MATH-6600, CLA Problem Set No. 9, 12-3-15

Name: Michael Hennessey

1. Let $A \in \mathbb{C}^{m\times m}$ be given, not necessarily hermitian. Show that a number $z \in \mathbb{R}$ is a Rayleigh quotient of A if and only if it is a diagonal entry of Q^*AQ for some unitary matrix Q. Thus Rayleigh quotients are just diagonal entries of matrices, once you transform orthogonally to the right coordinate system.

   Solution:

   $\Rightarrow$ We assume that z is a Rayleigh quotient of A and that it has the form

   $$z = \frac{x^*Ax}{x^*x}$$

   for some vector x. We can rewrite slightly to show that we are multiplying A by a vector of length 1:

   $$z = \frac{x^*}{\|x\|}A\frac{x}{\|x\|}.$$

   Setting $q = \dfrac{x}{\|x\|}$ results in the form:

   $$z = q^*Aq.$$

   We can find $m-1$ normalized m-dimensional vectors orthogonal to q and form a unitary matrix Q with these vectors and q as the columns. Then z will be the diagonal entry of the matrix Q^*AQ corresponding to the location of q such that if q is the ith column of Q, then z is the ii entry of Q^*AQ.

   $\Leftarrow$ We assume that z is a diagonal entry of Q^*AQ, for some unitary Q. To show that z is a Rayleigh quotient of A, we begin by expanding the numerator of the Rayleigh quotient:

   $$x^*Ax = x^*\begin{bmatrix} a_1^*x \\ a_2^*x \\ a_3^*x \end{bmatrix}$$

   Then it is easy to see that when we carry out the matrix multiplication Q^*AQ the diagonal entries are of this form:

   $$AQ = \begin{bmatrix} a_1^*q_1 & a_1^*q_2 & a_1^*q_3 \\ a_2^*q_1 & a_2^*q_2 & a_2^*q_3 \\ a_3^*q_1 & a_3^*q_2 & a_3^*q_3 \end{bmatrix}.$$

$$\{Q^*AQ\}_{ii} = q_i^* \begin{bmatrix} a_1^* q_i \\ a_2^* q_i \\ a_3^* q_i \end{bmatrix} = q_i^* A q_i.$$

Then since $q_i^* q_i = 1$, the diagonal entries of Q^*AQ are the Rayleigh quotients of A. Clearly this holds for any $m \times m$ matrices Q and A.

2. Let $A \in \mathbb{C}^{m \times m}$ have real eigenvalues

$$\lambda_1 > \lambda_2 \geq \lambda_3 \geq \cdots \geq \lambda_{m-1} > \lambda_m$$

and a complete set of eigenvectors. Consider the power method but for the matrix $A - \mu I$ with shift μ. What value of μ gives the fastest convergence to the eigenvector corresponding to λ_1? What is the rate of convergence?

$\mu \approx \lambda_2$ gives the fastest convergence to the eigenvector corresponding to λ_1. We show this by using the result found in class for the form of $v^{(k)}$:

$$v^{(k)} = c_k \lambda_1^k [a_1 q_1 + a_2 \left(\frac{\lambda_2}{\lambda_1}\right)^k q_2 + ... + a_m \left(\frac{\lambda_m}{\lambda_1}\right)^k q_m]$$

Then, if the λ_i are the eigenvalues of $B = A - \mu I$ and if we denote the eigenvalues of A as $\hat{\lambda}$ we can rewrite $v^{(k)}$ as

$$v^{(k)} = c_k (\hat{\lambda}_1^k - \mu)[a_1 q_1 + a_2 \left(\frac{\hat{\lambda}_2 - \mu}{\hat{\lambda}_1 - \mu}\right)^k q_2 + ... + a_m \left(\frac{\hat{\lambda}_m - \mu}{\hat{\lambda}_1 - \mu}\right)^k q_m]$$

Then our shifts must satisfy

$$\hat{\lambda}_1 - \mu > \hat{\lambda_2} - \mu \geq \hat{\lambda_3} - \mu \geq ... \geq \hat{\lambda}_{m-1} - \mu > \hat{\lambda}_m - \mu$$

and therefore

$$1 > \frac{\hat{\lambda_2} - \mu}{\hat{\lambda}_1 - \mu} \geq \frac{\hat{\lambda_3} - \mu}{\hat{\lambda_1} - \mu} \geq ... \geq \frac{\hat{\lambda_{m-1}} - \mu}{\hat{\lambda_1} - \mu} > \frac{\hat{\lambda}_m - \mu}{\hat{\lambda}_1 - \mu}$$

Then if all $\hat{\lambda_i} > 0$ the best shift to choose would be $\mu \approx \hat{\lambda}_m$. This is so because it would essentially eliminate the final term in the iteration of $v^{(k)}$. Similarly, if all $\hat{\lambda_i} < 0$, the best shift to choose for finding $\hat{\lambda_1}$ would be $\mu \approx \hat{\lambda_m}$. We make this choice here to force $\hat{\lambda_1} - \mu$ to be the largest eigenvalue in our shifted matrix.

However, the difficult case comes when $\hat{\lambda_2}$ and $\hat{\lambda_m}$ are of different signs. Then, the best choice of a shift is the average of those two eigenvalues. This is so because if we choose a shift $\mu \approx \hat{\lambda_2}$ then $\hat{\lambda_m} - \mu$ is going to be quite large, possibly larger than $\hat{\lambda_2}$. A similar problem occurs if we choose $\mu \approx \hat{\lambda_2}$. Thus, to minimize the overall maximum of the $\hat{\lambda_i} - \mu$ (which will be either of the cases above), we choose $\mu \approx \left|\frac{\hat{\lambda_m} - \hat{\lambda_2}}{2}\right|$.

The convergence rate will remain linear, as we are only removing one small term for the case when all eigenvalues are positive. For all eigenvalues negative, we are converging on the smallest eigenvalue in magnitude instead of the largest with our chosen shift, but this is essentially the same as the power iteration without a shift. Therefore we have linear convergence here as well.

3. Suppose that $A \in \mathbb{R}^{m\times m}$ is symmetric and tridiagonal. Show that if we perform the shifted QR step,

$$A - \mu I = QR,$$
$$\bar{A} = RQ + \mu I$$

then $\bar{A}$ is also symmetric and tridiagonal. This shows that the shifted QR algorithm does not destroy the banded structure of A.

Solution:

We begin by forming $\bar{A}$:

$$A - \mu I = QR \implies Q^*(A - \mu I) = R$$
$$\implies Q^*(A - \mu I)Q = RQ$$

Then,

$$Q^*(A - \mu I)Q + \mu I = RQ + \mu I$$
$$\bar{A} = Q^*(A - \mu I)Q + \mu I.$$

To show that $\bar{A}$ is symmetric, we show

$$\bar{A}^* = (Q^*(A - \mu I)Q + \mu I)^* = Q^*(A_\mu I)^*Q + \mu I = Q^*(A - \mu I)Q + \mu I = \bar{A}.$$

Then we use the fact that the QR factorization of $A-\mu I$ produces an upper-Hessenburg matrix Q to show that $\bar{A}$ is tridiagonal. Since Q is upper-Hessenburg, Q^* is lower-Hessenburg. Therefore,

$$Q^*(A_\mu I)$$

will produce a lower-Hessenburg matrix. Then, multiplying this lower-Hessenburg matrix by the upper-Hessenburg matrix Q, we are left with a tridiagonal matrix.

4. QR program. Let $A \in \mathbb{R}^{m\times m}$ be the real symmetric tridiagonal matrix with 2 on the diagonal and -1 on the sub- and super- diagonals,

$$A = \begin{bmatrix} 2 & 1 & & & \\ -1 & 2 & -1 & & \\ & -1 & 2 & -1 & \\ & & \ddots & \ddots & \ddots \\ & & & -1 & 2 \end{bmatrix}.$$

Let $\delta(k)$ denote the maximum of the absolute values of the off-diagonal entries of the QR matrix $A^{(k)}$.
Write a program that computes the eigenvalues of A using the unshifted QR eigenvalue algorithm using the Matlab `qr` function. After each 10 iterations, $k = 10, 20, 30, ...$, print k, $\delta(k)$ and the convergence ratio $r(k) = \dfrac{\delta(k)}{\delta(k-1)}$. For example, you could use a statement such as

```
fprintf('QR: k=%d : delta=%8.2e, ratio=%5.3f\n',k,delta,delta/deltaOld);
```

Continue iterating until $\delta(k) < tol = 10^{-5}$.

```
function [schurA,eigA,maxerror]=unshiftedQREigenvalue(A)
%This algorithm takes an arbitrary matrix A and iterates the pure QR
%algorithm to produce a Schur form for A
tol=10^(-5);
k=1;
[~,m]=size(A);
delta=1;
deltaOld=1;
while delta>=tol
    [Q,R]=qr(A);
    A=R*Q;
    deltaM=A;

    for i=1:m
        deltaM(i,i)=0;
    end
    delta=max(max(abs(deltaM)));

    if mod(k,10)==0
        fprintf('QR:k=%d : delta=%8.2e, ratio=%5.3f\n',k,delta,delta/deltaOld);
    end
    deltaOld=delta;
    k=k+1;
end
schurA=A;
eigA=diag(A);
E=eig(A);
x=eigA-E;
maxerror=max(abs(x));
```

- Show results for $m = 11$.
- After convergence print the eigenvalues found.
- Also print the maximum error in the eigenvalues compared to those obtained from the Matlab routine `eig`

```
[schurA,eigA,maxerror]=unshiftedQREigenvalue(A)
```

```
QR:k=10 : delta=4.83e-01, ratio=0.920
QR:k=20 : delta=1.77e-01, ratio=0.937
QR:k=30 : delta=8.19e-02, ratio=0.918
QR:k=40 : delta=4.56e-02, ratio=0.954
QR:k=50 : delta=2.79e-02, ratio=0.951
QR:k=60 : delta=1.68e-02, ratio=0.950
QR:k=70 : delta=1.00e-02, ratio=0.949
QR:k=80 : delta=5.95e-03, ratio=0.949
QR:k=90 : delta=3.53e-03, ratio=0.949
QR:k=100 : delta=2.10e-03, ratio=0.949
QR:k=110 : delta=1.24e-03, ratio=0.949
QR:k=120 : delta=7.39e-04, ratio=0.949
QR:k=130 : delta=4.39e-04, ratio=0.949
QR:k=140 : delta=2.60e-04, ratio=0.949
QR:k=150 : delta=1.55e-04, ratio=0.949
QR:k=160 : delta=9.18e-05, ratio=0.949
QR:k=170 : delta=5.45e-05, ratio=0.949
QR:k=180 : delta=3.23e-05, ratio=0.949
QR:k=190 : delta=1.92e-05, ratio=0.949
QR:k=200 : delta=1.14e-05, ratio=0.949

schurA =

  Columns 1 through 5

    3.9319   -0.0000   -0.0000   -0.0000   -0.0000
   -0.0000    3.7321   -0.0000    0.0000    0.0000
         0   -0.0000    3.4142   -0.0000    0.0000
         0         0   -0.0000    3.0000   -0.0000
         0         0         0   -0.0000    2.5176
         0         0         0         0   -0.0000
         0         0         0         0         0
         0         0         0         0         0
         0         0         0         0         0
         0         0         0         0         0
         0         0         0         0         0

  Columns 6 through 10

    0.0000    0.0000   -0.0000   -0.0000   -0.0000
   -0.0000    0.0000    0.0000   -0.0000   -0.0000
    0.0000   -0.0000   -0.0000    0.0000    0.0000
   -0.0000    0.0000    0.0000   -0.0000   -0.0000
   -0.0000   -0.0000   -0.0000    0.0000    0.0000
    2.0000   -0.0000   -0.0000   -0.0000    0.0000
   -0.0000    1.4824    0.0000   -0.0000   -0.0000
         0   -0.0000    1.0000    0.0000    0.0000
         0         0   -0.0000    0.5858    0.0000
         0         0         0   -0.0000    0.2679
         0         0         0         0    0.0000

eigA =

    3.9319
```

```
    3.7321
    3.4142
    3.0000
    2.5176
    2.0000
    1.4824
    1.0000
    0.5858
    0.2679
    0.0681


maxerror =

   4.7514e-10
```

- What is the numerically observed asymptotic convergence ratio $r(k)$? Is the convergence linear? Can you relate the convergence rate to the eigenvalues of A?

  Solution:

  The numerically observed asymptotic convergence ratio $r(k)$ approaches .95 as k gets large. The convergence appears to be linear. Interestingly, $r(k) \to \dfrac{\lambda_1}{\lambda_2}$ as $k \to \infty$.

5. Shifted QR program. Repeat question (4) but write a code using the shifted QR eigenvalue algorithm with deflation. Choose the shift based on the WIlkinson shift. The eienvalues should converge from the bottom right corner. When the off-diagonal entry next to the bottom corner has magnitude less than $tol = 10^{-5}$, declare the eigenvalue to be converged and deflate the matrix to be one dimension less by eliminating the last row and last column. Continue the shifted QR with deflation on the remaining smaller matrix.

```
        function [eigA,maxerror]=shiftedQREigenvalue(A)
%This algorithm takes an arbitrary matrix A and iterates the practical QR
%algorithm with shifts to produce a Schur form for A

%Initialize tolerance and variables
tol=10^(-5);
k=1;
delta=1;
deltaOld=1;
[~,m]=size(A);
E=eig(A);
%loop the practical QR algorithm until the tolerance is satisfied
while m>1
    %Choose the Wilkinson Shift for this step
    del=(A(m-1,m-1)-A(m,m))/2;
    if del==0
```

```
        mu=A(m,m)+A(m,m-1)^2/(abs(del)+sqrt(del^2+A(m,m-1)^2));
    else
        mu=A(m,m)-sign(del)*A(m,m-1)^2/(abs(del)+sqrt(del^2+A(m,m-1)^2));
    end

    %QR factorize A-mu I
    [Q,R]=qr(A-mu*eye(m));
    %Recombine factors in reverse order
    A=R*Q+mu*eye(m);

    delta=A(m,m-1);

    fprintf('QR:k=%d : shift=%5.3f, dimension=%d, delta=%8.2e, ratio=%5.3f\n',k,mu,m,


    %Test the convergence of the eigenvalue in the row m, save the
    %converged eigenvalue and deflate the matrix
    if delta<tol
        eigA(m)=A(m,m);
        if m==2
            eigA(m-1)=A(m-1,m-1);
        end

        A=A(1:m-1,1:m-1);
    end
    [~,m]=size(A);
    deltaOld=delta;
    k=k+1;
end
eigA=sort(eigA);
x=eigA'-E;
maxerror=max(abs(x));
```

- Show results for $m = 11$.
- After each QR iteration, $k = 1, 2, 3, ...$ print k, the shift $\mu(k)$, the current dimension of the matrix $m(k)$, the value of $\delta(k)$ and the convergence ratio $r(k) = \dfrac{\delta(k)}{\delta(k-1)}$, $k = 1, 2, 3, ...$ Continue iterating until all eigenvalues have been found.
- After convergence, print the eigenvalues found.
- Also print the maximum error in the eigenvalues compared to those obtained from the Matlab routine `eig`.

```
                [eigA,maxerror]=shiftedQREigenvalue(A)
    QR:k=1 : shift=3.000, dimension=11, delta=9.95e-17, ratio=0.000
    QR:k=2 : shift=1.000, dimension=10, delta=-2.87e-16, ratio=-2.888
    QR:k=3 : shift=3.000, dimension=9, delta=6.61e-01, ratio=-2301097811745637.000
    QR:k=4 : shift=3.257, dimension=9, delta=2.45e-01, ratio=0.370
    QR:k=5 : shift=3.420, dimension=9, delta=-1.78e-03, ratio=-0.007
    QR:k=6 : shift=2.363, dimension=8, delta=-1.03e-01, ratio=57.866
    QR:k=7 : shift=3.712, dimension=7, delta=-1.50e-02, ratio=0.146
```

```
QR:k=8 : shift=1.849, dimension=6, delta=9.52e-02, ratio=-6.329
QR:k=9 : shift=2.077, dimension=6, delta=-9.85e-03, ratio=-0.103
QR:k=10 : shift=1.465, dimension=5, delta=-3.62e-03, ratio=0.367
QR:k=11 : shift=3.929, dimension=4, delta=-1.28e-03, ratio=0.353
QR:k=12 : shift=0.582, dimension=3, delta=1.81e-03, ratio=-1.413
QR:k=13 : shift=0.586, dimension=3, delta=1.47e-09, ratio=0.000
QR:k=14 : shift=0.268, dimension=2, delta=-8.64e-18, ratio=-0.000

eigA =

  Columns 1 through 8

    0.0681    0.2680    0.5858    1.0000    1.4831    2.0079    2.5139
3.0000

  Columns 9 through 11

    3.4142    3.7275    3.9314


maxerror =

    0.0079
```

- How does the convergence of the shifted QR algorithm compare to the unshifted? Does the convergence rate look linear?

  The convergence of the shifted QR algorithm is considerably faster than the unshifted algorithm, though it does sacrifice a considerable amount of accuracy keeping the level of tolerance equal between the two. In fact, running the shifted algorithm with an absurd tolerance level, `tol=10^(-50)`, will bring the maximum error in the eigenvalues down to `maxerror = 4.3960e-11`, which is only one order of magnitude more accurate than the unshifted algorithm with `tol=10^(-5)`.

  As evidenced by the rapidity of the convergence, the convergence rate of the shifted QR algorithm is clearly nonlinear. Theoretically the convergence rate should be cubic, but the convergence rate oscillates wildly in our numeric result.