Chapter III

Jacobi's Method

6 The algorithm

Jacobi's method (JACOBI [1846]) is one of the oldest algorithms around. Although it has lost much of its popularity, it is still of interest because it involves some useful ideas that also show up in other approaches. The method is also still of interest for special situations. For instance, it has superior accuracy for the computation of eigenvalues. Also it lends itself for parallel implementation on a variety of parallel architectures.

For our discussion of the Jacobi method, we have used the following sources: Jacobi [1846], Wilkinson [1965, Chapter 5], Parlett [1980, Chapter 9], and Golub and Van Loan [1996, Chapter 8.4].

A symmetric matrix

$$A = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}$$

can be transformed by a similarity transformation with the rotation

$$J = \begin{pmatrix} \cos(\phi) & \sin(\phi) \\ -\sin(\phi) & \cos(\phi) \end{pmatrix}$$

to a diagonal matrix $J^T AJ$. Because the eigenvalues of A are invariant under similarity transformations, the eigenvalues of A appear on the diagonal of A. The fortunate aspect of this similarity transformation is that it is orthogonal, so that, in fact, we only rotate the coordinate system in order to get A in diagonal form.

The off-diagonal entries of $J^T AJ$ are easily computed as

$$(\cos^2(\phi) - \sin^2(\phi))\, a_{12} + \sin(\phi)\cos(\phi)\,(a_{11} - a_{22}),$$

and we see that these entries vanish when

$$\frac{\cos^2(\phi) - \sin^2(\phi)}{\sin(\phi)\cos(\phi)} = \frac{a_{22} - a_{11}}{a_{12}}.$$

Writing for short $\gamma = \cos(\phi)$, $\sigma = \sin(\phi)$, $\sigma = \alpha\gamma$, and $\tau = (a_{22} - a_{11})/(2a_{12})$, we have that

$$\frac{c^2 - \alpha^2\gamma^2}{\alpha\gamma^2} = \frac{1-\alpha^2}{\alpha} = 2\tau,$$

from which α can be solved as $\alpha = -\tau \pm \sqrt{1+\tau^2}$. Since $\sigma^2 + \gamma^2 = 1$, we have that

$$\gamma = \frac{1}{\sqrt{1+\alpha^2}}, \quad \sigma = \alpha\gamma.$$

Note that we have a choice for α. In order to avoid cancellation we select the α of maximum absolute value:

$$\alpha = \mathrm{sign}(\tau)\left(|\tau| + \sqrt{1+\tau^2}\right). \tag{6.1}$$

The idea in JACOBI [1846] was to use this for larger dimensional matrices in order to eliminate large off-diagonal entries. Unfortunately, if we apply this to eliminate the entry in position (i,j), then the similarity transformation has the effect that the rows i and j and the columns i and j are rotated. This may have as effect that previously eliminated entries in these rows or columns are rotated to some nonzero value again. Of course, Jacobi noticed this as well, but he also noticed that if we repeat this procedure, then the transformed matrix becomes more and more diagonally dominant. This was the purpose of Jacobi: he did not carry through this rotation process until the off-diagonal entries became very small, presumably because he was not aware of the relevant aspects of perturbation theory for eigenvalues. Instead, he solved the eigenproblem by trying to solve $Ax = \lambda x$ (Jacobi wrote this as a coefficient system, since matrix notation was not known yet at the time) with corrections for given approximations. Suppose we have some guess μ for the eigenvalue, and some guess y for the eigenvector, then he tried to compute the values ν and z, so that

$$A(y+z) = (\mu+\nu)(y+z).$$

Because he used the Gauss-Jacobi iteration procedure for the solution of z, he needed the matrix to be in diagonallly dominant form and that is what he used the rotations for. In fact, the early Jacobi process, as used by the master himself, may be viewed as a form of preconditioning for the Gauss-Jacobi iteration process. This is an interesting observation, because Jacobi's technique for eigenvalue (and eigenvector) computation has not been picked up by later researchers (until the 1990s); the exception seems to be BODEWIG [1956, p. 280-287], who described Jacobi's process very accurately, and he mentions that it is a powerful technique. We will come back to this later as we will show that Jacobi's original method led to the development of the Jacobi-Davidson method.

Anyway, in Jacobi's approach as well as in the modern iterative approach, the idea is to drive the matrix A to diagonal form by applying successive rotations. In order to annihilate, for instance, the entry in position (ℓ, m), we consider the submatrix

$$A_{\ell m} = \begin{pmatrix} a_{\ell\ell} & a_{\ell m} \\ a_{m\ell} & a_{mm} \end{pmatrix}$$

and compute the rotation, as described above. Of course we have to apply the rotation, embedded as

$$J_{\ell m} = \begin{pmatrix} 1 & & & & & & \\ & \ddots & & & & & \\ & & \gamma & & \sigma & & \\ & & & \ddots & & & \\ & & -\sigma & & \gamma & & \\ & & & & & \ddots & \\ & & & & & & 1 \end{pmatrix},$$

to the entire matrix which rotates the columns ℓ and m and the rows m and ℓ. If we denote the rotated matrix as

$$A_r = J_{\ell m}^T A J_{\ell m},$$

then the sum of squares of the off-diagonal entries has been decreased, through similarity transformation, by $2a_{\ell m}^2$. This can be easily seen for the generic case that $\ell = 1$, $m = 1$. Write

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{12}^T & A_{22} \end{bmatrix},$$

where A_{11} is the leading 2 by 2 part of A.

Then, the result of the Jacobi rotation is

$$\begin{bmatrix} J_{12} & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} A_{11} & A_{12} \\ A_{12}^T & A_{22} \end{bmatrix} \begin{bmatrix} J_{12}^T & 0 \\ 0 & I \end{bmatrix} = \begin{bmatrix} \begin{matrix} * & 0 \\ 0 & * \end{matrix} & J_{12}^T A_{12} \\ A_{12}^T J_{12} & A_{22} \end{bmatrix}.$$

The sum of squares of the off-diagonal entries in the leading block has been decreased by $2a_{12}^2$, and the sum of the squares of the remaining off-diagonal entries is left unchanged, because $J_{1,2}$ is unitary (which leaves the $\| \|_F$-norm unchanged).

7 Modern time developments

The preconditioning part of the method, as an iterative technique to diagonalize a matrix, was reinvented in 1949 by Goldstine, Murray, and von Neumann and described in an initially unpublished manuscript. After Ostrowski had pointed out that this was actually a rediscovery of Jacobi's method, the adapted manuscript was published only in 1959 (GOLDSTINE, MURRAY and VON NEUMANN [1959]). According to WILKINSON [1965, p. 343], Jacobi's method was already being used independently on desk computers at the National Physical Laboratory in 1947. From 1950 on the method got much attention. In the classical process, the maximal off-diagonal entry is annihilated, and this guarantees convergence. Since it is a time-consuming process to determine the maximal entry after each rotation, cyclic procedures were suggested, (cf. GREGORY [1953]). Later, threshold strategies were developed in order to avoid wasting time in trying to annihilate tiny entries (POPE and TOMKINS [1957]). Quadratic convergence for the cyclic Jacobi algorithm was proven, under various assumptions, by HENRICI [1958], SCHÖNHAGE [1961], WILKINSON [1965], and VAN KEMPEN [1966]. This rate of convergence sets in after a number of sweeps (that is $(n-1)n/2$ elementary rotations), but there is no rigorous bound on the number of sweeps required to achieve a specified accuracy. BRENT and LUK [1985] argued that this number is proportional to $\log(n)$, which is in line with Parlett's remark in PARLETT [1980, p. 181], that after three or four sweeps through all the off-diagonal entries convergence is usually very rapid.

The success of the Jacobi method for diagonalizing a symmetric matrix by orthogonal similarity transformations inspired many investigators to find a similar method for non-symmetric matrices. It was quickly realized that the Schur form was the appropriate decomposition. GREENSTADT [1955], was one of the earliest investigators to develop such a method (indeed, Greenstadt made von Neumann aware of this canonical form). Unfortunately, these earliest attempts were not successful. The QR method, that gained more popularity somewhat later, can be viewed, however, as a Jacobi like method, since it produces the Schur form via a sequence of Jacobi similarity transformations.

Rutishauser made an Algol60 implementation of Jacobi's process for symmetric matrices, as a contribution to the collection in WILKINSON and REINSCH [1971]. In the 1960s, the popularity of the method declined, because of the growing popularity first of the Givens method and slightly later, the Householder method: these latter two methods first reduce the symmetric matrix to tridiagonal form and then use an efficient procedure for computing the eigenvalues of the tridiagonal matrix. Interest in the Jacobi method returned with the advent of parallel computers, starting with a paper of SAMEH [1971], and followed by others in the 1980s.

Variants of Jacobi's method were proposed; we mention the extensions for normal matrices, by PAARDEKOOPER [1971], and to nonnormal matrices by EBERLEIN [1970]. The latter process was also part of the Wilkinson and Reinsch collection (the Algol60 procedure **eigen**). In order to improve data locality for distributed memory computers, block Jacobi methods were suggested, see BISCHOF [1987] for a discussion on the influence of the solution of the subproblems on the overall process.

Another interesting feature of the Jacobi method is its superior behaviour with respect to accuracy. Wilkinson analysed this and showed that the relative error in the eigenvalue approximations is eventually reduced to the order of the condition number of A times machine precision. This was perfected by DEMMEL and VESELIČ [1992], who showed that for symmetric positive definite matrices, the condition number of A could be replaced by that of the matrix symmetrically scaled by the diagonal. However, if one is satisfied with less accuracy, then for large scale computations, Jacobi's method is no longer regarded as competitive, not even for modern parallel computers.

CHAPTER IV

Methods for the Complete Eigenproblem

8 The Power Method

Unlike the situation for linear systems solving, there are no truly direct methods for the solution of the eigenproblem, in the sense that in general one cannot compute the eigenvalues (or eigenvectors) exactly in a finite number of floating point operations. The iterative methods come in two different classes, one in which the matrix is driven to diagonal form, or Schur form, by rotations, and one in which this is accomplished by detecting invariant subspaces. This second class of methods is explicitly based on the Power Method. The QR method is the most prominent member of this class; it converges so fast that the complete eigensystem of a matrix can be computed in a modest (but matrix dependent) factor times n^3 floating point operations. This is somehow similar to the situation for the direct methods for solving dense linear systems. The QR method is, for this reason, often referred to as a direct method, in order to distinct it from the essentially iterative subspace projection techniques. These iterative subspace projection techniques attempt to detect partial eigeninformation in much less than $\mathcal{O}(n^3)$ work. They will be described in forthcoming chapters.

Because the Power Method is important as a driving mechanism, although in hidden form, in the fast "direct" methods as well as in the subspace projection methods, we will discuss it in some more detail. The reader should keep in mind, however, that the Power Method is seldom competitive as a stand-alone method; we need it here for a better understanding of the more superior techniques to come.

The Power Method is based on the observation that if we multiply a given vector v by the matrix A, then each eigenvector component in v is multiplied by the corresponding eigenvalue of A.

Assume that A is real symmetric, then it has real eigenvalues and a complete set of orthonormal eigenvectors

$$Ax_k = \lambda_k x_k \quad , \quad \|x_k\|_2 = 1 \quad (k = 1, 2, \cdots, n).$$

We further assume that the largest eigenvalue in modulus is single and that

$$|\lambda_1| > |\lambda_2| \geq \cdots.$$

Now suppose we are given a vector v_1, which can be expressed in terms of the eigenvectors as $v_1 = \sum_i \gamma_i x_i$, and we assume that $\gamma_1 \neq 0$ (that means that v_1 has a nonzero component in the direction of the eigenvector corresponding λ_1).

Given this v_1, we compute Av_1, $A(Av_1)$, ..., and it follows for the Rayleigh quotient of these vectors that

$$\lim_{j\to\infty} \frac{w_j^T A w_j}{w_j^T w_j} = \lambda_1,$$

where $w_j := A^{j-1}v_1$.

The Power Method can be represented by the template given in Fig. 1. The sequence $\theta^{(i)}$ converges, under the above assumptions, to the dominant (in absolute value) eigenvalue A. The scaling of the iteration vectors is necessary in order to prevent over- or underflow. This scaling can be done a little bit cheaper by taking the maximum element of the vector instead of the norm. In that case the maximum element of v_i converges to the largest (in absolute value) eigenvalue of A.

$v = v_1/||v_1||_2$
for $i = 1, 2, ...$ **until convergence**
  $v_{i+1} = Av$
  $\theta^{(i)} = v^T v_{i+1}$
  $v = v_{i+1}/||v_{i+1}||_2$
end

Figure 1: The Power Method for symmetric A

It is not hard to see why the Rayleigh quotients converge to the dominant eigenvalue. We first write w_j in terms of the eigenvectors of A:

$$w_j = \sum_{i\geq 1} \gamma_i \lambda_i^{j-1} x_i$$

$$= \lambda_1^{j-1}\left\{\gamma_1 x_1 + \sum_{i\geq 2}\gamma_i\left(\frac{\lambda_i}{\lambda_1}\right)^{j-1} x_i\right\} .$$

Hence

$$\begin{aligned}\frac{w_j^T A w_j}{w_j^T w_j} &= \lambda_1 \frac{\gamma_1^2 + \sum_{i\geq 2}\gamma_i^2\left(\frac{\lambda_i}{\lambda_1}\right)^{2j-1}}{\gamma_1^2 + \sum_{i\geq 2}\gamma_i^2\left(\frac{\lambda_i}{\lambda_1}\right)^{2j-2}} \\ &= \lambda_1 \frac{\sum_{i\geq 1}\gamma_i^2 P_{2j-1}(\lambda_i)}{\sum_{i\geq 1}\gamma_i^2 P_{2j-2}(\lambda_i)} ,\end{aligned}$$

with $P_i(t) := (t/\lambda_1)^i$.

For unsymmetric matrices the situation is slightly more complicated, since A does not necessarily have an orthonormal set of eigenvectors. Let $A = XJX^{-1}$ denote the reduction to Jordan form, then $A^j v = XJ^jX^{-1}v$. If the largest eigenvalue, in modulus, is real and simple, then we see that this value will dominate and by similar arguments as above, we see convergence in the direction of the corresponding column of X. If the largest eigenvalue is complex, and if A is real, then there is a conjugate eigenpair. If there is only one eigenpair of the same maximal modulus, then the vector $A^j v$ will ultimately have an oscillatory behavior and it can be shown, see WILKINSON [1965, p. 579], that a combination of two successive vectors in the sequence $A^j v$ will converge to a subspace spanned by the two conjugate eigenvectors. The two eigenvectors can then be recovered by a least squares solution approach. For matrices with elementary divisors one can also show that the Power Method will converge, although very slowly, to the eigenvector associated with the eigenvalue of the divisor.

The Power Method, as a 'stand-alone method', received quite some attention until the early 1970s, as the only means to extract eigenvalue information from large matrices. Early attempts to exploit the information hidden in a successive number of iterands, as for instance by Aitken acceleration, were only partially successful. In particular, the method of LANCZOS [1950], which exploited all iterands of the Power Method, was suspect for a long time, mainly because of its not well understood behavior in non-exact arithmetic. We will come back to this later.

If systems with the shifted matrix, like

$$(A - \sigma I)x = b$$

can be solved efficiently, then it is very profitable to use the Power Method with shifts. If one has a good guess σ for the eigenvalue that one is interested in, then one can apply the Power Method with $(A - \sigma I)^{-1}$. Note that it is not necessary to compute this inverted matrix explicitly. In the computation of the next vector $v_{j+1} = (A - \sigma I)^{-1} v_j$ the vector v_{j+1} can be solved from

$$(A - \sigma I) v_{j+1} = v_j .$$

Assume that the largest eigenvalue (in absolute value) is λ_1, and the second largest is λ_2, and that σ is close to λ_1. The speed of convergence now depends on the ratio

$$\frac{|\lambda_1 - \sigma|}{|\lambda_2 - \sigma|},$$

and this ratio may be a good deal smaller than $|\lambda_2|/|\lambda_1|$. Even when the solution of a shifted linear system is significantly more expensive than a matrix vector multiplication with A, the much faster convergence may easily pay off for these additional costs.

We may update the shift as better approximations become available during the iteration process. That is, when we apply the Algorithm in Fig.1 with $(A - \sigma)^{-1}$ at step i, then $\theta^{(i)}$ is an approximation for $1/(\lambda_1 - \sigma)$. This means that the approximation for λ_1 becomes $\sigma + 1/\theta^{(i)}$ and we can use this value as the shift for iteration $i + 1$. This technique is known as *Rayleigh Quotient iteration*. Its convergence is ultimately cubic for symmetric matrices and quadratic for unsymmetric systems.

9 Simultaneous Iteration

We have already seen that for complex conjugate pairs it is necessary to work with three successive vectors in the Power iteration. This suggests that it may be a good idea to start with a block of vectors right from the start. So let us assume that we start with a set of independent vectors $U_k^{(0)} = [u_1, u_2, \ldots, u_k]$, and that we carry out the Power Method with $U_k^{(0)}$, which leads to the computation of

$$U_k^{(i)} = A U_k^{(i-1)}$$

per iteration. If we do this in a straightforward manner, then this will lead to unsatisfactory results because each of the columns of $U_k^{(0)}$ is effectively used as a starting vector for a single vector Power Method, and all these single vector processes will tend to converge towards the dominant eigenvector(s). This will make the columns of $U_k^{(i)}$ highly dependent in the course of the iteration. It is therefore a better idea to try to maintain better numerical independence between these columns and the most common technique for this is to make them orthonormal after each multiplication with A. This leads to the *Orthogonal Iteration* Method, as represented in Fig. 2.

start with orthonormal $U_k^{(1)}$
for $i = 1, .., $ **until convergence**
    $V_k = AU_k^{(i)}$
    orthonormalize the colums of V_k:
        $V_k = Q_k R_k$
    $U_k^{(i+1)} = Q_k$
end

Figure 2: The Orthogonal Iteration Method

The columns of $U_k^{(i)}$ converge to a basis of an invariant subspace of dimension k, under the assumption that the largest k (in absolute value) eigenvalues (counted according to multiplicity) are separated from the remainder of the spectrum. This can easily be seen from the same arguments as for the Power Method. If the eigenvalues are real, and the matrix is real, then the eigenvalues appear along the diagonal of R. For complex eigenvalues the situation is slightly different, we will come back to this later.

10 The QR Method

In order to simplify the derivation of the QR method, we will first assume that A has real eigenvalues, which helps to avoid complex arithmetic in the computation of the Schur forms.

We can apply the Orthogonal Iteration Method with a full set of starting vectors, say $U^{(0)} = I$, in order to try to reduce A to some convenient form. The matrix $U^{(i)}$ in the orthogonal subspace iteration converges to the matrix of Schur vectors, and the matrix $U^{(i)*}AU^{(i)}$ converges to Schur form. After one step of the algorithm: $AU^{(0)} = A = U^{(1)}R^{(1)}$, we can compute $A_1 = U^{(1)*}AU^{(1)}$, which is a similarity transform of the matrix A, and hopefully already a little bit more in the direction of Schur form than A itself. Therefore, it might be a better idea to continue the algorithm with A_1.

Let us define $Q_1 = U^{(1)}$. The matrix A_1 can be computed as

$$A_1 = Q_1^* A Q_1 = R^{(1)} Q_1,$$

which simply reverses the factors of $A_0 := A$.

The next step in the Orthogonal Iteration Method is to compute AQ_1 and factor this as $U^{(2)} R^{(2)}$, but since $A = Q_1 R^{(1)}$, we see that

$$AQ_1 = Q_1 R^{(1)} Q_1.$$

If we now factor $R^{(1)} Q_1$ as

$$R^{(1)} Q_1 = Q_2 R^{(2)}, \tag{10.1}$$

then we see that

$$AQ_1 = U^{(2)} R^{(2)}, \tag{10.2}$$

with $U^{(2)} = Q_1 Q_2$. Obviously, Q_2 gives the orthogonal transformation that corrects Q_1 to $U^{(2)}$. From equation (10.2) we see that

$$R^{(1)} Q_1 = A_1 = Q_1^* A Q_1 = Q_2 R^{(2)},$$

and hence the correction factor Q_2 had also been obtained if we had continued the iteration with A_1. This is a very nice observation, because this correction factor drives the matrix A_1 again further to Schur form, and we can repeat this in an iterative manner.

The nice consequence of working with the transformed matrices A_i is that we can compute the correction matrix Q_{i+1} from the matrix product of the reverted factors Q_i and $R^{(i)}$ from the previous iteration, as we see from (10.1). This leads to the famous QR iteration, which in its basic form is given by the template in Fig. 3.

The matrix A_i converges to Schur form and the product $Q_1 Q_2 \cdots Q_i$ of the correction matrices converges to the matrix of Schur vectors corresponding to A (remember that we are still assuming that all eigenvalues are real, in order to avoid complex arithmetic. We could have dropped this assumption, but then we should have used complex arithmetic; this can be avoided, however).

We can also apply the QR iteration with shifts, and this has very important consequences. Suppose, we apply in one particular step of the algorithm a shift σ close to an eigenvalue λ_j. That is, we do a QR factorization for $A_{i-1} - \sigma I$. If σ is close enough to an eigenvalue λ_j, then this implies that the matrix $A_{i-1} - \sigma I$, which is similar to $A - \sigma I$, is almost singular. This means

```
start with A_0 = A
for i = 1, ..., until convergence
    factor A_{i-1} = Q_i R^(i)
    compute A_i = R^(i) Q_i
end
```

Figure 3: The unshifted QR Method

that the matrix $R^{(i)}$ must be close to singular (since Q_i is orthogonal). If we do the QR factorization with pivoting, in order to get the smallest diagonal element at the bottom position, than this means that one of the columns of Q_i must be close to the smallest singular vector of $A_{i-1} - \sigma I$, or better: that column is almost an eigenvector of A_{i-1}, and consequently, σ is almost an eigenvalue, and that is what we are looking for.

Pivoting is usually not necessary. If a subdiagonal entry of A_i is very small then the eigenvalue computation can proceed for the two separate subblocks of A_i. If it is not small, then apparantly the first $n-1$ columns of A_i are independent (recall that A_i is of upper Hessenberg form) so that (near) dependence must show up through a small diagonal entry at the bottom position in $R^{(i)}$.

The speed of convergence with which the smallest diagonal entry of $R^{(i)}i$ goes to zero, for increasing i, is proportional to $|\lambda_{j+1}-\sigma|/|\lambda_j-\sigma|$, if λ_j is the eigenvalue closest to the shift and λ_{j+1} denotes the eigenvalue that is second closest. This is precisely the speed of convergence for the shift-and-invert Power Method and the surprising result is that we get the speed of shift and invert *without inverting anything.* A real shift can be easily incorporated in the QR algorithm. If we apply a shift in the ith iteration, then we have to QR-factor $A_{i-1} - \sigma I$, as

$$A_{i-1} - \sigma I \Rightarrow Q_i R^{(i)},$$

and, because of the orthogonality of Q_i, it follows that

$$Q_i^*(A_{i-1} - \sigma I)Q_i = R^{(i)}Q_i.$$

However, in the next step we want to continue with A_i (and possibly apply another shift), but that matrix is easily computed, since

$$A_i = Q_i^* A_{i-1} Q_i = R^{(i)}Q_i + \sigma I.$$

This leads to the QR iteration with shifts, given in Fig. 4.

```
start with A_0 = A
for i = 1, .., until convergence
    factor A_{i-1} - σ_i I = Q_i R^{(i)}
    compute A_i = R^{(i)} Q_i + σ_i I
end
```

Figure 4: The QR method with shifts σ_i

We have ignored some important aspects. One of these is that we have assumed that the real matrix, to which we applied the QR method, had real eigenvalues. This was necessary for the convergence of the matrix A_{i-1} (which is similar to A) to Schur form. For more general situations, that is when the real matrix has conjugate pairs of eigenvalues (and eigenvectors), it can be shown that in real arithmetic the matrix converges to *generalized real Schur form*.

Recall that a matrix is in generalized real Schur form if it is block upper triangular and if the diagonal blocks are either 2 by 2 or 1-dimensional. The 2 by 2 blocks along the diagonal of the generalized Schur form represent the complex conjugate eigenpairs of A (and the corresponding eigenvectors can be computed by combining the corresponding columns of the accumulated transformation matrices Q_i).

This leaves the question how to apply complex shifts, since that would lead to complex arithmetic. It can be shown that by combining two successive steps in the QR algorithm a pair of complex conjugate shifts can be used, without explicit complex arithmetic. A pair of such shifts is meaningful, because complex eigenvalues of a real matrix always appear in conjugate pairs. If we want to accelerate convergence for one complex eigenpair, then clearly we also have to treat the conjugate member of this pair.

For further details on the QR method, see, for instance Demmel [1997], Golub and Van Loan [1996], Parlett [1980], and Wilkinson [1965].

11 Efficient Implementation of QR

An important aspect of the QR algorithm is the computational complexity. If we apply it in the explained form then it is quite expensive, because we have to work with dense matrices in each iteration. Also we have, for instance, to compute dense QR factorizations. These costs can be significantly reduced by first bringing the given matrix A, by orthogonal transformations, to upper Hessenberg form: $H = Q^*AQ$. Note that this is equivalent to symmetric tridiagonal form if A is symmetric. The transformation to these forms can be done by Householder reflections or by Givens rotations (the Householder reflections are generally prefererred since they are cheaper).

A key observation is that the QR decomposition of an upper Hessenberg matrix (or a tridiagonal matrix) can be efficiently computed, because only the entries on the lower codiagonal have to be rotated to zero (successive rotations of rows of the matrix A_{i-1}). Another key observation is that the matrix $R^{(i)}Q_i$ is again in upper Hessenberg form (or tridiagonal form). This can be concluded from the observation that $R^{(i)}$ is upper triangular and that Q_i is an upper-Hessenberg matrix. Indeed, Q_i represents nothing else than the orthormalized columns of A_{i-1} and the normalization process preserves Hessenberg structure. Note that if A_i is symmetric tridiagonal, then also $Q_i^* A_{i-1} Q_i = R^{(i)}Q_i$ is symmetric, and therefore, $A_{i+1} = R^{(i)}Q_i$ must be tridiagonal (and $R^{(i)}$ is also tridiagonal).

Another important observation is that the matrix Q_i can be easily computed for the shifted matrix, which leads to an implicit shift strategy. This depends on the *Implicit Q theorem*:

Theorem 3 *Suppose that $Q^TAQ = H$ is an unreduced upper Hessenberg matrix. Then the first column of Q determines uniquely (up to signs) the other columns.*

For a proof of this theorem see, for instance Golub and Van Loan [1996, Chapter 7.4.5] or Demmel [1997, p. 168]. The implicit Q theorem can also be easily deduced from the Arnoldi process.

An implication of this theorem is that if both Q and $\widetilde{Q}$ bring A to upper Hessenberg form, and if $q_1 = \pm\widetilde{q}_1$, then the columns q_i and $\widetilde{q}_i$ of Q, $\widetilde{Q}$, respectively, satisfy the relations

$$q_i = \pm\widetilde{q}_i,$$

that is, the transformation is essentially unique.

The implicit Q theorem leads to a very cheap implementation of the shifts. First recall that we have to construct the QR factorization of the upper Hessenberg matrix $A_i - \sigma I$. Instead of subtracting σ from all diagonal elements, we only subtract it from the first diagonal element, which leaves us the first column of the matrix $A_i - \sigma I$, denoted as

$$a := (a_{1,1}^{(i)} - \sigma, a_{2,1}^{(i)}, 0, \ldots, 0)^T.$$

Clearly, since Q_i represents the orthonormalized columns of A_i, and since Q_i has to be of upper Hessenberg form, the conclusion is that the first column of Q_i is a multiple of a. The normalized vector a:

$$a/\|a\|_2 = (c_1, -s_1, 0, 0, \ldots, 0)^T$$

defines the first Givens rotation that we would have needed to transform $A_i - \sigma I$ to upper triangular form $R^{(i)}$. The product of the Givens rotations necessary for the reduction of $A_i - \sigma I$ to upper triangular form would have led to this first column of Q_i, if we denote the first Givens rotation as

$$G_1 = \begin{bmatrix} c_1 & s_1 & & & \\ -s_1 & c_1 & & & \\ & & 1 & & \\ & & & \ddots & \\ & & & & 1 \end{bmatrix}.$$

Since we also want to compute the resulting Hessenberg matrix $R_i Q_i$, *without computing explicitly* $R^{(i)}$, we proceed as follows. We need that $Q_i^T(A_i - \sigma I)Q_i$ is upper Hessenberg. If we form $G_1^T(A_i - \sigma I)G_1$ then we see that this matrix is almost upper Hessenberg, except for a nonzero entry in position $(3, 1)$. This entry is called a *bulge* in the context of the QR algorithm.

By realizing that Q_i removes the lower codiagonal in $A_i - \sigma I$, we can construct the Givens rotation

$$G_2 = \begin{bmatrix} 1 & & & & \\ & c_2 & s_2 & & \\ & -s_2 & c_2 & & \\ & & & \ddots & \\ & & & & 1 \end{bmatrix}.$$

for the removal of the bulge. Note that the ultimate form of the composite of these rotations tells us that it is in upper Hessenberg form. This leads to this form of G_2. Another rotation for the removal of the bulge, for instance a rotation in the plane spanned by the first and third dimension, would have destroyed the first column of the eventual Q_i. With G_2, the matrix $G_2^T G_1^T (A_i - \sigma I) G_1 G_2$ is again almost upper Hessenberg, now except for a bulge in position $(4, 2)$.

Proceeding in this manner, we see that the *bulge is chased* along the second codiagonal to the bottom of the matrix and eventually disappears. At the end of this process we have the desired Q_i as the product of the $n-2$ Givens rotations:

$$Q_i = G_1 G_2 \cdots G_{n-2}.$$

The idea of starting with only a shift on the first diagonal entry of A_i is referred to as the *implicit shift* procedure. It can also be carried out for the *Francis double shift* in order to capture complex eigenvalues. In that case the bulges appear as 2 by 2 blocks, but they can be handled in a similar way. The process of using successive Givens rotations, which exist because of the *Implicit Q Theorem*, Th. 3, for the further construction of Q_i is referred to as *bulge chasing*. For a detailed coverage of the QR method with these techniques, see Golub and Van Loan [1996], Demmel [1997], Watkins [1991], Watkins [1993], and Watkins [2001].

The costs for the QR method, for computing all eigenvalues and eigenvectors, of a real unsymmetric matrix, are in the order of n^3 arithmetic operations. A crude and conservative upper estimate is $25n^3$ flops, including the initial reduction of A to upper Hessenberg form. This reduces to about $10n^3$ if only the eigenvalues are desired. in that case it is not necessary to accumulate the Q_i's. In the symmetric case the costs are reduced by roughly a factor of 2. These costs make the method only feasable for matrices of order a few thousands at most. For larger matrices, the method is used in combination with a technique that first reduces the large matrices to much smaller matrices (while attempting to preserve the main characteristics of the large matrix, in particular the wanted eigenpairs).

Resuming: The QR method for eigenvalues and eigenvectors of a dense matrix is so effective because of the following ingredients:

1. orthogonalization of the iterated columns in the basic orthogonal simultaneous iteration

2. accumulation of the orthogonal transformations, and their use as similarity transformations (this leads to the QR-RQ step)

3. the effect of shifts and their efficient implementation

4. the initial reduction of A to upper Hessenberg (tridiagonal) form

The QR method has been implemented in the major subroutine libraries, in particular, in LAPACK, ScaLAPACK, IMSL, NAG, and MATLAB.

12 Generalized problems: the QZ method

The generalized eigenproblem

$$Ax = \lambda Bx, \tag{12.1}$$

with $A, B \in \mathbb{C}^{n\times n}$, can be reduced to several canonical forms, but, as in the standard unsymmetric case, the reduction to generalized Jordan form is often unsatisfactory, because of possible ill-conditioning of the transformation matrices. A straightforward approach is the following one. Let us suppose that B is nonsingular, then we can define $y = Bx$, so that the eigenproblem (12.1) reduces to the standard eigenproblem

$$AB^{-1}y = \lambda y. \tag{12.2}$$

This eigenproblem can be solved after a unitary reduction of AB^{-1} to Schur form with the QR method, which leads to

$$Q^*AB^{-1}Q = R, \tag{12.3}$$

with R an upper triangular matrix. For real A and B, the reduction can be realized to generalized real Schur form, which may have 2 by 2 nonzero blocks along the main diagonal.

However, working with AB^{-1} may be very unattractive, in particular when B is near singular. Also the matrix AB^{-1} may be very nonnormal or (close to) defective. This makes the resulting eigenproblem highly sensitive to perturbations. Finally, with this approach we cannot solve important classes of problems for which B is singular.

Moler and Stewart [1973] started with reducing $B^{-1}Q$ to upper triangular form S with a unitary Z:

$$B^{-1}Q = ZS,$$

so that

$$Q^*AZS = R, \quad \text{or} \quad AZ = QRS^{-1}.$$

They made the clever observation that, since RS^{-1} is an upper triangular matrix, and since $BZ = QS^{-1}$, and S^{-1} is also upper triangular, this leads to the interesting result that both A and B can be reduced to Schur form, albeit with two different unitary matrices Q and Z:

$$AZ = QR^A, \quad BZ = QR^B,$$

with R^A and R^B upper triangular matrices.

Moler and Stewart also proved that this reduction is possible for general B, by a more subtle argument. They consider a sequence of invertible $\{B_k\}$ that converges to B, and then they use the Bolzano-Weierstrass theorem for the convergence of a subsequence of the $\{Q_k, Z_k\}$ to the unitary Q, Z that accomplished the simultaneous reduction of A and B.

The implementation of this principle in a robust stable algorithm needs some other clever steps. In order to simplify the description a little bit, we will further assume that A and B are real matrices. It can be shown that A and B can be reduced in a finite number of steps to upper Hessenberg and triangular form by orthogonal transformations $\widetilde{Q}$ and $\widetilde{Z}$:

$$\widetilde{Q}^* A \widetilde{Z} = H \quad \text{and} \quad \widetilde{Q}^* B \widetilde{Z} = R,$$

with H an upper Hessenberg matrix, and R upper triangular. This is done in two major steps. First B is decomposed as $B = QR$, with R upper triangular, so that Q^*B is upper triangular.

We consider now the pencil Q^*A, Q^*B, and in the second step we attempt to reduce Q^*A to upper Hessenberg form, while preserving the upper triangular structure of the other member in the pencil. Note that in order to preserve the eigenvalues, we have to apply all transformations in the same manner to the matrices of the pencil.

This second step is more complicated and we decribe only one generic step of it: the reduction of the first column of Q^*A. With Givens rotations, all nonzero entries in positions $(3,1)$, $(4,1)$, ..., $(n,1)$ are transformed to zero, starting with the last one and working our way up in that column. For instance, the last entry in the first column is annilihated by a Givens rotation $G^L_{n-1,n}$, acting on rows n and $n-1$. This has the effect that $G^L_{n-1,n}Q^*A$ has a zero in position $(n,1)$. However, the matrix $G^L_{n-1,n}Q^*B$ has now a nonzero entry in position $(n, n-1)$. We apply a Givens rotation $G^R_{n-1,n}$ from the right in order to remove that nonzero entry. After this operation we have that $G^L_{n-1,n}Q^*AG^R_{n-1,n}$ has still a nonzero in position $(n,1)$, and $G^L_{n-1,n}Q^*BG^R_{n-1,n}$ is again in upper triangular form.

These two substeps are repeated when moving upwards in the first column until all elements below the second element in the first column have been zeroed. This process is repeated for the columns 2 to $n-2$. Note that all transformations involved are orthogonal, so that after completion of the process we have effectively constructed the required $\widetilde{Q}$ and $\widetilde{Z}$ that transform the pencil to an upper Hessenberg-triangular form.

There is another problem that needs to be conquered and that is singularity of the matrix B. After the reduction of this matrix to Schur form, the (near-) singularity is visible as a zero (or tiny) element on the diagonal. This element is pushed to the last position on the diagonal by Givens rotations again. After this step (see ,for an example of the process, GOLUB and VAN LOAN [1996, Chapter 7.7.5]), the element in the $(n, n-1)$ position of the simultaneously transformed upper Hessenberg form of A, for which the upper Hessenberg structure is preserved, is zero (or tiny). The QZ reduction can then be continued by considering only the leading $n-1$ by $n-1$ blocks of the reduced pencil.

A typical step in the QZ method follows then very much the same pattern as in the QR method. Given an upper Hessenberg-triangular pair A, B, we want to find orthonormal Q and Z, such that

$$Q^*(A - \lambda B)Z = \widetilde{A} - \lambda \widetilde{B},$$

with $\widetilde{A}, \widetilde{B}$ again an upper Hessenberg-triangular pair. The relation with the QR method is that the Q is the same orthogonal matrix that would have transformed the matrix AB^{-1} to $\widetilde{A}\widetilde{B}^{-1}$ after a single QR step. Note that this can be done since AB^{-1} has upper Hessenberg form. A problem is that the effect of Q on B is that it destroys the upper triangular form, but with a cleverly arranged sequence of Givens rotations the fill-in can be removed and this defines Z.

To make things even slightly more complicated: the whole procedure is usually carried out as a double step (including a double shift with the eigenvalues of the lower 2 by 2 subblock of AB^{-1}), so that the whole process can be done in real arithmetic for real matrices A and B. A nice aspect is that it is not necessary to form B^{-1} explicitly. It turns out that only two vectors have to be computed of the form $B^{-1}y$, and this can conveniently be done by solving an upper triangular system (which is a very stable operation in the sense that it leads to solutions of very nearby systems, even if B is ill-conditioned, see, for instance, HIGHAM [1996, Theorem 8.5]). For a detailed description of the implementation of the overall QZ algorithm, see MOLER and STEWART [1973] or GOLUB and VAN LOAN [1996, Chapter 7.7]. For a deeper understanding of the QZ method (and the QR method) it is advisable to read Watkins' paper: WATKINS [2001]. In this paper these methods are considered in a broader framework of more general transformations.

The total costs of the QZ algorithm, for the computation of the eigenvalues only, are about $30n^3$ flops. If one wants the eigenvectors as well, then the final Q and Z need to be evaluated (for the eigenvalues only, the products of all intermediate orthogonal transformations are not explicitly neccessary), which takes another $16n^3$ for Q and about $20n^3$ for Z (GOLUB and VAN LOAN [1996, Chapter 7.7.7].

The (generalized) Schur form $R^A - \lambda R^B$ reveals the eigen structure of the original eigenproblem $A - \lambda B$. The pairs of diagonal elements of R^A and R^B define the eigenvalues λ. Here we have to be careful; it is tempting to take the ratio's of these diagonal elements, and in view of the forgoing presentation this is certainly correct. However, it sometimes hides sensitivity issues. STEWART [1978] has pointed at the asymmetry in the usual presentation of the generalized eigenproblem with respect to the role of A and B. Instead of the form $Ax - \lambda Bx = 0$, we might also have considered $\mu Ax - Bx = 0$, which is equivalent for $\mu = \lambda^{-1}$. If, for instance B is singular, then the second form leads to the conclusion that there is an eigenvalue $\mu = 0$, which is equivalent to concluding that the first form has an eigenvalue $\lambda = \infty$. The important observation is that in this case, in rounding arithmetic, there will be a tiny diagonal element $r^B_{i,i}$ in R^B, so that the corresponding ratio $r^A_{i,i}/r^B_{i,i}$ leads to a very inaccurate eigenvalue approximation for λ, wheras the inverse leads to an accurate approximation for an eigenvalue μ.

The standard presentation for the generalized eigenproblem is quite appropriate for the type of problems that arise, for instance, in mechanical engineering, where A represents the *stiffness* matrix, with information of the differential operator, and B represents the *mass* matrix, which is largely determined by the chosen basis functions. For a well-chosen set of basis functions, the mass matrix is usually well-conditioned and then it is quite natural to consider the eigenvalue problem from the point of view of A. Stewart suggests that for a more general setting of generalized eigenproblems it is more fruitfull to consider the cross-product form

$$\beta Ax = \alpha Bx. \tag{12.4}$$

In this setting an infinite eigenvalue of the problem $Ax = \lambda Bx$ corresponds with a pair α, β, with $\beta = 0$, and, as Stewart points out, this is not essentially different from the situation that $\alpha = 0$. Note that if a pair (α, β) satisfies (12.4), then so does $\tau(\alpha, \beta)$, for any value of the scalar τ. Therefore, we use the following definition for the *generalized eigenvalue* $\langle \alpha, \beta \rangle$:

$$\langle \alpha, \beta \rangle := \{\tau(\alpha, \beta)^T \quad : \quad \tau \in \mathbb{C}\}, \tag{12.5}$$

in other words, the subspace spanned by the vector $(\alpha, \beta)^T$ is identified with the generalized eigenvalue. For an analysis of sensitivity issues of the generalized eigenvalue problem, in terms of generalized eigenvalues, we refer to STEWART and SUN [1990, Chapter VI].

13 Divide-and-Conquer

After the 1970s, the eigenproblem for dense matrices of moderate order seemed to be solved and further improvements were not expected, especially for symmetric dense matrices. However, with the ever increasing demand for higher efficiency and/or better accuracy, things still changed from time to time. CUPPEN [1981] proposed a divide and conquer algorithm for the solution of the eigenproblem for symmetric tridiagonal matrices. The idea was to split the tridiagonal matrix in two blocks, each of about half the original size, plus a rank-one update. Cuppen showed how the eigenproblems for each of the blocks could be combined for the original full problem by exploiting the rank-one update property, which led to the solution of a secular equation.

For our brief discussion of this approach we follow the exposition given in DEMMEL [1997, Chapter 5.3.3]. The key idea is to split a tridiagonal matrix T into two smaller tridiagonal matrices T_1 and T_2 in the following special way:

$$T = \begin{bmatrix} a_1 & b_1 & & & & & & \\ b_1 & \ddots & \ddots & & & & & \\ & \ddots & a_{m-1} & b_{m-1} & & & & \\ & & b_{m-1} & a_m & b_m & & & \\ & & & b_m & a_{m+1} & b_{m+1} & & \\ & & & & b_{m+1} & a_{m+2} & \ddots & \\ & & & & & \ddots & \ddots & b_{n-1} \\ & & & & & & b_{n-1} & a_n \end{bmatrix}$$

$$
= \begin{bmatrix}
a_1 & b_1 & & & & & & \\
b_1 & \ddots & \ddots & & & & & \\
& \ddots & a_{m-1} & b_{m-1} & & & & \\
& & b_{m-1} & a_m - b_m & & & & \\
& & & & a_{m+1} - b_m & b_{m+1} & & \\
& & & & b_{m+1} & a_{m+2} & \ddots & \\
& & & & & \ddots & \ddots & b_{n-1} \\
& & & & & & b_{n-1} & a_n
\end{bmatrix}
+ \begin{bmatrix}
& & & \\
& b_m & b_m & \\
& b_m & b_m & \\
& & &
\end{bmatrix}.
$$

Note that by subtracting b_m from the two interior diagonal elements, we have accomplished that there is only a rank-one correction for the splitting. This can be rewritten as

$$
T = \begin{bmatrix} T_1 & 0 \\ 0 & T_2 \end{bmatrix} + b_m u u^T,
$$

where u is the vector with zeros except for two ones at position m and $m+1$:

$$
u = (0, \ldots, 0, 1, 1, 0, \ldots, 0)^T.
$$

Now suppose that we have solved the eigenproblem (eigenvalues <u>and</u> eigenvectors) for T_1 and T_2, then all we have to do is glue the two together and try to take advantage of the two partial eigendecompositions.

Write $T_1 = Q_1 \Lambda_1 Q_1^T$ and $T_2 = Q_2 \Lambda_2 Q_2^T$. Then we can rewrite T as

$$
\begin{aligned}
T &= \begin{bmatrix} Q_1 \Lambda_1 Q_1^T & 0 \\ 0 & Q_2 \Lambda_2 Q_2^T \end{bmatrix} + b_m u u^T \\
&= \begin{bmatrix} Q_1 & 0 \\ 0 & Q_2 \end{bmatrix} \left(\begin{bmatrix} \Lambda_1 & 0 \\ 0 & \Lambda_2 \end{bmatrix} + b_m v v^T \right) \begin{bmatrix} Q_1^T & 0 \\ 0 & Q_2^T \end{bmatrix}, \qquad (13.1)
\end{aligned}
$$

where

$$
v = \begin{bmatrix} Q_1^T & 0 \\ 0 & Q_2^T \end{bmatrix} u.
$$

Note that, because of the special structure of u, the first part of length m of v is equal to the last column of Q_1^T; the remaining part is equal to the first column of Q_2^T.

Equation (13.1) implies that we have to solve the eigenproblem for the matrix

$$\begin{bmatrix} \Lambda_1 & 0 \\ 0 & \Lambda_2 \end{bmatrix} + b_m vv^T = D + b_m vv^T.$$

For the eigenvalues we use the characteristic equation

$$\det(D + b_m vv^T - \lambda I) = \det(D - \lambda I) \det(I + b_m (D - \lambda I)^{-1} vv^T) = 0,$$

where we have assumed that λ is not an eigenvalue of D.

For the solution of $\det(I + b_m (D - \lambda I)^{-1} vv^T) = 0$, we use the property that $\det(I + xy^T) = 1 + x^T y$, which leads to

$$\begin{aligned} f(\lambda) &= \det(I + b_m (D - \lambda I)^{-1} vv^T) \\ &= 1 + b_m v^T (D - \lambda I)^{-1} v \\ &= 1 + b_m \sum_{i=1}^{n} \frac{v_i^2}{d_i - \lambda}. \end{aligned} \tag{13.2}$$

Consequently, the eigenvalues of T are the roots of the so-called *secular equation*

$$f(\lambda) = 0. \tag{13.3}$$

This equation can be solved very efficiently by Newton's method, exploiting the fact that $f(\lambda)$ is a very steep and monotonic function in between its asymptotes at the points d_i. Finding all eigenvalues of T takes $\mathcal{O}(n^2)$ flops.

The eigenvectors of $D + b_m vv^T$ can be computed from the property that if λ is an eigenvalue, then $(D - \lambda I)^{-1} v$ is the corresponding eigenvector (Cuppen [1981], Demmel [1997]). Since D is a diagonal matrix, it takes only $\mathcal{O}(n)$ flops to compute an eigenvector.

In order to make the divide-and-conquer algorithm really practical, one has to use *deflation*. That is, if $d_i = d_{i+1}$, or if $v_i = 0$, then d_i is an eigenvalue of T. In practice, this is checked with some relative threshold in the order of the machine precision. This process of filtering out converged eigenvalues is called deflation. It makes the overall algorithm really fast, since it was born out in experiments that this deflation situation happens frequently.

Initially, this approach was not seen as a competitive algorithm by itself for general modest dimensional matrices, although Cuppen recognized that his algorithm was asymptotically (much) faster than QR. Further investigations by others were made on account of promising parallel properties. A major problem was that the original algorithm suffered from instabilities, especially for the eigenvectors belonging to close eigenvalues. Some scaling problems were rectified in DONGARRA and SORENSEN [1987], but the "right" implementation, according to DEMMEL[1997, Chapter 5.3.3], was not discovered until 1992 and published in 1995, in GU and EISENSTAT [1995]. Meanwhile, software for this algorithm found its way into LAPACK and ScaLAPACK.

As stated by Demmel again, the divide and conquer approach is now the fastest algorithm for computing all eigenvalues and eigenvectors of a symmetric matrix of order larger than 25; this also holds true for non-parallel computers. If the subblocks are of order greater than 25, then they are further reduced; else, the QR algorithm is used for computing the eigenvalues and eigenvectors of the subblock. For a full treatment of the modern variant of the divide and conquer method, we refer to DEMMEL [1997]. A recent discussion on parallel implementation aspects of this method can be found in TISSEUR and DONGARRA [1999].

There are still niches for other methods for dense symmetric matrices. Wilkinson advocated the bisection method for tridiagonal matrices if only a small subset of the eigenvalues is wanted. Inverse iteration may then be used to determine the corresponding eigenvectors. For the bisection method, based on the Sturm sequence property, the reader may find classical material in WILKINSON [1965, Chapter 5.37ff], a modern treatment of the inverse iteration (considered suspect by many because of the ill-conditioning of the shifted matrix) can be found in PARLETT [1980].