W.D. Henshaw **Math 6800: Solutions for Problem Set 9**

1. (10 pts) Let $A \in \mathbb{R}^{m\times m}$ be a real symmetric matrix.

(a) Prove that the Rayeligh quotient $r(x)$, for any vector $x \in \mathbb{R}^m$, lies in the interval $[\lambda_{\rm min}, \lambda_{\rm max}]$, where $\lambda_{\rm min}$ is the smallest eigenvalue and $\lambda_{\rm max}$ the largest eigenvalue of A.

(b) Suppose that the eigenvalues of A satisfy,

$$|\lambda_1| > |\lambda_2| > |\lambda_3| > \dots \tag{1}$$

and let q_i denote the corresponding orthonormal eigenvectors. Given an initial guess $v^{(0)}$, how fast would the power method converge (in exact arithmetic) if

$$q_1^T v^{(0)} = 0, \qquad q_i^T v^{(0)} \neq 0, \quad i = 2, 3, \dots, m? \tag{2}$$

Explain your result. What would likely happen using floating point arithmetic with finite precision?

Solution:

(a) Let $\lambda_i \in \mathbb{R}$ and $q_i \in \mathbb{R}^m$ denote the eigenvalues and eigenvectors of A with $q_i^T q_j = \delta_{ij}$. We can write x as

$$x = \sum_{i=1}^m a_i q_i \tag{3}$$

Then

$$x^T A x = \Big(\sum_{j=1}^m a_j q_j\Big)^T \Big(\sum_{i=1}^m a_i \lambda_i q_i\Big)^T = \sum_{i=1}^m a_i^2 \lambda_i \tag{4}$$

and

$$r(x) = \frac{x^T A x}{x^T x} = \frac{\sum_{i=1}^m a_i^2 \lambda_i}{\sum_{i=1}^m a_i^2} \tag{5}$$

Since

$$\lambda_{\rm min} = \frac{\sum_{i=1}^m a_i^2 \lambda_{\rm min}}{\sum_{i=1}^m a_i^2} \le \frac{\sum_{i=1}^m a_i^2 \lambda_i}{\sum_{i=1}^m a_i^2} \le \frac{\sum_{i=1}^m a_i^2 \lambda_{\rm max}}{\sum_{i=1}^m a_i^2} = \lambda_{\rm max} \tag{6}$$

then

$$\lambda_{\rm min} \le r(x) \le \lambda_{\rm max}. \tag{7}$$

(b) Let $v^{(0)}$ be written as

$$v^{(0)} = \sum_{i=2}^m a_i q_i \tag{8}$$

then

$$v^{(n)} = c_n A^n v^{(0)} = c_n \sum_{i=2}^{m} a_i A^n q_i = c_n \sum_{i=2}^{m} a_i \lambda_i^n q_i, \tag{9}$$

$$= c_n a_2 \lambda_2^n \left(q_2 + \sum_{i=3}^{m} \frac{a_i}{a_2} \left(\frac{\lambda_i}{\lambda_2} \right)^n q_i \right) \tag{10}$$

where c_n is chosen to make $\|v^{(n)}\| = 1$. The convergence will thus be

$$\|v^{(n)} - (\pm) q_2\| = \mathcal{O}\Big(\left| \frac{\lambda_3}{\lambda_2} \right|^n \Big), \tag{11}$$

and the asymptotic convergence rate is

$$\text{rate} = \frac{\lambda_3}{\lambda_2} \tag{12}$$

In floating point arithmetic it is likely that $q_1^T v^{(n)}$ will become non-zero due to round-off error and the power method will eventually converge to q_1 and λ_1 at rate λ_1/λ_2

2. (20 pts) Let A be the $m \times m$ tridiagonal matrix with entries

$$a_{i,i-1} = -1,$$
$$a_{ii} = 4 + i,$$
$$a_{i,i+1} = -1.$$

(a) Write a Matlab code to use the power method to find the largest eigenvalue (denoted by λ) and corresponding eigenvector (devoted by v). Take $m = 10$ and use an initial guess of $v^{(0)} = [1, 1, 1, \ldots, 1]^T$. Use the Matlab function `eig` to compute the exact answer for comparison. Perform `maxit=50` iterations, and at each iteration k, print the current estimate $\lambda^{(k)}$, the error in $\lambda^{(k)}$, the 2-norm of the error in $v^{(k)}$ and the ratio of the error in $v^{(k)}$ at step k to the previous step $k-1$. Use the following statement to output the result:

```
fprintf('k=%4d lambda=%18.14f error=%8.2e, v-err=%8.2e ratio=%8.5f\n',k,...
        lambda,abs(lambda-lambda1),vErr,ratio);
```

The ratio should approach a certain value. Explain where this value comes from.

(b) Write a Matlab code to use the Rayleigh quotient iteration to find a eigenvalue/eigenvector pair of A. Take $m = 10$ and choose the initial guess $v^{(0)} = [1, 1, 1, \ldots, 1]^T$, and $\lambda^{(0)} = (v^{(0)})^T A v^{(0)}$. Perform `maxit=5` iterations, and at each iteration k print the current estimate $\lambda^{(k)}$, the error in $\lambda^{(k)}$, the 2-norm of the error in the eigenvector $v^{(k)}$ and the ratio of the error in $v^{(k)}$ at step k to *cube of the error* at the previous step $k-1$ (e.g. `ratio=vErr/(vErrOld^3)`). Use the following statement to output the result:

```
fprintf('k=%4d lambda=%18.14f, error=%8.2e, v-err=%8.2e, ratio=%8.5f\n',k,...
        lambda,abs(lambda-lambda1),vErr,ratio);
```

Solution:

(a) The code for the power method is given below. Here are the results:

```
>> powerMethod
lambda1=14.746194, lambda2=13.210679, lambda2/lambda1 =  0.89587
k=   1 lambda=  9.73707533234860 error=5.01e+00, v-err=1.25e+00 ratio= 0.39165
k=   2 lambda= 11.05988335914097 error=3.69e+00, v-err=1.17e+00 ratio= 0.93520
k=   3 lambda= 11.89831252031907 error=2.85e+00, v-err=1.09e+00 ratio= 0.92760
k=   4 lambda= 12.50570116419184 error=2.24e+00, v-err=1.00e+00 ratio= 0.91975
k=   5 lambda= 12.98000433803850 error=1.77e+00, v-err=9.13e-01 ratio= 0.91246
k=   6 lambda= 13.35850150944157 error=1.39e+00, v-err=8.27e-01 ratio= 0.90632
k=   7 lambda= 13.66027653534853 error=1.09e+00, v-err=7.46e-01 ratio= 0.90152
k=   8 lambda= 13.89894732993262 error=8.47e-01, v-err=6.70e-01 ratio= 0.89801
k=   9 lambda= 14.08609281596197 error=6.60e-01, v-err=6.00e-01 ratio= 0.89560
k=  10 lambda= 14.23187724497717 error=5.14e-01, v-err=5.36e-01 ratio= 0.89406
k=  11 lambda= 14.34499977932975 error=4.01e-01, v-err=4.79e-01 ratio= 0.89315
k=  12 lambda= 14.43264659258538 error=3.14e-01, v-err=4.27e-01 ratio= 0.89270
k=  13 lambda= 14.50057846347998 error=2.46e-01, v-err=3.82e-01 ratio= 0.89256
k=  14 lambda= 14.55331497121092 error=1.93e-01, v-err=3.41e-01 ratio= 0.89261
k=  15 lambda= 14.59435296089282 error=1.52e-01, v-err=3.04e-01 ratio= 0.89278
k=  16 lambda= 14.62637667282058 error=1.20e-01, v-err=2.72e-01 ratio= 0.89302
k=  17 lambda= 14.65143932001440 error=9.48e-02, v-err=2.43e-01 ratio= 0.89328
k=  18 lambda= 14.67111067046611 error=7.51e-02, v-err=2.17e-01 ratio= 0.89356
k=  19 lambda= 14.68659271450290 error=5.96e-02, v-err=1.94e-01 ratio= 0.89382
k=  20 lambda= 14.69880839615397 error=4.74e-02, v-err=1.73e-01 ratio= 0.89407
k=  21 lambda= 14.70846887241110 error=3.77e-02, v-err=1.55e-01 ratio= 0.89430
k=  22 lambda= 14.71612420461681 error=3.01e-02, v-err=1.39e-01 ratio= 0.89450
k=  23 lambda= 14.72220150134280 error=2.40e-02, v-err=1.24e-01 ratio= 0.89469
k=  24 lambda= 14.72703365081295 error=1.92e-02, v-err=1.11e-01 ratio= 0.89485
k=  25 lambda= 14.73088102660057 error=1.53e-02, v-err=9.93e-02 ratio= 0.89500
```

The convergence rate approaches λ_2/λ_1 the ratio of the second largest to largest eigenvalue.

Listing 1: powerMethod.m

```
%
% Test the power method
%
clear;

m=10;
A=zeros(m,m);
v=zeros(m,1);

% ---- build the matrix and assign the initial guess ---
for i=1:m
  if( i>1 ) A(i-1,i)=-1.; end;
  A(i,i)=4+i;
  if( i<m ) A(i+1,i)=-1.; end;
  v(i)=1.;  % initial guess
end;

% --- Find the exact solution ----
[V,D] = eig(A);
lambdaTrue=diag(D); % these are sorted from smallest to largest

lambda1 = lambdaTrue(m); % largest
lambda2 = lambdaTrue(m-1); % next largest
fprintf('lambda1=%f, lambda2=%f, lambda2/lambda1 = %8.5f\n',lambda1,lambda2,lambda2/lambda1);
```

```

vTrue = V(:,m); % true eigenvector

vErrOld=norm(v-vTrue,2);

% ---- Power method ----
maxit=25;
for k=1:maxit
  v = A*v;
  v = v./norm(v,2);
  lambda = v'*A*v;

  vErr = norm(v-vTrue,2);
  ratio=vErr/vErrOld;
  vErrOld=vErr;
  fprintf('k=%4d␣lambda=%18.14f␣error=%8.2e,␣v-err=%8.2e␣ratio=%8.5f\n',k,lambda,abs(
      lambda-lambda1),vErr,ratio);

end;
```

(a) The code for the Rayleigh quotient method is given below. Here are the results:

```
>> rayleighQuotient
True: lambda= 10.00021752225710
k=   1 lambda= 10.30565571901764, error=3.05e-01, v-err=8.05e-01, ratio= 0.02971
k=   2 lambda= 10.14294003846173, error=1.43e-01, v-err=4.00e-01, ratio= 0.76525
k=   3 lambda= 10.00485453039090, error=4.64e-03, v-err=6.94e-02, ratio= 1.08508
k=   4 lambda= 10.00021762235017, error=1.00e-07, v-err=3.22e-04, ratio= 0.96383
k=   5 lambda= 10.00021752225710, error=1.78e-15, v-err=3.21e-11, ratio= 0.95997
```

Listing 2: rayleighQuotient.m

```
%
% Test the Rayleigh-Quotient method
%
clear;

m=10;
A=zeros(m,m);
As=zeros(m,m); % shifted matrix
v=zeros(m,1);

% ---- build the matrix and assign the initial guess ---
for i=1:m
  if( i>1 ) A(i-1,i)=-1.; end;
  A(i,i)=4+i;
  if( i<m ) A(i+1,i)=-1.; end;
  v(i)=1.;  % initial guess
end;

% --- Find the exact solution ----
[V,D] = eig(A);
lambdaTrue=diag(D); % these are sorted from smallest to largest

```

```
% Look for the eignvalue near lambda=10

lambda1 = lambdaTrue(6); % true
fprintf('True:␣lambda=%18.14f\n',lambda1);

vTrue = V(:,6); % true eigenvector


lambda=10.5; % initial guess
vErrOld=norm(v-vTrue,2);

% ---- Rayeligh-Quotient Iteration ----
maxit=5;
for k=1:maxit

  % -- form A-lambda*I
  As = A - lambda*eye(m);

  v = As\v;           % solve As*v = v
  v = v./norm(v,2);
  lambda = v'*A*v;

  vErr = min(norm(v-vTrue,2),norm(v+vTrue,2)); % sign may change
  ratio=vErr/(vErrOld^3);
  vErrOld=vErr;
  fprintf('k=%4d␣lambda=%18.14f,␣error=%8.2e,␣v-err=%8.2e,␣ratio=%8.5f\n',k,lambda,abs(
      lambda-lambda1),vErr,ratio);

end;
```

3. (20 pts) QR program.

The code for the unshifted and shifted algorithm is given below. Here are the results for the unshifted QR algorithm:

```
>> qrEigs

A =

     2    -1     0     0     0     0     0     0     0     0     0
    -1     2    -1     0     0     0     0     0     0     0     0
     0    -1     2    -1     0     0     0     0     0     0     0
     0     0    -1     2    -1     0     0     0     0     0     0
     0     0     0    -1     2    -1     0     0     0     0     0
     0     0     0     0    -1     2    -1     0     0     0     0
     0     0     0     0     0    -1     2    -1     0     0     0
     0     0     0     0     0     0    -1     2    -1     0     0
     0     0     0     0     0     0     0    -1     2    -1     0
     0     0     0     0     0     0     0     0    -1     2    -1
     0     0     0     0     0     0     0     0     0    -1     2

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
QR: DONE: max-off-diagonal: delta=9.74e-06, tol=1.000000e-05
QR: lambda=[3.93185165,3.73205081,3.41421356,3.00000000,2.51763809,2.00000000,1.48236191,
            1.00000000,0.58578644,0.26794919,0.06814835]
QR: Max error=4.75e-10
```

The asymptotic convergence ratio is $r(k) \approx .949$. The unshifted QR algorithm is seen to be converging linearly. The convergence rate is approximately equal to

$$\frac{\lambda_2}{\lambda_1} \approx \frac{3.73205081}{3.93185165} \approx .949$$

as would be the case for the power method.

4. (20 pts) Shifted QR program.

The code for the unshifted and shifted algorithm is given below. Here are the results for the shifted QR algorithm:

```
>> qrEigs

A =

     2    -1     0     0     0     0     0     0     0     0     0
    -1     2    -1     0     0     0     0     0     0     0     0
     0    -1     2    -1     0     0     0     0     0     0     0
     0     0    -1     2    -1     0     0     0     0     0     0
     0     0     0    -1     2    -1     0     0     0     0     0
     0     0     0     0    -1     2    -1     0     0     0     0
     0     0     0     0     0    -1     2    -1     0     0     0
     0     0     0     0     0     0    -1     2    -1     0     0
     0     0     0     0     0     0     0    -1     2    -1     0
     0     0     0     0     0     0     0     0    -1     2    -1
     0     0     0     0     0     0     0     0     0    -1     2

QR: k=1 : mu=1.000e+00 (Matrix size mm=11) delta=1.22e+00, ratio=1.225
QR: k=2 : mu=3.000e+00 (Matrix size mm=10) delta=1.41e+00, ratio=1.155
QR: k=3 : mu=1.000e+00 (Matrix size mm=9) delta=1.06e+00, ratio=0.750
QR: k=4 : mu=7.435e-01 (Matrix size mm=9) delta=1.23e+00, ratio=1.161
QR: k=5 : mu=5.798e-01 (Matrix size mm=9) delta=1.13e+00, ratio=0.916
QR: k=6 : mu=5.858e-01 (Matrix size mm=9) delta=1.43e+00, ratio=1.266
QR: k=7 : mu=2.671e-01 (Matrix size mm=8) delta=6.26e-01, ratio=0.439
QR: k=8 : mu=2.679e-01 (Matrix size mm=8) delta=6.38e-01, ratio=1.018
QR: k=9 : mu=7.032e-02 (Matrix size mm=7) delta=5.63e-01, ratio=0.883
QR: k=10 : mu=6.815e-02 (Matrix size mm=7) delta=5.00e-01, ratio=0.887
QR: k=11 : mu=1.483e+00 (Matrix size mm=6) delta=3.09e-01, ratio=0.618
QR: k=12 : mu=1.482e+00 (Matrix size mm=6) delta=1.69e-01, ratio=0.546
```

```
QR: k=13 : mu=2.000e+00 (Matrix size mm=5) delta=1.46e-01, ratio=0.866
QR: k=14 : mu=2.000e+00 (Matrix size mm=5) delta=1.25e-01, ratio=0.853
QR: k=15 : mu=2.518e+00 (Matrix size mm=4) delta=9.54e-02, ratio=0.765
QR: k=16 : mu=3.415e+00 (Matrix size mm=3) delta=5.70e-02, ratio=0.598
QR: k=17 : mu=3.414e+00 (Matrix size mm=3) delta=3.71e-02, ratio=0.650
QR: k=18 : mu=3.732e+00 (Matrix size mm=2) delta=3.67e-17, ratio=0.000
QR: DONE: max-off-diagonal: delta=3.67e-17, tol=1.000000e-05
QR: lambda=[3.93185165,3.73205081,3.41421356,2.51763809,2.00000000,1.48236191,0.06814835,0.26794919,0.58578644,
           3.00000000,1.00000000]
QR: Max error=2.22e-14
```

The shifted QR algorithm is converging much faster than the unshifted. The convergence rate does not appear to be linear. We are converging to a new eigenvalue every 1-4 iterations, which suggested rapid convergence for each eigenvalue.

Listing 3: qrEigs.m

```
%
% Compute eigenvalues by the QR algorithm
% using the unshifted or shifted algorithm
%

shift=1; % set shift=0 for no shift

m=11;
A=zeros(m,m);

for i=1:m
    if( i>1 ) A(i,i-1)=-1; end;
    A(i,i)=2;
    if( i<m ) A(i,i+1)=-1; end;
end;
A
Lambda=eig(A);
% pause

tol=1.e-5;
nit=500;
I=eye(m,m);

lambda=zeros(m,1);
mm=m; % current size of the deflated matrix
deltaOld=max(max(abs(A-diag(diag(A)))));
for k=1:nit

  if shift==1
    % mu=A(mm,mm);   % Rayleigh quotient shift
    % Wilkinson shift:
    a=A(mm-1,mm-1); b=A(mm-1,mm); c=A(mm,mm); delta=.5*(a-c);
    signDelta =sign(delta); if( signDelta==0 ) signDelta=1; end;
    mu = c - signDelta*b^2/( abs(delta)+ sqrt(delta^2 + b^2) );
  else
    mu=0.; % no shift
  end

  [Q,R]=qr(A-mu*I);
```

```
  A=R*Q+mu*I;

  delta=max(max(abs(A-diag(diag(A))))); % maximum of absolute values of off diagonals
  if( shift==0 && mod(k,10)==0 )
    fprintf('QR: k=%d : delta=%8.2e, ratio=%5.3f\n',k,delta,delta/deltaOld);
  elseif( shift==1 )
    fprintf('QR: k=%d : mu=%9.3e (Matrix size mm=%d) delta=%8.2e, ratio=%5.3f\n',k,mu,mm,
        delta,delta/deltaOld);
  end
  deltaOld=delta;

  if( shift==1 && mm>1 && abs(A(mm-1,mm))<tol )
    % When eigenvalue at lower right corner has converged deflate matrix
    lambda(mm)=A(mm,mm);
    mm=mm-1;
    A=A(1:mm,1:mm);
    I=eye(mm,mm);
  end;


  if( delta<tol ) break; end;

end
if( shift==1 )
  lambda(1)=A(1,1);
else
  lambda=diag(A);
end;

fprintf('QR: DONE: max-off-diagonal: delta=%8.2e, tol=%e\n',delta,tol);


fprintf('QR: lambda=[%10.8f',lambda(1));
for( i=2:m ) fprintf(',%10.8f',lambda(i)); end;
fprintf(']\n');

maxErr=max(abs(sort(lambda)-sort(Lambda)));
fprintf('QR: Max error=%8.2e\n',maxErr);
```