Computation of eigenvalues

Course Code and Name

Date of Submission

Your Name

Contents

1 Eigenvalues and Eigenvectors

Eigenvalues and eigenvectors are defined for Square matrices only. In linear algebra matrices are used to make linear transformations for a given vector. Eigenvectors are the special vectors for a given matrix whose direction remains unchanged and the magnitude of the vector may change when it is acted by the matrix. Eigenvalues quantify the stretching or compression along the original direction of those vectors.This is how it is depict mathematically:

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

Eigenvalues are the roots of the characteristic equation

$$det|A - \lambda I| = 0$$

2 Computation of Eigenvalues

There are various methods of computing eigenvalues. Generally people use different algorithms for computing eigenvalues for different types of matrices. For instance, to find eigenvalues for a 2×2 matrix or a 3×3 matrix as we already know the characteristic equation we can directly solve the equation to get all the eigenvalues. But this method is not feasible for larger matrices as it is computationally expensive. Lets see some of the various algorithms to get eigenvalues :

2.1 For small to medium sized matrices

2.1.1 QR algorithm

This method iteratively decomposes the matrix into QR factors where Q is a orthogonal matrix and R is a upper triangular matrices and keep updating it. This method can be used for any type of small to medium sized matrices. This method has a good accuracy for small and medium sized matrices but computational expensive for large matrices ($O\left(n^3\right)$) To make this method better there are some changes such that this method works better for some types of matrices.

2.1.2 Jacobi Method

This algorithms rotates pairs of elements in **symmetric matrices** to zero out off-diagonal elements. This method is simple and can be used only for the symmetric matrices. this shows slow convergence for large matrices.

2.1.3 Divide-and-conquer

This method splits the matrix into smaller blocks and solves for eigenvalues and combines results of each block. This method is also used only for **symmetric matrices**. Implementation of this method is hard.

Using the above method we can find all the eigenvalues for the specified type of matrices.

2.2 For large, sparse matrices

2.2.1 Power method

This method finds the largest eigenvalues by repeatedly multiplying a random vector by the matrix. this method will give you only the largest eigenvalue so it is not so useful in many cases but can be used sometimes.

2.2.2 Inverse iteration

Finds eigenvalues near a given shift μ bu solving $(A - \mu I)x = y$ iteratively. This method is effective if and only if we have a good initial guess and requires solving linear systems.

Remember the above methods are useful to get some eigenvalues of a given matrices.

3 QR algorithm

In this algorithm we decompose the given matrix A into a product of a orthogonal matrix(Q) and also a upper triangular matrix(R).

$$A = A_0 \tag{1}$$

QR decomposition

perform QR decomposition of A_k:

$$A_k = Q_k R_k \tag{2}$$

Recompute A:

Form a new matrix

$$A_{k+1} = R_k Q_l k$$

This is equivalent to

$$A_k = Q_k A_{k+1} Q_k^T$$

This transformation does not change eigenvalues of the matrices which means all A_k where $k = 0, 1, 2, \ldots$ have same eigenvalues. The reason for unchanged eigenvalues is due to similarity in all the A_k's i.e, As $A_{k+1} = Q^T A_k Q$ and if $A\mathbf{v} = \lambda\mathbf{v}$ then multiply with Q^T on both sides then we get

$$Q^T A_k \mathbf{v} = Q^T \lambda \mathbf{v}$$

which implies

$$A_{k+1}(Q^T\mathbf{v}) = \lambda(Q^T\mathbf{v})$$

Hence as we can see eigenvalues are preserved for the matrices A_{k+1} and A_k. Here wee made Q to be orthogonal but a invertible matrix also does not change the eigenvalues. The specialty of orthogonal matrices is that they are always invertible and also when acted to a vector does not change it's magnitude. Invertible matrices which are not orthogonal do not guarantee that. The orthogonal matrices also ensure stability of computation.
There is a very noticeable thing that every time we find A_{k+1} for A_k the elements below the diagonal of the matrix A_{k+1} are smaller than that of elements in A_k

Iterate: Repeat the decomposition and recombination until A_k converges to a upper triangular matrix or a diagonal matrix.

3.0.1 disadvantages

1. The standard QR algorithm is computationally expensive as every iteration involves decomposition and again recombination. The standard QR requires $O(n^3)$ operations and hence ineffective with large matrices.

2. **Normal standard QR method with out any modifications sometimes converges very slowly mainly when eigenvalues are close in magnitude or if the matrix has complex eigenvalue.**

3. This process need separate computation to find eigenvectors causing extra computational burden.

4 Jacobi Method

In case of small symmetric matrices Jacobi method is better than standard QR algorithm as Jacobi method is more precise as it has a excellent numerical stability for symmetric matrices. Also it converges well as it works by reducing the off-diagonal elements directly. And also Jacobi method is easier to implement than Standard QR algorithm.
Jacobi method is a repetitive algorithm which is used to find both eigenvalue and eigenvectors. Here we make matrix undergo a sequence of plane rotations and make it diagonal matrix from which we can easily get eigenvalues. The rotations are made by a series of orthogonal transitions. Here each rotation targets largest off-diagonal element to eliminate while preserving the eigenvalues.

4.1 Method

Let $A_0 = A$ where A is the original matrix. And let $A = VDV^T$ First we need to find the largest eelement in the given symmetric matrix let us say A_{pq} where $p \neq q$. This element is to be eliminated in this iteration. The rotation matrix in general form is

$$R(p, q, \theta) = I_n$$

Where I_n is a identity matrix of oreder n. and

$$R_{pp} = \cos\theta$$

$$R_{qq} = \cos\theta$$

$$R_{pq} = -\sin\theta$$

$$R_{qp} = \sin\theta$$

And $\tan 2\theta = \frac{2A_{pq}}{A_{pp}-A_{qq}}$ now update A as $A_{k+1} = J^T A_k J$ and update eigenvector martix as $V = VJ$. Repeat the process until A is approximatelyy a diagonal matrix.

5 Hessenberg Reduction followed by QR algorithm with shifts:

Feature	**Hessenberg Reduction + QR with Shifts**	**Standard QR Algorithm**
Efficiency	More efficient due to Hessenberg form (fewer nonzero elements below the diagonal)	Less efficient due to handling full matrix
Convergence Speed	Faster convergence due to shifts and reduced matrix size	Slower convergence, especially for large matrices
Computational Cost	Reduced cost per iteration (Hessenberg form reduces matrix size)	Higher cost due to processing full matrix
Stability	More stable due to Hessenberg form and appropriate shifts	Less stable, especially for large or ill-conditioned matrices
Number of Iterations	Fewer iterations required for convergence	More iterations required for convergence

Table 1: Comparison of Hessenberg Reduction + QR with Shifts and Standard QR Algorithm

Because of the reasons provided above the Hessenberg reduction followed by QR algorithm with shifts is better than normal QR method.

5.1 Hessenberg Reduction

The process of converting the matrix into hessenberg form i.e, all the elements bellow the first sub diagonal are zeros. **The usage of hessenberg reduction makes complexity to $O(n^2)$ from $O(n^3)$.** Here we are given a matrix A

Householder Transformations:

We use a sequence of reflections to make the elements below the first sub diagonal zeros. Compute a householder vector that zeros out elements below the kth row. Update A using the transformation $A \leftarrow H_k A H_k^T$ here H_k is Householder matrix. The matrix A is now transformed into an upper Hessenberg matrix H.

5.2 QR algorithm with Shifts

The QR algorithm is an repetitive method for finding the eigenvalues of a matrix. Using shift accelerates convergence mainly for matrices with closely spaced eigenvalues. **Choose a shift μ_k:**

The shift μ_k is mainly choose as a approx to a eigenvalue of A_k. Common choose is bottom right element. **Shifted QR Decomposition:**

for the shifted matrix $A_k - \mu_k I$ compute the QR decomposition. And for the next matrix $A_{k+1} = R_k Q_k + \mu_k I$. iterate until it converges.

6 Time complexity, Accuracy and suitability of the Chosen code

Time complexity:

The Jacobi method has a time complexity of $O(n^3)$ per iteration. And in Hessenberg Reduction followed by QR Algorithm with Shifts it is $O(n^2)$.

Accuracy:

In Jacobi It has good numerical stability because it involves orthogonal transformations (rotations), which help maintain numerical precision. And in Hessenberg Reduction followed by QR Algorithm with Shifts handles closely spaced eigenvalues better than the standard QR algorithm or Jacobi method, leading to improved precision.

Suitability:

Jacobi is best suitable for small to medium sized matrices. It works well when high precision is required and the matrix size is not excessively large. Hessenberg reduction followed by QR with swifts is suitable mainly for large matrices.

And comparison with Standard QR is there in respective sections of Jacobi and Hessenberg reduction followed by QR with shifts.