QR Algorithm

[What happens when you run $[\mathtt{V},\mathtt{D}] = \mathtt{eig}(\mathtt{A})$ in MatLab ?]

The QR algorithm

Let $A \in \mathbb{F}^{n \times n}$ be nonsingular where $\mathbb{F} = \mathbb{R}$ or $\mathbb{F} = \mathbb{C}$. Setting $A_0 = A$, the QR algorithm produces a sequence of matrices $A_1, A_2, \ldots$ by the following process:

for j =1,2, . . .
(i) Find a QR decomposition $A_{j-1} = Q_{j-1} R_{j-1}$ of A_{j-1} where R_{j-1} has positive diagonal entries.
(iii) Set $A_j = R_{j-1} Q_{j-1}$.

Observe that $A_j = Q_{j-1}^* A_{j-1} Q_{j-1}$. Thus *all* the matrices of the sequence $\{A_j\}$ are unitarily (orthogonally if $\mathbb{F} = \mathbb{R}$) similar to A.

The QR algorithm

Let $A \in \mathbb{F}^{n \times n}$ be nonsingular where $\mathbb{F} = \mathbb{R}$ or $\mathbb{F} = \mathbb{C}$. Setting $A_0 = A$, the QR algorithm produces a sequence of matrices $A_1, A_2, \ldots$ by the following process:

for j =1,2, . . .
(i) Find a QR decomposition $A_{j-1} = Q_{j-1} R_{j-1}$ of A_{j-1} where R_{j-1} has positive diagonal entries.
(iii) Set $A_j = R_{j-1} Q_{j-1}$.

Observe that $A_j = Q_{j-1}^* A_{j-1} Q_{j-1}$. Thus *all* the matrices of the sequence $\{A_j\}$ are unitarily (orthogonally if $\mathbb{F} = \mathbb{R}$) similar to A.

Under suitable conditions, the sequence $\{A_j\}$ converges to a Schur form of A.

The QR algorithm

Theorem Given any nonsingular matrix $A \in \mathbb{C}^{n\times n}$, let $A = Q_1 R_1$ and $A = Q_2 R_2$ be two QR decompositions of A. If $A_1 = Q_1^* A Q_1$ and $A_2 = Q_2^* A Q_2$, then there exists a unitary diagonal matrix D such that $A_2 = D^* A_1 D$.

The QR algorithm

Theorem Given any nonsingular matrix $A \in \mathbb{C}^{n \times n}$, let $A = Q_1 R_1$ and $A = Q_2 R_2$ be two QR decompositions of A. If $A_1 = Q_1^* A Q_1$ and $A_2 = Q_2^* A Q_2$, then there exists a unitary diagonal matrix D such that $A_2 = D^* A_1 D$.

Therefore in practice, we do not try to find a QR decomposition with R having positive diagonal entries.

The QR algorithm

The QR iterations may be carried out as follows:

for j =1,2, . . .

(i) Find reflectors $Q_{j-1}^{(1)}, Q_{j-1}^{(2)}, \ldots Q_{j-1}^{(n-1)}$ *such that*

$$Q_{j-1}^{(n-1)} \cdots Q_{j-1}^{(2)} Q_{j-1}^{(1)} A_{j-1} = R_{j-1}$$

(iii) Set $A_j = Q_{j-1}^{(n-1)} \cdots Q_{j-1}^{(2)} Q_{j-1}^{(1)} A_{j-1} Q_{j-1}^{(1)} Q_{j-1}^{(2)} \cdots Q_{j-1}^{(n-1)}$.

This costs $O(n^3)$ flops per iteration. But if we find a unitary (orthogonal if $\mathbb{F} = \mathbb{R}$) matrix Q such that $H = Q^* AQ$ is upper Hessenberg and start the QR algorithm on H instead of A then the cost per iteration comes down to O(n^2) flops.

If A is Hermitian or symmetric, then H is tridiagonal and the cost per iteration reduces further to O(n) flops per iteration.

The QR algorithm

Assume without loss of generality that A is a properly or irreducible upper Hessenberg matrix. Then A_j is of the form

$$A_j = \begin{bmatrix} a_{11}^{(j)} & a_{12}^{(j)} & \cdots & a_{1n}^{(j)} \\ a_{21}^{(j)} & a_{22}^{(j)} & \cdots & a_{2n}^{(j)} \\ & \ddots & \ddots & \vdots \\ & & a_{n,n-1}^{(j)} & a_{nn}^{(j)} \end{bmatrix}.$$

If $\lambda_1, \lambda_2, \ldots, \lambda_n$ be the eigenvalues of A ordered such that

$$|\lambda_1| \geq |\lambda_2| \geq \cdots \geq |\lambda_n|$$

and $|\lambda_k| > |\lambda_{k+1}|$ for some k, $1 \leq k \leq n-1$, then under suitable conditions $|a_{k+1,k}^{(j)}| \to 0$ as $j \to \infty$ linearly at the rate $|\lambda_{k+1}|/|\lambda_k|$.

The QR algorithm

If $k < n - 1$, then putting $a^{(j)}_{k+1,k}$ to 0 when it becomes very small reduces A_j to the form

$$A_j = \begin{bmatrix} A^{(j)}_{11} & A^{(j)}_{12} \\ 0 & A^{(j)}_{22} \end{bmatrix}$$

where $A^{(j)}_{11}$ is $k \times k$ and $A^{(j)}_{22}$ is $(n-k) \times (n-k)$ and the remaining QR iterations can proceed on $A^{(j)}_{11}$ and $A^{(j)}_{22}$ respectively.

The QR algorithm

If $k < n-1$, then putting $a^{(j)}_{k+1,k}$ to 0 when it becomes very small reduces A_j to the form

$$A_j = \begin{bmatrix} A^{(j)}_{11} & A^{(j)}_{12} \\ 0 & A^{(j)}_{22} \end{bmatrix}$$

where $A^{(j)}_{11}$ is $k \times k$ and $A^{(j)}_{22}$ is $(n-k) \times (n-k)$ and the remaining QR iterations can proceed on $A^{(j)}_{11}$ and $A^{(j)}_{22}$ respectively.

In particular, if $k = n-1$, then $|a^{(j)}_{n,n-1}|$ becomes very small for large enough j and consequently $a^{(j)}_{nn}$ approaches an eigenvalue of A. Once $a^{(j)}_{n,n-1}$ is small enough, it is put to zero. The eigenvalue $\lambda_n (\approx a^{(j)}_{nn})$ is extracted and the next QR iterations can occur with A_j replaced by A_j(1:n-1,1:n-1).

The QR algorithm

If $k < n-1$, then putting $a^{(j)}_{k+1,k}$ to 0 when it becomes very small reduces A_j to the form

$$A_j = \begin{bmatrix} A^{(j)}_{11} & A^{(j)}_{12} \\ 0 & A^{(j)}_{22} \end{bmatrix}$$

where $A^{(j)}_{11}$ is $k \times k$ and $A^{(j)}_{22}$ is $(n-k) \times (n-k)$ and the remaining QR iterations can proceed on $A^{(j)}_{11}$ and $A^{(j)}_{22}$ respectively.

In particular, if $k = n-1$, then $|a^{(j)}_{n,n-1}|$ becomes very small for large enough j and consequently $a^{(j)}_{nn}$ approaches an eigenvalue of A. Once $a^{(j)}_{n,n-1}$ is small enough, it is put to zero. The eigenvalue $\lambda_n (\approx a^{(j)}_{nn})$ is extracted and the next QR iterations can occur with A_j replaced by A_j(1:n-1,1:n-1).

The above processes are called *deflation*.