Lab Session 13

MA581: Numerical Computations Lab July-November, 2022 S. Bora

---

1. Write a function program $[\texttt{iter}, \texttt{lambda}] = \texttt{Powermethod}(\texttt{A}, \texttt{x}, \texttt{k})$ that performs $\texttt{k}$ iterations of the Power Method with $\texttt{A} \in \mathbb{C}^{\texttt{n}\times\texttt{n}}$ and initial vector $\texttt{x} \in \mathbb{C}^{\texttt{n}}$ and returns an $n \times k$ matrix $\texttt{iter}$ whose jth column is the jth iterate q_j and a scalar $\texttt{lambda}$ which is the dominant eigenvalue.

2. Run $[\texttt{iter}, \texttt{lambda}] = \texttt{Powermethod}(\texttt{A}, \texttt{x}, \texttt{k})$ with $x = [1\ 1\ 1]^T$ and the following matrices

$$(i)\, A = \begin{bmatrix} 1 & 1 & 1 \\ -1 & 9 & 2 \\ 0 & -1 & 2 \end{bmatrix} \qquad (ii)\, A = \begin{bmatrix} 1 & 1 & 1 \\ -1 & 9 & 2 \\ -4 & -1 & 2 \end{bmatrix} \qquad (iii)\, A = \begin{bmatrix} 1 & 1 & 1 \\ -1 & 3 & 2 \\ -4 & -1 & 2 \end{bmatrix}$$

   In each case check if for large enough values of j, $\|\texttt{iter}(:, j+1) - v\|/\|\texttt{iter}(:, j) - v\|$ agrees with the theoretical convergence rate of $|\lambda_2|/|\lambda_1|$ where λ_1 and λ_2 are the largest and second largest eigenvalues of A in magnitude and v is an eigenvector corresponding to λ_1. Try to explain your observations. (Type `format long e` to see more digits. You will have to run $[\texttt{V}, \texttt{D}] = \texttt{eig}(\texttt{A})$ and use the same scaling that you perform in your iterations on the column of `V` corresponding to λ_1 to find λ_1, λ_2 and v.)

3. Write a function program $[\texttt{iter}, \texttt{lambda}] = \texttt{Shiftinv}(\texttt{A}, \texttt{x}, \texttt{s}, \texttt{k})$ that *efficiently* performs $\texttt{k}$ iterations of Shift and Invert Method using $\texttt{A} \in \mathbb{C}^{\texttt{n}\times\texttt{n}}$, $\texttt{x} \in \mathbb{C}^{\texttt{n}}$ and shift `s` and returns an $n \times k$ matrix `iter` whose jth column is the jth iterate q_j and a scalar `lambda` is the the eigenvalue of A closest to `s`.

4. Write a function program $[\texttt{iter}, \texttt{lambda}] = \texttt{Rayleigh}(\texttt{A}, \texttt{x}, \texttt{k})$ that *efficiently* performs `k` iterations of inverse iterations with Rayleigh quotient shifts using $\texttt{A} \in \mathbb{C}^{\texttt{n}\times\texttt{n}}$ and $\texttt{x} \in \mathbb{C}^{\texttt{n}}$ and returns an $n \times k$ matrix `iter` whose jth column is the jth iterate q_j and a scalar `lambda` is the eigenvalue of A to which the Rayleigh quotient shifts converge.

   Note. *Use* $[\texttt{Q}, \texttt{H}] = \texttt{hess}(\texttt{A})$ *to find an upper Hessenberg matrix H and a unitary matrix Q such that $Q^*AQ = H$ and use H in place of A in the iterations. This will reduce the flop count in each iteration from $O(n^3)$ to $O(n^2)$. However, if the program will converges, it will be to an eigenvector v corresponding to H from which an eigenvector corresponding to A will have to be found by computing Qv.*