Chapter 8　矩阵特征对计算

张亚楠*

May 29, 2023

目标

设 $A\in\mathbb{R}^{n\times n}$, 寻求 $\lambda\in\mathbb{C}$ 和非零向量 $x\in\mathbb{R}^n$, 使得: $Ax=\lambda x$.

关于特征值的几个性质

Theorem 1 若$Ax=\lambda x$ 则:

1. $\forall a\neq 0,\& \ a\in\mathbb{R}$, ax也是λ对应的特征向量
2. $A-\mu I$ 的特征值为 $\lambda-\mu$
3. A^{-1}的特征值为$\dfrac{1}{\lambda}$

定义 1 设$A=(a_{ij})_{n\times n}$, 令:

1) $r_i=\sum\limits_{j=1,j\neq i}^{n}|a_{ij}|,\ i=1,2,...,n$

2) 集合 $D_i=\{z\,\big|\,|z-a_{ii}|\leq r_I,\ z\in\mathbb{C}\}$

所有 D_i 称之为 A的Gershgorin圆盘.

Theorem 2 设 $A=(a_{ij})\in\mathbb{R}^{n\times n}$,

1) A 的每个特征值必属于下述某个圆盘之中

$$|\lambda-a_{ii}|\leq r_i=\sum_{j=1,j\neq I}^{n}|a_{ij}|$$

或者说，A的特征值都在复平面上n个圆盘的并集中.

2) 如果A有m个圆盘组成一个连通的并集S, 且 S 与余下 $(n-m)$个圆盘是分离的，则S包含A的m的特征值.

*ynzhang@suda.edu.cn　苏州大学数学科学学院

Proof: 设 $Ax = \lambda x,\ x \neq 0\ \&\ x \in \mathbb{R}^n$

记 $|x_k| = \|x\|_\infty \neq 0$, 考虑$Ax = \lambda x$ 的第k个方程，

$$\sum_{j=1}^{n} a_{kj}x_j = a_{k1}x_1 + a_{k2}x_2 + \ldots + a_{kn}x_n = \lambda x_k$$

上式改写为：

$$(\lambda - a_{kk})x_k = \sum_{j\neq k} a_{kj}x_j$$

进而

$$|\lambda - a_{kk}| \cdot |x_k| \leq \|x\|_\infty \cdot \sum_{j\neq k} |a_{kj}|$$

即

$$|\lambda - a_{kk}| \leq \sum_{j\neq k} |a_{kj}|$$

例 1 画出矩阵

$$A = \begin{bmatrix} 4 & 1 & 0 \\ 1 & 0 & -1 \\ 1 & 1 & -4 \end{bmatrix}$$

的Gershgorin 圆盘，并用Matlab函数eig画出特征值在复平面上的坐标.

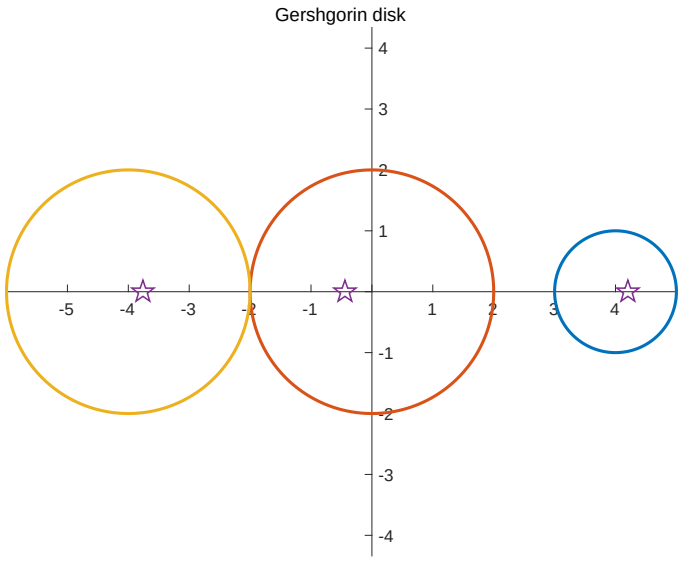

Figure 1: Gershgorin disk and eigenvalue "⋆" obtained by matlab

标注 1 对于上例可选择对角矩阵$diag(1, \sqrt{2}, \sqrt{2})$做相似正交变换，得到新的 矩阵$D^{-1}AD$以及相应的Gershgorin圆盘，每个圆盘都是 分离的。注意：圆盘定理表明每个特征值必然被某个圆盘盖住，但并非一个盘子盖住一个特征值， 也可能是一个盘子盖住多个特征值，有的盘子里没有特征值。

定义 2 对$x \in \mathbb{R}^n, \& x \neq$，记$r(x) = \frac{x^{\mathrm{T}} A x}{(x,x)}$ 为瑞利商(Rayleigh Quotient).

瑞利商[1]是矩阵特征值计算的一个重要概念，若x是矩阵A的特征向量，则瑞利商即是相应的特征值. 若$\|x\| = 1$, $r(x) = x^{\mathrm{T}} A x$.

Theorem 3 设对称矩阵A的特征值排列如下$\lambda_1 > \ldots > \lambda_n$, 则

$$\lambda_n \leq r(x) = \frac{x^{\mathrm{T}} A x}{(x,x)} \leq \lambda_1$$

且可以取到向量x使得上述不等式等号成立.

对称矩阵不光形式特殊，在实际问题中也经常出现. 以下两个关于对称矩阵的结论可在 普通线性代数教材上找到.

例 2 对称矩阵的特征值全是实数且特征向量可选择为正交组. 即

$$A = S * \Lambda * S^{\mathrm{T}}, \quad S * S^{\mathrm{T}} = I, \qquad \Lambda \text{ 为实对角矩阵}$$

记符号bar表示复数取共轭. 若λ 为A的特征值,

$$Ax = \lambda x \Rightarrow A\bar{x} = \bar{\lambda}\bar{x}$$

上述等式左乘以x^{T}

$$x^{\mathrm{T}} A\bar{x} = x^{\mathrm{T}}\bar{\lambda}\bar{x} = \bar{\lambda}\|x\|^2$$

上式再取一次共轭（注意A是实 、 对称）并转置， 得到

$$\bar{\lambda}\|x\|^2 = \lambda\|x\|^2 \quad \|x\| > 0, \qquad \rightarrow \qquad \lambda \in \mathbb{R}$$

即λ是实数.

例 3 若对称矩阵A满足

$$Ax = \lambda_1 x, \quad Ay = \lambda_2 y, \quad \lambda_1 \neq \lambda_2$$

证明:

$$\left(x, y\right) = x^{\mathrm{T}} y = 0$$

该例子说明不同特征值的特征向量是正交的.

$$\lambda_1\left(x, y\right) = \left(\lambda_1 x, y\right) = \left(Ax, y\right) = \left(x, Ay\right) = \left(x, \lambda_2 y\right) = \lambda_2\left(x, y\right) \rightarrow \left(x, y\right) = 0$$

1 幂法

本节介绍幂法(乘幂方法). 并假定A有完备的特征向量组,

$$Ax_j = \lambda_j x_j, \quad 1 \leq j \leq n,$$

[1]原始的 Rayleigh quotient 是对对称矩阵定义的，这里没有规定$A^{\mathrm{T}} = A$, 可以看作是广义的瑞利商。

且

$$|\lambda_1| > |\lambda_2| \geq |\lambda_3| ... \geq |\lambda_n|$$

该方法也适用于 $\lambda_1 = \lambda_2$, 但是不适用于$\lambda_1 = -\lambda_2$ 或者 $\lambda_1 = \bar{\lambda}_2$

任给初始非零向量[2] v_0,

$$v_0 = \sum_{j=1}^{n} \alpha_j x_j, \quad \alpha_1 \neq 0$$

因为$\alpha_j x_j$也是相应λ_j的特征向量，则上式可写为:

$$v_0 = \sum_{j=1}^{n} y_j = y_1 + y_2 + \cdots y_n$$

则

$$Av_0 = Ay_1 + Ay_2 + ... + Ay_n = \lambda_1 y_1 + ... + \lambda_n y_n$$

反复乘幂得到

$$v_k = A^k v_0 = \lambda_1^k y_1 + ... + \lambda_n^k y_n = \lambda_1^k \left[y_1 + \sum_{j=2}^{n} \left(\frac{\lambda_j}{\lambda_1} \right)^k y_j \right] \tag{1}$$

或者

$$\frac{v_k}{\lambda_1^k} = y_1 + \sum_{j=2}^{n} \left(\frac{\lambda_j}{\lambda_1} \right)^k y_j = y_1 + \varepsilon_k$$

以上通过反复乘幂求特征对的方法称之为Power method

注意到一旦得到主特征向量y_1或者 v_k，可按照瑞利商的表达式来计算特征值。 我们下面给出幂法计算特征值的收敛阶估计. 根据(1), 可得

$$v_{k+1} = \lambda_1^{k+1} \left[y_1 + \sum_{j=2}^{n} \left(\frac{\lambda_j}{\lambda_1} \right)^{k+1} y_j \right]$$

上式和(1)分别左乘 任意 非零行向量 [3] u^T 得到:

$$\frac{u^T v_{k+1}}{u^T v_k} = \frac{\lambda_1^{k+1} \left[u^T y_1 + \sum_{j=2}^{n} \left(\frac{\lambda_j}{\lambda_1} \right)^{k+1} u^T y_j \right]}{\lambda_1^k \left[u^T y_1 + \sum_{j=2}^{n} \left(\frac{\lambda_j}{\lambda_1} \right)^k u^T y_j \right]}$$

$$\approx \lambda_1 \frac{u^T y_1 + (\frac{\lambda_2}{\lambda_1})^{k+1} u^T y_2}{u^T y_1 + (\frac{\lambda_2}{\lambda_1})^k u^T y_2} = \lambda_1 \frac{1 + (\frac{\lambda_2}{\lambda_1})^{k+1} \frac{u^T y_1}{u^T y_2}}{1 + (\frac{\lambda_2}{\lambda_1})^k \frac{u^T y_1}{u^T y_2}} = \lambda_1 \frac{1 + \varepsilon * \frac{\lambda_2}{\lambda_1}}{1 + \varepsilon}$$

$$\approx \lambda_1 \left(1 + \mathbb{O}(\frac{\lambda_2}{\lambda_1})^k \right) \to \lambda_1$$

最后一步用到下式

$$\frac{1}{1-x} = 1 + x + x^2 + x^3 + ...$$

Theorem 4 幂法关于比值 $(\frac{\lambda_2}{\lambda_1})$ 线性收敛; 即第k步的迭代误差为 $\frac{u^T v_{k+1}}{u^T v_k} = \mathbb{O}\left((\frac{\lambda_2}{\lambda_1})^k \right)$

[2] 这里需要保证$\alpha_1 \neq 0$ 使得v_0在主特征向量方向的投影非零，一般取v_0为全1向量，或者随机向量即可。

[3] 还要求$u^T y_1 \neq 0$, 取$u = v_k$即可满足.

若A为对称矩阵， 可取$u = v_k = A^k v_0$, 则

$$u^T v_{k+1} = v_0^T * A^k * A^{k+1} v_0 = v_0^T A^{2k+1} v_0, \quad u^T v_k = v_0^T A^{2k} v_0$$

进而

$$\begin{aligned}
&\frac{u^T v_{k+1}}{u^T v_k} = \frac{v_0^T A^{2k+1} v_0}{v_0^T A^{2k} v_0} = \frac{v_0^T \cdot \left(\sum_{j=1}^n \lambda_j^{2k+1} y_j\right)}{v_0^T \cdot \left(\sum_{j=1}^n \lambda_j^{2k} y_j\right)} \\
=&\frac{\lambda_1^{2k+1}\left(v_0^T \cdot y_1 + \sum_{j=2}^n (\lambda_j/\lambda_1)^{2k+1} v_0^T \cdot y_j\right)}{\lambda_1^{2k+1}\left(v_0^T \cdot y_1 + \sum_{j=2}^n (\lambda_j/\lambda_1)^{2k} v_0^T \cdot y_j\right)} \\
\approx&\lambda_1 \cdot \frac{1 + (\lambda_2/\lambda_1)^{2k+1}(v_0^T \cdot y_2 / v_0^T \cdot y_1)}{1 + (\lambda_2/\lambda_1)^{2k}(v_0^T \cdot y_2 / v_0^T \cdot y_1)} \\
=&\lambda_1 \left[1 + \mathbb{O}\left((\frac{\lambda_2}{\lambda_1})^{2k}\right)\right]
\end{aligned}$$

Theorem 5 幂法计算对称矩阵的主特征值关于$(\frac{\lambda_2}{\lambda_1})$可达到平方收敛

幂法的算法实现:

若直接使用幂法计算主特征值而不加修正，则幂法可能不收敛

(1) 若$\rho(A) > 1$, 则 $A^k \cdot \begin{bmatrix} 1 \\ 1 \end{bmatrix} \to \infty$

(2) 若 $\rho(A) < 1$, 则 $A^k \cdot \begin{bmatrix} 1 \\ 1 \end{bmatrix} \to 0$

为了避免出现数值上出的溢出现象[4]， 我们对幂法的算法实现过程中加以规范化，保证特征向量不会趋于 零或无穷大. 例如每次乘幂之后将向量v 除以其最大分量，既能保证特征向量 在无穷范数意义下是单位向量[5].

$$\begin{cases}
v_1 = Av_0, & u_1 = \dfrac{v_1}{\max\{v_1\}} = \dfrac{Av_0}{\max\{Av_0\}} \\
v_2 = Au_1 = \dfrac{A^2 v_0}{\max\{Av_0\}}, & u_2 = \dfrac{v_2}{\max\{v_2\}} = \dfrac{A^2 v_0 / \max Av_0}{\max\{A^2 v_0 / \max Av_0\}} = \dfrac{A^2 v_0}{\max\{A^2 v_0\}} \\
\vdots & \vdots \\
v_k = Au_{k-1} = \dfrac{A^k v_0}{\max\{A^{k-1} v_0\}}, & u_k = \dfrac{A^k v_0}{\max\{A^k v_0\}}
\end{cases}$$

其中$\max\{v_k\}$ 指绝对值最大的那个分量.

分析可知:

$$v_0 = \sum_{j=1}^n \alpha_j x_j, \quad \alpha_1 \neq 0$$

则:

$$v_k = A^k v_0 = \lambda_1^k \left[\alpha_1 x_1 + \sum_{j=2}^n \alpha_j \left(\frac{\lambda_j}{\lambda_1}\right)^k x_j\right] = \lambda_1^k \left(\alpha_1 x_1 + \varepsilon_k\right)$$

[4]数值过大或者过小超出了浮点数表示的范围

[5]教材上给出的这一方法编程是相对麻烦的. 实际使用幂法迭代时 每次将迭代向量单位化即可.

$$u_k = \frac{A^k v_0}{\max\{A^k v_0\}} = \frac{\lambda_1^k\left[\alpha_1 x_1 + \varepsilon_k\right]}{\max\left\{\lambda_1^k\left[\alpha_1 x_1 + \varepsilon_k\right]\right\}} = \frac{x_1}{\max\{x_1\}} \longrightarrow \quad \text{eigen vector}$$

$$\max\{v_k\} = \max\left\{\frac{A^k v_0}{\max\{A^{k-1} v_0\}}\right\} = \frac{\max\{A^k v_0\}}{\max\{A^{k-1} v_0\}}$$

$$= \frac{\max\left\{\lambda_1^k\left[\alpha_1 x_1 + \varepsilon_k\right]\right\}}{\max\left\{\lambda_1^{k-1}\left[\alpha_1 x_1 + \varepsilon_{k-1}\right]\right\}} = \lambda_1 \cdot \frac{\max\left\{\lambda_1^{k-1}\left[\alpha_1 x_1 + \varepsilon_k\right]\right\}}{\max\left\{\lambda_1^{k-1}\left[\alpha_1 x_1 + \varepsilon_{k-1}\right]\right\}} \longrightarrow \quad \text{eig value} \quad \lambda_1$$

上述幂法的算法是比较繁琐的，实际上可按照如下方式以及更为简单的下列示例代码的方式进行 幂法迭代.

Algorithm:

$$\begin{cases} v_0 = u_0, \\ v_{k+1} = A u_k, \\ \lambda = \dfrac{v_{k+1}^T \cdot u_k}{u_k^T \cdot u_k} \quad \text{Rayleigh Quotient: approximation eigenvalue of A} \\ u_{k+1} = \dfrac{v_{k+1}}{\lambda}, \quad \text{OR} \quad u_{k+1} = \dfrac{v_{k+1}}{\|v_{k+1}\|} \qquad k = 0, 1, 2, \ldots \end{cases}$$

Output:

$$\boxed{\lambda \longrightarrow \lambda_1;\ u_k \longrightarrow x_1}$$

我们给出幂法迭代的代码. 测试矩阵为二阶差分矩阵，其主特征值见代码第2行. 读者测试下述代码可发现：代码有效但收敛较慢.

```
1 N = 8; T = toeplitz([-2,1,zeros(1,N-2)]),     % 2nd finite diff matirx
2 sig = -2 + 2*cos(pi*N/(N+1)),                 % largest eigenvalue lam_0
3 v0 = [1;zeros(N-1,1)];  maxI = 111;           % initial guess E vector v0
4 for j = 1:maxI
5     v1 = T*v0;                                % power meth / mat*vec
6     v0 = v1./norm(v1);                        % normalized vec
7 end
8 lam = v0'*T*v0;  e1 = abs(lam - sig)          % E value & check error
```

标注 2 幂法迭代算法有不同的实现方式，其中上段代码是最为简单的。建议读者将幂法理解为计算“主特征向量”的算法，有了主特征向量再“由瑞利商来计算特征值”。这更有助于理解幂法的各种变形。

例 4 用幂法计算

$$A=\begin{bmatrix}4 & 1 & 0\\ 1 & 0 & -1\\ 1 & 1 & -4\end{bmatrix}$$

的特征值，初值选全1向量。对比每步迭代的误差并观察收敛速度。

k	λ_k	Err_k
1	2.5517	1.6513e+00
11	4.1662	3.6802e-02
21	4.2495	4.6489e-02
31	4.2244	2.1368e-02
41	4.2107	7.6612e-03
51	4.2056	2.5847e-03
61	4.2039	8.5616e-04
71	4.2033	2.8194e-04
81	4.2031	9.2667e-05
91	4.2031	3.0439e-05
101	4.2030	9.9963e-06
111	4.2030	3.2826e-06

2 反幂法及其应用

反幂法

$Ax=\lambda x$, 则 两端同时左乘以$\frac{1}{\lambda}A^{-1}$得到： $A^{-1}x=\frac{1}{\lambda}x$; 说明矩阵$A$与其逆矩阵$A^{-1}$共享特征向量，且 A^{-1} 的特征值满足：

$$\frac{1}{|\lambda_n|}>\frac{1}{|\lambda_{n-1}|}>\ldots>\frac{1}{|\lambda_1|}$$

则幂法可直接应用于求A^{-1}的最大特征值，也即是A的最小特征值。

Algorithm 2:

$$\begin{cases} v_0=u_0, \\ \boxed{v_{k+1}=A^{-1}u_k,} \\ u_{k+1}=\frac{v_{k+1}}{\|v_{k+1}\|} \qquad k=0,1,2,\ldots \end{cases}$$

Output: $u_k\to x_n,\quad \lambda_n\approx u_k^{\mathrm{T}}Au_k$

```
1 A = [4,1,0; 1,0,-1; 1,1,-4];  v = ones(size(A,1),1);
2 ee = eig(A);  % eigenvalue
3 [~, ii] = sort( abs(ee) );   lamN = ee(ii(1));  % smallest eigvalue
4 maxit = 8;
5 for j = 1:maxit
6     u = A\v;
7     v = u/norm(u);
8 end
9 lam = v'*A*v;      err = abs(lam - lamN),
```

标注 3 实际算法中不求逆矩阵，而是通过求解线性方程组$Ax=b$实现$A^{-1}*b$. 对于涉及多次A^{-1}乘向量的算法，还应当先进行LU分解$PA=LU$, 即：

$$v_{k+1}=A^{-1}u_k, \Longrightarrow \begin{cases} L*y=Pu_k \\ U*v_{k+1}=y \end{cases}$$

读者测试代码可发现，反幂法比幂法收敛更快. 上述代码没有提前对T进行三角形分解，计算效率较低. 应当对反幂法作用的矩阵T进行分解从而提高求解线性方程组的效率. 读者可尝试修改代码第5行，提高计算效率. 我们用反幂法计算上例A的最小特征值，计算结果表明 对所计算的例子，反幂法比幂法的收敛速度更快. 读者可思考原因并进一步思考 反幂法是否一定比幂法 收敛更快？

k	λ_k	Err_k
1	-0.5283	8.5371e-02
2	-0.4662	2.3263e-02
3	-0.4461	3.1428e-03
4	-0.4433	3.4988e-04
5	-0.4430	4.3564e-05
6	-0.4429	4.8864e-06
7	-0.4429	6.0148e-07
8	-0.4429	6.8127e-08

反幂法的应用(平移)

若已知矩阵A的某个特征值λ的近似值μ，且其n个特征值分离良好，如何计算μ附近的特征值? 记$B=A-\mu\cdot I$, 则B的最小特征值为$(\lambda_A-\mu)$, 可用反幂法求解B的特征值λ_B；进而得到 A 的特征值$\lambda_B+\mu$. 实际上，A和B共享特征向量, 但是对B使用反幂法收敛更快. 我们可以通过反幂法求出该特征向量. 例如：通过测试幂法求主特征值的代码可知： 幂法迭代100步仍未得到高精度的主特征值, 反幂法迭代10余步即得到高精度的特征值.

由幂法的收敛估计式可知误差与$\frac{\lambda_2}{\lambda_1}$正相关, 反幂法与$\frac{\lambda_n}{\lambda_{n-1}}$正相关. 例如：矩阵$A$的特征值分别为1到10共十个自然数， 且事先只知道矩阵A 的某个特征值约为$\mu = 9.9$. 则对$B = A - \mu * I$作用反幂法的收敛速度为 $\left|\frac{10-9.9}{9-9.9}\right|^k \approx \frac{1}{10^k}$. 该收敛速度表明每迭代一次即可多一位有效数字. 若特征值的近似值为 $\mu = 9.99$, 则收敛速度可达 $\frac{1}{100^k}$, 只需几步即可达到机器精度.

```
clear, close all,
N = 8; T = toeplitz([-2,1,zeros(1,N-2)]);     % 2nd finite diff matirx
sig = -2 + 2*cos((N)*pi/(N+1)),               % largest eigenvalue lam_N
v0 = [1;zeros(N-1,1)];  maxI = 5;             % initial guess E vector v0
lam = -3.9;
% % % shift inverse power meth
B = T - lam*eye(N); dB = decomposition(B);    % dB is ldl decomposition of B
for j = 1:maxI
    v1 = dB \ v0;                             % super fast than B
    v0 = v1./norm(v1);
end
lam = v0'*T*v0, e1 = abs(lam - sig)
```

Rayleigh quotient 迭代法（选讲）

通过测试以上代码同样可以发现，近似特征值μ精度越高, 利用反幂法 计算$B = A - \mu * I$收敛越快. 一个很自然的想法是：每步迭代计算得到 近似的特征值μ_{new} 之后, 即得到新矩阵$B_{new} = A - \mu_{new} * I$. 若对$B_{new}$作用反幂法岂不是收敛更快?

这一方法称为Rayleigh商反幂法(Rayleigh Quotient Iteration). 我们给出测试代码. 提醒读者注意：这里给出的 幂法代码首要计算目标是特征向量，有了特征向量再给出相应矩阵的特征值.

```
N = 8; T = toeplitz([-2,1,zeros(1,N-2)]);     % 2nd finite diff matirx
sig = -2 + 2*cos((N)*pi/(N+1)),               % largest eigenvalue lam_N
v0 = [1;zeros(N-1,1)];  s = -3.9;             % initial guess Eigpair v0
B = T - s*eye(N);
for j = 1:4
    v1 = (T - s*eye(N) )\ v0;                 % B_new = B - s*I / shift each step
    v0 = v1./norm(v1);
    s = v0'*T*v0;                             % eigvalue of B
end
e1 = abs(s - sig)                             % v0 is vec of T, B & each T + s*I
```

标注 4 带位移的反幂法和瑞利商迭代法对初值的要求较高，给定某个特征值的 近似值μ之后，算法只会收敛到距离μ最近的那个特征值. 如果μ 的指向不明，可能收敛的结果并不是预期目标. 另外，瑞利商反幂法 是否有必要每步都要做位移加速也值得思考.

3 正交迭代和平移(选讲)

幂法和反幂法的优点是算法简单. 缺点是

1) 若矩阵A存在两个绝对值相等但符号相反的主特征值$\pm\lambda_1$, 则幂法不收敛;

2) 每次只能求出一个特征值

本节分别针对以上两个问题给出改进的算法.

平移算法

若A存在两个符号相反的主特征值，则可通过适当的平移算法使得新的矩阵可以使用幂法迭代. 由定理1，若矩阵$Av=\lambda v$, 则对任意实数μ, $B=A-\mu\cdot I$ 与A 有相同的特征向量v, 且B的特征值为$\lambda-\mu$. 通过平移(选择合适的μ), 可使得B只有一个主特征值，进而对B使用幂法求出 λ_B 和相应的特征向量v. 则$\lambda_A=\lambda_B+\mu$. 由于A,B共享特征向量， 也可通过求瑞利商$v^{\mathrm{T}}Av$即得到 λ_A.

例 5 Legendre 多项式$P_n(x)$ 的根等价于矩阵

$$A=\begin{bmatrix} 0 & \gamma_1 & & & \\ \gamma_1 & 0 & \gamma_2 & & \\ & \ddots & \ddots & \ddots & \\ & & \gamma_{n-2} & 0 & \gamma_{n-1} \\ & & & \gamma_{n-1} & 0 \end{bmatrix}$$

的特征值，其中 $\gamma_j=\frac{j}{\sqrt{(2j-1)(2j+1)}}=\sqrt{\frac{1}{4-1/j^2}}$.

由第三章正交多项式的理论可知$P_n(x)$在区间$(-1,1)$上有n个互异实根， 这些根是成对出现的且符号相反. 也即是A的两个主特征值是符号相反，因此幂法不收敛. 若考虑平移后的矩阵$B=A+I$, 则B的特征值在(0,2)内互异. 可用幂法求其主特征向量和主特征值[6].

```
N = 8;      beta = 0.5./sqrt(1-(2*(1:N)).^(-2));
A = diag(beta,1) + diag(beta,-1);                % legendre poly roots pts
lam = max(eig(A));     v0 = [1;zeros(N,1)];
for j = 1 : 66
    v1 = A*v0;         v0 = v1./norm(v1);
end
eg = v0'*A*v0;         e1 = lam - eg,            % power meth not work for A
% %  ===   power meth for shift A
B = A + eye(N+1);      v0 = [1;zeros(N,1)];
for j = 1 : 66
    v1 = B*v0;         v0 = v1./norm(v1);        % power meth for B gives
end                                              % the correct  eigen vector
 eg = v0'*A*v0;        e1 = lam - eg,
```

[6]后文介绍的基本QR算法 同样对A不收敛，对B收敛

标注 5 适当对矩阵进行平移可以扩大幂法的使用范围，也可加速幂法迭代的收敛性. μ 的选择需要对矩阵有一定的分析和了解. 原则上来说，对矩阵A了解的信息越多， 可以针对性地设计算法，效果也越好.

Gram-Schmidt正交化

借助于Modified Gram-Schmidt算法[7]，我们可以将一组线性无关的列向量正交化, 得到 一组标准正交向量组. 我们首先给出结论，并通过算法说明Gram-Schmidt的实现过程.

例 6 给定矩阵$A \in \mathbb{R}^{m\times n}$, $m \geq n$且$rank(A) = n$. 记span$\{A\}$为A的列向量张成的空间, 也称为列空间(column space). 通过Gram-Schmidt正交化过程可得矩阵 Q, 其中Q的各个列向量彼此正交且 span$\{A\}$ = span$\{Q\}$

```
function [Q,R] = mGSqr(A)
% % modified gram schmidt with double orthogonal
[m,n] = size(A);  Q = A;  R = zeros(n,n);
if m<n, warning('rows is less than cols'), return , end
for j = 1:n
    v = A(:,j);
    for k = 1:j-1
        R(k,j) = Q(:,k)'*v;
        v =  v - Q(:,k)*R(k,j);
    end
% %     reOrthgonal if nessesary
    for k = 1:j-1
        mu = Q(:,k)'*v;
        R(k,j) = R(k,j) + mu;
        v =  v - Q(:,k)*mu;
    end
%
    R(j,j) = norm(v,2);
    Q(:,j) = v / R(j,j);
end
```

标注 6 对A的列向量进行正交化的过程也是QR分解过程. 对列满秩的矩阵$A \in \mathbb{R}^{m\times n}$, 存在正交矩阵$Q \in \mathbb{R}^{m\times n}$ 及上三角矩阵R, 满足

$$A = QR, \quad Q^T * Q = I.$$

QR分解是矩阵分解的重要工具，是计算矩阵特征值的QR迭代算法的基础. 本小节只给出通过Schmidt正交化计算Q的过程. 关于QR算法的进一步讨论放在后文. 另外，Gram-Schmidt正交化和QR分解对复数矩阵也是适用的.

[7] 第五章第六章介绍的QR分解以及二次正交化算法都是基于GS正交化

正交迭代[8]

根据线性代数知识[9]，不同的特征值对应的特征向量是彼此正交的. 借助于这一结论，可以对幂法修正使算法同时可以求多个特征值和特征向量.

给定n阶方阵A且

$$|\lambda_1| > |\lambda_2| > |\lambda_3| ... > |\lambda_n|.$$

若$\{u_0, v_0\}$为两个彼此正交的向量, 对u_0, v_0进行幂法迭代得到 $\{u_1, v_1\}$ 并将$\{u_1, v_1\}$进行单位正交化，进行下一步迭代. 如此循环迭代得到 u_k, v_k. 根据幂法迭代的收敛性可知 u_k 会收敛到主特征向量. 问： v_k是否收敛?

u_k, v_k 收敛到前两个主特征向量！这一方法称为正交迭代. 该方法可 用于计算更多甚至全部特征向量. 理论分析参考Demmel或者Golub的专著.

对矩阵$A \in \mathbb{R}^{n\times n}$, 我们取$m(\leq n)$个$n$维向量做迭代初值，通过幂法结合 正交化迭代可得$A$的前$m$个特征对. 我们只给出算法过程，理论推导可参考文献[10] 这一方法特别适用于大型稀疏矩阵，且计算目标仅为某些特地的几个特征对, 例如最大或者最小的几个 特征对.

正交迭代

$$\begin{cases} V_{k+1} = A * Q_k \\ \text{span}(Q_{k+1}) = \text{span}(V_{k+1}); \quad //V_{k+1} = QR \end{cases}$$

算法第二步对V_{k+1}进行QR分解即可; 因为R_{k+1}不参与迭代，也可只保留正交向量组Q_{k+1}. 我们给出的代码没有输出R_{k+1}.

```
N = 8;      beta = 0.5./sqrt(1-(2*(1:N)).^(-2));
A = diag(beta,1) + diag(beta,-1);                % legendre poly roots pts
B = A + eye(N+1);                                % remove to [0,2]
N = size(A);   v0 = eye(N);  m = 4;  v0 = v0(:,1:m);  % initial m vecs
for j = 1 : 66
    v1 = B * v0;                                 % find the largest m eigs
    v0 = mGSqr(v1);                              % orthogonal & normalized
end
eg = diag(v0'*A*v0),                  % output eig values of A
lamA = sort( eig(A), 'descend') ;     % exact sol by matlab : eig
err1 = norm(eg - lamA(1:m), inf)      % check error
```

[8]本节内容本科生选学，研究生必修

[9]见本章例2

[10]Demmel, Applied Numerical Linear Algebra, SIAM; / Golub, Matrix Computations, Johns Hopkins University Press

上述代码计算了例3 的矩阵, 计算结果为矩阵A位于(0,1) 的特征值. 根据对称性，可得所有Legendre多项式的根. 读者测试代码可以发现上述基于幂法的正交迭代收敛较慢. 反幂法收敛相对较快， 上述算法可简单修改用于求解矩阵A 的前几个最小特征对.

正交迭代2

$$\begin{cases} V_{k+1} = A^{-1} * Q_k \\ \text{find unitary } Q_{k+1} \quad \text{s.t.} \quad span\{V_{k+1}\} = span\{Q_{k+1}\} \end{cases}$$

算法第一步是解线性方程组$AV_{k+1} = Q_k$. 在进行迭代之前应当 对A进行分解(例如LU分解，对称矩阵可选择LDL分解)，提高计算效率. 第二步同样对V_{k+1}进行QR分解.

```
N = 8;      beta = 0.5./sqrt(1-(2*(1:N)).^(-2));
A = diag(beta,1) + diag(beta,-1);                % legendre poly roots pts
B = A + eye(N+1);                                % remove to [0,2]
N = size(A);   v0 = eye(N);  m = 4;  v0 = v0(:,1:m);  % initial m vecs
dB = decomposition(B,'ldl');                     % decomposition of B by ldl
for j = 1 : 33
    v1 = dB \ v0;                                %  V1 = inv(B)*V0
    v0 = mGSqr(v1);                              %  orthogonal & normalized
end
eg = diag(v0'*A*v0),                 % output eig values of A
lamA = sort( eig(A), 'ascend') ;     % exact sol by matlab : eig
err1 = norm(eg - lamA(1:m), inf)     % check error
```

正交迭代的应用

幂法或者反幂法结合正交迭代特别适合求解矩阵的特征值分离情况好的矩阵. 对于有些特殊情况，可以一次求出矩阵的所有特征值.

例 7 Laguerre多项式满足如下递推式

$$(n+1)L_{n+1}(x) = (2n+1-x)L_n(x) - nL_{n-1}(x), \quad L_{-1} = 0, \quad L_0 = 1$$

改写上式为

$$x \cdot L_n(x) = -(n+1)L_{n+1}(x) + (2n+1)L_n(x) - nL_{n-1}(x), \qquad 0 \le n \le N-1$$

进而

$$x * \begin{bmatrix} L_0(x) \\ L_1(x) \\ \vdots \\ L_{N-1}(x) \end{bmatrix} = \begin{bmatrix} 1 & -1 & & & & \\ -1 & 3 & -2 & & & \\ & \ddots & \ddots & \ddots & & \\ & & -(N-2) & 2N-3 & -(N-1) & \\ & & & -(N-1) & 2N-1 & \end{bmatrix} \begin{bmatrix} L_0(x) \\ L_1(x) \\ \vdots \\ L_{N-1}(x) \end{bmatrix} - N * L_N(x) * e_N$$

其中$e_N=[0,...,0,1]^{\mathrm{T}}$. 当 $L_N(x^*)=0$时， x^*是$L_N(x)$的根也是上式中三对角矩阵的特征值. 由于Laguerre多项式的根(矩阵特征值)彼此分离，正交迭代可以一次性全部求出该矩阵的特征值.

```
N = 12;       j0 = -(1:N-1)';  j1 = 2*(1:N)'-1;
A = diag(j0,-1) + diag(j0,1) + diag(j1);      % eig(A) are roots of Laguerre
Q = eye(N);  dA = decomposition(A);      maxI = 211;
for k = 1:maxI
    v =   dA \ Q;
    Q = mGSqr(v);
end
ee = diag(Q'*A*Q);  err = A*Q - Q*diag(ee);  e1 = norm(err(:), inf)
```

标注 7 上述代码可一次性求出Laguerre多项式$L_N(x)$的N个根$\{x_j\}_{j=0}^{N-1}$. 这些根也即是Gauss-Laguerre积分公式的求积节点, 即

$$\int_0^{+\infty} e^{-x} f(x)dx = \sum_{j=0}^{N} \omega_j f(x_j)$$

利用插值性求积公式具有至少N次代数精度，给出关于ω_j的线性方程组并计算得到求积系数. 留作研究生编程作业.

4 基本QR算法

幂法和反幂法的优势在于方法简单，特别适用于大型稀疏矩阵. 但是缺点是每次只能求出一个 特征值，如果利用带位移的反幂法加速还需要特征值的近似值. 正交迭代可一次求出矩阵的几个或者全部特征对，但收敛速度相对较慢. 本章介绍的QR算法(也称为QR迭代)在大多数情况下优于前两种方法. QR 算法(实用算法为带位移的QR迭代)可以一次求出矩阵的所有特征值，特别适用于中小规模矩阵. 当然算法的理论和实现过程都更为复杂. 本节的学习以算法为主，收敛性分析等理论 不做要求.

例 8 *任给$A\in\mathbb{C}^{n\times n}$, $A=SJS^{-1}$为其Jordan标准型，证明 存在上三角矩阵 R 酉矩阵U, 使得*

$$A=URU^{\mathrm{H}}$$

例 9 *任给$A\in\mathbb{R}^{n\times n}$ 且 $A=XDX^{-1}$, 其中D为其相似对角化，X为可逆矩阵，证明 存在实数上三角矩阵 T 实正交矩阵Q, 使得*

$$A=QTQ^{\mathrm{T}}$$

给定矩阵A, 按照如下迭代格式

$$\begin{cases} A_k = Q_k\cdot R_k \\ A_{k+1} = R_k\cdot Q_k \end{cases}$$

其中Q是正交矩阵，R是上三角矩阵. 可得到矩阵序列 A_k $(k=0,1,2,\ldots)$.

读者可以多次测试以下代码并观察结果[11]. 通过观察测试代码思考:

1) 该算法为何奏效? 也即 A_{k+1}为何会变成上三角矩阵?

2) 每步迭代计算复杂度多大? $\mathbb{O}(n^3)$

3) 收敛速度如何? 慢！！

```
clear, n1 = 5; A = rand(n1);
for kk = 1:20
    [Q,R] = qr(A);
    A = R*Q,
end
A
```

Theorem 6 *上述QR迭代产生的矩阵序列A_k满足*

1) $A_{k+1}=Q_k^T\cdot A_k\cdot Q_k$ 且

$$A_{k+1}=Q_k^T\cdots Q_0^T\cdot A_0\cdot Q_0\cdots Q_k$$

[11]此处是调用了Matlab自带的QR函数，也可 用上节函数mGSqr.m

2) 记 $\hat{Q}_k = Q_0 Q_1 \cdots Q_{k-1}$, 则 $A_{k+1} = \hat{Q}_k^{\mathrm{T}} A_0 \hat{Q}_k$, 且

$$A^{k+1} = \hat{Q}_k \hat{R}_k$$

证明：1）是显然的递推式。我们证明 2). 记录QR迭代产生的 Q_k 和 R_k

$$\begin{aligned} \boldsymbol{A}_0 &= \boldsymbol{Q}_0 \boldsymbol{R}_0 \quad \boldsymbol{A}_1 = \boldsymbol{R}_0 \boldsymbol{Q}_0 \\ \boldsymbol{A}_1 &= \boldsymbol{Q}_1 \boldsymbol{R}_1 \quad \boldsymbol{A}_2 = \boldsymbol{R}_1 \boldsymbol{Q}_1 \\ \boldsymbol{A}_2 &= \boldsymbol{Q}_2 \boldsymbol{R}_2 \quad \boldsymbol{A}_3 = \boldsymbol{R}_2 \boldsymbol{Q}_2 \end{aligned}$$

检验：

$$\begin{aligned} \left(Q_0 Q_1 Q_2\right) * \left(R_2 R_1 R_0\right) &= Q_0 Q_1 A_2 R_1 R_0 \\ &= Q_0 Q_1 R_1 Q_1 R_1 R_0 \\ &= Q_0 R_0 Q_0 R_0 Q_0 R_0 \\ &= A^3 := \hat{Q}_2 \hat{R}_2 \end{aligned}$$

记 $\hat{Q}_k = Q_0 Q_1 \cdots Q_{k-1}$, 则可得 [12] $A^{k+1} = \hat{Q}_k \hat{R}_k$ ■

标注 8 SIAM news 报道QR算法是20世纪最重要的Top10算法之一. 关于算法的收敛性证明可参阅 Parlett, 1965NumerMath, Article: Convergence Of The QR Algorithm. 收敛性不够直观， 较难理解. 我们这里给一个粗略的解释.

假设A有完备的特征向量空间， 且特征值彼此分离 $|\lambda_1| > |\lambda_2| > \ldots > |\lambda_n| > 0$, 记

$$A = S \Lambda S^{-1}, \quad S = Q_s R_s, \quad S^{-1} = L \cdot U.$$

其中，分别对S, S^{-1}进行QR分解和LU分解. 代入上式可得

$$A^{k+1} = S * \Lambda^{k+1} * S^{-1} = S * \Lambda^{k+1} * L * U = S * \underbrace{\Lambda^{k+1} * L \Lambda^{-(k+1)}} * \underline{\Lambda^{(k+1)} * U}$$

现在说明三个事实

1. $\Lambda^{k+1} * L \Lambda^{-(k+1)} \to I$, 检验 $\lambda_i^{k+1} * l_{ij} * \lambda_j^{-(k+1)} = l_{ij} * (\lambda_i / \lambda_j)^{k+1} \to 0, \quad \text{as} \quad i > j$
2. $\Lambda^{k+1} * L \Lambda^{-(k+1)} * \Lambda^{(k+1)} * U := bigU$是上三角
3. 根据QR分解的唯一性(R的对角元符号确定的前提下)；

$$A^{k+1} = S * bigU = Q_s * (R_s * bigU) = \hat{Q}_k \hat{R}_k \quad \Rightarrow \quad Q_s = \hat{Q}_k \quad \text{as} \quad k \to \infty$$

结论：

$$A_{k+1} = \hat{Q}_k^{\mathrm{T}} * A_0 * \hat{Q}_k \approx Q_s^{\mathrm{T}} * (S \Lambda S^{-1}) * Q_s = \cancel{Q_s^{\mathrm{T}} * (Q_s R_s \Lambda R_s^{-1} Q_s^{-1}) * Q_s}$$

是上三角矩阵.

[12] 正交矩阵的乘积仍是正交矩阵；上三角矩阵的乘积仍是上三角.

标注 9 提示：此处推导只是为了帮助理解算法过程，不是严格的数学证明. 而且推导过程中加了很多条件. 有兴趣读者自行查阅文献仔细推导证明过程.

5 Householder变换

QR迭代算法的基础就是矩阵的QR分解， Gram-Schmidt正交化算法(修正的GS以及二次正交化GS)简单有效. 本节给出一种正交变换方法——Householder变换，可用于对矩阵进行QR分解且 算法是数值稳定的. Householder变换还可对一般矩阵进行正交相似变换约化其 为Hessenberg矩阵.

例 10 对任意模长为1的向量$w\in\mathbb{R}^n$, 检验 反射矩阵$A=I-2ww^T$,满足

$$A=A^T,\quad A*A=I.$$

即A 为对称正交矩阵。

记超平面（过圆点以w 为法向）

$$S=\{x\mid w^Tx=0,\quad \forall x\in\mathbb{R}^n\},$$

是$\mathbb{R}^n$子空间，维数为$(n-1)$. 任意向量$z\in\mathbb{R}^n$可在S和span$\{w\}$做正交分解

$$z=x+y,\quad x\in S,\quad \&\quad y=\alpha\cdot w,\quad \alpha\in\mathbb{R}$$

进而

$$Az=Ax+Ay=x+(I-2ww^T)y=x+y-2ww^T*\alpha w=x-y$$

几何上，上述反射矩阵可以看作是以S为镜面的反射, 有时也称Householder变换为反射变换.

Theorem 7 给定两个模长相等的向量$x,y\in\mathbb{R}^n$, 则存在反射变换P, 使得 $Px=y$.

证明：取$w=\frac{x-y}{\|x-y\|}$, 检验$P=I-2ww^T$满足要求.

$$Px=x-2\frac{(x-y)*(x-y)^{\mathrm{T}}}{\left(x-y,x-y\right)}x=x-(x-y)\frac{2(x-y)^{\mathrm{T}}x}{(x-y)^{\mathrm{T}}(x-y)}\tag{2}$$

由实内积的对称性可得

$$\frac{2(x-y)^{\mathrm{T}}x}{(x-y)^{\mathrm{T}}(x-y)}=\frac{2(x^{\mathrm{T}}x-y^{\mathrm{T}}x)}{x^{\mathrm{T}}x-x^{\mathrm{T}}y-y^{\mathrm{T}}x-y^{\mathrm{T}}y}=1.$$

代入上式到(2)即得 $Px=y$. ■

标注 10 定理7的结论是算法的基础. 我们可以利用该结论对矩阵进行QR分解，还可以将一般矩阵通过 正交相似变换变为特殊的上Hessenberg矩阵. 另外，若$x,y\in\mathbb{C}^n$，定理结论成立。但是取$P=I-2ww^{\mathrm{H}}$, $w=(x-y)/\|x-y\|$, 不能证明上述结果。需要推广反射变换的定义，例如

$$I-\frac{1}{w^{\mathrm{H}}x}ww^{\mathrm{H}}.$$

关于complex Householder变换的推广可参考 [13]

[13] Kuo-Liang Chung and Wen-Ming Yan, The Complex Householder Transform, IEEE TRANSACTIONS ON SIGNAL PROCESSING, VOL. 45, NO. 9, SEPTEMBER 1997

5.1 Householder变换实现QR分解

例 11 $\forall x \in \mathbb{R}^n$, 以及$\|x\| = \rho$, 存在Householder变换$P$得到

$$Px = [\pm\rho, 0, ..., 0]^T$$

由例11, 对任意给定方阵，可通过反射变换实现

$$\begin{bmatrix} \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \end{bmatrix} \xrightarrow{P_1 = I - 2w * w^T} \begin{bmatrix} \times & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & \times & \times & \times \end{bmatrix}$$

也即是将第一列对角线以下全变为零. 对小一阶规模的矩阵重复上述过程

$$\begin{bmatrix} I & 0 \\ 0 & P_2 \end{bmatrix} \cdot \begin{bmatrix} A_{11} & A_{12} \\ 0 & A_{22} \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} \\ 0 & P_2 \cdot A_{22} \end{bmatrix} = \begin{bmatrix} \times & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & 0 & \times & \times \\ 0 & 0 & \times & \times \end{bmatrix}$$

注意到$P_1, diag(I, P_2), ...$ 是对称正交矩阵，其乘积也时对称正交的. 反复利用反射变换将对角线以下的元素变为零， 最终得到三角形矩阵. 由于反射矩阵的乘积是正交矩阵，正交矩阵的逆是其转置也是正交矩阵，因此上述过程也实现了 矩阵的QR分解.

$$P_{n-1}P_2P_1 \cdot A = R, \quad PA = R, \quad A = P^T \cdot R = QR$$

```
n = 5;
% A = hilb(n);
% A = rand(n,n) - 1i*rand(n,n);
A = rand(n,n);
R = A;    Q = eye(n);
for k = 1:n
    if norm(R(k+1:n,k)) > 11*eps
        j1 = k:n;     x = R(j1,k);      rho = norm(x);
        if abs(x(1))>1e-11
            sgn = x(1)/abs(x(1));
        else
            sgn = 1;
        end
        w = x + sgn*[rho;zeros(n-k,1)]; w = w /norm(w);
            R(j1,:) =  R(j1,:) - 2*w*(w'*R(j1,:));
            Q(:,j1) =  Q(:,j1) - 2*(Q(:,j1)*w)*w';
    end
end
 e1 = norm( Q'*Q - eye(n) ), e2 = norm(A - Q*R)
```

测试上述代码可知道，基于Householder变换的QR分解比基本和修正的Gram-Schmidt算法更加稳定。 上述代码对复矩阵可用, 注意： $Px=[\rho;0,...,0]$适用； 一般的 $Px=y$ 不适用, 需要complex householder transform.

5.2 变换一般矩阵为Hessenberg矩阵

Hessenberg矩阵是数值计算中常用的一类特殊矩阵，在求解线性方程组问题和特征值问题时经常出现. 形式如下[14]

$$H=\begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ 0 & 0 & \times & \times & \times \\ 0 & 0 & 0 & \times & \times \end{bmatrix}$$

Theorem 8 设$A\in\mathbb{R}^{n\times n}$, 则存在正交矩阵$Q$, 上Hessenberg矩阵$H$, 满足

$$AQ=QH, \quad \Leftrightarrow \quad A=QHQ^{\mathrm{T}} \quad \Leftrightarrow \quad H=Q^{\mathrm{T}}AQ$$

与Schur分解和QR迭代的收敛性条件不同，一般的实数矩阵不一定正交相似于上三角矩阵。但是任意实数矩阵必正交相似于上 Hessenberg矩阵，而且有通用的方法来实现 这一过程——Hessenberg反射变换. 我们给出算法流程和说明.

$$\begin{bmatrix} \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \end{bmatrix} \xrightarrow[P\cdot A\rightarrow]{\text{Householder } P} \begin{bmatrix} \times & \times & \times & \times \\ \times & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & \times & \times & \times \end{bmatrix} \xrightarrow[\rightarrow]{PAP^{\mathrm{T}}} \begin{bmatrix} \times & \times & \times & \times \\ \times & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & \times & \times & \times \end{bmatrix}$$

上式第一步是将矩阵的第一列的$(2:n)$行进行反射变换，对角线第一个元素保持不变。 其对应的变换矩阵$\mathrm{diag}(I_1,P_{n-1})$是正交的。检验该变换对第一行不起作用, 进而 右乘反射矩阵只左右于$(2:n)$ 列，第一列保持不变。

由于$P^T=P^{-1}$, 上述过程是相似变换，A与PAP^{T} 具有同样的特征值。对小规模的$(n-1)$阶矩阵重复以上过程可得

$$A\rightarrow P_1AP_1\rightarrow P_2P_1AP_1P_2\rightarrow\cdots\rightarrow \text{Hessenberg form}$$

检验整个过程的计算量是$\mathbb{O}(n^3)$. 如果A本身是对称矩阵，则最终得到的Hessenberg矩阵为三对角矩阵. 以下代码是通过Householder变换将A变为上Hessenberg矩阵的例子. Matlab函数名为[P,H] = hess(A) 输出$AP=PH$.

```
function [P,H] = myHess(A)
% compute AP = PH, where P'*P = I; and H is Hessenberg
% For example:  A = rand(5);
% [P1,H1] = myHess(A);     [P, H ] = hess(A);
```

[14] 上Hessenberg矩阵是近似上三角矩阵，其中下三角部分只有次对角线非零.

```
% norm(P-P1), norm(H1-H)
n = size(A,1);  H = A;  P = eye(n);
for k  = 1:n-2
    j1 = k + 1 : n;     x = H(j1,k);
   if norm(H(k+2:n,k))>11*eps
       if abs(x(1))>1e-11
          cc = -x(1)/abs(x(1));
       else
          cc = -1;
       end
       y = [cc*norm(x); zeros(n-k-1,1)];
       [alp,w] = house(x,y);
H(j1,:) =  H(j1,:) - alp*w*(w'*H(j1,:));  % rows
H(:,j1) =  H(:,j1) - (H(:,j1)*w)*w'*alp'; % cols
P(:,j1) =  P(:,j1) - (P(:,j1)*w)*w'*alp'; % householer
   end
end
end
%
function [alp,w] = house(x,y)
% Px = y; |x| = |y|; P = I - alp*w*w'; for any complex vectors x,y
if (norm(x) - norm(y)) > 11*eps, error('|x| not equals |y|'), end
   w = x-y;  alp = 1 /(w'*x);
end
```

6 Givens 变换

例 12 证明： 矩阵 $G=\begin{bmatrix}\cos\theta & \sin\theta\\ -\sin\theta & \cos\theta\end{bmatrix}$ 是正交矩阵.

例 13 证明： k阶方阵 $G(k,\theta)=\begin{bmatrix}\cos\theta & & \sin\theta\\ & I_{k-1} & \\ -\sin\theta & & \cos\theta\end{bmatrix}$ 是正交矩阵, 其中 I_{k-1}表示$(k-1)$阶单位矩阵. 更进一步有

$$G(i,j,\theta)=\begin{bmatrix}I_{i-1} & & & & \\ & \cos\theta & & \sin\theta & \\ & & I_{j-i-1} & & \\ & -\sin\theta & & \cos\theta & \\ & & & & I_{n-j}\end{bmatrix}$$

也是正交矩阵. 其中I_0 为空, 此时 位置被$\cos\theta$ 占据.

例 14 给定向量 $x=[a,b]^{\mathrm{T}}$ 且$ab\neq 0$, 证明：存在矩阵

$$G=\begin{bmatrix} c & s \\ -s & c \end{bmatrix}, \qquad s.t. \qquad Gx=\begin{bmatrix} r \\ 0 \end{bmatrix}, \quad r=\|x\|_2=\sqrt{|a|^2+|b|^2}$$

Givens旋转变换的优点在于可以将某个指定位置的元素变为零，而仅仅需要改变某两行元素，其它元素保持不变； 读者可测试以下代码发现： （1）只有(i,j)两行发生变化； （2）G 是正交矩阵。 另外，注意若考虑复数运算，复内积要取共轭；则此时可取 [15] $G=[\bar{c},\bar{s};-s,c]$.

```
function [R,G] = givens(a,b,i,j,n)
% Givens rotation trans x to y = R*x
% for given x = [a; b], R*x = [||x||; 0]
% %  example
%     n = 5; A = rand(n,1); i = 1; j = 3;
%     a = A(i,1); b = A(j,1);
%     [R,G] = givens(a,b,i,j,n);
%     norm(G'*G - eye(n)), GA = G*A;  A([i,j]), GA([i,j])
x = [a; b];    r = norm(x);    c = a/r; s = b/r;
R = [c',s'; -s,c];
if nargin<3
   G = [];
else
   G = speye(n);    G([i,j],[i,j]) = R;
end
```

旋转变换也是正交变换，是数值稳定的；类似于反射变换，Givens旋转变换可以实现矩阵的QR分解.

```
function [Q,R] = myGivensQR(A)
% % qr decomp by Givens
% % example:
%         A = rand(4);  [Q,R] = myGivensQR(A);
%         Q'*Q - eye(4), A - Q*R, R,
[m,n] = size(A);
% if m<n, warning('rows is less than cols'), return, end
R = A;  Q = eye(m);
for i = 1:n
    for j = i+1:m
           a = R(i,i); b = R(j,i);
        if abs(b)>11*eps
           G = givens(a,b);
           R([i,j],:) = G*R([i,j],:);
           Q(:,[i,j]) = Q(:,[i,j])*G';
        end
```

[15] 实际上复数域上的 Givens 变换实现上例有无穷多组取法，取决于 r 的选择；例如$re^{i\theta}$ 均是保长度的.

```
    end
end
```

标注 11 由于Givens变换每约化一个零元素需要处理矩阵的两行，用于矩阵的QR分解时，主元所在的行每次都要参与变换，计算代价是Householder变换的两倍。因此，Givens变换更加适合于零元素较多(稀疏/sparse)的矩阵的QR分解，而稠密矩阵的QR分解最好采用 Householder反射变换或者 Gram-Schmidt(结合二次投影正交化)。

7 Hessenberg矩阵的QR迭代

由上节知识已知，任意矩阵$A \in \mathbb{R}^{n\times n}$, 均正交相似于一个上Hessenberg矩阵$H$, 即

$$AQ = QH$$

若已知 $Hx = x\lambda$, 则

$$AQx = QHx = Qx\lambda$$

即$y = Qx$为A的相应于λ的特征向量. 因此，计算矩阵A 的特征对就完全转化为计算H 的特征对.

Hessenberg矩阵的QR分解

Hessenberg矩阵是近似上三角矩阵，其中下三角部分只有次对角线非零。对这样一类特殊形式的矩阵进行QR分解，计算复杂度可降低为$\mathbb{O}(n^2)$. 如下所示，反射变换作用第一列时，上段代码中定义的w 只有两个非零元素。每次反射变换只需要处理矩阵的两行即可。每次乘除法次数为n, 进行n 步完成，总的工作量为$\mathbb{O}(n^2)$.

$$\begin{bmatrix} \times & \times & \times & \times \\ \times & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & 0 & \times & \times \end{bmatrix} \xrightarrow{H = I - 2w * w^T} \begin{bmatrix} \times & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & 0 & \times & \times \end{bmatrix}$$

针对Hessenberg矩阵进行QR分解的代码可简化如下

```
clear  % House QR decomp for hess mat
A = [1 2 3 4;  4 4 4 4; 0 1 -1 1; 0 0 2 3],
n = size(A,1);   R = A;    Q = eye(n);
for k = 1:n-1
   if abs(A(k+1,k))>11*eps
       x = R(k:k+1,k);    rho  =  norm(x);
       if abs(x(1))>1e-11, sgn = -x(1)/abs(x(1)); else, sgn = -1; end
       w = x - [sgn*rho; 0];  w = w/norm(w);
       R(k:k+1,:) =  R(k:k+1,:) - 2*w*(w'*R(k:k+1,:));   % H transf /2 rows
       Q(:,k:k+1) =  Q(:,k:k+1) - (Q(:,k:k+1)*w)*w'*2;
   end
```

```
end
norm(Q*R - A ), norm(Q'*Q - eye(n)), R
%%  Givens QR for hess
R = A; Q  = eye(n);
for k = 1:n-1
    a = R(k,k); b = R(k+1,k);
    if abs(b)>11*eps
       G = givens(a,b);
       R(k:k+1,:) = G*R(k:k+1,:);
       Q(:,k:k+1) = Q(:,k:k+1)*G';
    end
end
norm(Q*R - A ), norm(Q'*Q - eye(n)), R
```

可以检验对Hessenberg矩阵进行QR分解，产生的正交矩阵Q仍然是Hessenberg矩阵. 另外，Givens变换同样可以实现Hessenberg矩阵的QR分解. 由于 此时对角线下只有一个非零元，Givens变换的计算代价和Householder变换相同。 如果是形式更为简单的三对角矩阵，一步QR分解的计算复杂度进一步降低为$\mathbb{O}(n)$.

Hessenberg矩阵的基本QR算法

对于一般形式矩阵，求解其特征值可按照以下两步实现

1. 利用Householder变换化矩阵A为与其相似的Hessenberg矩阵H
2. 对H进行QR迭代
   1) 对Hessenberg矩阵进行QR分解: $H = QR$
   2) 更新$H_{new} = RQ$ 得到新的Hessenberg矩阵

标注 12 第一步$H = QR$的计算复杂度$\mathbb{O}(n^2)$. 第二步注意到R, Q分别是上三角和Hessenberg矩阵，则其乘积H_{new} 仍然是Hessenberg矩阵, 也即Hessenberg矩阵的QR算法产生的序列保持形式不变. 表面看每次矩阵乘积的计算复杂度[16]为 $n^3/6$. 实际上，RQ相当于$Q = (I - 2w * w^T)$ 对R做对应列变换，因为w 只有两个非零元，只要$\sum_{j=1}^{n} 2j = n^2$次乘法.

```
A = [1 2 3 4;  4 4 4 4; 0 1 -1 1; 0 0 2 3],
lam = eig(A);  Q = eye(size(A));  A0 = A;
n = size(A,1);
for k = 1 : 22
%     [Q,R] = myGivensQR(A);
```

[16]按照所得的Hessenberg矩阵逐列计算每列 的计算复杂度.

```
%       A = R*Q;
    for j = 1:n-1
        a = A(j,j); b = A(j+1,j);
        if abs(b) > 11*eps
           G = givens(a,b);
           A([j,j+1],:) = G*A([j,j+1],:);
           A(:,[j,j+1]) = A(:,[j,j+1])*G';
           Q(:,[j,j+1]) =  Q(:,[j,j+1])*G';
        end
    end

end
sort(lam) - sort(diag(A)),
norm( Q'*A0*Q - A )
```

对Hessenberg矩阵进行基本QR算法时，每次迭代计算量是$\mathbb{O}(n^2)$，将QR迭代的每步迭代 的计算量降到$\mathbb{O}(n^2)$是非常了不起的进步。 假设K为迭代步数，则总体运算量为 $K*n^2$, 实际计算时，基本QR算法进行Hessenberg矩阵往往需要多次迭代， 即 $K \gg n$ 则整体计算量仍然是超过$\mathbb{O}(n^3)$. 需要进一步提高收敛速度.

8 Hessenberg矩阵的带位移QR算法

由于基本QR迭代收敛较慢，实际计算时大多采用带 位移的QR算法(如: Rayleigh shift， Wilkinson shift). 带位移的QR迭代逐个计算每个特征值. 一旦得到位于矩阵右下角的特征值, 删去矩阵的最后一行从而降低矩阵的维数. 对小规模的矩阵进行新的迭代，直至求出所有特征值. 例如对Hessenberg矩阵

$$H = \begin{bmatrix} \times & \times & \times & \times \\ \times & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & 0 & \alpha & \lambda_n \end{bmatrix}$$

当$\alpha = 0$ 时， λ_n是H的一个特征值. 实际计算时，当$\alpha < \varepsilon$, 即可认为λ_n是所求特征值. 进而目标矩阵变为$(n-1)$阶的.

$$H_n = \begin{bmatrix} \times & \times & \times & \times \\ \times & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & 0 & 0 & \lambda_n \end{bmatrix} = \begin{bmatrix} H_{n-1} & \times \\ 0 & \lambda_n \end{bmatrix}$$

重复以上过程直至$n = 2$, 停止计算，求出2阶矩阵的特征值即可.

为了加快收敛速度, 在对H进行QR迭代时可选择位移加速

$$\begin{cases} \left(H-\sigma\cdot I\right)=Q\cdot R \\ H_{new}=R\cdot Q+\sigma\cdot I \end{cases} \tag{3}$$

其中I是单位矩阵，$\sigma\in\mathbb{C}$是位移. 可以检验上述过程满足$H_{new}=Q^{\mathrm{T}}\cdot H\cdot Q$.

(3)中 σ 的选择对收敛速度很重要. 若$\sigma=\lambda_n$ 是H 的特征值或者是其近似值， 则针对(近似)奇异矩阵$\left(H-\sigma*I\right)$的 QR分解得到上三角矩阵R, 有 $R_{nn}=\varepsilon(\approx)0$. 而新的$H_{new}=R*Q$形式如下

$$H_{new}=R*Q=\begin{bmatrix} \times & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & 0 & \times & \times \\ 0 & 0 & 0 & \varepsilon\approx 0 \end{bmatrix} * \begin{bmatrix} \times & \times & \times & \times \\ \times & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & 0 & \times & \times \end{bmatrix} = \begin{bmatrix} \times & \times & \times & \times \\ \times & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & 0 & \alpha\approx 0 & \lambda_n^{new} \end{bmatrix}$$

由矩阵的乘法规则可知 $\alpha\approx 0$, 则λ_n^{new}即是所求特征值. 删去矩阵的最后一行和最后一列的多新的H_{n-1}, 重复以上过程求出下一个特征值.

Rayleigh Quotient shift 算法

取$\sigma=H_{nn}$为矩阵H的右下角的元素，得到的算法称为是一类特殊的 Rayleigh Quetient 平移QR迭代. 读者可测试下面代码发现该算法比基本QR迭代收敛更快.

```
clear, %   --- shift QR algrithm
A = [1 2 3 4;  4 4 4 4; 0 1 -1 1; 0 0 2 3];      % test matrix
% N = 11;     beta = 0.5./sqrt(1-(2*(1:N)).^(-2));
% A = diag(beta,1) + diag(beta,-1);               % legendre poly roots pts
lam = sort( eig(A) );                            % exact eigenvalue by matlab
n1 = size(A,1);  eg = zeros(n1,1);
ij = 1;    Tol = 1e-9;   maxI = 20*n1;          % tol and max iter steps
while n1 > 1
    ee =  A(n1,n1);                              %  Rayleish shift
%   ee = eig_s2(A(n1-1:n1, n1-1:n1));            % get eigvalue of 2*2 mat
%   [~,i] = min(ee - A(n1,n1));  ee = ee(i);     % Wilkinson's shift lines 10-11
    [Q, R] = myGivensQR(A - ee*eye(n1) );
     A = R*Q + ee*eye(n1);                       % R is triangle, Q:hess
     corn = A(n1,n1-1);                          % check eps
   if abs(corn) < Tol
       eg(n1) = A(n1,n1);     n1 = n1-1;    A = A(1:n1,1:n1);
   end
   ij = ij + 1;                                  % plus iteration steps
   if ij > maxI, warning('Method may not work! '), return, end
end
eg(1) = A(1,1);  err1 =   sort(eg) - lam,
```

```
22 fprintf('\n whole itr steps is %d, error is %.1e \n', ij, norm(err1))
```

上述简单的平移算法并非永远奏效, 读者可测试对于代码3-4行定义的 对称三对角矩阵，该算法不收敛. 而下面的平移方法更加有效.

Wilkinson shift 算法

我们给出一个对大部分矩阵的特征值计算都有效的算法. 与上一算法的区别在于 位移σ 的选择. 记 $\alpha = \text{eig}(H(n1-1:n1, n1-1:n1))$ 为H右下角的2×2矩阵的两个特征值. 取位移$\sigma = \min\{\alpha - \lambda_n\}$ 为距离右下角较近的元素. 这一算法称为Wilkinson shift 算法. 由于2×2 矩阵的特征值有显式表达式， 所以该算法对比上一算法计算复杂度几乎没有增加. 另外，由于σ可能取到复数， 该算法也可计算矩阵的复特征值.

我们首先给一个简单的代码用于计算二阶矩阵的特征值.

```
1  function [sig] = eig_s2(A)
2  % A is a matirx of size 2 OR a number,
3  % sig is the eigenvalue of A
4  [m,n] = size(A);
5  if (m==1) && (n==1), sig = A;    return,    end
6  if (m~=2) || (n~=2), warning('Input A is not 2*2 matrix'),  return,  end
7    a11 = A(1,1); a22 = A(2,2); a12 = A(1,2); a21 = A(2,1);
8    % p_x =  (x - a11)*(x-a22) - a12*a21          % eig poly
9    b = -a11-a22; c = a11*a22 - a12*a21;          % x^2 + bx + c
10   del = sqrt( b*b - 4*c );   x1 =  -b + del;  x2 = -b - del;
11   sig = 0.5*[x1; x2];        sig = sort(sig);
12 end
```

利用上述代码选择合适的位移σ 可得Willkinson平移算法(见上一代码). 与上一小节算法对比，算法的区别只在于位移σ 的选择. 通过测试对比，该算法更加高效.

标注 13 本节介绍的Wilkinson平移QR算法是非常高效的算法，对大多数矩阵都有立方收敛阶. 关于矩阵的特征对计算还有许多方法，例如基于矩阵的LU分解的LR迭代法，古典的 Jacobi迭代，针对大型稀疏矩阵的Krylov子空间方法. 有兴趣的读者可以参考阅读: Demmel, Golub, Saad 等人的专著. 中文《数值线性代数》（徐数方，高立，张平文编著）也可作为阅读材料. Golub著作《Matrix Computation》已由袁亚湘等翻译为中文版.

标注 14 本节平移算法给出了计算 Hessenberg矩阵H的特征值的高效算法，一旦 得到H的特征值λ, 计算新的奇异的、Hessenberg矩阵$B = \lambda I - H$的零空间，即得到相应的特征向量x. 由于舍入误差的影响，$det(B) \approx 10^{-15} \neq 0$, 计算$Bx = 0$时往往只得到零解. 此时可以选择反幂法迭代计算一步或者两部$Bx = \epsilon$即得到特征向量x.

小结

再次总结一下QR算法的伟大之处：对于中小规模矩阵$A \in \mathbb{R}^{n\times n}$,

(1) Householder变换实现 $AQ = QH$; 计算量 $\mathbb{O}(n^3)$.

(2) 大约$\mathbb{O}(n)$步的Willkinson位移算法计算出H 所有特征值，总的计算量 $\mathbb{O}(n^3)$

(3) 逐一计算出对应H特征值λ_j 的特征向量x_j, 每个计算量为n^2. 算出所有 特征向量 $X = [x_1, ..., x_n]$ 的代价为$\mathbb{O}(n^3)$.

(4) 计算 A的所有特征向量 $Y = QX$ 一次矩阵相乘代价为$\mathbb{O}(n^3)$.

计算所有特征值和特征向量的计算复杂度只要 $\mathbb{O}(n^3)$. Amazing！！！

9 上三角(Hessenberg)矩阵的特征向量(略)

对Hessenberg矩阵进行基本QR算法计算特征值最终得到如下形式

$$H = QTQ^{\mathrm{T}}, \quad HQ = QT$$

其中T是上三角矩阵, $Q = Q_0Q_1...Q_k$为QR迭代产生的正交矩阵。 若

$$Tx = x\lambda,$$

则

$$H(Qx) = \lambda(Qx)$$

在进行QR迭代的过程中可记录Q，利用上三角矩阵 T 的特征向量，即得H 的特征向量。

9.1 上三角矩阵的特征向量

给定如下形式的矩阵

$$T = \begin{bmatrix} t_1 & + & + & + \\ 0 & t_2 & + & + \\ & & \ddots & \\ 0 & 0 & 0 & t_n \end{bmatrix}$$

由特征多项式定义知道T的 特征值即是对角元素 $\lambda_i = t_i$, 现采用如下方法计算 t_i相应的特征向量x_i. 记

$$B_i = T - t_iI = \begin{bmatrix} T_{i-1} & + & + \\ & 0 & + \\ & & T_{n-i} \end{bmatrix}, \quad T_{i-1} \in \mathbb{R}^{(i-1)\times(i-1)}, \quad T_{n-i} \in \mathbb{R}^{(n-i)\times(n-i)}$$

B_i的非零解[17] 即是x_i. 可按照如下方式求解:

$$\begin{bmatrix} T_{i-1} & + & + \\ & 0 & + \\ & & T_{n-i} \end{bmatrix} \begin{bmatrix} y_{i-1} \\ 1 \\ 0_{n-i} \end{bmatrix} = 0_n \quad \Rightarrow \begin{bmatrix} T_{i-1} & + \\ & 0 \end{bmatrix} \begin{bmatrix} y_{i-1} \\ 1 \end{bmatrix} = 0_i$$

注意此处是三角形矩阵，计算量为 $n^2/2$. 依次求出所有特征向量，总计算量$n^3/6$.

```
n = 4;  A = rand(n); A = triu(A);
[S,V] = eig(A);
D = diag(A); S1 = eye(n); E = S1;
for i = 2:n
   B = A - D(i)*E;
   S1(1:i-1,i) = -B(1:i-1,1:i-1)\B(1:i-1,i);
end
norm(A*S1 - S1*V )
```

[17] 也即是矩阵B_i零空间Null Space(核, kernel)的一组基。

9.2 上Hessenberg矩阵的特征向量

采用带位移的QR算法计算Hessenberg矩阵H的特征值时，可以保留与H同阶的三角形矩阵T, 以及记录 QR迭代过程中产生的正交矩阵 $G = G_0G_1...$ 计算出T的特征向量S，则 GS即为H的特征向量，进而 A 的特征向量 $X = QGS$. 这一过程比较繁琐，我们给出先算特征值再计算H特征向量的算法, 求解Hessenberg矩阵的零空间是相对容易的。

例 15 设σ 为Hessenberg矩阵H 的特征值，用反幂法计算其相应的特征向量x. 由特征对定义，x 为下述奇异方程组的非零解.

$$(H-\sigma I)*x=0$$

从数值计算上考虑，可按照反幂法计算特征向量

$$(H-\sigma I)*x=\epsilon \quad \text{OR} \quad (H-\tilde{\sigma} I)*x=\epsilon \quad ,\tilde{\sigma}\approx\sigma$$

后一表达式是为了使系数矩阵可逆，避免数值奇异型.

由于Hessenberg矩阵解线性方程组的代价是$\mathbb{O}(n^2)$, 上述反幂法迭代 只要一步或者两步即收敛，则求出所有特征向量的代价是$\mathbb{O}(n^3)$. 以下代码 给出了求解中小规模矩阵全部特征值的算法全过程. 已经修改正交变换为复数域的算法，对实数矩阵的复特征值和复数矩阵也同样适用。

```
function [Q,D] = myHessQReig(A,Tol)
% comput eigenpair of A by first trans to H and then
% the eigvalue by shift QR iteration, eigvector by inv power iter
% A*Q = Q*D, det(Q)~=0; D:diagonal matrix
if nargin < 2, Tol = 1e-11; end
if norm(triu(A,-1) - A,inf) > 11*eps
   [P,H] = myHess(A);
else
   H = A; P = eye(size(A,1));
end
A = H;          n = size(A,1);   sig = zeros(n,1);
i = 1;          maxI = 10*n;          % tol and max iter steps
while n > 1
    ee = eig_s2(A(n-1:n, n-1:n));             % get eigvalue of 2*2 mat
    [~,ii] = min(abs(ee - A(n,n)) );  ee = ee(ii);   % Wilkinson's shift
       [Q, R] = myGivensQR(A - ee*eye(n) );
       A = R*Q + ee*eye(n);                        % R is triangle, Q:hess
       corn = A(n,n-1);                            % check eps
    if abs(corn) < Tol
       sig(n) = A(n,n);
       n = n-1;     A = A(1:n,1:n);
    end
    i = i + 1;                                  % plus iteration steps
    if i > maxI, warning('Method may not work! '), return , end
```

```
end
fprintf('\n the number of all iteration steps is %d \n',i);
if ~isempty(A), sig(1) = A(1,1); end
% % % compute eigvector of H
sig = sort(sig,'descend');    D = diag(sig);    NL = length(sig);
E = eye(NL); r = 11*eps+zeros(NL,1);   S = 0*D;
for j = 1:NL
    B = round(sig(j),8)*E - H;
    v = B\r;   v = v./norm(v);
    v = B\v;   v = v./norm(v);
    S(:,j) = v;
end
Q = P*S;
```

10 三对角矩阵的性质补充(略)

三对角矩阵的特征值和特征向量计算在 科学计算中经常出现。 从计算量的角度看，可以达到最优。实际上三对角矩阵的特征值理论也很重要。本节给出一个与正交多项式有关的一个性质。

例 16 给定向量a,b,c, 记

$$B_n=\begin{bmatrix} b_1 & c_1 & & & \\ a_1 & b_2 & c_2 & & \\ & \ddots & \ddots & \ddots & \\ & & a_{n-2} & b_{n-1} & c_{n-1} \\ & & & a_{n-1} & b_n \end{bmatrix},\quad B_{n-1}=\begin{bmatrix} b_1 & c_1 & & & \\ a_1 & b_2 & c_2 & & \\ & \ddots & \ddots & \ddots & \\ & & a_{n-3} & b_{n-2} & c_{n-2} \\ & & & a_{n-2} & b_{n-1} \end{bmatrix},$$

则

$$B_n=\begin{bmatrix} B_{n-1} & c_{n-1}e_{n-1} \\ a_{n-1}e_{n-1}^{\mathrm{T}} & b_n \end{bmatrix}$$

其中 e_{n-1}代表$(n-1)$阶单位矩阵的最后一列. 检验

$$\det(B_n)=b_n*\det(B_{n-1})-a_{n-1}c_{n-1}*\det(B_{n-2})$$

根据上例结论，此处也给出一个计算三对角行列式的算法。

```
m = 11;   a =  rand(m-1,1) ; b = rand(m,1); c = rand(m-1,1);
A = diag(a,-1) + diag(b) + diag(c,1);
d1 = b(1);    d2 = b(1)*b(2) - a(1)*c(1);
for k = 1:m-2
    d3 = b(k+2)*d2 - a(k+1)*c(k+1)*d1;
    d1 = d2; d2 = d3;
end
d3 - det(A)
```

若存在另一个三对角矩阵T, 主对角线为b, 次对角线记为d, e, 且满足

$$a_j c_j = d_j e_j, \quad j = 1:n-1$$

则

$$\det(B_n) = \det(T) \Rightarrow B = STS^{-1}$$

一个有趣且重要的结果是:

> 若三对角矩阵T的次对角线对应乘积非负，则其必可对角化，且相似于对称矩阵
>
> $$T' = \text{tridiag}(\sqrt{ac}, b, \sqrt{ac})$$
>
> 所有正交多项式均有三项递推式，利用上述结论可将正交多项式求根问题全部转化为对称三对角 的特征值问题！

已知 n 次正交多项式都满足三项循环式且有n个互异实根， 改写循环式为

$$xp_{j-1} = \frac{c_j}{a_j} p_{j-2} - \frac{b_j}{a_j} p_{j-1} + \frac{1}{a_j} p_j$$

进而得到矩阵形式

$$x \cdot \begin{bmatrix} p_0 \\ p_1 \\ \vdots \\ p_{n-1} \end{bmatrix} = \begin{bmatrix} -b_1/a_1 & 1/a_1 & & \\ c_2/a_2 & -b_2/a_2 & 1/a_2 & \\ & \ddots & \ddots & \\ & & c_n/a_n & -b_n/a_n \end{bmatrix} \cdot P + \frac{p_n(x)}{a_n},$$

上式说明$p_n(x)$的n个根是上述三对角矩阵的特征值！

11 反射矩阵的应用：SVD分解(略)

矩阵的另一个特征称为奇异值(Singular Value)，特别是长方形矩阵本身没有特征值。 将矩阵$A \in \mathbb{R}^{m \times n}$ 看作一个算子，则 对任意向量$x \in \mathbb{R}^n$, $Ax \in \mathbb{R}^m$, 即A是空间R^n到R^m的一个映射。 由线性代数知识， $\mathbb{R}^n$空间是n维线性空间, 存在很多不同的基向量组，特殊的一类是(标准)正交基。一个自然(重要)的数学问题是:

> 是否存在一组$\mathbb{R}^n$空间的标准正交基组成的列向量组V, 满足 AV 是 $\mathbb{R}^m$的一组正交基?
>
> $$AV = U\Sigma, \quad \Sigma = \begin{bmatrix} \sigma_1 & & \\ & \sigma_2 & \\ & & \ddots \end{bmatrix}, \quad U^T U = I.$$

奇异值分解 将矩阵$\boldsymbol{A}$表示为三个特殊矩阵乘积的形式

$$\boldsymbol{A}=\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{\top}=\begin{bmatrix} | & | & & | \\ \boldsymbol{u}_1 & \boldsymbol{u}_2 & \cdots & \boldsymbol{u}_m \\ | & | & & | \end{bmatrix}_{m\times m}\begin{bmatrix} \sigma_1 & & & \\ & \sigma_2 & & \\ & & \ddots & \\ & & & \sigma_n \\ & & & \end{bmatrix}_{m\times n}\begin{bmatrix} - & \boldsymbol{v}_1^{\top} & - \\ - & \boldsymbol{v}_2^{\top} & - \\ & \vdots & \\ - & \boldsymbol{v}_n^{\top} & - \end{bmatrix}_{n\times n}$$

其中 $\boldsymbol{U}$ 是 $\boldsymbol{V}$ 正交的, $\boldsymbol{U}^{\top}\boldsymbol{U}=\boldsymbol{U}\boldsymbol{U}^{\top}=\boldsymbol{I}_m$ 以及 $\boldsymbol{V}^{\top}\boldsymbol{V}=\boldsymbol{V}\boldsymbol{V}^{\top}=\boldsymbol{I}_n$. 矩阵 $\boldsymbol{\Sigma}$ 的前n 行 是对角阵, 对角线是其奇异值 $\sigma_1\geq\sigma_2\geq\cdots\geq\sigma_n\geq 0$.

矩阵的奇异值分解在数值计算中有极其重要的位置。 问题：给定A, 如何寻找相应的奇异值σ以及两组标准正交基V, U？假设$A\in\mathbb{R}^{m\times n}$ 是瘦长型的矩阵, 即$m\geq n$, 则

$$A=U\Sigma V^T\rightarrow A^TA=V\Sigma^2V^T$$

即 $\sigma=\sqrt{\lambda(A^TA)}$, V为A^TA对应的特征向量。 A的奇异值分解转化为计算$A^{\top}A$特征对。步骤如下：

1. 计算$B=A^TA$的特征值σ^2, 及其对应特征向量V
2. $U=AV*(\Sigma)^{-1}$ 即可

Householder变换并不能直接计算奇异值(Singular values)， 但是可以像化简普通矩阵为Hessenberg矩阵一样， 将矩阵约化为两对角矩阵。 假设$A\in\mathbb{R}^{m\times n}$, $(m\leq n)$, 且 P,Q分别是m,n阶正交矩阵， 则[18]

$$A=U\Sigma V^T\rightarrow PAQ=(PU)\cdot\Sigma\cdot(V^TQ)$$

注意正交矩阵的乘积仍是正交矩阵，即PAQ与A具有同样的奇异值。 选择合适的反射矩阵P、Q，有希望使得PAQ形式较A更为简单。

例 17 给出Householder变换，完成如下计算

$$\begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \end{bmatrix}\xrightarrow[\rightarrow]{A\rightarrow B=PAQ}\begin{bmatrix} \times & \times & 0 & 0 & 0 \\ 0 & \times & \times & 0 & 0 \\ 0 & 0 & \times & \times & 0 \end{bmatrix}$$

A的奇异值通过计算B获得，$BB^{\top}$是对称正定三对角，QR分解 [19] 可求出其全部特征值， 其平方根为所求奇异值。顺带计算出$BB^{\top}$特征向量U, 进而计算出V。

```
function [P, H, Q] = myHousBidiag1(A)
% reduce A to P*H*Q, H is bidiag mat, P,Q is orthognal
% size(A,1) <= size(A,2)
% for example:
```

[18] Gilb, Strang, Linear Algebra and its Applications

[19] 对称三对角矩阵的特征值是相对容易的，$\mathbb{O}(n^2)$的计算量就可以完成所有特征对计算.

```
% A = [1 2 3 4 5 ;  4 4 4 4 5;  0 1 -1 1 1];
% [P H Q] = myHousBidiag1(A); H - P*A*Q
[n,m] = size(A);
if n>m, A = A'; [n,m] = size(A);  end
P = eye(n); Q = eye(m); H = A;
for k = 1:n
    j1 = k:n; % col
    x1 = H(j1,k);
    if norm(x1) > 11*eps
         if abs(x1(1)) > 1e-11
            cc =  -x1(1)/abs(x1(1));
         else
            cc = -1;
         end
    rho = -cc*norm(x1);
    w = x1 - [cc*rho; zeros(n-k,1)];
    eta = 1/(w'*x1);
%     w = w/norm(w);
    H(j1,:) = H(j1,:) - eta*w*(w'* H(j1,:));
    P(j1,:) = P(j1,:) - eta*w*(w'* P(j1,:));
    end
%     ---- row
    if k < m
    j2 = k+1:m;
    x2 = H(k,j2); rho = -sign(x2(1))*norm(x2);
    w2 = x2 - rho*[1, zeros(1,m-k-1)];      w2 = w2/norm(w2);
    H(:,j2) = H(:,j2) - 2*(H(:,j2)*w2')*w2;
    Q(:,j2) = Q(:,j2) - 2*(Q(:,j2)*w2')*w2;
    end
end
H =  triu(tril(H,1));
```

类似于变换一般矩阵为hessenberg矩阵，上述过程需要$\mathbb{O}(m\cdot n^2)$运算量。进而求解对称三对角矩阵的特征对，计算量为 $\mathbb{O}(n^2)$.

标注 15 一般三对角矩阵H的特征值计算无法使用对称性，$Q^\top HQ$一般不再是三对角；此时QR迭代的计算量和一般上Hessenberg矩阵的算法代价相同. 但是如果是对称三对角矩阵，可利用对称性减少计算量。

上机练习

⑴ 利用Householder变换约化矩阵

$$A=\begin{bmatrix}1 & 3 & 4\\ 3 & 1 & 2\\ 4 & 2 & 1\end{bmatrix}$$

为上Hessenberg矩阵H, 求出 反射矩阵P 使得

$$H=PAP$$

⑵ 用幂法、反幂法计算矩阵的最大最小特征对，以及QR算法计算

$$A=\begin{pmatrix}1 & 1 & 1/2\\ 1 & 1 & 1/4\\ 1/2 & 1/4 & 2\end{pmatrix}$$

的所有特征值和相应的特征向量.