On propagating orthogonal transformations in a product of 2×2 triangular matrices

Adam Bojanczyk[*] *and Paul Van Dooren*[†]

Abstract. **In this note, we propose an implicit method for applying orthogonal transformations on both sides of a product of upper triangular 2×2 matrices that preserve upper triangularity of the factors. Such problems arise in Jacobi type methods for computing the PSVD of a product of several matrices, and in ordering eigenvalues in the periodic Schur decomposition.**

Key Words. Orthogonal transformations, SVD, PSVD, Schur decompostion.

AMS(MOS) Subject Classifications. 15A23, 65F25.

1 Introduction

The problem of computing the singular value decomposition (SVD) of a product of matrices (PSVD) has been considered in [1],[2], [3], [10]. The computation proceeds in two stages. In the first stage the matrices are transformed into the upper triangular forms. In the second iterative stage an implicit Jacobi-type method is applied to the triangular matrices. It is important that after each iteration the matrices stay triangular [8].

A crucial aspect in such implicit Jacobi iterations is the accurate computation of the PSVD of a product of 2×2 triangular matrices. There two conditions have to be satisfied. First, one has to ensure that the orthogonal transformations applied to the triangular matrices must leave the matrices triangular, and second, that the transformations diagonalize the product accurately. It was shown in [1] and [2] that these two conditions are satisfied by a so-called *half-recursive* and *direct* method, respectively, for computing the SVD of the product of two matrices.

In this note we analyze an extension of the *half-recursive* method for computing the SVD of the product of many 2×2 triangular matrices. We also show that

[*]Cornell University, Department of Electrical Engineering, Ithaca, NY 14853-3801
[†]University of Illinois at Urbana-Champaign, Coordinated Science Lab, Urbana, IL 61801

the extension of the *half-recursive* method can be used for swapping eigenvalues in the periodic Schur decomposition described in [4]. For simplicity we assume real matrices and real eigenvalues, but all results are easily extended to the complex case.

2 Criterion for numerical triangularity

Suppose we are given k, $k > 1$, upper triangular matrices A_i, $i = 1, 2..., k$,

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

We denote the product of A_i, $i = 1, 2..., k$, by A,

$$A = A_1 \cdots A_k = \begin{pmatrix} a & b \\ 0 & d \end{pmatrix} .$$

Let the orthogonal matrices Q_1 and Q_{k+1} be such that

$$A' = Q_1 A Q_{k+1}^T = \begin{pmatrix} a' & b' \\ 0 & d' \end{pmatrix} \tag{2.1}$$

is upper triangular. In case we are interested in finding the *Singular Value Decomposition* of A, one imposes the additional condition that $b' = 0$. This defines uniquely the above decomposition up to permutations that interchange the diagonal elements of A'. In case we are interested in finding the *Schur Form* of A, one imposes the additional condition that $Q_1 = Q_{k+1}$. Again, this defines uniquely the above decomposition up to the ordering of the diagonal elements of A'. In both cases the transformations Q_1 and Q_{k+1} are thus defined by the choice of ordering of diagonal elements in the resulting matrix A'. Our objective now is to find orthogonal matrices Q_j, $j = 2, 3, ..., k$, such that

$$A'_i = Q_i A_i Q_{i+1}^T = \begin{pmatrix} a'_i & b'_i \\ 0 & d'_i \end{pmatrix} \tag{2.2}$$

are meanwhile maintained in upper triangular form as well. It is easy to see that if $abd \neq 0$ then for a given pair of orthogonal transformations Q_1 and Q_{k+1} there exist unique (up to the sign) orthogonal transformations Q_2,...,Q_k such that (2.2) is satisfied. There are many mathematically equivalent strategies of determining Q_2,...,Q_k. However, as it was shown in [1], [2] and [3], some strategies may produce numerically significantly different results than other strategies. We will consider a particular method numerically acceptable if the triangular matrices after transformations have been applied to them stay numerically triangular in the sense described below.

Let $\bar{A}$ be the computed A, and let $\bar{Q}_i$, $i = 1, 2, ..., k+1$ be the computed transformations. Define

$$\tilde{A}' := \bar{Q}_1 \bar{A} \bar{Q}_{k+1}^T = \begin{pmatrix} \tilde{a}' & \tilde{b}' \\ \tilde{e}' & \tilde{d}' \end{pmatrix} \tag{2.3}$$

and

$$\tilde{A}_i' := \bar{Q}_i A_i \bar{Q}_{i+1}^T = \begin{pmatrix} \tilde{a}_i' & \tilde{b}_j' \\ \tilde{e}_i' & d_i' \end{pmatrix} . \tag{2.4}$$

Let ϵ denote the relative machine precision. Assume that we are given $\bar{Q}_1$ and $\bar{Q}_{k+1}$ such that

$$|\tilde{e}'| = O(\epsilon ||\bar{A}||) \tag{2.5a}$$

We will say that $\tilde{A}_i'$ is numerically triangular if

$$|\tilde{e}_i'| = O(\epsilon ||A_i||) , \tag{2.5b}$$

We will propose a method for computing nearly orthogonal $\bar{Q}_i$, $i = 2, ..., k$, for which, under a slightly stronger version of the assumption (2.5a), the (2,1) element $\tilde{e_i}'$ of $\tilde{A}_i'$ will satisfy (2.5b). Condition (2.5b) justifies truncating the (2,1) element e_i' of $\tilde{A}_i'$ to zero. Thus, $\tilde{e}'$ is also forced to zero.

3 The Algorithm

Our algorithm is a generalization of the algorithms presented in [1] and [3] for computing the PSVD of two and three matrices respectively. There the orthogonal transformations all had the form

$$Q = \begin{pmatrix} s & c \\ -c & s \end{pmatrix} , \tag{3.1}$$

where $c^2 + s^2 = 1$. As we will build on the results presented in those papers we retain this particular choice of orthogonal transformations. While each transformation Q_i is defined by the cosine-sine pair $c_i = \cos\theta_i$ and $s_i = \sin\theta_i$, 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

$$c_i = \frac{1}{\sqrt{1 + t_i^2}} \quad \text{and} \quad s_i = t_i c_i . \tag{3.2}$$

Following the exposition in [1], [3], we consider the result of applying the left and right transformations Q_l (for the outer left transformation) and Q_r (for the outer right transformation) to a 2×2 upper triangular matrix A:

$$A' = Q_l A Q_r^T = \begin{pmatrix} a' & b' \\ e' & d' \end{pmatrix} = \begin{pmatrix} s_l & c_l \\ -c_l & s_l \end{pmatrix} \begin{pmatrix} a & b \\ 0 & d \end{pmatrix} \begin{pmatrix} s_r & c_r \\ -c_r & s_r \end{pmatrix}^T . \tag{3.3}$$

We can derive from (3.3) these four relations:

$$e' = c_l c_r(-a t_r + d t_l - b) , \tag{3.4a}$$

$$b' = c_l c_r(-a t_l + d t_r + b t_l t_r) , \tag{3.4b}$$

$$a' = c_l c_r(b t_l + d + a t_l t_r) , \tag{3.4c}$$

$$d' = c_l c_r (a - b t_r + d t_l t_r) \ , \tag{3.4d}$$

where $t_l = \tan\theta_l$ and $t_r = \tan\theta_r$.

The postulates that both e' and b' be zeros define two conditions on t_l and t_r, so that (3.3) represents an SVD of A [5]. The postulate that e' be zero and $t_l = t_r$ represent conditions for swapping eigenvalues of A.

The postulate that e' be zero defines a condition relating θ_l to θ_r, so that if one is known the other can be computed in order to reduce A' to an upper triangular form. For ease of exposition, we assume for now on that $abd \neq 0$. It implies that $c_l c_r \neq 0$, and so the postulate that $e' = 0$ in (3.4a) becomes

$$-a t_r + d t_l - b = 0 \ . \tag{3.5}$$

The consequence of (3.5) is that (3.4c) and (3.4d) simplify to

$$a' = c_l c_r (t_l^2 + 1) d \tag{3.6a}$$

and

$$d' = c_l c_r (t_r^2 + 1) a \ , \tag{3.6b}$$

respectively.

Assume that $Q_l = Q_1$ and $Q_r = Q_{k+1}$ are given, that is $t_l = t_1$ and $t_r = t_{k+1}$ are known. We will use relations of the type (3.5) with t_l and t_r as the reference tangents to compute the remaining transformations.

Our algorithm can be described recursively as follows. We split the sequence A_1, A_2,...,A_{k+1} into two subsequences of consecutive matrices A_1, A_2,...,A_m and A_{m+1}, A_{m+2},...,A_{k+1} where $1 < m < k+1$. Let us denote

$$A_l \equiv \begin{pmatrix} a_l & b_l \\ 0 & d_l \end{pmatrix} = \prod_{i=1}^{m} A_i \ \ and \ \ A_r \equiv \begin{pmatrix} a_r & b_r \\ 0 & d_r \end{pmatrix} = \prod_{i=m+1}^{k} A_{i+1} \ . \tag{3.7}$$

Suppose that

$$|t_l d| \leq |t_r a| \ .$$

Then we propose to compute t_m from the condition (3.5) by the forward substitution,

$$t_m = \frac{d_l t_l - b_l}{a_l} \tag{3.8a}$$

Otherwise, that is when

$$|t_l d| > |t_r a| \ ,$$

we propose to compute t_m from (3.5) by the backward substitution,

$$t_m = \frac{a_r t_r + b_r}{d_r} \ . \tag{3.8b}$$

Having defined the first step, the procedure can now be applied recursively to generate all the remaining orthogonal transformations Q_i, $i = 2, ..., k$. Note that there is a lot of freedom in splitting the sequence A_1, A_2,...,A_{k+1} into subsequent subsequences. This might be advantageous for a divide-and-conquer type of computation in a parallel environment.

As will be shown later, under mild conditions on Q_1 and Q_{k+1}, this particular way of generating orthogonal transformations Q_i, $i = 2, ..., k$, will guarantee that all A_i' will be numerically upper triangular in the sense that (2.5b) will be satisfied.

4 Error Analysis

In our error analysis, we 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 in this first-order analysis.

The function $\mathrm{fl}(a)$ will denote the floating point approximation of a. For the purpose of the analysis, a "bar" denotes a computed quantity which is perturbed as the result of inexact arithmetic. For example, instead of a, b and d, we have the perturbed values $\bar{a}$, $\bar{b}$ and $\bar{d}$ which result from floating point computation of $\prod_{i=1}^{k+1} A_i$. We assume that exact arithmetic may be performed using these perturbed values. The "tilde" symbol is used to denote conceptual values computed exactly from perturbed data.

We start our procedure by computing elements of the product matrix A as the product of $\bar{A}_l$ and $\bar{A}_r$ defined by (3.7):

$$\bar{a} := fl(\bar{a}_l \bar{a}_r) = \bar{a}_l \bar{a}_r (1+\alpha) \ , \tag{4.1a}$$

$$\bar{d} := fl(\bar{d}_l \bar{d}_r) = \bar{d}_l \bar{d}_r (1+\delta) \ , \tag{4.1b}$$

$$\bar{b} := fl(\bar{a}_l \bar{b}_r + \bar{b}_l \bar{d}_r) = \bar{a}_l \bar{b}_r (1+2\beta_1) + \bar{b}_l \bar{d}_l (1+2\beta_2) \ , \tag{4.1c}$$

where, according to our convention, the parameters α, δ, β_1, β_2, and β_3 are all quantities whose absolute values are bounded by ϵ.

Now we specify the condition that we impose on the computed Q_1 and Q_{k+1}.

Assumption I. Throught the rest of this note we will assume that the computed tangents $\bar{t}_l$ and $\bar{t}_r$ corresponding to the outer transformations $Q_l = Q_1$ and $Q_r = Q_{k+1}$ satisfy the following equality

$$\bar{a}(1+C\psi)\bar{t}_r - \bar{d}(1+C\phi)\bar{t}_l + \bar{b}(1+C\chi) = 0 \ , \tag{4.2a}$$

where $C = C(k)$.

□

Lemma 4.1. *The recurrence (3.8a) yields $\bar{t}_m$ such that*

$$\bar{a}_l(1+2\psi_1)\bar{t}_m - \bar{d}_l(1+\phi_1)\bar{t}_l + \bar{b}_l = 0 \ . \tag{4.3}$$

Likewise, the recurrence (3.8b) yields $\bar{t}_m$ such that

$$\bar{d}_r(1+2\phi_2)\bar{t}_m - \bar{a}_r(1+\psi_2)\bar{t}_r - \bar{b}_r = 0 \ . \tag{4.4}$$

□

Proof. The proof easily follows from (3.8a) and (3.8b).

□

Theorem4.2. *If $|\bar{t}_l \bar{d}| < |\bar{t}_r \bar{a}|$ and if $\bar{t}_m$ is computed via (3.8a) then $\bar{t}_m$ satisfies the relation*

$$\bar{a}_r(1+C_l\psi_l)\bar{t}_r - \bar{d}_r(1+C_l\phi_l)\bar{t}_m + \bar{b}_r(1+C_l\psi_l) = 0 \tag{4.5a}$$

where $C_l = C_l(k)$. Likewise, if $|\bar{t}_l \bar{d}| \geq |\bar{t}_r \bar{a}|$ and if $\bar{t}_m$ is computed via (3.8b) then $\bar{t}_m$ satisfies the relation

$$\bar{a}_l(1+C_r\psi_r)\bar{t}_m - \bar{d}_l(1+C_r\phi_r)\bar{t}_l + \bar{b}_l(1+C_r\chi_r) = 0 \tag{4.5b}$$

where $C_r = C_r(k)$.

Proof. We give a proof of the relation (4.5a) only as the relation (4.5b) can be proved in an analogous way.

First from (4.3)-(4.4) we get

$$\bar{a}_l(1+2\psi_1)\bar{t}_m - \bar{d}_l(1+\phi_1)\bar{t}_l + \bar{b}_l = 0 \ , \tag{4.6a}$$

while from Assumption I and (4.1a)-(4.1c) we have

$$\bar{a}_l\bar{a}_r(1+\alpha+C\psi)\bar{t}_r - \bar{d}_l\bar{d}_r(1+\delta+C\phi)\bar{t}_l +$$
$$\bar{a}_l\bar{b}_r(1+2\beta_1+C\chi) + \bar{b}_l\bar{d}_r(1+2\beta_2+C\chi) = 0 \ . \tag{4.6b}$$

By multiplying both sides of (4.6a) by $d_r(1+2\beta_2+C\chi)$ and subtracting from (4.6b) we obtain

$$\bar{a}_l\{\bar{a}_r(1+\alpha+C\psi)\bar{t}_r - \bar{a}_r\left(\frac{\bar{d}_l\bar{d}_r}{\bar{a}_l\bar{a}_r}\right)(\delta+C\phi-\phi_1-2\beta_2-C\chi)\bar{t}_l +$$
$$\bar{b}_r(1+2\beta_1+C\chi) - \bar{d}_r(1+2\beta_2+C\chi+2\psi_1)\bar{t}_m\} = 0 \ ,$$

or, since $\bar{a}_l \neq 0$,

$$\bar{a}_r(1+\alpha+C\psi)\bar{t}_r - \bar{a}_r\bar{t}_r\left(\frac{\bar{d}\bar{t}_l}{\bar{a}\bar{t}_r}\right)(\delta+C\phi-\phi_1+2\beta_2+C\chi) +$$
$$\bar{b}_r(1+2\beta_1+C\chi) - \bar{d}_r(1+2\beta_2+2\psi_1+C\chi)\bar{t}_m = 0 \ .$$

As we assumed that $|\bar{t}_l\bar{d}| < |\bar{t}_r\bar{a}|$, the above can be rewritten as

$$\bar{a}_r(1+C_l\psi_l)\bar{t}_r - \bar{d}_r(1+C_l\phi_l)\bar{t}_m + \bar{b}_r(1+C_l\chi_l) = 0 \tag{4.7}$$

where $C_l = C_l(k)$ completing the proof. □

We now justify why the (2,1) element in the computed matrix A_i' can be set to zero. Let the cosine and sine pairs $\tilde{c}_i$ and $\tilde{s}_i$ satisfy $\bar{t}_i = \tilde{s}_i/\tilde{c}_i$, for $i = l, m, r$. ¿From (4.2) we can derive that

$$\bar{c}_i := fl(\tilde{c}_i) = \tilde{c}_i(1+3\mu_i) \ , \tag{4.8a}$$

$$\bar{s}_i := fl(\tilde{s}_i) = \tilde{s}_i(1+4\nu_i) \ . \tag{4.8b}$$

Let $\tilde{A}_i'$ denote the exact updated matrix derived from $\bar{A}_i$, $i = l, r$, and $\bar{c}_i$, $\bar{s}_i$, $i = l, m, r$ that is

$$\tilde{A}_l' = \begin{pmatrix} \bar{s}_l & \bar{c}_l \\ -\bar{c}_l & \bar{s}_l \end{pmatrix}\begin{pmatrix} \bar{a}_l & \bar{b}_l \\ 0 & \bar{d}_l \end{pmatrix}\begin{pmatrix} \bar{s}_m & -\bar{c}_m \\ \bar{c}_m & \bar{s}_m \end{pmatrix} \ , \tag{4.9a}$$

and

$$\tilde{A}_r' = \begin{pmatrix} \bar{s}_m & \bar{c}_m \\ -\bar{c}_m & \bar{s}_m \end{pmatrix}\begin{pmatrix} \bar{a}_r & \bar{b}_r \\ 0 & \bar{d}_r \end{pmatrix}\begin{pmatrix} \bar{s}_r & -\bar{c}_r \\ \bar{c}_r & \bar{s}_r \end{pmatrix} \ . \tag{4.9b}$$

Our next result is a direct consequence of Theorem 4.2 and provides bounds on the elements $\tilde{e}_i'$, $i = l, r$, defined by the relations

$$\tilde{e}_l' := -\bar{c}_l\bar{s}_m a_l + \bar{s}_l\bar{c}_m d_l - \bar{c}_l\bar{c}_m b_l \ , \tag{4.10a}$$

$$\tilde{e}'_r := -\bar{c}_m \bar{s}_r a_r + \bar{s}_m \bar{c}_r d_r - \bar{c}_m \bar{c}_r b_r \ . \tag{4.10b}$$

Corollary 4.4: If $|\bar{t}_l \bar{d}| < |\bar{t}_r \bar{a}|$ and if $\bar{t}_m$ is computed via (3.8a) or if $|\bar{t}_l \bar{d}| \geq |\bar{t}_r \bar{a}|$ and if $\bar{t}_m$ is computed via (3.8b) then

$$|\tilde{e}'_i| \leq K_i \epsilon ||\bar{A}_i|| \ \ , \ \ for \ \ i = l, r \ . \tag{4.11}$$

□

Proof. We prove the corollary for the case when $|\bar{t}_l \bar{d}| < |\bar{t}_r \bar{a}|$ and when $\bar{t}_m$ is computed via (3.8a). The other case can be proved in an analogous manner.

Using (4.3a) we can rewrite (4.10a) as

$$\tilde{e}'_l = -\bar{c}_l \bar{s}_m \bar{a}_l + \bar{s}_l \bar{c}_m \bar{d}_l - \bar{c}_l \bar{c}_m \bar{b}_l +$$
$$\bar{c}_l \bar{c}_m (\bar{a}_l (1 + 2\psi_1) \bar{t}_m - \bar{d}_l (1 + \phi_1) \bar{t}_l + \bar{b}_l) \tag{4.12}$$

from which it follows that

$$|\tilde{e}'_l| \leq K_l \epsilon ||\bar{A}_l|| \ .$$

Similarly, using (4.5a) we can rewrite (4.10b) as

$$\tilde{e}'_r := -\bar{c}_m \bar{s}_r a_r + \bar{s}_m \bar{c}_r d_r - \bar{c}_r \bar{c}_m b_r +$$
$$\bar{c}_r \bar{c}_m (\bar{a}_r (1 + C_l \psi_l) \bar{t}_r - \bar{d}_r (1 + C_l \phi_l) \bar{t}_m + \bar{b}_r (1 + C_l \psi_l)) \tag{4.13}$$

and thus

$$|\tilde{e}'_r| \leq K_r \epsilon ||\bar{A}_r|| \ ,$$

completing the proof of (4.10a).

□

5 Numerical examples

The SVD algorithms for 2×2 upper triangular matrices in [1],[2] or [5] give $\bar{t}_l$ and $\bar{t}_r$ which satisfy Assumption I. We will illustrate that by using our new scheme triangularity of the transformed factors is preserved.

Consider the case of three matrices in the product. Assume that the given data matrices are

$$A_1 = \begin{pmatrix} 2.316797292247488e+00 & -1.437687878748196e-01 \\ 0 & -2.718295063593277e-02 \end{pmatrix} \ ,$$

$$A_2 = \begin{pmatrix} 1.222222234444442e+00 & 3.480474357220011e-01 \\ 0 & 5.674165405829751e+00 \end{pmatrix} \ ,$$

$$A_3 = \begin{pmatrix} 2.222222111111111e-01 & 1.732050807568877e+00 \\ 0 & 1.111111110000000e-12 \end{pmatrix} \ .$$

They generate the matrix product $\bar{A} := A1 \cdot A2 \cdot A3$

$$\bar{A} = \begin{pmatrix} 6.292535886949669e-01 & 4.904546363614013e+00 \\ 0 & -1.713783977472744e-13 \end{pmatrix} \ .$$

We are interested in computing orthogonal transformations Q_1, Q_2, Q_3 and Q_4 which satisfy (2.2) and (2.1) with the (1,2) element zero. The SVD algorithm for

the 2×2 upper triangular matrix $\bar{A}$ in [1] or [5] gives $\bar{t}_1 = 3.437688760727056e-14$ and $\bar{t}_4 = -7.794228673031074e+00$ which satisfy Assumption I. In fact we have

$$\bar{Q}_1 \bar{A} \bar{Q}_4 = \begin{pmatrix} -2.180909253067911e-14 & -7.494178599599612e-30 \\ 0 & 4.944748235423613e+00 \end{pmatrix}$$

We split $\bar{A}$ into the product of $A_{1,2} = A_1 A_2$ and A_3. We note that the ratio

$$\frac{\bar{t}_1 \bar{d}}{\bar{t}_4 \bar{a}} = 1.201223412093697e-27$$

If we compute t_3 from t_1 as indicated by the ratio, and next t_2 as specified by (3.8a) or (3.8b) then Corollary 5.4 will guarantee that the transformed factors will stay (numerically) triangular. Suppose however that we compute t_3 from t_4 and next t_2 from t_3. Then Lemma 4.1 will guarantee that $Q_2 A_2 Q_3^T$ and $Q_3 A_3 Q_4^T$ will stay numerically triangular. However, for the computed $Q_1 A_1 Q_2^T$ we have

$$Q_1 A_1 Q_2^T = \begin{pmatrix} -2.713066430028558e-02 & -1.685188387402401e-03 \\ -1.360106941575845e-04 & 2.321253786046106e+00 \end{pmatrix}$$

which cannot be considered upper triangular. An error of order 10^{-4} has to be introduced to truncate the (2,1) element in $Q_1 A_1 Q_2^T$ so it becomes upper triangular.

Acknowledgements
Adam Bojanczyk was partially supported by the Joint Services Electronics Program (Grant F49620–90–C–0039 monitored by AFOSR). Paul Van Dooren was partially supported by the Research Board of the University of Illinois at Urbana-Champaign (Grant P 1–2–68114) and by the National Science Foundation (Grant CCR 9209349).

References

[1] G.E. Adams, A.W. Bojanczyk and F.T. Luk, *Computing the PSVD of Two* 2×2 *Triangular Matrices*, submitted to SIMAX.

[2] Z. Bai and J.W. Demmel, *Computing the Generalized Singular Value Decomposition*, Report No UCB/CSD 91/645, Computer Science Division, University of California, Berkeley, August 1991.

[3] A.W. Bojanczyk, L.M. Ewerbring, F.T. Luk and P. Van Dooren, *An Accurate Product SVD Algorithm*, *Signal Processing*, 25 (1991), pp. 189-201.

[4] A.W. Bojanczyk, P. Van Dooren and G.H. Golub, *The periodic Schur decomposition. Algorithms and applications*, to appear in Proceedings SPIE, San Diego, July 1992.

[5] 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.

[6] 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.

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

[10] H. Zha, *A numerical algorithm for computing the restricted SVD of matrix triplets*, to appear in Linear Algebra and Its Applications.