Software Assignment

EE24BTECH11040 - Mandara Hosur

This report aims to detail a few popular algorithms used to find the eigenvalues of square matrices. Below is a list of the algorithms that have been outlined in the report:

1) The Characteristic Polynomial Method
2) The Power Iteration Algorithm
3) The Inverse Iteration Algorithm
4) The Jacobi Method
5) The QR Algorithm
6) The Divide And Conquer Algorithm

I. The Characteristic Polynomial Method

Algorithm Outline:

The characteristic polynomial $|A - \lambda I| = 0$ can be solved for λ. Here A denotes the given matrix, I denotes the identity matrix, and λ denotes the eigenvalue.

Some Advantages:

1) Solving a quadratic equation (for $2x2$ matrices) or a cubic equation (for $3x3$ matrices) is straightforward.
2) It works for any type of matrix.
3) Yields highly accurate results for smaller matrices.

Some Disadvantages:

1) Computing the determinant and solving the polynomial becomes impractical for larger matrices.
2) This method does not make sure of any explicit property that a matrix might have (such as sparsity, symmetry, etc.).
3) The calculated eigenvalues for larger matrices may have significant errors due to approximations and/or truncations that might have been made during computation.

II. The Power Iteration Algorithm

Algorithm Outline:

An arbitrary non-zero vector (of appropriate dimensions) is chosen and is recursively pre-multiplied by the given matrix until the multiplication process yields the same vector (say $\mathbf{V}$).

The largest element (in magnitude) in $\mathbf{V}$ represents the largest eigenvalue of the given matrix.

Some Advantages:

1) Easy to implement.
2) Highly efficient for sparse matrices (with many zero entries).
3) Can be implemented conveniently for large matrices as well.

Some Disadvantages:

1) May be slow to execute to completion when the second-largest eigenvalue is not far from the largest.
2) Does not calculate all eigenvalues; only the largest one.
3) The quality of convergence and precision of the eigenvalue calculated depend heavily on the choice of the initial arbitrary vector.

III. The Inverse Iteration Algorithm

Algorithm Outline:

This algorithm is an extension of the power iteration algorithm and can be used to find all eigenvalues of a given matrix.

Let A be the given matrix, λ be an eigenvalue, and $\mathbf{v}$ be an eigenvector. Then, by definition, we have

$$
\begin{aligned}
A\mathbf{v} &= \lambda\mathbf{v} \\
\implies A\mathbf{v} - \alpha I\mathbf{v} &= \lambda\mathbf{v} - \alpha\mathbf{v} \\
&\text{for some real constant } \alpha \\
\implies (A - \alpha I)\mathbf{v} &= (\lambda - \alpha)\mathbf{v} \\
&\text{Pre-multiplying with } (A - \alpha I)^{-1} \text{ gives} \\
\mathbf{v} &= (A - \alpha I)^{-1}(\lambda - \alpha)\mathbf{v} \\
\frac{1}{(\lambda - \alpha)}\mathbf{v} &= (A - \alpha I)^{-1}\mathbf{v}
\end{aligned}
$$

The closer the chosen value of α is to a particular eigenvalue of the matrix A, the more scaled $\frac{1}{(\lambda-\alpha)}$

becomes, making this, the largest eigenvalue of the new matrix equation.
This can be done repeatedly for each eigenvalue by appropriately varying the value of α.

Some Advantages:

1) Highly effective method to use if the approximate value of one or more eigenvalues is known.
2) This works for both symmetric and non-symmetric matrices.
3) If the initial guess for α is close to the actual eigenvalue, the method converges rapidly.

Some Disadvantages:

1) Requires inversion of a matrix, which is computationally expensive.
2) This method relies heavily on the choice of α. A poor choice will lead to slow convergence.
3) If the matrix $(A - \alpha I)$ is nearly singular, it may cause numerical issues.
4) This method is ineffective if the eigenvalues of the matrix are closely spaced.

IV. The Jacobi Method

Algorithm Outline:
This algorithm is used on symmetric matrices, by executing a spectral decomposition.
This means that we iteratively perform orthogonal transformations (rotation operations) on the initial matrix A until we get

$$A = VDV^{\top}$$

where V is a matrix containing all the eigenvectors, and D is a diagonal matrix where each diagonal element represents an eigenvalue.

Some Advantages:

1) For smaller matrices, the Jacobi method yields highly accurate results.
2) This method is guaranteed to converge, as long as the matrix A is symmetric.
3) Works well for dense matrices.

Some Disadvantages:

1) This method is inefficient for larger matrices, due to the high cost of computation.
2) This method does not apply to non-symmetric matrices.
3) Rate of convergence is slow.

V. The QR Algorithm

Algorithm Outline:
This algorithm involves decomposing the initial matrix A as

$$A = QR$$

where Q is an orthogonal matrix, and R is an upper triangular matrix.
Updating A as

$$A = RQ$$

gives a triangular matrix with eigenvalues as the diagonal elements.

Some Advantages:

1) Applies to any type of matrix.
2) Can yield both real and complex eigenvalues.
3) Computes all eigenvalues of a given matrix A.

Some Disadvantages:

1) Is computationally expensive for large matrices.
2) Rate of convergence can be slow.
3) The algorithm involves operations on the entire matrix, and leads to high memory consumption for larger matrices.

VI. The Divide And Conquer Algorithm

Algorithm Outline:
The given matrix A is first reduced to a tridiagonal matrix T by using orthogonal transformations.
Choose a pivot point and break the matrix T into sub-matrices.
Recursively execute the above-mentioned steps until the sub-matrices obtained are small enough to calculate eigenvalues for, and find their eigenvalues.
Use the eigenvalues of the sub-matrices to find the eigenvalue of the matrix.

Some Advantages:

1) This method is highly efficient due to its nature of dividing a huge matrix into smaller, more manageable matrices.
2) Yields results with satisfactory accuracy.

Some Disadvantages:

1) Only applies to symmetric matrices.
2) The algorithm is complex to implement.
3) Less efficient for small matrices.
4) Requires additional memory to store intermediate results.

Algorithm Implemented

For this assignment, I have chosen to implement the QR algorithm for calculating the eigenvalues of a given matrix. My code stores the matrix as a nested list of the matrix's rows and performs matrix operations accordingly.

While matrix operations can be performed more conveniently on NumPy arrays, I have chosen to use nested lists so that I can better portray my level of understanding of the way the algorithm works.

I have used the Gram-Schmidt algorithm to reduce the matrix input A into the required Q and R components.