Computing the CS and the Generalized Singular Value Decompositions

Charles Van Loan

TR 84-614
June 1984

Department of Computer Science
Cornell University
Ithaca, New York 14853

The research associated with this paper was partially supported by the Office of Naval Research contract N00014-83-K-0640.

COMPUTING THE CS AND THE GENERALIZED SINGULAR VALUE DECOMPOSITIONS

Charles Van Loan
Department of Computer Science
Cornell University
Ithaca, New York 14853

ABSTRACT

If the columns of a matrix are orthonormal and it is partitioned into a 2-by-1 block matrix, then the singular value decompositions of the blocks are related. This is the essence of the "CS decomposition". The computation of these related SVD's requires some care. Stewart has given an algorithm that uses the LINPACK SVD algorithm together with a Jacobi-type "clean-up" operation on a cross-product matrix. Our technique is equally stable and fast but avoids the cross product matrix. The simplicity of our technique makes it more amenable to parallel computation on systolic-type computer architectures. These developments are of interest because the best way to compute the generalized singular value decomposition of a matrix pair (A,B) is to compute the CS decompositiion of a certain orthogonal column matrix related to A and B.

The research associated with this paper was partially supported by the Office of Naval Research contract N00014-83-K-0640.

1. The C-S Decomposition

Suppose the columns of the real matrix

$$Q = \begin{array}{c} \begin{bmatrix} Q_1 \\ Q_2 \end{bmatrix} \\ p \end{array} \begin{array}{c} n_1 \\ n_2 \\ \end{array} \qquad n_1 \geq p$$

are orthonormal, i.e., $Q_1^T Q_1 + Q_2^T Q_2 = I_p$. The gist of the CS decomposition (CSD) is that the singular value decomposition (SVD) of Q_1 is related to the singular value decomposition of Q_2. In particular, there exist orthogonal matrices U_1 $(n_1 \times n_1)$, U_2 $(n_2 \times n_2)$, and V $(p \times p)$ such that

$$U_1^T Q_1 V = C = diag(c_1, \ldots, c_p)$$

and

$$U_2^T Q_2 V = S = diag(s_1, \ldots, s_q) \qquad q = \min\{p, n_2\}.$$

Since $C^T C + S^T S = I_p$, it follows that

$$\begin{array}{ll} c_i^2 + s_i^2 = 1 & i = 1,\ldots,q \\ |c_i| = 1 & i = q+1,\ldots,p \ . \end{array}$$

Thus, the singular values of Q_1 and Q_2 are cosines and sines accounting for the name of the decomposition. Without loss of generality, we may assume that the c_i and s_i are ordered as follows:

$$\begin{array}{c} 0 \leq c_1 \leq \cdots \leq c_q \leq c_{q+1} = \cdots = c_p = 1 \\ 1 \geq s_1 \geq \cdots \geq s_q \geq 0 \ . \end{array} \qquad (1.1)$$

This paper is about a new way to compute the CSD.

The CSD and its role in the analysis of various invariant subspace perturbation problems is discussed in Davis and Kahan [1], Stewart [10], and Van Loan [13]. For a proof of the CSD, see Stewart's paper.

Paige and Saunders [8] have shown that computing the CSD is crucial to the stable computation of the generalized singular value decomposition (GSVD). In the GSVD, we are given two matrices A $(n_1 \times t\ ,\ n_1 \geq t)$ and B $(n_2 \times t)$ and find orthogonal U_A $(n_1 \times n_1)$, orthogonal U_B $(n_2 \times n_2)$, and a nonsingular X $(t \times t)$ such that

$$A = U_A D_A X^T \quad , \quad D_A = diag(\alpha_1, \ldots, \alpha_t)$$

and

$$B = U_B D_B X^T \quad , \quad D_B = diag(\beta_1, \ldots, \beta_r)$$

where $r = \min\{t, n_2\}$. To compute this decomposition we first compute the SVD

$$M \equiv \begin{bmatrix} A \\ B \end{bmatrix} = Q \Sigma Z^T \ . \tag{1.2}$$

Assume that $rank(M) = p$, set $m = n_1 + n_2 - p$, and conformably partition Q, Σ, and Z as follows:

$$Q = \begin{array}{cc} \begin{bmatrix} Q_{11} & Q_{12} \\ Q_{21} & Q_{22} \end{bmatrix} & \begin{array}{c} p \\ m \end{array} \\ \begin{array}{cc} p & m \end{array} & \end{array} \qquad \Sigma = \begin{array}{cc} \begin{bmatrix} \Sigma_p & O \\ O & O \end{bmatrix} & \begin{array}{c} p \\ m \end{array} \\ \begin{array}{cc} p & t-p \end{array} & \end{array}$$

$$Z = \begin{array}{c} \begin{bmatrix} Z_1 , & Z_2 \end{bmatrix} \\ \begin{array}{cc} p & t-p \end{array} \end{array} .$$

Note that $Q_{11}^T Q_{11} + Q_{21}^T Q_{21} = I_p$. Let $Q_{11} = U_1 C V^T$ and $Q_{21} = U_2 S V^T$ be the CSD of Q_{11} and Q_{21}. Since

$$\begin{bmatrix} A \\ B \end{bmatrix} Z = \begin{bmatrix} AZ_1 & O \\ BZ_1 & O \end{bmatrix} = \begin{bmatrix} Q_{11}\Sigma_p & O \\ Q_{21}\Sigma_p & O \end{bmatrix} = \begin{bmatrix} U_1 C V^T \Sigma_p & O \\ U_2 S V^T \Sigma_p & O \end{bmatrix}$$

we have

$$\begin{bmatrix} A \\ B \end{bmatrix} = \begin{bmatrix} U_1 & O \\ O & U_2 \end{bmatrix} \begin{bmatrix} C & O \\ S & O \end{bmatrix} \begin{bmatrix} V^T \Sigma_p & O \\ O & W \end{bmatrix} Z^T$$

where W is an arbitrary $(t-p) \times (t-p)$ matrix. If W is nonsingular then the GSVD follows by setting $U_A = U_1$, $U_B = U_2$, $D_A = [C \ \ 0]$, $D_B = [S \ \ O]$, and

$$X^T = \begin{bmatrix} V^T \Sigma_p & O \\ O & W \end{bmatrix} Z^T \ .$$

Note that if $\Sigma_p = diag(\sigma_1, \ldots, \sigma_p)$ and $\sigma_1 \geq \cdots \geq \sigma_p > 0$, then the 2-norm condition of X satisfies $\kappa_2(X) \geq \sigma_1 / \sigma_p$. The lower bound can be achieved by setting $W = \sigma I_{t-p}$ for any σ that satisfies $\sigma_p \leq \sigma \leq \sigma_1$.

The GSVD and , hence, the CSD, are useful for solving various constrained and generalized least squares problems. However, the CSD is also useful in its own right. See Van Loan [12,13]. Stewart [11] devised the first stable CSD algorithm and the current paper arose by our desire to develop an implementation of his method suitable for systolic type architectures. These architectures are of interest in certain real time signal processing applications [3,4,9].

As we have shown, the problem of computing the GSVD boils down to the problem of computing the CSD. In §2 we describe several CSD algorithms each of which is flawed because of numerical problems that are associated with the orthonormalization of nearly orthogonal bases. In §3 two results are established that indicate when these problems can be circumvented. Our main algorithm is then presented in §4 while some aspects associated with its implementation are discussed in §5.

2. Some Obvious CSD Algorithms and Their Shortcomings

At first glance it appears that the CSD should be a rather easy decomposition to compute. After all, it just involves a pair of SVD's for which there are several efficient, stable methods. See [2,5]. In this section we show by example why the stable computation of the CSD is not straightforward.

Before we proceed, however, it is appropriate to state what we mean by a "stable" CSD algorithm. Let ϵ denote the machine precision and suppose an algorithm for computing the CSD generates $\hat{U}_1$, $\hat{U}_2$, $\hat{V}$, $\hat{C}$, and $\hat{S}$, computed versions of U_1, U_2, V, C, and S respectively. Assume that

$$||Q_1^TQ_1 + Q_2^TQ_2 - I_p||_2 \approx \epsilon , \tag{2.1}$$

i.e., assume that the columns of

$$Q = \begin{bmatrix} Q_1 \\ Q_2 \end{bmatrix}$$

are orthonormal to within roundoff error. We say that a CSD algorithm is *stable* if the following conditions hold:

$$||\hat{U}_1^T\hat{U}_1 - I_{n_1}||_2 \approx \epsilon \tag{2.2}$$

$$||\hat{U}_2^T\hat{U}_2 - I_{n_2}||_2 \approx \epsilon \tag{2.3}$$

$$||\hat{V}^T\hat{V} - I_p||_2 \approx \epsilon \tag{2.4}$$

$$\hat{C} = diag(\hat{c}_1, \ldots, \hat{c}_p) = \hat{U}_1^T(Q_1 + E_1)\hat{V} \quad , \quad ||E_1||_2 \approx \epsilon||Q_1||_2 \tag{2.5}$$

$$\hat{S} = diag(\hat{s}_1, \ldots, \hat{s}_q) = \hat{U}_2^T(Q_2 + E_2)\hat{V} \quad , \quad ||E_2||_2 \approx \epsilon||Q_2||_2 \ . \tag{2.6}$$

These assertions say that to within roundoff error, $\hat{U}_1$, $\hat{U}_2$, and $\hat{V}$ are orthogonal and $\hat{U}_1^TQ_1\hat{V}$ and $\hat{U}_2^TQ_2\hat{V}$ are diagonal. Using standard SVD perturbation theory [6,p285ff] one can also conclude from (2.2)-(2.6) that the $\hat{c}_i$ and $\hat{s}_i$ are the exact singular values of matrices that are relatively close to Q_1 and Q_2 respectively.

Having in mind the goal of a stable CSD algorithm, let us examine the numerical properties of two obvious CSD algorithms. For the sake of clarity we assume in our examples that both Q_1 and Q_2 are square and that Q_1 is nonsingular.

Algorithm 2.1 ($p = n_1 = n_2$)

1. Use the LINPACK SVD algorithm [2] to compute $U_2^TQ_2V = S = diag(s_1, \ldots, s_p)$. Note: the s_i are ordered from large to small.
2. Set $X = Q_1V$.
3. Set $C = diag(c_1, \ldots, c_p)$ where c_k is the 2-norm of X's k-th column.
4. Set $U_1 = XC^{-1}$

The rationale behind this algorithm is as follows. Since $X^TX = diag(1 - s_i^2)$, the columns of X are mutually orthogonal and the 2-norm of the i-th column equals $\sqrt{1 - s_i^2}$. By assumption, $X = Q_1V$ is nonsingular. Thus, the matrix $C = diag(\sqrt{1 - s_1^2}, \ldots, \sqrt{1 - s_p^2})$ is nonsingular, $U_1 = XC^{-1}$ is orthogonal, and $U_1^TQ_1V = C$.

The dangers of Algorithm 2.1 are highlighted in Stewart [11]. To see what they are, consider the example

$$Q_1 = \begin{bmatrix} .220508860423 & -.114095899416 & .001410518052 & .309131888087 \\ .075149984350 & .552192330457 & .309420137864 & .519525649668 \\ .346099513974 & -.465523358094 & -.147474170901 & .284504924779 \\ .200314808251 & .015869922033 & .063768831702 & .364621650530 \end{bmatrix}$$

$$Q_2 = \begin{bmatrix} -.149903307775 & .456869095895 & -.814555019070 & .205461483909 \\ -.132593956233 & .403919514293 & .374067025998 & -.294979263882 \\ .631588073183 & .226164206817 & .132173742848 & .047014825861 \\ -.588949720476 & -.205112923304 & .239887841318 & .537774110108 \end{bmatrix} \tag{2.7}$$

Q_1 and Q_2 satisfy (2.1) with $\epsilon = 10^{-12}$. Moreover,

$$\begin{aligned} c_1 &= .899999999985 \\ c_2 &= .799999999989 \\ c_3 &= .000020000000 \\ c_4 &= .000009999999 \end{aligned}$$

Using MATLAB [7] with an effective machine precision $\epsilon = 10^{-12}$ we found that criteria (2.3)-(2.6) were each satisfied. Unfortunately (2.2) fails to hold because

$$\hat{U}_1^T \hat{U}_1 = \begin{bmatrix} 0.99999999999 & 0.00671196274 & -0.00000000083 & -0.00000006623 \\ 0.00671196274 & 1.00000000000 & -0.00000019837 & 0.00000004765 \\ -0.00000000083 & -0.00000019837 & 1.00000000000 & -0.00000000000 \\ -0.00000006623 & 0.00000004765 & -0.00000000000 & 0.99999999999 \end{bmatrix}$$

The trouble stems from the fact that the first two columns in the computed $\hat{X}$ have norm $O(10^{-5})$. Consequently, errors of order $10^5\epsilon$ are introduced when the columns of $\hat{X}$ are normalized to produce $\hat{U}_1$.

A way to avoid this loss of orthogonality is to orthonormalize $\hat{X}$ by stably computing its QR factorization, say by using Householder matrices. This is the approach taken in our next algorithm.

Algorithm 2.2 $(p = n_1 = n_2)$

1. Use the LINPACK SVD algorithm to compute $U_2^T Q_2 V = S = diag(s_1, \ldots, s_p)$.
2. Set $X = Q_1 V$.
3. Use the LINPACK QR factorization algorithm to compute $X = U_1 R$ where U_1 is orthogonal and R is upper triangular with positive diagonal entries.
4. Set $C = diag(r_{11}, \ldots, r_{pp})$.

In exact arithmetic we should have $R = diag(r_{11}, \ldots, r_{pp})$ because a nonsingular upper triangular matrix whose columns are mutually orthogonal must be diagonal. It follows that $C = U_1^T X = U_1^T Q_1 V$.

Let us apply Algorithm 2.2 to the example (2.7) above. Again using MATLAB with an effective machine precision of $\epsilon = 10^{-12}$ we find that the quantities produced by Algorithm 2.2 satisfy (2.2)-(2.6) with the exception of (2.5):

$$||\hat{U}_1 \hat{C} \hat{V}^T - Q_1||_2 \approx 10^{-7} \approx \sqrt{\epsilon}$$

This is because the computed version of the matrix $\hat{R}$ is not diagonal as the theory predicts:

$$\hat{R} = \begin{bmatrix} .000010000300 & .000000134238 & -.000000000665 & -.000000059614 \\ .000000000000 & .000019999399 & -.000000158697 & .000000043290 \\ .000000000000 & .000000000000 & .799999999991 & -.000000000006 \\ .000000000000 & .000000000000 & .000000000000 & .899999999989 \end{bmatrix}$$

Thus, it appears that Algorithm 2.2 is an improvement over Algorithm 2.1 in that it renders a suitably orthogonal $\hat{U}_1$. Unfortunately, the price paid for this orthogonality is the violation of (2.5).

In Stewart's CSD algorithm a Jacobi-like "clean-up" operation that rectifies these problems. It entails working with the matrix $\hat{X}^T \hat{X}$. Our procedure is similar but it circumvents the cross-product matrix by exploiting some rather simple theorems that we present in the next section.

3. Safe Diagonalization

Suppose X $(m \times k)$ has rank k and let

$$X = [x_1, \ldots, x_k] \tag{3.1}$$

be a column partitioning. Assume that

$$X^T X = D^2 + E \tag{3.2}$$

where

$$D = diag(||x_1||_2, \ldots, ||x_k||_2) \ . \tag{3.3}$$

Let ϵ be the machine precision. The flawed algorithms of the previous section prompt us to ask the following two questions:

When does $X = (XD^{-1})D$ represent a stable QR factorization? That is, what conditions on D and E ensure that $U = XD^{-1}$ satisfies $||U^T U - I_k||_2 \approx \epsilon$?

If $X = UR$ is the QR factorization of X, then what conditions ensure that R is safely diagonal? By "safely diagonal" we mean that for all $i \neq j$ we have $|r_{ij}| \approx \epsilon \cdot ||R||_2$.

The following theorem answers the first of these two questions.

Theorem 3.1

If X $(m \times k)$ satisfies (3.1)-(3.3) and if $U = XD^{-1}$ then

$$||U^T U - I_k||_2 \leq \frac{||E||_2}{\min ||x_i||_2^2} \leq \frac{||E||_2}{\sigma_{\min}^2(X)}$$

where $\sigma_{\min}(\cdot)$ denotes the minimum singular value.

Proof.

$$U^T U = D^{-1}(X^T X)D^{-1} = D^{-1}(D^2 + E)D^{-1}$$

and so

$$||U^T U - I_k||_2 = ||D^{-1}ED^{-1}||_2 \leq ||D^{-1}||_2^2 ||E||_2 \leq ||E||_2 / \min ||x_i||_2^2 \ .$$

The proof is completed with the observation that $||x_i||_2 \geq \sigma_{\min}(X)$ for all i. □

This result essentially shows that QR via column normalization, i.e., Algorithm 2.1, is stable so long as (a) the matrix X has no small columns and (b) the off-diagonal elements of X^TX are small compared to the machine precision. Let us relate these comments to Algorithm 2.1. Since

$$\hat{X}^T\hat{X} = I_p - \hat{S}^2 + E \qquad ||E||_2 \leq \epsilon$$

where the "hats" designate computed quantities, we see that $\hat{U}_1$ will be orthogonal to within roundoff error provided none of the Q_2 singular values are near 1. (The roundoff error analysis details associated with the computation of $\hat{U}_1$ have been supressed as they are straightforward.)

We now focus on the second question posed above.

Theorem 3.2

Assume that (3.1)-(3.3) hold and that $X = QR$ where Q $(m \times m)$ is orthogonal and R $(m \times k)$ is upper triangular. If

$$X_i = [x_1, \ldots, x_i] \qquad i = 1,\ldots k$$

then for all i and j satisfying $j > i$ we have

$$|r_{ij}| \leq \min \{ ||x_j||_2 , ||E||_2 / \sigma_{\min}(X_i) \} .$$

Proof.

Let $G = X^TX$ and let R_i be the leading $i \times i$ principal submatrix of R. From the equation $R^TR = G$ it follows that

$$R_i^T \begin{bmatrix} r_{1j} \\ \cdot \\ \cdot \\ r_{ij} \end{bmatrix} = X_i^Tx_j \quad , \quad j = i+1,\ldots,k \ .$$

Thus,

$$|r_{ij}| \leq (r_{1j}^2 + \ldots + r_{ij}^2)^{1/2} \leq ||R_i^{-T}||_2 ||X_i^Tx_j||_2$$

Since $\sigma_{\min}(R_i) = \sigma_{\min}(X_i)$ and $||X_i^Tx_j||_2 \leq ||E||_2$ we have

$$|r_{ij}| \leq ||E||_2 / \sigma_{\min}(X_i) \ .$$

The theorem follows since

$$|r_{ij}| \leq (r_{1j}^2 + \cdots + r_{jj}^2)^{1/2} = ||x_j||_2 \qquad \square .$$

The theorem helps to explain why the matrices produced by Algorithm 2.2 may fail to satisfy (2.5). Let $\hat{X} = [\hat{x}_1, \ldots, \hat{x}_p]$ be the computed version of the matrix $X = [x_1, \ldots, x_p]$. A straightforward error analysis shows that

$$\hat{X}^T\hat{X} = diag(1 - \hat{s}_1^2, \ldots, 1 - \hat{s}_p^2) + E$$

where $||E||_2 \approx \epsilon$ and the $\hat{s}_i$ are the computed singular values of Q_2 . Moreover, the computed upper triangular matrix $\hat{R}$ turns out to satisfy

$$U_1^T(\hat{X} + F) = \hat{R}$$

where U_1 is exactly orthogonal and $||F||_2 \approx \epsilon||\hat{X}||_2$. By invoking the theorem and ignoring second order terms in ϵ we find for $i = 1,...,p$

$$|\hat{r}_{ij}| \approx \min\left\{\frac{\epsilon}{\sqrt{1 - \hat{s}_i^2}}, \hat{c}_j\right\} \approx \min\left\{\frac{\epsilon}{\hat{c}_i}, \hat{c}_j\right\}, \quad j = i + 1, \ldots, p .$$

Thus if we have $\hat{c}_i \approx \hat{c}_j \approx \sqrt{\epsilon}$ for some $j > i$, then we can expect trouble when trying to diagonalize Q_1 .

On the positive side, the theorem does indicate that if a well-conditioned matrix has nearly orthogonal columns, then it can be safely diagonalized by QR . Our algorithm for computing the CSD exploits this property.

4. A Stable Algorithm for the CSD

The new method for computing the CSD that we are about to describe requires a criteria for distinguishing between large and small singular values. This is because we will be invoking Algorithms 2.1 and 2.2 on certain well-conditioned subproblems. To this end we define a number σ to be *large* if $\sigma > 1/\sqrt{2}$ and to be *tiny* if $\sigma \leq 1/\sqrt{2}$. The rationale for choosing $1/\sqrt{2}$ as the dividing line between large and small numbers will be given later.

It is helpful to illustrate our method on a small example. Suppose Q_1 and Q_2 are each 4×4 and that Q_1 has two large and two tiny singular values.

Step 1. Compute the SVD of Q_2 ordering the singular values from *small* to *large*. Apply the right transformation to Q_1. This gives

$$Q_2 := U_2^T Q_2 V = \begin{bmatrix} T & \epsilon & \epsilon & \epsilon \\ \epsilon & T & \epsilon & \epsilon \\ \epsilon & \epsilon & L & \epsilon \\ \epsilon & \epsilon & \epsilon & L \end{bmatrix}$$

$$Q_1 := Q_2 V = \begin{bmatrix} \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \end{bmatrix}$$

Our notation is as follows. We use ϵ to indicate which matrix entries are of order machine precision. "T" stands for a tiny singular value, "L" stands for a large singular value, and "$\times$" denotes an arbitrary non-negligible entry. The reason for the "reverse" ordering of the singular values is that we want the resulting column norms in the updated Q_1 to range from large to small. This has the effect of introducing more negligible matrix entries in Q_1 in the next step.

Step 2. Compute the QR factorization of Q_1:

$$Q_1 := U_1^T Q_1 = \begin{bmatrix} L & \epsilon & \epsilon & \epsilon \\ \epsilon & L & \epsilon & \epsilon \\ \epsilon & \epsilon & r_{33} & r_{34} \\ \epsilon & \epsilon & \epsilon & r_{44} \end{bmatrix}$$

Bear in mind that we always have $Q_1^T Q_1 + Q_2^T Q_2 = I_p$ after every update of Q_1 and Q_2. Thus, the (1,1) and (2,2) entries of Q_1 are large since the norms of the first two columns of Q_2 are tiny. In view of Theorem 3.2, the superdiagonal entries in rows 1 and 2 are negligible. However, we cannot assert this for $|r_{34}|$ since $|r_{33}|$ and $|r_{44}|$ are each tiny.

Step 3. Compute the SVD of the lower 2×2 principal submatrix in Q_1 and apply the right transformation to Q_2:

$$Q_1 := \begin{bmatrix} I_2 & 0 \\ 0 & \tilde{U}_1 \end{bmatrix}^T Q_1 \begin{bmatrix} I_2 & 0 \\ 0 & \tilde{V} \end{bmatrix} = \begin{bmatrix} L & \epsilon & \epsilon & \epsilon \\ \epsilon & L & \epsilon & \epsilon \\ \epsilon & \epsilon & T & \epsilon \\ \epsilon & \epsilon & \epsilon & T \end{bmatrix}$$

$$Q_2 := Q_2 \begin{bmatrix} I_2 & 0 \\ 0 & \tilde{V} \end{bmatrix} = \begin{bmatrix} T & \epsilon & \epsilon & \epsilon \\ \epsilon & T & \epsilon & \epsilon \\ \epsilon & \epsilon & \times & \times \\ \epsilon & \epsilon & \times & \times \end{bmatrix}$$

Note that the trailing 2×2 submatrix of Q_2 has lost its diagonal form.

Step 4. The trailing 2×2 principal submatrix of Q_2 is well-conditioned since its smallest singular value is greater than $1/\sqrt{2}$. In view of Theorem 3.1 it can therefore be safely diagonalized by column normalization:

$$Q_2 := \begin{bmatrix} I_2 & 0 \\ 0 & \tilde{U}_2 \end{bmatrix}^T Q_2 = \begin{bmatrix} T & \epsilon & \epsilon & \epsilon \\ \epsilon & T & \epsilon & \epsilon \\ \epsilon & \epsilon & L & \epsilon \\ \epsilon & \epsilon & \epsilon & L \end{bmatrix}$$

At this stage, both Q_1 and Q_2 are diagonal.

Step 5. The orthogonal matrices generated in the above computations could, of course, be accumulated. If the ordering (1.1) is desired then it would be necessary to reverse the order of the columns in the accumulated U_1, U_2, and V as well as the order of the c_i and s_i.

It is clear from the above that this method of computing the CSD satisfies (2.2)-(2.6). This is because we only invoke Algorithms 2.1 and 2.2 on well-conditioned submatrices thereby avoiding the pitfalls of §2.

We're now set to specify our method in detail. The notation gets a little cumbersome because we are allowing for rectangular Q_1 and Q_2.

Algorithm 3.1

Given Q_1 $(n_1\times p\ ,\ n_1 \geq p)$ and Q_2 $(n_2\times p)$ satisfying $Q_1^TQ_2 + Q_2^TQ_2 = I_p$, this algorithm overwrites Q_1 and Q_2 with diagonal matrices $C = U_1^TQ_1V$ and $S = U_2^TQ_2V$ respectively where U_1, U_2, and V are each orthogonal. The diagonal entries of C and S are ordered according to (1.1).

Step 1. Compute orthogonal U_2 $(n_2\times n_2)$ and V $(p\times p)$ such that

$$U_2^T\, Q_2\, V = \begin{array}{cc} [\ 0 & \Delta] \\ p-q & q \end{array} \qquad q = \min\{n_2,p\}\ .$$

where

$$\Delta = diag(\delta_1, \ldots, \delta_q)\ .$$

Assume that the δ_i are in ascending order and that the index k is defined by

$$0 \leq \delta_1 \leq \cdots \leq \delta_k \leq \frac{1}{\sqrt{2}} < \delta_{k+1} \leq \cdots \leq \delta_q$$

Update:

$$Q_2 := U_2^TQ_2V$$
$$Q_1 := Q_1V$$

Step 2. Compute an orthogonal U_1 $(n_1 \times n_1)$ such that

$$U_1^T Q_1 = \begin{bmatrix} R \\ 0 \end{bmatrix}$$

where R $(p \times p)$ is upper triangular with positive diagonal entries. Since

$$R^T R = diag(\underbrace{1,\ldots,1}_{p-q}, 1-\delta_1^2, \ldots, 1-\delta_q^2)$$

it follows from Theorem 3.2 and the remarks thereafter that

$$|r_{ij}| \approx \epsilon \, ||R||_2 \qquad i = 1,\ldots,p-q+k \ , \ j = i+1,\ldots,p$$

Thus, after we perform the update

$$Q_1 := U_1^T Q_1$$

we find that Q_1 has the form

$$Q_1 = \begin{array}{cc} \begin{bmatrix} I & 0 & 0 \\ 0 & diag(\gamma_1,\ldots,\gamma_k) & 0 \\ 0 & 0 & R_1 \\ 0 & 0 & 0 \end{bmatrix} & \begin{array}{c} p-q \\ k \\ q-k \\ n_1-p \end{array} \\ \begin{array}{ccc} p-q & \qquad k & \qquad q-k \end{array} & \end{array}$$

where $\gamma_i = \sqrt{1-\delta_i^2}$ for $i = 1,\ldots,k$.

Step 3. Compute orthogonal $\tilde{U}_1$ and $\tilde{V}$ such that

$$\tilde{U}_1^T R_1 \tilde{V} = diag(\gamma_{k+1},\ldots,\gamma_q)$$

and update:

$$\begin{aligned} U_1 &:= U_1 \, diag(I_{p-q+k}, \tilde{U}_1, I_{n_1-p}) \\ V &:= V \, diag(I_{p-q+k}, \tilde{V}) \\ Q_1 &:= diag(I_{p-q+k}, \tilde{U}_1^T, I_{n_1-p}) \, Q_1 \, diag(I_{p-q+k}, \tilde{V}) \\ Q_2 &:= Q_2 \, diag(I_{p-q+k}, \tilde{V}) \end{aligned}$$

Note that at this stage,

$$Q_2 = \begin{array}{cc} \begin{bmatrix} 0 & diag(\delta_1,\ldots,\delta_k) & 0 \\ 0 & 0 & W \\ 0 & 0 & 0 \end{bmatrix} & \begin{array}{c} k \\ q-k \\ n_2-q \end{array} \\ \begin{array}{ccc} p-q & \qquad k & \qquad q-k \end{array} & \end{array}$$

where $W = diag(\delta_{k+1},..,\delta_q) \, \tilde{V}$.

Step 4. Since

$$W^T W = I_{q-k} - diag(\gamma_{k+1}^2, \ldots, \gamma_q^2)$$

and

$$\sigma_{\min}(W) = \delta_{k+1} \geq 1/\sqrt{2}$$

it follows that W can be safely diagonalized via column normalization. Thus, we compute an orthogonal $\tilde{U}_2$ such that $\tilde{U}_2^T W$ is upper triangular and update:

$$Q_2 := diag(I_k , \tilde{U}_2^T , I_{n_2-q})\, Q_2$$
$$U_2 := U_2\, diag(I_k , \tilde{U}_2 , I_{n_2-q})$$

Step 5. At this stage Q_1 and Q_2 have been overwritten by $U_1^T Q_1 V$ and $U_2^T Q_2 V$. Using the conventions in (1.1), these matrices have the form

$$Q_1 = \left[\begin{array}{ccc} c_p & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & c_1 \\ \hline & \bigcirc & \end{array}\right]$$

$$Q_2 = \begin{array}{cc} \left[\begin{array}{ccc|ccc} 0 & \cdots & 0 & s_q & \cdots & 0 \\ \vdots & & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & 0 & \cdots & s_1 \\ \hline & \bigcirc & & & \bigcirc & \end{array}\right] & \begin{array}{c} q \\ \\ n_2 - q \end{array} \\ \quad p-q \qquad\qquad q & \end{array} .$$

Thus, if the ordering (1.1) is desired, then it is necessary to reverse the order of the columns in U_1, U_2 , and V. The c_i and s_i are then the diagonal elements of the suitably permuted Q_1 and Q_2 .

Our choice of $1/\sqrt{2}$ as the dividing line between large and tiny singular values has the effect of minimizing the error bounds in (2.2)-(2.6). One may wish to play with this constant under certain circumstances since the overall amount of work depends on the size of the index k in the first step. For example, if $\sigma \geq .01$ is the definition of a large singular value, then smaller subproblems will result in Steps 2,3, and 4 of the algorithm. This reduces the amount of work, but increases the errors by a factor of about a hundred.

If we apply Algorithm 4.1 to the matrices in (2.7) we find

$$\hat{U}_1 = \begin{bmatrix} -.687726557625 & .545665008317 & -.409531855282 & .248124041805 \\ .335002082175 & .201264772176 & .243430456100 & .887697983245 \\ .554780323465 & .077373737916 & -.828391230010 & .000259642028 \\ -.327146113464 & -.809787314577 & -.294606928957 & .387848788835 \end{bmatrix}$$

$$\hat{U}_2 = \begin{bmatrix} -.937875930622 & .180798315164 & .216282039353 & .202293814616 \\ .274903644962 & .091872622366 & .938148480642 & .189380134924 \\ .207841469770 & .535652344879 & -.269374120814 & .772862259327 \\ -.040232426925 & -.819724317017 & -.023175218383 & .570873282926 \end{bmatrix}$$

$$\hat{V} = \begin{bmatrix} .259105212443 & .781804019979 & -.522164425396 & .221339729993 \\ -.262189089896 & .408991746550 & .702633018050 & .519893714448 \\ .884603850438 & -.238746378084 & .222655254207 & .333017766051 \\ -.285652582355 & -.405596341874 & -.429040548850 & .754863177726 \end{bmatrix}$$

$$\hat{U}_1^T Q_1 \hat{V} = \begin{bmatrix} .000010000000 & .000000000000 & -.000000000001 & .000000000003 \\ .000000000000 & .000020000000 & .000000000001 & -.000000000003 \\ .000000000000 & -.000000000004 & .799999999990 & .000000000001 \\ -.000000000000 & .000000000000 & -.000000000005 & .899999999991 \end{bmatrix}$$

$$\hat{U}_2^T Q_2 \hat{V} = \begin{bmatrix} .999999999937 & -.000000000002 & .000000000001 & .000000000000 \\ .000000000003 & .999999999788 & .000000000000 & -.000000000003 \\ -.000000000000 & .000000000001 & .599999999991 & .000000000001 \\ -.000000000001 & -.000000000000 & -.000000000000 & .435889894348 \end{bmatrix}$$

Since $\hat{U}_1$, $\hat{U}_2$, and $\hat{V}$ are orthogonal to working precision, we see that (2.2)-(2.6) are satisfied.

5. Implementation Details and Discussion.

A Fortran subroutine

$$CS(Q, qdim, n1, n2, p, U1, u1dim, U2, u2dim, V, vdim, c, s, work)$$

has been written that implements Algorithm 4.1. It relies heavily on LINPACK subroutines DQRDC (Housholder QR) and DSVDC (Golub-Reinsch SVD) as well as the BLAS (basic linear algebra subprograms). A few details follow.

The matrices Q_1 and Q_2 are passed to CS via the single array Q. (This is particularly handy in GSVD problems where Q is set up by applying DSVDC to the matrix M in (1.2).) DSVDC is used to compute the SVD required in Step 1 of the algorithm. Since this subroutine orders the singular values from large to small, a re-ordering must be performed.

Next, DQRDC is used to compute the QR factorization in Step 2. The Householder transformations are multiplied together and stored in $U1$.

The main computation in Step 3 is the diagonalization of matrix R_1 via SVD. This could be done via DSVDC. However, the matrix is already close to diagonal form making the 2-sided Jacobi SVD approach more efficient. Jacobi SVD methods are discussed in [3].

The central calculation in Step 4 is the diagonalization of the matrix W via QR. This could be done via DQRDC. However, it is cheaper to generate the orthogonal matrix by merely normalizing the columns of W. Since the smallest singular value of this matrix is bigger than $1/\sqrt{2}$, the resulting triangular form is safely diagonal in view of Theorem 3.1 .

Quantifying the overall amount of work is difficult as it depends upon the dimensions n_1, n_2, and p and the value of the problem dependent index k. Roughly speaking, however, the volume of computation required by Algorithm 4.1 is comparable to a DSVDC call with a matrix of size $(n_1 + n_2) \times p$.

Finally, we mention that we are developing an implementation of Algorithm 4.1 that can be mapped onto a systolic array. Our procedure relies solely upon Jacobi-type transformations and can be implemented on a slight modification of the SVD array proposed in [3]. This important convenience would not be possible if one had to form the cross-product matrix that is required by Stewart's CSD algorithm. Details will be reported elsewhere.

Acknowledgements

The author appreciates the many stimulating conversations that he had with Ira Kaplan and Frank Luk during the preparation of this paper.

References

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

[2] J. Dongarra, C.B. Moler, J.R. Bunch, and G.W. Stewart (1979), *LINPACK User's Guide,* Society for Industrial and Applied Mathematics, Philadelphia,

[3] R. Brent, F. Luk , and C. Van Loan (1982), *Computation of the singular value decomposition using mesh-connected processors* , Cornell Computer Science Technical Report TR 82-528 , Ithaca, New York 14853.

[4] R. Brent, F. Luk, and C. Van Loan (1983), *Computation of the generalized singular value decomposition using mesh-connected processors* , Cornell Computer Science Technical Report TR 83-563, Ithaca, New York 14853.

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

[6] G.H. Golub and C. Van Loan (1983), *Matrix Computations* , Johns Hopkins University Press, Baltimore, Md.

[7] C.B. Moler (1980), *MATLAB User's Guide,* Technical Report CS81-1, Department of Computer Science, University of New Mexico, Albuquerque, New Mexico, 87131.

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

[9] J. Speiser and H.J. Whitehouse (1983), *Techniques for spatial signal processing with systolic arrays,* Proceedings of the Workshop on the Applications of High Resolution Spatial Processing", Gulfport, MI.

[10] G.W. Stewart (1977), *On perturbation of pseudo-inverses, projections, and linear least squares problems,* SIAM Review, 19, 634-662.

[11] G.W. Stewart (1983), *An algorithm for computing the CS decomposition of a partitioned orthonormal matrix* , Numer. Math., 40, 297-306.

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

[13] C. Van Loan (1984), *Analysis of some matrix problems using the CS decomposition,* Cornell Computer Science Technical Report TR84-603, Ithaca, New York 14853.