Software Assignment-REPORT

EE24BTECH11045 - N.Tapasvi

AIM:
The aim of this report is to explore various methods for computing and analyzing eigenvalues of matrices.

THEORY:
Eigenvalues and Eigenvector: A set of scalars that describe how a matrix transforms a vector. They are also called characteristic values or characteristic roots. An eigenvalue of a matrix A is the value of λ which satisfies the equation

$$\mathrm{A}v = \lambda v$$

for some non-zero vector v. Here A is a $n \times n$ matrix,λ is eigenvalue and v is eigenvector. The eigenvectors do not change direction but may be scaled after matrix transformation.

For a given matrix A, eigenvectors are scaled by matrix A in the direction of the eigenvector. The amount by which the eigenvector is scaled depends on λ.

- $\lambda > 1$, the eigenvector is stretched.
- $\lambda = 1$, the length of eigenvector not changed.
- $0 < \lambda < 1$, the eigenvector is shrunk.
- $\lambda = 0$, the matrix transforms the eigenvector into a zero vector.
- $\lambda = -1$, the eigenvector is reflected.

QR ALGORITHM:
The QR Algorithm is particularly useful for dense,square matrices. In this method the matrix A is factorized into Q and R matrices where Q is an orthogonal matrix and R is upper triangular matrix.
Step 1: $A_0 = A$
Step 2: $A_k = Q_k R_k$
Step 3: $A_{k+1} = R_k Q_k$ where A_{k+1} is an upper triangular matrix whose diagonal elements are the eigen values of A.

Example 1:
Consider the matrix,

$$A = \begin{bmatrix} 6 & 2 \\ 2 & 5 \end{bmatrix}$$

The QR decomposition gives us:

$$Q_0 = \begin{bmatrix} \frac{3}{\sqrt{10}} & \frac{-13}{\sqrt{10}} \\ \frac{1}{\sqrt{1690}} & \frac{39}{\sqrt{1690}} \end{bmatrix} \quad \text{and} \quad R_0 = \begin{bmatrix} \frac{20}{\sqrt{10}} & \frac{11}{\sqrt{10}} \end{bmatrix}$$

Updated matrix A_1

$$A_1 = RQ = \begin{bmatrix} 7 & 0 \\ 0 & 4 \end{bmatrix}$$

Thus, the eigenvalues are 7 and 4.
Python code:Software Assignment/codes/e1.py.

Example 2:
Consider the matrix A,

$$A = \begin{bmatrix} 4 & 1 \\ 2 & 3 \end{bmatrix}$$

The QR decomposition gives us:

$$Q_0 = \begin{bmatrix} \frac{2}{\sqrt{5}} & \frac{1}{\sqrt{5}} \\ \frac{1}{\sqrt{5}} & \frac{3}{\sqrt{5}} \end{bmatrix} \quad \text{and} \quad R_0 = \begin{bmatrix} \sqrt{5} & \frac{11}{\sqrt{5}} \\ 0 & \sqrt{5} \end{bmatrix}$$

The updated matrix A_1 after the first QR iteration is:

$$A_1 = \begin{bmatrix} 4.618 & 1.382 \\ 1.382 & 3.618 \end{bmatrix}$$

After further iterations, the matrix converges to:

$$A_k = \begin{bmatrix} 5 & 0 \\ 0 & 2 \end{bmatrix}$$

Thus, the eigenvalues of the matrix A are 5 and 2.
Python code:Software Assignment/codes/e2.py.

Example 3:
Cosider the 3×3 matrix

$$A = \begin{bmatrix} 4 & -1 & 1 \\ -1 & 4 & -2 \\ 1 & -2 & 3 \end{bmatrix}$$

$$Q_0 = \begin{bmatrix} \frac{4}{\sqrt{18}} & \frac{11}{\sqrt{1251}} & \frac{-2}{\sqrt{6}} \\ \frac{-1}{\sqrt{18}} & \frac{31}{\sqrt{1251}} & \frac{1}{\sqrt{6}} \\ \frac{1}{\sqrt{18}} & \frac{-13}{\sqrt{1251}} & \frac{1}{\sqrt{6}} \end{bmatrix} \text{ and } R_0 = \begin{bmatrix} \sqrt{18} & 0 & \frac{-8}{\sqrt{6}} \\ 0 & \sqrt{15} & \frac{4}{\sqrt{6}} \\ 0 & 0 & \frac{-1}{\sqrt{6}} \end{bmatrix}$$

After repetitively applying QR algorithm,

$$A_k = \begin{bmatrix} 6 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 2 \end{bmatrix}$$

Thus, the eigenvalues are 6, 3 and 2.
Python code:Software Assignment/codes/e3.py.

Few of the other methods to find eigenvalues are:

- Power Iteration Algorithm
- Jacobi Method
- Davidson Method

Algorithm	Time Complexity	Convergence Rate	Suitable for
Power Iteration	$O(n^2)$ per iteration	Slow(converges linearly)	Large Matrices, dominant eigenvalues
QR Algorithm	$O(n^3)$	Fast(superlinear or cubic)	General,dense,Symmetric matrices
Jacobi Method	$O(n^3)$	Moderate(linear and sublinear)	Symmetric Matrices
Davidson Method	$O(mn^2)$ per iteration	Fast(superlinear or cubic)	Large sparse Matrices, few eigenvalues

TABLE I: Some properties of the algorithms

Algorithm	Pros	Cons
Power Iteration	Simple to implement, fast for dominant eigenvalue	Slow for finding all eigenvalues, requires good starting vector
QR Algorithm	Finds all eigenvalues, converges relatively quickly	High computational cost per iteration for large matrices
Jacobi Method	Highly accurate for symmetric matrices, easy to implement	Slower for large matrices, less efficient for non-symmetric
Davidson's Method	Efficient for large sparse matrices, computes few eigenvalues	Requires preconditioning, sensitive to poor initial guess

TABLE II: Pros and Cons of Various Eigenvalue Algorithms