第四讲　非对称特征值问题

1　幂迭代

2　正交迭代

3　QR 迭代

4　带位移的隐式 QR 迭代

5　特征向量的计算

6　广义特征值问题

设 $A \in \mathbb{R}^{n\times n}$ 是一个非对称的稠密矩阵.

本讲主要讨论如何计算 A 的全部特征值和 (或) 特征向量.

记 A 的特征值为 $\lambda_1, \lambda_2, \ldots, \lambda_n$. 本讲中我们总是假定

$$|\lambda_1| \geq |\lambda_2| \geq \cdots \geq |\lambda_n| \geq 0.$$

主要介绍以下方法:

- 幂迭代算法
- 位移策略与反迭代技巧
- 正交迭代法
- QR 算法 / 实用的 QR 算法

关于稠密矩阵特征值计算的参考资料有：

- J. H. Wilkinson, The Algebraic Eigenvalue Problem, 1965 [65]
- B. N. Parlett, The Symmetric Eigenvalue Problem, 2nd Eds., 1998 [49]
- G. W. Stewart, Matrix Algorithms, Vol II: Eigensystems, 2001 [58]
- G. H. Golub and C. F. Van Loan, Matrix Computations, 2013 [27]
- P. Arbenz, The course 252-0504-00 G, Numerical Methods for Solving Large Scale Eigenvalue Problems, 2014. online

1 幂迭代

1.1 幂迭代算法

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

算法 1.1 幂迭代算法 (Power Iteration)

Choose an initial guess $x^{(0)}$ with $\|x^{(0)}\|_2 = 1$
set $k = 0$
while not convergence **do**
    $y^{(k+1)} = Ax^{(k)}$
    $x^{(k+1)} = y^{(k+1)} / \|y^{(k+1)}\|_2$
    $\mu_{k+1} = (x^{(k+1)}, Ax^{(k+1)})$ % 内积
    $k = k + 1$
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 $\Lambda = \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n)$, $V = [v_1, v_2, \ldots, v_n] \in \mathbb{C}^{n\times n}$, 且 $\|v_i\|_2 = 1, i = 1, 2, \ldots, n$.

(2) 同时, 我们还假设 $|\lambda_1| > |\lambda_2| \geq |\lambda_3| \geq \cdots \geq |\lambda_n|$.

由于 V 的列向量组构成 $\mathbb{C}^n$ 的一组基, 因此 $x^{(0)}$ 可表示为

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

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

于是我们可得

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

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

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

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

收敛到 $e_1 = [1, 0, \ldots, 0]^T$.

所以向量 $x^{(k)} = A^k x^{(0)} / \|A^k x^{(0)}\|_2$ 收敛到 $\pm v_1$, 即 λ_1 的特征向量.

而 $\mu_k = (x^{(k)})^* A x^{(k)}$ 则收敛到 $v_1^* A v_1 = \lambda_1$.

† 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.

1.2 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 $|\lambda_2/\lambda_1|$

策略: 位移策略 (shift), 即计算 $A-\sigma I$ 的特征值

我们称 σ 为**位移** (shift), 满足

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

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

其中第一个条件保证最后所求得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

改进方法: 与反迭代相结合, 能起到很好的加速效果

1.3 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 1.2 反迭代算法 (Inverse Iteration)

Choose a scalar σ and an initial vector $x^{(0)}$ with $\|x^{(0)}\|_2 = 1$

set $k = 0$

while not convergence **do**

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

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

    $\mu_{k+1} = (x^{(k+1)}, Ax^{(k+1)})$

    $k = k + 1$

end while

显然: μ_k 收敛到距离 σ 最近的特征值, $x^{(k)}$ 收敛到对应的特征向量

事实上, 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.

问题:

- 怎样选取位移 σ? $\rightarrow$ Rayleigh 商: 动态选取, 自动调整

1.4 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 1.3 Rayleigh 商迭代 (Rayleigh Quotient Iteration (RQI))

Choose an initial vector $x^{(0)}$ with $\|x^{(0)}\|_2 = 1$
set $k = 0$
compute $\sigma = (x^{(0)})^* A x^{(0)}$
while not converge **do**
    $y^{(k+1)} = (A - \sigma I)^{-1} x^{(k)}$
    $x^{(k+1)} = y^{(k+1)} / \|y^{(k+1)}\|_2$
    $\mu_{k+1} = (x^{(k+1)}, A x^{(k+1)})$
    $\sigma = \mu_{k+1}$
    $k = k + 1$
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.

2 正交迭代

出发点: 同时计算多个特征值

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, 从而可以获得多个特征值.

算法 2.1 正交迭代算法 (Orthogonal Iteration)

Choose an $n \times p$ column orthogonal matrix Z_0
set $k = 0$
while not convergence **do**
    compute $Y_{k+1} = AZ_k$
    $Y_{k+1} = Z_{k+1}\hat{R}_{k+1}$ % QR 分解
    $k = k + 1$
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 $\mathrm{span}\{A^i Z_0\}$ 的一组正交基, 以确保算法的数值稳定性.

收敛性分析

假设 A 是可对角化的, 即 $A = V\Lambda V^{-1}$, 其中 $\Lambda = \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n)$, 且 $|\lambda_1| \geq \cdots \geq |\lambda_p| > |\lambda_{p+1}| \geq \cdots \geq |\lambda_n|$. 则可得

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

由此可知

$$\mathrm{span}\{Z_k\} = \mathrm{span}\{A^k Z_0\} = \mathrm{span}\{V\Lambda^k V^{-1} Z_0\}.$$

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

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

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

定理 给定正整数 p $(1 \le p \le n)$, 考虑算法 2.1 . 假设 A 是可对角化的, 且 $|\lambda_1| \ge \cdots \ge |\lambda_p| > |\lambda_{p+1}| \ge \cdots \ge |\lambda_n|$. 则 $\mathrm{span}\{Z_k\}$ 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 [63, 64].

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.

3 QR 迭代

3.1 算法介绍

3.2 QR 迭代与幂迭代的关系

3.3 QR 迭代与反迭代的关系

3.4 QR 迭代与正交迭代的关系

3.5 QR 迭代的收敛性

3.6 带位移的 QR 迭代

3.1 算法介绍

算法 3.1 QR 迭代算法 (QR Iteration)

```
Set A_1 = A and k = 1
while not convergence do
    A_k = Q_k R_k        % QR 分解
    compute A_{k+1} = R_k Q_k
    k = k + 1
end while
```

在 QR 迭代算法中, 我们有

$$A_{k+1} = R_k Q_k = (Q_k^T Q_k) R_k Q_k = Q_k^T (Q_k R_k) Q_k = Q_k^T A_k Q_k.$$

由这个递推关系可得

$$\begin{aligned} A_{k+1} = Q_k^T A_k Q_k &= Q_k^T Q_{k-1}^T A_{k-1} Q_{k-1} Q_k \\ &= \cdots \\ &= Q_k^T Q_{k-1}^T \cdots Q_1^T A Q_1 \cdots Q_{k-1} Q_k. \end{aligned}$$

记 $\boxed{\tilde{Q}_k = Q_1 \cdots Q_{k-1} Q_k} = \left[\tilde{q}_1^{(k)}, \tilde{q}_2^{(k)}, \dots, \tilde{q}_n^{(k)}\right]$, 则

$$\boxed{A_{k+1} = \tilde{Q}_k^T A \tilde{Q}_k,} \tag{4.1}$$

即 A_{k+1} 与 A 正交相似.

3.2 QR 迭代与幂迭代的关系

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

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

由此递推下去, 即可得

$$\tilde{Q}_k \tilde{R}_k = A^{k-1}\tilde{Q}_1\tilde{R}_1 = A^{k-1}Q_1R_1 = A^k. \tag{4.2}$$

故

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

这说明 QR 迭代与幂迭代有关.

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

因此, 当 k 充分大时, $A\tilde{q}_1^{(k)} \to \lambda_1 \tilde{q}_1^{(k)}$, 此时由 (4.1) 可知, A_{k+1} 的第一列为

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

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

3.3 QR 迭代与反迭代的关系

下面观察 $\tilde{Q}_k$ 的最后一列. 由 (4.1) 可知

$$A\tilde{Q}_k = \tilde{Q}_k A_{k+1} = \tilde{Q}_k Q_{k+1} R_{k+1} = \tilde{Q}_{k+1} R_{k+1},$$

所以有

$$\tilde{Q}_{k+1} = A\tilde{Q}_k R_{k+1}^{-1}.$$

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

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

观察等式两边矩阵的最后一列, 可得

$$\tilde{q}_n^{(k+1)} = c_1 \left(A^T\right)^{-1} \tilde{q}_n^{(k)},$$

其中 c_1 为某个常数. 依此类推, 可知

$$\tilde{q}_n^{(k+1)} = c \left(A^T\right)^{-k} \tilde{q}_n^{(1)},$$

其中 c 为某个常数. 假设 A 的特征值满足 $|\lambda_1| \geq |\lambda_2| \geq \cdots \geq |\lambda_{n-1}| > |\lambda_n| > 0$, 则 λ_n^{-1} 是 $\left(A^T\right)^{-1}$ 的模最大的特征值. 由幂迭代的收敛性可知, $\tilde{q}_n^{(k+1)}$ 收敛到 $\left(A^T\right)^{-1}$ 的模最大特征值 λ_n^{-1} 所对应的特征向量, 即当 k 充分大时, 有

$$\left(A^T\right)^{-1} \tilde{q}_n^{(k+1)} \to \lambda_n^{-1} \tilde{q}_n^{(k+1)}.$$

所以

$$A^T \tilde{q}_n^{(k+1)} \to \lambda_n \tilde{q}_n^{(k+1)}.$$

由 (4.1) 可知, A_{k+1}^T 的最后一列为

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

即 A_{k+1} 的最后一行的最后一个元素收敛到 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.

3.4 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 设正交迭代算法 2.1 和 QR 算法 3.1 中所涉及的 QR 分解都是唯一的. A_k 是由 QR 迭代算法 3.1 生成的矩阵, Z_k 是由正交迭代算法 2.1 (取 $Z_0 = I$) 生成的矩阵, 则有

$$A_{k+1} = Z_k^T A Z_k.$$

证明: 板书.

3.5 QR 迭代的收敛性

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

证明: 板书.

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_demo.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 A_k 的下三角部分中元素, 如果其绝对值小于某个阈值 tol, 则直接将其设为 0, 即

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

这里我们取 $tol = 10^{-6} \max_{1 \le i,j \le n} \{|a_{ij}^{(k)}|\}$, 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
```

3.6 带位移的 QR 迭代

为了加快 QR 迭代的收敛速度, 我们可以采用位移策略和反迭代思想.

算法 3.2 带位移的 QR 迭代算法 (QR Iteration with shift)

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

while not convergence **do**

  Choose a shift σ_k

  $A_k - \sigma_k I = Q_k R_k$ % QR 分解

  Compute $A_{k+1} = R_k Q_k + \sigma_k I$

  $k = k + 1$

end while

我们有

$$\begin{aligned}A_{k+1} = R_kQ_k + \sigma_k I &= (Q_k^TQ_k)R_kQ_k + \sigma_k I \\ &= Q_k^T(A_k - \sigma_k I)Q_k + \sigma_k I \\ &= Q_k^TA_kQ_k.\end{aligned}$$

所以, 带位移的 QR 算法中所得到的所有矩阵 A_k 都与 A 正交相似.

关于位移 σ_k

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

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

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 (见 `Eig_QR_shift.m`). 所有数据与设置与例 3.1 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 $\sigma_k = A_k(n-1,n-1)$. 依此类推, 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_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
```

4 带位移的隐式 QR 迭代

QR 迭代算法中需要考虑的另一个重要问题：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 $\mathcal{O}(n^4)$.

可以接受的运算量: 从 $\mathcal{O}(n^4)$ 减小到 $\mathcal{O}(n^3)$

为了实现这个目标, 我们可借助 Hessenberg 矩阵:

- 首先通过相似变化将 A 转化成一个上 Hessenberg 矩阵
- 对这个 Hessenberg 矩阵实施隐式 QR 迭代.

所谓隐式 QR 迭代, 就是在 QR 迭代中, 不需要进行显式的 QR 分解和矩阵乘积. 这样就可以将每一步的运算量从 $\mathcal{O}(n^3)$ 降低到 $\mathcal{O}(n^2)$. 从而总运算量降到 $\mathcal{O}(n^3)$.

4.1 上 Hessenberg 矩阵

设 $H = (h_{ij}) \in \mathbb{R}^{n\times n}$, 若当 $i > j+1$ 时, 有 $h_{ij} = 0$, 则称 H 为上 Hessenberg 矩阵.

定理 设 $A \in \mathbb{R}^{n\times n}$, 则存在正交矩阵 $Q \in \mathbb{R}^{n\times n}$, 使得 QAQ^T 是上 Hessenberg 矩阵.

下面我们给出具体的转化过程, 采用的工具为 Householder 变换.

我们以一个 5×5 的矩阵 A 为例.

第一步: 令 $Q_1 = \mathrm{diag}(I_{1\times 1}, H_1)$, 其中 H_1 是对应于向量 $A(2:5,1)$ 的 Householder 矩阵. 于是可得

$$Q_1 A = \begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ 0 & * & * & * & * \\ 0 & * & * & * & * \\ 0 & * & * & * & * \end{bmatrix}.$$

由于用 Q_1^T 右乘 $Q_1 A$ 时, 不会改变 $Q_1 A$ 第一列元素的值, 故

$$A_1 \triangleq Q_1 A Q_1^T = \begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ 0 & * & * & * & * \\ 0 & * & * & * & * \\ 0 & * & * & * & * \end{bmatrix}.$$

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

$$Q_2A_1 = \begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ 0 & * & * & * & * \\ 0 & 0 & * & * & * \\ 0 & 0 & * & * & * \end{bmatrix} \quad \text{和} \quad A_2 \triangleq Q_2A_1Q_2^T = \begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ 0 & * & * & * & * \\ 0 & 0 & * & * & * \\ 0 & 0 & * & * & * \end{bmatrix}.$$

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

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

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

下面是将任意一个矩阵转化成上 Hessenberg 矩阵的算法.

算法 4.1 上 Hessenberg 化算法 (Upper Hessenberg Reduction)

Set $Q = I$

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

  compute Householder matrix $H_k = I - \beta_k v_k v_k^T$ with respect to $A(k+1:n, k)$

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

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

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

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.

证明: 板书.

† 若 A 是奇异的, 也可以通过选取适当的 Q, 使得定理 4.2 结论成立.

由这个性质可知, 如果 $A \in \mathbb{R}^{n\times n}$ 是上 Hessenberg 矩阵, 则 QR 迭代中每一步的运算量可大大降低.

Hessenberg 矩阵另一个重要性质: 在 QR 迭代过程中保持不可约性.

定理 设 $A \in \mathbb{R}^{n\times n}$ 是不可约上 Hessenberg 矩阵, 其 QR 分解为 $A = QR$, 则 $\tilde{A} \triangleq RQ$ 也是不可约上 Hessenberg 矩阵.

证明. 留作练习. □

推论 设 $A \in \mathbb{R}^{n\times n}$ 是不可约上 Hessenberg 矩阵, 则在带位移的 QR 迭代中, 所有的 A_k 均是不可约上 Hessenberg 矩阵.

4.2 隐式 QR 迭代

在 QR 迭代中, 我们需要先做 QR 分解 $A_k = Q_k R_k$, 然后再计算 $A_{k+1} = R_k Q_k$. 但事实上, 我们可以将这个过程进一步简化, 即不用计算 A_k 的 QR 分解, 可以直接计算 A_{k+1}. 这就是隐式 QR 迭代.

我们这里考虑不可约的上 Hessenberg 矩阵, 即 A 的下次对角线元素都不为 0. 事实上, 若 A 是可约的, 则 A 就是一个块上三角矩阵, 这时 A 的特征值计算问题就转化成计算两个对角块的特征值问题.

隐式 QR 迭代的理论基础就是下面的隐式 Q 定理.

定理 (Implicit Q Theorem) 设 $H = Q^T AQ \in \mathbb{R}^{n\times n}$ 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).

证明: 板书.

由于 Q_k 的其它列都由 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 Q_k 的第一列相等, 且 $\tilde{Q}_k^T A_k \tilde{Q}_k$ 为上 Hessenberg 矩阵, 则由隐式 Q 定理可知 $\tilde{Q}_k = WQ_k$, 其中 $W = \mathrm{diag}(1, \pm 1, \ldots, \pm 1)$. 于是

$$\tilde{Q}_k^T A_k \tilde{Q}_k = W^T Q_k^T A_k Q_k W = W^T A_{k+1} W.$$

又 $W^T A_{k+1} W$ 与 A_{k+1} 相似, 且对角线元素相等, 而其它元素也至多相差一个符号, 所以不会影响 A_{k+1} 的收敛性, 即下三角元素收敛到 0, 对角线元素收敛到 A 的特征值.

因此在 QR 迭代算法中, 我们可以用 $\tilde{Q}_k^T A_k \tilde{Q}_k$ 代替 $Q_k A_k Q_k^T$. 这就是隐式 QR 迭代的基本思想.

在实际计算中, 我们只需要利用 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 $A_1 = A$ 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

$$G_1^T \triangleq G(1,2,\theta_1) = \begin{bmatrix} c_1 & s_1 & & & \\ -s_1 & c_1 & & & \\ & & 1 & & \\ & & & 1 & \\ & & & & 1 \end{bmatrix},$$

其第一列 $[c_1, s_1, 0, \ldots, 0]^T$ 就是 $A_1 - \sigma_1 I$ 的第一列 $[a_{11} - \sigma_1, a_{21}, 0, \ldots, 0]^T$ 的单位化后的列向量, 这里 σ_1 是位移. 于是有

$$G_1^T A = \begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ 0 & * & * & * & * \\ 0 & 0 & * & * & * \\ 0 & 0 & 0 & * & * \end{bmatrix} \quad \text{和} \quad A^{(1)} \triangleq G_1^T A G_1 = \begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ + & * & * & * & * \\ 0 & 0 & * & * & * \\ 0 & 0 & 0 & * & * \end{bmatrix}.$$

与 A_1 相比较, $A^{(1)}$ 在 $(3,1)$ 位置上多出一个非零元, 我们把它记为 " $+$ ", 并称之为 bulge. 在下面的计算过程中, 我们的目标就是将其 "赶" 出矩阵, 从而得到一个新的上 Hessenberg 矩阵, 即 A_2.

第二步: 为了消去这个 bulge, 我们可以构造 Givens 变换

$$G_2^T \triangleq G(2,3,\theta_2) = \begin{bmatrix} 1 & & & & \\ & c_2 & s_2 & & \\ & -s_2 & c_2 & & \\ & & & 1 & \\ & & & & 1 \end{bmatrix} \text{ 使得 } G_2^T A^{(1)} = \begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ 0 & * & * & * & * \\ 0 & 0 & * & * & * \\ 0 & 0 & 0 & * & * \end{bmatrix}.$$

为了保持与原矩阵的相似性, 需要再右乘 G_2, 所以

$$A^{(2)} \triangleq G_2^T A^{(1)} G_2 = \begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ 0 & * & * & * & * \\ 0 & + & * & * & * \\ 0 & 0 & 0 & * & * \end{bmatrix}.$$

此时, 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

$$G_3^T \triangleq G(3,4,\theta_3) = \begin{bmatrix} 1 & & & & \\ & 1 & & & \\ & & c_3 & s_3 & \\ & & -s_3 & c_3 & \\ & & & & 1 \end{bmatrix} \text{ 使得 } G_3^T A^{(2)} = \begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ 0 & * & * & * & * \\ 0 & 0 & * & * & * \\ 0 & 0 & 0 & * & * \end{bmatrix}.$$

这时

$$A^{(3)} \triangleq G_3^T A^{(2)} G_3 = \begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ 0 & * & * & * & * \\ 0 & 0 & * & * & * \\ 0 & 0 & + & * & * \end{bmatrix}.$$

于是, bugle 又从 $(4,2)$ 位置又被 “赶” 到 $(5,3)$ 位置.

第四步: 再次构造 Givens 变换

$$G_4^T \triangleq G(4,5,\theta_4) = \begin{bmatrix} 1 & & & & \\ & 1 & & & \\ & & 1 & & \\ & & & c_4 & s_4 \\ & & & -s_4 & c_4 \end{bmatrix} \text{ 使得 } G_4^T A^{(3)} = \begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ 0 & * & * & * & * \\ 0 & 0 & * & * & * \\ 0 & 0 & 0 & * & * \end{bmatrix}$$

于是

$$A^{(4)} \triangleq G_4^T A^{(3)} G_4 = \begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ 0 & * & * & * & * \\ 0 & 0 & * & * & * \\ 0 & 0 & 0 & * & * \end{bmatrix}.$$

现在, bulge 已经被 “赶” 出矩阵.

此时, 有

$$A^{(4)} = G_4^T G_3^T G_2^T G_1^T A_1 G_1 G_2 G_3 G_4 = \tilde{Q}_1^T A_1 \tilde{Q}_1,$$

其中 $\tilde{Q}_1 = G_1 G_2 G_3 G_4$. 通过直接计算可知, $\tilde{Q}_1$ 的第一列为 $[c_1, s_1, 0, 0, 0]^T$, 即 $A_1 - \sigma_1 I$ 的第一列的单位化. 根据隐式 Q 定理, $A_2 \triangleq A^{(4)} = \tilde{Q}_1^T A_1 \tilde{Q}_1$ 就是我们所需要的矩阵.

如果 $A \in \mathbb{R}^{n \times n}$ 是上 Hessenberg 矩阵, 则使用上面的算法, 带位移 QR 迭代中每一步的运算量为 $6n^2 + O(n)$.

4.3 位移的选取

通常, 位移越离特征值越近, 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, 这种位移选取方法就可能失效.

双位移策略

假设 $\sigma \in \mathbb{C}$ 是 A 的某个复特征值 λ 的一个很好的近似, 则其共轭 $\bar{\sigma}$ 也应该是 $\bar{\lambda}$ 的一个很好的近似. 因此我们可以考虑双位移策略, 即先以 σ 为位移迭代一次, 然后再以 $\bar{\sigma}$ 为位移迭代一次, 如此不断交替进行迭代.

这样就有

$$\begin{aligned} A_1 - \sigma I &= Q_1 R_1, \\ A_2 &= R_1 Q_1 + \sigma I, \\ A_2 - \bar{\sigma} I &= Q_2 R_2, \\ A_3 &= R_2 Q_2 + \bar{\sigma} I. \end{aligned} \tag{4.5}$$

容易验证

$$A_3 = Q_2^T A_2 Q_2 = Q_2^* Q_1^* A_1 Q_1 Q_2 = Q^* A_1 Q,$$

其中 $Q = Q_1 Q_2$.

我们注意到 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, 使的 $Q = Q_1Q_2$ 是实正交矩阵.

引理 在双位移 QR 迭代 (4.3) 中, 我们可以选取酉矩阵 Q_1 和 Q_2 使得 $Q = Q_1Q_2$ 是实矩阵.

证明: 板书.

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 $Q = Q_1Q_2$ 是实矩阵, 从而 $A_3 = Q^T A_1 Q$ 也是实矩阵. 因此我们无需计算 A_2, 而是直接由 A_1 计算出 A_3.

具体计算还是根据隐式 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 $A_3 = Q^T A_1 Q$ 是上 Hessenberg 矩阵即可.

由于 $A_1^2 - 2\mathrm{Re}(\sigma)A_1 + |\sigma|^2 I$ 的第一列为

$$\begin{bmatrix} a_{11}^2 + a_{12}a_{21} - 2\mathrm{Re}(\sigma)a_{11} + |\sigma|^2 \\ a_{21}(a_{11} + a_{22} - 2\mathrm{Re}(\sigma)) \\ a_{21}a_{32} \\ 0 \\ \vdots \end{bmatrix}. \tag{4.6}$$

所以 Q 的第一列是上述向量的单位化.

其它过程可以通过隐式 QR 迭代来实现. 但此时的 “bulge” 是一个 2×2 的小矩阵. 因此, 在双位移隐式 QR 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 $H_1 = \begin{bmatrix} \tilde{H}_1^T & 0 \\ 0 & I_{3\times 3} \end{bmatrix}$, 其中 $\tilde{H}_1 \in \mathbb{R}^{3\times 3}$, 使得其第一列与向量 (4.4) 平行. 于是有

$$H_1^T A = \begin{bmatrix} * & * & * & * & * & * \\ * & * & * & * & * & * \\ + & * & * & * & * & * \\ 0 & 0 & * & * & * & * \\ 0 & 0 & 0 & * & * & * \\ 0 & 0 & 0 & 0 & * & * \end{bmatrix} \quad \text{和} \quad A^{(1)} \triangleq H_1^T A H_1 = \begin{bmatrix} * & * & * & * & * & * \\ * & * & * & * & * & * \\ + & * & * & * & * & * \\ + & + & * & * & * & * \\ 0 & 0 & 0 & * & * & * \\ 0 & 0 & 0 & 0 & * & * \end{bmatrix}.$$

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, 即 A_1.

第二步: 令 $H_2 = \begin{bmatrix} I_{1\times 1} & 0 & 0 \\ 0 & \tilde{H}_2^T & 0 \\ 0 & 0 & I_{3\times 3} \end{bmatrix}$, 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

$$H_2^T A^{(1)} = \begin{bmatrix} * & * & * & * & * & * \\ * & * & * & * & * & * \\ 0 & * & * & * & * & * \\ 0 & + & * & * & * & * \\ 0 & 0 & 0 & * & * & * \\ 0 & 0 & 0 & 0 & * & * \end{bmatrix} \text{ 和 } A^{(2)} \triangleq H_2^T A^{(1)} H_2 = \begin{bmatrix} * & * & * & * & * & * \\ * & * & * & * & * & * \\ 0 & * & * & * & * & * \\ 0 & + & * & * & * & * \\ 0 & + & + & * & * & * \\ 0 & 0 & 0 & 0 & * & * \end{bmatrix}.$$

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 $H_3 = \begin{bmatrix} I_{2\times 2} & 0 & 0 \\ 0 & \tilde{H}_3^T & 0 \\ 0 & 0 & I_{1\times 1} \end{bmatrix}$, 其中 $\tilde{H}_3 \in \mathbb{R}^{3\times 3}$ 是对应于 $A(3:5,2)$ 的 Householder 变换, 使得

$$H_3^T A^{(2)} = \begin{bmatrix} * & * & * & * & * & * \\ * & * & * & * & * & * \\ 0 & * & * & * & * & * \\ 0 & 0 & * & * & * & * \\ 0 & 0 & + & * & * & * \\ 0 & 0 & 0 & 0 & * & * \end{bmatrix} \text{ 和 } A^{(3)} \triangleq H_3^T A^{(2)} H_3 = \begin{bmatrix} * & * & * & * & * & * \\ * & * & * & * & * & * \\ 0 & * & * & * & * & * \\ 0 & 0 & * & * & * & * \\ 0 & 0 & + & * & * & * \\ 0 & 0 & + & + & * & * \end{bmatrix}.$$

此时, bugle 又被向右下角方向 “赶” 了一个位置.

第四步: 令 $H_4 = \begin{bmatrix} I_{3\times 3} & 0 \\ 0 & \tilde{H}_4^T \end{bmatrix}$, 其中 $\tilde{H}_4 \in \mathbb{R}^{3\times 3}$ 是对应于 $A(4:6,3)$ 的 Householder 变换, 使得

$$H_4^T A^{(3)} = \begin{bmatrix} * & * & * & * & * & * \\ * & * & * & * & * & * \\ 0 & * & * & * & * & * \\ 0 & 0 & * & * & * & * \\ 0 & 0 & 0 & * & * & * \\ 0 & 0 & 0 & + & * & * \end{bmatrix} \text{ 和 } A^{(4)} \triangleq H_4^T A^{(3)} H_4 = \begin{bmatrix} * & * & * & * & * & * \\ * & * & * & * & * & * \\ 0 & * & * & * & * & * \\ 0 & 0 & * & * & * & * \\ 0 & 0 & 0 & * & * & * \\ 0 & 0 & 0 & + & * & * \end{bmatrix}.$$

第五步: 只需构造一个 Givens 变换 $G_5 = \begin{bmatrix} I_{4\times 4} & 0 \\ 0 & G(4,5,\theta)^T \end{bmatrix}$, 使得

$$
G_5^T A^{(4)} = \begin{bmatrix} * & * & * & * & * & * \\ * & * & * & * & * & * \\ 0 & * & * & * & * & * \\ 0 & 0 & * & * & * & * \\ 0 & 0 & 0 & * & * & * \\ 0 & 0 & 0 & 0 & * & * \end{bmatrix} \text{ 和 } A^{(5)} \triangleq G_5^T A^{(4)} G_5 = \begin{bmatrix} * & * & * & * & * & * \\ * & * & * & * & * & * \\ 0 & * & * & * & * & * \\ 0 & 0 & * & * & * & * \\ 0 & 0 & 0 & * & * & * \\ 0 & 0 & 0 & 0 & * & * \end{bmatrix}.
$$

现在, bulge 已经被全部消除, 且

$$
A^{(5)} = Q^T A Q,
$$

其中 $Q = H_1 H_2 H_3 H_4 G_5$. 通过直接计算可知, Q 的第一列即为 H_1 的第一列, 也就是向量 (4.4) 的单位化. 根据隐式 Q 定理, $A^{(5)} = Q^T A Q$ 就是双位移 QR 迭代 (4.3) 中的 A_3.

位移的具体选取问题

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.

一般来说, 采用 Francis 位移的 QR 迭代会使得迭代矩阵的右下角收敛到一个上三角矩阵 (两个实特征值) 或一个 2×2 的矩阵 (一对共轭特征值), 而且通常会有二次渐进收敛性. 在实际计算中, 计算一个特征值一般平均只需迭代两步.

注: 判断 QR 迭代算法是否收敛主要是看 $A_k(n-1,n-2)$ (或 $A_k(n,n-1)$) 是否趋向于 0.

需要指出的是, QR 迭代并不是对所有矩阵都收敛. 例如:

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

对于上面的矩阵, 采用 Francis 位移的 QR 迭代算法无效.

另外, 也可以考虑多重位移策略, 参见 [63].

4.4 收缩 Deflation

收缩 (deflation) 技术是实用 QR 迭代中的一个非常重要概念.

在隐式 QR 迭代过程中, 当矩阵 A_{k+1} 的某个下次对角线元素 $a_{i+1,i}$ 很小时, 我们可以将其设为 0.

由于 A_{k+1} 是上 Hessenberg 矩阵, 这时 A_{k+1} 就可以写成分块上三角形式, 其中两个对角块都是上 Hessenberg 矩阵.

因此我们可以将隐式 QR 迭代作用在这两个规模相对较小的矩阵上, 从而可以大大节约运算量.

5 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 对应于 λ 的特征向量 y 即可.

因为 R 的对角线元素即为 A 的特征值, 所以可设 $\lambda = R(i,i)$. 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

$$(R_{11} - \lambda I)y_1 + R_{12}y_2 + R_{13}y_3 = 0, \tag{4.7}$$

$$R_{23}y_3 = 0, \tag{4.8}$$

$$(R_{33} - \lambda I)y_3 = 0, \tag{4.9}$$

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, 则计算方法类似. 但如果 A 有复特征值, 则需要利用实 Schur 标准型, 计算较复杂.

6 广义特征值问题

6.1 广义 Schur 分解

6.2 QZ 迭代

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

$$Ax = \lambda Bx, \tag{4.10}$$

则称 λ 为矩阵对 (A, B) 的特征值, x 为相应的特征向量. 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

$$B^{-1}Ax = \lambda x \quad \text{或} \quad AB^{-1}y = \lambda y,$$

其中 $y = Bx$.

容易看出, λ 是 (A,B) 的一个特征值当且仅当

$$\det(A-\lambda B)=0. \tag{4.11}$$

若 (4.9) 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 (4.9) 是一个 n 次多项式, 因此恰好有 n 个特征值. 当 B 奇异时, 特征方程 (4.9) 的次数低于 n, 因此方程的解的个数小于 n. 但是, 注意到 $\lambda\neq 0$ 是 (A,B) 的特征值当且仅当 $\mu=\frac{1}{\lambda}$ 是 (B,A) 的特征值. 因此, 当 B 奇异时, $\mu=0$ 是 (B,A) 的特征值, 于是我们自然地把 $\lambda=\frac{1}{\mu}=\infty$ 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.

6.1 广义 Schur 分解

广义 Schur 分解 是矩阵对在酉等价变化下的最简形式.

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

$$Q^*AZ = R_A, \quad Q^*BZ = R_B, \tag{4.12}$$

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 [78]. □

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 (广义实 Schur 分解) 设 $A, B \in \mathbb{R}^{n\times n}$, 则存在正交矩阵 $Q, Z \in \mathbb{R}^{n\times n}$, 使得

$$Q^T AZ = T_A, \quad Q^T BZ = T_B, \tag{4.13}$$

其中 $T_A, T_B \in \mathbb{R}^{n\times n}$ 都是拟上三角矩阵.

证明. 参见 [78]. □

6.2 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 [42, 78].

课后习题

见讲义

参考文献

[1] J. O. Aasen, On the reduction of a symmetric matrix to tridiagonal form, BIT, 11 (1971), 233–242.

[2] M. Arioli, V. Pták, and Z. Strakoš, Krylov sequences of maximal length and convergence of GMRES, BIT, 38 (1998), 636–643.

[3] C. Ashcraft, R. G. Grimes and J. G. Lewis, Accurate symmetric indefinite linear equation solvers, SIAM Journal on Matrix Analysis and Applications, 20 (1998), 513–561.

[4] O. Axelsson, Iterative Solution Methods, Cambridge University Press, Cambridge, 1994.

[5] Z.-Z. Bai, G. H. Golub and M. K. Ng, Hermitian and skew-Hermitian splitting methods for non-Hermitian positive definite linear systems, SIAM Journal on Matrix Analysis and Applications, 24 (2003), 603–626.

[6] M. Van Barel, G. Heinig and P. Kravanja, A stabilized superfast solver for nonsymmetric toeplitz systems, SIAM Journal on Matrix Analysis and Applications, 23 (2001), 494–510. FORTRAN code http://people.cs.kuleuven.be/~marc.vanbarel/software/

[7] R. Barrett, et.al, Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods, SIAM, 1994. (http://www.netlib.org/templates/index.html)

[8] V. Barwell and A. George, A comparison of algorithms for solving symmetric indefinite systems of linear equations, ACM Transactions on Mathematical Software, 2 (1976), 242–251.

[9] Åke Björck, Solving linear least square problems by Gram-Schmidt orthogonalization, BIT, 7 (1967), 1–21.

[10] Åke Björck, Numerical Methods for Least Squares Problems, SIAM, Philadelphia, PA, 1996.

[11] J. R. Bunch, Analysis of the diagonal pivoting method, SIAM Journal on Numerical Analysis, 8 (1971), 656–680.

[12] J. R. Bunch and L. Kaufman, Some stable methods for calculating inertia and solving symmetric linear systems, Mathematics of Computation, 31 (1977), 163–179.

[13] J. R. Bunch and B. N. Parlett, Direct methods for solving symmetric indefinite systems of linear equations, SIAM Journal on Numerical Analysis, 8 (1971), 639–655.

[14] R. P. Brent, Stability of fast algorithms for structured linear systems, http://arxiv.org/pdf/1005.0671v1.pdf, 2010.

[15] G. Codevico, G. Heinig and M. Van Barel, A superfast solver for real symmetric Toeplitz systems using real trigonometric transformations, Numer. Linear Algebra Appl., 12 (2005), 699–713. MATLAB code http://people.cs.kuleuven.be/~marc.vanbarel/software/

[16] J. J. M. Cuppen, A Divide and Conquer Method for the Symmetric Tridiagonal Eigenproblem, Numerische Mathematik, 36 (1981), 177–195.

[17] J. W. Demmel, Applied Numerical Linear Algebra, SIAM, Philadelphia, PA, 1997.

[18] J. J. Dongarra, I. S. Duff, D. C. Sorensen and H. A. van der Vorst, Numerical Linear Algebra for High-Performance Computers, SIAM, Philadelphia, PA, 1998.

[19] Z. Drmač and K. Veselić, New fast and accurate jacobi SVD algorithm. I SIAM Journal on Matrix Analysis and Applications, 29 (2008), 1322–1342.

[20] Z. Drmač and K. Veselić, New fast and accurate jacobi SVD algorithm. II SIAM Journal on Matrix Analysis and Applications, 29 (2008), 1343–1362.

[21] V. Faber, W. Joubert, E. Knill and T. Manteuffel, Minimal residual method stronger than polynomial preconditioning, SIAM Journal on Matrix Analysis and Applications, 17 (1996), 707–729.

[22] K. Fernando and B. Parlett, Accurate singular values and differential qd algorithms, Numerische Mathematik, 67 (1994), 191–229.

[23] B. Fischer, Polynomial based iteration methods for symmetric linear systems, Wiley-Teubner Series Advances in Numerical Mathematics, John Wiley & Sons Ltd., Chichester, 1996.

[24] N. Gastinel, Linear Numerical Analysis, Kershaw Publishing, London, 1083.

[25] G. H. Golub, History of numerical linear algebra: A personal view, Stanford, 2007. Available at http://forum.stanford.edu/events/2007slides/plenary/history-revised-2007-03-19-golub.pdf

[26] G. H. Golub and W. Kahan, Calculating the singular values and pseudo-inverse of a matrix, SIAM Journal on Numerical Analysis, Series B, 2 (1965), 205–224.

[27] G. H. Golub and C. F. Van Loan, Matrix Computations, The 4th Editon, The Johns Hopkins University Press, Baltimore, MD, 2013.

[28] A. Greenbaum and L. Gurvits, Max-min properties of matrix factor norms, SIAM Journal on Scientific Computing, 15 (1994), 348–358.

[29] A. Greenbaum, V. Pták and Z. Strakoš, Any nonincreasing convergence curve is possible for GMRES, SIAM Journal on Matrix Analysis and Applications, 17 (1996), 465–469.

[30] A. Greenbaum and Z. Strakoš, Matrices that generate the same Krylov residual spaces, in Recent Advances in Iterative Methods, vol. 60 of IMA Vol. Math. Appl., Springer, New York, 1994, pp. 95–118.

[31] M. Gu and S. C. Eisenstat, A stable algorithm for the rank-1 modification of the symmetric eigenproblem, SIAM Journal on Matrix Analysis and Applications, 15 (1994), 1266–1276.
[32] M. Gu and S. C. Eisenstat, A Divide-and-Conquer algorithm for the bidiagonal SVD, SIAM Journal on Matrix Analysis and Applications, 16 (1995), 79–92.
[33] M. Gu and S. C. Eisenstat, A Divide-and-Conquer algorithm for the symmetric tridiagonal eigenproblem, SIAM Journal on Matrix Analysis and Applications, 16 (1995), 172–191.
[34] A. Hadjidimos, Accelerated overrelaxation method, Mathematics of Computation, 32 (1978), 149–157.
[35] Nicholas J. Higham, Accuracy and Stability of Numerical Algorithms, Second Edition, SIAM, Philadelphia, 2002.
[36] L. Hogben, Handbook of Linear Algebra, 2nd, CRC Press, 2014.
[37] R.A. Horn and C.R. Johnson, Matrix Analysis, Cambridge University Press, New York, 1985.
[38] R.A. Horn and C.R. Johnson, Topics in Matrix Analysis, Cambridge University Press, New York, 1991.
[39] W. Joubert, A robust GMRES-based adaptive polynomial preconditioning algorithm for nonsymmetric linear systems, SIAM Journal on Scientific Computing, 15 (1994), 427–439.

[40] W. Kahan Numerical Linear Algebra, Canadian Math. Bull., 9 (1966), 757–801.

[41] C. T. Kelley, Iterative Methods for Linear and Nonlinear Equations, SIAM, Philadelphia, 1995.

[42] D. Kressner, Numerical Methods for General and Structured Eigenvalue Problems, Lecture Notes in Computational Sciences and Engineering 46, Springer-Verlag, 2005.

[43] R. Lehoucq, Analysis and Implementation of an Implicitly Restarted Arnoldi Iteration, Ph.D. thesis, Rice University, Houston, TX, 1995.

[44] N. Levinson The Wiener RMS (root mean square) error criterion in filter design and prediction, J. Math. Phys., 25 (1946), 261–278.

[45] J. Liesen, Computable convergence bounds for GMRES, SIAM Journal on Matrix Analysis and Applications, 21 (2000), 882–903.

[46] J. Liesen and P. Tichý, Convergence analysis of Krylov subspace methods, GAMM-Mitteilungen, 27 (2004), 153–173.

[47] E. H. Moore, On the reciprocal of the general algebraic matrix, Bull. Amer. Math. Soc., 26 (1920), 394–395.

[48] Christopher C. Paige, Miroslav Rozložník and Zdeněk Strakoš, Modified Gram–Schmidt (MGS), least squares, and backward stability of MGS-GMRES, SIAM Journal on Matrix Analysis and Applications, (28) 2006, 264−-284.

[49] B. N. Parlett, The Symmetric Eigenvalue Problem, The 2nd Edition, SIAM, Philadelphia, PA, 1998.

[50] D. W. Peaceman and H. H. Rachford, Jr., The numerical solution of parabolic and elliptic differential equations, Journal of the Society for Industrial and Applied Mathematics, 3 (1955), 28–41.

[51] R. Penrose, A generalized inverse for matrices, Proc. Cambridge Philos. Soc., 51 (1955), 406–413.

[52] V. Britanak, P. Yip and K. Rao, Discrete Cosine and Sine Transforms: General properties, Fast algorithms and Integer Approximations, Academic Press, 2007.

[53] J. Rutter, A Serial Implementation of Cuppen'S Divide and Conquer Algorithm for the Symmetric Eigenvalue Problem, Master's Thesis, University of California, 1994.

[54] Y. Saad and M. H. Schultz, GMRES: A generalized minimal residual method for solving nonsymmetric linear systems, SIAM Journal on Scientific & Statistical Computing, 7 (1986), 856–869.

[55] Y. Saad, Numerical Methods for Large Eigenvalue Problems: Theory and Algorithms, Manchester University Press, Manchester, UK, 1992.

[56] D. Sorensen, Implicit application of polynomial filters in a k-step Arnoldi method, SIAM Journal on Matrix Analysis and Applications, 13 (1992), 357–385.

[57] G. W. Stewart, Matrix Algorithms, Vol I: Basic Decomposition, SIAM, Philadelphia, PA, 1998.

[58] G. W. Stewart, Matrix Algorithms, Vol II: Eigensystems, SIAM, Philadelphia, PA, 2001.

[59] G. W. Stewart and Ji-guang Sun, Matrix Perturbation Theory, Academic Press, New York, 1990.

[60] M. Stewart, Fast algorithms for structured matrix computations, in Handbook of Linear Algebra, 2nd, section 62, CRC Press, 2014.

[61] L. N. Trefethen and D. Bau, Numerical Linear Algebra, SIAM, Philadelphia, PA, 1997.

[62] L. N. Trefethen, Numerical Analysis, in Princeton Companion to Mathematics, Edited by T. Gowers, J. Barrow-Green and I. Leader, Princeton University Press, 2008.

[63] D. S. Watkins, The Matrix Eigenvalue Problem: GR and Krylov Subspace Methods, SIAM, Philadelphia, 2007.

[64] D. S. Watkins and L. Elsner, Convergence of algorithms of decomposition type for the eigenvalue problem, Linear Algebra and its Applications, 143 (1991), 19–47.

[65] J. H. Wilkinson, The Algebraic Eigenvalue Problem, Clarendon Press, Oxford University, Oxford, 1965.

[66] R. S. Varga, Matrix Iterative Analysis, Prentice-Hall, Englewood Cliffs, NJ, 1962. 2nd edition, Springer-Verlag, Berlin, 2000.

[67] D. M. Young, Iterative Methods for Solving Partial Difference Equations of Elliptic Type, PhD thesis, Harward University, 1950.

[68] D. M. Young, Iterative Solution of Large Linear Systems, Academic Press, New York, 1971.

[69] 北京大学数学系, 高等代数 (第三版), 高等教育出版社, 2003.

[70] 陈志明, 科学计算：科技创新的第三种方法, 中国科学院院刊, 27 (2012), 161–166.

[71] 戴华, 矩阵论, 科学出版社, 2001.

[72] 胡家赣, 线性代数方程组的迭代解法, 科学出版社, 1991.

[73] 蒋尔雄, 矩阵计算, 科学出版社, 2008.

[74] 李大明, 数值线性代数, 清华大学出版社.

[75] 孙继广, 矩阵扰动分析, 科学出版社, 北京, 2001.

[76] 魏木生, 广义最小二乘问题的理论与计算, 科学出版社, 北京, 2006.

[77] 徐树方, 矩阵计算的理论与方法, 北京大学出版社, 北京, 1995.

[78] 徐树方, 钱江, 矩阵计算六讲, 高等教育出版社, 北京, 2011.