Eigenvalue Algorithms: Analysis

Dwarak A
EE24BTECH11019

November 18, 2024

Abstract

This report investigates various algorithms for computing eigenvalues, analyzing their theoretical foundations and computational complexity. The primary focus is on iterative methods, matrix factorization techniques, and their respective trade-offs in terms of accuracy and efficiency.

Contents

1 Introduction **3**
1.1 Objective . . . 3
1.2 Motivation . . . 3

2 Theory **3**
2.1 Eigenvalues and Eigenvectors . . . 3
2.2 Special Matrices . . . 4
2.2.1 Unitary Matrices: . . . 4
2.2.2 Orthogonal Matrices: . . . 4
2.2.3 Hermitian Matrices: . . . 4
2.2.4 Skew-Hermitian Matrices: . . . 4
2.2.5 Positive Definite Matrices: . . . 4
2.3 Similarity Transformations . . . 4
2.3.1 Definition . . . 4
2.3.2 Properties . . . 5
2.3.3 Applications . . . 5
2.4 QR Decomposition . . . 5
2.5 Hessenberg Matrix . . . 5
2.5.1 Hessenberg Form Structure . . . 6
2.5.2 Time Complexity Reduction in the QR Algorithm . . . 6
2.5.3 Hessenberg Matrix . . . 6
2.5.4 Why Reduce to Hessenberg Form ? . . . 7

3 Implemented Eigenvalue Algorithm **7**
3.1 Hessenberg Reduction And QR Decomposition - Given's Rotations . . . 7
3.1.1 Givens Rotations in Eigenvalue Computations . . . 7
3.1.2 The Role of Givens Rotations in Eigenvalue Algorithms . . . 7
3.1.3 Mathematical Description of Givens Rotations . . . 8

3.1.4 Hessenberg Reduction Using Givens Rotations (Complex Values) 8
3.1.5 Complex QR Decomposition Using Givens Rotations 9
3.1.6 Efficiency and Stability for Complex Matrices 9
3.1.7 Significance in Eigenvalue Computations 9
3.2 Wilkinson Shift 10
3.2.1 Purpose of Wilkinson Shift 10
3.2.2 Mathematical Explanation 10
3.3 Main QR Algorithm - With Wilkinson Shifts 10

4 Other algorithms **11**
4.1 Hessenberg Reduction 11
4.1.1 Householder Transformations 11
4.1.2 Comparison with Givens Rotations 11
4.2 QR Decomposition 11
4.2.1 Householder Transformations 11
4.2.2 Gram-Schmidt 12
4.2.3 Divide and Conquer 13
4.2.4 Comparison with Givens Rotation 13
4.3 Shifts 13
4.3.1 Rayleigh Shift 13
4.3.2 Comparison with Wilkinson Shift 14

5 Code Implementation **14**

A References **14**

1 Introduction

Eigenvalues and eigenvectors are fundamental concepts in linear algebra, representing intrinsic properties of matrices and their associated transformations. They play a critical role in a wide range of scientific and engineering applications, from solving differential equations to analyzing complex systems in quantum mechanics, structural dynamics, and data science.

1.1 Objective

The objectives of this project are:

- To study and implement general eigenvalue computation algorithms.
- To analyze the time complexity and computational trade-offs of these algorithms.

1.2 Motivation

Eigenvalues are fundamental in understanding matrix behavior and have broad scientific and computational relevance:

- **Mathematical Importance:** Core to spectral theory and linear transformations, revealing intrinsic matrix properties.
- **Applications in Science and Engineering:** Crucial in vibrational analysis, system stability, and quantum mechanics.
- **Data Science and Machine Learning:** Drive techniques like Principal Component Analysis (PCA) and graph-based analytics.
- **Optimization and Numerical Solutions:** Applied in quadratic programming, differential equations, and matrix decomposition.
- **Interdisciplinary Relevance:** Integral to physics, economics, biology, and engineering for modeling and simulations.
- **Algorithmic Challenges:** Drive innovations in computational efficiency and large-scale matrix handling.

2 Theory

2.1 Eigenvalues and Eigenvectors

A matrix can be interpreted as a linear transformation that maps vectors in a vector space to other vectors within the same or different vector space. There exist specific vectors, known as *eigenvectors*, which, when transformed by the matrix, result in a vector that is a scalar multiple of the original vector. For a matrix $A \in \mathbb{C}^{n\times n}$, a vector $\mathbf{v} \in \mathbb{C}^n$ $(\mathbf{v} \neq \mathbf{0})$ is an eigenvector if there exists a scalar λ (called the *eigenvalue*) such that:

$$A\mathbf{v} = \lambda\mathbf{v} \tag{2.1.0.1}$$

Here, $\mathbf{v}$ retains (or reverses) its direction under the transformation, and λ represents the scaling factor.

2.2 Special Matrices

2.2.1 Unitary Matrices:

A matrix U is *unitary* if:

$$U^\dagger U = UU^\dagger = \mathbf{I} \tag{2.2.1.1}$$

where $U^\dagger$ is the conjugate transpose of U and $\mathbf{I}$ is the identity matrix.

2.2.2 Orthogonal Matrices:

A matrix Q is *orthogonal* if:

$$Q^T Q = QQ^T = \mathbf{I} \tag{2.2.2.1}$$

where Q^T is the transpose of Q. Orthogonal matrices preserve the dot product.

2.2.3 Hermitian Matrices:

A matrix A is *Hermitian* if:

$$A^\dagger = A \tag{2.2.3.1}$$

where $A^\dagger$ is the conjugate transpose of A.

2.2.4 Skew-Hermitian Matrices:

A matrix A is *skew-Hermitian* if:

$$A^\dagger = -A \tag{2.2.4.1}$$

2.2.5 Positive Definite Matrices:

A matrix A is *positive definite* if:

$$\mathbf{x}^\top A\mathbf{x} > 0 \tag{2.2.5.1}$$

for all non-zero vectors $\mathbf{x}$.

2.3 Similarity Transformations

In linear algebra, similarity transformations play a crucial role in simplifying matrices while preserving their fundamental properties, such as eigenvalues.

2.3.1 Definition

A square matrix $A \in \mathbb{R}^{n\times n}$ is said to be *similar* to another square matrix $B \in \mathbb{R}^{n\times n}$ if there exists an invertible matrix $P \in \mathbb{R}^{n\times n}$ such that

$$B = P^{-1}AP \tag{2.3.1.1}$$

The process of finding B via this transformation is called a **similarity transformation**.

2.3.2 Properties

1. Similar matrices have the same eigenvalues.

$$\begin{aligned} \left|P^{-1}AP - \lambda I\right| &= \left|P^{-1}AP - \lambda P^{-1}IP\right| && (2.3.2.1)\\ &= \left|P^{-1}(A - \lambda I)P\right| && (2.3.2.2)\\ &= \left|P^{-1}\right| \cdot \left|A - \lambda I\right| \cdot \left|P\right| && (2.3.2.3)\\ &= \left|A - \lambda I\right| && (2.3.2.4) \end{aligned}$$

   The characteristic polynomial of similar matrices is identical:

2. Similar matrices share the same trace and determinant:

$$\operatorname{tr}(A) = \operatorname{tr}(B), \quad \left|A\right| = \left|B\right|. \tag{2.3.2.5}$$

3. If A and B are similar, then they represent the same linear transformation under a change of basis.

2.3.3 Applications

Similarity transformations are particularly useful in diagonalizing matrices. A matrix A is diagonalizable if it is similar to a diagonal matrix D, where

$$D = P^{-1}AP \tag{2.3.3.1}$$

and the columns of P are the eigenvectors of A, while the diagonal entries of D are the eigenvalues of A.

2.4 QR Decomposition

QR decomposition is an essential algorithm in numerical linear algebra, where a matrix $A \in \mathbb{C}^{n\times n}$ is decomposed into:

$$A = QR, \tag{2.4.0.1}$$

where:

- $Q \in \mathbb{C}^{n\times n}$ is a unitary matrix ($Q^\dagger Q = I$), and
- $R \in \mathbb{C}^{n\times n}$ is an upper triangular matrix.

2.5 Hessenberg Matrix

The Hessenberg form simplifies eigenvalue computations by leveraging its special structure, reducing the computational effort required in iterative algorithms like the QR algorithm.

2.5.1 Hessenberg Form Structure

A Hessenberg matrix is a nearly triangular matrix:

- **Upper Hessenberg**: All elements below the first subdiagonal are zero.

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ 0 & a_{32} & a_{33} & \cdots & a_{3n} \\ 0 & 0 & a_{43} & \cdots & a_{4n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & a_{nn} \end{pmatrix} \tag{2.5.1.1}$$

- **Lower Hessenberg**: All elements above the first superdiagonal are zero.

$$\begin{pmatrix} a_{11} & a_{12} & 0 & \cdots & 0 \\ a_{21} & a_{22} & a_{23} & \cdots & 0 \\ a_{31} & a_{32} & a_{33} & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \cdots & a_{nn} \end{pmatrix} \tag{2.5.1.2}$$

2.5.2 Time Complexity Reduction in the QR Algorithm

The QR algorithm involves two main steps per iteration:

1. **QR Factorization**: Decompose H as QR, where Q is orthogonal, and R is upper triangular.
2. **Matrix Multiplication**: Update H as $H \leftarrow RQ$.

General Dense Matrix

For a general dense matrix:

- **QR Factorization**: Requires $O(n^3)$ operations.
- **Matrix Multiplication**: Also requires $O(n^3)$ operations.
- **Total per iteration**: $O(n^3)$.

2.5.3 Hessenberg Matrix

For a Hessenberg matrix:

- **QR Factorization**: Exploits the sparsity of H, requiring only $O(n^2)$ operations.
- **Matrix Multiplication**: Similarly benefits from sparsity, requiring $O(n^2)$ operations.
- **Total per iteration**: $O(n^2)$.

Thus, reducing a matrix to Hessenberg form results in significant computational savings for iterative processes. So for this project, the matrix is first transformed into the Upper Hessenberg form.

2.5.4 Why Reduce to Hessenberg Form ?

Transforming a general matrix to Hessenberg form:

- **One-time cost**: Requires $O(n^3)$ operations, typically via Given's rotations or Householder transformations.
- After this reduction, the subsequent iterations benefit from the $O(n^2)$ complexity, making the QR algorithm efficient for large-scale problems.

A crucial property of the QR algorithm is that it preserves the Hessenberg structure. This means the computational advantages of the Hessenberg form apply to every iteration.

By reducing the per-iteration complexity from $O(n^3)$ to $O(n^2)$, Hessenberg form enables scalable eigenvalue computations, which are essential for solving large problems in scientific computing and engineering.

3 Implemented Eigenvalue Algorithm

3.1 Hessenberg Reduction And QR Decomposition - Given's Rotations

3.1.1 Givens Rotations in Eigenvalue Computations

Givens rotations are an essential tool in numerical linear algebra, particularly in algorithms for matrix factorizations and eigenvalue computations. They provide a way to introduce zeros into specific positions of a matrix while preserving its orthogonality or structure. This technique is frequently employed in QR decomposition, which is a cornerstone of iterative eigenvalue methods.

3.1.2 The Role of Givens Rotations in Eigenvalue Algorithms

In eigenvalue computations, particularly when using the **QR algorithm**, the goal is to iteratively transform a matrix A into a form where the eigenvalues can be directly inferred from its diagonal entries. The QR algorithm works by repeatedly factorizing the matrix into the product of an orthogonal matrix Q and an upper triangular matrix R, followed by recomputing A as $A = RQ$.

- **Hessenberg Reduction**: Before applying the QR algorithm, the matrix A is often reduced to Hessenberg form (upper or lower). This step simplifies the computation by ensuring that only a small number of entries need to be zeroed out during the QR iterations. Givens rotations are particularly effective for this step because they can introduce zeros below the first subdiagonal in a row-by-row manner.
- **QR Iterations**: During each iteration of the QR algorithm, Givens rotations can be used to apply orthogonal transformations, either to compute Q and R or to refine the matrix further. For example:
  - When combined with **Wilkinson shifts**, Givens rotations help to maintain numerical stability while accelerating convergence.
  - They can selectively target specific entries to ensure numerical precision, especially when the matrix has nearly degenerate eigenvalues.

3.1.3 Mathematical Description of Givens Rotations

A Givens rotation matrix $G(i, k, \theta)$ is an orthogonal matrix that introduces a zero at the (i, k)-th position of a matrix or vector. It is defined as:

$$G(i,k,c,s) = \begin{pmatrix} 1 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & & \vdots & & \vdots \\ 0 & \cdots & c & \cdots & s & \cdots & 0 \\ \vdots & & \vdots & \ddots & \vdots & & \vdots \\ 0 & \cdots & -\overline{s} & \cdots & \overline{c} & \cdots & 0 \\ \vdots & & \vdots & & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & 0 & \cdots & 1 \end{pmatrix} \tag{3.1.3.1}$$

where c and s are complex values satisfying:

$$c\overline{c} + s\overline{s} = 1. \tag{3.1.3.2}$$

Here, $\overline{s}$ represents the complex conjugate of s, ensuring the rotation matrix G is unitary.

The Givens rotation G is applied to matrix A as $G^\top A$ (row transformation) or AG (column transformation).

3.1.4 Hessenberg Reduction Using Givens Rotations (Complex Values)

1. **Initialization**: Begin with a complex matrix A of size $n \times n$.
2. **Column Elimination**: For each column $k = 1, 2, \ldots, n-2$, apply the following steps:
   (a) Identify the elements $a_{i,k}$ for $i = k+2, k+3, \ldots, n$ that need to be zeroed.
   (b) For each such element $a_{i,k}$, compute a Givens rotation $G(i, k+1)$ that eliminates $a_{i,k}$.
3. **Applying the Givens Rotation**:
   The Given's Rotation $G(i, k+1, c, s)$ as previously defined is applied.
   To eliminate $a_{i,k}$, set:
   $$r = \sqrt{|a_{k+1,k}|^2 + |a_{i,k}|^2}, \quad c = \frac{a_{k+1,k}}{r}, \quad s = \frac{-a_{i,k}}{r}. \tag{3.1.4.1}$$
4. **Matrix Update**: Update the matrix A by applying the Givens rotation:
   $$A' = G^H A G, \tag{3.1.4.2}$$
   where G^H is the Hermitian transpose of G. This step ensures that the transformations are unitary, preserving the eigenvalues of the matrix.
5. **Repeat**: Continue this process for all columns $k = 1, 2, \ldots, n-2$ to achieve the Hessenberg form.

3.1.5 Complex QR Decomposition Using Givens Rotations

1. **Input**: A Hessenberg matrix $H \in \mathbb{C}^{n \times n}$.
2. **Initialization**:
   - Set $Q = I_n$ (the identity matrix).
   - Copy H into R for in-place updates.
3. **Process Each Sub-Diagonal Element**:
   (a) For each $i = 1, 2, \ldots, n-1$, compute the Givens rotation $G_{i,i+1}$ to zero out $h_{i+1,i}$.
   (b) Calculate rotation parameters c and s:

$$c = \frac{h_{ii}}{\sqrt{|h_{ii}|^2 + |h_{i+1,i}|^2}}, \quad s = \frac{h_{i+1,i}}{\sqrt{|h_{ii}|^2 + |h_{i+1,i}|^2}}. \tag{3.1.5.1}$$

   (c) Apply $G_{i,i+1}$ from the left to R:

$$R \leftarrow G_{i,i+1}^* R. \tag{3.1.5.2}$$

   (d) Apply $G_{i,i+1}$ from the right to Q:

$$Q \leftarrow Q G_{i,i+1}. \tag{3.1.5.3}$$

4. **Output**:
   - Q: A unitary matrix.
   - R: An upper triangular matrix such that $H = QR$.

The eigenvalues are approximated iteratively as the diagonal entries of A.

3.1.6 Efficiency and Stability for Complex Matrices

Givens rotations in the complex domain maintain numerical stability due to their unitary nature. They are especially effective in preserving the norm of complex vectors during transformations. This property is crucial for accurate eigenvalue computation, particularly when dealing with matrices with complex-valued entries.

3.1.7 Significance in Eigenvalue Computations

For complex matrices, Givens rotations provide:

1. **Numerical Stability**: The unitary nature of complex Givens rotations ensures stability in iterative algorithms.
2. **Precision with Shifts**: When combined with complex shifts (e.g., Wilkinson shifts), they enhance the accuracy of eigenvalue isolation.
3. **Efficiency**: The sparsity of Hessenberg matrices minimizes computational overhead, making Givens rotations ideal for large-scale eigenvalue problems.

3.2 Wilkinson Shift

3.2.1 Purpose of Wilkinson Shift

The Wilkinson shift is used to select a scalar shift σ that helps isolate the eigenvalue from the rest of the matrix during the QR iteration.

The idea is to use the bottom-right 2×2 submatrix (block) of the Hessenberg matrix to compute the shift, making it easier to extract the eigenvalues of the matrix. By applying this shift, the matrix's eigenvalues are pushed apart, improving the accuracy of the QR iteration.

3.2.2 Mathematical Explanation

Given a Hessenberg matrix $H \in \mathbb{C}^{n\times n}$, the Wilkinson shift σ is computed using the bottom-right 2-by-2 block of the matrix:

$$H = \begin{pmatrix} h_{n-1,n-1} & h_{n-1,n} \\ h_{n,n-1} & h_{n,n} \end{pmatrix} \tag{3.2.2.1}$$

The Wilkinson shift σ is defined as:

$$\sigma = \frac{h_{n,n} + h_{n-1,n-1}}{2} - \frac{\operatorname{sgn}(h_{n,n} - h_{n-1,n-1})}{2}\sqrt{(h_{n,n} - h_{n-1,n-1})^2 + 4|h_{n,n-1}|^2} \tag{3.2.2.2}$$

The Wilkinson shift σ is computed to be close to the larger of the two eigenvalues of the bottom-right 2-by-2 block of H. The formula involves two main components:

- The term $\frac{h_{n,n}+h_{n-1,n-1}}{2}$ represents the average of the diagonal elements, approximating the center of the eigenvalue pair.
- The square root term $\sqrt{(h_{n,n} - h_{n-1,n-1})^2 + 4|h_{n,n-1}|^2}$ measures the distance between the two eigenvalues, considering both the diagonal and off-diagonal elements. The factor $\operatorname{sgn}(h_{n,n} - h_{n-1,n-1})$ ensures that the larger eigenvalue is chosen as the shift.

3.3 Main QR Algorithm - With Wilkinson Shifts

In the QR algorithm, the Wilkinson shift is applied as follows:

1. Compute the shift σ.
2. Subtract σ from the hessenberg matrix H to create a shifted matrix $H_{\text{shifted}} = H - \sigma I$, where I is the identity matrix.
3. Perform a QR decomposition on the shifted matrix: $H_{\text{shifted}} = QR$.
4. Form the new matrix $H_{\text{new}} = RQ$, and repeat the QR process with the new matrix.
5. The QR iterations continue until the off-diagonal elements become sufficiently small, at which point the diagonal elements are the eigenvalues of the original matrix H.

The Wilkinson shift helps to accelerate the QR iteration process, particularly when the matrix has nearly degenerate eigenvalues, and improves the overall convergence.

4 Other algorithms

4.1 Hessenberg Reduction

In addition to the method of Givens rotations, the Hessenberg Reduction process can also be performed using other matrix factorization techniques, such as Householder transformations.

4.1.1 Householder Transformations

Householder transformations are widely used for orthogonal reduction of a matrix to a Hessenberg form. The basic idea is to use an orthogonal matrix Q to zero out the subdiagonal elements of the matrix H. This is done by reflecting the columns of H onto a vector that is parallel to the first column.

The Householder transformation for the i-th column of a matrix A is given by:

$$H_i = I - 2\frac{vv^T}{v^Tv} \tag{4.1.1.1}$$

where v is a vector chosen to zero out the subdiagonal elements of the i-th column of A, and I is the identity matrix of the same size. The resulting matrix H_i is an orthogonal matrix, meaning that $H_i^T H_i = I$.

To transform a matrix $A \in \mathbb{C}^{n\times n}$ to Hessenberg form, the Householder transformation is applied iteratively to each column, from left to right. After applying the transformations, the matrix is reduced to upper Hessenberg form H, where all elements below the first subdiagonal are zero.

4.1.2 Comparison with Givens Rotations

Both Householder and Givens methods are used to reduce a matrix to Hessenberg form, but they differ in their approach:

- **Householder Transformations**: These apply orthogonal reflections to eliminate subdiagonal elements in one step for each column. This method is more efficient for large, dense matrices.
- **Givens Rotations**: These use rotations to zero out individual elements and can be more efficient for sparse matrices. This algorithm can be parallelized.

4.2 QR Decomposition

4.2.1 Householder Transformations

After applying the Householder transformations iteratively, the QR decomposition of the Hessenberg matrix H is given by:

$$Q = H_1 H_2 \cdots H_{n-1} \tag{4.2.1.1}$$

where Q is an orthogonal matrix and H_k are the Householder transformations.

The upper triangular matrix R is obtained by applying the Householder transformations to the original matrix H:

$$R = H_{n-1}H_{n-2}\cdots H_1 H \tag{4.2.1.2}$$

and H_i is as previously defined.

4.2.2 Gram-Schmidt

Given a Hessenberg matrix $H \in \mathbb{C}^{n\times n}$, the Gram-Schmidt process can be used to find the QR decomposition $H = QR$, where Q is an orthogonal matrix and R is an upper triangular matrix.

1. **Input**: A Hessenberg matrix $H \in \mathbb{C}^{n\times n}$.
2. **Initialization**:
   - Let $Q = I_n$ (the identity matrix).
   - Let $R = 0$ (to store the upper triangular matrix).
3. **Orthogonalization Process**:
   (a) For $i = 1, 2, \ldots, n$, set the i-th column of H as a_i.
   $$a_i = H[:, i] \tag{4.2.2.1}$$
   (b) For $j = 1, 2, \ldots, i-1$, compute the projection of a_i onto $Q[:, j]$:
   $$r_{ij} = Q[:, j]^T a_i \tag{4.2.2.2}$$
   (c) Subtract the projection to make a_i orthogonal to the previous vectors:
   $$a_i \leftarrow a_i - r_{ij}Q[:, j] \quad \text{for each} \quad j < i \tag{4.2.2.3}$$
   (d) Compute the norm of a_i:
   $$r_{ii} = \|a_i\| \tag{4.2.2.4}$$
   (e) Normalize a_i to create the orthogonal vector q_i:
   $$q_i = \frac{a_i}{r_{ii}} \tag{4.2.2.5}$$
   (f) Store the orthogonal vector in Q:
   $$Q[:, i] = q_i \tag{4.2.2.6}$$
   (g) Store the upper triangular values in R:
   $$R_{ij} = r_{ij} \quad \text{for all} \quad j = 1, 2, \ldots, i \tag{4.2.2.7}$$
4. **Output**:
   - Q, the orthogonal matrix.
   - R, the upper triangular matrix.

4.2.3 Divide and Conquer

Given a Hessenberg matrix $H \in \mathbb{C}^{n\times n}$, the goal is to find matrices Q (unitary) and R (upper triangular) such that $H = QR$. The DC algorithm proceeds as follows:

1. **Split the Matrix:** Recursively divide the Hessenberg matrix H into two smaller sub-matrices. At each step, the matrix is split into two blocks, and the QR decomposition is computed on each block:

$$H = \begin{bmatrix} H_{11} & H_{12} \\ H_{21} & H_{22} \end{bmatrix} \tag{4.2.3.1}$$

   where H_{11} is a block of size $m \times m$, H_{22} is a block of size $(n-m) \times (n-m)$, and H_{12}, H_{21} are the off-diagonal blocks of appropriate sizes.

2. **Recursive QR Decomposition:** Perform QR decomposition on the smaller sub-matrices H_{11} and H_{22}:

$$H_{11} = Q_1 R_1, \quad H_{22} = Q_2 R_2 \tag{4.2.3.2}$$

   where Q_1, Q_2 are unitary matrices and R_1, R_2 are upper triangular matrices.

3. **Combine Results:** Once the QR decompositions of the smaller blocks are computed, the results are combined to form the QR decomposition of the entire matrix. The combined decomposition has the form:

$$H = Q_1 \begin{bmatrix} R_1 & R_1^{-1} H_{12} \\ 0 & R_2 \end{bmatrix} Q_2^T \tag{4.2.3.3}$$

   where the product of Q_1 and Q_2 forms the unitary matrix Q, and the matrix R is the upper triangular part of the combined result.

4. **Repeat Until Base Case:** The algorithm continues recursively until the sub-matrices are reduced to small enough sizes, at which point the base case can be directly computed using standard QR methods.

4.2.4 Comparison with Givens Rotation

Method	Time Complexity	Numerical Stability	Memory Usage
Householder	$O(n^3) \to O(n^2)$	High	Moderate
Divide and Conquer	$O(n^3) \to O(n^2 \log n)$	Moderate	Low
Givens Rotation	$O(n^3) \to O(kn^2)$	Very High	Low
Gram-Schmidt	$O(n^3)$	Low	Low

4.3 Shifts

4.3.1 Rayleigh Shift

The Rayleigh shift is a technique used to improve the convergence of the QR algorithm for eigenvalue computation. It involves shifting the matrix by a scalar value σ that approximates the eigenvalue of the matrix. This shift accelerates the convergence of the algorithm.

The Rayleigh quotient for a matrix $A \in \mathbb{C}^{n \times n}$ and a vector $v \in \mathbb{C}^n$ is defined as:

$$\rho(A, v) = \frac{v^H A v}{v^H v} \tag{4.3.1.1}$$

where v^H denotes the Hermitian (conjugate transpose) of v. For the QR algorithm, the Rayleigh shift σ is often chosen as the diagonal element of the matrix A. Specifically, σ can be taken as the element A_{nn}:

$$\sigma = A_{nn} \tag{4.3.1.2}$$

4.3.2 Comparison with Wilkinson Shift

- **Wilkinson shift**: generally requires fewer iterations, particularly in cases where eigenvalues are close or the matrix is ill-conditioned.
- The **Rayleigh shift**, while computationally simpler, may require more iterations when eigenvalues are close together or in ill-conditioned matrices.

5 Code Implementation

The Python code can be found at `https://github.com/Dwarak-A/EE1030/blob/main/Eigenvalues/code/eigen.py`

A References

1. Golub, G. H., and Van Loan, C. F. *Matrix Computations*. Johns Hopkins University Press.
2. Wikipedia contributors. *Eigenvalue Algorithm*. Wikipedia. Available at: `https://en.wikipedia.org/wiki/Eigenvalue_algorithm`
3. Arbenz, Peter. *Lecture Notes on Numerical Methods for Eigenvalue Problems, Chapter 4*. ETH Zurich, 2018. Available at: `https://people.inf.ethz.ch/arbenz/ewp/Lnotes/chapter4.pdf`