Computing the Generalized Singular Value Decomposition *

Zhaojun Bai† and James W. Demmel‡

Abstract

We present a new numerical method for computing the GSVD [36, 27] of two matrices A and B. This method is a variation on Paige's method [30]. It differs from previous algorithms in guaranteeing both backward stability and convergence. 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 by 2 triangular GSVD algorithm, which constitutes the inner loop of Paige's method. We present proofs of stability and convergence of our method, and demonstrate examples on which all previous algorithms fail.

Subject Classifications: AMS(MOS): 65F30; CR:G1.3

1 Introduction

The purpose of this paper is to describe a variation of the Paige's algorithm for computing the decomposition described in the following theorem.

Theorem 1. *Given a real $m \times n$ matrix A and $p \times n$ matrix B where $\mathrm{rank}((A^{\mathrm{T}}, B^{\mathrm{T}})^{\mathrm{T}}) = n$, the following generalized singular value decomposition (GSVD) of A and B exists:*

$$U^{\mathrm{T}} A Q = \Sigma_1 R, \quad V^{\mathrm{T}} B Q = \Sigma_2 R, \tag{1}$$

where U, V and Q are orthogonal matrices, R is a $n \times n$ nonsingular upper triangular, and

$$\Sigma_1 = \begin{array}{c} \\ r \\ q \\ m-r-q \end{array} \begin{array}{c} \begin{array}{ccc} r & q & n-r-q \end{array} \\ \begin{pmatrix} I_1 & & \\ & D_1 & \\ & & O_1 \end{pmatrix} \end{array}, \quad \Sigma_2 = \begin{array}{c} \\ p-n+r \\ q \\ n-r-q \end{array} \begin{array}{c} \begin{array}{ccc} r & q & n-r-q \end{array} \\ \begin{pmatrix} O_2 & & \\ & D_2 & \\ & & I_2 \end{pmatrix} \end{array}, \tag{2}$$

I_1 and I_2 are identity matrices, O_1 and O_2 are zero matrices,

$$D_1 = \mathrm{diag}(\alpha_{r+1}, \ldots, \alpha_{r+q}), \quad D_2 = \mathrm{diag}(\beta_{r+1}, \ldots, \beta_{r+q}), \tag{3}$$

$$1 > \alpha_{r+1} \geq \cdots \geq \alpha_{r+q} > 0, \quad 0 < \beta_{r+1} \leq \cdots \leq \beta_{r+q} < 1, \tag{4}$$

$$\alpha_i^2 + \beta_i^2 = 1, \quad i = r+1, \ldots, r+q.$$

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

If we denote

$$X = QR^{-1},$$

*This work was supported in part by NSF grants ASC-8715728 and ASC-9005933. The first author was also supported in part by NSF grant ASC-9102963.

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

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

then the GSVD can also be expressed as:

$$U^{\mathrm{T}}AX = \Sigma_1, \qquad V^{\mathrm{T}}BX = \Sigma_2. \tag{5}$$

The GSVD was first introduced by Van Loan [36] in form (5). Form (1) is used by Paige and Saunders [27], and is more amenable to numerical computation. 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 is equivalent to the SVD of AB^{-1}. The pairs (α_i, β_i) defined as

$$\begin{aligned} &\alpha_i = 1, \quad \beta_i = 0, \quad \text{for } i = 1, \ldots, r, \\ &\alpha_i \text{ and } \beta_i \text{ in (3)}, \quad \text{for } i = r+1, \ldots, r+q, \\ &\alpha_i = 0, \quad \beta_i = 1, \quad \text{for } i = r+q+1, \ldots, n, \end{aligned}$$

are called *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 definite pencil $A^{\mathrm{T}}A - \lambda B^{\mathrm{T}}B$.

Other closely related decompositions are the CS decomposition of an orthonormal matrix [34], and the product singular value decomposition (PSVD) of two matrices [21, 16]. The numerical technique developed in this paper can be extended to deal with the numerical computation of these decompositions. We will not go into the details in this paper.

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 [23], the linear constrained least-squares problem [38], the general Gauss-Markov linear model [29, 3], the generalized total least squares problem [22], and real time signal processing [32]. As a further generalization of the SVD, Ewerbring and Luk [13], Zha [39] proposed a SVD for the product of matrix triplets, and De Moor, Golub and Zha [7, 8] 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 [34] and Van Loan [37] proposed an algorithm for computing the GSVD. Their method has two phases: The first phase is to compute the QR decomposition (or the SVD if necessary) of $G = (A^{\mathrm{T}}, B^{\mathrm{T}})^{\mathrm{T}}$. The second phase is to compute the CS decomposition. In order to avoid possible numerical difficulties from combining matrices A and B with very different scalings, Paige proposed a Jacobi-Kogbetliantz approach [30], which we refer to as the *direct GSVD algorithm.* Paige's method applies orthogonal transformations to A and B separately. It also has two phases:

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

$$U^{\mathrm{T}}AP = \begin{array}{c} \\ r \\ q \\ m-r-q \end{array} \begin{array}{c} \begin{array}{ccc} r & q & n-r-q \end{array} \\ \begin{pmatrix} A_{11} & A_{12} & A_{13} \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \end{array} \equiv \begin{array}{c} \\ n \\ m-n \end{array} \begin{array}{c} \begin{array}{c} n \end{array} \\ \begin{pmatrix} R \\ 0 \end{pmatrix} \end{array}, \tag{6}$$

$$V^{\mathrm{T}}BP = \begin{array}{c} \\ r \\ q \\ p-r-q \end{array} \begin{array}{c} \begin{array}{ccc} r & q & n-r-q \end{array} \\ \begin{pmatrix} B_{11} & B_{12} & B_{13} \\ 0 & B_{22} & B_{23} \\ 0 & 0 & 0 \end{pmatrix} \end{array} \equiv \begin{array}{c} \\ n \\ p-n \end{array} \begin{array}{c} \begin{array}{c} n \end{array} \\ \begin{pmatrix} S \\ 0 \end{pmatrix} \end{array}, \tag{7}$$

where U and V are orthogonal matrices, P is a permutation matrix, A_{11} is nonsingular upper triangular, B_{11} is upper triangular, and if $q > 0$, B_{22} is nonsingular upper triangular.

(2). Compute the GSVD of two $n \times n$ upper triangular matrices by a generalized Kogbetliantz algorithm[1].

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

Phase 1 can be done first by the QR factorization with column pivoting of matrix A [18], meanwhile permuting the columns of matrix B in the same way, and then the QR factorization with column pivoting of the last $n - r$ columns of B; this yields forms (6) and (7). Phase 2 is iterative. In particular, if matrix B has full column rank, then phase 2 is mathematically equivalent to computing the SVD of the triangular matrix RS^{-1}.

In this paper, we will present a new numerical method for computing the GSVD. This method is a variation on the method of Paige just presented. Hereafter, we assume that A and B have been preprocessed to the upper trapezoidal forms (6) and (7). There are two innovations. The first is as follows: In Paige's paper [30], it is assumed (without providing detail) that in (7) the nonzero part of $V^{\mathrm{T}}BP$ has full row rank. It is known that it is complicated to choose V to guarantee this condition. In our algorithm, we do not require this condition, and so we can simply use conventional QR factorization with column pivoting.

The second innovation is a new 2 by 2 triangular GSVD algorithm, which constitutes the inner loop of Paige's method. We present proofs of stability and convergence of our method, and demonstrate examples on which all previous algorithms fail.

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

2 Paige's GSVD Algorithm

To describe Paige's direct GSVD algorithm, we first review the Kogbetliantz algorithm [24] 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 is 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 is a Jacobi-like scheme which is based on the following observation. At the kth (i,j) transformation, let

$$A_{ij} = \begin{pmatrix} a_{ii} & a_{ij} \\ 0 & a_{jj} \end{pmatrix}$$

be the 2×2 submatrix subtended by rows and columns i and j of A. Let the rotation matrices

$$U_k = \begin{pmatrix} c_u & s_u \\ -s_u & c_u \end{pmatrix}, \qquad V_k = \begin{pmatrix} c_v & s_v \\ -s_v & c_v \end{pmatrix},$$

be chosen so that

$$U_k^{\mathrm{T}} A_{ij} V_k = \begin{pmatrix} \gamma_{ii} & \\ & \gamma_{jj} \end{pmatrix}$$

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

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

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 [21, 20]. There is a literature on the different sweep orders for sequential and parallel computations besides the conventional row and column order, for example [26, 14].

Forsythe and Henrici [17] considered the convergence of the row cyclic Kogbetliantz algorithm. Fernando [15] proved a global convergence theorem under the weaker 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.,

$$\phi_k \in J \quad \text{or} \quad \psi_k \in J, \qquad k = 1, 2, \ldots \tag{8}$$

This is the condition that our algorithm will satisfy, thus guaranteeing its convergence. Furthermore, Paige and Van Dooren [31] have shown that the cyclic Kogbetliantz algorithm ultimately converges quadratically when no pathologically close singular values are present. In the case of repeated or clustered singular values, the quadratic convergence rate for a triangular matrix has been proved in [2, 6].

2.2 The Generalization of the Kogbetliantz Algorithm for the GSVD

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

$$U_1^{\mathrm{T}} C V_1 = C_1. \tag{9}$$

where C_1 is a lower triangular matrix. Recasting (9) as

$$U_1^{\mathrm{T}} A = C_1 V_1^T B,$$

we see that if we can determine an orthogonal matrix Q_1, which satisfies

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

where A_1, B_1 are lower triangular, then equivalently, we have

$$C_1 = A_1 B_1^{-1}.$$

This reveals that using a sweep of Kogbetliantz algorithm on an upper triangular C to get a lower triangular C_1 is equivalent to the problem of finding orthogonal matrices U_1, V_1 and Q_1 so that U_1 and V_1 transform the implicitly defined upper triangular matrix C to lower triangular form C_1, and meanwhile $U_1^{\mathrm{T}} A Q_1$ and $V_1^{\mathrm{T}} B Q_1$ are lower triangular. Heath *et al* [21], Paige [30] and Hari and Veselić [20] have shown that we may take advantages 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 B^{-1} and AB^{-1} explicitly. Specifically, at transformation (i,j) of the Kogbetliantz algorithm, the needed 2×2 submatrix C_{ij} of C can be obtained by

$$C_{ij} = A_{ij} B_{ij}^{-1} = \begin{pmatrix} a_{ii} & a_{ij} \\ & a_{jj} \end{pmatrix} \begin{pmatrix} b_{ii} & b_{ij} \\ & b_{jj} \end{pmatrix}^{-1}, \tag{10}$$

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. So using the SVD of C_{ij} ($U_{ij}^{\mathrm{T}} C_{ij} V_{ij} = \mathrm{diag}(\tilde{c}_{ii}, \tilde{c}_{jj})$) we have

$$U_{ij}^{\mathrm{T}} A_{ij} = \begin{pmatrix} \tilde{c}_{ii} & \\ & \tilde{c}_{jj} \end{pmatrix} V_{ij}^{\mathrm{T}} B_{ij}.$$

This shows that the corresponding row vectors of $U_{ij}^{\mathrm{T}}A_{ij}$ and $V_{ij}^{\mathrm{T}}B_{ij}$ are parallel. Hence if we choose rotation Q_{ij} so that $V_{ij}^{\mathrm{T}}B_{ij}Q_{ij}$ is lower triangular, then $U_{ij}^{\mathrm{T}}A_{ij}Q_{ij}$ must also be lower triangular, which is just the GSVD of the 2×2 triangular matrices A_{ij} and B_{ij} as desired. 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 $U_{12}U_{13},\ldots,U_{n-1,n}$, $V_{12}V_{13},\ldots,V_{n-1,n}$ and $Q_{12}Q_{13},\ldots,Q_{n-1,n}$, respectively. Let us illustrate the idea in detail for 4×4 matrices A and B. [2] $C = AB^{-1}$ can be written as

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ & a_{22} & a_{23} & a_{24} \\ & & a_{33} & a_{34} \\ & & & a_{44} \end{pmatrix} = \begin{pmatrix} c_{11} & c_{12} & c_{13} & c_{14} \\ & c_{22} & c_{23} & c_{24} \\ & & c_{33} & c_{34} \\ & & & c_{44} \end{pmatrix} \begin{pmatrix} b_{11} & b_{12} & b_{13} & b_{14} \\ & b_{22} & b_{23} & b_{24} \\ & & b_{33} & b_{34} \\ & & & b_{44} \end{pmatrix}. \tag{11}$$

For the (1,2) transformation, the needed 2×2 matrix C_{12} is

$$C_{12} = \begin{pmatrix} c_{11} & c_{12} \\ & c_{22} \end{pmatrix} = A_{12}B_{12}^{-1} = \begin{pmatrix} a_{11} & a_{12} \\ & a_{22} \end{pmatrix} \begin{pmatrix} b_{11} & b_{12} \\ & b_{22} \end{pmatrix}^{-1}.$$

By computing the SVD of the 2×2 matrix C_{12}: $U_{12}^{\mathrm{T}}C_{12}V_{12} = \mathrm{diag}(\tilde{c}_{11},\tilde{c}_{22})$, we have

$$U_{12}^{\mathrm{T}}A_{12} = \begin{pmatrix} \tilde{c}_{11} & \\ & \tilde{c}_{22} \end{pmatrix} V_{12}^{\mathrm{T}}B_{12}.$$

This implies that the corresponding row vectors of $U_{12}^{\mathrm{T}}A_{12}$ and $V_{12}^{\mathrm{T}}B_{12}$ are parallel. Hence if Q_{12} is chosen such that $V_{12}^{\mathrm{T}}B_{12}Q_{12}$ is lower triangular, then $U_{12}^{\mathrm{T}}A_{12}Q_{12}$ must also be lower triangular. Applying this to equation (11), we have

$$\hat{U}_{12}^{\mathrm{T}}A\hat{Q}_{12} = \hat{U}_{12}^{\mathrm{T}}C\hat{V}_{12}\cdot\hat{V}_{12}^{\mathrm{T}}B\hat{Q}_{12},$$

or componentwise,

$$\begin{pmatrix} \tilde{a}_{11} & 0 & \tilde{a}_{13} & \tilde{a}_{14} \\ \tilde{a}_{21} & \tilde{a}_{22} & \tilde{a}_{23} & \tilde{a}_{24} \\ & & \tilde{a}_{33} & \tilde{a}_{34} \\ & & & \tilde{a}_{44} \end{pmatrix} = \begin{pmatrix} \tilde{c}_{11} & 0 & \tilde{c}_{13} & \tilde{c}_{14} \\ & \tilde{c}_{22} & \tilde{c}_{23} & \tilde{c}_{24} \\ & & \tilde{c}_{33} & \tilde{c}_{34} \\ & & & \tilde{c}_{44} \end{pmatrix} \begin{pmatrix} \tilde{b}_{11} & 0 & \tilde{b}_{13} & \tilde{b}_{14} \\ \tilde{b}_{21} & \tilde{b}_{22} & \tilde{b}_{23} & \tilde{b}_{24} \\ & & \tilde{b}_{33} & \tilde{b}_{34} \\ & & & \tilde{b}_{44} \end{pmatrix}.$$

This completes the (1,2) transformation. Next for the (1,3) transformation, we have

$$C_{13} = \begin{pmatrix} \tilde{c}_{11} & \tilde{c}_{13} \\ & \tilde{c}_{33} \end{pmatrix} = \begin{pmatrix} \tilde{a}_{11} & \tilde{a}_{13} \\ & \tilde{a}_{33} \end{pmatrix} \begin{pmatrix} \tilde{b}_{11} & \tilde{b}_{13} \\ & \tilde{b}_{33} \end{pmatrix}^{-1}.$$

Just like (1,2) transformation, we determine rotation matrices U_{13}, V_{13} and Q_{13} such that

$$\hat{U}_{13}^{\mathrm{T}}(\hat{U}_{12}^{\mathrm{T}}A\hat{Q}_{12})\hat{Q}_{13} = \hat{U}_{13}^{\mathrm{T}}(\hat{U}_{12}^{\mathrm{T}}C\hat{V}_{12})\hat{V}_{13}\cdot\hat{V}_{13}^{\mathrm{T}}(\hat{V}_{12}^{\mathrm{T}}B\hat{Q}_{12})\hat{Q}_{13},$$

and componentwise,

$$\begin{pmatrix} \hat{a}_{11} & 0 & 0 & \hat{a}_{14} \\ \hat{a}_{21} & \hat{a}_{22} & \hat{a}_{23} & \hat{a}_{24} \\ \hat{a}_{31} & & \hat{a}_{33} & \hat{a}_{34} \\ & & & \hat{a}_{44} \end{pmatrix} = \begin{pmatrix} \hat{c}_{11} & 0 & 0 & \hat{c}_{14} \\ \hat{c}_{21} & \hat{c}_{22} & \hat{c}_{23} & \hat{c}_{24} \\ & & \hat{c}_{33} & \hat{c}_{34} \\ & & & \hat{c}_{44} \end{pmatrix} \begin{pmatrix} \hat{b}_{11} & 0 & 0 & \hat{b}_{14} \\ \hat{b}_{21} & \hat{b}_{22} & \hat{b}_{23} & \hat{b}_{24} \\ \hat{b}_{31} & & \hat{b}_{33} & \hat{b}_{34} \\ & & & \hat{b}_{44} \end{pmatrix}.$$

[2] By incorporating Gentleman's suggested row and column permutations [30] 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.

Following this procedure until the end of the row cyclic sweep, we obtain lower triangular matrices A_1 and B_1. 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.

Observing an individual transformation, we see we are actually carrying out the Kogbetliantz algorithm to diagonalize the implicitly defined matrix C. At convergence, this gives $U^{\mathrm{T}}(AB^{-1})V = \Sigma$, a diagonal matrix. That is

$$U^{\mathrm{T}} A Q = \Sigma \cdot V^{\mathrm{T}} B Q$$

with the i-th row of $U^{\mathrm{T}}AQ$ parallel to the i-th row of $V^{\mathrm{T}}BQ$, 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 this approach is not recommended, since the 2×2 matrix B_{ij} in (10) may be ill-conditioned or singular. To circumvent this, Paige [30] suggests using

$$C_{ij} = A_{ij} \cdot \mathrm{adj}(B_{ij}) = \begin{pmatrix} a_{ii} & a_{ij} \\ & a_{jj} \end{pmatrix} \begin{pmatrix} b_{jj} & -b_{ij} \\ & b_{ii} \end{pmatrix} \tag{12}$$

instead of C_{ij} in (10) in transformation (i,j), where $\mathrm{adj}(B_{ij})$ stands for the adjugate of matrix 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 (12) into the Kogbetliantz algorithm circumvents the problem of ill-conditioned B_{ij}. But it also introduces two questions. First, are there still rotation matrices U_{ij}, V_{ij} and Q_{ij} such that $U_{ij}^{\mathrm{T}} A_{ij} Q_{ij}$ and $V_{ij}^{\mathrm{T}} B_{ij} Q_{ij}$ are the GSVD of 2 by 2 matrices A_{ij} and B_{ij}? More generally, when B_{ij} is singular, does the scheme still gives the GSVD of 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 by 2 Triangular Matrices

As we showed in the last section, the kernel of computing the GSVD using a generalized Kogbetliantz algorithm is the computation of the GSVD of 2 by 2 matrices. Substituting $\mathrm{adj}(B_{ij})$ for B_{ij}^{-1} in the 2×2 GSVD circumvents the problem of ill-conditioned B_{ij}. However when the idea is extended to two $n \times n$ matrices A and B of the forms (6) and (7), the 2×2 matrix B_{ij} might well be exactly singular. This immediately raises the questions of how to compute the 2×2 GSVD, and how to guarantee eventual convergence starting from the forms (6) and (7). 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 presence of the floating point arithmetic.

3.1 The 2×2 GSVD in exact arithmetic

When A and B are processed to have upper trapezoidal forms (6) and (7), we see that transformation (i,j) in row order, the 2×2 matrices A and B are of the forms[3]

$$A = \begin{pmatrix} a_{11} & a_{12} \\ 0 & a_{22} \end{pmatrix} \quad \text{and} \quad B = \begin{pmatrix} b_{11} & b_{12} \\ 0 & b_{22} \end{pmatrix}, \tag{13}$$

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

[3]For simplicity of exposition, we drop the subscript ij from the 2 by 2 triangular matrices A_{ij} and B_{ij}, which are obtained by subtending rows and columns i and j of $n \times n$ matrices A and B.

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

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

is the GSVD of A *and* B. *Moreover*

(i). $\tilde{a}_{11} \neq 0$ *if* A *is nonzero.*

(ii). $\tilde{b}_{22} \neq 0$ *if both* A *and* B *are nonzero, unless* $A = \begin{pmatrix} a_{11} & a_{12} \\ 0 & 0 \end{pmatrix}$ *and* $B = \begin{pmatrix} b_{11} & b_{12} \\ 0 & 0 \end{pmatrix}$, *and first rows of* A *and* B *are parallel. In this case,* $U = V = I$, *and* Q *is chosen such that*

$$AQ = \begin{pmatrix} \tilde{a}_{11} & 0 \\ 0 & 0 \end{pmatrix}, \quad BQ = \begin{pmatrix} \tilde{b}_{11} & 0 \\ 0 & 0 \end{pmatrix}. \tag{14}$$

Proof. The proof proceeds by considering all possible cases. If B is nonsingular, the lemma follows immediately by the discussion in section 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:

(1). $B = \begin{pmatrix} b_{11} & b_{12} \\ 0 & 0 \end{pmatrix}$ with $b_{11} \neq 0$. In this case, $C = A \cdot \mathrm{adj}(B) = \begin{pmatrix} 0 & a_{12}b_{11} - a_{11}b_{12} \\ 0 & a_{22}b_{11} \end{pmatrix} \equiv \begin{pmatrix} 0 & c_{12} \\ 0 & c_{22} \end{pmatrix}$. 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 lemma for the special case (14). 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 $U^{\mathrm{T}}A$.

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

(2). $B = \begin{pmatrix} 0 & b_{12} \\ 0 & b_{22} \end{pmatrix}$ with $b_{22} \neq 0$. Hence $C = A \cdot \mathrm{adj}(B) = a_{11} \begin{pmatrix} b_{22} & -b_{12} \\ 0 & 0 \end{pmatrix}$. 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 $C = A \cdot \mathrm{adj}(B) = \begin{pmatrix} 0 & -a_{11}b_{12} \\ 0 & 0 \end{pmatrix}$. Then we can choose $U = I$, V being a permutation. Therefore the second row of $V^{\mathrm{T}}B$ is nonzero. The lemma follows by choosing Q to zero (1,2) entry of $U^{\mathrm{T}}A$. □

It has been shown by induction (see [30, 4]) that with the properties of Lemma 2, a sweep in row order with possible reordering takes the initial upper trapezoidal forms (6) and (7) of A and B into the forms

$$A_1 = U_1^{\mathrm{T}} A Q_1 = \begin{array}{c} \\ r \\ n-r \end{array} \begin{array}{c} \begin{array}{cc} r & n-r \end{array} \\ \begin{pmatrix} A_{11} & 0 \\ 0 & 0 \end{pmatrix} \end{array}, \quad B_1 = V_1^{\mathrm{T}} B Q_1 = \begin{array}{c} \\ r \\ n-r \end{array} \begin{array}{c} \begin{array}{cc} r & n-r \end{array} \\ \begin{pmatrix} B_{11} & 0 \\ B_{21} & B_{22} \end{pmatrix} \end{array}, \tag{15}$$

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 (15), we see that at transformation (i, j) of the column order sweep, the 2×2 matrices A and B are of the forms

$$A = \begin{pmatrix} a_{11} & 0 \\ a_{21} & a_{22} \end{pmatrix}, \quad B = \begin{pmatrix} b_{11} & 0 \\ b_{21} & b_{22} \end{pmatrix}, \tag{16}$$

where if A is singular, then A is either the zero matrix or its second row is zero. If $b_{22} = 0$, then $b_{21} = 0$. For the GSVD of these 2 by 2 lower triangular matrices, we have the following lemma.

Lemma 3. *There exist rotations U, V and Q such that*

$$\tilde{A} = U^{\mathrm{T}} A Q = \begin{pmatrix} \tilde{a}_{11} & \tilde{a}_{12} \\ & \tilde{a}_{22} \end{pmatrix}, \quad \tilde{B} = V^{\mathrm{T}} B Q = \begin{pmatrix} \tilde{b}_{11} & \tilde{b}_{12} \\ & \tilde{b}_{22} \end{pmatrix},$$

is the GSVD of A and B, and moreover

(i). $\tilde{a}_{11} \neq 0$ if A is nonzero.

(ii). $\tilde{b}_{22} \neq 0$, if both A and B are nonzero. If $\tilde{b}_{11} = 0$, then $\tilde{b}_{12} = 0$.

Proof. The proof is by analogy with the proof of Lemma 2. □

The proof of Lemma 2 and 3 suggest the following algorithm to compute the GSVD of 2 by 2 triangular matrices.

Algorithm 4 (The 2×2 GSVD algorithm).

form $C = A \cdot \mathrm{adj}(B)$;
compute the SVD of C: $U^{\mathrm{T}} C V = \mathrm{diag}(\sigma_1, \sigma_2)$;
form the products $G = U^{\mathrm{T}} A$, $H = V^{\mathrm{T}} B$;
if A and B are upper triangular matrices
  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
  $\tilde{A} = GQ$; $\tilde{B} = HQ$; $\tilde{a}_{12} = 0$; $\tilde{b}_{12} = 0$;
else if A and B are lower triangular matrices
  if A is nonsingular, then
    determine Q to zero out (2,1) entry of G;
  else
    determine Q to zero out (2,1) entry of H;
  end if
  $\tilde{A} = GQ$; $\tilde{B} = HQ$; $\tilde{a}_{21} = 0$; $\tilde{b}_{21} = 0$;
end if

By induction, it has been shown [30, 4] that under the properties stated in Lemma 3, after the second sweep in column order, we have

$$A_2 = U_2^{\mathrm{T}} A_1 Q_2 = \begin{array}{c} \\ r_1 \\ r_2 \\ n-r \end{array} \begin{array}{c} \begin{array}{ccc} r_1 & r_2 & n-r \end{array} \\ \begin{pmatrix} A_{11} & A_{12} & A_{13} \\ 0 & A_{22} & A_{23} \\ 0 & 0 & 0 \end{pmatrix} \end{array}, \quad B_2 = V_2^{\mathrm{T}} B_1 Q_2 = \begin{array}{c} \\ r_1 \\ r_2 \\ n-r \end{array} \begin{array}{c} \begin{array}{ccc} r_1 & r_2 & n-r \end{array} \\ \begin{pmatrix} 0 & 0 & 0 \\ 0 & B_{22} & B_{23} \\ 0 & 0 & B_{33} \end{pmatrix} \end{array} \tag{17}$$

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

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

This implies that the rest of computation is mathematically equivalent to computing the SVD of the implicitly defined matrix G. By the global convergence theory of the cyclic Kogbetliantz algorithm

(see section 2.1), we have

$$G \to \Sigma, \tag{18}$$

where Σ is a diagonal matrix, and the convergence is ultimately quadratic, provided the rotation angles of U and V obey (8). (18) implies that

$$A_2 \to \Sigma_1 R, \quad B_2 \to \Sigma_2 R$$

which is the desired GSVD of A and B.

3.2 The 2×2 GSVD in Presence of Floating Point Arithmetic

In this section, we will use the usual model of floating point arithmetic: barring over/underflow, $fl(x \circ y) = (1+\delta)(x \circ y)$ where $\circ$ is one of the basic operations $\{+, -, \times, \div\}$ and $|\delta| \leq \epsilon$ 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 4 not to be parallel. This means $\tilde{A}$ and $\tilde{B}$ are not part of 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 4 may be much larger than $O(\epsilon)\|A\|$ and $O(\epsilon)\|B\|$, respectively. Thus, the algorithms in [30, 21, 4], which use SVD of 2 by 2 triangular matrix to guarantee convergence, do not avoid numerical instability. On the other hand, to guarantee numerical stability, it has been suggested in [19, 5] that after computing the SVD of the 2×2 triangular matrix C: $U^{\mathrm{T}}CV = \mathrm{diag}(\sigma_1, \sigma_2)$, one should use U (say), to form $G = U^{\mathrm{T}}A$, then determine Q such that GQ is lower triangular, and finally determine $\tilde{V}$ such that $\tilde{V}^{\mathrm{T}}BQ$ is also lower triangular. However, $U^{\mathrm{T}}C\tilde{V}$ might not be diagonal at all, which results in nonconvergence. In section 5, we will present numerical examples illustrating the failures of these schemes. In this section, we will propose a new algorithm to overcome these shortcomings. We first discuss the two fundamental algorithmic building blocks: **SROTG** and **SLASV2**.

SROTG takes f and g as input and computes $c = \cos\theta$, $s = \sin\theta$ and r so that

$$\begin{pmatrix} c & s \\ -s & c \end{pmatrix} \begin{pmatrix} f \\ g \end{pmatrix} = \begin{pmatrix} r \\ 0 \end{pmatrix}. \tag{19}$$

Clearly one could simply compute $c = f/sqrt(f^2+g^2)$ and $s = g/sqrt(f^2+g^2)$, but this is subject to spurious over/underflow. A more robust way to compute c, s and r is as follows:

SROTG(f, g, c, s, r)
  if $(f = 0)$ then
    $c = 0$; $s = 1$; $r = g$;
  elseif $(|f| > |g|)$ then
    $t = g/f$; $y = sqrt(1 + t*t)$;
    $c = 1/y$; $s = t*c$; $r = f*y$;
  else
    $t = f/g$; $y = sqrt(1 + t*t)$;
    $s = 1/y$; $c = t*s$; $r = g*y$;
  end if

Barring underflow and overflow (which can only occur if the true value of r itself would nearly overflow), **SROTG** computes c, s and r to nearly full machine accuracy.

The second basic algorithm is **SLASV2**, which 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 high accuracy is essential in the proof of correctness of the overall algorithm. This algorithm was described briefly in [9], but not published in its entirety. For completeness, we include a listing in the appendix of this paper. As discussed in [9], the high accuracy of **SLASV2** is based on the fact that the algorithm uses formulas for the answers 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[4].

Using the algorithms **SROTG** and **SLASV2**, we now give a high-level description of an algorithm for computing the GSVD of 2 by 2 upper triangular matrices. Later we will show that the algorithm guarantees numerical stability and convergence. We will use the notation $|X|$ to denote the componentwise absolute value of the matrix X.

Algorithm GSVD22 (The GSVD of 2 by 2 upper triangular matrices): Let the 2×2 upper triangular matrices A and B be denoted

$$A = \begin{pmatrix} a_{11} & a_{12} \\ 0 & a_{22} \end{pmatrix}, \quad B = \begin{pmatrix} b_{11} & b_{12} \\ 0 & b_{22} \end{pmatrix}.$$

The following algorithm computes the orthogonal matrices

$$U = \begin{pmatrix} c_u & s_u \\ -s_u & c_u \end{pmatrix}, \quad V = \begin{pmatrix} c_v & s_v \\ -s_v & c_v \end{pmatrix}, \quad Q = \begin{pmatrix} c_q & s_q \\ -s_q & c_q \end{pmatrix},$$

such that

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

are the GSVD of A and B. In the interest of brevity, we omit the part for the 2 by 2 lower triangular matrices, which can be described similarly.

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

[4]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. This is why a different proof is needed on a Cray.

else

/ Choose Q to zero out (2,2) entries of $U^{\mathrm{T}}A$ and $V^{\mathrm{T}}B$ and then swap rows. */*

if $\hat{g}_{22}/(|g_{21}| + |g_{22}|) \leq \hat{h}_{22}/(|h_{21}| + |h_{22}|)$ *then*

call **SROTG**$(-g_{21}, g_{22}, c_q, s_q, r)$ */* Compute Q from $U^{\mathrm{T}}A$ */*

else

call **SROTG**$(-h_{21}, h_{22}, c_q, s_q, r)$ */* Compute Q from $V^{\mathrm{T}}B$ */*

end if

$\tilde{A} = G * Q; \tilde{B} = H * Q; \tilde{a}_{22} = 0; \tilde{b}_{22} = 0.$

/ Swap, where* $P = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}$ **/*

$\tilde{A} \leftarrow P * \tilde{A}; \tilde{B} \leftarrow P * \tilde{B};$

$U \leftarrow U * P; V \leftarrow V * P;$

end if

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

Theorem 5. *The $\tilde{A}$ and $\tilde{B}$ computed by Algorithm* **GSVD22** *have the following properties.*

(a) Both are triangular;

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

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

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

Proof. Without loss of generality, we assume that $\|A\| = 1$, $\|B\| = 1$. We only prove a branch of the algorithm where Q is computed from $U^{\mathrm{T}}A$ and used to zero out the (1,2) entries of $U^{\mathrm{T}}A$ and $V^{\mathrm{T}}B$; the proof for the other cases is similar.

We first note the following simple facts about the algorithm:

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.
2. The computed $\bar{U}$ and $\bar{V}$ from **SLASV2** satisfy $\bar{U} = U + \delta U$, $\bar{V} = V + \delta V$, where $U^{\mathrm{T}}\bar{C}V$ is the exact SVD of $\bar{C}$ and δU (δV) is a small componentwise relative perturbation of U (V).
3. The roundoff error in the computed $\bar{g}_{ij}$ ($\bar{h}_{ij}$) is bounded by $2\epsilon\bar{\hat{g}}_{ij}$ ($2\epsilon\bar{\hat{h}}_i$).

We note that triangularity (a) holds by construction. Let $\eta_a = \bar{\hat{g}}_{12}\epsilon/(|\bar{g}_{11}| + |\bar{g}_{12}|)$, 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 $U^{\mathrm{T}}A$ ($V^{\mathrm{T}}B$). In the branch of the algorithm we consider, $\eta_a \leq \eta_b$. There are two cases:

Case 1: $\eta_a = O(\epsilon)$. Then the computed $\bar{Q} = Q + \delta Q$ has relative errors of size $O(\epsilon)$ in each component; here Q is exact rotation such that $U^{\mathrm{T}}(A + \delta A_1)Q$ and $V^{\mathrm{T}}(B + \delta B_1)Q$ is the exact GSVD of $A + \delta A_1$ and $B + \delta B_1$. Since $\eta_a = O(\epsilon)$, $\bar{U}$ has $O(\epsilon)$ relative errors in each component, so

$$\begin{aligned} fl(\bar{G}\bar{Q}) &= \bar{G}\bar{Q} + O(\epsilon) = \bar{U}^{\mathrm{T}}A\bar{Q} + O(\epsilon) \\ &= (U + \delta U)^{\mathrm{T}}(A - \delta A_1 + \delta A_1)(Q + \delta Q) + O(\epsilon) = U^{\mathrm{T}}(A + \delta A_1)Q + O(\epsilon) \end{aligned}$$

is within $O(\epsilon)$ of the exact GSVD of $A + \delta A_1$. Similarly, $fl(\bar{H}\bar{Q}) = V^{\mathrm{T}}(B + \delta B_1)Q + O(\epsilon)$ is within $O(\epsilon)$ of the exact GSVD of $B + \delta B_1$. This proves assertion (c). Assertion (b) follows since the entries explicitly set to zero are $O(\epsilon)$ and so

$$\bar{\tilde{A}} = fl(\bar{G}\bar{Q}) + O(\epsilon) = U^{\mathrm{T}}(A + \delta A_1)Q + O(\epsilon) = U^{\mathrm{T}}(A + \delta A)Q$$

and

$$\bar{\tilde{B}} = fl(\bar{H}\bar{Q}) + O(\epsilon) = V^{\mathrm{T}}(B + \delta B_1)Q + O(\epsilon) = V^{\mathrm{T}}(B + \delta B)Q$$

where $\|\delta A\| = O(\epsilon)$ and $\|\delta B\| = O(\epsilon)$.

Case 2: $\eta_a \gg \epsilon$. First we prove assertion (c), then (b). The top rows of $\tilde{A}$ and $\tilde{B}$ are trivially parallel by construction (their second components are zero), so we only consider the bottom rows. From the full machine accuracy of the singular value decomposition of 2 by 2 triangular matrix computed by **SLASV2**, we know that bottom row of $U^{\mathrm{T}}(A+\delta A_1)$ and the bottom row of $V^{\mathrm{T}}(B+\delta B_1)$ are parallel. Thus the bottom rows of $\bar{G} = U^{\mathrm{T}}(A+\delta A_1)+O(\epsilon)$ and $\bar{H} = V^{\mathrm{T}}(B+\delta B_1)+O(\epsilon)$ are within $O(\epsilon)$ of being parallel. So for *any* $\bar{Q}$, the bottom row of $fl(\bar{G}\bar{Q}) = \bar{G}\bar{Q} + O(\epsilon)$ and the bottom row of $fl(\bar{G}\bar{Q}) = \bar{G}\bar{Q} + O(\epsilon)$ are within $O(\epsilon)$ of being parallel. This proves assertion (c).

To prove assertion (b), we need to show the (1,2) entry of $fl(\bar{H}\bar{Q})$, which is zeroed out to get $\tilde{B}$, is $O(\epsilon)$. (Q is chosen to accurately zero out the (1,2) entry of $fl(\bar{G}\bar{Q})$.)

$$fl((\bar{H}\bar{Q})_{12}) = \bar{h}_{11}\bar{s}_q(1 + O(\epsilon)) + \bar{h}_{12}\bar{c}_q(1 + O(\epsilon)) = \bar{h}_{11}\bar{s}_q + \bar{h}_{12}\bar{c}_q + O(\epsilon).$$

It is easy to see that

$$\bar{h}_{11} = h_{11} + O(\epsilon), \quad \bar{h}_{12} = h_{12} + O(\epsilon).$$

Also

$$|\bar{h}_{11}| + |\bar{h}_{12}| = \frac{\epsilon}{\eta_b}\bar{\tilde{h}}_{12} = \frac{O(\epsilon)}{\eta_b}.$$

Next, by construction, we know that $\bar{c}_q = c_q + \delta c_q$ and $\bar{s}_q = s_q + \delta s_q$, where $|\delta c_q|, |\delta s_q| \le \eta_a$. Thus

$$\begin{aligned}
|fl((\bar{H}\bar{Q})_{12})| &= |(h_{11} + O(\epsilon))(s_q + \delta s_q) + (h_{12} + O(\epsilon))(c_q + \delta c_q) + O(\epsilon)| \\
&= |(h_{11}s_q + h_{12}c_q) + (h_{11}\delta s_q + h_{12}\delta c_q) + O(\epsilon)| \\
&\le |h_{11}|\,|\delta s_q| + |h_{12}|\,|\delta c_q| + O(\epsilon) \\
&\le (|h_{11}| + |h_{12}|)\eta_a + O(\epsilon) \\
&\le \frac{O(\epsilon)}{\eta_b}\eta_a + O(\epsilon) = O(\epsilon)
\end{aligned}$$

since $\eta_a \le \eta_b$. This proves assertion (b). □.

4 Summary of the Complete Algorithm

Summarizing the above algorithms, we present a high-level version of Paige's direct GSVD algorithm for computing the GSVD of two upper triangular matrices A and B of the forms (6) and (7). 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 (The Direct GSVD Algorithm)

/* Initialization */

cycle := 0;

$l := r + q + 1$;

$U := I$; $V := I$; $Q := I$ if desired;

/* Main loop */
if *nonconvergence* and $cycle \le \kappa$ do
  $cycle := cycle + 1$;
  do (i,j)-loop
    /* 2×2 GSVD */
    call Algorithm GSVD22 to find U_{ij}, V_{ij}, Q_{ij} from a_{ii}, a_{ij}, a_{jj} and b_{ii}, b_{ij}, b_{jj};
    /* Updating */
    $A := U_{ij}^{\mathrm{T}} A Q_{ij}$;
    $B := V_{ij}^{\mathrm{T}} B Q_{ij}$;
    $U := UU_{ij}$; $V := VV_{ij}$; $Q := QQ_{ij}$ if desired;
    /* reordering */
    if creates a nonzero at (j,j) position of B, where $j > l$, then
      $A := AP_{lj}$;
      $B := P_{lj} B P_{lj}$;
      $V := VV_{lj}$; $Q := QP_{lj}$ if desired;
      $l := l + 1$;
    end if
  end of (i,j)-loop
  convergence test if cycle is even.
end if

The (i,j)-loop in the above algorithm can be simply chosen as the standard cyclic pivot sequence, i.e., in odd cycle, (i,j)-loop is row ordering (1,2), (1,3), ..., $(1,n)$, (2,3), ..., $(2,n)$, ..., $(n-1,n)$, and in even cycle, (i,j)-loop is column ordering (2,1), (3,1), ..., $(n,1)$, (3,2), ..., $(n,2)$, ..., $(n,n-1)$.

It is natural to use the parallelism (linear dependency) of the corresponding row vectors of $U_k^{\mathrm{T}} A Q_k$ and $V_k^{\mathrm{T}} B Q_k$ as stopping criterion of the iteration. To measure the parallelism of two k-vectors a and b in high accuracy and against possible over/underflow, we propose the following scheme: first compute the QR factorization of the $k \times 2$ matrix $\left(\frac{a}{\|a\|}, \frac{b}{\|b\|}\right)$:

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

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

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

measures the parallelism of these two vectors. If vectors a and b are exactly parallel, then $\gamma_2 = 0$.

Using the above described scheme as 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 even cycle, we take

$$\mathrm{error} = \sum_{i=1}^{n} \mathrm{par}(a_i, b_i)$$

as an indicator of convergence. For a given tolerance value tol, if

$$\mathrm{error} \le n \cdot tol,$$

then the corresponding row vectors of A and B are tol-parallel. This means that there are perturbations of size at most $n \cdot tol \cdot \|a_i\|$ in row a_i and $n \cdot tol \cdot \|b_i\|$ in row b_i that make them exactly parallel. This means that after making these perturbations, there would exist scalars α_i and β_i such that

$$\beta_i a_i = \alpha_i b_i, \quad i = 1, \ldots, n \tag{20}$$

where α_i and β_i can be chosen so that $\alpha_i^2 + \beta_i^2 = 1$. From (20), it is seen that there would then be an upper triangular matrix R, such that

$$U^{\mathrm{T}} A Q = \mathrm{diag}(\alpha_i) R, \quad V^{\mathrm{T}} B Q = \mathrm{diag}(\beta_i) R,$$

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

We note that the algorithm we have described does not give the GSV pairs in the order as stated in (4) but this can be achieved with an additional sweep after convergence.

5 Numerical Experiments

The numerical experiments we discuss here first compare our new 2×2 GSVD algorithm with previous 2×2 GSVD algorithms developed by Paige [30], Heath et at [21], Bai[4], Hammarling [19] and Bojanczyk et al [5] etc. Then we will evaluate our direct GSVD algorithm 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, and elapsed time when computing both GSV pairs and orthogonal matrices.

All tests were performed using FORTRAN 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 $tol = 10^{-14}$ 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 the direct GSVD algorithm. 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:

$$\|\bar{U}^{\mathrm{T}}\bar{U} - I\|_{\mathrm{F}} \approx \epsilon; \quad \|\bar{V}^{\mathrm{T}}\bar{V} - I\|_{\mathrm{F}} \approx \epsilon; \quad \|\bar{Q}^{\mathrm{T}}\bar{Q} - I\|_{\mathrm{F}} \approx \epsilon; \tag{21}$$

$$\frac{\|\bar{U}^{\mathrm{T}} A \bar{Q} - \bar{\Sigma}_1 \bar{R}\|_{\mathrm{F}}}{n \cdot \|A\|_{\mathrm{F}}} \approx \epsilon; \quad \frac{\|\bar{V}^{\mathrm{T}} B \bar{Q} - \bar{\Sigma}_2 \bar{R}\|_{\mathrm{F}}}{n \cdot \|B\|_{\mathrm{F}}} \approx \epsilon. \tag{22}$$

where $\|\cdot\|_{\mathrm{F}}$ 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 $\bar{U}^{\mathrm{T}} A \bar{Q}$ and $\bar{V}^{\mathrm{T}} B \bar{Q}$ are parallel.

The accuracy test of computed GSV pairs by the direct GSVD algorithm is based on Sun's [35] and Paige's [28] perturbation bound of the GSV pairs, which says that :

> *If* $rank(G) = rank(\tilde{G}) = n$, *where* $G = (A^{\mathrm{T}}, B^{\mathrm{T}})^{\mathrm{T}}$, *and* $\tilde{G} = (\tilde{A}^{\mathrm{T}}, \tilde{B}^{\mathrm{T}})^{\mathrm{T}} = ((A + E)^{\mathrm{T}}, (B + F)^{\mathrm{T}})^{\mathrm{T}}$, *and the generalized singular values 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 (4), then we have*

$$\sqrt{\sum_{i=1}^{n} [(\alpha_i - \tilde{\alpha}_i)^2 + (\beta_i - \tilde{\beta}_i)^2]} \leq \sqrt{2} \min\{\|G^{-1}\|_2, \|\tilde{G}^{-1}\|_2\} \left\| \begin{pmatrix} E \\ F \end{pmatrix} \right\|_{\mathrm{F}} \tag{23}$$

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

5.2 The numerical comparison of different 2×2 GSVD algorithms

Since the 2×2 GSVD is the inner loop of the direct GSVD algorithm, its accuracy and stability determine the accuracy and stability of the overall algorithm. Several versions have been proposed in the literature. There are essentially two kinds of schemes:

Scheme I: First compute the SVD of $C = A \cdot \mathrm{adj}(B)$: $U^{\mathrm{T}}CV = \Sigma$, then form the product of $G = U^{\mathrm{T}}A$ and $H = V^{\mathrm{T}}B$, 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 [30], Heath et al [21], Bai [4] fall in this category.

Scheme II: First compute the SVD of $C = A \cdot \mathrm{adj}(B)$: $U^{\mathrm{T}}CV = \Sigma$, form the product of $G = U^{\mathrm{T}}A$, 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 [19] and Bojanczyk et al [5] 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 [30], Heath et al [21] and Bai [4], for the computed $\bar{U}$, $\bar{V}$ and $\bar{Q}$, we have

$$\tilde{A} = \begin{pmatrix} \tilde{a}_1 \\ \tilde{a}_2 \end{pmatrix} = \bar{U}^{\mathrm{T}}A\bar{Q} = \begin{pmatrix} 0.70710677509009934D+00 & 0.21213203455917919D+01 \\ 0.00000000000000000D+00 & 0.28284271491319831D-07 \end{pmatrix},$$

$$\tilde{B} = \begin{pmatrix} \tilde{b}_1 \\ \tilde{b}_2 \end{pmatrix} = \bar{V}^{\mathrm{T}}B\bar{Q} = \begin{pmatrix} 0.31622776518779000D+00 & 0.94868330465141837D+00 \\ -0.33959487444334968D-08 & 0.31622776582710133D+01 \end{pmatrix}.$$

If we now set the (2,1) entry of $\tilde{B}$ to zero, the backward stability condition (22) is violated for matrix B, even though

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

To show how Scheme II can fail for the same example, using Hammarling's suggested method [19], we have

$$\tilde{A} = \begin{pmatrix} \tilde{a}_1 \\ \tilde{a}_2 \end{pmatrix} = \bar{U}^{\mathrm{T}}A\bar{Q} = \begin{pmatrix} 0.70710677509009934D+00 & 0.21213203455917919D+01 \\ 0.00000000000000000D+00 & 0.28284271491319831D-07 \end{pmatrix},$$

$$\tilde{B} = \begin{pmatrix} \tilde{b}_1 \\ \tilde{b}_2 \end{pmatrix} = \bar{V}^{\mathrm{T}}B\bar{Q} = \begin{pmatrix} -0.31622776518778994D+00 & -0.94868327069193081D+00 \\ 0.11102230246251565D-15 & -0.31622776684588585D+01 \end{pmatrix}.$$

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

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

which is not within $O(\epsilon)\|C\|$ of diagonal form. This means that the computed $\tilde{A}$ and $\tilde{B}$ are not the GSVD of A and B, and $\mathrm{par}(\tilde{a}_1, \tilde{b}_1) = 7.5935 \times 10^{-9}$.

But using our new scheme described in section 3.2, we have

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

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

and also

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

Thus both stability and convergence conditions are satisfied and $\mathrm{par}(\tilde{a}_1, \tilde{b}_1) = 7.0216 \times 10^{-17}$.

Recently, Bojanczyk et al [5] 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 $A_3 = I$. The BELV scheme does significantly improve Hammarling's method, but it still suffers from nonconvergence [5]. Using the BELV scheme, we see that for the following 2×2 matrices

$$A = \begin{pmatrix} 100 & 100 \\ 0 & 0.0001 \end{pmatrix}; \quad B = \begin{pmatrix} 100 & 100.000001 \\ 0 & 0.003 \end{pmatrix}.$$

the computed orthogonal matrices $\bar{U}$, $\bar{V}$ and $\bar{Q}$ by BELV scheme satisfy the stablity conditions (21) and (22):

$$\tilde{A} = \begin{pmatrix} \tilde{a}_1 \\ \tilde{a}_2 \end{pmatrix} = \bar{U}^{\mathrm{T}}A\bar{Q} = \begin{pmatrix} 0.70710678123422628D-04 & -0.16438067791142968D-02 \\ 0.00000000000000000D+00 & 0.14142135622777383D+03 \end{pmatrix}$$

$$\tilde{B} = \begin{pmatrix} \tilde{b}_1 \\ \tilde{b}_2 \end{pmatrix} = \bar{V}^{\mathrm{T}}B\bar{Q} = \begin{pmatrix} 0.21213204616848981D-02 & -0.49314206025121626D-01 \\ -0.14210854715202004D-13 & 0.14142134836229314D+03 \end{pmatrix}$$

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

$$\bar{U}^{\mathrm{T}}C\bar{V} = \begin{pmatrix} 0.99999994438265301D-02 & -0.64369760606822202D-14 \\ 0.19310930892552092D-12 & 0.30000001668520510D+00 \end{pmatrix}$$

since the off-diagonal elements are much larger than $O(\epsilon)\|C\| \approx 10^{-17}$, and $\mathrm{par}(\tilde{a}_1, \tilde{b}_1) = 5.6045 \times 10^{-11}$, i.e., the first rows of $\tilde{A}$ and $\tilde{B}$ are not parallel. But our new scheme algorithm **GSVD22** yields

$$\tilde{A} = \begin{pmatrix} \tilde{a}_1 \\ \tilde{a}_2 \end{pmatrix} = \bar{U}^{\mathrm{T}}A\bar{Q} = \begin{pmatrix} -0.14142135622777380D+03 & 0.00000000000000000D+00 \\ -0.16438067791142966D-02 & -0.70710678123422520D-04 \end{pmatrix}$$

$$\tilde{B} = \begin{pmatrix} \tilde{b}_1 \\ \tilde{b}_2 \end{pmatrix} = \bar{V}^{\mathrm{T}}B\bar{Q} = \begin{pmatrix} -0.14142134836229312D+03 & 0.00000000000000000D+00 \\ -0.49314206116154218D-01 & -0.21213204616848981D-02 \end{pmatrix}$$

and

$$\bar{U}^{\mathrm{T}}C\bar{V} = \begin{pmatrix} 0.30000001668520498D+00 & 0.13552527156068805D-19 \\ -0.42351647362715017D-21 & 0.99999994438265266D-02 \end{pmatrix}$$

which is stable and convergent, with $\mathrm{par}(\tilde{a}_1, \tilde{b}_1) = 0$, and $\mathrm{par}(\tilde{a}_2, \tilde{b}_2) = 9.0122 \times 10^{-17}$. It is possible to construct more extreme examples to show that the off diagonal elements of $\bar{U}^{\mathrm{T}}C\bar{V}$ of the BELV scheme are as large as order O(1).

The above examples show that our new scheme is superior to all previous known schemes.

5.3 Test matrix generation for testing backward stability

To test the backward stability of our **GSVD** algorithm, we used the LAPACK test matrix generation suite [10] to generate different types of upper triangular matrices A and B to test the backward stability of the algorithm. 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, `cond` ≥ 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 numbers in the range (1/`cond` , 1) such that 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 as follows:

Class 1: Matrix A was generated by taking `dist` = U, `cond` = 10, `mode` = 6, matrix B by taking `dist` = U, `cond` = 10, `mode` = 6.

Class 2: Matrix A was generated by taking `dist` = U, `cond` = 10^2, `mode` = 2, matrix B by taking `dist` = S, `cond` = 10, `mode` = 6.

Class 3: Matrix A was generated by taking `dist` = U, `cond` = 10^5, `mode` = 1, matrix B by taking `dist` = N, `cond` = 10, `mode` = 5.

Class 4: Matrix A was generated by taking `dist` = S, `cond` = 10^8, `mode` = 3, matrix B by taking `dist` = S, `cond` = 10, `mode` = 6.

Class 5: Matrix A was generated by taking `dist` = S, `cond` = 10^{12}, `mode` = 4, matrix B by taking `dist` = U, `cond` = 10, `mode` = 5.

Class 6: Matrix A was generated by taking `dist` = S, `cond` = 10^{14}, `mode` = 4, matrix B by taking `dist` = N, `cond` = 10, `mode` = 6.

Class 7: Matrix A was generated by taking `dist` = N, `cond` = 10, `mode` = 6, matrix B by taking `dist` = N, `cond` = 10^5, `mode` = 1.

Class 8: Matrix A was generated by taking `dist` = N, `cond` = 10, `mode` = 6, matrix B by taking `dist` = U, `cond` = 10^8, `mode` = 2.

Class 9: Matrix A was generated by taking `dist` = N, `cond` = 10, `mode` = 6, matrix B by taking `dist` = S, `cond` = 10^{12}, `mode` = 2.

Class 10: Matrix A was generated by taking `dist` = S, `cond` = 10, `mode` = 6, matrix B by taking `dist` = N, `cond` = 10^{14}, `mode` = 4.

Class 11: Matrix A was generated by taking `dist` = S, `cond` = 10^5, `mode` = 4, matrix B by taking `dist` = N, `cond` = 10^5, `mode` = 4.

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 1: Average Number of double sweeps

Class 12: Matrix A was generated by taking `dist` = S, `cond` = 10^3, `mode` = 3, matrix B by taking `dist` = N, `cond` = 10^4, `mode` = 4.

Thus classes 1-6 consist of well-conditioned matrices B, and the conditioning of matrix A is changed from well to ill-conditioned. Classes 7-10 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 random matrices, in each class of dimension 10 we generated 301 random matrix pairs, in each class of dimension 20 we generated 201 random matrix pairs, and in each class of dimension 50 we generated 101 random matrix pairs. This makes a total of 12,048 different test matrix pairs.

Table 1 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 (21) and (22) 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 matrices 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 our algorithm, 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 used the LAPACK test matrix generation suite [10] to generate random orthogonal matrices U, V and Q uniformly distributed with respect to Haar measure by the method of Stewart [33], and a random upper triangular matrix R with specified smallest singular value, and finally formed

$$A = U \cdot \hat{\Sigma}_1 \cdot R \cdot Q^{\mathrm{T}} \text{ and } B = V \cdot \hat{\Sigma}_2 \cdot R \cdot Q^{\mathrm{T}}. \tag{24}$$

Hence the GSV pairs of A and B are known to be (α_i, β_i). In this way we can generate random test matrices having any distribution of generalized singular value pairs.

We note that if the test matrices are given by (24), then

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

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	$1 - \frac{i-1}{n-1}(1 - \frac{1}{\mathrm{cond}}), 1$	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	$1, \mathrm{cond}^{-(i-1)/(n-1)}$	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 2: Average double sweeps and accuracy of computed GSV pairs

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 the direct GSVD algorithm, then the quantity

$$\Delta_1 \equiv \left\{ \sum_{i=1}^{n} [(\alpha_i - \bar{\alpha}_i)^2 + (\beta_i - \bar{\beta}_i)^2] \right\}^{1/2} \cdot \sigma_{\min}(R) \tag{25}$$

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

We designed six different distributions of GSV as illustrated in second column of Table 2, 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 (24). 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 random 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, 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 generalized singular values 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 random matrices, in each class of dimension 10 we generated 201 random matrices, in each class of dimension 20 we generated 101 random matrices, and in each class of dimension 40 we generated 51 random matrices, for a total of 10,772 different test matrices.

Table 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 (6) and (7) are performed using LINPACK QR decomposition with or without the column pivoting subroutine DQRDC [12]. In all test examples, the backward stability conditions (21) and (22) 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 $O(\|E\|, \|F\|) = O(10^{-14})$.

The third column in Table 3 is for different conditioned GSVD problems. 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. 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 (25). 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. The following table illustrates the required time for a 50 by 50 matrix pair A and B with 5 double sweeps to satisfy the stopping criterion $tol = 10^{-14}$.

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

Appendix

The Singular Value Decomposition of 2×2 Triangular Matrix - by Demmel and Kahan

In this appendix, for the convenience of the reader, we include Demmel and Kahan's 2 by 2 triangular SVD algorithm. The algorithm was used in their high relative accuracy bidiagonal SVD algorithm [9], but the algorithm details were not presented there.

It is known that the singular values of the 2 by 2 upper triangular matrix $\begin{bmatrix} f & g \\ 0 & h \end{bmatrix}$ 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 the rows of $(f^2 - \gamma_1^2, fg)$. 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.

```
      SUBROUTINE SLASV2( F, G, H, SSMIN, SSMAX, SNR, CSR, SNL, CSL )
      REAL       CSL, CSR, F, G, H, SNL, SNR, SSMAX, SSMIN
*
*  Computes singular value decomposition of 2 by 2 triangular matrix:
*    [ CSL  SNL ] . [ F G ] . [ CSR -SNR ] = [ SSMAX   0  ]
*    [-SNL  CSL ]   [ 0 H ]   [ SNR  CSR ]   [  0    SSMIN ]
*  Absolute value of SSMAX is larger singular value, Absolute value of
*  SSMIN is smaller singular value. Both CSR**2 + SNR**2 = 1 and
*  CSL**2 + SNL**2 = 1.
*
*  .. Parameters ..
      REAL       ZERO, HALF, ONE, TWO, FOUR
      PARAMETER  ( ZERO = 0.0, HALF = 0.5, ONE = 1.0, TWO = 2.0, FOUR = 4.0 )
*  .. Local Scalars ..
      LOGICAL    GASMAL, SWAP
      INTEGER    PMAX
      REAL       A, CLT, CRT, D, FA, FT, GA, GT, HA, HT, L, M,
     $           MM, R, S, SLT, SRT, T, TEMP, TSIGN, TT
*  .. Intrinsic Functions ..
      INTRINSIC  ABS, SIGN, SQRT
*
      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( ONE+( FA / GA ).EQ.ONE ) THEN   /* Case of very large GA */
            GASMAL = .FALSE.
            SSMAX = GA
            IF( HA.GT.ONE ) THEN
              SSMIN = FA / ( GA / HA )
            ELSE
              SSMIN = ( FA / GA )*HA
            END IF
            CLT = ONE
            SLT = HT / GT
            SRT = ONE
            CRT = FT / GT
          END IF
        END IF
        IF( GASMAL ) THEN     /* Normal case */
          D = FA - HA
          IF( D.EQ.FA ) THEN  /* Copes with infinite F or H */
            L = ONE
          ELSE
            L = D / FA
          END IF            /* Note that 0 .le. L .le. 1 */
          M = GT / FT         /* Note that abs(M) .le. 1/macheps */
          T = TWO - L         /* Note that T .ge. 1 */
          MM = M*M
          TT = T*T
          S = SQRT( TT+MM )   /* Note that 1 .le. S .le. 1 + 1/macheps */
          IF( L.EQ.ZERO ) THEN
            R = ABS( M )
          ELSE
            R = SQRT( L*L+MM )
          END IF              /* Note that 0 .le. R .le. 1 + 1/macheps */
          A = HALF*( S+R )    /* Note that 1 .le. A .le. 1 + abs(M) */
          SSMIN = HA / A
          SSMAX = FA*A
          IF( MM.EQ.ZERO ) THEN     /* Note that M is very tiny */
            IF( L.EQ.ZERO ) THEN
              T = SIGN( TWO, FT )*SIGN( ONE, GT )
            ELSE
              T = GT / SIGN( D, FT ) + M / T
            END IF
```

```
      ELSE
        T = ( M / ( S+T )+M / ( R+L ) )*( ONE+A )
      END IF
      L = SQRT( T*T+FOUR )
      CRT = TWO / L
      SRT = T / L
      CLT = ( CRT+SRT*M ) / A
      SLT = ( HT / FT )*SRT / A
    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
  /* Correct 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
```

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. and Its Appl., 104, pp.131-140(1988).

[3] Z. Bai, Numerical treatment of restricted Gauss-Markov linear model, Comm. in Statis. B17 No.2, pp.131-140(1988).

[4] Z. Bai, An improved algorithm for computing the generalized singular value decomposition, To appear in Numer. Math.

[5] A. W. Bojanczyk, M. Ewerbring, F. T. Luk and P. van Dooren, An accurate product SVD algorithm, preprint MCS-P171-0890, Argonne National Lab. 1990.

[6] J. P. Charlier and P. Van Dooren, On Kogbetliantz's SVD algorithm in the presence of Clusters, Lin. Alg. and Its Appl. 95, pp136-160(1987).

[7] B. L. R. De Moor and G. H. Golub, Generalized Singular Value Decompositions: A proposal for a standardized nomenclature, Manuscript NA-89-05, Numerical Analysis Project, Stanford University, Stanford, Calif., 1989

[8] 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.

[9] J. Demmel and W. Kahan, Accurate Singular Values of Bidiagonal Matrices, SIAM J. Sci. Stat. Comput. 11, pp873-912 (1990)

[10] J. Demmel and A. Mckenney, LAPACK Working Notes # 9: A test matrix generation suite, MCSD, Argonne National Lab. 1989.

[11] J. Demmel and K. Veselić, Jacobi's method is more accurate than QR, to appear in SIAM J. Mat. Anal. Appl.

[12] J. J. Dongarra, J. R. Bunch, C. B. Moler and G. W. Stewart, LINPACK User's Guide, Society for Industrial and Applied Mathematics, Philadelphia, 1978.

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

[14] K. V. Fernando, On equivalence and convergence of Jacobi and Kogbetliantz methods with odd-even orderings, NAG Tech. Rep. TR1/88, Numerical Algorithm Group Ltd, Oxford, 1988

[15] K. V. Fernando, Linear convergence of the row cyclic Jacobi and Kogbetliantz methods, Numer. Math. 56, 73-91, 1989.

[16] 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 at eds. SIAM Philadelphia, Penn. 1988, pp.128-140.

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

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

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

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

[21] 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, pp.1147-1159(1986).

[22] 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, pp294-315(1989).

[23] B. Kagstrom, The generalized singular value decomposition and $(A - \lambda B)$-problem. BIT 24, pp.568-583(1984)

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

[25] F. T. Luk, A parallel method for computing the generalized singular value decomposition, J. Para. Dist. Comp. 2, pp.250-260(1985)

[26] F. T. Luk and H. T. Park, On parallel Jacobi orderings, Tech. Rep. EE-CEG-86-5, School of Electrical Engineering, Cornell Univ. Ithaca, 1986.

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

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

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

[30] C. C. Paige, Computing the generalized singular value decomposition, SIAM J. Sci. Stat. Comput. 7(1986), pp.1126-1146.

[31] 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. and Appl. 77, pp.301-313(1986).

[32] 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).

[33] G. W. Stewart. On efficient generation of random orthogonal matrices with an application to condition estimation. *SIAM J. Numer. Anal.*, 17, pp.403-409(1980).

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

[35] Sun Ji-guang, Perturbation analysis for the generalized singular value problem, SIAM J. Numer. Anal. 20, pp.611-625(1983).

[36] C. F. Van Loan, Generalizing the singular value decomposition, SIAM J. Numer. Anal. 13, pp.76-83(1976).

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

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

[39] H. Zha, A numerical algorithm for computing restricted singular value decomposition of matrix triplets, to appear in Lin. Alg. & Appl. 1991.