Eigenvalue Calculation: QR Algorithm Analysis and Implementation

Roopansha Bugada(AI24BTECH11006)

November 18, 2024

1 Introduction

Eigenvalues and eigenvectors are fundamental concepts in linear algebra with applications in engineering, physics, and data science.

An eigenvalue of a square matrix A is a scalar λ such that there exists a non-zero vector $\mathbf{v}$ (called the eigenvector) satisfying the equation:

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

Here, λ represents how the eigenvector $\mathbf{v}$ is scaled during the linear transformation defined by A. Eigenvalues provide insights into the properties of matrices, such as stability, resonance, and dimensionality reduction.

This report focuses on computing eigenvalues using the QR algorithm, a widely used and efficient method. The QR algorithm was selected due to its robustness and versatility in handling dense matrices.

2 QR Algorithm Description

The QR algorithm is an iterative method to compute the eigenvalues of a matrix A. The process involves the following steps:

1. Factorize A into a product of an orthogonal matrix Q and an upper triangular matrix R such that $A = QR$.
2. Update A as $A \leftarrow RQ$.
3. Repeat the process until A converges to a diagonal matrix. The diagonal elements of the converged matrix are the eigenvalues of A.

The algorithm relies on the QR decomposition, ensuring numerical stability and convergence under appropriate conditions.

2.1 Pseudocode

1. **Input**: Matrix A, tolerance ϵ, and maximum iterations N.
2. **Output**: Eigenvalues of A.

3. Initialize $k = 0$.
4. While $k < N$ and $\|A_{k+1} - A_k\|_F > \epsilon$:
5. Compute Q_k, R_k such that $A_k = Q_k R_k$.
6. Update $A_{k+1} = R_k Q_k$.
7. Increment k.
8. Return the diagonal elements of A_k.

3 Time Complexity Analysis

The QR algorithm involves repeated QR decompositions, each with a time complexity of $\mathcal{O}(n^3)$ for an $n \times n$ matrix. Assuming convergence in m iterations, the total time complexity is $\mathcal{O}(mn^3)$.

4 Comparison of Algorithms

The QR algorithm is a robust and general-purpose method for eigenvalue computation. Below is a comparison of its advantages and disadvantages relative to other algorithms.

4.1 QR Algorithm

Advantages:

- High accuracy and numerical stability for dense matrices.
- Simultaneously computes all eigenvalues.
- Handles a wide range of matrices without requiring specific conditions like symmetry or sparsity.

Disadvantages:

- Computationally expensive with a complexity of $\mathcal{O}(n^3)$ per iteration.
- Not optimal for large or sparse matrices without modifications (e.g., Hessenberg reduction).

4.2 Power Iteration

Advantages:

- Simple and easy to implement.
- Efficient for finding the largest eigenvalue of a matrix.

Disadvantages:

- Can only find the largest eigenvalue, requiring modifications (e.g., deflation) for others.
- Slow convergence, especially when eigenvalues are close in magnitude.
- Sensitive to matrix scaling.

4.3 Inverse Iteration

Advantages:

- Efficient for finding eigenvalues near a specified value.
- High accuracy when combined with shifts (shifted inverse iteration).

Disadvantages:

- Requires computing matrix inverses, increasing computational cost.
- Needs a good initial guess for eigenvalue proximity, making it less general-purpose than QR.

4.4 Jacobi Method

Advantages:

- Highly accurate and stable for symmetric matrices.
- Easy to implement for such matrices.

Disadvantages:

- Inefficient for large matrices due to slow convergence.
- Not suitable for non-symmetric matrices, limiting its use cases.

4.5 Lanczos Method

Advantages:

- Very efficient for large, sparse symmetric matrices.
- Projects the matrix to a smaller Krylov subspace, reducing dimensionality.

Disadvantages:

- Limited to symmetric matrices.
- Can lose orthogonality in computed vectors, requiring re-orthogonalization for stability.

4.6 Arnoldi Iteration

Advantages:

- Suitable for large, non-symmetric sparse matrices.
- Extends the Lanczos method to handle non-symmetric cases.

Disadvantages:

- Memory-intensive, especially for large matrices.
- Slower convergence compared to other iterative methods.

5 Insights and Observations

The QR algorithm performs well for dense matrices, providing accurate results. However, it is computationally intensive for large matrices due to its $\mathcal{O}(n^3)$ complexity per iteration. For sparse matrices, alternative methods like the Lanczos or Arnoldi iteration are preferable. Convergence depends on the matrix properties, and deflation techniques can be used to improve performance.

6 Conclusion

The QR algorithm is a robust and versatile method for eigenvalue computation, particularly for dense matrices. While computationally demanding, its stability and accuracy make it a reliable choice in practical applications. Future work could explore optimizations such as Hessenberg transformations to reduce the computational burden.