QR Algorithm for Eigenvalue Problems

Rafikul Alam
Department of Mathematics
Indian Institute of Technology Guwahati
Guwahati - 781039, INDIA

Outline

- Basic QR algorithm
- Reduction to Hessenberg and tridiagonal forms
- Hessenberg QR algorithm

Eigenvalues via characteristic polynomial

Let $A \in \mathbb{C}^{n\times n}$ and $p(x) := \det(xI - A)$ be the characteristic polynomial of A. Then $\lambda \in \mathbb{C}$ is an eigenvalue of $A \iff p(\lambda) = 0$. Thus eigenvalues of A can be computed by finding the roots of $p(x)$. To see the efficacy of this method, consider $A := \mathrm{diag}(1, 2, \ldots, 22)$.

Eigenvalues via characteristic polynomial

Let $A \in \mathbb{C}^{n\times n}$ and $p(x) := \det(xI - A)$ be the characteristic polynomial of A. Then $\lambda \in \mathbb{C}$ is an eigenvalue of $A \iff p(\lambda) = 0$. Thus eigenvalues of A can be computed by finding the roots of $p(x)$. To see the efficacy of this method, consider $A := \mathrm{diag}(1, 2, \ldots, 22)$.

```
>> A = diag(1:22) ; % eigenvalues 1,2,...,22
>> p = charpoly(A); % coefficents of the poly p(x)
>> rt = roots(p); % roots of p(x)
>> rt(6:9) % displays four roots
ans =
17.564435730130054 + 0.661474607910510i
17.564435730130054 - 0.661474607910510i
15.388471193084563 + 0.581348923952655i
15.388471193084563 - 0.581348923952655i
```

These roots do not have even a single correct digit! Never compute roots of $\det(\lambda I - A)$ for computing eigenvalues of A.

Computation of eigenvalues and eigenvectors

The MATLAB coammand `[V, D] = eig(A)` computes eigenvalues and eigenvectors of A. The diagonal matrix D contains the computed eigenvalues and the columns of V are the computed eigenvectors of A such that

$$\|AV - VD\|_2/\|A\|_2 = \mathcal{O}(\mathbf{u}).$$

Computation of eigenvalues and eigenvectors

The MATLAB coammand `[V, D] = eig(A)` computes eigenvalues and eigenvectors of A. The diagonal matrix D contains the computed eigenvalues and the columns of V are the computed eigenvectors of A such that

$$\|AV - VD\|_2/\|A\|_2 = \mathcal{O}(\mathbf{u}).$$

However, there is a caveat: V may or may not be invertible.

Computation of eigenvalues and eigenvectors

The MATLAB coammand `[V, D] = eig(A)` computes eigenvalues and eigenvectors of A. The diagonal matrix D contains the computed eigenvalues and the columns of V are the computed eigenvectors of A such that

$$\|AV - VD\|_2/\|A\|_2 = \mathcal{O}(\mathbf{u}).$$

However, there is a caveat: V may or may not be invertible.

Define the residual $R := VD - AV$. MATLAB ensures that the relative residual error $\|R\|_2/\|A\|_2 = \mathcal{O}(\mathbf{u})$. Define $E := RV^{-1}$ when V is invertible.

Computation of eigenvalues and eigenvectors

The MATLAB coammand `[V, D] = eig(A)` computes eigenvalues and eigenvectors of A. The diagonal matrix D contains the computed eigenvalues and the columns of V are the computed eigenvectors of A such that

$$\|AV - VD\|_2/\|A\|_2 = \mathcal{O}(\mathbf{u}).$$

However, there is a caveat: V may or may not be invertible.

Define the residual $R := VD - AV$. MATLAB ensures that the relative residual error $\|R\|_2/\|A\|_2 = \mathcal{O}(\mathbf{u})$. Define $E := RV^{-1}$ when V is invertible.

Then, when A is diagonalizable, setting $E := RV^{-1}$, we have

$$R = VD - AV \Longrightarrow (A + E)V = VD \text{ and } \|E\|_2/\|A\|_2 \leq \|R\|_2\|V^{-1}\|_2/\|A\|_2.$$

Hence computed eigenvalues and eigenvectors of A are exact eigenvalues and eigenvectors of $A + E$ and $\|E\|_2/\|A\|_2 = \mathcal{O}(\mathbf{u})\|V^{-1}\|_2$.

Computation of eigenvalues and eigenvectors

The MATLAB coammand `[V, D] = eig(A)` computes eigenvalues and eigenvectors of A. The diagonal matrix D contains the computed eigenvalues and the columns of V are the computed eigenvectors of A such that

$$\|AV - VD\|_2/\|A\|_2 = \mathcal{O}(\mathbf{u}).$$

However, there is a caveat: V may or may not be invertible.

Define the residual $R := VD - AV$. MATLAB ensures that the relative residual error $\|R\|_2/\|A\|_2 = \mathcal{O}(\mathbf{u})$. Define $E := RV^{-1}$ when V is invertible.

Then, when A is diagonalizable, setting $E := RV^{-1}$, we have

$$R = VD - AV \implies (A+E)V = VD \text{ and } \|E\|_2/\|A\|_2 \leq \|R\|_2\|V^{-1}\|_2/\|A\|_2.$$

Hence computed eigenvalues and eigenvectors of A are exact eigenvalues and eigenvectors of $A + E$ and $\|E\|_2/\|A\|_2 = \mathcal{O}(\mathbf{u})\|V^{-1}\|_2$. Note that the backward error $\|E\|_2/\|A\|_2 = \mathcal{O}(\mathbf{u})$ when $\|V^{-1}\|_2 = \mathcal{O}(1)$ and in such a case the algorithm is backward stable.

Example

The command `A = gallery(3)` generates a 3×3 test matrix in MATLAB with known eigenvalues and is given by

$$A = \begin{bmatrix} -149 & -50 & -154 \\ 537 & 180 & 546 \\ -27 & -9 & -25 \end{bmatrix}.$$

Example

The command `A = gallery(3)` generates a 3×3 test matrix in MATLAB with known eigenvalues and is given by

$$A = \begin{bmatrix} -149 & -50 & -154 \\ 537 & 180 & 546 \\ -27 & -9 & -25 \end{bmatrix}.$$

The eigenvalues of A are $1, 2, 3$. The command `[V, D] = eig(A)` gives

$$V = \begin{bmatrix} 0.3162 & -0.4041 & -0.1391 \\ -0.9487 & 0.9091 & 0.9740 \\ -0.0000 & 0.1010 & -0.1789 \end{bmatrix}, \; D = \begin{bmatrix} 1.0000 & 0 & 0 \\ 0 & 2.0000 & 0 \\ 0 & 0 & 3.0000 \end{bmatrix}.$$

Example

The command `A = gallery(3)` generates a 3×3 test matrix in MATLAB with known eigenvalues and is given by

$$A = \begin{bmatrix} -149 & -50 & -154 \\ 537 & 180 & 546 \\ -27 & -9 & -25 \end{bmatrix}.$$

The eigenvalues of A are $1, 2, 3$. The command `[V, D] = eig(A)` gives

$$V = \begin{bmatrix} 0.3162 & -0.4041 & -0.1391 \\ -0.9487 & 0.9091 & 0.9740 \\ -0.0000 & 0.1010 & -0.1789 \end{bmatrix}, \; D = \begin{bmatrix} 1.0000 & 0 & 0 \\ 0 & 2.0000 & 0 \\ 0 & 0 & 3.0000 \end{bmatrix}.$$

Now $R := VD - AV$, $E := RV^{-1} = VDV^{-1} - A$ and $(A + E)V = VD$. In this case, the residual $\|R\|_2/\|A\|_2 = 1.9579 \times 10^{-16}$ is small but the backward error $\|E\|_2/\|A\|_2 = 2.0 \times 10^{-14}$ is large for double precision computation.

Example

The command `A = gallery(3)` generates a 3×3 test matrix in MATLAB with known eigenvalues and is given by

$$A = \begin{bmatrix} -149 & -50 & -154 \\ 537 & 180 & 546 \\ -27 & -9 & -25 \end{bmatrix}.$$

The eigenvalues of A are $1, 2, 3$. The command `[V, D] = eig(A)` gives

$$V = \begin{bmatrix} 0.3162 & -0.4041 & -0.1391 \\ -0.9487 & 0.9091 & 0.9740 \\ -0.0000 & 0.1010 & -0.1789 \end{bmatrix}, \; D = \begin{bmatrix} 1.0000 & 0 & 0 \\ 0 & 2.0000 & 0 \\ 0 & 0 & 3.0000 \end{bmatrix}.$$

Now $R := VD - AV$, $E := RV^{-1} = VDV^{-1} - A$ and $(A+E)V = VD$. In this case, the residual $\|R\|_2/\|A\|_2 = 1.9579 \times 10^{-16}$ is small but the backward error $\|E\|_2/\|A\|_2 = 2.0 \times 10^{-14}$ is large for double precision computation. The reason is that $\|V^{-1}\|_2 = 7.541 \times 10^2$ which shows that

$$\|R\|_2 \|V^{-1}\|_2 / \|A\|_2 = 1.4765 \times 10^{-13} \simeq \|E\|_2/\|A\|_2. \blacksquare$$

Perturbation of Jordan block

Let $\epsilon \geq 0$. Consider an $n \times n$ Jordan block A and its perturbation $A(\epsilon)$ given by

$$A := \begin{bmatrix} 2 & 1 & & \\ & \ddots & \ddots & \\ & & \ddots & 1 \\ & & & 2 \end{bmatrix} \in \mathbb{C}^{n\times n} \quad \text{and} \quad A(\epsilon) := \begin{bmatrix} 2 & 1 & & \\ & \ddots & \ddots & \\ & & \ddots & 1 \\ \epsilon & & & 2 \end{bmatrix} \in \mathbb{C}^{n\times n}.$$

Perturbation of Jordan block

Let $\epsilon \geq 0$. Consider an $n \times n$ Jordan block A and its perturbation $A(\epsilon)$ given by

$$A := \begin{bmatrix} 2 & 1 & & \\ & \ddots & \ddots & \\ & & \ddots & 1 \\ & & & 2 \end{bmatrix} \in \mathbb{C}^{n\times n} \quad \text{and} \quad A(\epsilon) := \begin{bmatrix} 2 & 1 & & \\ & \ddots & \ddots & \\ & & \ddots & 1 \\ \epsilon & & & 2 \end{bmatrix} \in \mathbb{C}^{n\times n}.$$

Then $p_\epsilon(x) := \det(xI - A(\epsilon)) = (x-2)^n + \epsilon$ shows that $\lambda := 2$ is the eigenvalue of $A(0)$ of multiplicity n and $A(\epsilon)$ has n distinct eigenvalues

$$\lambda_j(\epsilon) := 2 + \epsilon^{1/n}\, e^{(2j-1)\pi i/n}, \;\; j = 1:n, \;\; \text{when } \epsilon > 0.$$

Perturbation of Jordan block

Let $\epsilon \geq 0$. Consider an $n \times n$ Jordan block A and its perturbation $A(\epsilon)$ given by

$$A := \begin{bmatrix} 2 & 1 & & \\ & \ddots & \ddots & \\ & & \ddots & 1 \\ & & & 2 \end{bmatrix} \in \mathbb{C}^{n\times n} \quad \text{and} \quad A(\epsilon) := \begin{bmatrix} 2 & 1 & & \\ & \ddots & \ddots & \\ & & \ddots & 1 \\ \epsilon & & & 2 \end{bmatrix} \in \mathbb{C}^{n\times n}.$$

Then $p_\epsilon(x) := \det(xI - A(\epsilon)) = (x-2)^n + \epsilon$ shows that $\lambda := 2$ is the eigenvalue of $A(0)$ of multiplicity n and $A(\epsilon)$ has n distinct eigenvalues

$$\lambda_j(\epsilon) := 2 + \epsilon^{1/n}\, e^{(2j-1)\pi i/n}, \;\; j = 1:n, \;\; \text{when } \epsilon > 0.$$

Consequently, we have $|\lambda_j(\epsilon) - 2| = \epsilon^{1/n}$ for $j = 1 : n$.

Perturbation of Jordan block

Let $\epsilon \geq 0$. Consider an $n \times n$ Jordan block A and its perturbation $A(\epsilon)$ given by

$$A := \begin{bmatrix} 2 & 1 & & \\ & \ddots & \ddots & \\ & & \ddots & 1 \\ & & & 2 \end{bmatrix} \in \mathbb{C}^{n\times n} \quad \text{and} \quad A(\epsilon) := \begin{bmatrix} 2 & 1 & & \\ & \ddots & \ddots & \\ & & \ddots & 1 \\ \epsilon & & & 2 \end{bmatrix} \in \mathbb{C}^{n\times n}.$$

Then $p_\epsilon(x) := \det(xI - A(\epsilon)) = (x-2)^n + \epsilon$ shows that $\lambda := 2$ is the eigenvalue of $A(0)$ of multiplicity n and $A(\epsilon)$ has n distinct eigenvalues

$$\lambda_j(\epsilon) := 2 + \epsilon^{1/n}\, e^{(2j-1)\pi i/n}, \quad j = 1:n, \quad \text{when } \epsilon > 0.$$

Consequently, we have $|\lambda_j(\epsilon) - 2| = \epsilon^{1/n}$ for $j = 1:n$. For $n = 15$ and $\epsilon = 10^{-15}$, we have $|\lambda_j(10^{-15}) - 2| = 10^{-15/15} = 10^{-1}$ for $j = 1:n$.

Perturbation of Jordan block

Let $\epsilon \geq 0$. Consider an $n \times n$ Jordan block A and its perturbation $A(\epsilon)$ given by

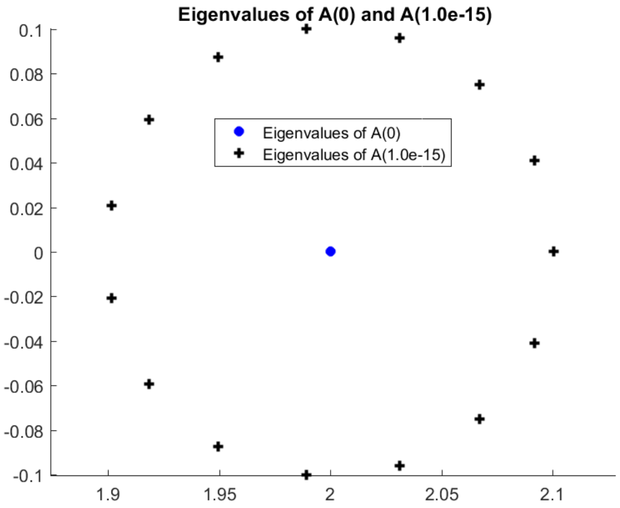
$$A := \begin{bmatrix} 2 & 1 & & \\ & \ddots & \ddots & \\ & & \ddots & 1 \\ & & & 2 \end{bmatrix} \in \mathbb{C}^{n\times n} \text{ and } A(\epsilon) := \begin{bmatrix} 2 & 1 & & \\ & \ddots & \ddots & \\ & & \ddots & 1 \\ \epsilon & & & 2 \end{bmatrix} \in \mathbb{C}^{n\times n}.$$

Then $p_\epsilon(x) := \det(xI - A(\epsilon)) = (x-2)^n + \epsilon$ shows that $\lambda := 2$ is the eigenvalue of $A(0)$ of multiplicity n and $A(\epsilon)$ has n distinct eigenvalues

$$\lambda_j(\epsilon) := 2 + \epsilon^{1/n}\, e^{(2j-1)\pi i/n}, \;\; j = 1:n, \text{ when } \epsilon > 0.$$

Consequently, we have $|\lambda_j(\epsilon) - 2| = \epsilon^{1/n}$ for $j = 1:n$. For $n = 15$ and $\epsilon = 10^{-15}$, we have $|\lambda_j(10^{-15}) - 2| = 10^{-15/15} = 10^{-1}$ for $j = 1:n$. This shows that the error 10^{-15} in the $(15, 1)$ entry of $A(0) = A$ is magnified 10^{14} times in the eigenvalues of $A(10^{-15})$.

Perturbation of Jordan block



Computation of Schur triangular form

Let $A \in \mathbb{C}^{n\times n}$. Then by Schur theorem there is a unitary matrix $U \in \mathbb{C}^{n\times n}$ such that U^*AU is upper triangular, that is,

$$U^*AU = \begin{bmatrix} t_{11} & t_{12} & \cdots & t_{1n} \\ & t_{22} & \cdots & t_{2n} \\ & & \ddots & \vdots \\ & & & t_{nn} \end{bmatrix} =: T.$$

The diagonal entries of T are the eigenvalues of A and the eigenvectors of A can be obtained from those of T. Indeed, if (λ, v) is an eigenpair of T then (λ, Uv) is an eigenpair of A.

Computation of Schur triangular form

Let $A \in \mathbb{C}^{n\times n}$. Then by Schur theorem there is a unitary matrix $U \in \mathbb{C}^{n\times n}$ such that U^*AU is upper triangular, that is,

$$U^*AU = \begin{bmatrix} t_{11} & t_{12} & \cdots & t_{1n} \\ & t_{22} & \cdots & t_{2n} \\ & & \ddots & \vdots \\ & & & t_{nn} \end{bmatrix} =: T.$$

The diagonal entries of T are the eigenvalues of A and the eigenvectors of A can be obtained from those of T. Indeed, if (λ, v) is an eigenpair of T then (λ, Uv) is an eigenpair of A.

In particular, if A is Hermitian then T is a real diagonal matrix and in such a case columns of U are orthonormal eigenvectors of A.

Computation of Schur triangular form

Let $A \in \mathbb{C}^{n\times n}$. Then by Schur theorem there is a unitary matrix $U \in \mathbb{C}^{n\times n}$ such that U^*AU is upper triangular, that is,

$$U^*AU = \begin{bmatrix} t_{11} & t_{12} & \cdots & t_{1n} \\ & t_{22} & \cdots & t_{2n} \\ & & \ddots & \vdots \\ & & & t_{nn} \end{bmatrix} =: T.$$

The diagonal entries of T are the eigenvalues of A and the eigenvectors of A can be obtained from those of T. Indeed, if (λ, v) is an eigenpair of T then (λ, Uv) is an eigenpair of A.

In particular, if A is Hermitian then T is a real diagonal matrix and in such a case columns of U are orthonormal eigenvectors of A.

As, eigenvectors of T are ei corresponding to each Lambda i

The eigenvectors of T can be computed easily by solving the system $(T - \lambda I)v = 0$. Thus the problem of solving the eigenvalue problem $Av = \lambda v$ boils down to computing Schur decomposition of A.

Basic QR algorithm

The QR algorithm constructs a sequence of unitary matrices (Q_m) and performs the similarity transformations $A_m := Q_m^* A_{m-1} Q_m$ with $A_0 = A$ such that $A_m \longrightarrow T$ and $\prod_{j=1}^{m} Q_j \longrightarrow Q$ as $m \to \infty$, where T is upper triangular.

Basic QR algorithm

The QR algorithm constructs a sequence of unitary matrices (Q_m) and performs the similarity transformations $A_m := Q_m^* A_{m-1} Q_m$ with $A_0 = A$ such that $A_m \longrightarrow T$ and $\prod_{j=1}^{m} Q_j \longrightarrow Q$ as $m \to \infty$, where T is upper triangular. The unitary matrix Q_m is computed from QR factorization of A_{m-1} and the resulting algorithm is called the QR algorithm.

Basic QR algorithm

The QR algorithm constructs a sequence of unitary matrices (Q_m) and performs the similarity transformations $A_m := Q_m^* A_{m-1} Q_m$ with $A_0 = A$ such that $A_m \longrightarrow T$ and $\prod_{j=1}^m Q_j \longrightarrow Q$ as $m \to \infty$, where T is upper triangular. The unitary matrix Q_m is computed from QR factorization of A_{m-1} and the resulting algorithm is called the QR algorithm.

Algorithm. (Basic QR algorithm)
Input: An $n \times n$ matrix A
Output: Upper triangular matrix $T = Q^* AQ$

$A_0 := A$
for $m = 1, 2, \ldots$
    $A_{m-1} = Q_m R_m$ % (QR factorization of A_{m-1})
    $A_m := R_m Q_m$ % (similarity transformation $Q_m^* A_{m-1} Q_m$)
end

Basic QR algorithm

The QR algorithm constructs a sequence of unitary matrices (Q_m) and performs the similarity transformations $A_m := Q_m^* A_{m-1} Q_m$ with $A_0 = A$ such that $A_m \longrightarrow T$ and $\prod_{j=1}^{m} Q_j \longrightarrow Q$ as $m \to \infty$, where T is upper triangular. The unitary matrix Q_m is computed from QR factorization of A_{m-1} and the resulting algorithm is called the QR algorithm.

Algorithm. (Basic QR algorithm)
Input: An $n \times n$ matrix A
Output: Upper triangular matrix $T = Q^* A Q$

$A_0 := A$
for $m = 1, 2, \ldots$
  $A_{m-1} = Q_m R_m$ % (QR factorization of A_{m-1})
  $A_m := R_m Q_m$ % (similarity transformation $Q_m^* A_{m-1} Q_m$)
end

Note that $A_m = R_m Q_m = Q_m^* Q_m R_m Q_m = Q_m^* A_{m-1} Q_m$.

Basic QR algorithm

The QR algorithm constructs a sequence of unitary matrices (Q_m) and performs the similarity transformations $A_m := Q_m^* A_{m-1} Q_m$ with $A_0 = A$ such that $A_m \longrightarrow T$ and $\prod_{j=1}^{m} Q_j \longrightarrow Q$ as $m \to \infty$, where T is upper triangular. The unitary matrix Q_m is computed from QR factorization of A_{m-1} and the resulting algorithm is called the QR algorithm.

Algorithm. (Basic QR algorithm)
Input: An $n \times n$ matrix A
Output: Upper triangular matrix $T = Q^* AQ$

$A_0 := A$
for $m = 1, 2, \ldots$
  $A_{m-1} = Q_m R_m$ % (QR factorization of A_{m-1})
  $A_m := R_m Q_m$ % (similarity transformation $Q_m^* A_{m-1} Q_m$)
end

Note that $A_m = R_m Q_m = Q_m^* Q_m R_m Q_m = Q_m^* A_{m-1} Q_m$. The cost of computing QR factorization $A_{m-1} = Q_m R_m$ is $4n^3/3$ flops and the cost of computing $A_m = R_m Q_m$ is $2n^3$ flops. Hence the cost of a QR-step is $10n^3/3$ flops. (MATLAB demo)

Example

The matrix `A = gallery(3)` from the MATLAB gallery is given by

$$A = \begin{bmatrix} -149 & -50 & -154 \\ 537 & 180 & 546 \\ -27 & -9 & -25 \end{bmatrix}.$$

The exact eigenvalues of A are $1, 2, 3$. We now apply basic QR algorithm to A and monitor converegence of strict lower triangular part of A_m to zero as the iteration progresses. We have the following results.

Example

The matrix `A = gallery(3)` from the MATLAB gallery is given by

$$A = \begin{bmatrix} -149 & -50 & -154 \\ 537 & 180 & 546 \\ -27 & -9 & -25 \end{bmatrix}.$$

The exact eigenvalues of A are $1, 2, 3$. We now apply basic QR algorithm to A and monitor converegence of strict lower triangular part of A_m to zero as the iteration progresses. We have the following results.

>> A_{60} =

```
3.0000e+00 -2.4098e+00 -8.0360e+02
-1.8461e-11 2.0000e+00 -1.5148e+02
3.0256e-20 -1.6389e-09 1.0000e+00
```

>> A_{85} =

```
3.0000e+00 -2.4098e+00 8.0360e+02
-7.3109e-16 2.0000e+00 1.5148e+02
-3.5709e-32 4.8844e-17 1.0000e+00
```

Strategy

The basic QR algorithm is too expensive as it costs $10n^3/3$ flops per QR step. Moreover, the convergence is quite slow and in some cases it may not even converge.

Strategy

The basic QR algorithm is too expensive as it costs $10n^3/3$ flops per QR step. Moreover, the convergence is quite slow and in some cases it may not even converge.

The cost of a QR step can be brought down by reducing a matrix to Hessenberg form or tridiagonal form by means of unitary similarity transformations. The eigenvalue computation proceeds as follows.

Strategy

The basic QR algorithm is too expensive as it costs $10n^3/3$ flops per QR step. Moreover, the convergence is quite slow and in some cases it may not even converge.

The cost of a QR step can be brought down by reducing a matrix to Hessenberg form or tridiagonal form by means of unitary similarity transformations. The eigenvalue computation proceeds as follows.

$$\text{General matrix} \longrightarrow \text{Hessenberg form} \longrightarrow \text{Upper triangular form}$$
$$\text{Symmetric matrix} \longrightarrow \text{tridiangoal form} \longrightarrow \text{diagonal form}$$

Strategy

The basic QR algorithm is too expensive as it costs $10n^3/3$ flops per QR step. Moreover, the convergence is quite slow and in some cases it may not even converge.

The cost of a QR step can be brought down by reducing a matrix to Hessenberg form or tridiagonal form by means of unitary similarity transformations. The eigenvalue computation proceeds as follows.

$$\begin{array}{c} \text{General matrix} \longrightarrow \text{Hessenberg form} \longrightarrow \text{Upper triangular form} \\ \text{Symmetric matrix} \longrightarrow \text{tridiangoal form} \longrightarrow \text{diagonal form} \end{array}$$

Schematically

$$\underbrace{\begin{bmatrix} \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \end{bmatrix}}_{\text{General matrix}} \to \underbrace{\begin{bmatrix} \times & \times & \times & \times \\ \times & \times & \times & \times \\ & \times & \times & \times \\ & & \times & \times \end{bmatrix}}_{\text{Upper Hessenberg}} \to \underbrace{\begin{bmatrix} \times & \times & \times & \times \\ & \times & \times & \times \\ & & \times & \times \\ & & & \times \end{bmatrix}}_{\text{Upper triangular}}$$

Hessenberg/tridiagonal form

Let $A \in \mathbb{C}^{n \times n}$. Then A is said to be upper Hessenberg if $a_{ij} = 0$ whenever $i > j + 1$. We refer to an upper Hessenberg matrix as Hessenberg matrix.

Hessenberg/tridiagonal form

Let $A \in \mathbb{C}^{n\times n}$. Then A is said to be upper Hessenberg if $a_{ij} = 0$ whenever $i > j+1$. We refer to an upper Hessenberg matrix as Hessenberg matrix.

On the other hand, A said to be tridiagonal if $a_{ij} = 0$ whenever $|i-j| > 1$.

Hessenberg/tridiagonal form

Let $A \in \mathbb{C}^{n\times n}$. Then A is said to be upper Hessenberg if $a_{ij} = 0$ whenever $i > j+1$. We refer to an upper Hessenberg matrix as Hessenberg matrix.

On the other hand, A said to be tridiagonal if $a_{ij} = 0$ whenever $|i-j| > 1$. Hessenberg and tridiagonal matrices have the forms:

$$\underbrace{\begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ & \times & \times & \times & \times \\ & & \times & \times & \times \\ & & & \times & \times \end{bmatrix}}_{\text{Hessenberg}} \quad \text{and} \quad \underbrace{\begin{bmatrix} \times & \times & & & \\ \times & \times & \times & & \\ & \times & \times & \times & \\ & & \times & \times & \times \\ & & & \times & \times \end{bmatrix}}_{\text{tridiagonal}}$$

Hessenberg/tridiagonal form

Let $A \in \mathbb{C}^{n\times n}$. Then A is said to be upper Hessenberg if $a_{ij} = 0$ whenever $i > j+1$. We refer to an upper Hessenberg matrix as Hessenberg matrix.

On the other hand, A said to be tridiagonal if $a_{ij} = 0$ whenever $|i-j| > 1$. Hessenberg and tridiagonal matrices have the forms:

$$\underbrace{\begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ & \times & \times & \times & \times \\ & & \times & \times & \times \\ & & & \times & \times \end{bmatrix}}_{\text{Hessenberg}} \quad \text{and} \quad \underbrace{\begin{bmatrix} \times & \times & & & \\ \times & \times & \times & & \\ & \times & \times & \times & \\ & & \times & \times & \times \\ & & & \times & \times \end{bmatrix}}_{\text{tridiagonal}}$$

The reduction of a matrix to Hessenberg/tridiagonal form is a finite process.

Hessenberg/tridiagonal form

Let $A \in \mathbb{C}^{n\times n}$. Then A is said to be upper Hessenberg if $a_{ij} = 0$ whenever $i > j+1$. We refer to an upper Hessenberg matrix as Hessenberg matrix.

On the other hand, A said to be tridiagonal if $a_{ij} = 0$ whenever $|i-j| > 1$. Hessenberg and tridiagonal matrices have the forms:

$$\underbrace{\begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ & \times & \times & \times & \times \\ & & \times & \times & \times \\ & & & \times & \times \end{bmatrix}}_{\text{Hessenberg}} \quad \text{and} \quad \underbrace{\begin{bmatrix} \times & \times & & & \\ \times & \times & \times & & \\ & \times & \times & \times & \\ & & \times & \times & \times \\ & & & \times & \times \end{bmatrix}}_{\text{tridiagonal}}$$

The reduction of a matrix to Hessenberg/tridiagonal form is a finite process. A finite number of unitary similarity transformations can be used to reduce a matrix to Hessenberg/tridiagonal form.

Reduction to Hessenberg form

Householder reflectors can be used to reduce A to Hessenberg/tridiagonal form in $n - 2$ steps.

Reduction to Hessenberg form

Householder reflectors can be used to reduce A to Hessenberg/tridiagonal form in $n-2$ steps. Construct a Householder reflector Q_1 that does not alter the first row of A and creates zeros in the first column below the (2, 1) entry. Then perform the similarity transformation $Q_1^* A Q_1$. Schematically, we have

$$\underbrace{\begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \end{bmatrix}}_{A} \rightarrow \underbrace{\begin{bmatrix} \times & \times & \times & \times & \times \\ * & * & * & * & * \\ & * & * & * & * \\ & * & * & * & * \\ & * & * & * & * \end{bmatrix}}_{Q_1^* A} \rightarrow \underbrace{\begin{bmatrix} \times & * & * & * & * \\ \times & * & * & * & * \\ & * & * & * & * \\ & * & * & * & * \\ & * & * & * & * \end{bmatrix}}_{Q_1^* A Q_1}.$$

We now repeat this process in the second column and construct a reflector Q_2 that leaves the first 2 rows unchanged and creates zeros in second column below the (3, 2) entry.

Reduction to Hessenberg form

Schematically,

$$\underbrace{\begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ & \times & \times & \times & \times \\ & \times & \times & \times & \times \\ & \times & \times & \times & \times \end{bmatrix}}_{Q_1^* A Q_1} \rightarrow \underbrace{\begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ & * & * & * & * \\ & & * & * & * \\ & & * & * & * \end{bmatrix}}_{Q_2^* Q_1^* A Q_1} \rightarrow \underbrace{\begin{bmatrix} \times & \times & * & * & * \\ \times & \times & * & * & * \\ & \times & * & * & * \\ & & * & * & * \\ & & * & * & * \end{bmatrix}}_{Q_2^* Q_1^* A Q_1 Q_2}.$$

Reduction to Hessenberg form

Schematically,

$$\underbrace{\begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ & \times & \times & \times & \times \\ & \times & \times & \times & \times \\ & \times & \times & \times & \times \end{bmatrix}}_{Q_1^* A Q_1} \rightarrow \underbrace{\begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ & * & * & * & * \\ & & * & * & * \\ & & * & * & * \end{bmatrix}}_{Q_2^* Q_1^* A Q_1} \rightarrow \underbrace{\begin{bmatrix} \times & \times & * & * & * \\ \times & \times & * & * & * \\ & \times & * & * & * \\ & & * & * & * \\ & & * & * & * \end{bmatrix}}_{Q_2^* Q_1^* A Q_1 Q_2}.$$

Again repeating the idea in the third column, we construct a reflector Q_3 that leaves the first 3 rows unchanged and creates zeros in third column below the $(4, 3)$ entry, and so on.

Reduction to Hessenberg form

Schematically,

$$\underbrace{\begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ & \times & \times & \times & \times \\ & \times & \times & \times & \times \\ & \times & \times & \times & \times \end{bmatrix}}_{Q_1^* A Q_1} \rightarrow \underbrace{\begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ & * & * & * & * \\ & & * & * & * \\ & & * & * & * \end{bmatrix}}_{Q_2^* Q_1^* A Q_1} \rightarrow \underbrace{\begin{bmatrix} \times & \times & * & * & * \\ \times & \times & * & * & * \\ & \times & * & * & * \\ & & * & * & * \\ & & * & * & * \end{bmatrix}}_{Q_2^* Q_1^* A Q_1 Q_2}.$$

Again repeating the idea in the third column, we construct a reflector Q_3 that leaves the first 3 rows unchanged and creates zeros in third column below the (4, 3) entry, and so on. Repeating this process $n-2$ times, the matrix A is reduced to Hessenberg form

$$H = \underbrace{Q_{n-2}^* \cdots Q_2^* Q_1^*}_{Q^*} A \underbrace{Q_1 Q_2 \cdots Q_{n-2}}_{Q} = Q^* A Q.$$

Householder reduction to Hessenberg form

Step-1: Partition A as

$$A = \begin{bmatrix} a_{11} & c^\top \\ b & \widehat{A} \end{bmatrix}.$$

Choose a reflector $\widehat{Q}_1 \in \mathbb{C}^{(n-1)\times(n-1)}$ such that $\widehat{Q}_1^* b = [-\sigma_1, 0, \ldots, 0]^\top$. Set $Q_1 := \mathrm{diag}(1, \widehat{Q}_1)$. Then

$$Q_1^* A Q_1 = \left[\begin{array}{c|c} a_{11} & c^\top \widehat{Q}_1 \\ \hline -\sigma_1 & \\ 0 & \\ \vdots & \widehat{Q}_1^* \widehat{A} \widehat{Q}_1 \\ 0 & \end{array}\right] = \left[\begin{array}{c|ccc} a_{11} & * & \cdots & * \\ \hline -\sigma_1 & & & \\ 0 & & & \\ \vdots & & \widehat{A}_1 & \\ 0 & & & \end{array}\right].$$

Note that because of the form of Q_1, the zeros in the first column of $Q_1^* A$ remain unchanged when $Q_1^* A$ is transformed to $Q_1^* A Q_1$.

Householder reduction to Hessenberg form

Step-2: The second step creates zeros in the first column of $\widehat{A}_1$. Thus we choose a reflector $\widehat{Q}_2 \in \mathbb{C}^{(n-2)\times(n-2)}$ in just the same way as in the first step, except that A is replaced by $\widehat{A}_1$. Then

$$\underbrace{\left[\begin{array}{cc|ccc} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ \hline 0 & 0 & & & \\ \vdots & \vdots & & \widehat{Q}_2 & \\ 0 & 0 & & & \end{array}\right]}_{Q_2} \text{ and } \underbrace{\left[\begin{array}{c|c|ccc} a_{11} & * & * & \cdots & * \\ \hline -\sigma_1 & * & * & \cdots & * \\ \hline 0 & -\sigma_2 & & & \\ \vdots & \vdots & & \widehat{Q}_2^* \widehat{A}_2 \widehat{Q}_2 & \\ 0 & 0 & & & \end{array}\right]}_{Q_2^* Q_1^* A Q_1 Q_2}.$$

Householder reduction to Hessenberg form

Step-2: The second step creates zeros in the first column of $\widehat{A}_1$. Thus we choose a reflector $\widehat{Q}_2 \in \mathbb{C}^{(n-2)\times(n-2)}$ in just the same way as in the first step, except that A is replaced by $\widehat{A}_1$. Then

$$\underbrace{\left[\begin{array}{cc|ccc} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ \hline 0 & 0 & & & \\ \vdots & \vdots & & \widehat{Q}_2 & \\ 0 & 0 & & & \end{array}\right]}_{Q_2} \text{ and } \underbrace{\left[\begin{array}{c|c|ccc} a_{11} & * & * & \cdots & * \\ \hline -\sigma_1 & * & * & \cdots & * \\ \hline 0 & -\sigma_2 & & & \\ \vdots & \vdots & & \widehat{Q}_2^* \widehat{A}_2 \widehat{Q}_2 & \\ 0 & 0 & & & \end{array}\right]}_{Q_2^* Q_1^* A Q_1 Q_2}.$$

The third step creates zeros in the third column, and so on. After $n-2$ steps, we have the unitary matrix $Q := Q_1 Q_2 \cdots Q_{n-2}$ and the Hessenberg matrix

$$H = \underbrace{Q_{n-2}^* \cdots Q_2^* Q_1^*}_{Q^*} A \underbrace{Q_1 Q_2 \cdots Q_{n-2}}_{Q} = Q^* A Q.$$

Algorithm for reduction to Hessenberg form

Algorithm: Householder reduction to Hessenberg form
Input: An $n \times n$ matrix $\mathtt{A}$
Output: Hessenberg matix $\mathtt{A} \leftarrow \mathtt{Q}^*\mathtt{AQ}$

$$
\begin{array}{l}
\mathtt{for}\ \mathtt{k} = 1\colon \mathtt{n}-2 \\
\quad \mathtt{x} := \mathtt{A}(\mathtt{k}+1\colon\mathtt{n}, \mathtt{k}) \qquad \text{\% choose } k\text{-th column of } A \\
\quad \mathtt{u} := \mathrm{sign}(\mathtt{x}_1)\|\mathtt{x}\|_2\mathtt{e}_1 + \mathtt{x} \qquad \text{\% Householder vector for reflector } Q_k \\
\quad \mathtt{u} := \mathtt{u}/\|\mathtt{u}\|_2 \\
\quad \mathtt{A}(\mathtt{k}+1\colon\mathtt{n}, \mathtt{k}\colon\mathtt{n}) := \mathtt{A}(\mathtt{k}+1\colon\mathtt{n}, \mathtt{k}\colon\mathtt{n}) - 2\mathtt{u}(\mathtt{u}^*\mathtt{A}(\mathtt{k}+1\colon\mathtt{n}, \mathtt{k}\colon\mathtt{n})) \\
\quad \mathtt{A}(1\colon\mathtt{n}, \mathtt{k}+1\colon\mathtt{n}) := \mathtt{A}(1\colon\mathtt{n}, \mathtt{k}+1\colon\mathtt{n}) - 2(\mathtt{A}(1\colon\mathtt{n}, \mathtt{k}+1\colon\mathtt{n})\mathtt{u})\mathtt{u}^* \\
\mathtt{end}
\end{array}
$$

Algorithm for reduction to Hessenberg form

Algorithm: Householder reduction to Hessenberg form
Input: An $n \times n$ matrix $\mathtt{A}$
Output: Hessenberg matix $\mathtt{A} \leftarrow \mathtt{Q}^*\mathtt{AQ}$

$\mathtt{for}\ \mathtt{k} = 1{:}\,\mathtt{n} - 2$

$\quad \mathtt{x} := \mathtt{A}(\mathtt{k}+1{:}\,\mathtt{n}, \mathtt{k})$ % choose k-th column of A

$\quad \mathtt{u} := \mathrm{sign}(\mathtt{x}_1)\|\mathtt{x}\|_2\mathtt{e}_1 + \mathtt{x}$ % Householder vector for reflector Q_k

$\quad \mathtt{u} := \mathtt{u}/\|\mathtt{u}\|_2$

$\quad \mathtt{A}(\mathtt{k}+1{:}\,\mathtt{n}, \mathtt{k}{:}\,\mathtt{n}) := \mathtt{A}(\mathtt{k}+1{:}\,\mathtt{n}, \mathtt{k}{:}\,\mathtt{n}) - 2\mathtt{u}(\mathtt{u}^*\mathtt{A}(\mathtt{k}+1{:}\,\mathtt{n}, \mathtt{k}{:}\,\mathtt{n}))$

$\quad \mathtt{A}(1{:}\,\mathtt{n}, \mathtt{k}+1{:}\,\mathtt{n}) := \mathtt{A}(1{:}\,\mathtt{n}, \mathtt{k}+1{:}\,\mathtt{n}) - 2(\mathtt{A}(1{:}\,\mathtt{n}, \mathtt{k}+1{:}\,\mathtt{n})\mathtt{u})\mathtt{u}^*$

$\mathtt{end}$

The above algorithm can be modified to store the Householder vectors in each step, which can be used to assemble the unitary matrix Q.

Algorithm for reduction to Hessenberg form

Algorithm: Householder reduction to Hessenberg form
Input: An $n \times n$ matrix $\mathtt{A}$
Output: Hessenberg matix $\mathtt{A} \leftarrow \mathtt{Q}^*\mathtt{AQ}$

```
for k = 1:n − 2
    x: = A(k + 1:n, k)            % choose k-th column of A
    u: = sign(x_1)||x||_2 e_1 + x     % Householder vector for reflector Q_k
    u: = u/||u||_2
    A(k + 1:n, k:n): = A(k + 1:n, k:n) − 2u(u* A(k + 1:n, k:n))
    A(1:n, k + 1:n): = A(1:n, k + 1:n) − 2(A(1:n, k + 1:n)u)u*
end
```

The above algorithm can be modified to store the Householder vectors in each step, which can be used to assemble the unitary matrix Q.

Cost: The update $A \leftarrow Q_k A$ requires $4(n-k)^2$ flops as it operates on $A(k+1:n, k:n)$. Total cost $\sum_{k=1}^{n-2} 4(n-k)^2 \sim 4n^3/3$ flops.

Algorithm for reduction to Hessenberg form

Algorithm: Householder reduction to Hessenberg form
Input: An $n \times n$ matrix $\mathtt{A}$
Output: Hessenberg matix $\mathtt{A} \leftarrow \mathtt{Q}^*\mathtt{AQ}$

```
for k = 1:n − 2
    x := A(k + 1:n, k)            % choose k-th column of A
    u := sign(x_1)||x||_2 e_1 + x    % Householder vector for reflector Q_k
    u := u/||u||_2
    A(k + 1:n, k:n) := A(k + 1:n, k:n) − 2u(u* A(k + 1:n, k:n))
    A(1:n, k + 1:n) := A(1:n, k + 1:n) − 2(A(1:n, k + 1:n)u)u*
end
```

The above algorithm can be modified to store the Householder vectors in each step, which can be used to assemble the unitary matrix Q.

Cost: The update $A \leftarrow Q_k A$ requires $4(n-k)^2$ flops as it operates on $A(k+1:n, k:n)$. Total cost $\sum_{k=1}^{n-2} 4(n-k)^2 \sim 4n^3/3$ flops.The update $A \leftarrow AQ_k$ requires $4n(n-k)$ flops as it operates on $A(1:n, k+1:n)$. Total cost $\sum_{k=1}^{n-2} 4n(n-k) \sim 2n^3$ flops.

Algorithm for reduction to Hessenberg form

Algorithm: Householder reduction to Hessenberg form
Input: An $n \times n$ matrix $\mathtt{A}$
Output: Hessenberg matix $\mathtt{A} \leftarrow \mathtt{Q}^*\mathtt{AQ}$

$\mathtt{for}\ \mathtt{k} = 1\colon \mathtt{n}-2$
  $\mathtt{x}\colon= \mathtt{A}(\mathtt{k}+1\colon \mathtt{n}, \mathtt{k})$ % choose k-th column of A
  $\mathtt{u}\colon= \mathrm{sign}(\mathtt{x}_1)\|\mathtt{x}\|_2\mathtt{e}_1 + \mathtt{x}$ % Householder vector for reflector Q_k
  $\mathtt{u}\colon= \mathtt{u}/\|\mathtt{u}\|_2$
  $\mathtt{A}(\mathtt{k}+1\colon \mathtt{n}, \mathtt{k}\colon \mathtt{n})\colon= \mathtt{A}(\mathtt{k}+1\colon \mathtt{n}, \mathtt{k}\colon \mathtt{n}) - 2\mathtt{u}(\mathtt{u}^*\mathtt{A}(\mathtt{k}+1\colon \mathtt{n}, \mathtt{k}\colon \mathtt{n}))$
  $\mathtt{A}(1\colon \mathtt{n}, \mathtt{k}+1\colon \mathtt{n})\colon= \mathtt{A}(1\colon \mathtt{n}, \mathtt{k}+1\colon \mathtt{n}) - 2(\mathtt{A}(1\colon \mathtt{n}, \mathtt{k}+1\colon \mathtt{n})\mathtt{u})\mathtt{u}^*$
$\mathtt{end}$

The above algorithm can be modified to store the Householder vectors in each step, which can be used to assemble the unitary matrix Q.

Cost: The update $A \leftarrow Q_k A$ requires $4(n-k)^2$ flops as it operates on $A(k+1:n, k:n)$. Total cost $\sum_{k=1}^{n-2} 4(n-k)^2 \sim 4n^3/3$ flops.The update $A \leftarrow AQ_k$ requires $4n(n-k)$ flops as it operates on $A(1:n, k+1:n)$. Total cost $\sum_{k=1}^{n-2} 4n(n-k) \sim 2n^3$ flops. Hence total cost for Householder reduction to Hessenberg form is $10n^3/3$ flops.

Example

The matlab command `[Q, H] = hess(A)` computes a unitary matrix $\mathtt{Q}$ and a Hessenberg matrix $\mathtt{H}$ such that $\mathtt{H} = \mathtt{Q}^*\mathtt{A}\mathtt{Q}$. For the matrix

$$A = \begin{bmatrix} -149 & -50 & -154 \\ 537 & 180 & 546 \\ -27 & -9 & -25 \end{bmatrix},$$

`[Q, H] = hess(A)` yields

$$\mathtt{Q} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -0.9987 & 0.0502 \\ 0 & 0.0502 & 0.9987 \end{bmatrix}, \ \mathtt{H} = \begin{bmatrix} -149.0000 & 42.2037 & -156.3165 \\ -537.6783 & 152.5511 & -554.9272 \\ 0 & 0.0728 & 2.4489 \end{bmatrix}.$$

Reduction to tridiagonal form

Suppose that A is Hermitian. Then the Hessenberg matrix $H = Q^*AQ$ is Hermitian $\Rightarrow H$ is tridiagonal. Consequently, the Householder reduction of A to Hessenberg form reduces A to tridiagonal form.

Reduction to tridiagonal form

Suppose that A is Hermitian. Then the Hessenberg matrix $H = Q^*AQ$ is Hermitian $\Rightarrow H$ is tridiagonal. Consequently, the Householder reduction of A to Hessenberg form reduces A to tridiagonal form.

The symmetry of A allows an efficient implementation of Householder reduction to tridiagonal form and brings down the cost of the reduction algorithm to $4n^3/3$ flops.

Reduction to tridiagonal form

Suppose that A is Hermitian. Then the Hessenberg matrix $H = Q^*AQ$ is Hermitian $\Rightarrow H$ is tridiagonal. Consequently, the Householder reduction of A to Hessenberg form reduces A to tridiagonal form.

The symmetry of A allows an efficient implementation of Householder reduction to tridiagonal form and brings down the cost of the reduction algorithm to $4n^3/3$ flops.

Indeed, in the k-th step, we compute the update $A \leftarrow Q_k A Q_k$, where $Q_k := I + \gamma uu^*$ with $\gamma := -2/u^*u$ is a reflector. Now

$$
\begin{aligned}
Q_k A Q_k &= (I + \gamma uu^*)A(I + \gamma uu^*) \\
&= A + \gamma uu^*A + \gamma Auu^* + \gamma^2 uu^*Auu^* \\
&= A + uv^* + vu^* + 2\delta uu^* \\
&= A + u(v + \delta u)^* + (v + \delta u)u^* \\
&= A + uw^* + wu^*
\end{aligned}
$$

where $v := \gamma Au, \delta := \dfrac{1}{2}\gamma u^* v$ and $w := v + \delta u$.

Reduction to tridiagonal form

Note that $A \leftarrow A + uw^* + wu^*$ is a Hermitian rank-2 update. By symmetry, the update operates on $A(k:n, k:n)$ and requires $2(n-k)^2$ flops

Reduction to tridiagonal form

Note that $A \leftarrow A + uw^* + wu^*$ is a Hermitian rank-2 update. By symmetry, the update operates on $A(k:n, k:n)$ and requires $2(n-k)^2$ flops and the computation of $v := \gamma Au$ requires $2(n-k)^2$ flops.

Reduction to tridiagonal form

Note that $A \leftarrow A + uw^* + wu^*$ is a Hermitian rank-2 update. By symmetry, the update operates on $A(k:n, k:n)$ and requires $2(n-k)^2$ flops and the computation of $v := \gamma Au$ requires $2(n-k)^2$ flops. Hence the total cost is $\sum_{k=1}^{n-2} 4(n-k)^2 \sim 4n^3/3$ flops.

Reduction to tridiagonal form

Note that $A \leftarrow A + uw^* + wu^*$ is a Hermitian rank-2 update. By symmetry, the update operates on $A(k:n, k:n)$ and requires $2(n-k)^2$ flops and the computation of $v := \gamma Au$ requires $2(n-k)^2$ flops. Hence the total cost is $\sum_{k=1}^{n-2} 4(n-k)^2 \sim 4n^3/3$ flops.

The 4×4 Hilbert matrix is given by

$A = \begin{bmatrix} 1 & 1/2 & 1/3 & 1/4 \\ 1/2 & 1/3 & 1/4 & 1/5 \\ 1/3 & 1/4 & 1/5 & 1/6 \\ 1/4 & 1/5 & 1/6 & 1/7 \end{bmatrix}$. The command `[Q, T] = hess(A)` gives

$$Q = \begin{bmatrix} -0.1589 & 0.7036 & -0.6926 & 0 \\ 0.7417 & -0.3780 & -0.5541 & 0 \\ -0.6516 & -0.6018 & -0.4618 & 0 \\ 0 & 0 & 0 & 1.0000 \end{bmatrix} \text{ and}$$

$$T = \begin{bmatrix} 0.0031 & -0.0068 & 0 & 0 \\ -0.0068 & 0.1806 & -0.2683 & 0 \\ 0 & -0.2683 & 1.3497 & -0.3609 \\ 0 & 0 & -0.3609 & 0.1429 \end{bmatrix}.$$

Hessenberg QR algorithm

Let $A \in \mathbb{C}^{n \times n}$ be a Hessenberg matrix. Suppose we perform a QR step on A to obtain $A = QR$ and $A_1 := RQ = Q^*AQ$.

Question: Is the matrix A_1 Hessenberg? In other words, does the QR step preserve the Hessenberg structure of A?

Hessenberg QR algorithm

Let $A \in \mathbb{C}^{n\times n}$ be a Hessenberg matrix. Suppose we perform a QR step on A to obtain $A = QR$ and $A_1 := RQ = Q^*AQ$.

Question: Is the matrix A_1 Hessenberg? In other words, does the QR step preserve the Hessenberg structure of A?

If the answer is NO then the reduction to Hessenberg form is of no use as far as QR algorithm is concerned.

Hessenberg QR algorithm

Let $A \in \mathbb{C}^{n\times n}$ be a Hessenberg matrix. Suppose we perform a QR step on A to obtain $A = QR$ and $A_1 := RQ = Q^*AQ$.

Question: Is the matrix A_1 Hessenberg? In other words, does the QR step preserve the Hessenberg structure of A?

If the answer is NO then the reduction to Hessenberg form is of no use as far as QR algorithm is concerned.

Theorem: Let A be a nonsingular Hessenberg matrix. Let A_1 be the result of a QR step on A, that is, $A = QR$ and $A_1 := RQ$. Then A_1 is a Hessenberg matrix.

Proof. Note that R is nonsingular and $A = QR \Longrightarrow Q = AR^{-1}$. Since R^{-1} is upper triangular and A is Hessenberg, it is easy to see that AR^{-1} is Hessenberg. Hence Q is Hessenberg. Consequently, $A_1 = RQ$ is Hessenberg. ■

Hessenberg QR algorithm

Let $A \in \mathbb{C}^{n\times n}$ be a Hessenberg matrix. Suppose we perform a QR step on A to obtain $A = QR$ and $A_1 := RQ = Q^*AQ$.

Question: Is the matrix A_1 Hessenberg? In other words, does the QR step preserve the Hessenberg structure of A?

If the answer is NO then the reduction to Hessenberg form is of no use as far as QR algorithm is concerned.

I.MP

Theorem: Let A be a nonsingular Hessenberg matrix. Let A_1 be the result of a QR step on A, that is, $A = QR$ and $A_1 := RQ$. Then A_1 is a Hessenberg matrix.

Proof. Note that R is nonsingular and $A = QR \implies Q = AR^{-1}$. Since R^{-1} is upper triangular and A is Hessenberg, it is easy to see that AR^{-1} is Hessenberg. Hence Q is Hessenberg. Consequently, $A_1 = RQ$ is Hessenberg. ■

A QR step applied to a nonsingular Hessenberg matrix A preserves the Hessenberg structure.

Hessenberg QR algorithm

However, a QR step applied to a singular Hessenberg matrix A may not preserve the Hessenberg structure. Consider

$$\underbrace{\begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}}_{A} = \underbrace{\begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}}_{Q} \underbrace{\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{R}$$

$$\underbrace{\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix}}_{A_1} = \underbrace{\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{R} \underbrace{\begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}}_{Q}.$$

Hessenberg QR algorithm

However, a QR step applied to a singular Hessenberg matrix A may not preserve the Hessenberg structure. Consider

$$\underbrace{\begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}}_{A} = \underbrace{\begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}}_{Q} \underbrace{\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{R}$$

$$\underbrace{\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix}}_{A_1} = \underbrace{\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{R} \underbrace{\begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}}_{Q}.$$

Note that A is singular and Hessenberg but A_1 is not Hessenberg. The QR factorization of a singular matrix is not unique. Among many QR factorization, some may not preserve the Hessenberg structure when they are used in a QR step.

Hessenberg QR algorithm

However, a QR step applied to a singular Hessenberg matrix A may not preserve the Hessenberg structure. Consider

$$\underbrace{\begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}}_{A} = \underbrace{\begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}}_{Q} \underbrace{\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{R}$$

$$\underbrace{\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix}}_{A_1} = \underbrace{\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{R} \underbrace{\begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}}_{Q}.$$

Note that A is singular and Hessenberg but A_1 is not Hessenberg. The QR factorization of a singular matrix is not unique. Among many QR factorization, some may not preserve the Hessenberg structure when they are used in a QR step. However, one can choose a QR factorization that preserves the Hessenberg structure.