Computing the Generalized Singular Value Decomposition*

Zhaojun Bai† James W. Demmel‡

Abstract

We present a variation of Paige's algorithm for computing the generalized singular value decomposition (GSVD) of two matrices A and B. There are two innovations. The first is a new preprocessing step which reduces A and B to upper triangular forms satisfying certain rank conditions. The second is a new 2×2 triangular GSVD algorithm, which constitutes the inner loop of Paige's algorithm. We present proofs of stability and high accuracy of the 2×2 GSVD algorithm, and demonstrate it using examples on which all previous algorithms fail.

1 Introduction

The purpose of this paper is to describe a variation of Paige's algorithm [28] for computing the following generalized singular value decomposition (GSVD) introduced by Van Loan [33], and Paige and Saunders [25]. This is also called the quotient singular value decomposition (QSVD) in [8].

Theorem 1.1 *Let $A \in \mathbb{R}^{m\times n}$ and $B \in \mathbb{R}^{p\times n}$ have* $\mathrm{rank}(A^T, B^T) = n$. [1] *Then there are orthogonal matrices U, V and Q such that*

$$U^TAQ = \Sigma_1 R, \qquad V^TBQ = \Sigma_2 R, \tag{1.1}$$

where R is a $n \times n$ upper triangular and nonsingular, and

$$\Sigma_1 = \begin{array}{c} \\ l \\ k \\ m-l-k \end{array} \begin{array}{c} \begin{array}{ccc} l & k & n-l-k \end{array} \\ \begin{pmatrix} I_1 & & \\ & D_1 & \\ & & O_1 \end{pmatrix} \end{array}, \qquad \Sigma_2 = \begin{array}{c} \\ p-n+l \\ k \\ n-l-k \end{array} \begin{array}{c} \begin{array}{ccc} l & k & n-l-k \end{array} \\ \begin{pmatrix} O_2 & & \\ & D_2 & \\ & & I_2 \end{pmatrix} \end{array}, \tag{1.2}$$

$I_1 \in \mathbb{R}^{l\times l}$ and $I_2 \in \mathbb{R}^{(n-l-k)\times(n-l-k)}$ are identity matrices, $O_1 \in \mathbb{R}^{(m-l-k)\times(n-l-k)}$ and $O_2 \in \mathbb{R}^{(p-n+l)\times l}$ are zero matrices,

$$D_1 = \mathrm{diag}(\alpha_{l+1}, \ldots, \alpha_{l+k}), \qquad D_2 = \mathrm{diag}(\beta_{l+1}, \ldots, \beta_{l+k}), \tag{1.3}$$

$$1 > \alpha_{l+1} \geq \cdots \geq \alpha_{l+k} > 0, \quad 0 < \beta_{l+1} \leq \cdots \leq \beta_{l+k} < 1, \quad \alpha_i^2 + \beta_i^2 = 1. \tag{1.4}$$

*This work was supported in part by NSF Grants ASC-9005933. The first author was also supported in part by NSF grant ASC-9102963. The second author was also supported in part by DARPA via a subcontract to the University of Tennessee under grant DAAL03-91-C-0047.

†Department of Mathematics, University of Kentucky, Lexington, KY 40506.

‡Computer Science Division and Mathematics Department, University of California, Berkeley, CA 94720.

[1] The assumption that $\mathrm{rank}(A^T, B^T) = n$ is not essential but simplifies exposition.

The GSVD is a generalization of the singular value decomposition (SVD) in the sense that if B is the identity matrix, then the GSVD of A and B is the SVD of A. Moreover, if B is nonsingular, then the GSVD of A and B reduces to the SVD of AB^{-1}. If $(A^T, B^T)^T$ has orthonormal columns, then the GSVD of A and B is the CS decomposition [31]. The pairs (α_i, β_i) defined by the diagonal elements of Σ_1 and Σ_2 are called the *generalized singular value pairs (GSV pairs)*. The quotient $\lambda_i = \alpha_i/\beta_i$ is called a *generalized singular value (GSV)*. Note that the λ_i are the square roots of the eigenvalues of the symmetric pencil $A^TA - \lambda B^TB$.

The GSVD of two matrices A and B is a tool used in many applications, such as the Kronecker canonical form of a general matrix pencil [22], the linearly constrained least-squares problem [35, 5], the general Gauss-Markov linear model [27, 3], the generalized total least squares problem [21], and real time signal processing [30]. As a further generalization of the SVD, Ewerbring and Luk [13], Zha [36] proposed a generalized SVD for matrix triplets, and De Moor, Golub and Zha [8, 9] have generalized the SVD into a factorization of any number of matrices. For all these applications and multi-matrix generalization of the SVD, the development of a stable and efficient algorithm for computing the GSVD of two matrices is a basic problem.

Stewart [31] and Van Loan [34] proposed two algorithms for computing the GSVD. Their algorithms have two phases: The first phase is to compute the QR decomposition (or the SVD if necessary) of $(A^T, B^T)^T$. The second phase is to compute the CS decomposition. Paige's algorithm is a Jacobi-Kogbetliantz approach [28], which applies orthogonal transformations to A and B separately without the CS decomposition. It also has two phases:

(1) Reduce matrices A and B to the following forms

$$U^TAP = \begin{array}{c} \\ r \\ q \\ m-t \end{array} \!\!\begin{array}{c} \begin{array}{ccc} r & q & n-t \end{array} \\ \begin{pmatrix} A_{11} & A_{12} & A_{13} \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \end{array}, \quad V^TBP = \begin{array}{c} \\ r \\ q \\ p-t \end{array} \!\!\begin{array}{c} \begin{array}{ccc} r & q & n-t \end{array} \\ \begin{pmatrix} B_{11} & B_{12} & B_{13} \\ 0 & B_{22} & B_{23} \\ 0 & 0 & 0 \end{pmatrix} \end{array}, \tag{1.5}$$

where $m \times m$ matrix U and $p \times p$ matrix V are orthogonal, P is a $n \times n$ permutation matrix, $A_{11} \in \mathbb{R}^{r \times r}$ is nonsingular upper triangular, $B_{11} \in \mathbb{R}^{r \times r}$ is upper triangular, $t = r + q$, and if $q > 0$, $B_{22} \in \mathbb{R}^{q \times q}$ is nonsingular upper triangular.

(2) Compute the GSVD of two $n \times n$ upper triangular matrices of forms (1.5) by a generalized Kogbetliantz algorithm[2].

Phase 1 can be done first by the QR factorization with column pivoting [17] of matrix A and determine the rank r of A, meanwhile permuting the columns of matrix B in the same way, and then applying the QR factorization with column pivoting to the block of the last $p - r$ rows and $n - r$ columns of B and obtain the rank q of the block; this yields the forms (1.5) [4]. Phase 2 is iterative.

In this paper, we will present a variation of Paige's algorithm for computing the GSVD. There are two innovations. The first is as follows: in [28], it is assumed (without providing detail) that in (1.5) the nonzero part of V^TBP has full row rank. It is known that it is complicated to choose V to guarantee this condition and P may not be a permutation matrix. However in the preprocessing step (1.5), we do not require this condition, and so we can simply use conventional QR factorization with column pivoting. Moreover, note that the GSVD is independent of column scaling of A and B. The forms (1.5) preserve this property.

The second innovation is a new 2×2 triangular GSVD algorithm, which constitutes the inner loop of Paige's algorithm. We will present proofs of stability and high accuracy of our method, and demonstrate it using examples on which all previous algorithms fail. Hereafter, we assume that A and B have been preprocessed to the upper trapezoidal forms (1.5).

[2] We may need to add zero rows or columns to get square matrices. This is not essential but it simplifies the description.

The numerical technique developed in this paper can be extended to deal with the numerical computation of other closely related decompositions such as the CS decomposition and the product SVD of two matrices [20, 15]. We will not go into the details.

The rest of the paper is organized as the follows: §2 reviews the Kogbetliantz algorithm for computing the SVD of a triangular matrix, and Paige's generalization of the Kogbetliantz algorithm for computing the GSVD. §3 explores the inner loop of Paige's algorithm, which includes the GSVD of a 2×2 matrix in terms of exact and floating point arithmetic. In §4, we describe the overall algorithm. The last section reports the results of numerical experiments. In the appendix, we include Demmel and Kahan's 2×2 triangular SVD code, which has not been published in its entirety before, and plays an important role in our algorithm.

2 Paige's GSVD Algorithm

To describe Paige's algorithm, we first review the Kogbetliantz algorithm [23] for computing the SVD of an upper triangular matrix A. Then we describe Paige's algorithm for computing the GSVD of A and B with B nonsingular. Finally, we discuss how to generalize the idea to the case where B is ill-conditioned or singular.

2.1 Kogbetliantz algorithm for the SVD of a triangular matrix

The Kogbetliantz algorithm [23] is a kind of Jacobi scheme. Assume that the kth transformation of the algorithm operates on the rows and columns i and j of A, let A_{ij} be the 2×2 submatrix subtended by rows and columns i and j of A. Let the rotation matrices $U_k = \mathrm{rot}(c_u, s_u)$ and $V_k = \mathrm{rot}(c_v, s_v)$ be chosen [3] so that

$$U_k^T A_{ij} V_k = \mathrm{diag}(\gamma_{ii}, \gamma_{jj})$$

is the SVD of A_{ij}, where $c_u = \cos\phi_k, s_u = \sin\phi_k$ and $c_v = \cos\psi_k, s_v = \sin\psi_k$. Let $\hat{U}_k$ and $\hat{V}_k$ be identity matrices with (i,i), (i,j), (j,i) and (j,j) elements replaced by the (1,1), (1,2), (2,1) and (2,2) elements of U_k and V_k respectively. Then let

$$A_{k+1} = \hat{U}_k^T A_k \hat{V}_k,$$

where $A_0 = A$. After the first sweep through all the (i,j) in row cyclic order, an upper triangular matrix A will become lower triangular. The second sweep will restore upper triangular form, and so on [20, 19]. There is a literature on the different sweep orders for sequential and parallel computations besides the conventional row and column order, for example [24].

Forsythe and Henrici [16] considered the convergence of the row cyclic Kogbetliantz algorithm. Fernando [14] proved a global convergence theorem under the assumption that one of the rotation angles $\{\phi_k, \psi_k\}$ at each (i,j) transformation lies in a closed interval $J \subset (-\pi/2, \pi/2)$, i.e.,

$$\phi_k \in J \quad \text{or} \quad \psi_k \in J, \qquad k = 1, 2, \ldots, . \tag{2.1}$$

This is the condition that our algorithm will satisfy. Furthermore, it has been proved that the cyclic Kogbetliantz algorithm ultimately converges quadratically [29, 2, 7].

[3] Throughout this paper, we use $\mathrm{rot}(c, s)$ to denote the rotation matrix $\begin{pmatrix} c & s \\ -s & c \end{pmatrix}$.

2.2 The Generalization of the Kogbetliantz Algorithm for the GSVD

We begin by computing the GSVD of two upper triangular matrices A and B with B nonsingular. It is known that this is equivalent to computing the SVD of the triangular matrix $C = AB^{-1}$. Of course, it is unwise to form C explicitly. We note that a sweep of the Kogbetliantz algorithm applied to C will make it lower triangular. This means that there are orthogonal matrices U_1 and V_1 such that

$$U_1^T C V_1 = C_1, \tag{2.2}$$

where C_1 is lower triangular. Recasting (2.2) as $U_1^T A = C_1 V_1^T B$, we see that if we can determine an orthogonal matrix Q_1 satisfying

$$U_1^T A Q_1 = A_1, \quad V_1^T B Q_1 = B_1,$$

where A_1 and B_1 are lower triangular, then $C_1 = A_1 B_1^{-1}$. This means that using a sweep of the Kogbetliantz algorithm on the upper triangular C to get the lower triangular C_1 is equivalent to the problem of finding orthogonal matrices U_1, V_1 and Q_1 so that $U_1^T A Q_1$ and $V_1^T B Q_1$ are lower triangular. Heath *et al* [20], Paige [28] and Hari and Veselić [19] have shown that we may take advantage of the triangular structures of A and B and the ordering of sweeps to get the desired orthogonal transformations U_1, V_1 and Q_1 without forming AB^{-1} explicitly. Specifically, at the (i,j) transformation, the needed 2×2 submatrix C_{ij} of C is given by

$$C_{ij} = A_{ij} B_{ij}^{-1} = \begin{pmatrix} a_{ii} & a_{ij} \\ 0 & a_{jj} \end{pmatrix} \begin{pmatrix} b_{ii} & b_{ij} \\ 0 & b_{jj} \end{pmatrix}^{-1}, \tag{2.3}$$

where a_{ij} and b_{ij} are the elements subtended by the rows and columns i and j of the updated A and B, respectively. By using the SVD of C_{ij}: $U_{ij}^T C_{ij} V_{ij} = \mathrm{diag}(\tilde{c}_{ii}, \tilde{c}_{jj})$, we have

$$U_{ij}^T A_{ij} = \mathrm{diag}(\tilde{c}_{ii}, \tilde{c}_{jj}) V_{ij}^T B_{ij}.$$

This shows that the corresponding rows of $U_{ij}^T A_{ij}$ and $V_{ij}^T B_{ij}$ are parallel. Hence if we choose rotation Q_{ij} so that $V_{ij}^T B_{ij} Q_{ij}$ is lower triangular, then $U_{ij}^T A_{ij} Q_{ij}$ must also be lower triangular, which is just the GSVD of the 2×2 triangular matrices A_{ij} and B_{ij}. With this observation, we see that after completing a sweep in row order, the desired U_1, V_1 and Q_1 are the products $U_{12}U_{13} \cdots U_{n-1,n}$, $V_{12}V_{13} \cdots V_{n-1,n}$ and $Q_{12}Q_{13} \cdots Q_{n-1,n}$, respectively. By the end of the row cyclic sweep, we obtain lower triangular matrices A_1 and B_1.[4] Then the next sweep consists of zeroing lower off-diagonal elements of $C_1 = A_1 B_1^{-1}$ in column order to return it to upper triangular form, and so on. Overall, we are actually carrying out the Kogbetliantz algorithm to diagonalize the implicitly defined matrix C. Upon convergence, this gives $U^T(AB^{-1})V = \Sigma$, a diagonal matrix. That is

$$U^T A\, Q = \Sigma \cdot V^T B\, Q,$$

i.e., the ith rows of $U^T AQ$ and $V^T BQ$ are parallel, which is the desired GSVD of A and B.

In general, if B is ill-conditioned with respect to inversion or B is singular after phase 1, then using B_{ij}^{-1} is not recommended. Paige [28] suggests using

$$C_{ij} = A_{ij} \cdot \mathrm{adj}(B_{ij}) = \begin{pmatrix} a_{ii} & a_{ij} \\ 0 & a_{jj} \end{pmatrix} \begin{pmatrix} b_{jj} & -b_{ij} \\ 0 & b_{ii} \end{pmatrix} \tag{2.4}$$

[4] By incorporating Gentleman's suggested row and column permutations [28] after each transformation, we need only use an upper triangular array to carry out the computation. But for clearer exposition, we will use the entire square array in this paper.

instead of C_{ij} in (2.3), where $\mathrm{adj}(B_{ij})$ stands for the adjugate of B_{ij}. Since $B_{ij}\cdot\mathrm{adj}(B_{ij}) = \det(B_{ij})I$, it seems to be direct and natural to use $\mathrm{adj}(B_{ij})$ instead of B_{ij}^{-1}. The incorporation of (2.4) into the above procedure circumvents the numerical difficulties when B_{ij} is ill-conditioned with respect to inversion or B_{ij} is singular. But it also introduces two questions. First, are there still rotation matrices U_{ij}, V_{ij} and Q_{ij} such that $U_{ij}^T A_{ij} Q_{ij}$ and $V_{ij}^T B_{ij} Q_{ij}$ are the GSVD of 2×2 matrices A_{ij} and B_{ij}? Second, does the scheme converge to our required GSVD forms of A and B? The following section will address these questions.

3 The GSVD of 2×2 Triangular Matrices

As we see in §2, the kernel of computing the GSVD using a generalized Kogbetliantz algorithm is the computation of the GSVD of 2×2 matrices. In this section, we first discuss the computation of the 2×2 GSVD for different possible 2×2 matrices A_{ij} and B_{ij} in exact arithmetic, and then we will discuss the computation in the presence of floating point arithmetic.

3.1 The 2×2 GSVD in exact arithmetic

When A and B are processed to have upper trapezoidal forms (1.5), we see that at the (i, j) transformation, the 2×2 matrices A and B are of the forms[5]

$$A = \begin{pmatrix} a_{11} & a_{12} \\ 0 & a_{22} \end{pmatrix} \quad \text{and} \quad B = \begin{pmatrix} b_{11} & b_{12} \\ 0 & b_{22} \end{pmatrix}, \tag{3.1}$$

where $a_{11} \neq 0$, if A is nonzero. We have the following lemma:

Lemma 3.1 *There exist 2×2 rotation matrices U, V and Q, such that*

$$\tilde{A} = U^T A Q = \begin{pmatrix} \tilde{a}_{11} & 0 \\ \tilde{a}_{21} & \tilde{a}_{22} \end{pmatrix}, \quad \tilde{B} = V^T B Q = \begin{pmatrix} \tilde{b}_{11} & 0 \\ \tilde{b}_{21} & \tilde{b}_{22} \end{pmatrix}$$

is the GSVD of A and B. Moreover,
(a) $\tilde{a}_{11} \neq 0$ if A is nonzero,
(b) $\tilde{b}_{22} \neq 0$ if both A and B are nonzero, except that
(c) if the first rows of A and B are parallel and the second rows are zero, then $U = V = I$, and Q can be chosen to zero the (1,2) entries of A and B simultaneously.

Proof. The proof proceeds by considering all possible cases. If B is nonsingular, the lemma follows immediately by §2.2. If A or B is zero, the results are trivial. The remaining cases are for B singular but not zero. This includes the following three cases, where $C = A \cdot \mathrm{adj}(B)$:

(1) $B = \begin{pmatrix} b_{11} & b_{12} \\ 0 & 0 \end{pmatrix}$ with $b_{11} \neq 0$. In this case, $C = \begin{pmatrix} 0 & a_{12}b_{11} - a_{11}b_{12} \\ 0 & a_{22}b_{11} \end{pmatrix} \equiv \begin{pmatrix} 0 & c_{12} \\ 0 & c_{22} \end{pmatrix}$. If $c_{12} = 0$, i.e., the first row vectors of A and B are parallel, then if c_{22} is also equal to zero, $U = V = I$. Q_{ij} is chosen to zero (1,2) entry of A and must also zero the (1,2) entry of B, yielding the result (c). If $c_{22} \neq 0$, then both U and V are chosen as permutation matrices. Q is chosen to zero the (1,2) entry of $U^T A$.

If $c_{12} \neq 0$, then U is chosen to zero (2,2) entry of C and $V = \mathrm{rot}(0, 1)$. $V^T B$ has second row nonzero. The lemma follows by choosing Q to zero (1,2) entry of $U^T A$.

---

[5]For simplicity of exposition, we drop the subscript ij from the 2×2 triangular matrices A_{ij} and B_{ij}.

(2) $B = \begin{pmatrix} 0 & b_{12} \\ 0 & b_{22} \end{pmatrix}$ with $b_{22} \neq 0$. Hence $C = a_{11} \begin{pmatrix} b_{22} & -b_{12} \\ 0 & 0 \end{pmatrix}$. Then $U = I$, and V is chosen to zero (1,2) entry of C, i.e. to zero the (1,2) entry of B. The lemma follows by choosing Q to zero (1,2) entry of A.

(3) $B = \begin{pmatrix} 0 & b_{12} \\ 0 & 0 \end{pmatrix}$ with $b_{12} \neq 0$. We see that $C = \begin{pmatrix} 0 & -a_{11}b_{12} \\ 0 & 0 \end{pmatrix}$. Then we can choose $U = I$, $V = \mathrm{rot}(0,1)$. Therefore the second row of $V^T B$ is nonzero. The lemma follows by choosing Q to zero (1,2) entry of $U^T A$. ∎

It has been shown by induction (see [28, 4]) that with the properties of Lemma 3.1, a sweep in row order with possible reordering takes the initial upper trapezoidal forms (1.5) of A and B into the forms

$$A_1 = U_1^T A Q_1 = \begin{array}{c} \\ r \\ n-r \end{array} \!\!\begin{array}{c} \begin{array}{cc} r & n-r \end{array} \\ \begin{pmatrix} A_{11} & 0 \\ 0 & 0 \end{pmatrix} \end{array}, \quad B_1 = V_1^T B Q_1 = \begin{array}{c} \\ r \\ n-r \end{array} \!\!\begin{array}{c} \begin{array}{cc} r & n-r \end{array} \\ \begin{pmatrix} B_{11} & 0 \\ B_{21} & B_{22} \end{pmatrix} \end{array}, \tag{3.2}$$

where A_{11}, B_{11} and B_{22} are lower triangular, and A_{11}, B_{22} are nonsingular. B_{11} may be singular, but there must exist nonzero diagonal elements in the nonzero rows of B_{11}.

From (3.2), we see that at (i, j) transformation in column ordering, the 2×2 matrices A and B are lower triangular matrices, where if A is singular, then A is either the zero matrix or its second row is zero, and moreover, if $b_{22} = 0$, then $b_{21} = 0$. By a similar argument as in Lemma 3.1, we can show that there are 2×2 orthogonal matrices U, V and Q such that $\tilde{A} = U^T A Q$ and $\tilde{B} = V^T B Q$ both are upper triangular, and the GSVD of A and B. The proof of Lemma 3.1 suggests the following algorithm, where for brevity, we omit the part for lower triangular matrices.

Algorithm 1 (The 2×2 GSVD algorithm).
form $C = A \cdot \mathrm{adj}(B)$;
compute the SVD of C: $U^T C V = \mathit{diag}(\sigma_1, \sigma_2)$;
form the products $G = U^T A$, $H = V^T B$;
if A *is nonzero, then*
  determine Q *to zero out (1,2) entry of* G;
else
  determine Q *to zero out (1,2) entry of* H;
end if
$\tilde{A} = GQ$; $\tilde{B} = HQ$; $\tilde{a}_{12} = 0$; $\tilde{b}_{12} = 0$;

Again, from [28, 4], at the end of the second sweep, we have $A_2 = U_2^T A_1 Q_2$ and $B_2 = V_2^T B_1 Q_2$, such that

$$A_2 = \begin{array}{c} \\ r_1 \\ r_2 \\ n-r \end{array} \!\!\begin{array}{c} \begin{array}{ccc} r_1 & r_2 & n-r \end{array} \\ \begin{pmatrix} A_{11} & A_{12} & A_{13} \\ 0 & A_{22} & A_{23} \\ 0 & 0 & 0 \end{pmatrix} \end{array}, \quad B_2 = \begin{array}{c} \\ r_1 \\ r_2 \\ n-r \end{array} \!\!\begin{array}{c} \begin{array}{ccc} r_1 & r_2 & n-r \end{array} \\ \begin{pmatrix} 0 & 0 & 0 \\ 0 & B_{22} & B_{23} \\ 0 & 0 & B_{33} \end{pmatrix} \end{array}, \tag{3.3}$$

where A_{11}, A_{22}, B_{22} and B_{33} are upper triangular matrices and nonsingular, $r_1 + r_2 = r$. Hence there is a unique $(n - r_1) \times (n - r_1)$ upper triangular matrix T such that

$$\begin{pmatrix} A_{22} & A_{23} \\ 0 & 0 \end{pmatrix} = T \begin{pmatrix} B_{22} & B_{23} \\ 0 & B_{33} \end{pmatrix}.$$

This implies that the rest of computation is essentially equivalent to computing the SVD of the implicitly defined matrix T. By the global convergence theory of the cyclic Kogbetliantz algorithm (see §2.1), we have

$$T \rightarrow \Sigma, \tag{3.4}$$

where Σ is a diagonal matrix, and the convergence is ultimately quadratic, provided the rotation angles of U and V obey (2.1). (3.4) implies that there exists diagonal matrices Σ_1 and Σ_2 with $\Sigma_1^2 + \Sigma_2^2 = I$, and an upper triangular matrix R, such that

$$A_2 \rightarrow \Sigma_1 R, \quad \text{and} \quad B_2 \rightarrow \Sigma_2 R,$$

which gives the desired GSVD of A and B.

3.2 The 2×2 GSVD in Floating Point Arithmetic

In this section, we will use the usual model of floating point arithmetic: barring over/underflow, $\text{fl}(x \circ y) = (1+\delta)(x \circ y)$ where $\circ$ is one of the basic operations $\{+, -, \times, \div\}$ and $|\delta| \le \epsilon$ where ϵ is the machine roundoff. This model eliminates machines like Crays without guard digits, but with some effort all the results can be extended to these machines as well.

When using floating point arithmetic, roundoff can cause the row vectors of $\tilde{A}$ and $\tilde{B}$ computed by Algorithm 1 not to be parallel. This means $\tilde{A}$ and $\tilde{B}$ are not the GSVD of the 2×2 matrices A and B, or in short, the algorithm is not convergent. Another possibility is that the computation may not be backward stable, because the entries $\tilde{a}_{12}$ or $\tilde{b}_{12}$ ($\tilde{a}_{21}$ or $\tilde{b}_{21}$) which are explicitly set to zero by Algorithm 1 may be much larger than $O(\epsilon)\|A\|$ and $O(\epsilon)\|B\|$, respectively.[6] Thus, the algorithms in [28, 20, 4], which use the SVD of 2×2 triangular matrix to guarantee convergence, are potentially numerical unstable. On the other hand, to guarantee numerical stability, it is suggested in [18, 6] that after computing the SVD of the 2×2 triangular matrix C, one uses U (say) to form $G = U^T A$, then determines Q such that GQ is lower triangular, and finally determines $\tilde{V}$ such that $\tilde{V}^T BQ$ is also lower triangular. However, in practice, $U^T C \tilde{V}$ might not be diagonal, which results in divergence. In §5, we will present numerical examples illustrating the failures of these schemes. In this section, we propose a new algorithm to overcome these shortcomings. We first discuss the two fundamental algorithmic building blocks: SLASV2 and SLARTG.

SLASV2 computes the SVD of a 2×2 upper triangular matrix

$$\begin{pmatrix} c_u & s_u \\ -s_u & c_u \end{pmatrix} \begin{pmatrix} f & g \\ 0 & h \end{pmatrix} \begin{pmatrix} c_v & -s_v \\ s_v & c_v \end{pmatrix} = \begin{pmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{pmatrix}.$$

Barring over/underflow, SLASV2 computes all of c_u, s_u, c_v, s_v, σ_1 and σ_2 to nearly full machine precision. This algorithm was described briefly in [10], but not published in its entirety. For completeness, we include a listing of Fortran code in the appendix, and a statement and proof sketch of its error analysis. As discussed in [10], the high accuracy of SLASV2 is based on the fact that the algorithm uses formulas that only contain products, quotients, square roots, sums of terms of like sign, differences of computed quantities only when cancellation is impossible, and the difference $|f| - |h|$ of the input data, which, if cancellation occurs, is exact[7].

SLARTG(f, g, c, s, r) generates a rotation matrix $\text{rot}(c, s)$ from f and g to zero g, i.e., $c = f/r$ and $s = g/r$, $r = \sqrt{f^2 + g^2}$, but this is subject to spurious over/underflow if we directly compute them from these formulas. A more robust way to compute c, s and r can be found in [17]:

---

[6] Throughout, $\|\cdot\|$ will denote the matrix 2-norm.

[7] This exact cancellation property, which is essential for the accuracy claim of SLASV2, requires a guard digit and so fails on machines like the Cray. On a Cray we retain backward stability of SLASV2, but lose forward stability. Since the proof uses forward stability of SLASV2 in an important way, it does not apply to Cray. However, there is a more complicated proof which does work on the Cray. The reader is invited to try to find it.

(handle $f = 0$ and $g = 0$ as special cases)
if $|f| > |g|$ then
  $t = g/f$; $tt = \sqrt{1+t^2}$; $c = 1/tt$; $s = t * c$; $r = f * tt$
else
  $t = f/g$; $tt = \sqrt{1+t^2}$; $s = 1/tt$; $c = t * s$; $r = g * tt$;
endif

The same techniques used to analyze SLASV2 in the appendix can be straightforwardly used to show that the relative error in the computed c and s is bounded by 6ϵ.

Using SLARTG and SLASV2, we present a high-level description of an algorithm for computing the 2×2 GSVD. Later we will show that the proposed algorithm guarantees numerical stability and convergence. We will use the notation $|X| = (|x_{ij}|)$.

Algorithm GSVD22: Let A and B be 2×2 upper triangular matrices. The following algorithm computes the orthogonal matrices $U = \mathrm{rot}(c_u, s_u)$, $V = \mathrm{rot}(c_v, s_v)$ and $Q = \mathrm{rot}(c_q, s_q)$, such that

$$\tilde{A} = U^T A Q = \begin{pmatrix} \tilde{a}_{11} & 0 \\ \tilde{a}_{21} & \tilde{a}_{22} \end{pmatrix}, \quad \tilde{B} = V^T B Q = \begin{pmatrix} \tilde{b}_{11} & 0 \\ \tilde{b}_{21} & \tilde{b}_{22} \end{pmatrix}$$

are the GSVD of A and B. For brevity, we omit the part for lower triangular matrices, which can be described similarly.

compute $C = A \cdot \mathrm{adj}(B)$;
use SLASV2 to compute the SVD of C: $U^T C V = \Sigma$;
compute $G = U^T A$; $H = V^T B$;
compute $\hat{G} = |U|^T |A|$; $\hat{H} = |V|^T |B|$;
/* *The angles of U and V are chosen to satisfy the convergence condition (3.4).* */
if $|c_u| \geq |s_u|$ *or* $|c_v| \geq |s_v|$ *then*
  /* *Choose Q to zero out (1,2) entries of* $U^T A$ *and* $V^T B$ */
  if $\hat{g}_{12}/(|g_{11}| + |g_{12}|) \leq \hat{h}_{12}/(|h_{11}| + |h_{12}|)$ *then*
    call $SLARTG(-g_{11}, g_{12}, c_q, s_q, r)$ /* *Compute Q from* $U^T A$ */
  else
    call $SLARTG(-h_{11}, h_{12}, c_q, s_q, r)$ /* *Compute Q from* $V^T B$ */
  end if
  $\tilde{A} = GQ$; $\tilde{B} = HQ$; $\tilde{a}_{12} = 0$; $\tilde{b}_{12} = 0$.
else
  /* *Choose Q to zero out (2,2) entries of* $U^T A$ *and* $V^T B$ *and then swap rows.* */
  if $\hat{g}_{22}/(|g_{21}| + |g_{22}|) \leq \hat{h}_{22}/(|h_{21}| + |h_{22}|)$ *then*
    call $SLARTG(-g_{21}, g_{22}, c_q, s_q, r)$ /* *Compute Q from* $U^T A$ */
  else
    call $SLARTG(-h_{21}, h_{22}, c_q, s_q, r)$ /* *Compute Q from* $V^T B$ */
  end if
  $\tilde{A} = GQ$; $\tilde{B} = HQ$; $\tilde{a}_{22} = 0$; $\tilde{b}_{22} = 0$.
  /* *Swap, where* $P = \mathrm{rot}(0, 1)$ */
  $\tilde{A} \leftarrow P\tilde{A}$; $\tilde{B} \leftarrow P\tilde{B}$;
  $U \leftarrow UP$; $V \leftarrow VP$;
end if

We now present a theorem about the stability and convergence of the above algorithm. Quantities with bars (like $\bar{C}$) denote computed quantities.

Theorem 3.1 *The $\tilde{A}$ and $\tilde{B}$ computed by Algorithm GSVD22 have the following properties.*

(a) Both are triangular;

(b) $\bar{U}^T\bar{C}\bar{V}$ is within $132\epsilon\|\bar{C}\|$ of being diagonal.

(c) The rows of $\bar{\tilde{A}}$ and $\bar{\tilde{B}}$ are within $87\epsilon\|A\|$ and $87\epsilon\|B\|$, respectively, of being parallel.

(d) They are computed stably, i.e., there exist δA and δB, where $\|\delta A\| \le 377\epsilon\|A\|$ and $\|\delta B\| = 377\epsilon\|B\|$, and orthogonal U, V and Q such that

$$\bar{\tilde{A}} = U^T(A+\delta A)Q, \quad \bar{\tilde{B}} = V^T(B+\delta B)Q,$$

Proof. We only prove a branch of the algorithm where Q is computed from U^TA and used to zero out the (1,2) entries of U^TA and V^TB; the proof for the other cases is similar. We will also leave some of the more tedious details of error analysis to the ambitious reader.

We first note the following facts about the algorithm:

Fact 1. $\bar{C} = (A+\delta A_1)\cdot \mathrm{adj}(B+\delta B_1)$ where δA_1 and δB_1 are small componentwise relative perturbations of A and B:

$$\bar{C} = \begin{bmatrix} a_{11}b_{22}(1+\epsilon_1) & -a_{11}b_{12}(1+2\epsilon_2)+a_{12}b_{11}(1+2\epsilon_3) \\ 0 & a_{22}b_{11}(1+\epsilon_4) \end{bmatrix} = \begin{bmatrix} a'_{11}b_{22} & -a'_{11}b'_{12}+a'_{12}b'_{11} \\ 0 & a_{22}b'_{11} \end{bmatrix}$$

where $a'_{11} = a_{11}(1+\epsilon_1)$, $b'_{11} = b_{11}(1+\epsilon_4)$, $a'_{12} = a_{12}(1+2\epsilon_3)/(1+\epsilon_4)$ and $b'_{12} = b_{12}(1+2\epsilon_2)/(1+\epsilon_1)$. (The ϵ_i are independent quantities bounded in magnitude by the machine precision ϵ.) So there is at most a 3 ulp perturbation in any entry, and also $\|\delta A_1\| \le 4\epsilon\|A\|$ and $\|\delta B_1\| \le 4\epsilon\|B\|$.

Fact 2. The computed $\bar{U}$ and $\bar{V}$ from SLASV2 satisfy $\bar{U} = U+\delta U$, $\bar{V} = V+\delta V$, where $U^T\bar{C}V$ is the exact SVD of $\bar{C}$ and δU (δV) is a small componentwise relative perturbation of U (V), bounded by 46.5ε in each component (see the proposition in the appendix). This also implies $\|\delta U\| \le \sqrt{2}\cdot 46.5\epsilon < 66\epsilon$ and $\|\delta V\| < 66\epsilon$.

Fact 3. The error in the $\bar{g}_{ij}$ ($\bar{h}_{ij}$) is bounded by $48.5\epsilon\bar{\tilde{g}}_{ij}$ ($48.5\epsilon\bar{\tilde{h}}_{ij}$). In the factor 48.5, 2 comes from the roundoff in computing $\mathrm{fl}(U^TA)$ or $\mathrm{fl}(V^TB)$, and 46.5 comes from the errors in U and V.

Fact 4. Using simple geometry, one can show that changing f to $f+\delta f$ and g to $g+\delta g$ can change $c = f/\sqrt{f^2+g^2}$ and $s = g/\sqrt{f^2+g^2}$ to $c+\delta c$ and $s+\delta s$, respectively, where $\sqrt{\delta c^2+\delta s^2} \le 2((\delta f^2+\delta g^2)/(f^2+g^2))^{1/2}$.

Fact 5. Subroutine SLARTG computes $c = f/\sqrt{f^2+g^2}$ and $s = g/\sqrt{f^2+g^2}$ with relative errors bounded by 6ϵ. This means the 2×2 matrix $\mathrm{rot}(c,s)$ has an error bounded in norm by $\sqrt{2}\cdot 6\epsilon < 9\epsilon$.

Fact 6. If X and Y are 2-by-2 matrices, then $\|\mathrm{fl}(X\cdot Y) - X\cdot Y\| \le 4\cdot\epsilon\cdot\|X\|\cdot\|Y\|$.

We note that triangularity (a) holds by construction. We prove (b) as follows. Near diagonality of $\bar{U}^T\bar{C}\bar{V}$ holds by the high accuracy of $\bar{U}$ and $\bar{V}$:

$$\bar{U}^T\bar{C}\bar{V} = (U+\delta U)^T\bar{C}(V+\delta V) = U^T\bar{C}V + F_1$$

where Fact 2 tells us that to first order in ϵ

$$\|F_1\| \le \|\delta U^T\bar{C}V\| + \|U^T\bar{C}\delta V^T\| \le 66\epsilon\|\bar{C}\| + 66\epsilon\|\bar{C}\| = 132\epsilon\|\bar{C}\|.$$

Next we prove assertion (c). The top rows of $\bar{\tilde{A}}$ and $\bar{\tilde{B}}$ are trivially parallel by construction (their second components are zero), so we only consider the bottom rows. We know by construction that the bottom rows of $U^T(A+\delta A_1)$ and $V^T(B+\delta B_1)$ are parallel. Thus the bottom rows of

$$\bar{G} = \mathrm{fl}(\bar{U}^T A) = \bar{U}^T A + F_2 = (U^T + \delta U^T)(A + \delta A_1 - \delta A_1) + F_2 = U^T(A+\delta A_1) + F_3$$

and $\bar{H} = V^T(B+\delta B_1) + F_4$ are within $\|F_3\| \le 74\epsilon\|A\|$ and $\|F_4\| \le 74\epsilon\|B\|$, respectively, of being parallel. Here we have used Facts 1, 2 and 6.

So for *any* $\bar{Q} = Q + \delta Q$ that is within 9ϵ in norm of an orthogonal matrix Q, the bottom rows of $\bar{\tilde{A}}$ and $\bar{\tilde{B}}$ are the same as the bottom rows of

$$\mathrm{fl}(\bar{G}\bar{Q}) = \bar{G}\bar{Q} + F_5 = (U^T(A+\delta A_1) + F_3)(Q+\delta Q) + F_5 = U^T(A+\delta A_1)Q + F_6$$

and $\mathrm{fl}(\bar{H}\bar{Q}) = V^T(B+\delta B_1)Q + F_7$, which are within $\|F_6\| \le 87\epsilon\|A\|$ and $\|F_7\| \le 87\epsilon\|B\|$ of being parallel; we have used our bounds on $\|F_3\|$ and $\|F_4\|$, and Facts 5 and 6. This proves assertion (c).

Let $\eta_a = \bar{\tilde{g}}_{12}\epsilon/(|\bar{g}_{11}| + |\bar{g}_{12}|)$, and $\eta_b = \bar{\hat{h}}_{12}\epsilon/(|\bar{h}_{11}| + |\bar{h}_{12}|)$. Then η_a (η_b) is an approximate bound on relative error of Q if it is computed from $U^T A$ ($V^T B$). In the branch of the algorithm we consider, $\eta_a \le \eta_b$, and the algorithm chooses to compute Q from $U^T A$. The remarkable fact is that even if $\epsilon \ll \eta_a$, so that the forward error in Q is large, the backward error in B is small.

To finally prove this assertion (d), we need to show the (1,2) entry of $\mathrm{fl}(\bar{H}\bar{Q})$, which is zeroed out to get $\tilde{B}$, is at most $286\epsilon\|B\|$. (Q is chosen to accurately zero out the (1,2) entry of $\mathrm{fl}(\bar{G}\bar{Q})$.) Earlier we showed that $\bar{h}_{11} = h_{11} + 74\epsilon_8\|B\|$ and $\bar{h}_{12} = h_{12} + 74\epsilon_9\|B\|$, where h_{11} and h_{12} are the exact entries of $V^T(B+\delta B_1)$. Now write $\bar{c}_q = c_q + \delta c_q$ and $\bar{s}_q = s_q + \delta s_q$, where c_q and s_q are the exact cosine and sine computed from $U^T(A+\delta A_1)$. Then

$$\begin{aligned} |\mathrm{fl}((\bar{H}\bar{Q})_{12})| &= |(h_{11} + 76\epsilon_{10}\|B\|)(s_q + \delta s_q) + (h_{12} + 76\epsilon_{11}\|B\|)(c_q + \delta c_q)| \\ &= |(h_{11}s_q + h_{12}c_q) + (h_{11}\delta s_q + h_{12}\delta c_q) + \sqrt{2}\cdot 76\epsilon_{12}\|B\|| \\ &\le |h_{11}|\ |\delta s_q| + |h_{12}|\ |\delta c_q| + 108\epsilon\|B\| \end{aligned} \qquad (3.5)$$

There are two cases, $\eta_a < \epsilon$ and $\eta_a \ge \epsilon$. In the first case, we will show δs_q and δc_q are both bounded by 175ϵ, and so $|\mathrm{fl}((\bar{H}\bar{Q})_{12})| \le 286\epsilon$. To see this, use Fact 3 to write

$$(48.5\epsilon\bar{g}_{11})^2 + (48.5\epsilon\bar{\tilde{g}}_{12})^2 = (48.5\epsilon\bar{g}_{11})^2 + (48.5\epsilon\bar{\tilde{g}}_{12})^2 \le (48.5\epsilon)^2(\bar{g}_{11}^2 + (|\bar{g}_{11}| + |\bar{g}_{12}|)^2)$$

so by Facts 4 and 5, $\sqrt{|\delta s_q|^2 + |\delta c_q|^2}$ can be at most

$$9\epsilon + 2\cdot 48.5\epsilon \left(\frac{\bar{g}_{11}^2 + (|\bar{g}_{11}| + |\bar{\tilde{g}}_{12}|^2)}{\bar{g}_{11}^2 + \bar{g}_{12}^2}\right)^{1/2} \le 178\epsilon\ .$$

Now we use this bound in inequality (3.5) to get $|\mathrm{fl}((\bar{H}\bar{Q})_{12})| \le 286\epsilon$ as desired.

In the second case, we bound $\sqrt{|\delta s_q|^2 + |\delta c_q|^2}$ by

$$9\epsilon + 2\left(\frac{48.5^2((\epsilon\bar{g}_{11})^2 + (\eta_a(|\bar{g}_{11}| + |\bar{g}_{12}|))^2)}{\bar{g}_{11}^2 + \bar{g}_{12}^2}\right)^{1/2} \le 178\eta_a\ .$$

Plugging in to inequality (3.5) and using

$$|\bar{h}_{11}| + |\bar{h}_{12}| = \frac{\epsilon}{\eta_b}\bar{\hat{h}}_{12} \le \frac{\epsilon\|B\|}{\eta_b}$$

yields the upper bound

$$|\mathrm{fl}((\bar{H}\bar{Q})_{12})| \leq (|h_{11}| + |h_{12}|) \cdot 178\eta_a + 108\epsilon\|B\| \leq \frac{\epsilon\|B\|}{\eta_b} \cdot 178\eta_a + 108\epsilon\|B\| = 286\epsilon\|B\|$$

as before, since $\eta_b \geq \eta_a$.

This means that we can write the final output

$$\bar{\bar{B}} = V^T(B + \delta B_1)Q + F_7 + F_8 = V^T(B + \delta B_1 + VF_7Q^T + VF_8Q^T)Q \equiv V^T(B + \delta B)Q$$

where F_8 zeroes out the (1,2) entry of $\bar{\bar{B}}$ and leaves the others unchanged. We just showed $\|F_8\| \leq 286\epsilon\|B\|$ and combing this with our earlier bounds of $\|F_7\| \leq 87\epsilon\|B\|$ and $\|\delta B_1\| \leq 4\epsilon\|B\|$ yields the final result $\|\delta B\| \leq 377\epsilon\|B\|$. We can similarly show that $\bar{\bar{A}} = U^T(A+\delta A)Q$ with $\|\delta A\| \leq 107\epsilon\|A\|$, using the fact that Q is computed to directly zero out the (1,2) entry of $\bar{\bar{A}}$. This complete the proof of assertion (d). ∎

The constants in these error bounds could doubtless be decreased by a more detailed analysis.

4 Summary of the Complete Algorithm

In this section, we present a high-level description of our version of Paige's algorithm for computing the GSVD of two upper triangular matrices A and B of the forms (1.5). Let κ be a user chosen parameter specifying the maximum number of cycles the algorithm may perform (say, $\kappa = 20$). Let P_{ij} be the identity matrix with rows i and j interchanged.

Algorithm GSVD

```
/* Initialization */
cycle := 0;
ξ := r + q + 1; /* r and q are defined in (1.5) */
U := I; V := I; Q := I if desired;
/* Main loop */
if nonconvergence and cycle ≤ κ do
   cycle := cycle + 1;
   do (i, j)-loop
      /* 2 × 2 GSVD */
      Use GSVD22 to find U_ij, V_ij, Q_ij from a_ii, a_ij, a_jj and b_ii, b_ij, b_jj;
      /* Updating */
      A := U_ij^T A Q_ij;
      B := V_ij^T B Q_ij;
      U := U U_ij; V := V V_ij; Q := Q Q_ij if desired;
      /* reordering */
      if the (j, j) entry of B is nonzero, where j > l, then
         A := A P_ξj;
         B := P_ξj B P_ξj;
         V := V V_ξj;   Q := Q P_ξj if desired;
         ξ := ξ + 1;
      end if
   end of (i, j)-loop
   convergence test if cycle is even.
end if
compute α_i and β_i.
```

The (i, j)-loop can be simply chosen as the standard cyclic pivot sequence. It is natural to use the parallelism (linear dependency) of the corresponding row vectors of A and B at the end of an even cycle as the stopping criterion of the iteration. To measure the parallelism of two k-vectors a and b to high accuracy and despite possible over/underflow, we propose the following scheme: first compute the QR factorization of the $k \times 2$ matrix $\left(\frac{a}{\|a\|}, \frac{b}{\|b\|}\right)$:

$$Q^T \left(\frac{a}{\|a\|}, \frac{b}{\|b\|}\right) = \begin{pmatrix} \mu_{11} & \mu_{12} \\ 0 & \mu_{22} \\ 0 & 0 \end{pmatrix},$$

and then compute the singular values $\gamma_1 \geq \gamma_2 \geq 0$ of the 2×2 upper triangular (μ_{ij}). It is clear that

$$\mathrm{par}(\frac{a}{\|a\|}, \frac{b}{\|b\|}) \equiv \gamma_2$$

measures the parallelism of these two vectors. Vectors a and b are exactly parallel iff $\gamma_2 = 0$.

Using the above described scheme as the stopping criterion in Algorithm GSVD, let a_i and b_i be the i-th row vectors of A and B, respectively, at the end of an even cycle. For a given tolerance value τ, we take

$$\mathrm{error} = \sum_{i=1}^{n} \mathrm{par}(\frac{a_i}{\|a_i\|}, \frac{b_i}{\|b_i\|}) \leq n\tau.$$

This means that there are perturbations of size at most $n\tau\|a_i\|$ in row a_i and $n\tau\|b_i\|$ in row b_i that makes them exactly parallel. This means that after making these perturbations, there would exist scalars α_i and β_i such that

$$\beta_i a_i = \alpha_i b_i, \quad i = 1, \ldots, n, \tag{4.1}$$

where α_i and β_i can be chosen so that $\alpha_i^2 + \beta_i^2 = 1$. From (4.1), it is seen that there is an upper triangular matrix R, such that

$$U^T AQ = \mathrm{diag}(\alpha_i)R, \quad V^T BQ = \mathrm{diag}(\beta_i)R,$$

which is the desired GSVD of matrices A and B, where α_i and β_i are the GSV pairs.

5 Numerical Experiments

The numerical experiments we discuss here first compare Algorithm GSVD22 with previous algorithms developed by Paige [28], Heath et at [20], Bai [4], Hammarling [18] and Bojanczyk *et al* [6]. Then we will evaluate Algorithm GSVD for different cases of random matrices A and B, measuring the backward stability, accuracy, average total number of sweeps, rate of convergence, elapsed time when computing GSV pairs only, or both GSV pairs and transformation matrices.

All tests were performed using FORTRAN 77 on a SUN sparc station 1+. The arithmetic was IEEE standard double precision [1], with a machine precision of $\epsilon = 2^{-53} \approx 10^{-16}$ and over/underflow threshold $10^{\pm 307}$. We use $\tau = 10^{-14}$ as the stopping criterion.

5.1 Backward Stability and Accuracy

Before we proceed, it is appropriate to state what we mean by the *backward stability* and the *accuracy* of Algorithm GSVD. The backward stability is defined as follows: Let the computed orthogonal matrices be $\bar{U}$, $\bar{V}$ and $\bar{Q}$, the diagonal matrices be $\bar{\Sigma}_1$ and $\bar{\Sigma}_2$, and the upper triangular matrix be $\bar{R}$. Then the following conditions should be satisfied:

$$\|\bar{U}^T\bar{U} - I\|_F \approx \epsilon; \quad \|\bar{V}^T\bar{V} - I\|_F \approx \epsilon; \quad \|\bar{Q}^T\bar{Q} - I\|_F \approx \epsilon; \tag{5.1}$$

$$\|\bar{U}^T A\bar{Q} - \bar{\Sigma}_1\bar{R}\|_{\rm F} \approx n\epsilon\|A\|_{\rm F}; \quad \|\bar{V}^T B\bar{Q} - \bar{\Sigma}_2\bar{R}\|_{\rm F} \approx n\epsilon\|B\|_{\rm F}, \tag{5.2}$$

where $\|\cdot\|_{\rm F}$ is Frobenius norm. These assertions say that to within roundoff error, the computed matrices $\bar{U}$, $\bar{V}$ and $\bar{Q}$ are orthogonal, and the rows of $\bar{U}^T A\bar{Q}$ and $\bar{V}^T B\bar{Q}$ are parallel.

The accuracy test of computed GSV pairs by Algorithm GSVD is based on Sun and Paige's perturbation bound of the GSV pairs [32, 26], which says that: if $\mathrm{rank}(G) = \mathrm{rank}(\tilde{G}) = n$, where $G = (A^T, B^T)^T$, and $\tilde{G} = (\tilde{A}^T, \tilde{B}^T)^T = ((A+E)^T, (B+F)^T)^T$, and the GSV pairs (α_i, β_i) of A and B, and $(\tilde{\alpha}_i, \tilde{\beta}_i)$ of $\tilde{A}$ and $\tilde{B}$ are ordered as in (1.4), respectively, then we have

$$\sqrt{\sum_{i=1}^{n}[(\alpha_i - \tilde{\alpha}_i)^2 + (\beta_i - \tilde{\beta}_i)^2]} \le \sqrt{2}\min\{\|G^\dagger\|_2, \|\tilde{G}^\dagger\|_2\} \left\| \begin{pmatrix} E \\ F \end{pmatrix} \right\|_{\rm F}. \tag{5.3}$$

If we generate the matrices A and B with known GSV pairs, then the above perturbation bound can be used to measure the accuracy of the computed GSV pairs.

5.2 The numerical comparison of different 2×2 GSVD algorithms

Several versions have been proposed for computing the 2×2 GSVD. There are essentially two kinds of schemes:

Scheme I: First compute the SVD of $C = A \cdot \mathrm{adj}(B)$: $U^T C V = \Sigma$, then form the product of $G = U^T A$ and $H = V^T B$, and finally compute Q from G such that the (1,2) or (2,1) entry of GQ is zero. Mathematically, it is known that the (1,2) or (2,1) entry of HQ is automatically zero. The algorithms proposed by Paige [28], Heath *et al* [20], and Bai [4] fall in this category.

Scheme II: First compute the SVD of $C = A \cdot \mathrm{adj}(B)$: $U^T C V = \Sigma$, form the product of $G = U^T A$, compute Q so such the (1,2) or (2,1) entry of GQ is zero, and finally compute V to zero out the (1,2) or (2,1) entry of BQ. The algorithms proposed by Hammarling [18] and Bojanczyk *et al* [6] fall in this category.

To demonstrate the failure of the first kind of scheme, the following example shows that in finite precision, the (1,2) or (2,1) entry of the final B may be much larger than $O(\epsilon)\|B\|$:

$$A = \begin{pmatrix} 2 & 0 \\ 1 & 10^{-8} \end{pmatrix}, \quad B = \begin{pmatrix} 1 & 0 \\ 3 & 1 \end{pmatrix}.$$

With the scheme described by Paige [28], Heath *et al* [20] and Bai [4], for the computed $\bar{U}$, $\bar{V}$ and $\bar{Q}$, we have

$$\begin{aligned} \bar{U}^T A\bar{Q} &= \begin{pmatrix} \bar{a}_1 \\ \bar{a}_2 \end{pmatrix} = \begin{pmatrix} 0.70710677509009934\mathrm{E}+00 & 0.21213203455917919\mathrm{E}+01 \\ 0.00000000000000000\mathrm{E}+00 & 0.28284271491319831\mathrm{E}-07 \end{pmatrix}, \\ \bar{V}^T B\bar{Q} &= \begin{pmatrix} \bar{b}_1 \\ \bar{b}_2 \end{pmatrix} = \begin{pmatrix} 0.31622776518779000\mathrm{E}+00 & 0.94868330465141837\mathrm{E}+00 \\ -0.33959487444334968\mathrm{E}-08 & 0.31622776582710133\mathrm{E}+01 \end{pmatrix}. \end{aligned}$$

If we now set the (2,1) entry of $\bar{B} = \bar{V}^T B\bar{Q}$ to zero, the backward stability condition (5.2) is violated for matrix B, even though

$$\bar{U}^T C\bar{V} = \begin{pmatrix} 0.22360679640833827\mathrm{E}+01 & 0.00000000000000000\mathrm{E}+00 \\ 0.17888543605335784\mathrm{E}-16 & 0.89442719636647925\mathrm{E}-08 \end{pmatrix}.$$

To show how Scheme II can fail for the same example, using Hammarling's method [18], we have

$$\begin{aligned} \bar{U}^T A\bar{Q} &= \begin{pmatrix} \bar{a}_1 \\ \bar{a}_2 \end{pmatrix} = \begin{pmatrix} 0.70710677509009934\mathrm{E}+00 & 0.21213203455917919\mathrm{E}+01 \\ 0.00000000000000000\mathrm{E}+00 & 0.28284271491319831\mathrm{E}-07 \end{pmatrix}, \\ \bar{V}^T B\bar{Q} &= \begin{pmatrix} \bar{b}_1 \\ \bar{b}_2 \end{pmatrix} = \begin{pmatrix} -0.31622776518778994\mathrm{E}+00 & -0.94868327069193081\mathrm{E}+00 \\ 0.11102230246251565\mathrm{E}-15 & -0.31622776684588585\mathrm{E}+01 \end{pmatrix}. \end{aligned}$$

Thus the stability is achieved, but for the computed $\bar{U}$ and $\bar{V}$, we have

$$\bar{U}^T C \bar{V} = \begin{pmatrix} -0.22360679640833823\mathrm{E}+01 & -0.24012983681642603\mathrm{E}-07 \\ 0.78163392273857838\mathrm{E}-16 & -0.89442719636647908\mathrm{E}-08 \end{pmatrix}$$

which is not within $O(\epsilon)\|C\|$ of diagonal form. This means that the computed $\bar{A} = \bar{U}^T A \bar{Q}$ and $\bar{B} = \bar{V}^T B \bar{Q}$ are not the GSVD of A and B. In fact, $\mathrm{par}(\frac{\bar{a}_1}{\|A\|}, \frac{\bar{b}_1}{\|B\|}) \approx \mathrm{par}(\frac{\bar{a}_1}{\|\bar{a}_1\|}, \frac{\bar{b}_1}{\|\bar{b}_1\|}) \approx 7.59 \times 10^{-9}$.

But using Algorithm GSVD22 in §3.2, we have

$$\bar{U}^T A \bar{Q} = \begin{pmatrix} \bar{a}_1 \\ \bar{a}_2 \end{pmatrix} = \begin{pmatrix} 0.70710677736817096\mathrm{E}+00 & 0.21213203448324349\mathrm{E}+01 \\ 0.30374288814267665\mathrm{E}-16 & 0.28284271491319831\mathrm{E}-07 \end{pmatrix},$$

$$\bar{V}^T B \bar{Q} = \begin{pmatrix} \bar{b}_1 \\ \bar{b}_2 \end{pmatrix} = \begin{pmatrix} 0.31622776620657461\mathrm{E}+00 & 0.94868330431182346\mathrm{E}+00 \\ 0.00000000000000000\mathrm{E}+00 & 0.31622776582710133\mathrm{E}+01 \end{pmatrix},$$

and

$$\bar{U}^T C \bar{V} = \begin{pmatrix} 0.22360679640833827\mathrm{E}+01 & 0.00000000000000000\mathrm{E}+00 \\ 0.17888543605335784\mathrm{E}-16 & 0.89442719636647925\mathrm{E}-08 \end{pmatrix}.$$

Thus both stability and convergence conditions are satisfied, where $\mathrm{par}(\frac{\bar{a}_1}{\|A\|}, \frac{\bar{b}_1}{\|B\|}) \approx \mathrm{par}(\frac{\bar{a}_1}{\|\bar{a}_1\|}, \frac{\bar{b}_1}{\|\bar{b}_1\|}) \approx 7.02 \times 10^{-17}$.

Recently, Bojanczyk *et al* [6] proposed a variation of Scheme II, which we refer to as the BELV scheme. The BELV scheme was originally designed for treating a matrix-triple (A_1, A_2, A_3). It is easy to see that the 2×2 GSVD of two matrices is a special case when one of the matrices (say, A_3 is the identity). The BELV scheme does significantly improve Hammarling's method, but it still suffers from possible nonconvergence. For example, using the BELV scheme, we see that for the following 2×2 matrices

$$A = \begin{pmatrix} 100000 & 10000 \\ 0 & 0.0001 \end{pmatrix}; \quad B = \begin{pmatrix} 100000 & 10000.0000000001 \\ 0 & 0.003 \end{pmatrix},$$

the computed orthogonal matrices $\bar{U}$, $\bar{V}$ and $\bar{Q}$ by BELV scheme satisfy the stability conditions (5.1) and (5.2):

$$\bar{U}^T A \bar{Q} = \begin{pmatrix} \bar{a}_1 \\ \bar{a}_2 \end{pmatrix} = \begin{pmatrix} 0.99503719020998935\mathrm{E}-04 & -0.12189168086858831\mathrm{E}-03 \\ 0.00000000000000000\mathrm{E}+00 & 0.10049875621120891\mathrm{E}+06 \end{pmatrix}$$

$$\bar{V}^T B \bar{Q} = \begin{pmatrix} \bar{b}_1 \\ \bar{b}_2 \end{pmatrix} = \begin{pmatrix} 0.29851115706299699\mathrm{E}-02 & -0.36499576550546638\mathrm{E}-02 \\ 0.00000000000000000\mathrm{E}+00 & 0.10049875621120886\mathrm{E}+06 \end{pmatrix}$$

However, the computed $\bar{U}$ and $\bar{V}$ do not diagonalize the matrix C:

$$\bar{U}^T C \bar{V} = \begin{pmatrix} 0.99999999999999964\mathrm{E}+01 & -0.67590597725531451\mathrm{E}-09 \\ 0.20277179317658482\mathrm{E}-07 & 0.30000000000000023\mathrm{E}+03 \end{pmatrix},$$

since the off-diagonal elements are much larger[8] than $O(\epsilon)\|C\| \approx 10^{-14}$, $\mathrm{par}(\frac{\bar{a}_1}{\|\bar{a}_1\|}, \frac{\bar{b}_1}{\|\bar{b}_1\|}) = 6.44 \times 10^{-4}$, and even $\mathrm{par}(\frac{\bar{a}_1}{\|A\|}, \frac{\bar{b}_1}{\|B\|}) = 1.43 \times 10^{-12}$, so that the first rows of $\bar{A}$ and $\bar{B}$ are not parallel.

[8] In [6] it is proven that the off diagonal elements should be $O(\epsilon)\|A\|\,\|B\| \approx 10^{-6}$, which is attained. Since $\|C\| = \|A\,\mathrm{adj}(B)\| \ll \|A\|\,\|B\|$, our bound is much tighter.

But Algorithm GSVD22 yields

$$\bar{U}^T A \bar{Q} = \begin{pmatrix} \bar{a}_1 \\ \bar{a}_2 \end{pmatrix} = \begin{pmatrix} -0.10049875621120894\mathrm{E}+06 & 0.00000000000000000\mathrm{E}+00 \\ -0.12189168086858835\mathrm{E}-03 & -0.99503719020998963\mathrm{E}-04 \end{pmatrix}$$

$$\bar{V}^T B \bar{Q} = \begin{pmatrix} \bar{b}_1 \\ \bar{b}_2 \end{pmatrix} = \begin{pmatrix} -0.10049875621120886\mathrm{E}+06 & -0.18189894035458565\mathrm{E}-11 \\ -0.36567504260576521\mathrm{E}-02 & -0.29851115706299699\mathrm{E}-02 \end{pmatrix}$$

and

$$\bar{U}^T C \bar{V} = \begin{pmatrix} 0.30000000000000028\mathrm{E}+03 & 0.16940658945086007\mathrm{E}-20 \\ 0.00000000000000000\mathrm{E}+00 & 0.99999999999999982\mathrm{E}+01 \end{pmatrix}$$

with $\mathrm{par}(\frac{\bar{a}_1}{\|A\|}, \frac{\bar{b}_1}{\|B\|}) = \mathrm{par}(\frac{\bar{a}_1}{\|\bar{a}_1\|}, \frac{\bar{b}_1}{\|\bar{b}_1\|}) = 0$, and $\mathrm{par}(\frac{\bar{a}_2}{\|A\|}, \frac{\bar{b}_2}{\|B\|}) < \mathrm{par}(\frac{\bar{a}_2}{\|\bar{a}_2\|}, \frac{\bar{b}_2}{\|\bar{b}_2\|}) = 1.72 \times 10^{-16}$.

The above examples show that Algorithm GSVD22 is superior to all previous schemes.

5.3 Test matrix generation for testing backward stability

To test the backward stability of Algorithm GSVD, we used the LAPACK test matrix generation suite [11] to generate different types of upper triangular matrices A and B. The conditioning of a generated upper triangular matrix can be controlled by the following parameters:

`dist` specifies the type of probability distribution to be used to generate the random matrices:

= U: uniform distribution on (0, 1);

= S: uniform distribution on (-1, 1);

= N: normal distribution on (0, 1).

`cond` specifies the reciprocal of the condition number of generated matrix, $\texttt{cond} \geq 1$.

`mode` describes how the singular values d_i of generated matrix are to be distributed:

= 1: sets $d_1 = 1$ and $d_i = 1/\texttt{cond}$, $i = 2, \ldots, n$;

= 2: sets $d_i = 1$, $i = 1, \ldots, n-1$ and $d_n = 1/\texttt{cond}$;

= 3: sets $d_i = \texttt{cond}^{-(i-1)/(n-1)}$, $i = 1, \ldots, n$;

= 4: sets $d_i = 1 - \frac{i-1}{n-1}\left(1 - \frac{1}{\texttt{cond}}\right)$, $i = 1, \ldots, n$;

= 5: sets d_i to random in ($1/\texttt{cond}$, 1), their logarithms are uniformly distributed;

= 6: sets d_i to random numbers from same distribution as the rest of the matrix.

We generated 12 separate classes of upper triangular matrices A and B according to different choices of parameters `dist,` `cond` and `mode`, since this allows us to form different types of matrices to fairly test the behavior of the algorithm. The 12 classes are listed in Table 5.1. Thus classes 1 to 6 consist of well-conditioned matrices B, and the conditioning of matrix A is changed from well to ill-conditioned. Classes 7 to 10 consist of well-conditioned matrices A and the conditioning of matrix B is changed from moderate to ill-conditioned. Classes 11 and 12 consist of moderately conditioned matrices A and B.

5.4 Test Results

We tested the above 12 classes of matrix pairs of dimension of $n = 5, 10, 20, 50$. In each class of dimension 5 we generated 401 matrix pairs, in each class of dimension 10 we generated 301 matrix pairs, in each class of dimension 20 we generated 201 matrix pairs, and in each class of dimension 50 we generated 101 matrix pairs. This makes a total of 12,048 test matrix pairs.

	A			B		
class	`dist`	`cond`	`mode`	`dist`	`cond`	`mode`
1	U	10	6	U	10	6
2	U	10^2	2	S	10	6
3	U	10^5	1	N	10	5
4	S	10^8	3	S	10	6
5	S	10^{12}	4	U	10	5
6	S	10^{14}	4	N	10	6
7	N	10	6	N	10^5	1
8	N	10	6	U	10^8	2
9	N	10	6	S	10^{12}	2
10	S	10	6	N	10^{14}	4
11	S	10^5	4	N	10^5	4
12	S	10^3	3	N	10^4	4

Table 5.1: Test matrices

Class		1	2	3	4	5	6	7	8	9	10	11	12
n	5	2.29	2.40	2.02	2.00	2.19	2.18	2.07	2.05	2.00	2.12	2.01	1.93
	10	3.00	3.01	2.99	2.01	3.01	3.00	2.97	3.01	2.00	2.99	3.00	2.00
	20	3.26	3.50	3.07	2.19	3.53	3.30	3.05	3.21	2.98	3.23	3.21	2.20
	50	4.00	4.01	3.99	3.00	4.00	4.00	3.89	4.01	3.00	4.00	4.00	3.00

Table 5.2: Average Number of double sweeps

Table 5.2 illustrates the average number of double sweeps required to converge with the tolerance value $\tau = 10^{-14}$, where a double sweep consists of a sweep of row ordering and a sweep of column ordering. None of 12,048 test matrix pairs failed to converge. The observed largest number of double sweeps required to converge was 5. The backward stability conditions (5.1) and (5.2) held throughout the test. The following quadratic convergence rate of the algorithm is typical of what we observed:

cycle	2	4	6	8
error $= \sum_{i=1}^{n} \mathrm{par}(a_i, b_i)$	1.5094	$1.0252 \cdot 10^{-2}$	$9.4356 \cdot 10^{-9}$	$6.4874 \cdot 10^{-16}$

where A and B are 50×50 matrices, the condition numbers for both matrices are about 10^4.

5.5 Test matrix generation for testing accuracy

To test accuracy of Algorithm GSVD, we generated random matrices A and B with known GSV pairs. Specifically, let $\Sigma_1 = \mathrm{diag}(\alpha_i)$ and $\Sigma_2 = \mathrm{diag}(\beta_i)$ be the given GSV pairs. Then we generated random orthogonal matrices U, V and Q uniformly distributed with respect to Haar measure, and a random upper triangular matrix R with specified smallest singular value, and finally formed

$$A = U\Sigma_1 RQ^T \quad \text{and} \quad B = V\Sigma_2 RQ^T. \tag{5.4}$$

Hence the GSV pairs of A and B are known to be $(\alpha_i,\, \beta_i)$.

Type	α_i, β_i	$\sigma_{\min}(R)$	double sweeps for different n				Δ_1
			5	10	20	40	
1	U(0,1), U(0,1)	10	2.28	3.02	3.53	4.02	$1.51 \cdot 10^{-14}$
		10^{-6}	2.02	3.00	3.23	4.00	$1.21 \cdot 10^{-15}$
		10^{-12}	2.07	3.04	3.45	4.21	$8.64 \cdot 10^{-15}$
2	$1/i^2, 1$	10	2.01	2.62	3.00	3.00	$2.56 \cdot 10^{-15}$
		10^{-6}	2.00	2.61	3.00	3.00	$2.65 \cdot 10^{-15}$
		10^{-12}	2.00	2.61	3.00	3.10	$9.99 \cdot 10^{-15}$
3	$i, 1$	10	2.48	3.06	3.99	4.06	$2.52 \cdot 10^{-14}$
		10^{-6}	2.07	3.01	3.99	4.02	$9.71 \cdot 10^{-15}$
		10^{-12}	2.75	3.38	4.01	4.53	$3.89 \cdot 10^{-16}$
4	$1 + \mathrm{mod}(i, n/4+1), 1$	10	1.03	2.03	3.00	4.00	$7.33 \cdot 10^{-14}$
		10^{-6}	1.00	2.26	3.04	4.02	$5.11 \cdot 10^{-15}$
		10^{-12}	1.08	3.51	3.50	4.59	$1.95 \cdot 10^{-15}$
5	$1 - \frac{i-1}{n-1}(1 - \frac{1}{\mathtt{cond}}), 1$	10	2.06	3.00	3.55	4.14	$3.29 \cdot 10^{-14}$
		10^{-6}	2.01	3.01	3.62	4.18	$4.23 \cdot 10^{-15}$
		10^{-12}	2.01	3.01	3.62	4.20	$6.05 \cdot 10^{-15}$
6	$1, \mathtt{cond}^{-(i-1)/(n-1)}$	10	2.28	3.00	3.28	4.00	$1.51 \cdot 10^{-14}$
		10^{-6}	2.00	2.77	3.00	3.17	$2.98 \cdot 10^{-16}$
		10^{-12}	2.00	2.00	3.00	3.20	$1.14 \cdot 10^{-15}$

Table 5.3: Average double sweeps and accuracy of computed GSV pairs

In this way we can generate random test matrices having any distribution of the GSV pairs, and

$$\|G^\dagger\|_2^{-1} = \sigma_{\min}(G) = \sigma_{\min}(R).$$

Hence $\sigma_{\min}(R)$ (the smallest singular value) gives the conditioning of the designed test matrix pair. If $\bar{\alpha}_i$ and $\bar{\beta}_i$ are computed the GSV pairs by Algorithm GSVD, then the quantity

$$\Delta_1 \equiv \left\{ \sum_{i=1}^{n} [(\alpha_i - \bar{\alpha}_i)^2 + (\beta_i - \bar{\beta}_i)^2] \right\}^{1/2} \sigma_{\min}(R) \tag{5.5}$$

should be $O(\tau)$, where $\tau = 10^{-14}$ is our stopping criterion.

We designed six different distributions of the GSV pairs as illustrated in the second column of Table 5.3, where α_i and β_i are normalized so that $\alpha_i^2 + \beta_i^2 = 1$ for $i = 1, \ldots, n$ if necessary, (U(0,1),U(0,1)) means that GSV pairs (α_i, β_i) comes from the normalization of a pair of random numbers from a uniform distribution on the interval (0,1). `cond` is the reciprocal of the smallest singular value of the matrix R in (5.4). Note that some of the distributions of GSV are well separated, some of them are highly clustered or multiple.

5.6 Test Results

We generated several categories of matrix pairs according to three parameters: the dimension n, the smallest singular value of R ($\sigma_{\min}(R)$), and the type of distribution of GSV. We first separated test matrices with three possible values of $\sigma_{\min}(R) = 1, 10^{-6}, 10^{-12}$, i.e., corresponding to well,

Timing in seconds with $\tau = 10^{-14}$		
	without U, V, Q	with U, V, Q
preprocessing	0.28	1.11
iteration	13.11	20.99

Table 5.4: CPU timing of the GSVD of two 50×50 matrices

moderately, and ill-conditioned GSVD problems. For each $\sigma_{\min}(R)$, we tested matrices of dimension $n = 5, 10, 20, 40$ with six different distributions of GSV pairs as showed in table 1. This makes a total of $3 \times 4 \times 6 = 72$ different classes of matrices. In each class of dimension 5 we generated 301 matrices, in each class of dimension 10 we generated 201 matrices, in each class of dimension 20 we generated 101 matrices, and in each class of dimension 40 we generated 51 matrices, for a total of 10,772 test matrix pairs.

Table 5.3 illustrates the average number of double sweeps and accuracy of the algorithm for different size of matrices. The preprocessing orthogonal transformations of A and B to upper trapezoidal forms (1.5) are performed using LINPACK QR decomposition subroutine DQRDC [12]. In all tests, the backward stability conditions (5.1) and (5.2) are satisfied, so we do not report the details here. Given the backward stability, we can assume that the backward errors E of A and F of B satisfy $O(\|E\|,\ \|F\|) = O(10^{-14})$. For each type of GSV distribution, we let the conditioning (i.e., $\sigma_{\min}(R)$) of the GSVD problems vary from well to moderate to ill-conditioned, as indicated in column 3 of Table 5.3. The numbers in column 4 to 7 are the average numbers of double sweeps needed for convergence. The last column of the table is the largest value of Δ_1 computed from the formula (5.5). We see that all computed results are as accurate as predicted.

Finally, we briefly report timing results. The codes have not been polished intensively in order to reduce the execution time. Table 5.4 illustrates the required time for a 50 by 50 matrix pair A and B with 5 double sweeps to satisfy the stopping criterion.

Appendix: The SVD of 2×2 Triangular Matrix

In this appendix, for the convenience of the reader, we include Demmel and Kahan's 2×2 triangular SVD algorithm. The algorithm was used in their high relative accuracy bidiagonal SVD algorithm [10], but the algorithm details were not presented there.

It is known that the singular values of the 2×2 upper triangular matrix $\begin{pmatrix} f & g \\ 0 & h \end{pmatrix}$ are the values of the unobvious expression $\frac{1}{2}|\sqrt{(f+h)^2+g^2} \pm \sqrt{(f-h)^2+g^2}|$, of which the bigger is γ_1 and the smaller is $\gamma_2 = |fh|/\gamma_1$. The right singular vector row $(-s_v, c_v)$ turns out to be parallel to $(f^2 - \gamma_1^2, fg)$. After computing a right singular vector, the corresponding left singular vector is determined by $(c_u, s_u) = (fc_v + gs_v, hs_v)/\gamma_1$. But computing the singular values/vectors directly from these expressions is unwise because roundoff can destroy all relative accuracy, and they can suffer from over/underflow in the squared subexpressions even when the singular values/vectors are far from over/underflow thresholds. Demmel and Kahan have carefully reorganized the computation as described in the following so that barring over/underflow and assuming a guard digit in subtraction, all output quantities are correct to within a few units in the last place (ulps). In IEEE arithmetic [1], the code works correctly even if one matrix entry is infinite. Overflow is impossible unless the largest singular value itself overflows, or is within a few ulps of overflow. (On machines with partial overflow, like the Cray, overflow may occur if the largest singular value is within a factor of 2 of overflow.) Underflow is harmless if underflow is gradual. Otherwise, results may correspond to a matrix modified by perturbations of size near the underflow threshold.

The error analysis of the main path of the code depends on the fact that all the operations except two are

multiplication and division, where the relative error of the result is at most 1 ulp larger than the sum of the relative errors of the inputs,
addition of positive quantities, where the relative error of the result is at most 1 ulp larger than the maximum of the relative errors of the inputs, and
square root, where the relative error of the result is at most 1 ulp more than half the relative error of the input.

There are also two subtractions in the main code path. The first subtracts original data D = FA-HA, and so has a 1 ulp error. In the second, T=2-L with $0 \le$ L ≤ 1, the relative error in T can only be 1 ulp larger than the relative error in L. These rules are sufficient to straightforwardly bound the error in the main code path, provided we ignore second order terms. There is another path corresponding to the case where the offdiagonal g is much larger than the other two matrix entries, which is analyzed much more easily. Summarizing all these considerations we can easily prove the

Proposition. *Barring over/underflow, and assuming there is a guard digit in subtraction, the relative errors in the computed singular values are at most 7 ulps, and the relative errors in the computed singular vectors are at most 46.5 ulps in each component.*

The comments in the following code indicate the error bound in ulp of each computed quantity.

```
      SUBROUTINE SLASV2( F, G, H, SSMIN, SSMAX, SNR, CSR, SNL, CSL )
      REAL CSL, CSR, F, G, H, SNL, SNR, SSMAX, SSMIN
C
C     Computes singular value decomposition of 2 × 2 triangular matrix:
C     [ CSL SNL ] . [ F G ] . [ CSR -SNR ] = [ SSMAX 0 ]
C     [-SNL CSL ]   [ 0 H ]   [ SNR CSR ]    [ 0 SSMIN ]
C     Absolute value of SSMAX is larger singular value, Absolute value of SSMIN
C     is smaller singular value. Both CSR**2+SNR**2=1 and CSL**2+SNL**2=1.
C
C     .. Parameters ..
      REAL      ZERO, HALF, ONE, TWO, FOUR
      PARAMETER ( ZERO = 0.0, HALF = 0.5, ONE = 1.0, TWO = 2.0, FOUR = 4.0 )
C     .. Local Scalars ..
      LOGICAL  GASMAL, SWAP
      INTEGER  PMAX
      REAL      A, CLT, CRT, D, FA, FT, GA, GT, HA, HT, L, M,
      REAL      MM, R, S, SLT, SRT, T, TEMP, TSIGN, TT
C     .. Intrinsic Functions ..
      INTRINSICABS, SIGN, SQRT
C
      FT = F
      FA = ABS( FT )
      HT = H
      HA = ABS( H )
      PMAX = 1                              /* PMAX points to maximum absolute entry of matrix */
      SWAP = ( HA.GT.FA )
      IF( SWAP ) THEN
         PMAX = 3
         TEMP = FT
         FT = HT
         HT = TEMP
         TEMP = FA
         FA = HA
         HA = TEMP
      END IF                                /* Now FA .ge. HA */
      GT = G
      GA = ABS( GT )
      IF( GA.EQ.ZERO ) THEN                 /* Diagonal matrix */
         SSMIN = HA
         SSMAX = FA
         CLT = ONE
         CRT = ONE
         SLT = ZERO
         SRT = ZERO
      ELSE
         GASMAL = .TRUE.
         IF( GA.GT.FA ) THEN
            PMAX = 2
            IF( ABS( FA / GA ).LE.EPS ) THEN   /* Case of very large GA, EPS is machine epsilon */
               GASMAL = .FALSE.
               SSMAX = GA                   /* 1 ulp error */
               IF( HA.GT.ONE ) THEN
```

```
                SSMIN = FA / ( GA / HA )               /* 2 ulps error */
            ELSE
                SSMIN = ( FA / GA )*HA                 /* 2 ulps error */
            END IF
            CLT = ONE                                  /* 1 ulp error */
            SLT = HT / GT                              /* 1 ulp error */
            SRT = ONE                                  /* 1 ulp error */
            CRT = FT / GT                              /* 1 ulp error */
        END IF
    END IF
    IF( GASMAL ) THEN                                  /* Normal case */
        D = FA - HA                                    /* 1 ulp error */
        IF( D.EQ.FA ) THEN                             /* Copes with infinite F or H */
            L = ONE                                    /* 0 ulps error */
        ELSE
            L = D / FA                                 /* 2 ulps error */
        END IF                                         /* Note that 0 ≤ L ≤ 1 */
        M = GT / FT                                    /* 1 ulp error; Note that |M| ≤ 1/EPS */
        T = TWO - L                                    /* 3 ulps error; Note that T ≥ 1 */
        MM = M*M                                       /* 3 ulps error */
        TT = T*T                                       /* 7 ulps error */
        S = SQRT( TT+MM )                              /* 5 ulps error; Note that 1 ≤ S ≤ 1 + 1/EPS */
        IF( L.EQ.ZERO ) THEN
            R = ABS( M )                               /* 0 ulps error */
        ELSE
            R = SQRT( L*L+MM )                         /* 3.5 ulps error */
        END IF                                         /* Note that 0 ≤ R ≤ 1 + 1/EPS */
        A = HALF*( S+R )                               /* 6 ulps error; Note that 1 ≤ A ≤ 1 + |M| */
        SSMIN = HA / A                                 /* 7 ulps error */
        SSMAX = FA*A                                   /* 7 ulps error */
        IF( MM.EQ.ZERO ) THEN                          /* Note that M is very tiny */
            IF( L.EQ.ZERO ) THEN
                T = SIGN( TWO, FT )*SIGN( ONE, GT )/* 0 ulps error */
            ELSE
                T = GT / SIGN( D, FT ) + M / T         /* 6 ulps error */
            END IF
        ELSE
            T = ( M / ( S+T )+M / ( R+L ) )*( ONE+A )/* 17 ulps error */
        END IF
        L = SQRT( T*T+FOUR )                           /* 18.5 ulps error */
        CRT = TWO / L                                  /* 19.5 ulps error */
        SRT = T / L                                    /* 36.5 ulps error */
        CLT = ( CRT+SRT*M ) / A                        /* 46.5 ulps error */
        SLT = ( HT / FT )*SRT / A                      /* 45.5 ulps error */
    END IF
END IF
IF( SWAP ) THEN
    CSL = SRT
    SNL = CRT
    CSR = SLT
    SNR = CLT
ELSE
    CSL = CLT
    SNL = SLT
    CSR = CRT
    SNR = SRT
END IF
C   Correct the signs of SSMAX and SSMIN
IF( PMAX.EQ.1 ) TSIGN = SIGN( ONE, CSR )*SIGN( ONE, CSL )*SIGN( ONE, F )
IF( PMAX.EQ.2 ) TSIGN = SIGN( ONE, SNR )*SIGN( ONE, CSL )*SIGN( ONE, G )
IF( PMAX.EQ.3 ) TSIGN = SIGN( ONE, SNR )*SIGN( ONE, SNL )*SIGN( ONE, H )
SSMAX = SIGN( SSMAX, TSIGN )
SSMIN = SIGN( SSMIN, TSIGN*SIGN( ONE, F )*SIGN( ONE, H ) )
RETURN
END
```

Acknowledgement The authors are grateful to S. Hammarling and W. Kahan for valuable comments. We would also like to express thanks to the referees, whose comments led to improvement in the presentation.

References

[1] *IEEE Standard for Binary Floating Point Arithmetic*. ANSI/IEEE, New York, Std 754-1985 edition, 1985.

[2] Z. Bai, Note on the quadratic convergence of Kogbetliantz algorithm for computing the singular value decomposition, Lin. Alg. Appl., 104:131–140(1988).

[3] Z. Bai, Numerical treatment of restricted Gauss-Markov linear model, Comm. Statis. B17 2:131–140(1988).

[4] Z. Bai, The direct GSVD algorithm and its parallel implementation. Ph.D. thesis, Fudan University, China, 1987. Also available as Comput. Sci. TR-1901, Univ. of Maryland, College Park, 1987

[5] J. L. Barlow, Error analysis and implementation aspects of deferred correction for equality constrained least squares problems. SIAM J. Numer. Anal., 25:1340–1358(1988).

[6] A. W. Bojanczyk, M. Ewerbring, F. T. Luk and P. van Dooren, An accurate product SVD algorithm, in SVD and Signal Processing, II, Algorithms, Analysis and Applications, R. J. Vaccaro ed. Elsevier Sci. Publishers, Holland, 1991.

[7] J. P. Charlier and P. Van Dooren, On Kogbetliantz's SVD algorithm in the presence of Clusters, Lin. Alg. Appl. 95:136–160(1987).

[8] B. L. R. De Moor and G. H. Golub, Generalized Singular Value Decompositions: A proposal for a standardized nomenclature, Manuscript NA-89-05, Stanford Univ. , Stanford, CA, 1989

[9] B. L. R. De Moor and H. Zha, A tree of generalizations of the ordinary singular value decomposition, ESAT-SISTA report 1989-21, Katholieke Universiteit Leuven, Belgium.

[10] J. Demmel and W. Kahan, Accurate Singular Values of Bidiagonal Matrices, SIAM J. Sci. Stat. Comput. 11:873–912(1990).

[11] J. Demmel and A. Mckenney, LAPACK working notes # 9, a test matrix generation suite, Argonne National Lab. MCS-P69-0389, 1989.

[12] J. J. Dongarra, J. R. Bunch, C. B. Moler and G. W. Stewart, LINPACK User's Guide, SIAM, Philadelphia, PA, 1978.

[13] L. M. Ewerbring and F. T. Luk, Canonical correlations and generalized SVD: Applications and New Algorithms, J. Comput. Appl. Math. 27:37-52(1989).

[14] K. V. Fernando, Linear convergence of the row cyclic Jacobi and Kogbetliantz methods, Numer. Math. 56:73–91(1989).

[15] K. V. Fernando and S. J. Hammarling, A product induced singular value decomposition for two matrices and balanced realization, in Linear Algebra in Signals Systems and Control, B. N. Datta *et al* eds. SIAM, Philadelphia, PA. pp.128–140(1988).

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

[17] G. H. Golub and C. F. Van Loan, Matrix Computations (2nd ed), The Johns Hopkins Univ. Press, Baltimore, MD, 1989

[18] S. J. Hammarling, private communications, 1989.

[19] V. Hari and K. Veselić: On Jacobi methods for singular value decomposition, SIAM J. of Sci. Stat. Comp. 6:741–754(1987)

[20] M. T. Heath, J. A. Laub, C. C. Paige and R. C. Ward, Computing the singular value decomposition of a product of two matrices. SIAM J. Sci. Stat. Comput. 7:1147–1159(1986).

[21] S. V. Huffel and J. Vandewalle, Analysis and properties of the generalized total least squares problem $AX \approx B$ when some or all columns in A are subject to error, SIAM J. Mat. Anal. Appl. 10:294–315(1989).

[22] B. Kågström, The generalized singular value decomposition and $(A - \lambda B)$-problem. BIT, 24:568–583(1984).

[23] E. G. Kogbetliantz, Solution of linear equations by diagonalization of coefficients matrix, Quart. Appl. Math. 13:123–132(1955).

[24] F. T. Luk and H. T. Park, On parallel Jacobi orderings, SIAM J. Sci. Stat. Comput. 10:18–26(1989).

[25] C. C. Paige and M. A. Saunders, Towards a generalized singular value decomposition, SIAM J. Numer. Anal. 18:398–405(1981).

[26] C. C. Paige, A note on a result of Sun Ji-guang: sensitivity of the CS and GSV decomposition, SIAM J. Numer. Anal. 21:186–191(1984).

[27] C. C. Paige, The general linear model and generalized singular value decomposition, Lin. Alg. Appl. 70:269–284(1985).

[28] C. C. Paige, Computing the generalized singular value decomposition, SIAM J. Sci. Stat. Comput. 7:1126–1146(1986).

[29] C. C. Paige and P. Van Dooren, A note on the convergence of Kogbetliantz's iterative algorithm for obtaining the singular value decomposition, Lin. Alg. Appl. 77:301–313(1986).

[30] J. M. Speiser and C. F. Van Loan, Signal processing computations using the generalized singular value decomposition, Proc. SPIE Vol.495, Real Time Signal Processing VII, pp.47-55(1984).

[31] G. W. Stewart, Computing the CS-decomposition of a partitioned orthonormal matrix, Numer. Math. 40:297–306(1982).

[32] J. Sun, Perturbation analysis for the generalized singular value problem, SIAM J. Numer. Anal. 20:611–625(1983).

[33] C. F. Van Loan, Generalizing the singular value decomposition, SIAM J. Numer. Anal. 13:76–83(1976).

[34] C. F. Van Loan, Computing the CS and the generalized singular value decomposition, Numer. Math. 46:479–491(1985).

[35] C. F. Van Loan, On the method of weighting for equality- constrained least-squares problems, SIAM J. Numer. Anal. 22:851–864(1985).

[36] H. Zha, A numerical algorithm for computing restricted singular value decomposition of matrix triplets, Lin. Alg. Appl. 168:1-26(1992).