# Software Assignment

Rudrax Kumar Garwa
AI24BTECH11029

November 17, 2024

## Methods to calculate Eigenvalues of a Matrix

After extensively searching for different algorithms I found the following methods to find eigenvalues of a matrices are widely used:

### Characteristic Equation

For a square matrix of size $n \times n$ eigenvalues $\lambda$ is given by the solution of equation

$$\det(A - \lambda I) = 0$$

where $I$ is identity matrix of same size as $A$ and $\det(A - \lambda I)$ denote determinant of $(A - \lambda I)$. The determinant is a polynomial in $\lambda$, and the roots of this polynomial are the eigenvalues.
The time complexity of this method depends on the cost of calculating the determinant and solving the resulting polynomial equation.
Methods like co factor expansion takes exponential time but methods like Gaussian elimination or LU decomposition can reduce it to $O(n^3)$. Solving polynomial equation can have varying time complexity depending on method used.
Characteristic equation is a fundamental way but can be computationally expensive for larger matrices, especially when calculating exact values.

### Diagonalization

Diagonalization is a mathematical process used to simplify a square matrix by converting it into a diagonal form, which makes it easier to find its eigenvalues.
A matrix $A$ is said to be diagonalizable if it can be written as:

$$A = PDP^{-1}$$

where $A$ is the original matrix, $P$ is the matrix of eigenvectors of $A$ and $D$ is the diagonal matrix with eigenvalues of $A$ on the diagonal.
To diagonalize a matrix $A$, we first need to find its eigenvalues, which are solutions to the characteristic equation:

$$\det(A - \lambda I) = 0$$

For each eigenvalue $\lambda$, find the corresponding eigenvector $v$ by solving the system:

$$(A - \lambda I)v = 0$$

Form the matrix $P$ eigenvectors as columns in the matrix. The matrix $A$ is diagonalized as $A = PDP^{-1}$.
Diagonalization is particularly efficient when the matrix $A$ has special properties such as symmetric or sparse matrices.
To find the eigenvalues we need to solve characteristics equation which involves finding the determinant of the matrix $A - \lambda I$. IT have a time complexity of $O(n^3)$ using standard methods and $O(n^2)$ for special methods used for special matrices.
Then for each eigenvalue we solve a linear system $(A - \lambda I)v = 0$. It has a time complexity of $O(n^3)$ using Gaussian elimination or similar methods. Inverting a the matrix $P$ has a time complexity of $O(n^3)$.
So, overall time complexity of diagonalizing a $n \times n$ matrix is $O(n^3)$.

## Power Iteration Method

Power iteration method is simple method to find the eigenvalue having largest absolute value. It is particularly useful for large, sparse matrices where other methods like direct eigenvalue decomposition may be computationally expensive.
The method is based on the idea that repeated matrix-vector multiplication will cause a vector to converge to the eigenvector corresponding to the largest eigenvalue. Given a square matrix $A \in \mathbb{R}^{n \times n}$, start with an arbitrary vector $b_0$.
For each $k$:

$$b_{k+1} = Ab_k$$

Normalize the vector $b_{k+1}$ to prevent overflow or underflow:

$$b_{k+1} = \frac{b_{k+1}}{||b_{k+1}||}$$

After convergence, when the change in the vector $b_k$ becomes sufficiently small, the largest eigenvalue $\lambda_{max}$ is given by:

$$\lambda_{max} = \frac{{b_k}^T A b_k}{{b_k}^T b_k}$$

This is called Rayleigh quotient, which gives good estimation of dominant eigenvalue.
For matrix multiplication $Ab_k$, time complexity is $O(n^2)$. The number of iterations ($k$) depends on the difference between the largest and second-largest eigenvalues in magnitude. If this magnitude is is large, convergence is faster. The method converges roughly at a rate proportional to $(\frac{\lambda_2}{\lambda_1})^k$, where $\lambda_1$ is

largest and $\lambda_2$ is second largest eigenvalue. The number of iterations required for convergence is $O(\log(\frac{\lambda_1}{\lambda_2}))$. So overall time complexity is $O(n^2 \cdot k)$, where $k$ is number of iterations and $n$ is the size of the matrix.
This method is easy to understand and memory efficient. Converges rapidly if the difference in largest and second largest eigenvalue is large.
Power Iteration gives only the largest eigenvalue. converges very slowly if the difference in largest and second largest eigenvalue is small. The convergence rate can depend on initial vector. If the initial vector is orthogonal to the dominant eigenvector, the method will not converge to the correct eigenvalue.
Power Iteration method is computationally efficient for finding the largest eigenvalue of large matrices. If additional eigenvalues are needed, deflation techniques can be applied, but this increases computational cost.

### Jacobi Method

The Jacobi method is an iterative algorithm used to compute the eigenvalues of a symmetric matrix. The method diagonalizes the matrix by applying a series of rotations to eliminate off-diagonal elements, eventually converging to a diagonal matrix where the diagonal elements are the eigenvalues.
Begin with matrix $A \in \mathbb{R}^{n \times n}$. At each step, identify the largest off-diagonal element $A_{ij}$ of matrix $A$. Compute a rotation matrix $R$ that will zero out the off-diagonal element $A_{ij}$. Apply the rotation matrix $R$to the matrix $A$, updating both $A$ and the eigenvectors associated with $A$. Repeat until the off-diagonal elements of the matrix are sufficiently small. The diagonal elements of the resulting matrix are the required eigenvalues.
Finding the largest off-diagonal element takes $O(n^2)$ time. Each rotation matrix is computed in constant time, but applying the rotation to the matrix $A$ involves $O(n^2)$ operations. In the worst case, the Jacobi method requires $O(n^3)$ iterations for convergence. The overall time complexity is $O(n^4)$.
The Jacobi method may converge slowly for large matrices due to $O(n^4)$ time complexity. Jacobi method is suited only for symmetric matrices.

## My Code

I am using C to implement QR algorithm to calculate eigenvalues because its basic and effective at the same time. Also is efficient for moderately large matrices. Unlike other algorithm it is easy to implement and gives all eigenvalues.

### QR Algorithm

The QR algorithm is an iterative method used to find the eigenvalues of a matrix. The method is based on the factorization of a matrix into a product of an orthogonal matrix $Q$ and an upper triangular matrix $R$, and is a powerful method to calculate eigenvalues. Over successive iterations, the QR algorithm converges to a diagonal matrix whose diagonal elements are the eigenvalues of

the original matrix.
Start with a square matrix $A \in \mathbb{R}^{n \times n}$. Find matrices $Q$(orthogonal) and $R$(upper triangular) such that:

$$A = QR$$

Update the matrix $A$ using the product $A_{new} = RQ$. The matrix $A_{new}$ is similar to the original matrix $A$, meaning they have the same eigenvalues. Repeat until the matrix $A$ converges to a diagonal form, where the diagonal elements are the original matrix.
The QR decomposition of an $n \times n$ matrix takes $O(n^3)$ operations.the number of iterations needed for convergence depends on the matrix. In general, the QR algorithm converges ii $O(n^2)$ iterations for most matrices. the overall time complexity for the QR algorithm os $O(n^5)$.
I chose QR algorithm because for matrices with distinct eigenvalues, it typically converges relatively quickly, especially with the use of shifts. However, matrices with repeated eigenvalues or ill-conditioned matrices may require more iterations to converge. Computationally expensive with a time complexity of $O(n^5)$, making it less practical for very large matrices. Convergence can be slower for matrices with repeated eigenvalues or matrices that are nearly defective.