SVD AND SIGNAL PROCESSING, II
Algorithms, Analysis and Applications
R.J. Vaccaro (Editor)
© Elsevier Science Publishers B.V., 1991

An Accurate Product SVD Algorithm

Adam W. Bojanczyk[1], Magnus Ewerbring[2],
Franklin T. Luk[1], Paul Van Dooren[3]

[1]Cornell University
School of Electrical Engineering
Ithaca, NY 14853, USA

[2]Argonne National Laboratory
Mathematics and Computer Science Division
Argonne, IL 60439, USA

[3]Philips Research Laboratory
Avenue Albert Einstein 4
B-1348 Louvain-la-Neuve, Belgium

Abstract

In this paper, we propose a new algorithm for computing a singular value decomposition of a product of three matrices. We show that our algorithm is numerically desirable in that all relevant residual elements will be numerically small.

1. Introduction

The problem of computing the singular value decomposition (SVD) of a product of matrices occurs in many applications, e.g., weighted least squares, canonical correlations, linear prediction, and balanced realization (cf. Ewerbring and Luk [4], Fernando and Hammarling [7], and Heath, Laub, Paige and Ward [8]). In [4] Ewerbring and Luk proposed to perform the computation in two steps. The first involves a reduction of all matrices to upper triangular forms, and the second, an SVD of a product of three matrices. It is of utmost importance that the three matrices be kept triangular, so that an efficient, implicit Jacobi-SVD method can be used. In addition, the method is easily amenable to parallel computing (Ewerbring and Luk [5]). There are many ways to preserve the triangular property of the individual matrices. Ewerbring [3] compared various approaches and showed why each has its own strengths and weaknesses. We propose here a new algorithm for the product SVD problem, and we prove that our algorithm is numerically accurate in that all the relevant residual elements will be numerically small.

This paper is organized as follows. In Section 2 we describe a generalization of the SVD to explain how the product SVD problem may arise, and in Section 3 we present our new algorithm. We give a criterion for numerical stability in Section 4 and a detailed error analysis in Section 5. Finally, we discuss a few numerical examples in Section 6.

2. HK Singular Value Decomposition

Van Loan [12] first generalized the SVD. Recently, there has been much interest in further generalizations; see, e.g., De Moor and Golub [2] and Ewerbring and Luk [4]. In this section, we present the details of one such generalization to explain how the product SVD problem may arise. We call our generalization the HK singular value decomposition (HK-SVD) [3], and it concerns the simultaneous diagonalization of three matrices. Given three real matrices A $(n \times p)$, H $(n \times n)$, and K $(p \times p)$, where H and K are symmetric and positive semi-definite and

$$\operatorname{rank}(H) = r, \quad \operatorname{rank}(K) = s, \quad \text{and} \quad r \geq s,$$

our aim is to find an $n \times r$ transformation Y and a $p \times s$ transformation Z, such that

$$\begin{pmatrix} Y & 0 \\ 0 & Z \end{pmatrix}^T \begin{pmatrix} H & A \\ A^T & K \end{pmatrix} \begin{pmatrix} Y & 0 \\ 0 & Z \end{pmatrix} = \begin{pmatrix} I_r & D \\ D^T & I_s \end{pmatrix}, \tag{2.1}$$

where the matrix D is diagonal. When $H = I_n$ and $K = I_p$, we get the familiar singular value decomposition of the matrix A. One possible application when $p = s$, i.e., when K is nonsingular, is weighted least squares:

$$\| A\vec{x} - \vec{b} \|_{H^+} = \min \quad \text{s.t.} \quad \| \vec{x} \|_K = \min .$$

The problem simplifies to

$$\| D\vec{y} - \vec{d} \|_2 = \min \quad \text{s.t.} \quad \| \vec{y} \|_2 = \min ,$$

where $\vec{d} = Y^T\vec{b}$, which can be easily solved by standard means. The solution vector $\vec{x}$ is then given by $\vec{x} = Z\vec{y}$.

To compute the HK-SVD, we start by reducing the two matrices H and K. Since they are both symmetric and semi-definite, we can find their square roots as upper trapezoidal matrices, viz., $H^{1/2}$ $(r \times n)$ and $K^{1/2}$ $(s \times p)$, respectively, satisfying

$$H = (H^{1/2})^T H^{1/2} \quad \text{and} \quad K = (K^{1/2})^T K^{1/2} .$$

Using pseudo-inverses, we simplify the two-by-two block matrix to one with identity matrices in the diagonal positions:

$$X^T \begin{pmatrix} H & A \\ A^T & K \end{pmatrix} X = \begin{pmatrix} I_r & B \\ B^T & I_s \end{pmatrix}, \tag{2.2}$$

where

$$X = \begin{pmatrix} (H^{1/2})^+ & 0 \\ 0 & (K^{1/2})^+ \end{pmatrix}$$

and

$$B = ((H^{1/2})^+)^T A (K^{1/2})^+ . \tag{2.3}$$

Next, we aim to diagonalize the $r \times s$ matrix B without disturbing the diagonal identity blocks, a feat accomplished by an SVD of B, i.e.,

$$B = U\Sigma V^T ,$$

where U $(r \times r)$ and V $(s \times s)$ are orthogonal and Σ $(r \times s)$ is diagonal. The desired transformations Y and Z for the HK-SVD are given in product form by

$$Y = (H^{1/2})^+ U \quad \text{and} \quad Z = (K^{1/2})^+ V . \tag{2.4}$$

Hence the given problem simplifies to an SVD of a product of three matrices. Details are given in [6] on how the matrix product B can be reduced to one where all three factors have equal dimensions (here $s \times s$) and where the pseudo-inverses are replaced by inverses. Our job then is to find an SVD of C, where

$$C = E^{-1} F G^{-1} , \tag{2.5}$$

and E, F, and G are all $s \times s$ and upper triangular. For obvious numerical reasons, we wish to avoid finding $s \times s$ inverses and forming $s \times s$ matrix products. The trick is to utilize a Jacobi-SVD method that has been developed for triangular matrices [9], [11]. Then we need to work with only 2×2 submatrices. By applying the transformations and data permutations in some special order, viz., the so-called outer rotations and odd-even ordering [9], we can guarantee convergence of the overall algorithm [10].

To be specific, let $\hat{E}$, $\hat{F}$, and $\hat{G}$ denote the three 2×2 submatrices extracted from the ith and $(i+1)$st rows and columns of the matrices E, F, and G, respectively. Let $\hat{C}$ denote the corresponding submatrix of C. Since the matrices are upper triangular, it follows that $\hat{C}$ can be found directly as

$$\hat{C} = \hat{E}^{-1} \hat{F} \hat{G}^{-1} . \tag{2.6}$$

For the purpose of finding rotations to diagonalize $\hat{C}$ we can further simplify (2.6) by replacing each inverse by an adjoint (abbrev. adj), an approach advocated in [11] for the generalized singular value decomposition (GSVD). For example, if

$$\hat{E} = \begin{pmatrix} \alpha & \beta \\ 0 & \gamma \end{pmatrix} ,$$

then

$$\mathrm{adj}(E) = \begin{pmatrix} \gamma & -\beta \\ 0 & \alpha \end{pmatrix} .$$

We then find rotations to diagonalize the matrix product:

$$\hat{C} = \mathrm{adj}(\hat{E})\hat{F}\,\mathrm{adj}(\hat{G}) .$$

Hence from here on, we can consider the SVD problem for a product of three 2×2 upper triangular matrices without any need to include matrix inverses

3. New Algorithm

In this section, we propose a new algorithm for the product SVD problem. Our algorithm can be extended to a product of a larger number of matrices. Suppose that the three given upper triangular matrices are

$$A_1 = \begin{pmatrix} a_1 & b_1 \\ 0 & d_1 \end{pmatrix} ,$$

$$A_2 = \begin{pmatrix} a_2 & b_2 \\ 0 & d_2 \end{pmatrix} ,$$

$$A_3 = \begin{pmatrix} a_3 & b_3 \\ 0 & d_3 \end{pmatrix} .$$

We call the product A:

$$A = A_1 A_2 A_3 ,$$

and let

$$A = \begin{pmatrix} a & b \\ 0 & d \end{pmatrix} .$$

Our objective is to find four orthogonal matrices Q_1, Q_2, Q_3, Q_4 such that

$$A' = Q_1 A Q_4^T = \begin{pmatrix} a' & 0 \\ 0 & d' \end{pmatrix} \tag{3.1}$$

and

$$A'_i = Q_i A_i Q_{i+1}^T = \begin{pmatrix} a'_i & b'_i \\ 0 & d'_i \end{pmatrix} , \tag{3.2}$$

for $i = 1, 2, 3$. The two equations (3.1) and (3.2) imply that

$$A' = A'_1 A'_2 A'_3 .$$

In words, we would like to find four transformations Q_1, Q_2, Q_3 and Q_4 to zero out five elements, namely, the off-diagonal elements of A and the sub-diagonal elements of A_1, A_2 and A_3. The extra requirement, although mathematically feasible, may cause numerical difficulty if not treated with care. The goal of this paper is to develop an algorithm so that properties (3.1) and (3.2) will be satisfied except for very small numerical errors.

Our tool for the computation is a transformation discussed by Charlier, Vanbegin, and Van Dooren [1]:

$$Q = \begin{pmatrix} s & c \\ -c & s \end{pmatrix} , \tag{3.3}$$

where $c^2 + s^2 = 1$. We may regard the transformation as a permuted reflection:

$$Q = \begin{pmatrix} c & s \\ s & -c \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} .$$

While each transformation Q_i is defined by the cosine sine pair

$$c_i = \cos\theta_i \quad \text{and} \quad s_i = \sin\theta_i ,$$

we also associate Q_i with the tangent

$$t_i = \tan\theta_i .$$

Given t_i, we can easily recover c_i and s_i using the relations

$$c_i = \frac{1}{\sqrt{1 + t_i^2}} \quad \text{and} \quad s_i = t_i c_i . \tag{3.4}$$

In general, consider the result of applying the left and right transformations Q_l and Q_r to a 2×2 upper triangular matrix A:

$$A' = Q_l A Q_r^T = \begin{pmatrix} a' & b' \\ c' & d' \end{pmatrix} = \begin{pmatrix} s_l & c_l \\ -c_l & s_l \end{pmatrix} \begin{pmatrix} a & b \\ 0 & d \end{pmatrix} \begin{pmatrix} s_r & c_r \\ -c_r & s_r \end{pmatrix}^T . \tag{3.5}$$

We can derive from (3.5) these two relations:

$$c' = c_l c_r (-a t_r + d t_l - b) , \tag{3.6a}$$

$$b' = c_l c_r (-a t_l + d t_r + b t_l t_r) , \tag{3.6b}$$

where $t_l = \tan\theta_l$ and $t_r = \tan\theta_r$. The postulates that both c' and b' be zeros define two conditions on t_l and t_r so that they can be determined explicitly, whereas the postulate that c' be zero defines a condition that relates θ_l to θ_r so that one can be computed if the other is known.

For the case of exposition, assume for now that $abd \neq 0$; this condition will be removed in Section 5.2. This assumption implies that $c_l c_r \neq 0$, and so the postulate that $c' = 0$ in (3.6a) becomes

$$-a t_r + d t_l - b = 0 . \tag{3.6c}$$

For the SVD problem, both c' and b' are zeros, and we can use (3.6c) to reduce (3.6b) to an equation either in t_l:

$$b' = c_l c_r \left(\frac{bd}{a}\right)\left(t_l^2 + t_l \sigma_l - 1\right) , \tag{3.7a}$$

where

$$\sigma_l = \frac{1}{2d}\left(\frac{d^2 - a^2}{b} - b\right),$$

or in t_r:

$$b' = c_l c_r \left(\frac{ab}{d}\right)\left(t_r^2 + t_r\sigma_r - 1\right), \tag{3.7b}$$

where

$$\sigma_r = \frac{1}{2a}\left(\frac{d^2 - a^2}{b} + b\right).$$

From (3.7a) we get the quadratic equation by setting b' to zero:

$$t_l^2 + 2\sigma_l t_l - 1 = 0, \tag{3.7c}$$

and from (3.7b) we get

$$t_r^2 + 2\sigma_r t_r - 1 = 0. \tag{3.7d}$$

We propose to solve the two equations (3.7c) and (3.7d) using these formulas:

$$r = \frac{(d-a)(d+a)}{b}, \tag{3.8a}$$

$$\sigma_l = \frac{r-b}{2d}, \tag{3.8b}$$

$$\sigma_r = \frac{r+b}{2a}, \tag{3.8c}$$

$$t_l = \frac{1}{\sigma_l + \operatorname{sign}(\sigma_l)\sqrt{\sigma_l^2 + 1}}, \tag{3.8d}$$

$$t_r = \frac{1}{\sigma_r + \operatorname{sign}(\sigma_r)\sqrt{\sigma_r^2 + 1}}. \tag{3.8e}$$

In finite-precision arithmetic, either one of σ_l and σ_r can be computed with a higher relative precision. In particular, if

$$\operatorname{sign}(r) = -\operatorname{sign}(b), \tag{3.9a}$$

then (3.8b) will produce a very accurate σ_l, whereas if

$$\operatorname{sign}(r) = \operatorname{sign}(b), \tag{3.9b}$$

then (3.8c) will produce a very precise σ_r. If $r = 0$, then both t_l and t_r will be computed with the same relative accuracy. So, let $r \neq 0$. Note that since

$$r = (d^2 - a^2)/b,$$

the condition (3.9a) is equivalent to the inequality:

$$|a| > |d|, \tag{3.10a}$$

and (3.9b) to

$$|a| < |d|. \tag{3.10b}$$

Now, (3.10a) implies that

$$|\sigma_l| > |\sigma_r|,$$

and so from (3.8d) and (3.8e) that

$$|t_l| < |t_r|.$$

Similarly, the condition (3.9b) will lead to

$$|t_l| > |t_r|.$$

We summarize the results in the next lemma.

Lemma 3.1. Let $abdr \neq 0$. If $|a| > |d|$, then $|t_l| < |t_r|$. Conversely, if $|a| < |d|$, then $|t_l| > |t_r|$. □

Thus, our algorithm will always choose the smaller angular rotation. To summarize, we do a two-stage computation. In the first stage, we calculate the product A explicitly:

$$a = a_1 a_2 a_3, \tag{3.11a}$$

$$b = a_1 a_2 b_3 + a_1 b_2 d_3 + b_1 d_2 d_3, \tag{3.11b}$$

$$d = d_1 d_2 d_3. \tag{3.11c}$$

We use (3.8a) to calculate r, and then compute either σ_l or σ_r, depending on the signs of r and b. Hence we obtain either Q_1 or Q_4. In the second stage we use the relation (3.6e) to compute the remaining transformations. Suppose that t_1 is known. Then t_2, t_3, t_4 are generated by the forward substitution

$$t_{i+1} = \frac{d_i t_i - b_i}{a_i}. \tag{3.12a}$$

On the other hand, if t_4 is known, then t_3, t_2, t_1 are generated by the backward substitution

$$t_i = \frac{a_i t_{i+1} + b_i}{d_i}. \tag{3.12b}$$

4. Criterion for Numerical Stability

In this paper, unless otherwise stated, we use the vector and matrix 2-norms:

$$\|\cdot\| = \|\cdot\|_2.$$

We also assume, without loss of generality, that

$$\|A_1\| = \|A_2\| = \|A_3\| = 1.$$

Recall that A_1', A_2', A_3', and A' denote the four matrices A_1, A_2, A_3, and A, respectively, after the equivalence transformations as defined in (3.1) and (3.2) have been performed. Let ϵ denote the relative precision of the floating-point arithmetic, and let $\bar{A}_1'$, $\bar{A}_2'$, $\bar{A}_3'$, and $\bar{A}'$ represent the computed A_1', A_2', A_3', and A', respectively. We wish A' to be a diagonal matrix:

$$A' = A_1'A_2'A_3' = \begin{pmatrix} a' & 0 \\ 0 & d' \end{pmatrix} . \tag{4.1a}$$

Assume that, given the exact upper triangular matrices A_i', $i = 1,2,3$, we wish to compute in floating-point arithmetic the product

$$\bar{A}' := \mathrm{fl}(\prod A_i') . \tag{4.1b}$$

Because of rounding errors, we can hope only for

$$\bar{A}' = \begin{pmatrix} \bar{a}' & \bar{b}' \\ 0 & \bar{d}' \end{pmatrix} , \tag{4.2a}$$

where $\bar{b}'$ satisfies the relation:

$$|\bar{b}'| = O(\epsilon \prod_{i=1}^{3} \| A_i \|) . \tag{4.2b}$$

This is the case even when $\| A_1'A_2'A_3' \| \ll 1$. Thus, the relative error in $\bar{A}'$ may be very large. A more desirable result would be to get

$$\| \bar{A}_i' - A_i' \| = O(\epsilon) , \tag{4.3a}$$

and the following relation for $\bar{b}'$:

$$|\bar{b}'| = O(\epsilon \; \| A_1'A_2'A_3' \|) . \tag{4.3b}$$

However, it is difficult to satisfy (4.3a) and (4.3b), unless the element b of A can be computed with a high relative accuracy. This seems to be difficult to achieve, as all operations are performed in the same floating-point arithmetic, and hence the computed b may suffer from cancellations. Hence (4.2b) defines the maximal relative numerical accuracy feasible for the implicit product SVD problem. We shall show in the next section, viz., Theorem 5.1, that condition (4.2b) will be satisfied.

In addition to (4.2b), we also wish to preserve the triangularity of the individual matrices A_i':

$$A_i' = \begin{pmatrix} a_i' & b_i' \\ 0 & d_i' \end{pmatrix} ,$$

for $i = 1,2,3$. Now, assume that orthogonal transformations Q_i and Q_{i+1} satisfying (3.2) exactly are given, and that we compute the product $Q_iA_iQ_{i+1}^T$ using finite precision arithmetic. Our best hope is that

$$\bar{A}_i' = \begin{pmatrix} \bar{a}_i' & \bar{b}_i' \\ \bar{c}_i' & \bar{d}_i' \end{pmatrix} , \tag{4.4a}$$

with

$$|\bar{c}_i'| = O(\epsilon \; \| A_i \|) , \tag{4.4b}$$

for $i = 1,2,3$. We shall show in the next section, viz., Theorem 5.2, that this relation will hold.

To summarize, we shall prove that, using our new product SVD algorithm, the four computed matrices $\bar{A}_1'$, $\bar{A}_2'$, $\bar{A}_3'$, and $\bar{A}'$ will satisfy conditions (4.2b) and (4.4b), which provide the maximal numerical accuracy that is feasible in the finite precision computation.

5. Backward Error Analysis

In this section, we present a backward error analysis of our computation. We assume that our initial parameters are perturbed, and use the "bar" symbol. For example, instead of initial values a, b and d, we have the perturbed values $\bar{a}$, $\bar{b}$ and $\bar{d}$. We assume further that exact arithmetic will be performed by using these perturbed initial values. We use the "tilde" symbol for the exact values based on the perturbed data. For example, $\tilde{r}$ will denote the exact result using formula (3.8a) for the perturbed data $\bar{a}$, $\bar{b}$ and $\bar{d}$.

The symbol fl(a) will be used to denote the computed result of the parameter a. In our error analysis, we shall adopt a convention that involves a liberal use of Greek letters. For example, by α we mean a relative perturbation of an absolute magnitude not greater than ϵ, where ϵ denotes the machine precision. All terms of order ϵ^2 or higher will be ignored.

We start our procedure by computing elements of the product matrix A. For the elements of the computed product matrix A we have

$$\bar{a} := \mathrm{fl}(a) = a_1a_2a_3(1 + 2\alpha_1) , \tag{5.1a}$$

$$\bar{d} := \mathrm{fl}(d) = d_1d_2d_3(1 + 2\delta_1) , \tag{5.1b}$$

$$\bar{b} := \mathrm{fl}(b) = a_1a_2b_3(1 + 4\beta_1) + a_1b_2d_3(1 + 4\beta_2) + b_1d_2d_3(1 + 3\beta_3) , \tag{5.1c}$$

where, according to our convention, the parameters α_1, δ_1, β_1, β_2, and β_3 are all quantities whose absolute values are bounded by ϵ. Our analysis is divided into two parts. In Section 5.1, we consider a regular case where all elements of the computed product matrix are numerically significant with respect to the maximal in magnitude element, i.e.,

$$\min(|\bar{a}|, |\bar{b}|, |\bar{d}|) > \epsilon \max(|\bar{a}|, |\bar{b}|, |\bar{d}|) . \tag{5.2}$$

In Section 5.2, we consider special cases where at least one element of the computed A is numerically insignificant.

5.1. Regular Case

In this subsection, we assume that $rb < 0$, i.e., sign(r) = −sign(b). First, we show that equation (3.7c) will be solved very accurately. Conversely, if $rb \geq 0$, then we can prove in a similar fashion that equation (3.7d) can be solved very accurately.

Lemma 5.1. Let $\hat{t}_1$ and $\bar{t}_1$ be the exact and computed solutions, respectively, of equation (3.7c) with data $\tilde{a}, \tilde{b}, \tilde{d}$. Moreover, let $\hat{c}_1, \hat{s}_1$ and $\bar{c}_1, \bar{s}_1$ be the exact and the computed cosines and sines using (3.4) with the tangent value $\bar{t}_1$. Then

$$\bar{t}_1 = \hat{t}_1(1 + 10\epsilon_5)\,, \tag{5.3a}$$

$$\bar{c}_1 = \hat{c}_1(1 + 3\mu_1)\,, \tag{5.3b}$$

$$s_1 = \hat{s}_1(1 + 3\mu_1)(1 + \nu_1)\,, \tag{5.3c}$$

where $|\epsilon_5| < \epsilon$, $|\mu_1| < \epsilon$, and $|\nu_1| < \epsilon$.

Proof. Let $\hat{r}, \hat{\sigma}_1$ be exact values using (3.8a) and (3.8b) with data $\tilde{a}, \tilde{b}$, and $\tilde{d}$. For the computed values of $\hat{r}, \hat{\sigma}_1, \hat{t}_1$, we get

$$\bar{r} := \mathrm{fl}(\hat{r}) = \left(\frac{(\tilde{d} - \tilde{a})(\tilde{d} + \tilde{a})}{\tilde{b}}\right)(1 + 4\epsilon_1) = \hat{r}(1 + 4\epsilon_1)\,,$$

$$\bar{\sigma}_1 := \mathrm{fl}(\hat{\sigma}_1) = \left(\frac{\bar{r} - \tilde{b}}{2\tilde{d}}\right)(1 + 2\epsilon_2) = \left(\frac{\hat{r} - \tilde{b}}{2\tilde{d}}\right)(1 + 6\epsilon_3) = \hat{\sigma}_1(1 + 6\epsilon_3)\,,$$

$$\bar{t}_1 := \mathrm{fl}(\hat{t}_1) = \left(\frac{1}{\bar{\sigma}_1 + \mathrm{sign}(\bar{\sigma}_1)\sqrt{\bar{\sigma}_1^2 + 1}}\right)(1 + 4\epsilon_1) = \hat{t}_1(1 + 10\epsilon_5)\,.$$

Similarly, we can use the formulas in (3.4) to prove relations (5.3b) and (5.3c) for $\bar{c}_1$ and $\bar{s}_1$. □

In words, Lemma 5.1 states that the procedure (3.8a) (3.8e) for solving (3.7c) is numerically stable in the forward sense. Note that, due to the way they are defined, the parameters $\bar{t}_1$, $\bar{c}_1$ and $\bar{s}_1$ may not satisfy (3.4). Three lemmas follow, leading to our main result of Theorem 5.1.

Lemma 5.2. Let $\hat{\sigma}_1$ and $\hat{t}_1$ be exact values corresponding to given data $\tilde{a}$, $\tilde{b}$ and $\tilde{d}$, and let $t_1 := \mathrm{fl}(\hat{t}_1)$. Define a residual r_1 by

$$r_1 := \frac{\tilde{b}\tilde{d}}{\tilde{a}}(\bar{t}_1^2 + 2\bar{\sigma}_1\bar{t}_1 - 1)\,. \tag{5.4}$$

Then

$$|r_1| \le K_1\epsilon \prod_{i=1}^{3} \| A_i \|\,, \tag{5.5}$$

where K_1 is a positive constant.

Proof. Substitute $\bar{t}_1$ into equation (3.7c) and define a corresponding residual element:

$$p_1 := \bar{t}_1^2 + 2\bar{\sigma}_1\bar{t}_1 - 1\,.$$

Using Lemma 5.1, we get

$$p_1 = 20\epsilon_5\hat{t}_1 + 20\epsilon_5\hat{t}_1\hat{\sigma}_1 = 30\epsilon_6\,, \tag{5.6}$$

since from (3.8d) we have $|\hat{t}_1| \le 1$ and $|\hat{t}_1\hat{\sigma}_1| \le \frac{1}{2}$. The desired residual is given by

$$r_1 = \left(\frac{\tilde{b}\tilde{d}}{\tilde{a}}\right)p_1\,,$$

and so

$$|r_1| \le 30\epsilon_6\,|\tilde{b}| \tag{5.7}$$

from (5.6) and (3.10a). Finally, we use (5.1c) on $\tilde{b}$ to get

$$|r_1| \le K_1\epsilon \prod_{i=1}^{3} \| A_i \|$$

□

Lemma 5.3. The recurrence (3.12a) yields $\bar{t}_{i+1}$, $i = 1, 2, 3$, such that

$$\tilde{a}_i\bar{t}_{i+1} - \tilde{d}_i\bar{t}_i + \tilde{b}_i = 0\,, \tag{5.8a}$$

with

$$\tilde{a}_i = a_i(1 + 2\psi_i)\,, \tag{5.8b}$$

$$\tilde{b}_i = b_i\,, \tag{5.8c}$$

$$\tilde{d}_i = d_i(1 + \phi_i)\,. \tag{5.8d}$$

Proof. From (3.12a) the computed $\bar{t}_{i+1}$ satisfies the relation

$$\bar{t}_{i+1} := \mathrm{fl}(\hat{t}_{i+1}) = \frac{d_i\bar{t}_i(1 + \phi_i) - b_i}{a_i(1 + 2\psi_i)}\,. \tag{5.9}$$

Rewriting the relation leads to

$$d_i\bar{t}_i(1 + \phi_i) - a_i(1 + 2\psi_i)\bar{t}_{i+1} = b_i\,. \tag{5.10}$$

Defining $\tilde{d}_i := d_i(1 + \phi_i)$, $\tilde{a}_i := a_i(1 + 2\psi_i)$, and $\tilde{b}_i := b_i$, we obtain the desired results.

□

Lemma 5.4. The computed $\bar{t}_1$ and $\bar{t}_4$ satisfy exactly the following relation:

$$\tilde{a}\bar{t}_4 - \tilde{d}\bar{t}_1 + \tilde{b} = 0\,, \tag{5.11}$$

where

$$\tilde{a} := a_1a_2a_3(1 + 2\psi_1)(1 + 2\psi_2)(1 + 2\psi_3)\,, \tag{5.12a}$$

$$\tilde{d} := d_1d_2d_3(1 + \phi_1)(1 + \phi_2)(1 + \phi_3)\,, \tag{5.12b}$$

$$\tilde{b} := a_1a_2b_3(1 + 2\psi_1)(1 + 2\psi_2) + a_1b_2d_3(1 + 2\psi_1)(1 + \phi_3) + b_1d_2d_3(1 + \phi_2)(1 + \phi_3)\,. \tag{5.12c}$$

Proof. This lemma is a direct consequence of the preceding one. □

Theorem 5.1. Suppose that the given tangent values are $\bar{t}_1$ and $\bar{t}_4$. Let $\tilde{c}_1$, $\tilde{s}_1$, $\tilde{c}_4$ and $\tilde{s}_4$ be the corresponding exact cosine and sine values. Also, let $\tilde{e}'$ and $\tilde{b}'$ denote the corresponding exact values of e' and b', respectively; that is,

$$\tilde{e}' := \tilde{c}_1\tilde{c}_4[-\bar{a}\bar{t}_4 + \bar{d}\bar{t}_1 - \bar{b}] , \tag{5.13}$$

$$\tilde{b}' := \tilde{c}_1\tilde{c}_4[-a\bar{t}_1 + \bar{d}\bar{t}_4 + \bar{b}\bar{t}_1\bar{t}_4] \tag{5.14}$$

Then

$$|\tilde{e}'| \leq K_2\epsilon \prod_{i=1}^{3} \| A_i \| , \tag{5.15}$$

$$|\tilde{b}'| \leq K_3\epsilon \prod_{i=1}^{3} \| A_i \| , \tag{5.16}$$

where K_2 and K_3 are some positive constants.

Proof. First, from Lemma 5.4 we get

$$\tilde{e}' = \tilde{c}_1\tilde{c}_4[(-\bar{a}\bar{t}_4 + \bar{d}\bar{t}_1 - \bar{b}) + (\tilde{a}\bar{t}_4 - \tilde{d}\bar{t}_1 + \tilde{b})] = (\tilde{a} - \bar{a})\tilde{c}_1\tilde{s}_4 - (\tilde{d} - \bar{d})\tilde{s}_1\tilde{c}_1 + (\tilde{b} - \bar{b})\tilde{c}_1\tilde{c}_4 .$$

Using (5.1) and (5.12), we prove the inequality

$$|\tilde{e}'| \leq K\epsilon\,(|a| + |d| + |\bar{b}| + |\tilde{b}|) \leq K_2\epsilon \prod_{i=1}^{3} \| A_i \| . \tag{5.17}$$

Second, rewrite (5.4) as

$$r_1 = \frac{1}{\bar{a}}[\bar{d}\bar{b}\bar{t}_1^2 + \bar{t}_1(\bar{d}^2 - \bar{a}^2 - \bar{b}^2) - \bar{d}\bar{b}] = \frac{1}{\bar{a}}[(\bar{d}\bar{t}_1 - \bar{b})(\bar{b}\bar{t}_1 + \bar{d}) - \bar{t}_1\bar{a}^2] . \tag{5.18}$$

From (5.13) we get

$$\frac{1}{\bar{a}}(\bar{d}\bar{t}_1 - \bar{b}) = \bar{t}_4 + \frac{\tilde{e}'}{\tilde{c}_1\tilde{c}_4\bar{a}} . \tag{5.19}$$

Substituting (5.19) into (5.18) and rearranging terms, we get

$$-\bar{a}\bar{t}_1 + \bar{d}\bar{t}_4 + \bar{b}\bar{t}_1\bar{t}_4 = r_1 - \frac{\tilde{e}'(\bar{b}\bar{t}_1 + \bar{d})}{\tilde{c}_1\tilde{c}_4\bar{a}} ,$$

and so

$$\tilde{b}' = \tilde{c}_1\tilde{c}_4 r_1 - \frac{\tilde{e}'(\bar{b}\bar{t}_1 + \bar{d})}{\bar{a}} . \tag{5.20}$$

From (3.8d) we get

$$|\bar{t}_1\bar{\sigma}_1| \leq \frac{1}{2} ,$$

and from (3.8b) we get

$$|\bar{\sigma}_1| = \left|\frac{\bar{r} - \bar{b}}{2\bar{d}}\right| \geq \left|\frac{\bar{b}}{2\bar{d}}\right| .$$

It follows that

$$|\bar{t}_1| \leq \left|\frac{\bar{d}}{\bar{b}}\right| < \left|\frac{\bar{a}}{\bar{b}}\right| , \tag{5.21}$$

since $|\bar{d}| < |\bar{a}|$ from (3.10a). Finally, recall from (5.3) that $\hat{t}_1 = \bar{t}_1(1 + 10\epsilon_5)$, and use (5.20) to get

$$|\tilde{b}'| \leq |r_1| + 2\,|\tilde{e}'| \leq K_3\epsilon \prod_{i=1}^{3} \| A_i \| , \tag{5.22}$$

thus completing the proof □

We now wish to justify setting the element $\hat{e}_i'$ in $\hat{A}_i'$ to zero. What have we done so far? From Lemma 5.3 we get that for $i = 1,2,3$,

$$-\tilde{a}_i\bar{t}_{i+1} + \tilde{d}_i\bar{t}_i - \tilde{b}_i = 0 , \tag{5.23}$$

with

$$a_i = \tilde{a}_i(1 - 2\psi_i) , \; d_i = \tilde{d}_i(1 - \phi_i) . \tag{5.24}$$

Let the cosine and sine pairs $\tilde{c}_i$ and $\tilde{s}_i$ satisfy $\bar{t}_i = \tilde{s}_i/\tilde{c}_i$, for $i = 2,3,4$. From (3.4) we can derive that

$$\hat{c}_i := \mathrm{fl}(\tilde{c}_i) = \tilde{c}_i(1 + 3\mu_i) , \tag{5.25a}$$

$$\hat{s}_i := \mathrm{fl}(\tilde{s}_i) = \tilde{s}_i(1 + 3\mu_i)(1 + \nu_i) . \tag{5.25b}$$

Our next result provides a bound on the element $\hat{e}_i'$, $i = 1,2,3$, defined by the relation

$$\hat{e}_i' := -\hat{c}_i\hat{s}_{i+1}a_i + \hat{s}_i\hat{c}_{i+1}d_i - \hat{c}_i\hat{c}_{i+1}b_i . \tag{5.26}$$

Since in actual computation we simply set $\hat{e}_i'$ equal to zero, we want to justify our action by showing that $|\hat{e}_i'|$ corresponds to relative, elementwise perturbation of A_i' of the order of ϵ.

Theorem 5.2. For $i = 1,2,3$, the matrix $\hat{A}_i'$ is almost upper triangular in that its (2,1) element $\hat{e}_i'$ satisfies the inequality

$$|\hat{e}_i'| \leq 3\,\epsilon\, \| A_i \| . \tag{5.27}$$

Proof. Using (5.25) and (5.26), we get

$$\hat{e}_i' = (1 + 3\mu_i)(1 + 3\mu_{i+1})[-\tilde{c}_i\tilde{s}_{i+1}a_i(1 + \nu_{i+1}) + \tilde{s}_i\tilde{c}_{i+1}d_i(1 + \nu_i) - \tilde{c}_i\tilde{c}_{i+1}b_i] . \tag{5.28}$$

Substituting (5.24) into (5.28), we obtain

$$\hat{e}_i' = (1 + 3\mu_i)(1 + 3\mu_{i+1})[-\tilde{c}_i\tilde{s}_{i+1}\tilde{a}_i(1 + \nu_{i+1})(1 - 2\psi_i) + \tilde{s}_i\tilde{c}_{i+1}\tilde{d}_i(1 + \nu_i)(1 - \phi_i) - \tilde{c}_i\tilde{c}_{i+1}b_i] . \tag{5.29}$$

From (5.23) we find that

$$-\tilde{a}_i\tilde{c}_i\tilde{s}_{i+1} + \tilde{d}_i\tilde{s}_i\tilde{c}_{i+1} - \tilde{c}_i\tilde{c}_{i+1}b_i = 0 .$$

Hence (5.29) simplifies to

$$\tilde{c}_i' = (1 + 3\mu_i)(1 + 3\mu_{i+1})[-\tilde{c}_i\tilde{s}_{i+1}\tilde{a}_i(\nu_{i+1} - 2\psi_i) + \tilde{s}_i\tilde{c}_{i+1}\tilde{d}_i(\nu_i - \phi_i)] .$$

We derive the inequality

$$|\tilde{c}_i'| \leq \left\| (-\tilde{c}_i \quad \tilde{s}_i)\, D \begin{pmatrix} \tilde{s}_{i+1} \\ \tilde{c}_{i+1} \end{pmatrix} \right\| ,$$

where

$$D = \begin{pmatrix} (\nu_{i+1} - 2\psi_i)\tilde{a}_i & 0 \\ 0 & (\nu_i - \phi_i)\tilde{d}_i \end{pmatrix}$$

Hence

$$|\tilde{c}_i'| \leq 3\epsilon \; \| A_i \| ,$$

completing our proof. □

In summary, we have proved two results using backward error analysis. First, the computed matrix product $\tilde{A}'$ is almost diagonal in that inequalities (5.15) and (5.16) both hold. Second, we can safely set each computed matrix $\tilde{A}_i'$, $i = 1, 2, 3$, to a triangular form because (5.27) is valid. As a final note, even though we have assumed that $rb < 0$, we can easily prove similar results for the case where $rb \geq 0$.

5.2. Special Cases

In this subsection, we assume that inequality (5.2) is violated. To be specific, define

$$\gamma := \min(|\tilde{a}|, |\tilde{b}|, |\tilde{d}|), \tag{5.30}$$

$$\Gamma := \max(|\tilde{a}|, |\tilde{b}|, |\tilde{d}|). \tag{5.31}$$

Hence

$$\gamma \leq \epsilon \Gamma, \tag{5.32}$$

i.e., one of the elements of $\tilde{A}$ is numerically insignificant. This situation requires modifications to our algorithm, since the proposed formulas may break down. In particular, we shall not solve a quadratic equation to determine either $\tilde{t}_1$ or $\tilde{t}_4$. Instead, we set one of the two tangents to zero and attempt to compute all the other tangents from the recurrences. We divide the special cases into three groups, one where

$$|\tilde{a}| + |\tilde{d}| \neq 0 \quad \text{and} \quad |\tilde{b}| \neq 0, \tag{5.33}$$

one where

$$|\tilde{a}| + |\tilde{d}| = 0 \quad \text{and} \quad |\tilde{b}| \neq 0, \tag{5.34}$$

and the last where

$$|\tilde{b}| = 0. \tag{5.35}$$

First, assume that (5.33) holds. Hence at least one, but not all, of the following conditions hold:

$$\gamma = \tilde{b}, \quad \gamma = a \quad \text{or} \quad \gamma = \tilde{d} .$$

We shall set $\tilde{t}_1$ to zero if

$$|\tilde{a}| > |\tilde{d}|, \tag{5.36}$$

and set $\tilde{t}_4$ to zero if

$$|\tilde{a}| \leq |\tilde{d}| \tag{5.37}$$

Thus, the sizes of the diagonal elements of A will be compared to decide which one of $\tilde{t}_1$ or $\tilde{t}_4$ should be zeroed. Without loss of generality, assume that (5.36) holds; hence, $\tilde{t}_1$ becomes the reference angle. So, $\tilde{t}_2$, $\tilde{t}_3$, and $\tilde{t}_4$ are computed from recurrence (3.12a), and relation (5.11) will be satisfied. Further, since $\tilde{t}_1 = 0$ it follows that $\tilde{t}_4 = -b/a$. Substituting these values into (5.13) and (5.14), we can verify that Theorem 5.1 holds. Similarly, Theorem 5.2 follows from the fact that equation (5.23) will be satisfied. We note that it is very important to decide which reference angle to choose, even for the case when $\tilde{b}$ is numerically zero. At first, the choice of reference angle may seem arbitrary for a "small" $\tilde{b}$, since either $\tilde{t}_1$ or $\tilde{t}_4$ can be set to zero. However, as will be illustrated in Example 6.1, an unnecessarily large error may occur unless we pay special care.

Second, assume thus that (5.34) holds. Then, at least one of the a_i's equals zero and at least one of the d_j's also equals zero, for $i, j = 1, 2, 3$. A solution is to permute either the rows or the columns, in order to ensure that the transformed product is diagonal and that the data are reordered. Hence for this case, we may set the two extreme tangents $\{\tilde{t}_1, \tilde{t}_4\}$ to $\{0, \infty\}$, resulting in one transformation matrix being the identity and the other a ninety degree rotation. To be specific, consider the case where one or more a_i's equal zero. If $a_1 = 0$, set $\tilde{t}_1 = 0$ and set $\tilde{t}_2 = \tilde{t}_3 = \tilde{t}_4 = \infty$. If $a_1 \neq 0$ and $a_2 = 0$, set $\tilde{t}_1 = 0$, compute $\tilde{t}_2$ from the forward recurrence, and set $\tilde{t}_3 = \tilde{t}_4 = \infty$. The remaining case is when $a_1 a_2 \neq 0$ and $a_3 = 0$. Again set $\tilde{t}_1 = 0$, calculate $\tilde{t}_2$ and $\tilde{t}_3$ via the recurrence scheme, and set $\tilde{t}_4 = \infty$. Note that we may also choose to determine the tangents using the values of the d_j's. In an actual implementation, we may decide to interleave the tests on a_i and d_j so as to minimize work; i.e., if $a_1 \neq 0$, then test to see whether $d_3 \neq 0$ and so forth.

Third, assume that (5.35) holds. We need to account for the fact that we are really solving an $n \times n$ problem. Although the 2×2 subproblem is already numerically diagonal, it is not sufficient to set $\tilde{t}_1 = \tilde{t}_4 = \infty$, which will leave the 2×2 product unchanged. The $n \times n$ data need to be reordered, calling for $\tilde{t}_1 = \tilde{t}_4 = 0$; i.e., the affected rows and columns will be permuted. Unfortunately, while applying the symmetric permutation, the triangular structures of both $\tilde{A}_1$ and $\tilde{A}_3$ are destroyed. Therefore, $\tilde{t}_2$ and $\tilde{t}_3$ are determined from the recurrences.

6. Numerical Examples

In this section, we present a few examples to show why we have paid so much attention to special cases and why we think we have developed a superior numerical scheme. The first example illustrates how a different reference angle can lead to a much larger numerical error.

Example 6.1. Consider the case when $| b |$ is numerically "small". For instance, let

$$A = \begin{pmatrix} a & b \\ 0 & d \end{pmatrix} = \begin{pmatrix} 10^{-10} & -10^{-17} \\ 0 & 1 \end{pmatrix} .$$

The product is numerically diagonal and thus the diagonal elements need only be permuted. On the surface, it seems like either t_1 or $\bar{t}_4$ can be set to zero. Suppose that t_1 is selected as the reference angle. Then the recurrence formula yields

$$t_1 \leftarrow 0 ,$$

$$t_4 = \frac{dt_1 - b}{a} = 10^{-7} .$$

On the other hand, choosing $\bar{t}_4$ as the reference angle results in the answers

$$\bar{t}_4 \leftarrow 0 ,$$

$$\bar{t}_1 = \frac{a\bar{t}_4 + b}{d} = -10^{-17} .$$

Taking into account limited word length, we find that the first scheme gives the updated product

$$\mathrm{fl}\left(\xi_1 \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} A \begin{pmatrix} 10^{-7} & -1 \\ 1 & 10^{-7} \end{pmatrix}\right) = \mathrm{fl}\left(\xi_1 \begin{pmatrix} 1 & 10^{-7} \\ 0 & -10^{-10} + 10^{-21} \end{pmatrix}\right) ,$$

where $\xi_1 = 1/\sqrt{1 + 10^{-14}}$. On the other hand, the second choice yields

$$\mathrm{fl}\left(\xi_2 \begin{pmatrix} -10^{-17} & 1 \\ -1 & -10^{-17} \end{pmatrix} A \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}\right) = \begin{pmatrix} 1 & 10^{-27} \\ 0 & -10^{-10} \end{pmatrix} ,$$

where $\xi_2 = 1/\sqrt{1 + 10^{-34}}$. Hence, if $| a | < | d |$, we should set $\bar{t}_4$ to zero and compute $\bar{t}_1$ from the recurrence formula. □

A seemingly obvious way to solve the product SVD problem is to first calculate $\bar{t}_1$ and t_4 via an SVD of A, and then determine $\bar{t}_2$ and $\bar{t}_3$ to restore the triangularity of $Q_1 A_1$ and $A_3 Q_4^T$. The transformed middle matrix A_2' must be triangular from mathematical relationships. Indeed, we enforce its triangularity by "truncating" its (2,1) element to zero. The next example show how this scheme may give rise to a large "truncation" error.

Example 6.2. Assume that the given data matrices are

$$A_1 = \begin{pmatrix} 0.21131896972656 & 0.75987243652344 \\ 0 & 0.00872802734375 \end{pmatrix} ,$$

$$A_2 = \begin{pmatrix} 0.80964660644531 & 0.45243835449219 \\ 0 & 0.80749511718750 \end{pmatrix} ,$$

and

$$A_3 = \begin{pmatrix} 1 & -1 \\ 0 & 10^{-10} \end{pmatrix} .$$

They generate the matrix product

$$\bar{A} = \begin{pmatrix} 0.17109368671663 & -0.17109368664571 \\ 0 & 0.705 \cdot 10^{-12} \end{pmatrix} .$$

We use equation (3.8d) to calculate $\bar{t}_1$, equation (3.6c) to find t_4 from $\bar{t}_1$, equation (3.12a) to find $\bar{t}_2$ from $\bar{t}_1$, and equation (3.12b) to determine $\bar{t}_3$ from t_4. The updated data matrices become

$$A_1' = \begin{pmatrix} 0.00233850233086 & -0.00840891599370 \\ 0 & 0.78870896200965 \end{pmatrix} ,$$

$$\bar{A}_2' = \begin{pmatrix} 0.70680263726181 & 0.39635737357002 \\ 0.900558 \cdot 10^{-8} & 0.92499011351931 \end{pmatrix} ,$$

$$\bar{A}_3' = \begin{pmatrix} 0.301512 \cdot 10^{-9} & 1.37477296630288 \\ 0 & 0.33166141036119 \end{pmatrix} .$$

Before the $(2,1)$ element of $\bar{A}_2'$ is set to zero, the matrix product is

$$\bar{A}' = \bar{A}_1' \bar{A}_2' \bar{A}_3' = \begin{pmatrix} 0.498 \cdot 10^{-12} & 0.87 \cdot 10^{-18} \\ 0.21 \cdot 10^{-17} & 0.21196301214092 \end{pmatrix} ,$$

which is numerically diagonal. Now, we need to commit an error of $O(10^{-8})$ in truncating $\bar{A}_2'$ to a triangular matrix, say $\hat{A}_2'$. The actual matrix product is thus

$$\bar{A}' = \bar{A}_1' \hat{A}_2' \bar{A}_3' = \begin{pmatrix} 0.498 \cdot 10^{-12} & 0.101108 \cdot 10^{-9} \\ 0 & 0.21196300237621 \end{pmatrix} ,$$

which is not quite diagonal. Hence, the off-diagonal mass of the matrix product has increased substantially as a result of truncation of the middle matrix. □

Finally, we wish to illustrate the need for choosing the correct initial tangent. We shall compare our algorithm, calling it Method Recur, against an approach that always starts the recurrence from t_1, calling it Method Recur-Left.

Example 6.3. We use 8×8 data matrices, all of which are of full rank but are possibly ill-conditioned. The product matrix C to be diagonalized is of the form

$$C = E^{-1} F G^{-1} .$$

Upon convergence, we have found orthogonal transformations, say U and V, such that

$$U^T E^{-1} F G^{-1} V = D ,$$

where D is a diagonal matrix. We choose to compare accuracy based on the error term

$$\| F - EUDV^TG \|_F$$

so that no 8×8 matrix inversion needs to be performed. To justify our promotion of implicit algorithms, we include a third scheme which involves an SVD of the explicitly formed matrix product. We refer to this approach as Method Explicit.

In Table 1, we present results from one set of tests. We use $\kappa(M)$ to denote the condition number of the matrix M with respect to the 2 norm. For each run, we fix

$$E = G \quad \text{and} \quad \| E \|_F = 1 ,$$

and we increase the value of $\kappa(E)$. The middle matrix F stays constant throughout the entire test set:

$$\| F \|_F = 1 \quad \text{and} \quad \kappa(F) = 109 .$$

We ran each algorithm for six sweeps and assumed convergence.

As expected, we find that Method Recur provides more accurate results than Method Explicit. Indeed, it appears that the errors of the former method are proportional to $\kappa(E)$, while those for the latter to $\kappa(C)$. Perhaps surprisingly, Method Recur-Left gives rise to large errors. On close analysis, one finds that the method fails to correctly treat 2×2 problems of the kind portrayed in Example 6.1. In fact, convergence is lost in that some nontrivial numerical quantity is moved between the diagonal and the strictly upper triangular parts of C. For a detailed discussion of this phenomenon, see [3]. □

Table 1. Product SVD: Error Comparison

		$\| F - EUDV^TG \|_F$		
$\kappa(E)$	$\kappa(C)$	Recur	Recur-Left	Explicit
$1.00 \cdot 10^{+2}$	$5.85 \cdot 10^{+01}$	$5.22 \cdot 10^{-15}$	$5.88 \cdot 10^{-12}$	$3.90 \cdot 10^{-14}$
$1.00 \cdot 10^{+4}$	$5.69 \cdot 10^{+08}$	$5.83 \cdot 10^{-13}$	$6.96 \cdot 10^{-08}$	$4.00 \cdot 10^{-10}$
$1.00 \cdot 10^{+6}$	$5.69 \cdot 10^{+12}$	$5.10 \cdot 10^{-11}$	$9.50 \cdot 10^{-04}$	$4.39 \cdot 10^{-06}$
$1.00 \cdot 10^{+8}$	$4.95 \cdot 10^{+16}$	$4.38 \cdot 10^{-09}$	$6.27 \cdot 10^{+00}$	$6.31 \cdot 10^{-02}$

7. Acknowledgements

The authors thank their research sponsors: A. W. Bojanczyk was supported in part by the Army Research Office under grant DAAL03-90-G-0092; L. M. Ewerbring was supported by the Applied Mathematical Sciences Subprogram of the Office of Energy Research, U.S. Department of Energy, under contract W-31-109-Eng-38; and F. T. Luk was supported by the Joint Services Electronics Program monitored by the Air Force Office of Scientific Research.

8. References

[1] J. P. Charlier, M. Vanbegin and P. Van Dooren, "On efficient implementations of Kogbetliantz's algorithm for computing the singular value decomposition," *Numer. Math.*, 52 (1988), pp. 279–300.

[2] 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.

[3] L. M. Ewerbring, "A new generalization of the singular value decomposition: Algorithms and applications," Ph.D. dissertation, School of Electrical Engineering, Cornell University, Ithaca, New York, 1989.

[4] L. M. Ewerbring and F. T. Luk, "Canonical correlations and generalized SVD: Applications and new algorithms," *J. Comput. Applied Math.*, 27 (1989), pp. 37–52.

[5] L. M. Ewerbring and F. T. Luk, "Computing the singular value decomposition on the Connection Machine," *IEEE Trans. Computers*, 39 (1990), 152–155.

[6] L. M. Ewerbring and F. T. Luk, "The HK singular value decomposition of rank-deficient matrix triplets," Report MCS-P125-0190, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, Illinois, 1990.

[7] K. V. Fernando and S. J. Hammarling, "A product induced singular value decomposition for two matrices and balanced realisation," in *Linear Algebra in Signals, Systems and Control*, B. N. Datta et al., Eds., SIAM, Philadelphia, Penn., 1988, pp. 128–140.

[8] M. T. Heath, A. J. Laub, C. C. Paige, and R. C. Ward, "Computing the SVD of a product of two matrices," *SIAM J. Sci. Statist. Comput.*, 7 (1986), pp. 1147–1159.

[9] F. T. Luk, "A triangular processor array for computing singular values," *Lin. Alg. Applics.*, 77 (1986), pp. 259–273.

[10] F. T. Luk and H. Park, "A proof of convergence for two parallel Jacobi SVD algorithms," *IEEE Trans. Computers*, 38 (1989), pp. 806–811.

[11] C. C. Paige, "Computing the generalized singular value decomposition," *SIAM J. Sci. Statist. Comput.*, 7 (1986), pp. 1126–1146.

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