbb{R}$ 上的一个线性空间, 定义一个从 $\mathbb{S}\times\mathbb{S}$ 到 $\mathbb{R}$ 的代数运算, 记为 "$(\cdot,\cdot)$", 即对任意 $x,y\in\mathbb{S}$, 都存在唯一的 $f\in\mathbb{R}$, 使得 $f=(x,y)$. 如果该运算满足

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

(2) $(\alpha x,y)=\alpha(x,y),\quad \forall\,\alpha\in\mathbb{R},\ x,y\in\mathbb{S}$;

(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**.

✍ 如果是复数域 $\mathbb{C}$, 则条件 (1) 需改为 $(y,x)=\overline{(x,y)}$.

例 1.8 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

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

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**.

✍ 需要指出的是, 同一个线性空间中可以定义多个内积.

✍ 以上定义的内积是 $\mathbb{R}^n$ 上的标准内积, 若不特别声明, 我们一般采用标准内积.

例 1.9 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 1.8 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.

内积导出范数

定理 1.9 (Cauchy-Schwarz 不等式) 设 $\mathbb{S}$ 是 $\mathbb{R}$ 上的内积空间, 则对任意 $x,y\in\mathbb{S}$, 有

$$|(x,y)|^2\le\|x\|_2^2\cdot\|y\|_2^2. \tag{1.3}$$

✍ 该结论在复数域也成立.

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 1.10 $\mathbb{R}^n$ 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.

1.2 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. 误差分析也是科学计算的重要研究内容.

误差 可用来描述数值计算中近似解的精确程度, 其来源主要有以下几个方面:

- **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**: 也称 **方法误差**, 是指对数学模型进行数值计算时产生的误差.
- **舍入误差**: 由于计算机的机器字长有限, 做算术运算时存在一定的精度限制, 也会产生误差.

✍ 本讲义中, 我们只考虑截断误差和舍入误差对计算结果的影响.

例 1.11 近似计算 $\int_0^1 e^{-x^2}\,\mathrm{d}x$ 的值.

解. 这里我们利用 Taylor 展开进行计算, 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**, 它是由数值算法所造成的.

在计算 S_4 的值时, 假定要求保留 4 位有效数字, 则

$$S_4 = 1 - \frac{1}{3} + \frac{1}{10} - \frac{1}{42} \approx 1.0000 - 0.33333 + 0.10000 - 0.023810 \approx 0.7429$$

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**. □

1.2.1 绝对误差和绝对误差限

精度是数值计算中的重要概念, 衡量精度主要依靠两个指标: 绝对误差和相对误差.

定义 1.9 *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*

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

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, 注意这只是一种表示方法, 不是数学意义上的严格等式.

关于误差和误差限的几点说明

- 绝对误差可能是正的, 也可能是负的.
- 由于精确值通常是不知道的, 因此误差一般也是不可知的, 在做误差估计时, 我们所计算的通常是误差限.
- 误差限不唯一, 通常是越小越好, 一般是指 **所能找到的最小上界**.

1.2.2 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 1.10 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.

1.2.3 有效数字

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

例 1.12 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, 而近似值 $x_3 = 3.1415$ 只有 4 位有效数字.

从上面的例子可以看出, 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 1.10 *设 $\tilde{x}$ 是 x 的近似值, 若 $\tilde{x}$ 可表示为*

$$\tilde{x} = \pm 0.a_1 a_2 \cdots a_n \cdots \times 10^m,$$

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

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

则 $\tilde{x}$ 恰好有 $m-k$ 位有效数字.

在上面的例 1.12 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 1.13 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 1.11 *设 $\tilde{x}$ 是 x 的近似值, 若 $\tilde{x}$ 可表示为*

$$\tilde{x} = \pm 0.a_1 a_2 \cdots a_n \cdots \times 10^m,$$

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

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

则 $\tilde{x}$ 至少有 $m-k$ 位有效数字.

例 1.14 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

$$0.5 \times 10^{-4} < |x_1 - x| = 0.818... \times 10^{-4} \leq 0.5 \times 10^{-3},$$

$$0.5 \times 10^{-5} < |x_2 - x| = 0.181... \times 10^{-4} \leq 0.5 \times 10^{-4}.$$

又 $x_1 = 0.27182 \times 10^1$ 和 $x_2 = 0.27183 \times 10^1$, 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 1.12 (有效数字与相对误差限) 设 $\tilde{x}$ 是 x 的近似值, 若 $\tilde{x}$ 可表示为

$$\tilde{x} = \pm 0.a_1 a_2 \dots a_n \dots \times 10^m,$$

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

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

反之, 若 $\tilde{x}$ 的相对误差满足

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

则 $\tilde{x}$ 至少有 n 位有效数字. (留作课外自习)

✍ 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, 舍入误差也是不一样的. 目前科学计算中常用的浮点数是双精度数和单精度数, 相对误差分别是 $\varepsilon_u \approx 1.1 \times 10^{-16}$ 和 $\varepsilon_u \approx 6.0 \times 10^{-8}$, 即两个浮点数做数值计算 (加减乘除等) 时, 实际计算值 $\tilde{f}$ 与精确值 f 满足 $|\tilde{f} - f| \leq \varepsilon_u |f|$, 工程中也记为 $f = \tilde{f}(1 \pm \varepsilon_u)$.

1.2.4 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.

记 $\tilde{x}$ 为 x 的近似值, 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:

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

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.

1.2.5 问题的适定性和算法的稳定性

数学问题的适定性

定义 1.11 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.

病态问题与条件数

定义 1.12 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 1.15 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. □

注记

- 病态是问题本身固有的性质, 与数值算法无关.
- 对于病态问题, 选择数值算法时需要更加谨慎.

算法的稳定性

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

例 1.16 近似计算 (Demo11_Stablity.m)

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

解. 通过观察可知

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

因此,

$$S_n = \frac{1}{n} - 5S_{n-1}. \tag{1.4}$$

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

$$S_1 = 0.0900, \quad S_2 = 0.0500, \quad S_3 = 0.0833, \quad S_4 = -0.166,$$
$$S_5 = 1.03, \quad S_6 = -4.98, \quad S_7 = 25.0, \quad S_8 = -125.$$

另一方面, 我们有

$$\frac{1}{6(n+1)} = \int_0^1 \frac{x^n}{6}\,\mathrm{d}x \le \int_0^1 \frac{x^n}{x+5}\,\mathrm{d}x \le \int_0^1 \frac{x^n}{5}\,\mathrm{d}x = \frac{1}{5(n+1)}. \tag{1.5}$$

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

设 $\tilde{S}_n$ 是 S_n 的近似值, 则

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

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

事实上, 递推公式 (1.4) 可以改写为

$$S_{n-1} = \frac{1}{5n} - \frac{1}{5}S_n. \tag{1.6}$$

因此, 我们可以先估计 S_8 的值, 然后通过反向递推, 得到其它值. 这样, 误差就以 $\frac{1}{5}$ 的速度衰减.

首先, 可以根据 (1.5) 对 S_8 做简单的估计, 即

$$S_8 \approx \frac{1}{2}\left(\int_0^1 \frac{x^8}{6}\,\mathrm{d}x + \int_0^1 \frac{x^8}{5}\,\mathrm{d}x\right) \approx 0.0204.$$

然后通过递推公式 (1.6) 计算可得

$$S_7 = 0.0209, \quad S_6 = 0.0244, \quad S_5 = 0.0285, \quad S_4 = 0.0343,$$
$$S_3 = 0.0431, \quad S_2 = 0.0580, \quad S_1 = 0.0884, \quad S_0 = 0.182.$$

可以验证, 从 S_1 至 S_5 都至少具有 3 位有效数字.

为了获得更精确的结果 (使得 S_6, S_7, S_8 也具有更多的有效数字), 我们可以从较大的 n 算起, 比如取 $n = 11$, 则

$$S_{11} \approx \frac{1}{2}\left(\int_0^1 \frac{x^{11}}{6}\,\mathrm{d}x + \int_0^1 \frac{x^{11}}{5}\,\mathrm{d}x\right) \approx 0.0153.$$

通过递推公式 (1.6) 计算可得

$$S_8 = 0.0188, S_7 = 0.0212, \quad S_6 = 0.0243, \quad S_5 = 0.0285, \quad S_4 = 0.0343,$$
$$S_3 = 0.0431, \quad S_2 = 0.0580, \quad S_1 = 0.0884, \quad S_0 = 0.182.$$

此时所有的 S_i 都至少具有 3 位有效数字. 计算过程是稳定的. □

✍ 在数值计算中, 误差不可避免, 算法的稳定性非常重要, 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)

1.2.6 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 1.17 已知 $\sqrt{9.01}=3.001666\cdots\approx 3.00$, 计算 $\sqrt{9.01}-3$ 的值, 计算过程中保留 3 位有效数字.

解. 如果直接计算的话, 可得

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

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

但如果换一种计算方法, 如

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

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

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

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

例 1.18 计算 $y=\left(\dfrac{\sqrt{101}-10}{\sqrt{101}+10}\right)^2$.

解. 方法一: 分母有理化

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

方法二: 分子有理化

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

方法三: 直接将 $\sqrt{101}$ 的近似值代入计算.

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

$\sqrt{101}$	方法一	方法二	方法三
10.04	79.400	6.1911×10^{-6}	3.9840×10^{-6}
10.05	-1.0000	6.1880×10^{-6}	6.2189×10^{-6}
10.06	-81.400	6.1849×10^{-6}	8.9462×10^{-6}

精确值为 $y=6.188042227\cdots\times10^{-6}$. 由此可见, 方法二能较好地避免有效数字的损失. □

例 1.19 在 MATLAB 中用双精度数计算 (Demo12_Significance.m)

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

解. 计算结果如下:

x	E_1	E_2
1.0000000000	0.649223205204762	0.649223205204762
0.1000000000	0.501252086288577	0.501252086288571
0.0100000000	0.500012500208481	0.500012500208336
0.0010000000	0.500000124992189	0.500000125000021
0.0001000000	0.499999998627931	0.500000001250000
0.0000100000	0.500000041386852	0.500000000012500
0.0000010000	0.500044450291337	0.500000000000125
0.0000001000	0.499600361081322	0.500000000000001
0.0000000100	0.000000000000000	0.500000000000000
0.0000000010	0.000000000000000	0.500000000000000
0.0000000001	0.000000000000000	0.500000000000000

从计算结果可以看出, 当 x 趋于 0 时, 按照 E_1 的方式计算得到的结果显然是错误的, 而按 E_2 方式则能计算出很好的近似结果. □

例 1.20 计算 $y=\ln 2$. (Demo13_ln.m)

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

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

将 $x=1$ 代入后计算结果为

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

计算到第 100 项, 误差仍有 0.05.

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

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

将 $x = 1/3$ 代入后计算结果为

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

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

(2) 避免数量级相差很大的数相除

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 1.21 计算 $S = 1 + 2 + 3 + \cdots + 100 + 10^{16}$. (Demo14_Sum.m)

从小到大计算, 结果为

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

从大到小计算, 结果为

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

(4) 简化计算

尽量减少运算次数, 从而减少误差的积累.

例 1.22 多项式计算. 设多项式 (Demo15_Poly.m)

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

试计算 $p(3)$ 的值.

解. 如果直接计算的话, 可得

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

需要做 15 次乘法和 5 次加法.

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

事实上, 我们可以将多项式改写为

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

这样就只需做 5 次乘法和 5 次加法. 显然这是更佳的计算方案. □

✍ 在计算多项式的值时, 我们都是将多项式改写成

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

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

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

1.3 课后练习

练习 1.1 设 $x = \begin{bmatrix} 3.1 \\ -9.4 \\ 2.2 \\ 8.4 \end{bmatrix}$, 计算 $\|x\|_1, \|x\|_2, \|x\|_\infty$.

练习 1.2 设矩阵 $A = \begin{bmatrix} 5 & -4 & 2 \\ -4 & 5 & -2 \\ 2 & -2 & 1 \end{bmatrix}$. 计算 $\|A\|_1, \|A\|_2, \|A\|_\infty, \|A\|_F$.

练习 1.3 证明:

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

(2) 对任意的相容矩阵范数 $\|\cdot\|$, 有 $\|I\| \geq 1$.

练习 1.4 设 $\|\cdot\|$ 是 $\mathbb{R}^m$ 空间上的一个向量范数, $A \in \mathbb{R}^{m\times n}$, 且 $\mathrm{rank}(A) = n$.

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

练习 1.5 设 $A \in \mathbb{R}^{m\times n}$, $B \in \mathbb{R}^{n\times m}$, 证明: $\mathrm{tr}(AB) = \mathrm{tr}(BA)$.

练习 1.6 分别用 $x_1 = \dfrac{22}{7}$ 和 $x_2 = \dfrac{355}{113}$ 作为 $\pi = 3.1415926535\cdots$ 的近似值, 各有几位有效数字?

练习 1.7 设 $\tilde{x}_1 = 3.1$ 和 $\tilde{x}_2 = 0.7$ 都是四舍五入后得到得近似值, 试估计 $\tilde{x}_1 + \tilde{x}_2$, $\tilde{x}_1\tilde{x}_2$ 和 $\tilde{x}_1/\tilde{x}_2$ 的误差.

练习 1.8 求方程 $\dfrac{1}{2}x^2 - 10x + \dfrac{1}{2} = 0$ 的两个根, 要求至少保留 3 位有效数字. (取 $\sqrt{11} \approx 3.32$)

练习 1.9* 设 $A \in \mathbb{R}^{n\times n}$. 试证明对任意矩阵范数都有 $\rho(A) \le \|A\|$.

练习 1.10* 设 $A \in \mathbb{R}^{n\times n}$ 是对称正定矩阵, 证明

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

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

$$(x, y) = y^{\mathsf{T}} A x.$$

2 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**. 本讲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

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

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

$$f(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n = 0,$$

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.

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.

我们本讲所研究内容就是在**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 **求解区间**, 即计算指定区间内的解.

2.1 对分法

2.1.1 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 2.1 *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 2.1. 对分法

1: 给定函数 $f(x)$ 和求解区间 $[a,b]$, 以及精度要求 $\mathsf{tol}>0$
2: 令 $a_1=a, b_1=b$
3: 计算 $f(a_1)$ 和 $f(b_1)$, 如果 $|f(a_1)|<\mathsf{tol}$, 则返回数值解 $x=a_1$ 并停止计算; 如果 $|f(b_1)|<\mathsf{tol}$, 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 $|f(x_k)|<\mathsf{tol}$ 或者 $|b_k-a_k|<\mathsf{tol}$, 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.

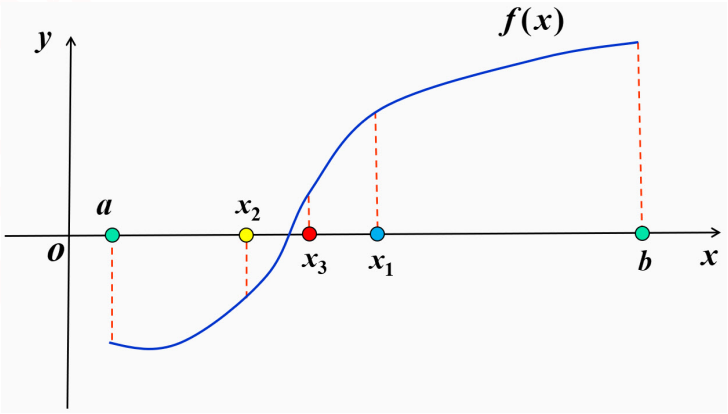


图 2.1. 对分法的几何表示

2.1.2 对分法的收敛性

记算法在第 k 步时得到的有解区间为 $[a_k, b_k]$, 中点为 x_k, 则 $a_1 = a, b_1 = b$, 且

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

由于每次都是对分有解区间, 因此有

$$b_k - a_k = \frac{1}{2}(b_{k-1} - a_{k-1}) = \frac{1}{2^2}(b_{k-2} - a_{k-2}) = \cdots = \frac{1}{2^{k-1}}(b_1 - a_1).$$

因此

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

所以

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

即算法收敛. 因此我们就有下面的收敛性结论.

定理 2.2 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.

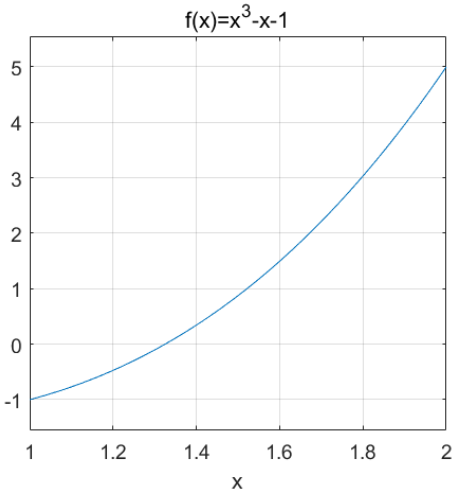
关于对分法的几点注记

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

解. 易知 $f(1) = -1 < 0, f(2) = 5 > 0$, 所以 $f(x)$ 在 $[1, 2]$ 内存在零点 (见下图). 下表中列出了对分法迭代 10 步的数值结果, 其中 $f(a)$ 和 $f(b)$ 只标出其正负号. 由表中可以看出, 迭代 10 步后得到的近似解为 $x_* \approx 1.3252$.

k	$a/f(a)$	$b/f(b)$	x	$f(x)$
1	1.0000/-	2.0000/+	1.5000	0.8750
2	1.0000/-	1.5000/+	1.2500	-0.2969
3	1.2500/-	1.5000/+	1.3750	0.2246
4	1.2500/-	1.3750/+	1.3125	-0.0515
5	1.3125/-	1.3750/+	1.3438	0.0826
6	1.3125/-	1.3438/+	1.3281	0.0146
7	1.3125/-	1.3281/+	1.3203	-0.0187
8	1.3203/-	1.3281/+	1.3242	-0.0021
9	1.3242/-	1.3281/+	1.3262	0.0062
10	1.3242/-	1.3262/+	1.3252	0.0020



□

2.2 不动点迭代法

2.2.1 基本思想

不动点迭代法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:

$$x_{k+1}=\varphi(x_k),\quad k=0,1,2,\ldots, \tag{2.1}$$

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**, $\varphi(x)$ 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 x_* 是 $\varphi(x)$ 的一个不动点.

✍ 不动点迭代法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.

2.2.2 收敛性分析

设 $\varphi(x)$ 连续, 不动点迭代法 (2.1) 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, 则不动点迭代法 (2.1) 是 **发散** 的.

全局收敛性

定理 2.3 (不动点迭代法的全局收敛性) 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

$$|\varphi(x)-\varphi(y)|\le L|x-y|.$$

则对任意初始值 $x_0\in[a,b]$, 不动点迭代法 (2.1) 收敛, 且

$$|x_k-x_*|\le\frac{L}{1-L}|x_k-x_{k-1}|\le\frac{L^k}{1-L}|x_1-x_0|,$$

其中 x_* 是 $\varphi(x)$ 在 $[a,b]$ 内的唯一不动点.

证明. 由条件 (1) 可知, $x_k\in[a,b]$, $k=0,1,2,\ldots$. 再根据条件 (2), 我们有

$$|x_k-x_*|=|\varphi(x_{k-1})-\varphi(x_*)|\le L|x_{k-1}-x_*|. \tag{2.2}$$

依此类推, 可得

$$|x_k-x_*|\le L^k|x_0-x_*|.$$

由于 $0<L<1$, 所以

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

由 (2.2) 直接可得

$$|x_k - x_*| \le L|x_{k-1} - x_*| \le L(|x_k - x_{k-1}| + |x_k - x_*|).$$

因为 $L < 1$, 所以

$$|x_k - x_*| \le \frac{L}{1-L}|x_k - x_{k-1}|.$$

根据不动点迭代格式和条件 (1), 可得

$$|x_k - x_{k-1}| = |\varphi(x_{k-1}) - \varphi(x_{k-2})| \le L|x_{k-1} - x_{k-2}|.$$

不断递推下去, 最后可得

$$|x_k - x_{k-1}| \le L^{k-1}|x_1 - x_0|.$$

所以

$$|x_k - x_*| \le \frac{L}{1-L}|x_k - x_{k-1}| \le \frac{L^k}{1-L}|x_1 - x_0|.$$

□

✍ 条件 $|\varphi(x) - \varphi(y)| \le L|x - y|$ 称为 **Lipschitz 条件**, 当 $L < 1$ 时, 称 $\varphi(x)$ 为 **压缩映射**.
✍ 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

$$|\varphi(y) - \varphi(x)| \le \max_{\xi \in [a,b]} |\varphi'(\xi)| \cdot |y - x|.$$

于是我们可以立即得到下面的结论.

推论 2.4 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

$$|\varphi'(x)| \le L < 1, \quad \forall\, x \in [a, b],$$

则对任意初始值 $x_0 \in [a, b]$, 不动点迭代法 (2.1) 收敛, 且

$$|x_k - x_*| \le \frac{L}{1-L}|x_k - x_{k-1}| \le \frac{L^k}{1-L}|x_1 - x_0|.$$

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 2.2 构造不动点迭代格式, 计算 $f(x) = x^3 - x - 1$ 在 $[1, 2]$ 中的零点. (`NLS_fixpoint_01.m`)

解. 构造 $f(x) = 0$ 的等价形式 $x = \sqrt[3]{x+1} \triangleq \varphi(x)$, 则对任意 $x \in [1, 2]$, 有

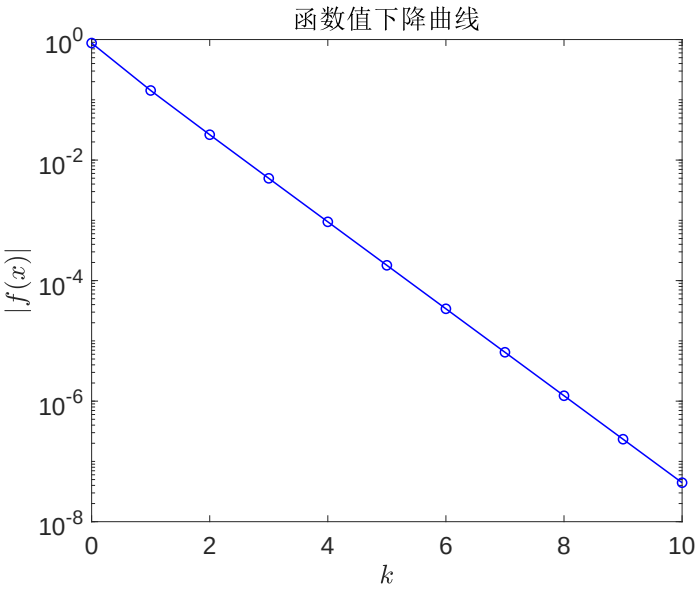
(1) $1 \le \varphi(x) \le 2$;
(2) $|\varphi'(x)| = \left|\frac{1}{3}(x+1)^{-2/3}\right| \le \frac{1}{3}\sqrt[3]{0.25} < 1$.

因此, 不动点迭代 $x_{k+1} = \varphi(x_k) = \sqrt[3]{x_k + 1}$ 全局收敛.

取中点为初始值, 即 $x_0 = 1.5$, 计算结果见下表. 由表中数据可知, 迭代 10 步后, 函数值已经

非常小, 达到了 10^{-8} 量级, 显然比对分法要快很多.

k	x	$\lvert f(x)\rvert$
0	1.5000	8.75e-01
1	1.3572	1.43e-01
2	1.3309	2.63e-02
3	1.3259	4.98e-03
4	1.3249	9.44e-04
5	1.3248	1.79e-04
6	1.3247	3.41e-05
7	1.3247	6.47e-06
8	1.3247	1.23e-06
9	1.3247	2.33e-07
10	1.3247	4.43e-08



思考: 如果取迭代函数 $\varphi(x) = x^3 - 1$, 则结果怎样? □

✍ 需要指出的是, 定理 2.3 和推论 2.4 中给出的是充分条件, 不是充要条件.

局部收敛性

定义 2.1 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 $x_0 \in U_\delta(x_*)$, 不动点迭代法 (2.1) 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 2.5 设 x_* 是 $\varphi(x)$ 的不动点, 若 $\varphi'(x)$ 在 x_* 的某个邻域内连续且

$$|\varphi'(x_*)| < 1,$$

则不动点迭代法 (2.1) 局部收敛.

证明. 由条件 $\varphi'(x)$ 在 x_* 的某个邻域内连续且 $|\varphi'(x_*)| < 1$ 可知, 存在某个领域 $U_\delta(x_*) \triangleq [x_* - \delta, x_* + \delta]$ 和正常数 $L < 1$ 使得

$$|\varphi'(x)| \le L < 1, \quad \forall\, x \in U_\delta(x_*).$$

此外, 对任意 $x \in U_\delta(x_*)$, 有

$$|\varphi(x) - x_*| = |\varphi(x) - \varphi(x_*)| \le L|x - x_*| < \delta,$$

所以 $\varphi(x) \in U_\delta(x_*)$. 根据推论 2.4, 对任意 $x_0 \in U_\delta(x_*)$, 迭代法均收敛. □

2.2.3 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 2.2 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 2.5 可以立即得到下面的结论.

推论 2.6 设 x_* 是 $\varphi(x)$ 的不动点, 若 $\varphi'(x)$ 在 x_* 的某个邻域内连续且

$$|\varphi'(x_*)| < 1,$$

则不动点迭代法 (2.1) 至少局部线性收敛.

关于收敛阶的判定, 我们有下面的定理.

定理 2.7 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

$$\varphi'(x_*) = \varphi''(x_*) = \cdots = \varphi^{(p-1)}(x_*) = 0, \quad \varphi^{(p)}(x_*) \neq 0, \tag{2.3}$$

则不动点迭代法 (2.1) 是 p 阶局部收敛的, 且有

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

✍ 注意: 使用定理 2.7 讨论收敛阶时, 一定要确保 x_* 是不动点.

例 2.3 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, 不动点迭代法 (局部) 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:

$$x_{k+1} = \varphi(x_k, x_{k-1}, ..., x_{k-\ell+1}), \quad k = \ell - 1, \ell, \ell + 1, \ldots. \tag{2.4}$$

我们称之为**多点迭代法**. 比如后面会提到的割线法和抛物线法, 都是多点迭代. 显然, 多点迭代法一开始需要给定 ℓ 个初始值, 即 $x_0, x_1, \ldots, x_{\ell-1}$.

2.3 Steffensen 迭代法

不动点迭代法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.

2.3.1 Aitken 加速技巧

已有不动点迭代

$$x_{k+1} = \varphi(x_k). \tag{2.5}$$

给定初值 x_0, 可得

$$x_1 = \varphi(x_0), \quad x_2 = \varphi(x_1).$$

于是

$$x_1 - x_* = \varphi(x_0) - \varphi(x_*) = \varphi'(\xi_1)(x_0 - x_*),$$
$$x_2 - x_* = \varphi(x_1) - \varphi(x_*) = \varphi'(\xi_2)(x_1 - x_*).$$

如果 $\varphi'(x)$ 变化不大, 或者 ξ_1 和 ξ_2 非常接近, 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:

$$y_{k+1} = x_k - \frac{(x_{k+1} - x_k)^2}{x_{k+2} - 2x_{k+1} + x_k}. \tag{2.6}$$

这就是 **Aitken 加速方法**. 这时我们就得到两个迭代序列: $\{x_k\}_{k=0}^{\infty}$ 和 $\{y_k\}_{k=0}^{\infty}$.

定理 2.8 假定迭代 2.5 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_*.

注记

Aitken 加速也称为 **Aitken 外推** (Aitken extrapolation), 或 Aitken's delta-squared process, 由 Alexander Aitken 于 1926 年提出, 是迭代算法中重要的加速技巧.

2.3.2 Steffensen 迭代法

将 Aitken 加速技巧与不动点迭代法相结合, 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 2.9 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, 若原不动点迭代法是线性收敛的, 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 2.4 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 2.5 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)

2.4 Newton 法

Newton 法是当前求解非线性方程 (组) 的一个常用方法, 也是一般情况下的首选方法.

2.4.1 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

$$x_{k+1}=x_k-\frac{f(x_k)}{f'(x_k)}. \tag{2.7}$$

这就是 Newton 法的迭代格式, 其几何意义可以用下面的图像表示:

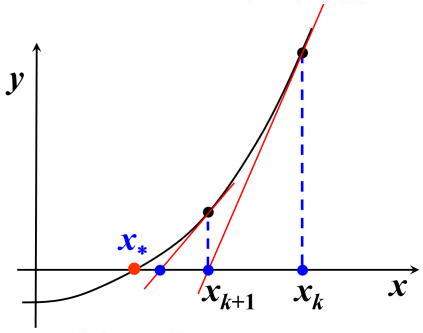


图 2.2. Newton 法的几何含义

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

算法 2.2. 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
    end if
    $x_0=x_1$
end for

2.4.2 Newton 法的收敛性

由迭代格式 (2.7) 可知, Newton 法也是不动点迭代法, 其迭代函数为

$$\varphi(x) = x - \frac{f(x)}{f'(x)}.$$

通过直接计算可得

$$\varphi'(x_*) = 1 - \frac{f'(x_*)f'(x_*) - f(x_*)f''(x_*)}{(f'(x_*))^2} = 0, \quad \varphi''(x_*) = \frac{f''(x_*)}{f'(x_*)}.$$

根据定理 2.7, 我们可以立即得到下面的结论.

定理 2.10 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

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

例 2.6 编写程序, 用 Newton 法求 $f(x) = xe^x - 1$ 的零点. (NLS_Newton_01.m)

解. 迭代格式为

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

取初值 $x_0 = 0.5$, 迭代结果见下表.

k	x	$\lvert f(x)\rvert$
	0.50000000	1.76e-01
1	0.57102044	1.07e-02
2	0.56715557	3.39e-05
3	0.56714329	3.41e-10
4	0.56714329	2.22e-16

从表中数据可以看出, Newton 法迭代 4 步就达到机器精度了, 收敛速度非常快. □

例 2.7 用 Newton 法求 $f(x) = x^2 - C = 0$ 的正根, 并判断收敛性, 其中 $C > 0$.

解. 易知, Newton 法的迭代格式为

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)} = \frac{1}{2}\left(x_k + \frac{C}{x_k}\right).$$

因此可得

$$x_{k+1} - \sqrt{C} = \frac{1}{2x_k}(x_k - \sqrt{C})^2, \quad x_{k+1} + \sqrt{C} = \frac{1}{2x_k}(x_k + \sqrt{C})^2.$$

所以

$$\frac{x_{k+1} - \sqrt{C}}{x_{k+1} + \sqrt{C}} = \left(\frac{x_k - \sqrt{C}}{x_k + \sqrt{C}}\right)^2.$$

以此类推, 可知

$$\frac{x_{k+1}-\sqrt{C}}{x_{k+1}+\sqrt{C}}=\left(\frac{x_0-\sqrt{C}}{x_0+\sqrt{C}}\right)^{2^k}\triangleq q^{2^k},$$

其中 $q=\dfrac{x_0-\sqrt{C}}{x_0+\sqrt{C}}$. 假定 $x_0>0$, 则 $|q|<1$. 直接求解可得

$$x_k-\sqrt{C}=2\sqrt{C}\frac{q^{2^k}}{1-q^{2^k}}\to 0\quad (k\to\infty).$$

由此可知, 只要 $x_0>0$, 则 Newton 法总是收敛到正根 $x_*=\sqrt{C}$, 也就是说, 此时 Newton 法在 $(0,+\infty)$ 内是全局收敛的. □

思考: 如果 $x_0<0$, 则结果会怎样?

✍ 一般来说 Newton 法只是局部收敛, 如果初值离真解太远可能就不收敛, 因此初值的选取很重要但也比较困难. 幸运得是, 对于计算平方根, Newton 法是全局收敛的, 因此是安全的.

✍ Newton 法的优点是收敛速度快 (至少二阶局部收敛), 特别是当迭代点充分靠近精确解时. 但缺点是

- 对重根收敛速度较慢, 只有线性收敛;
- 对初值的选取很敏感, 要求初值相当接近真解, 因此在实际使用时, 可以先用其它方法获取一个近似解, 然后使用 Newton 法加速;
- 每一次迭代都需要计算导数, 难度和工作量都可能会比较大.

2.4.3 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.

2.4.4 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

$$|f(x_{k+1})|<|f(x_k)|, \tag{2.8}$$

即保持函数的绝对值是下降的, 这样就能保证全局收敛性. 具体做法是加入一个 **下山因子** 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 (2.8) 为止.

2.4.5 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 2.8 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)

2.5 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).

2.5.1 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:

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

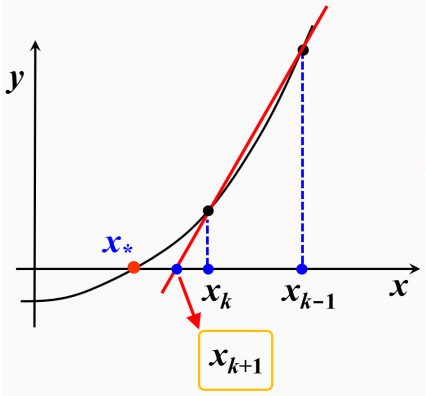


图 2.3. 割线法的几何含义

✍ 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 2.11 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.

2.5.2 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}.

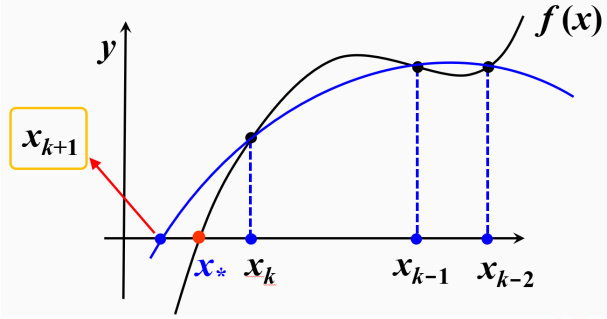


图 2.4. 抛物线法的几何含义

✍ 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 (关于插值方法, 可参见 "第 六 讲 函数插值"), 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**.

2.6 课后练习

练习 2.1 用对分法求方程 $f(x) = x^2 + x - 1$ 在 $[0, 1]$ 中的零点近似值 $\tilde{x}$, 要求 $|f(\tilde{x})| < 0.005$. (可使用计算机或计算器辅助计算, 计算过程中保留 4 位有效数字)

练习 2.2 用下面的迭代格式计算 $f(x) = x^3 - 2x^2 + 1$ 的零点 $x_* = 1$, 试分析各迭代法的 (局部) 收敛性.

(1) $x_{k+1} = 2 - 1/x_k^2$

(2) $x_{k+1} = \sqrt[3]{2x_k^2 - 1}$

(3) $x_{k+1} = \dfrac{1}{\sqrt{2 - x_k}}$

练习 2.3 设 $f(x) \in C^1[-\infty, +\infty]$, 且 $0 < m \le f'(x) \le M$. 试证明: 若 $\alpha \in (0, 2/M)$, 则对任意初值 $x_0 \in \mathbb{R}$, 迭代方法

$$x_{k+1} = x_k - \alpha f(x_k), \quad k = 0, 1, 2, \ldots$$

收敛, 且其极限是 $f(x)$ 的零点 x_*.

练习 2.4 应用 Newton 法于 $f(x) = x^3 - a = 0$, 其中 $a \ge 0$. 导出计算 $\sqrt[3]{a}$ 的迭代公式, 并讨论其全局收敛性, 即对任意给定的非零 $x_0 \in \mathbb{R}$, 迭代是否收敛?

练习 2.5 设 $a > 0$, 并假定 x_0 充分靠近 $\sqrt{a}$. 证明迭代公式

$$x_{k+1} = \frac{x_k(x_k^2 + 3a)}{3x_k^2 + a}, \quad k = 0, 1, 2, \ldots$$

是计算 $\sqrt{a}$ 的三阶方法, 并计算

$$\lim_{k\to\infty} \frac{x_{k+1} - x_k}{(x_k - x_{k-1})^3}.$$

练习 2.6* 设 x_* 是 $f(x)$ 的单重零点, $\varphi(x) = x - p(x)f(x) - q(x)f^2(x)$. 试确定 $p(x)$ 和 $q(x)$, 使得以 $\varphi(x)$ 为迭代函数的求解 $f(x) = 0$ 的不动点迭代法至少具有三阶 (局部) 收敛速度.

练习 2.7* 设 x_* 是 $f(x)$ 的单重零点, 对于 Newton 法 $x_{k+1} = x_k - f(x_k)/f'(x_k)$, 证明

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

(这里假定 x_k 是收敛到 x_* 的)

练习 2.8* 设函数 $\varphi(x) \in C^1[-\infty, +\infty]$, 且满足 $|\varphi'(x)| < 1$, 则迭代格式

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

是否收敛? (如果收敛则给出证明, 否则给出反例)

3 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

$$Ax = b, \quad A \in \mathbb{R}^{n\times n},\ b \in \mathbb{R}^n. \tag{3.1}$$

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.

✍ 在本讲中, 我们总是假定系数矩阵 A 是非奇异的, 即线性方程组 (3.1) 的解存在且唯一.

3.1 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, 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.

例 3.1 求解下面的线性方程组

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

解. 利用 **Gauss 消去法**求解: 先写出增广矩阵, 然后通过初等变换将其转换为阶梯形, 最后通过回代求解. 具体过程可写为

$$\left[\begin{array}{ccc|c} 1 & -2 & 2 & -2 \\ 2 & -3 & -3 & 4 \\ 4 & 1 & 6 & 3 \end{array}\right] \xrightarrow[③-①\times 4]{②-①\times 2} \left[\begin{array}{ccc|c} 1 & -2 & 2 & -2 \\ 0 & 1 & -7 & 8 \\ 0 & 9 & -2 & 11 \end{array}\right] \xrightarrow{③-②\times 9} \left[\begin{array}{ccc|c} 1 & -2 & 2 & -2 \\ 0 & 1 & -7 & 8 \\ 0 & 0 & 61 & -61 \end{array}\right]$$

通过回代求解可得

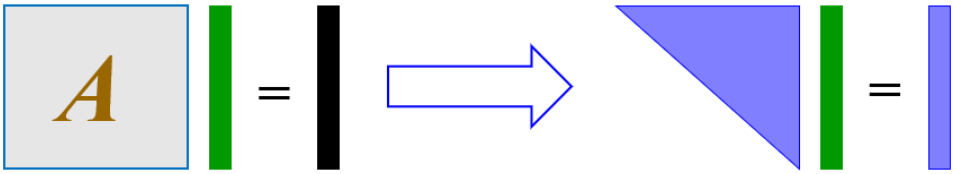
$$\begin{cases} x_3 = -1, \\ x_2 = 8 + 7x_3 = 1, \\ x_1 = -2 + 2x_2 - 2x_3 = 2. \end{cases}$$

□

将以上的做法推广到一般线性方程 $Ax = b$, 即

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

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.

3.1.1 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 $A^{(k)}$:

$$A^{(k)} = \left[\begin{array}{ccccc|c} a_{11}^{(1)} & \cdots & a_{1k}^{(1)} & \cdots & a_{1n}^{(1)} & b_1^{(1)} \\ & \ddots & \vdots & & \vdots & \vdots \\ & & a_{kk}^{(k)} & \cdots & a_{kn}^{(k)} & b_k^{(k)} \\ & & \vdots & \ddots & \vdots & \vdots \\ & & a_{nk}^{(k)} & \cdots & a_{nn}^{(k)} & b_n^{(k)} \end{array}\right], \tag{3.2}$$

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

$$a_{ij}^{(k+1)} = a_{ij}^{(k)} - l_{ik}a_{kj}^{(k)}, \quad b_i^{(k+1)} = b_i^{(k)} - l_{ik}b_k^{(k)}, \quad i,j = k+1, k+2, \ldots, n. \tag{3.3}$$

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 3.1 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 3.2 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.

3.1.2 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.

3.2 矩阵分解法

3.2.1 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 (3.2) 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 (3.3) 可得

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

其中

$$L_k = \begin{bmatrix} 1 & & & & \\ & \ddots & & & \\ & & 1 & & \\ & & -l_{k+1,k} & 1 & \\ & & \vdots & & \ddots \\ & & -l_{n,k} & & 1 \end{bmatrix}, \quad l_{ik} = \frac{a_{ik}^{(k)}}{a_{kk}^{(k)}},\ i = k+1, k+2, \ldots, n. \tag{3.4}$$

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 3.3 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

$$A=LU. \tag{3.5}$$

由引理 3.3 可知 L 是单位下三角矩阵, U 是非奇异上三角矩阵.

由推论 3.2 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 (3.5) 也存在. 事实上, 我们有下面的结论.

定理 3.4 (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)

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

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

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

充分性: 用归纳法.

当 $n=1$ 时, 结论显然成立.

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

考虑 n 阶的矩阵 A, 写成分块形式

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

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

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

令

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

则

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

易知 U 非奇异, 所以 A 存在 LU 分解.

下面证明**唯一性**. 设 A 存在两个不同的 LU 分解:

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

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

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

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

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

即 $\tilde{L} = L, \tilde{U} = U$.

由归纳法可知, 结论成立. □

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 3.1. LU 分解

```
Set L = I, U = 0    % 将 L 设为单位矩阵, U 设为零矩阵
for k = 1 to n − 1 do
    for i = k to n do
        u_ki = a_ki    % 计算 U 的第 k 行
    end for
    for i = k + 1 to n do
        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 3.2. 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
```

3.2.2 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 3.2 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 3.3 求解线性方程组 $Ax = b$, 其中 $A = \begin{bmatrix} 0.02 & 61.3 \\ 3.43 & -8.5 \end{bmatrix}$, $b = \begin{bmatrix} 61.5 \\ 25.8 \end{bmatrix}$, 要求在运算过程中保留 3 位有效数字.

解. 根据 LU 分解算法 3.1, 我们可得

$$l_{11} = 1.00,\ l_{21} = a_{21}/a_{11} \approx 1.72 \times 10^2,\ l_{22} = 1.00,$$

$$u_{11} = a_{11} = 2.00 \times 10^{-2},\ u_{12} = a_{12} = 6.13 \times 10,$$
$$u_{22} = a_{22} - l_{21}u_{12} \approx -8.5 - 1.05 \times 10^4 \approx -1.05 \times 10^4,$$

即

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

解方程组 $Ly = b$ 可得

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

解方程组 $Ux = y$ 可得

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

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

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 3.3. *列主元 Gauss 消去法*

1: **for** $k = 1$ to $n - 1$ **do**
2: $\quad a_{i_k,k} = \max_{k\le i\le n}|a_{i,k}|$ % 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 3.5 (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

$$PA = LU, \tag{3.6}$$

其中 L 为单位下三角矩阵, U 为上三角矩阵. (留作课外自习)

得到 A 的 PLU 分解 (3.6) 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 3.4 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)

解. 由于 $|a_{21}| > |a_{11}|$, 根据部分选主元 LU 分解算法, 我们需要将第一行与第二行交换, 即取

$P=\begin{bmatrix}0&1\\1&0\end{bmatrix}$, 然后计算 $\tilde{A}=PA=\begin{bmatrix}3.43&-8.5\\0.02&61.3\end{bmatrix}$ 的 LU 分解, 即令

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

直接比较等式两边可得

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

即

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

解方程组 $Ly=Pb$ 可得

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

解方程组 $Ux=y$ 可得

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

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

✍ 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 消去法和 PLU

考虑列主元 Gauss 消去法, 设第 k 步选主元时交换了第 k 行与第 i_k 行, 对应的置换矩阵记为 P_k, 则 $i_k\geq k$. 如果 $i_k=k$, 则表示不需要行交换. 整个列主元 Gauss 消去过程可表示为

$$\tilde{L}_{n-1}P_{n-1}\tilde{L}_{n-2}P_{n-2}\cdots\tilde{L}_2P_2\tilde{L}_1P_1A=U,$$

其中 $\tilde{L}_k$ 表示第 k 次 Gauss 消元时对应的单位下三角矩阵, U 是上三角矩阵. 为了能将上式改为成 $PA=LU$ 形式, 我们将上式左端改写为

$$\tilde{L}_{n-1}(P_{n-1}\tilde{L}_{n-2}P_{n-1}^{\mathsf{T}})(P_{n-1}P_{n-2}\tilde{L}_{n-3}P_{n-2}^{\mathsf{T}}P_{n-1}^{\mathsf{T}})P_{n-1}P_{n-2}P_{n-3}\tilde{L}_{n-4}\cdots\tilde{L}_1P_1A.$$

记 $\hat{L}_{n-1}\triangleq\tilde{L}_{n-1}$,

$$\hat{L}_k\triangleq P_{n-1}\cdots P_{k+1}\tilde{L}_kP_{k+1}^{\mathsf{T}}\cdots P_{n-1}^{\mathsf{T}},\quad k=n-2,n-3,\ldots,1,$$

则可知 $\hat{L}_k$ 为单位下三角矩阵, 且与 $\tilde{L}_k$ 具有相同的形状. 于是列主元 Gauss 消去过程可写为

$$\hat{L}_{n-1}\hat{L}_{n-2}\cdots\hat{L}_1P_{n-1}P_{n-2}\cdots P_1A=U.$$

记

$$L\triangleq\hat{L}_1^{-1}\cdots\hat{L}_{n-1}^{-1},\quad P=P_{n-1}\cdots P_1,$$

则有 $PA=LU$.

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 3.4. 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 3.5. 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 3.4)
$y_1 = b_{p_1}$ % 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}
end for
```

✍ 算法 3.5 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. 这就是全主元 Gauss 消去法.

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.

3.2.3 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 3.6 (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)

证明. 首先证明存在性, 我们用数学归纳法来构造矩阵 L.

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

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

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

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

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

易知, L 是对角线元素均为正的下三角矩阵, 且

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

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

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

唯一性可以采用反证法, 留做练习. □

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

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

由此可得计算公式

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

根据这个公式即可得下面的算法描述.

---

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

---

for $j = 1$ to n **do**
  for $k = 1$ to $j-1$ **do**
    $a_{jj} = a_{jj} - a_{jk}^2$
  end for
  $a_{jj} = \sqrt{a_{jj}}$
  for $i = j+1$ to n **do**
    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 3.7. 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 3.6)
$y_1 = b_1/a_{11}$ % 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 $L^{\mathsf{T}}x = y$
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$^{\mathsf{T}}$ 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

$$A = LDL^{\mathsf{T}},$$

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$^{\mathsf{T}}$ 分解**. 由待定系数法, 设

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

比较等式两边的元素, 可得

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

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

5: **end for**
6: **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 3.8. 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 $DL^{\mathsf{T}}x = y$
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, D 的对角线元素都是正的, 如果允许出现负数, 则可得下面的结论.

定理 3.7 *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*

$$A = LDL^{\mathsf{T}},$$

其中 L 为单位下三角矩阵, D 为对角矩阵.

3.2.4 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

$$|b_1| > |c_1| > 0, \quad |b_n| > |a_{n-1}| > 0, \tag{3.7}$$

且

$$|b_i| \geq |a_{i-1}| + |c_i|, \quad a_i c_i \neq 0, \quad i = 1, \ldots, n-1. \tag{3.8}$$

即 A 是 **不可约弱对角占优** 的 (不可约见定义 4.5, 对角占优见定义 4.6). 此时, 我们可以得到下面的三角分解 (Crout 分解)

$$A = \begin{bmatrix} b_1 & c_1 & & \\ a_1 & \ddots & \ddots & \\ & \ddots & \ddots & c_{n-1} \\ & & a_{n-1} & b_n \end{bmatrix} = \begin{bmatrix} \alpha_1 & & & \\ a_1 & \alpha_2 & & \\ & \ddots & \ddots & \\ & & a_{n-1} & \alpha_n \end{bmatrix} \begin{bmatrix} 1 & \beta_1 & & \\ & 1 & \ddots & \\ & & \ddots & \beta_{n-1} \\ & & & 1 \end{bmatrix} \triangleq LU. \tag{3.9}$$

由待定系数法, 我们可以得到递推公式:

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

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 3.8 设三对角矩阵 A 满足条件 (3.7) 和 (3.8). 则 A 非奇异, 且

(1) $|\alpha_1| = |b_1| > 0$;

(2) $0 < |\beta_i| < 1, i = 1, 2, \ldots, n-1$;

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

由定理 3.8 可知, 分解 (3.9) 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 3.9. 追赶法

$\alpha_1 = b_1$
$\beta_1 = c_1/b_1$
$y_1 = f_1/b_1$
for $i = 2$ to $n-1$ **do**
    $\alpha_i = b_i - a_{i-1}\beta_{i-1}$
    $\beta_i = c_i/\alpha_i$
    $y_i = (f_i - a_{i-1}y_{i-1})/\alpha_i$
end for
$\alpha_n = b_n - a_{n-1}\beta_{n-1}$
$y_n = (f_n - a_{n-1}y_{n-1})/\alpha_n$
$x_n = y_n$
for $i = n-1$ to 1 **do**
    $x_i = y_i - \beta_i x_{i+1}$
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

$$A = \begin{bmatrix} b_1 & c_1 & & \\ a_1 & \ddots & \ddots & \\ & \ddots & \ddots & c_{n-1} \\ & & a_{n-1} & b_n \end{bmatrix} = \begin{bmatrix} 1 & & & \\ \gamma_1 & 1 & & \\ & \ddots & \ddots & \\ & & \gamma_{n-1} & 1 \end{bmatrix} \begin{bmatrix} \alpha_1 & c_1 & & \\ & \alpha_2 & \ddots & \\ & & \ddots & c_{n-1} \\ & & & \alpha_n \end{bmatrix}. \tag{3.10}$$

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 (3.9); 如果 A 是列对角占优, 则采用分解 (3.10).

3.3 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**.

3.3.1 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 3.5 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 3.1 *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 $\kappa_2(A) = \dfrac{\max\limits_{1\le i\le n} |\lambda_i|}{\min\limits_{1\le i\le n} |\lambda_i|}$.

例 3.6 计算例 3.5 中系数矩阵 A 的条件数 $\kappa_\infty(A)$ 和 $\kappa_2(A)$.

解. 通过直接计算可得 $A^{-1} = \begin{bmatrix} 10001 & -10000 \\ -10000 & 10000 \end{bmatrix}$. 所以 $\kappa_\infty(A) = \|A\|_\infty \cdot \|A^{-1}\|_\infty \approx 4 \times 10^4$.

由于 A 是对称的, 且 A 的特征值为

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

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

引理 3.9 条件数具有以下性质:

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

3.3.2 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

$$\|\delta x\|\leq\|A^{-1}\|\cdot\|\delta b\|. \tag{3.11}$$

又由 $Ax_*=b$ 可知

$$\|b\|\leq\|A\|\cdot\|x_*\|,\quad \text{即}\quad \frac{1}{\|x_*\|}\leq\frac{\|A\|}{\|b\|}. \tag{3.12}$$

由 (3.11) 和 (3.12) 两边相乘可得

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

定理 3.10 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. 这个倍数就是系数矩阵的条件数, 这也是为什么条件数是衡量线性方程组是否病态的重要指标. 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

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

为了确保问题的适定性, 我们假定 $A+\delta A$ 是非奇异的.

定理 3.11 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 3.12 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），假定 $\|A^{-1}\|\cdot\|\delta A\|<1$, 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

$$\frac{\|\delta x\|}{\|\tilde{x}\|}\leq\kappa(A)\frac{\|\delta A\|}{\|A\|}\tag{3.13}$$

例 3.7 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

$$H_nx=b,$$

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 $n\geq20$ 时, Cholesky 分解计算出来的近似解已经没有任何意义了.

3.4 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.

3.4.1 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.

3.4.2 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.

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.

3.4.3 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

$$Az = b - A\tilde{x} = r. \tag{3.14}$$

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 (3.14) 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 3.10. 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.

3.5 课后练习

练习 3.1 下列矩阵是否存在 LU 分解 (L 单位下三角, U 非奇异上三角)

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

练习 3.2 设 $A = [a_{ij}] \in \mathbb{R}^{n\times n}$, 且 $a_{11} \neq 0$, 经过一次 Gauss 消去法后得到 $A^{(2)} = \begin{bmatrix} a_{11} & * \\ 0 & A_2 \end{bmatrix}$.

证明: (1) 若 A 对称, 则 A_2 也对称; (2) 若 A 对称正定, 则 A_2 也对称正定.

练习 3.3 设矩阵 $A = \begin{bmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 12 \end{bmatrix}$, 计算 A 的 LU 分解和 PLU 分解.

练习 3.4 计算 $A=\begin{bmatrix}4&2&4\\2&37&8\\4&8&14\end{bmatrix}$ 的 Cholesky 分解, 并求解 $Ax=b$, 其中 $b=\begin{bmatrix}6\\-9\\7\end{bmatrix}$.

练习 3.5 设矩阵 $A=\begin{bmatrix}2&-1&0\\-1&2&a\\0&a&2\end{bmatrix}$, 问: 当 a 取何值时, A 存在 Cholesky 分解?

练习 3.6 用追赶法解三对角方程组 $Ax=b$, 其中

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

练习 3.7 设矩阵 $A=\begin{bmatrix}1&2\\2&3\end{bmatrix}$, 计算 $\kappa_2(A)$ 和 $\kappa_\infty(A)$.

练习 3.8 设矩阵 $A=\begin{bmatrix}\lambda&2\lambda\\1&1\end{bmatrix}$ $(\lambda\neq 0)$, 当 λ 取何值时, $\kappa_\infty(A)$ 达到最小.

练习 3.9 证明以下结论:

(1) $\kappa(A)\geq 1,\quad \forall\, A\in\mathbb{R}^{n\times n}$.

(2) 对任意非零常数 $\alpha\in\mathbb{R}$, 都有 $\kappa(\alpha A)=\kappa(A)$.

(3) 对任意正交矩阵 $Q\in\mathbb{R}^{n\times n}$, 有 $\kappa_2(Q)=1$.

(4) 设 Q 是正交矩阵, 则对任意矩阵 A 有 $\kappa_2(QA)=\kappa_2(A)=\kappa_2(AQ)$.

练习 3.10 设 $A\in\mathbb{R}^{n\times n}$ 非奇异, 证明: $\kappa_2(A^{\mathsf{T}}A)=\Big(\kappa_2(A)\Big)^2$.

练习 3.11 将 $A\in\mathbb{R}^{n\times n}$ 写成分块形式

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

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

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

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

练习 3.12* 设 $A,B\in\mathbb{R}^{n\times n}$ 是两个上三角矩阵, $\alpha\in\mathbb{R}$ 是给定常数, 且 $AB-\alpha I$ 非奇异. 试设计求解 $(AB-\alpha I)x=b$ 的算法, 使得运算量为 $\mathcal{O}(n^2)$.

4 线性方程组迭代解法

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. 对于大规模线性方程组, 由于运算量太大, 除了有特殊结构的线性方程组外, 一般采用 **迭代法** 求解.

迭代法基本思想

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

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

迭代法分类

目前常用的迭代法主要有两类, 一类是 **定常迭代法** (也称 **基本迭代法**), 如 Jacobi 迭代法, Gauss-Seidel 迭代法, SOR 迭代法等. 另一类是 **子空间迭代法**, 如共轭梯度法等.

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_*.

一类常用的定常迭代法是基于矩阵分裂的迭代法.

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**:

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

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

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

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

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

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

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

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)`

解. **Jacobi 迭代法**: 迭代格式为

$$\begin{cases} x_1^{(k+1)} = \frac{1}{2}\left(1 + x_2^{(k)}\right) \\ x_2^{(k+1)} = \frac{1}{3}\left(8 + x_1^{(k)} + x_3^{(k)}\right) \\ x_3^{(k+1)} = \frac{1}{2}\left(-5 + x_2^{(k)}\right) \end{cases}$$

直接计算可得

$$x^{(1)} = [0.5000, 2.6667, -2.5000]^{\mathsf{T}}, \ \ldots, \ x^{(\mathbf{21})} = [2.0000, 3.0000, -1.0000]^{\mathsf{T}}.$$

G-S 迭代法: 迭代格式为

$$\begin{cases} x_1^{(k+1)} = \frac{1}{2}\left(1 + x_2^{(k)}\right) \\ x_2^{(k+1)} = \frac{1}{3}\left(8 + x_1^{(k+1)} + x_3^{(k)}\right) \\ x_3^{(k+1)} = \frac{1}{2}\left(-5 + x_2^{(k+1)}\right) \end{cases}$$

直接计算可得

$$x^{(1)} = [0.5000, 2.8333, -1.0833]^{\mathsf{T}}, \ \ldots, \ x^{(\mathbf{9})} = [2.0000, 3.0000, -1.0000]^{\mathsf{T}}.$$

SOR 迭代法: 迭代格式为

$$\begin{cases} x_1^{(k+1)} = x_1^{(k)} + \dfrac{\omega}{2}\left(1 - 2x_1^{(k)} + x_2^{(k)}\right) \\ x_2^{(k+1)} = x_2^{(k)} + \dfrac{\omega}{3}\left(8 + x_1^{(k+1)} - 3x_2^{(k)} + x_3^{(k)}\right) \\ x_3^{(k+1)} = x_3^{(k)} + \dfrac{\omega}{2}\left(-5 + x_2^{(k+1)} - 2x_3^{(k)}\right) \end{cases}$$

令 $\omega = 1.1$, 直接计算可得

$$x^{(1)} = [0.5500, 3.1350, -1.0257]^{\mathsf{T}}, \ \ldots, \ x^{(\mathbf{7})} = [2.0000, 3.0000, -1.0000]^{\mathsf{T}}.$$

□

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)

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

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

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

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

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 $\left\{A^{(k)}\right\}$, 其中

$$A^{(k)} = \begin{bmatrix} a & 1 \\ 0 & a \end{bmatrix}^k, \quad k = 1,2,\ldots.$$

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

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

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

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

证明. 先证明**必要性**. 由条件 $\lim_{k\to\infty} A^{(k)} = 0$ 可知, 对任意算子范数都有 $\lim_{k\to\infty} \|A^{(k)}\| = 0$. 因此, 对

任意 $x \in \mathbb{R}^n$ 有

$$\|A^{(k)}x\| \leq \|A^{(k)}\| \cdot \|x\| \to 0 \ (k \to \infty) \quad \text{即} \quad \lim_{k\to\infty} A^{(k)}x = 0.$$

下面证明**充分性**. 取 $x = e_i$, 即单位矩阵的第 i 列, 则由 $\lim\limits_{k\to\infty} A^{(k)}e_i = 0$ 可知, $A^{(k)}$ 的第 i 列的极限为 0. 令 $i = 1, 2, \ldots, n$, 则可得 $\lim\limits_{k\to\infty} A^{(k)} = 0$. □

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

证明. 由条件 $\|A\| < 1$ 可知,

$$\|A^k\| \leq \|A\|^k \to 0 \ (k \to \infty).$$

所以 $\lim\limits_{k\to\infty} A^k = 0$. □

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 $\rho(A) < 1$.

证明. 先证明**必要性**, 反证法. 假设 $\rho(A) \geq 1$, 则 A 存在特征值 λ, 满足 $|\lambda| \geq 1$. 设其对应的特征向量为 $x \neq 0$, 即 $Ax = \lambda x$, 则 $A^kx = \lambda^k x$. 由于 $|\lambda| \geq 1$, 当 $k \to \infty$ 时 $\lambda^k x$ 不可能收敛到 0, 这与条件矛盾. 所以结论成立, 即 $\rho(A) < 1$.

下面证明**充分性**. 令 $\varepsilon = \frac{1}{2}\big(1 - \rho(A)\big)$, 则 $\varepsilon > 0$. 因此, 由定理 1.7 可知, 存在某个矩阵范数 $\|\cdot\|_\varepsilon$, 使得

$$\|A\|_\varepsilon \leq \rho(A) + \varepsilon = \frac{1}{2}(1 + \rho(A)) < 1.$$

所以由定理 4.3 可知 $\lim\limits_{k\to\infty} A^k = 0$. □

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.

证明. 对迭代格式 (4.3) 两边取极限可得

$$x_* = \lim_{k\to\infty} x^{(k+1)} = \lim_{k\to\infty} \big(M^{-1}Nx^{(k)} + M^{-1}b\big) = M^{-1}Nx_* + M^{-1}b.$$

整理后可得 $(M - N)x_* = b$, 即 $Ax_* = b$, 结论成立. □

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 $\rho(G) < 1$.

证明. 先证明**必要性**. 对任意向量 $\tilde{x} \in \mathbb{R}^n$, 令 $x^{(0)} = \tilde{x} - x_*$, 则

$$x^{(k)} - x_* = (Gx^{(k-1)} + g) - (Gx_* + g) = G(x^{(k-1)} - x_*) = \cdots = G^k(x^{(0)} - x_*) = G^k\tilde{x}.$$

由迭代法 (4.3) 的收敛性可知, $G^k\tilde{x} = x^{(k)} - x_* \to 0$ $(k \to \infty)$. 根据定理 4.2, 我们有 $\lim\limits_{k\to\infty} G^k = 0$. 再由定理 4.4 可知 $\rho(G) < 1$.

下面证明**充分性**. 由条件 $\rho(G) < 1$ 可得 $\lim\limits_{k\to\infty} G^k = 0$. 所以对任意 $x^{(0)} \in \mathbb{R}^n$, 当 $k \to \infty$ 时, 有

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

故 $\lim\limits_{k\to\infty} x^{(k)} = x_*$, 即迭代法 (4.3) 收敛. □

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

✍ 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 4.3.

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 (G-S, SOR) 迭代法收敛的 **充要条件**: $\rho(G_{\mathrm{J}})<1$ ($\rho(G_{\mathrm{GS}})<1$, $\rho(G_{\mathrm{SOR}})<1$)
- Jacobi (G-S, SOR) 迭代法收敛的 **充分条件**: $\|G_{\mathrm{J}}\|<1$ ($\|G_{\mathrm{GS}}\|<1$, $\|G_{\mathrm{SOR}}\|<1$)

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 4.5 (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 $\ge n-1$.

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

$$\begin{cases}A_{22}y_2=f_2,\\ A_{11}y_1=f_1-A_{12}y_2.\end{cases}$$

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.

定义 4.6 (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 4.10 设 $A \in \mathbb{R}^{n\times n}$ 严格对角占优 (或不可约对角占优), 则 A 非奇异.

定理 4.11 设 $A \in \mathbb{R}^{n\times n}$, 若 A 严格对角占优 (或不可约对角占优), 则

- Jacobi 迭代法收敛;
- G-S 迭代法收敛;
- 当 $0 < \omega \leq 1$ 时, SOR 迭代法收敛.

✍ 事实上, SOR 迭代法收敛的一个必要条件是 $0 < \omega < 2$.

4.3.2 对称正定情形

定理 4.12 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;
- G-S 迭代法收敛的充要条件是 A 正定;
- 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 迭代法收敛的充要条件.

解. 易知 A 对称且对角线部分都为正, 因此 Jacobi 迭代收敛的充要条件是 A 和 $2D - A$ 都正定. 通过计算 A 的各阶顺序主子式可知, A 正定的充要条件是

$$\det\left(\begin{bmatrix} 1 & a \\ a & 1 \end{bmatrix}\right) = 1 - a^2 > 0, \quad \det(A) = 1 + 2a^3 - 3a^2 = (a-1)^2(2a+1) > 0,$$

即

$$-\frac{1}{2} < a < 1.$$

类似地, 可知 $2D-A$ 正定的充要条件是

$$-1<a<\frac{1}{2}.$$

所以 Jacobi 迭代收敛的充要条件是

$$-\frac{1}{2}<a<\frac{1}{2},$$

G-S 收敛的充要条件是 A 正定, 即

$$-\frac{1}{2}<a<1,$$

SOR 收敛的充要条件是 A 正定且 $0<\omega<2$, 即

$$-\frac{1}{2}<a<1,\quad 0<\omega<2.$$

□

4.4 共轭梯度法

本节考虑对称正定线性方程组的求解. 共轭梯度法是子空间迭代法的典型代表, 最早由 Hestenes 和 Stiefel 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, 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.

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:

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

定理 4.13 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.

证明. 首先证明**必要性**. 设 x_* 是 $Ax=b$ 的解, 即 $Ax_*=b$, 并注意到 A 对称正定, 则

$$\begin{aligned}\Phi(x)-\Phi(x_*)&=\frac{1}{2}x^{\mathsf{T}}Ax-b^{\mathsf{T}}x-\frac{1}{2}x_*{}^{\mathsf{T}}Ax_*+b^{\mathsf{T}}x_*\\&=\frac{1}{2}x^{\mathsf{T}}Ax-x_*{}^{\mathsf{T}}Ax-\frac{1}{2}x_*{}^{\mathsf{T}}Ax_*+x_*{}^{\mathsf{T}}Ax_*\\&=\frac{1}{2}(x-x_*)^{\mathsf{T}}A(x-x_*)\geq 0,\end{aligned}$$

等号当且仅当 $x-x_*=0$ 时成立. 因此 x_* 是 $\Phi(x)$ 的唯一最小值点.

下面证明**充分性**. 设 x_* 是 $\Phi(x)$ 的最小值点, 则

$$\begin{aligned}\Phi(x)-\Phi(x_*)&=\frac{1}{2}(x-x_*)^{\mathsf{T}}A(x-x_*)+x^{\mathsf{T}}Ax_*-x_*{}^{\mathsf{T}}Ax_*-b^{\mathsf{T}}x+b^{\mathsf{T}}x_*\\&=\frac{1}{2}(x-x_*)^{\mathsf{T}}A(x-x_*)-(x-x_*)^{\mathsf{T}}(b-Ax_*).\end{aligned}$$

记 $r \triangleq b - Ax_*$, 如果 $r \neq 0$, 则取 $x = x_* + \alpha r$, 其中 $\alpha > 0$, 于是

$$\Phi(x) - \Phi(x_*) = \frac{\alpha^2}{2} r^{\mathsf{T}} A r - \alpha \|r\|_2.$$

当 α 足够小时, 上式右端小于 0, 这与 x_* 是 $\Phi(x)$ 的最小值点矛盾. 所以 $r = 0$, 即 $Ax_* = b$. □

✍ 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

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

由于 A 对称正定, 直接计算可得

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

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

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

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方向, 因此也称为**负梯度方向法**.

✍ 实际计算时, 我们无需对下降方向 p 进行单位化.

最速下降法的一般格式可表示为

$$x^{(k)} = x^{(k-1)} + \alpha_k p_k, \quad k = 1, 2, \ldots,$$

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

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 4.14 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 $\|x\|_A \triangleq \sqrt{x^{\mathsf{T}} A x}$.

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 $x^{(0)} = \begin{bmatrix} -0.5 \\ 0 \end{bmatrix}$.

解. 计算结果如下: (Iter_steepest_descent_01.m)

k	x	relres
1	[0.94896898, 1.06454864]	3.54e-02
2	[0.99757851, 0.99838567]	1.61e-03
3	[0.99991762, 1.00010420]	5.71e-05
4	[0.99999609, 0.99999739]	2.61e-06
5	[0.99999987, 1.00000017]	9.21e-08

表格中 "k" 表示迭代步数, "relres" 表示相对残量

$$\text{relres} = \frac{\|b - Ax^{(k)}\|_2}{\|b - Ax^{(0)}\|_2}.$$

□

由步长的选取方法可知, 最速下降法的迭代解具有下面的最优性

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

✍ 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.

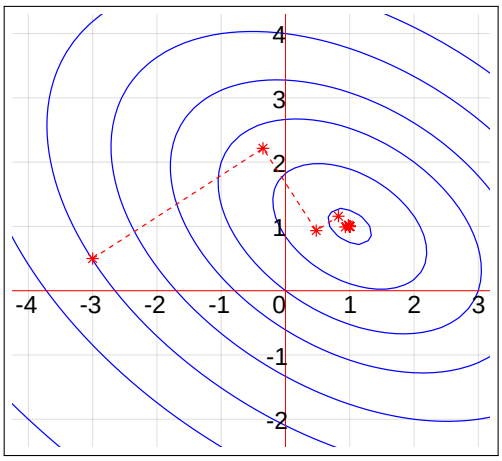
4.4.2 共轭梯度法

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 $p_1, p_2, \ldots$ 线性无关, 这意味着迭代过程可能会出现 "之" 字型, 即同一个下降方向可能会多次出现, 这就会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)

解. 计算结果如下: (Iter_steepest_descent_02.m)

k	x	relres
1	$[-0.3498, 2.2148]$	2.70e-01
2	$[0.4784, 0.9348]$	1.30e-01
3	$[0.8240, 1.1584]$	3.52e-02
4	$[0.9320, 0.9915]$	1.70e-02
5	$[0.9771, 1.0207]$	4.59e-03
6	$[0.9911, 0.9989]$	2.22e-03
⋮	⋮	⋮
14	$[1.0000, 1.0000]$	6.41e-07



计算结果显示, 迭代步数明显增加, 右图的收敛曲线呈 "之" 字型. □

✍ 如果初值取 $x^{(0)} = \begin{bmatrix} 1 \\ 1 \end{bmatrix} + \begin{bmatrix} -2.552 \\ 1.577 \end{bmatrix}$, 则计算结果如何? 为什么?

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. 我们首先给出共轭的定义.

定义 4.7 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.

✍ 若 $A\in\mathbb{R}^{n\times n}$ 对称正定, 则

$$f(x,y)\triangleq y^{\mathsf{T}}Ax$$

是 $\mathbb{R}^n$ 上的内积.

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

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

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.

✍ 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.

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

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

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 4.15 设 $x^{(k)}$ 是由共轭方向法得到的迭代解, 则

$$x^{(k)}=\operatorname*{argmin}_{x\in x^{(0)}+\operatorname{span}\{p_1,p_2,\dots,p_k\}}\Phi(x).$$

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 4.16 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 次迭代后, 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 $p_1, p_2, \dots, p_k, \dots$. 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 $p_1, p_2, \dots, p_k$ 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 $p_1, p_2, \dots, p_{k-1}$ 都 A-共轭 (见定理 4.17).

于是共轭梯度法可描述为

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

其中 $p_1 = r_0$.

定理 4.17 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) $r_k^{\mathsf{T}} r_i = 0, \quad i = 0, 1, 2, \dots, k-1;$
(2) $r_k^{\mathsf{T}} p_i = 0, \quad i = 1, 2, \dots, k;$
(3) $p_{k+1}^{\mathsf{T}} A p_i = 0, \quad i = 1, 2, \dots, k;$
(4) $\operatorname{span}\{r_0, r_1, \dots, r_k\} = \operatorname{span}\{p_1, p_2, \dots, p_{k+1}\} = \operatorname{span}\{r_0, Ar_0, \dots, A^k r_0\}.$

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 4.18 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

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

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 $x^{(k)} = x^{(k-1)} + \alpha_k p_k$, 其中 $\alpha_k = \dfrac{r_{k-1}^{\mathsf{T}} r_{k-1}}{p_k^{\mathsf{T}} A p_k}$
5: 计算 $r_k = r_{k-1} - \alpha_k A p_k$
6: 计算 $p_{k+1} = r_k + \beta_k p_k$, 其中 $\beta_k = \dfrac{r_k^{\mathsf{T}} r_k}{r_{k-1}^{\mathsf{T}} r_{k-1}}$
7: **end while**

对于共轭梯度法, 我们有下面的收敛性质.

定理 4.19 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

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

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

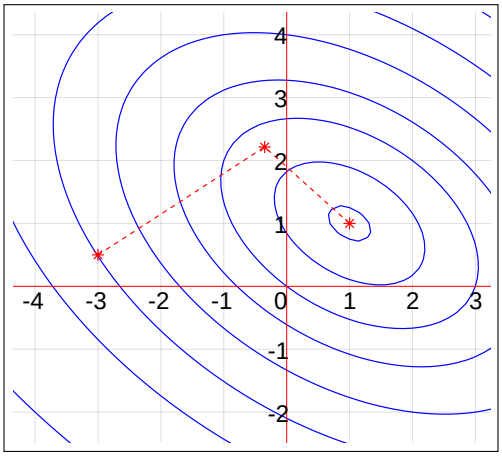
由此可知, 共轭梯度法每次迭代的误差下降率为 $\dfrac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}$.

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 $x^{(0)} = \begin{bmatrix} -3 \\ 0.5 \end{bmatrix}$.

解. 计算结果如下: (Iter_CG_01.m)

k	x	relres
1	$[-0.3498, 2.2148]$	2.70e-01
2	$[1.0000, 1.0000]$	4.39e-17

计算结果显示, 共轭梯度法迭代 2 步就收敛, 收敛曲线见下图.



□

✍ 由于共轭梯度法和最速下降法的第一个下降方向是一样的, 所以第一次迭代后的解也一样.

例 4.11 用共轭梯度法求解离散二维 Poission 方程.

(Iter_CG_gallery.m) (Iter_CG_Jacobi_GS_SOR_Poisson2D.m)

4.5 课后练习

练习 4.1 已知线性方程组 $\begin{bmatrix} 3 & 1 \\ 1 & 3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 3 \\ -1 \end{bmatrix}$, 迭代法

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

试问: α 取何值时迭代收敛? 何时收敛最快?

练习 4.2 设线性方程组 $Ax = b$, 其中 A 对称正定且 $0 < \alpha \leq \lambda(A) \leq \beta$, 迭代法

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

试证明: 当 $0 < \omega < \frac{2}{\beta}$ 时, 迭代法收敛.

练习 4.3* 设 $\|\cdot\|_\alpha$ 是定义在 $\mathbb{R}^{n\times n}$ 上的矩阵范数, 试证明: 存在 $\mathbb{R}^n$ 上的向量范数 $\|\cdot\|_\beta$ 使得

$$\|Ax\|_\beta \leq \|A\|_\alpha \|x\|_\beta, \quad \forall\, A \in \mathbb{R}^{n\times n},\ x \in \mathbb{R}^n.$$

练习 4.4* 设 $A \in \mathbb{R}^{n\times n}$, 常数 $\alpha > 0$. 记 $B = (\alpha I + A)^{-1}(\alpha I - A)$, 证明: 若 A 对称正定, 则 $\rho(B) < 1$. (思考: 若 A 仅仅是正定的, 则结论 $\rho(B) < 1$ 是否成立?)

练习 4.5 已知线性方程组

$$(1)\begin{cases} 5x_1 + 2x_2 + x_3 = 3, \\ -2x_1 + 4x_2 + 2x_3 = 7, \\ 3x_1 - 5x_2 + 8x_3 = 2. \end{cases} \qquad (2)\begin{cases} x_1 + 0.4x_2 + 0.5x_3 = 2, \\ 0.4x_1 + x_2 + 0.8x_3 = 1, \\ 0.5x_1 + 0.8x_2 + x_3 = 3. \end{cases}$$

考察 Jacobi 迭代法和 Gauss-Seidel 迭代法的收敛性.

练习 4.6 已知线性方程组

$$\begin{cases} 2x_1 + 4x_2 - 3x_3 = 3, \\ x_1 + 2x_2 + x_3 = 1, \\ 2x_1 + 2x_2 + 2x_3 = 2. \end{cases}$$

考察 Jacobi 迭代法和 Gauss-Seidel 迭代法的收敛性.

练习 4.7 设线性方程组

$$\begin{cases} a_{11}x_1 + a_{12}x_2 = b_1, \\ a_{21}x_1 + a_{22}x_2 = b_2, \end{cases}$$

其中 $a_{11}a_{22} \neq 0$. 证明: 求解该线性方程组的 Jacobi 迭代法和 Gauss-Seidel 迭代法具有相同的收敛性, 并求两种方法的收敛速度之比.

练习 4.8 已知线性方程组 $Ax = b$, 其中

$$A = \begin{bmatrix} 8 & \alpha & 0 \\ \beta & 8 & \beta \\ 0 & \alpha & 4 \end{bmatrix}$$

非奇异, 试给出 Jacobi 迭代法和 Gauss-Seidel 迭代法收敛的充要条件.

练习 4.9 已知线性方程组:

$$\begin{bmatrix} 5 & 2 & 2 \\ 2 & 5 & 4 \\ 2 & 4 & 5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 2 \\ 6 \\ 7 \end{bmatrix}.$$

写出求解该线性方程组的 Jacobi, Gauss-Seidel 和 SOR 迭代法的分量格式, 并判断这三个算法的收敛性.

练习 4.10 设 $A \in \mathbb{R}^{n\times n}$ 对称正定. 证明: 最速下降法的下降方向是 "之" 字型的, 即 $p_{k+1}^{\mathsf{T}} p_k = 0$, $k = 1, 2, \ldots$. (思考: 下降方向是否相互正交 (即 $p_i^{\mathsf{T}} p_j = 0, i \neq j$)?

练习 4.11* 设 $A \in \mathbb{R}^{n\times n}$ 对称正定. 试证明: 在最速下降法中, 如果初值取为 $x^{(0)} = x_* + q$, 其中 x_* 是真解, q 是 A 的一个特征向量, 则 $x^{(1)} = x_*$.

练习 4.12*(推论 4.18) 证明: 共轭梯度法 (4.13) 中的 α_k 和 β_k 满足

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

练习 4.13* 能否构造一类带参数的迭代法, 使得 Jacobi, G-S 和 SOR 都是该迭代法的特例?

练习 4.14* 初值的选取对迭代法的收敛速度也有着很大的影响, 请以 Jacobi 迭代法为例, 初值取什么时收敛最快 (真解除外)? 什么初值收敛最慢?

5 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课题.

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): 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.

最后, 我们将介绍最小二乘问题在函数逼近、数据拟合、信号恢复、图像处理等方面的应用.

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**.

5.1 问题介绍

线性最小二乘问题就是求解下面的最值问题:

$$\min_{x\in\mathbb{R}^n} \|Ax-b\|_2^2, \tag{5.1}$$

其中 $A\in\mathbb{R}^{m\times n}$, $b\in\mathbb{R}^m$. 问题 (5.1) 的解称为**最小二乘解**.

- 当 $m>n$ 时, 约束个数大于未知量个数, 此时我们称问题 (5.1) 为**超定**的;
- 当 $m<n$ 时, 未知量个数大于约束个数, 此时我们称问题 (5.1) 为**欠定** (或亚定) 的.

我们这里主要考虑超定情形. 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 (5.1) 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, 这就是最小二乘问题 (5.1) 对应的正规方程.

✍ 如果 A 不是满秩, 则 $A^{\mathsf{T}}A$ 半正定, 此时解不唯一. 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.

5.2 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**, 比如线性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.

5.2.1 Householder 变换

定义 5.1 我们称矩阵

$$H = I - \frac{2}{v^*v}vv^* = I - \frac{2}{\|v\|_2^2}vv^*, \quad 0 \neq v \in \mathbb{C}^n, \tag{5.2}$$

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 (5.2) 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.

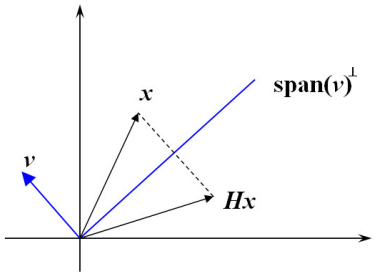
从几何上看, 一个 Householder 变换是一个关于超平面 $\mathrm{span}\{v\}^\perp$ 的反射. 由于 $\mathbb{C}^2 = \mathrm{span}\{v\} \oplus \mathrm{span}\{v\}^\perp$, 因此对任意一个向量 $x \in \mathbb{C}^n$, 都可写成

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

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 5.1 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.

定理 5.2 设 $x=[x_1,x_2,\dots,x_n]^{\mathsf{T}}\in\mathbb{R}^n$ 是非零向量, 则存在 Householder 矩阵 $H(v)$ 使得 $H(v)x=\alpha e_1$, 其中 $\alpha=\|x\|_2$ (或 $-\|x\|_2$), $e_1=[1,0,\dots,0]^{\mathsf{T}}\in\mathbb{R}^n$. (取 $v=x-\alpha e_1$ 即可)

✍ 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 5.2 中 Householder 矩阵 $H(v)$ 所对应的 Householder 向量 v. 由引理 ?? 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

$$\alpha=-\operatorname{sign}(x_1)\cdot\|x\|_2. \tag{5.3}$$

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

$$\alpha=\operatorname{sign}(x_1)\|x\|_2,\quad v_1=x_1-\alpha=\frac{x_1^2-\|x\|_2^2}{x_1+\alpha}=\frac{-(x_2^2+x_3^2+\cdots+x_n^2)}{x_1+\alpha}. \tag{5.4}$$

在 v_1 的两种计算方法 (5.3) 和 (5.4) 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

$$\beta=\frac{2}{v^*v}=\frac{2}{(x_1-\alpha)^2+x_2^2+\cdots+x_n^2}=\frac{2}{2\alpha^2-2\alpha x_1}=-\frac{1}{\alpha v_1}.$$

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 5.1. 计算 Householder 向量

% Given $x\in\mathbb{R}^n$, compute $v\in\mathbb{R}^n$ such that $Hx=\|x\|_2e_1$, where $H=I-\beta vv^*$

function $[\beta,v]$ = **House**(x)
$n=$ `length`(x) (here `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

11: **else**
12: $\alpha = \sqrt{x_1^2 + \sigma}$ % $\alpha = \|x\|_2$
13: **if** $x_1 < 0$ **then**
14: $v_1 = x_1 - \alpha$
15: **else**
16: $v_1 = -\sigma/(x_1 + \alpha)$
17: **end if**
18: $\beta = 2/(v_1^2 + \sigma)$
19: **end if**

稳定性

可以证明, 上述算法具有很好的数值稳定性, 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.

5.2.2 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.

易知, 交换 $G(i,j,\theta)$ 的第 1 行与第 i 行, 再交换第 1 列与第 i 列, 然后交换第 2 行与第 j 行, 交换第 2 列与第 j 列, 则转化为 $\begin{bmatrix} c & s & \\ -s & c & \\ & & I_{n-2} \end{bmatrix}$.

定理 5.3 $G(i,j,\theta)$ 是正交矩阵, 且 $\det(G(i,j,\theta)) = 1$.

✍ 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.
✍ 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 5.1 (Givens 变换化零) 设 $x = [x_1, x_2]^\mathsf{T} \in \mathbb{R}^2$, 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 $Gx = [r, 0]^\mathsf{T}$, 其中 c, s 和 r 的值如下:

$$c = \frac{x_1}{r}, \quad s = \frac{x_2}{r}, \quad r = \sqrt{x_1^2 + x_2^2},$$

即 $G = \frac{1}{r}\begin{bmatrix} x_1 & x_2 \\ -x_2 & x_1 \end{bmatrix}$

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 Givens 变换, 将 x 中除第一个分量外的所有元素都化为 0.

算法 5.2. 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 $r = \|x\|_2$

function $[c, s]$ = **givens**(a, b)
if $b = 0$ **then**
  if $a \geq 0$ **then**
    $c = 1, \quad s = 0$
  else
    $c = -1, \quad s = 0$
  end if
else
  if $|b| > |a|$ **then** % 绝对值大的数作为分母
    $\tau = \dfrac{a}{b}, \quad s = \dfrac{\operatorname{sign}(b)}{\sqrt{1+\tau^2}}, \quad c = s\tau$
  else
    $\tau = \dfrac{b}{a}, \quad c = \dfrac{\operatorname{sign}(a)}{\sqrt{1+\tau^2}}, \quad s = c\tau$
  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 5.1 和 5.2), 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.

5.2.3 正交变换的舍入误差分析

引理 5.4 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 5.5 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.

5.3 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.

5.3.1 QR 分解的存在性与唯一性

定理 5.6 (QR 分解) 设 $A \in \mathbb{R}^{m\times n}$ $(m \ge n)$. 则存在一个单位列正交矩阵 $Q \in \mathbb{R}^{m\times n}$ (即 $Q^{\mathsf{T}}Q = I_{n\times n}$) 和一个上三角矩阵 $R \in \mathbb{R}^{n\times n}$, 使得

$$A = QR. \tag{5.5}$$

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 (5.5) 成立, 且此时 QR 分解唯

一, 即 Q 和 R 都唯一.

证明. 设 $A=[a_1,a_2,\dots,a_n]\in\mathbb{C}^{m\times n}$. 若 A 列满秩, 即 $\operatorname{rank}(A)=n$. 则 QR 分解 (5.5) 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 5.3).

算法 5.3. 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 5.3 可知: $a_1=r_{11}q_1$,

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

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

$$r_{ij}=\begin{cases}q_i^*a_j, & \text{for } i\le j\\ 0, & \text{for } i>j\end{cases}\tag{5.6}$$

于是 Gram-Schmidt 过程可表示为

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

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 (5.6). 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 $\operatorname{rank}(A)=l<n$, 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 5.3) 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

$$Q_1=Q_2R_2R_1^{-1}. \tag{5.7}$$

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 $\|Q_1\|_F^2=\operatorname{tr}(Q_1^*Q_1)=n$, 所以由 (5.7) 可知

$$\|R_2R_1^{-1}\|_F^2=\|Q_2R_2R_1^{-1}\|_F^2=\|Q_1\|_F^2=n.$$

由于 $R_2R_1^{-1}$ 的对角线元素都是 1, 所以 $R_2R_1^{-1}$ 只能是单位矩阵, 即 $R_2=R_1$. 因此 $Q_2=AR_2^{-1}=AR_1^{-1}=Q_1$, 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 $l=\operatorname{rank}(A)$. 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 5.7 (列主元 QR 分解) 设 $A\in\mathbb{C}^{m\times n}$ $(m\geq n)$, 且秩为 l $(0\leq l\leq n)$, 则存在置换矩阵 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 5.7 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 5.3 的运算量大约为 $2mn^2$.

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.

5.3.2 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 5.4. 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 $2mn^2$.

5.3.3 基于 Householder 变换的 QR 分解

由定理 5.2 可知, 通过 Householder 变换, 我们可以将任何一个非零变量 $x \in \mathbb{R}^n$ 转化成 $\|x\|_2 e_1$, 即除第一个元素外, 其它都为零. 下面我们就考虑通过 Householder 变换来实现矩阵的 QR 分解.

我们首先考虑 $m = n$ 时的情形. 设矩阵 $A \in \mathbb{R}^{n\times n}$, 令 $H_1 \in \mathbb{R}^{n\times n}$ 为一个 Householder 变换, 满足

$$H_1 \begin{bmatrix} a_{11} \\ a_{21} \\ \vdots \\ a_{n1} \end{bmatrix} = \begin{bmatrix} r_1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}.$$

于是

$$H_1 A = \left[\begin{array}{c|ccc} r_1 & \tilde{a}_{12} & \cdots & \tilde{a}_{1n} \\ \hline 0 & & & \\ \vdots & & \tilde{A}_2 & \\ 0 & & & \end{array}\right],$$

其中 $\tilde{A}_2 \in \mathbb{R}^{(n-1)\times(n-1)}$. 同样地, 我们可以构造一个 Householder 变换 $\tilde{H}_2 \in \mathbb{R}^{(n-1)\times(n-1)}$, 将 $\tilde{A}_2$ 的第一列中除第一个元素外的所有元素都化为 0, 即

$$\tilde{H}_2 \tilde{A}_2 = \left[\begin{array}{c|ccc} r_2 & \tilde{a}_{23} & \cdots & \tilde{a}_{2n} \\ \hline 0 & & & \\ \vdots & & \tilde{A}_3 & \\ 0 & & & \end{array}\right].$$

令

$$H_2 = \begin{bmatrix} 1 & 0 \\ 0 & \tilde{H}_2 \end{bmatrix}.$$

则 $H_2 \in \mathbb{R}^{n\times n}$, 且

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

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 5.5. 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.

5.3.4 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.

5.3.5 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.

Björck 证明了, 通过 MGS 计算的矩阵 Q 满足

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

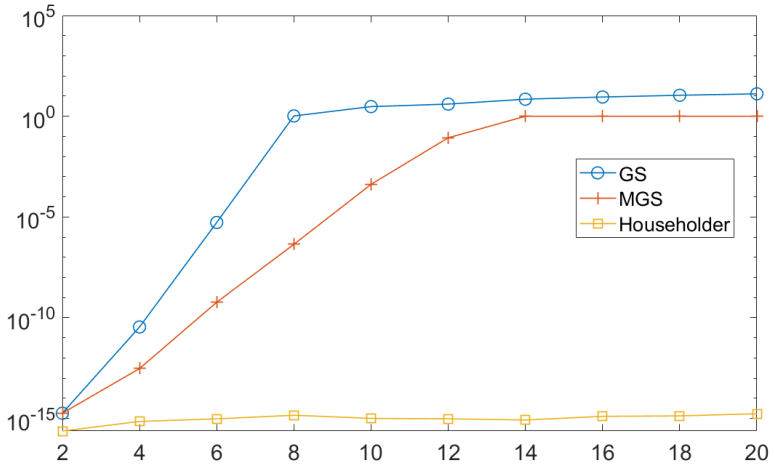
而 Householder 变换计算的矩阵 Q 满足

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

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 5.2 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)

	GS		MGS		Householder	
n	$\|\tilde{Q}\tilde{R} - H\|_2$	$\|\tilde{Q}^{\mathsf{T}}\tilde{Q} - I\|_2$	$\|\tilde{Q}\tilde{R} - H\|_2$	$\|\tilde{Q}^{\mathsf{T}}\tilde{Q} - I\|_2$	$\|\tilde{Q}\tilde{R} - H\|_2$	$\|\tilde{Q}^{\mathsf{T}}\tilde{Q} - I\|_2$
2	0.00e+00	1.81e-15	0.00e+00	1.81e-15	1.24e-16	2.36e-16
4	3.93e-17	3.45e-11	5.55e-17	2.98e-13	2.46e-16	7.08e-16
6	6.21e-17	5.33e-06	2.78e-17	5.81e-10	1.49e-16	9.49e-16
8	7.48e-17	1.03e+00	6.59e-17	4.38e-07	2.57e-16	1.44e-15
10	6.52e-17	3.00e+00	7.49e-17	4.16e-04	6.36e-16	1.00e-15
12	7.87e-17	4.00e+00	7.55e-17	8.52e-02	4.68e-16	9.52e-16
14	6.54e-17	7.00e+00	8.54e-17	9.96e-01	5.71e-16	8.35e-16



5.4 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.

5.4.1 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 5.8 (SVD) 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

$$U^*AV = \begin{bmatrix} \Sigma \\ 0 \end{bmatrix} \quad \text{或} \quad A = U \begin{bmatrix} \Sigma \\ 0 \end{bmatrix} V^*, \tag{5.8}$$

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 (5.8) 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**.

✍ 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.

由 (5.8) 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 $U=[u_1,u_2,\dots,u_m]$ 和 $V=[v_1,v_2,\dots,v_n]$ 的列向量分别称为 A 的**左奇异向量**和**右奇异向量**, 即存在关系式

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

由定理 5.8 可知

$$A=U\begin{bmatrix}\Sigma\\0\end{bmatrix}V^*=\sigma_1u_1v_1^*+\sigma_2u_2v_2^*+\cdots+\sigma_nu_nv_n^*=\sum_{i=1}^{n}\sigma_iu_iv_i^*.$$

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

$$A=U_n\Sigma V^*. \tag{5.9}$$

这就是所谓的**细 SVD** (**thin SVD**) 或**降阶 SVD** (**reduced SVD**), 有的文献将 (5.9) 称为奇异值分解.

设 $k<n$, 我们称

$$A_k=\sigma_1u_1v_1^*+\sigma_2u_2v_2^*+\cdots+\sigma_ku_kv_k^*=\sum_{i=1}^{k}\sigma_iu_iv_i^*$$

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

$$A=A_r=\sum_{i=1}^{r}\sigma_iu_iv_i^*.$$

奇异值的应用

- 计算矩阵范数, 条件数, 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, ...

5.4.2 奇异值的性质与应用

定理 5.9 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, $i=1,2,\dots,n$;

(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, $i=1,2,\dots,m$;

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

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

定理 5.10 设 A 非奇异, 则 $\kappa_2(A)=\sigma_1/\sigma_n$.

SVD 的一个重要应用是计算矩阵的低秩逼近.

定理 5.11 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

$$\min_{B\in\mathbb{C}^{m\times n},\ \mathrm{rank}(B)=k} \|A-B\|_2 \tag{5.10}$$

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**.

✍ 定理中的 (5.10) 式也可以改写为

$$\min_{B\in\mathbb{C}^{m\times n},\ \mathrm{rank}(B)\le k} \|A-B\|_2 \tag{5.11}$$

5.5 最小二乘问题的求解方法

5.5.1 正规方程

定理 5.12 设 $A \in \mathbb{R}^{m\times n}$ $(m \ge n)$, 则 $x_* \in \mathbb{R}^n$ 是线性最小二乘问题 (5.1) 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

$$A^{\mathsf{T}}(b - Ax) = 0 \quad 或 \quad A^{\mathsf{T}}Ax = A^{\mathsf{T}}b. \tag{5.12}$$

证明. 充分性: 设 x_* 是正规方程 (5.12) 的解. 对任意向量 $y \in \mathbb{R}^n$, 由 $(b - Ax_*) \perp \mathrm{Ran}(A)$ 可知

$$\begin{aligned}\|Ay - b\|_2^2 &= \|(Ax_* - b) + A(y - x_*)\|_2^2\\ &= \|Ax_* - b\|_2^2 + \|A(y - x_*)\|_2^2\\ &\ge \|Ax_* - b\|_2^2.\end{aligned}$$

因此, x_* 是线性最小二乘问题 (5.1) 的解.

必要性: 设 x_* 是线性最小二乘问题 (5.1) 的解. 用反证法, 假定 $z \triangleq A^{\mathsf{T}}(b - Ax_*) \ne 0$. 取 $y = x_* + \alpha z$, 其中 $\alpha > 0$, 则有

$$\|Ay - b\|_2^2 = \|Ax_* - b + \alpha Az\|_2^2 = \|Ax_* - b\|_2^2 - 2\alpha\|z\|_2^2 + \alpha^2\|Az\|_2^2.$$

由于 $\|z\|_2 > 0$, 当 α 充分小时, 有 $2\|z\|_2^2 > \alpha\|Az\|_2^2$, 即上式右端小于 $\|Ax_* - b\|_2^2$. 这与 x_* 是最小二乘解相矛盾. 所以 $z = 0$, 即 $A^{\mathsf{T}}(b - Ax_*) = 0$. □

由定理 5.12 可知, 求线性最小二乘问题 (5.1) 的解等价于求正规方程 (5.12) 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 5.13 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 $\operatorname{rank}(A) = n$. 此时, 线性最小二乘问题 (5.1) 的解是唯一的, 其表达式为

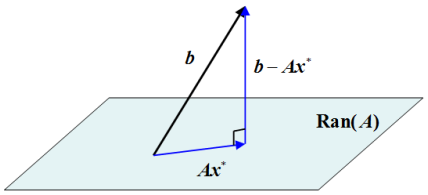
$$x_* = (A^{\mathsf{T}}A)^{-1}A^{\mathsf{T}}b.$$

最小二乘解的几何含义

根据定理 5.12, 我们可以把 b 写成

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

所以 Ax_* 就是 b 在 $\operatorname{Ran}(A)$ 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.

5.5.2 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 5.3 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.

5.5.3 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 5.12 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.

5.5.4 奇异值分解法

设 $A \in \mathbb{R}^{m\times n}$ 列满秩, $A = U_n\Sigma V^{\mathsf{T}}$ 是 A 的降阶奇异值分解, 则

$$x_* = (A^{\mathsf{T}}A)^{-1}A^{\mathsf{T}}b = \left(V\Sigma U_n^{\mathsf{T}}U_n\Sigma V^{\mathsf{T}}\right)^{-1}V\Sigma U_n^{\mathsf{T}}b = (V\Sigma^{-2}V^{\mathsf{T}})V\Sigma U_n^{\mathsf{T}}b = V\Sigma^{-1}U_n^{\mathsf{T}}b.$$

✍ 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 5.4 分别用三种方法求解最小二乘问题, 比较运算时间. (LS_3methods.m)

三种方法的运算时间如下 ($A \in \mathbb{R}^{2n\times n}, b \in \mathbb{R}^n$, 以秒为单位):

n	正规方程法	QR 分解法	奇异值分解法
500	0.0050	0.0220	0.0370
1000	0.0160	0.0340	0.1440
1500	0.0490	0.1330	0.5530
2000	0.0870	0.2070	1.4840
2500	0.1910	0.4430	3.1160
3000	0.2500	0.6950	5.9600
3500	0.4640	1.2130	10.0500
4000	0.4940	1.5700	14.8750
4500	0.6690	2.1680	20.6410
5000	1.0720	2.9350	28.6360

✍ 这里结果可能受计算机系统和软件优化影响, 并不一定能准确反映各种方法的实际运算量.

5.6 数据拟合

数据拟合, 也称 **曲线拟合**, 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

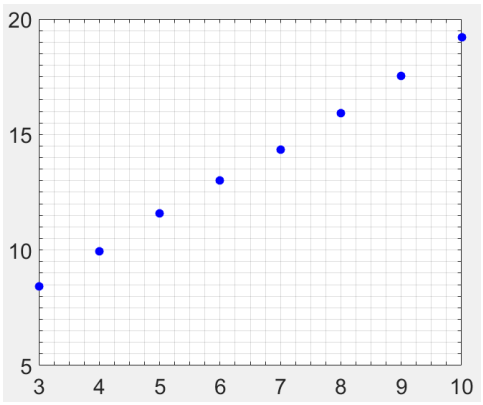
数据, 如果能够找到一个连续的函数（即曲线）或者更加密集的离散方程, 使得实验数据与方程的曲线能够在最大程度上近似吻合, 就可以根据曲线方程对数据进行理论分析和数值预测, 对某些不具备测量条件的位置的结果进行估算.

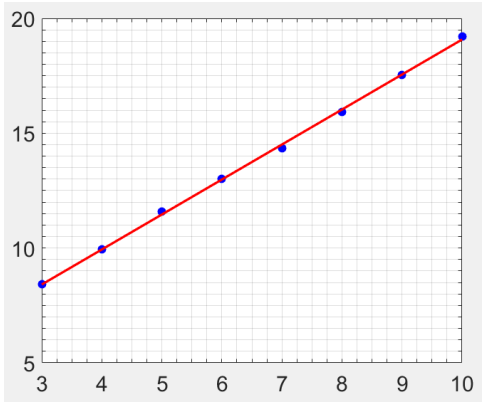
我们首先看一个简单的线性数据拟合问题.

例 5.5 回想一下中学物理课的 “速度与加速度” 实验: 假设某物体正在做加速运动, 加速度未知, 实验人员从时间 $t_0 = 3$ 秒时刻开始, 每隔 1 秒时间对这个物体进行测速, 得到一组速度和时间的离散数据 (见下表). 请根据实验数据推算该物体的加速度.

时间 t (秒)	3	4	5	6	7	8	9	10
速度 v (米/秒)	8.41	9.94	11.58	13.02	14.33	15.92	17.54	19.22

解. **实验法:** 在坐标纸中画出这些点, 如下图 (左图) 所示.





可以看出, 测量结果呈现典型的线性特征. 沿着该线性特征画一条直线, 使尽量多的测量点能够位于直线上, 或者与直线的偏差尽量小, 见上图 (右图). 这条直线就是我们根据测量结果拟合的速度与时间的函数关系. 最后测量出直线的斜率 k, 它就是被测物体的加速度. 经过测量, 我们实验测到的物体加速度值约为 1.52 米/秒 ^{2}.

数学方法: 设加速度为 a (米/秒 ^{2}), 则速度 v 与时间 t 之间的关系式为

$$v = v_0 + a(t - t_0).$$

其中 $v_0 = 8.41$, $t_0 = 3$. 将实验数据 (t_i, v_i) 依次代入可得

$$\begin{cases} 9.94 = 8.41 + a \\ 11.58 = 8.41 + 2a \\ 13.02 = 8.41 + 3a \\ 14.33 = 8.41 + 4a \\ 15.92 = 8.41 + 5a \\ 17.54 = 8.41 + 6a \\ 19.22 = 8.41 + 7a \end{cases}$$

显然, 这个方程组是无解的.

事实上, 由于实验存在误差, 上面的每个方程并不需要严格成立, 因此我们只需要求偏差尽可能地小即可. 也就是说, 使得偏差平方和尽可能地小, 即转化为下面的最小化问题

$$\min_{a\in\mathbb{R}} \sum_{i=1}^{7} |v_i - v_0 - a(t_i - t_0)|^2.$$

这是一个最小二乘问题, 这就是 **数据拟合的最小二乘法**. □

数据拟合

如果只知道函数在部分点上的值 (即数据), 且这些数据带有一定的误差, 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 **数据拟合**, 也称为 **曲线拟合**.

5.6.1 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

$$\sum_{i=0}^{m} |S^*(x_i) - y_i|^2 = \min_{S(x)\in\Phi} \sum_{i=0}^{m} |S(x_i) - y_i|^2. \tag{5.13}$$

这里的 n 通常远远小于 m, 即 $n \ll m$. 因此, 曲线拟合问题就转化为求解一个最小二乘问题, 这就是 **数据拟合的最小二乘法**.

✍ 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.

下面考虑问题 (5.13) 的求解. 对任意 $S(x) \in \Phi = \operatorname{span}\{\varphi_0(x), \varphi_1(x), \ldots, \varphi_n(x)\}$, 可设

$$S(x) = \alpha_0\varphi_0(x) + \alpha_1\varphi_1(x) + \cdots + \alpha_n\varphi_n(x),$$

于是原问题就转化为求下面的多元函数的最小值点:

$$I(\alpha_0, \alpha_1, \ldots, \alpha_n) \triangleq \sum_{i=0}^{m} |S(x_i) - y_i|^2 = \sum_{i=0}^{m} \left[\sum_{j=0}^{n} \alpha_j\varphi_j(x_i) - y_i\right]^2.$$

由于 $I(\alpha_0,\alpha_1,\dots,\alpha_n)$ 是正定的, 因此其最小值点就是其驻点. 令偏导数为零, 可得

$$0=\frac{\partial I(\alpha_0,\alpha_1,\dots,\alpha_n)}{\partial \alpha_k}=2\sum_{i=0}^{m}\varphi_k(x_i)\left[\sum_{j=0}^{n}\alpha_j\varphi_j(x_i)-y_i\right],\quad k=0,1,2,\dots,n.$$

整理后可写为

$$\sum_{j=0}^{n}\left[\sum_{i=0}^{m}\varphi_k(x_i)\varphi_j(x_i)\right]\alpha_j=\sum_{i=0}^{m}y_i\varphi_k(x_i),\quad k=0,1,2,\dots,n.$$

引入记号

$$(\varphi_j,\varphi_k)\triangleq\sum_{i=0}^{m}\varphi_j(x_i)\varphi_k(x_i),\quad (y,\varphi_k)\triangleq\sum_{i=0}^{m}y_i\varphi_k(x_i),\tag{5.14}$$

则上面的方程可简写为

$$\sum_{j=0}^{n}(\varphi_j,\varphi_k)\alpha_j=(y,\varphi_k),\quad k=0,1,2,\dots,n.$$

写成矩阵形式即可得 **法方程**:

$$G\alpha=d,\tag{5.15}$$

其中

$$G=\begin{bmatrix}(\varphi_0,\varphi_0) & (\varphi_0,\varphi_1) & \cdots & (\varphi_0,\varphi_n)\\ (\varphi_1,\varphi_0) & (\varphi_1,\varphi_1) & \cdots & (\varphi_1,\varphi_n)\\ \vdots & \vdots & & \vdots\\ (\varphi_n,\varphi_0) & (\varphi_n,\varphi_1) & \cdots & (\varphi_n,\varphi_n)\end{bmatrix},\quad \alpha=\begin{bmatrix}\alpha_0\\ \alpha_1\\ \vdots\\ \alpha_n\end{bmatrix},\quad d=\begin{bmatrix}(y,\varphi_0)\\ (y,\varphi_1)\\ \vdots\\ (y,\varphi_n)\end{bmatrix}.$$

将法方程的解记为 $\alpha_0^*,\alpha_1^*,\alpha_2^*,\dots,\alpha_n^*$, 则最佳平方逼近函数为

$$S^*(x)=\alpha_0^*\varphi_0(x)+\alpha_1^*\varphi_1(x)+\cdots+\alpha_n^*\varphi_n(x).$$

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 (5.14) 中引入的记号 (φ_j,φ_k) 并不构成 $C[a,b]$ 或 Φ 中的内积.

定理 5.14 设 $\varphi_0(x),\varphi_1(x),\dots,\varphi_n(x)\in C[a,b]$ 线性无关. 如果 $\varphi_0(x),\varphi_1(x),\dots,\varphi_n(x)$ 的任意 (非零) 线性组合在点集 $\{x_0,x_1,\dots,x_m\}$ 上至多只有 n 个不同的零点, 则 G 非奇异, 此时法方程 (5.15) 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 5.6 已知数据表如下, 求 $f(x)$ 的最小二乘拟合函数 $S^*(x)$, 并给出拟合误差.

x_i	0.24	0.65	0.95	1.24	1.73	2.01	2.23	2.52	2.77	2.99
y_i	0.102	0.103	0.156	0.167	0.274	0.448	0.562	0.620	0.799	1.00

证明. 分两步走: 先确定一组基, 然后求解法方程. (Approxi_Datafit_01.m)

在坐标平面上描出数据点, 根据点的分布情况, 选取基函数

$$\varphi_0(x)=\ln x,\quad \varphi_1(x)=\cos x,\quad \varphi_2(x)=e^x.$$

所以 $n=2$. 又根据数据表可知 $m=8$, 直接计算可得

$$(\varphi_0,\varphi_0)=\sum_{i=0}^{m}\varphi_0^2(x_i)\approx 6.794,\quad (\varphi_0,\varphi_1)=(\varphi_1,\varphi_0)=\sum_{i=0}^{m}\varphi_0(x_i)\varphi_1(x_i)\approx -5.347,$$

$$(\varphi_0,\varphi_2)=(\varphi_2,\varphi_0)=\sum_{i=0}^{m}\varphi_0(x_i)\varphi_2(x_i)\approx 63.26,\quad (\varphi_1,\varphi_1)=\sum_{i=0}^{m}\varphi_1^2(x_i)\approx 5.108,$$

$$(\varphi_1,\varphi_2)=(\varphi_2,\varphi_1)=\sum_{i=0}^{m}\varphi_1(x_i)\varphi_2(x_i)\approx -49.01,\quad (\varphi_2,\varphi_2)=\sum_{i=0}^{m}\varphi_2^2(x_i)\approx 1003,$$

$$(y,\varphi_0)=\sum_{i=0}^{m}y_i\varphi_0(x_i)\approx 3.234,\quad (y,\varphi_1)=\sum_{i=0}^{m}y_i\varphi_1(x_i)\approx -2.489,$$

$$(y,\varphi_2)=\sum_{i=0}^{m}y_i\varphi_2(x_i)\approx 51.77.$$

所以法方程为

$$\begin{bmatrix} 6.794 & -5.347 & 63.26 \\ -5.347 & 5.108 & -49.01 \\ 63.26 & -49.01 & 1003 \end{bmatrix}\begin{bmatrix}\alpha_0\\ \alpha_1\\ \alpha_2\end{bmatrix}=\begin{bmatrix}3.234\\ -2.489\\ 51.77\end{bmatrix}.$$

解得

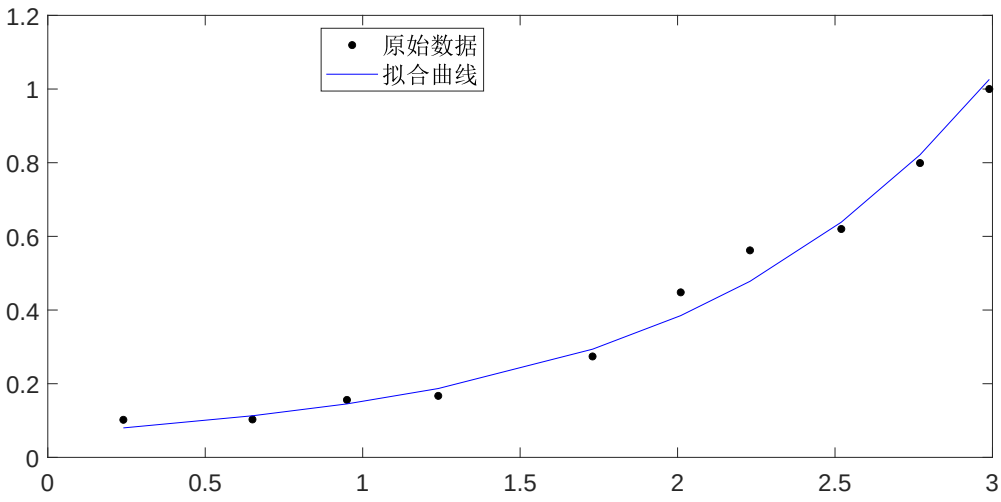
$$\alpha_0=0.00224,\quad \alpha_1=0.0172,\quad \alpha_2=0.0523.$$

所以最小二乘拟合函数为

$$S^*(x)=0.00224\ln x+0.0172\cos x+0.0523e^x.$$

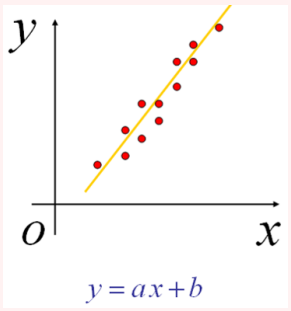
拟合误差为

$$\sum_{i=0}^{m}|S^*(x_i)-y_i|^2\approx 0.0141.$$

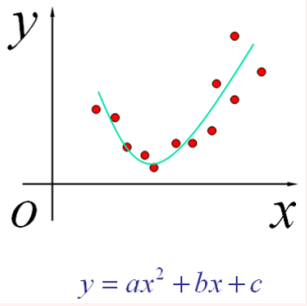


□

✍ 对于数据拟合问题, 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.



$y = ax + b$



$y = ax^2 + bx + c$

5.6.2 多项式数据拟合

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

$$\begin{bmatrix} \sum\limits_{i=0}^{m} \omega_i & \sum\limits_{i=0}^{m} \omega_i x_i & \cdots & \sum\limits_{i=0}^{m} \omega_i x_i^n \\ \sum\limits_{i=0}^{m} \omega_i x_i & \sum\limits_{i=0}^{m} \omega_i x_i^2 & \cdots & \sum\limits_{i=0}^{m} \omega_i x_i^{n+1} \\ \vdots & \vdots & & \vdots \\ \sum\limits_{i=0}^{m} \omega_i x_i^n & \sum\limits_{i=0}^{m} \omega_i x_i^{n+1} & \cdots & \sum\limits_{i=0}^{m} \omega_i x_i^{2n} \end{bmatrix} \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \vdots \\ \alpha_n \end{bmatrix} = \begin{bmatrix} \sum\limits_{i=0}^{m} \omega_i y_i \\ \sum\limits_{i=0}^{m} \omega_i x_i y_i \\ \cdots \\ \sum\limits_{i=0}^{m} \omega_i x_i^n y_i \end{bmatrix}.$$

设解为 $[\alpha_0^*, \alpha_1^*, \ldots, \alpha_n^*]^{\mathsf{T}}$, 则

$$S^*(x) = \alpha_0^* + \alpha_1^* x + \cdots + \alpha_n^* x^n$$

即为 $f(x)$ 的 **n 次最小二乘拟合多项式**.

例 5.7 已知数据表如下, 求 2 次最小二乘拟合多项式.

x_i	0	0.25	0.50	0.75	1.00
y_i	1.0000	1.2840	1.6487	2.1170	2.7183

证明. 设 2 次最小二乘拟合多项式为 $p_2(x) = \alpha_0 + \alpha_1 x + \alpha_2 x^2$. 直接计算得法方程

$$\begin{bmatrix} 5 & 2.5 & 1.875 \\ 2.5 & 1.875 & 1.5625 \\ 1.875 & 1.5625 & 1.3825 \end{bmatrix} \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \end{bmatrix} = \begin{bmatrix} 8.7680 \\ 5.4514 \\ 4.4015 \end{bmatrix}.$$

解得

$$\alpha_0 = 1.0052, \quad \alpha_1 = 0.8641, \quad \alpha_2 = 0.8437.$$

所以此组数据的 2 次最小二乘拟合多项式为

$$S_2^*(x) = 1.0052 + 0.8641x + 0.8437x^2.$$

□

✍ 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.

5.7 信号恢复与图像处理

5.7.1 信号去噪

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, 即

$$b = x + \eta,$$

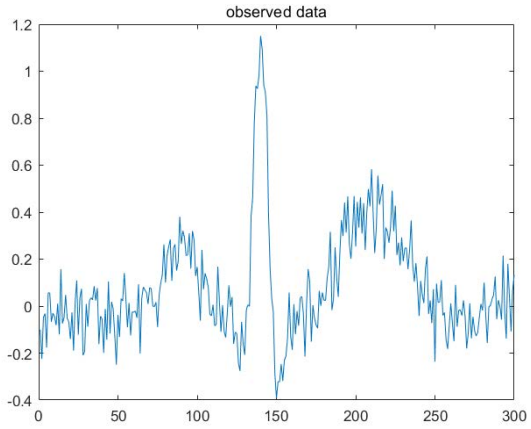
其中 x 是真实的信号, b 是观察到的信号, η 是噪声.

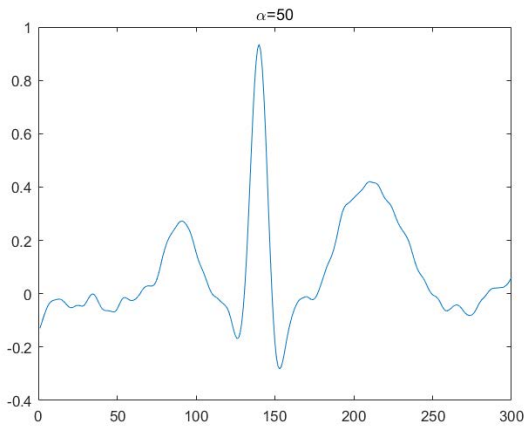
去噪 是数字信号和图像处理中的一个基本问题, 其中一个有效方法就是加权最小二乘法, 即

$$\min_x \frac{1}{2}\|x-b\|_2^2 + \frac{1}{2}\alpha\|Dx\|_2^2,$$

其中 D 是离散的二阶导算子或 TV 算子.

例 5.8 数字信号去噪. (LS_denoise.m)





5.7.2 图像恢复

除了带噪声以外, 在获取数字图像时也经常会由于各种原因 (设备仪器, 拍摄环境等) 导致图像模糊. 通常数字图像的获取模型为

$$f = \mathcal{B}(x) + \eta,$$

其中 x 是真实图像, f 是观察到的图像, $\mathcal{B}(\cdot)$ 是卷积算子 (代表模糊机制), η 是噪声.

由于问题本身是不适定的, 因此求解时需要进行正则化, 常用的模型有 **Tikhonov 正则化**模型:

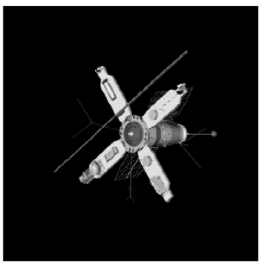
$$\min \|\mathcal{B}(x) - f\|_2^2 + \mu^2\|x\|_2^2$$

或**加权正则化**模型:

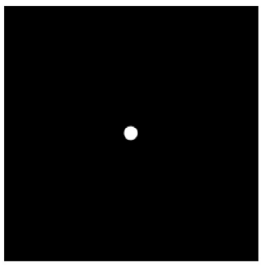
$$\min \|\mathcal{B}(x) - f\|_2^2 + \mu^2 \|Dx\|_2^2,$$

其中 D 是广义加权矩阵, 可以是非负对角矩阵或 TV 算子等.

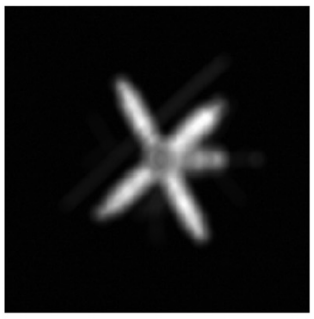
例 5.9 数字图像去噪和去模糊.

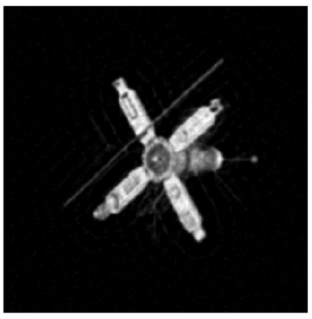


(a) 真实图像



(b) 模糊机制 (out-of-focus)





(a) 获取到的图像 (带有噪声和模糊)
(b) 恢复后的图像 (基于 Tikhonov 正则化模型)
(c) 恢复后的图像 (基于加权正则化模型)

5.8 课后习题

练习 5.1 设 $A \in \mathbb{R}^{n\times n}$ 是正交矩阵, 则 A 可表示成至多 n 个 Householder 变换的乘积.

练习 5.2 设 $A \in \mathbb{R}^{n\times n}$ 是正交矩阵, 则 A 可表示成至多 $\frac{1}{2}n(n-1)$ 个 Givens 变换的乘积.

练习 5.3 设 $x = [x_1, x_2, \ldots, x_n]^{\mathsf{T}} \in \mathbb{R}^n$ 是一个非零向量, H 是 Householder 矩阵, 满足 $Hx = \alpha e_1$.
证明: H 的第一列与 x 平行.

练习 5.4 设 $H_k \in \mathbb{R}^{k\times k}$ 是 Householder 变换, 其中 $k < n$.
证明: $H_n = \begin{bmatrix} I_{n-k} & 0 \\ 0 & H_k \end{bmatrix}$ 是 n 阶 Householder 变换.

练习 5.5 设 $R = \begin{bmatrix} r_{11} & r_{12} \\ 0 & r_{22} \end{bmatrix} \in \mathbb{R}^{2\times 2}$, 求一个 Givens 变换 G, 使得 $G^{\mathsf{T}}RG = \begin{bmatrix} r_{22} & r_{12} \\ 0 & r_{11} \end{bmatrix}$.

练习 5.6 (**极分解**) 设 $A \in \mathbb{C}^{n\times n}$. 证明:

(1) 存在酉矩阵 U 和唯一的 Hermitian 半正定矩阵 P, 使得 $A = PU$.

(2) 进一步, 若 A 非奇异, 则 U 也唯一.

练习 5.7 用 Householder 变换计算矩阵 $A = \begin{bmatrix} 1 & 1 & 1 \\ 2 & -1 & -1 \\ 2 & -4 & 10 \end{bmatrix}$ 的 QR 分解.

练习 5.8 设 $A \in \mathbb{R}^{n\times n}$ 是一个对角加边矩阵

$$A = \begin{bmatrix} a_1 & b_2 & b_3 & \cdots & b_n \\ c_2 & a_2 & & & \\ c_3 & & a_3 & & \\ \vdots & & & \ddots & \\ c_n & & & & a_n \end{bmatrix}.$$

试给出用 Givens 变换计算 A 的 QR 分解的详细算法, 使得运算量为 $\mathcal{O}(n^2)$.

练习 5.9 已知实验数据如下:

x_i	19	25	31	38	44
y_i	19.0	32.3	49.9	73.3	97.8

用最小二乘法求形如 $S(x) = a + bx^2$ 的拟合函数, 并计算拟合误差 $\sum_{i=0}^{m} |S(x_i) - y_i|^2$.

(计算结果保留 4 位有效数字)

练习 5.10* 设 $A \in \mathbb{R}^{m\times n}$, $W \in \mathbb{R}^{p\times n}$, 试给出矩阵 $A^{\mathsf{T}}A + W^{\mathsf{T}}W$ 非奇异的一个充要条件.

6 函数插值

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

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

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

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**, **三角插值**, ...

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

$$a \le x_0 < x_1 < \cdots < x_n \le b$$

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**

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

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:

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

解. 由于 $p(x)$ 是一次多项式, 即代表一条直线. 因此由点斜式可知

$$\begin{aligned} p(x) &= y_0+\frac{y_1-y_0}{x_1-x_0}(x-x_0)\\ &= y_0-\frac{y_0(x-x_0)}{x_1-x_0}+y_1\frac{x-x_0}{x_1-x_0}\\ &= y_0\frac{x-x_1}{x_0-x_1}+y_1\frac{x-x_0}{x_1-x_0}. \end{aligned}$$

记 $l_0(x)=\dfrac{x-x_1}{x_0-x_1}$, $l_1(x)=\dfrac{x-x_0}{x_1-x_0}$, 则 $p(x)$ 就可以表示成 $l_0(x)$ 和 $l_1(x)$ 的线性组合, 即

$$p(x)=y_0l_0(x)+y_1l_1(x).$$

我们进一步观察可知

$$\begin{aligned} l_0(x_0)&=1,\quad l_0(x_1)=0;\\ l_1(x_0)&=0,\quad l_1(x_1)=1. \end{aligned}$$

□

例 6.3 抛物线插值: 求一个二次多项式 $p(x)$, 满足:

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

解. 借鉴线性插值思想, 如果能构造出三个二次多项式 $l_0(x), l_1(x), l_2(x)$, 满足

$$\begin{aligned} &l_0(x_0)=1,\quad l_0(x_1)=0,\quad l_0(x_2)=0;\\ &l_1(x_0)=0,\quad l_1(x_1)=1,\quad l_1(x_2)=0;\\ &l_2(x_0)=0,\quad l_2(x_1)=0,\quad l_2(x_2)=1. \end{aligned}$$

则由插值条件可知, $p(x)$ 可以表示成

$$p(x)=y_0l_0(x)+y_1l_1(x)+y_2l_2(x).$$

现在的问题是如何构造 $l_0(x), l_1(x), l_2(x)$. 我们可以使用待定系数法.

由于 $l_0(x)$ 是二次多项式, 且满足 $l_0(x_1)=l_0(x_2)=0$, 因此 $l_0(x)$ 可以写成

$$l_0(x)=\alpha(x-x_1)(x-x_2),$$

其中 α 是待定系数 (常数). 将 $l_0(x_0)=y_0$ 代入可得 $\alpha=\dfrac{1}{(x_0-x_1)(x_0-x_2)}$. 所以

$$l_0(x)=\frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)}.$$

同理可得

$$\begin{aligned} l_1(x)&=\frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)},\\ l_2(x)&=\frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)}. \end{aligned}$$

□

我们注意到, $p(x)$ 之所以可以写成 $l_0(x), l_1(x)$ 和 $l_2(x)$ 的线性组合, 是因为 $l_0(x), l_1(x), l_2(x)$ 组成了线性空间

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

的一组基, 而 $p(x)\in\mathbb{H}_2(x)$, 因此 $p(x)$ 可以由 $l_0(x), l_1(x), l_2(x)$ 线性表出. 这种利用基函数来计算插值函数的方法就是**基函数插值法**.

基函数插值法的两个关键问题

(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**.

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

$$l_k(x) = \frac{(x - x_0)\cdots(x - x_{k-1})(x - x_{k+1})\cdots(x - x_n)}{(x_k - x_0)\cdots(x_k - x_{k-1})(x_k - x_{k+1})\cdots(x_k - x_n)} = \prod_{i=0,\, i\neq k}^{n} \frac{x - x_i}{x_k - x_i}.$$

- 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.

解. 为了减小截断误差, 通常选取离插值点 x 比较近的点作为插值节点.

线性插值: 取插值节点 $x_0=0.5$, $x_1=0.6$. 根据 Lagrange 插值公式, 可得 $f(x)$ 在区间 $[0.5,0.6]$ 上的线性插值为

$$L_1(x)=y_0\frac{x-x_1}{x_0-x_1}+y_1\frac{x-x_0}{x_1-x_0}\approx 0.1823x-1.6046.$$

将 $x=0.54$ 代入可得 $\ln(0.54)\approx -0.6202$.

抛物线插值: 取插值节点 $x_0=0.4$, $x_1=0.5$, $x_2=0.6$. 根据 Lagrange 插值公式, 可得 $f(x)$ 在 $x=0.54$ 上的近似值为

$$\begin{aligned}\ln(0.54)&\approx L_2(0.54)\\&=y_0\frac{(0.54-x_1)(0.54-x_2)}{(x_0-x_1)(x_0-x_2)}+y_1\frac{(0.54-x_0)(0.54-x_2)}{(x_1-x_0)(x_1-x_2)}+y_2\frac{(0.54-x_0)(0.54-x_1)}{(x_2-x_0)(x_2-x_1)}\\&\approx -0.6153.\end{aligned}$$

□

✍ 我们将需要插值的点称为 **插值点**, 比如上面例题中的 0.54. 在实际计算中, 一般不需要给出插值多项式的具体表达式, 可以直接将插值点代入进行计算, 得到近似值.

✍ Lagrange 插值简单方便, 只要给定插值节点就可写出基函数, 易于计算机实现.

6.2.2 插值余项

记 Lagrange 插值多项式的余项为 $R_n(x)\triangleq f(x)-L_n(x)$.

定理 6.2 *设 $f(x)\in C^n[a,b]$ (即存在 n 阶连续导数), 且 $f^{(n+1)}(x)$ 在 (a,b) 内存在, 则对 $\forall x\in[a,b]$, 都存在 $\xi_x\in(a,b)$ 使得*

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

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

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.

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.

推论 6.3 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 $f(x)=x^m$ $(0\le m\le n)$, 则由推论 6.3 可知 $f(x)=L_n(x)$, 于是我们立即有下面的结论.

推论 6.4 当 $l_k(x)$ 是 Lagrange 基函数, 则有

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

特别地, 当 $m=0$ 时, 有

$$\sum_{k=0}^{n}l_k(x)=1.$$

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.

解. **线性插值**: 取插值节点 $x_0=0.5$, $x_1=0.6$, 则插值余项

$$\begin{aligned}|R_1(x)|&=\left|\frac{f^{(2)}(\xi_x)}{2!}(x-x_0)(x-x_1)\right|\\&=\left|\frac{\xi_x^{-2}}{2}(0.54-0.5)(0.54-0.6)\right|\qquad(\xi_x\in(x_0,x_1))\\&\le 2\times 0.04\times 0.06=0.048.\end{aligned}$$

抛物线插值: 取插值节点 $x_0=0.4$, $x_1=0.5$, $x_1=0.6$, 插值余项:

$$\begin{aligned}|R_2(x)|&=\left|\frac{f^{(3)}(\xi_x)}{3!}(x-x_0)(x-x_1)(x-x_2)\right|\\&=\left|\frac{2\xi_x^{-3}}{3!}(0.54-0.4)(0.54-0.5)(0.54-0.6)\right|\qquad(\xi_x\in(x_0,x_2))\end{aligned}$$

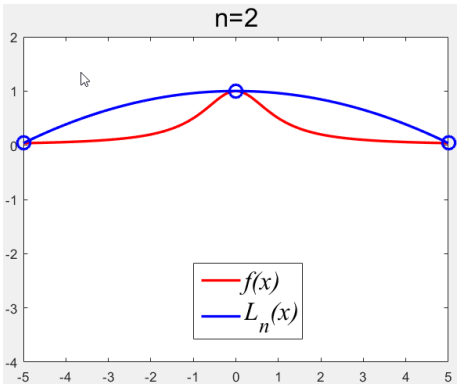
$$\leq \frac{125}{8} \times 0.14 \times 0.04 \times 0.06 = 0.00175.$$

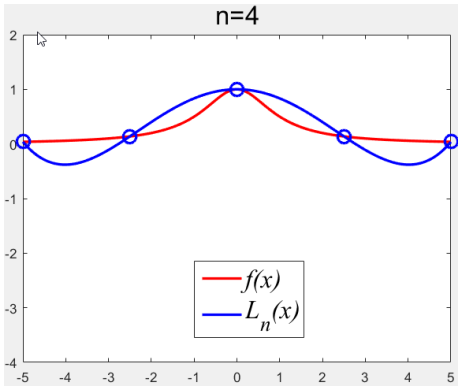
□

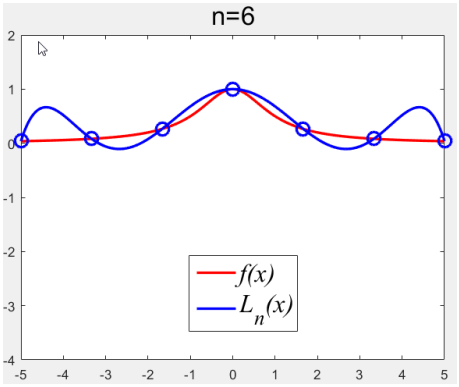
思考：由上例可知，抛物线插值要优于线性插值，是不是插值多项式次数越高，精度就越好？

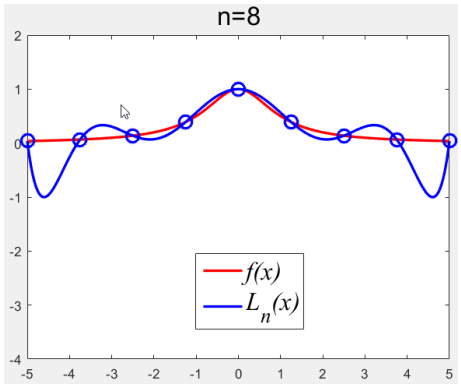
例 6.6 Runge 现象：已知函数 $f(x) = \dfrac{1}{1+x^2}$，插值区间 $[-5,5]$，取等距插值节点，即 $x_i = -5 + \dfrac{10i}{n}$，$i = 0,1,2,\ldots,n$，画出当 $n = 2,4,6,8,10,12$ 时，插值多项式 L_n 的图像. 并由此说明，插值多项式并不一定是次数越高就越好! (Interp_Lagrange_Runge.m)

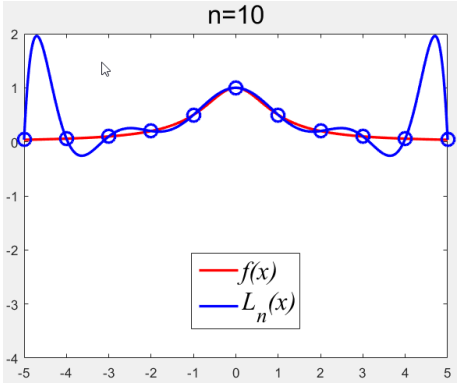
解. 当 $n = 2,4,6,8,10$ 时，原函数 $f(x)$ 和插值多项式 L_n 的图像如下：

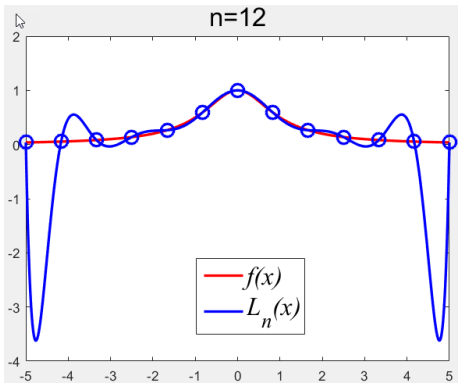












□

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.

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 $\mathbb{H}_n(x)$ 的一组基, 其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?

要解决以上问题, 我们需用到**差商**.

定义 6.4 (差商) 设节点 $x_0, x_1, \dots, x_n$, 我们称

$$f[x_i, x_j] = \frac{f(x_j) - f(x_i)}{x_j - x_i}$$

为 $f(x)$ 关于点 x_i, x_j 的**一阶差商**; 称

$$f[x_i, x_j, x_k] = \frac{f[x_j, x_k] - f[x_i, x_j]}{x_k - x_i}$$

为 $f(x)$ 关于点 x_i, x_j, x_k 的**二阶差商**; 一般地, 称

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

为 $f(x)$ 关于点 $x_0, x_1, \dots, x_k$ 的 **k 阶差商**.

差商的计算

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 6.7 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

解. 通过计算可得差商表如下:

x_i	$f(x_i)$	一阶差商	二阶差商	三阶差商
-2	5			
-1	3	-2		
1	17	7	3	
2	21	4	-1	-1

□

6.3.1 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

$$f(x)=f(x_0)+f[x,x_0](x-x_0). \tag{6.8}$$

同理, 由

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

可得

$$f[x,x_0]=f[x_0,x_1]+f[x,x_0,x_1](x-x_1). \tag{6.9}$$

以此类推, 我们有

$$f[x,x_0,x_1]=f[x_0,x_1,x_2]+f[x,x_0,x_1,x_2](x-x_2) \tag{6.10}$$

$$\vdots$$

$$f[x, x_0, \ldots, x_{n-1}] = f[x_0, x_1, \ldots, x_n] + f[x, x_0, x_1, \ldots, x_n](x - x_n). \tag{6.11}$$

将等式 (6.8)-(6.11) 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

$$N_n(x) = a_0 + a_1(x - x_0) + \cdots + a_n(x - x_0) \cdots (x - x_{n-1}), \tag{6.12}$$

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

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

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.

定理 6.5 设 $f(x)\in C^n[a,b]$ (n 阶连续可导), 且 $f^{(n+1)}(x)$ 在 (a,b) 内存在. 则对 $\forall x\in[a,b]$, 存在 $\xi_x\in(a,b)$ 使得

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

例 6.8 已知函数 $f(x)=\ln(x)$ 的函数值如下: (Interp_newton_01.m)

x	0.4	0.5	0.6	0.7	0.8
$f(x)$	-0.9163	-0.6931	-0.5108	-0.3567	-0.2231

试分别用 Newton 线性插值和抛物线插值计算 $\ln(0.54)$ 的近似值.

解. 取插值节点 $x_0=0.5, x_1=0.6, x_2=0.4$, 做差商表:

x_i	$f(x_i)$	一阶差商	二阶差商
0.5	**-0.6931**		
0.6	-0.5108	**1.8230**	
0.4	-0.9163	2.0275	**-2.0450**

于是可得, Newton 线性插值在 $x=0.54$ 上的近似值为

$$N_1(x)=f(x_0)+f[x_0,x_1](x-x_0)\approx -0.6931+1.8230(x-0.5)\approx -0.6202.$$

Newton 抛物线插值在 $x=0.54$ 上的近似值为

$$N_2(x)=N_1(x)+f[x_0,x_1,x_2](x-x_0)(x-x_1)\approx -0.6153.$$

□

思考: 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, Newton 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 Lagrange 插值相比, Newton 插值具有灵活增加插值节点的优点!

注记: Hermite 插值

在许多实际应用中, 不仅要求函数值相等, 而且还要求若干阶导数也相等, 如机翼设计等.
设插值节点为 $x_0,x_1,\dots,x_n$, 如果要求插值多项式满足

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

则计算这类插值多项式的方法就称为**Hermite 插值**.

✍ 在 Hermite 插值中, 并不一定需要在所有插值节点上的导数都相等, 在有些情况下, 可能只需要在部分插值点上的导数值相等即可.

6.4 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, 我们发现插值余项由两部分组成: $f(x)$ 的 $n+1$ 阶导数和 $\omega_{n+1}(x)$, 即

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

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.

6.4.1 分段线性插值

定义 6.5 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

$$I_h(x) = \frac{x - x_k}{x_{k-1} - x_k} f_{k-1} + \frac{x - x_{k-1}}{x_k - x_{k-1}} f_k, \quad x \in [x_{k-1}, x_k], \qquad k = 1,2,\ldots,n, \tag{6.13}$$

且在 $[x_{k-1}, x_k]$ 上余项满足

$$\max_{x_{k-1}\le x\le x_k} |f(x) - I_h(x)| \le \frac{1}{2!} \max_{x_{k-1}\le x\le x_k} |f''(x)| \max_{x_{k-1}\le x\le x_k} |(x - x_{k-1})(x - x_k)|$$

$$\leq \frac{h_k^2}{8} \max_{x_{k-1}\leq x\leq x_k} |f''(x)|, \tag{6.14}$$

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

定理 6.6 若 $f(x) \in C^2[a,b]$, 则分段线性插值函数 $I_h(x)$ 的误差满足

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

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

$$\lim_{h\to 0} I_h(x) = f(x)$$

在 $[a,b]$ 上一致成立, 即 $I_h(x)$ 在 $[a,b]$ 上一致收敛到 $f(x)$.

证明. 由 (6.14) 可知

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

所以

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

□

✍ 分段线性插值的优点: 简单易用, 关于 h 二阶收敛; 缺点: 插值函数 $I_h(x)$ 在插值节点处不可导.

思考: 如何构造分段抛物线插值? 误差是多少?

6.5 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 6.6 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 $f(x_k) = f_k$, $k = 0,1,\ldots,n$. 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**.

6.5.1 三次样条函数

定义 6.7 (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

$$s_k(x_{k-1}) = f_{k-1}, \quad s_k(x_k) = f_k. \tag{6.15}$$

于是

$$S(x) = \begin{cases} s_1(x), & x \in [x_0, x_1] \\ s_2(x), & x \in [x_1, x_2] \\ \quad \vdots & \\ s_n(x), & x \in [x_{n-1}, x_n] \end{cases}. \tag{6.16}$$

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

$$s_k'(x_k^-) = s_{k+1}'(x_k^+), \quad s_k''(x_k^-) = s_{k+1}''(x_k^+), \quad k = 1, 2, \ldots, n-1. \tag{6.17}$$

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 (6.15) 和 (6.17) 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**.

6.5.2 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.

6.5.3 三次样条函数的计算

由于 $S(x)$ 二阶可导, 所以可设

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

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

$$s_k(x)=\frac{(x_k-x)^3}{6h_k}M_{k-1}+\frac{(x-x_{k-1})^3}{6h_k}M_k+c_1x+c_2, \tag{6.18}$$

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

$$c_1=\frac{1}{h_k}(f_k-f_{k-1})-\frac{h_k}{6}(M_k-M_{k-1})=\frac{1}{h_k}\left[\left(f_k-\frac{M_kh_k^2}{6}\right)-\left(f_{k-1}-\frac{M_{k-1}h_k^2}{6}\right)\right],$$

$$c_2=f_{k-1}-\frac{M_{k-1}h_k^2}{6}-c_1x_{k-1}=\frac{x_k}{h_k}\left(f_{k-1}-\frac{M_{k-1}h_k^2}{6}\right)-\frac{x_{k-1}}{h_k}\left(f_k-\frac{M_kh_k^2}{6}\right).$$

代入 (6.18), 整理后可得

$$\begin{aligned}s_k(x)=&\frac{(x_k-x)^3}{6h_k}M_{k-1}+\frac{(x-x_{k-1})^3}{6h_k}M_k\\&+\frac{x_k-x}{h_k}\left(f_{k-1}-\frac{M_{k-1}h_k^2}{6}\right)+\frac{x-x_{k-1}}{h_k}\left(f_k-\frac{M_kh_k^2}{6}\right).\end{aligned} \tag{6.19}$$

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 (6.19), 整理后可得

$$\begin{aligned}s_k(x)=&\frac{M_k-M_{k-1}}{6h_k}(x-x_{k-1})^3+\frac{M_{k-1}}{2}(x-x_{k-1})^2\\&+\left(\frac{f_k-f_{k-1}}{h_k}-\frac{h_k(M_k+2M_{k-1})}{6}\right)(x-x_{k-1})+f_{k-1}.\end{aligned} \tag{6.20}$$

✍ 现在, 问题转化为如何确定 $M_0,M_1,\dots,M_n$ 的值?

由于 $S(x)\in C^2[a,b]$, 所以在节点处的一阶导数存在, 故

$$S'(x_k^-)=S'(x_k^+),\quad k=1,2,\dots,n-1,$$

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

$$\mu_k = \frac{h_k}{h_k + h_{k+1}}, \quad \lambda_k = \frac{h_{k+1}}{h_k + h_{k+1}},$$
$$d_k = \frac{6(f[x_k, x_{k+1}] - f[x_{k-1}, x_k])}{h_k + h_{k+1}} = 6f[x_{k-1}, x_k, x_{k+1}],$$

则上面的方程可写为

$$\mu_k M_{k-1} + 2M_k + \lambda_k M_{k+1} = d_k, \quad k = 1, 2, \ldots, n-1. \tag{6.21}$$

✍ 上述方程中的系数有个重要性质: $\mu_k + \lambda_k = 1$

✍ 方程 (6.21) 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 (6.21) 联立可得方程组

$$\begin{bmatrix} 2 & 1 & & & \\ \mu_1 & 2 & \lambda_1 & & \\ & \ddots & \ddots & \ddots & \\ & & \mu_{n-1} & 2 & \lambda_{n-1} \\ & & & 1 & 2 \end{bmatrix} \begin{bmatrix} M_0 \\ M_1 \\ \vdots \\ M_{n-1} \\ M_n \end{bmatrix} = \begin{bmatrix} d_0 \\ d_1 \\ \vdots \\ d_{n-1} \\ d_n \end{bmatrix}. \tag{6.22}$$

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

$$\begin{bmatrix} 2 & \lambda_1 & & & \\ \mu_2 & 2 & \lambda_2 & & \\ & \ddots & \ddots & \ddots & \\ & & \mu_{n-2} & 2 & \lambda_{n-2} \\ & & & \mu_{n-1} & 2 \end{bmatrix} \begin{bmatrix} M_1 \\ M_2 \\ \vdots \\ M_{n-2} \\ M_{n-1} \end{bmatrix} = \begin{bmatrix} d_1-\mu_1 f_0'' \\ d_2 \\ \vdots \\ d_{n-2} \\ d_{n-1}-\lambda_{n-1} f_n'' \end{bmatrix}. \tag{6.23}$$

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 (6.21) 联立可得方程组

$$\begin{bmatrix} 2 & \lambda_1 & & & \mu_1 \\ \mu_2 & 2 & \lambda_2 & & \\ & \ddots & \ddots & \ddots & \\ & & \mu_{n-1} & 2 & \lambda_{n-1} \\ \lambda_n & & & \mu_n & 2 \end{bmatrix} \begin{bmatrix} M_1 \\ M_2 \\ \vdots \\ M_{n-1} \\ M_n \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_{n-1} \\ d_n \end{bmatrix}. \tag{6.24}$$

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 (6.22), (6.23) 和 (6.24) 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 (6.20), 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 (6.20) 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 6.9 函数 $f(x)$ 定义在 $[27.7, 30]$ 上, 插值节点及相应函数值下表, 试求三次样条插值多项式 $S(x)$, 满足边界条件 $S'(27.7) = 3.0$, $S'(30) = -4.0$. (`Interp_spline_01.m`)

x	27.7	28	29	30
$f(x)$	4.1	4.3	4.1	3.0

解. 做差商表

x_i	$f(x_i)$	一阶差商	二阶差商
27.7	4.1		
28	4.3	$2/3$	
29	4.1	-0.2	$-2/3$
30	3.0	-1.1	$-9/20$

由题意可知 $h_0 = 0.3, h_1 = 1.0, h_2 = 1.0$, 所以

$$\mu_1 = \frac{h_0}{h_0 + h_1} = \frac{3}{13}, \quad \lambda_1 = 1 - \mu_1 = \frac{10}{13},$$

$$\mu_2 = \frac{h_1}{h_1 + h_2} = \frac{1}{2}, \quad \lambda_2 = 1 - \mu_2 = \frac{1}{2},$$

$$d_0 = \frac{6}{h_0}(f[x_0, x_1] - f'_0) = 20\left(\frac{2}{3} - 3.0\right) = -\frac{140}{3},$$

$$d_1 = 6f[x_0, x_1, x_2] = -4,$$

$$d_2 = 6f[x_1, x_2, x_3] = -2.7,$$

$$d_3 = \frac{6}{h_2}(f'_3 - f[x_2, x_3]) = 6(-4 + 1.1) = -17.4.$$

因此可得线性方程组

$$\begin{bmatrix} 2 & 1 & & \\ \frac{3}{13} & 2 & \frac{10}{13} & \\ & \frac{1}{2} & 2 & \frac{1}{2} \\ & & 1 & 2 \end{bmatrix} \begin{bmatrix} M_0 \\ M_1 \\ M_2 \\ M_3 \end{bmatrix} = \begin{bmatrix} -\frac{140}{3} \\ -4 \\ -2.7 \\ -17.4 \end{bmatrix},$$

解得 (追赶法)

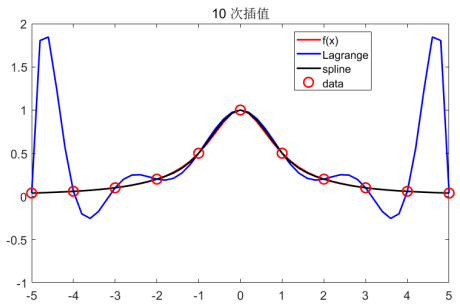
$$M_3 = -\frac{4603}{505} \approx -9.115, \quad M_2 = \frac{419}{505} \approx 0.830,$$
$$M_1 = -\frac{40}{101} \approx 0.396, \quad M_0 = -\frac{7130}{303} \approx -23.531.$$

代入 $s_k(x)$ 的表达式 (6.19) 可得

$$S(x) = \begin{cases} 13.293(x-27.7)^3 - 11.766(x-27.7)^2 + 3.000(x-27.7) + 4.1, & x \in [27.7, 28] \\ 0.072(x-28)^3 + 0.198(x-28)^2 - 0.470(x-28) + 4.3, & x \in [28, 29] \\ -1.657(x-29)^3 + 0.415(x-29)^2 + 0.143(x-29) + 4.1, & x \in [29, 30]. \end{cases}$$

□

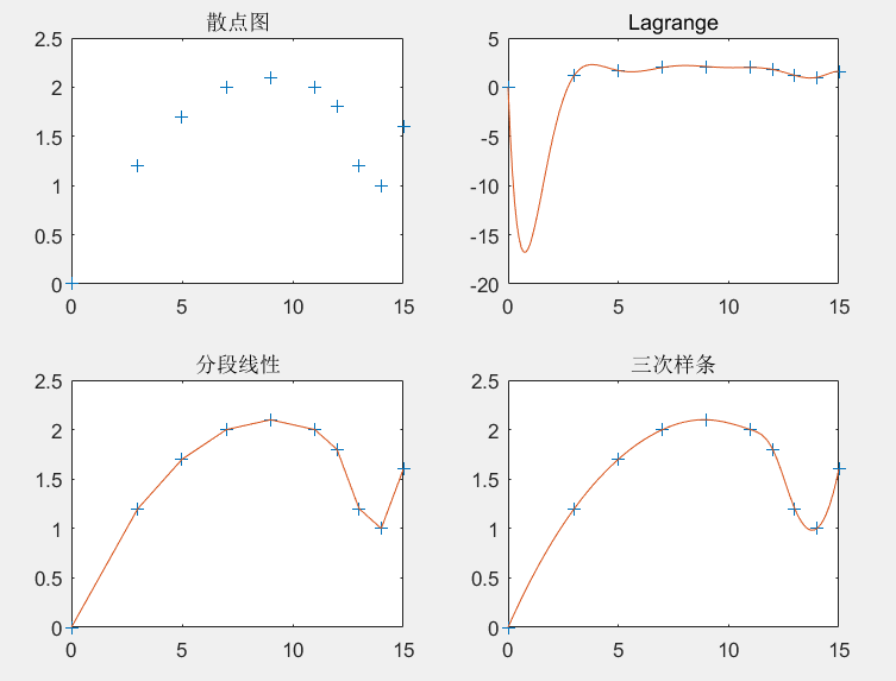
例 6.10 函数 $f(x) = \dfrac{1}{1+x^2}$, 插值区间 $[-5, 5]$, 取 11 个等距节点 (10 等分), 试画出 10 次插值多项式 $L_{10}(x)$ 与三次样条插值多项式 $S(x)$ 的函数图形. (Interp_spline_02.m)



例 6.11 机床加工. 待加工零件的外形根据工艺要求由一组数据 (x, y) 给出, 用程控铣床加工时每一刀只能沿 x 方向和 y 方向走非常小的一步, 这就需要从已知数据得到加工所要求的步长很小的 (x, y) 坐标. 下表中给出的 x, y 数据位于机翼断面的下轮廓线上, 假设需要得到 x 坐标每改变 0.1 时的 y 坐标. 试完成加工所需数据, 画出曲线. 要求用 Lagrange, 分段线性和三次样条三种插值方法计算. (Interp_spline_03.m)

x	0	3	5	7	9	11	12	13	14	15
y	0	1.2	1.7	2.0	2.1	2.0	1.8	1.2	1.0	1.6

解. 图形为



可以看出, Lagrange 插值的结果根本不能用. 分段线性插值的光滑性较差 (特别是在 $x = 14$ 附近弯曲处), 建议选用三次样条插值的结果. □

6.6 课后练习

练习 6.1 已知 $f(x) = \mathrm{e}^x$ 在 $[-4, 4]$ 上的等距节点函数值表, 若用二次插值求 e^x 的近似值 ($x \in [-4, 4]$) , 要使截断误差不超过 10^{-6}, 问步长 h 应取多少?

练习 6.2 已知 $f(x) = x^7 + x^4 + 3x + 1$, 求 $f[2^0, 2^1, \ldots, 2^7]$ 和 $f[2^0, 2^1, \ldots, 2^8]$.

练习 6.3 已知 $f(-1) = -3$, $f(1) = 0$, $f(2) = 4$, 分别用以下三种方法计算 $f(x)$ 的二次插值多项式:

(1) 用单项式基函数, 即 $\{1, x, x^2\}$,

(2) 用 Lagrange 基函数,

(3) 用 Newton 基函数.

练习 6.4 求次数不超过 3 的多项式 $p(x)$, 满足

$$p(x_0) = f(x_0), \quad p(x_1) = f(x_1), \quad p'(x_0) = f'(x_0), \quad p''(x_0) = f''(x_0).$$

练习 6.5 求次数不超过 3 的多项式 $p(x)$, 满足

$$p(0) = 0,\ p'(0) = 1, \quad p(1) = 1,\ p'(1) = 2.$$

练习 6.6 证明两点三次 Hermite 插值余项为

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

其中 $\xi_x \in (x_0, x_1)$ 且与 x 相关. 并由此给出分段三次 Hermite 插值的误差限.

练习 6.7 求一个次数不超过 4 的多项式 $p(x)$, 满足

$$p(0) = p'(0) = 0, \quad p(1) = p'(1) = 1, \quad p(2) = 1.$$

(提示: 这里需要使用非标准的插值计算方法, 可以采用一些技巧, 不要死算.)

练习 6.8 设 $f(x) = \dfrac{1}{1+x^2}$, 在 $[-5, 5]$ 上取 n 等分点做分段线性插值, 计算 $n = 10$ 时插值函数 $I_h(x)$ 在各区间中点处的值, 并估计误差.

练习 6.9 求 $f(x) = x^2$ 在 $[a, b]$ 上分段线性插值函数 $I_h(x)$, 并估计误差.

练习 6.10 给定数据表如下:

x_k	0.25	0.30	0.40	0.45	0.53
y_k	0.5000	0.5477	0.6245	0.6708	0.7280

试求三次样条插值函数 $S(x)$, 满足

(1) $S'(0.25) = 1.0000$, $S'(0.53) = 0.6868$;

(2) $S''(0.25) = S''(0.53) = 0$.

7 数值积分与数值微分

考虑定积分

$$I(f) \triangleq \int_a^b f(x)\,\mathrm{d}x. \tag{7.1}$$

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

(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.

7.1 数值积分基本概念

7.1.1 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

$$\int_a^b f(x)\,\mathrm{d}x \approx \sum_{i=0}^{n} h_i f(\xi_i). \tag{7.2}$$

这就是目前常用的近似求积方法.

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 α_i 和 x_i)

$$I(f) = \int_a^b f(x)\,\mathrm{d}x \approx \sum_{i=0}^{n} \alpha_i f(x_i) \triangleq I_n(f). \tag{7.3}$$

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 α_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 7.1 (矩形公式) 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

$$\int_a^b f(x)\,\mathrm{d}x = f(\xi)(b-a).$$

但 ξ 的取值往往是不可知的. 因此我们可以考虑用 $[a,b]$ 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**: $\int_a^b f(x)\,\mathrm{d}x \approx f\left(\frac{a+b}{2}\right)(b-a)$.

7.1.2 代数精度

根据 Weierstrass 逼近定理, 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 7.1 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 $f(x) = 1, x, x^2, \ldots, x^m$ 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 7.2 试确定系数 α_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.

$$\int_{-1}^{1} f(x)\,\mathrm{d}x \approx \alpha_0 f(-1) + \alpha_1 f(0) + \alpha_2 f(1).$$

解. 分别取 $f(x) = 1, x, x^2$, 令求积公式精确成立, 可得

$$\begin{cases} \alpha_0 + \alpha_1 + \alpha_2 = b - a \\ -\alpha_0 + \alpha_2 = 0 \\ \alpha_0 + \alpha_2 = \dfrac{2}{3} \end{cases}$$

求解该方程组, 可得 $\alpha_0 = \frac{1}{3}$, $\alpha_1 = \frac{4}{3}$, $\alpha_2 = \frac{1}{3}$. 因此求积公式为

$$\int_{-1}^{1} f(x)\,\mathrm{d}x \approx \frac{1}{3} f(-1) + \frac{4}{3} f(0) + \frac{1}{3} f(1).$$

将 $f(x) = x^3$ 代入可得, 公式左边 $= 0 =$ 右边. 将 $f(x) = x^4$ 代入可得, 公式左边为 $\frac{2}{5}$, 右边为 $\frac{2}{3}$. 因此该求积公式的代数精度为 3. □

✍ 通常情况下, 由于求积公式 (7.3) 中含有 $n+1$ 个参数, 因此可以选取合适的 α_i 的值, 使得求积公式 (7.3) 至少具有 n 次代数精度.

引理 7.1 *设机械求积公式 (7.3) 具有 m 次代数精度 ($m \geq 0$), 则有*

$$\alpha_0 + \alpha_1 + \cdots + \alpha_n = b - a. \tag{7.4}$$

证明. 将 $f(x) = 1$ 代入求积公式 (7.3), 令等式精确成立即可. □

✍ 上面的结论是机械求积公式的一个基本性质.

7.1.3 收敛性与稳定性

定义 7.2 *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 (即分割的模趋于 0 时) 极限存在, 该极限就是定积分.

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 7.3 考虑机械求积公式 (7.3), 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

$$\left| \sum_{k=0}^{n} \alpha_k f(x_k) - \sum_{k=0}^{n} \alpha_k \tilde{f}_k \right| < \varepsilon,$$

则称机械求积公式 (7.3) 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 7.2 若机械求积公式 (7.3) 具有 0 次或以上代数精度, 且求积系数 α_i 都是正数, 则求积公式是稳定的.

7.2 插值型求积公式

构造求积公式的一个常用方法就是使用插值多项式. 设 $L_n(x)$ 是 $f(x)$ 关于节点 $a \le x_0 < x_1 < \cdots < x_n \le b$ 的 n 次插值多项式, 则

$$\begin{aligned} \int_a^b f(x)\,\mathrm{d}x \approx \int_a^b L_n(x)\,\mathrm{d}x &= \int_a^b \sum_{k=0}^{n} l_k(x) f(x_k)\,\mathrm{d}x \\ &= \sum_{k=0}^{n} \left(\int_a^b l_k(x)\,\mathrm{d}x \right) f(x_k) \triangleq \sum_{k=0}^{n} \alpha_k f(x_k). \end{aligned} \tag{7.5}$$

这就是**插值型求积公式**, 其中 $l_k(x)$ 是 n 次 Lagrange 基函数, $\alpha_k = \int_a^b l_k(x)\,\mathrm{d}x$.

由多项式插值余项公式可知, 插值型求积公式 (7.5) 的余项为

$$R[f] = \int_a^b \big(f(x) - L_n(x)\big)\,\mathrm{d}x = \int_a^b \frac{f^{(n+1)}(\xi_x)}{(n+1)!} \omega_{n+1}(x)\,\mathrm{d}x, \tag{7.6}$$

其中

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

✍ 由于 ξ_x 未知, 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

$$\big|\, R[f] \,\big| = \left| \int_a^b \frac{f^{(n+1)}(\xi_x)}{(n+1)!} \omega_{n+1}(x)\,\mathrm{d}x \right| \le \frac{M_{n+1}}{(n+1)!} \int_a^b |\omega_{n+1}(x)|\,\mathrm{d}x,$$

其中 $M_{n+1} = \max\limits_{a \le x \le b} |f^{(n+1)}(x)|$.

引理 7.3 插值型求积公式 (7.5) 至少具有 n 次代数精度. (板书)

证明. 依次将 $f(x) = 1, x, x^2, \dots, x^n$ 代入, 可得 $f^{(n+1)}(\xi_x) = 0$. 故 $R[f] = 0$, 所以求积公式对 $f(x) = 1, x, x^2, \dots, x^n$ 精确成立, 即代数精度至少为 n. □

事实上, 我们有下面的性质.

定理 7.4 机械求积公式 (7.3) 至少具有 n 次代数精度的充要条件是该公式是插值型的.

✍ 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.

7.3 Newton-Cotes 公式

定义 7.4 如果插值型求积公式 (7.5) 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

$$I_n(f) = (b-a)\sum_{k=0}^{n} C_k^{(n)} f(x_k), \tag{7.7}$$

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) $\sum_{k=0}^{n} C_k^{(n)} = 1$; (2) $C_k^{(n)} = C_{n-k}^{(n)}$, $k = 0, 1, 2, \ldots, n$.

7.3.1 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 $C_0^{(1)} = C_1^{(1)} = \frac{1}{2}$, 此时的 Newton-Cotes 公式为

$$I_1(f) = \frac{b-a}{2}\left(f(a) + f(b)\right). \tag{7.8}$$

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

$$I_2(f) = \frac{b-a}{6}\left(f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)\right). \tag{7.9}$$

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

$$I_3(f) = \frac{b-a}{8}\big(f(x_0) + 3f(x_1) + 3f(x_2) + f(x_3)\big). \tag{7.10}$$

该公式称为 **Simpson 3/8 公式** 或 **Boole 公式** 或 **Milne 公式**.

✍ 当 $n > 7$ 时, Cotes 系数中会出现负数, 会导致算法的不稳定, 因此不考虑 $n > 7$.
✍ 梯形公式和抛物线公式简单易用, 因此很受欢迎.

定理 7.5 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.

梯形公式的余项

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

$$\int_a^b f(x)\,\mathrm{d}x = \frac{b-a}{2}\Big(f(a)+f(b)\Big) - \frac{(b-a)^3}{12} f''(\eta), \quad \eta \in (a,b). \tag{7.11}$$

Simpson 公式的余项

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

$$\int_a^b f(x)\,\mathrm{d}x = \frac{b-a}{6}\left(f(a)+4f\left(\frac{a+b}{2}\right)+f(b)\right) - \frac{(b-a)^5}{2880} f^{(4)}(\eta), \quad \eta \in (a,b). \tag{7.12}$$

7.4 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.

7.4.1 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

$$\int_a^b f(x)\,\mathrm{d}x = \sum_{k=0}^{n-1}\int_{x_k}^{x_{k+1}} f(x)\,\mathrm{d}x \approx \sum_{k=0}^{n-1}\frac{h}{2}(f(x_k)+f(x_{k+1}))$$

$$= h\left(\frac{f(a)+f(b)}{2}+\sum_{k=1}^{n-1} f(x_k)\right).$$

这就是**复合梯形公式** (Composite Trapezoidal rule), 通常记为 T_n, 即

$$T_n = h\left(\frac{f(a)+f(b)}{2}+\sum_{k=1}^{n-1} f(x_k)\right).$$

设 $f(x)\in C^2[a,b]$, 则在每个小区间 $[x_k, x_{k+1}]$ 上的余项为 $-\frac{h^3}{12}f''(\eta_k)$, 所以整体余项为

$$R_n[f] = \int_a^b f(x)\,\mathrm{d}x - T_n = -\frac{h^3}{12}\sum_{k=0}^{n-1} f''(\eta_k).$$

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.

7.4.2 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 7.3 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 $\int_0^1 f(x)\,\mathrm{d}x$ 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

7.5 Gauss 求积公式

> Gauss quadrature, one of the jewels of numerical analysis, is a beautiful and powerful idea.
>
> — L.N. Trefethen, SIAM Review, 2008.

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 7.4 试确定 α_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.

$$\int_{-1}^{1} f(x)\,\mathrm{d}x \approx \alpha_0 f(x_0) + \alpha_1 f(x_1). \tag{7.13}$$

由此可见, 采用不等距节点的两点求积公式 (7.13) 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.

7.5.1 一般 Gauss 求积公式

定义 7.5 设 $\rho(x)$ 是 $[a,b]$ 上的权函数, 若求积公式

$$\int_a^b \rho(x) f(x)\,\mathrm{d}x \approx \sum_{i=0}^{n} \alpha_i f(x_i), \tag{7.14}$$

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**, α_i 称为 **Gauss 系数**.

✍ 需要指出的是, 求积公式 (7.14) 的右端只包含 $f(x)$ 的函数值, 与权函数无关.

求积公式 (7.14) 中含有 $2n+2$ 的待定参数, 即 α_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 (7.14) 精确成立, 然后解出 α_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 (7.14) 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 $\sum\limits_{i=0}^{n} \alpha_i f(x_i) = 0$, 但显然

$$\int_a^b \rho(x) f(x)\,\mathrm{d}x > 0,$$

即求积公式 (7.14) 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 7.6 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 7.4 可以看出, 我们可以将 $f(x)=1,x,x^2,\dots,x^{2n+1}$ 代入, 并令求积公式 (7.14) 精确成立, 这样就能解出 α_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 α_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 α_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 7.7 设节点 $a\le x_0<x_1<\cdots<x_n\le b$, 则插值型求积公式 (7.14) 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

$$\int_a^b \rho(x)\omega_{n+1}(x)p(x)\,\mathrm{d}x=0,\quad \forall\, p(x)\in\mathbb{H}_n.$$

计算 Gauss 点的一般方法

(1) 设 $\omega_{n+1}(x)=x^{n+1}+a_nx^n+a_{n-1}x^{n-1}+\cdots+a_1x+a_0$;

(2) 利用 $\omega_{n+1}(x)$ 与 $p(x)=1,x,x^2,\dots,x^n$ 正交 (带权) 的性质, 得到 $n+1$ 个线性方程, 解出 a_k, $k=0,1,2,\dots,n$, 这样就能确定多项式 $\omega_{n+1}(x)$;

(3) 求出多项式 $\omega_{n+1}(x)$ 的 $n+1$ 个零点, 这就是 Gauss 点.

例 7.5 试确定 α_i 和 x_i, 使得下面的求积公式具有尽可能高的代数精度

$$\int_0^1 \sqrt{x}f(x)\,\mathrm{d}x\approx\alpha_0f(x_0)+\alpha_1f(x_1).$$

Gauss 求积公式的余项

设 $p_{2n+1}(x)$ 是 $f(x)$ 关于点 $x_0,x_1,x_2,\dots,x_n$ 的 $2n+1$ 次 Hermite 插值多项式, 满足

$$p_{2n+1}(x_i)=f(x_i),\quad p'_{2n+1}(x_i)=f'(x_i),\quad i=0,1,2,\dots,n.$$

若 $f(x)\in C^{2n+2}[a,b]$, 则可以验证, 插值余项为

$$R_n(x)\triangleq f(x)-p_{2n+1}(x)=\frac{f^{(2n+2)}(\xi_x)}{(2n+2)!}\omega_{n+1}^2.$$

由于求积公式 (7.14) 具有 $2n+1$ 次代数精度, 故

$$\int_a^b \rho(x)p_{2n+1}(x)\,\mathrm{d}x=\sum_{i=0}^{n}\alpha_ip_{2n+1}(x_i).$$

所以, Gauss 求积公式的余项为

$$\begin{aligned} R_n[f] &\triangleq \int_a^b \rho(x)f(x)\,\mathrm{d}x - \sum_{i=0}^{n}\alpha_i f(x_i) \\ &= \int_a^b \rho(x)f(x)\,\mathrm{d}x - \sum_{i=0}^{n}\alpha_i p_{2n+1}(x_i) \\ &= \int_a^b \rho(x)\Big(f(x) - p_{2n+1}(x)\Big)\,\mathrm{d}x \\ &= \int_a^b \rho(x)\frac{f^{(2n+2)}(\xi_x)}{(2n+2)!}\omega_{n+1}^2\,\mathrm{d}x. \end{aligned}$$

假定 $f^{(2n+2)}(\xi_x)$ 在 $[a,b]$ 上关于 x 是连续的, 则由积分中值定理可知, 存在 $\eta\in(a,b)$, 使得

$$R_n[f] = \frac{f^{(2n+2)}(\eta)}{(2n+2)!}\int_a^b \rho(x)\omega_{n+1}^2\,\mathrm{d}x. \tag{7.15}$$

Gauss 公式的收敛性与稳定性

定理 7.8 设 $f(x)\in C[a,b]$, 则 Gauss 求积公式是收敛的, 即

$$\lim_{n\to\infty}\sum_{i=0}^{n}\alpha_i f(x_i) = \int_a^b \rho(x)f(x)\,\mathrm{d}x.$$

定理 7.9 Gauss 求积公式中的系数 α_i 全是正数, 因此 Gauss 求积公式是稳定的.

7.6 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.

7.6.1 插值型求导公式

设 $p_n(x)$ 是 $f(x)$ 基于节点 $x_0,x_1,x_2,\ldots,x_n$ 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

$$f'(x_i) - p_n'(x_i) = \frac{f^{(n+1)}(\xi_x)}{(n+1)!}\omega_{n+1}'(x_i) = \frac{f^{(n+1)}(\xi_x)}{(n+1)!}\prod_{k=0,k\neq i}^{n}(x_i - x_k). \tag{7.16}$$

✍ 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.

7.6.2 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

$$f'(x_0) = p_1'(x_0) + \frac{f''(\xi_0)}{2}(x_0 - x_1) = \frac{1}{h}(f(x_1) - f(x_0)) - \frac{h}{2}f''(\xi_0), \tag{7.17}$$

$$f'(x_1) = p_1'(x_1) + \frac{f''(\xi_1)}{2}(x_1 - x_0) = \frac{1}{h}(f(x_1) - f(x_0)) + \frac{h}{2}f''(\xi_1). \tag{7.18}$$

公式 (7.17) 称为 **向前差分** (forward difference) 公式, 公式 (7.18) 称为 **向后差分** (backward difference) 公式.

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

$$p_2'(x_0) = \frac{1}{2h}\big(-3f(x_0) + 4f(x_1) - f(x_2)\big),$$

$$p_2'(x_1) = \frac{1}{2h}\big(f(x_2) - f(x_0)\big),$$

$$p_2'(x_2) = \frac{1}{2h}\big(f(x_0) - 4f(x_1) + 3f(x_2)\big).$$

由公式 (7.16) 可知

$$f'(x_0) = \frac{1}{2h}\big(-3f(x_0) + 4f(x_1) - f(x_2)\big) + \frac{h^2}{3}f^{(3)}(\xi_0),$$

$$f'(x_1)=\frac{1}{2h}\big(f(x_2)-f(x_0)\big)-\frac{h^2}{6}f^{(3)}(\xi_1),\tag{7.19}$$
$$f'(x_2)=\frac{1}{2h}\big(f(x_0)-4f(x_1)+3f(x_2)\big)+\frac{h^2}{3}f^{(3)}(\xi_2).$$

这就是三点求导公式, 特别中间的公式 (7.19), 是常用的计算一阶导数的 **中心差分** (centered difference) 公式.

7.6.3 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 (7.16) 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.

7.7 课后练习

练习 7.1 确定下列求积公式中的待定参数, 使其具有尽可能高的代数精度, 并指出所构造的求积公式的代数精度.

(1) $\displaystyle\int_{-1}^{1}f(x)\,\mathrm{d}x\approx\alpha_{-1}f(-1)+\alpha_0f(0)+\alpha_1f(1)$

(2) $\displaystyle\int_{-2}^{2}f(x)\,\mathrm{d}x\approx\alpha_{-1}f(-1)+\alpha_0f(0)+\alpha_1f(1)$

(3) $\displaystyle\int_{-1}^{1}f(x)\,\mathrm{d}x\approx\frac{1}{3}\Big(f(-1)+2f(x_1)+3f(x_2)\Big)$

(4) $\displaystyle\int_{0}^{h}f(x)\,\mathrm{d}x\approx\frac{h}{2}\Big(f(0)+f(h)\Big)+ah^2\Big(f'(0)-f'(h)\Big)$

练习 7.2 分别用复化梯形法 T_4 和复化抛物线方法 S_2 计算下列定积分:

(1) $\displaystyle\int_0^2\frac{x}{4+x^2}\,\mathrm{d}x,$ （注: 不要做近似计算）

(2) $\displaystyle\int_1^7\sqrt{x}\,\mathrm{d}x,$ （注: 计算过程中保留小数点后两位数字）

练习 7.3 用 Simpson 公式计算定积分 $\displaystyle\int_1^2e^{-x}\,\mathrm{d}x$, 并估计误差.
(提示: 用余项公式估计误差)

练习 7.4 设 $f''(x)>0, x\in[a,b]$, 证明用梯形公式计算定积分 $I=\displaystyle\int_a^bf(x)\,\mathrm{d}x$ 所得结果比准确值大, 并说明其几何意义.

练习 7.5 已知 $\rho(x)=\dfrac{1}{\sqrt{x}}$ 是 $[0,2]$ 上的权函数, 试构造 Gauss 求积公式

$$\int_0^2 \frac{1}{\sqrt{x}} f(x) \approx \alpha_0 f(x_0) + \alpha_1 f(x_1).$$

(提示: 采用标准方法, 即先求 Gauss 点, 然后求 Gauss 系数.)

练习 7.6 用 $n=2$ 的 Gauss-Legendre 求积公式计算定积分:

$$\int_0^2 e^x \sin(x)\,\mathrm{d}x,$$

计算过程中保留小数点后两位数字.

练习 7.7 设 $f(x)\in C^4[a,b]$, 给出计算 $\displaystyle\int_a^b f(x)\,\mathrm{d}x$ 的复合两点 Gauss-Legendre 求积公式的余项公式 (求积区间 n 等分).

练习 7.8 用三点公式计算 $f(x)$ 在 $x=1.2$ 处的一阶导数和二阶导数. 函数值如下: $f(1.1)=0.227$, $f(1.2)=0.207$, $f(1.3)=0.189$.