

第四讲 非对称特征值问题

- 1 幂迭代方法
- 2 反迭代方法
- 3 QR 迭代方法
- 4 带位移的隐式 QR 迭代
- 5 应用：多项式求根 *

非对称矩阵特征值/特征向量计算

基本约定 1: $A \in \mathbb{R}^{n \times n}$ 、非对称、稠密

基本约定 2: $|\lambda_1| \geq |\lambda_2| \geq \dots \geq |\lambda_n|$

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）
- QR 迭代方法（带位移的隐式 QR 迭代）



关于稠密矩阵特征值计算的参考资料

- J. H. Wilkinson, **The Algebraic Eigenvalue Problem**, 1965
- B. N. Parlett, **The Symmetric Eigenvalue Problem**, 2nd Eds., 1998
- G. W. Stewart, **Matrix Algorithms, Vol II: Eigensystems**, 2001
- G. H. Golub and C. F. Van Loan, **Matrix Computations**, 2013
- Z. Bai, J. Demmel, J. Dongarra, A. Ruhe, and H. van der Vorst, 2000
Templates for the Solution of Algebraic Eigenvalue Problems: A Practical Guide
- P. Arbenz, The course 252-0504-00 G,
Numerical Methods for Solving Large Scale Eigenvalue Problems, 2018.
(该课程的主页, 有电子讲义)



1 | 幂迭代方法

幂迭代 是计算特征值和特征向量的一种简单易用的算法。

虽然简单,但它却建立了计算特征值和特征向量的一个基本框架。

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: $x^{(k+1)} = y^{(k+1)} / \|y^{(k+1)}\|_2$
 - 6: $\mu_{k+1} = (x^{(k+1)}, Ax^{(k+1)})$ % 内积
 - 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

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

假设 2: $|\lambda_1| > |\lambda_2| \geq |\lambda_3| \geq \dots \geq |\lambda_n|$.



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

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

我们假定 $\alpha_1 \neq 0$, 即 $x^{(0)}$ 不属于 $\text{span}\{v_2, v_3, \dots, v_n\}$

(由于 $x^{(0)}$ 是随机选取的, 从概率意义上讲, 这个假设通常是成立的).

于是我们可得

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



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

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

故当 k 趋向于无穷大时, 向量

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

收敛到 $e_1 = [1, 0, \dots, 0]^T$.

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



幂迭代的不足:

- 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 \iff 尽可能地减小 $|\lambda_2/\lambda_1|$

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

其中第一个条件保证最后所求得特征值是我们所需要的, 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



2 | 反迭代方法

将幂迭代作用到 A^{-1} 上, 从而计算 A 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 a scalar σ and an initial vector $x^{(0)}$ with $\|x^{(0)}\|_2 = 1$
 - 2: set $k = 0$
 - 3: **while** not convergence **do**
 - 4: $y^{(k+1)} = (A - \sigma I)^{-1}x^{(k)}$
 - 5: $x^{(k+1)} = y^{(k+1)} / \|y^{(k+1)}\|_2$
 - 6: $\mu_{k+1} = (x^{(k+1)}, Ax^{(k+1)})$
 - 7: $k = k + 1$
 - 8: **end while**
-

 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。



👉 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:

- (1) 若 σ 与某个特征值 λ_k 非常接近, 则反迭代的收敛速度非常快.
- (2) 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:

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



Rayleigh 商迭代

出发点 使得位移 σ 尽可能地接近所求的特征值。

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 商为位移的反迭代, 简称 RQI

理由: μ_k 会逐渐收敛到某个特征值 \leftarrow 幂迭代的收敛性



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 $\sigma = (x^{(0)})^* Ax^{(0)}$
 - 4: **while** not converge **do**
 - 5: $y^{(k+1)} = (A - \sigma I)^{-1} x^{(k)}$
 - 6: $x^{(k+1)} = y^{(k+1)} / \|y^{(k+1)}\|_2$
 - 7: $\mu_{k+1} = (x^{(k+1)}, Ax^{(k+1)})$
 - 8: $\sigma = \mu_{k+1}$
 - 9: $k = k + 1$
 - 10: **end while**
-



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.

 RQI 通常应用于 **三对角矩阵** 的特征值计算.



3 | QR 迭代方法

- 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 迭代的收敛性
- 3.5 带位移的 QR 迭代



3.1 算法介绍

基本思想 通过一系列正交相似变换, 将 A 转化为 (拟) 上三角形式

算法 3.1 QR 迭代算法 (QR Iteration)

- 1: Set $A_1 = A$ and $k = 1$
 - 2: **while** not convergence **do**
 - 3: $[Q_k, R_k] = \text{qr}(A_k)$ % QR 分解
 - 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

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

由这个递推关系可得

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

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

$$A_{k+1} = \tilde{Q}_k^T A \tilde{Q}_k \quad (4.1)$$

即 A_{k+1} 与 A 正交相似.



3.2 QR 迭代与幂迭代的关系

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

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

由此递推下去, 即可得

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

故

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



假设 $|\lambda_1| > |\lambda_2| \geq \dots \geq |\lambda_n|$, 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.



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 $A_{k+1} = \tilde{Q}_k^T A \tilde{Q}_k$ 可知, A_{k+1} 的第一列

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

结论

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

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

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

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

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

观察等式两边矩阵的最后一列, 可得

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

依此类推, 可知

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



假定 $|\lambda_{n-1}| > |\lambda_n| > 0$, 则 λ_n^{-1} 是 $(A^\top)^{-1}$ 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

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

所以

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

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

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

结论

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 迭代的收敛性

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

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

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

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



A₁₂ =

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

A₁₃ =

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

A₂₂ =

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



A₂₈ =

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



3.5 带位移的 QR 迭代

可以采用 **位移策略** 和 **反迭代** 思想来加快 QR 迭代的收敛速度.

算法 3.2 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: $[Q_k, R_k] = \text{qr}(A_k - \sigma_k I)$ % QR 分解
 - 5: Compute $A_{k+1} = R_k Q_k + \sigma_k I$
 - 6: $k = k + 1$
 - 7: **end while**
-



正交相似性

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



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 (见 [Eig_QR_shift.m](#)).

所有数据和设置与例 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 $\sigma_k = A_k(n - 1, n - 1)$.

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

4.1 上 Hessenberg 矩阵

4.2 隐式 QR 迭代

4.3 位移的选取

4.4 收缩 Deflation



直接实施 QR 方法的困难: 运算量太大

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 $O(n^3)$



实现方法: 两步走

- (1) 首先通过相似变化将 A 转化成一个 **上 Hessenberg 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, 而是通过特殊手段实现从 A_k 到 A_{k+1} 的迭代, 并且将运算量控制在 $\mathcal{O}(n^2)$ 量级, 从而将总运算量降到 $\mathcal{O}(n^3)$.



4.1 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^T 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 $Q_1 = \text{diag}(I_{1 \times 1}, H_1)$, 其中 H_1 是对应于向量 $A(2 : 5, 1)$ 的 Householder 矩阵. 于是可得

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

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

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



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

$$Q_2 A_1 = \begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ 0 & * & * & * & * \\ 0 & 0 & * & * & * \\ 0 & 0 & * & * & * \end{bmatrix} \quad \text{和} \quad A_2 \triangleq Q_2 A_1 Q_2^T = \begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ 0 & * & * & * & * \\ 0 & 0 & * & * & * \\ 0 & 0 & * & * & * \end{bmatrix}.$$



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

$$Q_3 A_2 = \begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ 0 & * & * & * & * \\ 0 & 0 & * & * & * \\ 0 & 0 & 0 & * & * \end{bmatrix} \quad \text{和} \quad A_3 \triangleq Q_3 A_2 Q_3^T = \begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ 0 & * & * & * & * \\ 0 & 0 & * & * & * \\ 0 & 0 & 0 & * & * \end{bmatrix}.$$

这时, 我们就将 A 转化成一个上 Hessenberg 矩阵, 即 $Q A Q^T = A_3$ 其中 $Q = Q_3 Q_2 Q_1$ 是正交矩阵, A_3 是上 Hessenberg 矩阵.



上 Hessenberg 化算法

算法 4.1 上 Hessenberg 化算法 (Upper Hessenberg Reduction)

- 1: Set $Q = I$
- 2: **for** $k = 1$ to $n - 2$ **do**
- 3: compute Householder matrix H_k with respect to $A(k + 1 : n, k)$
- 4: $A(k + 1 : n, k : n) = H_k \cdot A(k + 1 : n, k : n)$
 $= A(k + 1 : n, k : n) - \beta_k v_k (v_k^\top A(k + 1 : n, k : n))$
- 5: $A(1 : n, k + 1 : n) = A(1 : n, k + 1 : n) \cdot H_k^\top$
 $= A(1 : n, k + 1 : n) - \beta_k A(1 : n, k + 1 : n) v_k v_k^\top$
- 6: $Q(k + 1 : n, k : n) = H_k \cdot Q(k + 1 : n, k : n)$
 $= Q(k + 1 : n, k : n) - \beta_k v_k (v_k^\top Q(k + 1 : n, k : n))$
- 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. 这样, 在进行 QR 分解时, 运算量可大大降低.



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 $a_{i+1,i} \neq 0, i = 1, 2, \dots, n-1$. 设其 QR 分解为 $A = QR$, 则 $\tilde{A} \triangleq RQ$ 的下次对角线元素也都非零. (证明留作练习)

👉 我们只需考虑不可约情形, 因此 A 的下次对角线均非零.

推论 设 $A \in \mathbb{R}^{n \times n}$ 是上 Hessenberg 矩阵且下次对角线元素均非零, 则在带位移的 QR 迭代中, 所有的 A_k 的下次对角线元素均非零.



4.2 隐式 QR 迭代

在 QR 迭代中, 先做 QR 分解 $A_k = Q_k R_k$, 然后计算 $A_{k+1} = R_k Q_k$.

但事实上, 我们可以将这个过程进行简化, 即在不对 A_k 进行 QR 分解的前提下, 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 (Implicit Q Theorem) 设 $H = Q^T A Q \in \mathbb{R}^{n \times n}$ 是一个不可约上 Hessenberg 矩阵, 其中 $Q \in \mathbb{R}^{n \times n}$ 是正交矩阵, 则 Q 的第 2 至第 n 列均由 Q 的第一列所唯一确定 (可相差一个符号). (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 $\tilde{Q}_k = W Q_k$, 其中 $W = \text{diag}(1, \pm 1, \dots, \pm 1)$. 于是

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

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

在 QR 迭代算法中, 如果我们直接令 $A_{k+1} = \tilde{Q}_k^T A_k \tilde{Q}_k$, 则其收敛性与原 QR 迭代算法没有任何区别! 这就是隐式 QR 迭代的基本思想.

由于 A 是上 Hessenberg 矩阵, 因此实际计算中只需 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

$$G_1^T \triangleq G(1, 2, \theta_1) = \begin{bmatrix} c_1 & s_1 & & & \\ -s_1 & c_1 & & & \\ & & I_3 & & \end{bmatrix} \quad (c_1, s_1 \text{ 待定})$$

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

$$G_2^T \triangleq G(2, 3, \theta_2) = \begin{bmatrix} 1 & & & & \\ & c_2 & s_2 & & \\ & -s_2 & c_2 & & \\ & & & I_2 & \\ & & & & & \end{bmatrix} \quad \text{使得 } G_2^T A^{(1)} = \begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ 0 & * & * & * & * \\ 0 & 0 & * & * & * \\ 0 & 0 & 0 & * & * \end{bmatrix} .$$

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, bulge 从 (3, 1) 位置被“赶”到 (4, 2) 位置.



第三步: 与第二步类似, 构造 Givens 变换

$$G_3^T \triangleq G(3, 4, \theta_3) = \begin{bmatrix} I_2 & & & & \\ & c_3 & s_3 & & \\ & -s_3 & c_3 & & \\ & & & & \\ & & & & 1 \end{bmatrix} \quad \text{使得 } G_3^T A^{(2)} = \begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ 0 & * & * & * & * \\ 0 & 0 & * & * & * \\ 0 & 0 & 0 & * & * \end{bmatrix}.$$

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

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

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 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: $A_1 = Q_1 R_1, A_2 = R_1 Q_1 \implies A_2 = Q_1^T A_1 Q_1$

根据前面的计算过程, 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 $\tilde{Q}_1 = G_1 G_2 G_3 G_4 \implies A^{(4)} = \tilde{Q}_1^T A_1 \tilde{Q}_1$

通过直接计算可知, \tilde{Q}_1 的第一列为

$$[c_1, s_1, 0, 0, 0]^T.$$

如果将其取为 A_1 的第一列 $[a_{11}, a_{21}, 0, \dots, 0]^T$ 单位化后的向量, 则 \tilde{Q}_1 的第一列与 Q_1 的第一列相同! $\implies A^{(4)} = W^T A_2 W$



针对带位移的 QR 方法, 我们取 $A_1 - \sigma_1 I$ 的第一列

$$[a_{11} - \sigma_1, a_{21}, 0, \dots, 0]^T$$

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 $6n^2 + O(n)$.



4.3 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, 这种位移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 $\bar{\sigma}$ 也应该是 $\bar{\lambda}$ 的一个很好近似. 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, 不断交替, 这样就有

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

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 , 使得 $Q = Q_1Q_2$ 是实正交矩阵.

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



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. 因此我们希望 **不计算 A_2 , 而是直接从 A_1 得到 A_3 .**

实现方法:

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

$$A_1^2 - 2\operatorname{Re}(\sigma)A_1 + |\sigma|^2 I$$

的第一列平行, 并且 $A_3 = Q^T A_1 Q$ 是上 Hessenberg 矩阵即可.



易知, $A_1^2 - 2\text{Re}(\sigma)A_1 + |\sigma|^2I$ 的第一列为

$$\begin{bmatrix} a_{11}^2 + a_{12}a_{21} - 2\text{Re}(\sigma)a_{11} + |\sigma|^2 \\ a_{21}(a_{11} + a_{22} - 2\text{Re}(\sigma)) \\ a_{21}a_{32} \\ 0 \\ \vdots \end{bmatrix}. \quad (4.5)$$

所以 Q 的第一列是上述向量的单位化.

- 其它过程可通过隐式 QR 迭代实现, 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 $H_1 = \begin{bmatrix} \tilde{H}_1^\top & 0 \\ 0 & I_3 \end{bmatrix}$, 其中 $\tilde{H}_1 \in \mathbb{R}^{3 \times 3}$, 使得其第一列与 $A_1^2 - 2\text{Re}(\sigma)A_1 + |\sigma|^2I$ 的第一列平行. 于是有

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

与 A_1 相比较, $A^{(1)}$ 在 $(3, 1)$, $(4, 1)$ 和 $(4, 2)$ 位置上出现 **bulge**. 在下面的计算过程中, 我们的目标就是要把它们“赶”出矩阵, 从而得到一个新的上 Hessenberg 矩阵.



第二步: 令 $H_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \tilde{H}_2^T & 0 \\ 0 & 0 & I_2 \end{bmatrix}$, 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

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

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 $H_3 = \begin{bmatrix} I_2 & 0 & 0 \\ 0 & \tilde{H}_3^T & 0 \\ 0 & 0 & 1 \end{bmatrix}$, 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

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

此时, bugle 又被向右下角方向“赶”了一个位置。



第四步: 令 $H_4 = \begin{bmatrix} I_3 & 0 \\ 0 & \tilde{H}_4^\top \end{bmatrix}$, 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

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



第五步: 只需构造一个 Givens 变换 $G_5 = \begin{bmatrix} I_4 & 0 \\ 0 & G(4, 5, \theta)^T \end{bmatrix}$, 使得

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

现在, bulge 已经被全部消除, 且

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

其中 $Q = H_1 H_2 H_3 H_4 G_5$. 通过直接计算可知, Q 的第一列即为 H_1 的第一列. 根据隐式 Q 定理, 可以直接令 $A_3 \triangleq A^{(5)} = Q^T A Q$



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 A_k 的右下角收敛到一个上三角矩阵 (两个实特征值) 或一个具有复共轭特征值的 2 阶矩阵, 而且通常会有二次收敛性. 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, 采用 Francis 位移的 QR 迭代并不是对所有矩阵都收敛.

另外, 也可以考虑多重位移策略, 参见 [Watkins 2007].



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 | 应用：多项式求根 *

考虑 n 次多项式

$$q_n(x) = x^n + c_{n-1}x^{n-1} + \cdots + c_1x + c_0, \quad c_i \in \mathbb{R}.$$

- 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

* 本节内容只需了解。



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 \iff 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, 运算量降为 $O(n^2)$
 - 将 A 写成一个酉矩阵与秩一矩阵之差, 参见相关文献