# 2 QR algorithm

We saw in the previous lectures that a Schur factorization of a matrix $A \in \mathbb{C}^{n\times n}$ directly gives us the eigenvalues. More precisely, if we can compute $P$ and $U$ such that

$$A = PUP^*,$$

where $P^*P = I$ and $U$ is upper triangular, then the eigenvalues of $A$ are given by the diagonal elements of $U$.

The QR method developed by Francis in 1960's [4] is classified as one of the top-ten developments in computation in the 20th century. The performance and robustness is still actively improved within the numerical linear algebra research community.

We will now introduce the QR-method, which is sometimes called the QR-algorithm or Francis's QR-steps [4]. The goal of the method is to compute a Schur factorization by means of similarity transformations. The total complexity of the algorithm is essentially $O(n^3)$, which can only be achieved in practice after several improvements are appropriately taken into account. Most of this chapter is devoted to improving the basic QR-method. The fundamental results for the convergence are based on connections with the power method and simultaneous iteration and will be covered later in the course.

Although the QR-method can be successfully adapted to arbitrary complex matrices, we will here for brevity concentrate the discussion on the case where the matrix has only real eigenvalues.

## 2.1 Basic variant of QR-method

As the name suggests, the QR-method is tightly coupled with the QR-factorization. Consider for the moment a QR-factorization of the matrix $A$,

$$A = QR$$

where $Q^*Q = I$ and $R$ is upper triangular. We will now reverse the order of multiplication product of $Q$ and $R$ and eliminate $R$,

$$RQ = Q^*AQ. \tag{2.1}$$

Since $Q^*AQ$ is a similarity transformation of $A$, $RQ$ has the same eigenvalues as $A$. More importantly, we will later see that by repeating this process, the matrix $RQ$ will become closer and closer to upper

triangular, such that we eventually can read off the eigenvalues from the diagonal. That is, the QR-method generates a sequence of matrices $A_k$ initiated with $A_0 = A$ and given by

$$A_k = R_k Q_k,$$

where $Q_k$ and $R_k$ represents a QR-factorization of $A_{k-1}$,

$$A_{k-1} = Q_k R_k.$$

Idea of *basic QR-method*: compute a QR-factorization and reverse the order of multiplcation of $Q$ and $R$.

**Algorithm 1** Basic QR algorithm

**Input:** A matrix $A \in \mathbb{C}^{n\times n}$
**Output:** Matrices $U$ and $T$ such that $A = UTU^*$.
Set $A_0 := A$ and $U_0 = I$
**for** $k = 1, \ldots$ **do**
Compute QR-factorization: $A_{k-1} =: Q_k R_k$
Set $A_k := R_k Q_k$
Set $U_k := U_{k-1} Q_k$
**end for**
Return $T = A_\infty$ and $U = U_\infty$

Although the complete understanding of convergence of the QR-method requires theory developed in later parts of this course, it is instructive to already now formulate the character of the convergence. Let us assume the eigenvalues are distinct in modulus and ordered as $|\lambda_1| > |\lambda_2| > \cdots > |\lambda_n|$. Under certain assumptions, the elements of the matrix $A_k$ below the diagonal will converge to zero according to

$$|a_{ij}^{(k)}| = \mathcal{O}(|\lambda_i/\lambda_j|^k) \text{ for all } i > j. \tag{2.2}$$

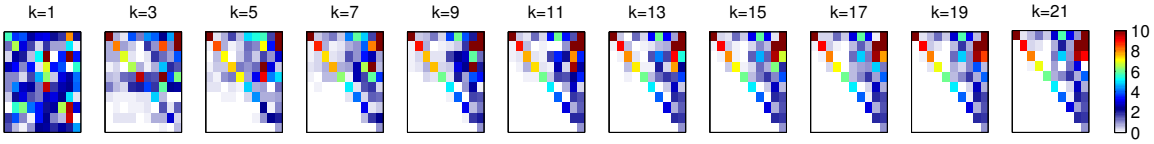
## *Example (basic QR-method)*

We conduct a simple matlab experiment to illustrate the convergence. To that end we construct a matrix with eigenvalues $1, 2, \ldots, 10$ and run the basic QR-method.

```
D=diag(1:10);
rand('seed',0);
S=rand(10); S=(S-0.5)*2;
A=S*D*inv(S);
for i=1:100
    [Q,R]=qr(A);
    A=R*Q
```

```
end
norm(sortrows(diag(A))-sortrows(diag(D)))
```

In the last step, this program returns $5 \cdot 10^{-5}$, suggesting that we do find reasonably accurate eigenvalues. This is confirmed by the fact that the matrix $A$ does indeed approach an upper triangular matrix, which can be seen in the following illustration of $A_k$.




Moreover, the illustration suggests that $A_k$ approaches a triangular matrix where the diagonal elements are ordered by (descending) magnitude. Since the diagonal elements of a triangular matrix are the eigenvalues, our observation is here $a_{11}^{(k)} \to \lambda_1 = 10$, $a_{22}^{(k)} \to \lambda_2 = 9, \ldots,$ $a_{10}^{(k)} \to \lambda_{10} = 1$ (which we will be able to show later in this course).

We can also observe the convergence claimed in (2.2) by considering the quotient of the elements below the diagonal of two consecutive iterations. Let `A20` and `A21` be the matrices generated after $k = 20$ and $k = 21$ iterations. In MATLAB notation we have

```
>> A21./A20
ans =
  1.0002  0.9313  1.0704  0.9854  1.0126  0.9929  0.9952  0.9967  1.2077 -1.0087
  0.9118  1.0003  0.9742  1.0231  0.8935  1.0534  1.0614  1.0406  1.0082 -0.9828
  0.8095  0.8917  1.0003  2.7911  1.4191  0.9689  0.9999  0.9508  0.9941 -1.0260
  0.7005  0.7718  0.8676  0.9992  0.9918  1.0509  0.2971  1.0331  0.8996 -0.2654
  0.5959  0.6746  0.7411  0.8436  1.0001  1.0022  0.9901  1.0007  0.9650 -1.0036
  0.5013  0.5602  0.6303  0.7224  0.8309  0.9997  1.0349  0.9901  0.9993 -1.0113
  0.4005  0.4475  0.4993  0.5904  0.6669  0.7908  1.0000  1.0035  1.0022 -0.9996
  0.3007  0.3344  0.3738  0.4355  0.5002 -1.9469  0.7460  0.9998  1.0006 -1.0007
  0.2002  0.2227  0.2493  0.2899  0.3332  0.4044  0.4994  0.6660  1.0003 -0.9994
 -0.1001 -0.1119 -0.1259 -0.1426 -0.1669 -0.1978 -0.2500 -0.3332 -0.5000  1.0000
```

The elements below the diagonal in the output are consistent with (2.2) in the sense that the $(i, j)$ element of the output is

$$|a_{i,j}^{(21)}/a_{i,j}^{(20)}| \approx |\lambda_i/\lambda_j|, \quad i \geq j+1.$$

## *Downsides with the basic QR-method*

Although the basic QR-method in general converges to a Schur factorization when $k \to \infty$, it is not recommended in practice.

The basic QR-method is often slow, in the sense that the number of iterations required to reach convergence is in general high. It is in general expensive in the sense that the computational effort required per step is high.

**Disadvantage 1.** One step of the basic QR-method is relatively expensive. More precisely,

the complexity of one step of the basic QR-method = $\mathcal{O}(n^3)$.

Hence, even in an overly optimistic situation where the number of steps would be proportional to $n$, the algorithm would need $\mathcal{O}(n^4)$ operations.

**Disadvantage 2.** Usually, many steps are required to have convergence; certainly much more than $n$. In fact, the basic QR-method can be arbitrarily slow if the eigenvalues are close to each other. It is often slow in practice.

## 2.2 *The two-phase approach*

The disadvantages of the basic QR-method suggest that several improvements are required to reach a competitive algorithm. The Hessenberg matrix structure will be crucial for these improvements.

**Definition 2.2.1** *A matrix $H \in \mathbb{C}^{n\times n}$ is called a* Hessenberg matrix *if its elements below the lower off-diagonal are zero,*

$$h_{i,j} = 0 \quad \textit{when } i > j+1.$$

Structure of a Hessenberg matrix:

$$H = \begin{bmatrix} \times & \cdots & \cdots & \cdots & \times \\ \times & \ddots & & & \vdots \\ 0 & \ddots & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \times \\ 0 & \cdots & 0 & \times & \times \end{bmatrix}$$

*The matrix $H$ is called an* unreduced Hessenberg matrix *if additionally $h_{i,i+1} \neq 0$ for all $i = 1, \ldots, n-1$.*

Our improved QR-method will be an algorithm consisting of two phases, illustrated as follows:

$$\begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \end{bmatrix} \xrightarrow[\text{Phase 1}]{} \begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ & \times & \times & \times & \times \\ & & \times & \times & \times \\ & & & \times & \times \end{bmatrix} \xrightarrow[\text{Phase 2}]{} \begin{bmatrix} \times & \times & \times & \times & \times \\ & \times & \times & \times & \times \\ & & \times & \times & \times \\ & & & \times & \times \\ & & & & \times \end{bmatrix}$$

**Phase 1. (Section 2.2.1).** In the first phase we will compute a Hessenberg matrix $H$ (and orthogonal $U$) such that

$$A = UHU^*.$$

Unlike the Schur factorization ($A = UTU^*$ where $T$ is upper triangular) such a reduction can be done with a finite number of operations.

**Phase 2. (Section 2.2.2).** In the second phase we will apply the basic QR-method to the matrix $H$. It turns out that several improvements can be done when applying the basic QR-method to a Hessenberg matrix such that the complexity of one step is $\mathcal{O}(n^2)$, instead of $\mathcal{O}(n^3)$ in the basic version.

### 2.2.1 *Phase 1. Hessenberg reduction*

In the following it will be advantageous to use the concept of Householder reflectors.

**Definition 2.2.2** *A matrix $P \in \mathbb{C}^{n\times n}$ of the form*

$$P = I - 2uu^* \quad \text{where } u \in \mathbb{C}^n \text{ and } \|u\| = 1$$

*is called a* Householder reflector.

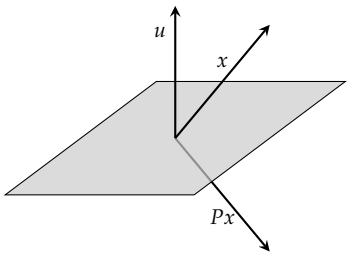



Figure 2.1: A Householder reflector is a "reflector" in the sense that the multiplication of $P$ with a vector gives the mirror image with respect to the (hyper) plane perpendicular to $u$.

Some properties of Householder reflectors:

- A Householder reflector is always hermitian $P = P^*$
- Since $P^2 = I$, a Householder reflector is always orthogonal and $P^{-1} = P^* = P$
- If $u$ is given, the corresponding Householder reflector can be multiplied by a vector in $\mathcal{O}(n)$ operations:

$$Px = x - 2u(u^*x) \tag{2.3}$$

#### *Householder reflectors satisfying $Px = \alpha e_1$*

We will now explicitly solve this problem: Given a vector $x$, construct a Householder reflector such that $Px$ is a multiple of the first unit vector. This tool will be used several times throughout this chapter.

**Lemma 2.2.3** *Suppose $x \in \mathbb{R}^m \backslash \{0\}$ and let $\rho = \pm 1$ and $\alpha := \rho\|x\|$. Let*

$$u = \frac{x - \alpha e_1}{\|x - \alpha e_1\|} = \frac{z}{\|z\|}, \tag{2.4}$$

*where*

$$z := x - \alpha e_1 = \begin{bmatrix} x_1 - \rho\|x\| \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

The choice of $u$ in (2.4) is the normal vector of a plane such that the mirror image of $x$ is in the direction of the first unit vector.

*Then, the matrix $P = I - 2uu^T$ is a Householder reflector and*

$$Px = \alpha e_1.$$

**Proof** The matrix $P$ is a Householder reflector since $u$ is normalized. From the definition of $z$ and $\alpha$ we have $z^Tz = (x - \alpha e_1)^T(x - \alpha e_1) = 2(\|x\|^2 - \rho\|x\|x_1)$. Similarly, $z^Tx = \|x\|^2 - \rho\|x\|x_1$. Hence

$$uu^Tx = \frac{z}{\|z\|}\frac{z^T}{\|z\|}x = \frac{z^Tx}{z^Tz}z = \frac{1}{2}z$$

and $(I - 2uu^T)x = x - z = \alpha e_1$, which is the conclusion of the lemma.

■

In principle, $\rho$ can be chosen freely (provided $|\rho| = 1$). The choice $\rho = -\text{sign}(x_1)$, is often better from the perspective of round-off errors. With this specific choice of $\rho$, Lemma 2.2.3 also holds in complex arithmetic.

## *Repeated application of Householder reflectors*

By carrying out $n-2$ orthogonality transformations with (cleaverly constructed) Householder reflectors we will now be able to reduce the matrix $A$ to a Hessenberg matrix.

In the first step we will carry out a similarity transformation with a Householder reflector $P_1$, given as follows

$$P_1 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \end{bmatrix} = \begin{bmatrix} 1 & 0^T \\ 0 & I - 2u_1u_1^T \end{bmatrix}. \tag{2.5}$$

Note: $I - 2u_1u_1^T \in \mathbb{C}^{(n-1)\times(n-1)}$ is the Householder reflector associated with $u_1 \in \mathbb{C}^{n-1}$ and $P_1 \in \mathbb{C}^{n\times n}$ in (2.5) is the Householder reflector associated with $[0, u_1^T] \in \mathbb{C}^n$.

The vector $u_1$ will be constructed from elements of the matrix $A$. More precisely, they will be given by (2.4) with $x^T = [a_{21}, \ldots, a_{n1}]$ such that the associated Householder reflector satisfies

$$(I - 2u_1u_1^T)\begin{bmatrix} a_{21} \\ \vdots \\ a_{n1} \end{bmatrix} = \alpha e_1.$$

Hence, multiplying $A$ from the left with $P_1$ inserts zeros in desired positions in the first column,

$$P_1A = \begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ & \times & \times & \times & \times \\ & \times & \times & \times & \times \\ & \times & \times & \times & \times \end{bmatrix}$$

In order to have a similarity transformation, we must also multiply from the right with $P_1^*$. Recall that $P_1^* = P_1$ since Householder reflectors are Hermitian. Because of the non-zero structure in $P_1$ the non-zero structure is unchanged and we have a matrix $P_1AP_1$ which is similar to $A$ and has desired zero-entries,

$$P_1AP_1^* = P_1AP_1 = \begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ & \times & \times & \times & \times \\ & \times & \times & \times & \times \\ & \times & \times & \times & \times \end{bmatrix}.$$

In the first step of the Hessenberg reduction: A similarity transformation is constructed such that $P_1AP_1^*$ has the desired (Hessenberg) zero-structure in the first column.

The process can be repeated and in the second step we set

$$P_2 = \begin{bmatrix} 1 & 0 & 0^T \\ 0 & 1 & 0^T \\ 0 & 0 & I - 2u_2u_2^T \end{bmatrix}$$

where $u_2$ is constructed from the $n-2$ last elements of the second column of $P_1AP_1^*$.

$$P_1AP_1 = \begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ & \times & \times & \times & \times \\ & \times & \times & \times & \times \\ & \times & \times & \times & \times \end{bmatrix} \xrightarrow[\text{left with } P_2]{\text{mult. from}} \begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ & \times & \times & \times & \times \\ & & \times & \times & \times \\ & & \times & \times & \times \end{bmatrix} \xrightarrow[\text{right with } P_2]{\text{mult. from}} \begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ & \times & \times & \times & \times \\ & & \times & \times & \times \\ & & \times & \times & \times \end{bmatrix} = P_2P_1AP_1P_2$$

After $n-2$ steps we have completed the Hessenberg reduction since

$$P_{n-2}P_{n-3}\cdots P_1AP_1P_2\cdots P_{n-2} = H,$$

where $H$ is a Hessenberg matrix. Note that $U = P_1P_2\cdots P_{n-1}$ is orthogonal and $A = UHU^*$ and $H$ have the same eigenvalues.

The complete algorithm is provided in Algorithm 2.2.1. In the algorithm we do not explicitly compute the orthogonal matrix $U$ since it is not required unless also eigenvectors are to be computed. In every step in the algorithm we need $\mathcal{O}(n^2)$ operations and consequently

the total complexity of the Hessenberg reduction = $\mathcal{O}(n^3)$.

**Algorithm 2** Reduction to Hessenberg form

**Input:** A matrix $A \in \mathbb{C}^{n\times n}$
**Output:** A Hessenberg matrix $H$ such that $H = U^*AU$.
**for** $k = 1, \ldots, n-2$ **do**
Compute $u_k$ using (2.4) where $x^T = [a_{k+1,k}, \cdots, a_{n,k}]$
Compute $P_kA$: $A_{k+1:n,k:n} := A_{k+1:n,k:n} - 2u_k(u_k^*A_{k+1:n,k:n})$
Compute $P_kAP_k^*$: $A_{1:n,k+1:n} := A_{1:n,k+1:n} - 2(A_{1:n,k+1:n}u_k)u_k^*$
**end for**
Let $H$ be the Hessenberg part of $A$.

The Hessenberg reduction in Algorithm 2.2.1 is implemented by overwriting the elements of $A$. In this way less memory is required.

### 2.2.2 *Phase 2. Hessenberg QR-method*

In the second phase we will apply the QR-method to the output of the first phase, which is a Hessenberg matrix. Considerable improvements of the basic QR-method will be possible because of the structure.

Let us start with a simple example:

```
>> A=[1 2 3 4; 4 4 4 4;0  1 -1 1;   0 0 2 3]
A =
     1     2     3     4
     4     4     4     4
     0     1    -1     1
     0     0     2     3
>> [Q,R]=qr(A);
>> A=R*Q
A =
    5.2353   -5.3554   -2.5617   -4.2629
   -1.3517    0.2193    1.4042    2.4549
         0    2.0757    0.6759    3.6178
         0         0    0.8325    0.8696
```

The code corresponds to one step of the basic QR method applied to a Hessenberg matrix. Note that the result is also a Hessenberg matrix. This obsevation is true in general. (The proof is postponed.)

**Theorem 2.2.4** *If the basic QR-method (Algorithm 2.1) is applied to a Hessenberg matrix, then all iterates $A_k$ are Hessenberg matrices.*

A basic QR-step preserves the Hessenberg structure. Hence, the basic QR-method also preserves the Hessenberg structure.

We will now explicitly characterize a QR-step for Hessenberg matrices, which can be used in all subsequent QR-steps.

Fortunately, the QR-decomposition of a Hessenberg matrix has a particular structure which indeed be characterized and exploited. To this end we use the concept of Givens rotations.

**Definition 2.2.5** *Let $c^2 + s^2 = 1$. The matrix $G(i,j,c,s) \in \mathbb{R}^{n\times n}$ corresponding to a Givens rotation is defined by*

$$G(i,j,c,s) := I + (c-1)e_ie_i^T - se_ie_j^T + se_je_i^T + (c-1)e_je_j^T = \begin{bmatrix} I & & & & \\ & c & & -s & \\ & & I & & \\ & s & & c & \\ & & & & I \end{bmatrix} \tag{2.6}$$

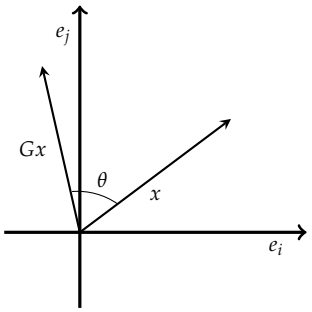



Figure 2.2: A Givens rotation $G = G(i,j,\cos(\theta),\sin(\theta))$ is a "rotation" in the sense that the application onto a vector corresponds to a rotation with angle $\theta$ in the plane spanned by $e_i$ and $e_j$.

We note some properties of Givens rotations:

- $G(i,j,c,s)$ is an orthogonal matrix.
- $G(i,j,c,s)^* = G(i,j,c,-s)$
- The operation $G(i,j,c,s)x$ can be carried out by only modifying two

elements of $x$,

$$G(i,j,c,s)\begin{bmatrix} x_1 \\ \vdots \\ x_{i-1} \\ x_i \\ x_{i+1} \\ \vdots \\ x_{j-1} \\ x_j \\ x_{j+1} \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} x_1 \\ \vdots \\ x_{i-1} \\ cx_i - sx_j \\ x_{i+1} \\ \vdots \\ x_{j-1} \\ sx_i + cx_j \\ x_{j+1} \\ \vdots \\ x_n \end{bmatrix} \tag{2.7}$$

The QR-factorization of Hessenberg matrices can now be explicitly expressed and computed with Givens rotations as follows.

**Theorem 2.2.6** *Suppose $A \in \mathbb{C}^{n\times n}$ is a Hessenberg matrix. Let $H_i$ be generated as follows $H_1 = A$*

$$H_{i+1} = G_i^T H_i, \quad i = 1, \ldots, n-1$$

*where $G_i = G(i, i+1, (H_i)_{i,i}/r_i, (H_i)_{i+1,i}/r_i)$ and $r_i = \sqrt{(H_i)_{i,i}^2 + (H_i)_{i+1,i}^2}$ and we assume $r_i \neq 0$. Then, $H_n = R$ is upper triangular and*

$$A = (G_1 G_2 \cdots G_{n-1}) H_n = QR$$

*is a QR-factorization of $A$.*

Theorem 2.2.6 implies that the $Q$-matrix in a QR-factorization of a Hessenberg matrix can be factorized as a product of $n-1$ Givens rotations.

**Proof** It will be shown that the matrix $H_i$ is a reduced Hessenberg matrix where the first $i-1$ lower off-diagonal elements are zero. The proof is done by induction. The start of the induction for $i = 1$ is trivial. Suppose $H_i$ is a Hessenberg matrix with $(H_i)_{k+1,k} = 0$ for $k = 1, \ldots, i-1$. Note that the application of $G_i$ only modifies the $i$th and $(i+1)$st rows. Hence, it is sufficient to show that $(H_{i+1})_{i+1,i} = 0$. This can be done by inserting the formula for $G$ in (2.6),

$$\begin{aligned} (H_{i+1})_{i+1,i} &= e_{i+1}^T G_i^T H_i e_i \\ &= e_{i+1}^T [I + (c_i - 1)e_i e_i^T + (c_i - 1)e_{i+1}e_{i+1}^T \\ &\qquad - s_i e_{i+1} e_i^T + s_i e_i e_{i+1}^T] H_i e_i \\ &= (H_i)_{i+1,i} + (c_i - 1)(H_i)_{i+1,i} - s_i (H_i)_{i,i} \\ &= c_i (H_i)_{i+1,i} - s_i (H_i)_{i,i} = 0 \end{aligned}$$

Use the definition of $c_i = (H_i)_{i,i}/r$ and $s_i = (H_i)_{i+1,i}/r$

By induction we have shown that $H_n$ is a triangular matrix and $H_n = G_{n-1}^T G_{n-2} \cdots G_1^T A$ and $G_1 \cdots G_{n-1} H_n = A$.

■

The theorem gives us an explicit form for the $Q$ matrix. The theorem also suggests an algorithm to compute a QR-factorization by applying the Givens rotations and reaching $R = H_n$. Since the application of a Givens rotator can be done in $\mathcal{O}(n)$, we can compute QR-factorization of a Hessenberg matrix in $O(n^2)$ with this algorithm. In order to carry

Idea Hessenberg-structure exploitation: Use factorization of $Q$-matrix in terms of product of Givens rotations in order to compute $RQ$ with less operations.

out a QR-step, we now reverse the multiplication of $Q$ and $R$ which leads to

$$A_{k+1} = RQ = Q^* A_k Q = H_n G_1 \cdots G_{n-1}$$

The application of $G_1 \cdots G_{n-1}$ can also be done in $\mathcal{O}$ and consequently

$$\text{the complexity of one Hessenberg QR step} = \mathcal{O}(n^2)$$

The algorithm is summarized in Algorithm 2.2.2. In the algorithm `[c,s]=givens(a,b)` returns $c = a/\sqrt{a^2+b^2}$ and $s = b/\sqrt{(a^2+b^2)}$.

**Algorithm 3** Hessenberg QR algorithm

**Input:** A Hessenberg matrix $A \in \mathbb{C}^{n\times n}$
**Output:** Upper triangular $T$ such that $A = UTU^*$ for an orthogonal matrix $U$.

```
Set A_0 := A
for k = 1, ... do
    // One Hessenberg QR step
    H = A_{k-1}
    for i = 1, ..., n-1 do
        [c_i, s_i] = givens(h_{i,i}, h_{i+1,i})
        H_{i:i+1,i:n} = [ c_i  s_i ; -s_i  c_i ] H_{i:i+1,i:n}
    end for
    for i = 1, ..., n-1 do
        H_{1:i+1,i:i+1} = H_{1:i+1,i:i+1} [ c_i  -s_i ; s_i  c_i ]
    end for
    A_k = H
end for
Return T = A_∞
```

## 2.3 *Acceleration with shifts and deflation*

In the previous section we saw that the QR-method for computing the Schur form of a matrix $A$ can be executed more efficiently if the matrix $A$ is first transformed to Hessenberg form.

The next step in order to achieve a competitive algorithm is to improve the convergence speed of the QR-method. We will now see that the convergence can be dramatically improved by considering a QR-step applied to the matrix formed by subtracting a multiply of the identity matrix. This type of acceleration is called *shifting*.

First note the following result for singular Hessenberg matrices.

**Lemma 2.3.1** *Suppose $H \in \mathbb{C}^{n\times n}$ is an irreducible Hessenberg matrix. Let $QR = H$ be a QR-decomposition of $H$. If $H$ is singular, then the last diagonal element of $R$ is zero,*

$$r_{n,n} = 0.$$

The $R$-matrix in the QR-decomposition of a singular unreduced Hessenberg matrix has the structure

$$R = \begin{bmatrix} \times & \times & \times & \times & \times \\ & \times & \times & \times & \times \\ & & \times & \times & \times \\ & & & \times & \times \\ & & & & 0 \end{bmatrix}.$$

As a justification for the shifting procedure, suppose for the moment that $\lambda$ is an eigenvalue of the irreducible Hessenberg matrix $H$. We will now characterize the consequence of shifting $H$, applying one step of the QR-method and subsequently reverse the shifting:

$$H - \lambda I = QR \tag{2.8a}$$
$$\bar{H} = RQ + \lambda I \tag{2.8b}$$

By rearringing the equations, we find that

$$\bar{H} = RQ + \lambda I = Q^*(H - \lambda I)Q + \lambda I = Q^* H Q.$$

Hence, similar to the basic QR-method, one shifted QR step (2.8) also corresponds to a similarity transformation. They do however correspond to different similarity transformations since the $Q$-matrix is generated from the QR-factorization of $H - \lambda I$ instead of $H$.

The result of (2.8) has more structure.

The shifted QR-step (2.8), where $\lambda$ is an eigenvalue of $H$, generates a reduced hessenberg matrix with structure

$$\bar{H} = \begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ & \times & \times & \times & \times \\ & & \times & \times & \times \\ & & & & \lambda \end{bmatrix}.$$

**Lemma 2.3.2** *Suppose $\lambda$ is an eigenvalue of the Hessenberg matrix $H$. Let $\bar{H}$ be the result of one shifted QR-step (2.8). Then,*

$$\bar{h}_{n,n-1} = 0$$
$$\bar{h}_{n,n} = \lambda.$$

**Proof** The matrix $H - \lambda I$ is singular since $\lambda$ is an eigenvalue of $H$. From Lemma 2.3.1 we have that $r_{n,n} = 0$. The product $RQ$ in (2.8b) implies that the last row of $RQ$ is zero. Hence, $\bar{h}_{n,n} = \lambda$ and $\bar{h}_{n,n-1} = 0$.

■

Since $\bar{H}$ has the eigenvalue $\lambda$, we conclude that the variant of the QR-step involving shifting in (2.8), generates an exact eigenvalue after one step.

## *Rayleigh quotient shifts*

The shifted QR-method can at least in theory be made to give an exact eigenvalue after only one step. Unfortunately, we cannot choose perfect shifts as in (2.8) since we obviously do not have the eigenvalues of the matrix available. During the QR-method we do however have estimates of the eigenvalues. In particular, if the off diagonal elements of $H$ are sufficiently small, the eigenvalues may be estimated by the

The name *Rayleigh qoutient shifts* comes from the fact that there is a connection with the Rayleigh qoutient iteration.

diagonal elements. In the heuristic called the *Rayleigh quotient shift* we select the last diagonal element of $H$ as a shift,

$$\sigma_k = h_{n,n}^{(k-1)}.$$

The shifted algorithm is presented in Algorithm 2.3. The steps involving a QR factorization can also be executed with Givens rotations as in Section 2.2.2. The algorithm features a type of deflation; instead of carrying out QR-steps on $n \times n$ matrices, once $n - m$ eigenvalues have converged we iterate with a matrix of size $m \times m$.

Deflation here essentially means that once an eigenvalue is detected to be of sufficient accuracy, the iteration is continued with a smaller matrix from which the converged eigenvalue has (in a sense) been removed.

**Algorithm 4** Hessenberg QR algorithm with Rayleigh quotient shift and deflation

**Input:** A Hessenberg matrix $A \in \mathbb{C}^{n\times n}$
  Set $H^{(0)} := A$
  **for** $m = n, \ldots, 2$ **do**
    $k = 0$
    **repeat**
      $k = k + 1$
      $\sigma_k = h_{m,m}^{(k-1)}$
      $H_{k-1} - \sigma_k I =: Q_k R_k$
      $H_k := R_k Q_k + \sigma_k I$
    **until** $|h_{m,m-1}^{(k)}|$ is sufficiently small
    Save $h_{m,m}^{(k)}$ as a converged eigenvalue
    Set $H^{(0)} = H_{1:(m-1),1:(m-1)}^{(k)} \in \mathbb{C}^{(m-1)\times(m-1)}$
  **end for**

## 2.4 *Convergence of the QR algorithm*

For didactic reasons we here assume that the matrix $A$ is symmetric. We prove the convergence by introducing a new algorithm: Unnormalized simultaneous iteration (USI). We show convergence of USI in Section 2.4.1 and show its equivalence with the QR-method.

### 2.4.1 *Unnormalized simultaneous iteration*

Let $V^{(0)} \in \mathbb{C}^{n\times n}$ be a starting and let $V^{(k)}$, $k = 1, \ldots$, be defined by

$$V^{(k)} := AV^{(k-1)}.$$

Moreover, we let $Q^{(k)} \in \mathbb{C}^{n\times n}$ and $R^{(k)} \in \mathbb{C}^{n\times n}$ be the QR-factorization of $V^{(k)}$,

$$Q^{(k)}R^{(k)} = V^{(k)}.$$

We now show that the columns of $Q^{(k)}$ converge to the matrix consisting of eigenvectors, corresponding to the eigenvalues ordered by magnitude.

**Theorem 2.4.1 (Convergence of USI)** *Suppose the eigenvalues of $A \in \mathbb{R}^{n\times n}$ are distinct in modulus, and let $|\lambda_1| < \cdots < |\lambda_n|$. Let the columns of $X$ be eigenvectors of $A$ consistently ordered with the eigenvalues. Let $V^{(0)} \in \mathbb{R}^{n\times n}$ be a given matrix such that all leading submatrices of $X^{-1}V^{(0)}$ are non-singular. Let $V^{(k)} := AV^{(0)}$, be the iterates of USI. Then, there exists a sequence of QR-factorizations of $V^{(k)} = Q^{(k)}R^{(k)}$ such that*

$$\|Q^{(k)} - X\| = \mathcal{O}(C^k)$$

*where $C = \max_{\ell=1,\ldots,n-1} |\lambda_\ell|/|\lambda_{\ell+1}|$.*

Theorem 2.4.1 is given in the textbook: TB Thm 28.1

**Proof** Let the LU-factorization of $X^T V^{(0)}$ be

$$LU = X^{-1}V^{(0)} \tag{2.9}$$

which exists since the leading submatrices of $X^{-1}V^{(0)}$ are non-singular, by assumption in the theorem. We can now reformulate the iterates of the unnormlized simultaneous iteration:

$$\begin{aligned} V^{(k)} &= A^k V^{(0)} \\ &= X\Lambda^k X^{-1} V^{(0)} \\ &= X\Lambda^k LU \end{aligned} \tag{2.10}$$

The matrix $\Lambda^k L$ can be further analyzed by using a technical result regarding the QR-factorization of a diagonal matrix and an upper triangular matrix (with identity on the diagonal) explicitly given in appendix Lemma 2.6.2. Essentially it states that if $k$ is large, the Q-matrix in the QR-factorization of $\Lambda^k L$ is close to the identity matrix,

$$(I + \Delta_k)U^{(k)} = \Lambda^k L.$$

where $\Delta_k = \mathcal{O}(C^k) \to 0$. Hence, by inserting into formula (2.10), a we can explicitly write down a QR-factorization of $V^{(k)}$. More precisely, $\hat{Q}^{(k)}\hat{R}^{(k)} = V^{(k)}$ where

$$\hat{Q}^{(k)} = X(I + \Delta_k) = X + \mathcal{O}(C^k).$$

■

### 2.4.2 *USI-QR equivalence and QR-algorithm convergence*

Read TB pages 215-218.

## 2.5 *Further reading*

The variant of the QR-method that we presented here can be improved considerably in various ways. Many of the improvements already carefully implemented in the high performance computing software such

as LAPACK [1]. If the assumption that the eigenvalues are real is removed, several issues must be addressed. If $A$ is real, the basic QR method will in general only converge to a real Schur form, where $R$ is only block triangular. In the acceleration with shifting, it is advantageous to use double shifts that preserve the complex conjugate structure. The speed can be improved by introducing deflation at an early, see aggressive early deflation cite-kressner. Various aspects of the QR-method is given more attention in the text-books. In [3] the QR-method is presented including further discussion of connections with the LR-method and numerical stability. Some of the presentation above was inspired by the online material of a course at ETH Zürich [2]. The book of Watkins [5] has a recent comprehensive discussion of the QR-method, with a presentation which does not involve the basic QR-method.

## 2.6 *Appendix: QR factorization results*

Lemma 2.6.1 implies that a small perturbation in a matrix results in a small perturbation in the associated QR-decomposition

The proof of Lemma 2.6.1 and Lemma 2.6.2 are not a part of the course.

**Lemma 2.6.1 (Smoothness of QR-factorization)** *Suppose $A(\varepsilon) = A(0) + O(\varepsilon) \in \mathbb{R}^{m\times m}$. Then there is a $Q$ such that*

$$Q(\varepsilon) = Q(0) + O(\varepsilon).$$

*and $Q(\varepsilon)R(\varepsilon) = A(\varepsilon)$ is a QR-factorization of $A(\varepsilon)$.*

**Proof** The Householder QR-factorization algorithm involves a finite number of operations with products, sums and division by square roots. Those operations are all continuously differentiable if defined, which they are since otherwise the QR-factorization of $A(0)$ would fail.

■

**Lemma 2.6.2 (QR-factorization for diagonally scaled triangular matrix)** *Suppose $\Lambda = \text{diag}(\lambda_1, \ldots, \lambda_m)$ where $|\lambda_1| < |\lambda_2| < \cdots < |\lambda_m|$. Let $L$ be a lower triangular matrix with unit diagonal elements. Then, there exists a QR-factorization*

$$\Lambda^k L = Q^{(k)} R^{(k)}. \tag{2.11}$$

*such that*

$$Q^{(k)} = I + \Delta_k$$

*where*

$$\Delta_k = \mathcal{O}(\max_{i=1,\ldots,m-1} |\lambda_i|/|\lambda_{i+1}|)^k. \tag{2.12}$$

**Proof** We start by showing (2.11) only for the first column, and proceed with induction. By multiplying (2.11) with $e_1$, we have

$$\Lambda^k L e_1 = Q^{(k)} R^{(k)} e_1 = r_{11}^{(k)} q_1^{(k)} \tag{2.13}$$

Hence,

$$q_1^{(k)} = \Lambda^k L e_1 / r_{11}^{(k)} = e_1 + O(\lambda_2/\lambda_1)^k$$

since (2.13) is the power method for a diagonal matrix. We have selected the sign in the first column of the QR-decomposition such that $q_1^T e_1 \geq 0$. We multiply

$$(I - q_1^{(k)} q_1^{(k)T}) \Lambda^k L = [0, q_2^{(k)}, \ldots, q_n^{(k)}] R^{(k)} = [q_2^{(k)}, \ldots, q_n^{(k)}] R_+^{(k)}$$

$$[0, I_{n-1}] q_1^{(k)} = [0, I_{n-1}] e_1 + O(\lambda_1/\lambda_2)^k = O(\lambda_1/\lambda_2)^k$$

$$\Lambda_-^k L_- + O(\lambda_2/\lambda_1) = Q_-^{(k)} R_-^{(k)}$$

which via smoothness (Lemma 2.6.1) implies that $q_2^{(k)} = e_2 + \mathcal{O}(\lambda_2/\lambda_1)^k + \mathcal{O}(\lambda_3/\lambda_2)^k$. The process is repeated.

■

## 2.7 *Bibliography*

[1] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen. *LAPACK Users' Guide*. Society for Industrial and Applied Mathematics, Philadelphia, PA, third edition, 1999.

[2] P. Arbenz. The course 252-0504-00 G, "Numerical Methods for Solving Large Scale Eigenvalue Problems", ETH Zürich. online material, 2014.

[3] G. Dahlquist and Å. Björck. *Numerical Methods in Scientific Computing Volume II*. Springer, 2014.

[4] J. Francis. The QR transformation. a unitary analogue to the LR transformation. I, II. *Comput. J.*, 4:265–271,332–345, 1961.

[5] D. S. Watkins. *Fundamentals of matrix computations. 3rd ed*. Wiley, 2010.