MATRIX THEORY - EE1030
Software Assignment

ee24btech11058 - P.Shiny Diavajna

Aim : To compute the eigenvalues of a given matrix using C code.

Introduction :

- **What are Eigenvalues ?**
  Eigenvalues are associated with a square matrix A and represent scalars,λ, such that when a matrix multiplies a vector v, the result is the vector v scaled by λ. This relationship is expressed as:

$$Av = \lambda v$$

Chosen Algorithm to calculate Eigenvalues:

QR Algorithm with Gram Schmidt Orthogonalization

- The QR algorithm is an iterative method for calculating eigenvalues of a matrix. It repeatedly factors a given matrix A into a product of an orthogonal matrix Q and an upper triangular matrix R, then updates A as $A = RQ$. Over iterations, A converges to a quasi-diagonal matrix, where the diagonal elements approximate the eigenvalues.
- **Gram-Schmidt Orthogonalization** is used to compute the orthogonal matrix Q during QR factorization. It ensures numerical stability by orthogonalizing the columns of A explicitly.

Time Complexity:

- **QR Factorization with Gram-Schmidt:** For an $n \times n$matrix, the Gram-Schmidt process has a complexity of $O(n^3)$.
- **Overall QR Algorithm :** Since the QR factorization is performed iteratively ,the total complexity depends on the number of iterations k. For k iterations, the complexity is $O(kn^3)$. For matrices that converge quickly (e.g., symmetric matrices), k is often small.

Other Aspects :

1) **Memory Usage:**The QR algorithm requires storing the matrices Q and R, making memory usage approximately $O(n^2)$.

2) **Convergence Rate :** The convergence of the QR algorithm depends on the type of matrix
   - The algorithm converges rapidly , often quadratically, without additional enhancements.
   - Convergence can be slower and may require shifts or deflation techniques to accelerate. The rate also depends on how well-separated the eigenvalues requiring more iterations.

3) **Suitability:**
   - **Symmetric Matrices:** Excellent due to faster convergence and numerical stability.
   - **Sparse Matrices:** the algorithm may not preserve sparsity, leading to increased computational costs.

- **Large Matrices:** High computational cost per iteration makes it less practical for very large matrices. However,it can handle moderately large matrices effectively.

Pros of the QR Algorithm:

1) **Widely Applicable :** Works for general square matrices, including non-symmetric and non-diagonalizable ones.

2) **Stability:** Numerically stable for many types of matrices, especially symmetric matrices.

3) **High efficiency for Small to Medium Matrices:**
   - Scales well for matrices of moderate size.
   - Reduces the matrix iteratively while preserving eigenvalues, avoiding explicit determinant calculations.

4) **Simplicity:** Straightforward to implement and adapt for specific matrix types (e.g., Hessenberg or tridiagonal matrices).

Cons of the QR Algorithm:

1) **Computational Complexity:** Requires $O(n^3)$ operations per iteration for a full matrix, which may be inefficient for very large matrices.

2) **Memory Usage:** Needs significant memory for intermediate calculations, especially for dense matrices.

3) **Less efficient for Sparse Matrices :** Algorithms like Arnoldi Iteration or Lanczos Method are better suited for sparse matrices.

4) **Slow Convergence for some Matrices:** Without shifts, convergence can be slow or fail for certain matrices.

Comparison with other algorithms:

Algorithm	Time Complexity	Accuracy	Suitability
QR Algorithm	$O(kn^3)$	High for all matrices	Best for dense, symmetric matrices
Power Iteration	$O(n^2k)$	Good for dominant eigenvalue	Best for largest eigenvalue only
Jacobi Method	$O(n^3 \log n)$	High for symmetric matrices	Best for symmetric matrices
Lanczos Method	$O(kn^2)$	Moderate	Best for sparse symmetric matrices
Divide and Conquer	$O(n^3)$	High	Suitable for symmetric matrices

Why Choose the QR Algorithm ?

- **General Purpose:** It handles most types of matrices reliably,making it a go-to method when matrix-specific optimizations aren't critical.

- **Accuracy:** Produces highly accurate eigenvalues, particularly for symmetric or well-conditioned matrices.

- **Iterative Improvement** By reducing the matrix iteratively, it avoids explicitly solving high-degree characteristic polynomials, which can be unstable numerically.

When is QR algorithm the best choice ?

- The matrix size is small to medium.
- The matrix is dense,symmetric or requires precise eigen values.

Conclusion:

The *QR* algorithm with Gram-Schmidt orthogonalization is a robust and accurate method for eigenvalue computation, particularly for symmetric and dense matrices. However, its $O(kn^3)$ complexity makes it less efficient for very large or sparse matrices, where specialized methods like Lanczos are more suitable. For practical applications, matrix properties and computational resources dictate the choice of algorithm.