HW8

Saturday, April 13, 2024 8:09 PM

1. **[Q1]** $(25 = 10 + 10 + 5$ pts) **(a)** Let H be the initial input matrix for the shifted QR iteration. Show that $(H - \mu^{(k)}I)\cdots(H-\mu^{(2)}I)(H-\mu^{(1)}I) = \underline{Q}^{(k)}\underline{R}^{(k)}$ (in practice, H is upper Hessenberg, but we do not need this assumption here)

   (b) Other than the single real Wilkinson shift, we may also let $\mu^{(k)} = h_{nn}^{(k-1)}$ if $h_{n(n-1)}^{(k-1)}$ is small. Assume, for example, that $H^{(k-1)} = \begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ 0 & 0 & \times & \times & \times \\ 0 & 0 & 0 & \delta & h_{nn}^{(k-1)} \end{bmatrix}$. After the application of $n-2$ Givens rotations to $H^{(k-1)} - h_{nn}^{(k-1)}I$, we have the intermediate matrix $H_{tmp}^{(k-1)} = \begin{bmatrix} \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ 0 & 0 & \times & \times & \times \\ 0 & 0 & 0 & a & b \\ 0 & 0 & 0 & \delta & 0 \end{bmatrix}$ (make sure you understand why it is of this form), and the last Givens rotation is needed on the left before we compute $H^{(k)}$ by transposed Givens rotations. Show that the new matrix $H^{(k)} = R^{(k)}Q^{(k)} + h_{nn}^{(k-1)}I$ satisfies $h_{n(n-1)}^{(k)} = -\frac{b\delta^2}{a^2+\delta^2}$. What does this observation suggest, if $\left|h_{n(n-1)}^{(k-1)}\right| = |\delta| \ll 1$, and either $|b| < 2|a|$ (δ can be arbitrary) or if $|\delta| < \frac{a^2}{|b|}$?

   (c) What can we say about $h_{n(n-1)}^{(k)}$ if A is real symmetric, such that $H^{(k-1)}$ is also real symmetric (hence tridiagonal)? In particular, does this entry decrease more slowly or more rapidly in the symmetric case than in the nonsymmetric case?

a)
$$\begin{aligned}
\underline{Q}^{(k)}\underline{R}^{(k)} &= Q^{(1)}\cdots Q^{(k)}R^{(k)}\cdots R^{(1)} \\
&= Q^{(1)}\cdots Q^{(k-1)}(A^{(k-1)}-\mu^{(k)}I)R^{(k-1)}\cdots R^{(1)} \\
&= Q^{(1)}\cdots Q^{(k-1)}Q^{(k-1)T}(A^{(k-2)}-\mu^{(k)}I)Q^{(k-1)}R^{(k-1)}\cdots R^{(1)} \\
&= Q^{(1)}\cdots Q^{(k-2)}(A^{(k-2)}-\mu^{(k)}I)(A^{(k-2)}-\mu^{(k-1)}I)R^{(k-2)}\cdots R^{(1)} \\
&= Q^{(1)}\cdots Q^{(k-2)}Q^{(k-2)T}(A^{(k-3)}-\mu^{(k)}I)Q^{(k-2)}Q^{(k-2)T}(A^{(k-3)}-\mu^{(k)}I)Q^{(k-2)}R^{(k-2)}\cdots R^{(1)} \\
&= Q^{(1)}\cdots Q^{(k-3)}(A^{(k-3)}-\mu^{(k)}I)(A^{(k-3)}-\mu^{(k-1)}I)(A^{(k-3)}-\mu^{(k-2)}I)R^{(k-3)}\cdots R^{(1)} \\
&\ \vdots \\
&= \Pi_{j=k}^{1}(A-\mu^{(j)}I)
\end{aligned}$$

bi) Since $h_{n(n-1)}^{(k)}$ is only updated by the last Given's rotation, consider,

$$\left[\begin{array}{c|c} I_3 & 0 \\ \hline 0 & \begin{matrix} c & -s \\ s & c \end{matrix} \end{array}\right]\left[\begin{array}{c|c} \nabla & X \\ \hline 0 & \begin{matrix} a & b \\ \delta & 0 \end{matrix} \end{array}\right]\left[\begin{array}{c|c} I_3 & 0 \\ \hline 0 & \begin{matrix} c & s \\ -s & c \end{matrix} \end{array}\right]$$

where $\begin{bmatrix} c & -s \\ s & c \end{bmatrix}$ is the Given's rotation based on $\begin{bmatrix} a \\ \delta \end{bmatrix}$

$$= \left[\begin{array}{c|c} \nabla & X \\ \hline 0 & \begin{matrix} \frac{a^2-b^2}{\sqrt{a^2+\delta^2}} & \frac{ba}{\sqrt{a^2+\delta^2}} \\ 0 & \frac{\delta b}{\sqrt{a^2+\delta^2}} \end{matrix} \end{array}\right]\left[\begin{array}{c|c} I_3 & 0 \\ \hline 0 & \begin{matrix} c & s \\ -s & c \end{matrix} \end{array}\right]$$

$$\begin{bmatrix} & \mid & 0 & \frac{\delta b}{\sqrt{a^2+\delta^2}} \end{bmatrix}\begin{bmatrix} 0 & \mid & -s & c \end{bmatrix}$$

$$= \left[\begin{array}{c|cc} \nabla & \multicolumn{2}{c}{X} \\ \hline 0 & X & X \\ & \frac{-b\delta^2}{a^2+\delta^2} & X \end{array}\right] \quad \text{Thus, } h^{(k)}_{n(n-1)} = \frac{-b\delta^2}{a^2+\delta^2}$$

bii) $\frac{b\delta^2}{a^2+\delta^2} < \frac{4b\delta^2}{b^2+4\delta^2} < \frac{4b\delta^2}{b^2+4\delta^2} < \frac{4\delta^2}{b} < 4\delta^2 << 4$

biii) $\frac{b\delta^2}{a^2+\delta^2} < \frac{a^2}{b(a^2+\delta^2)} < \frac{\delta}{a^2+\delta^2} < \delta << 1$

This suggest $h^{(k)}_{n(n-1)} \to 0$

c) $$\left[\begin{array}{c|cc} I_3 & \multicolumn{2}{c}{0} \\ \hline 0 & c & -s \\ & s & c \end{array}\right]\left[\begin{array}{ccc|cc} x & x & o & \multicolumn{2}{c}{0} \\ o & x & x & & \\ o & \delta & x & & \\ \hline \multicolumn{3}{c|}{0} & a & \delta \\ & & & \delta & 0 \end{array}\right]\left[\begin{array}{c|cc} I_3 & \multicolumn{2}{c}{0} \\ \hline 0 & c & s \\ & -s & c \end{array}\right]$$ where $\begin{bmatrix} c & -s \\ s & c \end{bmatrix}$ is the Given's rotation based on $\begin{bmatrix} a \\ \delta \end{bmatrix}$

$$= \left[\begin{array}{c|cc} \nabla & \multicolumn{2}{c}{X} \\ \hline 0 & \frac{a^2-\delta^2}{\sqrt{a^2+\delta^2}} & \frac{\delta a}{\sqrt{a^2+\delta^2}} \\ & 0 & \frac{\delta^2}{\sqrt{a^2+\delta^2}} \end{array}\right]\left[\begin{array}{c|cc} I_3 & \multicolumn{2}{c}{0} \\ \hline 0 & c & s \\ & -s & c \end{array}\right]$$

$$= \left[\begin{array}{c|cc} \nabla & \multicolumn{2}{c}{X} \\ \hline 0 & X & X \\ & -\frac{\delta^3}{a^2+\delta^2} & X \end{array}\right] \quad \text{Thus, } h^{(k)}_{n(n-1)} = -\frac{\delta^3}{a^2+\delta^2}$$

Which will converge faster to zero.

2. **[Q2]** (15 pts) Implement the single-shift QR step in MATLAB; that is, given an upper Hessenberg $H^{(k-1)}$ and shift $\mu^{(k)}$, we compute $Q^{(k)}R^{(k)} = H^{(k-1)} - \mu^{(k)}I$ by Givens rotations and then use these Givens rotations to compute $H^{(k)} = R^{(k)}Q^{(k)} + \mu^{(k)}I$. Make simple changes in my code to enforce the use of single (Wilkinson) shift only, even if complex arithmetic is needed. Assemble your single shift code with the uploaded subroutines. Test it with the matrix obtained by

```
load west0479;
A = full(west0479);
```

2. **[Q2]** (15 pts) Implement the single-shift QR step in MATLAB; that is, given an upper Hessenberg $H^{(k-1)}$ and shift $\mu^{(k)}$, we compute $Q^{(k)}R^{(k)} = H^{(k-1)} - \mu^{(k)}I$ by Givens rotations and then use these Givens rotations to compute $H^{(k)} = R^{(k)}Q^{(k)} + \mu^{(k)}I$. Make simple changes in my code to enforce the use of single (Wilkinson) shift only, even if complex arithmetic is needed. Assemble your single shift code with the uploaded subroutines. Test it with the matrix obtained by

```
load west0479;
A = full(west0479);
```

Compare the eigenvalues of your final $H^{(k)}$ (use `ordeig`) with those of A. Be aware that the ordering of eigenvalues must be consistent to make a meaningful comparison.

Calculating $\frac{\|\Lambda(A)-\Lambda(H)\|}{\|\Lambda(A)\|} \sim O(10^{-11})$ which about the same as I got testing with Matlab's QR.

3. **[Q3]** (10 pts) Implement the Arnoldi's method without and with reorthogonalization, and test the orthogonality of the column vectors in U_{50} for the matrix A generated by `u = cos((0:2048)/2048*pi); A = vander(u);` Is the reorthogonalization effective for generating an orthonormal basis?

Use Arnoldi with reorthogonalization to compute the 11 dominant eigenvalues and eigenvectors of `aerofoil_new`, using $m = 30, 60, 100,$ and 150 dimensional Krylov subspaces. For each m, plot all eigenvalues $\{\lambda_i\}_{i=1}^n$ of A together with the eigenvalues $\{\mu_i\}_{i=1}^m$ of H_m on the complex plane. Intuitively, how do $\{\mu_i\}_{i=1}^m$ approximate $\{\lambda_i\}_{i=1}^n$ as m increases? Give the relative eigenresidual norm $\frac{\|AU_mw_i-\mu_iU_mw_i\|_2}{\|AU_mw_i\|_2}$ $(1 \le i \le 11)$ of the desired eigenpairs for each m in a table.

a) Reorthogonalization was effective $\|V'V - I\| \sim O(\varepsilon_m)$

The eigvals seem to cluster at the beginning where the most real eigvals are, as m increases, there is more spread to cover each part of the spectrum.

Pictures and tabel in code

```
load west0479;
C=full(west0479);

[H,Q,iter] = HW8_QReig(C);
E=ordeig(H);
E=sort(E);
D=eig(C);
D=sort(D);
norm(E-D)/norm(D)

u=cos((0:2048)/2048*pi);
B=vander(u);
flag=true;

[V,H]=AR(B,50,flag);

norm(V'*V-eye(51))

l=zeros(11,4);
flag=true;

m=30;
[V,H]=AR(A,m,flag);
[Veig,Deig]=eig(H(1:end-1,:));
[Deig,idx]=sort(diag(Deig),'descend');
Veig=Veig(:,idx);
Veig=V(:,1:end-1)*Veig;

for i=1:11
    l(i,1)=norm(A*Veig(:,i)-Deig(i)*Veig(:,i))/norm(A*Veig(:,i));
end

tiledlayout(3,2)
nexttile
plot(real(ev),imag(ev),"o")
nexttile
plot(real(Deig),imag(Deig),"o")

m=60;
[V,H]=AR(A,m,flag);
[Veig,Deig]=eig(H(1:end-1,:));
[Deig,idx]=sort(diag(Deig),'descend');
Veig=Veig(:,idx);
Veig=V(:,1:end-1)*Veig;

for i=1:11
    l(i,2)=norm(A*Veig(:,i)-Deig(i)*Veig(:,i))/norm(A*Veig(:,i));
end

nexttile
plot(real(Deig),imag(Deig),"o")
```

```
m=100;
[V,H]=AR(A,m,flag);
[Veig,Deig]=eig(H(1:end-1,:));
[Deig,idx]=sort(diag(Deig),'descend');
Veig=Veig(:,idx);
Veig=V(:,1:end-1)*Veig;

for i=1:11
    l(i,3)=norm(A*Veig(:,i)-Deig(i)*Veig(:,i))/norm(A*Veig(:,i));
end

nexttile
plot(real(Deig),imag(Deig),"o")

m=150;
[V,H]=AR(A,m,flag);
[Veig,Deig]=eig(H(1:end-1,:));
[Deig,idx]=sort(diag(Deig),'descend');
Veig=Veig(:,idx);
Veig=V(:,1:end-1)*Veig;

for i=1:11
    l(i,4)=norm(A*Veig(:,i)-Deig(i)*Veig(:,i))/norm(A*Veig(:,i));
end

nexttile
plot(real(Deig),imag(Deig),"o")
l

function [V,H]=AR(A,m,flag)
    [n,~]=size(A);
    r0=zeros(n,1);
    r0(1)=1;
    H=zeros(m+1,m);
    V=zeros(n,m+1);
    V(:,1)=r0/norm(r0);
    for k=1:m
        w=A*V(:,k);
        for i=1:k
            H(i,k)=V(:,i)'*w;
            w=w-H(i,k)*V(:,i);
        end
        if flag==true
            for i=1:k
                dH(i,k)=V(:,i)'*w;
                w=w-dH(i,k)*V(:,i);
                H(i,k)=H(i,k)+dH(i,k);
            end
        end
        H(k+1,k)=norm(w);
        V(:,k+1)=w/norm(w);
    end
end
```

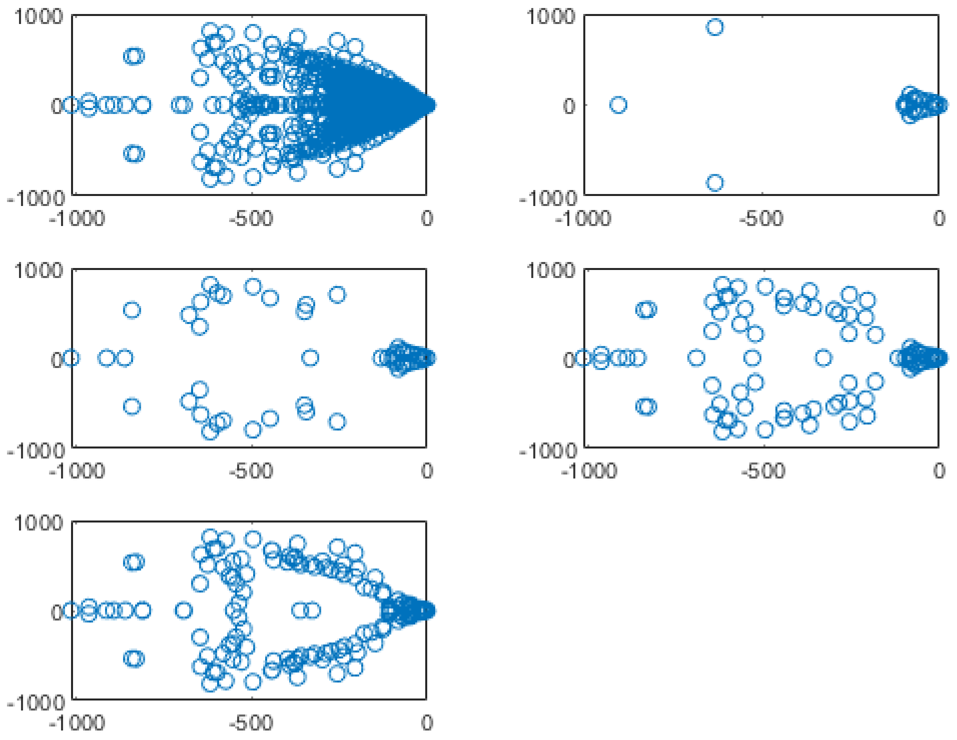
```
function [H,Q,iter] = HW8_QReig(A)

[m,n] = size(A);
if m ~= n
    error('Input matrix must be square!');
end
tol = eps/4;

A = full(A);

tstart1 = tic;
[H,Q] = HW8_HHrdcUH(A);
tend1 = toc(tstart1);
fprintf('Phase I: matrix reduced to a similar upper Hessenberg in %.2f
secs.\n',tend1);

maxiter = n*4;
q = 0;
tmpv = randn(length(A),1);  tmpv = tmpv/norm(tmpv);
if isreal(A) && norm(A*tmpv-(tmpv'*A)')/norm(A,'fro') >= 4*eps
    maxq = n-2;     real_nonsymm = true;
else
    maxq = n-1;     real_nonsymm = false;
end
iter = 1;
tstart2 = tic;
while q < maxq && iter <= maxiter

    for k = 1 : n-1
        if abs(H(k+1,k)) <= tol*(abs(H(k,k))+abs(H(k+1,k+1)))
            H(k+1,k) = 0;
        end
    end

    oldq = q;
    for j = n-oldq : -1 : 2
        if H(j,j-1) == 0 || (j > 2 && H(j,j-1) ~= 0 && H(j-1,j-2) == 0 &&
real_nonsymm)
            q = q + 1;
        else
            break;
        end
    end
    subdgH1n2 = diag(H(1:n-q,1:n-q),-1);
    p = find(subdgH1n2 == 0,1,'last');
    if isempty(p),  p = 0;  end
    sizeH22 = n-p-q;
    if q < maxq
        if sizeH22 >= 2
            evs = eig(H(n-q-1:n-q,n-q-1:n-q));
        else
            evs = H(n-q,n-q);
        end
        if isreal(evs) || ~real_nonsymm
```

```
            [~,idx] = min(abs(evs-H(n-q,n-q)));
            [H(p+1:n-q,p+1:n-q),GCS] = HW8_SingleShiftedQRstep(H(p+1:n-
q,p+1:n-q),evs(idx));
            for k = 1 : sizeH22-1
                Gt = [conj(GCS(1,k)) -GCS(2,k); conj(GCS(2,k))
conj(GCS(1,k))];
                Q(:,p+k:p+k+1) = Q(:,p+k:p+k+1)*Gt;
                H(1:p,p+k:p+k+1) = H(1:p,p+k:p+k+1)*Gt;
                H(p+k:p+k+1,n-q+1:end) = Gt'*H(p+k:p+k+1,n-q+1:end);
            end
        else
            [H(p+1:n-q,p+1:n-q),HVs] = HW8_DoubleShiftedQRstep(H(p+1:n-
q,p+1:n-q),H(n-q-1:n-q,n-q-1:n-q));
            for k = 1 : sizeH22-2
                Q(:,p+k:p+k+2) = Q(:,p+k:p+k+2)-
Q(:,p+k:p+k+2)*(2*HVs(:,k))*HVs(:,k)';
                H(1:p,p+k:p+k+2) = H(1:p,p+k:p+k+2)-
H(1:p,p+k:p+k+2)*(2*HVs(:,k))*HVs(:,k)';
                H(p+k:p+k+2,n-q+1:end) = H(p+k:p+k+2,n-
q+1:end)-(2*HVs(:,k))*(HVs(:,k)'*H(p+k:p+k+2,n-q+1:end));
            end
            Q(:,n-q-1:n-q) = Q(:,n-q-1:n-q)-Q(:,n-q-1:n-
q)*(2*HVs(1:2,end))*HVs(1:2,end)';
            H(1:p,n-q-1:n-q) = H(1:p,n-q-1:n-q)-H(1:p,n-q-1:n-
q)*(2*HVs(1:2,end))*HVs(1:2,end)';
            H(n-q-1:n-q,n-q+1:end) = H(n-q-1:n-q,n-
q+1:end)-(2*HVs(1:2,end))*(HVs(1:2,end)'*H(n-q-1:n-q,n-q+1:end));
        end
    end
    if iter == 1 || mod(iter,100) == 0 || q >= maxq
        fprintf('Iteration %d: %d eigenvalues have converged.\n',iter,q);
    end
    if q >= maxq
        fprintf('All %d eigenvalues have converged in Schur form in %d
iterations.\n',n,iter);
        break;
    end
    iter = iter + 1;
end

tend2 = toc(tstart2);

if q < maxq
    fprintf('Maximum iteration (%d) reached; %d eigenvalues have
converged.\n',maxiter,q);
end

fprintf('Phase II: the shifted QR iteration took %.2f seconds.\n',tend2);
end

function [H,V] = HW8_simreductHess(A)

[m,n] = size(A);
if m ~= n
```

```
    error('A must be a square matrix.');
end
tmpv = randn(length(A),1);  tmpv = tmpv/norm(tmpv);
if norm(A*tmpv-(tmpv'*A)')/norm(A,'fro') <= 2*eps
    symm = true;
else
    symm = false;
end
V = zeros(n,n-2);
for k = 1 : n-2
    xk = A(k+1:n,k);
    signxk1 = sign(xk(1));
    if signxk1 == 0,    signxk1 = 1;    end
    vk = -signxk1*norm(xk)*eye(n-k,1)-xk;
    vk = vk/norm(vk);   vk = vk/norm(vk);
    A(k+1:n,k) = -signxk1*norm(xk)*eye(n-k,1);
    A(k+1:n,k+1:n) = A(k+1:n,k+1:n) - 2*vk*(vk'*A(k+1:n,k+1:n));
    A(:,k+1:n) = A(:,k+1:n) - 2*(A(:,k+1:n)*vk)*vk';
    V(k+1:n,k) = vk;
end
if ~symm,   H = A;
else,       H = tril(A,1);
end

end

function [H,Q] = HW8_SingleShiftedQRstep(H0,evs)
    [m,n] = size(H0);
    mu = evs*eye(m);
    Q=eye(m);
    R0=(H0-mu);
    GCS=zeros(n-1,2);
    for k=1:n-1
        [c,s] = givens(R0(k:k+1,k));
        G=[c,-s;s,c];
        R0(k:k+1,k:end)=G*R0(k:k+1,k:end);
        R0(k+1,k)=0;

        Q(1:k+1,k:k+1)=Q(1:k+1,k:k+1)*G';
        GCS(k,:)=[c,s];
    end
    H=R0*Q+mu;
    for j=1:n
        for i=j:n
            if abs(H(i,j))<10^-16
                H(i,j)=0;
            end
        end
    end
end

function [c,s] = givens(u)
    if u(2) == 0
        c = 1;  s = 0;
    else
        if abs(u(2)) > abs(u(1))
            tau = -u(1)/u(2);   s = 1/sqrt(1+tau^2);    c = s*tau;
        else
            tau = -u(2)/u(1);   c = 1/sqrt(1+tau^2);    s = c*tau;
        end
    end
end

function [H,Q] = HW8_HHrdcUH(A)

% Householder similar reduction of a square matrix A into upper Hessenberg
%
% Input:  A is the input square matrix, real or complex
% Output: H is an upper Hessenberg, similar to A
%         Q is orthogonal/unitary, such that Q'*A*Q = H numerically
%
% Copyright (c) 2017, F. Xue
%

[m,n] = size(A);
if m ~= n
    error('A must be an mxn matrix where n >= n.');
end
V = zeros(n,n-2);
for k = 1 : n-2
    xk = A(k+1:n,k);
    signxk1 = sign(xk(1));
    if signxk1 == 0,    signxk1 = 1;    end
    vk = signxk1*norm(xk)*eye(n-k,1)+xk;
    vk = vk/norm(vk);   vk = vk/norm(vk);
    A(k+1:n,k) = -signxk1*norm(xk)*eye(n-k,1);
    A(k+1:n,k+1:n) = A(k+1:n,k+1:n) - 2*vk*(vk'*A(k+1:n,k+1:n));
    A(:,k+1:n) = A(:,k+1:n) - 2*(A(:,k+1:n)*vk)*vk';
    V(k+1:n,k) = vk;
end

H = A;

Q = eye(n,n);
for k = n-2:-1:1
    Q(k+1:n,k+1:n) =
Q(k+1:n,k+1:n)-2*V(k+1:n,k)*(V(k+1:n,k)'*Q(k+1:n,k+1:n));
end

end

function [Hnew,HVs] = HW8_DoubleShiftedQRstep(H,Mu2)

[m2,n2] = size(Mu2);
if m2 ~= 2 || n2 ~= 2 || ~isreal(Mu2)
    error('Wrong input Mu2: it must be a 2x2 real matrix.');
end
[m,n] = size(H);
```

```
if m ~= n
    error('Input H is not a square matrix!');
end
if nnz(tril(H,-2)) > 0
    error('Input H is not upper Hessenberg!');
end
if ~isreal(H)
    error('H is not real!');
end

HVs = zeros(3,n-1);

s = Mu2(1,1)+Mu2(2,2);
t = Mu2(1,1)*Mu2(2,2)-Mu2(1,2)*Mu2(2,1);

x = H(1,1)^2+H(1,2)*H(2,1)-s*H(1,1)+t;
y = H(2,1)*(H(1,1)+H(2,2)-s);
z = H(2,1)*H(3,2);
for k = 0 : n-3
    if y == 0 && z == 0
        continue;
    end
    signx = sign(x);
    if signx == 0,   signx = 1;  end
    vk = signx*norm([x; y; z])*eye(3,1) + [x; y; z];
    vk = vk/norm(vk);   vk = vk/norm(vk);
    HVs(:,k+1) = vk;
    q = max([1 k]);
    H(k+1:k+3,q:n) = H(k+1:k+3,q:n) - (2*vk)*(vk'*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) - (H(1:r,k+1:k+3)*(2*vk))*vk';
    x = H(k+2,k+1);
    y = H(k+3,k+1);
    if k < n-3
        z = H(k+4,k+1);
    end
end
if y ~= 0
    signx = sign(x);
    if signx == 0,  signx = 1;  end
    vk = signx*norm([x; y])*eye(2,1) + [x; y];
    vk = vk/norm(vk);   vk = vk/norm(vk);
    HVs(1:2,n-1) = vk;
    H(n-1:n,n-2:n) = H(n-1:n,n-2:n)-(2*vk)*(vk'*H(n-1:n,n-2:n));
    H(1:n,n-1:n) = H(1:n,n-1:n)-(H(1:n,n-1:n)*(2*vk))*vk';
end
Hnew = triu(H,-1);
end
```

```
Phase I: matrix reduced to a similar upper Hessenberg in 1.14 secs.
Iteration 1: 0 eigenvalues have converged.
Iteration 100: 43 eigenvalues have converged.
Iteration 200: 103 eigenvalues have converged.
Iteration 300: 163 eigenvalues have converged.
Iteration 400: 234 eigenvalues have converged.
Iteration 500: 308 eigenvalues have converged.
Iteration 600: 373 eigenvalues have converged.
Iteration 700: 443 eigenvalues have converged.
Iteration 768: 477 eigenvalues have converged.
All 479 eigenvalues have converged in Schur form in 768 iterations.
Phase II: the shifted QR iteration took 10.95 seconds.

ans =

     2.840478422734245e-11


ans =

     1.017962789714332e-15


l =

  Columns 1 through 3

   0.041784219538352   0.000007049484270   0.000000000000017
   0.041784219538352   0.000007049484270   0.000000000000017
   0.476845655943187   0.000041294951621   0.000000000007971
   0.007086813340415   0.000097920756724   0.000000000011333
   0.007086813340415   0.000097920756724   0.000000000011333
   0.003469836862336   0.013307600226945   0.000000003901824
   0.003469836862336   0.013307600226945   0.000000003901824
   0.027645650955478   0.000108792962341   0.000000003829891
   0.027645650955478   0.000108792962341   0.000000003829891
   0.003212962275781   0.005447000020748   0.000003653642646
   0.003212962275781   0.007180199074471   0.000003653642646

  Column 4

   0.000000000000003
   0.000000000000003
   0.000000000000007
   0.000000000000006
   0.000000000000006
   0.000000000000004
   0.000000000000004
   0.000000000000006
   0.000000000000006
   0.000000000000005
   0.000000000000005
```



Published with MATLAB® R2023b