Eigenvalue Computation Using Householder and Givens Rotation Methods

Your Name

November 16, 2024

Introduction

Eigenvalue computation is a cornerstone of numerical linear algebra with widespread applications across sciences and engineering. This report delves into the implementation of eigenvalue computation using Householder transformations and Givens rotations. Additionally, the theoretical foundation of these methods is explored, and the impracticality of solving the characteristic polynomial $\det(A - \lambda I) = 0$ directly is discussed.

Theoretical Background

Householder Transformations

Householder transformations are used to simplify a given matrix into a Hessenberg form (for general matrices) or tridiagonal form (for symmetric matrices). A Householder transformation is defined by:

$$H = I - 2\frac{uu^T}{u^T u},$$

where u is a carefully chosen vector such that when H is applied to a vector x, the result aligns with the standard basis vector.

This orthogonal transformation has the following properties:

- H is unitary ($H^T H = I$), ensuring numerical stability.
- Householder transformations are efficient, requiring $O(n^2)$ operations for each step, leading to an $O(n^3)$ total complexity for reducing a matrix to Hessenberg form.

In eigenvalue computations, transforming a matrix into Hessenberg form significantly simplifies subsequent steps, as the structure reduces the number of computations needed.

Givens Rotations

Givens rotations are another class of orthogonal transformations, particularly suited for introducing zeros into matrices. A Givens rotation matrix $G(i, j, \theta)$ affects only rows i and j of a matrix, and is given by:

$$G(i, j, \theta) = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}.$$

When G is applied to a matrix A, it rotates the rows i and j to eliminate specific elements.
Why Givens Rotations?

- Givens rotations work well with sparse matrices as they affect only two rows or columns at a time.
- They are numerically stable because they avoid amplifying rounding errors.
- In the QR algorithm, Givens rotations are particularly useful for computing the decomposition of Hessenberg matrices.

Why Not Solve $\det(A - \lambda I) = 0$ Directly?

The characteristic polynomial $\det(A - \lambda I) = 0$ theoretically provides all eigenvalues of A. However, solving it directly is impractical for several reasons:

- **Polynomial Root-Finding:** Solving a general polynomial of degree $n \geq 5$ exactly is impossible using radicals (as per Galois theory). Numerical methods must be used instead.
- **Ill-Conditioning:** The roots of the characteristic polynomial are highly sensitive to small perturbations in the coefficients, leading to numerical instability.
- **Computational Cost:** Constructing the determinant is computationally expensive ($O(n!)$ in naive methods), whereas modern algorithms exploit matrix structure for efficiency.

Chosen Algorithm

The eigenvalue computation algorithm combines:

1. **Householder Transformations:** To reduce the matrix to Hessenberg form, simplifying subsequent computations.
2. **QR Iterations with Givens Rotations:** To iteratively compute an upper triangular matrix, where the diagonal elements converge to the eigenvalues.

Time Complexity Analysis

Householder Transformations

Reducing an $n \times n$ matrix to Hessenberg form takes $O(n^3)$ operations.

QR Iterations

Each QR decomposition requires $O(n^2)$ operations. Assuming $O(n)$ iterations for convergence, this step also requires $O(n^3)$ operations.

Overall Complexity

The total time complexity of the combined approach is $O(n^3)$, making it competitive with other eigenvalue computation methods.

Other Insights

- **Memory Usage:** Requires $O(n^2)$ memory for storing matrices.
- **Convergence Rate:** Quadratic convergence is achieved for well-conditioned matrices. Complex eigenvalues or nearly identical eigenvalues may require more iterations.
- **Suitability:** Effective for dense matrices but less optimal for sparse or structured matrices where specialized algorithms may outperform.

Comparison of Algorithms

Algorithm	**Time Complexity**	**Accuracy**	**Suitability**
Power Iteration	$O(n^2)$ per iteration	Low for non-dominant eigenvalues	Dominant eigenv
QR Algorithm	$O(n^3)$	High	Dense, general ma
Jacobi Method	$O(n^3)$	High for symmetric matrices	Symmetric mat
Arnoldi Iteration	$O(k^2n)$	Moderate	Sparse, large ma
Householder + Givens	$O(n^3)$	High	Dense, general ma

Table 1: Comparison of eigenvalue computation algorithms.

1 Code Implementation and Explanation

The goal of the code is to compute the eigenvalues of a given complex matrix using the Householder transformation followed by QR decomposition with Givens rotations. Below, we provide a detailed explanation of the implemented methods and their theoretical basis.

1.1 Overview of the Code

The code is divided into several functions, each responsible for a specific part of the eigenvalue computation process:

1. **Matrix Initialization:** The matrix is initialized with complex numbers to demonstrate the algorithm's capability to handle complex matrices.
2. **Householder Transformation:** Used to transform the matrix into an upper Hessenberg form.
3. **QR Decomposition:** Performed using Givens rotations to obtain a factorization of the Hessenberg matrix.

4. **Iterative QR Algorithm:** The eigenvalues are extracted by iteratively applying QR decomposition and recombining the results.

The code outputs the computed eigenvalues and intermediate results for verification purposes.

1.2 Functions Explained

1.2.1 Householder Transformation

The `householder` function transforms the input matrix into an upper Hessenberg form, which simplifies subsequent QR decomposition steps. The steps are as follows:

- For each column of the matrix (excluding the last), a Householder reflector is computed. This reflector zeroes out all elements below the diagonal in the current column.
- The reflector is constructed as $P = I - 2uu^T$, where u is a unit vector.
- The matrix is updated using $A' = P^T AP$, maintaining symmetry for Hermitian matrices.

This transformation reduces computational complexity for QR decomposition from $O(n^4)$ to $O(n^3)$ and preserves eigenvalues.

1.2.2 QR Decomposition with Givens Rotations

The `upper_triangular` function computes the QR decomposition of the Hessenberg matrix using Givens rotations. The steps are:

- A Givens rotation is applied to zero out specific elements in the matrix, targeting subdiagonal entries one at a time.
- The rotation matrix G is constructed such that $G^T R$ introduces zeros in the desired positions.
- The process is repeated for all subdiagonal elements.
- The product of the Givens rotation matrices forms Q, and the upper triangular matrix forms R.

The QR decomposition is essential for iterative eigenvalue computation.

1.2.3 Iterative Eigenvalue Computation

The iterative process repeatedly applies QR decomposition and recombines Q and R to update the matrix as $A = RQ$. This convergence method ensures the diagonal elements approximate the eigenvalues.

1.2.4 Supporting Functions

Several utility functions support the main computation:

- `matscale`: Scales a matrix by a constant factor.
- `matrix_multiply`: Multiplies two matrices.
- `transpose`: Computes the conjugate transpose of a matrix.
- `eye`: Generates an identity matrix.
- `print_matrix`: Outputs a matrix for debugging purposes.

1.3 Why Use Householder and Givens?

Householder Transformations:

- Efficient for reducing matrices to a simpler form (upper Hessenberg or tridiagonal).
- Provides numerical stability.
- Reduces the complexity of subsequent steps, like QR decomposition.

Givens Rotations:

- Ideal for zeroing specific matrix elements without affecting others.
- Well-suited for sparse matrices.
- Simple to implement and numerically stable.

1.4 Challenges in Solving $\det(A - \lambda I) = 0$

The direct computation of eigenvalues by solving $\det(A - \lambda I) = 0$ is impractical for several reasons:

- The polynomial roots are highly sensitive to perturbations, leading to numerical instability.
- For large matrices, the degree of the polynomial is high, making root-finding algorithms computationally expensive.
- Closed-form solutions exist only for polynomials of degree 4 or lower.

Iterative methods like the one implemented in this code are preferred due to their numerical stability and scalability for large matrices.

1.5 Output Verification

The code produces intermediate outputs, such as the Q and R matrices during each QR iteration, to verify correctness. The final eigenvalues are extracted from the diagonal of the matrix after sufficient iterations.

Conclusion

The combination of Householder transformations and Givens rotations provides a robust and efficient method for eigenvalue computation, especially for dense matrices. While other methods may be tailored for specific matrix types, this approach balances numerical stability, computational efficiency, and broad applicability.

References

1. G. H. Golub and C. F. Van Loan, *Matrix Computations*, Johns Hopkins University Press.

2. Trefethen, L. N., and Bau, D., *Numerical Linear Algebra*, SIAM.

3. Higham, N. J., *Accuracy and Stability of Numerical Algorithms*, SIAM.