Eigenvalue Calculation

EE24BTECH11031-Jashwanth

I. Introduction

Eigenvalues are one of the most significant concepts in linear algebra and have widespread applications in diverse fields such as physics, engineering, machine learning, and numerical analysis. They provide critical insights into the behavior of systems modeled by matrices, such as stability, vibration modes, or principal directions in data.

Given a square matrix $\mathbf{A}$, eigenvalues λ are scalars that satisfy: $A\mathbf{v} = \lambda\mathbf{v}$,

where $\mathbf{v} \neq \mathbf{0}$ is the corresponding eigenvector. This report focuses on implementing and explaining a numerical approach to compute the eigenvalues of $\mathbf{A}$ using the QR decomposition method.

II. Eigenvalues

To find the eigenvalues of a matrix $\mathbf{A}$, the characteristic polynomial $\det A - \lambda I = 0$ is solved. However, solving this polynomial analytically becomes computationally challenging for matrices larger than 4×4 because there are no general formulas for roots of polynomials of degree greater than four.

Numerical methods like the QR decomposition provide a more efficient and stable alternative. These methods iteratively refine approximations to the eigenvalues by leveraging matrix factorization techniques.

III. QR Decomposition

QR decomposition is a technique to factorize a matrix $\mathbf{A}$ into the product of:

- $\mathbf{Q}$: An orthogonal (or unitary) matrix, where $\mathbf{Q}^T\mathbf{Q} = I$, and
- $\mathbf{R}$: An upper triangular matrix.

For a square matrix $\mathbf{A}$, this decomposition satisfies:

$$\mathbf{A} = \mathbf{Q}R$$

. The QR method for eigenvalue calculation is based on the idea that if $\mathbf{A}$ is repeatedly decomposed into $\mathbf{QR}$ and then reassembled as $\mathbf{RQ}$, the resulting sequence of matrices will converge to an upper triangular matrix, where the diagonal elements are the eigenvalues of $\mathbf{A}$.

IV. QR Algorithm for Eigenvalues

We start with the given matrix $\mathbf{A}$. This matrix is subjected to iterative transformations to produce sequences of $\mathbf{Q}$ and $\mathbf{R}$ matrices.

A. 1. QR Decomposition

The matrix $\mathbf{A}$ is decomposed into $\mathbf{Q}$ and $\mathbf{R}$ using the Gram-Schmidt orthogonalization process:

- The columns of $\mathbf{A}$ are orthogonalized to construct $\mathbf{Q}$, ensuring each column is perpendicular to the others.
- The projections of $\mathbf{A}$'s columns onto $\mathbf{Q}$'s columns yield $\mathbf{R}$, which is upper triangular.

B. 2. Matrix Update

Using the decomposition $\mathbf{A} = \mathbf{QR}$, the matrix is updated as:

$$\mathbf{A}' = \mathbf{RQ}$$

. This step effectively shifts eigenvalue information progressively closer to the diagonal.

C. 3. Iteration

Steps 2 and 3 are repeated iteratively until **A** converges to a quasi-diagonal matrix. Convergence is typically determined when the off-diagonal elements are sufficiently close to zero (below a predefined tolerance, e.g., 10^{-10}).

D. 4. Extraction of Eigenvalues

Once the matrix has converged, the eigenvalues of **A** are found along the diagonal of the resulting matrix.

V. Key Components in the Code

The provided implementation includes the following critical components:

A. 1. Matrix Multiplication

This function multiplies two matrices **A** and **B** to produce their product **C**. It is used in reconstructing $\mathbf{A}' = \mathbf{RQ}$ during each iteration.

B. 2. Dot Product

The dot product function computes the projection of one column of the matrix onto another. This is essential for calculating the entries of **R** and ensuring orthogonalization during the Gram-Schmidt process.

C. 3. QR Decomposition

The QR decomposition function performs the Gram-Schmidt process:

- **Orthogonalization**: Subtracts projections of a column onto previous **Q** -columns to produce orthogonal vectors.
- **Normalization**: Scales the orthogonal vectors to unit length.

D. 4. Iterative Update

In each iteration, the matrix **A** is decomposed into **Q** and **R**, and then recombined as **RQ**. The process continues until **A** converges to a quasi-diagonal form.

VI. Advantages of the QR Method

A. 1. Numerical Stability

The QR algorithm avoids explicit computation of the characteristic polynomial, reducing the numerical instability associated with root-finding algorithms for higher-degree polynomials.

B. 2. Efficiency

The QR method is computationally efficient for dense matrices, particularly when combined with techniques like the Hessenberg reduction for preprocessing **A**.

C. 3. Generality

This approach can handle both symmetric and non-symmetric matrices. For symmetric matrices, the algorithm converges more rapidly, and the resulting eigenvalues are real.

D. 4. Practical Utility

The QR method is widely applicable in solving problems in signal processing, data compression (e.g., Principal Component Analysis), and structural analysis.

VII. Limitations

While the QR method is robust, it has some challenges:

- **Convergence Rate**: The method may require many iterations to converge for certain matrices, especially those with close or repeated eigenvalues.
- **Computational Cost**: Each iteration involves matrix multiplications and QR decompositions, which can be costly for very large matrices.
- **Preprocessing Requirements**: For large or sparse matrices, preprocessing steps like Hessenberg reduction are often required to improve efficiency.

VIII. Conclusion

The QR decomposition method is a powerful and practical technique for calculating eigenvalues. It relies on the iterative refinement of a matrix to a diagonal form by leveraging orthogonal transformations. The method is robust, numerically stable, and widely applicable across scientific and engineering domains.

The provided code uses the Gram-Schmidt process for QR decomposition and iteratively applies the **QR** method to compute eigenvalues. While efficient for moderate-sized matrices, optimizing for large-scale problems may involve further refinements, such as introducing shift strategies or leveraging sparsity.

This implementation serves as an excellent introduction to eigenvalue computation using numerical methods and highlights the utility of linear algebra in computational problem-solving.