QR iteration

Numerical Linear Algebra 151119 / 041219

Schur (real) NF of a matrix

We consider $A \in \mathbb{R}^{n\times n}$.

Theorem. There exists $Q \in \mathbb{R}^{n\times n}$ orthogonal such that

$$Q^t A Q = \begin{pmatrix} R_{11} & R_{12} & \dots & R_{1n} \\ 0 & R_{22} & \dots & R_{2n} \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & R_{nn} \end{pmatrix}$$

where R_{ii} is either a real number or a 2×2 block matrix having complex conjugate eigenvalues. $\square$

Remark. It is *useful* because the eigenvalues are in the diagonal R_{ii} blocks. Moreover, Q orthogonal implies that Q^tAQ is numerically stable (compare with Jordan NF for example!).

Ex.1 Consider

$$M = \begin{pmatrix} A & B \\ 0 & C \end{pmatrix}$$

Show that if λ is an eigenvalue of M then λ is either an eigenvalue of A or of C.

How to compute the Schur NF? The QR-iteration.

Consider the iteration:

$$\begin{array}{rll} A =: A_1 = & Q_1R_1 & \leftarrow QR\text{-factorization of } A = A_1 \\ A_2 := & R_1Q_1 = Q_2R_2 & \leftarrow \text{ We define } A_2 \text{ and consider its } QR\text{-factorization} \\ A_3 := & R_2Q_2 = Q_3R_3 & \\ & \vdots & \end{array}$$

i.e. $A_k = Q_kR_k$, $A_{k+1} = R_kQ_k$.

Ex.2 Show that $A_{k+1} \sim A_k$ for all $k \geq 1$.

The previous iteration is not very useful from a computational point of view because the QR-factorization of a full matrix requires $\mathcal{O}(n^3)$ flops. But we can apply the iteration scheme to a matrix $A_1 = QAQ^{-1}$ instead (here Q can be any matrix, but we can choose it to be orthogonal, see exercise below). The idea is to take Q so that the computation of the QR-factorization becomes cheaper and that this happens at each step of the iteration. This is the reason why one performs a reduction to upper Hessenberg form H of A and apply the iteration to H: the QR-factorization of an upper Hessenberg matrix requires $\mathcal{O}(n^2)$ flops (e.g. using Givens rotations).

Ex.3 The reduction of A to a Hessenberg form can be computed using of orthogonal transformations. Implement it using the Householder transformations. Then we obtain Q orthogonal so that $H = QAQ^t$ is upper Hessenberg.

Ex.4 Check (numerically) that the upper Hessenberg form is preserved by the QR-iteration.

Convergence? Under "quite general conditions", for example, if

(i) $|\lambda_1| > |\lambda_2| > \cdots > |\lambda_n|$,

(ii) A is diagonalizable by a conjugating matrix X and,

(iii) $\exists$ LU-factorization of X and X^{-1},

the iterates A_k converge to an upper triangular matrix (essential convergence with the eigenvalues in the diagonal).[1]

More general, we say that the QR-iteration for an upper Hessenberg matrix $H_1 = (h_{i,j}^{(1)})_{ij}$ converges if the sequence $H_k = Q_k R_k$, $H_{k+1} = R_k Q_k$, satisfies

$$h_{j+1,j}^{(k)} h_{j,j+1}^{(k)} \to 0, \text{ as } k \to \infty, \text{ for } j = 1, 2, \ldots, n-1.$$

(that is, at least one element out of each adjacent pair of subdiagonal elements must vanish).

The following result gives a necessary and sufficient condition for convergence of the QR-iteration.

Theorem.[2] The QR-iteration applied to H upper Hessenberg (unreduced) matrix converges if, and only if, among each set of eigenvalues of H with equal modulus there are at most two of even multiplicity and at most two of odd multiplicity. $\square$.

Ex.5 Consider a matrix with $\bar{\lambda}_1 = \lambda_2 \in \mathbb{C}$, $\lambda_3 \in \mathbb{R}$, $\bar{\lambda}_4 = \lambda_5 \in \mathbb{C}$ and $\lambda_6 \in \mathbb{R}$. Check numerically that the QR-iteration converges.

Ex.6 Construct an example for which the QR-iteration does not converge to the real Schur NF.

QR-iteration with shifts strategies: the Francis QR iteration

The speed of convergence of the QR-algorithm depends on the ratio $|\lambda_{r+1}/\lambda_r|$. Mainly:

1. If A has eigenvalues of different modulus then the QR-iteration converges exponentially fast (if the conditions for essential convergence hold). In this case the improvement in precision per iterate is linear.

2. If A has eigenvalues of the same modulus and the QR-iteration converges, then it converges very slowly (in a polynomial way indeed). In this case the improvement in precision per iterate is logarithmic.

General rule: a small gap between $|\lambda_r|$ and $|\lambda_{r+1}|$ implies slow convergence.

[1] J.H. Wilkinson, Convergence of the LR, QR and related algorithms, The Computer Jour., 8:(77–84),1965

[2] B. Parlett, Global Convergence of the Basic QR algorithm on Hessenberg Matrices, Mathematics of Computation, 22(104), 1968.

How to speed up convergence? Use shift strategies.

Simple shift strategy. Consider an upper Hessenberg matrix H and $\mu \in \mathbb{R}$. An step of the QR-iteration with simple shift strategy consists in

$$H_k - \mu I = Q_k R_k, \qquad H_{k+1} = R_k Q_k + \mu I,$$

where I is the identity matrix. In this case, the r-th subdiagonal element converges to zeros with rate $|(\lambda_{r+1} - \mu)/(\lambda_r - \mu)|^k$. The main problem is how to choose μ.

Ex.7 Consider the matrix

$$\begin{pmatrix} 9 & -1 & -2 \\ 2 & 6 & -2 \\ 0 & 1 & 5 \end{pmatrix}$$

and compute one-step of the QR-algorithm with shift $\mu = 6$.

One uses different values of μ at each step. A general strategy is to choose $\mu = h_{n,n}$ (this is maybe a good approximation of λ_n).

Drawback of this strategy: If the matrix

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

has complex eigenvalues, then $\mu = h_{n,n}$ is a bad estimate of λ_n.

Double shift strategy. One computes the eigenvalues of G in (1), say a_1 and a_2 (they can be complex). Then one performs a double shift step:

$$H_k - a_1 I = Q_k R_k,\ H_{k+1} = R_k Q_k + a_1 I$$
$$H_{k+1} - a_2 I = Q_{k+1} R_{k+1},\ H_{k+2} = R_{k+1} Q_{k+1} + a_2 I.$$

One has,

$$M = (H_k - a_1 I)(H_k - a_2 I) = (Q_k Q_{k+1})(R_{k+1} R_k)$$

hence the QR-factorization of $M = H_k^2 - \mathrm{tr}(G) H_k + \det(G) I$ has real eigenvalues (independetly of the eigenvalues of G).

The last formula is not a practical one from a numerical point of view because to form M it requires $\mathcal{O}(n^3)$ flops. There is an alternative way (Francis step) to compute one step of the double shift QR-algorithm with $\mathcal{O}(n^2)$ flops.[3]

Ex.8 Use the functions `scipy.linalg.hessenberg` i `scipy.linalg.schur` to compute the Schur decomposition of a matrix. The last one implements the QR-iteration to compute the Schur decomposition.

[3]See Golub-Van Loan (3rd edition, chap.7, p.358) for details