Lab Session 8

MA-423 : Matrix Computations July-November 2013 S. Bora

Last Lab Assignment! Cheers!

1. Write a function program $[\texttt{iterate}, \texttt{sigma}] = \texttt{powermethod}(\texttt{A}, \texttt{q}, \texttt{k})$ for performing k iterations of Power method for finding the dominant eigenvalue of a matrix A with starting vector q which returns a matrix `iterate` of dimension $n \times k$ such that $\texttt{iterate}(:, \texttt{j})$ is the vector obtained at the end of the j^{th} iteration of the Power method and sigma is the scaling factor at the end of k iterations. If `sigma` is chosen as explained in your theory class it will approximate the dominant eigenvalue of A whenever such an eigenvalue exists.

   Run the above program with $q = [1; 1; 1]$

$$A = \begin{bmatrix} 2.4347 & -0.8641 & 1.4661 \\ -0.8641 & 2.6630 & -3.1263 \\ 1.4661 & -3.1263 & 6.9023 \end{bmatrix}$$

   and your own choice of k. In how many iterations do you see convergence?

   The vector $v = [0.2507; -0.4556; 0.8542]$ is a unit vector corresponding to the dominant eigenvalue. Thus if $y = x - \langle x, v \rangle v$, for any arbitrary choice of x, then y has no component in the direction of v in exact arithmetic. Repeat the above experiment with q equal to such a y and keep increasing the value of k. What do you observe now?

2. Write a function program $[\texttt{iterate}, \texttt{sigma}] = \texttt{shiftinv}(\texttt{A}, \texttt{q}, \texttt{k}, \texttt{s})$ which performs k steps of Shift and invert Method with shift `s` on a matrix A with starting vector q to produce an $n \times k$ matrix `iterate` whose j^{th} column is the vector obtained at the end of the the j^{th} iteration and a scalar `sigma` which approximates an eigenvalue of $\texttt{A} - \texttt{s} * \texttt{eye}(\texttt{n})$.

   You should remember to convert A to upper Hessenberg form by using $[\texttt{Q}, \texttt{H}] = \texttt{hess}(\texttt{A})$ and perform the iterations with the upper Hessenberg matrix H. At the end you have to recover the eigenvector of A from the one produced by the algorithm for H.

   Run the above program for the same matrix A as given in the previous exercise with your own choices of shifts, starting vectors and number of iterations k (at least 5 different combinations), given the information that the exact eigenvalues of A are $9, 2$, and 1.

3. Write a function program $[\texttt{iterate}, \texttt{rho}] = \texttt{Rayleigh}(\texttt{A}, \texttt{q}, \texttt{k})$ which performs k steps of Rayleigh quotient iteration on A with initial vector q with output `iterate` as in the previous exercises and a scalar `rho` which is the Rayleigh quotient obtained at the end of the k iterations.

   Run the above program with $q = [1; 0; 0]$ and

$$\begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

   and your own choice of k. What do you observe. Repeat the experiment with at least 5 other choices of q and note your observations. P.T.O.

Run the above program with $q = [1; 0; 1]$ and

$$A = \begin{bmatrix} 1 & 1 & 1 \\ -1 & 9 & 2 \\ 0 & -1 & 2 \end{bmatrix}$$

and your own choice of k. What kind of convergence do you see now?

4. Write a program $[\mathtt{B}, \mathtt{p}] = \mathtt{deflateQR}(\mathtt{A}, \mathtt{tol}, \mathtt{N})$ that executes iterations of unshifted QR algorithm on an $n \times n$ properly upper Hessenberg matrix A, until it produces a matrix B for which $B(n, n-1)$ and/or $B(n-1, n-2)$ is close enough to zero and either one eigenvalue or a pair of eigenvalues of B (and hence of A) are revealed. The input `tol` should be a small number used to decide when the appropriate entries of the matrix produced at any stage are small enough to stop the iterations. Also `N` should be a positive integer that indicates the maximum number of iterations allowed. The output p should be 0 if the matrix B cannot be produced with the given values of `tol` and `N` and 1 otherwise. In case of failure the program should return B as the matrix obtained after the N^{th} iteration. You may use the built in MATLAB command $[\mathtt{Q}, \mathtt{R}] = \mathtt{qr}(\mathtt{A})$ in the program.

   Write two more programs $[\mathtt{B}, \mathtt{p}] = \mathtt{deflateQRR}(\mathtt{A}, \mathtt{tol}, \mathtt{N})$ and $[\mathtt{B}, \mathtt{p}] = \mathtt{deflateQRW}(\mathtt{A}, \mathtt{tol}, \mathtt{N})$ with exactly the same features as above that execute QR iteration with Rayleigh Quotient shifting strategy and Wilkinson shifting strategy respectively.

   Test the above programs on different moderately sized matrices and observe the differences when you choose real matrices with all eigenvalues that are real vis a vis those which have complex eigenvalues.