Eigenvalue Computation of Complex Matrices

Agamjot Singh
EE24BTECH11002
IIT Hyderabad

November 18, 2024

Abstract

This report presents an efficient algorithm for finding the eigenvalues of matrices with complex entries using a combination of Householder transformations and QR decomposition (via Givens rotation algorithm). The implementation is in C, and we discuss the theory and the approach taken.

1 Introduction

Finding eigenvalues of complex matrices is a critical task in various scientific and engineering applications. This report explores an efficient approach using a combination of:

- **Householder Transformations:** to reduce the matrix to Hessenberg form.
- **Givens Rotations:** for QR decomposition of the Hessenberg matrix.

2 Theory

We start off by defining some basic concepts that we will be using throughout:

2.1 Eigenvectors and Eigenvalues

When we multiply a vector $\mathbf{v} \in \mathbb{C}^m$, $\mathbf{v} \neq \mathbf{0}$ by a matrix $A \in \mathbb{C}^{m \times m}$ we get a resultant vector $\mathbf{x} = A\mathbf{v}$, $\mathbf{x} \in \mathbb{C}^m$. This transformation rotates, stretches, or shears the vector $\mathbf{v}$. Now we choose the vector $\mathbf{v}$ such that this linear transformation only stretches, with no rotation or shear. This chosen vector is the **eigenvector** of the matrix A. The corresponding **eigenvalue** is defined as the factor by which the eigenvector has been stretched.
Mathematically speaking,

$$A\mathbf{v} = \lambda\mathbf{v} \tag{2.1}$$

where $\mathbf{v}$ is the eigenvector and $\lambda \in \mathbb{C}$ is the corresponding eigenvalue.

Equation (2.1) can also be stated as,

$$(A - \lambda I)\,\mathbf{v} = \mathbf{0} \tag{2.2}$$

where $I \in \mathbb{C}^{m\times m}$ is the identity matrix of order m.

Equation (2.2) has non zero solution $\mathbf{v}$ if and only if,

$$\det(A - \lambda I) = 0 \tag{2.3}$$

2.2 Similarity Transformation

A similarity transformation on a matrix $A \in \mathbb{C}^{m\times m}$ with transformation matrix $A' \in \mathbb{C}^{m\times m}$ is given by,

$$A' = X^* A X \tag{2.4}$$

where X is a unitary matrix and X^* is the conjugate transpose of X. Note that, for X to be unitary, it satisfies,

$$X^*X = XX^* = I \tag{2.5}$$

We will now show that a similarity transformation preserves the eigenvalues of any matrix.
Say $\lambda' \in \mathbb{C}$ is an eigenvalue of the matrix A', hence by equation (2.3) and (2.5),

$$\det(A' - \lambda' I) = 0 \tag{2.6}$$
$$\det(X^*AX - \lambda' X^*X) = 0 \tag{2.7}$$
$$\det(X^*(A - \lambda' I)X) = 0 \tag{2.8}$$
$$\implies \det(X^*)\det(A - \lambda' I)\det(X) = 0 \tag{2.9}$$
$$\tag{2.10}$$

From equation (2.5), X is invertible with $X^{-1} = X^*$. Hence $\det(X) = \frac{1}{\det(X^*)} \neq 0$.
Using this fact in equation (2.9), we get,

$$\det(A - \lambda' I) = 0 \tag{2.11}$$

This proves that eigenvalues of A' are exactly same as the eigenvalues of A. Similarily, it can also be proved that eigenvalues of A are same as that of A'.
Similarily transformations are the basic building block of computing eigenvalues efficiently which will be extensively used in this implementation.

2.3 QR decomposition

QR decomposition is a decomposition of a matrix $A \in \mathbb{C}^{m\times m}$ into a product $A = QR$, such that $Q \in \mathbb{C}^{m\times m}$ is an square unitary matrix with all of its columns having unit norm and $R \in \mathbb{C}^{m\times m}$ is an upper triangular matrix.

Geometrically speaking, the matrix A can be thought of as a set of vectors (columns of A) in the m-dimensional space. The matrix Q represents an orthonormal basis for the column space of A. It "replaces" the original vectors of A with a new set of orthonormal vectors. The matrix R provides the coefficients needed to express the original vectors of A as linear combinations of the new orthonormal vectors in Q.

3 Pros and Cons of Householder transformation and Givens rotation

3.1 Time complexity

- Householder reflections: $O(n^3)$
- Givens rotations for QR decomposition: $O(n^3)$ for dense matrices, $O(n)$ per rotation (efficient for sparse matrices).
- Total: $O(n^3)$

3.2 Numerical Stability

- High stability due to orthogonal transformations.

3.3 Applicability

- Suitable for dense matrices.
- Can handle both real and complex matrices effectively.

3.4 Memory Usage

- Moderate, requires storing Householder vectors and applying Givens rotations in place.

Algorithm	**Pros**	**Cons**	**Time Complexity**	**Conditions**
Householder + Givens	Stable, precise; preserves sparsity.	Complex preprocessing for Hessenberg form.	$O(n^3)$	Any square matrix.
Jacobi Method	Accurate for symmetric matrices; simple.	Slow for non-symmetric matrices; inefficient for large ones.	$O(n^3)$ (dense)	Symmetric (real/complex).
Lanczos Algorithm	Efficient for sparse symmetric matrices.	Sensitive to rounding; limited to Hermitian matrices.	$O(kn^2)$	Sparse, symmetric/Hermitian.
QR Algorithm	Robust for eigenvalue computation.	High cost for large dense matrices.	$O(n^3)$	Any square matrix.
Arnoldi Algorithm	Handles non-symmetric sparse matrices.	Requires reorthogonalization; less stable.	$O(kn^2)$	Sparse, any matrix.
Power Iteration	Simple, finds dominant eigenvalue.	Only computes the largest eigenvalue; slow for close eigenvalues.	$O(kn^2)$	Any square matrix.

3.5 Why did we choose this particular algorithm?

- Stability: It offers high numerical stability, making it suitable for dense, symmetric matrices.
- Efficient Hessenberg Reduction: Householder transforms reduce the matrix to Hessenberg form, simplifying eigenvalue calculations.
- Orthogonality: Maintains orthogonality during transformations, preserving accuracy.
- Full Spectrum Calculation: Efficient for computing all eigenvalues, especially in iterative methods like QR iteration.
- Numerically Robust: Handles ill-conditioned matrices better than simpler methods like Power Iteration.

4 Algorithm Overview

First of all, why not just take the determinant of the matrix and find the roots of $\det(A - \lambda I) = 0$ **?**
Well, it's $O(m!)$, and that speaks for itself. It will blow up with extreme rates with increasing m.
Therefore, we employ a multi-step approach to find the eigenvalues of a complex matrix $A \in \mathbb{C}^{m \times m}$:

4.1 The Basic QR Algorithm - Schur Decomposition

In this algorithm, a sequence of matrices $\{A_k\}$, $A_k \in \mathbb{C}^{m \times m}$ is generated iteratively, converging towards an upper triangular matrix. Under specific conditions, the convergence of off-diagonal elements follows a rate based on the ratio of eigenvalues.

As discussed in section 2.3, any matrix A_k can be expressed as,

$$A_k = Q_k R_k \tag{4.1}$$

where $Q_k^* Q_k = I$.
Now we take A_{k+1} such that,

$$A_{k+1} = R_k Q_k = Q_k^* A_k Q_k \tag{4.2}$$

We can observe that the above transformation is a similarity transformation. Hence the eigenvalues of any matrix in the sequence $\{A_k\}$ are identical.
The matrix sequence $\{A_k\}$ converges (although very slowly in its current form) towards an upper triangular matrix, with its eigenvalues as diagonal elements. This algorithm computes an upper triangular matrix T and a unitary matrix U such that $A = UTU^*$ is the Schur decomposition of A.

$$A_k = Q_k^* A_{k-1} Q_k = Q_k^* Q_{k-1}^* A_{k-2} Q_{k-1} Q_k = \cdots = Q_k^* \dots Q_1^* A Q_1 \dots Q_k \tag{4.3}$$

We can observe that $U_k = Q_1 Q_2 Q_3 \dots Q_k$, and $U = U_\infty$. Also note that $T = A_\infty$.

Let the eigen values of A be λ_1, λ_2, $\lambda_3 \dots \lambda_m$, such that

$$|\lambda_1| \geq |\lambda_2| \geq |\lambda_3| \cdots \geq |\lambda_m| \tag{4.4}$$

We state, without proof, that the subdiagonal elements of A converge like

$$\left|a_{ij}^{(k)}\right| = O\left(\left|\frac{\lambda_i}{\lambda_j}\right|^k\right), i > j \tag{4.5}$$

4.2 Householder Transformations to Hessenberg form

As stated before, the vanilla QR algorithm converges very slowly without any tweaking involved. One such tweak is reducing the matrix to a hessenberg matrix form using similarity transforms and then apply the QR algorithm. This is achieved by Householder Transformations on the original matrix to convert it into hessenberg form.
Note that a Hessenberg matrix is of the form,

$$H = \begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ 0 & 0 & \times & \times & \times \\ 0 & 0 & 0 & \times & \times \end{bmatrix} \tag{4.6}$$

4.2.1 Householder reflectors

A Householder reflector matrix is of the form

$$P = I - 2\mathbf{u}\mathbf{u}^* \tag{4.7}$$

where $P \in \mathbb{C}^{m \times m}$ and $\mathbf{u} \in \mathbb{C}^m$, $\|\mathbf{u}\| = 1$.

Note that Householder reflectors are Hermitian, i.e. $P^* = P$ and $P^2 = I$. As $P^*P = I$, we also come to the conclusion that P is a unitary matrix.
We want the Householder reflector to transform any vector $\mathbf{x} \in \mathbb{C}^m$ to a multiple of $\mathbf{e_1}$, where $\mathbf{e_n}$ is the impulse vector with $n = 1$.

$$P\mathbf{x} = \mathbf{x} - 2\mathbf{u}\left(\mathbf{u}^*\mathbf{x}\right) = \alpha\mathbf{e_1} \tag{4.8}$$

Since P is unitary, $\|P\mathbf{x}\| = \|\mathbf{x}\|$.
Hence by taking norm on both sides on equation (4.8), we get $|\alpha| = \|\mathbf{x}\|$. Therefore, $\alpha = \rho\|\mathbf{x}\|$, where $\rho \in \mathbb{C}$, $|\rho| = 1$.
By rearranging equation (4.8),

$$\mathbf{x} - \rho\|\mathbf{x}\|\mathbf{e_1} = 2\mathbf{u}\left(\mathbf{u}^*\mathbf{x}\right) \tag{4.9}$$

As $\mathbf{u}$ is unit norm,

$$\mathbf{u} = \frac{\mathbf{x} - \rho\|\mathbf{x}\|\mathbf{e_1}}{\|\mathbf{x} - \rho\|\mathbf{x}\|\mathbf{e_1}\|} = \frac{1}{\|\mathbf{x} - \rho\|\mathbf{x}\|\mathbf{e_1}\|}\begin{pmatrix} x_1 - \rho\|\mathbf{x}\| \\ x_2 \\ \vdots \\ x_n \end{pmatrix} \tag{4.10}$$

Selection of ρ is flexible as long as $|\rho| = 1$. To ease out the process, we take $\rho = \frac{x_1}{|x_1|}$, $x_1 \neq 0$. If $x_1 = 0$, we take $\rho = 1$.

4.2.2 Reduction to Hessenberg Form using Householder reflectors

Consider the initial matrix $A \in \mathbb{C}^{5\times 5}$ for the sake of explanation:

$$A = \begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \end{bmatrix} \tag{4.11}$$

Let P_1 have the structure,

$$P_1 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \end{bmatrix} = \begin{bmatrix} 1 & \mathbf{0}^* \\ \mathbf{0} & I_4 - 2\mathbf{u_1}\mathbf{u_1^*} \end{bmatrix} \tag{4.12}$$

and let P_2 have the structure

$$P_2 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & \times & \times & \times \\ 0 & 0 & \times & \times & \times \\ 0 & 0 & \times & \times & \times \end{bmatrix} = \begin{bmatrix} 1 & 0 & \mathbf{0}^* \\ 0 & 1 & \mathbf{0}^* \\ \mathbf{0} & \mathbf{0} & I_3 - 2\mathbf{u_2}\mathbf{u_2^*} \end{bmatrix} \tag{4.13}$$

Similarily P_3, $P_4 \ldots$ can also be found using relevant $\mathbf{u}$. The Householder vector $\mathbf{u_1}$ is found taking $\mathbf{x}$ in equation (4.10) as $\begin{pmatrix} a_{21} \\ a_{31} \\ a_{41} \\ a_{51} \end{pmatrix}$.
Similary $\mathbf{u_2}$, $\vec{u_3} \ldots$ can also be found taking elements below the diagonal of each column as $\mathbf{x}$.
The multiplication of P_1 from the left inserts the desired zeros in the first column of A. The multiplication from the right is necessary in order to have similarity and preserve eigen values. Because of the structure of P_1 the first column of $P_1 A$ is not affected. The reduction happens in the following way:

$$P_1 A P_1 = \begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \end{bmatrix} \tag{4.14}$$

$$P_2 \left(P_1 A P_1\right) P_2 = \begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ 0 & 0 & \times & \times & \times \\ 0 & 0 & \times & \times & \times \end{bmatrix} \tag{4.15}$$

$$P_3\left(P_2P_1AP_1P_2\right)P_3 = \begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ 0 & 0 & \times & \times & \times \\ 0 & 0 & 0 & \times & \times \end{bmatrix} = H \tag{4.16}$$

which H is the transformed matrix in the Hessenberg form.

4.3 QR Decomposition using Givens Rotations

Now that we have reached the hessenberg form, we have to perform the QR algorithm to converge it into an upper triangular matrix. For QR decomposition, i.e finding both Q and R such that $H = QR$, we use Givens Rotations to zero out subdiagonal elements to find upper triangular R, and then calculate $\tilde{H} = RQ$. Note that the QR algorithm preserves the Hessenberg form of the matrix on every iteration.

The Givens rotation matrix $G\left(i, j, c, s\right)$ is defined by

$$G\left(i, j, c, s\right) = \begin{bmatrix} 1 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & \cdots & c & \cdots & s & \cdots & 0 \\ \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & \cdots & -\overline{s} & \cdots & \overline{c} & \cdots & 0 \\ \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & 0 & \cdots & 1 \end{bmatrix} \tag{4.17}$$

where $c, s \in \mathbb{C}$ and $|c|^2 + |s|^2 = 1$. We can see that G is a unitary matrix.
Say we take a vector $\mathbf{x} \in \mathbb{C}^m$, and $\mathbf{y} = G\left(i, j, c, s\right)\mathbf{x}$, then

$$y_k = \begin{cases} cx_i + sx_j, & k = i \\ -\overline{s}x_i + \overline{c}x_j, & k = j \\ x_k, & k \neq i, j \end{cases} \tag{4.18}$$

For y_j to be zero, we set

$$c = \frac{\overline{x_i}}{\sqrt{|x_i|^2 + |x_j|^2}} = c_{ij} \tag{4.19}$$

$$s = \frac{\overline{x_j}}{\sqrt{|x_i|^2 + |x_j|^2}} = s_{ij} \tag{4.20}$$

Using this Givens rotation matrix, we zero out elements of subdiagonal in the hessenberg matrix H. The algorithm for that is given as follows:

$$H = \begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ 0 & 0 & \times & \times & \times \\ 0 & 0 & 0 & \times & \times \end{bmatrix} \xrightarrow{G(1,2,c_{12},s_{12})} \begin{bmatrix} \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ 0 & 0 & \times & \times & \times \\ 0 & 0 & 0 & \times & \times \end{bmatrix}$$

$$\xrightarrow{G(2,3,c_{23},s_{23})} \begin{bmatrix} \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ 0 & 0 & \times & \times & \times \\ 0 & 0 & \times & \times & \times \\ 0 & 0 & 0 & \times & \times \end{bmatrix} \xrightarrow{G(3,4,c_{34},s_{34})} \begin{bmatrix} \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ 0 & 0 & \times & \times & \times \\ 0 & 0 & 0 & \times & \times \\ 0 & 0 & 0 & \times & \times \end{bmatrix}$$

$$\xrightarrow{G(4,5,c_{45},s_{45})} \begin{bmatrix} \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ 0 & 0 & \times & \times & \times \\ 0 & 0 & 0 & \times & \times \\ 0 & 0 & 0 & 0 & \times \end{bmatrix} = R \quad (4.21)$$

Let $G_k = G\left(k, k+1, c_{k,k+1}, s_{k,k+1}\right)$, then we deduce that

$$G_4 G_3 G_2 G_1 H = R \quad (4.22)$$

$$H = G_1^* G_2^* G_3^* G_4^* R \quad (4.23)$$

$$H = QR, \text{ where } Q = G_1^* G_2^* G_3^* G_4^* \quad (4.24)$$

Note that we only presented this algorithm with a 5×5 matrix, but it can be easily generalized to any $m \times m$ matrix. In general, if H is a $m \times m$ matrix, $(m-1)$ givens rotations are required to transform H into R.
Now that we have the QR decomposition of H, we can implement the basic QR algorithm iteratively till the Hessenberg form converges to an upper triangular form.

4.4 Improving the QR convergence rate with Rayleigh Shifts

The implementation as of now is efficient but it can be drastically improved by using injecting shifts in every iteration. One such shift is the Rayleigh Shift, which shifts by an amount called the **Rayleigh Quotient** which changes every iteration.

4.4.1 Similarity transformation in shifts

Let matrix $\tilde{A} \in \mathbb{C}^{m \times m}$,

$$A = QR - \sigma I \quad (4.25)$$

$$\tilde{A} = RQ + \sigma I \quad (4.26)$$

We can see that $\tilde{A}$ is similar to A,

$$\tilde{A} = Q^* \left(A - \sigma I\right) Q + \sigma I = Q^* A Q \quad (4.27)$$

4.4.2 Rayleigh Quotient Shift

We take the shift σ_k in the k^{th} step of the QR algorithm,

$$\sigma_k = H_{(n,n)}, \text{ where } n \in \{2, 3, \dots m\} \tag{4.28}$$

The algorithm below further clarifies the building of σ_k and the entire structure of the Householder transform + Givens rotation in a simple manner:

Algorithm 1 Eigenvalue Computation using Givens Rotations

```
A ∈ ℂ^{m×m}
i = 0
for n = m to 2 do
    while i < max_iterations do
        if A is upper triangular then
            break
        end if
        σ_n = A[n][n]
        A = A − σ_n I                          ▷ Apply Rayleigh shift
        A ← hessenberg(A)                      ▷ Apply Householder Transformations
        A ← givens(A)                          ▷ Apply Givens rotation
        A = A + σ_n I                          ▷ Undo Rayleigh shift
        i = i + 1
        if |A[n][n − 1]| = 0 then
            break
        end if
    end while
end for
```

4.5 Defects with convergence of QR algorithm

4.5.1 Quasi-Triangular matrices

The QR algorithm will converge to matrix which is of the form where some subdiagonal elements will not converge to 0. This form is known as Quasi-Triangular form. Fortunately, this case only occurs when the matrix has only real entries but has complex eigenvalues.

4.5.2 Jordan blocks

When a quasi-triangular form occurs, there will be 2×2 blocks protruding the diagonal of the matrix.

$$\begin{bmatrix} \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ 0 & 0 & 0 & \times & \times \\ 0 & 0 & 0 & 0 & \times \end{bmatrix}$$

These Jordan Blocks can easily be solved be taking all such blocks, and then taking the eigenvalues of this sub 2×2 matrix block of all blocks. Note that the eigenvalues of this block will be a conjugate complex pair.

5 C implementation

The implementation written in C can be found at
`https://github.com/agamjotsingh1/eigen`.

References

[1] ETH Zurich *QR Algorithms*. Available at: `https://people.inf.ethz.ch/arbenz/ewp/Lnotes/chapter4.pdf`.

[2] MIT OpenCourseWare *Computing Eigenvalues and Singular values*. Available at: `https://youtu.be/d32WV1rKoVk`.

[3] 3Blue1Brown *Essence of Linear Algebra Playlist*. Available at: `https://youtube.com/playlist?list=PLZHQObOWTQDPD3MizzM2xVFitgF8hE_ab&si=oxNwAmrJMTRl0K5V`.

[4] Wikipedia contributors *Eigenvalue algorithm*. Available at: `https://en.wikipedia.org/wiki/Eigenvalue_algorithm#Hessenberg_and_tridiagonal_matrices`.