EigenValues of a Matrix

EE24BTECH11048-NITHIN.K

1 Introduction

Eigenvalues are scalar values associated with a square matrix A that satisfy:

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

where $\mathbf{v}$ is a non-zero eigenvector. Eigenvalues are fundamental to many scientific and engineering applications, including vibration analysis, quantum mechanics, and dimensionality reduction techniques like Principal Component Analysis (*PCA*).
The QR algorithm is a robust iterative method for eigenvalue computation. It iteratively transforms a matrix into an upper triangular form, from which the eigenvalues can be read directly from the diagonal. This report explores the QR algorithm's implementation.

2 Brief Description about Algorithm

The QR algorithm is an iterative method to compute the eigenvalues (and sometimes eigenvectors) of a square matrix. It relies on the QR decomposition, which represents a matrix as a product of an orthogonal matrix (Q) and an upper triangular matrix (R).

Steps of the QR Algorithm:

1) Start with a Matrix A: Begin with the given matrix A for which eigenvalues are to be computed.
2) QR Decomposition: At each iteration, perform the QR decomposition of A_k, where $A_k = Q_k R_k$.
3) Recompose the Matrix: Form a new matrix A_{k+1} by multiplying $R_k Q_k$:

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

4) Convergence: Repeat steps 2 and 3 until A_k converges to a triangular matrix (or nearly triangular). The eigenvalues of A_k are then the diagonal elements of the resulting matrix.

3 Key Insights

Memory Usage:

Efficiency: The QR algorithm requires storing the matrices A_K, Q_k and R_k. For an nxn matrix Memory usage is approximately $O\left(n^2\right)$, dominated by the storage of A_k.

Convergence Rate:

Standard QR Algorithm: Converges linearly for general matrices. convergence is faster for matrices with well-separated eigenvalues (large gaps between eigenvalues).

Challenges:Slow convergence for matrices with close eigenvalues or clusters of eigenvalues.

Suitability for different Type of Matrices:
Dense Matrices: The QR algorithm is well-suited for small-to-moderate-sized dense matrices. For large dense matrices, computational costs can become prohibitive.
Sparse Matrices: Direct application is inefficient because the QR steps often destroy sparsity.
Symmetric Matrices: For symmetric matrices, QR decomposition can converge faster because eigenvalues are guaranteed to be real, and the orthogonal structure simplifies computations.
If a matrix has defective eigenvalues (eigenvalues without a complete set of eigenvectors), the QR algorithm may converge poorly or fail to compute the full eigenvalue spectrum.

Computational Cost:
Each QR iteration involves:Decomposition: $O\left(n^3\right)$ operations, Matrix recomposition: $O\left(n^3\right)$ operations.
For k iterations, the total cost is approximately $O\left(kn^3\right)$, where k depends on the matrix's eigenvalue distribution.
Stability:
Orthogonal transformations used in QR decomposition maintain numerical stability by avoiding large round-off errors.

4 Time Complexity:

For an nxn matrix, QR decomposition using methods like Gram-Schmidt, Householder reflections, or Givens rotations has a computational cost of: $O\left(n^3\right)$.
Multiplying the matrices R_k and Q_k requires another: $O\left(n^3\right)$.
Total per iteration cost: $O\left(n^3\right) + O\left(n^3\right) = O\left(n^3\right)$.
For most practical implementations: $k \approx O(n)$ for the standard QR algorithm and $k \approx O(\log(n))$ for shifted QR and its variants. The total time complexity is the product of the cost per iteration and the number of iterations: $T(n) = k \cdot O\left(n^3\right)$
Standard QR Algorithm: $T(n) = O(n) \cdot O\left(n^3\right) = O\left(n^4\right)$
Shifted QR Algorithm: $T(n) = O(\log(n)) \cdot O\left(n^3\right) = O\left(n^3 \log(n)\right)$

5 Comparison to other Methods:

Power Iteration: $O\left(kn^2\right)$, efficient for computing a single eigenvalue but slow for the full spectrum.
Lanczos/Arnoldi: $O\left(kn^2\right)$, suitable for large sparse matrices.
Jacobi Algorithm: $O\left(n^4\right)$ slower than QR for large dense matrices.

Algorithm Variant	Time Complexity	Key Use Cases
Standard QR	$O\left(n^4\right)$	Small-to-moderate dense matrices
Shifted QR	$O\left(n^3 \log\left(n\right)\right)$	Moderate-sized dense matrices
Hessenberg-Reduced QR	$O\left(n^3\right)$	Symmetric matrices, practical usage
Tridiagonal QR	$O\left(n^3\right)$	(for symmetric)

I chose the Shifted QR Algorithm to solve the eigenvalue problem for a matrix due to the following key reasons:

Eigenvalue Decomposition: The goal is to compute the eigenvalues of a matrix. While there are several methods (like power iteration or direct methods), the Shifted QR Algorithm is specifically designed for this purpose, efficiently finding eigenvalues of large, sparse, or dense matrices.

Numerical Stability: The Shifted QR Algorithm is numerically stable and avoids the problems associated with other methods like direct computation methods or methods that rely on iteration without shifts, which can converge slowly.

The Shifted QR Algorithm uses orthogonal transformations, which preserve the norm of the matrix and are computationally efficient. This makes the algorithm relatively memory-friendly compared to more memory-heavy alternatives like Jacobi's method for eigenvalue problems or methods that require storing all eigenvectors explicitly.

The Shifted QR Algorithm is robust against issues like small numerical errors during the calculation of eigenvalues. The shifts act as a form of acceleration, improving the stability of the algorithm.

6 Alternatives

1) **Power Iteration**: is a simple algorithm used to find the dominant eigenvalue (the largest in magnitude) and its corresponding eigenvector of a matrix. The method works by repeatedly multiplying a random vector v_0 by the matrix A and normalizing the result after each multiplication. As the iterations progress, the vector converges to the eigenvector associated with the largest eigen value of A. The dominant eigenvalue can be estimated by taking the ratio of consecutive vector norms. Power iteration is efficient for sparse matrices, but it only finds the largest eigenvalue, requiring modifications (e.g., deflation) to find additional eigenvalues.
2) **Jacobi's Method**: is an iterative algorithm used to find the eigenvalues and eigenvectors of a matrix, particularly for symmetric matrices. It works by repeatedly applying a series of orthogonal transformations (rotations) to the matrix, aiming to diagonalize it. In each iteration, Jacobi's method selects the largest off-diagonal element, applies a rotation that zeroes it, and updates the matrix. The process continues until the matrix is sufficiently diagonal, with the diagonal elements representing the eigenvalues, and the transformation matrices providing the eigenvectors. While Jacobi's method is highly accurate, it is computationally expensive with an $O\left(n^3\right)$ complexity, making it inefficient for large matrices.
3) **Divide-and-Conquer Methods**: for eigenvalue computation are designed to break down a large matrix into smaller, more manageable submatrices, which can be solved independently and then combined. These methods are particularly effective

for symmetric matrices. The idea is to recursively divide the matrix into blocks or use matrix decompositions (e.g., Schur decomposition) to reduce the problem's complexity. Once the subproblems are solved, the results are merged to obtain the final eigenvalues and eigenvectors. These methods offer better performance than direct approaches like Jacobi's method, particularly for large matrices, with a computational complexity of $O\left(n^3\right)$. However, they require more sophisticated algorithms and are typically used when high accuracy and speed are needed.

4) The QR decomposition can be further made robust by using householder transformations,converting the matrix to hessenberg form reduces QR iterations and use of rayleigh quotient shift.

   Householder Transformation is a linear algebra technique used to reduce a matrix to a simpler form, often a triangular or Hessenberg matrix. It works by applying a sequence of orthogonal transformations to eliminate elements below the main diagonal of the matrix. The transformation is represented by a Householder matrix, which is symmetric and orthogonal, and is used to create zeroes in specific positions of the matrix while preserving its eigenvalues.

   To improve convergence, the QR algorithm can be enhanced by applying the Rayleigh quotient shift. This technique selects a shift σ close to an eigenvalue of **A**, effectively speeding up convergence and reducing the number of iterations.

   The Rayleigh quotient shift involves the following steps:

   a) Compute the shift:

   $$\sigma = A_{n,n},$$

   where $A_{n,n}$ is the bottom-right entry of the current matrix A, approximating an eigenvalue.

   b) Update the matrix by subtracting the shift:

   $$A - \sigma I,$$

   where I is the identity matrix.

   c) Perform QR decomposition on the shifted matrix:

   $$A - \sigma I = QR.$$

   d) Recalculate **A** by applying the inverse shift:

   $$A_{\text{new}} = RQ + \sigma I.$$

   This shifting mechanism "pushes" the matrix closer to its triangular form, accelerating convergence.