CS513, Fall 2000

Prof. Amos Ron

Homework #5 Solutions

Prepared by Houssain Kettani

1. (a) To find the number of eigenvalues of A in the interval (0,1) using Method I, we LU-factorize A and $A' = A - I$, so that $A = LU$ and $A' = L'U'$. We find:

$$L = \begin{pmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ 0 & \frac{1}{2} & 1 \end{pmatrix},$$

$$U = \begin{pmatrix} -1 & 1 & 0 \\ 0 & 2 & 1 \\ 0 & 0 & \frac{3}{2} \end{pmatrix},$$

$$L' = \begin{pmatrix} 1 & 0 & 0 \\ -\frac{1}{2} & 1 & 0 \\ 0 & 2 & 1 \end{pmatrix},$$

and

$$U' = \begin{pmatrix} -2 & 1 & 0 \\ 0 & \frac{1}{2} & 1 \\ 0 & 0 & -1 \end{pmatrix}.$$

Thus, by observing U and U', A has two positive eigenvalues and A' has two negative eigenvalues. Therefore, there is only one eigenvalue of A in the given interval.
Using Method II, the sequence of main principal minors of A is {1, -1, -2, -3}, whereas that of A' is {1, -2, -1, 1}. There is one sign change in the first sequence, while the second has two sign changes. Therefore, A has one eigenvalue in the given interval.

(b) We see here that the number of arithmetic operations (flops) that each method performs is close to each other. Mainly, 75 for the first method and 86 for the second one.

(c) The following two MATLAB codes were developed to answer this question.

```
function [e,f] = method1(A,s1,s2)
% This function returns the number of evals "e" a matrix A has in the
% given interval [s1,s2], and returns the number of flops "f" used.
% the method used is LU-factorizing a symmetric matrix A, then
% inspect the diagonal of U.
flops(0)
n = length(A);
I = eye(n);
A1 = A - s1*I;
[L1,U1] = sym_lu(A1);   % LU-factorizing A1 using the symmetric
                        % algorithm given in HW5 solutions
```

```
sig1 = sign(diag(U1));   % checking the signature of A1
peval1 = (sum(sig1)+n)/2; % # of positive evals of A1

A2 = A - s2*I;
[L2,U2] = sym_lu(A2);   % LU-factorizing A2 using the symmetric
                        % algorithm given shown in HW5 solutions
sig2 = sign(diag(U2));   % checking the signature of A2
peval2 = (sum(sig2)+n)/2; % # of positive evals of A2

e = peval1 - peval2;
f = flops;


function [e,f] = method2(A,s1,s2)
% This function returns the number of evals "e" of a tridiagonal
% matrix A has in the
% given interval [s1,s2], and returns the number of flops "f" used.
% the main principal minor method is used.
flops(0)
n = length(A);
I = eye(n);
A1 = A - s1*I;
b= zeros(1,n+1); % this vector is to contain the determinants of the main
       % principal minors of A1
b(1) = 1;
b(2) = A1(1,1);

for i = 2:n
  % since A is tridiagonal:
  b(i+1) = A1(i,i)*b(i) - A1(i,i-1)*A1(i-1,i)*b(i-1);
end

sig = sign(b);
neval1 = 0;  % # of negative evals of A1

for j = 1:n
  if sig(j)+sig(j+1) == 0
    neval1 = neval1+1;
  end
end

A2 = A - s2*I;
b= zeros(1,n+1); % this vector is to cantain the determinants of the main
       % principal minors of A2
b(1) = 1;
```

```
b(2) = A2(1,1);

for i = 2:n
  b(i+1) = A2(i,i)*b(i) - A2(i,i-1)*A2(i-1,i)*b(i-1);
end

sig = sign(b);
neval2 = 0;  % # of negative evals of A2

for j = 1:n
  if sig(j)+sig(j+1) == 0
    neval2 = neval2+1;
  end
end

e = neval2 - neval1;
f = flops;
```

To generate a `20x20` symmetric tridiagonal matrix A, we write:

```
A = rand(20);
A = A + A';    % make A symmetric
A = hess(A)    % substitute A by the corresponding Hessenberg matrix
```

Applying both methods to this matrix A, the output of the first and second codes are `[e,f] = [3,7385]` and `[e,f] = [3,2008]`, respectively. Clearly for large matrices the second method is much more efficient.

2. (a) Since the n Gerschgoun's discs are pairwise disjoint, each disc has to contain at least one eigenvalue, and A is nxn, the cardinality of $\sigma(A)$ is n. Since the eigenvalues are disjoint, each has an algebraic multiplicity equals one. We know that $m_g(\lambda_i) \leq m_a(\lambda_i)$, but $m_a(\lambda_i) = 1$, therefore $m_g(\lambda_i) = 1$.

   (b) Leon is right, since $m_g(\lambda_i) = m_a(\lambda_i)$.

   (c) Note that A is a real matrix, thus Gerschgoun's discs are centered on the real axis, and if λ is complex with $\lambda \in \sigma(A)$, then $\lambda^* \in \sigma(A)$. Thus, if A had a complex eigenvalue, one of the discs will contain two eigenvalues, which is a contradiction. Therefore, Nina is right too.

   (d) Elvis is wrong. For the power method to converge, we need $|\lambda_1| > |\lambda_i|$, for $i > 1$. But, we can have, for example, $\lambda_2 = -\lambda_1 = 1$. Note that the power method does not converge in this case, although the Gerschgoun's discs are disjoint.

3. The algorithms are already given in the notes and the book, therefore are not presented here for brevity. The actual eigenvalues using MATLAB `eig` function are given in the following vector:

$$\begin{pmatrix} 6.1165 \\ -1.2866 \\ 0.0367 + 1.0294i \\ 0.0367 - 1.0294i \\ 0.7571 + 0.4518i \\ 0.7571 - 0.4518i \\ 0.3545 + 0.4643i \\ 0.3545 - 0.4643i \\ -0.3441 \\ 0.0265 + 0.3537i \\ 0.0265 - 0.3537i \\ -0.1153 \end{pmatrix},$$

with corresponding magnitude:

$$\begin{pmatrix} 6.1165 \\ 1.2866 \\ 1.0301 \\ 1.0301 \\ 0.8816 \\ 0.8816 \\ 0.5842 \\ 0.5842 \\ 0.3441 \\ 0.3547 \\ 0.3547 \\ 0.1153 \end{pmatrix}.$$

We apply the power method to get the first two largest eigenvalues in magnitude, with the corresponding eigenvectors. the first eigenpair is very close to to that obtained with MATLAB `eig` function, considering that the stopping criteria was that the error is less than 10^{-2}. The second eigenpair (λ_1, v_2) also was okay, except the first entry of the eigenvector (0.0139 vs. 0.0291). Recall that the matrix A was deflated, obtaining the matrix A_1. We then applied the power method to obtain w_2. The eigenvalue of A_1 corresponding to the largest eigenvalue in magnitude. this eigenvector is v_2 withot the first component, which was computed as described in the notes. Multiplying the first row of A_1 with w_2 one can see that the errors in each element of w_2 are essentially added up in the computation. So when one adds up 11 elements with error on the order of 0.001, one should expect an error on the order of 0.01 in the result. The accumulation of thsi error in the algorithm makes things even worse.

This is a big problem, and setting the tolerance to a smaller level does not seem to help. This eigenvalue is close in magnitude to some of the other complex pairs of eigenvalues, and the deflation method does not seem to handle the problem well. One way to try to avoid this problem would be to get a rough estimate x of this second-largest eigenvalue, by applying Gershgoun's theorem, and then use the inverse power method on A - Ix to find the answer.

This should work well, because those complex evalues whose magnitudes are close to that of our desired eigenvalue will be "far away" from the real value x in the complex plane. Hence, when the matrix is shifted, they will still have large magnitudes, but the desired eigenvalue will have a small magnitude and the inverse power method will find it. The convergence should be better, and the results will then be better too, since fast convergence implies fewer operations, which implies less round-off error.

We next apply the power method to get the first two smallest eigenvalues in magnitude, with the corresponding eigenvectors. the fisrt eigenpair is very close to to that obtained with MATLAB `eig` function, considering that the stopping criteria was that the error is less than 10^{-2}. The second eigenpair is another story however. The resaltant eigenvector has nothing to do with the actual eigenvector, and the algorithm took a long time to stop.

It is not surprising that this eigenvector is a garbage result. When the machine takes an extremely long time to compute something when it normally doesn't require that much time, that may mean that the algorithm is running into problems computing the output for that particular input. This computation took 188 iterations as compared to ¡10 for most inputs. Round-off error builds up, and it may even be minor compared to some other error that is a source or symptom of the difficult computation.The first eigenpair took 2078 flops to compute, the second one took 20533, the third took 6431, but the forth took 98525.

Again, a great way to circumvent this problem is to use shifting. What have happened here is that there is another eigenvalue with the same magnitude as the eigenvalue we are interested in. If the eigenvalue of interest can be estimated, the inverse power method on the shifted matrix produces the eigenpair with good results. In this case, a shift of $2 * \lambda_3$ produced good results.

4. (a) Recall that in the QR factorization of the mxm matrix A, we write

$$H_{m-1}H_{m-2}\dots H_1A = R,$$

where

$$Q = H_{m-1}H_{m-2}\dots H_1.$$

The construction of the Householder matrix H_j is as follows: write $A = [a_1|\dots|a_m]$, pick $x = a_j$. Since A is tridiagonal, $x = [0\dots 0x_{j-1}x_jx_{j+1}0\dots 0]^T$. Then $y = [0\dots 0x_{j-1}st0\dots 0]^T$, where st (stands for something!) is such that $\|x\| = \|y\|$. Thus, $w \approx x - y = [0\dots 0stx_{j+1}]^T$. Therefore,

$$H_j = I - 2ww^* = \begin{pmatrix} I & 0 & 0 \\ 0 & H^{(j)} & 0 \\ 0 & 0 & I \end{pmatrix},$$

where,

$$H^(j) = \begin{pmatrix} h_{j,j} & h_{j,j+1} \\ h_{j+1,j} & h_{j+1,j+1} \end{pmatrix}.$$

This shows that Q is a Hessenberg matrix, i.e. Q is the matrix with entries $q_{k,l}$, such that $q_{k,l} = 0$ for $l > k + 1$. Recall also that R is an upper triangular matrix with entries $r_{ij} = (a_i, q_j)$. Therefore, R has non-zero entries on its diagonal, and on the first and second super-diagonals.

(b) Note that this problem refers to Theorem 28.4, which assumes that A is symmetric. Let $A = QR$ as mentioned before. Let $B = RQ$. Note that B is a Hessenberg matrix, thus to show that B is tridiagonal, it suffices to show that B is symmetric. But

$$B = Q^*QRQ = Q^*AQ,$$

thus

$$B^* = Q^*A^*Q = Q^*AQ = B.$$

Therefore, B is tridiagonal.