矩阵计算

潘建瑜

August 24, 2019

1 课程主要内容

- 线性方程组的直接解法
- 线性最小二乘问题的数值算法
- 非对称矩阵的特征值计算
- 对称矩阵特征值计算与奇异值分解
- 线性方程组迭代算法
- 特征值问题的迭代算法 (部分特征值和特征向量)
- 稀疏矩阵计算

主要参考资料

- G. H. Golub and C. F. van Loan, “Matrix Computations (4th),”2013. 对应中文版为:《矩阵计算》(第三版), 袁亚湘等译, 2001.
- J. W. Demmel, “Applied Numerical Linear Algebra,”1997. 对应中文版为:《应用数值线性代数》, 王国荣译, 2007.
- L. N. Trefethen and D. Bau, III, “Numerical Linear Algebra,”1997. 对应中文版为:《数值线性代数》, 陆金甫等译, 2006.
- 徐树方, “矩阵计算的理论与方法,”北京大学出版社, 1995.
- 曹志浩, “数值线性代数,”复旦大学出版社, 1996.

课程主页

http://math.ecnu.edu.cn/ jypan/Teaching/MatrixComp/

2 引言

计算数学, 也称数值分析或计算方法.

1947 年 Von Neumann 和 Goldstine 在《美国数学会通报》发表了题为“高阶矩阵的数值求逆”的著名论文, 开启了现代计算数学的研究

一般来说, 计算数学主要研究如何求出数学问题的近似解 (数值解), 包括算法的设计与分析 (收敛性, 稳定性, 复杂性等)

> 计算数学主要研究内容:
>
> 数值逼近, 数值微积分, 数值代数, 微分方程数值解, 最优化等

2.1 为什么学计算数学

科学计算是 20 世纪重要科学技术进步之一, 已与理论研究和实验研究相并列成为科学研究的第三种方法. 现今科学计算已是体现国家科学技术核心竞争力的重要标志, 是国家科学技术创新发展的关键要素。

——国家自然科学基金•重大项目指南, 2014

计算科学是 21 世纪确保国家核心竞争能力的战略技术之一。

——计算科学: 确保美国竞争力,2005

科学计算的核心/数学基础: 计算数学.

2.2 矩阵计算 (数值线性代数) 的重要性

If any other mathematical topic is as fundamental to the mathematical sciences as calculus and differential equations, it is numerical linear algebra.

——Trefethen and Bau, 1997.

国家自然科学基金委员会关于计算数学的分类 (2018):

• 计算数学与科学工程计算 (A0117)

- 偏微分方程数值解 (A011701)
- 流体力学中的数值计算 (A011702)
- 一般反问题的计算方法 (A011703)
- 常微分方程数值计算 (A011704)
- 数值代数 (A011705)

- 数值逼近与计算几何 (A011706)
- 谱方法及高精度数值方法 (A011707)
- 有限元和边界元方法 (A011708)
- 多重网格技术与区域分解 (A011709)
- 自适应方法 (A011720)
- 并行计算 (A011711)

2.3 计算数学的主要任务

- 算法设计: 构造求解各种数学问题的数值方法
- 算法分析: 收敛性、稳定性、复杂性、计算精度等
- 算法实现: 编程实现、软件开发

好的数值方法一般需要满足以下几点：

- 有可靠的理论分析, 即收敛性、稳定性等有数学理论保证
- 有良好的计算复杂性 (时间和空间)
- 易于在计算机上实现
- 要有具体的数值试验来证明是行之有效的

2.4 矩阵计算基本问题

数值代数: 数值线性代数 (矩阵计算) 和数值非线性代数

数值线性代数 (矩阵计算) 主要研究以下问题:

- 线性方程组求解

$$Ax = b, A \in R^{n\times n}\text{非奇异}$$

- (线性) 最小二乘问题

$$\min_{x\in\mathbb{R}^n} \|Ax - b\|_2 \quad , \quad A \in \mathbb{R}^{m\times n}, m \geq n$$

- 矩阵特征值问题

$$Ax = \lambda x \quad , \quad A \in \mathbb{R}^{n\times n}, \lambda \in \mathbb{C}, x \in \mathbb{C}^n, x \neq 0$$

- 矩阵奇异值问题

$$A^\top Ax = \sigma^2 x \quad , \quad A \in \mathbb{R}^{m\times n}, \sigma \geq 0, x \in \mathbb{R}^n, x \neq 0$$

- 其他问题：

广义特征值问题, 二次特征值问题, 非线性特征值问题, 矩阵方程, 特征值反问题, 张量计算, ...

† 数值方法一般都是近似方法, 求出的解是带有误差的, 因此误差分析非常重要.

2.5 矩阵计算常用方法 (技术或技巧)

- 矩阵分解
- 矩阵分裂
- 扰动分析

† 问题的特殊结构对算法的设计具有非常重要的影响.

† 在编程实现时, 要充分利用现有的优秀程序库.

二十世纪十大优秀算法 (SIAM News, 2000)

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

3 线性代数基础

3.1 线性空间与内积空间

- 数域, 如: Q, R, C
- 线性空间, 如: $R^n, C^n, R^{m\times n}$
- 线性相关与线性无关, 秩, 基, 维数
- 线性子空间

- 像空间 (列空间, 值域) Ran(A), 零空间 (核) Ker(A)
- 张成子空间:$span x_1, x_2, ..., x_k, span(A) = Ran(A)$

3.1.1 直和

设 S_1, S_2 是子空间, 若 $S_1 + S_2$ 中的任一元素都可唯一表示成

$$x = x_1 + x_2, x_1 \in S1, x2 \in S2,$$

则称 $S_1 + S_2$ 为直和, 记为 $S_1 \oplus S_2$.

定理 3.1 *设 S_1 是 S 的子空间, 则存在另一个子空间 S_2, 使得*

$$S = S_1 \oplus S_2.$$

例： 设 $A \in \mathbb{C}^{m \times n}$, 则

$$\mathbb{C}^n = \text{Ker}(A) \oplus \text{Ran}\,(A^*), \quad \mathbb{C}^m = \text{Ker}\,(A^*) \oplus \text{Ran}(A)$$

3.1.2 内积空间

- 内积, 内积空间, 欧氏空间, 酉空间
- 常见内积空间:
  - $C^n : (x, y) = y * x$
  - $R^n : (x, y) = y^T x$
  - $R^{m \times n} : (A, B) = tr(B^T A)$

3.1.3 正交与正交补

- 正交: 向量正交, 子空间正交
- 正交补空间

3.2 向量范数与矩阵范数

定义 1 *(向量范数) 若函数 $f : C^n \to R$ 满足*

(1) $f(x) \geq 0, \forall x \in C^n$, *等号当且仅当* $x = 0$ *时成立;*

(2) $f(x) = ||\cdot f(x), \forall x \in C^n, \in C$;

(3) $f(x + y) \leq f(x) + f(y), \forall x, y \in C^n$;

则称 f(x) 为 C^n 上的范数, 通常记作 $||\cdot||$

相类似地，我们可以定义实数空间 R^n 上的向量范数。

常见的向量范数：

- 1-范数:$||x||_1 = |x_1| + |x_2| + ... + |x_n|$
- 2-范数:$||x||_2 = \sqrt{|x_1|^2 + |x_2|^2 + ... + |x_n|^2}$
- ∞-范数:$\|x\|_\infty = \max_{1\le i\le n} |x_i|$
- p-范数:$\|x\|_p = (\sum_{i=1}^n |x_i|^p)^{1/p}, \quad 1 \le p < \infty$

定义 2 *(范数等价性)* $\mathbb{C}^n$ *上的向量范数* $|||\dot{|}||_\alpha$ *与* $|||\dot{|}||_\beta$ *等价: 存在正常数* c_1, c_2, *使得*

$$c_1\|x\|_\alpha \le \|x\|_\beta \le c_2\|x\|_\alpha, \quad \forall x \in \mathbb{C}^n$$

定理 3.2 $\mathbb{C}^n$ *空间上的所有向量范数都是等价的, 特别地, 有*

$$\|x\|_2 \le \|x\|_1 \le \sqrt{n}\|x\|_2$$

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

$$\|x\|_\infty \le \|x\|_1 \le n\|x\|_\infty$$

定理 3.3 *(Cauchy-Schwartz 不等式)* *设* $(.,.)$ *是* $\mathbb{C}^n$ *上的内积, 则对任意* $x, y \in \mathbb{C}^n$, *有*

$$|(x,y)|^2 \le (x,x)\cdot(y,y)$$

推论 1 *设* $(.,.)$ *是* $\mathbb{C}^n$ *上的内积,则* $||x|| \triangleq \sqrt{(x,x)}$ *是* $\mathbb{C}^n$ *上的一个向量范数*

定理 3.4 *设* $||.||$ *是* $\mathbb{C}^n$ *上的一个向量范数,则* $f(x) \triangleq ||x||$ *是* $\mathbb{C}^n$ *上的连续函数。*

3.2.1 矩阵范数

定义 3 *(矩阵范数) 若函数* $f : \mathbb{C}^{n\times n} \to R$ *满足* $(1) f(A) \ge 0, \forall A \in \mathbb{C}^{n\times n}$, *等号当且仅当* $A = 0$ *时成立;* $(2) f(A) = |\cdot| f(A), \forall A \in \mathbb{C}^{n\times n}, \in \mathbb{C}$; $(3) f(A+B) \le f(A) + f(B), \forall A, B \in \mathbb{C}^{n\times n}$; *则称* $f(x)$ *为* $\mathbb{C}^{n\times n}$ *上的范数, 通常记作* $||.||$。

相容的矩阵范数：f(AB) ≤ f(A)f(B), ∀ A, B ∈$\mathbb{C}^{n\times n}$。

若未明确指出, 讲义所涉及矩阵范数都指相容矩阵范数

引理 1 *设* $\|\cdot\|$ *是* C^n *上的向量范数, 则*

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

是 $\mathbb{C}^{n\times n}$ *上的范数, 称为算子范数, 或诱导范数, 导出范数。*

† 算子范数都是相容的, 且

$$\|Ax\| \leq \|A\| \cdot \|x\|, \quad A \in \mathbb{C}^{n\times n}, x \in \mathbb{C}^n$$

† 算子范数都是相容的, 且

$$\|Ax\| \leq \|A\| \cdot \|x\|, \quad A \in \mathbb{C}^{n\times n}, x \in \mathbb{C}^n$$

† 类似地, 我们可以定义 $\mathbb{C}^{m\times n}, \mathbb{R}^{n\times n}, \mathbb{R}^{m\times n}$ 上的矩阵范数.

引理 2 *可以证明:*

(1) 1-范数 (列范数):$\|A\|_1 = \max_{1\leq j\leq n}\left(\sum_{i=1}^n |a_{ij}|\right)$

(2) ∞-范数 (行范数): $\|A\|_\infty = \max_{1\leq i\leq n}\left(\sum_{j=1}^n |a_{ij}|\right)$

(3) 2-范数 (谱范数):$\|A\|_2 = \sqrt{\rho\left(A^\top A\right)}$

另一个常用范数 F-范数 $\|A\|_F = \sqrt{\sum_{i=1}^n \sum_{j=1}^n |a_{ij}|^2}$

定理 3.5 *(矩阵范数的等价性)* $\mathbb{R}^{n\times n}$ *空间上的所有范数都是等价的, 特别地, 有*

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

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

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

$$\frac{1}{\sqrt{n}}\|A\|_1 \leq \|A\|_F \leq \sqrt{n}\|A\|_2,$$

3.2.2 矩阵范数的一些性质

• 对任意的算子范数 $||.||$, 有 $||I|| = 1$ • 对任意的相容范数 $||.||$, 有 $||I|| \leq 1$ • F-范数是相容的, 但不是算子范数 • $||.||_2$ 和 $||.||_F$ 酉不变范数 • $\left\|A^\top\right\|_2 = \|A\|_2, \left\|A^\top\right\|_1 = \|A\|_\infty$ • 若 A 是正规矩阵, 则 $\|A\|_2 = \rho(A)$

3.2.3 向量序列的收敛

设 $x^{(k)}{}_{k=1}^{\inf}$ 是 $\mathbb{C}^n$ 中的一个向量序列, 如果存在 x∈$\mathbb{C}^n$, 使得

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

则称 $x^{(k)}$(按分量) 收敛到 x, 记为 $\lim_{k\to\infty} x^{(k)} = x$

定理 3.6 *(矩阵范数的等价性) 设* $||.||$ *是* $\mathbb{C}^n$ *上的任意一个向量范数, 则* $\lim_{k\to\infty} x^{(k)} = x$ *的充要条件是*

$$\lim_{k\to\infty} \left\|x^{(k)} - x\right\| = 0$$

3.2.4 收敛速度

设点列 $k_{k=1}^{\inf}$ 收敛, 且 $\lim_{k=\infty}\varepsilon_k = 0$. 若存在一个有界常数 $0 < c < \infty$, 使得 $\lim_{k\to\infty}\frac{|\varepsilon_{k+1}|}{|\varepsilon_k|^p} = c$ 则称点列 $_k$ 是 p 次 (渐进) 收敛的. 若 $1 < p < 2$ 或 $p = 1$ 且 $c = 0$, 则称点列是超线性收敛的.

† 类似地, 我们可以给出矩阵序列的收敛性和判别方法.

3.3 矩阵的投影

3.3.1 特征值与特征向量

- 特征多项式, 特征值, 特征向量, 左特征向量, 特征对
- n 阶矩阵 A 的谱: $(A)\square 1, 2, ..., n$
- 代数重数和几何重数, 特征空间
- 最小多项式
- 可对角化, 特征值分解
- 可对角化的充要条件
- 特征值估计: Bendixson 定理, 圆盘定理

3.3.2 Bendixson 定理

设 $A \in \mathbb{C}^{n\times n}$, 令 $H = \frac{1}{2}(A + A^*), S = \frac{1}{2}(A - A^*)$. 则有

$$\lambda_{\min}(H) \le \mathrm{Re}(\lambda(A)) \le \lambda_{\max}(H)$$
$$\lambda_{\min}(iS) \le \mathrm{Im}(\lambda(A)) \le \lambda_{\max}(iS)$$

其中 $Re(.)$ 和 $Im(.)$ 分别表示实部和虚部。

† 一个矩阵的特征值的实部的取值范围由其 Hermite 部分确定, 而虚部则由其 Skew-Hermite 部分确定.

3.3.3 Gerschgorin 圆盘定理

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

$$\mathcal{D}_i \triangleq \left\{ z \in \mathbb{C} : |z - a_{ii}| \le \sum_{j=1, j\neq i}^{n} |a_{ij}| \right\}, \quad i = 1, 2, \ldots, n$$

这就是 A 的 n 个 Gerschgorin 圆盘。

定理 3.7 *(Gerschgorin 圆盘定理) 设 $A = [a_{ij}] \in \mathbb{C}^{n\times n}$. 则 A 的所有特征值都包含在 A 的 Gerschgorin 圆盘的并集中, 即 $\sigma(A) \subset \bigcup_{i=1}^{n}\mathcal{D}_i$*

3.3.4 投影变换与投影矩阵

设 $S = S_1 \oplus S_2$, 则 S 中的任意向量 x 都可唯一表示为 $x = x_1 + x_2, x_1 \in S_1, x_2 \in S_2$. 我们称 x_1 为 x 沿 S_2 到 S_1 上的投影, 记为 $x|_{S_1}$. 设线性变换 $P : S \to S$. 如果对任意 $x \in S$, 都有 $Px = x|_{S_1}$, 则称 P 是从 S 沿 S_2 到 S_1 上的投影变换 (或投影算子), 对应的变换矩阵称为投影矩阵.

引理 3 *设 $P \in \mathbb{R}^{n\times n}$ 是一个投影矩阵, 则*

$$\mathbb{R}^n = \mathrm{Ran}(P) \oplus \mathrm{Ker}(P) \tag{1}$$

反之, 若 (1.3) 成立, 则 P 是沿 $Ker(P)$ 到 $Ran(P)$ 上的投影

投影矩阵由其像空间和零空间唯一确定.

引理 4 *若 S_1 和 S_2 是 $\mathbb{R}^n$ 的两个子空间, 且 $\mathbb{R}^n = S_1 \oplus S_2$, 则存在唯一的投影矩阵 P, 使得*

$$\mathrm{Ran}(P) = \mathcal{S}_1, \quad \mathrm{Ker}(P) = \mathcal{S}_2$$

3.3.5 投影矩阵的判别

定理 3.8 *矩阵 $P \in \mathbb{R}^{n\times n}$ 是投影矩阵的充要条件是 $P^2 = P$*

3.3.6 投影算子的矩阵表示

设 S_1 和 S_2 是 $\mathbb{R}^n$ 的两个 m 维子空间. 如果 $S_1 \oplus \mathcal{S}_2^{\perp} = \mathbb{R}^n$, 则存在唯一的投影矩阵 P, 使得

$$\mathrm{Ran}(P) = \mathcal{S}_1, \quad \mathrm{Ker}(P) = \mathcal{S}_2^{\perp}$$

此时, 我们称 P 是 S1 上与 S2 正交的投影矩阵, 且有

$$P = V\left(W^{\top}V\right)^{-1}W^{\top}$$

其中 $V = [v_1, v_2, ..., v_m]$ 和 $W = [w_1, w_2, ..., w_m]$ 的列向量组分别构成 S_1 和 S_2 的一组基.

3.3.7 正交投影

设 S_1 是内积空间 S 的一个子空间, $x \in S$, 则 x 可唯一分解成

$$x = x_1 + x_2, x_1 \in S_1, x_2 \in S_1^{\perp}$$

, 其中 x_1 称为 x 在 S_1 上的正交投影.

• 若 P 是沿 $S_1^{\perp}$ 到 S_1 上的投影变换, 则称 P 为 S_1 上的正交投影变换 (对应的矩阵为正交投影矩阵), 记为 P_{S_1}

• 如果 P 不是正交投影变换, 则称其为斜投影变换

定理 3.9 *投影矩阵 $P \in \mathbb{R}^{n\times n}$ 是正交投影矩阵的充要条件 $P = P$.*

定理 3.10 *投影矩阵 $P \in \mathbb{R}^{n\times n}$ 是正交投影矩阵的充要条件 $P^\top = P$.*

推论 2 *设 P 是子空间 $S1$ 上的正交投影变换. 令 $v_1, v_2, ..., v_m$ 是 S_1 的一组标准正交基, 则*

$$P = VV^\top$$

其中 $V = [v_1, v_2, ..., v_m]$.

性质 1 *设 $P \in \mathbb{R}^{n\times n}$ 是一个正交投影矩阵, 则*

$$||P||_2 = 1$$

且对 $\forall x \in \mathbb{R}^n$, 有

$$||x||_2^2 = ||Px||_2^2 + ||(I - P)x||_2^2$$

3.3.8 正交投影矩阵的一个重要应用

定理 3.11 *设 S_1 是 R_n 的一个子空间,$z \in R_n$ 是一个向量. 则最佳逼近问题*

$$\min_{x\in\mathcal{S}_1} \|x - z\|_2$$

的唯一解为

$$x_* = P_{\mathcal{S}_1} z$$

即 S_1 中距离 z 最近 (2-范数意义下) 的向量是 z 在 S_1 上的正交投影.

推论 3 *设矩阵 $A \in \mathbb{R}^{n\times n}$ 对称正定, 向量 $x* \in S_1 \subseteq R_n$. 则 x_* 是最佳逼近问题*

$$\min_{x\in\mathcal{S}_1} \|x - z\|_A$$

的解的充要条件是

$$A\,(x_* - z) \perp \mathcal{S}_1$$

这里 $\|x - z\|_A \triangleq \left\|A^{\frac{1}{2}}(x - z)\right\|_2$

3.3.9 不变子空间

设 $A \in \mathbb{R}^{n\times n}$, S 是 $\mathbb{R}^n$ 的一个子空间, 记

$$A\mathcal{S} \triangleq \{Ax : x \in \mathcal{S}\}$$

定义 4 *若 $AS \subseteq S$, 则称 S 为 A 的一个不变子空间.*

定理 3.12 *设 $x_1, x_2, ..., x_m$ 是 A 的一组线性无关特征向量, 则*

$$\text{span}\,\{x_1, x_2, \ldots, x_m\}$$

是 A 的一个 m 维不变子空间.

3.3.10 不变子空间的一个重要性质

定理 3.13 *设 $A\in\mathbb{R}^{n\times n}, X\in\mathbb{R}^{n\times k}$ 且 $rank(X)=k$. 则 $span(X)$ 是 A 的不变子空间的充要条件是存在 $B\in\mathbb{R}^{k\times k}$ 使得*

$$AX = XB,$$

此时, B 的特征值都是 A 的特征值.

推论 4 *设 $A\in\mathbb{R}^{n\times n}, X\in\mathbb{R}^{n\times k}$ 且 $rank(X)=k$. 若存在一个矩阵 $B\in\mathbb{R}^{k\times k}$ 使得 $AX=XB$, 则 $(,v)$ 是 B 的一个特征对当且仅当 $(,Xv)$ 是 A 的一个特征对.*

3.4 矩阵标准型

计算矩阵特征值的一个基本思想是通过相似变换, 将其转化成一个形式尽可能简单的矩阵, 使得其特征值更易于计算. 其中两个非常有用的特殊矩阵是 $Jordan$ 标准型和 $Schur$ 标准型.

定理 3.14 *设 $A\in\mathbb{C}^{n\times n}$ 有 p 个不同特征值, 则存在非奇异矩阵 $X\in\mathbb{C}^{n\times n}$, 使得*

$$X^{-1}AX = \begin{bmatrix} J_1 & & & \\ & J_2 & & \\ & & \ddots & \\ & & & J_p \end{bmatrix} \triangleq J$$

其中 J_i 的维数等于 $_i$ 的代数重数, 且具有下面的结构

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

这里 $_i$ 为 $_i$ 的几何重数, J_{ik} 称为 $Jordan$ 块, 每个 $Jordan$ 块对应一个特征向量

† Jordan 标准型在理论研究中非常有用, 但数值计算比较困难, 目前还没有找到十分稳定的数值算法.

推论 5 *所有可对角化矩阵组成的集合在所有矩阵组成的集合中是稠密的.*

3.4.1 Schur 标准型

定理 3.15 *设 $A \in \mathbb{C}^{n\times n}$, 则存在一个酉矩阵 $U\in\mathbb{C}^{n\times n}$ 使得*

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

其中 $\lambda_1, \lambda_2, \ldots, \lambda_n$ 是 A 的特征值 (排序任意).

关于 Schur 标准型的几点说明:

- Schur 标准型可以说是酉相似变化下的最简形式
- U 和 R 不唯一, R 的对角线元素可按任意顺序排列
- A 是正规矩阵当且仅当定理 (3.15) 中的 R 是对角矩阵;
- A 是 Hermite 矩阵当且仅当定理 (3.15) 中的 R 是实对角矩阵.

3.4.2 实 Schur 标准型

定理 3.16 *设 $A\in\mathbb{R}^{n\times n}$, 则存在正交矩阵 $Q\in\mathbb{R}^{n\times n}$, 使得*

$$Q^\top AQ = T$$

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

3.5 几类特殊矩阵

3.5.1 对称正定矩阵

设 $A\in\mathbb{C}^{n\times n}$.

A 是半正定 $\Longleftrightarrow \mathrm{Re}\,(x^*Ax) \geq 0, \forall x \in \mathbb{C}^n$

A 是正定 $\Longleftrightarrow \mathrm{Re}\,(x^*Ax) > 0, \forall x \in \mathbb{C}^n, x \neq 0$

A 是 Hermite 半正定 $\Longleftrightarrow$ A Hermite 且半正定

A 是 Hermite 正定 $\Longleftrightarrow$ A Hermite 且正定

† 正定和半正定矩阵不要求是对称或 Hermite 的

定理 3.17 *设 $A\in\mathbb{C}^{n\times n}$. 则 A 正定 (半正定) 的充要条件是矩阵 $H = \frac{1}{2}(A + A*)$ 正定 (半正定).*

定理 3.18 *设 $A\in\mathbb{R}^{n\times n}$. 则 A 正定 (或半正定) 的充要条件是对任意非零向量 $x\in\mathbb{R}^n$ 有 $x^\top Ax > 0$(或 $x^\top Ax\geq0$).*

3.5.2 矩阵平方根

定理 3.19 *设 $A \in \mathbb{C}^{n\times n}$ 是 Hermite 半正定, k 是正整数. 则存在唯一的 Hermite 半正定矩阵 $B \in \mathbb{C}^{n\times n}$ 使得*

$$B^k = A.$$

同时, 我们还有下面的性质: (1) $BA = AB$, 且存在一个多项式 $p(t)$ 使得 $B = p(A)$; (2) $rank(B) = rank(A)$, 因此, 若 A 是正定的, 则 B 也正定; (3) 如果 A 是实矩阵的, 则 B 也是实矩阵.

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

Hermite 正定矩阵与内积之间有下面的关系

定理 3.20 *设 $(\cdot,\cdot)$ 是 $\mathbb{C}^n$ 上的一个内积, 则存在一个 Hermite 正定矩阵 $A\mathbb{C}^{n\times n}$ 使得*

$$(x, y) = y^* Ax.$$

反之, 若 $A \in \mathbb{C}^{n\times n}$ 是 Hermite 正定矩阵, 则

$$f(x, y) \triangleq y^* Ax$$

是 $\mathbb{C}^n$ 上的一个内积.

† 上述性质在实数域中也成立.

3.5.3 对角占优矩阵

定义 5 *设 $A \in \mathbb{C}^{n\times n}$, 若*

$$|a_{ii}| \geq \sum_{j\neq i} |a_{ij}|$$

对所有 $i = 1, 2, ..., n$ 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. 若对所有 $i = 1, 2, ..., n$ 不等式都严格成立, 则称 A 是严格行对角占优. 通常简称为弱对角占优和严格对角占优.

† 类似地, 可以定义弱列对角占优和严格列对角占优.

3.5.4 可约与不可约

设 $A \in \mathbb{R}^{n\times n}$, 若存在置换矩阵 P, 使得 $PAP^\top$ 为块上三角, 即

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

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

定理 3.21 *设 $A \in \mathbb{R}^{n\times n}$, 指标集 $\mathbb{Z}_n = \{1, 2, \ldots, \ldots, n\}$. 则 $\boldsymbol{A}$ 可约的充要条件是存在非空指标集 $J \subset \mathbb{Z}_n$ 且 $J \neq \mathbb{Z}_n$, 使得*

$$a_{ij} = 0, \quad i \in J且j \in \mathbb{Z}_n \backslash J$$

这里 $\mathbb{Z}_n$ J 表示 $\boldsymbol{J}$ 在 $\mathbb{Z}_n$ 中的补集.

定理 3.22 *若 $A \in \mathbb{C}^{n\times n}$ 严格对角占优, 则 $\boldsymbol{A}$ 非奇异*

定理 3.23 *若 $A \in \mathbb{C}^{n\times n}$ 不可约对角占优, 则 $\boldsymbol{A}$ 非奇异*

3.5.5 其他常见特殊矩阵

• 带状矩阵: $a_{ij} \neq 0$ only if $-b_u \leq i - j \leq b_l$, 其中 b_u 和 b_l 为非负整数, 分别称为下带宽和上带宽, $b_u + b_l + 1$ 称为 A 的带宽

• 上 Hessenberg 矩阵: $a_{ij} = 0 \quad for \quad i-j > 1$,

$$\begin{bmatrix} * & * & * & \ldots & * \\ * & * & * & \ldots & * \\ & * & * & \ldots & * \\ & & \ddots & \ddots & \vdots \\ & & & * & * \end{bmatrix}$$

• 下 Hessenberg 矩阵

• Toeplitz 矩阵

$$T = \begin{bmatrix} t_0 & t_{-1} & \ldots & t_{-n+1} \\ t_1 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & t_{-1} \\ t_{n-1} & \cdots & t_1 & t_0 \end{bmatrix}$$

• 循环矩阵 (circulant):

$$C = \begin{bmatrix} c_0 & c_{n-1} & c_{n-2} & \cdots & c_1 \\ c_1 & c_0 & c_{n-1} & \cdots & c_2 \\ c_2 & c_1 & c_0 & \cdots & c_3 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ c_{n-1} & c_{n-2} & c_{n-3} & \cdots & c_0 \end{bmatrix}$$

• Hankel 矩阵:

$$H = \begin{bmatrix} h_0 & h_1 & \cdots & h_{n-2} & h_{n-1} \\ h_1 & \ddots & \ddots & \ddots & h_n \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ h_{n-2} & \cdots & \cdots & \cdots & h_{2n-2} \\ h_{n-1} & h_n & \dots & h_{2n-2} & h_{2n-1} \end{bmatrix}$$

3.6 Kronecker 积

定义 6 *设 $A \in \mathbb{C}^{m\times n}, B \in \mathbb{C}^{p\times q}$, 则 A 与 B 的 Kronecker 积定义为*

$$A \otimes B = \begin{bmatrix} a_{11}B & a_{12}B & \cdots & a_{1n}B \\ a_{21}B & a_{22}B & \cdots & a_{2n}B \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1}B & a_{m2}B & \cdots & a_{mn}B \end{bmatrix} \in \mathbb{C}^{mp\times nq}$$

Kronecker 积也称为直积, 或张量积.

† 任意两个矩阵都存在 Kronecker 积, 且 $A⊗B$ 和 $B⊗A$ 是同阶矩阵, 但通常 $A \otimes B \neq B \otimes A$

3.6.1 基本性质

(1) $(A)⊗B = A⊗(B) = (A⊗B), ∀∈\mathbb{C}$

(2) $(A⊗B)^{□} = A^{□}⊗B^{□}, (A⊗B)^* = A^*⊗B^*$

(3) $(A⊗B)⊗C = A⊗(B⊗C)$

(4) $(A+B)⊗C = A⊗C + B⊗C$

(5) $A⊗(B+C) = A⊗B + A⊗C$

(6) 混合积:$(A⊗B)(C⊗D) = (AC)⊗(BD)$

(7) $(A_1⊗A_2⊗\cdots⊗A_k)(B_1⊗B_2⊗\cdots⊗B_k) = (A_1B_1)⊗(A_2B_2)⊗\cdots⊗(A_kB_k)$

(8) $(A_1⊗B_1)(A_2⊗B_2)\cdots(A_k⊗B_k) = (A_1A_2\cdots A_k)⊗(B_1B_2\cdots B_k)$

(9) $rank(A⊗B) = rank(A)rank(B)$

定理 3.24 *设 $A∈\mathbb{C}^{m×m}, B∈\mathbb{C}^{n×n}$, 并设 $(,x)$ 和 $(,y)$ 分别是 A 和 B 的一个特征对, 则 $(,x⊗y)$ 是 $A⊗B$ 的一个特征对. 由此可知,$B⊗A$ 与 $A⊗B$ 具有相同的特征值.*

定理 3.25 *设 $A∈\mathbb{C}^{m×m}$, $B∈\mathbb{C}^{n×n}$, 则*

(1) $tr(A\otimes B)=tr(A)tr(B)$ *;*

(2) $det(A\otimes B)=det(A)^n det(B)^m$*;*

(3) $A\otimes I_n+I_m\otimes B$ *的特征值为* $_i+_j$ *, 其中* $_i$ *和* $_j$ *分别为* A *和* B *的特征值;*

(4) 若 A *和* B *都非奇异, 则* $(A\otimes B)-1=A^{-1}\otimes B^{-1}$*;*

推论 6 *设* $A=Q_1\Lambda_1Q_1^{-1}, B=Q_2\Lambda_2Q_2^{-1}$*, 则*

$$A\otimes B=(Q_1\otimes Q_2)(\Lambda_1\otimes\Lambda_2)(Q_1\otimes Q_2)^{-1}$$

定理 3.26 *设* $A\in\mathbb{C}^{m\times m}, B\in\mathbb{C}^{n\times n}$*, 则存在* $m+n$ *阶置换矩阵* P *使得*

$$P^\top(A\otimes B)P=B\otimes A$$

定理 3.27 *设矩阵* $X=[x1,x2,...,xn]\in\mathbb{R}^{m\times n}$*, 记* $vec(X)$ *为* X *按列拉成的* mn *维列向量, 即*

$$\mathrm{vec}(X)=\left[x_1^\top,x_2^\top,\ldots,x_N^\top\right]^\top$$

则有

$$vec(AX)=(I\otimes A)vec(X), vec(XB)=(B^\top\otimes I)vec(X),$$

以及

$$(A\otimes B)vec(X)=vec(BXA^\top)$$

4 线性方程组直接解法

线性方程组的求解方法

- 直接法: LU 分解, $Cholesky$ 分解, ...
- 迭代法: 古典迭代法, $Krylov$ 子空间迭代法

本章介绍直接法, 即 Gauss 消去法或 PLU 分解

直接法优点: 稳定可靠 → 在工程界很受欢迎

直接法缺点: 运算量大 $O(n^3)$ → 不适合大规模稀疏线性方程组 (针对特殊结构矩阵的快速方法除外)

4.1 Gauss 消去法和 LU 分解

4.1.1 LU 分解

4.1.2 LU 分解的实现

4.1.3 IKJ 型 LU 分解

4.1.4 待定系数法计算 LU 分解

4.1.5 三角方程求解

算法 4.1:	Gauss 消去法
1:	将 A 进行 LU 分解:
2:	$A = LU$, 其中 L 为单位下三角矩阵, U 为非奇异上三角矩阵;
3:	向前回代: 求解 $Ly = b$, 即得 $y = L^{-1}b$
4:	向后回代: 求解 $Ux = y$, 即得 $x = U^{-1}y = (LU)^{-1}b = A^{-1}b$.

4.1.6 选主元 LU 分解

4.1.7 矩阵求逆

4.1.1 LU 分解

考虑线性方程组

$$Ax = b \tag{2}$$

其中 $A \in \mathbb{R}^{n\times n}$ 非奇异, $b \in \mathbb{R}^n$ 为给定的右端项.

Gauss 消去法本质上就是对系数矩阵 A 进行 LU 分解:

$$A = LU \tag{3}$$

其中 L 是单位下三角矩阵, U 为非奇异上三角矩阵.

分解 () 就称为 LU 分解

$$Ax = b \Longleftrightarrow \begin{cases} Ly = b \\ Ux = y \end{cases} \mid \Longrightarrow \text{只需求解两个三角方程组} \tag{4}$$

† 需要指出的是: A 非奇异, 则解存在唯一, 但并不一定存在 LU 分解!

定理 4.1 *(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, $k = 1, 2, ..., n$.*

4.1.2 LU 分解的实现—矩阵初等变换

给定一个矩阵

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & & \ddots & \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} \in \mathbb{R}^{n\times n}$$

- 第一步: 假定 $a_{11} \neq 0$, 构造矩阵

$$L_1 = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ l_{21} & 1 & 0 & \cdots & 0 \\ l_{31} & 0 & 1 & \cdots & 0 \\ \vdots & & & \ddots & \\ l_{n1} & 0 & 0 & \cdots & 1 \end{bmatrix}$$, 其中 $l_{i1} = \frac{a_{i1}}{a_{11}}, i = 2, 3, \ldots, n$

易知 L_1 的逆为

$$L_1^{-1} = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ -l_{21} & 1 & 0 & \cdots & 0 \\ -l_{31} & 0 & 1 & \cdots & 0 \\ \vdots & & \ddots & & \\ -l_{n1} & 0 & 0 & \cdots & 1 \end{bmatrix}$$

用 L_1^{-1} 左乘 A, 并将所得到的矩阵记为 $A(1)$, 则

$$A^{(1)} = L_1^{-1}A \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ 0 & a_{22}^{(1)} & \cdots & a_{2n}^{(1)} \\ \vdots & \vdots & \ddots & \\ 0 & a_{n2}^{(1)} & \cdots & a_{nn}^{(1)} \end{bmatrix}$$

即左乘 L_1^{-1} 后, A 的第一列中除第一个元素外其它都变为 0.

• 第二步: 将上面的操作作用在 $A^{(1)}$ 的子矩阵 $A^{(1)}(2:n, 2:n)$ 上, 将其第一列除第一个元素外都变为 0 : 假定 $a_{22}^{(1)} \neq 0$, 构造矩阵

$$L_2 = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & l_{32} & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \\ 0 & l_{n2} & 0 & \cdots & 1 \end{bmatrix}$$, 其中 $l_{i2} = \frac{a_{i2}^{(1)}}{a_{22}^{(1)}}, i = 3, 4, \ldots, n$

用 L_2^{-1} 左乘 $A^{(1)}$, 并将所得到的矩阵记为 $A^{(2)}$, 则

$$A^{(2)} = L_2^{-1}A = L_2^{-1}L_1^{-1}A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & \ldots & a_{1n} \\ 0 & a_{22}^{(1)} & a_{23}^{(1)} & \ldots & a_{2n}^{(1)} \\ 0 & 0 & a_{33}^{(2)} & \ldots & a_{3n}^{(2)} \\ \vdots & \vdots & \vdots & \ddots & \\ 0 & 0 & a_{n3}^{(2)} & \cdots & a_{nn}^{(2)} \end{bmatrix}$$

依此类推, 假定 $a_{kk}^{(k-1)} \neq 0 (k = 3, 4, \ldots, n-1)$, 则我们可以构造一系列的矩阵

算法 4.2: LU 分解

```
for k = 1 to n − 1 do
    for i = k + 1 to n do
        l_ik = a_ik/a_kk % 计算 L 的第 k 列
    end for
    for j = k to n do
        u_kj = a_kj % 计算 U 的第 k 行
    end for
    end for i = k + 1 to n do
        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
```

$L_3, L_4, ..., L_{n-1}$, 使得

$$L_{n-1}^{-1} \cdots L_2^{-1} L_1^{-1} A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ 0 & a_{22}^{(1)} & a_{23}^{(1)} & \dots & a_{2n}^{(1)} \\ 0 & 0 & a_{33}^{(2)} & \cdots & a_{3n}^{(2)} \\ \vdots & \vdots & \vdots & \ddots & \\ 0 & 0 & 0 & \cdots & a_{nn}^{(n-1)} \end{bmatrix} \triangleq U \to \text{上三角}$$

于是可得 $A = LU$ 其中

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

Gauss 消去法的运算量

由算法 4.2 可知, LU 分解的运算量 (加减乘除) 为

$$\sum_{i=1}^{n-1} \left(\sum_{j=i+1}^{n} 1 + \sum_{j=i+1}^{n} \sum_{k=i+1}^{n} 2 \right) = \sum_{i=1}^{n-1} \left(n - i + 2(n-i)^2 \right) = \frac{2}{3} n^3 + O\left(n^2\right)$$

加上回代过程的运算量 $O\left(n^2\right)$, 总运算量为 $\frac{2}{3} n^3 + O\left(n^2\right)$

† 评价算法的一个主要指标是执行时间, 但这依赖于计算机硬件和编程技巧等, 因此直接给出算法执行时间是不太现实的. 所以我们通常是统计算法中算术运算 (加减乘除) 的次数.

† 在数值算法中, 大多仅仅涉及加减乘除和开方运算. 一般地, 加减运算次数与乘法运算次数具有相同的量级, 而除法运算和开方运算次数具有更低的量级.

† 为了尽可能地减少运算量, 在实际计算中, 数, 向量和矩阵做乘法运算时的先后执行次序为: 先计算数与向量的乘法, 然后计算矩阵与向量的乘法, 最后才计算矩阵与矩阵的乘法.

矩阵 L 和 U 的存储

当 A 的第 i 列被用于计算 L 的第 i 列后, 在后面的计算中不再被使用.

同样地, A 的第 i 行被用于计算 U 的第 i 行后, 在后面计算中也不再使用

为了节省存储空间, 在计算过程中将 L 的第 i 列存放在 A 的第 i 列, 将 U 的第 i 行存放在 A 的第 i 行, 这样就不需要另外分配空间存储 L 和 U.

计算结束后, A 的上三角部分为 U, 其绝对下三角部分为 L 的绝对下三角部分 (L 的对角线全部为 1, 不需要存储).

算法 4.3: LU 分解

```
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
```

† 根据指标的循环次序, 算法 4.3 也称为 KIJ 型 LU 分解. 实际计算中一般不建议使用: 对指标 k 的每次循环, 都需要更新 A 的第 $k+1$ 至第 n 行, 这种反复读取数据的做法会使得计算效率大大降低. 对于按行存储的数据结构, 一般采用后面介绍的 IKJ 型 LU 分解.

```
% Matlab code 1 : LU 分解
function A = mylu(A)
n=size(A,1);
for k=1:n-1
if A(k,k) == 0
fprintf('Error: A(%d,%d)=0!\n', k, k);
return;
end
for i=k+1:n
A(i,k)=A(i,k)/A(k,k);
for j=k+1:n
A(i,j)=A(i,j)-A(i,k)*A(k,j);
end
end
end
```

为了充分利用 Matlab 的向量运算优势, 提高运算效率, 程序可改写为

```
% Matlab code 2 : LU 分解
function A = mylu(A)
n=size(A,1);
for k=1:n-1
if A(k,k) == 0
fprintf('Error: A(%d,%d)=0!\n', k, k);
return;
end
A(k+1:n,k)=A(k+1:n,k)/A(k,k);
A(k+1:n,k+1:n)=A(k+1:n,k+1:n)-A(k+1:n,k)*A(k,k+1:n);
end
```

算法 4.4: LU 分解	
1:	for i = 2 to n do
2:	for k = 1 to i − 1 do
3:	$a_{ik} = a_{ik}/a_{kk}$
4:	for j = k + 1 to n do
5:	$a_{ij} = a_{ij} - a_{ik}a_{kj}$
6:	end for
7:	end for
8:	end for

4.1.3 IKJ 型 LU 分解

如果数据是按行存储的, 如 C/C++, 我们一般采用下面的 IKJ 型 LU 分解.

上述算法可以用下图来描述.

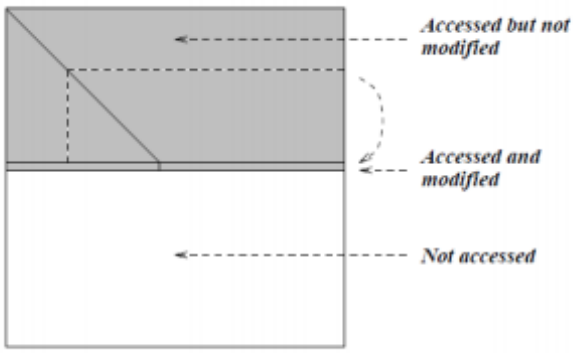


图 1

思考:如果数据按列存储, 如 FORTRAN/MATLAB, 如何设计算法?

4.1.4 待定系数法计算 LU 分解

设 $A = LU$, 即

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \cdots & a_{3n} \\ \vdots & & & \ddots & \vdots \\ a_{n1} & a_{x2} & a_{x3} & \cdots & a_{nn} \end{bmatrix} = \begin{bmatrix} 1 & & & & \\ l_{21} & 1 & & & \\ l_{31} & l_{32} & 1 & & \\ \vdots & & & \ddots & \\ l_{n1} & l_{n2} & \cdots & l_{n,n-1} & \end{bmatrix} \begin{bmatrix} u_{11} & u_{12} & u_{13} & \cdots & u_{1n} \\ & u_{22} & u_{23} & \cdots & u_{2n} \\ & & u_{33} & \cdots & u_{2n} \\ & & & \ddots & \vdots \\ & & & & u_{nni} \end{bmatrix}$$

(1) 比较等式两边的第一行, 可得

算法 4.5 LU 分解 (待定系数法或 *Doolittle* 方法)	
1：	for k = 1 to n do
2：	$a_{kj} = a_{kj} - \sum_{i=1}^{k-1} a_{ki}a_{ij}, \quad j = k, k+1, \ldots, n$
3：	$a_{ik} = \frac{1}{a_{kk}} \left(a_{ik} - \sum_{j=1}^{k-1} a_{ij}a_{jk} \right), \quad i = k+1, k+2, \ldots, n$
4：	end for

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

再比较等式两边的第一列, 可得

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

(2) 比较等式两边的第二行, 可得

$a_{2j} = l_{21}u_{1j} + a_{2j} \Rightarrow u_{2j} = a_{2j} - l_{21}u_{1j}, \quad j = 2, 3, \ldots, n$

再比较等式两边的第二列, 可得

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

(3) 以此类推, 第 k 步时, 比较等式两边的第 k 行, 可得

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

比较等式两边的第 k 列, 可得

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

直到第 n 步, 即可计算出 L 和 U 的所有元素.

同样, 我们可以利用 A 来存储 L 和 U. 算法描述如下:

```
% Matlab code 2 : 待定系数法 LU 分解
function A = mylu2(A)
[n,n]=size(A);
for k=1:n
A(k,k)=A(k,k)-A(k,1:k-1)*A(1:k-1,k);
if (A(k,k)==0)
fprintf('Error: A(%d,%d)=0!\n', i,i);
return;
end
A(k,k+1:n)=A(k,k+1:n)-A(k,1:k-1)*A(1:k-1,k+1:n);
A(k+1:n,k)=A(k+1:n,k)-A(k+1:n,1:k-1)*A(1:k-1,k);
A(k+1:n,k)=A(k+1:n,k)/A(k,k);
end
```

4.1.5 三角方程求解

得到 A 的 LU 分解后, 我们最后需要用回代法求解两个三角方程组

$$Ly = b, \quad Ux = y$$

如果数据是按列存储的, 则采用列存储方式效率会高一些.

算法 4.6	向前回代求解 $Ly = b$(假定 L 是一般的非奇异下三角矩阵)
1：	$y_1 = b_1/l_{11}$
2：	for i = 2 : n do
3：	for j = 1 : i − 1 do
4:	$b_i = b_i - l_{ij}y_j$
5：	end for
6:	$y_i = b_i/l_{ii}$
7:	end for

算法 4.7	向前回代求解 $Ly = b$(假定 L 是一般的非奇异下三角矩阵)
1：	for i = n : −1 : 1 do
2：	$x_i = y_i/u_{ii}$
3：	for j = i − 1 : −1 : 1 do
4:	$y_j = y_j - x_i u_{ji}$
5：	end for
6:	end for

下面是按列存储方式求解上三角方程组.

这两个算法的运算量均为 $n^2 + O(n)$

4.1.6 选主元 LU 分解

• 在 LU 分解算法 1.2 中, 我们称 $a_{kk}^{(k-1)}$ 为主元. 如果 $a_{kk}^{(k-1)} = 0$, 则算法就无法进行下去.

• 即使 $a_{kk}^{(k-1)}$ 不为零, 但如果 $\left|a_{kk}^{(k-1)}\right|$ 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.

• 此时我们就需要通过选主元来解决这个问题.

例 用 LU 分解求解线性方程组 Ax = b, 其中

$$A = \begin{bmatrix} 0.02 & 61.3 \\ 3.43 & -8.5 \end{bmatrix}, \quad b = \begin{bmatrix} 61.5 \\ 25.8 \end{bmatrix} \tag{5}$$

要求在运算过程中保留 3 位有效数字.

$(x_1 \approx -20.7, x_2 \approx 1.01)$

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. 所以我们需要选主元.

算法 4.7　向前回代求解 $Ly = b$(假定 L 是一般的非奇异下三角矩阵)

```
p = 1 : n % 用于记录置换矩阵
for k = 1 to n − 1 do
    [a_max, l] = max_{k≤i≤n} |a_ik| % 选列主元, 其中 l 表示主元所在的行
    if l ≠ k then
        for j = 1 to n do
            tmp = a_kj, a_kj = a_lj, a_lj = tm p % 交换第 k 行与第 l 行
        end for
        tm p = p(k), p(k) = p(l), p(l) = tm p % 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 k 列
    end for
    for i = k + 1 to n do
        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
    end for
end for
```

4.2　选主元 LU 分解

定理 4.2 *设 $A \in R^{n\times n}$ 非奇异, 则存在置换矩阵 P_L,P_R, 以及单位下三角矩阵 L 和非奇异上三角矩阵 U, 使得 $P_L A P_R = LU$. 其中 P_L 和 P_R 中只有一个是必需的.*

第 k 步时, 如何选取置换矩阵 $P_L^{(k)}$ 和 $P_R^{(k)}$?

选法一. 选取 $P_L^{(k)}$ 和 $P_R^{(k)}$ 使得主元为剩下的矩阵中绝对值最大, 这种选取方法称为“全主元 $Gauss$ 消去法”, 简称 $GECP$ (Gaussian elimination with complete pivoting);

选法二. 选取 $P_L^{(k)}$ 和 $P_R^{(k)}$ 使得主元为第 k 列中第 k 到第 n 个元素中, 绝对值最大, 这种选取方法称为“部分选主元 $Gauss$ 消去法”, 简称 $GEPP$ (Gaussian elimination with partial pivoting), 此时 $P_R^{(k)} = I$, 因此也称为列主元 $Gauss$ 消去法.

† (1) $GECP$ 比 $GEPP$ 更稳定, 但工作量太大, 在实际应用中通常使用 $GEPP$ 算法.

(2) $GEPP$ 算法能保证 L 所有的元素的绝对值都不超过 1.

```
% Matlab code 2 : 部分选主元 LU 分解
function [A,p] = myplu(A)
```

```
[n,n]=size(A); p=1:n;
for i=1:n-1
[a,k]=max(abs(A(i:n,i)));
if a==0
error('Error: 第 %d 步的列主元为 0!\n', i);
end
k=k+i-1;
if k~=i
tmp=A(i,:); A(i,:)=A(k,:); A(k,:)=tmp;
tmp=p(i); p(i)=p(k); p(k)=tmp;
end
A(i+1:n,i)=A(i+1:n,i)/A(i,i);
A(i+1:n,i+1:n)=A(i+1:n,i+1:n)-A(i+1:n,i)*A(i,i+1:n);
end
```

例 用 LU 分解求解线性方程组 Ax = b, 其中

$$A = \begin{bmatrix} 0.02 & 61.3 \\ 3.43 & -8.5 \end{bmatrix}, \quad b = \begin{bmatrix} 61.5 \\ 25.8 \end{bmatrix} \tag{6}$$

要求在运算过程中保留 3 位有效数字.

$(x_1 \approx 10.0, x_2 \approx 0.998)$

4.2.1 矩阵求逆

我们可以通过部分选主元 LU 分解来计算矩阵的逆. 设 $PA = LU$, 则

$$A^{-1} = P^T U^{-1} L^{-1}$$

等价于求解下面 $2n$ 个三角线性方程组

$$Ly_i = Pe_i, \quad Ux_i = y_i, \quad i = 1, 2, \ldots, n$$

也可以分别计算 L^{-1} 和 U^{-1}, 然后相乘. 哪种方法划算?

4.3 特殊方程组的求解

2.1 对称正定线性方程组

2.2 对称不定线性方程组

2.3 三对角线性方程组

2.4 带状线性方程组

2.5 Toeplitz 线性方程组

4.3.1 对称正定线性方程组

我们首先给出对称正定矩阵的几个基本性质.

定理 4.3 *设 $A\in\mathbb{R}^{n\times n}$.* • *$A$ 对称正定当且仅当 A 对称且所有特征值都是正的;*

• *A 对称正定当且仅当 $X^{\top}AX$ 对称正定, 其中 $X\in\mathbb{R}^{n\times n}$ 是一个任意的非奇异矩阵;*

• *若 A 对称正定, 则 A 的任意主子矩阵都对称正定;*

• *若 A 对称正定, 则 A 的所有对角线元素都是正的, 且*

$$\max_{i\neq j}\{|a_{ij}|\}<\max_{i}\{a_{ii}\},$$

即绝对值最大的元素出现在对角线上.

Cholesky 分解

定理 4.4 *(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为正的下三角矩阵 L, 使得*

$$A=LL^{\top}$$

该分解称为 Cholesky 分解.

Cholesky 分解的实现 设 $A=LL^{\top}$, 即

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n}\\ a_{21} & a_{22} & \cdots & a_{2n}\\ \vdots & & \ddots & \vdots\\ a_{n1} & a_{n2} & \cdots & a_{nn}\end{bmatrix}=\begin{bmatrix} l_{11} & & & \\ l_{21} & l_{22} & & \\ \vdots & \vdots & \ddots & \\ l_{n1} & l_{n2} & \cdots & l_{nn}\end{bmatrix}\begin{bmatrix} l_{11} & l_{21} & \cdots & l_{n1}\\ & l_{22} & \cdots & l_{n2}\\ & & \ddots & \vdots\\ & & & l_{nn}\end{bmatrix} \tag{7}$$

直接比较等式两边的元素可得

$$a_{ij}=\sum_{k=1}^{n}l_{ik}l_{jk}=l_{jj}l_{ij}+\sum_{k=1}^{j-1}l_{ik}l_{jk},\quad i,j=1,2,\ldots,n \tag{8}$$

根据上面的计算公式, 可得下面的算法:

几点说明

• 与 LU 分解一样, 可以利用 A 的下三角部分来存储 L;

• $Cholesky$ 分解算法的运算量为 $\frac{1}{3}n^3+O(n^2)$, 大约为 LU 分解的一半;

• $Cholesky$ 分解算法是稳定的(稳定性与全主元 $Gauss$ 消去法相当), 故不需要选主元.

算法 4.8	Cholesky 分解算法
1：	for j = 1 to n do
2：	$l_{jj} = \left(a_{jj} - \sum_{k=1}^{j-1} l_{jk}^2\right)^{1/2}$
3：	for i = j + 1 to n do
4:	$l_{ij} = \left(a_{ij} - \sum_{k=1}^{j-1} l_{ik} l_{jk}\right) / l_{jj}$
5：	end for
6:	end for

改进的 Cholesky 分解算法 为了避免开方运算, 我们可以将 A 分解为: $A = LDL^\top$, 即

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} = \begin{bmatrix} 1 & & & \\ l_{21} & 1 & & \\ \vdots & & \ddots & \\ l_{n1} & \cdots & l_{n,n-1} & 1 \end{bmatrix} \begin{bmatrix} d_1 & & & \\ & d_2 & & \\ & & \ddots & \\ & & & d_n \end{bmatrix} \begin{bmatrix} 1 & l_{21} & \cdots & l_{n1} \\ & 1 & \cdots & l_{n2} \\ & & \ddots & \vdots \\ & & & 1 \end{bmatrix} \tag{9}$$

通过待定系数法可得

$$a_{ij} = \sum_{k=1}^{n} l_{ik} d_k l_{jk} = d_j l_{ij} + \sum_{k=1}^{j-1} l_{ik} d_k l_{jk}, \quad i, j = 1, 2, \ldots, n \tag{10}$$

基于以上分解来求解对称正定线性方程组的算法称为改进的平方根法:

4.3.2 对称不定线性方程组

$A\rightarrow$ 非奇异, 对称不定

若 A 存在 LU 分解, 即 $A = LU$, 则可写成 $A = LDL^\top$

然而, 当 A 不定时, 其 LU 分解不一定存在.

若采用选主元 LU 分解, 则其对称性将被破坏. 为了保持对称性, 在选主元时必须对行列进行同样的置换, 即选取置换矩阵 P, 使得

$$PAP^\top = LDL^\top \tag{11}$$

通常称 (2.4) 为对称矩阵的 $LDL^\top$ 分解.

不幸的是, 这样的置换矩阵可能不一定存在, 即分解 (2.4) 不一定存在.

算法 4.9	改进的平方根法
1：	for j = 1 to n do % 先计算分解
2：	$d_j = a_{jj} - \sum_{k=1}^{j-1} l_{jk}^2 d_k$
3：	for i = j + 1 to n do
4:	$l_{ij} = \left(a_{ij} - \sum_{k=1}^{j-1} l_{ik} d_k l_{jk}\right) / d_j$
5：	end for
6:	end for
7:	$y_1 = b_1$ % 解方程组: Ly = b 和 $DL^\top x = y$
8：	for i = 2 to n do
9：	$y_i = b_i - \sum_{k=1}^{i-1} l_{ik} y_k$
10：	end for
11：	$x_n = y_n / d_n$
12：	for i = n − 1 to 1 do
13：	$x_i = y_i / d_i - \sum_{k=i+1}^{n} l_{ki} x_k$
14：	end for

例 设对称矩阵

$$A = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix} \tag{12}$$

由于 A 的对角线元素都是 0, 对任意置换矩阵 P, 矩阵 $PAP^\top$ 的对角线元素仍然都是 0. 因此, 矩阵 A 不存在 $LDL^\top$ 分解.

Aasen 算法

1971 年, Aasen 提出了下面的分解

$$PAP^\top = LTL^\top \tag{13}$$

其中 P 为置换矩阵, L 为单位下三角矩阵, T 为对称三对角矩阵.

分解 (2.5) 本质上与部分选主元 LU 分解是一样的.

块 $LDL^\top$ 分解

设 A 对称非奇异, 则存在置换矩阵 P 使得

$$PAP^\top = \begin{bmatrix} B & E^\top \\ E & C \end{bmatrix} \tag{14}$$

其中 $B \in \mathbb{R}$ 或 $B \in \mathbb{R}^{2\times 2}$, 且非奇异. 因此可以对 $PAP^\top$ 进行块对角化, 即

$$PAP^\top = \begin{bmatrix} I & 0 \\ EB^{-1} & I \end{bmatrix} \begin{bmatrix} B & 0 \\ 0 & C - EB^{-1}E^\top \end{bmatrix} \begin{bmatrix} I & B^{-1}E^\top \\ 0 & I \end{bmatrix} \tag{15}$$

其中 $C - EB^{-1}E^\top$ 是 $Schur$ 补。

不断重复以上过程, 就可以得到 A 的块 $LDL^\top$ 分解:

$$PAP^\top = L\tilde{D}L^\top \tag{16}$$

其中 $\tilde{D}$ 是拟对角矩阵, 即块对角矩阵且对角块的大小为 1 或 2

选主元块 $LDL^\top$ 分解 与选主元 LU 分解类似, 我们需要考虑块 $LDL^\top$ 分解的选主元策略, 即如何选取置换矩阵. 目前常用的策略有

• 全主元策略: 由 $Bunch$ 和 $Parlett$ 于 1971 年提出, 并证明了其稳定性. 但需要进行 $n^3/6$ 次比较运算, 代价比较昂贵.

• 部分选主元策略: 由 $Bunch$ 和 $Kaufman$ 于 1977 年提出, 将比较运算复杂度降低到 $O(n^2)$ 量级, 而且具有较满意的向后稳定性. 因此被广泛使用.

• Rook 策略: 由 Ashcraft, Grimes 和 Lewis 于 1998 年提出, 整体上与部分选主元类似, 但在选主元时加了一层迭代, 从而能提供更高的精度

目前大部分软件都采用部分选主元块 $LDL^\top$ 分解算法

4.3.3 三对角线性方程组

$$A = \begin{bmatrix} b_1 & c_1 & & \\ a_1 & \ddots & \ddots & \\ & \ddots & \ddots & c_{n-1} \\ & & a_{n-1} & b_n \end{bmatrix} \tag{17}$$

我们假定

$$|b_1| > |c_1| > 0, \quad |b_n| > |a_{n-1}| > 0 \tag{18}$$

$$|b_i| \geq |a_{i-1}| + |c_i|, \quad a_i c_i \neq 0, \quad i = 1, \ldots, n-1 \tag{19}$$

即 A 是不可约弱对角占优

如果 A 可约, 怎么处理?

此时, 我们可以得到下面的三角分解

$$A=\begin{bmatrix} b_1 & c_1 & & \\ a_1 & \ddots & \ddots & \\ & \ddots & \ddots & c_{n-1} \\ & & a_{n-1} & b_n \end{bmatrix}=\begin{bmatrix} \alpha_1 & & & \\ a_1 & \alpha_2 & & \\ & \ddots & \ddots & \\ & & a_{n-1} & \alpha_n \end{bmatrix}\begin{bmatrix} 1\beta_1 & & & \\ 1 & \ddots & & \\ & \ddots & \beta_{n-1} & \\ & & 1 & \end{bmatrix}\triangleq LU \tag{20}$$

递推公式：

$$\begin{aligned} &\alpha_1=b_1 \\ &\beta_1=c_1/\alpha_1=c_1/b_1 \\ &\begin{cases} \alpha_i=b_i-a_{i-1}\beta_{i-1} \\ \beta_i=c_i/\alpha_i=c_i/\left(b_i-a_{i-1}\beta_{i-1}\right), \quad i=2,3,\ldots,n-1 \end{cases} \\ &\alpha_n=b_n-a_{n-1}\beta_{n-1} \end{aligned} \tag{21}$$

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 4.5 *设三对角矩阵 A 满足条件 (2.6) 和 (2.7). 则 A 非奇异, 且*

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

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

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

† 追赶法 (也称为 Thomas 算法) 的运算量大约为 $8n-6$.

† 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

$$A=\begin{bmatrix} b_1 & c_1 & & \\ a_1 & \ddots & \ddots & \\ & \ddots & \ddots & c_{n-1} \\ & & a_{n-1} & b_n \end{bmatrix}=\begin{bmatrix} 1 & & & \\ \gamma_1 & 1 & & \\ & \ddots & \ddots & \\ & & \gamma_{n-1} & 1 \end{bmatrix}\begin{bmatrix} \alpha_1 & c_1 & & \\ & \alpha_2 & \ddots & \\ & & \ddots & c_{n-1} \\ & & & \alpha_n \end{bmatrix} \tag{22}$$

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 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, 则采用前面的分解;

如果 A 是列对角占优, 则采用分解 (2.11).

算法 4.9	改进的平方根法
1：	$\beta_1 = c_1/b_1$
2：	$y_1 = f_1/b_1$
3：	: for i = 2 to n−1 do
4:	$\alpha_i = b_i - a_{i-1}\beta_{i-1}$
5：	$\beta_i = c_i/\alpha_i$
6:	$y_i = (f_i - a_{i-1}y_{i-1})/\alpha_i$
7:	end for
8：	$\alpha_n = b_n - a_{n-1}\beta_{n-1}$
9：	$y_n = (f_n - a_{n-1}y_{n-1})/\alpha_n$
10：	$x_n = y_n$
11：	for i = n−1 to 1 do
12：	$x_i = y_i - \beta_i x_{i+1}$
13：	end for

4.3.4 带状线性方程组

设 $A \in R^{n\times n}$ 是带状矩阵, 其下带宽为 b_L, 上带宽为 b_U , 即

$$a_{ij} = 0 \quad \text{for} \quad i > j + b_L \text{ or } i < j - b_U \tag{23}$$

对于带状矩阵, 其 LU 分解有如下性质:

定理 4.6 *设 $A \in R^{n\times n}$ 是带状矩阵, 其下带宽为 b_L, 上带宽为 b_U . 若 $A = L_U$ 是不选主元的 L_U 分解, 则 L 为下带宽为 b_L 的带状矩阵, U 为上带宽为 b_U 的带状矩阵.*

统计求解带状矩阵 $Ax = b$ 的运算量.

若采用部分选主元的 LU 分解, 则有

定理 4.7 *设 $A \in Rn\times n$ 是带状矩阵, 其下带宽为 b_L, 上带宽为 b_U . 若 $PA = LU$ 是部分选主元的 LU 分解, 则 U 为上带宽不超过 $b_L + b_U$ 的带状矩阵, L 为下带宽为 b_L 的"基本带状矩阵", 即 L 每列的非零元素不超过 $b_L + 1$ 个.*

4.3.5 Toeplitz 线性方程组

$$T_n = \begin{bmatrix} t_0 & t_{-1} & \cdots & t_{-n+1} \\ t_1 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & t_{-1} \\ t_{n-1} & \cdots & t_1 & t_0 \end{bmatrix} \tag{24}$$

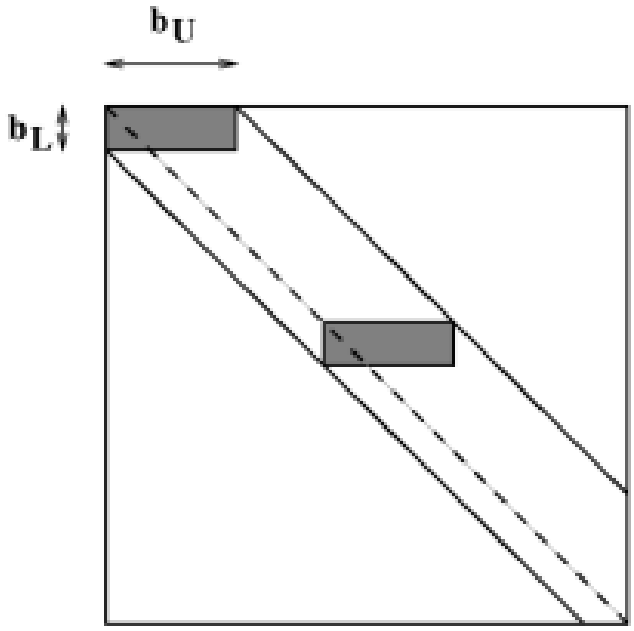


图 2

T_n 是反向对称 (persymmetric) 矩阵. 记 J_n 为 n 阶反向单位矩阵:

$$J_n = \begin{bmatrix} & & & 1 \\ & & 1 & \\ & \cdots & & \\ 1 & & & \end{bmatrix} \tag{25}$$

易知 $J_n^\top = J_n^{-1} = J_n$

引理 5 *矩阵 $A \in \mathbb{R}^{n\times n}$ 是反向对称矩阵当且仅当*

$$A = J_n A^\top J_n \quad \text{或} \quad J_n A = A^\top J_n \tag{26}$$

若 A 可逆, 则可得

$$A^{-1} = J_n^{-1}\left(A^\top\right)^{-1} J_n^{-1} = J_n\left(A^{-1}\right)^\top J_n \tag{27}$$

即反向对称矩阵的逆也是反向对称矩阵.

† Toeplitz 矩阵的逆是反向对称矩阵, 但不一定是 Toeplitz 矩阵.

Yule-Walker 方程组

假定 T_n 对称正定, 考虑线性方程组

$$T_n x = -r_n \tag{28}$$

其中 $r_n = [t_1, t_2, ..., t_{n-1}, t_n]^\top$. 这类线性方程组称为 $Yule - Walker$ 方程组, 其中 t_n 为任意给定的实数.

由于 T_n 对称正定, 所以 $t_0 > 0$. 因此我们可以对 T_n 的对角线元素进行单位化. 不失一般性, 我们假定 T_n 的对角线元素为 1, 即

$$T_n = \begin{bmatrix} 1 & t_1 & \cdots & t_{n-1} \\ t_1 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & t_1 \\ t_{n-1} & \cdots & t_1 & 1 \end{bmatrix} \tag{29}$$

由于方程组右端项的特殊性, 我们可以通过递推来求解.

记 $T_k x = -r_k$ 的解为 $\mathcal{X}^{(k)}$. 设 $T_{k+1} x = -r_{k+1}$ 的解 $x^{(k+1)} = \begin{bmatrix} z^{(k)} \\ \alpha_k \end{bmatrix}$。

代入后可得递推公式：

$$\alpha_k = \frac{-t_{k+1} - r_k^\top J_k x^{(k)}}{1 + r_k^\top x^{(k)}}, \quad z^{(k)} = x^{(k)} + \alpha_k J_k x^{(k)} \quad k = 1, 2, \ldots \tag{30}$$

因此, 我们就可以从一阶 $Yule - Walker$ 方程出发, 利用递推公式 (2.13) 计算 $T_n x = -r_n$ 的解. 总的运算量 (乘法与加减运算) 大约为 $3n^2$.

为了减少运算量, 我们引入一个变量 $\beta_k \triangleq 1 + r_k^\top x^{(k)}$, 则：

$$\begin{aligned} \beta_{k+1} &= 1 + r_{k+1}^\top x^{(k+1)} \\ &= 1 + \left[r_k^\top, t_{k+1}\right] \begin{bmatrix} x^{(k)} + \alpha_k J_k x^{(k)} \\ \alpha_k \end{bmatrix} \\ &= 1 + r_k^\top x^{(k)} + \alpha_k \left(t_{k+1} + r_k^\top J_k x^{(k)}\right) \\ &= \left(1 - \alpha_k^2\right) \beta_k \end{aligned} \tag{31}$$

总运算量降为 $2n^2$. 这就是求解 Yule-Walker 方程组的 Durbin 算法.

一般右端项的对称正定 Toeplitz 线性方程组

考虑一般右端项的方程组 $T_n x = b$, 其中 T_n 对称正定.

我们利用递推方法来求解: 假定 $x^{(k)}$ 和 $y^{(k)}$ 分别是下面两个方程组的解:

$$T_k x = [b_1, b_2, \ldots, b_k]^\top, \quad T_k y = -[t_1, t_2, \ldots, t_k]^\top \tag{32}$$

设 $x^{(k+1)} = \begin{bmatrix} z^{(k)} \\ \mu_k \end{bmatrix}$ 是 $T_{k+1} x = b^{(k+1)}$ 的解,则可得

$$z^{(k)} = x^{(k)} + \mu_k J_k y^{(k)}, \quad \mu_k = \frac{b_{k+1} - r_k^\top J_k x^{(k)}}{1 + r_k^\top y^{(k)}} \tag{33}$$

算法	求解 Yule-Walker 方程组的 Durbin 算法
1：	输入数据: $t = [t_1, t_2, ..., t_n]$ % 注: 这里假定 t0 = 1
2：	$x(1) = -t_1, = 1, = -t_1$
3：	for for k = 1 to n − 1 do
4:	$\beta = (1 - \alpha^2)\beta$
5：	$\alpha = -\left(t_{k+1} - \sum_{i=1}^{k} t_{k+1-i}x(i)\right)/\beta$
6:	$x(1:k) = x(1:k) + \alpha x(k:-1:1)$
7:	$x(k+1) = \alpha$
8：	end for

算法 4.9	求解对称正定 Toeplitz 线性方程组的 Levinson 算法
1：	输入数据: $t = [t_1, t_2, ..., t_n]$ % 假定 t0 = 1
2：	$y(1) = -t_1, x(1) = b_1, \beta = 1, \alpha = -t_1$
3：	for k = 1 to n−1 do
4:	$\beta = (1 - \alpha^2)\beta$
5：	$\mu = \left(b_{k+1} - \sum_{i=1}^{k} t_{k+1-i}x(i)\right)/\beta$
6:	$x(1:k) = x(1:k) + \mu y(k:-1:1), \quad x(k+1) = \mu$
7:	if k < n − 1 then
8：	$\alpha = -\left(t_{k+1} + \sum_{i=1}^{k} t_{k+1-i}y(i)\right)/\beta$
9:	$y(1:k) = y(1:k) + \alpha y(k:-1:1)$
10:	$y(k+1) = \alpha$
11:	end for
12:	end for

所以, 我们可以先计算 $T_k x = b^{(k)}$ 和 $T_k x = -r_k$ 的解，, 然后利用上述公式得到 $T_{k+1}x = b^{(k+1)}$ 的解,这就是 Levinson 算法, 总运算量大约为 $4n^2$.

在数学与工程的许多应用中都会出现 $Toeplitz$ 线性方程组. $Levinson$ 算法是较早的关于对称正定 $Toeplitz$ 线性方程组的快速算法, 但并不稳定 (只具有弱稳定性). 后来人们提出了各种各样的快速和超快速算法:

• Fast : Levinson-Durbin (1946), Trench (1964), ...

• Fast stable: Bareiss (1969), Gohberg, Kailath and Olshevsky (1995), Chandrasekaran and Sayed (1998), Gu (1998), ...

• Superfast: Brent, Gustavson and Yun (1980), Bitmead and Anderson (1980),Morf (1980), de Hoog (1987), Ammar and Gragg (1988), ...

• Superfast Preconditioners: Strang, Chan, Chan, Tyrtyshnikov, ... 6

表 1: 改变表格任一列宽

方法	运算量	存储量
Fast stable	≥$20n^2$	≥$n^2/2$
Fast but unstable	≥$3n^2$	≥$4n$
Superfast and “unstable”	$O(nlog^2n)$	O(n)
Superfast preconditioner	O(n log n)	O(n)

4.4 扰动分析

3.1 x 与 $\hat{x}$ 的关系

3.2 x 与 x_* 的关系

3.3 x 与残量的关系

3.4 相对扰动分析

考虑线性方程组

$$Ax = b \tag{34}$$

设 $x*$ 是精确解, $\hat{x}$ 是通过数值计算得到的近似解. 假定 $\hat{x}$ 满足线性方程组

$$(A + \delta A)\hat{x} = b + \delta b \tag{35}$$

$\delta x \triangleq \hat{x} - x$ 的大小, 即向后误差分析.

4.4.1 x 与 $\hat{x}$ 的关系

定理 4.8 *设 $||.||$ 任一向量范数(当该范数作用在矩阵上时就是相应的导出范数), 则 x 与 $\hat{x}$ 满足下面的关系式*

$$\frac{\|\delta x\|}{\|\hat{x}\|} \leq \left\|A^{-1}\right\| \cdot \|A\| \left(\frac{\|\delta A\|}{\|A\|} + \frac{\|\delta b\|}{\|A\| \cdot \|\hat{x}\|}\right) \tag{36}$$

当 $b = 0$ 时, 有

$$\frac{\|\delta x\|}{\|\hat{x}\|} \leq \kappa(A)\frac{\|\delta A\|}{\|A\|} \tag{37}$$

4.4.2 x 与 x_* 的关系

引理 6 *设 $||.||$ 是任一算子范数, $X \in \mathbb{R}^{n\times n}$. 若 $\|X\| < 1$, 则 $I - X$ 可逆, 且有*

$$(I - X)^{-1} = \sum_{k=0}^{\infty} X^k \quad 和 \quad \left\|(I - X)^{-1}\right\| \leq \frac{1}{1 - \|X\|} \tag{38}$$

定理 4.9 *设 $A\in R^{n\times n}$ 非奇异且 $\|A^{-1}\|\cdot\|\delta A\|<1$, 则*

$$\frac{\|\delta x\|}{\|x_*\|}\leq\frac{\kappa(A)}{1-\kappa(A)\cdot\frac{\|\delta A\|}{\|A\|}}\left(\frac{\|\delta A\|}{\|A\|}+\frac{\|\delta b\|}{\|b\|}\right) \tag{39}$$

如果 $||\delta A||=0$, 则

$$\frac{1}{\kappa(A)}\frac{\|\delta b\|}{\|b\|}\leq\frac{\|\delta x\|}{\|x_*\|}\leq\kappa(A)\frac{\|\delta b\|}{\|b\|} \tag{40}$$

定理 4.10 *设 $A\in\mathbb{R}^{n\times n}$ 非奇异, 则有*

$$\min\left\{\frac{\|\delta A\|_2}{\|A\|_2}:A+\delta A奇异\right\}=\frac{1}{\kappa_2(A)} \tag{41}$$

† 上述定理中的结论对所有 $p-$ 范数都成立.

† 度量

$$disp_p(A)\triangleq\min\left\{\frac{\|\delta A\|_p}{\|A\|_p}:A+\delta A奇异\right\}=\frac{1}{\kappa_p(A)} \tag{42}$$

表示 A 距离奇异矩阵集合的相对距离.

4.4.3 x 与残量的关系

记残量 (残差) 为 $r=b-A\hat{x}$, 则有

$$\delta x=\hat{x}-x_*=\hat{x}-A^{-1}b=A^{-1}(A\hat{x}-b)=-A^{-1}r \tag{43}$$

所以可得

$$\|\delta x\|\leq\left\|A^{-1}\right\|\cdot\|r\| \tag{44}$$

实际计算中 r 是可以计算的, 因此比较实用.

4.4.4 相对扰动分析

前面给出的误差 δx 与条件数 $\kappa(A),\delta A$ 和 δb 成比例. 许多情况下, 这个界是令人满意的. 但有时相差很大, 不能很好的反映实际计算中解的误差.

例 设 $A=\begin{bmatrix}\gamma&0\\0&1\end{bmatrix},b=\begin{bmatrix}\gamma\\1\end{bmatrix}$,其中 $\gamma>1$, 则 $Ax=b$ 的精确解为 $x_*=\begin{bmatrix}1\\1\end{bmatrix}$, 任何合理的直接法求得的解的误差都很小. 但系数矩阵的谱条件数为 $\kappa_2(A)=\gamma$, 当 γ 很大时,$\kappa_2(A)$ 也很大, 因此误差界 (2.16) 和 (2.18) 可以是很大。

针对这个问题, 我们按分量进行分析. 记

$$\delta A=\begin{bmatrix}\delta a_{11}\\\delta a_{22}\end{bmatrix},\quad\delta b=\begin{bmatrix}\delta b_1\\\delta b_2\end{bmatrix} \tag{45}$$

,

并设 $|\delta a_{ij}| \leq \varepsilon |a_{ij}|, |\delta b_i| \leq \varepsilon |b_i|$, 则

$$\delta x = \begin{bmatrix} \hat{x}_1 - x_1 \\ \hat{x}_2 - x_2 \end{bmatrix} = \begin{bmatrix} \frac{\delta b_1 + b_1}{\delta a_{11} + a_{11}} - 1 \\ \frac{\delta b_2 + b_2}{\delta a_{22} + a_{22}} - 1 \end{bmatrix} = \begin{bmatrix} \frac{\delta b_1 + \gamma}{\delta a_{11} + \gamma} - 1 \\ \frac{\delta b_2 + 1}{\delta a_{22} + 1} - 1 \end{bmatrix} = \begin{bmatrix} \frac{\delta b_1 - \delta a_{11}}{\delta a_{11} + \gamma} \\ \frac{\delta b_2 - \delta a_{22}}{\delta a_{22} + 1} \end{bmatrix} \tag{46}$$

故

$$\|\delta x\|_\infty \leq \frac{2\varepsilon}{1 - \varepsilon} \tag{47}$$

如果 $b = 0$, 则

$$\|\delta x\|_\infty \leq \frac{\varepsilon}{1 - \varepsilon} \tag{48}$$

这个界与 (2.16) 或 (2.18) 相差约 γ 倍.

相对条件数

为了得到更好误差界, 我们引入相对条件数 $\kappa_{cr}(A)$, 即

$$\kappa_{cr}(A) \triangleq \left\| \left| A^{-1} \right| \cdot |A| \right\| \tag{49}$$

有时也称为 Bauer 条件数或 Skeel 条件数.

假定 A 和 b 满足 $|A| \leqq |A|$ 和 $|b| \leqq |b|$. 由 $(A + A)\hat{x} = b + b$ 可得

$$\begin{aligned} |\delta x| &= \left| A^{-1}(-\delta A \hat{x} + \delta b) \right| \\ &\leq \left| A^{-1} \right| \cdot (|\delta A| \cdot |\hat{x}| + |\delta b|) \\ &\leq \left| A^{-1} \right| \cdot (\varepsilon |A| \cdot |\hat{x}| + \varepsilon |b|) \\ &= \varepsilon \left| A^{-1} \right| \cdot (|A| \cdot |\hat{x}| + |b|) \end{aligned} \tag{50}$$

若 $b = 0$, 则有

$$\|\delta x\| = \||\delta x|\| \leq \varepsilon \left\| \left| A^{-1} \right| \cdot |A| \cdot |\hat{x}| \right\| \leq \varepsilon \left\| \left| A^{-1} \right| \cdot |A| \right\| \cdot \|\hat{x}\| \tag{51}$$

即

$$\frac{\|\delta x\|}{\|\hat{x}\|} \leq \left\| \left| A^{-1} \right| \cdot |A| \right\| \cdot \varepsilon = \kappa_{cr}(A) \cdot \varepsilon \tag{52}$$

相对条件数有下面的性质

引理 7 *设 $A \in R^{n \times n}$ 非奇异, $D \in R^{n \times n}$ 为非奇异对角矩阵, 则*

$$\kappa_{cr}(DA) = \kappa_{cr}(A) \tag{53}$$

定理 4.11 *设 $A \in R^{n \times n}$ 非奇异, 使得 $|\delta A| \leq \varepsilon |A|, |\delta b| \leq \varepsilon |b|$ 成立, 且满足*

$$(A + \delta A)\hat{x} = b + \delta b \tag{54}$$

的最小的 >0 称为按分量的相对向后误差, 其表达式为

$$\varepsilon = \max_{1\le i\le n} \frac{|r_i|}{(|A|\cdot|\hat{x}|+|b|)_i} \tag{55}$$

, 其中 $r=b-A\hat{x}$

更多关于数值计算的稳定性和矩阵扰动分析方面的知识, 可以参考相关资料.

4.5 误差分析

4.1 LU 分解的舍入误差分析

4.2 Gauss 消去法的舍入误差分析

4.5.1 LU 分解的舍入误差分析

关于 LU 分解的舍入误差分析, 我们有下面的结果.

定理 4.12 *假定 $A\in\mathbb{R}^{n\times n}$ 的所有顺序主子式都不为 0, 则带舍入误差的 LU 分解可表示为*

$$A = LU + E \tag{56}$$

其中误差 E 满足

$$|E| \le \gamma_n |L|\cdot|U| \tag{57}$$

这里 $\gamma_n = \frac{n\varepsilon_u}{1-n\varepsilon_u}, \varepsilon_u$ 表示机器精度.

4.5.2 Gauss 消去法的舍入误差分析

引理 8 (High02) *设 $\hat{y}$ 和 $\hat{x}$ 分别是由向前回代和向后回代得到的数值解, 则*

$$\begin{aligned} (L+\delta L)\hat{y} &= b, \quad |\delta L| \le \gamma_n |L| \\ (U+\delta U)\hat{x} &= \hat{y}, \quad |\delta U| \le \gamma_n |U| \end{aligned} \tag{58}$$

该引理表明, 向前回代算法和向后回代算法都是稳定的.

† 在绝大多数情况下, 部分选主元 Gauss 消去法是向后稳定的, 但理论上也存在失败的例子.

† 全主元 Gauss 消去法是数值稳定的. 在大部分实际应用中, 部分选主元 Gauss 消去法与全主元 Gauss 消去法具有同样的数值稳定性.

4.6 解的改进和条件数估计

5.1 高精度运算

5.2 矩阵元素缩放 (Scaling)

5.3 迭代改进法

4.6.1 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.

4.6.2 矩阵元素缩放 (Scaling)

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

例 考虑线性方程组

$$\begin{bmatrix} -4000 & 2000 & 2000 \\ 2000 & 0.78125 & 0 \\ 2000 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 400 \\ 1.3816 \\ 1.9273 \end{bmatrix} \tag{59}$$

用部分选主元 Gauss 法求解, 计算过程保留 8 位有效数字, 求得数值解

$$\overline{x} = [0.00096365, -0.698496, 0.90042329]^\top \tag{60}$$

与精确解 $x = [1.9273\cdots, -0.698496\cdots, 0.9004233\cdots]^\top$ 误差较大

考虑矩阵元素缩放, 即同乘对角阵 $D = diag(0.00005, 1, 1)$, 得新方程组

$$DADy = Db \tag{61}$$

最后令 $\tilde{x} = Dy$, 即可求得比较精确的数值解.

4.6.3 迭代改进法

设近似解 $\hat{x}$, 残量 $r = b - A\hat{x}$. 当 $\hat{x}$ 没达到精度要求时, 可以考虑方程 $Az = r$. 如果 z 该方程的精确解, 则

$$A(\hat{x} + z) = A\hat{x} + Az = (b - r) + r = b \tag{62}$$

’ 因此 $\hat{x} + z$ 就是原方程的精确解. 在实际计算中, 我们只能得到近似解 $\hat{z}$, 但 $||r - A\hat{z}||$ 很小, 特别地, 应该比 $||r||$ 更小. 因此 $\hat{x} + \hat{z}$ 应该比 x 更接近精确解.

如果新的近似解 $\hat{x} + \hat{z}$ 还不满足精度要求, 则可重复以上过程. 这就是通过迭代来提高解的精度.

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

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

算法 5.1	通过迭代改进解的精度
1：	设 $PA = LU, \hat{x}x$ 是 $Ax = b$ 的近似解
2：	while 近似解 $\hat{x}$ 不满足精度要求, do
3：	计算 $r = b - A\hat{x}$
4:	求解 $Ly = Pr$, 即 $y = L^{-1}Pr$
5：	求解 $Uz = y$, 即 $z = U^{-1}y$
6:	令 $\hat{x} = \hat{x} + z$
7:	end while

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

最小二乘问题

- 线性最小二乘问题
- 总体最小二乘问题
- 约束最小二乘问题
- $\cdots$

最小二乘问题在统计学, 最优化问题, 材料与结构力学, 信号与图像处理等方面都有着广泛的应用, 是计算数学的一个重要研究分支, 也是一个活跃的研究领域.

本讲主要介绍求解线性最小二乘问题的三种直接法.

5 线性最小二乘问题

5.1 引言

考虑线性最小二乘问题

$$\min_{x\in\mathbb{R}} ||Ax - b||_2^2 \tag{63}$$

其中 $A \in \mathbb{R}^{m\times n}, b \in \mathbb{R}^m$. 问题 (1) 的解称为最小二乘解.

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

为了讨论方便，本讲总是假定 A 是满秩的.

5.1.1 欠定方程组

若 $m < n$, 则 $Ax = b$ 存在无穷多个解，这时我们通常寻求最小范数解, 于是问题就转化为下面的约束优化问题

$$\min_{Ax=b} \frac{1}{2}\|x\|_2^2 \tag{64}$$

对应的Lagrange 函数为

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

其中 $\lambda = [\lambda_1, \lambda_2, \ldots, \lambda_m]^\top$ 是Lagrange 乘子.

此时优化问题 (2) 的解就是 $\mathcal{L}(x, \lambda)$ 的鞍点，即下面方程组的解：

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

写成矩阵形式为

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

如果 A(行) 满秩，即 $rank(A) = m$, 则系数矩阵非奇异, 上述方程组存在唯一解.

5.1.2 超定方程组

当 $m > n$ 时，线性方程组 $Ax = b$ 的解可能不存在. 记

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

易知 $J(x)$ 是关于 x 的二次凸函数. 因此，x_* 是问题 (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

$$A^\top Ax - A^\top b = 0$$

于是 (1) 就转化为一个 (半正定) 线性方程组.

本讲我们主要讨论超线性最小二乘问题的求解.

5.2 初等变换矩阵

矩阵计算的一个基本思想就是把较复杂的问题转化为等价的较简单的, 易于求解问题. 而完成这个转化的基本工具就是初等变换矩阵, 其中常用的有三个: Gauss 变换, Householder 变换和 Givens 变换.

5.2.1 初等矩阵

我们考虑初等矩阵

$$E(u,v,\tau)=I-\tau uv^*$$

其中 $u,v\in\mathbb{C}^n$ 是非零向量，τ 是一个非零复数. 事实上，$E(u,v,\tau)$ 是单位矩阵的一个秩 1 扰动.

定理 设 $E(u,v,\tau)$ 是一个初等矩阵，我们有

(1)$\det(E(u,v,\tau))=1-\tau v^*u$

(2) 若 $1-\tau v^*u\neq 0$, 则 $E(u,v,\tau)$ 非奇异，且

$$(E(u,v,\tau))^{-1}=E(u,v,\tau)$$

其中 $\gamma=\frac{\tau}{\tau v^*u-1}$

5.2.2 Gauss 变换

设 $l_j=[0,\ldots,0,l_{j+1,j},\ldots,l_{n,j}]^\top, j=1,2,\ldots,n$，则Gauss 变换定义为

$$L(l_j)\triangleq E(l_j,e_j,-1)=I+l_je_j^\top=\begin{bmatrix}1&&&&\\&\ddots&&&\\&&1&&\\&&l_{j+1,j}&1&\\&&\vdots&&\ddots\\&&l_{n,j}&&1\end{bmatrix}$$

向量 l_j 称为Gauss 向量，由定理2.1可知

$$\det(L(l_j))=1,\quad (L(l_j))^{-1}=E(l_j,e_j,1)=E(-l_j,e_j,-1)=L(-l_j)$$

Gauss 变换主要用于矩阵的 LU 分解

5.2.3 Householder 变换

定义 我们称矩阵

$$H=I-\frac{2}{v^*v}vv^*=I-\frac{2}{\|v\|_2^2}vv^*,\quad 0\neq v\in\mathbb{C}^n \tag{65}$$

为Householder 矩阵（或Householder 变换，或 Householder 反射），向量 v 称为Householder 向量. 我们通常将矩阵 (3) 记为 $H(v)$

从几何上看, Householder 变换就是一个关于超平面 $\text{span}\{v\}^\perp$ 的反射，对任意一个向量 $x\in\mathbb{C}^n$, 可将其写为

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

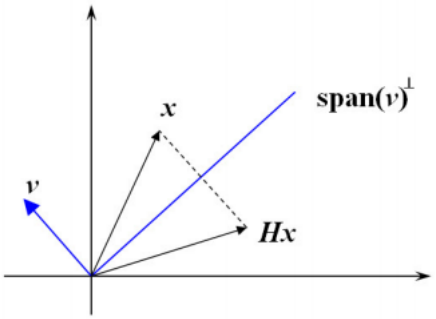


图 2.1 Householder 变换的几何意义

其中 $\alpha v \in \text{span}\{v\}, y \in \text{span}\{v\}^{\perp}$. 则

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

即 Hx 与 x 在 $\text{span}\{v\}^{\perp}$ 方向有着相同的分量, 而在 v 方向的分量正好相差一个符号. 也就是说,Hx 是 x 关于超平面 $\text{span}\{v\}^{\perp}$ 的镜面反射, 见图2.1. 因此, Householder 矩阵也称为反射矩阵.

Householder 矩阵的几个基本性质

定理 设 $H \in \mathbb{C}^{n\times n}$ 是一个 Householder 矩阵, 则 (1)$H^* = H$, 即 H Hermite 的;

(2)$H^*H = I$, 即 H 是酉矩阵;

(3)$H^2 = I$, 所以 $H^{-1} = H$;

(4)$det(H) = -1$;

(5)H 有两个互异的特征值:$\lambda = 1$ 和 $\lambda = -1$, 其中 $\lambda = 1$ 的代数重数为 $n-1$

Householder 矩阵的重要应用: 将一个向量除第一个元素以外的所有元素都化为零

引理 设 $x, y \in mathbbC^n$n 为任意两个互异的向量, 则存在一个 Householder 矩阵 $H(x)$ 使得 $y = H(v)x$ 的充要条件是 $\|x\|_2 = \|y\|_2 \mathbb{E} x^*y \in \mathbb{R}$(取 $v = x - y$ 即可)

定理 设 $x = [x_1, x_2, \ldots, x_n]^{\top} \in \mathbb{R}^n$ 是一个非零向量, 则存在 Householder 矩阵 H 使得 $Hx = \alpha e_1$, 其中 $\alpha = \|x\|_2$(或 $\alpha = -\|x\|_2$),$e_1 = [1, 0, \ldots, 0]^{\top} \in \mathbb{R}^n$

设 $x = [x_1, x_2, \ldots, x_n]^{\top} \in \mathbb{R}^n$ 是一个实的非零向量, 下面讨论如何计算定理中的 Householder 矩阵 $H(v)$. 由引理的证明过程可知

$$v = x - \alpha e_1 = [x_1 - \alpha, x_2, \ldots, x_n]^{\top}$$

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法运算, 否则就会损失有效数字. 因此, 我通常取

$$\alpha = -\,\text{sign}\,(x_1) \cdot \|x\|_2$$

事实上, 我们也可以取 $\alpha = \text{sign}\,(x_1)\,\|x\|_2$ 但此时为了减少舍入误差, 我们需要通过下面的公式来计算 v 的第一个分量 v_1

$$\alpha = \text{sign}\,(x_1)\,\|x\|_2$$

$$v_1 = x_1 - \alpha = \frac{x_1^2 - \|x\|_2^2}{x_1 + \alpha} = \frac{-\left(x_2^2 + x_3^2 + \cdots + x_n^2\right)}{x_1 + \alpha}$$

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

无论怎样选取 α, 我们都有 $H = I - \beta v v^*$ 其中

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

算法 2.1计算 Householder 向量
%Given $x \in \mathbb{R}^n$,compute β,v suth that $Hx = \|x\|_2 e_1$ with $H = I - \beta v v^*$

1: function[β, v]=house(x)
2: n = length(x) (here length(x) denotes the dimension of x)
3: $\sigma = x_2^2 + x_3^2 + \cdots + x_n^2$
4: $v = x$
5: if $\sigma = 0$ then
6: if $x_1 < 0$ then
7: $v_1 = 2x, \beta = 2/v_1^2$
8: else
9: $v_1 = 0, \beta = 0$
10: end if
11: else
12: $\alpha = \sqrt{x_1^2 + \sigma}$ % $\alpha = \|x\|_2$
13: if $x_1 < 0$ then
14: $v_1 = x_1 - \alpha$
15: else
16: $v_1 = -\alpha/(x_1 + \alpha)$
17: end if
18: $\beta = 2/(v_1^2 + \alpha)$
19: end if

总运算量大约为 3n, 且具有很好的数值稳定性

在实际计算时, 我们可以将向量 v 单位化, 使得 $v_1 = 1$, 这样, 我们就无需为 v 另外分配空间, 而是将 $v(2:n)$ 存放在 $x(2:n)$ 中, 因为变换后的向量 x 除第一个分量外, 其它都为零.

为了避免可能产生的溢出, 可事先将 x 单位化,即令 $x = x/\|x\|_2$

Householder 变换的运算量
设 $A \in \mathbb{R}^{m\times n}, H = I - \beta vv^* \in \mathbb{R}^m$, 则

$$HA = (I - \beta vv^*)\, A = A - \beta vv^* A = A - \beta v\,(A^* v)^*$$

因此, 在做 Householder 变换时, 并不需要生成 Householder 矩阵, 只需要 Householder 向量即可. 上面矩阵相乘的总运算量大约为 4mn.

5.2.4 **Givens 变换**

$$G(i,j,\theta) = \begin{bmatrix} 1 & & & & & & \\ & \ddots & & & & & \\ & & c & & s & & \\ & & & \ddots & & & \\ & & -s & & c & & \\ & & & & & \ddots & \\ & & & & & & 1 \end{bmatrix} \in \mathbb{R}^{n\times n}\ (\theta \in [0, 2\pi], i \leqslant j)$$

Givens 变换 (或 Givens 旋转, 或 Givens 矩阵), 其中 $c = \cos(\theta), s = \sin(\theta)$

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

左乘 Givens 矩阵: 只会影响第 i 行和第 j 的元素.
右乘 Givens 矩阵: 只会影响第 i 和第 j 列的元素.

例　设 $x = [x_1, x_2]^\top \in \mathbb{R}^2$, , 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 $Gx = [r, 0]^\top$,其中 c, s 和 r 的值如下:

- 若 $x_1 = x_2 = 0$, 则 $c = 1, s = 0, r = 0$;
- 若 $x_1 = 0$ 但 $x_2 \neq 0$, 则 $c = 0, s = x_2/|x_2|, r = |x_2|$
- 若 $x_1 \neq 0$ 但 $x_2 = 0$, 则 $c =$sign$(x_1), s = 0, r = |x_1|$;
- 若 $x_1 \neq 0$ 且 $x_2 \neq 0$, 则 $c = x_1/r, s = x_2/r, r = \sqrt{x_1^2 + x_2^2}$.

通过 Givens 变换, 我们可以将向量 $x \in \mathbb{R}^2$ 的第二个分量化为 0.

事实上, 对于任意一个向量 $x \in \mathbb{R}^n$, 我们都可以通过 Givens 变换将其任意一个位置上的分量化为 0. 更进一步, 我们也可以通过若干个 Givens 变换, 将 x 中除第一个分量外的所有元素都化为 0.

算法 2.2Givens 变换
% Given $x = [a, b]^\top$,compute c,s such that $Gx = [r, 0]^\top$ where $r = \|x\|_2$

function $[c, s]$ = givens(a, b)
if $b = 0$ then
    if $a \geq 0$then
        $c = 1, s = 0$
    else
        $c = -1, s = 0$
    end if
else
    if $|b| > |a|$ then
        $\tau = a/b, \quad s = \text{sign}(b)/\sqrt{1+\tau^2}, \quad c = s\tau$
    else
        $\tau = b/a, \quad c = \text{sign}(a)/\sqrt{1+\tau^2}, \quad s = c\tau$
    end if
end if

5.2.5 正交变换的舍入误差分析

引理 设 $P \in \mathbb{R}^{n\times n}$ 是一个精确的 Householder 或 Givens 变换, $\tilde{P}$ 是其浮点运算近似, 则

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

其中 $\|E\|_2 = \mathcal{O}(\varepsilon_u)\cdot\|A\|_2, \|F\|_2 = \mathcal{O}(\varepsilon_u)\cdot\|A\|_2$

这说明对一个矩阵做 Householder 变换或 Givens 变换是向后稳定的.

考虑对矩阵 A 做一系列的正交变换, 则有

$$\text{fl}\left(\tilde{P}_k\cdots\tilde{P}_1A\tilde{Q}_1\cdots\tilde{Q}_k\right) = P_k\cdots P_1(A+E)Q_1\cdots Q_k$$

其中 $\|E\|_2 = \mathcal{O}(\varepsilon_u) \cdot (k\|A\|_2)$,这说明整个计算过程是向后稳定的

一般地, 假设 X 是一个非奇异的线性变换, $\tilde{X}$ 是其浮点运算近似. 当 X 作用到 A 上时, 我们有

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

其中 $\|E\|_2 = O(\varepsilon_u) \cdot \|XA\|_2 \leq O(\varepsilon_u) \cdot \|X\|_2 \cdot \|A\|_2$, 故

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

因此, 舍入误差将被放大 $k_2(X)$ 倍. 当 X 是正交变换时,$k_2(X)$ 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

5.3.1 QR 分解的存在唯一性

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

$$A = QR \tag{66}$$

证明: 设 $A = [a_1, a_2, \ldots, a_n] \in \mathbb{C}^{m\times n}$
若 A 列满秩, 即 rank(A) = n. 则 QR 分解 (4) 就是对 A 的列向量组进行 Gram-Schmidt 正交化过程的矩阵描述. 具体过程见下面的算法.
算法 3.1Gram-schmidt Process

1: $r_{11} = ||a_1||_2$

2: $q_1 = a_1/r_{11}$

3: for $j = 2$ to n do

4: $q_j = a_j$

5: for $i = 1$ to $j - 1$ do

6: $r_{ij} = q_i^* a_j$

7: $q_j = q_j - r_{ij} q_i$

8: end for

9: $r_{jj} = ||q_j||_2$

10: $q_j = q_j / r_{jj}$

11: end for

如果 A 不是列满秩，我们可以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 $j = 2, 3, \cdots$ 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 $Q = [q_1, q_2, ..., q_n]$ 列正交 (但不是单位列正交，列向量中可能有零向量). 这里的 $R = [r_{ij}]_{n \times n}$ 是上三角矩阵，定义如下

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

易知，如果 Q 的某一列 $q_k = 0$, 则 R 中对应的第 k 行全为 0
设 $rank(A) = l < n$, 则 rank$(Q) = l$ 即 Q 只有 l 个非零列，不妨设为 $q_{i1}, q_{i2}, ...q_{il}$, 他们构成 $\mathbb{C}^m$ 的一个单位正交向量组，将其扩展成 $\mathbb{C}^m$ 中的一组标准正交基，即

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

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 总的第二个零列，依次类推，将 Q 中的所有零列都替换掉. 将最后得到的矩阵记为 $\tilde{Q}$, 于是 $\tilde{Q} \in \mathbb{C}^{m \times n}$ 单位列正交. 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

$$\tilde{Q}R = QR = A$$

这就是 A 的 QR 分解

满秩矩阵 QR 分解的存在唯一性

定理 若 A 列满秩，并要求 R 的对角线元素都为正，则 A 的 QR 分解存在且唯一.

† 若 A 是实矩阵，则所有运算都是实运算，因此 Q 和 R 都是实矩阵.

† 有时也将 QR 分解定义为：存在酉矩阵 $Q \in \mathbb{C}^{m\times m}$ 使得

$$A = A\begin{bmatrix} R_{11} \\ 0 \end{bmatrix}$$

其中 $R_{11} \in \mathbb{R}^{n\times n}$ 为上三角矩阵.

若 A 不满秩，存在置换矩阵 P，使得 AP 的前 l 列线性无关，其中 $l = rank(A)$. 对 AP 进行 QR 分解，可得：
推论　设 $A \in \mathbb{C}^{m\times n}(m \geq n)$ 则存在一个置换矩阵 P，使得

$$AP = Q\begin{bmatrix} R_{11} & R_{12} \\ 0 & 0 \end{bmatrix}_{n\times n}$$

其中 $Q \in \mathbb{C}^{m\times n}$ 单位列正交，$R_{11} \in \mathbb{C}^{l\times l}$ 是非奇异上三角矩阵.
† 上述结论可简化为

$$AP = Q\begin{bmatrix} R_{11} & R_{12} \end{bmatrix}$$

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.
推论 (满秩分解) 设 $A \in \mathbb{C}^{m\times n}$, 且 $rank(A) = l \leq min\{m, n\}$, 则存在满秩矩阵 $F \in \mathbb{C}^{m\times l}$ 和 $G \in \mathbb{C}^{l\times n}$ ，使得

$$A = FG$$

† 如果 A 是非奇异方阵，则 QR 分解可用来求解线性方程组 $Ax = b$

† 基于 G-S 正交化的 QR 分解的运算量大约为 $2mn^2$.

在后面，我们会介绍基于 Householder 变换的 QR 分解，在不需要计算 Q 的情况下，运算量大约为 $2mn^2 - 2n^3/3$, 如果需要计算 Q，则需另外大约 $2mn^2 - 2n^3/3$ 运算量.

5.3.2　基于 MGS 的 QR 分解

由于数值稳定性方面的原因, 在实际计算中，我们一般不采用 Gram-Schmidt 过程，取而代之的是修正的是Gram-Schmidt 过程, 本算法的运算量大约为 $2mn^2$
算法 3.2 基于 MGS 的 QR 分解
% Given $A \in \mathbb{R}^{m\times n}$, compute $Q = [q_1, ..., q_n]$ and R such that $A = QR$

1. Set $R = [r_{ik}] = 0_{n\times n}$(the $n \times n$zero matrix)
2. if $a_1 = 0$ then

3. $q_1 = 0$
4. else
5. $r_{11} = ||a_1||_2$
6. $q_1 = a_1/||a_1||_2$
7. end if
8. for $k = 2$ to n do
9. $q_k = a_k$
10. for $i = 1$ to $k - 1$ do
11. $r_{ik} = q_i^T q_k$
12. $q_k = q_k - r_{ik} q_i$
13. end for
14. if $q_k \neq 0$ then
15. $r_{kk} = ||q_k||_2$
16. $q_k = q_k/r_{kk}$
17. end if
18. end for

† MGS 得到的 QR 分解中，$Q \in \mathbb{R}^{n\times n}, R \in \mathbb{R}^{n\times n}$

5.3.3 基于 Householder 变换的 QR 分解

由 Householder 变换的性质可知，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，其他都为零.
假定 $m = n$，即 $A \in \mathbb{R}^{n\times n}$ 令 $H_1 \in \mathbb{R}^{n\times n}$ 为 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

$$H_1 A = \left[\begin{array}{c|ccc} r_1 & \tilde{a}_{12} & \cdots & \tilde{a}_{1n} \\ \hline 0 & & & \\ \vdots & & \tilde{A}_2 & \\ 0 & & & \end{array}\right], \tilde{A}_2 \in \mathbb{R}^{(n-1)\times(n-1)}$$

同样，构造 Householder 变换 $\tilde{H}_2 \in \mathbb{R}^{(n-1)\times(n-1)}$，使得

$$\tilde{H}_2\tilde{A}_2 = \left[\begin{array}{c|ccc} r_1 & \tilde{a}_{12} & \cdots & \tilde{a}_{1n} \\ \hline 0 & & & \\ \vdots & & \tilde{A}_2 & \\ 0 & & & \end{array}\right], \tilde{A}_2 \in \mathbb{R}^{(n-1)\times(n-1)}$$

令 $H_2 = \begin{bmatrix} 1 & 0 \\ 0 & \tilde{H}_2 \end{bmatrix} \in \mathbb{R}^{n\times n}$，则

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

不断重复上述过程，这样，我们就得到一系列的矩阵

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

使得

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

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

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

如果不需要生成 Q，则运算量大约为 $2mn^2 - 2/3n^3$.

矩阵 Q 的计算
可通过下面的算法实现

$$Q = I_n, \quad Q = QH_k, \quad k = 1, 2, \ldots, n-1$$

若保留了所有的 Householder 向量，则 Q 可以通过下面的向后累积法实现：

$$Q = I_n, \quad Q = H_kQ, \quad k = n-1, n-2, \ldots, 1$$

优点：一开始 Q 会比较稀疏，随着迭代的进行，Q 才会慢慢变满
运算量大约为 $4(m^2n - mn^2 + \frac{1}{3}n^3)$.

† 若 $m > n$ 则由 Householder 变换得到的 QR 分解中，$Q \in \mathbb{R}^{m\times m}, R \in \mathbb{R}^{m\times n}$

算法 3.3基于 Householder 变换的 QR 分解
%Given$A \in \mathbb{R}^{m\times\times n}$,compute$Q \in \mathbb{R}^{m\times m}$,$R \in \mathbb{R}^{m\times n}$ such that $A = QR$ % The upper triangular part of R is stored in the upper triangular part of A

1: Set $Q = I_{m\times m}$

2: for $k = 1$ to n do

3: $x = A(k : m, k)$

4: $[\beta, v_k]$ = house(x)

5: $v_k = v_k||v||2$

6: $A(k : m, k : n) = (I_{m-k+1} - 2v_k v_k^T)A(k : m, k : n)$

7: $Q(1 : k-1, k : m) = Q(1 : k-1, k : m)(I_{m-k+1} - 2v_k v_k^{\square})$

8: $Q(k : m, k : m) = Q(k : m, k : m)(I_{m-k+1} - 2v_k v_k^T)$

9: end for

上面的算法只是一个简单描述, 并没有考虑运算量问题.

† 在实际计算时, 我们通常会保留所有的 Householder 向量.

由于第 k 步中 $\hat{H}_k$ 所对应的 Householder 向量 v_k 的长度为 $m-k+1$, 因此我们先把 v_k 单位化, 使得 v_k 的第一元素为 1, 这样就只要存储 $v_k(2 : end)$, 共 $m-k$ 个元素.

这样, 我们就可以把所有的 Householder 向量存放在 A 的严格下三角部分, 而 A 的上三角部分仍然存放 R.

在计算 Q 时采用向后累积的方法, 所以总的运算量大约为 $4m^2n - 2mn^2 + 2n^3$. 若 $m = n$, 则运算量大约为 $8n^3$.

† 我们也可以考虑分块 Householder QR 分解, 以便充分利用 3 级 BLAS 运算, 提高计算效率.

5.3.4 列主元 QR 分解

当 A 不是满秩时, 我们可以进行列主元 QR 分解.
定理 (列主元 QR 分解) 设 $A \in \mathbb{C}^{m\times n}(m \geq n)$，且 $\text{rank}(A) = k < n$，则存在置换矩阵 P,

正交矩阵 $Q \in \mathbb{C}^{m\times m}$, 使得

$$AP = Q \begin{bmatrix} R_{11} & R_{12} \\ 0 & 0 \end{bmatrix}_{m\times n}$$

其中 $R_{11} \in \mathbb{C}^{k\times k}$ 是非奇异上三角矩阵, 且对角线元素满足 $r_{11} \geq r_{22} \geq \cdots \geq r_{kk} > 0$

列主元 QR 分解的实现过程与 QR 分解基本类似, 只是每一步需要选个列主元, 同时做一个列交换

假设经过 l 步后, 我们得到下面的分解

$$AP^{(l)} = Q^{(l)} \begin{bmatrix} R_{11}^{(l)} & R_{12}^{(l)} \\ 0 & R_{22}^{(l)} \end{bmatrix} \triangleq Q^{(l)} R^{(l)} \square\square \left(Q^{(l)}\right)^{\top} AP^{(l)} = R^{(l)}$$

其中 $P(l)$ 为置换矩阵, $Q(l)$ 为正交矩阵, $R_{12}^{(l)} \in \mathbb{R}^{(l)\times(l)}$ 非奇异上三角矩阵.
下面进行第 $l+1$ 步:

1. 计算 $R_{22}^{(l)}$ 所有列的范数, 若都为 0, 则算法结束, 此时必有 $l = k$.
2. 若 $l < k$, 则 $R_{22}^{(l)} \neq 0$. 设范数最大的列为第 i_{l+1} 列 (若有相等的, 取其中一列即可), 范数为 $r_{l+1,l+1}$. 若 $i_{l+1} \neq 1$, 则交换 $R(l)$ 的第 $l+1$ 列与第 $i_{l+1}+l$ 列, 并记相应的置换矩阵为 P_{l+1}.
3. 以 $R_{22}^{(l)}$ 的第 1 列构造 Householder 变换 $\bar{H}_{l+1}$
   令 $H_{l+1} = Blkdiag(I_l, \bar{H}_{l+1}), P^{(l+1)} = P^{(l)}P_{l+1}$. 则

$$H_{l+1}\left(Q^{(l)}\right)^{\top} AP^{(l+1)} = H_{l+1}R^{(l)}P_{l+1} = \begin{bmatrix} R_{11}^{(l)} & \tilde{R}_{12}^{(l)} \\ 0 & \tilde{R}_{22}^{(l)} \end{bmatrix} \triangleq R^{(l+1)}$$

其中 $\tilde{R}_{22}^{(l)}$ 的第一列除第一个元素外, 其余都是零, 且该元素等于 $r_{l+1,l+1}$, 即

$$R^{(l+1)} = \left[\begin{array}{cc|c} R_{11}^{(l)} & * & * \\ 0 & r_{l+1,l+1} & * \\ \hline 0 & 0 & * \end{array}\right] \triangleq \begin{bmatrix} R_{11}^{(l+1)} & R_{12}^{(l+1)} \\ 0 & R_{22}^{(l+1)} \end{bmatrix}$$

其中 $R_{11}^{(l+1)} \in \mathbb{R}^{(l+1)\times(l+1)}$ 为非奇异上三角矩阵
记 $Q^{(l+1)} \triangleq Q^{(l)}H_{l+1}^{\top}$, 则

$$AP^{(l+1)} = Q^{(l+1)}R^{(l+1)}$$

依此类推, 直到第 k 步, 即可得 A 的列主元 QR 分解.
矩阵 R11 的对角线元素的递减关系可由列主元的选取方法推出.

5.3.5 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 $A \in \mathbb{R}^{n\times n}$,构造 Givens 变换 G_{21}, 作用在 A 的最前面的两行上, 使得

$$G_{21}\begin{bmatrix} a_{11} \\ a_{21} \\ a_{31} \\ \vdots \\ a_{n1} \end{bmatrix} = \begin{bmatrix} \tilde{a}_{11} \\ 0 \\ a_{31} \\ \vdots \\ a_{n1} \end{bmatrix}$$

构造 Givens 变换 G_{31}, 作用在 $G_{21}A$ 的第 1 行和第 3 行上, 将 a_{31} 化为零. 由于 G_{31} 只改变第 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.
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, 将转化成一个上三角矩阵 R, 即

$$R = G_{n,n-1}\cdots G_{21}A, \square A = (G_{n,n-1}\cdots G_{21})^{\top} R \triangleq QR$$

† 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 House- holder 变换要多很多.
基于 Givens 变换的 QR 分解主要用于当矩阵的非零下三角元素相对较少时的情形, 比如对上 Hessenberg 矩阵进行 QR 分解
† 如果 $A \in \mathbb{R}^{m\times n}, m > n$, 仍然可以通过 Givens 变换进行 QR 分解

算法 3.4基于 Givens 变换的 QR 分解
%Given $A \in \mathbb{R}^{m\times n}$,compute$Q \in \mathbb{R}^{m\times\times m}$ and$R \in \mathbb{R}m \times n$ such that $A = QR$
% The upper triangular part of R is stored in the upper triangular part of A

1: Set $Q = I_{m\times m}$

2: for $k = 1$ to n do

3: for i = k + 1 to m do

4: $[c, s] = givens(a_{kk}, a_{ik})$

5: $$\begin{bmatrix} A(k,k:n) \\ A(i,k:n) \end{bmatrix} = G \begin{bmatrix} A(k,k:n) \\ A(i,k:n) \end{bmatrix} \text{ where } G = \begin{bmatrix} c & s \\ -s & c \end{bmatrix}$$

6: $$[Q(1:m,k), Q(1:m,i)] = [Q(1:m,k), Q(1:m,i)]G^{\top}$$

7: end for

8: end for

5.3.6 QR 分解的稳定性

基于 Householder 变换和 Givens 变换的 QR 分解都具有很好的数值稳定性, 基于 MGS 的 QR 分解也是向后稳定的
† 如果需要计算 Q, 则 MGS 的运算量相对较少, 因此当 A 的列向量具有很好的线性无关性时, 我们可以使用 MGS 来计算 QR 分解.
Björck [Björck 1967] 证明了, 通过 MGS 计算的矩阵 Q 满足

$$Q^{\top}Q = I + E_{MGS}, \quad \|E_{MGS}\|_2 \approx \varepsilon_u \kappa_2(A)$$

而 Householder 变换计算的矩阵 Q 满足

$$Q^{\top}Q = I + E_H, \quad \|E_H\|_2 \approx \varepsilon_u$$

因此, 如果正交性至关重要, 建议使用 Householder 变换, 特别是当 A 的列向量接近线性相关时

5.4 奇异值分解

定理 (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

$$U^*AV = \begin{bmatrix} \Sigma \\ 0 \end{bmatrix}, or A = U \begin{bmatrix} \Sigma \\ 0 \end{bmatrix} V^* \tag{67}$$

其中 $\Sigma = \text{diag}\,(\sigma_1, \sigma_2, \ldots, \sigma_n) \in \mathbb{R}^{n\times n}$, 且 $\sigma_1 \geq \ldots \geq \sigma_n \geq 0$

† 该定理也可以通过 Hermite 半正定矩阵的特征值分解来证明.

定义 分解 (3.6) 称为 A 的奇异值分解 (SVD),$\sigma_1, \sigma_2, \ldots, \sigma_n$ 称为 A 的奇异值, 它们是矩阵 A^*A 的特征值的平方根.

† 在不做特别说明的情况下, 我们总是假定 $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n \geq 0$
† 如果 $A \in \mathbb{R}^{m\times n}$ 是实矩阵, 则 U , V 也都可以是实矩阵 [Horn-Johnson1985].

奇异向量
矩阵 $U=[u_1,u_2,\ldots,u_m]$ 和 $V=[v_1,v_2,\ldots,v_n]$ 的列向量分别称为 A 的左奇异向量和右奇异向量, 即存在关系式

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

SVD 的其他形式

$$A=U\begin{bmatrix}\Sigma\\0\end{bmatrix}V^*=\sigma_1u_1v_1^*+\sigma_2u_2v_2^*+\cdots+\sigma_nu_nv_n^*$$

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

$$A=U_n\Sigma V^* \tag{68}$$

这就是所谓的细 SVD (thin SVD) 或降阶 SVD (reduced SVD), 有的文献将上式称为奇异值分解.
当 $k<n$ 时, 我们称

$$A_k=\sigma_1u_1v_1^*+\sigma_2u_2v_2^*+\cdots+\sigma_ku_kv_k^*$$

为 A 的截断 SVD (truncated SVD).

奇异值分解基本性质
定理设 $A=U\begin{bmatrix}\Sigma\\0\end{bmatrix}V^*$ 是 $A\in\mathbb{C}^{m\times n}(m\geq n)$ 的奇异值分解, 则

(1) A^*A 的特征值是 σ_i^2 , 对应的特征向量是 v_i

(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

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

(4) 若 $\mathrm{rank}(A)=r\leq n$,则 $\mathrm{Ran}(A)=\mathrm{span}\{u_1,u_2,\ldots,u_r\}$,$\mathrm{Ker}(A)=\mathrm{span}\{v_{r+1},v_{r+2},\ldots,v_n\}$

(5) 设 $x\in\mathbb{C}^n$,且 $\|x\|_2=1$ 和 $Y\in\mathbb{C}^{n\times n}$ 是酉矩阵,则 $\sigma_i(X^*AY)=\sigma_i(A)$

(6) (酉不变性)$X\in\mathbb{C}^{m\times m}$ 和 $Y\in\mathbb{C}^{n\times n}$ 是酉矩阵,则 $\sigma_i(X^*AY)=\sigma_i(A)$

定理 设 $A=U\Sigma V^*$ 是 $A\in\mathbb{C}^{n\times n}$ 的奇异值分解, 则:

(1) $|\det(A)|=\sigma_1\sigma_2\cdots\sigma_n$

(2) 若 A 非奇异, 则 $\|A^{-1}\|_2=\sigma_n^{-1},\kappa_2(A)=\sigma_1/\sigma_n$

(3) 若 A 是 Hermite 的, 且 $A=U\Lambda U^*$ 是 A 的酉特征值分解, 设 $|\lambda_1|\geq|\lambda_2|\geq\cdots\geq|\lambda_n|$, 则 $A=U\Sigma V$ 是 A 的奇异值分解, 其中 $\sigma_i=|\lambda_i|,v_i=\mathrm{sign}(\lambda_i)u_i$, 若 $\lambda_i=0$, 则 $v_i=u_i$

(4) 矩阵 $\begin{bmatrix} 0 & A^* \\ A & 0 \end{bmatrix}$ 的特征值是 $\pm\sigma_i$，对应单位特征向量 $\frac{1}{\sqrt{2}}\begin{bmatrix} v_i \\ \pm u_i \end{bmatrix}$

† 矩阵条件数取决于奇异值，不是特征值，见讲义上的例子

低秩逼近

定理设 $A = U_n \Sigma V^*$ 是 $A \in \mathbb{C}^{m\times n}$ 的奇异值分解. 令

$$A_k = \sum_{i=1}^{k} \sigma_i u_i v_i^*$$

则 A_k 是

$$\min_{B\in\mathbb{C}^{m\times n},\text{rank}(B)=k} \|A-B\|_2 \tag{69}$$

的一个解，且

$$\|A - A_k\|_2 = \sigma_{k+1}$$

此时，我们称 Ak 是 A 的一个秩 k 逼近
† 对于 Frobenius 范数，我们有类似的结论.

定理 (Weyl)设 $A, B \in \mathbb{C}^{m\times n}(m \geq n)$ 且 $\text{rank}(B) = k$，则有

$$\max_{x\in\text{Ker}(B),\|x\|_2=1} \|Ax\|_2 \geq \sigma_{k+1}(A) \tag{70}$$

和

$$\min_{x\in\text{Ker}(B),\|x\|_2=1} \|Ax\|_2 \leq \sigma_{n-k}(A) \tag{71}$$

因此，

$$\sigma_1(A-B) \geq \sigma_{k+1}(A), \quad \sigma_n(A+B) \leq \sigma_{n-k}(A) \tag{72}$$

且

$$\sigma_{i+j-1}(A) \leq \sigma_i(B) + \sigma_j(A-B), \quad i = 1, 2, \ldots, n, j = 1, 2, \ldots, n-i+1 \tag{73}$$

根据 Weyl 定理，我们可以得到矩阵奇异值的最小最大定理.

定理设 $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n \geq 0$ 是矩阵 $A \in \mathbb{C}^{m\times n}(m \geq n)$ 的奇异值，则

$$\sigma_k(A) = \min_{\dim(\mathcal{L})=n-k+1} \max_{x\in\mathcal{L},\|x\|_2=1} \|Ax\|_2, \quad k = 1, 2, \ldots, n$$

和

$$\sigma_k(A) = \max_{\dim(\mathcal{L})=k} \min_{x\in\mathcal{L},\|x\|_2=1} \|Ax\|_2, \quad k = 1, 2, \ldots, n$$

其中 $\mathcal{L}$ 表示 $\mathbb{C}^n$ 的一个子空间.

引理 (交错不等式)设 $A \in \mathbb{C}^{m\times n}, A_r$ 是由 A 去除 r 列 (或 r 行) 后得到的子矩阵，则

$$\sigma_i(A) \geq \sigma_i(A_r) \geq \sigma_{i+r}(A), \quad i = 1, 2, \ldots, \min\{m, n\}$$

这里，当下标 k 大于矩阵的维数时，我们令 $\sigma_k = 0$. 更进一步，如果 $B \in \mathbb{C}^{(m-r)\times(n-s)}$ 是 A 的子矩阵，则

$$\sigma_i(A) \geq \sigma_i(B) \geq \sigma_{i+r+s}(A)$$

引理设 $A \in \mathbb{C}^{n\times n}, 1 \leq k \leq n$ 则对任意的单位列正交矩阵 $U_k \in \mathbb{C}^{n\times k}$ 和 $V_k \in \mathbb{C}^{n\times k}$，有

$$\sigma_i(U_k^* A V_k) \leq \sigma_i(A), \quad i = 1, 2, \ldots, k \tag{74}$$

因此，

$$|\det(U_k^* A V_k)| \leq \sigma_1(A)\sigma_2(A)\cdots\sigma_k(A) \tag{75}$$

定理 (Weyl 不等式，1949)设 $A \in \mathbb{C}^{n\times n}$ 其奇异值和特征值分别为 $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n \geq 0$ 和 $|\lambda_1| \geq |\lambda_2| \geq \cdots \geq |\lambda_n|$ 则

$$|\lambda_1\lambda_2\cdots\lambda_k| \leq \sigma_1\sigma_2\cdots\sigma_k, \quad k = 1, 2, \ldots, n$$

且当 $k = n$ 时，等号成立.

奇异值扰动分析

定理 设 $A, E \in \mathbb{C}^{m\times n}(m \geq n)$，则

$$|\sigma_i(A + E) - \sigma_i(A)| \leq \|E\|_2, \quad i = 1, 2, \ldots, n$$

下面是上述定理的一个推论，也是 SVD 的一个重要应用

推论 设 $A \in \mathbb{C}^{n\times n}, \|\cdot\|$ 是任一相容矩阵范数，则对任意的 $\varepsilon > 0$ 总存在矩阵 A_ε，使得 $\|A - A_\varepsilon\| \leq \varepsilon$，其中 A_ε 具有互不相同的特征值.
因此，可对角化矩阵在所有矩阵组成的集合中是稠密的

5.5 线性最小二乘问题的求解方法

5.1 正规方程法

5.2 QR 分解法

5.3 奇异值分解法

5.5.1 正规方程法

定理 设 $A \in \mathbb{R}^{m\times n}(m \geq n)$，则 $x_* \in \mathbb{R}^n$ 是线性最小二乘问题的解当且仅当残量 $r = b - Ax_*$ 与 $\mathrm{Ran}(A)$ 正交，即 x_* 是下面的正规方程的解

$$A^\top(b - Ax) = 0 \quad \text{或} \quad A^\top Ax = A^\top b \tag{76}$$

由于

$$A^\top b \in \text{Ran}\left(A^\top\right) = \text{Ran}\left(A^\top A\right)$$

因此正规方程 $A^\top Ax = A^\top b$ 总是相容的，即最小二乘解总是存在的，且当 A 满秩时，这个解也是唯一的.

定理 设 $A \in \mathbb{R}^{m\times n}(m > n)$，则 $A^\top A$ 对称正定当且仅当 A 是列满秩的，即 $\text{rank}(A) = n$，此时，线性最小二乘问题的解是唯一的，其表达式为

$$x = \left(A^\top A\right)^{-1} A^\top b$$

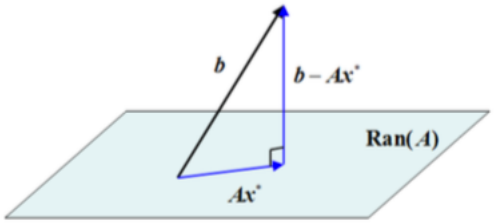
当 A 列满秩时，我们就可以使用 Cholesky 分解来求解正规方程，总的运算量大约为 $mn^2 + 1/3n^3 + \mathbb{O}(n^2)$，其中大部分的运算量 (mn^2) 是用来计算 A^TA(由于 A^TA 对称，因此只需计算其下三角部分).

最小二乘的几何含义

我们可以把 b 写成

$$b = Ax_* + r, \text{其中} r \perp \text{Ran}(A) \tag{77}$$

所以 Ax_* 就是 b 在 $Ran(A)$ 上的正交投影:



† 最小二乘解可能并不唯一，但 b 的上述分解总是唯一的.

最小二乘与鞍点问题

线性最小二乘问题 (3.1) 等价于

$$A^\top r = 0, \quad r = b - Ax$$

即

$$\begin{bmatrix} I & A \\ A^\top & 0 \end{bmatrix} \begin{bmatrix} r \\ x \end{bmatrix} = \begin{bmatrix} b \\ 0 \end{bmatrix} \tag{78}$$

这就是线性最小二乘问题 (3.1) 的增广方程 (augmented system). 事实上，方程组 (16) 是下面方程组的一种特殊情形

$$\begin{bmatrix} B & A \\ A^\top & 0 \end{bmatrix} \begin{bmatrix} r \\ x \end{bmatrix} = \begin{bmatrix} f \\ g \end{bmatrix}$$

其中 $B \in \mathbb{R}^{m\times m}$ 对称半正定. 这就是通常所说的鞍点问题. 这个方程组存在唯一解当且仅当 A 列满秩且矩阵 [B, A] 行满秩.
如果 B 对称正定, 则 $r = B^{-1}(f - Ax)$, 代入第二个方程可得

$$A^\top B^{-1} A x = A^\top B^{-1} f - g$$

这就是广义的正规方程. 其所对应的广义线性最小二乘问题是

$$\min_{x\in\mathbb{R}^n} \frac{1}{2}\|Ax - f\|_{B^{-1}}^2 + g^\top x$$

其中范数 $\|\cdot\|_{B^{-1}}$ 的定义是 $\|x\|_{B^{-1}}^2 = x^\top B^{-1} x$.

5.5.2 QR 分解法

假定 $A \in \mathbb{R}^{m\times n}(m \geq n)$ 满秩,用三种不同方法来推导

(1) 将 Q 的扩充成一个正交矩阵, 记为 $[Q, \hat{Q}] \in \mathbb{R}^{m\times m}$ 于是有

$$\begin{aligned}
\|Ax - b\|_2^2 &= \left\|[Q,\hat{Q}]^\top (Ax - b)\right\|_2^2 \\
&= \left\|[Q,\hat{Q}]^\top (QRx - b)\right\|_2^2 \\
&= \left\| \begin{bmatrix} Rx - Q^\top b \\ -\hat{Q}^\top b \end{bmatrix} \right\|_2^2 \\
&= \left\|Rx - Q^\top b\right\|_2^2 + \left\|\hat{Q}^\top b\right\|_2^2 \geq \left\|\hat{Q}^\top b\right\|_2^2
\end{aligned}$$

等号成立当且仅当 $Rx = Q^\top b$ 所以最小二乘解为

$$x_* = R^{-1} Q^\top b$$

(2) 将 b 写成 $b = \left(QQ^\top + I - QQ^\top\right) b$,则

$$\begin{aligned}
Ax - b &= Ax - \left(QQ^\top + I - QQ^\top\right) b \\
&= \left(Ax - QQ^\top b\right) - \left(I - QQ^\top\right) b
\end{aligned}$$

由于 QQ^T 是 $Ran(A)$ 上的正交投影,因此 $Ax - QQ^\top b$ 与 $\left(I - QQ^\top\right) b$ 正交. 所以

$$\begin{aligned}
\|Ax - b\|_2^2 &= \left\|Ax - QQ^\top b\right\|_2^2 + \left\|\left(I - QQ^\top\right) b\right\|_2^2 \\
&= \left\|Rx - Q^\top b\right\|_2^2 + \left\|\left(I - QQ^\top\right) b\right\|_2^2 \\
&\geq \left\|\left(I - QQ^\top\right) b\right\|_2^2
\end{aligned}$$

等号成立当且仅当 $Rx = Q^\top b$ 所以最小二乘解为

$$x_* = R^{-1} Q^\top b$$

(3) 解正规方程. 由定理 5.1 可知, 最小二乘解为

$$\begin{aligned} x_* &= \left(A^\top A\right)^{-1} A^\top b \\ &= \left(R^\top Q^\top Q R\right)^{-1} R^\top Q^\top b \\ &= \left(R^\top R\right)^{-1} R^\top Q^\top b \\ &= R^{-1} Q^\top b \end{aligned}$$

† 用 QR 分解来求最小二乘解的运算量大约为 $2mn^2$ (如果采用 Householder 变换的话, 运算量大约为 $2mn^2 - 2/3n^3$). 当 $m \gg n$ 时, 大约为正规方程的两倍. 当 $m = n$ 时, 几乎相同.
† 通常 QR 算法比较稳定, 是求解最小二乘问题的首选方法, 特别是当 A 条件数较大 (病态) 时.

5.5.3 奇异值分解法

设 $A \in \mathbb{R}^{m\times n}$ 列满秩, $A = U \begin{bmatrix} \Sigma \\ 0 \end{bmatrix} V^\top$ 是 A 的奇异值分解, $U = \left[U_n, \tilde{U}\right]$ 则

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

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

$$x = \left(\Sigma V^\top\right)^{-1} U_n^\top b = V \Sigma^{-1} U_n^\top b$$

这就是线性最小二乘问题 (3.1) 的解.

5.6 广义逆与最小二乘

5.6.1 广义逆的定义

广义逆的概念最早由 Moore 于 1920 年提出:

设 $A \in \mathbb{C}^{m\times n}$, 若 $X \in \mathbb{C}^{n\times m}$ 满足

$$AX = P_{\mathrm{Ran}(A)}, \quad XA = P_{\mathrm{Ran}(X)} \tag{79}$$

即 AX 和 XA 分别为 Ran(A) 和 Ran(X) 上的正交投影, 则称 X 是 A 的广义逆.

1955 年, Penrose 利用四个矩阵方程给出了广义逆的定义.

定义 设 $A \in \mathbb{C}^{m\times n}$,若 $X \in \mathbb{C}^{n\times m}$ 满足

$$AXA = A \tag{80}$$

$$XAX = X \tag{81}$$

$$(AX)^* = AX \tag{82}$$

$$(XA)^* = XA \tag{83}$$

则称 X 为 A 的广义逆 (或 Moore-Penrose 逆, 简称 MP 逆), 记为 $A\dagger$

† 可以证明, 以上两个定义是等价的.

† 若 $A \in \mathbb{C}^{m\times n}$ 非奇异,则 $A^\dagger = A^{-1}$

定义 设 $A \in \mathbb{C}^{m\times n}$ 则满足矩阵方程 (18)-(21) 的矩阵 $X \in \mathbb{C}^{n\times m}$ 存在唯一,即广义逆存在且惟一.

广义逆与 SVD
设 $A \in \mathbb{C}^{m\times n}\,\mathrm{rank}(A) = r > 0, A$ 的 SVD 为

$$A = U\begin{bmatrix}\Sigma_1 & 0\\ 0 & 0\end{bmatrix}V^*, \quad \Sigma_1 = \mathrm{diag}\,(\sigma_1, \sigma_2, \ldots, \sigma_r) \in \mathbb{R}^{r\times r}$$

则容易验证

$$A^\dagger = V\begin{bmatrix}\Sigma_1^{-1} & 0\\ 0 & 0\end{bmatrix}U^*$$

5.7 广义逆的基本性质

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

(1) $(A^\dagger)^\dagger = A$

(2) $(A^T)^\dagger = (A^\dagger)^T, (A^*)^\dagger = (A^\dagger)^*$

(3) $\mathrm{rank}(A) = \mathrm{rank}\left(A^\dagger\right) = \mathrm{rank}\left(A^\dagger A\right)$

(4) $(AA^*)^\dagger = (A^*)^\dagger A^\dagger, (A^*A)^\dagger = A^\dagger (A^*)^\dagger$

(5) $(AA^*)^\dagger AA^* = AA^\dagger, (A^*A)^\dagger A^*A = A^\dagger A$

(6) $A^{\dagger}=\left(A^{*}A\right)^{\dagger}A^{*}=A^{*}\left(AA^{*}\right)^{\dagger}$ 特别地, 若 A 列满秩, 则 $A^{\dagger}=\left(A^{*}A\right)^{-1}A^{*}$. 若 A 行满秩, 则 $A^{\dagger}=A^{*}\left(AA^{*}\right)^{-1}$

(7) 若 U, V 是酉矩阵, 则 $(UAV)^{\dagger}=V^{*}A^{\dagger}U^{*}$

† 一般来说, 当 A, B 是方阵时,

- $(AB)^{\dagger}\neq B^{\dagger}A^{\dagger}$
- $AA^{\dagger}\neq A^{\dagger}A$
- $\left(A^{k}\right)^{\dagger}\neq\left(A^{\dagger}\right)^{k}$
- A 和 $A^{\dagger}$ 的非零特征值并不是互为倒数

设 $A\in\mathbb{C}^{m\times n}$, 则

$$\begin{aligned}&\mathrm{Ran}\left(AA^{\dagger}\right)=\mathrm{Ran}\left(AA^{*}\right)=\mathrm{Ran}(A)\\&\mathrm{Ran}\left(A^{\dagger}A\right)=\mathrm{Ran}\left(A^{*}A\right)=\mathrm{Ran}\left(A^{*}\right)=\mathrm{Ran}\left(A^{\dagger}\right)\\&\mathrm{Ker}\left(AA^{\dagger}\right)=\mathrm{Ker}\left(AA^{*}\right)=\mathrm{Ker}\left(A^{*}\right)=\mathrm{Ker}\left(A^{\dagger}\right)\\&\mathrm{Ker}\left(A^{\dagger}A\right)=\mathrm{Ker}\left(A^{*}A\right)=\mathrm{Ker}(A)\end{aligned}$$

5.7.1 广义逆的计算

- 利用 SVD, 但运算量较大.
- 利用满秩分解
  定理 设 $A\in\mathbb{R}^{m\times n}$

  (1) 若 A 是列满秩矩阵, 则 $A^{\dagger}=\left(A^{*}A\right)^{-1}A^{*}$
  (2) 若 A 是行满秩矩阵, 则 $A^{\dagger}=A^{*}\left(AA^{*}\right)^{-1}$
  (3) 若 A 的秩是 $r\leq\min\{m,n\}$ 且其满秩分解为 $A=FG$ 其中 $F\in\mathbb{R}^{m\times r},G\in\mathbb{R}^{r\times n}$, 则
  $$A^{\dagger}=G^{\dagger}F^{\dagger}=G^{*}\left(GG^{*}\right)^{-1}\left(F^{*}F\right)^{-1}F^{*}$$
- 利用 QR 分解
  定理 设 $A\in\mathbb{R}^{m\times n}(m\geq n)$ 是列满秩矩阵, 其 QR 分解为 $A=QR$, 其中 $Q\in\mathbb{R}^{m\times n},R\in\mathbb{R}^{n\times n}$, 则
  $$A^{\dagger}=R^{-1}Q^{*}$$
- 其他算法
  其他比较重要的算法有: Greville 递推算法, Cline 算法等.

5.7.2 广义逆与线性最小二乘

定理 设 $A \in \mathbb{R}^{m\times n}$ 则线性最小二乘问题的解为

$$x = A^{\dagger}b + (I - P_{A^{\top}})\, z, \quad \forall z \in \mathbb{R}^n \tag{84}$$

† 通常，线性最小二乘问题的解 (22) 不是唯一的，但当 A 列满秩时，$P_{A^{\top}} = I$，此时解唯一.

定理 设 $A \in \mathbb{R}^{m\times n}$ 的解集为 $\mathcal{S}$，则

$$\min_{x\in\mathcal{S}} \|x\|_2 \tag{85}$$

存在唯一解, 即满足 (23) 的线性最小二乘问题 (3.1) 的解存在且唯一

5.8 最小二乘扰动分析

定理 设 $A \in \mathbb{R}^{m\times n}$ 且 $\mathrm{rank}(A) = n$ 设 x 是线性最小二乘问题 (3.1) 的解，$\tilde{x}$ 极小化 $\|(A+\delta A)\tilde{x} - (b+\delta b)\|_2$，则

$$\frac{\|\tilde{x} - x\|_2}{\|x\|_2} \le \varepsilon \cdot \left\{ \frac{2\kappa_2(A)}{\cos\theta} + \kappa_2^2(A)\tan\theta \right\} + O\left(\varepsilon^2\right)$$

其中 $\kappa_2(A) = \sigma_1(A)/\sigma_n(A), \theta$ 为 b 与 $\mathrm{Ran}(A)$ 的夹角，

$$\varepsilon \triangleq \max\left\{ \frac{\|\delta A\|_2}{\|A\|_2}, \frac{\|\delta b\|_2}{\|b\|_2} \right\}$$

并假定 $\varepsilon \cdot \kappa_2(A) < 1$(确保 $A+\delta A$ 满秩, 从而 $\tilde{x}$ 唯一确定)
我们记

$$\kappa_{LS} \triangleq \frac{2\kappa_2(A)}{\cos\theta} + \kappa_2^2(A)\tan\theta$$

这就是最小二乘问题的条件数. 当 $\theta = 0, b \in \mathrm{Ran}(A)$，此时 $\kappa_{LS} = 2\kappa_2(A)$，当 $\theta = \pi/2$ 时，$b \perp \mathrm{Ran}(A)$，此时最小二乘解为 $x = 0$，而 $\kappa_{LS} = \infty$；当 $0 < \theta < \pi/2$ 时，$\kappa_{LS} = O\left(\kappa_2^2(A)\right)$

定义残量 $r = b - Ax, \tilde{r} = (b+\delta b) - (A+\delta A)\tilde{x}$，我们有下面的性质 [Higham2002]

$$\frac{\|\tilde{r} - r\|_2}{\|r\|_2} \le 1 + 2\varepsilon \cdot \kappa_2(A)$$

当我们使用 QR 分解或 SVD 分解求解最小二乘问题时, 由于采用的是正交变换, 它们都是数值稳定的. 而正规方程涉及求解方程组 $A^{\top}Ax = A^{\top}b$, 其精度依赖于条件数 $\kappa_2\left(A^{\top}A\right) = \kappa_2^2(A)$, 因为其误差是以 $\kappa_2^2(A)$ 倍数增长. 因此当 A 的条件数较大时, 正规方程法的精度会大大降低.

5.9 最小二乘问题的推广及其应用

5.9.1 正则化

在求解超定线性方程组时, 我们极小化 $\|Ax-b\|_2^2$ 而对于欠定线性方程组, 我们极小化的是 $\|x\|_2^2$. 两者合起来就是

$$\min_{x\in\mathbb{R}^n\times n} \frac{1}{2}\|Ax-b\|_2^2+\frac{\alpha}{2}\|x\|_2^2 \tag{86}$$

其中 $\alpha>0$ 是权系数. 对应的目标函数记为

$$J(x)=\|Ax-b\|_2^2+\alpha\|x\|_2^2$$

当 $\alpha>0$ 时，$J(x)$ 是一个严格凸的二次函数, 因此存在唯一的最小值点. 关于 x 求导后, 令其等于零, 可得

$$\left(A^\top A+\alpha I\right)x=A^\top b$$

由于 $A^\top A+\alpha I$ 对称正定, 故非奇异. 所以问题 (24) 的唯一解为

$$x=\left(A^\top A+\alpha I\right)^{-1}A^T b$$

5.9.2 加权正则化

一类应用更广泛的问题是下面的加权正则化问题

$$\min_{x\in\mathbb{R}^n\times n} \frac{1}{2}\|Ax-b\|_2^2+\frac{\alpha}{2}\|Wx\|_2^2 \tag{87}$$

其中 $W\in\mathbb{R}^{p\times n}$ 是广义加权矩阵, 可以是非负对角矩阵, 对称正定矩阵, 也可以是一般矩阵 (如一阶差分算子, 二阶差分算子等). 注意, W 不一定要求是方阵.
经过类似的推导，问题 (25) 的解满足

$$\left(A^\top A+\alpha W^\top W\right)x=A^\top b$$

如果 $A^\top A+\alpha W^\top W$ 非奇异, 则存在唯一解

$$x=\left(A^\top A+\alpha W^\top W\right)^{-1}A^T b$$

5.9.3 约束最小二乘问题

考虑带有约束的最小二乘问题

$$\begin{aligned}&\min\nolimits_{x\in\mathbb{R}^{n\times n}} \tfrac{1}{2}\|Ax-b\|_2^2\\ &\text{s.t.}\quad Bx=f\end{aligned} \tag{88}$$

其中 $Bx=f$ 是约束条件. 对应的 Lagrange 函数为

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

分别对 x 和 λ 求一阶导数, 并令其等于零, 可得

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

如果 $A^\top A$ 非奇异, 且 B 行满秩, 则存在唯一解

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

5.9.4 应用: 多项式数据拟合

已知 n 个点 $\{(t_i, f_i)\}_{i=1}^n$, 寻找一个低次多项式来拟合这些数据. 设拟合多项式为

$$p(x) = a_0 + a_1 t + a_2 t^2 + \cdots + a_m t^m$$

通常 $m \ll n$ 将上述 n 个点代入可得

$$\begin{bmatrix} 1 & t_1 & t_1^2 & \cdots & t_1^m \\ 1 & t_2 & t_2^2 & \cdots & t_2^m \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & t_n & t_n^2 & \cdots & t_n^m \end{bmatrix}_{n\times(m+1)} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_m \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_n \end{bmatrix} \quad or \quad Ax = f$$

其中 $A \in \mathbb{R}^{n\times(m+1)}, x = [a_0, a_1, \ldots, a_m]^\top$, 由于 $m \ll n$ 该方程组是超定的, 解通常是不存在的. 因此, 我们寻找一个近似解, 使得残量 $\|f - Ax\|_2$ 最小, 即求解最小二乘问题

$$\min_{x\in\mathbb{R}^{m+1}} \|f - Ax\|_2^2$$

5.9.5 应用: 线性预测

预测一个时间序列的未来走向, 一个常用方法就是线性预测. 假定在 t_k 时刻的值 f_k 线性依赖于其前 m 个时刻的值 $f_{k-1}, f_{k-2}, \ldots, f_{k-m}$, 即

$$f_k = a_1 f_{k-1} + a_2 f_{k-2} + \cdots + a_m f_{k-m} \tag{89}$$

现在已经测得该时间序列的前 n 个值 $f_i, i = 0, 1, 2, \ldots, n-1$, 其中 $n \gg m$.如何预测其未来的取值? 将现有的数据代入上述关系式可得

$$\begin{bmatrix} f_{m-1} & f_{m-2} & f_{m-3} & \cdots & f_0 \\ f_m & f_{m-1} & f_{m-2} & \cdots & f_1 \\ \vdots & \vdots & \vdots & & \vdots \\ f_{n-2} & f_{n-3} & f_{n-4} & \cdots & f_{n-m-1} \end{bmatrix}_{(n-m)\times m} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_m \end{bmatrix} = \begin{bmatrix} f_m \\ f_{m+1} \\ \vdots \\ f_{n-1} \end{bmatrix} \square Ax = f$$

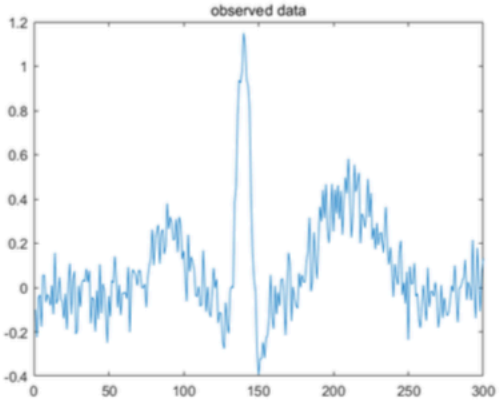
这也是一个超定问题, 因此需要求解最小二乘 $\min_{x\in\mathbb{R}^m} \|f - Ax\|_2^2$

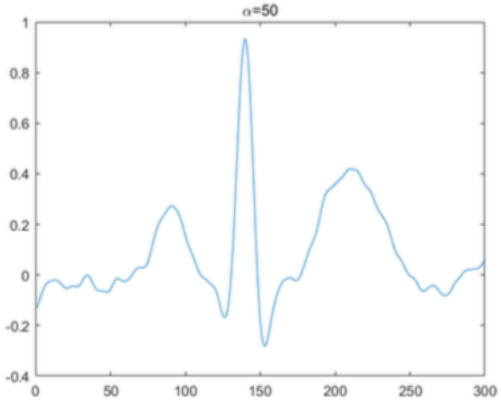
5.9.6 应用：信号处理

- 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．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．

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

其中 D 是离散的二阶导算子．





- 图像去模糊
一般模型：

$$f = K(x) + n$$

其中 x 表示真实图像，f 表示观察到的图像，$K(\cdot)$ 是一个卷积算子，代表模糊机制，n 代表噪声．

- 图像压缩

5.10 非对称矩阵的特征值问题

非对称矩阵特征值/特征向量的计算
基本约定 1：$A \in \mathbb{R}^{n\times n}$、非对称、稠密
基本约定 2: $|\lambda_1| \geq |\lambda_2| \geq \cdots \geq |\lambda_n| \geq 0$

本讲主要讨论如何计算 A 的全部特征值和/或特征向量
主要介绍以下方法：

- 幂迭代方法
- 反迭代方法 (位移策略，Rayleigh 商迭代)
- 正交迭代方法
- QR 方法

关于稠密矩阵特征值计算的参考资料有:

- J.H.Wilkinson,TheAlgebraicEigenvalueProblem,1965
- B.N.Parlett,TheSymmetricEigenvalueProblem,2ndEds.,1998
- G.W.Stewart,MatrixAlgorithms,VolII:Eigensystems,2001
- G.H.GolubandC.F.VanLoan,MatrixComputations,2013
- P.Arbenz,Thecourse252-0504-00G,

Numerical Methods for Solving Large Scale Eigenvalue Problems, 2018. (该课程的主页)

5.11 幂迭代

幂迭代 是计算特征值和特征向量的一种简单易用的算法.
虽然简单, 但它却建立了计算特征值和特征向量的算法的一个基本框架.
算法 1.1 幂迭代算法 (Power Iteration)

1: Choose an initial guess $x(0)$ with $||x(0)||_2 = 1$
2: set $k = 0$
3: while not convergence do
4: $y^{(k+1)} = Ax^{(k)}$
5: $x^{(k+1)} = y^{(k+1)} / \left\|y^{(k+1)}\right\|_2$
6: $\mu_{k+1} = \left(x^{(k+1)}, Ax^{(k+1)}\right)$
7: $k = k + 1$
8: end while

幂迭代的收敛性
假设 1:$A \in \mathbb{R}^{n\times n}$ 可对角化, 即 $A = V\Lambda V^{-1}$, 其中

$$\Lambda = \text{diag}\left(\lambda_1, \ldots, \lambda_n\right), \quad V = [v_1, \ldots, v_n] \in \mathbb{C}^{n\times n}, \quad \|v_i\|_2 = 1$$

假设 2:$|\lambda_1| > |\lambda_2| \geq |\lambda_3| \geq \cdots \geq |\lambda_n|$ 由于 V 的列向量组构成 $\mathbb{C}^n$ 的一组基, 因此 $x^{(0)}$ 可表示为

$$x^{(0)} = \alpha_1 v_1 + \alpha_2 v_2 + \cdots + \alpha_n v_n = V\left[\alpha_1, \alpha_2, \ldots, \alpha_n\right]^\top$$

我们假定 $\alpha_1 \neq 0$，即 $x^{(0)}$ 不属于 $\operatorname{span}\{v_2, v_3, \ldots, v_n\}$ (由于 $x^{(0)}$) 是随机选取的，从概率意义上讲，这个假设通常是成立的).
于是我们可得

$$A^k x^{(0)} = \left(V\Lambda V^{-1}\right)^k V \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_n \end{bmatrix} = V\Lambda^k \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_n \end{bmatrix} = V \begin{bmatrix} \alpha_1\lambda_1^k \\ \alpha_2\lambda_2^k \\ \vdots \\ \alpha_n\lambda_n^k \end{bmatrix}$$

$$= \alpha_1\lambda_1^k V \begin{bmatrix} 1 \\ \frac{\alpha_2}{\alpha_1}\left(\frac{\lambda_2}{\lambda_1}\right)^k \\ \vdots \\ \frac{\alpha_n}{\alpha_1}\left(\frac{\lambda_n}{\lambda_1}\right)^k \end{bmatrix}$$

又 $|\lambda_i/\lambda_1| < 1, i = 2, 3, \ldots, n$，所以

$$\lim_{k\to\infty}\left(\frac{\lambda_i}{\lambda_1}\right)^k = 0, \quad i = 2, 3, \ldots, n$$

故当 k 趋向于无穷大时，向量

$$\left[1, \frac{\alpha_2}{\alpha_1}\left(\frac{\lambda_2}{\lambda_1}\right)^k, \ldots, \frac{\alpha_n}{\alpha_1}\left(\frac{\lambda_n}{\lambda_1}\right)^k\right]^\top, \quad k = 0, 1, 2, \ldots$$

收敛到 $e_1 = [1, 0, \ldots, 0]^\top$
所以向量 $x^{(k)} = A^k x^{(0)} / \left\|A^k x^{(0)}\right\|_2$ 收敛到 $\pm v_1$，即 λ_1 的特征向量. 而 $\mu_k = \left(x^{(k)}\right)^* A x^{(k)}$ 则收敛到 $v_1^* A v_1 = \lambda_1$ † 幂迭代的收敛快慢取决于 $|\lambda_2/\lambda_1|$ 的大小，$|\lambda_2/\lambda_1|$ 越小，收敛越快.

- 幂迭代只能用于计算 (模) 最大的特征值和其相应的特征向量
- 当 $|\lambda_2/\lambda_1|$ 接近于 1 时，收敛速度会非常慢
- 如果模最大的特征值是一对共轭复数，则幂迭代可能会失效.

加速技巧：位移策略
出发点：加快幂迭代算法的收敛速度 $\Longleftrightarrow$ 尽可能地减小 $|\lambda_2/\lambda_1|$
位移策略：计算 $A - \sigma I$ 的特征值
我们称 σ 为位移，满足

(1) $\lambda_1 - \sigma$ 是 $A - \sigma I$ 的模最大特征值

(2) $\max_{2\le i\le n}\left|\frac{\lambda_i - \sigma}{\lambda_1 - \sigma}\right|$ 尽可能地小

其中第一个条件保证最后所求得的特征值是我们所要的，第二个条件用于加快幂迭代的收敛速度.
缺点：(1)σ 很难选取；(2) 加速效果有限
改进：与反迭代相结合，能起到很好的加速效果

5.12 反迭代

用幂迭代求 A^{-1} 的模最小特征值，这就是反迭代
算法 2.1 反迭代算法 (Inverse Iteration)

1: Choose an initial guess $x(0)$ with $||x(0)||_2 = 1$

2: set $k = 0$

3: while not convergence do

4: $y^{(k+1)} = (A - \sigma I)^{-1} x^{(k)}$

5: $x^{(k+1)} = y^{(k+1)} / \left\| y^{(k+1)} \right\|_2$

6: $\mu_{k+1} = \left(x^{(k+1)}, Ax^{(k+1)} \right)$

7: $\sigma = \mu_{k+1}, k = k + 1$

8: end while

显然：μ_k 收敛到 σ 最近的特征值，$x(k)$ 收敛到对应的特征向量
†理论上，反迭代 + 位移策略，可以计算矩阵的任意一个特征值
优点：

- 若 σ 与某个特征值 λ_k 非常接近，则反迭代算法的收敛速度非常快
- 只要选取合适的位移 σ，就可以计算 A 的任意一个特征值.

缺点：

- 每步迭代需要解一个线性方程组 $(A - \sigma I)y^{(k+1)} = x^{(k)}$ 这需要对 $A - \sigma I$ 做 LU 或 PLU 分解
- 与幂迭代一样，反迭代算法一次只能求一个特征值
- 怎样选取位移 σ? $\rightarrow$ Rayleigh 商动态选取，自动调整

5.12.1 Rayleigh 商迭代

出发点：使得 σ 与所求的特征值越靠近越好.

期望能直接给出一个理想位移是不太现实的. 比较现实的方法就是动态调整，使得位移逐渐靠近某个特征值.
Rayleigh 商迭代：以 Rayleigh 商 μ_k 为第 k 步的位移
理由：μ_k 会逐渐收敛到某个特征值.

算法 2.2 Rayleigh 商迭代 (Rayleigh Quotient Iteration, RQI)

1. Choose an initial vector $x(0)$ with $||x(0)||_2 = 1$
2. set $k = 0$
3. compute $\sigma = \left(x^{(0)}\right)^* Ax^{(0)}$
4. while not convergence do
5. $\quad y^{(k+1)} = (A - \sigma I)^{-1} x^{(k)}$
6. $\quad x^{(k+1)} = y^{(k+1)} / \left\| y^{(k+1)} \right\|_2$
7. $\quad \mu_{k+1} = \left(x^{(k+1)}, Ax^{(k+1)}\right)$
8. $\quad \sigma = \mu_{k+1}$
9. $\quad k = k + 1$
10. end while

RQI 算法的收敛性

一般来说, 如果 Rayleigh 商迭代收敛到 A 的一个单特征值, 则至少是二次收敛的, 即具有局部二次收敛性. 如果 A 是对称的, 则能达到局部三次收敛, 详情见后面的对称特征值问题.

缺点:
由于每次迭代的位移是不同的, 因此每次迭代需要求解一个不同的线性方程组, 这使得运算量大大增加.
因此通常应用于 三对角矩阵 的特征值计算

5.13 正交迭代

出发点: 同时计算多个特征值/特征向量
策略: 同时采用多个初始向量, 希望收敛到 A 的一个不变子空间
算法 3.1 正交迭代算法 (Orthogonal Iteration)

1: Choose an initial vector$n \times p$column orthogonal matrix Z_0
2: set $k = 0$
3: while not convergence do
4: $\quad$ compute $Y^{(k+1)} = AZ_{(k)}$
5: $\quad Y_{(k+1)} = Z_{(k+1)} \hat{R}_{k+1}$
6: $\quad k = k + 1$
7: end while

说明:
在算法中使用 QR 分解是为了保持 z_k 的列正交性, 使得其列向量组构成子空间 span$\{A^k Z_0\}$ 的一组正交基. 一方面提高算法的数值稳定性, 另一方面避免所有列都收敛到最大特征值所对应的特征向量.
收敛性分析
假设 A 是可对角化的, 即 $A = V\Lambda V^{-1}$, 其中 $\Lambda = \mathbf{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n)$, 且 $|\lambda_1| \geq \cdots \geq |\lambda_p| > |\lambda_{p+1}| \geq \cdots \geq |\lambda_n|$. 则可得

$$\mathbf{span}\{Z_k\} = \mathbf{span}\{Y_k\} = \mathbf{span}\{AZ_{k-1}\}, \quad k = 1, 2, \ldots$$

由此可知

$$\mathbf{span}\{Z_k\} = \mathbf{span}\left\{A^k Z_0\right\} = \mathbf{span}\left\{V\Lambda^k V^{-1} Z_0\right\}$$

我们注意到

$$\Lambda^k V^{-1} Z_0 = \lambda_p^k \begin{bmatrix} (\lambda_1/\lambda_p)^k & & & & \\ & \ddots & & & \\ & & 1 & & \\ & & & \ddots & \\ & & & & (\lambda_n/\lambda_p)^k \end{bmatrix} V^{-1} Z_0 \triangleq \lambda_p^k \begin{bmatrix} W_p^{(k)} \\ W_{n-p}^{(k)} \end{bmatrix}$$

由于当 $i > p$ 时有 $|\lambda_i/\lambda_p| < 1$,所以当 k 趋于无穷大时,$W_{n-p}^{(k)}$ 趋向于 0,令 $V = [V_p, V_{n-p}]$, 则

$$V\Lambda^k V^{-1} Z_0 = \lambda_p^k [V_p, V_{n-p}] \begin{bmatrix} W_p^{(k)} \\ W_{n-p}^{(k)} \end{bmatrix} = \lambda_p^k \left(V_p W_p^{(k)} + V_{n-p} W_{n-p}^{(k)}\right)$$

所以当 $k \to \infty$ 时,有

$$\begin{aligned} \mathbf{span}\{Z_k\} = \mathbf{span}\left\{V\Lambda^k V^{-1} Z_0\right\} = \mathbf{span}\left\{V_p W_p^{(k)} + V_{n-p} W_{n-p}^{(k)}\right\} \\ \to \mathbf{span}\left\{V_p W_p^{(k)}\right\} = \mathbf{span}\{V_p\} \end{aligned}$$

即 span$\{Z_k\}$ 趋向于 A 的一个 p 维不变子空间 span$\{V_p\}$

定理 给定正整数 $p(1 \leq p \leq n)$, 考虑算法 3.1, 假设 A 是可对角化的, 且 $|\lambda_1| \geq \cdots \geq |\lambda_p| > |\lambda_{p+1}| \geq \cdots \geq |\lambda_n|$. 则 span$\{Z_k\}$ 收敛到 A 的一个 p 维不变子空间.
说明:
如果 A 不可对角化, 利用 Jordan 标准型, 可以到同样的结论, 见 [Watkins 2007, Watkins-Elsner 1991].
† 在正交迭代中, 如果我们取 $Z_0 = I$, 则可得到一类特殊的正交迭代算法. 此时, 在一定条件下, 正交迭代会收敛到 A 的 Schur 标准型.

5.14 QR 迭代

5.14.1 算法介绍

基本思想:通过不断的正交相似变换, 将 A 转化为 (拟) 上三角形式
算法 4.1 QR 迭代算法 (QR Iteration)

1: Set $A_1 = A$ and $k = 1$

2: while not convergence do

3: $[Q_k, R_k] = qr(A_k)$

4: compute $A_{k+1} = R_k Q_k$

5: $k = k + 1$

6: end while

正交相似性在 QR 迭代算法中, 我们有

$$A_{k+1} = R_k Q_k = \left(Q_k^\top Q_k\right) R_k Q_k = Q_k^\top \left(Q_k R_k\right) Q_k = Q_k^\top A_k Q_k$$

由这个递推关系可得

$$A_{k+1} = Q_k^\top A_k Q_k = \cdots = Q_k^\top Q_{k-1}^\top \cdots Q_1^\top A Q_1 \cdots Q_{k-1} Q_k$$

记 $\tilde{Q}_k = Q_1 \cdots Q_{k-1} Q_k = \left[\begin{array}{cccc} \tilde{q}_1^{(k)}, & \tilde{q}_2^{(k)}, \ldots, \tilde{q}_n^{(k)} \end{array}\right]$ 则

$$A_{k+1} = \tilde{Q}_k^\top A \tilde{Q}_k \tag{90}$$

即 A_{k+1} 与 A 正交相似

5.14.2 QR 迭代与幂迭代的关系

记 $\tilde{R}_k = R_k R_{k-1} \cdots R_1$, 则有

$$\begin{aligned} \tilde{Q}_k \tilde{R}_k &= \tilde{Q}_{k-1} \left(Q_k R_k\right) \tilde{R}_{k-1} = \tilde{Q}_{k-1} \left(A_k\right) \tilde{R}_{k-1} \\ &= \tilde{Q}_{k-1} \left(\tilde{Q}_{k-1}^\top A \tilde{Q}_{k-1}\right) \tilde{R}_{k-1} \\ &= A \tilde{Q}_{k-1} \tilde{R}_{k-1} \end{aligned}$$

由此递推下去, 即可得

$$\tilde{Q}_k \tilde{R}_k = A^{k-1} \tilde{Q}_1 \tilde{R}_1 = A^{k-1} Q_1 R_1 = A^k$$

故

$$\tilde{Q}_k \tilde{R}_k e_1 = A^k e_1$$

假设 $|\lambda_1| > |\lambda_2| \geq \cdots \geq |\lambda_n|$, 则当 k 充分大时, $A^k e_1$ 收敛到 A 的模最大特征值 λ_1 所对应的特征向量.
$\rightarrow$ 故 $\tilde{Q}_k$ 的第一列 $\tilde{q}_1^{(k)}$ 也收敛到 λ_1 所对应的特征向量
因此, 当 k 充分大时, $A\tilde{q}_1^{(k)} \rightarrow \lambda_1 \tilde{q}_1^{(k)}$
由 $A_{k+1} = \tilde{Q}_k^\top A \tilde{Q}_k$ 可知 A_{k+1} 的第一列

$$A_{k+1}(:, 1) = \tilde{Q}_k^\top A \tilde{q}_1^{(k)} \rightarrow \lambda_1 \tilde{Q}_k^\top \tilde{q}_1^{(k)} = \lambda_1 e_1$$

结论
A_{k+1} 的第一列的第一个元素收敛到 λ_1, 而其他元素都趋向于 0. 收敛速度取决于 $|\lambda_2/\lambda_1|$ 的大小

5.14.3 QR 迭代与反迭代的关系

观察 $\tilde{Q}_k$ 的最后一列. 由 $A_{k+1}=\tilde{Q}_k^{\top}A\tilde{Q}_k$ 可知

$$A\tilde{Q}_k=\tilde{Q}_kA_{k+1}=\tilde{Q}_kQ_{k+1}R_{k+1}=\tilde{Q}_{k+1}R_{k+1}$$

所以有

$$\tilde{Q}_{k+1}=A\tilde{Q}_kR_{k+1}^{-1}$$

由于 $\tilde{Q}_{k+1}$ 和 $\tilde{Q}_k$ 都是正交矩阵,上式两边转置后求逆,可得

$$\tilde{Q}_{k+1}=\left(\tilde{Q}_{k+1}^{\top}\right)^{-1}=\left(\left(R_{k+1}^{-1}\right)^{\top}\tilde{Q}_k^{\top}A^{\top}\right)^{-1}=\left(A^{\top}\right)^{-1}\tilde{Q}_kR_{k+1}^{\top}$$

观察等式两边矩阵的最后一列,可得

$$\tilde{q}_n^{(k+1)}=c_1\left(A^{\top}\right)^{-1}\tilde{q}_n^{(k)}(c_1\text{为某个常数})$$

以此类推,可知

$$\tilde{q}_n^{(k+1)}=c\left(A^{\top}\right)^{-k}\tilde{q}_n^{(1)}(c\text{为某个常数})$$

假定 $|\lambda_1|\geq\cdots\geq|\lambda_{n-1}|>|\lambda_n|>0$ 则 λ_n^{-1} 是 $\left(A^{\top}\right)^{-1}$ 的模最大特征值. 由幂迭代可知 $\tilde{q}_n^{(k+1)}$ 收敛到 λ_n^{-1} 所对应的特征向量,即

$$\left(A^{\top}\right)^{-1}\tilde{q}_n^{(k+1)}\to\lambda_n^{-1}\tilde{q}_n^{(k+1)}\quad(k\to\infty)$$

所以

$$A^{\top}\tilde{q}_n^{(k)}\to\lambda_n\tilde{q}_n^{(k)}\quad(k\to\infty)$$

由 $A_{k+1}=\tilde{Q}_k^{\top}A\tilde{Q}_k$ 可知 $A_{k+1}^{\top}$ 的最后一列

$$A_{k+1}^{\top}(:,n)=\tilde{Q}_k^{\top}A^{\top}\tilde{q}_n^{(k)}\to\lambda_n\tilde{Q}_k^{\top}\tilde{q}_n^{(k)}=\lambda_ne_n$$

□□

A_{k+1} 的最后一行的最后一个元素收敛到 λ_n, 而其它元素都趋向于 0. 收敛速度取决于 $|\lambda_n/\lambda_{n-1}|$ 的大小

5.14.4 QR 迭代与正交迭代的关系

下面的定理给出了 QR 迭代算法与正交迭代算法 ($Z_0=I$) 之间的关系.

定理 假定正交迭代算法 3.1 和 QR 算法 4.1 中所涉及的 QR 分解都是唯一的. A_k 是由 QR 迭代算法 4.1 生成的矩阵, Z_k 是由正交迭代算法 3.1 (取 $Z_0=I$) 生成的矩阵, 则有

$$A_{k+1}=Z_k^{\top}AZ_k$$

5.14.5 QR 迭代的收敛性

定理 设 $A=V\Lambda V^{-1}\in\mathbb{R}^{n\times n}$, 其中 $\Lambda=\mathrm{diag}\left(\lambda_1,\lambda_2,\ldots,\lambda_n\right)$, 且 $|\lambda_1|>|\lambda_2|>\cdots>|\lambda_n|$. 若 V^{-1} 的所有顺序主子矩阵都非奇异 (即 V^{-1} 存在 LU 分解), 则 A_k 的对角线以下的元素收敛到 0

说明：
需要指出的是, 由于 D_k 的元素不一定收敛, 故 A_{k+1} 对角线以上 (不含对角线) 的元素不一定收敛, 但这不妨碍 A_{k+1} 的对角线元素收敛到 A 的特征值 (即 A_{k+1} 的对角线元素是收敛的)
例 QR 迭代算法演示 (见 $Eig_QR.m$). 设

$$A = X \begin{bmatrix} 9 & & & \\ & 5 & & \\ & & 3 & \\ & & & 1 \end{bmatrix} X^{-1}$$

其中 X 是由 MATLAB 随机生成的非奇异矩阵.
在迭代过程中, 对于 Ak 的下三角部分中元素, 如果其绝对值小于某个阈值 tol, 则直接将其设为 0, 即

$$a_{ij}^{(k)} = 0 \quad \text{if} \quad i > j \text{ and } \left|a_{ij}^{(k)}\right| < tol$$

这里我们取 $tol = 10^{-6} \max_{1\le i,j\le n} \left\{\left|a_{ij}^{(k)}\right|\right\}$, 迭代过程如下：

```
A =
   6.5629e+00   3.1505e+00   2.4882e+00  -4.5006e+00
   3.1564e+00   4.6079e+00   1.4346e+00  -2.9295e+00
  -3.5367e-02   9.7647e+00   7.7607e+00  -8.7044e+00
   3.7514e+00   2.4217e+00   5.2685e-01  -9.3141e-01
A_7 =
   1.0079e+01   2.0598e+00  -8.7382e-02  -1.4010e+01
  -2.6356e+00   3.9694e+00   5.3709e+00   2.8474e+00
  -1.0317e-02  -1.8888e-02   2.9523e+00  -1.4913e+00
            0  -1.4296e-05   1.3377e-03   9.9898e-01
A_8 =
   9.8306e+00   3.5979e+00  -1.4282e+00   1.4272e+01
  -1.1084e+00   4.1983e+00   5.1778e+00   7.8545e-01
  -2.9432e-03  -1.2199e-02   2.9714e+00   1.5095e+00
            0            0  -4.5563e-04   9.9966e-01

A_12 =
   9.0830e+00   4.6472e+00  -2.4491e+00   1.3798e+01
  -7.2867e-02   4.9207e+00   4.7783e+00   3.7229e+00
  -2.9534e-05  -1.5694e-03   2.9963e+00   1.5315e+00
            0            0            0   1.0000e+00
A_13 =
   9.0460e+00   4.6811e+00   2.4859e+00  -1.3767e+01
  -3.9787e-02   4.9562e+00  -4.7591e+00  -3.8330e+00
            0   9.3992e-04   2.9978e+00   1.5328e+00
            0            0            0   1.0000e+00
A_22 =
   9.0002e+00   4.7219e+00  -2.5302e+00   1.3729e+01
  -1.9625e-04   4.9998e+00   4.7355e+00   3.9669e+00
            0            0   3.0000e+00   1.5346e+00
            0            0            0   1.0000e+00

A_28 =
   9.0000e+00   4.7221e+00  -2.5304e+00   1.3729e+01
            0   5.0000e+00   4.7354e+00   3.9675e+00
            0            0   3.0000e+00   1.5346e+00
            0            0            0   1.0000e+00
```

5.14.6 带位移的 QR 迭代

为了加快 QR 迭代的收敛速度, 可以采用位移策略 和反迭代的思想

算法 4.2 带位移的 QR 迭代算法 (QR Iteration with shift)

1: Set $A_1 = A$ and $k = 1$

2: while not convergence do

3: Choose a shift σ_k

4: $[Q_k, R_k] = qr(A_k - \sigma_k I)$

5: compute $A_{k+1} = R_k Q_k + \sigma_k I$

6: $k = k + 1$

7: end while

正交相似性

$$\begin{aligned} A_{k+1} = R_k Q_k + \sigma_k I &= \left(Q_k^\top Q_k\right) R_k Q_k + \sigma_k I \\ &= Q_k^\top \left(A_k - \sigma_k I\right) Q_k + \sigma_k I \\ &= Q_k^\top A_k Q_k \end{aligned}$$

位移 σ_k 的选取

在前面的分析可知,$A_{k+1}(n,n)$ 收敛到 A 的模最小特征值.

若 σ_k 就是 A 的一个特征值,则 $A_k - \sigma_k I$ 的模最小特征值为 0, 故 QR 算法迭代一步就收敛. 此时

$$A_{k+1} = R_k Q_k + \sigma_k I = \begin{bmatrix} A_{k+1}^{(n-1)\times(n-1)} & * \\ 0 & \sigma_k \end{bmatrix}$$

A 的其它特征值可通过对 $A_{k+1}^{(n-1)\times(n-1)}$ 使用带位移 QR 迭代算法得到.

通常, 如果 σ_k 与 A 的某个特征值非常接近, 则收敛速度通常会很快. 由于 $A_k(n,n)$ 收敛到 A 的一个特征值, 所以在实际使用中, 一个比较直观的位移选择策略是 $\sigma_k = A_k(n,n)$. 事实上, 这样的位移选取方法通常会使得 QR 迭代算法有二次收敛速度.

例 带位移的 QR 迭代算法演示 ($Eig_Q R_s hift.m$).

所有数据和设置与例 4.1 相同, 在迭代过程中, 取 $\sigma_k = A_k(n,n)$. 如果 $A_k(n,n)$ 已经收敛, 则取 $\sigma_k = A_k(n-1, n-1)$

5.15 带位移的隐式 QR 迭代

直接实施 QR 方法的困难: 运算量

每一步迭代需要做一次 QR 分解和矩阵乘积, 运算量为 $O(n^3)$ 即使每计算一个特征值只需迭代一步, 则总运算量为 $O(n^4)$

我们的目标: 从 $O(n^4)$ 减小到 $O(n^3)$

```
A =
   6.5629e+00   3.1505e+00   2.4882e+00  -4.5006e+00
   3.1564e+00   4.6079e+00   1.4346e+00  -2.9295e+00
  -3.5367e-02   9.7647e+00   7.7607e+00  -8.7044e+00
   3.7514e+00   2.4217e+00   5.2685e-01  -9.3141e-01
A_5 =
   5.5186e+00  -3.0411e-01   4.4529e+00  -5.1700e+00
  -4.9782e+00   8.5660e+00   3.0148e+00   1.3331e+01
  -3.9116e-02  -1.7945e-03   2.9153e+00  -1.4587e+00
            0            0            0   1.0000e+00

A_7 =
   9.4467e+00   4.2553e+00  -2.0222e+00  -1.4068e+01
  -4.6678e-01   4.5533e+00   4.9737e+00  -2.5126e+00
            0            0   3.0000e+00  -1.5346e+00
            0            0            0   1.0000e+00
A_10 =
   9.0000e+00  -4.7221e+00   2.5304e+00   1.3729e+01
            0   5.0000e+00   4.7354e+00  -3.9676e+00
            0            0   3.0000e+00  -1.5346e+00
            0            0            0   1.0000e+00
```

实现方法:两个步骤

(1) 首先通过相似变化将 A 转化成一个上 H essenberg 矩阵

(2) 对这个 Hessenberg 矩阵实施隐式 QR 迭代

隐式 QR 迭代:
在 QR 迭代算法中, 并不进行显式的 QR 分解和矩阵乘积, 而是通过特殊手段来实现从 A_k 到 A_{k+1} 的迭代,并且将运算量控制在 $O\left(n^2\right)$ 量级,从而将总运算量降到 $O(n^3)$

5.15.1 上 Hessenberg 矩阵

上 Hessenberg 矩阵:$H=[h_{ij}] \in \mathbb{R}^{n\times n}$ 当 $i>j+1$ 时,有 $h_{ij}=0$
定理设 $A \in \mathbb{R}^{n\times n}$,则存在正交矩阵 $Q \in \mathbb{R}^{n\times n}$ 使得 $QAQ^{\top}$ 是上 Hessenberg 矩阵
下面我们以一个 5×5 的矩阵 A 为例, 给出具体的转化过程, 采用的工具为 Householder 变换.
第一步:令 $Q_1=\operatorname{diag}\left(I_{1\times 1}, H_1\right)$,其中 H_1 是对应于向量 $A(2:5,1)$ 的 Householder 矩阵. 于是可得

$$Q_1A=\begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ 0 & * & * & * & * \\ 0 & * & * & * & * \\ 0 & * & * & * & * \end{bmatrix}$$

由于用 Q_1^T 右乘 Q_1A, 不会改变 Q_1A 的第一列元素的值, 故

$$A_1 \triangleq Q_1AQ_1^{\top}=\begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ 0 & * & * & * & * \\ 0 & * & * & * & * \\ 0 & * & * & * & * \end{bmatrix}$$

第二步: 令 $Q_2=\operatorname{diag}\left(I_{2\times 2}, H_2\right)$, 其中 H_2 是对应于向量 $A_1(3:5,2)$ 的 Householder 矩阵, 则用 Q_2 左乘 A_1 时, 不会改变 A_1 的第一列元素的值. 用 $Q_2^{\top}$ 右乘 Q_2A_1 时,不会改变 Q_2A_1 前两列元素的值. 因此

$$Q_2A_1=\begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ 0 & * & * & * & * \\ 0 & 0 & * & * & * \\ 0 & 0 & * & * & * \end{bmatrix} \square A_2 \triangleq Q_2A_1Q_2^{\top}=\begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ 0 & * & * & * & * \\ 0 & 0 & * & * & * \\ 0 & 0 & * & * & * \end{bmatrix}$$

第三步：令 $Q_3 = \text{diag}\,(I_{3\times3}, H_3)$，其中 H_3 是对应于向量 $A_2(4:5,3)$ 的 Householder 矩阵，则有

$$Q_3A_2 = \begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ 0 & * & * & * & * \\ 0 & 0 & * & * & * \\ 0 & 0 & 0 & * & * \end{bmatrix} \square A_3 \triangleq Q_3A_2Q_3^\top = \begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ 0 & * & * & * & * \\ 0 & 0 & * & * & * \\ 0 & 0 & 0 & * & * \end{bmatrix}$$

这时我们就将 A 转化成一个上 Hessenberg 矩阵，即 $QAQ^\top = A_3$，其中 $Q = Q_3Q_2Q_1$ 是正交矩阵，A_3 是上 Hessenberg 矩阵.

第三步： 令 $Q_3 = \text{diag}\,(I_{3\times3}, H_3)$，其中 H_3 是对应于向量 $A_2(4:5,3)$ 的 Householder 矩阵，则有

$$Q_3A_2 = \begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ 0 & * & * & * & * \\ 0 & 0 & * & * & * \\ 0 & 0 & 0 & * & * \end{bmatrix} \text{和} A_3 \triangleq Q_3A_2Q_3^\top = \begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ 0 & * & * & * & * \\ 0 & 0 & * & * & * \\ 0 & 0 & 0 & * & * \end{bmatrix}$$

这时，我们就将 A 转化成一个上 Hessenberg 矩阵，即 $QAQ^T = A_3$，其中 $Q = Q_3Q_2Q_1$ 是正交矩阵，A_3 是上 Hessenberg 矩阵。

上 Hessenberg 化算法

算法 5.1 上 Hessenberg 化算法 (Upper Hessenberg Reduction)

1: set $Q = I$

2: for $k = 1$ to $n-2$ do

3: compute Hessenberg matrix H_k with respect to $A(k+1:n,k)$

4: $$\begin{aligned} A(k+1:n,k:n) &= H_k \cdot A(k+1:n,k:n) \\ &= A(k+1:n,k:n) - \beta_k v_k \left(v_k^\top A(k+1:n,k:n)\right) \end{aligned}$$

5: $$\begin{aligned} A(1:n,k+1:n) &= A(1:n,k+1:n) \cdot H_k^\top \\ &= A(1:n,k+1:n) - \beta_k A(1:n,k+1:n) v_k v_k^\top \end{aligned}$$

6: $$\begin{aligned} Q(k+1:n,k:n) &= H_k \cdot Q(k+1:n,k:n) \\ &= Q(k+1:n,k:n) - \beta_k v_k \left(v_k^\top Q(k+1:n,k:n)\right) \end{aligned}$$

7: end for

说明：

- 在实际计算时，我们不需要显式地形成 Householder 矩阵 H_k。

- 上述算法的运算量大约为 $\frac{14}{3}n^3 + \mathcal{O}(n^2)$。如果不需要计算特征向量，则正交矩阵 Q 也不用计算，此时运算量大约为 $\frac{10}{3}n^3 + \mathcal{O}(n^2)$。
- 上 Hessenberg 矩阵的一个很重要的性质就是在 QR 迭代中保持形状不变。

定理　设 $A \in \mathbb{R}^{n\times n}$ 是非奇异上 Hessenberg 矩阵，其 QR 分解为 $A = QR$，则 $\tilde{A} \triangleq RQ$ 也是上 Hessenberg 矩阵。

若 A 是奇异的，也可以通过选取适当的 Q，使得上述结论成立。

由此可知，如果 A 是上 Hessenberg 矩阵，则 QR 迭代中的每一个 A_k 都是上 Hessenberg 矩阵矩阵。这样在进行 QR 分解时，运算量可大大降低。

Hessenberg 矩阵另一重要性质：在 QR 迭代中保持下次对角线元素非零。

定理　设 $A \in \mathbb{R}^{n\times n}$ 是上 Hessenberg 矩阵且下次对角线元素均非零，即 $a_{i+1,i} \neq 0, i = 1, 2, \ldots, n-1$。设其 QR 分解为 $A = QR$，则 $\tilde{A} \triangleq RQ$ 的下次对角线元素也都非零。

若 A 村咋子某个下次对角线元素为零，则 A 一定可约。因此，我们只需考虑下次对角线均非零的情形。

推论　$\tilde{A} \triangleq RQ$ 则在带位移的 QR 迭代中，所有的 A_k 的下次对角线元素均非零。

5.15.2　隐式 QR 迭代

在 QR 迭代中，我们要先做 QR 分解 $A_k = Q_k R_k$，然后计算 $A_{k+1}k = Q_k R_k$. 但事实上，我们可以直接计算出 A_{k+1}。这就是隐式 QR 迭代。

不失一般性，我们假定 A 是不可约的上 Hessenberg 矩阵。

隐式 QR 迭代的理论基础就是下面的隐式 Q 定理。

定理 (ImplicitQTheorem)　设 $H = Q^\top AQ \in \mathbb{R}^{n\times n}$ 是一个不可约上 Hessenberg 矩阵，其中 $Q \in \mathbb{R}^{n\times n}$ 是正交矩阵，则 Q 的第 2 至第 n 列均由 Q 的第一列所唯一确定（可相差一个符号）。

由于 Q_k 的其他列都由 Q_k 的第一列唯一确定（至多相差一个符号），所以我们只要找到一个正交矩阵 $\tilde{Q}_k$ 使得其第一列与 $\tilde{Q}_k$ 的第一列相等，且 $\tilde{Q}_k^\top A_k \tilde{Q}_k$ 为上 Hessenberg 矩阵，则由隐式 Q 定理可知 $\tilde{Q}_k = WQ_k$，其中 $W = \mathrm{diag}(1, \pm 1, \ldots, \pm 1)$，于是

$$\tilde{Q}_k^\top A_k \tilde{Q}_k = W^\top Q_k^\top A_k Q_k W = W^\top A_{k+1} W$$

。又 $W^\top A_{k+1} W$ 与 A_{k+1} 相似，且对角线元素相等，而其他元素也至多相差一个符号，所以不会影响 A_{k+1} 的收敛性，即下三角元素收敛到 0，对角线元素收敛到 A 的特征值。

在 QR 迭代算法中，如果我们直接令 $A_{k+1} = \tilde{Q}_k^\top A_k \tilde{Q}_k$，则其收敛性与原 QR 迭代算法没有任何区别！这就是隐式 QR 迭代的基本思想。

由于 A 是上 Hessenberg 矩阵，因此在实际计算中，我们只需 Givens 变换。

下面我们举一个例子，具体说明如何利用隐式 Q 定理，由 A_1 得到 A_2。

设 $A \in \mathbb{R}^{5\times 5}$ 是一个不可约上 Hessenberg 矩阵，即

$$A_1 = A = \begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ 0 & * & * & * & * \\ 0 & 0 & * & * & * \\ 0 & 0 & 0 & * & * \end{bmatrix}$$

第一步： 构造一个 Givens 变换

$$G_1^\top \triangleq G(1,2,\theta_1) = \begin{bmatrix} c_1 & s_1 & & \\ -s_1 & c_1 & & \\ & & & I_3 \end{bmatrix} \qquad (c_1, s_1 \text{待定})$$

于是有

$$G_1^\top A = \begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ 0 & * & * & * & * \\ 0 & 0 & * & * & * \\ 0 & 0 & 0 & * & * \end{bmatrix} \text{和} A^{(1)} \triangleq G_1^\top A G_1 = \begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ + & * & * & * & * \\ 0 & 0 & * & * & * \\ 0 & 0 & 0 & * & * \end{bmatrix}$$

与 A_1 相比较，$A^{(1)}$ 在 $(3,1)$ 位置上多出一个非零元，我们把它记为“+”，并称之为bulge。在下面的计算过程中，我们的目标就是将其“赶”出矩阵，从而得到一个新的上 Hessenberg 矩阵，即 A_2。

第二步： 为了消去这个 bulge，我们可以构造 Givens 变换

$$G_2^\top \triangleq G(2,3,\theta_2) = \begin{bmatrix} 1 & & & \\ & c_2 & s_2 & \\ & -s_2 & c_2 & \\ & & & I_2 \end{bmatrix} \text{使得} G_2^\top A^{(1)} = \begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ 0 & * & * & * & * \\ 0 & 0 & * & * & * \\ 0 & 0 & 0 & * & * \end{bmatrix}$$

为了保持与原矩阵的相似性，需要再右乘 G_2，所以

$$A^{(2)} \triangleq G_2^\top A^{(1)} G_2 = \begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ 0 & * & * & * & * \\ 0 & + & * & * & * \\ 0 & 0 & 0 & * & * \end{bmatrix}$$

此时，bugle 从 $(3,1)$ 位置被“赶”到 $(4,2)$ 位置。

第三步： 与第二步类似，构造 Givens 变换

$$G_3^\top \triangleq G(3,4,\theta_3) = \begin{bmatrix} I_2 & & & \\ & c_3 & s_3 & \\ & -s_3 & c_3 & \\ & & & 1 \end{bmatrix} \text{使得} G_3^\top A^{(2)} = \begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ 0 & * & * & * & * \\ 0 & 0 & * & * & * \\ 0 & 0 & 0 & * & * \end{bmatrix}$$

这时

$$A^{(3)} \triangleq G_3^\top A^{(2)} G_3 = \begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ 0 & * & * & * & * \\ 0 & 0 & * & * & * \\ 0 & 0 & + & * & * \end{bmatrix}$$

于是，bugle 又从 $(4,2)$ 位置被“赶”到 $(5,3)$ 位置。

第四步： 再次构造 Givens 变换

$$G_4^\top \triangleq G(4,5,\theta_4) = \begin{bmatrix} 1 & & & & \\ & 1 & & & \\ & & 1 & & \\ & & & c_4 & s_4 \\ & & & -s_4 & c_4 \end{bmatrix} \text{使得} G_4^\top A^{(3)} = \begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ 0 & * & * & * & * \\ 0 & 0 & * & * & * \\ 0 & 0 & 0 & * & * \end{bmatrix}$$

这时

$$A^{(4)} \triangleq G_4^\top A^{(3)} G_4 = \begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ 0 & * & * & * & * \\ 0 & 0 & * & * & * \\ 0 & 0 & 0 & * & * \end{bmatrix}$$

现在，bulge 被“赶”出矩阵，$A(4)$ 就是我们所要的矩阵！

算法分析，以及 c_1, s_1 的取值

常规 QR 迭代：$A_1 = Q_1 R_1, A_2 = R_1 Q_1 \Longrightarrow A_2 = Q_1^\top A_1 Q_1$

根据前面的计算过程，有

$$A^{(4)} = G_4^\top G_3^\top G_2^\top G_1^\top A_1 G_1 G_2 G_3 G_4 = \tilde{Q}_1^\top A_1 \tilde{Q}_1$$

，其中 $\tilde{Q}_1 = G_1 G_2 G_3 G_4 \Longrightarrow A^{(4)} = \tilde{Q}_1^\top A_1 \tilde{Q}_1$

通过直接计算可知，$\tilde{Q}_1$ 的第一列为

$$[c_1, s_1, 0, 0, 0]^\top$$

如果将其取为 A_1 的第一列 $[a_{11}, a_{21}, 0, \ldots, 0]^\top$ 单位化后的向量，则 $\tilde{Q}_1$ 的第一列与 Q_1 的第一列相同！$\Longrightarrow A^{(4)} = W^\top A_2 W$

针对带位移的 QR 方法，我们取 $A_1 - \sigma_1 I$ 的第一列

$$[a_{11} - \sigma_1, a_{21}, 0, \ldots, 0]^\top$$

单位化后的向量作为 G_1 的第一列即可。

运算量：

如果 $A \in \mathbb{R}^{n\times n}$ 是上 Hessenberg 矩阵，则使用上面的算法，带位移 QR 迭代中每一步的运算量为 $6n^2 + O(n)$。

5.15.3 位移的选取

通常，位移越离某个特征值越近，则收敛速度就越快。

由习题 4.10 可知，如果位移 σ 与某个特征值非常接近，则 $A_k(n,n) - \sigma$ 就非常接近于 0。

这说明$A_k(n,n)$ 通常会首先收敛到 A 的一个特征值。所以$\sigma = A_k(n,n)$ 是一个不错的选择。但是，如果这个特征值是复数，这种唯一选取方法就可能失效。

双位移策略

设 $\sigma \in \mathbb{C}$ 是 A 的某个复特征值 λ 的一个很好的近似，则其共轭 $\overline{\sigma}$ 也应该是 $\overline{\lambda}$ 的一个很好的近似。因此我们可以考虑双位移策略，即先以 λ 为位移迭代一次，然后再以 $\overline{\sigma}$ 为位移迭代一次，如此不断交替进行迭代。

这样就有

$$\begin{aligned} A_1 - \sigma I &= Q_1 R_1 \\ A_2 &= R_1 Q_1 + \sigma I \\ A_2 - \overline{\sigma} I &= Q_2 R_2 \\ A_3 &= R_2 Q_2 + \overline{\sigma} I \end{aligned}$$

容易验证

$$A_3 = Q_2^\top A_2 Q_2 = Q_2^* Q_1^* A_1 Q_1 Q_2 = Q^* A_1 Q$$

其中 $Q = Q_1 Q_2$

我们注意到 σ 可能是复的，所以 Q_1 和 Q_2 都可能是复矩阵。但我们却可以选取适当的 Q_1 和 Q_2，使得 $Q = Q_1 Q_2$ 是实矩阵。

双位移策略的实现

由前面的结论可知，存在 Q_1 和 Q_2，使得 $Q = Q_1 Q_2$ 是实矩阵，从而

$$A_3 = Q^\top A_1 Q$$

也是实矩阵。因此我们希望不计算 A_2，而是直接从 A_1 得到 A_3

实现方式：

根据隐式 Q 定理：只要找到一个实正交矩阵 Q，使得其第一列与

$$A_1^2 - 2\,\mathrm{Re}(\sigma) A_1 + |\sigma|^2 I$$

的第一列平行，并且 $A_3 = Q^\top A_1 Q$ 是上 Hessenberg 矩阵即可。

易知，$A_1^2 - 2\,\mathrm{Re}(\sigma)A_1 + |\sigma|^2 I$ 的第一列为

$$\begin{bmatrix} a_{11}^2 + a_{12}a_{21} - 2\,\mathrm{Re}(\sigma)a_{11} + |\sigma|^2 \\ a_{21}\left(a_{11} + a_{22} - 2\,\mathrm{Re}(\sigma)\right) \\ a_{21}a_{32} \\ 0 \\ \vdots \end{bmatrix} \tag{91}$$

所以 Q 的第一列是上述向量的单位化。

其他过程可以通过隐式 QR 迭代来实现。但此时的“bulge” 是一个 2×2 的小矩阵。因此，在双位移隐式 R 迭代过程中，需要使用 Householder 变换。

需要指出的是，双位移 QR 迭代算法中的运算都是实数运算。

下面通过一个例子来说明如何在实数运算下实现双位移隐式 QR 迭代。

设 $A \in \mathbb{R}^{6\times 6}$ 是一个不可约上 Hessenberg 矩阵，即

$$A_1 = A = \begin{bmatrix} * & * & * & * & * & * \\ * & * & * & * & * & * \\ 0 & * & * & * & * & * \\ 0 & 0 & * & * & * & * \\ 0 & 0 & 0 & * & * & * \\ 0 & 0 & 0 & 0 & * & * \end{bmatrix}$$

第一步： 构造一个正交矩阵 $H_1 = \begin{bmatrix} \tilde{H}_1^\top & 0 \\ 0 & I_3 \end{bmatrix}$，其中 $\tilde{H}_1 \in \mathbb{R}^{3\times 3}$，使得第一列与 $A_1^2 - 2\,\mathrm{Re}(\sigma)A_1 + |\sigma|^2 I$ 的第一列平行。于是有

$$H_1^\top A = \begin{bmatrix} * & * & * & * & * & * \\ * & * & * & * & * & * \\ + & * & * & * & * & * \\ 0 & 0 & * & * & * & * \\ 0 & 0 & 0 & * & * & * \\ 0 & 0 & 0 & 0 & * & * \end{bmatrix} \text{和} A^{(1)} \triangleq H_1^\top A H_1 = \begin{bmatrix} * & * & * & * & * & * \\ * & * & * & * & * & * \\ + & * & * & * & * & * \\ + & + & * & * & * & * \\ 0 & 0 & 0 & * & * & * \\ 0 & 0 & 0 & 0 & * & * \end{bmatrix}$$

与 A_1 相比较，$A^{(1)}$ 在 $(3,1),(4,1)$ 和 $(4,2)$ 位置上出现 bulge。在下面的计算过程中，我们的目标就是把它们”赶“出矩阵，从而得到一个新的上 Hessenberg 矩阵。

第二步： 令 $H_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \tilde{H}_2^\top & 0 \\ 0 & 0 & I_2 \end{bmatrix}$，其中 $\tilde{H}_2 \in \mathbb{R}^{3\times 3}$ 是对应于 $A(2:4,1)$ 的

Householder 变换, 使得

$$H_2^\top A^{(1)} = \begin{bmatrix} * & * & * & * & * & * \\ * & * & * & * & * & * \\ 0 & * & * & * & * & * \\ 0 & + & * & * & * & * \\ 0 & 0 & 0 & * & * & * \\ 0 & 0 & 0 & 0 & * & * \end{bmatrix} \text{和} A^{(2)} \triangleq H_2^\top A^{(1)} H_2 = \begin{bmatrix} * & * & * & * & * & * \\ * & * & * & * & * & * \\ 0 & * & * & * & * & * \\ 0 & + & * & * & * & * \\ 0 & + & + & * & * & * \\ 0 & 0 & 0 & 0 & * & * \end{bmatrix}$$

这时，我们将 bugle 向右下角方向”赶“了一个位置。

第三步　与第二步类似，令 $H_3 = \begin{bmatrix} I_2 & 0 & 0 \\ 0 & \tilde{H}_3^\top & 0 \\ 0 & 0 & 1 \end{bmatrix}$，其中 $\tilde{H}_3 \in \mathbb{R}^{3\times 3}$ 是对应于 $A(3:5,2)$ 的 Householder 变换, 使得

$$H_3^\top A^{(2)} = \begin{bmatrix} * & * & * & * & * & * \\ * & * & * & * & * & * \\ 0 & * & * & * & * & * \\ 0 & 0 & * & * & * & * \\ 0 & 0 & + & * & * & * \\ 0 & 0 & 0 & 0 & * & * \end{bmatrix} \text{和} A^{(3)} \triangleq H_3^\top A^{(2)} H_3 = \begin{bmatrix} * & * & * & * & * & * \\ * & * & * & * & * & * \\ 0 & * & * & * & * & * \\ 0 & 0 & * & * & * & * \\ 0 & 0 & + & * & * & * \\ 0 & 0 & + & + & * & * \end{bmatrix}$$

这时，bugle 又被向右下角方向”赶“了一个位置。

第四步　令 $H_4 = \begin{bmatrix} I_3 & 0 \\ 0 & \tilde{H}_4^\top \end{bmatrix}$，其中 $\tilde{H}_4 \in \mathbb{R}^{3\times 3}$ 是对应于 $A(4:6,3)$ 的 Householder 变换, 使得

$$H_4^\top A^{(3)} = \begin{bmatrix} * & * & * & * & * & * \\ * & * & * & * & * & * \\ 0 & * & * & * & * & * \\ 0 & 0 & * & * & * & * \\ 0 & 0 & 0 & * & * & * \\ 0 & 0 & 0 & + & * & * \end{bmatrix} \text{和} A^{(4)} \triangleq H_4^\top A^{(3)} H_4 = \begin{bmatrix} * & * & * & * & * & * \\ * & * & * & * & * & * \\ 0 & * & * & * & * & * \\ 0 & 0 & * & * & * & * \\ 0 & 0 & 0 & * & * & * \\ 0 & 0 & 0 & + & * & * \end{bmatrix}$$

第五步　只需构造一个 Givens 变换 $G_5 = \begin{bmatrix} I_4 & 0 \\ 0 & G(4,5,\theta)^\top \end{bmatrix}$，使得

$$G5^\top A^{(4)} = \begin{bmatrix} * & * & * & * & * & * \\ * & * & * & * & * & * \\ 0 & * & * & * & * & * \\ 0 & 0 & * & * & * & * \\ 0 & 0 & 0 & * & * & * \\ 0 & 0 & 0 & 0 & * & * \end{bmatrix} \text{和} A^{(5)} \triangleq G_5^\top A^{(4)} G_5 = \begin{bmatrix} * & * & * & * & * & * \\ * & * & * & * & * & * \\ 0 & * & * & * & * & * \\ 0 & 0 & * & * & * & * \\ 0 & 0 & 0 & * & * & * \\ 0 & 0 & 0 & 0 & * & * \end{bmatrix}$$

现在，bulge 已经被全部消除，且

$$A^{(5)} = Q^\top A Q$$

，其中 $Q = H_1 H_2 H_3 H_4 G_5$。通过直接计算可知，Q 的第一列即为 H_1 的第一列。根据隐式 Q 定理，可以直接令 $A_3 \triangleq A^{(5)} = Q^\top A Q$。

位移的具体选取

在单位移 QR 迭代算法中，若 A 的特征值都是实的，则取 $\sigma_k = A_k(n,n)$. 推广到复共轭特征值上，我们可以取 A_k 的右下角矩阵

$$\begin{bmatrix} A_k(n-1,n-1) & A_k(n-1,n) \\ A_k(n,n-1) & A_k(n,n) \end{bmatrix}$$

的复共轭特征值作为双位移。这样选取的位移就是Francis 位移。

如果上述矩阵的两个特征值都是实的，则选取其中模较小的特征值做单位移。

采用 Francis 位移的 QR 迭代会使得 A_k 的右下角收敛到一个上三角矩阵 (两个实特征值) 或一个 2 阶的矩阵 (一对复共轭特征值)，而且通常会有二次收敛性。在实际计算中，一个特征值一般平均只需迭代两步。

收敛性判断：

判断收敛性主要是看 $A_k(n-1,n-2)$（或 $A_k(n,n-1)$）是否趋向于 0。

需要指出的是，QR 迭代并不是对所有的矩阵都收敛。例如：

$$A = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

对于上面的矩阵，采用 Francis 位移的 QR 迭代算法无效。另外，也可以考虑多重位移策略，参见[Watkins 2007]。

5.15.4　收缩 Deflation

收缩 (deflation) 技术是实用 QR 迭代中的一个非常重要概念。

隐式 QR 迭代过程中，当矩阵 A_{k+1} 的某个下次对角线元素 a_{i+1}，i 很小时，我们可以将其设为 0。

由于 A_{k+1} 是上 Hessenberg 矩阵，这时 A_{k+1} 就可以写成分块上三角形式，其中两个对角块都是上 Hessenberg 矩阵。

因此我们可以将隐式 QR 迭代作用在这两个规模相对较小的矩阵上，从而可以大大节约运算量。

5.16 特征向量的计算

设 A 的特征值都是实的，$R = Q^T AQ$ 是其 Schur 标准型。若 $Ax = \lambda x$，则 $Ry = \lambda y$，其中 $y = Q^T x$ 或 $x = Qy$。故只需计算 R 的特征向量 y 即可。

因为 R 的对角线元素即为 A 的特征值，不妨设 $\lambda = R(i,i)$。

假定 λ 是单重特征值，则方程 $(R - \lambda I)y = 0$ 即为

$$\begin{bmatrix} R_{11} - \lambda I & R_{12} & R_{13} \\ 0 & 0 & R_{23} \\ 0 & 0 & R_{33} - \lambda I \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = 0$$

即

$$(R_{11} - \lambda I)\, y_1 + R_{12} y_2 + R_{13} y_3 = 0 \tag{92}$$

$$R_{23} y_3 = 0 \tag{93}$$

$$(R_{33} - \lambda I)\, y_3 = 0 \tag{94}$$

其中 $R_{11} \in \mathbb{R}^{(i-1)\times(i-1)}, R_{33} \in \mathbb{R}^{(n-i)\times(n-i)}$。由于 λ 是单重特征值，故 $R_{33} - \lambda I$ 非奇异，因此 $y_3 = 0$。令 $y_2 = 1$，则可得

$$y_1 = (R_{11} - \lambda I)^{-1} R_{12}$$

因此计算特征向量 y 只需求解一个上三角线性方程组。

若 λ 是多重特征值，则据算方法类似。但如果 A 有负特征值，则需要利用实 Schur 标准型，计算较复杂。

5.17 广义特征值问题

设 $A, B \in \mathbb{R}^{n\times n}$，若存在 $\lambda \in \mathbb{C}$ 和非零向量 $x \in \mathbb{C}^n$ 使得

$$Ax = \lambda Bx$$

则称 λ 为矩阵对 (A, B) 的特征值，x 为对应的特征向量。

计算矩阵对 (A, B) 的特征值和特征向量就是广义特征值问题

当 B 非奇异时，广义特征值问题就等价于标准特征值问题

$$B^{-1}Ax = \lambda x 或 AB^{-1}y = \lambda y$$

其中 $y = Bx$。

容易看出，λ 是 (A, B) 的一个特征值当且仅当

$$\det(A - \lambda B) = 0 \tag{95}$$

若95对所有 $\lambda \in \mathbb{C}$ 都成立，则称矩阵对 (A, B) 是奇异矩阵对，否则称为正则矩阵对。

当 B 非奇异时，特征方程95是一个 n 次多项式，因此恰好有 n 个特好找呢个字。当 B 奇异时，特征方程95的次数低于 n，因此方程的解的个数小于 n。但是，注意带 $\lambda \neq 0$ 是 (A, B) 的 t 特征值当且仅当 $\mu = \frac{1}{\lambda}$ 是 (B, A) 的特征值。因此，当 B 奇异时，$\mu = 0$ 是 (B, A) 的特征值，于是我们自然的把 $\lambda = \frac{1}{u} = \infty$ 当作是 (A, B) 的特征值。所以广义特征值不是分布在 $\mathbb{C}$ 上，而是分布在 $\mathbb{C} \cup \{\infty\}$ 上。

容易验证，若 U, V 非奇异，则矩阵对 (U^*AV, U^*BV) 的特征值与 (A, B) 是一样的。因此我们称这种变换为矩阵对的等价变换。如果 U, V 是酉矩阵，则称为酉等价变换。

5.17.1 广义 Schur 分解

广义 Schur 分解是矩阵对在酉等价变换下的最简形式。

定理 (广义 Schur 分解) 设 $A, B \in \mathbb{C}^{n\times n}$，则存在酉矩阵 $Q, Z \in \mathbb{C}^{n\times n}$，使得

$$Q^*AZ = R_A, \quad Q^*BZ = R_B \tag{96}$$

其中 $R_A, R_B \in \mathbb{C}^{n\times n}$ 都是上三角矩阵。此时矩阵对 (A, B) 的特征值为 R_A 和 R_B 的对角线元素的比值，即

$$\lambda_i = \frac{R_A(i,i)}{R_B(i,i)}, \quad i = 1, 2, \ldots, n$$

当 $R_B(i, i) = 0$ 时，对应的特征值 $\lambda_i = \infty$。

证明参见[Xu-Qian 2011]。

与实 Schur 分解类似，当 A, B 都是实矩阵时，我们有相应的广义实 Schur 分解。

定理 (广义 Schur 分解) 设 $A, B \in \mathbb{R}^{n\times n}$，则存在酉矩阵 $Q, Z \in \mathbb{R}^{n\times n}$，使得

$$Q^TAZ = T_A, \quad Q^TBZ = T_B \tag{97}$$

其中 $T_A, T_B \in \mathbb{R}^{n\times n}$ 都是拟上三角矩阵。

证明参见[Xu-Qian 2011]。

5.17.2 QZ 迭代

QZ 迭代是用于计算 (A, B) 的广义 Schur 分解的算法，是 QR 算法的自然推广，实质上可以看作是将 QR 算法作用到矩阵 AB^{-1} 上。

详细算法可参见[Kressner 2005, Xu-Qian 2011]。

5.18 应用：多项式求根

考虑 n 次多项式

$$q_n(x) = x^n + c_{n-1}x^{n-1} + \cdots + c_1 x + c_0, \quad c_i \in \mathbb{R}$$

- 由代数学基本定理可知，$p_n(x)$ 在复数域中有且仅有 n 的零点
- $n \geq 5$ 时，不存在求根公式
- 非线性迭代方法求解
- MATLAB 中的roots命令: 通过特征值计算方法求出所有零点

友矩阵

$$A = \begin{bmatrix} 0 & & -c_0 & \\ 1 & 0 & & -c_1 \\ \ddots & \ddots & \vdots & \\ & & 1 & -c_{n-1} \end{bmatrix}$$

多项式$q_n(x)$的零点 $\Longleftrightarrow$ A的特征值

- 无需上 Hessenberg 化
- A 非常稀疏，但经过一步 QR 迭代后，上三角部分的零元素会消失，总运算量仍是 $O(n^3)$
- 快速 QR 方法：利用 A 的特殊结构，运算量 $O(n^2)$

将 A 写成一个酉矩阵与秩一矩阵之差, 具体实现参见相关文献。

6 对称矩阵的特征值问题

关于对称特征值问题的常用算法有（直接法）：

- Jacobi 迭代: 最古老, 收敛速度较慢, 但精度较高, 且很适合并行计算。
- Rayleigh 商迭代: 一般具有三次收敛性, 但需要解方程组。
- 对称 QR 迭代: 对称矩阵的 QR 方法. 对于对称三对角矩阵, 若只计算特征值, 则速度最快 (运算量为 $\mathcal{O}(n^2)$)。如果还需要计算特征向量，则运算量约为 $6n^3$。
- 分而治之法：同时计算特征值和特征向量的一种快速算法。基本思想时将大矩阵分解形成小矩阵，然后利用递归思想求特征值。在最坏的情形下，运算量为 $\mathcal{O}(n^3)$。在实际应用中，平均为 $\mathcal{O}(n^{2.3})$。如果使用快速多极子算法 (FMM) 后，理论上的运算量可降低到 $\mathcal{O}(nlog^p n)$，其中 p 时一个较小的整数，这使得分而治之算法陈给目前计算对称三对角矩阵的 特征值和特征向量的首选方法。

- 对分法和反迭代：对分法主要用于求解对称三交矩阵在某个区间中的特征值，运算量约为 $\mathcal{O}(kn)$。其中 k 为所需计算的特征值的个数。反迭代用于计算特征向量，在最佳情况下，即特征值“适当分离”时，运算量约为 $\mathcal{O}(kn)$，但在最差情况下，即特征值成串的紧靠在一起时，运算量约为 $\mathcal{O}(k^2n)$，而且不能保证特征向量的精度（虽然实际上它几乎是精确的）。

处理 Jacobi 迭代和 Rayleigh 商迭代外，其余算法都需要先将对称矩阵三对角化，这个过程大约需花费 $\frac{4}{3}n^3$ 的工作量，如果需要计算特征向量的话，则运算量约为 $\frac{8}{3}n^3$。

6.1 Jacobi 迭代

基本思想　通过一系列的 Jacobi 旋转将 A 正交相似于一个对角矩阵：

$$A^{(0)} = A, A(K+1) = J_k^T A^{(k)} J_k, k = 0, 1, ...,$$

且 $A(0)$ 收敛到一个对角矩阵，其中 J^k 为 Jacobi 旋转，即 Givens 变换：

$$J_k = G(i_k, j_k, \theta_k) = \left[\begin{array}{c|ccc|c} I & & & & \\ \hline & cos\theta_k & \cdots & -sin\theta_k & \\ & \vdots & \ddots & \vdots & \\ & sin\theta_k & \cdots & cos\theta_k & \\ \hline & & & & I \end{array}\right] \tag{98}$$

$$J_k = G(i_k, j_k, \theta_k) = \left[\begin{array}{c|ccc|c} I & & & & \\ \hline & \cos\theta_k & \cdots & -\sin\theta_k & \\ & \vdots & \ddots & \vdots & \\ & \sin\theta_k & \cdots & \cos\theta_k & \\ \hline & & & & I \end{array}\right] \begin{array}{l} \\ i_k \\ \\ j_k \\ \\ \end{array}$$

（i_k 标在第 2 行，j_k 标在第 4 行）

引理　设 $A \in \mathbb{R}^{2\times 2}$ 是对角矩阵。则存在 Givens 变换 $G \in \mathbb{R}^{2\times 2}$ 使得 $G^T AG$ 为对角阵。

为了使 $A^{(K)}$ 收敛到一个对角矩阵，其非对角素必须趋向于 0。

记 $off(A)$ 为所有非对角元素的平方和，即

$$off(A) = \sum_{i\neq j} a_{ij}^2 = \|A^2\|_F^2 - \sum_{i=1}^{n} a_{ii}^2$$

我们的目标就是使得 $off(A)$ 尽快趋向于 0。

引理　设 $A = [a_{ij}]_{n\times n} \in \mathbb{R}^{n\times n}$ 是对称矩阵。$\widehat{A} = [a_{ij}]_{n\times n} = J^T AJ$,$J = G(i.j,\theta)$, 其中 θ 的选取使得 $\widehat{a}_{ij} = \widehat{a}_{ji} = 0$, 则

$$off(\widehat{A}) = off(A) - 2a_{ij}^2$$

算法　1.1　Jacobi 迭代算法

Given a symmetric matrix $A \in \mathbb{R}^{n\times n}$
if eigenvectors are desired then
    set $J = I$ and $shift = 1$
end if
while not converge do
    choose an index pair (i, j)such that $a_{ij} \neq 0$
    $\tau = (a_{ii} - a_{jj})/(2a - jj)$
    $t = sign(\tau/(|\tau| + \sqrt{1 + \tau^2})$
    $c = 1/\sqrt{1 + t^2}, s = c{\cdot}t$
    $A = G(i, j, \theta)^T AG(i, j, \theta)$
    if $shift = 1$ then
        $J = J{\cdot}G(i, j, \theta)$
    end if
end while

a_{ij} 的选取问题

一种直观的选取方法就是使得 a_{ij} 为所有非对角元素中绝对值最大的一个，这就是经典 Jacobi 算法。

算法　1.2　经典 Jacobi 迭代算法

Given a symmetric matrix $A \in \mathbb{R}^{n\times n}$
if eigenvectors are desired then
    set $J = I$ and $shift = 1$
end if
while $off(A) > tol$ do
    choose (i, j)such that $a_{ij} = max_{k\neq l}|a_{kl}|$

7: $\tau = (a_{ii} - a_{jj})/(2a - jj)$

8: $t = sign(\tau/(|\tau| + \sqrt{1+\tau^2})$

9: $c = 1/\sqrt{1+t^2}, s = c{\cdot}t$

10: $A = G(i,j,\theta)^T AG(i,j,\theta)$

11: if $shift = 1$ then

12: $J = J{\cdot}G(i,j,\theta)$

13: end if

14: end while

可以证明，经典 Jacobi 算法至少是线性收敛的。

定理　经典 Jacobi 算法1.2是 N 步局部二次收敛的，即对足够大的 k，有

$$off(A^{(k+N)}) = \mathcal{O}(off^2(A^{(k)}))$$

循环 Jacobi　经典 Jacobi 算法的每一步都要寻找绝对值最大的非对角元，费时不实用。改进：逐行扫描。

算法　1.3　经典 Jacobi 迭代算法 (逐行扫描)

1: Given a symmetric matrix $A \in \mathbb{R}^{n\times n}$

2: if eigenvectors are desired then

3: set $J = I$ and $shift = 1$

4: end if

5: while $off(A) > tol$ do

6: for $i = 1$ to $n-1$ do

7: for$j = i+1$ to n do

8: if $a_{ij} \neq 0$ then

9: $\tau = (a_{ii} - a_{jj})/(2a - jj)$

10: $t = sign(\tau/(|\tau| + \sqrt{1+\tau^2})$

11: $c = 1/\sqrt{1+t^2}, s = c{\cdot}t$

12: $A = G(i,j,\theta)^T AG(i,j,\theta)$

13: if $shift = 1$ then

14: $J = J{\cdot}G(i,j,\theta)$

15: end if

16: end if

17: end for

18: end for

19: end while

循环 Jacobi 也具有局部二次收敛性。

6.2 Raylaogh 商迭代

反迭代方法中，以 Rayleigh 商作为位移。

关于 Rayleigh 商迭代的收敛性，我们有下面的结论。

定理 设 $A \in \mathbb{R}^{n\times n}$ 对称，且特征值都是单重的。则当误差足够小时，Rayleigh 商迭代中每步迭代所得的正确数字的位数曾至三倍，即 Rayleigh 商迭代是局部三次收敛。

6.3 对称 QR 迭代

将带位移的隐式 QR 方法运用到对称矩阵，就得到对称 QR 迭代方法。基础步骤：

1. 对称三对角化：利用 Householder 变换，将 A 化为对称三对角矩阵，即计算正交矩阵 Q 使得 $T = QAQ^T$ 为对称三对角矩阵；
2. 使用带（单）位移的隐式 QR 迭代算法计算 T 的特征值与特征向量；
3. 计算 A 的特征向量。

对称三对角化

任何一个对称矩阵 $A \in \mathbb{R}^{n\times n}$ 都可以通过正交变换转化成一个对称三对角矩阵 T。这个过程可以通过 Householder 变换来实现，也可以通过 Givens 变换来实现。
对称 QR 迭代算法的运算量

- 三对角化 $4n^3/3 + \mathcal{O}(n^2)$, 若需计算特征向量，则为 $8n^3/3 + \mathcal{O}(n^2)$；
- 对 T 做带位移的隐式 QR 迭代，每次迭代的运算量为 $6n$；
- 计算特征值，假定每个平均迭代 2 步，则总运算量为 $12n^2$；
- 若要计算 T 的所有特征值和特征向量，则运算量为 $6n^3 + \mathcal{O}(n^2)$；
- 若只要计算 A 的所有特征值，运算量为 $4n^3/3 + \mathcal{O}(n^2)$；
- 若计算 A 的所有特征值和特征向量，则运算量为 $26n^3/3 + \mathcal{O}(n^2)$；

位移的选取——Wilkinson 位移

位移的好坏直接影响到算法的收敛速度。我们可以通过下面的方式来选取位移。设

$$A^{(k)} = \begin{bmatrix} a_1^{(k)} & b_1^{(k)} & & \\ b_1^{(k)} & \ddots & \ddots & \\ & \ddots & \ddots & b_{n-1}^{k} \\ & & b_{n-1}^{(k)} & a_n^{(k)} \end{bmatrix}$$

一种简单的位移选取策略就是令 $\sigma_k = a_n(K)$。事实上，$a_n^{(k)}$ 就是收敛到特征向量的迭代向量的 Rayleigh 商。这种位移选取方法几乎对所有的矩阵都有三次渐进收敛速度。但也存在不收敛的例子，故我们需要对其做改进。

Wilkinson 位移：取 $\begin{bmatrix} a_{n-1}^{(k)} & b_{n-1}^{(k)} \\ b_{n-1}^{(k)} & a_n^{(k)} \end{bmatrix}$ 的最接近 $a_n^{(k)}$ 的特征值作为位移。通过计算可得 Wilkinson 位移为

$$\sigma = a_n^{(k)} + \delta - sign(\delta)\sqrt{\delta^2 + (b_{n-1}^{(k)})^2}$$

，其中 $\delta = \ frac12(a_{n-1}^{(k)} - a_n^{(K)})$ 出于稳定性方面的考虑，我们通常用下面的计算公式

$$\sigma = a_n^{(k)} - \frac{(b_{n-1}(k))^2}{\delta + sign(\delta)\sqrt{\delta^2 + (b_{n-1}^{(k)})^2}} \tag{99}$$

定理　采用 Wilkinson 位移的 QR 迭代时整体收敛的，且至少是线性收敛。事实上，几乎所有的矩阵都是渐进三次收敛的。

例　带 Wilkinson 位移的隐式 QR 迭代算法收敛性演示。

MATLAB 代码：$Eig_TriQR.m$

6.4　分而治之法

分而治之法由 Cuppen 于 1981 年首次提出，但直到 1995 年才出现稳定的实现方式，是目前计算所有特征值和特征向量的最快算法。

考虑不可约对称三对角矩阵

$$T=\left[\begin{array}{cccc|cccc}
a_1 & b_1 & & & & & & \\
b_1 & \ddots & \ddots & & & & & \\
 & \ddots & a_{m-1} & b_{m-1} & & & & \\
 & & b_{m-1} & a_m & b_m & & & \\
\hline
 & & & b_m & a_{m+1} & b_{m+1} & & \\
 & & & & b_{m+1} & \ddots & \ddots & \\
 & & & & & \ddots & \ddots & b_{n-1} \\
 & & & & & & b_{n-1} & a_n
\end{array}\right]$$

$$=\left[\begin{array}{cccc|cccc}
a_1 & b_1 & & & & & & \\
b_1 & \ddots & \ddots & & & & & \\
 & \ddots & a_{m-1} & b_{m-1} & & & & \\
 & & b_{m-1} & a_m-b_m & & & & \\
\hline
 & & & & a_{m+1}-b_m & b_{m+1} & & \\
 & & & & b_{m+1} & \ddots & \ddots & \\
 & & & & & \ddots & \ddots & b_{n-1} \\
 & & & & & & b_{n-1} & a_n
\end{array}\right]+\left[\begin{array}{cccc|cccc}
 & & & & & & & \\
 & & & & & & & \\
 & & & & & & & \\
 & & & & & b_m & b_m & \\
\hline
 & & & & & b_m & b_m & \\
 & & & & & & & \\
 & & & & & & & \\
 & & & & & & &
\end{array}\right]$$

$$=\left[\begin{array}{c|c}
T_1 & 0 \\
\hline
0 & T_1
\end{array}\right]+b_m vv^T$$

其中 $v=[0,...,0,1,1,0,...,0]^T$。假定 T_1 和 T_2 的特征值已经计算出来，即 $T_1=Q_1\mathbf{\Lambda}_1Q_1^T$，$T_2=Q_2\mathbf{\Lambda}_2Q_2^T$，下面考虑 T 的特征值分解。

$$T=\begin{bmatrix} T_1 & 0 \\ 0 & T_2 \end{bmatrix}+b_m vv^{\mathrm{T}}=\begin{bmatrix} Q_1\Lambda_1Q_1^{\mathrm{T}} & 0 \\ 0 & Q_2\Lambda_2Q_2^{\mathrm{T}} \end{bmatrix}+b_m vv^{\mathrm{T}}$$

$$=\begin{bmatrix} Q_1 & 0 \\ 0 & Q_2 \end{bmatrix}\left(\begin{bmatrix} \Lambda_1 & 0 \\ 0 & \Lambda_2 \end{bmatrix}+b_m uu^{\mathrm{T}}\right)\begin{bmatrix} Q_1 & 0 \\ 0 & Q_2 \end{bmatrix}^{\top}$$

其中

$$u=\begin{bmatrix} Q_1 & 0 \\ 0 & Q_2 \end{bmatrix}^{\top},v=\begin{bmatrix} Q_1^T\text{的最后一列} \\ Q_2^T\text{的第一列} \end{bmatrix}$$

令 $a=b_m$，$D=diag(\Lambda_1,\Lambda_2)=diag(d_1,d_2,...,d_n)$，并假定 $d_1\geq d_2\geq ...\geq d_n$，则 T 的特征值于 $D+\alpha uu^T$ 的特征值相同。考虑 $D+\alpha uu^T$ 的特征值

设 λ 是 $D+\alpha uu^T$ 的一个特征值，若 $D-\lambda I$ 非奇异，则

$$det(D+\alpha uu^T-\lambda I)=det(D-\lambda I)\cdot det(I+\alpha(D-\lambda I)^{-1}uu^T)$$

故 $det(D+\alpha uu^T-\lambda I)=0$。

引理　设 $x,y\in\mathbb{R}^n$，则 $det(I+xy^T)=1+y^Tx$。

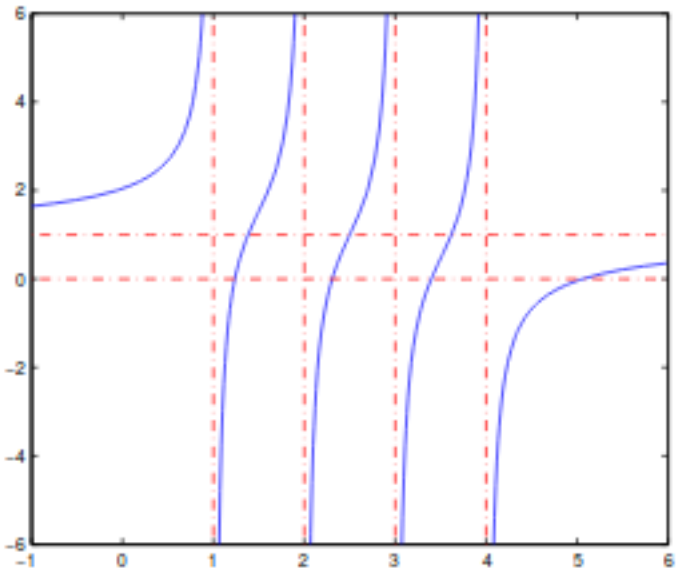
于是

$$det(I+\alpha(D-\lambda I)^{-1}uu^T)=1+\alpha u^T(D-\lambda I)^{-1}u=1+\alpha\sum_{i=1}^{n}\frac{u_i^2}{d_i-\lambda}\triangleq f(\lambda)$$

故求 A 的特征值等价于求特征方程 $f(\lambda)=0$ 的根。
由于

$$f'(\lambda)=\alpha\sum_{i=1}^{n}\frac{u_i^2}{(d_i-\lambda)}$$

当所有的 d_i 都互不相同，且所有的 u_i 都不为零时，$f(\lambda)$ 在 $\lambda\neq d_i$ 处都是严格单调的。



所以 $f(\lambda)$ 在每隔区间 (d_{i+1},d_i) 内都有一个根，共 $n-1$ 个，另一个根在 (d_1,∞)（若 $\alpha>0$）或（$-\infty,d_n$）若 $\alpha<0$）中。

由于 $f(\lambda)$ 在每个区间 (d_{i+1},d_i) 内光滑且严格单调递增（$\alpha>0$）或递减（$\alpha<0$），所以在实际计算中，可以使用对分法，牛顿法及其变形，或有理逼近等算法求解。通常都很快收敛，一般只需迭代几步即可。

因此，计算一个特征值的运算量约为 $\mathcal{O}(n)$，计算 $D+\alpha\ uu^T$ 的所有特征向量。

引理　设 $D\in\mathbb{R}^{n\times n}$ 为对角矩阵，$u\in\mathbb{R}^n$，$\alpha\in\mathbb{R}$，若 λ 是 $D+\alpha\ uu^T$ 的特征值，且 $\lambda\neq d_i, i=1,2,...,n$，则 $(D-\lambda I)^{-1}u$ 是其对应的特征向量。

算法 4.1　计算对称三对角矩阵的特征值和特征向量的分而治之法

1: function $[Q,\Lambda]=dc\_eig(T)$　　$\%T=Q\Lambda Q^T$

2: if $T$is of 1×1then

3: $Q = 1$,$\Lambda = T$

4: return

5: end if

6: form $T = \begin{bmatrix} T_1 & 0 \\ 0 & T_2 \end{bmatrix} + b_m v v^\top$

7: $[Q_1, \Lambda_1] = \mathrm{d}c\_\,\mathrm{eig}\,(T_1)$

8: $[Q_2, \Lambda_2] = \mathrm{d}c\_\,\mathrm{eig}\,(T_2)$

9: form $D + \alpha u u^T$ from $\Lambda_1, \Lambda_2, Q_1, Q_2$

10: compute the eigenvalues Λ and eigenvectors $\hat{Q}$ of $D + \alpha u u^T$

11: compute the eigenvalues of T with $Q = \begin{bmatrix} Q_1 & 0 \\ 0 & Q_2 \end{bmatrix} \cdot \hat{Q}$

12: end

在分而治之法中, 计算特征值和计算特征向量是同时进行的。

下面我们详细讨论分而治之算法的几个细节问题：

(1) 如何减少运算量；

(2) 如何求解特征方程 $f(\lambda) = 0$；

(3) 如何稳定的计算特征向量。

(1) 如何减小运算量——收缩技巧（deflation）

分而治之算法的计算复杂性分析如下：用 $t(n)$ 表示对 n 阶矩阵调用函数 dc_eig 的运算量，则

$$\begin{aligned} t(n) =& 2t(n/2) && \text{递归调用} dc_eig \text{两次} \\ &+ \mathcal{O}(n^2) && \text{计算} D + \alpha u u^T \text{的特征值和特征向量} \\ &+ c{\cdot}n^3 && \text{计算} Q \end{aligned}$$

如果计算 Q 时使用的是稠密矩阵乘法，则 $c = 2$；若不计 $\mathcal{O}(n^2)$ 项，则由递归公式 $t(n) = 2t(n/2) + c{\cdot}n^3$ 可得 $t(n) \approx c{\cdot}4n^3/3$。

但事实上，由于收缩现象的存在，常熟 c 通常比 1 小得多。

在前面的算法描述过程中，我们假定 d_i 互不相等且 u_i 不能等于零。

事实上，当 $d_i = d_{i+1}$ 或 $u_i = 0$ 时，d_i 即为 $D + \alpha u u^T$ 的特征值，这种现象我们成为收缩。

在实际计算时，当 $d_i - d_{i+1}$ 或 $|u_i|$ 小于一个给定的阈值时，我们就金斯认为 d_i 为 $D + \alpha u u^T$ 的特征值，即出现收缩现象。

在实际计算中，收缩现象会经常发生，而且会非常频繁，所以我们可以而且应该利用这种有点加快分而治之算法的速度。

由于主要的计算量集中在计算 Q，即算法的最后一步的矩阵的乘积。如果 $u_i = 0$，则 d_i 为特征值，其对应的特征向量 e_i，即 $\hat{Q}$ 的第 i 列为 e_i，故计算 Q 的第 i 列时不需要做任何的计算。

当 $d_i = d_{i+1}$ 时，也存在一个类似的简化。

(2) 特征方程求解

通常我们可以使用牛顿法来计算特征方程 $f(\lambda) = 0$ 的解。当 $d_i \neq d_{i+1}$ 且 $u_i \neq 0$ 时，用牛顿法计算 $f(\lambda)$ 在 (d_{i+1}, d_i) 中的零点 λ_i。如果 $|u_i|$ 小于给定的阈值时，我们可直接将 d_i 作为特征值 λ_i 的一个近似。但当 u_i 很小（却大于给定的阈值）时，此时 $f(\lambda)$ 在区间 $[d_{i+1}, d_i]$ 中的大部分处的斜率几乎为 0（见下图）。这是，如果任取 $[d_{i+1}, d_i]$ 中的一个点作为迭代初始点，经过一次牛顿迭代后，迭代解可能会跑到区间 $[d_{i+1}, d_i]$ 的外面，造成不收敛。

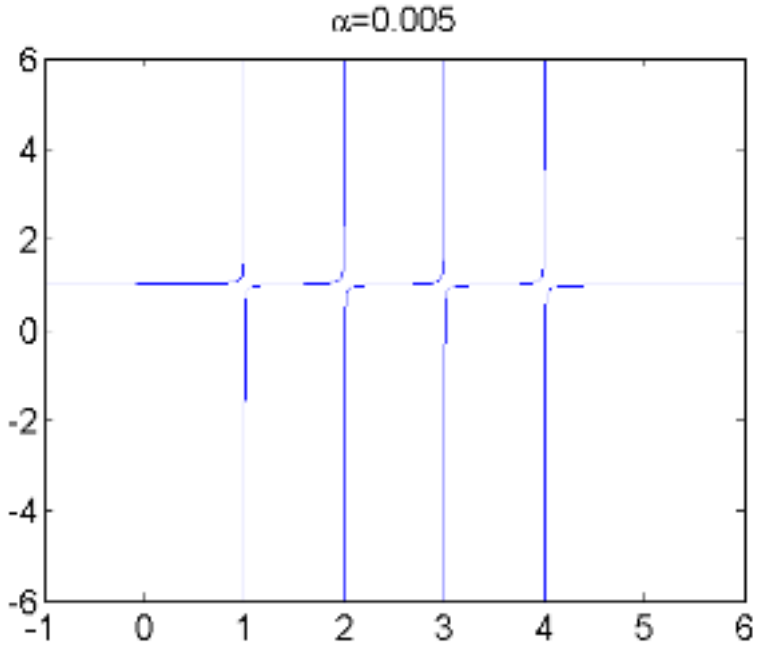


图 3: $f(\lambda) = 1 + 0.005(\frac{1}{4-\lambda} + \frac{1}{3-\lambda} + \frac{1}{2-\lambda} + \frac{1}{1-\lambda})$ 的图像

这时需要采用修正的牛顿法。假设我们已经计算出 λ_i 的一个近似 $\tilde{\lambda}$，下面我们需要从 $\tilde{\lambda}$ 出发，利用牛顿迭代计算下一个近似，直至收敛。我们知道牛顿法的基本原理是使用 $f(\lambda)$ 在点 $\tilde{\lambda}$ 的切线来近似 $f(\lambda)$，并将切线的零点作为下一个近似，即用直线来近似曲线 $f(\lambda)$。

当 u_i 很小时，这种近似方法会出现问题，此时不能使用直线来近似 $f(\lambda)$。这时我们可以寻找其他简单函数 $h(\lambda)$ 来近似 $f(\lambda)$，然后用 $h(\lambda)$ 的零点作为 $f(\lambda)$ 零点的近似，并不断迭代下去，直至收敛。

当然，$h(\lambda)$ 需要满足一定的要求：

（1）必须容易构造；

（2）其零点容易计算；

（3）尽可能与 $f(\lambda)$ 相近。

下面给出构造 $h(\lambda)$ 的一种方法。

因为 d_i 和 d_{i+1} 是 $f(\lambda)$ 的奇点，所以我们令

$$h(\lambda)=\frac{c_1}{d_i-\lambda}+\frac{c_2}{d_{i+1}-\lambda}+c_3$$

其中 c_1,c_2,c_3 为参数。显然，$h(\lambda)$ 的零点很容易计算（与 Newton 法相差无几）。在选取这些参数时，要使得 $h(\lambda)$ 在 $\tilde{\lambda}$ 附近尽可能地接近 $f(\lambda)$。记

$$\begin{aligned}f(\lambda)=1+\alpha\sum_{k=1}^{n}\frac{u_k^2}{d_k-\lambda}&=1+\alpha(\sum_{k=1}^{i}\frac{u_k^2}{d_k-\lambda}+1+\alpha\sum_{k=i+1}^{n}\frac{u_k^2}{d_k-\lambda})\\&\triangleq 1+\alpha\left(\Psi_1(\lambda)+\Psi_2(\lambda)\right)\end{aligned}$$

当 $\lambda\in(d_{i+1},d_i)$ 时，$\Psi_1(\lambda)$ 为正项和，$\Psi_2(\lambda)$ 为负项的和，因此它们都可以较精确地计算。但如果把它们加在一起时可能会引起对消，从而失去相对精度。因此我们也将 $h(\lambda)$ 写成

$$h(\lambda)=1+\alpha(h_1(\lambda)+h_2(\lambda))$$

其中

$$h_1(\lambda)=\frac{c_1}{d_i-\lambda}+\hat{c}_1,\quad h_2(\lambda)=\frac{c_2}{d_{i+1}-\lambda}+\hat{c}_2$$

满足

$$\begin{aligned}h_1(\tilde{\lambda})&=\Psi_1(\tilde{\lambda}),\quad h_1'(\tilde{\lambda})=\Psi_1'(\tilde{\lambda})\\h_2(\tilde{\lambda})&=\Psi_2(\tilde{\lambda}),\quad h_2'(\tilde{\lambda})=\Psi_2'(\tilde{\lambda})\end{aligned}$$

即 $h_1(\lambda)$ 和 $h_2(\lambda)$ 分别在点 $\tilde{\lambda}$ 与 $\Psi_1(\lambda)$ 和 $\Psi_2(\lambda)$ 相切。这在数值插值中是常见的条件。容易计算可得

$$\begin{cases}c_1=\Psi_1'(\tilde{\lambda})\left(d_i-\tilde{\lambda}\right)^2, & \hat{c}_1=\Psi_1(\tilde{\lambda})-\Psi_1'(\tilde{\lambda})\left(d_i-\tilde{\lambda}\right)\\ c_2=\Psi_2'(\tilde{\lambda})\left(d_{i+1}-\tilde{\lambda}\right)^2, & \hat{c}_2=\Psi_2(\tilde{\lambda})-\Psi_2'(\tilde{\lambda})\left(d_{i+1}-\tilde{\lambda}\right)\end{cases}\tag{100}$$

所以，最后取

$$h(\lambda)=1+\alpha\left(\hat{c}_1+\hat{c}_2\right)+\alpha\left(\frac{c_1}{d_i-\lambda}+\frac{c_2}{d_{i+1}-\lambda}\right)\tag{101}$$

这就是迭代函数。算法 4.2　修正的 Newton 算法

1: set$k=0$

2: choose an initial guess $\lambda_0\in[d_{i+1},d_i]$

3: while not convergence do

4:　　let$\tilde{\lambda}=\lambda_k$and compute $c_1,c_2,\hat{c}_1,\hat{c}_2$from 100

5: set $k = k + 1$

6: compute the solution $\lambda_k$of $h(\lambda)$defined by 101

7: end while

(3) 计算特征向量的稳定算法

设 λ_i 是 $D + \alpha uu^T$ 的特征值，则根据引理 4.2，可利用公式 $(D - \lambda_i I)^{-1} u$ 来计算其对应的特征向量。但遗憾的是，当相邻的两个特征值非常接近时，这个公式可能不稳定。即当 λ_i 与 λ_{i+1} 非常接近时，他们都靠近 d_{i+1}（这里假定 $\lambda_i \in (d_{i+1}, d_i)$），在计算 $d_{i+1} - \lambda_i$ 和 $d_{i+1} - \lambda_{i+1}$ 时会存在对消，这就可能损失有效数字，产生较大的相对误差，从而导致 $(D - \lambda_i I)^{-1} u$ 与 $(D - \lambda_{i+1} I)^{-1} u$ 的计算时不准确的，正交性也会失去。下面的定理可以解决这个问题。

定理（Löwner） 设对角阵 $D = \operatorname{diag}(d_1, d_2, \ldots, d_n)$ 满足 $d_1 > d_2 > \cdots > d_n$，若矩阵 $\hat{D} = D + \hat{u}\hat{u}^\top$ 的特征值 $\lambda_1, \lambda_2, \ldots, \lambda_n$ 满足交错性质

$$\lambda_1 > d_1 > \lambda_2 > d_2 > \cdots > \lambda_n > d_n \tag{102}$$

则向量 $\hat{u}$ 的分量满足

$$|\hat{u}_i| = \left(\frac{\prod_{k=1}^{n} (\lambda_k - d_i)}{\prod_{k=1, k \neq i}^{n} (d_k - d_i)} \right)^{1/2} \tag{103}$$

因此，我们可以采用公式103来计算特征向量。这样就尽可能的避免了出现分母很小的情形。

箭型分而治之法

分而治之算法于 1981 年被首次提出，但直到 1995 年才由 Gu 和 Eisenstat 给出了一种快速稳定的实现方式，称为箭型分而治之法 (Arrowhead Divide-and-Conquer, ADC). 他们做了大量的数值试验，在试验中，当矩阵规模不超过 6 时，就采用对称 QR 迭代来计算特征值和特征向量。在对特征方程求解时，他们采用的是修正的有理逼近法。数值结果表明，ADC 算法的计算精度可以与其他算法媲美，而计算速度通常比对称 QR 迭代快 5 至 10 倍，比 Cuppen 的分而治之法快 2 倍。详细介绍参见相关文献。

6.5 对分法和逆迭代

对分法的基本思想是利用惯性定理来计算所需的部分特征值。

定义 设 A 为对称矩阵，则其惯性定义为

$$(A) = (\nu, \zeta, \pi)$$

其中 ν, ζ, π 分别表示 A 的负特征值，零特征值和正特征值的个数。

定理 (Sylvester 惯性定理) 设 $A \in \mathbb{R}^{n \times n}$ 是对称矩阵，$X \in \mathbb{R}^{n \times n}$ 非奇异，则 $X^T A X$ 与 A 有相同的惯性。

利用 LU 分解可得 $A - zI = LDL^\top$，其中 L 为奇异下三角矩阵，D 为对角阵，则

$$(A - zI) = \text{ Inertia } (D)$$

由于 D 时对角矩阵，所以 Inertia (D) 很容易计算。

设 $\alpha \in \mathbb{R}^n$，记 Negcount(A, α) 为小于 α 的 A 的特征值的个数，即

$$\text{Negcount}(A, \alpha) = \#(\lambda(A) < \alpha)$$

设 $\alpha_1 < \alpha_2$，则 A 在区间 $[\alpha_1, \alpha_2)$ 中的特征值个数为

$$(A, \alpha_2) - \text{Negcount}\,(A, \alpha_1)$$

。

如果 $\alpha_2 - \alpha_1 < tol$（其中 $tol \ll 1$ 为事先给定的阈值），且 A 在 $[\alpha_1, \alpha_2)$ 中有特征值，则我们可将 $[\alpha_1, \alpha_2)$ 中的任意一个值作为 A 在该区间中的特征值的近似。

由此我们可以给出下面的对分法。

算法 5.1　计算 A 在 $[a, b)$ 中的所有特征值

Let tolbe a given threshold
compute n_a =Negcount (A, a)
compute n_b =Negcount (A, b)
if $n_a = n_b$then
    return　　% 此时 $[a, b)$ 中没有 A 的特征值
end if
put (a, n_a, b, n_b)onto worklist
    %worklist 中的元素时“四元素对，即由四个数组成的数对
while worklist not empty do
    remove$(low, n_{low}, up, n_{up})$from the worklist
    %$(low, n_{low}, up, n_{up})$ 是 worklisth 中的任意一个元素
    if$(up - low) < tol$ then
        print ”There are $n_{up} - n_{low}$eigenvalues in[low,up)”
    else
        compute $mid = (low + up)/2$
        computen_{mid} = Negcount (A, mid)
        if $(n_{mid} > n_{low})$ then
            put $(low, n_{low}, mid, n_{mid})$onto worklist
        end if

20: if $(n_{up} > n_{mid})$ then

21: $(mid, n_{mid}, up, n_{up})$onto worklist

22: end if

23: end if

24: end while

对分法的主要运算量集中在计算 Negcount (A, z)。通常是事先将 A 转化成对称三对角矩阵，这样计算 $A - zI$ 的 $\mathrm{LDL}^{\mathrm{T}}$ 分解就非常简单：

$$A - zI = \begin{bmatrix} a_1 - z & b_1 & & \\ b_1 & \ddots & \ddots & \\ & \ddots & \ddots & b_{n-1} \\ & & b_{n-1} & a_n - z \end{bmatrix}$$

$$= \begin{bmatrix} 1 & & & \\ l_1 & \ddots & & \\ & \ddots & \ddots & \\ & & l_{n-1} & 1 \end{bmatrix} \begin{bmatrix} d_1 & & & \\ & \ddots & & \\ & & \ddots & \\ & & & d_n \end{bmatrix} \begin{bmatrix} 1 & l_1 & & \\ & \ddots & \ddots & \\ & & \ddots & l_{n-1} \\ & & & 1 \end{bmatrix} \triangleq LDL^{\top}$$

利用待定系数法，可以得到下面的递推公式

$$d_1 = a_1 - z, \quad d_i = (a_i - z) - \frac{b_{i-1}^2}{d_{i-1}}, \quad i = 2, 3, \ldots, n \tag{104}$$

用上面的公式计算 d_i 的运算量约为 $4n$。

注意这里没有选主元，但针对对称三对角矩阵，该算法是非常稳定的，即使当 d_i 有可能很小时，算法依然很稳定。

定理 [Demmel '97]利用公式104计算所得的 d_i 与精确计算 $\hat{A}$ 的 $\hat{d}_i$ 有相同的符号，故有相同的惯性。这里 $\hat{A}$ 与 A 非常接近，即

$$\hat{A}(i,i) = a_i, \quad \hat{A}(i,i+1) = b_i(1 + \varepsilon_i)$$

其中，$|\varepsilon_i| \leq 2.5\varepsilon + O(\varepsilon^2)$，这里 ε 为机器精度。

- 由于单独调用一次 Negcount 的运算量为 $4n$，故计算 k 个特征值的总运算量约为 $O(kn)$；
- 当当特征值计算出来后，我们可以使用带位移的逆迭代来计算对应的特征向量。通常只需迭代 1 至 2 次即可，由于 A 是三对角矩阵，故计算每个特征向量的运算量为 $O(n)$；
- 当特征值紧靠在一起时，计算出来的特征向量可能会失去正交性，此时需要进行再正交化，可通过 MGS 的 QR 分解来实现。

6.6 奇异值分解

奇异值分解 (SVD) 具有十分广泛的应用背景, 因此, 如何更好更快地计算一个给定矩阵的 SVD 是科学与工程计算领域中的一个热门研究课题, 吸引了众多专家进行这方面的研究, 也涌现出了许多奇妙的方法. 本章主要介绍计算 SVD 的常用算法。

对任意矩阵 $A \in \mathbb{R}^{m\times n}$,其奇异值与对称矩阵 A^TA, AA^T 和 $\begin{bmatrix} 0 & A^\top \\ A & 0 \end{bmatrix}$ 的特征值是密切相关的,故理论上计算对称特征值的算法都可以用于计算奇异值。但在实际计算中,我们通过可以利用 SVD 的特殊结构使得算法更加有效和准确。

与计算对称矩阵的特征值累死,计算一个矩阵 A 的奇异值分解的算法通常分为一下几个步骤(Jacobi 算法除外):

1. 将 A 二对角化:$B = U_1^TAV_1$,其中 B 为上二对角矩阵,U_1, V_1 为正交阵;
2. 计算 B 的 SVD:$B = U_2\sum V_2^T$,其中 $\sum$ 为对角阵,U_2, V_2 为正交阵;
3. 合并得到 A 的 SVD:$A = U_1BV_1^T = (U_1U_2)B(V_1V_2)^T$。

6.6.1 二对角化

我们知道。对称矩阵可以通过一系列 Householder 变换转化为对称三对角矩阵。对于一般矩阵 $A \in \mathbb{R}^{m\times n}$, 我们也可以通过 Householder 变换,将其转化为而对角矩阵,即计算正交矩阵 U_1 和 V_1 使得

$$U_1^TAV_1 = B \tag{105}$$

其中 B 是一个实(上)二对角矩阵。这个过程就称为二对角化。

需要注意的是,与对称矩阵的对称三对角化不同,A 与 B 是不相似的。

设 $A \in \mathbb{R}^{m\times n}$,二对角化过程大致如下:

(1) 首先确定一个 Household 矩阵 $H_1 \in \mathbb{R}^{m\times n}$,使得 H_1A 的第一节除第一个元素外,其他分量都为零,即

$$H_1A = \begin{bmatrix} * & * & * & * & * \\ 0 & * & * & \cdots & * \\ 0 & * & * & \cdots & * \\ 0 & * & * & \cdots & * \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & * & * & \cdots & * \end{bmatrix}$$

(2) 再确定一个 Household 矩阵 $\tilde{H}_1 \in \mathbb{R}^{(n-1)\times(n-1)}$,把 H_1A 的第一行的第 3 至第 n 个

元素化为零，即

$$H_1 A \begin{bmatrix} 1 & 0 \\ 0 & \tilde{H}_1 \end{bmatrix} = \begin{bmatrix} * & * & 0 & \cdots & 0 \\ 0 & * & \cdots & * & \\ 0 & * & \cdots & * & \\ 0 & * & \cdots & * & \\ \vdots & \vdots & & \vdots & \\ 0 & * & \cdots & * & \end{bmatrix}$$

(3) 重复上面的过程，直到把 A 最终化为而对角矩阵。

有了分解105以后，我们可得

$$A^\top A = \left(U_1 B V_1^\top\right)^\top U_1 B V_1^\top = V_1 B^\top B V_1^\top$$

即 $V_1^\top A^\top A V_1 = B^\top B$。由于 $B^T B$ 是对称三对角的，所以这就相当于将 $A^T A$ 三对角化。

整个二对角化过程的运算量约为 $4mn^2 + 4m^2 n - 4n^3/3$。若不需要计算 U_1 和 V_1，则运算量约为 $4mn^2 - 4n^3/3$。

二对角矩阵的奇异值分解

设 $B \in \mathbb{R}^{n\times n}$ 是一个而对角矩阵 $B = \begin{bmatrix} a_1 & b_1 & & & \\ & \ddots & \ddots & & \\ & & \ddots & & \\ & & \ddots & b_{n-1} & \\ & & & a_n & \end{bmatrix}$，则下面三种方法均可将计算 B 的 SVD 转化成计算对称三对角矩阵的特征分解：

(1) 令 $A = \begin{bmatrix} 0 & B^\top \\ B & 0 \end{bmatrix}$，置换阵 $P = [e_1, e_{n+1}, e_2, e_{n+2}, \ldots, e_n, e_{2n}]$，则 $T_{ps} = P^\top A P$ 是对称三对角矩阵，且 T_{ps} 的主对角线元素全为 0，次对角线元素为 $a_1, b_1, a_2, b_2, \ldots, a_{n-1}, b_{n-1}, a_n$。若 (λ_i, x_i) 是 T_{ps} 的一个特征对，则

$$\lambda_i = \pm\sigma_i, \quad Px_i = \frac{1}{\sqrt{2}} \begin{bmatrix} v_i \\ \pm u_i \end{bmatrix}$$

，其中 σ_i 为 B 一个奇异值，u_i 和 v_i 分别为对应的左和右奇异向量。

(2) 令 $T_{BB^\top} = BB'$，则

$$T_{BB^\top} = \begin{bmatrix} a_1^2 + b_1^2 a_2 b_1 & & & \\ a_2 b_1 & \ddots & \ddots & \\ & \ddots & a_{n-1}^2 + b_{n-1}^2 a_n b_{n-1} & \\ & & a_n b_{n-1} & a_n^2 \end{bmatrix}$$

$T_{BB^\top}$ 的特征值为 B 的奇异值的平方，且 $T_{BB^\top}$ 的特征向量为 B 的左奇异向量。

(3) 令 $T_{BB^\top} = BB'$，则

$$T_{B^\top B} = \begin{bmatrix} a_1^2 & a_1 b_1 & & \\ a_1 b_1 & a_2^2 + b_1^2 & \ddots & \\ & \ddots & \ddots & a_{n-1} b_{n-1} \\ & & a_{n-1} b_{n-1} & a_n^2 + b_{n-1}^2 \end{bmatrix}$$

$T_{B^\top B}$ 的特征值为 B 的奇异值的平方，且 $T_{B^\top B}$ 的特征向量为 B 的右奇异向量。

理论上，我们可以直接使用 QR 迭代、分而治之法或带反迭代的对分法，计算三对角矩阵的 $T_{ps}, T_{BB^\top}$ 和 $T_{B^\top B}$ 的特征值和特征向量。但一般来说，这种做法并不是最佳的，原因如下：

(1) 对 T_{ps} 做 QR 迭代并不划算，因为 QR 迭代计算所有的特征值和特征向量，而事实上只要计算正的特征值即可；

(2) 直接构成 $T_{BB^\top}$ 或 $T_{B^\top B}$ 是数值不稳定的。事实上，这样做可能会使得 B 的小奇异值的精度丢失一半。

下面是一些奇异值分解的比较实用的算法。

1. Golub-Kahan SVD 算法: 由 Golub 和 Kahan 于 1965 年提出, 是一种十分稳定且高效的计算 SVD 的算法。主要思想是将带位移的对称 QR 迭代算法隐式地用到 $B^T B$ 上, 在该算法中, 并不需要显示地把 $B^T B$ 计算出来。该算法也通常就称为 SVD 算法, 是一个基本且实用的算法, 目前仍然是计算小规模矩阵奇异值分解的常用算法。

2. dqds 算法: 由 Fernando 和 Parlett 于 1994 年提出, 是计算二对角矩阵所有奇异值的最快算法, 而且能达到很高的相对精度, 包括奇异值很小的情形。该算法主要基于对 $B^T B$ 的 Cholesky 迭代, 可以看作是 LR 迭代算法的改进。由于 LR 迭代算法在一定条件下与对称 QR 算法是等价的, 因此该算法也可以看作是 QR 迭代的变形。

3. 分而治之法：该算法是计算维数 $n \geq 25$ 的矩阵的所有奇异值和奇异向量的最快算法，但不能保证小奇异值的相对精度，即 σ_i 的相对精度为 $O(\varepsilon)\sigma_1$，而不是 $O(\varepsilon)\sigma_i$。

4. 对分法和反迭代: 主要用于计算某个区间内的奇异值及对应的奇异向量, 能保证较高的相对精度。

5. Jacobi 迭代：可隐式地对 AA^T 或 $A^T A$ 实施对称 Jacobi 迭代，能保证较高的相对精度。最近，Z.Drmac 和 K.Veselić 改进了最初的 Jacobi 算法, 使其变成一个速度快、精度高的实用算法。

在这里，我们简要介绍 Golub-Kahan SVD 算法, dqds 算法和 Jacobi 迭代。

6.6.2 Golub-Kahan SVD 算法

该算法主要思想是将带位移的对称 QR 迭代算法隐式地用到 B^TB 上，而无需将 B^TB 显示的计算出来。
算法基本框架

Golub-Kahan SVD 算法有时也简称 SVD 算法, 其基本框架是:

- 将矩阵 A 二对角化，得到上二对角矩阵 B；
- 用隐式 QR 迭代计算 B^TB 的特征值分解，即

$$B^\top B = Q\Lambda Q^\top, \quad \Lambda = \text{diag}\left(\sigma_1^2, \sigma_2^2, \ldots, \sigma_n^2\right) \tag{106}$$

- 计算 BQ 的列主元 QR 分解，即

$$(BQ)P = UR \tag{107}$$

其中 P 是置换矩阵，U 是正交矩阵，R 是上三角矩阵。

由106可知

$$(BQ)^\top BQ = \Lambda$$

因此 BQ 是列正交矩阵 (但不是单位列正交)。再由107可知 $R = U^T(BQ)P$ 也是列正交矩阵。又 R 是上三角矩阵，所以 R 必定是对角矩阵。令 $V = QP$，则由107可知

$$U^TBV = R$$

这就是二对角矩阵 B 的奇异值分解。

算法的具体实现参见相关文献

6.6.3 dqds 算法

我们首先介绍针对实对称正定矩阵的 LR 算法, 该算法思想与 QR 迭代算法类似, 但提出时间更早。
算法 6.1　带位移的 LR 算法

Let T_0 be a given real symmetric positive definite matrix
set $i = 0$
while not converge do
    choose a shiftτ_i^2 satisfying $\tau_i^2 < \min\{\lambda(T_i)\}$
    compute B_i such that $T_i - \tau_i^2 I = B_i^\top B_i$　　%Cholesky factorization
    $T_{i+1} = B_iB_i^\top + \tau_i^2 I$
    $i = i + 1$

8: end while

LR 迭代算法在形式上与 QR 迭代算法非常类似。事实上，对于不带位移的 LR 迭代算法，我可以证明，两步 LR 迭代等价于一步 QR 迭代。

引理　设 $\tilde{T}$ 是不带位移的 LR 算法迭代两步后生成的矩阵，$\hat{T}$ 是不带唯一的 QR 算法迭代一步后生成的矩阵，则 $\tilde{T} = \hat{T}$。

(1) LR 算法中要求 T_0 对称正定，但并不一定是三对角矩阵；

(2) 由该引理可知，QR 算法与 LR 算法有相同的收敛性。

dqds 算法

该算法是针对三对角的对称正定矩阵 B^TB，其中 B 是而对角矩阵。在数学上，dqds 算法与 LR 算法是等价的，但在该算法中，我们是直接通过 B_i 来计算 B_{i+1}，从而避免计算中间矩阵 T_{i+1}，这样也就尽可能的避免了由于计算 $B_iB_i^T$ 而可能带来的数值不稳定性。

下面推导如何从 B_i 直接计算 B_{i+1}。设

$$B_i = \begin{bmatrix} a_1 & b_1 & & \\ & a_2 & \ddots & \\ & & \ddots & b_{n-1} \\ & & & a_n \end{bmatrix}, \quad B_{i+1} = \begin{bmatrix} \tilde{a}_1\tilde{b}_1 & & \\ \tilde{a}_2 & \ddots & \\ & \ddots & \tilde{b}_{n-1} \\ & & \tilde{a}_n \end{bmatrix}$$

为了书写方便，我们记 $b_0 = b_n = \tilde{b}_0 = \tilde{b}_n = 0$。由 LR 算法 6.1 可知

$$B_{i+1}^\top B_{i+1} + \tau_{i+1}^2 I = B_i B_i^\top + \tau_i^2 I$$

比较等式两边矩阵的对角线和上对角线元素，可得

$$\tilde{a}_k^2 + \tilde{b}_{k-1}^2 + \tau_{i+1}^2 = a_k^2 + b_k^2 + \tau_i^2, \quad k = 1, 2, \ldots, n$$

$$\tilde{a}_k\tilde{b}_k = a_{k+1}b_k \quad 或 \quad \tilde{a}_k^2\tilde{b}_k^2 = a_{k+1}^2 b_k^2, \quad k = 1, 2, \ldots, n-1$$

记 $\delta = \tau_{i+1}^2 - \tau_i^2, p_k = a_k^2, q_k = b_k^2, \tilde{p}_k = \tilde{a}_k^2, \tilde{q}_k = \tilde{b}_k^2$，则可得 qds 算法：

算法 6.2　qds 算法的单步 $(B_i \to B_{i+1})$

1: $\delta = \tau_{i+1}^2 - \tau_i^2$

2: for $k = 1$ to $n-1$ do

3: $\quad \tilde{p}_k = p_k + q_k - \tilde{q}_{k-1} - \delta$

4: $\tilde{q}_k = q_k \cdot (p_{k+1}/\tilde{p}_k)$

5: end for

6: $\tilde{p}_n = p_n - \tilde{q}_{n-1} - \delta$

qds 算法中的每个循环仅需 5 个浮点运算，所以运算量较少。

为了体验算法的精确性，我们引入一个辅助变量 $d_k \triangleq p_k - \tilde{q}_{k-1} - \delta$，则

$$\begin{aligned} d_k &= p_k - \tilde{q}_{k-1} - \delta \\ &= p_k - \frac{q_{k-1}p_k}{\tilde{p}_{k-1}} - \delta \\ &= p_k \cdot \frac{\tilde{p}_{k-1} - q_{k-1}}{\tilde{p}_{k-1}} - \delta \\ &= p_k \cdot \frac{p_{k-1} - \tilde{q}_{k-2} - \delta}{\tilde{p}_{k-1}} - \delta \\ &= \frac{p_k}{\tilde{p}_{k-1}} \cdot d_{k-1} - \delta \end{aligned}$$

于是就可得到 dqds 算法。算法 6.3　dqds 算法的单步 $(B_i \to B_{i+1})$

1: $\delta = \tau_{i+1}^2 - \tau_i^2$

2: $d_1 = p_1 - \delta$

3: for $k = 1$ to $n - 1$ do

4: $\quad \tilde{p}_k = d_k + q_k$

5: $\quad t = p_{k+1}/\tilde{p}_k$

6: $\quad \tilde{q}_j = q_k \cdot t$

7: $\quad d_{k+1} = d_k \cdot t - \delta$

8: end for

9: $\tilde{p}_n = d_n$

dqds 算法的运算量与 dqs 差不多，但更精确。

下面的定理显示了 dqds 算法的高精度性质。

定理　以浮点运算对 B 做单步 dqds 迭代，得到矩阵 $\tilde{B}$，该过程等价于

1. 对 B 的每个元素座椅而小德相对扰动（不超过 1.5ε），得到 $\tilde{B}$；
2. 对 $\tilde{B}$ 应用精确的 dqds 算法的单步，得到 $\overline{B}$；
3. 对 $\overline{B}$ 的每个元素做一个小的相对扰动（不超过 ε），得到 $\tilde{B}$。

因此，B 和 $\tilde{B}$ 的奇异值满足高的相对精度。

关于 dqds 算法中位移的选取，以及如何判断收敛性，可以参见相关文献。

6.6.4 **Jacobi 算法**

本节讨论对矩阵 $M = A^T A$ 实施隐式的 Jacoi 算法来计算 A 的奇异值。

我们知道，Jacobi 算法的每一步就是对矩阵作 Jacobi 旋转，即 $A^T A \to J^T A^T A J$，其中 J 的选取将两个非对角元化为 0。在实际计算中，我们只需计算 AJ，故该算法称为单边 Jacobi 旋转。

算法 6.4　单边 Jacobi 旋转的单步

% 对 $M = A^T A$ 作 Jacobi 旋转，将 $M(i.j), M(j,i)$ 化为 0

1: Compute $m_{ii} = (A^T A)_{ii}, m_{ij} = (A^T A)_{ij}, m_{jj} = (A^T A)_{jj}$

2: if m_{ij} is not small enough then

3: $\tau = (m_{ii} - m_{jj}) / (2 \cdot m_{ij})$

4: $t = \text{sign}(\tau) / \left(|\tau| + \sqrt{1 + \tau^2}\right)$

5: $c = 1/\sqrt{1 + t^2}$

6: $s = c \cdot t$

7: $A = AG(i, j, \theta)$　　% $G(i, j, \theta)$ 为 Givens 变换

8: if eigenvectors are desired then

9: $J = J \cdot G(i, j, \theta)$

10: end if

11: end if

在上面算法的基础上，我们可以给出完整的单边 Jacobi 算法。blue算法 6.5　单边 Jacobi 算法：计算 $A = U \sum V^T$

1: while $A^T A$ is not diagonal enough do

2: for $i = 1$ to $n - 1$do

3: for $j = i + 1$to n do

4: 调用单边 Jacobi 旋转

5: end for

6: end for

7: end while

8: compute $\sigma_i = \|A(:, i)\|_2, i = 1, 2, \ldots n$

9: $U = [u_1, \ldots, u_n]$with $u_i = A(:, i)/\sigma_i$

10: $V = J$

Jacobi 算法的特点

- 不需要双对角化，这样可以避免双对角化引入的误差；
- 可以达到相对较高的计算精度；
- 速度较慢。（目前已有快速的改进算法）

定理　设 $A = DX \in \mathbb{R}^{n\times n}$，其中 D 为非奇异对角阵，X 非奇异。设 $\hat{A}$ 是按浮点运算单边 Jacobi 旋转 m 次后所得到的矩阵。若 A 和 $\hat{A}$ 的奇异值分别为 $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_n$ 和 $\hat{\sigma}_1 \geq \hat{\sigma}_2 \geq \ldots \geq \hat{\sigma}_n$，则

$$\frac{|\hat{\sigma}_i - \sigma_i|}{\sigma_i} \leq O(m\varepsilon)\kappa(X)$$

故 X 的条件数越小，计算矩阵 A 的奇异值时相对误差越小。

6.7 扰动分析

设 $A \in \mathbb{R}^{n\times n}$ 是对称矩阵，则存在一个蒸饺矩阵 Q 使得

$$A = Q\Lambda Q^T$$

其中 $\Lambda = \mathbf{diag}\,(\lambda_1, \lambda_2, \ldots, \lambda_n)$ 是一个实对角矩阵。

这里的 λ_i 就是 A 的特征值，我们假设 $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$。令 $Q = [q_1, q_2, \ldots, q_n]$，则 q_i 就是 λ_i 对应的单位正交特征向量。

关于对称矩阵特征值问题的扰动理论，这里只做一些简单介绍，若要深入了解这方面的信息，可以参考相关文献。

6.7.1 特征值与 Rayleigh 商

定义　设 $A \in \mathbb{R}^{n\times n}$ 是对称矩阵，向量 $x \in \mathbb{R}^n$ 非零，则 x 关于 A 的Rayleigh 商定义为：

$$\rho(x, A) = \frac{x^\top Ax}{x^\top x}$$

有时简记为 $\rho(x)$。

下面是关于 Rayleigh 商的一些基本性质：

(1) $\rho(\alpha x) = \rho(x), \forall \alpha \in \mathbb{R}, \alpha \neq 0$;

(2) $\rho\,(q_i) = \lambda_i, i = 1, 2, \ldots, n$;

(3) 设 $x = \alpha_1 q_1 + \alpha_2 q_2 + \cdots + \alpha_n q_n$，则

$$\rho(x) = \frac{\alpha_1^2\lambda_1 + \alpha_2^2\lambda_2 + \cdots + \alpha_n^2\lambda_n}{\alpha_1^2 + \alpha_2^2 + \cdots + \alpha_n^2}$$

;

(4) $\lambda_n \le \rho(x) \le \lambda_1, |\rho(x)| \le \|A\|_2$。

Courant-Fischer 极小极大定理

实对称矩阵的特征值与 Rayleigh 商之间的一个基本性质是 Courant-Fischer 极小极大定理。

定理 (Courant-Fischer)　设 $A \in \mathbb{R}^{n\times n}$ 是对称矩阵，其特征值为 $\lambda_1 \ge \lambda_2 \ge \cdots \ge \lambda_n$，则有

$$\lambda_k = \max_{\mathrm{U}\in\mathbb{S}_k^n} \min_{x\in\mathbb{U}, x\neq 0} \frac{x^\top Ax}{x^\top x} = \min_{\mathrm{V}\in\mathbb{S}_{n-k+1}^n} \max_{x\in\mathbb{V}, x\neq 0} \frac{x^\top Ax}{x^\top x}$$

其中 S_i^n 表示 $\mathbb{R}^n$ 中所欲 i 维子空间构成的集合，当

$$\mathbb{U} = \operatorname{span}\{q_1, \ldots, q_k\}, \quad \mathbb{V} = \operatorname{span}\{q_k, \ldots, q_n\}, \quad x = q_k$$

时，上式中的等号成立。

Rayleigh-Ritz 定理

当 $k = 1$ 和 $k = n$ 时, 就可以得到下面的定理.

定理 (Rayleigh-Ritz)　设 $A \in \mathbb{R}^{n\times n}$ 是对称矩阵，其特征值为 $\lambda_1 \ge \lambda_2 \ge \cdots \ge \lambda_n$，则有

$$\lambda_1 = \max_{x\in\mathbb{R}^n, x\neq 0} \frac{x^\top Ax}{x^\top x}, \quad \lambda_n = \min_{x\in\mathbb{R}^n, x\neq 0} \frac{x^\top Ax}{x^\top x}$$

特征值分割定理

由极大极小定理，我们可以得到下面的特征值分隔定理。

定理（分割定理）　设 $A \in \mathbb{R}^{n\times n}$ 是对称矩阵，$B = Q^T AQ$，其中 $Q \in \mathbb{R}4^{n\times(n-1)}$ 满足 $Q^TQ = I_{n-1}$。再设 A 和 B 的特征值分别为

$$\lambda_1 \ge \lambda_2 \ge \cdots \ge \lambda_n \quad 和 \quad \tilde{\lambda}_1 \ge \tilde{\lambda}_2 \ge \cdots \ge \tilde{\lambda}_{n-1}$$

则有

$$\lambda_1 \ge \tilde{\lambda}_1 \ge \lambda_2 \ge \tilde{\lambda}_2 \cdots \ge \tilde{\lambda}_{n-1} \ge \lambda_n$$

。

特别地，在上述定理中，取 $Q = [e_1, \ldots, e_{i-1}, e_{i+1}, \ldots, e_n]$，则可以得到下面的结论。

推论　设 $A \in \mathbb{R}^{n\times n}$ 是对称矩阵，$\tilde{A}$ 是 A 的一个 $n-1$ 阶主子矩阵，A 和 $\tilde{A}$ 的特征值分别为

$$\lambda_1 \ge \lambda_2 \ge \cdots \ge \lambda_n \quad 和 \quad \tilde{\lambda}_1 \ge \tilde{\lambda}_2 \ge \cdots \ge \tilde{\lambda}_{n-1}$$

则有

$$\lambda_1 \ge \tilde{\lambda}_1 \ge \lambda_2 \ge \tilde{\lambda}_2 \cdots \ge \tilde{\lambda}_{n-1} \ge \lambda_n$$

。

反复应用上面的推论，即可得到下面的结论。

推论　设 $A \in \mathbb{R}^{n\times n}$ 是对称矩阵，$\tilde{A}$ 是 A 的一个 k 阶主子矩阵 $(1 \le k \le n-1)$，A 和 $\tilde{A}$ 的特征值分别为

$$\lambda_1 \ge \lambda_2 \ge \cdots \ge \lambda_n \quad 和 \quad \tilde{\lambda}_1 \ge \tilde{\lambda}_2 \ge \cdots \ge \tilde{\lambda}_{n-1}$$

则有

$$\lambda_i \geq \tilde{\lambda}_i \geq \lambda_{n-k+i}, \quad i = 1, 2, \ldots, k$$

。

6.7.2 对称矩阵特征值的扰动分析

设 $A \in \mathbb{R}^{n\times n}$ 是对称矩阵，扰动矩阵 $E \in \mathbb{R}^{n\times n} E \in \mathbb{R}^{n\times n}$ 都是对火车呢就在，下面讨论 $A + E$ 的特征值与 A 的特征值之间的关系。

由极小极大定理，我们可以证明下面的性质。

定理　设 $A \in \mathbb{R}^{n\times n}$ 和 $B = A + E \in \mathbb{R}^{n\times n}$ 都是对称矩阵，其特征值分别为

$$\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n \quad 和 \quad \tilde{\lambda}_1 \geq \tilde{\lambda}_2 \geq \cdots \geq \tilde{\lambda}_{n-1}$$

假定 E 的最大特征值和最小特征值分别为 μ_1 和 μ_n，则有

$$\lambda_i + \mu_1 \geq \tilde{\lambda}_i \geq \lambda_i + \mu_n, \quad i = 1, 2, \ldots, n$$

Weyl 定理

根据这个定理，我们可以得到下面的 Weyl 定理。

定理（Weyl）　设 $A \in \mathbb{R}^{n\times n}$ 和 $B = A + E \in \mathbb{R}^{n\times n}$ 都是对称矩阵，其特征值分别为 $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$ 和 $\tilde{\lambda}_1 \geq \tilde{\lambda}_2 \geq \cdots \geq \tilde{\lambda}_{n-1}$，则

$$\left|\tilde{\lambda}_j - \lambda_j\right| \leq \|E\|_2, \quad j = 1, 2, \ldots, n$$

。

该定理的结论可以推广到奇异值情形。

我们首先给出下面的引理。

引理　设 $A \in \mathbb{R}^{m\times n}(m \geq n)$ 的奇异值分解为 $A = U \sum V$，其中 $U = [u_1, ..., u_n] \in \mathbb{R}^{m\times n}$ 为列正交矩阵，$V = [v_1, ...v_n] \in \mathbb{R}^{n\times n}$ 为正交矩阵，$\sum = diag(\sigma_1, \ldots, \sigma_n)$。将 U 扩展成 $n \times n$ 的正交矩阵 $[U, \tilde{U}] = [u_1, \ldots, u_n, \tilde{u}_1, \ldots, \tilde{u}_{m-n}]$，令

$$H = \begin{bmatrix} 0 & A^\top \\ A & 0 \end{bmatrix} \in \mathbb{R}^{(m+n)\times(m+n)} 4$$

则 H 对称，且特征值为 $\pm\sigma_i$ 和 0(其中 0 至少为 $m - n$ 重特征值)，对应的特征向量分别为 $\frac{\sqrt{2}}{2}\begin{bmatrix} v_i \\ \pm u_i \end{bmatrix}, i = 1, 2, \ldots, n, \begin{bmatrix} 0 \\ \tilde{u}_j \end{bmatrix}, j = 1, 2, \ldots, m - n$。

由上面的引理和 Weyl 定理立即可得

定理　设 $A□B \in \mathbb{R}^{m\times n}(m \geq n)$, 他们的奇异值分解为 $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n$ 和 $\tilde{\sigma}_1 \geq \tilde{\sigma}_2 \geq \cdots \geq \tilde{\sigma}_n$，则

$$|\tilde{\sigma}_j - \sigma_j| \leq \|B - A\|_2, \quad j = 1, 2, \ldots, n$$

6.7.3 对称矩阵特征向量的扰动

定义　设 $A \in \mathbb{R}^{n\times n}$ 的特征值为 $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$，则 λ_i 与其余特征值之间的间隙 (gap)定义为

$$\operatorname{gap}(\lambda_i, A) = \min_{j\neq i} |\lambda_j - \lambda_i|$$

有时简记为 $\operatorname{gap}(\lambda_i)$。

特征向量的没干系那个依赖于其对应的特征值得 gap，一般来说，gap 越小，特征向量越敏感。

例　设

$$A = \begin{bmatrix} 1+g & \\ & 1 \end{bmatrix}, \quad E = \begin{bmatrix} 0 & \\ & \varepsilon \\ & & \varepsilon \end{bmatrix}, \quad (0 < \varepsilon < g)$$

则 A 的特征值为 $\lambda_1 = 1+g, \lambda_2 = 1$，对应的单位特征向量为 $q_1 = e_1 \square q_2 = e_2$。$A+E$ 的特征值为 $\hat{\lambda}_{1,2} = 1 + \left(g \pm \sqrt{g^2+4\varepsilon^2}\right)/2$，对应的单位特征向量为

$$\begin{aligned} \hat{q}_1 = \beta_1 \begin{bmatrix} 1 \\ \frac{\sqrt{1+4\varepsilon^2/g^2}-1}{2\varepsilon/g} \end{bmatrix} &= \beta_1 \begin{bmatrix} 1 \\ \frac{\sqrt{(1+2\varepsilon^2/g^2)^2-4(\varepsilon/g)^4}-1}{2\varepsilon/g} \end{bmatrix} \\ &\approx \beta_1 \begin{bmatrix} 1 \\ \frac{\left(1+2\varepsilon^2/g^2\right)-1}{2\varepsilon/g} \end{bmatrix} \\ &= \frac{1}{\sqrt{1+\varepsilon^2/g^2}} \begin{bmatrix} 1 \\ \varepsilon/g \end{bmatrix} \\ \hat{q}_2 = \beta_2 \cdot \begin{bmatrix} 1 \\ \frac{1}{2\varepsilon/g} \end{bmatrix} &\approx \frac{1}{\sqrt{1+\varepsilon^2/g^2}} \begin{bmatrix} -\varepsilon/g \\ 1 \end{bmatrix} \end{aligned}$$

其中 β_1, β_2 为规范化因子。故特征向量的扰动约为 ε/g，与特征值的间隙 $\operatorname{gap}(\lambda_i, A) = g$ 成反比。

定理　设 $A = Q\Lambda Q^T he$ 和 $A+E = \tilde{Q}\tilde{\Lambda}\tilde{Q}^{\mathrm{T}}$ 分别为对称矩阵 $A \in \mathbb{R}^{n\times n}$ 和 $A+E \in \mathbb{R}^{n\times n}$ 的特征值分解，其中 $Q = [q_1, q_2, \ldots, q_n]$ 和 $\tilde{Q} = [\tilde{q}_1, \tilde{q}_2, \ldots, \tilde{q}_n]$ 均为正交矩阵，且 $\tilde{q}_i$ 为 q_i 对应的扰动特征向量。用 θ_i 表示 q_i 和 $\tilde{q}_i$ 之间的锐角，则当 $\operatorname{gap}(\lambda_i, A) > 0$ 时

$$\frac{1}{2}\sin 2\theta_i \leq \frac{\|E\|_2}{\operatorname{gap}(\lambda_i, A)}$$

类似的，当 $\operatorname{gap}\left(\tilde{\lambda}_i, A+E\right) > 0$ 时

$$\frac{1}{2}\sin 2\theta_i \leq \frac{\|E\|_2}{\operatorname{gap}\left(\tilde{\lambda}_i, A+E\right)}$$

。

- 当 $\theta_i \ll 1$ 时，$\frac{1}{2}\sin 2\theta_i \approx \theta_i \approx \sin\theta_i$；
- 当 $\|E\|_2 \geq \frac{1}{2}\operatorname{gap}(\lambda_i, A)$ 时，定理中给出的上界就失去了实际意义；
- 在该定理中，没有对特征值进行排序；
- 在实际计算中，我们通常所知道的是 $\operatorname{gap}\left(\tilde{\lambda}_i, A+E\right)$。

6.7.4 Rayleigh 商逼近

定理　设对称矩阵 $A \in \mathbb{R}^{n\times n}$ 的特征值为 $\lambda_1, \lambda_2, \ldots, \lambda_n$。

(1) 若 $x \in \mathbb{R}^n$ 是单位向量，$\beta \in \mathbb{R}$，则

$$\min_{1\leq i\leq n} |\lambda_i - \beta| \leq \|Ax - \beta x\|_2 \tag{108}$$

(2) 给定非零向量 $x \in \mathbb{R}^n$，当 $\beta = \rho(x)$ 时，$\|Ax - \beta x\|_2$ 达到最小，即

$$\min_{\beta\in\mathbb{R}} \|Ax - \beta x\|_2 = \|Ax - \rho(x)x\|_2 \tag{109}$$

(3) 令 $r = Ax - \rho(x)x$，设 λ_i 是离 $\rho(x)$ 最近的特征值，$\operatorname{gap}' = \min_{j\neq i}|\lambda_j - \rho(x)|$，$\theta$ 是 x 和 q_i 之间的锐角，其中 q_i 是 λ_i 对应的单位特征向量，则

$$\sin\theta \leq \frac{\|r\|_2}{gap'} \quad \mathbb{H} \quad |\lambda_i - \rho(x)| \leq \frac{\|r\|_2^2}{gap'} \tag{110}$$

由108可知，在幂迭代和反迭代中可以使用残量 $\|Ax - \tilde{\lambda}x\|_2 < tol$ 作为停机准则，这里 $\tilde{\lambda}$ 是迭代过程中计算得到的近似特征值。等式109则解释了为什么用 Rayleigh 商来近似特征值。

不等式110表明 $|\lambda_i - \rho(x)|$ 的值与残量范数 $\|r\|_2$ 的平方成正比，这个结论是 Rayleigh 商迭代局部三次收敛的基础。

6.7.5 相对扰动分析

这里主要讨论 A 和 X^TAX 的特征值和特征向量之间的扰动关系，其中 X 非奇异且满足 $\left\|X^\top X - I\right\|_2 = \varepsilon$。这是因为在计算特征向量时，由于舍入误差的原因，最后得到的正交矩阵 Q 会带有误差，从而失去正交性。

定理 (相对 Weyl 定理)　设对称矩阵 A 和 X^TAX 的特征值分别为 $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$ 和 $\tilde{\lambda}_1 \geq \tilde{\lambda}_2 \geq \cdots \geq \tilde{\lambda}_n$，令 $\varepsilon = \left\|X^\top X - I\right\|_2$，则

$$\left|\tilde{\lambda}_i - \lambda_i\right| \leq \varepsilon\,|\lambda_i| \text{ 或} \frac{\left|\tilde{\lambda}_i - \lambda_i\right|}{|\lambda_i|} \leq \varepsilon \quad (\text{ if } \lambda_i \neq 0)$$

当 X 正交时，$\varepsilon = 0$，故 X^TAX 与 A 有相同的特征值。当 X 几乎正交时，ε 很小，此时 X^TAX 与 A 的特征值几乎相同。

推论　设 G 和 Y^TGX 的奇异值分别为 $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n$ 和 $\tilde{\sigma}_1 \geq \tilde{\sigma}_2 \geq \cdots \geq \tilde{\sigma}_n$，令 $\varepsilon = \max\left\{\left\|X^\top X - I\right\|_2, \left\|Y^\top Y - I\right\|_2\right\}$，则

$$|\tilde{\sigma}_i - \sigma_i| \leq \varepsilon |\sigma_i| \text{ 或} \frac{|\tilde{\sigma}_i - \sigma_i|}{|\sigma_i|} \leq \varepsilon \quad (\text{ if } \sigma_i \neq 0)$$

下面给出特征向量的相对扰动性质。

定义　设 $A \in \mathbb{R}^{n\times n}$ 的特征值为 $\lambda_1, \lambda_2, \ldots, \lambda_n$，若 $\lambda_i \neq 0$，则 λ_i 与其余特征值之间的相对间隙 (relative gap)定义为

$$\text{relgap}\left(\lambda_i, A\right) = \min_{j\neq i} \frac{|\lambda_j - \lambda_i|}{|\lambda_i|}$$

定理　设 $A \in \mathbb{R}^{n\times n}$ 和 $X^\top AX \in \mathbb{R}^{n\times n}$ 的特征值分解分别为 $A = Q\Lambda Q^\top$ 和 $X^\top AX = \tilde{Q}\tilde{\Lambda}\tilde{Q}^T$，其中 $Q = [q_1, q_2, \ldots, q_n]$ 和 $\tilde{Q} = [\tilde{q}_1, \tilde{q}_2, \ldots, \tilde{q}_n]$ 均为正交矩阵，$\Lambda = \text{diag}\left(\lambda_1, \lambda_2, \ldots, \lambda_n\right), \tilde{\Lambda} = \text{diag}\left(\tilde{\lambda}_1, \tilde{\lambda}_2, \ldots, \tilde{\lambda}_n\right)$ 且 $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n, \tilde{\lambda}_1 \geq \tilde{\lambda}_2 \geq \cdots \geq \tilde{\lambda}_n$。设 θ_i 表示 q_i 和 $\tilde{q}_i$ 之间的锐角，令 $\varepsilon_1 = \left\|I - X^{-T}X^{-1}\right\|_2, \varepsilon_2 = \|X - I\|_2$，若 $\varepsilon_1 < 1$ 且 $\text{relgap}\left(\tilde{\lambda}_i, X^\top AX\right) > 0$，则

$$\frac{1}{2} sin2\theta_i \leq \frac{\varepsilon_1}{1-\varepsilon_1} \cdot \frac{1}{\text{relgap}\left(\tilde{\lambda}_i, X^\top AX\right)} + \varepsilon_2$$

7　线性方程组的迭代解法

方法综述

- 直接法　PLU 分解, LDLT 分解, Cholesky 分解法
- 迭代法
  -经典 (定常，不动点) 迭代法:Jacobi/Gauss-Seidel, SOR, AOR 等
- -Krylov 子空间迭代法: CG, MIRES, GMRES, BiCGStab 等
- 快速算法
  -基于快速变换, 如 FFT, DCT, DST 等
  -代数多重网格法 (Algebraic multigrid)
  -快速多极子算法 (Fast multipole)

有些方法可能只是对某类方程有效, 如快速算法. 在实际应用中, 这些方法可以结合使用, 如混合 (hybrid) 算法, 预处理算法 (preconditioning) 等

本讲主要介绍定常迭代算法

更多迭代方法可参见Templates for the Solution of Linear Systems:Building Blocks for Iterative Methods,SLAM,1994

7.1 离散 Poisson 方程

在本讲中, 我们以一个典型的线性方程组为例, 逐个介绍各种迭代方法, 并比较它们之间的性能. 这个方程组就是二维 Poisson 方程经过五点差分离散后得到的线性方程组.

7.1.1 一维 Poisson 方程

考虑如下带 Dirichlet 边界条件的一维 Poisson 方程

$$\begin{cases} \frac{\mathrm{d}^2 u(x)}{\mathrm{d}x^2} = f(x), 0 < x < 1, \\ u(0) = a, u(1) = b \end{cases} \tag{6.1}$$

其中 $f(x)$ 是给定的函数,$u(x)$ 是需要计算的未知函数.
差分离散
取步长 $h = \frac{1}{n+1}$, 为节点 $x_i = ih, i = 0, 1, 2, ..., n+1$ 我们采用中心差分离散，可得 $(i = 1, 2, ..., n)$

$$-\left.\frac{d^2u(x)}{dx^2}\right|_{x_i} = \frac{2u(x_i) - u(x_{i-1}) - u(x_{i+1})}{h^2} + O\left(h^2 \cdot \left\|\frac{d^4u}{dx^4}\right\|_\infty\right)$$

将其代入 (6.1), 舍去高阶项后可得 Poisson 方程在 x_i 点的近似离散方程

$$\boxed{-u_{i-1} + 2u_i - u_{i+1} = h^2 f_i,}$$

其中 $fi = f(xi), ui$ 为 $u(xi)$ 的近似.
令 $i = 1, 2, ..., n$, 则可 0 得 n 个线性方程, 写成矩阵形式

$$T_n u = f, \tag{6.2}$$

其中

$$T_n = \begin{bmatrix} 2 & -1 & & \\ -1 & \ddots & \ddots & \\ & \ddots & \ddots & -1 \\ & & -1 & 2 \end{bmatrix}, u = \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_{n-1} \\ u_n \end{bmatrix}, f = \begin{bmatrix} f_1 + u_0 \\ f_2 \\ \vdots \\ f_{n-1} \\ f_n + u_{n+1} \end{bmatrix} \tag{6.3}$$

系数矩阵 T_n 的性质

引理　T_n 的特征值和对应的特征向量分别为

$$\lambda_k = 2 - 2cos\frac{k\pi}{n+1},$$

$$z_k = \sqrt{\frac{2}{n+1}} \cdot \left[\sin\frac{k\pi}{n+1}, \sin\frac{2k\pi}{n+1}, \ldots, \sin\frac{nk\pi}{n+1}\right]^T$$

即 $Tn = ZAZ_T$, 其中 $A = diag(\lambda_1, \lambda_2, ..., \lambda_n), Z = [z_1, z_2, ..., z_n]$.

证明. 直接带入验证即可.

引理　更一般的, 设 $T = tridiag(a, b, c) \in \mathbb{R}^{n\times n}$, 则 T 的特征值为

$$\lambda_k = b - 2\sqrt{ac}cos\frac{k\pi}{n+1}, k = 1, 2, ..., n$$

对应的特征向量为 z_k, 其第 j 个分量为

$$z_k(j) = (\frac{a}{c})^{\frac{j}{2}} sin\frac{jk\pi}{n+1}$$

特别地, 若 $a = c = 1$, 则对应的单位特征向量为

$$z_k = \sqrt{\frac{2}{n+1}} \cdot \left[\sin\frac{k\pi}{n+1}, \sin\frac{2k\pi}{n+1}, \ldots, \sin\frac{nk\pi}{n+1}\right]^T$$

由前面的结论可知,T_n 是对称正定的, 其最大特征值为

$$2\left(1 - \cos\frac{n\pi}{n+1}\right) = 4\sin^2\frac{n\pi}{2(n+1)} \approx 4,$$

最小特征值为

$$2\left(1 - \cos\frac{\pi}{n+1}\right) = 4\sin^2\frac{\pi}{2(n+1)} \approx \left(\frac{\pi}{n+1}\right)^2$$

因此, 当 n 很大时,T_n 的谱条件数约为

$$\kappa_2(T_n) \approx \frac{4(n+1)^2}{\pi^2}$$

矩阵 T_n 可以分解为 $T_n = DD^T$, 其中

$$D = \begin{bmatrix} -1 & 1 & & & \\ & -1 & 1 & & \\ & & \ddots & \ddots & \\ & & & -1 & 1 \end{bmatrix} \in \mathbb{R}^{n\times(n+1)}$$

矩阵 D 也通常称为差分矩阵. 需要注意的是,D 不是方阵, 因此不能用这个分解来求解线性方程组 $T_n x = b$.

7.1.2　二维 Poisson 方程

现在考虑二维 Poisson 方程

$$\begin{cases} -\Delta u(x,y) = -\frac{\partial^2 u(x,y)}{\partial x^2} - \frac{\partial^2 u(x,y)}{\partial y^2} = f(x,y), & (x,y) \in \Omega \\ u(x,y) = u_0(x,y), & (x,y) \in \partial\Omega \end{cases} \tag{6.4}$$

其中 $\Omega = [0,1] \times [0,1]$ 为求解区域,$\partial\Omega$ 表示 Ω 的边界.

五点差分离散

为了简单起见, 我们在 $x-$ 方向和 $y-$ 方向取相同的步长 $h=\frac{1}{n+1}$, 节点设为 $x_i=ih, y_j=jh, i,j=0,1,2,\ldots,n$. 在 $x-$ 方向和 $y-$ 方向同时采用中心差分离散可得

$$\left.\frac{\partial^2 u(x,y)}{\partial x^2}\right|_{(x_i,y_j)} \approx \frac{2u\left(x_i,y_j\right)-u\left(x_{i-1},y_j\right)-u\left(x_{i+1},y_j\right)}{h^2}$$

$$\left.\frac{\partial^2 u(x,y)}{\partial y^2}\right|_{(x_i,y_j)} \approx \frac{2u\left(x_i,y_j\right)-u\left(x_i,y_{j-1}\right)-u\left(x_i,y_{j+1}\right)}{h^2}$$

代入 (6.4), 即得二维 Poisson 方程在 (x_i,y_j) 点的近似离散方程

$$\boxed{4u_{i,j}-u_{i-1,j}-u_{i+1,j}-u_{i,j-1}-u_{i,j+1}=h^2 f_{i,j}}$$

其中 $f_{ij}=f\left(x_i,y_j\right), u_{i,j}$ 为 $u\left(x_i,y_j\right)$ 的近似。写成矩阵形式即为

$$T_N u=h^2 f \tag{6.5}$$

其中

$$T_N \triangleq I\otimes T_n+T_n\otimes I, \quad N=n^2,$$

$$u=\left[u_{1,1},\ldots,u_{n,1},u_{1,2},\ldots,u_{n,2},\ldots,u_{1,n},\ldots,u_{n,n}\right],$$

在后面介绍的算法时, 我们都以二维离散 Poisson 方程 (6.5) 为例.

系数矩阵 T 的性质

因为 $T_N=I\otimes T_n+T_n\otimes I$ 由 Kronecker 乘积的性质即得定理　设 $T_n=Z\Lambda Z^T$ 其中 $Z=[z_1,z_2,\ldots,z_n]$ 为正交阵,$\Lambda=\mathrm{diag}\left(\lambda_1\lambda_2,\ldots,\lambda_n\right)$ 为对角阵, 则 T 的特征值分解为

$$T_N=(Z\otimes Z)(I\otimes\Lambda+\Lambda\otimes I)(Z\otimes Z)^T$$

即 T 的特征值为 $\lambda_i+\lambda_j$, 对应的特征向量为 $z_i\otimes z_j, i,j=1,2,\ldots,n$ 条件数

$$\kappa\left(T_N\right)=\frac{\lambda_{\max}\left(T_N\right)}{\lambda_{\min}\left(T_N\right)}=\frac{1-\cos\frac{n\pi}{n+1}}{1-\cos\frac{\pi}{n+1}}=\frac{\sin^2\frac{n\pi}{2(n+1)}}{\sin^2\frac{\pi}{2(n+1)}}\approx\frac{4(n+1)^2}{\pi^2}$$

故当 n 越来越大时 $\kappa\left(T_N\right)\to\infty$, 即 T_N 越来越病态.

二维离散 Poisson 方程的常用算法

	方法	串行时间	存储空间
直接法	稠密 Cholesky 分解	$O(N^3)$	$O(N^2)$
	显式求逆	$O(N^2)$	$O(N^2)$
	带状 Cholesky 分解	$O(N^2)$	$O(N^(3/2))$
	稀疏 Cholesky 分解	$O(N^(3/2)))$	$O(N\log N)$
经典迭代	Jacobi	$O(N^3)$	$O(N)$
	Gauss-Seidel	$O(N^3)$	$O(N)$
	SOR	$O(N^(3/2)))$	$O(N)$
	带 Chebyshev 加速的 SSOR	$O(N^(5/4)))$	$O(N)$
Krylov 子空间迭代	CG (共轭梯度法)	$O(N^{3/2})$	$O(N)$
	CG (带修正 IC 预处理)	$O(N^(5/4)))$	$O(N)$
快速算法	FFT(快速 Fourier 变换)	$O(N\log N)$	$O(N)$
	块循环约化	$O(N\log N)$	$O(N)$
	Multigrid	$O(N)$	$O(N)$

7.2 定常迭代方法

当直接求解方程组 $Ax=b$ 较困难时, 我们可以求解一个近似方程组

$$Mx=b$$

设其解为 $x^{(1)}$. 易知它与真解之间的误差满足

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

如果 $x^{(1)}$ 已经满足精度要求, 则停止计算, 否则需要修正. 设修正量为 Δx. 显然 Δx 满足方程 $A\Delta x=b-Ax^{(1)}$ 但由于直接求解该方程比较困难, 因此我们还是求解近似

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

于是得到修正后的近似解

$$x^{(2)}=x^{(1)}+\Delta x=x^{(1)}+M^{-1}\left(b-Ax^{(1)}\right)$$

若 $x^{(2)}$ 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, 我们就得到一个序列

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

满足以下递推关系

$$x^{(k+1)}=x^{(k)}+M^{-1}\left(b-Ax^{(k)}\right),\quad k=1,2,\ldots$$

由于每次迭代的格式是一样的, 因此称为 定常迭代.
通常, 构造一个好的定常迭代, 需要考虑以下两点:

- (1) 以 M 为系数矩阵的线性方程组必须要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, 或者迭代序列 x_k 要收敛

下面我们就介绍几个常见的基于矩阵分裂的定常迭代方法

- Jacobi 算法
- Gauss-Seidel 算法
- SOR(Successive Over-Relaxation) 算法
- SSOR(Symmetric SOR) 算法
- AOR(Accelerated over-relaxation) 算法

7.2.1 矩阵分裂迭代方法

迭代方法的基本思想: 给定一个迭代初始值 $x_{(0)}$, 通过一定的迭代格式生成一个迭代序列 $\left\{x^{(k)}\right\}_{k=0}^{\infty}$, 使得 $\lim_{k\to\infty} x^{(k)} = x_* \triangleq A^{-1}b$

定义(矩阵分裂 Matrix splitting)

设 $A \in \mathbb{R}^{n\times n}$ 非奇异, 称

$$A = M - N$$

为 A 的一个矩阵分裂, 其中 M 非奇异
原方程组等价于 $Mx = Nx + b$. 于是我们就可以构造出以下的迭代格式

$$x^{(k+1)} = M^{-1}Nx^{(k)} + M^{-1}b \triangleq Gx^{(k)} + g \quad , \quad k = 0, 1, \ldots, \tag{6.7}$$

其中 $G = M^{-1}N$ 称为该迭代格式的迭代矩阵

7.2.2 Jacobi 迭代

将矩阵 A 分裂为

$$A = D - L - U$$

其中 D 为 A 的对角部分,$-L$ 和 $-U$ 分别为 A 的严格下三角和严格上的三角部分
在矩阵分裂 $A = M - N$ 中取 $M = D, N = L + U$,则可得到Jacobi 迭代算法:

$$x^{(k+1)} = D^{-1}(L + U)x^{(k)} + D^{-1}b \quad , \quad k = 0, 1, 2, \ldots \tag{6.8}$$

迭代矩阵为

$$G_1 = D^{-1}(L + U)$$

写成分量形式即为

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

由于 Jacobi 迭代中 $x_i^{(k+1)}$ 的更新顺序与 i 无关，即可以按照顺序 $i=1,2,...,n$ 计算。因此 Jacobi 迭代非常适合并行计算。

算法 2.1求解线性方程组的 Jacobi 迭代方法

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

我们也可以将 Jacobi 迭代格式写成

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

其中 $r_k \triangleq b-Ax^{(k)}$ 是 k 次迭代后的残量。

二维离散 poisson 方程 Jacobi 迭代方法

算法 2.2求解二维离散 poisson 方程的 Jacobi 迭代方法

1：Choose an initial guess $v^{(0)}$
2: while not converge do
3: for $i=1$ to N do
4: for $j=1$ to N do
5: $x_i^{(k+1)}=\left(b_i-\sum_{j=1,j\neq i}^{n}a_{ij}x_j^{(k)}\right)/a_{ii}$
6: end for
7: end for
8:end while

7.2.3 Gauss-Seidel 迭代

取 $M=D-L, N=U$, 即可得 colorblueGauss-Seidel (G-S) 迭代算法：

$$x^{(k+1)}=(D-L)^{-1}Ux^{(k)}+(D-L)^{-1}b \tag{6.9}$$

迭代矩阵为

$$G_{GS}=(D-L)^{-1}U$$

将 $G-S$ 迭代改写为

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

即可得分量形式

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

算法 2.1求解线性方程组的 Jacobi 迭代方法

1：Choose an initial guess $x^{(0)}$

2: while not converge do

3: for $i = 1$ to n do

4: $x_i^{(k+1)} = \frac{1}{a_{ii}}\left(b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij}x_j^{(k)}\right)$

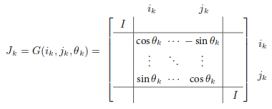
5: end for

6:end while

$G-S$ 算法的主要优点是充分利用了已经获得的最新数据。
但由于 $G-S$ 算法中未知量的更新是按自然顺序进行的。因此不适合并行计算。

G-S 算法的并行计算：红黑排序

下面我们介绍一种适合并行计算的更新顺序：红黑排序，即将二维网络点依次做红黑记号，如右图
在计算过程中，对未知量的值进行更新时，我们可以先更新红色节点，此时所使用的只是黑色节点的数据，然后再更新黑色节点，这时使用的是红色节点的数据. 于是我们得到红黑排序 G-S 迭代方法.



下面我们介绍一种适合并行计算的更新顺序：红黑排序，即将二维网络点依次做红黑记号，如右图
在计算过程中，对未知量的值进行更新时，我们可以先更新红色节点，此时所使用的只是黑色节点的数据，然后再更新黑色节点，这时使用的是红色节点的数据. 于是我们得到红黑排序 G-S 迭代方法.

$$J_k = G(i_k, j_k, \theta_k) = \begin{bmatrix} & i_k & & j_k & \\ I & & & & \\ & \cos\theta_k & \cdots & -\sin\theta_k & & i_k \\ & \vdots & \ddots & \vdots & & \\ & \sin\theta_k & \cdots & \cos\theta_k & & j_k \\ & & & & I & \end{bmatrix}$$

图 4

算法 2.4求解二维离散 poisson 方程的红黑排序 G-S 迭代方法

1：Choose an initial guess $v^{(0)}$

2: while not converge do

3: for (i,j) 为红色节点 do

4: $u_{i,j}^{(k+1)} = \frac{1}{4}\left(h^2 f_{i,j} + u_{i+1,j}^{(k)} + u_{i-1,j}^{(k)} + u_{i,j+1}^{(k)} + u_{i,j-1}^{(k)}\right)$

5: end for

6: for (i,j) 为黑色节点 do

7: $u_{i,j}^{(k+1)} = \frac{1}{4}\left(h^2 f_{i,j} + u_{i+1,j}^{(k+1)} + u_{i-1,j}^{(k+1)} + u_{i,j+1}^{(k+1)} + u_{i,j-1}^{(k+1)}\right)$

8: end for

9:end while

7.2.4 SOR 迭代

在 G-S 算法的基础上, 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) 算法, 即将 G-S 算法中的第 $k+1$ 步近似解与第 k 步近似解做一个加权平均:

$$x^{(k+1)} = (1-\omega)x^{(k)} + \omega\left(D^{-1}\left(Lx^{(k+1)} + Ux^{(k)}\right) + D^{-1}b\right) \tag{6.10}$$

整理后即为

$$x^{(k+1)} = (D-\omega L)^{-1}((1-\omega)D + \omega U)x^{(k)} + \omega(D-\omega L)^{-1}b \tag{6.11}$$

其中 ω 称为松弛参数。
当 $\omega = 1$ 时，SOR 即为 G-S 算法，当 $\omega <$ 时，称为低松弛 (under relaxation)算法，当 $\omega > 1$ 时，称为超松弛 (over relaxation)算法.
SOR 算法曾经在很长一段时间内是科学计算中求解线性方程组的首选方法。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

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

对应的矩阵分裂为

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

由 (6.11) 可得 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 2.5求解线性方程组的 SOR 迭代方法

1:Choose an initial guess $x^{(0)}$

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 算法最大的优点是引入了松弛参数 ω, 通过选取适当的 ω 可以大大提高算法的收敛速度.
但是 SOR 算法最大的难点就是如何选取最优的参数

算法 2.4求解二维离散 poisson 方程的红黑排序 G-S 迭代方法

1:Choose an initial guess $v^{(0)}$

2: while not converge do

3: for (i,j) 为红色节点 do

4: $u_{i,j}^{(k+1)} = (1-\omega)v_{i,j}^{(k)} + \omega\left(h^2 f_{i,j} + u_{i+1,j}^{(k)} + u_{i-1,j}^{(k)} + u_{i,j+1}^{(k)} + u_{i,j-1}^{(k)}\right)/4$

5: end for

6: for (i,j) 为黑色节点 do

7: $u_{i,j}^{(k+1)} = (1-\omega)v_{i,j}^{(k)} + \omega\left(h^2 f_{i,j} + u_{i+1,j}^{(k+1)} + u_{i-1,j}^{(k+1)} + u_{i,j+1}^{(k+1)} + u_{i,j-1}^{(k+1)}\right)/4$

8: end for

9:end while

7.2.5 SSOR 迭代方法

将 SOR 算法中的 L 和 U 相交换, 即可得迭代格式

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

将这个迭代格式与 SOR 相结合, 就可以得到下面的两步迭代方法

$$\begin{cases} x^{\left(k+\frac{1}{2}\right)} = (D-\omega L)^{-1}[(1-\omega)D + \omega U]x^{(k)} + \omega(D-\omega L)^{-1}b \\ x^{(k+1)} = (D-\omega U)^{-1}[(1-\omega)D + \omega L]x^{\left(k+\frac{1}{2}\right)} + \omega(D-\omega U)^{-1}b \end{cases}$$

这就是SSOR 迭代(对称超松弛) 算法, 相当于将 L 与 U 同等看待, 交替做两次 SOR 迭代.
消去中间迭代量 $x^{\left(k+\frac{1}{2}\right)}$, 可得

$$x^{(k+1)} = G_{\text{SSOR}}x^{(k)} + g$$

其中迭代矩阵

$$G_{\text{SSOR}} = (D-\omega U)^{-1}[(1-\omega)D + \omega L](D-\omega L)^{-1}[(1-\omega)D + \omega U]$$

对应的矩阵分裂为

$$\begin{aligned} M &= \frac{1}{\omega(2-\omega)}\left[D-\omega(L+U)+\omega^2 LD^{-1}U\right] \\ &= \frac{1}{\omega(2-\omega)}(D-\omega L)D^{-1}(D-\omega U) \\ N &= \frac{1}{\omega(2-\omega)}[(1-\omega)D+\omega L]D^{-1}[(1-\omega)D+\omega U] \end{aligned}$$

对于某些特殊问题, SOR 算法不收敛, 但仍然可能构造出收敛的 SSOR 算法.
一般来说, SOR 算法的渐进收敛速度对参数 ω 比较敏感, 但 SSOR 对参数 ω 不太敏感. (Poisson SOR omega.m,Poisson SSOR omega.m))

7.2.6 AOR 迭代

Hadjidimos 于 1978 年提出了 AOR (Accelerated over-relaxation, 快速松弛) 算法, 迭代矩阵为

$$G_{\text{AOR}} = (D-\gamma L)^{-1}[(1-\omega)D+(\omega-\gamma)L+\omega U]$$

其中 γ 和 ω 为松弛参数. 对应的矩阵分解为

$$M = \frac{1}{\omega}(D-\gamma L), \quad N = \frac{1}{\omega}[(1-\omega)D+(\omega-\gamma)L+\omega U]$$

(1) 当 $\gamma=\omega$ 时, AOR 算法即为 SOR 算法;

(2) 当 $\gamma=\omega=1$ 时, AOR 算法即为 G-S 算法;

(3) 当 $\gamma=0,\omega=1$ 时, AOR 算法即为 Jacobi 算法

与 SSOR 类似, 我们也可以定义 SAOR 算法.

7.2.7 Richardson 算法

Richardson 算法是一类形式非常简单的算法, 其迭代格式为

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

对应的矩阵分裂和迭代矩阵分别为

$$M = \frac{1}{\omega}I, \quad N = \frac{1}{\omega}I - A, \quad G_{\text{R}} = I-\omega A$$

如果在每次迭代时取不同的参数, 即

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

则称为 nonstationary Richardson 算法. 定理设 $A\in\mathbb{R}^{n\times n}$ 是对称正定矩阵,λ_1 和 λ_n 分别是 A 的最大和最小特征值, 则 Richardson 算法收敛当且仅当

$$0<\omega<\frac{1}{\lambda_1}$$

最优参数为

$$\omega_* = \arg\min_{\omega} \rho\left(G_{\mathrm{R}}\right) = \frac{2}{\lambda_1 + \lambda_n}$$

即当 $\omega = \omega_*$ 时, 迭代矩阵的谱半径达到最小, 且有

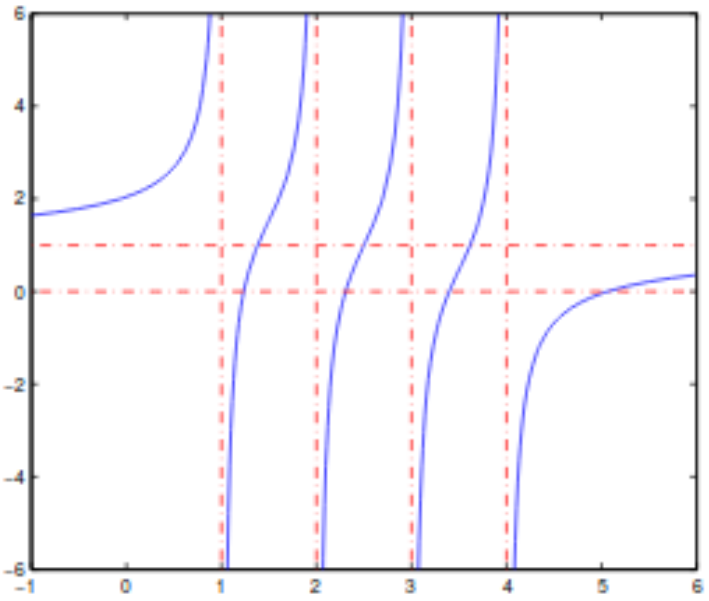
$$\rho\left(G_{\mathrm{R}}\right) = \begin{cases} 1 - \omega\lambda_n & \text{if } \omega \leq \omega \\ \frac{\lambda_1 - \lambda_n}{\lambda_1 + \lambda_n} = \frac{\kappa(A)-1}{\kappa(A)+1} & \text{if } \omega = \omega \\ \omega\lambda_1 - 1 & \text{if } \omega \geq \omega \end{cases}$$

7.2.8 分块迭代方法

前面介绍的迭代方法可以推广到分块情形. 将 A 写成如下的分块形式

$$A = \begin{bmatrix} A_{11} & A_{12} & \cdots & A_{1p} \\ A_{21} & A_{22} & \cdots & A_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ A_{p1} & A_{p2} & \cdots & A_{PP} \end{bmatrix}$$

设 $A = D - L - U$, 其中 $D, -L, -U$ 分别是 A 的快对角,块严格下三角矩阵和块严格上三角矩阵。则相应的分块 Jacobi, 分块 Gauss-Seidel 和分块 SOR 算法分别为



- 分块 Jacobi 迭代

$$A_{ii}\boldsymbol{x}_i^{(k+1)} = \boldsymbol{b}_i - \sum_{j=1, j\neq i}^{p} A_{ij}\boldsymbol{x}_j^{(k)}, \quad i = 1, 2, \ldots, p$$

- 分块 Gauss-seidel 迭代

$$A_{ii}\boldsymbol{x}_i^{(k+1)} = \boldsymbol{b}_i - \sum_{j=1}^{i-1} A_{ij}\boldsymbol{x}_j^{(k+1)} - \sum_{j=i+1}^{p} A_{ij}\boldsymbol{x}_j^{(k)}, \quad i = 1, 2, \ldots, p$$

- 分块 SOR 迭代

$$\boldsymbol{x}_i^{(k+1)} = (1-\omega)\boldsymbol{x}_i^{(k)} + \omega A_{ii}^{-1}\left(\boldsymbol{b}_i - \sum_{j=1}^{i-1} A_{ij}\boldsymbol{x}_j^{(k+1)} - \sum_{j=i+1}^{p} A_{ij}\boldsymbol{x}_j^{(k)}\right)$$
$$i = 1, 2, \ldots, p$$

7.3 收敛性分析

7.3.1 定常迭代方法的收敛性

定义（迭代方法的收敛性）如果对于任意的初始向量 $x^{(0)}$, 都有

$$\lim_{k\to\infty} x^{(k)} \to x_*$$

则称迭代格式6.7是收敛的, 否则就称其为发散的。
基于矩阵分裂的迭代方法, 其收敛性取决于迭代矩阵的谱半径.

矩阵谱半径

设 $A \in \mathbb{R}^{n\times n}$, 则称

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

为 A 的谱半径, 其中 $\sigma(A)$ 表示 A 的所有特征值组成的集合。
谱半径与矩阵范数之间有如下的关系.
引理 (谱半径与范数的关系)设 $G \in \mathbb{R}^{n\times n}$, 则

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

由此, 我们可以立即得到下面的结论.
定理设矩阵 $G \in \mathbb{R}^{n\times n}$, 则 $\lim_{k\to\infty} G^k = 0$ 当且仅当 $\rho(G) < 1$. 下面的结论是谱半径与算子范数之间的一个非常重要的性质。
引理设 $G \in \mathbb{R}^{n\times n}$, 则对任意算子范数 $\|\cdot\|$, 有

$$\rho(G) = \lim_{k\to\infty} \left\|G^k\right\|^{\frac{1}{k}}$$

迭代方法收敛性判断

首先给出一个迭代方法收敛的充分条件
引理若存在算子范数 $\|\cdot\|$, 使得 $|G\| < 1$, 则迭代方法 6.7 收敛.
我们记 $e^{(k)} \triangleq x^{(k)} - x$ 为第 k 步迭代解 $x^{(k)}$ 的误差向量.
定理 (收敛性定理)对任意迭代初始向量 $x^{(0)}$, 迭代方法 6.7 收敛的充要条件是 $\rho(G) < 1$.
定义设 G 是迭代矩阵, 则迭代方法 6.7 的平均收敛速度定义为

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

, 渐进收敛速度定义为

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

平均收敛速度与迭代步数和所用的范数有关, 但渐进收敛速度只依赖于迭代矩阵的谱半径.
定理考虑算法 6.7. 如果存在某个算子范数 $\|\cdot\|$ 使得 $\|G\| = q < 1$, 则

- (1)$\left\|x^{(k)} - x_*\right\| \leq q^k \left\|x^{(0)} - x_*\right\|$
- (2)$\left\|x^{(k)} - x_+\right\| \leq \frac{q}{1-q} \left\|x^{(k)} - x^{(k-1)}\right\|$
- (3)$\left\|x^{(k)} - x_*\right\| \leq \frac{q^k}{1-q} \left\|x^{(1)} - x^{(0)}\right\|$

一般来说, 好的迭代方法应该满

- (1)$\rho(G)$ 很小;
- (2) 以 M 为系数矩阵的线性方程组比较容易求解

7.3.2 二维离散 Poisson 方程情形

充要条件:迭代矩阵的谱半径小于 1.
充分条件:迭代矩阵的某个算子范数小于 1.
对于二维离散 Poisson 方程, 系数矩阵为

$$A = T = I \otimes T_n + T_n \otimes I$$

故 Jacobi 算法的迭代矩阵为

$$G_{\mathrm{J}} = D^{-1}(L+U) = (4I)^{-1}(4I - T) = I - \frac{1}{4}T \tag{6.12}$$

由于 T 的特征值为

$$\lambda_i + \lambda_j = 2\left(1 - \cos\frac{\pi i}{n+1}\right) + 2\left(1 - \cos\frac{\pi j}{n+1}\right)$$

所以 G_{J} 的特征值为

$$1-\frac{1}{4}\left(\lambda_i+\lambda_j\right)=\frac{1}{2}\left(\cos\frac{\pi i}{n+1}+\cos\frac{\pi j}{n+1}\right)$$

故

$$\rho\left(G_{\mathrm{J}}\right)=\frac{1}{2}\max_{i,j}\left\{\left|\cos\frac{\pi i}{n+1}+\cos\frac{\pi j}{n+1}\right|\right\}=\cos\frac{\pi}{n+1}<1$$

即 Jacobi 算法是收敛的. 注意当 n 越来越大时,$\kappa(T)\to\infty$, 即 T 越来越病态, 此时 $\rho\left(G_1\right)\to 1$, 即 Jacobi 算法收敛越来越慢.
问题越病态可能就越难求解

G-S 算法和 SOR 算法

性质设 G_{GS} 和 G_{SOR} 分别表示求解二维 Poisson 方程的红黑排序的 G-S 算法和 SOR 算法的迭代矩阵, 则有

$$\rho\left(G_{\mathrm{GS}}\right)=\rho\left(G_{\mathrm{J}}\right)^2=\cos^2\frac{\pi}{n+1}<1 \tag{6.13}$$

$$\rho\left(G_{\mathrm{SOR}}\right)=\frac{\cos^2\frac{\pi}{n+1}}{\left(1+\sin\frac{\pi}{n+1}\right)^2}<1,\quad \omega=\frac{2}{1+\sin\frac{\pi}{n+1}} \tag{6.14}$$

在上述结论中, SOR 算法中的 ω 是最优参数, 即此时的 $\rho\left(G_{\mathrm{SOR}}\right)$ 最小.

SOR 与 Jacobi

由 Taylor 公式可知, 当 n 很大时, 有

$$\rho\left(G_{\mathrm{J}}\right)=\cos\frac{\pi}{n+1}\approx 1-\frac{\pi^2}{2(n+1)^2}=1-O\left(\frac{1}{n^2}\right)$$

$$\rho\left(G_{\mathrm{SOR}}\right)=\frac{\cos^2\frac{\pi}{n+1}}{\left(1+\sin\frac{\pi}{n+1}\right)^2}\approx 1-\frac{2\pi}{n+1}=1-O\left(\frac{1}{n}\right)$$

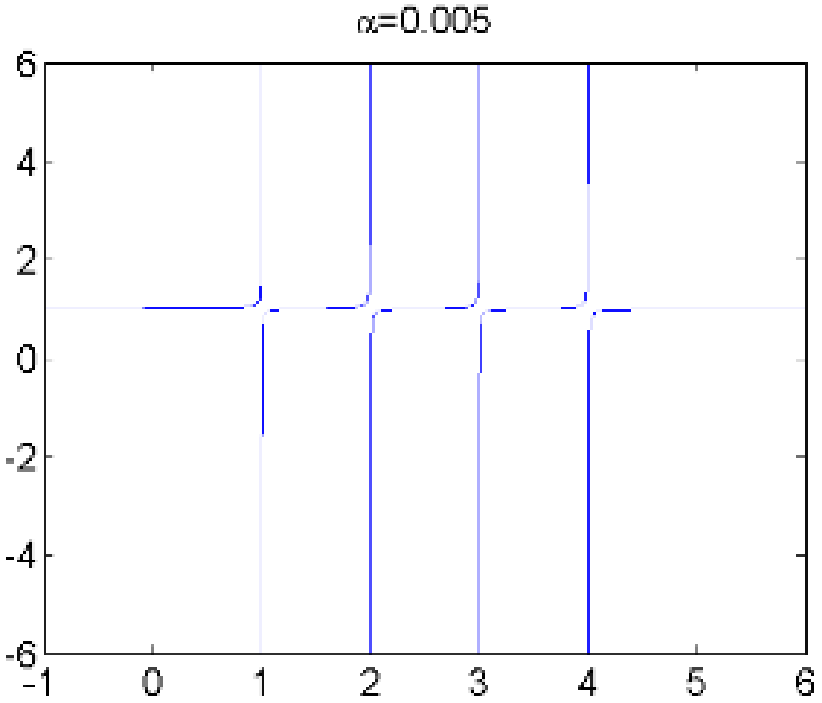
由于当 n 很大时有

$$\left(1-\frac{1}{n}\right)^k\approx 1-\frac{k}{n}=1-\frac{kn}{n^2}\approx\left(1-\frac{1}{n^2}\right)^{kn}$$

即 SOR 迭代 k 步后误差的减小量与 Jacobi 迭代 kn 步后误差减小量差不多. 因此, 取最优参数的 SOR 算法的收敛速度大约是 $Jacobi$ 算法的 n 倍.
事实上, 当 n 很大时, 这三个算法的收敛速度都很慢.
例 已知二维 Poisson 方程

$$\begin{cases}-\Delta u(x,y)=-1, & (x,y)\in\Omega\\ u(x,y)=\frac{x^2+y^2}{4}, & (x,y)\in\partial\Omega\end{cases}$$

其中 $\Omega=(0,1)\times(0,1)$. 该方程的解析解是 $u(x,y)=\frac{x^2+y^2}{4}$. 用五点差分格式离散后得到一个线性方程组, 分别用 Jacobi, G-S 和 SOR 算法计算这个方程组的解, 并比较收敛效果



7.3.3 不可约对角占优

这里我们考虑 A 是严格对角占优或不可约弱对角占优情形. 定理　设 $A \in \mathbb{R}^{n\times n}$,若 A 严格对角占优, 则 Jacobi 算法和 G-S 算法都收敛, 且

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

定理　设 $A \in \mathbb{R}^{n\times n}$ 若 A 是弱对角占优且不可约, 则 Jacobi 算法和 G-S 算法都收敛, 且 $\rho(G_{\mathrm{GS}}) < \rho(G_{\mathrm{J}}) < 1$ 二维离散 Poisson 方程是弱行对角占优且不可约, 故对 Jacobi 算法和 G-S 算法都收敛.
上述定理中的结论对一般矩阵并不成立:对某些矩阵, Jacobi 算法收敛, 但 G-S 算法却不一定收敛.
关于 SOR 方法, 我们有下面的结论
设 $A \in \mathbb{R}^{n\times n}$,若 A 严格对角占优且 $0 < \omega \leq 1$, 则 SOR 方法收敛.
定理　设 $A \in \mathbb{R}^{n\times n}$ 若 A 是弱对角占优且不可约, 且 $0 < \omega \leq 1$ 则 SOR 方法收敛.

7.3.4 对称正定矩阵

在给出收敛性结论之前, 也介绍两个需要用到的引理.
引理　设 $A \in \mathbb{C}^{n\times n}$,Hermite 对称, 且 $A = M - N$ 是 A 的一个矩阵分裂, 则 $M^* + N$ 也是 Hermite 对称, 且对任意 $x \in \mathbb{C}^n$ 有

$$x^*Ax - \tilde{x}^*A\tilde{x} = u^*\left(M^* + N\right)u$$

其中 $\tilde{x} = M^{-1}Nx, u = x - \tilde{x}$ 定理　设 $A \in \mathbb{R}^{n\times n}$ 对称,且 $A = M - N$ 是 A 的一个矩阵分裂.

- (1) 如果 A 和 $M^T + N$ 都是正定矩阵, 则 M 非奇异且 $\rho(M^{-1}N) < 1$

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

我们首先给出 SOR 迭代收敛的一个必要条件. 定理对于 SOR 算法, 有 $\rho(G_{\text{SOR}})\geq|1-\omega|$, 故 SOR 算法收敛的必要条件是 $0<\omega<2$. 定理　设 $A\in\mathbb{R}^{n\times n}$ 对称正定.

- (1) 若 $2D$ 正定且 Jacobi 迭代收敛, 则 A 正定
- (2) 若 D 正定, 且存在 $\omega\in(0,2)$ 使得 SOR (或 SSOR) 收敛, 则 A 正定
- (3) 若 D 正定, 且 G-S 迭代收敛, 则 A 正定.

7.3.5 相容次序矩阵

针对一类特殊的矩阵, 这三种迭代方法的谱半径之间存在一种特殊关系.
定义设 $A\in\mathbb{R}^{n\times n}$, 如果存在一个置换矩阵 P, 使得

$$PAP^T=\begin{bmatrix}D_1 & F\\ E & D_2\end{bmatrix} \tag{6.15}$$

其中 D_1,D_2 为对角矩阵, 则称 A 具有性质 A. 例　对于二维离散 Poisson 方程, 系数矩阵 T_{N^2} 具有性质 A. 事实上, 设 $\tilde{T}_{N^2}$ 为模型问题采用红黑排序后的系数矩阵, 则 $\tilde{T}_{N^2}$ 具有 (6.15) 的结构.
我们首先给出一个性质.
引理　设 $B\in\mathbb{R}^{n\times n}$ 具有下面的结构

$$B=\begin{bmatrix}0 & B_{12}\\ B_{21} & 0\end{bmatrix}$$

令 B_L 和 B_U 分别表示 B 的下三角和上三角部分, 则

- (1) 若 μ 是 B 的特征值, 则 $-\mu$ 也是 B 的特征值;
- (2)$B(\alpha)$ 的特征值与无关, 其中

$$B(\alpha)=\alpha B_L+\frac{1}{\alpha}B_U,\qquad \alpha\neq 0$$

$B(\alpha)+\beta I$ 的特征值也与 α 无关, 其中 β 为任意常数.
该结论可以推广到块三对角形式, 见习题. 设 $A\in\mathbb{R}^{n\times n}$ 的对角线元素全不为零, 记 $\tilde{L}=D^{-1}L,\tilde{U}=D^{-1}U$.
定义设 $A\in\mathbb{R}^{n\times n}$ 的对角线元素全不为零,$A=D(I-\tilde{L}-\tilde{U})$ 若矩阵 $G(\alpha)=\alpha\tilde{L}+\frac{1}{\alpha}\tilde{U}$ 的特征值与 α 无关, 则称 A 具有相容次序.
设 A 的对角线元素全不为零, 若 A 具有性质 A, 则存在置换矩阵 P, 使得 PAP^T 具有相容次序.
定理　设 A 有相容次序且 $\omega\neq 0$, 则下列命题成立

- (1)Jacobi 迭代矩阵 G_1 的特征值正负成对出现;

- (2) 若 μ 是 G_{J} 的特征值且 λ 满足

$$(\lambda+\omega-1)^2=\lambda\omega^2\mu^2 \tag{6.16}$$

则 λ 是 SOR 迭代矩阵 G_{SOR} 的一个特征值;

- (3) 反之, 若 $\lambda\neq 0$ 是 G_{SOR} 的一个特征值且 μ 满足 (6.16), 则 μ 是 G_{J} 的一个特征值.

推论　若 A 具有相容次序, 则 $\rho\left(G_{\mathrm{GS}}\right)=\rho\left(G_{\mathrm{J}}\right)^2$, 即当 Jacobi 算法收敛时,G-S 算法比 Jacobi 算法快一倍例　　采用红黑排序的二维离散 Poisson 方程, 系数矩阵 $\tilde{T}_{N^2}$ 具有相容次序, 故有 $\rho\left(G_{\mathrm{GS}}\right)=\rho\left(G_{\mathrm{J}}\right)^2$.
下面是关于 SOR 算法的最优参数选取.
定理　　设 A 具有相容次序,G_{l} 的特征值全部为实数, 且 $\rho_J=\rho\left(G_{\mathrm{J}}\right)<1$, 则 SOR 算法的最优参数为

$$\omega_{opt}=\frac{2}{1+\sqrt{1-\rho_J^2}}$$

此时

$$\rho\left(G_{\mathrm{SOR}}\right)=\omega_{opt}-1=\frac{\rho_J^2}{\left(1+\sqrt{1-\rho_J^2}\right)^2}$$

进一步, 有

$$\rho\left(G_{\mathrm{SOR}}\right)=\begin{cases}\omega-1, & \omega_{opt}\leq\omega\leq 2\\ 1-\omega+\frac{1}{2}\omega^2\rho_J^2+\omega\rho_J\sqrt{1-\omega+\frac{1}{4}\omega^2\rho_J^2}, & 0<\omega\leq\omega_{opt}\end{cases}$$

例　　采用红黑排序的二维离散 Poisson 问题的系数矩阵 $\tilde{T}_{N^2}$ 具有相容次序, 且 G_{l} 是对称的, 即 G_{l} 的特征值都是实的. 又由系数矩阵的弱对角占优和不可约性质可知 $\rho\left(G_1\right)<1$, 故上述定理的条件均满足.

7.4　加速算法

当迭代解 $x^{(0)},x^{(1)},x^{(2)},\ldots,x^{(k)}$ 已经计算出来后, 我们可以对其进行组合, 得到一个新的近似解, 这样就可以对原算法进行加速

7.4.1　外推技术

设原迭代格式为

$$x^{(k+1)}=Gx^{(k)}+b \tag{6.17}$$

由 $x^{(k)}$ 和 $x^{(k+1)}$ 加权组合后可得新的近似解

$$x^{(k+1)}=(1-\omega)x^{(k)}+\omega\left(Gx^{(k)}+b\right) \tag{6.18}$$

其中 ω 是参数. 这种加速方法就称为外推算法.
为了使得迭代格式 (6.18) 尽可能快地收敛, 需要选择 ω 使得其迭代矩阵 $G_\omega\triangleq(1-$

$\omega)I+\omega G$ 的谱半径尽可能地小. 假设 G 的特征值都是实数, 且最大特征值和最小特征值分别为 λ_1 和 λ_n. 于是

$$\rho\left(G_\omega\right)=\max_{\lambda\in\sigma(G)}\left|(1-\omega)+\omega\lambda\right|=\max\left\{\left|1-\omega+\omega\lambda_1\right|,\left|1-\omega+\omega\lambda_n\right|\right\}$$

定理　设 G 的特征值都是实数, 其最大和最小特征值分别为 λ_1 和 λ_n, 且 $1\notin[\lambda_n,\lambda_1]$, 则

$$\omega_*=\arg\min_\omega\rho\left(G_\omega\right)=\frac{2}{2-\left(\lambda_1+\lambda_n\right)}$$

此时

$$\rho\left(G_{\omega_*}\right)=1-\left|\omega_*\right|d$$

其中 d 是 1 到 $[\lambda_n,\lambda_1]$ 的距离, 即当 $\lambda_n\leq\lambda_1<1$ 时,$d=\lambda_n-1$.
由定理可知,$\rho\left(G_{\omega_*}\right)=1-\left|\omega_*\right|d$ 且当 $\omega_*\neq1$ 时, 外推迭代 (6.18) 比原迭代方法收敛要更快一些.
最优参数依赖于原迭代矩阵 G 的特征值, 因此实用性不强. 在实际应用时可以估计特征值所在的区间 $[a,b]$, 然后用 a,b 代替 λ_n 和 λ_1.

JOR 算法

对 $Jacobi$ 迭代进行外推加速, 则可得 $JOR(Jacobiover-relaxation)$ 算法:

$$\begin{aligned}x^{(k+1)}&=(1-\omega)x^{(k)}+\omega\left(D^{-1}(L+U)x^{(k)}+D^{-1}b\right)\\&=x^{(k)}+\omega D^{-1}\left(b-Ax^{(k)}\right),\quad k=0,1,2,\ldots\end{aligned}$$

定理　设 A 对称正定. 若

$$0<\omega<\frac{2}{\rho\left(D^{-1}A\right)}$$

则 JOR 算法收敛.

7.5 Chebyshev 加速

本节对外推技巧进行推广.
假定通过迭代格式 (6.17) 已经计算出 $x^{(0)},x^{(1)},\ldots,x^{(k)}$, 下面考虑如何将这些近似解进行组合, 以便得到更精确的近似解.
记 $\varepsilon_k=x^{(k)}-x_*$ 为第 k 步迭代解的误差, 则有

$$\varepsilon_k=G\varepsilon_{k-1}=G^2\varepsilon_{k-2}=\cdots=G^k\varepsilon_0$$

设 $\tilde{x}^{(k)}$ 为 $x^{(0)},x^{(1)},\ldots,x^{(k)}$ 的一个线性组合, 即

$$\tilde{x}^{(k)}=\alpha_0x^{(0)}+\alpha_1x^{(1)}+\cdots+\alpha_kx^{(k)}\tag{6.19}$$

其中 α_i 为待定系数, 且满足 $\sum_{i=0}^k\alpha_i=1$. 于是

$$\tilde{x}^{(k)}-x_*=\alpha_0\varepsilon_0+\alpha_1G\varepsilon_0+\cdots+\alpha_kG^k\varepsilon_0\triangleq p_k(G)\varepsilon_0\tag{6.20}$$

其中 $p_k(t)=\sum_{i=0}^{k}\alpha_i t^i$ 为 k 次多项式, 且满足 $p_k(1)=1$.
我们希望通过适当选取参数 α_i, 使得 $\tilde{x}^{(k)}-x_*$ 尽可能地小, 即使得 $\tilde{x}^{(k)}$ 收敛到 x_* 速度远远快于 $x^{(k)}$ 收敛到 x 速度. 这种加速方法就称为多项式加速或半迭代方法 (semi-iterative method)
例　　设 $p_n(t)$ 为 G 的特征多项式, 则 $p_n(G)=0$, 所以选取 α_i 为 p_n 的系数, 则 $\tilde{x}^{(n)}-x_*=0$. 但这种选取方法不实用, 原因是:

- (1)$p_n(t)$ 的系数并不知道;
- (2) 我们通常希望收敛所需的迭代步数 $\ll n$.

下面讨论参数 α_i 的较实用的选取方法. 由 (6.20) 可知

$$\left\|\tilde{x}^{(k)}-x_*\right\|_2=\|p_k(G)\varepsilon_0\|_2\leq\|p_k(G)\|_2\cdot\|\varepsilon_0\|_2$$

因此我们需要求解下面的极小化问题

$$\min_{p\in\mathbb{P}_k,p(1)=1}\|p(G)\|_2 \tag{6.21}$$

其中 $\mathbb{P}_k$ 表示所有次数不超过 k 的多项式组成的集合.
一般来说, 这个问题是非常困难的.
但在一些特殊情况下, 我们可以给出其最优解.
假设迭代矩阵 G 是对称的, 即 G 存在特征值分解 $Schur$ 标准型为

$$G=U\Lambda U^*$$

其中 Λ 是实对角矩阵,U 是酉矩阵. 于是有

$$\begin{aligned}\min_{p\in\mathbb{P}_{k,P}(1)=1}\|p(G)\|_2&=\min\nolimits^{\min}_{p\in\mathbb{P}_{k,p}(1)=1}\|p(\Lambda)\|_2\\&=\min_{p\in\mathbb{P}_{k,p}(1)=1}\max_{1\leq i\leq n}\{|p(\lambda_i)|\}\\&\leq\min_{p\in\mathbb{P}_{k,p}(1)=1}\max_{\lambda\in[\lambda_n,\lambda_1]}\{|p(\lambda)|\}\end{aligned} \tag{6.22}$$

其中 λ_1,λ_n 分别表示 G 的最大和最小特征值.
这是带归一化条件的多项式最佳一致逼近问题 (与零的偏差最小).
该问题的解与著名的Chebyshev 多项式有关.

Chebyshev 多项式

Chebyshev 多项式 $T_k(t)$ 可以通过下面的递归方式来定义:

$$\begin{aligned}&T_0(t)=1,\quad T_1(t)=t\\&T_k(t)=2tT_{k-1}(t)-T_{k-2}(t),k=2,3,\ldots\end{aligned} \tag{6.23}$$

也可以直接由下面的式子定义

$$T_k(t)=\frac{1}{2}\left[\left(t+\sqrt{t^2-1}\right)^k+\left(t+\sqrt{t^2-1}\right)^{-k}\right]$$

或者

$$T_k(t)=\begin{cases}\cos(k\ \arccos t\ ), & |t|\le 1\\ \cosh(k\,\operatorname{arccosh} t), & |t|>1\end{cases}$$

Chebyshev 的一个重要性质是下面的最小最大性质.
定理设 $\eta\in\mathbb{R}$ 满足 $|\eta|>1$, 则下面的最小最大问题

$$\min_{p(t)\in\mathbb{P}_k,p(\eta)=1}\max_{-1\le t\le 1}|p(t)|$$

的唯一解为

$$\tilde{T}_k(t)\triangleq\frac{T_k(t)}{T_k(\eta)}$$

通过简单的仿射变换, 该定理的结论可以推广到一般区间. 定理　　设 $\alpha,\beta,\eta\in\mathbb{R}$ 满足 $\alpha<\beta$ 且 $|\eta|\notin[\alpha,\beta]$. 则下面的最小最大问题

$$\min_{p(t)\in\mathbb{P}_k,p(\eta)=1}\max_{\alpha\le x\le\beta}|p(t)|$$

的唯一解为

$$\hat{T}_k(t)\triangleq\frac{T_k\left(\frac{2t-(\beta+\alpha)}{\beta-\alpha}\right)}{T_k\left(\frac{2\eta-(\beta+\alpha)}{\beta-\alpha}\right)}$$

Chebyshev 加速方法

(1) 迭代矩阵 G 的特征值都是实数;

(2) 迭代矩阵谱半径 $\rho=\rho(G)<1$, 故 $\lambda(G)\in[-\rho,\rho]\subset(-1,1)$

于是最小最大问题 (6.22) 就转化为

$$\min_{p\in\mathbb{P}_k,p(1)=1}\max_{\lambda\in[-\rho,\rho]}\{|p(\lambda)|\}$$

由于 $1\neq[-\rho,\rho]$ 根据定理 4.4, 上述问题的解为

$$p_k(t)=\frac{T_k(t/\rho)}{T_k(1/\rho)}$$

$\tilde{x}^{(k)}$ 的计算

我们无需先计算 $x^{(0)},x^{(1)},\ldots,x^{(k)}$, 然后通过线性组合 (6.19) 来计算 $\tilde{x}^{(k)}$ 事实上, 我们可以通过 Chebyshev 多项式的三项递推公式 (6.23), 由 $\tilde{x}^{(k-1)}$ 和 $\tilde{x}^{(k-2)}$ 直接计算出 $\tilde{x}^{(k)}$. 这样做的另一个好处是无需存储所有的 $\tilde{x}^{(i)}$. 具体的推导公式如下:
令 $\mu_k=\frac{1}{T_k(1/\rho)}$, 即 $T_k(1/\rho)=\frac{1}{\mu_k}$. 由三项递推公式 (6.23) 可得

$$\frac{1}{\mu_k}=\frac{2}{\rho}\cdot\frac{1}{\mu_{k-1}}-\frac{1}{\mu_{k-2}}$$

所以

$$\begin{aligned}\tilde{x}^{(k)}-x_*=p_k(G)\varepsilon_0=\mu_k T_k(G/\rho)\varepsilon_0\\ =\mu_k\left[\frac{2G}{\rho}\cdot\frac{1}{\mu_{k-1}}\left(\tilde{x}^{(k-1)}-x_*\right)-\frac{1}{\mu_{k-2}}\left(\tilde{x}^{(k-2)}-x_*\right)\right]\end{aligned}$$

整理后可得

$$\boxed{\tilde{x}^{(k)}=\frac{2\mu_k}{\mu_{k-1}}\cdot\frac{G}{\rho}\tilde{x}^{(k-1)}-\frac{\mu_k}{\mu_{k-2}}\tilde{x}^{(k-2)}+d_k}$$

其中

$$\begin{aligned}d_k&=x_*-\frac{2\mu_k}{\mu_{k-1}}\cdot\frac{G}{\rho}x_*+\frac{\mu_k}{\mu_{k-2}}x_*\\&=x^*-\frac{2\mu_k}{\mu_{k-1}}\cdot\frac{x_*-g}{\rho}+\frac{\mu_k}{\mu_{k-2}}x_*\\&=\mu_k\left(\frac{1}{\mu_k}-\frac{2}{\rho\mu_{k-1}}+\frac{1}{\mu_{k-2}}\right)x_*+\frac{2\mu_k g}{\mu_{k-1}\rho}\\&=\frac{2\mu_k g}{\mu_{k-1}\rho}\end{aligned}$$

由此, 我们可以得到迭代格式 (6.17) 的 Chebyshev 加速算法.

算法 4.1Chebyshev 加速算法

1:Set $\mu_0=1,\mu_1=\rho=\rho(G),\tilde{x}^{(0)}=x^{(0)},k=1$

2:compute$\tilde{x}^{(1)}=Gx^{(0)}+g$

3:while not converge do

4: $k=k+1$

5: $\mu_k=\left(\frac{2}{\rho}\cdot\frac{1}{\mu_{k-1}}-\frac{1}{\mu_{k-2}}\right)^{-1}$

6: $\tilde{x}^{(k)}=\frac{2\mu_k}{\mu_{k-1}}\cdot\frac{G}{\rho}\tilde{x}^{(k-1)}-\frac{\mu_k}{\mu_{k-2}}\tilde{x}^{(k-2)}+\frac{2\mu_k}{\mu_{k-1}\rho}\cdot g$

7:end while

该算法的每步迭代的整体运算量与原迭代格式基本相当.

设 $\lambda(G)\in[\alpha,\beta]$, 且 $-1<\alpha\leq\beta<1$, 则我们也可以构造出相应的 Chebyshev 加速算法.

SSOR 算法的 Chebyshev 加速

SSOR 迭代矩阵为

$$G_{\text{SSOR}}=(D-\omega U)^{-1}[(1-\omega)D+\omega L](D-\omega L)^{-1}[(1-\omega)D+\omega U]$$

当 A 对称时, 有 $L=U^T$, 故

$$\begin{aligned}
&(D-\omega U)G_{\mathrm{SSOR}}(D-\omega U)^{-1}\\
&=[(1-\omega)D+\omega L](D-\omega L)^{-1}\left[(1-\omega)D+\omega L^T\right]\left(D-\omega L^T\right)^{-1}\\
&=[(2-\omega)D(D-\omega L)^{-1}-I]\left[(2-\omega)D\left(D-\omega L^T\right)^{-1}-I\right]\\
&=I-(2-\omega)D\left[(D-\omega L)^{-1}+\left(D-\omega L^T\right)^{-1}\right]\\
&\quad+(2-\omega)^2D(D-\omega L)^{-1}D\left(I-\omega L^T\right)^{-1}
\end{aligned}$$

假定 D 的对角线元素全是正的, 则

$$\begin{aligned}
&D^{-1/2}(D-\omega U)G_{\mathrm{SSOR}}(D-\omega U)^{-1}D^{1/2}\\
&=I-(2-\omega)D^{-1/2}\left[(D-\omega L)^{-1}+\left(D-\omega L^T\right)^{-1}\right]D^{1/2}\\
&\quad+(2-\omega)^2D^{-1/2}(D-\omega L)^{-1}D\left(I-\omega L^T\right)^{-1}D^{1/2}
\end{aligned}$$

这是一个对称矩阵, 故 G_{SSOR} 具有实特征值. 所以我们可以对其实行 Chebyshev 加速. 但我们需要估计 G_{SSOR} 的谱半径. 若存在矩阵 W 使得 $W^{-1}AW$ 是对称矩阵, 则称 A 是可对称化的, 即 A 相似于一个对称矩阵.

7.6 交替方向与 HSS 算法

7.6.1 多步迭代法

设 $A=M_1-N_1=M_2-N_2$ 是 A 的两个矩阵分裂, 则可以构造迭代格式

$$\begin{cases}
M_1x^{\left(k+\frac{1}{2}\right)}=N_1x^{(k)}+b,\\
M_2x^{(k+1)}=N_2x^{\left(k+\frac{1}{2}\right)}+b
\end{cases}\qquad k=0,1,2,\ldots \tag{6.24}$$

这就是两步迭代方法, 对应的分裂称为二重分裂.
易知, 两步迭代格式 (6.24) 的迭代矩阵为

$$G=M_2^{-1}N_2M_1^{-1}N_1$$

因此, 其收敛的充要条件是 $\rho\left(M_2^{-1}N_2M_1^{-1}N_1\right)<1$. 类似地, 我们可以推广到多步迭代方法.

7.6.2 交替方向法

交替方向 (ADI)本质上是一个两步迭代方法.
设 $A=A_1+A_2$, 则 ADI 迭代格式为

$$\begin{cases}
\left(\alpha I+A_1\right)x^{\left(k+\frac{1}{2}\right)}=\left(\alpha I-A_2\right)x^{(k)}+b,\\
\left(\alpha I+A_2\right)x^{(k+1)}=\left(\alpha I-A_1\right)x^{\left(k+\frac{1}{2}\right)}+b,
\end{cases}\qquad k=0,1,2,\ldots \tag{6.25}$$

其中 $\alpha \in \mathbb{R}$ 是迭代参数. 易知 ADI 算法的迭代矩阵为

$$G_{\text{ADI}} = (\alpha I + A_2)^{-1}(\alpha I - A_1)(\alpha I + A_1)^{-1}(\alpha I - A_2)$$

定理　设 $A \in \mathbb{R}^{n\times n}$ 对称正定,$A = A_1 + A_2$, 其中 A_1 和 A_2 中有一个是对称正定, 另一个是对称半正定, 则对任意正数 $\alpha > 0$, 有 $\rho(G_{\text{ADI}}) < 1$, 即 ADI 迭代方法 (6.25) 收敛.

7.6.3 HSS 方法

设 $A = H + S$, 其中 H 和 S 分别是 A 的对称与斜对称部分, 即

$$H = \frac{A + A^T}{2}, \quad S = \frac{A - A^T}{2}$$

该分裂就称为 HS 分裂, 即 HSS.
类似于 ADI 方法, 我们可得下面的 HSS 方法

$$\begin{cases} (\alpha I + H)x^{\left(k+\frac{1}{2}\right)} = (\alpha I - S)x^{(k)} + b, \\ (\alpha I + S)x^{(k+1)} = (\alpha I - H)x^{\left(k+\frac{1}{2}\right)} + b \end{cases} \quad k = 0, 1, 2, \ldots$$

定理　设 $A \in \mathbb{R}^{n\times n}$ 正定, 则对任意正数 $\alpha > 0$, HSS 迭代方法都收敛

参数 α 的选取

定理　设 $A \in \mathbb{R}^{n\times n}$ 正定, 则极小极大问题

$$\min_{\alpha>0} \max_{\lambda_{\min}(H)\le\lambda\le\lambda_{\max}(H)} \left|\frac{\alpha - \lambda}{\alpha + \lambda}\right|$$

的解为

$$\alpha_* = \sqrt{\lambda_{\max}(H)\lambda_{\min}(H)}$$

此时

$$\sigma(\alpha_*) = \frac{\sqrt{\lambda_{\max}(H)} - \sqrt{\lambda}_{\min}(H)}{\sqrt{\lambda_{\max}(H)} + \sqrt{\lambda}_{\min}(H)} = \frac{\sqrt{\kappa(H)} - 1}{\sqrt{\kappa(H)} + 1}$$

HSS 推广: PSS, NSS, AHSS 等, 感兴趣的读者可以参考相关文献.

7.7 快速 Poisson 算法

如果已经知道 A 的特征值分解 $A = X\Lambda X^{-1}$, 则 $Ax = b$ 的解可表示为

$$x = A^{-1}b = X\Lambda^{-1}X^{-1}b$$

如果 A 是正规矩阵, 即 X 是酉矩阵, 则

$$x = A^{-1}b = X\Lambda^{-1}X^{*}b$$

一般来说, 我们不会采用这种特征值分解的方法来解线性方程组, 因为计算特征值分解通常比解线性方程组更困难.
但在某些特殊情况下, 我们可以由此得到快速算法.
考虑二维离散 Poisson 方程

$$Tu = h^2 f \tag{6.26}$$

其中

$$T = I \otimes T_n + T_n \otimes I = (Z \otimes Z)(I \otimes \Lambda + \Lambda \otimes I)(Z \otimes Z)^T$$

这里 $Z = [z_1, z_2, \ldots, z_n]$ 是正交矩阵,

$$z_k = \sqrt{\frac{2}{n+1}} \cdot \left[\sin\frac{k\pi}{n+1}, \sin\frac{2k\pi}{n+1}, \ldots, \sin\frac{nk\pi}{n+1}\right]^T, \quad k = 1, 2, \ldots, n$$

所以, 方程 (6.26) 的解为

$$u = T^{-1}h^2 f = \left[(Z \otimes Z)(I \otimes \Lambda + \Lambda \otimes I)^{-1}(Z \otimes Z)^T\right] h^2 f$$

因此, 主要的运算是 $Z \otimes Z$ 与向量的乘积, 以及 $(Z \otimes Z)^T$ 与向量的乘积. 而这些乘积可以通过快速 Sine 变换来实现.

离散 Sine 变换

离散 Sine 变换有多种定义, 这里只介绍与求解 Poission 方程有关的一种. 设 $x = [x_1, x_2, \ldots, x_n]^T \in \mathbb{R}^n$, 其离散 Sine 变换 (DST)定义为 $y = \mathrm{DST}(x) = [y_1, y_2, \ldots, y_n]^T \in \mathbb{R}^n$, 其中

$$y_k = \sum_{j=1}^{n} x_j \sin\left(\frac{kj\pi}{n+1}\right), \quad k = 1, 2, \ldots, n$$

对应的离散 Sine 反变换记为 IDST, 即 $x = \mathrm{IDST}(\mathrm{y})$, 其中

$$x_j = \frac{2}{n+1}\sum_{k=1}^{n} y_k \sin\left(\frac{jk\pi}{n+1}\right), \quad j = 1, 2, \ldots, n$$

DST 和 IDST 满足下面的性质:

$$\mathrm{IDST}(\mathrm{DST}(x)) = x, \quad \mathrm{DST}(\mathrm{IDST}(y)) = y$$

在 MATLAB 中, 计算 DST 和 IDST 的函数分别为dst和idst, 即:y=dst(x),x=idst(y).
(测试代码见DST_ test.m)

Possion 方程与 DST

我们首先考虑矩阵 Z 与一个任意给定向量 b 的乘积. 设 $y = Zb$, 则

$$y_k = \sum_{j=1}^{n} Z(k, j)b_j = \sqrt{\frac{2}{n+1}}\sum_{j=1}^{n} b_j \sin\left(\frac{kj\pi}{n+1}\right) = \sqrt{\frac{2}{n+1}} \cdot \mathrm{DST}(b)$$

因此, 乘积 $y=Zb$ 可以通过 DST 来实现. 类似地, 乘积 $y=Z^Tb=Z^{-1}b$ 可以通过离散 Sine 反变换 IDST 实现, 即

$$y=Z^Tb=Z^{-1}b=\left(\sqrt{\frac{2}{n+1}}\right)^{-1}\text{IDST}(b)$$

所以对于一维离散 Poisson 方程, 其解为

$$u=T_n^{-1}\left(h^2f\right)=\left(Z\Lambda^{-1}Z^T\right)\left(h^2f\right)=h^2Z\Lambda^{-1}Z^Tf=h^2\cdot\text{DST}\left(\Lambda^{-1}\,\text{IDST}(b)\right)$$

而对于二维离散 Poisson 方程, 我们需要计算 $(Z\otimes Z)b$ 和 $\left(Z^T\otimes Z^T\right)b$. 它们对应的是二维离散 Sine 变换和二维离散 Sine 反变换.
设 $b=\left[b_1^T,b_2^T,\ldots,b_n^T\right]^T\in\mathbb{R}^{n^2}$, 其中 $b_k\in\mathbb{R}^{R\times n}$. 令 $B=[b_1,b_2,\ldots,b_n]\in\mathbb{R}^{n\times n}$, 则由 Kronecker 乘积的性质可知

$$(Z\otimes Z)b=(Z\otimes Z)\,\text{vec}(B)=\text{vec}\left(ZBZ^T\right)=\text{vec}\left(\left(Z(ZB)^T\right)^T\right)$$

因此, 我们仍然可以使用 DST 来计算 $(Z\otimes Z)b$. 类似地, 我们可以使用 IDST 来计算 $\left(Z^T\otimes Z^T\right)b$.

算法 6.1二维离散 Poisson 方程的快速算法

1: 计算 $b=h^2f$
2: $B=\text{reshape}\,(b,n,n)$
3: $B_1=\left(Z^TB\right)^T=(\text{IDST}(B))^T$
4: $B_2=\left(Z^TB_1\right)^T=(\text{IDST}\,(B_1))^T$
5: $b_1=(I\otimes\Lambda+\Lambda\otimes I)^{-1}\,\text{vec}\,(B_2)$
6: $B_3=\text{reshape}\,(b_1,n,n)$
7: $B_4=(ZB_3)^T=(\text{DST}\,(B_3))^T$
8: $B_5=(ZB_4)^T=(\text{DST}\,(B_4))^T$
9: $u=\text{reshape}\,(B_5,n^2,1)$

MATLAB 程序见Poisson_ DST.m

子空间迭代方法

基本思想

在一个维数较低的子空间中寻找解析解的一个最佳近似. 子空间迭代算法的主要过程可以分解为下面三步:

(1) 寻找合适的子空间;

(2) 在该子空间中求“最佳近似”;

(3) 若这个近似解满足精度要求, 则停止计算; 否则, 重新构造一个新的子空间, 并返回第 (2) 步.

这里主要涉及到的两个关键问题是:

1. 如果选择和更新子空间;
2. 如何在给定的子空间中寻找“最佳近似”.

关于第一个问题, 目前较成功的解决方案就是使用Krylov 子空间.

7.8 Krylov 子空间

设 $A \in \mathbb{R}^{n\times n}, r \in \mathbb{R}^n$, 则由 A 和 r 生成的 m 维Krylov 子空间定义为

$$\boxed{\mathcal{K}_m = \mathcal{K}_m(A,r) \triangleq \text{span}\left\{r, Ar, A^2r, \ldots, A^{m-1}r\right\}, \quad m \le n}$$

设 $\dim \mathcal{K}_m = m$, 令 $v_1, v_2, \ldots, v_m$ 是 $\mathcal{K}_m$ 的一组基, 则 $\forall x \in \mathcal{K}_m$ 可表示为

$$x = y_1v_1 + y_2v_2 + \cdots + y_mv_m \triangleq V_m y$$

寻找“最佳近似”$x^{(m)}$转化为

1. 寻找一组合适的基 $v_1, v_2, \ldots, v_m$;
2. 求出 $x^{(m)}$ 在这组基下面的表出系数 $y^{(m)}$.

基的选取: Arnoldi 过程

最简单的基:$\{r, Ar, A^2r, \ldots, A^{m-1}r\} \longmapsto$ 非正交, 稳定性得不到保证.
Arnoldi 过程: 将 $\{r, Ar, A^2r, \ldots, A^{m-1}r\}$ 单位正交化

1:$v_1 = r/\|r\|_2$
2:for $j = 1$ to m do
3: $z = Av_j$
4: for $i = 1$ to j do % MGS 正交化过程
5: $h_{i,j} = (v_i, z), \quad z = z - h_{i,j}v_i$
6: end for
7: $h_{j+1,j} = \|z\|_2$ % if $h_{j+1,j} = 0$break, endif
8:$v_{j+1} = z/h_{j+1,j}$
9:end for

Arnoldi 过程的矩阵表示

记 $V_m = [v_1, v_2, \ldots, v_m]$

$$H_{m+1,m} = \begin{bmatrix} h_{1,1} & h_{1,2} & h_{1,3} & \cdots & h_{1,m} \\ h_{2,1} & h_{2,2} & h_{2,3} & \cdots & h_{2,m} \\ & h_{3,2} & h_{3,3} & \cdots & h_{3,m} \\ & & \ddots & \ddots & \vdots \\ & & & h_{m,m-1} & h_{m,m} \\ & & & & h_{m+1,m} \end{bmatrix} \in \mathbb{R}^{(m+1)\times m}$$

则由 Arnoldi 过程可知

$$Av_j = h_{1,j}v_1 + h_{2,j}v_2 + \cdots + h_{j,j}v_j + h_{j+1,j}v_{j+1}$$

所以有

$$AV_m = V_{m+1}H_{m+1,m} = V_m H_m + h_{m+1,m}v_{m+1}e_m^T \tag{7.1}$$

其中 $H_m = H_{m+1,m}(1:m, 1:m), e_m = [0, \ldots, 0, 1]^T \in \mathbb{R}^m$. 由于 V_m 是列正交矩阵, 上式两边同乘 V_m^T 可得

$$V_m^T A V_m = H_m \tag{7.2}$$

等式 (7.1) 和 (7.2) 是 Arnoldi 过程的两个重要性质.

Lanczos 过程

若 A 对称, 则 H_m 为对称三对角, 记为 T_m, 即

$$T_m = \begin{bmatrix} \alpha_1 & \beta_1 & & \\ \beta_1 & \ddots & \ddots & \\ & \ddots & \ddots & \beta_{m-1} \\ & & \beta_{m-1} & \alpha_m \end{bmatrix} \tag{7.3}$$

Lanczos 过程的性质与三项递推公式(令 $v_0 = 0$ 和 $\beta_0 = 0$)

$$AV_m = V_m T_m + \beta_m v_{m+1} e_m^T \tag{7.4}$$

$$V_m^T A V_m = T_m \tag{7.5}$$

$$\beta_j v_{j+1} = Av_j - \alpha_j v_j - \beta_{j-1}v_{j-1}, \quad j = 1, 2, \ldots$$

Lanczos 过程

1:Set$v_0 = 0$and$\beta_0 = 0$

2:$v_1 = r/\|r\|_2$

3:for $j = 1$ to m do

4: $z = Av_j$

5: $\alpha_j = (v_j, z)$

6: $z = z - \alpha_j v_j - \beta_{j-1} v_{j-1}$

7: $\beta_j = \|z\|_2$

8: if $\beta_(j) = 0$then break ,end if

9: $v_{j+1} = z/\beta_j$

10:end for

Krylov 子空间算法的一般过程

(1) 令 $m = 1$;

(2) 定义 Krylov 子空间 $\mathcal{K}_m(A, r_0)$;

(3) 找出仿射空间 $x^{(0)} + \mathcal{K}_m$中的“最佳近似”解;

(4) 如果这个近似解满足精度要求, 则迭代结束;
否则令 $m \leftarrow m + 1$, 返回第 (2) 步.

Krylov 子空间迭代算法基本框架

1: 选取初始向量 $x^{(0)}$

2: 计算 $r_0 = b - Ax^{(0)}, v_1 = r_0/\|r_0\|_2$

3: 寻找“最佳近似”解:$x^{(1)} \in x^{(0)} + \mathcal{K}_1 = x^{(0)} + \text{span}\{v_1\}$

4:if $x^{(1)}$ 满足精度要求 then

5: 终止迭代

6:end if

7:for $m = 2$to n do

8: 调用 Arnoldi 或 Lanczos 过程计算向量 v_m

9: 寻找“最佳近似”解:$x^{(m)} \in x^{(0)} + \mathcal{K}_m = x^{(0)} + \text{span}\{v_1, \ldots, v_m\}$

10: if $x^{(m)}$ 满足精度要求 then

11: 终止迭代

12: end if

13:end for

如何计算 $x^{(0)}+\mathcal{K}_m$ 中的“最佳近似”$x^{(m)}$

首先, 我们必须给出“最佳”的定义, 不同的定义会导致不同的算法. 最直接的方式:$\left\|x^{(m)}-x_*\right\|_2$ 达到最小. 但由于 x_* 不知道, 因此不实用.

什么是“最佳”

(1) $\|r_m\|_2=\left\|b-Ax^{(m)}\right\|_2$ 达到最小 A 对称 $\rightarrow$ MINRES,A 非对称 $\rightarrow$GMRES

(2) A 对称正定, 极小化 $\left\|x_*-x^{(m)}\right\|_A \rightarrow$ CG(共轭梯度法)

本讲主要介绍GMRES算法和CG算法.

7.9 GMRES 算法

GMRES 算法是目前求解非对称线性方程组的最常用算法之一.“最佳近似”解的判别方法为使得 $\|r_m\|_2=\left\|b-Ax^{(m)}\right\|_2$ 最小
对任意向量 $x\in x^{(0)}+\mathcal{K}_m$, 可设 $x=x^{(0)}+V_m y$, 其中 $y\in\mathbb{R}^m$. 于是

$$r=b-Ax=r_0-AV_m y=V_{m+1}\left(\beta e_1-H_{m+1,m}y\right)$$

这里 $\beta=\|r_0\|_2$. 由于 V_{m+1} 列正交, 所以

$$\|r\|_2=\|V_{m+1}\left(\beta e_1-H_{m+1,m}y\right)\|_2=\|\beta e_1-H_{m+1,m}y\|_2$$

于是最优性条件就转化为

$$y^{(m)}=\arg\min_{y\in\mathbb{R}^m}\|\beta e_1-H_{m+1,m}y\|_2 \tag{7.6}$$

用基于 Givens 变换的 QR 分解来求解即可.
GMRES 算法的基本框架

算法 2.1GMRES 迭代算法基本框架

1: 选取初值 $x^{(0)}$, 停机标准 $\varepsilon > 0$, 以及最大迭代步数 IterMax
2:$r_0 = b - Ax^{(0)}, \beta = \|r_0\|_2$
3:$v_1 = r_0/\beta$
4:for$j = 1$to IterMax do 5: $w = Av_j$
6: for $i = 1$ to $j$do % Arnoldi 过程
7: $h_{i,j} = (v_i, w)$
8: $w = w - h_{i,j}v_i$
9: end for
10: $h_{j+1,j} = \|w\|_2$
11: if $h_{j+1,j} = 0$ then
12: $m = j,$ break
13: end if
14: $v_{j+1} = w/h_{j+1,j}$
15: 5 relres $= \|r_j\|_2/\beta$
16: if relres $< \varepsilon$ then
17: $m = j,$ break
18: end if
19: end for
20: 解最小二乘问题 (7.6), 得到 y
21:$x^{(m)} = x^{(0)} + V_m y^{(m)}$

实施细节

需要解决下面两个问题:

(1) 如何计算残量 $r_m \triangleq b - Ax^{(m)}$ 的范数?

(2) 如何求解最小二乘问题 (7.6)?

这两个问题可以同时处理.

最小二乘问题的求解

设 $H_{m+1,m}$ 的 QR 分解为

$$H_{m+1,m} = Q_{m+1}^T R_{m+1,m}$$

其中 Q_{m+1} 是正交矩阵,$R_{m+1,m} \in \mathbb{R}^{(m+1)} \times m$ 是上三角矩阵. 则

$$\|\beta e_1 - H_{m+1,m} y\|_2 = \|\beta Q_{m+1} e_1 - R_{m+1,m} y\|_2 = \left\| \beta q_1 - \begin{bmatrix} R_m \\ 0 \end{bmatrix} y \right\|_2$$

其中 $R_m \in \mathbb{R}^{m \times m}$ 非奇异 (假定 $H_{m+1,m}$ 不可约). 所以

$$y^{(m)} = \beta R_m^{-1} q_1(1:m)$$
$$\|r_m\|_2 = \left\|b - Ax^{(m)}\right\|_2 = \left\|\beta e_1 - H_{m+1,m} y^{(m)}\right\|_2 = \beta \cdot |q_1(m+1)|$$

其中 $q_1(m+1)$ 表示 q_1 的第 $m+1$ 个分量

$H_{m+1,m}$ 的 QR 分解的递推计算方法

由于 $H_{m+1,m}$ 是上 Hessenberg 矩阵, 因此我们采用 Givens 变换.

(1) 当 $m = 1$ 时 $H_{21} = \begin{bmatrix} h_{11} \\ h_{21} \end{bmatrix}$, 构造 Givens 变换 G_1 使得 $\overline{G}_1 H_{21} = \begin{bmatrix} * \\ 0 \end{bmatrix} = R_{21}$, 即 $H_{21} = G_1^T R_{21}$

(2) 假定存在 $G_1, G_2, \ldots, G_{m-1}$ 使得

$$(G_{m-1} \cdots G_2 G_1) H_{m,m-1} = R_{m,m-1}$$

即

$$H_{m,m-1} = (G_{m-1} \cdots G_2 G_1)^T R_{m,m-1} \triangleq Q_m^T R_{m,m-1}$$

为了书写方便, 这里假定 Gi 的维数自动扩张, 以满足矩阵乘积的需要.

(3) 考虑 $H_{m+1,m}$ 的 QR 分解. 易知

$H_{m+1,m} = \begin{bmatrix} H_{m,m-1} & h_m \\ 0 & h_{m+1,m} \end{bmatrix}$ 其中 $h_m = [h_{1m}, h_{2m}, \ldots, h_{mm}]^T$

所以有

$$\begin{bmatrix} Q_m & 0 \\ 0 & 1 \end{bmatrix} H_{m+1,m} = \begin{bmatrix} R_{m,m-1} & Q_m h_m \\ 0 & h_{m+1,m} \end{bmatrix} = \begin{bmatrix} R_{m-1} & \tilde{h}_{m-1} \\ 0 & \hat{h}_{mm} \\ 0 & h_{m+1,m} \end{bmatrix}$$

其中 $\tilde{h}_{m-1}$ 是 $Q_m h_m$ 的前 m 个元素组成的向量,$\hat{h}_{mm}$ 是 $Q_m h_m$ 的最后一个元素.
构造 Givens 变换 G_m:

$$G_m = \begin{bmatrix} I_{m-1} & 0 & 0 \\ 0 & c_m & s_m \\ 0 & -s_m c_m & \end{bmatrix} \in \mathbb{R}^{(m+1)\times(m+1)}$$

其中 $c_m = \frac{\hat{h}_{m,m}}{\tilde{h}_{m,m}}, s_m = \frac{h_{m+1,m}}{\tilde{h}_{m,m}}, \tilde{h}_{m,m} = \sqrt{\hat{h}_{m,m}^2 + h_{m+1,m}^2}$ 令

$$Q_{m+1} = G_m \begin{bmatrix} Q_m & 0 \\ 0 & 1 \end{bmatrix}$$

则

$$Q_{m+1}H_{m+1,m} = G_m \begin{bmatrix} R_{m-1} & \tilde{h}_{m-1} \\ 0 & \hat{h}_{j,j} \\ 0 & h_{m+1,m} \end{bmatrix} = \begin{bmatrix} R_{m-1} & \tilde{h}_{m-1} \\ 0 & \tilde{h}_{j,j} \\ 0 & 0 \end{bmatrix} \triangleq R_{m+1,m}$$

所以可得 $H_{m+1,m}$ 的 QR 分解 $H_{m+1,m} = Q_{m+1}^T R_{m+1,m}$.

由 $H_{m,m-1}$ 的 QR 分解到 $H_{m+1,m}$ 的 QR 分解，我们需要

(1) 计算 $Q_m h_m$, 即将之前的 $m-1$ 个 Givens 变换作用到 $H_{m+1,m}$ 的最后一列的前 m 个元素上, 所以我们需要保留所有的 Givens 变换;

(2) 残量计算:$\|r_m\|_2 = |\beta q_1(m+1)| = |\beta Q_{m+1}(m+1,1)|$, 即

$$G_m G_{m-1} \cdots G_2 G_1 (\beta e_1)$$

的最后一个分量的绝对值. 由于在计算 r_{m-1} 时就已经计算出 $G_{m-1} \cdots G_2 G_1 (\beta e_1)$ 因此这里只需做一次 Givens 变换即可;

(3) $y^{(m)}$ 的计算: 当相对残量满足精度要求时, 需要计算 $y^{(m)} = R_m^{-1} q_1(1:m)$ 而 q_1 即为 $G_m G_{m-1} \cdots G_2 G_1 (\beta e_1)$

实用 GMRES 算法

算法 2.2实用 GMRES 算法

1: 给定初值 $x^{(0)}$, 停机标准 $\varepsilon > 0$, 最大迭代步数 IterMax

2:$r_0 = b - Ax^{(0)}, \beta = \|r_0\|_2$

3:if $\beta < \epsilon$ then

4: 停止计算,输出近似解 $x^{(0)}$

5:end if

6:$v_1 = r_0/\beta$

7:$\xi = \beta e_1$ 记录 q_1

8:for$j = 1$to IterMax do 9: $w = Av_j$

10: for $i = 1$ to $j$do % Arnoldi 过程

11: $h_{i,j} = (v_i, w)$

12: $w = w - h_{i,j}v_i$

13: end for

14: $h_{j+1,j} = \|w\|_2$

15: if $h_{j+1,j} = 0$ then % 迭代中断

16: $m = j,$ break

17: end if

18: $v_{j+1} = w/h_{j+1,j}$

19: for$i = 1$to$j - 1$do % 计算 $G_{j-1} \cdots G_2 G_1 H_{j+1,j}(1:j, j)$

20: $\begin{bmatrix} h_{ij} \\ h_{i+1,j} \end{bmatrix} = \begin{bmatrix} c_i & s_i \\ -s_i & c_i \end{bmatrix} \begin{bmatrix} h_{ij} \\ h_{i+1,j} \end{bmatrix}$

21: end for 22: if$|h_{jj}| > |h_{j+1,j}|$then % 构造 Givens 变换 G_j

23: $\tau = h_{j+1,j}/h_{jj}, c_j = 1/\sqrt{1+\tau^2}, s_j = c_j\tau$

24: else

25: $\tau = h_{jj}/h_{j+1,j}, s_j = 1/\sqrt{1+\tau^2}, c_j = s_j\tau$

26: end if

27: $h_{jj} = c_j h_{jj} + s_j h_{j+1,j}$ % 计算 $G_j H_{j+1,j}(1:j, j)$

28: $h_{j+1,j} = 0$ 29: $\begin{bmatrix} \xi_j \\ \xi_{j+1} \end{bmatrix} = \begin{bmatrix} c_j & s_j \\ -s_j & c_j \end{bmatrix} \begin{bmatrix} \xi_j \\ 0 \end{bmatrix}$ % 计算 $G_j\left(\beta G_{j-1} \cdots G_2 G_1 e_1\right)$

30: relres $= |\xi_{j+1}|/\beta$ % 相对残量

31: if $relres < \epsilon$ then

32: $m = j,$ break

33: end if

34:end for

35:$m = j$

36:$y^{(m)} = H(1:m, 1:m) \backslash \xi(1:m)$ % 最小二乘问题,回代求解

GMRES 算法的中断

在上面的 GMRES 算法中, 当执行到某一步时有 $h_{j+1,j}=0$, 则算法会中断 (breakdown). 如果出现这种中断, 则我们就可以找到精确解

定理　设 $A\in\mathbb{R}^{n\times n}$ 非奇异且 $r_0\neq 0$. 若 $h_{i+1,i}\neq 0, i=1,2,\ldots,k-1$ 则 $h_{k+1,k}=0$ 当且仅当 $x^{(k)}$ 是方程组的精确解. (不考虑舍入误差)

带重启的 GMRES 算法

由于随着迭代步数的增加, GMRES 算法的每一步所需的运算量和存储量都会越来越大. 因此当迭代步数很大时, GMRES 算法就不太实用.

重启技术

事先设定一个重启迭代步数 k, 如果 GMRES 达到这个迭代步数时仍不收敛, 则计算出 $x^{(0)}+\mathcal{K}_k$ 中的最佳近似解 $x^{(k)}$, 然后令 $x^{(0)}=x^{(k)}$, 重新开始新的 GMRES 迭代.

算法 2.3带重启的 GMRES 算法

1: 设定重启步数 $k(\ll n)$

2: 给定初值 $x^{(0)}$, 停机标准 $\varepsilon > 0$, 最大迭代步数 IterMax

3:$r_0 = b - Ax^{(0)}, \beta = \|r_0\|_2$

4:if $\beta < \varepsilon$ then

5: 停止计算, 输出近似解 $x = x^{(0)}$

6:end if

7:for $iter = 1$ to $ceil(IterMax/k)$ do

8: $v_1 = r_0/\beta$

9: $\xi = \beta e_1$

10: for $j = 1$ to k do

11: 调用 GMRES 循环

12: end for

13: $m = j$

14: $y^{(m)} = H(1:m, 1:m)\backslash\xi(1:m)$

15: $x^{(m)} = x^{(0)} + V_m y^{(m)}$

16: if $relres < \varepsilon$ then % 收敛。退出循环

17: break 18: end if 19: $x^{(0)} = x^{(m)}$% 重启 GMRES

20: $r_0 = b - Ax^{(0)}, \beta = \|r_0\|_2$

21:end for

22:if $relres < \varepsilon$ then

23: 输出近似解 $x^{(m)}$ 及相关信息

24:else

25: 输出算法失败信息

26:end if

带重启的 GMRES 算法需要注意的问题

(1) 如何选取合适的重启步数 k?
一般只能依靠经验来选取, 如 $k = 20, 50$.

(2) 不带重启的 GMRES 算法能保证算法的收敛性, 但带重启的 GM-RES 算法却无法保证, 有时可能出现停滞现象 (stagnation).

7.10 共轭梯度法 (CG)

“最佳近似”:$\left\|x_* - x^{(m)}\right\|_A$ 最小

首先给出“最佳近似”解 $x^{(m)}$ 的一个性质.

定理 设 A 对称正定, 则

$$x^{(m)} = \arg \min_{x \in x^{(0)} + \mathcal{K}_m} \|x - x_*\|_A \tag{7.7}$$

当且仅当

$$x^{(m)} \in x^{(0)} + \mathcal{K}_m \quad \mathbb{H} \quad b - Ax^{(m)} \perp \mathcal{K}_m \tag{7.8}$$

Lanczos 过程

Lanczos 过程的三项递推公式:

$$\begin{aligned} AV_m &= V_{m+1} T_{m+1,m} = V_m T_m + \beta_m v_{m+1} e_m^T \\ V_m^T A V_m &= T_m \end{aligned}$$

其中 $T_m = \text{tridiag}\left(\beta_i, \alpha_{i+1}, \beta_{i+1}\right)$

由前面的结论可知, 此时我们需要在 $x^{(0)} + \mathcal{K}_m$ 寻找最优解 $x^{(m)}$, 满足

$$b - Ax^{(m)} \perp \mathcal{K}_m \tag{7.9}$$

下面就根据这个性质推导 CG 算法的迭代公式.

CG 算法的推导

首先, 设 $x^{(m)} = x^{(0)} + V_m z^{(m)}$, 其中 $z^{(m)} \in \mathbb{R}^m$. 由 (7.9) 可知

$$0 = V_m^T \left(b - Ax^{(m)}\right) = V_m^T \left(r_0 - AV_m z^{(m)}\right) = \beta e_1 - T_m z^{(m)}$$

因此,

$$z^{(m)} = T_m^{-1} \left(\beta e_1\right)$$

设 T_m 的 LDL^T 分解为 $T_m = L_m D_m L_m^T$. 于是

$$x^{(m)} = x^{(0)} + V_m z^{(m)} = x^{(0)} + V_m T_m^{-1} \left(\beta e_1\right) = x^{(0)} + \left(V_m L_m^{-T}\right) \left(\beta D_m^{-1} L_m^{-1} e_1\right)$$

如果 $x^{(m)}$ 满足精度要求, 则计算结束. 否则我们需要计算

$$x^{(m+1)} = x^{(0)} + V_{m+1} T_{m+1}^{-1} \left(\beta e_1\right) = x^{(0)} + \left(V_{m+1} L_{m+1}^{-T}\right) \left(\beta D_{m+1}^{-1} L_{m+1}^{-1} e_1\right)$$

这里 $T_{m+1} = L_{m+1} D_{m+1} L_{m+1}^T$

记

$$\begin{aligned} \tilde{P}_m &\triangleq V_m L_m^{-T} = [\tilde{p}_1, \tilde{p}_2, \ldots, \tilde{p}_m] \in \mathbb{R}^{n \times m} \\ y_m &\triangleq \beta D_m^{-1} L_m^{-1} e_1 = [\eta_1, \ldots, \eta_m]^T \in \mathbb{R}^m \end{aligned}$$

$\tilde{P}_m$ 和 y_m 的递推关系式(由 T_{m+1} 的 LDL^{T} 分解可得)

$$\begin{aligned} \tilde{P}_{m+1} &\triangleq V_{m+1} L_{m+1}^{-T} = \left[\tilde{P}_m, \tilde{p}_{m+1}\right] \\ y_{m+1} &\triangleq \beta D_{m+1}^{-1} L_{m+1}^{-1} e_1 = \left[y_m^T, \eta_{m+1}\right]^T, \quad m = 1, 2, \ldots \end{aligned}$$

$\tilde{P}_{m+1}$ 的递推关系式

$$\tilde{p}_{m+1} = -l_m \tilde{p}_m + v_{m+1}$$

x^{m+1} 的递推关系式

$$x^{(m+1)} = \tilde{P}_{m+1} y_{m+1} = \left[\tilde{P}_m, \tilde{p}_{m+1}\right] \begin{bmatrix} y_m \\ \eta_{m+1} \end{bmatrix} = x^{(m)} + \eta_{m+1}\tilde{p}_{m+1}$$

$m+1$ 的递推关系式(收敛性判断)

$$r_{m+1} = b - Ax^{(m+1)} = b - A\left(x^{(m)} + \eta_{m+1}\tilde{p}_{m+1}\right) = r_m - \eta_{m+1}A\tilde{p}_{m+1}$$

另一方面, 我们有

$$r_m = b - Ax^{(m)} = r_0 - AV_m z^{(m)} = -\beta_m \left(e_m^T z^{(m)}\right) v_{m+1}$$

即 r_m 与 $v_m + 1$ 平行. 记$r_m = T_m v_{m+1}$, 其中

$$\tau_0 = \beta = \|r_0\|_2, \quad \tau_m = -\beta_m \left(e_m^T z^{(m)}\right), \quad m = 1, 2, \ldots$$

p^{m+1} 的递推关系式(定义 $p_m = \tau_{m-1}\tilde{p}_m$)

$$p_{m+1} = \tau_m \tilde{p}_{m+1} = \tau_m \left(v_{m+1} - l_m \tilde{p}_m\right) = r_m + \mu_m p_m \tag{7.10}$$

其中 $\mu_m = -l_m \tau_m / \tau_{m-1}, m = 1, 2, \ldots$ $x^{(m+1)}$ 和 r_{m+1} 的新递推关系式

$$x^{(m+1)} = x^{(m)} + \eta_{m+1}\tilde{p}_{m+1} = x^{(m)} + \xi_{m+1} p_{m+1} \tag{7.11}$$

$$r_{m+1} = r_m - \eta_{m+1} A\tilde{p}_{m+1} = r_m - \xi_{m+1} A p_{m+1} \tag{7.12}$$

其中 $\xi_{m+1} = \eta_{m+1}/\tau_m, m = 1, 2, \ldots$

系数 $\xi m + 1$ 和 μ_m 的计算方法

引理　下面的结论成立:

(1) $r_1, r_2, \ldots, r_m$ 相互正交;

(2) $p_1, p_2, \ldots, p_m$ 相互 $A-$ 共轭 ($A-$ 正交), 即当 $i \neq j$ 时有 $p_i^T A p_j = 0$.

在等式 (7.10) 两边同时左乘 $p_{m+1}^T A$ 可得

$$p_{m+1}^T A p_{m+1} = p_{m+1}^T A r_m + \mu_m p_{m+1}^T A p_m = r_m^T A p_{m+1}$$

再用 r_m^T 左乘方程 (7.12) 可得

$$0 = r_m^T r_{m+1} = r_m^T r_m - \xi_{m+1} r_m^T A p_{m+1}$$

于是

$$\xi_{m+1} = \frac{r_m^T r_m}{r_m^T A p_{m+1}} = \frac{r_m^T r_m}{p_{m+1}^T A p_{m+1}} \tag{7.13}$$

等式 (7.10) 两边同时左乘 $p_m^T A$ 可得

$$0 = p_m^T A p_{m+1} = p_m^T A r_m + \mu_m p_m^T A p_m \Longrightarrow \mu_m = -\frac{r_m^T A p_m}{p_m^T A p_m}$$

为了进一步减少运算量, 将上式简化. 用 r_{m+1}^T 左乘方程 (7.12) 可得

$$r_{m+1}^T r_{m+1} = r_{m+1}^T r_m - \xi_{m+1} r_{m+1}^T A p_{m+1} = -\xi_{m+1} r_{m+1}^T A p_{m+1}$$

于是

$$\xi_{m+1} = -\frac{r_{m+1}^T r_{m+1}}{r_{m+1}^T A p_{m+1}} \Longrightarrow \xi_m = -\frac{r_m^T r_m}{r_m^T A p_m}$$

即 $r_m^T A p_m = -r_m^T r_m / \xi_m$ 于是

$$\mu_m = -\frac{r_m^T A p_m}{p_m^T A p_m} = \frac{r_m^T r_m}{p_m^T A p_m} \cdot \frac{1}{\xi_m} = \frac{r_m^T r_m}{r_{m-1}^T r_{m-1}} \tag{7.14}$$

注意, 以上递推公式是从 $m = 1$ 开始的. 因此 $m = 0$ 时需要另外推导. 首先, 由 $\tilde{p}_1$ 的定义可知

$$\tilde{p}_1 = \tilde{P}_1 = V_1 L_1^{-T} = v_1 \Longrightarrow \qquad p_1 = \tau_0 \tilde{p}_1 = \beta v_1 = r_0$$

其次, 由 Lanczos 过程可知 $T_1 = \alpha_1 = v_1^T A v_1$. 注意到 $\beta = r_0^T r_0$, 于是

$$x^{(1)} = x^{(0)} + V_1 T_1^{-1} (\beta e_1) = x^{(0)} + \frac{\beta}{v_1^T A v_1} v_1 = x^{(0)} + \frac{r_0^T r_0}{p_1^T A p_1} p_1$$

令 $\xi_1 = \frac{r_0^T r_0}{p_1^T A p_1}$(注: 之前的 $\xi m + 1$ 计算公式 (7.13) 只对 $m \geq 1$ 有定义), 则当 $m = 0$ 时关于 $x^{(m+1)}$ 的递推公式仍然成立.
最后考虑残量. 易知

$$r_1 = b - A x^{(1)} = b - A x^{(0)} - \frac{r_0^T r_0}{p_1^T A p_1} A p_1 = r_0 - \xi_1 A p_1$$

即当 $m = 0$ 时关于 r_{m+1} 的递推公式也成立.

共轭梯度法

算法 2.2实用 GMRES 算法

1: 给定初值 $x^{(0)}$, 停机标准 $\varepsilon > 0$, 最大迭代步数 IterMax
2:$r_0 = b - Ax^{(0)}$,
3:$\beta = \|r_0\|_2$
4:if $\beta < \epsilon$ then
5: 停止计算,输出近似解 $x^{(0)}$
6:end if
7:for $m = 1$ to $IterMax$ do
8: $\rho = r_{m-1}^T r_{m-1}$
9: if $m > 1$ then
10: $\mu_{m-1} = \rho/\rho_0$
11: $p_m = r_{m-1} + \mu_{m-1} p_{m-1}$
12: else
13: $p_m = r_0$
14: end if
15: $q_m = Ap_m$
16: $\xi_m = \rho/\left(p_m^T q_m\right)$
17: $x^{(m)} = x^{(m-1)} + \xi_m p_m$
18: $r_m = r_{m-1} - \xi_m q_m$
19: relres $= \|r_m\|_2 / \beta$
20: if relres $< \varepsilon$ then
21: 停止迭代, 输出近似解 $x^{(m)}$
22: end if
23: $\rho_0 = \rho$
24:end for
25:if relres $< \varepsilon$ then
26: 输出近似解 $x^{(m)}$ 及相关信息
27:else
28: 输出算法失败信息 29:end if

CG 算法的每个迭代步的主要运算为一个矩阵向量乘积和两个向量内积;

7.11 收敛性分析

CG 算法的收敛性

设 x_* 是解析解,$x^{(m)}$ 是 CG 算法在 $x^{(0)}+\mathcal{K}_m$ 中找到的近似解, 即

$$x^{(m)}=\arg\min_{x\in x^{(0)}+\mathcal{K}_m}\|x-x_*\|_A$$

记 $\mathbb{P}_k$ 为所有次数不超过 k 的多项式的集合. 对任意 $x\in x^{(0)}+\mathcal{K}_m$, 存在 $p(t)\in\mathbb{P}_{m-1}$, 使得

$$x=x^{(0)}+p(A)r_0$$

于是有

$$x-x_*=\varepsilon_0+p(A)\left(b-Ax^{(0)}\right)=\varepsilon_0+p(A)\left(Ax_*-Ax^{(0)}\right)\triangleq q(A)\varepsilon_0$$

其中 $\varepsilon_0=x^{(0)}-x_*$ 多项式 $q(t)=1-tp(t)\in\mathbb{P}_m$ 且 $q(0)=1$. 所以

$$\|x-x_*\|_A^2=\varepsilon_0^Tq(A)^TAq(A)\varepsilon_0$$

设 $A=Q\Lambda Q^T,\Lambda=\operatorname{diag}(\lambda_1,\lambda_2,\ldots,\lambda_n)$, 记 $y=[y_1,y_2,\ldots,y_n]^T\triangleq Q^T\varepsilon_0$.

$$\begin{aligned}
\left\|x^{(m)}-x_*\right\|_A^2&=\min_{x\in x^{(0)}+\mathcal{K}_m}\|x-x_*\|_A^2\\
&=\min_{q\in\mathbb{P}_m,q(0)=1}\varepsilon_0^TQq(\Lambda)^T\Lambda q(\Lambda)Q^T\varepsilon_0\\
&=\min_{q\in\mathbb{P}_m,q(0)=1}\sum_{i=1}^n y_i^2\lambda_iq\left(\lambda_i\right)^2\\
&\le\min_{q\in\mathbb{P}_m,q(0)=1}\max_{1\le i\le n}\left\{q\left(\lambda_i\right)^2\right\}\sum_{i=1}^n y_i^2\lambda_i\\
&=\min_{q\in\mathbb{P}_m,q(0)=1}\max_{1\le i\le n}\left\{q\left(\lambda_i\right)^2\right\}y^T\Lambda y\\
&=\min_{q\in\mathbb{P}_m,q(0)=1}\max_{1\le i\le n}\left\{q\left(\lambda_i\right)^2\right\}\varepsilon_0^TA\varepsilon_0\\
&=\min_{q\in\mathbb{P}_m,q(0)=1}\max_{1\le i\le n}\left\{q\left(\lambda_i\right)^2\right\}\|\varepsilon_0\|_A^2
\end{aligned}$$

引理　设 $x^{(m)}$ 是 CG 算法迭代 m 步后得到的近似解. 则

$$\frac{\left\|x^{(m)}-x_*\right\|_A}{\left\|x^{(0)}-x_*\right\|_A}\le\min_{q\in\mathbb{P}_m,q(0)=1}\max_{1\le i\le n}\left|q\left(\lambda_i\right)\right|$$

当 A 的特征值不知道时, 可用区间代替, 即

$$\frac{\left\|x^{(m)}-x_*\right\|_A}{\left\|x^{(0)}-x_*\right\|_A}\le\min_{q\in\mathbb{P}_m,q(0)=1}\max_{\lambda_n\le\lambda\le\lambda_1}|q(\lambda)|$$

由 Chebyshev 多项式的最佳逼近性质可知, 上式的解为

由 Chebyshev 多项式的最佳逼近性质可知, 上式的解为

$$\tilde{q}(t)=\frac{T_m\left(\frac{2t-(\lambda_1+\lambda_n)}{\lambda_1-\lambda_n}\right)}{T_m\left(-\frac{\lambda_1+\lambda_n}{\lambda_1-\lambda_n}\right)} \Longrightarrow |\tilde{q}(t)| \leq \frac{1}{2}\left(\frac{\sqrt{\kappa(A)}+1}{\sqrt{\kappa(A)}-1}\right)^m$$

定理　设 $A\in\mathbb{R}^{n\times n}$ 对称正定,$x^{(m)}$ 是 CG 算法迭代 m 步后得到的近似解. 则

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

其中 $\kappa(A)=\lambda_1/\lambda_n$

CG 算法的超收敛性

如果我们能够获得 A 的更多的特征值信息, 则能得到更好的误差限.

定理　设 $A\in\mathbb{R}^{n\times n}$ 对称正定, 特征值为

$$0<\lambda_n\leq\cdots\leq\lambda_{n+1-i}\leq b_1\leq\lambda_{n-i}\leq\cdots\leq\lambda_{j+1}\leq b_2\leq\lambda_j\leq\cdots\leq\lambda_1$$

则当 $m\geq i+j$ 时有

$$\frac{\left\|x^{(m)}-x_*\right\|_A}{\left\|x^{(0)}-x_*\right\|_A} \leq 2\left(\frac{b-1}{b+1}\right)^{m-i-j}\max_{\lambda\in[b_1,b_2]}\left\{\prod_{k=n+1-i}^{n}\left(\frac{\lambda-\lambda_k}{\lambda_k}\right)\prod_{k=1}^{j}\left(\frac{\lambda_k-\lambda}{\lambda_k}\right)\right\}$$

其中 $b=(b_2/b_1)^{\frac{1}{2}}\geq 1$.

由此可知, 当 b_1 与 b_2 非常接近时, 迭代 $i+j$ 步后,CG 收敛会非常快!

推论　设 A 对称正定, 特征值为

$$\begin{aligned}0<\delta\leq\lambda_n\leq\cdots\leq\lambda_{n+1-i}\leq&\\ 1-\varepsilon\leq\lambda_{n-i}\leq\cdots\leq\lambda_{j+1}&\leq 1+\varepsilon\\ &\leq\lambda_j\leq\cdots\leq\lambda_1\end{aligned}$$

则当 $m\geq i+j$ 时有

$$\frac{\left\|x^{(m)}-x_*\right\|_A}{\left\|x^{(0)}-x_*\right\|_A} \leq 2\left(\frac{1+\varepsilon}{\delta}\right)^i\varepsilon^{m-i-j} \tag{7.16}$$

GMRES 算法的收敛性

正规矩阵情形:$A=U\Lambda U^*$

定理　设 $A\in\mathbb{R}^{n\times n}$ 是正规矩阵,$x^{(m)}$ 是 GMRES 得到的近似解, 则

$$\frac{\left\|b-Ax^{(m)}\right\|_2}{\left\|r_0\right\|_2} \leq \min_{q\in\mathbb{P}_m,q(0)=1}\max_{1\leq i\leq n}|q(\lambda_i)| \tag{7.17}$$

需要指出的是, 上界 (7.17) 是紧凑的.

设 $\Omega \subset \mathbb{C}$ 是包含 A 的所有特征值的一个区域 (不能包含原点), 则

$$\frac{\left\|b - Ax^{(m)}\right\|_2}{\|r_0\|_2} \leq \min_{q\in\mathbb{P}_k, q(0)=1} \max_{\lambda\in\Omega} |q(\lambda)|$$

通常 Ω 必须是连通的, 否则求解非常困难, 即使两个区间的并都没法求解.

非正规情形

设 $A \in \mathbb{R}^{n\times n}$ 可对角化, 即 $A = X\Lambda X^{-1}$, 则

$$\left\|b - Ax^{(k)}\right\|_2 = \min_{x\in x^{(0)}+\mathcal{K}_k(A,r_0)} \|b - Ax\|_2 = \min_{q\in\mathbb{P}_k, q(0)=1} \|q(A)r_0\|_2 \tag{7.18}$$

相类似地, 我们可以得到下面的结论.

定理　设 $A = X\Lambda X^{-1}$ 其中 $X \in \mathbb{C}^{n\times n}$ 非奇异,Λ 是对角矩阵,$x^{(k)}$ 是 GMRES 算法得到的近似解, 则

$$\begin{aligned}\frac{\left\|b - Ax^{(k)}\right\|_2}{\|r_0\|_2} &\leq \|X\|_2 \left\|X^{-1}\right\|_2 \min_{q\in P_k, q(0)=1} \max_{1\leq i\leq n} |q(\lambda_i)| \\ &= \kappa(X) \min_{q\in P_k, q(0)=1} \max_{1\leq i\leq n} |q(\lambda_i)|\end{aligned} \tag{7.19}$$

其中 $\kappa(X)$ 是 X 的谱条件数.

如果 A 接近正规, 则 $\kappa(X) \approx 1$. 此时上界 (7.19) 在一定程度上能描述 GMRES 的收敛速度.

当如果 X 远非正交, 则 $\kappa(X)$ 会很大, 此时该上界就失去实际意义了.

需要指出的是, 上面的分析并不意味着非正规矩阵就一定比正规矩阵收敛慢. 事实上, 对任意一个非正规矩阵, 总存在一个相应的正规矩阵, 使得 GMRES 算法的收敛速度是一样的.

虽然 GMRES 算法的收敛性与系数矩阵的特征值有关, 但显然并不仅仅取决于特征值的分布. 事实上, 我们有下面的结论.

定理　对于任意给定的特征值分布和一条不增的收敛曲线, 则总存在一个矩阵 A 和一个右端项 b, 使得 A 具有指定的特征值分布, 且 GMRES 算法的收敛曲线与给定的收敛曲线相同.

例　考虑线性方程组 $Ax = b$ 其中

$$A = \begin{bmatrix} 0 & 1 & & & \\ & 0 & 1 & & \\ & \ddots & \ddots & & \\ & & & 0 & 1 \\ a_0 & a_1 & a_2 & \cdots & a_{n-1} \end{bmatrix}, \quad b = e_1$$

当 $a_0 \neq 0$ 时,A 非奇异. 易知,A 的特征值多项式为

$$p(x) = \lambda^n - a_{n-1}\lambda^{n-1} - a_{n-2}\lambda^{n-2} - \cdots - a_1\lambda - a_0$$

方程组的精确解为

$$x = [-a_1/a_0, 1, 0, \ldots, 0]^\top$$

以零向量为迭代初值, 则 GMRES 迭代到第 n 步时才收敛. (前 $n-1$ 步残量范数不变)

如果 A 不可以对角化

我们在分析 GMRES 算法的收敛性时, 通常会想办法用一个新的极小化问题来近似原来的极小化问题 (7.18). 当然, 这个新的极小化问题应该是比较容易求解的.

事实上, 我们有

$$\begin{aligned}
\frac{\left\|b - Ax^{(k)}\right\|_2}{\|r_0\|_2} &= \frac{\min_{q\in\mathbb{P}_k, q(0)=1} \|q(A)r_0\|_2}{\|r_0\|_2} \\
&\leq \max_{\|v\|_2=1} \min_{q\in P_k, q(0)=1} \|q(A)v\|_2 \\
&\leq \min_{q\in\mathbb{P}_k, q(0)=1} \|q(A)\|_2
\end{aligned}$$

不等式 (7.20) 右端代表的是在最坏情况下的 GMRES 收敛性, 而且是紧凑的, 即它是所能找到的不依赖于 r_0 的最好上界. 但我们仍然不清楚, 到底是 A 的那些性质决定着这个上界 [?]

可以证明, 当 A 是正规矩阵时, 上界 (7.20) 和 (7.21) 是相等的 [? ?]. 但是, 对于大多数非正规矩阵而言, 这两者是否相等或者非常接近, 迄今仍不太清楚

最后需要指出的是, 算法的收敛性也依赖于迭代初值和右端项. 所以上定理中的上界描述的都是最坏情况下的收敛速度. 也就是说, 在实际计算中, 算法的收敛速度可能会比预想的要快得多.

7.12 其它 Krylov 子空间迭代算法

对称	CG (1952)	对称正定, 正交投影法 (Galerkin)
	MINRES (1975	对称不定, 斜投影法 (Petrov-Galerkin)
	SYMMLQ (1975)	对称不定
	SQMR (1994)	对称不定
非对称	FOM (1981)	正交投影法, Arnoldi
	GMRES (1984)	斜投影法 (Petrov-Galerkin), Arnoldi
	BiCG (1976)	双正交 (biorthogonalization
	QMR (1991)	双正交 (biorthogonalization
	CGS (1989)	Transpose free
	BiCGStab (1992)Tr	Transpose free, smoother convergence than C
	TFQMR (1993)	Transpose free, smoother convergence than C
	FGMRES (1993)	
正规方程	CGLS (1982)	最小二乘 (法方程)
	LSQR (1982)	最小二乘 (法方程)