Software Assignment

AI24BTECH11017 - MAANYA SRI REDDY

1 Various Algorithms To Compute Eigen Values

1) **Power Iteration:**
The Power Iteration method is a simple and efficient algorithm used to compute the largest eigenvalue and its corresponding eigenvector of a matrix.
METHOD:
Let A be an $n \times n$ matrix with eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_n$, sorted so that

$$|\lambda_1| > |\lambda_2| \geq \cdots \geq |\lambda_n|, \tag{1.1}$$

and corresponding eigenvectors $v_1, v_2, \ldots, v_n$.
Given any initial vector b_0 (not orthogonal to v_1), the matrix A applied iteratively to b_0 amplifies the contribution of the eigenvector corresponding to λ_1 due to its dominance in magnitude:

$$b_k = A^k b_0 \approx \lambda_1^k v_1. \tag{1.2}$$

After normalization, b_k converges to the eigenvector v_1, and the corresponding eigenvalue can be estimated as:

$$\lambda_1 = \frac{b_k^\top A b_k}{b_k^\top b_k}. \tag{1.3}$$

Using the power iteration method:

$$b_1 = Ab_0, \tag{1.4}$$
$$b_2 = Ab_1, \tag{1.5}$$
$$b_3 = Ab_2, \tag{1.6}$$

and so on . At each step, the Rayleigh quotient provides an approximation of the dominant eigenvalue:

$$\lambda_1 \approx \frac{b_k^\top A b_k}{b_k^\top b_k}. \tag{1.7}$$

Time Complexity
Each iteration involves a matrix-vector multiplication, which has a complexity of $O(n^2)$ for a dense $n \times n$ matrix. For m iterations, the total complexity is:

$$O(mn^2).$$

Sparse matrices significantly reduce this cost, as only the non-zero elements are involved in computation.
Advantages

- **Simplicity**: Easy to implement with minimal computational overhead.
- **Efficiency**: Works well for sparse matrices or when only the largest eigenvalue is required.
- **Memory Efficiency**: Requires storing only the matrix A and the vector b_k, making it suitable for large matrices.

Limitations

- **Single Eigenvalue**: Finds only the dominant eigenvalue; other methods are required to find the rest.
- **Slow Convergence**: Convergence rate depends on the ratio $\left|\frac{\lambda_2}{\lambda_1}\right|$. If these values are close, the method converges slowly.
- **Not Suitable for Complex Eigenvalues**: Works best with real and dominant eigenvalues.

2) **QR Algorithm:**
The algorithm works by iteratively factorizing a matrix into its QR decomposition (a product of an orthogonal matrix Q and an upper triangular matrix R), and then updating the matrix through these factorizations. The primary purpose of the QR algorithm is to find the eigenvalues of a matrix by transforming it into a simpler form.
METHOD:
QR Decomposition
The QR decomposition is a factorization of a matrix A into a product of an orthogonal matrix Q and an upper triangular matrix R, i.e.

$$A = QR, \tag{2.1}$$

Iteration Process
After computing the QR decomposition of the matrix A, we update the matrix by performing:

$$A_{k+1} = R_k Q_k, \tag{2.2}$$

where A_k is the matrix at the k-th iteration, and Q_k and R_k are the orthogonal and upper triangular matrices from the QR decomposition of A_k, respectively.
The idea is that as the iterations progress, the matrix A_k converges to a form where the diagonal elements are the eigenvalues of the original matrix A.
Convergence
After several iterations, the matrix A_k becomes close to an upper triangular matrix, and the eigenvalues of A can be read from its diagonal entries. If A is a real, symmetric matrix, the QR algorithm converges more quickly, and the diagonal elements of A_k converge to the eigenvalues of A. **Computational Complexity**
Each QR decomposition requires $O(n^3)$ operations, where n is the size of the matrix. If m iterations are needed for convergence, the total computational cost of the algorithm is:

$$O(mn^3). \tag{2.3}$$

Advantages

- **General Applicability**: The QR algorithm can be applied to any square matrix, whether symmetric or non-symmetric.
- **Accuracy**: The algorithm can provide accurate results, particularly for symmetric matrices.
- **Diagonalization**: It can not only compute eigenvalues but also provides the basis for computing eigenvectors.

Limitations

- **Slow Convergence for Non-Symmetric Matrices**: The convergence can be slow, especially when the matrix is non-symmetric or has eigenvalues that are close in magnitude.
- **Computationally Intensive**: The method can be computationally expensive, especially for large matrices, as the QR decomposition has a cubic time complexity.

3) **Lanczos Method:**
The Lanczos method is an iterative algorithm used to compute eigenvalues and eigenvectors of large symmetric matrices.The Lanczos method is based on the idea of orthogonalizing a sequence of vectors (starting from a random initial vector) that spans a subspace of the matrix. These vectors are used to build a tridiagonal matrix whose eigenvalues approximate the eigenvalues of the original matrix.
METHOD:
Lanczos Method:
The Lanczos method is an iterative algorithm used to compute eigenvalues and eigenvectors of large symmetric matrices. It is particularly efficient for sparse symmetric matrices.
Initialization: Start with a symmetric matrix A and an initial vector b_0, which should not be orthogonal to the eigenvector corresponding to the largest eigenvalue. Normalize b_0 to obtain the first vector v_1:

$$||b_0|| = 1 \quad \text{and} \quad v_1 = b_0$$

Iterative Process: For each iteration k, the Lanczos method builds an orthogonal basis for the Krylov subspace spanned by the vectors $Av_1, Av_2, \ldots, Av_k$. The key steps of the iteration are:

$$\begin{aligned} \text{1. Apply the matrix } A \text{ to the vector } v_k: &\quad w_k = Av_k \\ \text{2. Compute the coefficients } \alpha_k \text{ and } \beta_{k-1}: &\quad \alpha_k = v_k^\top A v_k \\ \text{3. Orthogonalize } w_k: &\quad w_k = w_k - \alpha_k v_k - \beta_{k-1} v_{k-1} \\ \text{4. Normalize } w_k \text{ to form the next vector: } &\quad \beta_k = ||w_k||, \quad v_{k+1} = \frac{w_k}{\beta_k} \end{aligned}$$

The vector v_{k+1} is added to the orthogonal set, and the process repeats for $k = 1, 2, 3, \ldots$.
Tridiagonal Matrix: After k iterations, the Lanczos method produces a tridiagonal matrix T_k formed by the coefficients α_k on the diagonal and β_k on the subdiagonal

and superdiagonal:

$$T_k = \begin{pmatrix} \alpha_1 & \beta_1 & 0 & \cdots & 0 \\ \beta_1 & \alpha_2 & \beta_2 & \cdots & 0 \\ 0 & \beta_2 & \alpha_3 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & \alpha_k \end{pmatrix} \tag{3.1}$$

Convergence: After several iterations, the algorithm provides increasingly better approximations of the eigenvalues of A. The Lanczos method is often used in conjunction with other techniques (e.g., the Arnoldi method) to refine the solution or to compute more eigenvalues.

Advantages

- **Efficiency for Sparse Matrices:** Requires only matrix-vector multiplications, making it computationally efficient for large sparse matrices.
- **Memory Efficiency:** Only a few vectors need to be stored, making it suitable for very large matrices.
- **Quick Convergence for Symmetric Matrices:** Converges quickly to the largest or smallest eigenvalues of symmetric matrices.

Limitations

- **Symmetry Requirement:** The Lanczos method works only for symmetric matrices, not for general (non-symmetric) matrices.
- **Loss of Orthogonality:** Over multiple iterations, the vectors v_k can lose orthogonality, causing numerical instability (known as Lanczos breakdown). Reorthogonalization can mitigate this issue but increases computational cost.
- **Limited to Few Eigenvalues:** Typically used to compute only the largest or smallest eigenvalues. For all eigenvalues, other methods might be more efficient.

4) **Characteristic Polynomial Method :**
It is the most simplest way of finding eigen values of a simple matrix.
METHOD:
Construct the Matrix $(A - \lambda I)$:
Subtract λI, which is a diagonal matrix with λ along the diagonal, from the matrix A:

$$A - \lambda I = \begin{bmatrix} a_{11} - \lambda & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} - \lambda & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} - \lambda \end{bmatrix}. \tag{4.1}$$

The resulting matrix depends on the variable λ.
Formulate the Determinant:
Compute the determinant of $(A - \lambda I)$. This will result in a polynomial in λ of degree n, called the characteristic polynomial

$$p(\lambda) = \det(A - \lambda I). \tag{4.2}$$

Solve the Characteristic Equation:
Solve the characteristic equation:

$$p(\lambda) = 0, \tag{4.3}$$

to find the roots λ. These roots are the eigenvalues of the matrix A.
Advantages
Conceptual Simplicity:

- The method is straightforward and directly based on the definition of eigenvalues as roots of the characteristic polynomial:

$$\det(A - \lambda I) = 0. \tag{4.4}$$

- Easy to understand and implement for small matrices.

General Applicability:

- Can be applied to any square matrix (symmetric or non-symmetric).
- Does not require the matrix to have specific properties like sparsity or symmetry.

No Need for Preconditioning:

- Unlike iterative methods such as Power Iteration or Lanczos, this approach does not require selecting an initial vector or constructing subspaces.

Disadvantages
Computational Complexity:

- Computing the determinant recursively for $(A - \lambda I)$ is computationally expensive. For an $n \times n$ matrix, this has a time complexity of $O(n!)$ for exact calculations, making it impractical for large matrices.
- Even with numerical approximations, evaluating determinants repeatedly for different λ values can be slow.

Numerical Instability:

- Determinants can be sensitive to small changes in λ, leading to inaccuracies in finding roots.
- Small perturbations in the matrix elements can significantly affect the computed eigenvalues.

Approximation Issues:

- The method relies on numerical root-finding, which might miss closely spaced eigenvalues if the step size for λ is too large.
- Requires fine-tuning of the step size and range of λ, which can be time-consuming.

2 Algorithm I Chose: QR Algorithm

The method I chose is the QR Algorithm because the remaining won't work for all the matrices. For example, with Power iteration method we can only find the maximum eigen value and should use another methods for finding remaining values. Jacobi Method and Divide-and-Conquer Algorithm works only for symmetric matrices. Characteristic Polynomial Method works efficiently only for small matrices upto n=2 or 3 . QR Algorithm is the only works for both symmetric and non-symmetric matrices and also for large size matrices.

We aim to find the eigenvalues of the matrix A using the QR algorithm.

$$A = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} \tag{4.5}$$

Step 1: Compute the QR Decomposition of A
The QR decomposition expresses A as $A = QR$, where:

- Q is an orthogonal matrix ($Q^T Q = I$).
- R is an upper triangular matrix.

For $A = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}$:

$$q_1 = \frac{\begin{bmatrix} 2 \\ 1 \end{bmatrix}}{\|\begin{bmatrix} 2 \\ 1 \end{bmatrix}\|} = \frac{\begin{bmatrix} 2 \\ 1 \end{bmatrix}}{\sqrt{2^2 + 1^2}} = \begin{bmatrix} \frac{2}{\sqrt{5}} \\ \frac{1}{\sqrt{5}} \end{bmatrix}, \tag{4.6}$$

$$q_2 = \frac{\text{column 2 of } A - (\text{projection onto } q_1)}{\|\cdots\|} \tag{4.7}$$

$$= \begin{bmatrix} -\frac{1}{\sqrt{5}} \\ \frac{2}{\sqrt{5}} \end{bmatrix}. \tag{4.8}$$

Thus:

$$Q = \begin{bmatrix} \frac{2}{\sqrt{5}} & -\frac{1}{\sqrt{5}} \\ \frac{1}{\sqrt{5}} & \frac{2}{\sqrt{5}} \end{bmatrix}, \tag{4.9}$$

$$R = Q^T A = \begin{bmatrix} \sqrt{5} & \frac{4}{\sqrt{5}} \\ 0 & \sqrt{5} \end{bmatrix}. \tag{4.10}$$

Therefore:

$$A = QR. \tag{4.11}$$

Step 2: Update A
Compute:

$$A_1 = RQ, \tag{4.12}$$

which gives:

$$A_1 = \begin{bmatrix} 3 & 0 \\ 0 & 1 \end{bmatrix}. \tag{4.13}$$

Step 3: Extract Eigenvalues
When A_1 converges to an upper triangular matrix, the diagonal elements are the eigenvalues. In this case:

$$\text{Eigenvalues: 3 and 1.} \tag{4.14}$$

Summary

Using the QR algorithm:

$$\text{Eigenvalues of } A = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} \text{ are 3 and 1.} \tag{4.15}$$