Numerical Algebra：Homework 13

HUANG Wenchong
3200100006

See my codes in my Github!

June 1, 2022

1 书面题（第六章）

1.1 习题 6.23

题面. 设 $A \in \mathbb{R}^{n\times n}$ 是一个具有互不相同对角元的上三角阵，给出计算 A 的全部特征向量的详细算法.

分析. 注意到上三角阵的对角元就是它的所有特征值，因此 A 具有 n 个互不相同的特征值. 设其对角元依次为 $\lambda_1, \lambda_2, ..., \lambda_n$. 现考虑计算 λ_k 对应的特征向量 v_k. 事实上就是求方程:

$$(A - \lambda_k I)x = 0$$

的任一个非零解. 为此，记:

$$A - \lambda_k I = \begin{pmatrix} U_1 & u & B \\ & 0 & v^T \\ & & U_2 \end{pmatrix}$$

其中 $L_1 \in \mathbb{R}^{(k-1)\times(k-1)}, L_2 \in \mathbb{R}^{(n-k)\times(n-k)}$ 是对角元非零的上三角阵，u, v 是 $k-1$ 维向量. 再记:

$$x = \begin{pmatrix} x_1 \\ c \\ x_2 \end{pmatrix}$$

其中 x_1 是 $k-1$ 维向量，x_2 是 $n-k$ 维向量，c 是一个实数. 那么:

$$(A - \lambda_k I)x = \begin{pmatrix} U_1x_1 + cu + Bx_2 \\ v^T x_2 \\ U_2 x_2 \end{pmatrix}$$

而 U_1, U_2 对角元均非零，故满秩，$U_2x_2 = 0$ 仅有零解，这就确定了 $x_2 = 0$. 再取 $c = 1$，则有:

$$U_1 x_1 = -u$$

由于 U_1 满秩，上述方程有唯一解. 对其求解就确定了 x_1，进而确定了一个特征向量 v_k.

整理上述讨论，得出如下算法.

Algorithm 1: Eigenvectors of an Upper Triangular Matrix with Distinct Diagonal Elements

---

$v_1 \leftarrow (1, 0, 0, \cdots, 0)^T$
for $k = 2 : n$ **do**
  $B \leftarrow A - A(k,k)I$
  Solve: $B(1:k-1, 1:k-1)x = -B(1:k-1, k)$
  $v_k \leftarrow (x^T, 1, 0, \cdots, 0)^T$
end

1.2 习题 6.25

题面. 设 H 是上 Hessenberg 矩阵, 并且假定已经用列主元 Gauss 消去法求得分解 $PH = LU$, 其中 P 是排列方阵, L 是单位下三角阵, U 是上三角阵. 证明: $\tilde{H} = U(P^T L)$ 仍是上 Hessenberg 矩阵, 并且相似于 H.

证明.

由列主元法的变换过程, 知道 $P = P_{n-1} \cdots P_2 P_1$, 其中 P_k 是第 k 步消去的选主元行交换变换, 而由于 H 具有上 Hessenberg 形式, 要么 $P_k = I$, 要么 $P_k = I_{k,(k+1)}$. 从而有:

$$P^T L = P^T P (L_{n-1} P_{n-1} \cdots L_1 P_1)^{-1} = P_1 L_1^{-1} \cdots P_{n-1} L_{n-1}^{-1}$$

其中 L_k 是 Gauss 消元第 k 步中的变换, 由于 H 具有上 Hessenberg 形式, 其变换为将第 k 行的某一倍数加给第 $k+1$ 行, 因此 L_k 具有下述形式:

$$L_k = \begin{array}{cl} \begin{pmatrix} 1 & & & & & \\ & \ddots & & & & \\ & & 1 & & & \\ & & * & 1 & & \\ & & & & \ddots & \\ & & & & & 1 \end{pmatrix} & \begin{array}{l} \\ \\ k \\ k+1 \\ \\ \\ \end{array} \end{array}$$

我们来证明 $P^T L$ 是上 Hessenberg, 为此, 我们断言 $P_{n-k} L_{n-k}^{-1} \cdots P_{n-1} L_{n-1}^{-1}$ 具有如下形式:

$$P_{n-k} L_{n-k}^{-1} \cdots P_{n-1} L_{n-1}^{-1} = \begin{array}{cl} \begin{pmatrix} 1 & & & & & \\ & \ddots & & & & \\ & & * & * & \cdots & * \\ & & * & * & \ddots & \vdots \\ & & & \ddots & \ddots & * \\ & & & & * & * \end{pmatrix} & \begin{array}{l} \\ \\ n-k \\ \\ \\ \\ \end{array} \end{array}$$

对于 $k = 1$ 显然成立, 假设对于某一 $k \geq 1$ 成立, 考虑

$$L_{n-k-1}^{-1} P_{n-k} L_{n-k}^{-1} \cdots P_{n-1} L_{n-1}^{-1} = \begin{array}{cl} \begin{pmatrix} 1 & & & & & & \\ & \ddots & & & & & \\ & & 1 & & & & \\ & & * & * & * & \cdots & * \\ & & & * & * & \ddots & \vdots \\ & & & & \ddots & \ddots & * \\ & & & & & * & * \end{pmatrix} & \begin{array}{l} \\ \\ \\ n-k \\ \\ \\ \\ \end{array} \end{array}$$

再进行一次可能的 $n-k-1$ 行与 $n-k$ 行的交换：

$$P_{n-k}L_{n-k-1}^{-1}P_{n-k}L_{n-k}^{-1}\cdots P_{n-1}L_{n-1}^{-1} = \begin{array}{@{}c@{}l@{}} \begin{pmatrix} 1 & & & & & & \\ & \ddots & & & & & \\ & & 1 & * & * & \cdots & * \\ & & * & * & * & \cdots & * \\ & & & * & * & \ddots & \vdots \\ & & & & \ddots & \ddots & * \\ & & & & & * & * \end{pmatrix} & \begin{array}{l} \\ \\ \\ {\scriptstyle n-k} \\ \\ \\ \\ \end{array} \end{array}$$

这样就得到了 $k+1$ 时我们断言的形式，因此由归纳法可以得证 P^TL 是上 Hessenberg 阵. 而 U 是上三角阵，这样就不难验证 $\tilde{H} = U(P^TL)$ 是上 Hessenberg 阵.

又由 $H = P^TLU$，即得到 $P^TL = HU^{-1}$，从而 $\tilde{H} = UHU^{-1}$，即得相似.

1.3 习题 6.26

题面. 设

$$A = \begin{pmatrix} \alpha_{11} & v^T \\ 0 & T \end{pmatrix} \in \mathbb{R}^{3\times 3}$$

其中 T 是一个有一对复共轭特征值的 2×2 矩阵. 设计一种算法计算一个三阶正交矩阵 Q，使得

$$Q^TAQ = \begin{pmatrix} \tilde{T} & u \\ 0 & \alpha_{11} \end{pmatrix}$$

其中 $\lambda(\tilde{T}) = \lambda(T)$.

分析. 设 x 是矩阵 A^T 对应于特征值 α_{11} 的一个单位特征向量. 即 $A^Tx = \alpha_{11}x$. 由线性空间的基扩展定理，可以取得 $u_1, u_2 \in \mathbb{R}^2$，使 $\{u_1, u_2, x\}$ 构成线性空间 $\mathbb{R}^3$ 的一组单位正交基. 记 $U = (u_1, u_2)$，则有 $x^TU = (x^Tu_1, x^Tu_2) = (0, 0)$，从而：

$$\begin{pmatrix} U^T \\ x^T \end{pmatrix} A \begin{pmatrix} U & x \end{pmatrix} = \begin{pmatrix} U^TAU & U^TAx \\ x^TAU & x^TAx \end{pmatrix} = \begin{pmatrix} U^TAU & U^TAx \\ \alpha_{11}x^TU & \alpha_{11}x^Tx \end{pmatrix} = \begin{pmatrix} U^TAU & U^TAx \\ 0 & \alpha_{11} \end{pmatrix}$$

上述变换并不改变 A 的任何特征值，因此取正交阵 $Q = (u_1, u_2, x)$，再取 $\tilde{T} = U^TAU$，即符合题目要求. 现在考虑如何计算 x, u_1, u_2. 设：

$$\tilde{x} = \begin{pmatrix} c \\ x_1 \end{pmatrix}$$

其中 x_1 是一个二维向量，那么由方程 $(A^T - \alpha_{11}I_3)\tilde{x} = 0$，得：

$$0 = \begin{pmatrix} 0 & 0 \\ v & T^T - \alpha_{11}I_2 \end{pmatrix} \begin{pmatrix} c \\ x_1 \end{pmatrix} = \begin{pmatrix} 0 \\ cv + T^Tx_1 - \alpha_{11}x_1 \end{pmatrix}$$

取 $c = 1$，则方程

$$(T^T - \alpha_{11}I_2)x_1 = -v$$

具有唯一解，这就确定了 x_1，从而确定 $\tilde{x}$，对向量组 $(\tilde{x}, e_2, e_3)$ 进行施密特正交化，然后单位化，即可得到 (x, u_1, u_2).

整理上述讨论，得到如下算法：

Algorithm 2:

Solve: $(T^T - \alpha_{11}I_2)x_1 = -v$

$\tilde{x} \leftarrow (1, x_1^T)^T$

$(x^*, u_1^*, u_2^*) \leftarrow$ **Schmidt Orthogonalization** $(\tilde{x}, e_2, e_3)$

$Q = \left(\frac{x^*}{||x^*||_2}, \frac{u_1^*}{||u_1^*||_2}, \frac{u_2^*}{||u_2^*||_2} \right)$

1.4 习题 6.27

题面. 设 $A \in \mathbb{R}^{n\times n}$ 是一个如下形状的拟上三角阵:

$$A = \begin{pmatrix} A_{11} & A_{12} & \cdots & A_{1k} \\ & A_{22} & \cdots & A_{2k} \\ & & \ddots & \vdots \\ & & & A_{kk} \end{pmatrix}$$

其中 A_{ii} $(i = 1, ..., k)$ 是 1×1 的或是一个有一对复共轭特征值的 2×2 矩阵. 设计一种计算 A 的全部特征向量的数值方法.

分析. 若 A_{ii} 是 1×1 的矩阵，那么它包含了 A 的一个实特征值，此时可以用反幂法求得对应的特征向量 x_i.

现在若 A_{ii} 是 2×2 的矩阵，则存在一对共轭特征值 $a \pm ib$，对应于特征向量 x 和 $\bar{x}$（$\bar{x}$ 表示将 x 的所有分量取共轭）. 现在考虑计算特征向量

用复数形式下的反幂法:

$$(A - aI - ibI)y_{k+1} = u_k$$
$$u_{k+1} = \frac{y_{k+1}}{||y_{k+1}||\infty}$$

记 $y_{k+1} = v_{k+1} + iw_{k+1}$，$u_k = p_k + iq_k$，代入并分别令实部虚部相等，得到:

$$(A - aI)v_{k+1} + bw_{k+1} = p_k \quad (1)$$
$$(A - aI)w_{k+1} - bv_{k+1} = q_k \quad (2)$$

联立消去 w_{k+1}，得到:

$$\left((A - aI)^2 + b^2 I\right) v_{k+1} = (A - aI)p_k - bq_{k+1}$$

取定 u_k 后，由上式算得 v_{k+1}，再由 (1) 式算得 w_{k+1} 即可得到 y_{k+1}. 计算过程需要求解的方程组的系数矩阵是 $(A - aI)^2 + b^2 I$，该矩阵是上 Hessenberg 阵，其求解的计算量为 $O(n^2)$. 一次迭代后即得到达到机器精度的近似特征向量.

2 书面题（第七章）

2.1 习题 7.9

题面. 设 $A \in \mathbb{R}^{n\times n}$ 是对称的，并假定 A 有分解 $A = QTQ^T$，其中 Q 是正交的，T 是对称三对角阵. 试利用比较等式 $AQ = QT$ 两边列向量所得到的公式来设计一个直接计算 Q 和 T 的算法.

分析. 设 $Q = (q_1, q_2, \cdots, q_n)$，以及：

$$T = \begin{pmatrix} \alpha_1 & \beta_1 & & \\ \beta_1 & \alpha_2 & \ddots & \\ & \ddots & \ddots & \beta_{n-1} \\ & & \beta_{n-1} & \alpha_n \end{pmatrix}$$

那么有：

$$AQ = (Aq_1, Aq_2, \cdots, Aq_n)$$

$$QT = (\alpha_1 q_1 + \beta_1 q_2,\ \beta_1 q_1 + \alpha_2 q_2 + \beta_2 q_3,\ \cdots,\ \beta_{n-1} q_{n-1} + \alpha_n q_n)$$

取 $q_1 = e_1$，$\alpha_1 = e_1^T A e_1$，由 $Aq_1 = \alpha_1 q_1 + \beta_1 q_2$，取：

$$\beta_1 = ||Aq_1 - \alpha_1 q_1||$$

$$q_2 = \frac{1}{\beta_1}(Aq_1 - \alpha_1 q_1)$$

对于第 k 列，由 $Aq_k = \beta_{k-1} q_{k-1} + \alpha_k q_k + \beta_k q_{k+1}$，两边左乘 q_k^T，得到：

$$\alpha_k = q_k^T A q_k$$

进而得到：

$$\beta_k = ||Aq_k - \alpha_k q_k - \beta_{k-1} q_{k-1}||$$

$$q_{k+1} = \frac{1}{\beta_k}(Aq_k - \alpha_k q_k - \beta_{k-1} q_{k-1})$$

对 $k = 2, 3, ..., n-1$ 依次执行上述步骤，最后再由第 n 列的关系 $Aq_n = \beta_{n-1} q_{n-1} + \alpha_n q_n$ 两边左乘 q_n^T，得到：

$$\alpha_n = q_n^T A q_n$$

这就完成了计算.

2.2 习题 7.11

题面. 设

$$T=\begin{pmatrix}\alpha_1 & \varepsilon\\ \varepsilon & \alpha_2\end{pmatrix},\qquad \alpha_1\neq\alpha_2,\ \varepsilon\ll 1$$

并假定 $\tilde{T}$ 是以 $\mu=\alpha_2$ 为位移进行了一次对称 QR 迭代得到的矩阵. 试证：$\tilde{T}(2,1)=O(\varepsilon^3)$. 如果改用 Wilkinson 位移，$\tilde{T}(2,1)=?$

证明. 对 $T-\alpha_2 I$ 进行 QR 分解，得到：

$$Q=\frac{1}{\sqrt{(\alpha_1-\alpha_2)^2+\varepsilon^2}}\begin{pmatrix}\alpha_1-\alpha_2 & -\varepsilon\\ \varepsilon & \alpha_1-\alpha_2\end{pmatrix}$$

$$R=\frac{1}{\sqrt{(\alpha_1-\alpha_2)^2+\varepsilon^2}}\begin{pmatrix}(\alpha_1-\alpha_2)^2+\varepsilon^2 & (\alpha_1-\alpha_2)\varepsilon\\ 0 & -\varepsilon^2\end{pmatrix}$$

于是进行一次迭代得到：

$$\tilde{T}=RQ+\alpha_2 I=\frac{1}{(\alpha_1-\alpha_2)^2+\varepsilon^2}\begin{pmatrix}* & *\\ -\varepsilon^3 & *\end{pmatrix}$$

于是由 $\alpha_1\neq\alpha_2$ 以及 $\varepsilon\ll 1$ 得：

$$\tilde{T}(2,1)=\frac{-\varepsilon^3}{(\alpha_1-\alpha_2)^2+\varepsilon^2}=O(\varepsilon^3)$$

如果改用 Wilkinson 位移，则应取：

$$\delta=(\alpha_1-\alpha_2)/2$$
$$\mu=\frac{\alpha_1+\alpha_2}{2}-\mathrm{sgn}(\delta)\sqrt{\delta^2+\varepsilon^2}$$

记 $M=\mathrm{sgn}(\delta)\sqrt{\delta^2+\varepsilon^2}$，从而：

$$T-\mu I=\begin{pmatrix}\delta+M & \varepsilon\\ \varepsilon & -\delta+M\end{pmatrix}$$

对其进行 QR 分解，得：

$$Q=\frac{1}{\sqrt{(\delta+M)^2+\varepsilon^2}}\begin{pmatrix}\delta+M & -\varepsilon\\ \varepsilon & \delta+M\end{pmatrix}$$

$$R=\frac{1}{\sqrt{(\delta+M)^2+\varepsilon^2}}\begin{pmatrix}(\delta+M)^2+\varepsilon^2 & 2\varepsilon M\\ 0 & -\varepsilon^2+(M-\delta)^2\end{pmatrix}$$

于是进行一次迭代得到：

$$\tilde{T}=RQ+\mu I=\frac{1}{\sqrt{(\delta+M)^2+\varepsilon^2}}\begin{pmatrix}* & *\\ -\varepsilon^3+\varepsilon(M-\delta)^2 & *\end{pmatrix}$$

于是：

$$\tilde{T}(2,1)=\frac{2(\delta-M)\delta\varepsilon}{\sqrt{(\delta+M)^2+\varepsilon^2}}=\begin{cases}O(\varepsilon^2), & \mathrm{sgn}(\delta)=1\\ O(1), & \mathrm{sgn}(\delta)=-1\end{cases}$$

2.3　习题 7.18

题面. 设

$$A = \begin{pmatrix} \alpha_1 & \beta_1 & & \\ \gamma_1 & \alpha_2 & \ddots & \\ & \ddots & \ddots & \beta_{n-1} \\ & & \gamma_{n-1} & \alpha_n \end{pmatrix}$$

其中 $\gamma_i\beta_i > 0$. 证明：存在对角阵 D，使得 $D^{-1}AD$ 为对称三对角阵.

分析. 设 $D = \mathrm{diag}(d_1, d_2, \cdots, d_n)$，且对角元非零，则：

$$D^{-1}AD = \begin{pmatrix} \alpha_1 & d_1^{-1}d_2\beta_1 & & \\ d_2^{-1}d_1\gamma_1 & \alpha_2 & \ddots & \\ & \ddots & \ddots & d_{n-1}^{-1}d_n\beta_{n-1} \\ & & d_n^{-1}d_{n-1}\gamma_{n-1} & \alpha_n \end{pmatrix}$$

要使其成为对称三对角阵，则以下方程应满足：

$$\begin{cases} d_1^{-1}d_2\beta_1 = d_2^{-1}d_1\gamma_1 \\ d_2^{-1}d_3\beta_2 = d_3^{-1}d_2\gamma_2 \\ \vdots \\ d_{n-1}^{-1}d_n\beta_{n-1} = d_n^{-1}d_{n-1}\gamma_{n-1} \end{cases}$$

取 $d_1 = 1$，依次代入方程各式得：

$$\begin{cases} d_2 = \sqrt{\dfrac{\gamma_1}{\beta_1}} > 0 \\ d_3 = d_2\sqrt{\dfrac{\gamma_2}{\beta_2}} > 0 \\ \vdots \\ d_n = d_{n-1}\sqrt{\dfrac{\gamma_{n-1}}{\beta_{n-1}}} > 0 \end{cases}$$

按上式递推计算即可得到符合要求的 D.

3 上机题报告（第六章）

3.1 隐式 QR 算法求实矩阵全部特征值的通用子程序设计

主程序如下：

```
function D = eigen(A)
% 求矩阵的特征根
    n = size(A,1);
    [H,~] = realSchur(A);
    D = zeros(n,1);
    i = 1;
    while i<=n
        if i==n || H(i+1,i)==0
            D(i) = H(i,i);
            i = i+1;
        else
            D(i:i+1) = getComplexEigen(H(i:i+1,i:i+1));
            i = i+2;
        end
    end
end
```

其中 realSchur 是计算实 Schur 分解的程序，如下：

```
function [H,Q] = realSchur(A)
% REALSCHUR 计算实矩阵的实Schur分解：隐式QR算法
    n = size(A,1);
    [H,Q] = hessenberg(A);
    u = 1e-16;
    while true
        % 收敛性判定
        for i = 2:n
            if abs(H(i,i-1)) <= (abs(H(i,i))+abs(H(i-1,i-1)))*u
                H(i,i-1) = 0;
            end
        end
        m = 0;
        while m<n
            if m==n-1 || abs(H(n-m,n-m-1))<u
                m = m+1;
            else
                if (m==n-2 || abs(H(n-m-1,n-m-2))<u) && isComplexEigen(H(n-m-1:n-m,
                    n-m-1:n-m))
                    m = m+2;
                else
                    break;
                end
            end
```

```
        end
        if m==n
            break;
        end
        l = n-m;
        while l>1 && abs(H(l,l-1))>=u
            l = l-1;
        end
        l = l-1;
        % 进入双重步隐式QR迭代
        [H(l+1:n-m,l+1:n-m),P] = doubleQR(H(l+1:n-m,l+1:n-m));
        Q = Q * blkdiag(eye(l),P,eye(m));
        H(1:l,l+1:n-m) = H(1:l,l+1:n-m)*P;
        H(l+1:n-m,n-m+1:n) = P'*H(l+1:n-m,n-m+1:n);
    end
    for k = 3:n
        H(k,1:k-2) = zeros(1,k-2);
    end
end
```

程序中的 hessenberg() 是将矩阵上 Hessenberg 化的程序，doubleQR 是双重步位移隐式 QR 迭代的程序，如下：

```
function [H,Q] = hessenberg(A)
% 计算矩阵A的上Hessenberg分解, H=Q'AQ
    n = size(A,1);
    Q = eye(n);
    for k = 1:n-2
        [v,beta] = householder(A(k+1:n,k));
        Hk = (eye(n-k)-beta*(v*v'));
        A(k+1:n,k:n) = Hk*A(k+1:n,k:n);
        A(1:n,k+1:n) = A(1:n,k+1:n)*Hk;
        Q = Q * blkdiag(eye(k), Hk);
    end
    H = A;
    for k = 3:n
        H(k,1:k-2) = zeros(1,k-2);
    end
end

function [H,P] = doubleQR(H)
% Francis双重步位移QR迭代算法
    n = size(H,1);
    P = eye(n);
    m = n-1;
    s = H(m,m)+H(n,n);
    t = H(m,m)*H(n,n)-H(m,n)*H(n,m);
    x = H(1,1)*H(1,1)+H(1,2)*H(2,1)-s*H(1,1)+t;
```

```
    y = H(2,1)*(H(1,1)+H(2,2)-s);
    if n >= 3
        z = H(2,1)*H(3,2);
    end
    for k = 0:n-3
        [v,beta] = householder([x,y,z]');
        q = max([1,k]);
        Pk = eye(3)-beta*(v*v');
        H(k+1:k+3,q:n) = Pk*H(k+1:k+3,q:n);
        r = min([k+4,n]);
        H(1:r,k+1:k+3) = H(1:r,k+1:k+3)*Pk;
        x = H(k+2,k+1);
        y = H(k+3,k+1);
        if k < n-3
            z = H(k+4,k+1);
        end
        P = P * blkdiag(eye(k),Pk,eye(n-k-3));
    end
    [v,beta] = householder([x,y]');
    Pk = eye(2)-beta*(v*v');
    if n >= 3
        H(n-1:n,n-2:n) = Pk*H(n-1:n,n-2:n);
    end
    H(1:n,n-1:n) = H(1:n,n-1:n)*Pk;
    P = P * blkdiag(eye(n-2),Pk);
    for k = 3:n
        H(k,1:k-2) = zeros(1,k-2);
    end
end
```

程序中用到了 isComplexEigen 和 getComplexEigen 函数，是用一元二次方程判别式来判断 2×2 的矩阵是否具有复特征根，并且计算其复特征根的程序，如下：

```
function res = isComplexEigen(A)
% 判断一个2*2实矩阵的特征值是不是复数
    res = ((A(1,1)+A(2,2))*(A(1,1)+A(2,2)) - 4*det(A) < 0);
end

function res = getComplexEigen(A)
% 求一个2*2实矩阵的复特征值
    res = zeros(2,1);
    delta = (A(1,1)+A(2,2))*(A(1,1)+A(2,2)) - 4*det(A);
    res(1) = (A(1,1)+A(2,2)+sqrt(delta))/2;
    res(2) = (A(1,1)+A(2,2)-sqrt(delta))/2;
end
```

3.2 上机习题 6.2(2)

以下程序被设计用于求解多项式的全部根

```
function [root] = allroot(a)
    n = size(a,1);
    A = zeros(n,n);
    A(2:n,1:n-1) = eye(n-1);
    A(1:n,n) = -a;
    root = eigen(A);
end
```

以下脚本用于求解本题（附件 ex_6_2_2.m）：

```
n = 41;
x = zeros(41,1);
x(1) = 1;
x(4) = 1;
allroot(x)
```

运行结果如下：

```
ans =


           1.01430466733554 +    0.0809230298548768i
           1.01430466733554 -    0.0809230298548768i
          0.987183858031794 +     0.240354383678765i
          0.987183858031794 -     0.240354383678765i
           0.93366404473162 +     0.392546384549113i
           0.93366404473162 -     0.392546384549113i
          0.855158027643731 +     0.532633535495491i
          0.855158027643731 -     0.532633535495491i
          0.753719530443121 +     0.655380276002736i
          0.753719530443121 -     0.655380276002736i
          0.632339827064371 +     0.753401199923816i
          0.632339827064371 -     0.753401199923816i
          0.507568639467913 +     0.810573438852718i
          0.507568639467913 -     0.810573438852718i
          0.417151578255587 +     0.871067309516253i
          0.417151578255587 -     0.871067309516253i
           0.28981213106784 +     0.946423674668758i
           0.28981213106784 -     0.946423674668758i
          0.139165434976384 +     0.992476733700879i
          0.139165434976384 -     0.992476733700879i
        -0.0197286322077255 +      1.00935215687363i
        -0.0197286322077255 -      1.00935215687363i
         -0.180206043239115 +     0.997962232725531i
         -0.180206043239115 -     0.997962232725531i
          -0.33698432323146 +     0.959227612653095i
```

```
         -0.33698432323146 -     0.959227612653095i
        -0.485280239299887 +     0.894538369839528i
        -0.485280239299887 -     0.894538369839528i
        -0.620672505470617 +     0.805889307058931i
        -0.620672505470617 -     0.805889307058931i
        -0.739100565190882 +     0.695904353165252i
        -0.739100565190882 -     0.695904353165252i
        -0.836863004990835 +     0.567825669085266i
        -0.836863004990835 -     0.567825669085266i
        -0.910511134933716 +     0.425528151315794i
        -0.910511134933716 -     0.425528151315794i
        -0.956339011328335 +     0.273776209087026i
        -0.956339011328335 -     0.273776209087026i
        -0.968140341518281 +     0.120866955421148i
        -0.968140341518281 -     0.120866955421148i
        -0.952483875214081 +                     0i
```

3.3 上机习题 6.2(3)

测试代码如下（修改 x 的值可以对 $x = 0.9, 1.0, 1.1$ 分别进行测试）（附件 ex_6_2_3.m）：

```
x = 0.9;
A = [[9.1,3.0,2.6,4.0];[4.2,5.3,4.7,1.6];[3.2,1.7,9.4,x];[6.1,4.9,3.5,6.2]];
eigen(A)
```

测试结果如下：

```
x = 0.9,  ans =
          17.4396781909386 +                     0i
          2.87040173577024 +     0.642891129472461i
          2.87040173577024 -     0.642891129472461i
          6.81951833752094 +                     0i
x = 1.0,  ans =
          17.4764849155294 +                     0i
          2.86799924645605 +     0.688747355259161i
          2.86799924645605 -     0.688747355259161i
          6.78751659155854 +                     0i
x = 1.1,  ans =
           17.513028071474 +                     0i
          2.86545787067426 +     0.732169739115249i
          2.86545787067426 -     0.732169739115249i
          6.75605618717752 +                     0i
```

可以看到 A 的特征值变化并不大. 随着 x 在 0.1 的尺度下变化，特征值的变化不超过 0.05

4 上机题报告（第七章）

4.1 隐式对称 QR 算法求实对称矩阵全部特征值和特征向量的通用子程序设计

主程序如下：

```
function [T,Q] = symmetricEigen(A)
% 计算实对称矩阵的特征值和特征向量
    n = size(A,1);
    [T,Q] = tridiag(A);
    u = 1e-16;
    while true
        % 收敛性判定
        for i = 1:n-1
            if abs(T(i+1,i)) <= (abs(T(i,i))+abs(T(i+1,i+1)))*u
                T(i+1,i) = 0;
                T(i,i+1) = 0;
            end
        end
        m = 0;
        while m<n && (m==n-1 || abs(T(n-m,n-m-1))<u)
            m = m+1;
        end
        if m==n
            break;
        end
        l = n-m;
        while l>1 && abs(T(l,l-1))>=u
            l = l-1;
        end
        l = l-1;
        % 进入Wilkinson位移隐式对称QR迭代
        [T(l+1:n-m,l+1:n-m),G] = wilkinsonQR(T(l+1:n-m,l+1:n-m));
        Q = Q * blkdiag(eye(l),G',eye(m));
    end
    T = diag(diag(T));
end
```

在运算前需要将矩阵三对角化，程序中的 tridiag 就是实对称矩阵三对角化的程序，如下：

```
function [A,Q] = tridiag(A)
% 计算矩阵的三对角分解
    n = size(A,1);
    Q = eye(n);
    for k = 1:n-2
        [v,beta] = householder(A(k+1:n,k));
        u = beta*(A(k+1:n,k+1:n)*v);
        w = u-(beta*(u'*v)/2)*v;
        A(k+1,k) = norm(A(k+1:n,k),2);
```

```
        A(k,k+1) = A(k+1,k);
        A(k+1:n,k+1:n) = A(k+1:n,k+1:n)-v*w'-w*v';
        A(k+2:n,k) = zeros(n-k-1,1);
        A(k,k+2:n) = zeros(1,n-k-1);
        Q = Q * blkdiag(eye(k),eye(n-k)-beta*(v*v'));
    end
end
```

主程序中用到带 Wilkinson 位移的隐式对称 QR 迭代，如下：

```
function [T,Q] = wilkinsonQR(T)
% 带Wilkinson位移的隐式对称QR迭代
    n = size(T,1);
    d = (T(n-1,n-1)-T(n,n))/2;
    mu = T(n,n)+d-sign(d)*sqrt(d*d+T(n,n-1)*T(n,n-1));
    x = T(1,1)-mu;
    z = T(2,1);
    Q = eye(n);
    for k = 1:n-1
        G = givensMatrix(x,z,k,k+1,n);
        T = G*T*G';
        Q = G*Q;
        if k < n-1
            x = T(k+1,k);
            z = T(k+2,k);
        end
    end
end
```

其中用到的 givensMatrix 是计算 givens 变换矩阵的程序，如下：

```
function [c,s] = givens(a,b)
% givens变换
    if b==0
        c = 1; s = 0;
    else
        if abs(b)>abs(a)
            tau = a/b; s = 1/sqrt(1+tau^2); c = s*tau;
        else
            tau = b/a; c = 1/sqrt(1+tau^2); s = c*tau;
        end
    end
end

function G = givensMatrix(a,b,i,j,n)
% 返回对i,j行列变换的givens变换矩阵
    [c,s] = givens(a,b);
    G = eye(n);
    G(i,i) = c;
```

```
    G(i,j) = s;
    G(j,i) = -s;
    G(j,j) = c;
end
```

4.2 上机习题 7.1(2)

测试代码如下（附件 ex_7_1.m）：

```
n = 100;
A = zeros(n,n);
for i = 1:n-1
    A(i,i) = 4;
    A(i+1,i) = 1;
    A(i,i+1) = 1;
end
A(n,n) = 4;
[D,Q] = symmetricEigen(A)
max(max(abs(A*Q-Q*D)))
```

程序将会输出 Q, D，满足 $A = QDQ^T$，其中 D 是对角元为 A 的所有特征值的对角阵，Q 是正交阵，其第 j 列是 D 的第 j 个对角元对应的特征向量. 可以调整 $n = 50, 51, ..., 100$ 以测试各阶数. 经测试当 $n = 50$ 时迭代次数为 113，而当 $n = 100$ 时迭代次数为 207. 现考虑测试结果的精度. 用：

$$||Ax - \lambda x||_\infty$$

来衡量对于特征值 λ 和对应特征向量的计算精度. 用：

$$\max_\lambda ||Ax - \lambda x||_\infty = \max_{i,j} |(AQ - QD)_{ij}|$$

来衡量对于 A 的所有特征值和特征向量的计算精度. 测试时，设置机器精度 $u = 10^{-16}$，测试结果表明，当 $n = 50$ 时上式的值为 6.44×10^{-15}，当 $n = 100$ 时上式的值为 7.66×10^{-15}. 这在预设的机器精度下是极好的结果.