Software Assignment: Report

DESABOINA SRI SATHWIK-AI24BTECH11007
ai24btech11007@iith.ac.in

Used algorithm: Shifted QR algorithm

1 Introduction

The algorithm which I used to calculate the eigen values of the matrices of any type is **Shifted QR algorithm**. This report presents a detailed analysis of the shifted QR algorithm for eigenvalue computation, including its implementation, complexity analysis, and comparison with other prominent methods. We examine the algorithm's convergence properties, memory requirements, and suitability for different types of matrices. The analysis includes comparisons with power method, inverse iteration, and other variants of the QR algorithm in terms of computational efficiency and numerical stability. Here, I also used two shift strategies which Wilkinson shift and Rayleigh shift.

Eigenvalue computation is a fundamental problem in numerical linear algebra with applications ranging from principal component analysis to quantum mechanics. The shifted QR algorithm represents a powerful method for computing all eigenvalues of a matrix simultaneously.

2 Algorithm Analysis

2.1 Implementation Overview

The implemented shifted QR algorithm uses the following key components:

Algorithm 1 Shifted QR Algorithm

1: Initialize matrix A
2: **while** not converged **do**
3: Compute shift μ
4: A $\leftarrow$ A - μI
5: Compute QR decomposition of A
6: A $\leftarrow$ RQ + μI
7: **end while**

2.2 Time Complexity Analysis

The computational complexity can be broken down as follows:

- QR decomposition using Givens rotations: O(n^3) per iteration
- Shift computation: O(1)
- Matrix multiplication (RQ): O(n^3) per iteration

Total complexity: O(kn^3) where k is the number of iterations needed for convergence.

2.3 Memory Usage

Memory requirements:

- Original matrix: n^2 doubles
- Working matrix: n^2 doubles
- Eigenvalue storage: n doubles
- Additional temporary storage: O(1)

Total memory complexity: O(n^2)

2.4 Convergence Analysis

The convergence rate depends on the eigenvalue separation:

$$\text{Rate} \approx \left|\frac{\lambda_{i+1}}{\lambda_i}\right|$$

where λ_i are the eigenvalues ordered by magnitude.

For the Wilkinson shift strategy:

- Cubic convergence for well-separated eigenvalues
- Linear convergence for clustered eigenvalues
- Typically requires 10-30 iterations per eigenvalue

3 Comparison with Other Methods

Table 1: Comparison of Eigenvalue Methods

Method	Time Complexity	Memory	Convergence Rate
Power Method	O(kn^2)	O(n)	Linear
Inverse Iteration	O(kn^3)	O(n^2)	Linear/Quadratic
QR without shifts	O(kn^3)	O(n^2)	Linear
Shifted QR	O(kn^3)	O(n^2)	Cubic
Divide & Conquer	O(n^3)	O(n^2)	-

3.1 Suitability Analysis

3.1.1 Dense Symmetric Matrices

- Shifted QR: Excellent (maintains symmetry)
- Power Method: Poor (finds only dominant eigenvalue)
- Divide & Conquer: Excellent (exploits structure)

3.1.2 Sparse Matrices

- Shifted QR: Poor (fills in zeros)
- Power Method: Excellent (preserves sparsity)
- Arnoldi/Lanczos: Excellent (exploits sparsity)

3.1.3 Non-symmetric Matrices

- Shifted QR: Good (handles complex eigenvalues)
- Power Method: Limited (real dominant eigenvalue only)
- Arnoldi: Good (handles non-symmetry well)

4 Performance Analysis

4.1 Accuracy Considerations

The implemented algorithm achieves high accuracy due to:

- Use of Givens rotations (numerically stable)
- Wilkinson shift strategy (improved convergence)
- Deflation technique (reduces accumulation of errors)

Error bound for computed eigenvalues λ:

$$|\lambda - \lambda| \leq O(\epsilon\|A\|)$$

where ϵ is machine precision and

$$\|A\|$$

is the matrix norm.

4.2 Special Cases

4.2.1 Ill-conditioned Matrices

For matrices with condition number K:

$$\text{Relative Error} \leq O(k\epsilon)$$

4.2.2 Matrices with Multiple Eigenvalues

- Convergence slows for repeated eigenvalues
- Accuracy may decrease due to sensitivity
- Special handling needed for clustered eigenvalues

5 Implementation Optimizations

5.1 Current Optimizations

- Deflation to reduce problem size
- Adaptive shift strategy
- Early termination for converged eigenvalues
- Minimal memory allocation/deallocation

5.2 Potential Improvements

- Block operations for cache efficiency
- Parallel implementation of rotations
- Adaptive convergence criteria
- Specialized handling for sparse matrices

6 Conclusion

The implemented shifted QR algorithm provides a robust and efficient method for computing eigenvalues. Its O(n^3) complexity is competitive with other methods, while offering better convergence properties for general matrices. The method is particularly suitable for dense matrices where all eigenvalues are needed.

Key advantages:

- Simultaneous computation of all eigenvalues

- Cubic convergence with good shift strategy
- Numerically stable implementation

Primary limitations:

- High memory requirements
- Not optimal for sparse matrices
- Performance degradation for clustered eigenvalues