# EIGEN VALUE CALCULATION

B.Prajwal

ai24btech11005@iith.ac.in

## 1 Eigen values

An eigenvalue of a square matrix A is a scalar $\lambda$ such that there exists a nonzero vector v(called the eigenvector) satisfying the equation $A\overrightarrow{V} = \lambda\overrightarrow{V}$ Here $\lambda$ is called the eigen values for the matrix A. $\lambda$ represents how much the eigenvector is stretched or compressed during the transformation by A.

$\lambda$ represents how much the eigenvector is stretched or compressed during the transformation by A.

Eigenvalues characterize fundamental properties of the matrix, such as its scaling behavior and stability. It helps us to solve system of linear differential equations, describe natural frequencies of vibrations, seperate modes of motion, distinguish states of energy and in many important applications etc.
There are many algorithms which can be used to find the eigen values , but each of them will have it's own pros and cons . Here are some algorithms which can be used to find the eigen values.

### 1.1 QR-ALGORITHM

The QR algorithm computes a Schur decomposition of a matrix. It is certainly one of the important algorithm in eigen value computations.. The total complexity of the algorithm is essentially $O(n^3)$ which can only be achieved in practice after several improvements are appropriately taken into account

## 1.2 Basic Form of QR Algorithm

Consider the QR factorization of the matrix A

$$A = QR$$

where $Q{*}Q = I$ and R is upper triangular. We will now reverse the order of multiplication product of Q and R and eliminate R

$$RQ = Q^{*}AQ$$

With using the similarity condition RQ has the same eigen values as A.Hence by repeating this process several times the RQ matrix will become closer and closer to upper triangular matrix , then we can get the eigen values directly from the diagonal elements.

Initiate with$A_0 = A$

$A_k = R_k Q_k$

where $Q_k$and$R_k$ represents a QRfactorization of QR factorization of $A_{k1}$

$A_{k-1} = Q_k R_k$

### 1.2.1 Process followed

Finally we get an Matrices U and T such that $A = UTU^{*}$

Set $A_0 = A$ and $U_0 = I$ for $K = 1....$

Compute QR factorization: $A_{k-1} = Q_k R_k$

Set $A_k = R_k Q_k$

Set $U_k = U_{k-1} Q_k$

Finally we will get $T = A_\infty$ and $U = U_\infty$

The elements of the matrix $A_k$ below the diagonal will converge to zero according to $|a_{ij}^{k} = O(|\lambda_i/\lambda_j|^k)$ for all $i > j$

### 1.2.2 Disadvantages with Basic QR method

1: The steps of the basic QR method is relatively expensive.

The complexity of one step of the basic QR method=$O(n^3)$

In an overly optimistic situation steps would be proportional to n, the algorithm would need $O(n^4)$ operations.

2:Many steps are required for the convergence ”more than n”. And the other rigourous problem is **if the eigen values are close to each other** it is often slow.

## 2 Improving Basic QR method

### 2.1 Two-Phase Approach

#### 2.1.1 Phase 1

In the first phase we will compute a Hessenberg matrix H such that $A = UHU^*$, as H is not an upper triangular matrix , such a reduction can be done with a finite number of operations. **By carrying out n-2 orthogonality transformations with Householder reflections we will now be able to reduce the matrix A to a Hessenberg matrix.**
Hence we will get a matrix H(Hessenberg part of A) such that $A = UHU^*$ So A and H have same eigen values as they are known as similar.

#### 2.1.2 Phase 2

Now we will apply the basic QR algorithm to the matrix H. Now one of the major improvement is that the complexity of one step is $O(n^2)$.
Apply **Wilkinson shifts** to accelerate convergence.
Perfor the QR decomposition of $H - \sigma I$ and update H to $H = RQ$. Hence, after the convergence the eigen values are found on the diagonal of the Hessenberg matrix. Hence it is a powerful method because it reduces computational cost and improves convergence with the wilkinsons shift.It has better numerical stability so it is an efficient way yo compute eigen values of higher dimensions with better numerical stability.

## 3 Chosen Algorithm

Hessenberg reduction with Householder reflections QR of matrix in Hessenberg form with Jacobi rotations but we should consider shifts and deflation (for symmetric matrices Wilkinson’s shift , for non-symetric doubly implicit shift) We should also consider deflation

## 3.1 Process involved

1: Hessenberg Reduction

2: QR Iterations on Hessenberg Matrices

3: Jacobi Rotations for Symmetric Matrices

4: Shifts

5:Deflation

6:Convergence

### 3.1.1 Computation

1:**Hessenberg Reduction**

Using HouseHolder Reflections:

Update $H_i = H_{i-1} - 2w(w^T H_{i-1}$ it is applied to both rows to maintain symmetry.

Hence after n-2 steps , A is transformed to Hessenberg form H.

2:**QR Algorithm with shifts** In each iteration we should do:

1: Computing the QR decomposition of H.

$$H - \mu I = QR$$

2: Update the Hessenberg matrix:

$$H \leftarrow RQ + \mu I$$

3:**Convergence**

For symmetric matrices: H becomes diagonal

For non-symmetric H becomes upper triangular(Schur form)

4: **Wilkinson's shift(For symmetric Matrices)**

It accelerates convergence by choosing the most obvious value of $\mu$ for the shift based on the eigen values of the trailing 2X2 block: $H_k = \begin{bmatrix} h_{n-1,n-1} & h_{n-1,n} \\ h_{n,n-1} & h_{n,n} \end{bmatrix}$

Compute the eigenvalues:

$\lambda_{1,2} = \frac{h_{n-1,n-1}+h_{n,n}}{2} \pm \sqrt{(\frac{h_{n-1,n-1}+h_{n,n}}{2})^2 + (h_{n-1,n})^2}$.

Choose the $\mu$, The eigen valueclosest to $h_{n,n}$. because it ensures the rapid deflation of $h_{n,n-1}$ which is last subdiagonal entry.

5: **Implicit Double shift(For non-symmetric matrices)**

This shift is based on the characteristic polynomial of the trailing 2X2 block: $p(\mu) = |H_k - \mu I$. This shifts helps us to handle complex eigen values effectively.

Now the matrix is transformed as:

$H \leftarrow RQ + \mu I$

4:**Deflation**

It is applied to reduce the size of the problem when subdiagonal elements become negligible.

6:**Jacobi Rotations for Symmetric Matrices**

Jacobi rotations ensure the non-diagonal elements of H decrease systematically for symmetric matrices.

A given rotation matric=x $G(i, j, \theta)$ is defined as:

$G(i, j, \theta) = I - \sin\theta(e_i e_j^T + e_j e_i^T) + (1 - cos\theta)(e_i e_i^T + e_j e_j^T)$

where $\theta$ is chosen to zero out $h_{i,j}$.

Update: $H \leftarrow G^T HG$

## 3.2 Total Time Complexity

The total time complexity will depend on the matrix size and shifts used.

### 3.2.1 Symmetric matrices

$*$ Hessenberg reduction:$O(n^3)$

$*$ QR iterations with Wilkinson's shift:$O(n^2 logn)$

$*$ Total:$O(n^3)$

#### 3.2.2 Non-symmetric Matrices

$*$ Hessenberg reduction:$O(n^3)$

$*$ QR iterations with implicit doubleshift:$O(n^2 logn)$

$*$ Total:$O(n^3)$

## 4 Advantages of this method

#### 4.0.1 Efficiency

$*$ Improves the computational efficiency.

$*$ **Implicit double shifts** reduce the number of iterations required for convergence.

$*$ **Deflation** minimizes the effective problem size dynamically, improving performance for matrices with seperable eigenvalues.

#### 4.0.2 Accuracy

$*$ Implicit double shifts improve numerical stability by avoiding explicit shifts that can give us rounding errors.

$*$ Handles both symmetric and non-symmetric matrices with high precision.

#### 4.0.3 Suitability

$*$ Well-suited for dense symmetric and non-symmetric matrices.

$*$ Can compute all eigenvalues effectively for matrices of small to medium size.

$*$ It is not suitable for very large or sparse matrices due to its $O(n^3)$ preprocessing cost.

## 5 Comparision of various algorithms

### 5.1 QR Algorithm (Basic)

#### 5.1.1 Time Complexity

$O(n^3)$ processing, $O(n^2)$ per iteration.

#### 5.1.2 Accuracy

High for general matrices. May converge slowly for clustered eigenvalues.

#### 5.1.3 Best Used for

General dense matrices.

### 5.2 QR Algorithm (Hessenberg+shifts

#### 5.2.1 Complexity

$O(n^3)$ preprocessing, $O(n^2)$ per iteration.

#### 5.2.2 Accuracy

High, faster convergence than basic QR. Handles symmetric and non-symmetric matrices.

#### 5.2.3 Best Used for

General dense matrices, symmetric or non-symmetric.

### 5.3 QR Algorithm (Hessenberg + Implicit Double Shifts + Deflation)

#### 5.3.1 Complexity

$O(n^3) preprocessing$ , $O(n^2)$ per iteration with faster convergence.

#### 5.3.2 Accuracy

High for dense matrices. Implicit shifts accelerate convergence. Deflation reduces effective problem size.

#### 5.3.3 Best used for

Dense symmetric and non-symmetric matrices. Preferred for computing all eigenvalues.

## 5.4 Jacobi

### 5.4.1 Complexity

$O(n^3)$

### 5.4.2 Accuracy

Very high for symmetric matrices .

### 5.4.3 Best used for

Small-to-medium symmetric matrices.

## 5.5 Lanczos

### 5.5.1 Complexity

$O(nk), k << n$

### 5.5.2 Accuracy

High for a subset of eigenvalues. May need reorthogonalization for many eigenvalues.

### 5.5.3 Best Used for

Sparse symmetric matrices, few eigenvalues.

## 5.6 Arnoldi

### 5.6.1 Complexity

$O(n^2K), k << n$ .

### 5.6.2 Accuracy

High for a subset of eigenvalues. Stability depends on reorthogonalization.

### 5.6.3 Best Used for

Sparse non-symmetric matrices, few eigenvalues.

## 5.7 Divide-and-conquer

### 5.7.1 Complexity

$O(n^3)$

### 5.7.2 Accuracy

High accuracy. Parallelizable.

### 5.7.3 Best Used for

Dense symmetric matrices, large-scale computation.