SIAM J. SCI. STAT. COMPUT.
Vol. 7, No. 4, October 1986

COMPUTING THE GENERALIZED SINGULAR VALUE DECOMPOSITION*

C. C. PAIGE†

Abstract. An algorithm is described for computing the generalized singular value decomposition of $A(m \times n)$ and $B(p \times n)$. Unitary matrices U, V and Q are developed so that U^HAQ and V^HBQ have as many nonzero parallel rows as possible, and these correspond to the common row space of the two matrices. The algorithm consists of an iterative sequence of cycles where each cycle is made up of the serial application of 2×2 generalized singular value decompositions. Convergence appears to be at least quadratic. With the correct choice of ordering the algorithm can be implemented using systolic array processors (Gentleman, personal communication). The algorithm can also be used to compute any CS decomposition of a unitary matrix.

Key words. generalized singular values, CS decomposition, unitary matrices, matrix decompositions, matrix parallel computations

AMS(MOS) subject classifications. Primary 65F30; secondary 15-04, 15A03, 15A18, 15A23, 65F25

1. Introduction.

The generalized singular value decomposition (GSVD) was introduced by van Loan [18]. Paige and Saunders [15] extended van Loan's idea in order to handle all possible cases, and presented a form of the decomposition which was more suitable for computation than that in [18]. This is the GSVD we will consider here. Van Loan [18] discussed several uses of the GSVD, for example the singular value pairs (α, β), see [15], of $A(m \times n)$ and $B(p \times n)$ solve $\det(\beta^2A^HA - \alpha^2B^HB) = 0$. Paige [14] gave an expository introduction to the GSVD and its relation to the general Gauss-Markov linear model (y, Xb, σ^2FF^T) used in regression analysis, and showed how it not only reveals the structure and sensitivity of the model, but provides the best linear unbiased estimator and the structure of the covariance of the error of this estimator. For these and other problems there is a need for a good algorithm for computing the GSVD.

When B is square and nonsingular the GSVD of $A(m \times n)$ and $B(n \times n)$ corresponds to the singular value decomposition (SVD) of AB^{-1}. If A or B is ill conditioned with respect to solution of equations, then computing AB^{-1} would usually lead to unnecessarily large numerical errors, and so this approach is not recommended in general. When B is not square, or is singular, then the SVD of AB^+, where B^+ denotes the Moore-Penrose pseudoinverse of B, does not necessarily correspond to the GSVD of A and B, and a different approach is definitely needed.

If $[A^H, B^H]^H$ is a block of columns of a unitary matrix then the GSVD of A and B gives what is called the CS decomposition for this partitioning; see Stewart [17]. This decomposition is implicit in the work of Davis and Kahan [3]. It was stated explicitly in [16] for the slightly restricted case of $A(n \times n)$, and was derived in [15] for the general case, where it was seen to provide one approach to computing the GSVD of general $A(m \times n)$ and $B(p \times n)$. Briefly that approach computes the SVD or similar unitary factorization of general

$$(1.1) \qquad \begin{pmatrix} A \\ B \end{pmatrix} = P_1RQ_1^H, \qquad P_1 = \begin{pmatrix} P_{11} \\ P_{21} \end{pmatrix},$$

$R(k \times k)$ square and nonsingular, $P_1^HP_1 = Q_1^HQ_1 = I$; followed by the CS decomposition

* Received by the editors June 21, 1984, and in revised form August 15, 1985.

† Computer Science, McGill University, Montreal, Quebec, Canada H3A 2K6. This work was supported by the Canadian Natural Sciences and Engineering Research Council under grant A8652.

of $P_{11}(m \times k)$ and $P_{21}(p \times k)$. Methods for computing the CS decomposition are given by Stewart [17] and van Loan [19], and these can be combined with (1.1) to produce the GSVD of A and B, see [15].

The above two stage process for computing the GSVD has definite drawbacks. First A and B are combined in (1.1), an approach which ignores the good numerical practice of applying unitary transformations to A and B separately where possible. Distinctly different scaling of A and B could lead to difficulties. Next a rank decision must be made in (1.1), and so it is advisable to use the SVD; but again this rank decision could be affected by the relative sizes of A and B. This results in two separate iterative algorithms with an important and possibly difficult rank decision between them, and in difficult cases there will be no clear criteria on which to make this rank decision. Accordingly we have sought one unified iterative algorithm which applies unitary transformations to A and B separately.

One approach [1] implicitly applied the QR algorithm for the SVD in [5] to the equivalent of AB^{-1}, somewhat like the QZ algorithm of Moler and Stewart [13] does for the generalized eigenvalue problem. Although this appeared to work quite satisfactorily, it was so complicated we chose not to publish it or to pursue it further. To obtain some idea of the difficulty, it suffices to note that the QR algorithm for the SVD of a given matrix C in [5] has two levels of implicitness, one because it implicitly applies the QR algorithm to C^HC while working with C rather than forming C^HC, and a second because it carries out implicit shifts in these QR steps. If now we want the SVD of $C = AB^{-1}$ without forming AB^{-1}, this gives a third level of implicitness. In contrast the QZ algorithm only has two levels of implicitness.

The simplest unified approach we have found comes from an idea of Kogbetliantz [10], [11], for finding the singular value decomposition of a square matrix. Just as Jacobi's method [9] for computing the eigenvalues of a symmetric matrix solves a sequence of 2×2 symmetric eigenproblems, Kogbetliantz' method solves a sequence of 2×2 SVD problems. It is beautifully simple and surprisingly fast, and apparently has ultimate quadratic convergence. This has been proven when there are no pathologically close singular values [21]. However it is not usually as fast as the method of [5] and [6]. The fine analysis of Forsythe and Henrici [4] not only proved convergence of the serial-cyclic Jacobi method for the symmetric eigenproblem, but also of the serial-cyclic Kogbetliantz method, both under restrictions on the choice of the angles of rotation at each step. Kogbetliantz' method is discussed further in [8].

The method to be described here computes U^HAQ and V^HBQ, where unitary U, V and Q are built up in an iterative manner which when for example B is square and nonsingular corresponds to a refinement of Kogbetliantz' method of transforming $U^HAB^{-1}V$ to diagonal form. There is clearly only one level of implicitness in this. The method is similar in approach to that in [8], and owes much to the work on that paper. In fact reference will be made to that work where appropriate, instead of repeating some material here.

In § 2 we give a brief account of the serial-cyclic version of Kogbetliantz' algorithm applied to a general square matrix, and then to the special case of an upper triangular matrix, an improvement suggested in [8]. We then show how these ideas can be used to compute the GSVD of square A and B when one is nonsingular. This is the core of the algorithm, and in § 3 it is extended to give an algorithm for general A and B. Section 4 shows that this extension gives the correct result for 2×2 matrices, and § 5 uses this to show how the algorithm behaves for general A and B, a summary of the algorithm being included for ease in following the proofs. Section 6 contains numerical examples, and some comments are made in § 7. Finally as a result of ideas of Gentleman

[22], it is shown how the algorithm given here can be implemented for parallel computation in § 8. His approach is particularly suitable for systolic array processors.

The GSVD algorithm proposed here can be applied directly to the relevant submatrices of a unitary matrix to obtain any required CS decomposition.

2. The basic algorithm. Kogbetliantz [11] computes the SVD of a square matrix in an iterative sequence of cycles. In one cycle of the serial-cyclic method, $n(n-1)/2$ SVDs of 2×2 matrices centred on the diagonal are computed to eliminate elements in the following order ($n=4$),

$$\begin{pmatrix} 0 & 1 & 2 & 3 \\ 1 & 0 & 4 & 5 \\ 2 & 4 & 0 & 6 \\ 3 & 5 & 6 & 0 \end{pmatrix}. \tag{2.1}$$

We call this the row ordering; there is obviously a column version. The elements marked 1 are eliminated first, then those marked 2, in which case those marked 1 could reappear. Of course, the diagonal elements are not eliminated; in fact their sum of squares does not decrease, and this, together with a restriction we will discuss, ensures the convergence of the algorithm to diagonal form [4]. Convergence is found in practice to be at least quadratic, and 5 cycles are often sufficient to give full accuracy on for example the VAX 11/750 using double precision.

The unitary reduction of a 2×2 matrix to diagonal form requires a unitary rotation from the left through an angle ϕ say, and one of the right through ψ say. If the serial-cyclic approach (2.1), or the column equivalent, is used then [4] shows the $n\times n$ matrix converges to diagonal form if each

$$\phi\in J, \qquad \psi\in J, \tag{2.2}$$

where J is a fixed closed interval interior to the open interval $(-\pi/2, \pi/2)$. So a method which just requires the singular values of the 2×2 diagonal to be ordered in a fixed way does not necessarily satisfy this condition.

If the initial matrix is upper triangular then one cycle takes the following form, where now only one element is eliminated by each 2×2 SVD, and is circled immediately prior to elimination.

$$\begin{array}{l} \begin{array}{cccc} \times & \otimes & \times & \times \\ & \times & \times & \times \\ & & \times & \times \\ & & & \times \end{array} \rightarrow \begin{array}{cccc} \times & & \otimes & \times \\ & \times & \times & \times \\ & & \times & \times \\ & & & \times \end{array} \rightarrow \begin{array}{cccc} \times & & & \otimes \\ \times & \times & \times & \times \\ & & \times & \times \\ & & & \times \end{array} \rightarrow \begin{array}{cccc} \times & & & \\ \times & \times & \otimes & \times \\ \times & & \times & \times \\ & & & \times \end{array} \\ \qquad\qquad \rightarrow \begin{array}{cccc} \times & & & \\ \times & \times & & \otimes \\ \times & & \times & \times \\ & & & \times \end{array} \rightarrow \begin{array}{cccc} \times & & & \\ \times & \times & & \\ \times & \times & \times & \otimes \\ \times & & & \times \end{array} \rightarrow \begin{array}{cccc} \times & & & \\ \otimes & \times & & \\ \times & \times & \times & \\ \times & \times & & \times \end{array} \end{array} \tag{2.3}$$

This leads to lower triangular form. The next cycle uses exactly the same ordering as (2.1) and produces an upper triangular matrix again This refinement leads to a saving in computation, and since this is really the same method as for the full case, the same comments on convergence hold. A more complete description of these algorithms is given in [8].

Now we present a method which we will then show implicitly carries out one cycle of Kogbetliantz' algorithm on upper triangular $C = AB^{-1}$, when both A and B are initially upper triangular and B is nonsingular. We first describe what we will refer to as the (i, j) transformation. If A_{ij} and B_{ij} are 2×2 matrices, whose elements lie in rows i and j and columns i and j of A and B, and U and V are unitary so that

$$U^H A_{ij} B_{ij}^{-1} V = S \tag{2.4}$$

is diagonal, then

$$U^H A_{ij} = S V^H B_{ij}. \tag{2.5}$$

The dependence of these 2×2 U, V and S on i and j will not be stated explictly in these first four sections.

As a result, the first row of $U^H A_{ij}$ is parallel to the first row of $V^H B_{ij}$, and the second row of $U^H A_{ij}$ is parallel to that of $V^H B_{ij}$. Thus if Q is unitary so that $V^H B_{ij} Q$ is lower triangular, that is

$$(V^H B_{ij})Q = \begin{pmatrix} \overset{\curvearrowleft}{\times \quad \otimes} \\ \times \quad \times \end{pmatrix} = \begin{pmatrix} \times & \\ \times & \times \end{pmatrix}, \tag{2.6}$$

then $U^H A_{ij} Q$ is also lower triangular. For n by n upper triangular $C = AB^{-1}$ we carry out $n(n-1)/2$ such 2×2 transformations in the same order as in (2.1) and (2.3). The kth such transformation acting on its relevant 2×2 submatrices takes the form, with C_{ij} defined to be $A_{ij} B_{ij}^{-1}$,

$$\begin{array}{ccccc} A_{ij} & = & C_{ij} & \cdot & B_{ij} \\ & & k' & & \\ k\left(\begin{matrix} \times & \times \\ \square & \times \end{matrix}\right. & = & k\left(\begin{matrix} \overset{\curvearrowleft}{\times \quad \otimes} \\ \quad\quad \times \end{matrix}\right. & \cdot & k'\left(\begin{matrix} \times & \times \\ \square & \times \end{matrix}\right. \\ k'' & & & & k'' \\ \begin{matrix} \overset{\curvearrowleft}{\times \quad \otimes} \\ \times \quad \times \end{matrix} & = & \begin{matrix} \times & \\ & \times \end{matrix} & \cdot & \begin{matrix} \overset{\curvearrowleft}{\times \quad \otimes} \\ \times \quad \times \end{matrix} \end{array} \tag{2.7}$$

where k corresponds to U^H, k' corresponds to V^H, and k'' corresponds to Q.

Note that we would design Q on the larger of the two possible rows in the figure. This is the obvious choice for numerical precision, but should be supported by a rounding error analysis. This kth transformation will be denoted

$$\begin{matrix} \underline{\times} & \underline{\times} \\ & \underline{\times} \end{matrix} = \begin{matrix} \times & \times \\ & \times \end{matrix} \cdot \begin{matrix} \underline{\times} & \underline{\times} \\ & \underline{\times} \end{matrix} \rightarrow \begin{matrix} \times & \\ \times & \times \end{matrix} = \begin{matrix} \times & \\ & \times \end{matrix} \cdot \begin{matrix} \times & \\ \times & \times \end{matrix} \tag{2.8}$$

Using this notation with that of (2.3), and taking $n = 3$ for illustration, one cycle starting with upper triangular A and B takes the form

$$
\begin{array}{ccccccccccccc}
 & A & & = & & C & & \cdot & & B & & & \\
\underline{\times} & \underline{\times} & \times & = & \times & \otimes & \times & \cdot & \underline{\times} & \underline{\times} & \times & \rightarrow \\
 & \underline{\times} & \times & & & \times & \times & & & \underline{\times} & \times & \\
 & & \times & & & & \times & & & & \times & \\
 & & & & & & & & & & & \\
\underline{\times} & & \underline{\times} & = & \times & & \otimes & \cdot & \underline{\times} & & \underline{\times} & \rightarrow \\
\times & \times & \times & & & \times & \times & & \times & \times & \times & \\
 & & \underline{\times} & & & & \times & & & & \underline{\times} & \\
 & & & & & & & & & & & \\
\times & & & = & \times & & & \cdot & \times & & & \rightarrow \\
\times & \underline{\times} & \underline{\times} & & \times & \times & \otimes & & \times & \underline{\times} & \underline{\times} & \\
\times & & \underline{\times} & & & & \times & & \times & & \underline{\times} & \\
 & & & & & & & & & & & \\
\underline{\times} & & & = & \times & & & \cdot & \underline{\times} & & & \\
\underline{\times} & \underline{\times} & & & \otimes & \times & & & \underline{\times} & \underline{\times} & & \\
\times & \times & \times & & \times & & \times & & \times & \times & \times &
\end{array}
\tag{2.9}
$$

which results in lower triangular A and B. It can be seen from the above forms of A and B that the elements of the 2×2 $C_{ij}=A_{ij}B_{ij}^{-1}$ are just the elements in rows and columns i and j of $C=AB^{-1}$. The proof that this is true for larger n follows by induction, noting that after the off-diagonal elements in the first row of each of A and B have been eliminated, the remaining rotations are designed on the $n-1\times n-1$ upper triangular matrices in the bottom right-hand corners of A and B. When A and B are lower triangular the process is essentially the same and follows the ordering in (2.1), except now the equivalent of (2.6) is

$$
(V^H B_{ij})Q=\begin{pmatrix} \times & \times \\ \otimes \curvearrowright & \times \end{pmatrix}=\begin{pmatrix} \times & \times \\ & \times \end{pmatrix}
\tag{2.10}
$$

and the equivalent of (2.8) becomes

$$
\begin{array}{ccccccccccccccccc}
\underline{\times} & & = & \times & & \cdot & \underline{\times} & & \rightarrow & \times & \times & = & \times & & \cdot & \times & \times \\
\underline{\times} & \underline{\times} & & \times & \times & & \underline{\times} & \underline{\times} & & & \times & & & \times & & & \times
\end{array}
\tag{2.11}
$$

so that at the end of the cycle the A and B matrices are upper triangular again.

A comparison of C in (2.9) with (2.3) shows that this process is mathematically equivalent to applying Kogbetliantz' serial-cyclic method to upper triangular $C=AB^{-1}$, and all the previous comments on convergence hold. As a result we have given a method for finding the GSVD of two matrices, at least one of which is nonsingular. It will be seen in § 6 that this is very effective, giving as much accuracy as the problem and computer allows, in a reasonable time.

The computational cost per cycle is about the same as the method in [8] for finding the SVD of the product of two triangular matrices. That is, if we use r-multiplication rotations, one cycle as in (2.9) costs about rn^3 multiplications and $2n^3$ additions just to form the new A and B. In addition it would require $rn^3/2$ multiplications and n^3 additions for each of U, V, and Q that we chose to update. Since the method may sometimes take more than 5 cycles to converge, but rarely much more, it can be quite expensive for large n.

To counter-balance this expense it is clear that such methods are ideal for special architectures such as systolic arrays, and this can result in very efficient implementations.

Kogbetliantz' SVD method [11] for a full square matrix A has been suggested by Brent, Luk and van Loan [2] for systolic array computation. Their resulting method requires $O(n^2)$ processors and $O(n \log n)$ time to execute. On hearing the ideas presented here, Morven Gentleman [22] showed how the algorithms of the present paper can be implemented very effectively using systolic array processors.

3. The general case. For A and B square and B nonsingular we saw convergence of the method in § 2 gave $U^HAB^{-1}V = S$, a diagonal matrix, with V^HBQ triangular. That is

$$U^HAQ = SV^HBQ, \qquad S = \operatorname{diag}(\sigma_1, \cdots, \sigma_n), \tag{3.1}$$

with the ith row of U^HAQ parallel to the ith row of V^HBQ. If we ordered the singular values $\sigma_1 \geqq \sigma_2 \geqq \cdots \geqq \sigma_n \geqq 0$, and A had rank s, then U^HAQ would be zero in all but the first s rows, and so would be upper trapezoidal if the algorithm took an even number of cycles.

For simplicity we will take A and B square, by initial unitary reduction to upper trapezoidal form and adding zero rows if necessary, and extend the algorithm to handle singular B. We will describe an algorithm which effectively produces on convergence after an even number of cycles, nonsingular upper triangular R and positive definite S_A and S_B such that

$$\underset{k\times k}{R} = \begin{pmatrix} R_{11} & R_{12} & R_{13} \\ 0 & R_{22} & R_{23} \\ 0 & 0 & R_{33} \end{pmatrix} \begin{array}{l} \}r_1 \\ \}r_2, \\ \}s \end{array} \qquad r = r_1 + r_2, \tag{3.2}$$

$$S_A = \operatorname{diag}(\alpha_{r_1+1}, \cdots, \alpha_r), \qquad S_B = \operatorname{diag}(\beta_{r_1+1}, \cdots, \beta_r), \tag{3.3}$$

$$S_A^2 + S_B^2 = I, \tag{3.4}$$

$$U^HAQ = \begin{pmatrix} R_{11} & R_{12} & R_{13} & 0 \\ 0 & S_AR_{22} & S_AR_{23} & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \tag{3.5}$$

$$V^HBQ = \begin{pmatrix} 0_B & 0 & 0 & 0 \\ 0 & S_BR_{22} & S_BR_{23} & 0 \\ 0 & 0 & R_{33} & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}. \tag{3.6}$$

Here 0_B is an $r_1 \times r_1$ zero matrix so $[S_BR_{22}, S_BR_{23}]$ in (3.6) is in the same position as $[S_AR_{22}, S_AR_{23}]$ in (3.5). These then correspond to the nonzero parallel rows of U^HAQ and V^HBQ, in analogy with (3.1). It follows from the theorem in [15, § 2] that this is the GSVD of A and B each with n columns.

To obtain (3.5) and (3.6) for any $A(m \times n)$ and $B(p \times n)$ we first transform with unitary matrices U and V and a permutation matrix Q so that U^HAQ and V^HBQ are upper trapezoidal; see for example [7]. To the resulting nonzero trapezoids we add zero rows if necessary to give square matrices. This is not essential but it simplifies the description and we can apply the algorithm very much as in § 2. We will assume the initial transformations we start with are identically partitioned

$$\underset{n\times n}{A} = \begin{pmatrix} A_{11} & A_{12} & A_{13} \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \begin{array}{l} \}r \\ \\ \\ \end{array}, \qquad \underset{n\times n}{B} = \begin{pmatrix} B_{11} & B_{12} & B_{13} \\ 0 & B_{22} & B_{23} \\ 0 & 0 & 0 \end{pmatrix} \begin{array}{l} \}r \\ \}q, \\ \\ \end{array} \tag{3.7a}$$

(3.7b) A_{11}, B_{11}, B_{22} are upper triangular, A_{11} is nonsingular, the nonzero part of B is at the top and has full row rank, and if $q>0$, B_{22} is nonsingular.

This can be done by designing the permutations in Q first during the reduction of A until A_{11} is produced, then from the part of the reduction of B that gives B_{22}.

The only reason the algorithm of § 2 cannot be applied directly here is that some of the 2×2 matrices B_{ij} in (2.4) will be singular. To circumvent this we merely need to define

$$A_{ij}=\begin{pmatrix}\alpha_{ii} & \alpha_{ij}\\ \alpha_{ji} & \alpha_{jj}\end{pmatrix},\quad B_{ij}=\begin{pmatrix}\beta_{ii} & \beta_{ij}\\ \beta_{ji} & \beta_{jj}\end{pmatrix},\quad C_{ij}=\begin{pmatrix}\gamma_{ii} & \gamma_{ij}\\ \gamma_{ji} & \gamma_{jj}\end{pmatrix},$$
$$C_{ij}=A_{ij}\operatorname{adj}(B_{ij})=A_{ij}\begin{pmatrix}\beta_{jj} & -\beta_{ij}\\ -\beta_{ji} & \beta_{ii}\end{pmatrix}, \tag{3.8}$$

where adj stands for *adjugate* (or *adjoint*, depending on which book you read last). So $A\operatorname{adj}(A)=\det(A)I$, α_{ij} are the elements of A, and β_{ij} are the elements of B. We then replace (2.4) by

$$U^HC_{ij}V=S=\operatorname{diag}(\sigma_{ii},\sigma_{jj}),\qquad \sigma_{ii},\sigma_{jj}\geqq 0, \tag{3.9}$$

taking care that

$$U=V=I\quad\text{when } C_{ij}=0. \tag{3.10}$$

With these changes the general algorithm then proceeds exactly as the algorithm in § 2, and somewhat surprisingly is found in practice to converge to the desired result (3.5) and (3.6) when we have an even number of cycles, or the corresponding lower triangular forms when we have an odd number of cycles. An outline of the algorithm is given in § 6; it includes a possible reordering of the elements to ensure (3.2) to (3.6) result.

This approach treats A and B essentially equally, for if we define

$$K=\begin{pmatrix}0 & 1\\ -1 & 0\end{pmatrix} \tag{3.11}$$

then it is easy to check that

$$B_{ij}\operatorname{adj}(A_{ij})=K^TC_{ij}^TK=\operatorname{adj}(C_{ij}), \tag{3.12}$$

so this has the same elements, apart from sign, as (3.8), and both have the same singular values and eigenvalues. That is, working with (3.8) is essentially the same as working with (3.12). At first glance (3.5) and (3.6) seem to treat A and B differently, but if W is unitary so that RW is lower triangular, see (3.2), then if W is applied to the right of (3.5) and (3.6) the roles of A and B will essentially be reversed. But this is just the form resulting from the odd-numbered steps of the algorithm, and so we can say A and B are essentially treated equally in the algorithm.

We have suggested a reasonable method, but now to see if it gives us what we want, we first show that (3.8) to (3.10) give U^HA_{ij} and V^HB_{ij} parallel rows, and so give the GSVD of 2×2 A_{ij} and B_{ij}. We will then use this result to justify the algorithm.

4. The GSVD of 2×2 A and B.

We will show that (3.8) and (3.9) result in U^HA_{ij} and V^HB_{ij} having parallel rows, so that (2.10) results in the GSVD of A_{ij} and B_{ij}, see (3.2) to (3.6). This makes (3.8) and (3.9) a reasonable choice for the 2×2 computations in our algorithm in § 3. That is, we compute the GSVD of two general matrices by an iterative sequence of 2×2 GSVDs, in analogy with the Jacobi and Kogbetliantz methods. We will need the following result.

LEMMA 1. *If A and B are nonzero 2×2 matrices, then*

$$A\,\mathrm{adj}\,(B)=0 \Leftrightarrow A=fd^T \text{ and } B=hd^T, \tag{4.1}$$

that is, $\mathrm{rank}\,(A)=\mathrm{rank}\,(B)=1$ *and A and B have the same row space.*

Proof. If $B=hd^T=[\eta_1,\eta_2]^T[\delta_1,\delta_2]$ then

$$\mathrm{adj}\,(B)=[\delta_2,-\delta_1]^T[\eta_2,-\eta_1]. \tag{4.2}$$

Thus if $A=fd^T$ then $A\,\mathrm{adj}\,(B)=0$. On the other hand if $A\,\mathrm{adj}\,(B)=0$ with A and B nonzero 2×2 matrices, then $\mathrm{rank}\,(A)=\mathrm{rank}\,(B)=1$. Write $B=hd^T$, $A=ft^T=f[\tau_1,\tau_2]$, so that

$$0=A\,\mathrm{adj}\,(B)=f\cdot(\tau_1\delta_2-\tau_2\delta_1)\cdot[\eta_2,-\eta_1],$$

and since f and h are nonzero, $\det([t,d])=0$ so that t and d are parallel. The result follows.

THEOREM 1. *If A and B are 2×2 matrices and U and V are unitary so that*

$$U^HA\,\mathrm{adj}\,(B)V=S=\mathrm{diag}\,(\sigma_1,\sigma_2),\qquad \sigma_1\geqq\sigma_2\geqq 0, \tag{4.3}$$

then the ith row of U^HA is parallel to the ith row of V^HB, $i=1,2$. If also

$$A\,\mathrm{adj}\,(B)\neq 0 \tag{4.4}$$

and $U=[u_1,u_2]$, $V=[v_1,v_2]$, then

$$u_1^HA\neq 0,\qquad v_2^HB\neq 0, \tag{4.5}$$

$$\mathrm{rank}\,(A)=1\Rightarrow u_2^HA=0, \tag{4.6}$$

$$\mathrm{rank}\,(B)=1\Rightarrow v_1^HB=0. \tag{4.7}$$

Proof. From (4.3) we see that

$$U^HA\det(B)=SV^HB. \tag{4.8}$$

We enumerate all the possibilities:

(i) If A or B is zero the results are trivial.

(ii) If A is nonsingular the results follow from (4.8) and Lemma 1.

(iii) If $\mathrm{rank}\,(B)=1$ we write $B=hd^T$ for nonzero h and d. Together with (4.3) and (4.8) this gives

$$SV^HB=0,\quad V^HB=ed^T,\quad e\neq 0,\quad S=\begin{pmatrix}\sigma_1 & \\ & 0\end{pmatrix}, \tag{4.9}$$

so (4.2) and (4.3) give

$$u_2^HA\begin{pmatrix}\delta_2\\ -\delta_1\end{pmatrix}=0. \tag{4.10}$$

With this case of $\mathrm{rank}\,(B)=1$ we have the following subcases.

(a) If A is nonsingular then $\sigma_1>0$ in (4.3), and so in (4.9)

$$V^HB=\begin{pmatrix}0\\ \varepsilon_2d^T\end{pmatrix},\quad \varepsilon_2d^T\neq 0\quad\text{for some scalar } \varepsilon_2, \tag{4.11}$$

proving (4.5) and (4.7). But from (4.10) we have nonsingular

$$(4.12)\qquad U^HA=\begin{pmatrix} * \\ \phi_2 d^T\end{pmatrix}$$

for some scalar ϕ_2, and so the results follow.

(b) If $A=fd^T$ then $U^HA=U^Hfd^T$ and $V^HB=V^Hhd^T$ automatically have parallel rows, but from Lemma 1 (4.4) does not hold.

(c) Finally if

$$(4.13)\qquad A=ft^T,\quad B=hd^T,\quad \det([t,d])\neq 0,\quad f\neq 0,\quad h\neq 0,$$

then from Lemma 1 $A\,\mathrm{adj}\,(B)\neq 0$, so $\sigma_1>0$ in (4.3) and again (4.11) holds. But now (4.10) and (4.13) imply $u_2^Hf=0$, and so

$$(4.14)\qquad U^HA=\begin{pmatrix}\phi_1 t^T\\ 0\end{pmatrix},\qquad V^HB=\begin{pmatrix}0\\ \varepsilon_2 d^T\end{pmatrix},$$

which are necessarily nonzero with parallel rows. Also (4.4) holds, and (4.14) shows that (4.5) to (4.7) hold, completing the theorem.

We see that (4.3) corresponds to (3.9) in the algorithm of § 3, and so these results hold for our algorithm. The use of (3.10) in the algorithm leads to another important result.

THEOREM 2. *If* (3.8) *to* (3.10) *hold with unitary* $U=[u_1,u_2]$, $V=[v_1,v_2]$, $I=[e_1,e_2]$, *then*

$$(4.15)\qquad e_2^TA_{ij}=0\Rightarrow u_2^HA_{ij}=0,$$

$$(4.16)\qquad e_1^TB_{ij}=0\Rightarrow v_1^HB_{ij}=0,$$

$$(4.17)\qquad e_2^TB_{ij}=0\Rightarrow\begin{cases} U=V=I & \textit{if } C_{ij} \textit{ is zero},\\ \textit{or } V=[e_2,e_1] & \textit{if } C_{ij} \textit{ is nonzero}.\end{cases}$$

Proof. If C_{ij} in (3.8) is nonzero (4.15) and (4.16) follow from (4.6) and (4.7). If C_{ij} is zero (4.15) to (4.17) follow from (3.10). If C_{ij} is nonzero then zero $e_2^TB_{ij}$ implies B_{ij} has rank unity and then (4.17) follows from (4.7) and the fact that V is unitary.

5. Properties of the algorithm.

One approach to finding the GSVD is to transform A and B to separate the row subspaces, and then solve the GSVD of two square nonsingular triangular matrices using the method of § 2. This would be an easy method to follow and a theoretically correct algorithm, but it would require two rank decisions which in practice could be difficult to make. To avoid this we carry out the QR factorization of each of A and B as the only preprocessing, giving square $A^{(0)}$ and $B^{(0)}$ of the form in (3.7). Here we can be somewhat casual on the rank decisions and just use tolerances. We then apply the general iterative algorithm of § 3 to these, resulting in square $A^{(k)}$ and $B^{(k)}$ after the kth cycle. It is far from trivial to show that this gives the correct result in theory, and so we will summarize the main results first before proving the details. We use $S(A)$ to denote the row space of A.

We will show that after the first cycle we will have

$$(5.1)\qquad A^{(1)}=\begin{pmatrix}A_{1.}^{(1)}\\ 0\\ 0\end{pmatrix}=\begin{pmatrix}A_{11}^{(1)} & 0 & 0\\ 0&0&0\\ 0&0&0\end{pmatrix}\begin{matrix}\}r\\ \}s,\\ \end{matrix}\qquad B^{(1)}=\begin{pmatrix}B_{1.}^{(1)}\\ B_{2.}^{(1)}\\ 0\end{pmatrix}=\begin{pmatrix}B_{11}^{(1)} & 0 & 0\\ B_{21}^{(1)} & B_{22}^{(1)} & 0\\ 0&0&0\end{pmatrix}\begin{matrix}\}r\\ \}s\\ \end{matrix}$$

where $A_{11}^{(1)}$ is nonsingular and lower triangular, and $B_{11}^{(1)}$, $B_{22}^{(1)}$ are square and lower triangular. It follows that

$$S(B_{1.}^{(1)}) \subset S(A^{(1)}). \tag{5.2}$$

A reordering step in the algorithm also ensures

$$B_{2.}^{(1)} \text{ has full row rank and } B_{22}^{(1)} \text{ is nonsingular.} \tag{5.3}$$

After every subsequent step we will show we have identically partitioned matrices

$$A^{(k)} = \begin{pmatrix} A_{1.}^{(k)} \\ A_{2.}^{(k)} \\ 0 \\ 0 \end{pmatrix} \begin{matrix} \}r_1 \\ \}r_2 \\ \\ \\ \end{matrix}, \qquad B^{(k)} = \begin{pmatrix} 0 \\ B_{2.}^{(k)} \\ B_{3.}^{(k)} \\ 0 \end{pmatrix} \begin{matrix} \}r_1 \\ \}r_2 \\ \}s \\ \\ \end{matrix}, \qquad k > 1, \tag{5.4}$$

where each nonzero block has full row rank and the dimensions are related to (5.1) via

$$r_1 + r_2 = r. \tag{5.5}$$

The even numbered cycles will result in identically partitioned matrices with the row partitioning of (5.4)

$$A^{(k)} = \begin{pmatrix} A_{11}^{(k)} & A_{12}^{(k)} & A_{13}^{(k)} & 0 \\ 0 & A_{22}^{(k)} & A_{23}^{(k)} & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \qquad B^{(k)} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & B_{22}^{(k)} & B_{23}^{(k)} & 0 \\ 0 & 0 & B_{33}^{(k)} & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \tag{5.6}$$

k even, $k > 0$,

$$A_{11}^{(k)},\ A_{22}^{(k)},\ B_{22}^{(k)},\ B_{33}^{(k)} \text{ nonsingular upper triangular,} \tag{5.7}$$

while the odd numbered cycles will result in matrices with identical partitioning to (5.6)

$$A^{(k)} = \begin{pmatrix} A_{11}^{(k)} & 0 & 0 & 0 \\ A_{21}^{(k)} & A_{22}^{(k)} & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \qquad B^{(k)} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ B_{21}^{(k)} & B_{22}^{(k)} & 0 & 0 \\ B_{31}^{(k)} & B_{32}^{(k)} & B_{33}^{(k)} & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \qquad k \text{ odd}, k > 1, \tag{5.8}$$

$$A_{11}^{(k)},\ A_{22}^{(k)},\ B_{22}^{(k)},\ B_{33}^{(k)} \text{ nonsingular low triangular.} \tag{5.9}$$

In computing (5.6) from (5.8) or the identically partitioned $B^{(1)}$, $B_{3.}^{(k)}$ is not touched by the left transformations V^H, while in computing (5.8) from (5.6) $A_{1.}^{(k)}$ is not touched by the left transformation U^H.

When we have proven (5.1) to (5.9) we can prove the following key result. From (5.6) and (5.7)

$$S(A_{2.}^{(k-1)}) \subset S(B^{(k-1)}), \quad k-1 \text{ even}, \quad k > 1, \tag{5.10}$$

so the comment following (5.9) ensures in (5.8)

$$S(A_{2.}^{(k)}) \subset S(B^{(k)}), \quad k \text{ odd}, \quad k > 1. \tag{5.11}$$

But $B_{33}^{(k)}$ is nonsingular in (5.8), and so

$$A_{2.}^{(k)} = C_2^{(k)} B_{2.}^{(k)}, \quad k \text{ odd}, \quad k > 1.$$

In a similar way we obtain a result for even k, and using (5.7) and (5.9) we have

$$A_{2.}^{(k)} = C_2^{(k)} B_{2.}^{(k)}, \quad C_2^{(k)} \text{ nonsingular}, \quad k = 2, 3, 4, \cdots \tag{5.12}$$

where we use (5.2) for the case $k = 2$.

We see that (5.12) gives the common row space of maximum dimension since $A_{11}^{(k)}$ is nonsingular in (5.6), and $B_{33}^{(k)}$ is nonsingular in (5.8). In theory this means that after the first two cycles of the algorithm only the rows of $A_{2.}^{(k)}$ and $B_{2.}^{(k)}$ are affected by transformations from the left, and so we have carried out a direct algorithm for separating subspaces. In practice we can still allow $B_{3.}^{(k)}$ and $B_{2.}^{(k)}$ to be combined in the transformation of (5.6), and $A_{1.}^{(k)}$ and $A_{2.}^{(k)}$ to be combined in the transformation of (5.8), giving possibly superior final results in the presence of rounding errors to those obtained by working with $A_{2.}^{(k)}$ and $B_{2.}^{(k)}$ alone in the iterative part of the algorithm.

In the third and subsequent cycles, an examination of the individual rotations shows that we are implicitly carrying out Kogbetliantz' algorithm to diagonalize the implicitly defined nonsingular trangular matrix $C_2^{(k)}$ in (5.12), and this necessarily converges if the angles of rotation obey (2.2). This means if (5.1) to (5.9) hold then we have the required proof for this algorithm in the absence of rounding errors.

We summarize the algorithm prior to showing the results of this section hold.

(5.13) ALGORITHM

```
begin {GSVD algorithm, $A$, $B$, $n$ given as in (3.7)}
  $U := I_n$; $V := I_n$; $Q := I_n$; {if wanted}
  cycle := 0;
  while nonconvergence and (cycle < 10) do
  begin {general step}
    cycle := cycle + 1;
    for $i := 1$ to $n-1$ do
      for $j := i+1$ to $n$ do
      begin {$(i, j)$ transformation. Here $\alpha_{ij}$ and $\beta_{ij}$ are the $(i, j)$ elements of $A$ and $B$
        respectively. $U_{ij}$, $V_{ij}$, $Q_{ij}$ are $2\times 2$ unitary matrices.}
        $A_{ij} := \begin{pmatrix} \alpha_{ii} & \alpha_{ij} \\ \alpha_{ji} & \alpha_{jj} \end{pmatrix}$   $B_{ij} := \begin{pmatrix} \beta_{ii} & \beta_{ij} \\ \beta_{ji} & \beta_{jj} \end{pmatrix}$;
        $U_{ij}^H A_{ij} \operatorname{adj}(B_{ij}) V_{ij} = \operatorname{diag}(\sigma_1, \sigma_2)$, $\sigma_1, \sigma_2 \geqq 0$, {where $\sigma_1 \geqq \sigma_2$ if cycle $\leqq 2$, else
        (2.2) should hold}, with $U_{ij} = V_{ij} = I$ if $A_{ij} \operatorname{adj}(B_{ij}) = \operatorname{diag}(\sigma_1, \sigma_1)$;
        Choose $Q_{ij}$ so $U_{ij}^H A_{ij} Q_{ij}$ and $V_{ij}^H B_{ij} Q_{ij}$ are lower triangular if cycle is odd,
        else upper triangular;
        Let $U_n$, $V_n$, $Q_n$ be $n\times n$ unit matrices with $(i, i), (i, j), (j, i), (j, j)$ elements
        replaced by the $(1, 1), (1, 2), (2, 1), (2, 2)$ elements respectively of $U_{ij}$, $V_{ij}$, $Q_{ij}$
        respectively;
        $A := U_n^H A Q_n$; $B := V_n^H B Q_n$;
        $U := U U_n$; $V := V V_n$; $Q := Q Q_n$; {as required}
      end; {$(i, j)$ transformation}
    if cycle = 1 then do
    begin {reorder}
      Let $r$ and $s$ be as in (5.1);
      for $i := r+1$ to $r+s$ do
        if the $i$th row of $B$ is zero, move it and the $i$th column to the end of $B$,
        bringing all remaining rows and columns up one;
    end; {reorder}
  end; {general step}
end. {GSVD algorithm}
```

The remainder of this section is hard going. There is undoubtedly a more straightforward way of proving that this algorithm works, and I hope someone else can find it.

Theorem 1 proves that both $U_{ij}^H A_{ij}$ and $V_{ij}^H B_{ij}$ in the algorithm are triangularized by the one unitary matrix Q_{ij}, and as a result the progress of each odd numbered cycle is correctly described by A and B in (2.9), with the corresponding result for even numbered cycles. This means that $A^{(k)}$ and $B^{(k)}$ are lower triangular for odd k and upper triangular for even k, so (5.1) and (5.6) to (5.9) have the correct triangular form. But Theorem 2 shows that all the zero rows at the bottom of $A^{(k-1)}$ stay there, so from (3.7) the leading r rows of $A^{(k)}$ in (5.1) and (5.4) have full row rank, and (5.2) holds. This gives nonsingularity of $A_{11}^{(1)}$ in (5.1) and the A matrices in (5.9). Theorem 2 also shows that in an (i, j) transformation a zero row of $B^{(k)}$ can be exchanged with a nonzero row above it, but not with one below it, and a nonzero row cannot be combined with a zero row to give two nonzero rows. Combined with (3.7) this gives:

RESULT 1. *The collection of nonzero rows of any $B^{(k)}$ has full row rank, and apart from the reordering step in the algorithm nonzero rows of B can only move downward.*

The insistence that A_{11} and B_{22}, if it exists, be nonsingular upper triangular in (3.7) ensures in (5.1) that the diagonal elements beyond the rth corresponding to nonzero rows of $B^{(1)}$ are themselves nonzero. We prove this as a slightly more general result which also applies to the cycle which takes (5.6) to (5.8). We see (5.6) is a special case of (3.7a), and (5.8) is a special case of (5.1).

RESULT 2. *Let A_{11} and B_{22} be nonsingular upper triangular in* (3.7a), *and let* (5.1) *be the result of applying one cycle of the algorithm* (5.13) *to these, then $B^{(1)}$ has nonzero diagonal elements in the nonzero rows beyond the rth.*

Proof. Two classes of (i, j) transformations are relevant. When $i<j\leqq r$, A_{ij} is nonsingular and so the (i, i) and (j, j) elements of A remain nonzero. When $i\leqq r<j$ there is no U_{ij} transformation of A_{ij}, and the Q_{ij} transformation gives the (i, i) element of A size $\|[\alpha_{ii}, \alpha_{ij}]\|_2>0$, since α_{ii} is nonzero. We will show that if the jth row of B is zero and it is exchanged with the ith row this results in a nonzero (j, j) element of B, while if the (j, j) element of B is nonzero it stays so. Since initially B_{22} is nonsingular upper triangular in (3.7a) this will prove Result 2 by induction. When j corresponds to a zero row of B it will only be exchanged with the ith row if the top rows of A_{ij} and B_{ij} are not parallel; consequently the Q_{ij} which makes the $(1, 2)$ element of $A_{ij}Q_{ij}$ zero cannot also make the new (j, j) element of B zero. Next suppose before the (i, j) transformation that the (j, j) elements of B is nonzero. This implies that A_{ij} and $V_{ij}^H B_{ij}$ have parallel first rows, and since the $(1, 1)$ element of A_{ij} is nonzero the $(2, 2)$ element of $V_{ij}^H B_{ij}$ must be nonzero, since it could only be zero if V_{ij} were an exchange putting $[0, *]$ in the first row. Since $A_{ij}Q_{ij}$ has zero $(1, 2)$ element Q_{ij} does not exchange the columns, and $V_{ij}^H B_{ij} Q_{ij}$ cannot have zero $(2, 2)$ element, completing the proof of Result 2.

Before we can use Result 2 to say $B_{22}^{(1)}$ in (5.1) and $B_{33}^{(k)}$ in (5.8) are nonsingular we have to show that after the reordering step they are still lower triangular and $B_{2\cdot}^{(1)}$ in (5.1) and $B_{3\cdot}^{(k)}$ in (5.4) have no zero rows.

Let B' be the B matrix prior to the reordering step that results in $B^{(1)}$. Occasionally the nonzero rows after the rth in B' will not be adjacent, considerably complicating our analysis. Here we show that if $j>r$ and the jth row of B' is zero then so is the jth column, so the reordering step moves both these to the end. The jth column of $A^{(1)}$ is already zero, and the exchange has no effect on it. The remaining rows and columns of B' each move up one so the triangular form of $B_{22}^{(1)}$ is maintained and finally (5.3) follows from Result 2. Suppose the jth row of B' is zero, $j>r$, but B' has at least one nonzero row with higher index. In the (i, j) transformations in any cycle, zero rows of B can only be moved upwards, and since (j, t) transformations, $t>j$, have $A_{jt}=0$, a zero row cannot be moved into the jth row from below. It follows that the jth row and all below it were zero at the start, and the jth row has remained zero through all the transformations. Now consider an (i, j) transformation in the first cycle, with $i<j$. We must have

$$A_{ij}=\begin{pmatrix} \times & \times \\ 0 & 0 \end{pmatrix}, \qquad B_{ij}=\begin{pmatrix} \times & \times \\ 0 & 0 \end{pmatrix}. \tag{5.14}$$

The top rows of these are parallel, otherwise the nonzero ith row of B would be exchanged with the jth, which we have just shown did not happen. It follows from Lemma 1 that $U_{ij}=V_{ij}=I$, and then $A_{ij}Q_{ij}$ and $B_{ij}Q_{ij}$ have zero $(1,2)$ elements, so the (i,j) element of B is made zero. This ith row can now only be affected in (i,t) transformations with $t>j$. If the tth row of B were zero it could be made nonzero by exchanging it with this ith row, but the (i,j) and (t,j) elements would remain zero. If the tth row of B were nonzero it must have initially been the result of an earlier exchange like this giving it zero jth element. Clearly if the ith and tth rows of B are combined in the (i,t) transformation, the (i,j) and (t,j) elements of B are zero before and after this transformation. It follows that B' has zero jth column as stated. The reordering step then gives $B_{2.}^{(1)}$ full row rank in (5.1) with $B_{22}^{(1)}$ lower triangular and nonsingular proving (5.3).

We have seen that all the rows of $B^{(k)}$ beyond the $(r+s)$th must henceforth remain zero, and since no zero rows can be moved downward or combined with nonzero rows, and the nonzero rows maintain full row rank, $B_{3.}^{(k)}$ in (5.4) must have full row rank for $k>1$. Also if $j>r+s$, A_{ij} and B_{ij} from (5.1), (5.6) and (5.8) have zero second columns and rows, so these will not be transformed and the columns beyond the $(r+s)$th remain zero after all cycles. This with the full row rank of $B_{3.}^{(k)}$ shows that $B_{33}^{(k)}$ in (5.6) is nonsingular.

In theory this reordering step can only have an effect immediately following the first cycle; however in practice with the use of tolerances it could also be useful in later cycles. We now want to show $B_{2.}^{(k)}$ in (5.4) has full row rank.

An (i,j) transformation on (5.1) or (5.8) with $1\leqq i<j\leqq r$ transforms a nonsingular lower triangular A_{ij} into necessarily nonsingular upper triangular form, while from Theorem 2 if B_{ij} has nonzero first row and zero second these are exchanged. Thus if there are exactly r_1 zero rows in $B_{1.}^{(1)}$ in (5.1), these are all put at the top of $B^{(2)}$ in (5.6) by the (i,j) transformations with $i=1,\cdots,r_1$, leading to full row rank $B_{2.}^{(2)}$ in (5.4). But with Result 1 this shows $B_{2.}^{(k)}$ has full row rank for all $k>1$, so $B_{22}^{(k)}$ in (5.7) is nonsingular.

Now we show that $A_{11}^{(k)}$ and $A_{22}^{(k)}$ in (5.6), and $B_{22}^{(k)}$ in (5.8), are nonsingular. An (i,j) transformation on (5.8) with $i<j\leqq r$ leaves the (i,i) and (j,j) elements of A nonzero, while with $i\leqq r<j$ the $(1,1)$ element of A_{ij} is nonzero and the $(2,2)$ element of B_{ij} is nonzero, so with an analogous argument to that in the proof of Result 2, the (i,i) element of A stays nonzero. This completes the proof of (5.7). A similar argument completes the proof of (5.9).

It remains for us to support the sentence following (5.9). We see that in computing (5.6) from (5.8) $B_{3.}^{(k)}$ could only be affected by the left transformations V^H in (i,j) transformations with $r<j\leqq r+s$. But in these A_{ij} has zero last row so corresponding rows of the lower triangular A_{ij} and B_{ij} are parallel, and from (4.1) $A_{ij}\operatorname{adj}(B_{ij})=0$, and so $U_{ij}=V_{ij}=I$ in the algorithm. The same sort of argument holds for $A_{1.}^{(k)}$ in going from (5.6) to (5.8). This completes all the results we required, and so the algorithm is theoretically correct.

6. Numerical examples. The examples here were generated using the MATLAB system of Moler [12] on a VAX 11/750. They were not chosen to be particularly testing of the numerical stability of the algorithm, but rather to show how it handled different ranks and common row spaces.

In each of the examples we took $A(n\times n)$ $B(n\times n)$, with possibly some zero rows at the bottom, and carried out the QR decomposition to give upper triangular A and B. We took $n=6$ and built up our matrices as the products of random matrices of the ranks we chose to test. Since the algorithm is designed to produce parallel rows, we used this as a measure of convergence. For two vectors a and b we defined

$$\text{par}(a,b)=\begin{cases}0 & \text{if } \|a\|<\text{tol or } \|b\|<\text{tol},\\ \min\left\{\text{norm}\left(\dfrac{a}{\|a\|}\pm\dfrac{b}{\|b\|}\right)\right\}, & \text{otherwise};\end{cases} \tag{6.1}$$

and if a_i^T and b_i^T were the ith rows of our matrices at the end of a cycle, we took as an indicator of convergence

$$\text{error}=\sum_{i=1}^{n}\text{par}(a_i,b_i). \tag{6.2}$$

Here $\|a\|=(a^Ha)^{1/2}$ and we took tol $=10^{-14}$ for these test problems. We stopped when

$$\text{error}\leqq\text{tol} \tag{6.3}$$

and then computed

$$\sigma_i=\begin{cases}0 & \text{if } \|a_i\|=0 \text{ or } \|b_i\|=0,\\ \|a_i\|/\|b_i\| & \text{otherwise},\end{cases} \tag{6.4}$$

to give the equivalent of $\sigma_i=\alpha_i/\beta_i$ in (3.3). We write $s=[\sigma_1,\cdots,\sigma_n]$. When B is nonsingular we give t as the vector of computed singular values of AB^{-1} and cond (A) as the ratio of the largest to smallest singular value of A. One cycle takes A and B from upper triangular to lower triangular form, or vice versa; see (2.9).

Example 1. *Nonsingular A and B*, cond $(A)=24$, cond $(B)=21$.

cycle	1	2	3	4	5
error	2.5	.52	.0018	3×10**(−10)	2×10**(−16)

```
s = 4.356373396988140   3.492138554949318   1.858034577501202   1.001892079494734
     .300018648851983    .197330532251905

t = 4.356373396988142   3.492138554949318   1.858034577501202   1.001892079494734
     .300018648851983    .197330532251905
```

The algorithm gives as much accuracy as possible, but it does take a while to start converging.

Example 2. rank $(A)=4$, rank $(B)=3$, *common row space dimension* 2.

cycle	1	2	3	4	5
error	.47	2×10**(−17)			

After the QR decomposition the original matrices were:

```
COLUMNS 1 THRU 4
A = -1.483754207808792   -2.754569846168734   -1.901063717347766   -3.134725173442366
     .000000000000000    -.660367822613970    -.558467086690151    -.472306743711252
     .000000000000000     .000000000000000    -.190188592370571    -.444428846555170
     .000000000000000     .000000000000000     .000000000000000    -.047939573588338
     .000000000000000     .000000000000000     .000000000000000     .000000000000000
     .000000000000000     .000000000000000     .000000000000000     .000000000000000

COLUMNS 5 THRU 6
A = -1.814262790205398   -2.047678638962671
    -.263343099599686    -.264594139637406
    -.220470422637046    -.284218707808010
    -.149658467473861    -.071714348021438
     .000000000000000     .000000000000000
     .000000000000000     .000000000000000

COLUMNS 1 THRU 4
B = -2.152554420079003   -3.894294758406499   -2.368033003532614   -4.429539352490643
     .000000000000000     .074280121258419     .133400322312574     .209165610430773
     .000000000000000     .000000000000000     .073602580523686    -.176428622793232
     .000000000000000     .000000000000000     .000000000000000     .000000000000000
     .000000000000000     .000000000000000     .000000000000000     .000000000000000
     .000000000000000     .000000000000000     .000000000000000     .000000000000000

COLUMNS 5 THRU 6
B = -3.459002749639446   -3.411337868703009
     .079934376871628     .121676462551819
    -.387079235508506    -.368994080158650
    -.000000000000001    -.000000000000001
     .000000000000000     .000000000000000
     .000000000000000     .000000000000000
```

Note the small nonzero elements in positions (4, 5) and (4, 6) of B. These have been introduced by rounding errors.

After 2 cycles of the GSVD algorithm the matrices were:

```
COLUMNS 1 THRU 4
A = -.200073697315396    -.216325788985202    -.138100394035475    -.593297250589336
     .000000000000000    -.222361539578086     .118655281512143     .580017836681617
     .000000000000000     .000000000000000     .389711441881607     .218509802415124
     .000000000000000     .000000000000000     .000000000000000     .199123441354269
     .000000000000000     .000000000000000     .000000000000000     .000000000000000
     .000000000000000     .000000000000000     .000000000000000     .000000000000000

COLUMNS 5 THRU 6
A =  2.269980552374192    .000000000000000
    -4.039759246493302    .000000000000000
    -2.527788530781247    .000000000000000
    -1.805072397942648    .000000000000000
     .000000000000000     .000000000000000
     .000000000000000     .000000000000000

COLUMNS 1 THRU 4
B = .000000000000000      .000000000000000     .000000000000000     .000000000000000
    .000000000000000      .000000000000000     .000000000000000     .000000000000000
    .000000000000000      .000000000000000     .128833790821756     .072236642683449
    .000000000000000      .000000000000000     .000000000000000     .489752152327848
    .000000000000000      .000000000000000     .000000000000000     .000000000000000
    .000000000000000      .000000000000000     .000000000000000     .000000000000000
```

```
COLUMNS 5 THRU 6
B = -.000000000000001    .000000000000000
     .000000000000001    .000000000000000
    -.835655677041274    .000000000000000
   -4.439648521477558    .000000000000000
    6.954830268687245    .000000000000000
     .000000000000000    .000000000000000

s = .000000000000000     .000000000000000    3.024916362360086    .406580022992879
    .000000000000000     .000000000000000
```

Only two cycles were required, since the first separates out the common row space of dimension 2, and then the second produces parallel rows in this row space. This is because a 2×2 GSVD is solved exactly in one step. Note the small nonzeros in positions $(1, 5)$ and $(2, 5)$ of B. These were ignored by the use of the tolerance.

Example 3. rank (A) = rank $(B) = 4$, *common row space dimension* $= 3$.

cycle	1	2	3	4	5
error	.33	.003	2×10**(−9)	9×10**(−17)	

After the QR decomposition, the original matrices were:

```
COLUMNS 1 THRU 4
A = -1.483754207808792   -2.754569846168734   -1.901063717347766   -3.134725173442366
     .000000000000000    -.660367822613970    -.558467086690151    -.472306743711252
     .000000000000000     .000000000000000    -.190188592370571    -.444428846555170
     .000000000000000     .000000000000000     .000000000000000    -.047939573588338
     .000000000000000     .000000000000000     .000000000000000     .000000000000000
     .000000000000000     .000000000000000     .000000000000000     .000000000000000

COLUMNS 5 THRU 6
A = -1.814262790205398   -2.047678638962671
    -.263343099599686    -.264594139637406
    -.220470422637046    -.284218707808010
    -.149658467473861    -.071714348021438
     .000000000000000     .000000000000000
     .000000000000000     .000000000000000

COLUMNS 1 THRU 4
B = -3.123355429329773   -5.020366343436982   -2.972191466297970   -5.812154899974686
     .000000000000000     .207884167869281     .158033933963608     .182365676077579
     .000000000000000     .000000000000000     .107328952460706    -.021768478702607
     .000000000000000     .000000000000000     .000000000000000     .219692246210581
     .000000000000000     .000000000000000     .000000000000000     .000000000000000
     .000000000000000     .000000000000000     .000000000000000     .000000000000000

COLUMNS 5 THRU 6
B = -4.217468529709810   -4.339624584281506
     .227500717178881     .164471791036765
    -.275289576213577    -.216390069125860
     .269744345263956     .300087815368396
     .000000000000000     .000000000000000
     .000000000000000     .000000000000000
```

After 4 steps of the GSVD algorithm the matrices were:

```
     COLUMNS 1 THRU 4
     A = -.152535367220157    .215310202586866   -.277325505745866   -.659393620451283
          .000000000000000    .427810977395549   -.029228351378672   -.206360377382993
          .000000000000000    .000000000000000   -.212975875737842    .175581097281234
          .000000000000000    .000000000000000    .000000000000000    .194134471721972
          .000000000000000    .000000000000000    .000000000000000    .000000000000000
          .000000000000000    .000000000000000    .000000000000000    .000000000000000

     COLUMNS 5 THRU 6
    A = -3.535137973645832    .000000000000000
        -3.172167996024390    .000000000000000
         1.906135608940945    .000000000000000
         2.258917563058958    .000000000000000
          .000000000000000    .000000000000000
          .000000000000000    .000000000000000

     COLUMNS 1 THRU 4
       B = .000000000000000    .000000000000000    .000000000000000    .000000000000000
           .000000000000000    .121957526020080   -.008332225239963   -.058827852542294
           .000000000000000    .000000000000000    .144065810553351   -.118770414771316
           .000000000000000    .000000000000000    .000000000000000    .491824577188100
           .000000000000000    .000000000000000    .000000000000000    .000000000000000
           .000000000000000    .000000000000000    .000000000000000    .000000000000000

     COLUMNS 5 THRU 6
     B =   .000000000000000    .000000000000000
          -.904300687351262    .000000000000000
         -1.289390033379685    .000000000000000
          5.722791864318372    .000000000000000
          8.899102416565753    .000000000000000
           .000000000000000    .000000000000000

       s = .000000000000000   3.507868610954851   1.478323517008020    .394722998252534
           .000000000000000    .000000000000000
```

Again the algorithm behaved as well as could be hoped.

These examples show that the theoretical behaviour described in § 5 also occurs in practice. This is not too surprising since each of A and B is subject to unitary transformations only. There is one caveat here, and that is that the nonzero singular values of A and B are all in a reasonable range, so none of the final nonzero rows is very small. If they were, we would not be able to use (6.3) as our test for convergence, because rounding errors would stop (6.2) being that small. It is clear that improvements will be needed in a production code.

7. Comments. We have suggested a straightforward algorithm for computing the GSVD of given matrices $A(m\times n)$ and $B(p\times n)$, or the CSD of a partitioned unitary matrix. The approach is a unified one whereby the A and B matrices are transformed separately with unitary transformations (orthogonal transformations when A and B are real) and no difficult rank decisions need be made during the computation. Because of its simplicity the algorithm is fairly easy to program, however the proof of correctness here was long and involved, and it is hoped that someone else can derive a more brief and straightforward proof.

For most problems the criterion (2.2) need not be enforced, however it would appear to be theoretically necessary as the following example suggests. Let

$$C=\begin{pmatrix}\alpha & 0 & \beta\\ 0 & \alpha & 0\\ 0 & 0 & \alpha\end{pmatrix}, \qquad (7.1)$$

and apply Kogbetliantz' algorithm to this, exchanging rows i and j, and columns i and j, each (i,j) transformation. After 6 such transformations the matrix will be identical to its starting value (7.1), so there is no convergence.

The numerical behaviour of the algorithm has so far been exemplary. By following the work of Wilkinson [20] we know that with correct use of unitary rotations our computed matrices $A^{(k)}$ and $B^{(k)}$ will be the exact result of applying unitary matrices $U^{(k)}$, $V^{(k)}$ and $Q^{(k)}$ to slightly perturbed initial data $A+\delta A^{(k)}$ and $B+\delta B^{(k)}$, so that if we obtain convergence we can be confident about our results. We have not proven convergence in the presence of rounding errors, but all the tests run so far have converged pleasingly swiftly, rarely taking more than 6 cycles to give full precision using MATLAB on a VAX 11/750.

The ultimate convergence appears to be at least quadratic. The method implicitly applies Kogbetliantz' method, for which quadratic convergence has recently been proven when there are no pathologically close singular values [21], and work on the general case is in progress. One delightful observation that may contribute to the understanding of convergence is the following.

RESULT 3. Kogbetliantz' serial-cyclic algorithm applied to nonsingular triangular C also implicitly applies the algorithm to C^{-1}.

This means the algorithm is diagonalizing C and C^{-1} in exactly the same way at the same time, and this could help convergence. The result can be proven by examining the form of C and C^{-1} during the elimination of the offdiagonal elements of the first row of C as in (2.3), and using induction. Note however that the rate of convergence is similar for triangular and full C.

For our algorithm the above result means that if A and B are nonsingular the algorithm is implicitly diagonalizing AB^{-1} and BA^{-1} simultaneously, so that A and B are treated completely equally. However the iterative algorithm also works with singular square A and B, and in this case we have the following.

RESULT 4. The algorithm (5.13) implicitly applies Kogbetliantz' algorithm to both $A\,\mathrm{adj}\,(B)$ and $B\,\mathrm{adj}\,(A)$.

This is not a very significant result if for example the rank of B is less than $n-1$, for then $\mathrm{adj}\,(B)=0$. However it does indicate that this is an equal opportunity algorithm.

8. Systolic array implementation.

A brief sketch will be given here describing how this GSVD algorithm can be implemented for systolic array computations. This was first worked out by W. M. Gentleman [22], but since he did not have time to write it up, this outline is included to round out the presentation. The term "rotation" will be used loosely to mean either a 2×2 unitary rotation or a 2×2 elementary unitary hermitian.

The important step is to implement Kogbetliantz' basic algorithm in a way that can be transferred effectively to systolic array processors. The physical ordering presented in (2.1) and (2.3) cannot be used because it combines rows and columns i and j where the pair (i,j) cycles through $(1,2)$, $(1,3),\cdots,(1,n)$; $(2,3),\cdots,(2,n)$; $\cdots$; $(n-1,n)$, which does not allow a lot of concurrency. The ideal ordering will only apply rotations to adjacent rows and to adjacent columns. The ordering suggested by Gentleman starts with an upper triangular matrix as in (2.3), and gives theoretically identical results to the one in (2.3), but one cycle leaves the matrix in upper triangular form rather than lower triangular form as in (2.3). In fact if

$$\begin{pmatrix}-c_1 & s_1\\ s_1 & c_1\end{pmatrix} \quad\text{and}\quad \begin{pmatrix}-c_2 & s_2\\ s_2 & c_2\end{pmatrix} \tag{8.1}$$

are the nontrivial elements of the left and right rotations respectively for one 2×2 SVD in (2.3), then in Gentleman's ordering the left and right rotations have the diagonal blocks

$$\begin{pmatrix} s_1 & c_1 \\ -c_1 & s_1 \end{pmatrix} \quad \text{and} \quad \begin{pmatrix} s_2 & -c_2 \\ c_2 & s_2 \end{pmatrix} \tag{8.2}$$

respectively.

Thus the ordering corresponds to the same rotations on the same elements as in (2.3), with each rotation followed by a permutation. We illustrate this with the equivalent diagram to (2.3), but include indices which indicate the position of each corresponding element in (2.3). The element to be eliminated is underlined immediately prior to elimination.

$$\begin{array}{cccc} 11 & \underline{12} & 13 & 14 \\ & 22 & 23 & 24 \\ & & 33 & 34 \\ & & & 44 \end{array} \rightarrow \begin{array}{cccc} 22 & & 23 & 24 \\ & 11 & \underline{13} & 14 \\ & & 33 & 34 \\ & & & 44 \end{array} \rightarrow \begin{array}{cccc} 22 & 23 & 21 & 24 \\ & 33 & & 34 \\ & & 11 & \underline{14} \\ & & & 44 \end{array} \rightarrow \begin{array}{cccc} 22 & \underline{23} & 24 & 21 \\ & 33 & 34 & 31 \\ & & 44 & \\ & & & 11 \end{array} \tag{8.3}$$

$$\rightarrow \begin{array}{cccc} 33 & & 34 & 31 \\ & 22 & \underline{24} & 21 \\ & & 44 & \\ & & & 11 \end{array} \rightarrow \begin{array}{cccc} 33 & \underline{34} & 32 & 31 \\ & 44 & & 41 \\ & & 22 & 21 \\ & & & 11 \end{array} \rightarrow \begin{array}{cccc} 44 & & 42 & 41 \\ & 33 & 32 & 31 \\ & & 22 & \underline{21} \\ & & & 11 \end{array}$$

Thus the rotations are in the (i, j) plane where (i, j) goes through $(1, 2), (2, 3), (3, 4), \cdots, (n-1, n)$; $(1, 2), (2, 3), \cdots, (n-2, n-1)$; $\cdots$; $(1, 2), (2, 3)$; $(1, 2)$. Only adjacent rows or columns are combined, so this is suitable for implementation using systolic array processors. However, if sequential computations are used, this approach can be made to give numerically identical results to that in (2.3). This parallels the ordering that Stewart [23] uses for the unsymmetric eigenvalue problem.

If we refer to (8.3) as the forward cycle, then the next cycle is referred to as the reverse cycle, and has rotations in the planes $(n-1, n)$; $(n-2, n-1), \cdots, (1, 2)$; $(n-1, n), (n-2, n-1), \cdots, (2, 3)$; $\cdots$; $(n-1, n), (n-2, n-1)$; $(n-1, n)$. Choosing the angles correctly will mean that this gives exactly the same upper triangular matrix as the two cycles described by (2.3) and its following paragraph. Sweep would perhaps be a more satisfactory term than cycle here.

The way this method can be applied to obtaining the GSVD of A and B, or the SVD of AB as in [8], is now easy to see. If A, B and C are upper triangular, and A_{ij}, B_{ij} and C_{ij} are 2×2 matrices with the (i, i), (i, j), (j, i) and (j, j) elements of A, B and C respectively, and $j = i+1$, then

$$A_{ij} = C_{ij}B_{ij} \quad \text{if } A = CB \quad \text{and} \tag{8.4}$$

$$C_{ij} = A_{ij}B_{ij} \quad \text{if } C = AB. \tag{8.5}$$

Thus if we keep A and B upper triangular throughout the computation, the three implicitly defined elements of C required for each 2×2 SVD in (8.3) will be available just as they were in (2.7) for example. If we now concentrate on the GSVD of A and B, the computation corresponding to (2.7) will have the rotation k in (2.7) followed by a permutation, and the rotation k' followed by a permutation, but k'' will now be chosen to eliminate the $(2, 1)$ elements of A_{ij} and B_{ij}, rather than the $(1, 2)$ element as in (2.7). Since with $j = i+1$ this gives the correct transformation for C in (8.3), and keeps both A and B upper triangular, the computations for C in (8.3) can be carried through implicitly by only applying rotations to adjacent rows and to adjacent columns of both A and B. And it can be seen from (2.9) that these occur in exactly the same order as for C in (8.3). What is more with sequential computations this can be made numerically identical to the earlier GSVD, since the permutations after k and k' in (2.7) ensure the new k'' is acting on the same elements as k'' in (2.7).

We will leave the details of the systolic array implementation to others. It suffices here to point out in (8.3) for example, that while the element marked $\underline{12}$ is being eliminated in this forward cycle, the element in the position marked $\underline{34}$ in that first matrix can be eliminated as part of the previous reverse cycle. Similarly $\underline{14}$ and $\underline{23}$ in the 3rd and 4th matrices can be eliminated simultaneously, as can $\underline{34}$ in the 6th and $\underline{21}$ (corresponding to the reverse cycle) in the 7th.

In conclusion this very nice ordering initially proposed by Stewart for the eigenproblem [23], and now by Gentleman [22] for the SVD and GSVD, is a very natural and easy one to follow and implement, and appears to be the correct approach to use in implementing such algorithms using systolic array processors.

Acknowledgments. I am deeply indebted to Binay Bhattacharya and Mike Saunders for the many insights that arose from the hours of discussion and computing that went into our earlier attempts at this problem. I am also grateful to Mike Heath for his steadfast insistence on treating both matrices with complete equality in problems like this. It was after working with Mike Heath, Alan Laub and Bob Ward on [8] that I realized the correct approach to the algorithm here. Gene Golub contributed very positively during the difficult proofs of § 5. Morven Gentleman rounded out the work very nicely by showing how to implement the algorithm using systolic array processors.

REFERENCES

[1] B. K. Bhattacharya, C. C. Paige and M. A. Saunders, unpublished code for the generalized singular value decomposition, 1980.

[2] R. P. Brent, F. T. Luk and C. van Loan, *Computation of the singular value decomposition using mesh-connected processors*, Technical Report CS-52, Dept Computer Science, Cornell Univ., Ithaca, NY, 1983.

[3] C. Davis and W. M. Kahan, *The rotation of eigenvectors by a perturbation.* III, SIAM J. Numer. Anal., 7 (1970), pp. 1-46.

[4] G. E. Forsythe and P. Henrici, *The cyclic Jacobi method for computing the principal values of a complex matrix*, Trans. Amer. Math. Soc., 94 (1960), pp. 1-23.

[5] G. Golub and W. Kahan, *Calculating the singular values and pseudo-inverse of a matrix*, SIAM J. Numer. Anal., 2 (1965), pp. 205-224.

[6] G. H. Golub and C. Reinsch, *Singular value decomposition and least squares solutions*, Numer. Math., 14 (1970), pp. 403-420.

[7] G. H. Golub and C. F. van Loan, *Matrix Computations*, The Johns Hopkins Univ. Press, Baltimore, MD, 1983.

[8] M. T. Heath, A. J. Laub, C. C. Paige and R. C. Ward, *Computing the* SVD *of product of two matrices*, this Journal, 7 (1986), pp. 1147-1159.

[9] C. G. J. Jacobi, *Über eine neue Auflösungsart der bei der Methode der kleinsten Quadrate vorkommenden linearen Gleichungen*, (*On a new way of solving linear equations occurring in the method of least squares*), Astronom. Nachr., 22 (1845), pp. 297-306. (Werke, Vol III, 1884, pp. 468-478.)

[10] E. G. Kogbetliantz, *Diagonalization of general complex matrices as a new method for solution of linear equtions*, Proc. Internat. Congr. Math., Amsterdam, 2 (1954), pp. 356-357.

[11] ———, *Solution of linear equations by diagonalization of coefficients matrix*, Quart. Appl. Math., 13 (1955), pp. 123-132.

[12] C. B. Moler, MATLAB *Users' Guide*, Technical Report CS81-1, Dept. of Computer Science, Univ. New Mexico, Albuquerque, NM, 1981.

[13] C. B. MOLER AND G. W. STEWART, *An algorithm for generalized matrix eigenvalue problems*, SIAM J. Numer. Anal., 10 (1973), pp. 241-256.

[14] C. C. PAIGE, *The general linear model and the generalized singular value decomposition*, Linear Algebra Appl., 70 (1985), pp. 269-284.

[15] C. C. PAIGE AND M. A. SAUNDERS, *Towards a generalized singular value decomposition*, SIAM, J. Numer. Anal., 18 (1981), pp. 398-405.

[16] G. W. STEWART, *On the perturbation of pseudo-inverses, projections and linear least squares problems*, SIAM Rev., 19 (1977), pp. 634-662.

[17] ———, *Computing the CS decompostion of a partitioned orthonormal matrix*, Numer. Math., 40 (1982), pp. 297-306.

[18] C. F. VAN LOAN, *Generalizing the singular value decomposition*, SIAM J. Numer. Anal., 13 (1976), pp. 76-83.

[19] ———, *Computing the CS and generalized singular value decompositions*, Technical Report CS-604, Dept. Computer Science, Cornell Univ., Ithaca, NY, 1984.

[20] J. H. WILKINSON, *The Algebraic Eigenvalue Problem*, Clarendon Press, Oxford, 1965.

[21] C. C. PAIGE AND P. VAN DOOREN, *On the quadratic convergence of Kogbetliantz's algorithm for computing the singular value decomposition*, Linear Algebra Appl., to appear.

[22] W. M. GENTLEMAN, personal communication at the IXth Gatlinburg meeting on Numerical Linear Algebra, University of Waterloo, Ontario, Canada, July 8-14, 1984.

[23] G. W. STEWART, Computer Science Technical Report 1321, Univ. Maryland, College Park, MD, 1983.