Numer. Math. 52, 279–300 (1988)

Numerische Mathematik
© Springer-Verlag 1988

On Efficient Implementations of Kogbetliantz's Algorithm for Computing the Singular Value Decomposition

J.P. Charlier, M. Vanbegin, and P. Van Dooren
Philips Research Laboratory, Av. Van Becelaere 2, Box 8, B-1170 Brussels, Belgium

Summary. In this paper we compare several implementations of Kogbetliantz's algorithm for computing the SVD on sequential as well as on parallel machines. Comparisons are based on timings and on operation counts. The numerical accuracy of the different methods is also analyzed.

Subject Classifications: AMS(MOS): 65F15; CR: G1.3

1. Introduction

In this paper we analyze Kogbetliantz's algorithm [10, 11] for computing the singular value decomposition (SVD) of a real or complex $m \times n$ matrix A (where we assume $m \geq n$):

$$A = U \cdot \begin{pmatrix} \Sigma \\ 0 \end{pmatrix} \cdot V^*. \tag{1.1}$$

Here U and V are unitary (resp. orthogonal) and * denotes the conjugate transposed (resp. transposed) in the complex (resp. real) case and Σ is diagonal and real. This algorithm has received a great deal of attention recently because of its efficiency as a parallel algorithm [1, 2] and also because of its possible extensions to various other decompositions [9, 18]. In this paper we take a closer look at the implementation details of the method (whose basics are given in Sect. 2) and present a few variants which may compare favourably with the "standard" form in terms of speed on a parallel or sequential machine.

For a Hermitian (symmetric) matrix one can derive (1.1) from the eigenvalue decomposition obtained by Jacobi's cyclic method (see e.g. [2]). A major advantage here is that one has to process only half of the matrix throughout the iterations. On a sequential machine or pipeline machine this then reduces the amount of computing time roughly by a factor 2, while on an array of processors it reduces the number of processors approximately by a factor 2. For a general matrix similar savings are obtained when using a variant [13, 18] where a preliminary QR decomposition with column pivoting is performed:

$$A = Q \cdot \begin{pmatrix} R \\ 0 \end{pmatrix} \cdot E \tag{1.2}$$

(where thus Q is a unitary matrix and E a permutation matrix), in order to reduce the $m \times n$ matrix A to a triangular matrix R of dimension n. We assume here that the diagonal elements of R are real (and positive if requested) in the above decomposition, which can always be obtained by an appropriate choice of Q. On a regular array of processors (e.g., systolic array) column pivoting should be left out because it can not be implemented efficiently on such an architecture. A QR decomposition *without* pivoting, on the other hand, can be implemented efficiently in $O(m+n)$ steps using n^2 processors [7, 13]. Here also the diagonal elements of R can be chosen real. It turns out that Kogbetliantz's cyclic method (which is in fact very similar to Jacobi's cyclic method for symmetric matrices and indeed reduces to it when A is symmetric) can be reorganized such that at each stage of the recursion one needs only to store and process a triangular matrix. In Sect. 3 we recall the reorganization of the classical "row by row" scheme that achieves this [13].

On this triangular matrix R one can now still define different ways of implementing the "reordered" Kogbetliantz's algorithm. Indeed one can use different schemes to implement the elementary 2×2 SVD's which are the basic building blocks of the algorithm. In Sect. 4 we present some possible variants of computing the SVD of a 2×2 matrix exactly or approximately, and in Sect. 5 we have a closer look at their stability properties. Each of these can now be "plugged in" to obtain a new variant of Kogbetliantz's algorithm. The speed of these different methods on sequential and parallel machines is then compared in the last Section before the concluding remarks.

As a result of this we finally recommend a "triangular approximated" Kogbetliantz algorithm as the algorithm which nicely combines the advantages of being faster, more flexible and easier to implement.

2. Kogbetliantz's Algorithm

Kogbetliantz's method consists of generating a sequence of matrices $A^{(k)}$ as follows

$$\begin{array}{lll} U^{(0)} := I_m & V^{(0)} := I_n & A^{(0)} := A \\ U^{(k+1)} := U_k \cdot U^{(k)} & V^{(k+1)} := V_k \cdot V^{(k)} & A^{(k+1)} := U_k \cdot A^{(k)} \cdot V_k^* \end{array} \tag{2.1}$$

such that the "off-norm" of the matrix $A^{(k)}$:

$$\operatorname{off}(A) = \Big\{ \sum_{i \neq j} |a_{i,j}|^2 \Big\}^{\frac{1}{2}} \tag{2.2}$$

decreases and eventually becomes negligible, i.e. of the order of δ, the relative precision of the machine one is working with. In (2.1) the updating transformations U_k and V_k are chosen as real or complex elementary rotations acting only on the pair of rows and columns (i_k, j_k), in order to yield zeros in the positions (i_k, j_k) and (j_k, i_k):

$$A^{(k+1)} := U_k \cdot A^{(k)} \cdot V_k^* = \begin{pmatrix} x & \dots & x & \dots & x & \dots & x \\ \vdots & & \vdots & & & & \vdots \\ x & \dots & x & \dots & 0 & & x \\ \vdots & & \vdots & & & & \vdots \\ x & \dots & 0 & \dots & x & & x \\ \vdots & & \vdots & & & & \vdots \\ x & \dots & x & \dots & x & & x \end{pmatrix}. \tag{2.3}$$

Let us denote such an elementary rotation acting on the rows (or columns) i and j (where we always assume $i<j$) by $G_{i,j}(\theta, e)$[1], i.e.

$$G_{i,j}(\theta, e) = \begin{pmatrix} 1 & \dots & 0 & \dots & 0 & \dots & 0 \\ \vdots & & \vdots & & \vdots & & \\ 0 & \dots & \cos\theta & \dots & e\cdot\sin\theta & \dots & 0 \\ \vdots & & \vdots & & \vdots & & \\ 0 & \dots & -\bar{e}\cdot\sin\theta & \dots & \cos\theta & \dots & 0 \\ \vdots & & \vdots & & \vdots & & \\ 0 & \dots & 0 & \dots & 0 & \dots & 1 \end{pmatrix}, \quad |e| = 1 \tag{2.4}$$

where $\bar{e}$ denotes the complex conjugate of e. Notice here that a general 2×2 unitary matrix has a more general form than $G_{i,j}(\theta, e)$ (see e.g. [6]) but it can be reduced to it by merely a diagonal unitary scaling e.g. of the rows. Such a scaling of course does not affect the off-norm (2.2) of the matrix nor its zero pattern in (2.3), but meanwhile simplifies formulas (2.5) and (2.6) given below. Thus for (2.3) one takes $U_k := G_{i_k, j_k}(\phi_k, c_k)$ and $V_k := G_{i_k, j_k}(\psi_k, d_k)$ with ϕ_k, ψ_k, c_k and d_k satisfying [6]:

$$\begin{aligned} &-d_k\cdot\cos\phi_k\cdot\sin\psi_k\cdot a^{(k)}_{i_k,i_k} + \cos\phi_k\cdot\cos\psi_k\cdot a^{(k)}_{i_k,j_k} \\ &\quad - c_k\cdot d_k\cdot\sin\phi_k\cdot\sin\psi_k\cdot a^{(k)}_{j_k,i_k} + c_k\cdot\sin\phi_k\cdot\cos\psi_k\cdot a^{(k)}_{j_k,j_k} = 0 \end{aligned} \tag{2.5}$$

$$\begin{aligned} &-\bar{c}_k\cdot\sin\phi_k\cdot\cos\psi_k\cdot a^{(k)}_{i_k,i_k} - \bar{c}_k\cdot\bar{d}_k\cdot\sin\phi_k\cdot\sin\psi_k\cdot a^{(k)}_{i_k,j_k} \\ &\quad + \cos\phi_k\cdot\cos\psi_k\cdot a^{(k)}_{j_k,i_k} + \bar{d}_k\cdot\cos\phi_k\cdot\sin\psi_k\cdot a^{(k)}_{j_k,j_k} = 0 \end{aligned} \tag{2.6}$$

in order to "annihilate" the elements $a^{(k+1)}_{i_k,j_k}$ and $a^{(k+1)}_{j_k,i_k}$ in (2.3). Notice here that when n is strictly smaller than m, $a^{(k)}_{i_k,j_k}$ and $a^{(k)}_{j_k,j_k}$ will be non-existent for $j_k > n$. In this case V_k is in fact not computed and ψ_k taken to be 0 in (2.5) and (2.6) [1]. A method to solve these equations in the general complex case is e.g. given in [6]. Here we will only be interested in special forms of these equations. This, however, still allows us to treat the general complex case as will be shown later on.

[1] The letter G is used to denote these rotations because they were popularized by the work of Givens

First, when A is real, all computations can be made real by choosing $c_k = d_k = 1$ for all k. One could also choose $c_k = -1$ and or $d_k = -1$ but this does not affect any of the properties of the obtained algorithm, which is why we do not consider these possibilities anymore in the sequel. Equations (2.5) and (2.6) then become:

$$\begin{aligned} -\cos\phi_k\cdot\sin\psi_k\cdot a^{(k)}_{i_k,i_k} + \cos\phi_k\cdot\cos\psi_k\cdot a^{(k)}_{i_k,j_k} \\ -\sin\phi_k\cdot\sin\psi_k\cdot a^{(k)}_{j_k,i_k} + \sin\phi_k\cdot\cos\psi_k\cdot a^{(k)}_{j_k,j_k} = 0 \end{aligned} \tag{2.7}$$

$$\begin{aligned} -\sin\phi_k\cdot\cos\psi_k\cdot a^{(k)}_{i_k,i_k} - \sin\phi_k\cdot\sin\psi_k\cdot a^{(k)}_{i_k,j_k} \\ +\cos\phi_k\cdot\cos\psi_k\cdot a^{(k)}_{j_k,i_k} + \cos\phi_k\cdot\sin\psi_k\cdot a^{(k)}_{j_k,j_k} = 0. \end{aligned} \tag{2.8}$$

When A is Hermitian one usually chooses $c_k = d_k$ and $\phi_k = \psi_k$ for all k since then $U_k = V_k$ and hence all $A^{(k)}$ are also Hermitian. The diagonal elements that are then eventually obtained in $A^{(\infty)}$ are the *eigenvalues* of A instead of its *singular values*. For Hermitian matrices these are equal up to a sign and one then often prefers to compute the eigenvalue decomposition since symmetry allows one to process only half of the matrix A (see next section for more details on this) and involves only one transformation matrix U. Equations (2.5) and (2.6) thus become identical and moreover boil down to the real equation:

$$\cos\phi_k\cdot\sin\phi_k\cdot(a^{(k)}_{i_k,i_k} - a^{(k)}_{j_k,j_k}) - (\cos^2\phi_k - \sin^2\phi_k)\cdot|a^{(k)}_{i_k,j_k}| = 0 \tag{2.9}$$

when choosing $c_k = a^{(k)}_{i_k,j_k}/|a^{(k)}_{i_k,j_k}|$, since $a^{(k)}_{i_k,i_k}$ and $a^{(k)}_{j_k,j_k}$ are already real. Notice that when A is real *and* Hermitian (i.e. symmetric) then one can drop the modulus signs in the above equation. This algorithm then also becomes identical to Jacobi's method for diagonalizing a real symmetric matrix A [2].

Finally in the upper triangular case with *real* diagonal elements, one again reduces (2.5) and (2.6) easily to *real* equations by choosing again $c_k = d_k = a^{(k)}_{i_k,j_k}/|a^{(k)}_{i_k,j_k}|$. Indeed the Eq. (2.5) resp. (2.6) then become (up to the nonzero scalar c_k, resp. $\bar{c}_k$):

$$-\cos\phi_k\cdot\sin\psi_k\cdot a^{(k)}_{i_k,i_k} + \cos\phi_k\cdot\cos\psi_k\cdot|a^{(k)}_{i_k,j_k}| + \sin\phi_k\cdot\cos\psi_k\cdot a^{(k)}_{j_k,j_k} = 0 \tag{2.10}$$

$$-\sin\phi_k\cdot\cos\psi_k\cdot a^{(k)}_{i_k,i_k} - \sin\phi_k\cdot\sin\psi_k\cdot|a^{(k)}_{i_k,j_k}| + \cos\phi_k\cdot\sin\psi_k\cdot a^{(k)}_{j_k,j_k} = 0. \tag{2.11}$$

Here also one can drop the modulus signs in (2.10) and (2.11) for the real triangular case. Unlike for the previous two cases the *structure* of the matrices $A^{(k)}$ (i.e. here their triangularity) is not necessarily preserved in subsequent steps. In the next section we recall how this can be ensured via an appropriate choice of the index sets (i_k, j_k) [13, 18].

It is now easily shown that because of (2.1–4) one has for the three above schemes:

$$\mathrm{off}(A^{(k+1)})^2 = \mathrm{off}(A^{(k)})^2 - |a^{(k)}_{i_k,j_k}|^2 - |a^{(k)}_{j_k,i_k}|^2 \tag{2.12}$$

and that linear convergence of this algorithm is guaranteed [5, 6, 15] at least when the angles ϕ_k and ψ_k remain strictly inside the interval $(-\pi/2, +\pi/2)$ and the $(2m-n-1)\cdot n/2$ possible pairs of indices (i_k, j_k) are scanned in a cyclic ordering, by rows or by columns. E.g., in the cyclic "row by row" method, all possible pairs have a turn in what is called a "sweep", in the following order:

```
for $i_k = 1, n$
    for $j_k = i_k + 1, m$
        $A^{(k+1)} := G_{i_k, j_k}(\phi_k, c_k) \cdot A^{(k)} \cdot G_{i_k, j_k}(\psi_k, d_k)^*$
        $k = k + 1$
    end $j_k$;
end $i_k$;                                                        (2.13)
```

In practice one observes that with Kogbetliantz's algorithm the off-norm of $A^{(k)}$ in fact decreases *quadratically* from one sweep to another (this is also *proved* in [19] for the case that A has *no* repeated singular values). One thus only needs a moderate number of sweeps in order to obtain a δ-small off-norm and hence have a δ-close estimate of the singular values of A.

3. A Parallel Scheme

In this section we recall some of the features of Jacobi's and Kogbetliantz's methods which made them such popular algorithms for implementation on a machine with some sort of parallel architecture.
Using the different schemes (2.7–11) one proceeds as follows for the computation of the SVD of a general complex or real matrix A. When A is Hermitian (resp. symmetric) we compute the Hermitian (resp. symmetric) eigenvalue decomposition:

$$A = U \cdot \Lambda \cdot U^* \tag{3.1}$$

(where Λ is always real) with Jacobi's method. The singular value decomposition (1.1) is then derived from (3.1) by merely extracting the signs of the eigenvalues into a diagonal sign matrix which is e.g., absorbed into the right factor U^* thus constituting V^*. When A has not such a structure, we first compute the triangular decomposition (1.2) and then proceed with the triangular version of Kogbetliantz's algorithm in order to obtain:

$$R = U \cdot \Sigma \cdot V^* \tag{3.2}$$

which together with (1.2) then finally yields the requested SVD of A. Notice that in both cases only a triangle with real diagonals must be processed since for the Hermitian (symmetric) case $A^{(k)}$ is completely determined by its upper triangular part. A second advantage is that in both cases $m = n$ which avoids the need of "one-sided" transformations and thus reduces the amount of work needed for one sweep to $(n-1) \cdot n/2$ rotations on rows and columns of length $n/2$ on the average (provided one works only on a triangle). This then turns out to be advantageous as well on sequential machines as on parallel machines. As now shown below, the possibility to work only on a triangle depends on an appropriate ordering of the index pairs (i_k, j_k) within one sweep.

For sequential machines one of the most popular orderings of rotations used to be the "row-by-row" scheme mentioned above in (2.13). Unfortunately, this scheme is not that appropriate for e.g. systolic arrays. For such architectures it is important to perform transformations on adjacent rows and columns only, i.e. on index pairs of the type $(i_k, i_k + 1)$. While apparently this is not compatible with the requirement that *all* index pairs (i_k, j_k), $i_k < j_k$ should have a turn, it is shown e.g. in [13, 20] that the following scheme (from now on we assume $m = n$):

```
for $i = 1, n-1$
    for $i_k = 1, n-i$
        $A^{(k+1)} := P_{i_k} \cdot G_{i_k, i_k+1}(\phi_k, c_k) \cdot A^{(k)} \cdot G_{i_k, i_k+1}(\psi_k, d_k)^* \cdot P_{i_k}^*$
        $k = k+1$
    end $i_k$;
end $i$;                                                        (3.3)
```

(where P_i is the permutation matrix that swaps columns (or rows) i and $i+1$) is in fact a relabelling of the row-by-row scheme (2.13). Indeed, the permutations in (3.3) "shuffle" the rows and columns of $A^{(k)}$ in such a way that each index pair (i_k, j_k) in (2.13) becomes a pair of the type (i_k, i_k+1) in (3.3) when it is its turn to be processed (see e.g. [20] for more details). A side effect of this relabelling is that after one sweep the order of rows and columns in (3.3) is the reverse of that in (2.13). If one now performs the following "reversed" sweep:

```
for $i = 1, n-1$
    for $i_k = n-1, i\ step -1$
        $A^{(k+1)} := P_{i_k} \cdot G_{i_k, i_k+1}(\phi_k, c_k) \cdot A^{(k)} \cdot G_{i_k, i_k+1}(\psi_k, d_k)^* \cdot P_{i_k}^*$
        $k = k+1$
    end $i_k$;
end $i$;                                                        (3.4)
```

subsequently to (3.3) then the final matrix $A^{(k)}$ obtained after two sweeps of (2.13) is identical to that obtained after one sweep (3.3) followed by one sweep (3.4). Since the above method is essentially "equivalent" [14] to the cyclic by rows method, convergence of the latter also implies convergence of the former [15].

Notice that the relabelled schemes (3.3) and (3.4) now contain *only* transformation between adjacent rows and columns. They also have the same complexity as the previous scheme (2.13) because the permutations involved are performed simultaneously with the rotations, this at no extra computational cost. A direct consequence of this is that for a *sequential* machine the application of the elementary rotations and permutations $P_{i_k} \cdot G_{i_k, i_k+1}(\phi_k, c_k)$ and $G_{i_k, i_k+1}(\psi_k, d_k)^* \cdot P_{i_k}^*$ can easily be performed by two "BLAS-like"[2] calls (one for the $n - i_k - 1$ rotations ϕ_k in rows i_k and i_k+1 and one for the $i_k - 1$ rotations ψ_k in columns i_k and i_k+1) resulting in a total of $4n-8$ "flops". On a triangular matrix $A^{(k)}$ processed row-by-row this would require more BLAS calls since nonzero elements do not remain adjacent there [18] while on a full matrix $A^{(k)}$ this would require twice as many flops. Moreover, while performing these rotations one scans here adjacent elements in the matrix $A^{(k)}$ which reduces the number of "page faults" [12] in comparison to the row-by-row scheme. Roughly the same comments also hold for array processors or computers with pipeline architecture [4]. For systolic arrays, the reordering performed in (3.3–4) becomes essential in order

[2] This refers to the Basic Linear Algebra Subroutines package described in [12]

to avoid *transmission* delays between elements that have to be present in one processor at a given time [2]. It is shown e.g. in [20] that the above reordered scheme can be implemented on a square grid of processors (see Fig. 4.3 in [20]). There each processor of the grid contains four adjacent elements of the matrix. After a processor has finished generating or applying a pair of elementary rotations, it passes its transformed elements to neighboring processors. For more details we refer to [17, 20] and to [2] where an alternative reordering is proposed. The use of triangular matrices moreover allows one to reduce the amount of processors needed in the array by approximately $n^2/2$ [13] since only the processors above the diagonal are processing nonzero elements. Finally, there are also numerical advantages to the use of the "triangular" version of Kogbetliantz's algorithm. One of them is the possibility to develop good approximate 2×2 SVD's as discussed in the next section.

4. Elementary 2×2 SVD's

For an array of processors, the computational cost of the parallel Kogbetliantz algorithm is essentially determined by the 2×2 SVD's performed by the processors on the diagonal. Indeed, each of these requires in general three square roots for the construction of two elementary rotations. A modified Jacobi algorithm is proposed in [16] using approximate decompositions where one only needs to compute two square roots. This section deals with the extension of those results to Kogbetliantz's algorithm (*approximation* 1 and 2 below). Moreover a new formula is introduced (*approximation* 3) because of its better properties of accuracy and, at least in the triangular case, of convergence.

A priori the decrease of computing time for elementary rotations is worthwhile as far as the possible associated increase in number of sweeps does not compensate it. A compromise has to be reached between the accuracy of a given approximation and its computational complexity. On a sequential machine the cost of the 2×2 SVD's is not as crucial. However an approximate scheme can be of interest there as well. These points will be illustrated in Sect. 6.

For convenience, one 2×2 SVD will be written in the abbreviated form:

$$\begin{pmatrix} a'_{11} & a'_{12} \\ a'_{21} & a'_{22} \end{pmatrix} := \begin{pmatrix} -\bar{c}\cdot\sin\phi & \cos\phi \\ \cos\phi & c\cdot\sin\phi \end{pmatrix} \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} d\cdot\sin\psi & \cos\psi \\ \cos\psi & \bar{d}\cdot\sin\psi \end{pmatrix} \tag{4.1}$$

which includes the column and row exchanges of (3.3) and (3.4)[3]. The exact decomposition gives $a'_{12}=0$ and $a'_{21}=0$ by a proper choice of ϕ and ψ. The convergence of Kogbetliantz's algorithm does not require it in general. As shown in [6], the following condition:

$$|a'_{12}|^2+|a'_{21}|^2 \leqq t(|a_{12}|^2+|a_{21}|^2) \tag{4.2}$$

with $0\leqq t<1$, is sufficient for linear convergence. In the sequel we use this to discuss the convergence of the approximate schemes.

[3] In case of a real matrix, this section is to be read with a_{12} instead of $|a_{12}|$ and $\bar{a}_{12}$

4.1. Upper Triangular Case with Real Diagonal Elements

The analog of (2.10) is the equation:

$$a'_{21} := \frac{a_{12}}{|a_{12}|} \cdot (-\cos\phi \cdot \sin\psi \cdot a_{11} + \cos\phi \cdot \cos\psi \cdot |a_{12}| + \sin\phi \cdot \cos\psi \cdot a_{22}) = 0 \tag{4.3}$$

to be satisfied exactly, such that the upper triangular structure is preserved, while (2.11) corresponds to:

$$a'_{12} := \frac{\bar{a}_{12}}{|a_{12}|} \cdot (-\sin\phi \cdot \cos\psi \cdot a_{11} - \sin\phi \cdot \sin\psi \cdot |a_{12}| + \cos\phi \cdot \sin\psi \cdot a_{22}) = 0 \tag{4.4}$$

to be satisfied exactly or approximately (if only (4.2) is requested). From (4.3) $\tan\psi$ (resp. $\tan\phi$) can be expressed as a function of $\tan\phi$ (resp. $\tan\psi$). Then (4.4) leads to an exact or approximate expression for $\tan\phi$ (resp. $\tan\psi$). Finally:

$$\cos\psi = \frac{1}{\sqrt{1+\tan^2\psi}}, \qquad \sin\psi = \cos\psi \cdot \tan\psi \tag{4.5}$$

and

$$\cos\phi = \frac{1}{\sqrt{1+\tan^2\phi}}, \qquad \sin\phi = \cos\phi \cdot \tan\phi \tag{4.6}$$

allow to compute a'_{12} by (4.4), and the new diagonal elements by:

$$a'_{11} = \sin\phi \cdot \sin\psi \cdot a_{11} - \sin\phi \cdot \cos\psi \cdot |a_{12}| + \cos\phi \cdot \cos\psi \cdot a_{22}, \tag{4.7}$$

$$a'_{22} = \cos\phi \cdot \cos\psi \cdot a_{11} + \cos\phi \cdot \sin\psi \cdot |a_{12}| + \sin\phi \cdot \sin\psi \cdot a_{22}. \tag{4.8}$$

The latter reduces to:

$$a'_{11} = a_{22} \frac{\cos\psi}{\cos\phi}, \qquad a'_{22} = a_{11} \frac{\cos\phi}{\cos\psi} \tag{4.9}$$

by use of (4.3) only (i.e. no matter if (4.4) is satisfied exactly or approximately).

Remark. It follows from (4.5–8) that the diagonal elements remain real, and positive if taken so initially since then all quantities in (4.9) are positive. This is an elegant property of this scheme and ensures convergence to positive diagonal values, which was not certified for earlier schemes [6].

Let us detail the situation where an expression for $\tan\psi$ is obtained from (4.3):

$$\tan\psi = \frac{a_{22} \cdot \tan\phi + |a_{12}|}{a_{11}}. \tag{4.10}$$

Then $\tan\phi$ is solution of:

$$-\sigma \cdot \tan^2\phi - \tan\phi + \sigma = 0, \tag{4.11}$$

where σ is defined as:

$$\sigma = \frac{a_{22}\,|a_{12}|}{a_{11}^2 - a_{22}^2 + |a_{12}|^2}. \tag{4.12}$$

The exact formula is chosen to be the smaller root of (4.11):

$$\tan\phi = \frac{2\sigma}{1+\sqrt{1+4\sigma^2}}. \tag{4.13}$$

If an approximation of that solution is considered, the ratio of successive off-diagonal elements is easily derived from (4.4) and (4.10):

$$\frac{a'_{12}}{\bar{a}_{12}} = \frac{a_{22}}{a_{11}} \frac{-\sigma\cdot\tan^2\phi - \tan\phi + \sigma}{\sigma} \cos\phi\cdot\cos\psi. \tag{4.14}$$

This expression is now used to compare three different approximations of (4.13). We first assume $|a_{22}| \leqq |a_{11}|$ the reason of which will become clear in the sequel. *Approximation* 1 corresponds to: $\tan\phi = \sigma$. From (4.10) one has then:

$$|\cos\psi| = \frac{|a_{11}|\cdot|\sin\psi|}{|a_{22}\,\sigma + |a_{12}||} \leqq \left|\frac{a_{22}}{a_{11}}\right| \frac{1}{|\sigma|} |\sin\psi|. \tag{4.15}$$

Together with (4.14) this gives:

$$\left|\frac{a'_{12}}{a_{12}}\right| = \left|\frac{a_{22}}{a_{11}}\right| \sigma^2\cdot\cos\phi\cdot\cos\psi \leqq \frac{a_{22}^2}{a_{11}^2} \frac{|\sigma|}{\sqrt{1+\sigma^2}} |\sin\psi| \tag{4.16}$$

which is smaller than 1 if $|a_{22}| < |a_{11}|$. The ratio of successive off-diagonal elements may approach 1 only if $|a_{22}| \to |a_{11}|$ and $|\sigma| \to \infty$, in a somewhat particular way, as discussed briefly now. Let $|a_{22}| = |a_{11}| - \varepsilon$, $(0 < \varepsilon \ll |a_{11}|)$. Then from (4.12) we have:

$$|\sigma| \approx \frac{(|a_{11}| - \varepsilon)|a_{12}|}{2|a_{11}|\varepsilon + |a_{12}|^2}. \tag{4.17}$$

Since $|\sigma| \leqq |a_{22}|/|a_{12}|$, $|\sigma| \to \infty$ implies $|a_{12}| \to 0$. Moreover, by (4.17), $|a_{12}|$ must go to 0 slower than $(2|a_{11}|\varepsilon + |a_{12}|^2)$, thus slower than ε, for the ratio (4.16) to tend to 1. From that and (4.2), it can be concluded that convergence is assured when $|a_{22}| \leqq |a_{11}|$: the ratio of successive off-diagonal elements may possibly be equal to 1 only if they are both zero.

Finally, we note that $a'_{12}/\bar{a}_{12} = O(\sigma^2)$ upon convergence i.e. when ϕ and ψ have become very small, as indicated by (4.16).

Approximation 2 corresponds to the choice: $\tan\phi = \sigma/(1+|\sigma|)$. Equation (4.14) then becomes:

$$\frac{a'_{12}}{\bar{a}_{12}} = \frac{a_{22}}{a_{11}} \frac{|\sigma|}{(1+|\sigma|)^2} \cos\phi\cdot\cos\psi. \tag{4.18}$$

Taking into account (4.6), the following bound is obtained:

$$\left|\frac{a'_{12}}{a_{12}}\right| \leqq \left|\frac{a_{22}}{a_{11}}\right| \frac{|\sigma|}{(1+|\sigma|)\sqrt{(1+|\sigma|)^2+\sigma^2}}. \tag{4.19}$$

Here also, the ratio is smaller than 1 if $|a_{22}|<|a_{11}|$. The function of σ appearing in (4.19) is maximum at $\sigma=\pm 0.8295$. Then:

$$\left|\frac{a'_{12}}{a_{12}}\right| \leqq 0.2257 \left|\frac{a_{22}}{a_{11}}\right|. \tag{4.20}$$

We remark that the use of approximation 2 leads to an error of $a'_{12}/\bar{a}_{12}=O(\sigma)$ upon convergence, which is greater than for approximation 1.

The first two approximations were proposed earlier [16] for symmetric matrices. Now a new formula (approximation 3) is presented. Equation (4.13) is approximated in the form: $\tan\phi=\sigma/(1+\sigma^2)$, and the use of (4.14) gives:

$$\frac{a'_{12}}{\bar{a}_{12}}=\frac{a_{22}}{a_{11}}\frac{\sigma^4}{(1+\sigma^2)^2}\cos\phi\cdot\cos\psi \tag{4.21}$$

which is bounded by:

$$\left|\frac{a'_{12}}{a_{12}}\right| \leqq \left|\frac{a_{22}}{a_{11}}\right| \frac{\sigma^4}{(1+\sigma^2)\sqrt{(1+\sigma^2)^2+\sigma^2}} \tag{4.22}$$

According to the same argument as for approximation 1, convergence is assured if $|a_{22}|\leqq|a_{11}|$. The error due to approximation 3 is $a'_{12}/\bar{a}_{12}=O(\sigma^4)$ upon convergence, which is smaller than for approximations 1 and 2.

If $|a_{11}|\leqq|a_{22}|$, a development similar to the above one can be done when inverting the role of ϕ and ψ. Here an expression for $\tan\phi$ (in place of $\tan\psi$) is derived from (4.3):

$$\tan\phi=\frac{a_{11}\cdot\tan\psi-|a_{12}|}{a_{22}}. \tag{4.23}$$

Then $\tan\psi$ is solution of:

$$-\sigma\cdot\tan^2\psi+\tan\psi+\sigma=0, \tag{4.24}$$

σ being given now by:

$$\sigma=\frac{a_{11}|a_{12}|}{a_{22}^2-a_{11}^2+|a_{12}|^2}. \tag{4.25}$$

The smaller root of (4.24) is identical to (4.13), except for the sign. It is easy to see that approximation 1 ($\tan\psi=-\sigma$), approximation 2 ($\tan\psi=-\sigma/(1+|\sigma|)$), and approximation 3 ($\tan\psi=-\sigma/(1+\sigma^2)$) are convergent since $|a_{11}|\leqq|a_{22}|$.

Thus we conclude that the three approximations lead to convergent Kogbetliantz processes, seemingly at the additional expense of initial tests on the order of $|a_{11}|$ and $|a_{22}|$. In reality the same expense is required for the exact method, in order to guarantee the stable computation of σ (Sect. 5). In the sequel, it will be stressed and illustrated that approximation 3 is the most adequate to compute the singular value decomposition of a triangular matrix.

4.2. *Hermitian Case*

In (4.1), we now take $\phi=\psi$, $a_{12}=\bar{a}_{21}$, and $a'_{12}=\bar{a}'_{21}$. The diagonal elements a_{11} and a_{22} are real, and $c=d=a_{12}/|a_{12}|$, according to Sect. 2. For a real, thus symmetric, matrix, approximations 1 and 2 have been investigated in [16]. The material below for complex matrices is completely analogous.

The transformed off-diagonal element has to satisfy:

$$a'_{12} := \frac{\bar{a}_{12}}{|a_{12}|}\cdot[\sin\phi\cdot\cos\phi\cdot(a_{22}-a_{11})+(\cos^2\phi-\sin^2\phi)\cdot|a_{12}|]=0 \tag{4.26}$$

exactly or approximately. Then the new diagonal elements are:

$$a'_{11}=\sin^2\phi\cdot a_{11}-2\sin\phi\cdot\cos\phi\cdot|a_{12}|+\cos^2\phi\cdot a_{22} \tag{4.27}$$

$$a'_{22}=\cos^2\phi\cdot a_{11}+2\sin\phi\cdot\cos\phi\cdot|a_{12}|+\sin^2\phi\cdot a_{22} \tag{4.28}$$

which are still real. From (4.26), $\tan\phi$ is solution of (4.11), of which we choose again the smaller root (4.13), with:

$$\sigma=\frac{|a_{12}|}{a_{11}-a_{22}}. \tag{4.29}$$

In practice, the values of σ given by (4.29) will presumably be greater in many situations than those given by (4.12) for corresponding upper triangular matrices, thus leading to a slower convergence. If $\tan\phi$ is not approximated, then (4.27–28) have the simpler form:

$$\begin{pmatrix}a'_{11}\\ a'_{22}\end{pmatrix}=\begin{pmatrix}a_{11}\\ a_{22}\end{pmatrix}+\frac{|a_{12}|}{\tan\phi}\begin{pmatrix}-1\\ 1\end{pmatrix}=\begin{pmatrix}a_{22}\\ a_{11}\end{pmatrix}+|a_{12}|\tan\phi\begin{pmatrix}-1\\ 1\end{pmatrix}. \tag{4.30}$$

The ratio of successive off-diagonal elements is:

$$\frac{a'_{12}}{\bar{a}_{12}}=\frac{-\sigma\cdot\tan^2\phi-\tan\phi+\sigma}{\sigma(1+\tan^2\phi)}. \tag{4.31}$$

For approximation 1 ($\tan\phi=\sigma$), this ratio becomes:

$$\frac{a'_{12}}{\bar{a}_{12}}=\frac{-\sigma^2}{1+\sigma^2} \tag{4.32}$$

whose absolute value is bounded by 1. If $a_{11}=a_{22}$, then $\phi=\pi/2$ and the transformed matrix differs from the initial one only by $a'_{12}=-\bar{a}_{12}$. If a_{11} is close to a_{22}, thus ϕ close to $\pi/2$, only a slight modification of the matrix is produced. In these situations, convergence may be absent or slow. In [16] it is suggested to start the computation with the exact method and to apply the approximation when σ is small enough. In case of repeated or very close eigenvalues, such a procedure may require the use of the exact method until the final result is reached since some of the diagonal elements will tend to the same value. Then the possible advantage in computation time of the approximate scheme would be lost. That difficulty does not occur with a triangular matrix, as mentioned in the previous subsection: the ratio is strictly smaller than 1, unless the off-diagonal element is zero.

Introduction of approximation 2 ($\tan\phi = \sigma/(1+|\sigma|)$) in (4.31) gives:

$$\frac{a'_{12}}{\bar{a}_{12}} = \frac{|\sigma|}{(1+|\sigma|)^2+\sigma^2}. \tag{4.33}$$

As proved in [16], the maximum occurs at $\sigma = 1/\sqrt{2}$, where $a'_{12}/\bar{a}_{12} = (\sqrt{2}-1)/2$. By this, convergence is assured. However the asymptotic error is greater than with the other approximations.

Finally approximation 3 ($\tan\phi = \sigma/(1+\sigma^2)$) leads to the smallest asymptotic error, since:

$$\frac{a'_{12}}{\bar{a}_{12}} = \frac{\sigma^4}{(1+\sigma^2)^2+\sigma^2}. \tag{4.34}$$

If $a_{11} = a_{22}$, $\phi = 0$ and the transformed matrix is simply a permutation of the initial matrix. There is no decrease of the off-diagonal elements. Convergence can be questioned in the same way as for approximation 1 for large values of σ.

We conclude that the three approximations are much less attractive for Hermitian matrices than for triangular matrices. This also appears from our test results (Sect. 6).

4.3. Arbitrary Matrix

For an arbitrary complex matrix, the determination of elementary rotations is a much more intricate process as follows from [6, Theorem 5']. Arguments and moduli of the elements a_{ij} ($i, j = 1, 2$) are first to be isolated, then combined to obtain the coefficients c and d in (4.1). Finally those quantities allow to compute the rotation angles ϕ and ψ. In order to avoid that additional computational cost, it is recommended to perform a QR factorization of the whole matrix before the SVD such that the simpler arithmetic of Subsection 4.1 can be used.

Moreover the preliminary QR factorization has further advantages. The direct application of approximate schemes to arbitrary matrices leads to the same difficulties as for Hermitian matrices. Also it can be shown [3] that the off-norm of a triangular matrix is bounded by the spread of its singular values (up to a constant), while this is not true for full matrices.

Nevertheless the QR factorization may be unnecessarily costly when for instance the whole matrix is almost diagonal already. An alternative solution would be to use a complex analog of the method described in [1, 8] for arbitrary real matrices where one first symmetrizes the 2×2 matrix prior to its exact diagonalization.

5. Error Analysis of the Different Schemes

This section[4] is devoted to the error analysis of some elementary schemes of the previous section. Namely error bounds will be derived for the exact method

[4] In case of a real matrix, this section is to be read with a_{12} instead of $|a_{12}|$ and $\bar{a}_{12}$

and approximate formulae in the upper triangular case. The analysis is close to Wilkinson's [21] to whom we refer for the Hermitian case.

All along, δ_i will denote quantities smaller in absolute value than δ, the relative precision of the machine used. Quantities stored in the computer will be denoted by a hat ($\hat{\ }$). Moreover, $\tan\phi$, $\cos\phi$, $\sin\phi$ are abbreviated to t_ϕ, c_ϕ, s_ϕ respectively, and similar abbreviations hold for functions of ψ.

5.1. Upper Triangular Case; Exact Method

Our main purposes are to establish that:

(i) $\hat{a}'_{12}$ and $\hat{a}'_{21}$ are δ-small. Indeed they are not evaluated but replaced by zero in practice.

(ii) transformation matrices are δ-close to unitary.

We assume that $|a_{22}| \leqq |a_{11}|$, for convenience. First, $\hat{\sigma}$ is not computed in the form (4.12), which would be unstable, but as:

$$\hat{\sigma} = \frac{a_{22}|a_{12}|(1+\delta_6)}{[(a_{11}+a_{22})(1+\delta_1)(a_{11}-a_{22})(1+\delta_2)(1+\delta_3)+|a_{12}|^2(1+\delta_4)](1+\delta_5)}(1+\delta_7), \tag{5.1}$$

where each of the $(1+\delta_i)$ stands for a rounding error of one elementary operation [21]. From this, one easily derives:

$$\hat{\sigma} = \sigma(1+\Delta_\sigma), \tag{5.2}$$

with $|\Delta_\sigma| \leqq 7\delta$. Then use of (4.13) gives:

$$\hat{t}_\phi = \frac{2\hat{\sigma}}{[1+\sqrt{[1+4\hat{\sigma}^2(1+\delta_8)](1+\delta_9)}(1+\delta_{10})](1+\delta_{11})}(1+\delta_{12}) = t_\phi(1+\Delta_{t\phi}), \tag{5.3}$$

with $|\Delta_{t\phi}| \leqq 11\delta$. From (4.6) and due to $|t_\phi| \leqq 1$, it follows:

$$\hat{c}_\phi = \frac{1}{\sqrt{[1+\hat{t}_\phi^2(1+\delta_{13})](1+\delta_{14})}(1+\delta_{15})}(1+\delta_{16}) = c_\phi(1+\Delta_{c\phi}) \tag{5.4}$$

$$\hat{s}_\phi = \hat{c}_\phi\,\hat{t}_\phi(1+\delta_{17}) = s_\phi(1+\Delta_{s\phi}) \tag{5.5}$$

where $|\Delta_{c\phi}| \leqq 8.25\delta$ and $|\Delta_{s\phi}| \leqq 20.25\delta$. As a consequence:

$$|\hat{c}_\phi^2 + \hat{s}_\phi^2 - 1| \leqq 28.5\delta. \tag{5.6}$$

On the other hand, evaluation of (4.10) yields:

$$\hat{t}_\psi = \frac{[a_{22}\,\hat{t}_\phi(1+\delta_{18})+|a_{12}|](1+\delta_{19})}{a_{11}}(1+\delta_{20}) = t_\psi(1+\Delta_{t\psi})+\varepsilon_{t\psi} \tag{5.7}$$

with $|\Delta_{t\psi}| \leqq 2\delta$ and $|\varepsilon_{t\psi}| \leqq 12\delta$. From (4.5) one obtains:

$$\hat{c}_\psi = \frac{1}{\sqrt{[1+\hat{t}_\psi^2(1+\delta_{21})](1+\delta_{22})(1+\delta_{23})}}(1+\delta_{24}) = c_\psi(1+\Delta_{c\psi}) \tag{5.8}$$

$$\hat{s}_\psi = \hat{c}_\psi\,\hat{t}_\psi(1+\delta_{25}) = s_\psi(1+\Delta_{s\psi}) + c_\psi\,\varepsilon_{t\psi} \tag{5.9}$$

where $|\Delta_{c\psi}| \leqq 11\delta$ and $|\Delta_{s\psi}| \leqq 14\delta$. Thus:

$$|\hat{c}_\psi^2 + \hat{s}_\psi^2 - 1| \leqq 52\delta. \tag{5.10}$$

Together (5.6) and (5.10) ensure (ii) to hold.

Then the transformed matrix is computed by means of (4.9) unregarded which one of $|a_{11}|$ or $|a_{22}|$ is the smallest. Since transformation matrices are not only δ-close to unitary but, according to (5.4–5) and (5.8–9), also δ-close to the true 2×2 transformation, tight bounds are obtained for the transformed matrix. In particular one easily proves the result claimed in (i).

5.2. *Upper Triangular Case; Approximation Schemes*

The error analyses of the three approximate schemes are similar. For brevity and due to its better properties, we only examine approximation 3.

In contrast to purpose (i) of the previous subsection here we only want to prove a'_{21} to be δ-small. Again $|a_{22}| \leqq |a_{11}|$ is assumed and bound (5.2) for $\hat{\sigma}$ remains valid. For the computation of t_ϕ, we now have:

$$\hat{t}_\phi = \frac{\hat{\sigma}}{[1+\hat{\sigma}^2(1+\delta_1)](1+\delta_2)}(1+\delta_3) = t_\phi(1+\Delta_{t\phi}) \tag{5.11}$$

where $|\Delta_{t\phi}| \leqq 10\delta$. Since $|t_\phi| \leqq 1/2$ it comes:

$$\begin{aligned} \hat{c}_\phi &= c_\phi(1+\Delta_{c\phi}), & |\Delta_{c\phi}| &\leqq 4.6\delta. \\ \hat{s}_\phi &= s_\phi(1+\Delta_{s\phi}), & |\Delta_{s\phi}| &\leqq 15.6\delta. \end{aligned} \tag{5.12}$$

We have thus:

$$|\hat{c}_\phi^2 + \hat{s}_\phi^2 - 1| \leqq 13.6\delta. \tag{5.13}$$

Evaluations (5.7–9) are now:

$$\hat{t}_\psi = t_\psi(1+\Delta_{t\psi}) + \varepsilon_{t\psi}, \qquad |\Delta_{t\psi}| \leqq 2\delta, \qquad |\varepsilon_{t\psi}| \leqq 5.5\delta, \tag{5.14}$$

and

$$\begin{aligned} \hat{c}_\psi &= c_\psi(1+\Delta_{c\psi}), & |\Delta_{c\psi}| &\leqq 7.75\delta. \\ \hat{s}_\psi &= s_\psi(1+\Delta_{s\psi}) + c_\psi\,\varepsilon_{t\psi}, & |\Delta_{s\psi}| &\leqq 10.75\delta \end{aligned} \tag{5.15}$$

which yields:

$$|\hat{c}_\psi^2 + \hat{s}_\psi^2 - 1| \leqq 32.5\delta. \tag{5.16}$$

Then a'_{11} and a'_{22} are computed by (4.9), and a'_{12} by (4.4). Again it is justified to replace a'_{21} by zero. Notice that error bounds are smaller for approximation 3 than for the exact method.

6. Timing and Operation Count

Here results of numerical experiments are given in order to illustrate the material of the previous sections. For simplicity all examples were taken real. First, it is inferred that (i) the cost of the SVD is greater for a full matrix than for a triangular one, and (ii) approximate schemes, specially approximation 3 introduced above, may be of interest for triangular matrices. Secondly, operation counts and timings are discussed for the SVD of a 2×2 triangular matrix by exact and approximate schemes, and potential advantages of the latter are emphasized.

6.1. Number of Sweeps

Again three classes of matrices are distinguished: arbitrary full, symmetric, and triangular. Arbitrary matrices have already been discussed in Subsection 4.3. Due e.g. to the number of rotations per sweep and to the complexity of the rotations, a preliminary QR factorization was recommended. So they are not considered further.

As a first experiment, the number of sweeps required by the SVD of a real $(n\times n)$ matrix, either symmetric or triangular, was evaluated for several methods. For each of a set of n values, ranging from 3 to 50, the average number of sweeps was computed over a variety (10 or 20) of matrices. Each matrix was constructed randomly, its elements being smaller than 1 in absolute value. Computations were done in double precision arithmetic ($\delta = 1.4\cdot 10^{-17}$). The final number of sweeps was noted when the (Frobenius) off-norm, as estimated after each sweep, was smaller than 10^{-15}. Figure 6.1 shows the results obtained for symmetric matrices. Sweeps (3.3) and (3.4) were applied alternately, and elementary rotations were performed by one of the four following methods: TRUE corresponds to the exact determination of the angle of rotation, leading to (4.30), while APPR1, APPR2, and APPR3 are the approximate schemes termed approximations 1, 2, and 3 in Subsection 4.2. In Fig. 6.2, similar results are given for (upper) triangular matrices, which were obtained by QR factorization (without pivoting) of arbitrary random matrices. Again exact (TRUE) and approximate (APPR1, APPR2, APPR3) schemes are considered, referring now to Subsection 4.1. It is to be noted that in case of QR factorization with pivoting (results not shown here), the average number of sweeps is smaller only by a fraction of a sweep (0.1 to 0.3 typically) than that indicated in Fig. 6.2. For regular arrays of processors (e.g. systolic arrays) where pivoting is not possible, this is only a marginal disadvantage.

From these two figures, it appears that approximate rotations in general, and the approximation 3 in particular, are a priori of greater interest for triangular than for symmetric matrices. As a consequence, only approximate schemes for triangular matrices are considered in the sequel.

On a sequential machine, the increase in number of sweeps incurred by the approximate schemes is acceptable only if it is (over)compensated by a smaller complexity of the elementary rotations with respect to the exact method.

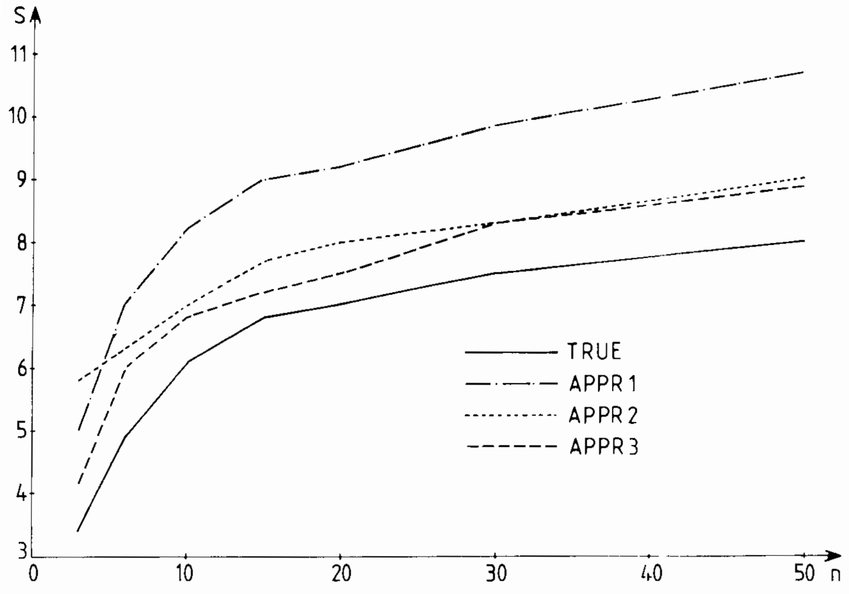

Fig. 6.1. SVD of $(n \times n)$ symmetric matrices. Number of sweeps (S) versus n

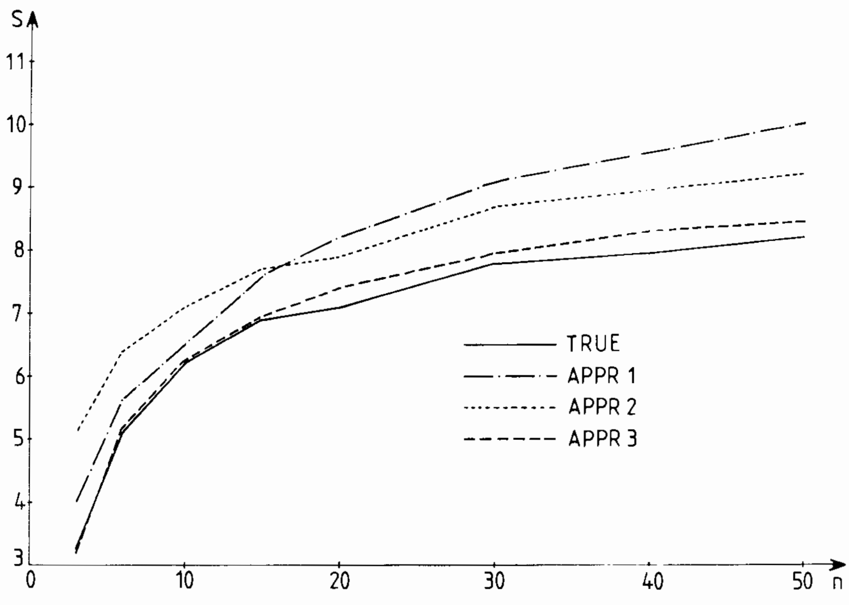

Fig. 6.2. SVD of $(n \times n)$ triangular matrices. Number of sweeps (S) versus n

On an array of processors, a slightly different type of compensation may have to be verified. If the communication between processors is limited, as in systolic-like implementations, it can be impossible to define an inexpensive termination criterion, interacting with the computational process. Rather one would stop after a predetermined number of sweeps [1] (possibly dependent on the dimension of the matrix). Such a criterion would probably be the same for both the exact method and the approximation 3 (see Fig. 6.2). The potential advantage of this approximation would thus be augmented, since it would be sufficient to compare its complexity with that of the exact scheme.

6.2. Complexity and Timing of Triangular 2×2 SVD's

We now turn to the comparison of the arithmetical complexity and the timing of various (exact and approximate) elementary SVD's of a triangular real 2×2 matrix. Five methods are considered, the source code of which is given in appendix under the same label as here:

- TRUE is the exact diagonalization scheme which is described in Subsection 4.1.
- APPR1, APPR2, and APPR3 correspond to the approximation 1, 2, and 3 discussed in subsection 4.1.
- SYM is another exact scheme used by Brent et al. ([1], algorithm USVD) in the general case, and adapted here for triangular matrices. It consists in two steps: a first rotation symmetrizes the matrix, then a second one performs the diagonalization (according to (4.30)).

It is expected that the approximate methods are less expensive than the exact ones, since the latter involve three square roots and the former only two. We try to evaluate more precisely the influence of this on the computational cost.

Arithmetical complexity of the above schemes is compared in Tables 6.1 and 6.2, where only the balance of operation counts is presented, i.e. the difference of the number of arithmetical operations from one method to the other(s). SYM and TRUE exhibit approximately the same complexity (Table 6.2). Comparison of TRUE and the approximate schemes (Table 6.1) depends on the cost of SQRT and / with respect to $\pm$ and $\times$. In general, the former operations are (much)

Table 6.1. Relative complexity of TRUE and the approximate schemes. Only differences in operation counts are mentioned

Method	TRUE	APPR1	APPR2	APPR3
Balance	1 /, 1 SQRT	3 $\times$	3 $\times$, 1 /, 1 +, 1 ABS	4 $\times$, 1 /, 1 +

Table 6.2. Relative complexity of SYM and TRUE

Method	SYM	TRUE
Balance	4 $\times$, 3 $\pm$	2 /, 1 IF

Table 6.3. Execution time of 2×2 SVD's, in percentage (TRUE: base 100)

Method	SYM	TRUE	APPR1	APPR2	APPR3
OPT	102.3	100	86.8	90.1	89.3
NOOPT	103.3	100	88.8	90.4	90.8

more expensive than the latter ones. But an exact comparison strongly depends on the particular computer and on the particular implementation. By way of illustration, an example of timing results is shown in Table 6.3. The five schemes were applied to the same data, and the corresponding execution times were measured. Programs were written in FORTRAN (see appendix) and ran on a VAX 11/780. Each program was enclosed in a subroutine, such that the measure included time for a CALL. Results in Table 6.3 are given in percentage (TRUE: base 100), for single precision arithmetic, and for two compilation modes: optimized (OPT) or not optimized (NOOPT). In these situations, a saving of time of more or less 10% is observed for the approximate schemes.

In summary, we have for triangular matrices that:

- As far as the performance of the various methods can be reflected by the product of the time of an elementary SVD by the number of sweeps for obtaining a given accuracy, approximate schemes (specially approximation 3) may be advantageous at least on some computer architectures. This would be specially true for machines that allow broadcasting.
- If the number of sweeps is fixed, as it might be the case on arrays of processors, and if the square root operation is costly, then approximation 3 seems to be the best candidate. As noted above this fixed number would be the same overestimate for both methods since approximation 3 and the true scheme in Fig. 6.2 differ by only a fraction of a sweep

7. Conclusion

In this paper various possible implementations of Kogbetliantz's algorithm were analyzed and compared for their efficiency and accuracy. The following conclusions can be drawn from it.

- For an Hermitian matrix it is more appropriate for reasons of efficiency to preserve symmetry throughout the computation and hence to compute the eigenvalue decomposition $A = U\Lambda U^*$ (the SVD can easily be derived from it). Approximate schemes may improve the speed of the algorithm on arrays of processors [16] but their efficiency is not always guaranteed and they may have to be mixed with the exact scheme (see Sect. 4.2 and [16]).
- For an arbitrary matrix it is recommended to use a preliminary QR decomposition for several reasons:
  1. First it can severely reduce the computation burden (at least by a factor 2 but even more when $m \gg n$): the number of operations of a sequential

machine (see also [13]), and the number of processors on a systolic-like array of processors and hence also the travel time of each "wave" of transformations through these processors.

2. Approximate schemes for triangular matrices can be constructed which may improve the speed of the computation as well on a sequential machine as on parallel machines (Sect. 6).
3. Convergence to positive values on diagonal is easily ensured by merely imposing a choice of positive diagonal elements in the QR decomposition.
4. The off-norm may severely drop by the mere application of the QR decomposition and convergence properties seem to be better in the presence of clusters [3].
5. The code for the triangular 2×2 SVD's (exact or approximate) is compact, numerically stable, and trivially adapts to cope with SVD's of submatrices (where certain rows or columns should not intermingle with each other [1]).

As a consequence we recommend the use of the eigenvalue decomposition for Hermitian matrices (with a preference for the exact scheme) and the use of the triangular SVD algorithm for arbitrary matrices (with a preference for the approximate scheme APPR3).

Acknowledgment. We thank C. Semaille for the careful drawing of the figures.

Appendix

Source Codes of Elementary SVD's

For definiteness, we gather here the source codes of SVD's applied to the 2×2 real triangular matrix

$$\begin{pmatrix} a_{11} & a_{12} \\ 0 & a_{22} \end{pmatrix}$$

including permutation of columns and rows, as in (4.1). Language is FORTRAN. Complex versions are trivial extensions of these ones. Tests for division by zero are not included.

True

```
      A=A11
      C=A12
      B=A22
      IF(ABS(A) .LT. ABS(B)) GO TO 5
      SIGMA=2*B*C/((A-B)*(A+B)+C*C)
      TPHI=SIGMA/(1+SQRT(1+SIGMA*SIGMA))
      TPSI=(C+TPHI*B)/A
      GO TO 10
```

```
 5    SIGMA=2*A*C/((B-A)*(B+A)+C*C)
      TPSI=-SIGMA/(1+SQRT(1+SIGMA*SIGMA))
      TPHI=(TPSI*A-C)/B
10    CPHI=1/SQRT(1+TPHI*TPHI)
      SPHI=TPHI*CPHI
      CPSI=1/SQRT(TPSI*TPSI)
      SPSI=TPSI*CPSI
      X=CPSI/CPHI
      A11=B*X
      A12=0
      A22=A/X
```

APPR1, APPR2, APPR3

For APPR1:

```
      A=A11
      C=A12
      B=A22
      IF(ABS(A).LT. ABS(B)) GO TO 5
      SIGMA=B*C/((A-B)*(A+B)+C*C)
 1    TPHI=SIGMA
      TPSI=(C+TPHI*B)/A
      GO TO 10
 5    SIGMA=A*C/((B-A)*(B+A)+C*C)
 6    TPSI=-SIGMA
      TPHI=(TPSI*A-C)/B
10    CPHI=1/SQRT(1+TPHI*TPHI)
      SPHI=TPHI*CPHI
      CPSI=1/SQRT(TPSI*TPSI)
      SPSI=TPSI*CPSI
      X=CPSI/CPHI
      A11=B*X
      A12=-SPHI*(CPSI*A+SPSI*C)+SPSI*CPHI*B
      A22=A/X
```

For APPR2, the code is the same as for APPR1, apart from the following two instructions:

```
 1    TPHI=SIGMA/(1+ABS(SIGMA))
```

and

```
 6    TPSI=-SIGMA/(1+ABS(SIGMA))
```

For APPR3, the code is the same as for APPR1 and APPR2, apart from:

```
 1    TPHI=SIGMA/(1+SIGMA*SIGMA)
```

and

```
 6    TPSI=-SIGMA/(1+SIGMA*SIGMA)
```

SYM

```
    A=A11
    C=A12
    B=A22
```

First step: symmetrization (to the left).

```
    TSYM=-C/(A+B)
    CSYM=1/SQRT(1+TSYM*TSYM)
    SSYM=TSYM*CSYM
    AS=A*CSYM
    CS=-A*SSYM
    BS=-C*SSYM+B*CSYM
```

Second step: diagonalization.

```
    SIGMA=2*CS/(AS-BS)
    TPSI=SIGMA/(1+SQRT(1+SIGMA*SIGMA))
    CPSI=1/SQRT(1+TPSI*TPSI)
    SPSI=TPSI*CPSI
    CPHI=CPSI*CSYM-SPSI*SSYM
    SPHI=CPSI*SSYM+SPSI*CSYM
    X=CS*TPSI
    A11=BS-X
    A12=0
    A22=AS+X
```

References

1. Brent, R., Luk, F., Van Loan, C.: Computation of the singular value decomposition using mesh-connected processors. J. VLSI Comput. Syst. **1**, 242–270 (1984)
2. Brent, R., Luk, F.: The solution of singular value and symmetric eigenvalue problems on multiprocessor arrays. SIAM J. Sci. Stat. Comput. **6**, 69–84 (1985)
3. Charlier, J.-P., Van Dooren, P.: On Kogbetliantz's SVD algorithm in the presence of clusters. Linear Algebra Appl. **95**, 135–160 (1987)
4. Du Croz, J.: Adapting the NAG library to vector-processing machines. NAG Newsletter **2** (1983)
5. Fernando, K., Hammarling, S.: Kogbetliantz methods for parallel SVD computation: architecture, algorithms, and convergence. NAG Techn. Rept. TR9/86. Numerical Algorithms Group Ltd, Oxford 1986
6. Forsythe, G., Henrici, P.: The cyclic Jacobi method for computing the principal values of a complex matrix. Trans. Am. Math. Soc. **94**, 1–23 (1960)
7. Gentleman, W., Kung, H.: Matrix triangularization by systolic arrays. Proc. SPIE Symp. 1981, **298**, Real Time Signal Processing IV, 19–26 (1981)
8. Golub, G., Van Loan, C.: Matrix computations. Oxford: North Oxford Academic 1983
9. Heath, M., Laub, A., Paige, C., Ward, R.: Computing the singular value decomposition of a product of two matrices. SIAM J. Sci. Stat. Comput. **7**, 1147–1159 (1986)
10. Kogbetliantz, E.: Diagonalization of general complex matrices as a new method for solution of linear equations. Proc. Intern. Congr. Math., Amsterdam **2**, 356–357 (1954)
11. Kogbetliantz, E.: Solution of linear equations by diagonalization of coefficient matrices. Quart. Appl. Math. **13**, 123–132 (1955)
12. Lawson, C., Hanson, R., Kincaid, D., Krogh, F.: Basic Linear Algebra Subprograms for Fortran usage. ACM Trans. Math. Soft. **5**, 308–323 (1979)

13. Luk, F.: A triangular processor array for computing the singular value decomposition. Linear Algebra Appl. (Special Issue on Parallel Computation) **77**, 259–273 (1986)
14. Luk, F., Park, H.: On parallel Jacobi orderings. Techn. Rept. EE-CEG-86-5, Computer Engineering Group, Cornell University 1986
15. Luk, F., Park, H.: A proof of convergence for two parallel Jacobi SVD algorithms. Tehn. Rept. EE-CEG-86-12, Computer Engineering Group, Cornell University 1986
16. Modi, J., Pryce, J.: Efficient implementation of Jacobi's diagonalization method on the DAP. Numer. Math. **46**, 443–454 (1985)
17. O'Leary, D., Stewart, G.: Data-flow algorithms for parallel matrix computations. Comm. ACM **28**, 840–853 (1985)
18. Paige, C.: Computing the generalized singular value decomposition. SIAM J. Sci. Stat. Comput. **7**, 1126–1146 (1986)
19. Paige, C., Van Dooren, P.: On the quadratic convergence of Kogbetliantz's algorithm for computing the singular value decomposition. Linear Algebra Appl. (Special Issue on Parallel Computation) **77**, 301–313 (1986)
20. Stewart, G.: A Jacobi-like algorithm for computing the Schur decomposition of a non-Hermitian matrix. SIAM J. Sci. Stat. Comput. **6**, 853–864 (1985)
21. Wilkinson, J.: The algebraic eigenvalue problem. Oxford: Clarendon Press 1965

Received May 20, 1986 / September 23, 1987