A PARALLEL METHOD FOR COMPUTING THE GENERALIZED SINGULAR VALUE DECOMPOSITION

Franklin T. Luk

School of Electrical Engineering, Cornell University
Ithaca, New York 14853

Abstract

We describe a new parallel algorithm for computing the generalized singular value decomposition of two $n \times n$ matrices, one of which is nonsingular. Our procedure requires $O(n)$ time and one triangular array of $O(n^2)$ processors.

Introduction

In this paper we describe a linear-time algorithm for computing a generalized singular value decomposition (GSVD) of a partitioned matrix

$$E \equiv \begin{pmatrix} A \\ B \end{pmatrix}. \qquad (1)$$

Only the simple case where A and B are both square $(n \times n)$ and B is nonsingular will be considered.

The GSVD of E is a simultaneous diagonalization of both A and B by two orthogonal matrices U and V and a nonsingular matrix X:

$$U^T AX = D_A \equiv diag\left(\alpha_1, \cdots, \alpha_n\right) \qquad (2)$$

and

$$V^T BX = D_B \equiv diag\left(\beta_1, \cdots, \beta_n\right). \qquad (3)$$

The GSVD is useful for solving various constrained and generalized least squares problems (Golub and Van Loan [6]). In the special case where the columns of E are orthonormal, i.e.,

$$A^T A + B^T B = I,$$

the transformation X may be taken to be orthogonal and the two diagonal matrices will satisfy

$$D_A^T D_A + D_B^T D_B = I;$$

the resulting factorization (2-3) is called a CS-decomposition (CSD). The CSD is useful for analyzing invariant subspace perturbation problems (Davis and Kahan [5], Stewart [13], and Van Loan [16]). If we first compute a singular value decomposition (SVD) of the matrix E:

$$E = U_E D_E V_E^T,$$

where U_E $(2n \times n)$ has orthonormal columns, D_E $(n \times n)$ is diagonal and V_E $(n \times n)$ is orthogonal, and if we then determine a CSD of the matrix U_E, we shall obtain a GSVD of the given matrix E. This approach is often recommended for computing the GSVD (Paige and Saunders [12], Stewart [14] and Van Loan [17]). Indeed, stable CSD algorithms have been derived in Stewart [14] and Van Loan [17] for this purpose. The first direct GSVD procedure is given by Paige [11]. It implicitly applies a Jacobi-SVD algorithm to the matrix $C \equiv AB^{-1}$, and is numerically appealing in that only orthogonal transformations are applied to A and B and that the matrices B^{-1} and C are never explicitly formed.

The advent of real time signal processing has aroused much interest in parallel GSVD algorithms (Bromley and Speiser [4]). Parallel implementations of Van Loan's CSD algorithm are discussed in Kaplan and Van Loan [7] and in Luk and Qiao [10]. While the former paper uses the "parallel" ordering of Brent and Luk [1], the latter one chooses an "odd-even" ordering that is due to Stewart [15]. A systolic array implementation of Stewart's CSD algorithm is sketched in Brent, Luk and Van Loan [2]. Paige's GSVD procedure is not amenable to parallel computations as it uses a cyclic-by-rows ordering. In this paper we modify his algorithm to adopt the "odd-even" ordering. Besides a parallel implementation, our new algorithm is easier to program and understand. It can be implemented on a triangular processor array of Luk [9]: with $O(n^2)$ processors the time requirement for a GSVD is only $O(n)$. Since this processor array also computes in linear time the QR-decomposition [9], the SVD [9] and the CSD [10], it satisfies many of the the computational needs of real time signal processing [4].

Jacobi-SVD Algorithms

Jacobi-like SVD procedures for square matrices are first proposed by Kogbetliantz [8]. Their implementation on systolic arrays is discussed in Brent et al.[3] and Luk [9]. The basic tool is a 2×2 plane rotation:

$$P(\alpha) \equiv \begin{pmatrix} \cos\alpha & \sin\alpha \\ -\sin\alpha & \cos\alpha \end{pmatrix},$$

as the basic problem concerns the diagonalization of a 2×2 matrix by the rotations $J(\theta)$ and $K(\phi)$:

$$J(\theta)^T \begin{pmatrix} w & x \\ y & z \end{pmatrix} K(\phi) = \begin{pmatrix} d_1 & 0 \\ 0 & d_2 \end{pmatrix}. \qquad (4)$$

A two-stage procedure for finding θ and ϕ is advocated in Brent et al.[3]. First, find a rotation $S(\psi)$ to symmetrize the 2×2 matrix:

$$S(\psi)^T \begin{pmatrix} w & x \\ y & z \end{pmatrix} = \begin{pmatrix} p & q \\ q & r \end{pmatrix}.$$

If $x=y$ set $\psi=0$, otherwise compute

$$\rho = \frac{w+z}{x-y} \equiv \mathrm{ctn}\psi,$$
$$\sin\psi = \mathrm{sign}(\rho) / \sqrt{1+\rho^2},$$
$$\cos\psi = \rho \sin\psi.$$

Second, diagonalize the result:

$$K(\phi)^T \begin{pmatrix} p & q \\ q & r \end{pmatrix} K(\phi) = \begin{pmatrix} d_1 & 0 \\ 0 & d_2 \end{pmatrix}.$$

Suppose $q \neq 0$ (else choose either $\phi=0$ or $\phi=\pi/2$). It is well known that $t \equiv \tan\phi$ satisfies the quadratic equation:

$$t^2 + 2\rho t - 1 = 0, \qquad (5)$$

where

$$\rho = \frac{r-p}{2q} \equiv \mathrm{ctn}2\phi.$$

The two solutions to (5) are

$$t = \mathrm{sign}(\rho) / [\,|\rho| + \sqrt{1+\rho^2}\,],$$
$$\cos\phi = 1 / \sqrt{1+t^2}, \qquad (6)$$
$$\sin\phi = t \cos\phi$$

and

$$t = -\mathrm{sign}(\rho) [\,|\rho| + \sqrt{1+\rho^2}\,],$$
$$\cos\phi = 1 / \sqrt{1+t^2}, \qquad (7)$$
$$\sin\phi = t \cos\phi.$$

The rotation $J(\theta)$ is given by

$$J(\theta)^T = K(\phi)^T S(\psi)^T \quad (\text{i.e.}, \theta = \phi + \psi).$$

The angle ϕ associated with (6) is the smaller of the two possibilities; it satisfies $0 \leqslant |\phi| < \pi/4$. The angle associated with (7) satisfies $\pi/4 \leqslant |\phi| < \pi/2$. We refer to a rotation through the smaller angle as an "inner rotation" and one through the larger angle as an "outer rotation" (see Stewart [15]). For a given matrix that is diagonal ($x=y=0$) an "inner rotation" means $\phi=0$ and an "outer rotation" implies $\phi=\pi/2$. In the former case the matrix stays unchanged, and in the latter the singular values are interchanged:

$$\begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} w & 0 \\ 0 & z \end{pmatrix} \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} = \begin{pmatrix} z & 0 \\ 0 & w \end{pmatrix}.$$

While "inner rotations" are usually chosen for a (presumably) faster rate of convergence, "outer rotations" have found acceptance as an integral part of many parallel algorithms [9,10,15].

An SVD of an $n\times n$ matrix A is computed by solving an appropriate sequence of 2×2 SVD problems. The basic Jacobi transformation is

$$T_{ij} \;:\; A \leftarrow J_{ij}^T \, A \, K_{ij}, \qquad (8)$$

where J_{ij} and K_{ij} are rotations in the (i,j) plane chosen to annihilate the (i,j) and (j,i) elements of A. If we define

$$\mathit{off}(C) \equiv \sum_{p\neq q} c_{pq}^2 \quad \text{for } C = (c_{pq}),$$

the transformation T_{ij} will produce a matrix B satisfying

$$\mathit{off}(B) = \mathit{off}(A) - a_{ij}^2 - a_{ji}^2.$$

That is, the matrix B has become more "diagonal" than A. The value of (i,j) is determined according to some ordering, to be selected such that all the off-diagonal elements will be annihilated once in any group of $n(n-1)/2$ rotations (called a sweep). Choosing the well known cyclic-by-rows ordering, we obtain Kogbetliantz's method [8]:

Algorithm SVD1.
do until convergence
  for $i = 1, 2, \cdots, n-1$ do
    for $j = i+1, i+2, \cdots, n$ do
      $A \leftarrow J_{ij}^T A K_{ij}$. □

By convergence we mean that the parameter $\mathit{off}(A)$ has fallen below some pre-selected tolerance. In the settings of parallel computations, it is difficult to monitor $\mathit{off}(A)$ and we may decide to stop iterations after a sufficiently large number (say ten) of sweeps.

A square SVD processor array implementing the "parallel" ordering of Brent and Luk [1] is described in Brent et al.[3], and a triangular SVD processor array for rectangular matrices is given in

Luk [9]. The triangular array implements a two-stage algorithm. First, a QR-decomposition is computed of the given matrix as it is fed into the array. Second, a Jacobi-SVD method based on the "odd-even" ordering of Stewart [15] is applied to the resultant triangular form. "Outer rotations" must be used to ensure convergence. Here is the SVD algorithm in Luk [9] for an upper triangular A:

Algorithm SVD2.

do until convergence
  begin
  { "outer rotations" are used }
  for $i = 1, 3, \cdots$ (i odd) do
    $A \leftarrow J_{i,i+1}^T \, A \, K_{i,i+1}$;
  for $i = 2, 4, \cdots$ (i even) do
    $A \leftarrow J_{i,i+1}^T \, A \, K_{i,i+1}$
  end. □

Details on the two SVD arrays are given in the three papers [1,3,9]. Important points worth emphasizing are that only nearest neighbor connections are required of the $O(n^2)$ processors, that broadcasting can be avoided through a staggering of computations, and that one sweep of the associated SVD algorithm is implementable in time $O(n)$.

Extensive numerical experiments have been performed with various Jacobi-SVD methods [3,9]. The rate of convergence was at least quadratic, and only eight or fewer sweeps were required for $n \leqslant 200$. The SVD of an $n \times n$ matrix is thus computable in effectively linear time.

A Jacobi-GSVD Method

We describe here the novel GSVD algorithm of Paige [11]. Recall that the matrices A and B are square, and that B is nonsingular. In addition, assume both matrices to be upper triangular (do two preparatory QR-decompositions if necessary). Orthogonal transformations U, V and Q are to be determined such that the two resulting matrices $U^T AQ$ and $V^T BQ$ have parallel rows, i.e.,

$$U^T AQ = D \cdot V^T BQ \,, \qquad (9)$$

where D is some diagonal matrix. Defining the nonsingular matrix $X \equiv B^{-1}V$, we get the desired GSVD:

$$\begin{aligned} V^T BX &= I, \\ U^T AX &= U^T AQ \cdot Q^T X \\ &= D \cdot V^T BQ \cdot Q^T B^{-1} V \\ &= D. \end{aligned}$$

On the other hand, note that

$$U^T (AB^{-1}) V = D. \qquad (10)$$

So the transformations U and V can be obtained via an SVD procedure applied to $C \equiv AB^{-1}$. The gist of Paige's method lies in its implicit application of Algorithm SVD1 to C without explicitly forming the matrices B^{-1} and C.

The paper [11] discusses in detail the effect of Algorithm SVD1 on a triangular matrix. It describes how an initially upper (respectively lower) triangular matrix will gradually lose its structure and become lower (respectively upper) triangular. For an initially upper triangular matrix C, sweep #1 of Algorithm SVD1 will make it lower triangular, sweep #2 will return it to upper triangular form, and so on. Let us examine how Paige takes advantage of this structural transformation. Consider a transformation in the (i,j) plane and denote by M_{ij} the 2×2 matrix formed by the intersection of rows i, j and columns i, j of an $n \times n$ matrix M. With a judicious choice of the third orthogonal transformation Q (to be described), Paige asserts that

$$C_{ij} = A_{ij}(B^{-1})_{ij} \quad , \; (B^{-1})_{ij} = (B_{ij})^{-1}. \quad (11)$$

Property (11) is very important. It means that we need not compute the matrices B^{-1} and C, for the submatrices A_{ij} and B_{ij} are sufficient for generating the rotations U and V for the 2×2 SVD:

$$U^T C_{ij} V = S \,,$$

where S is diagonal. Then

$$U^T A_{ij} = S \cdot V^T B_{ij} \,,$$

i.e., the first (resp. second) row of $U^T A_{ij}$ is parallel to the first (resp. second) row of $V^T B_{ij}$. Thus, if another rotation Q is chosen so that $V^T B_{ij} Q$ is lower (upper) triangular, then so is $U^T A_{ij} Q$ (the numerical aspects of this mathematical relation are not investigated in Paige [11]). Paige claims that A_{ij} and B_{ij} always assume the same triangular structure, and he chooses Q to reduce both $U^T A_{ij} Q$ and $V^T B_{ij} Q$ to lower (resp. upper) triangular forms for given matrices A_{ij} and B_{ij} that are upper (resp. lower) triangular. With

$$A \equiv (a_{ij}),\ B \equiv (b_{ij}),\ C \equiv (c_{ij}),$$

and letting U_{ij}, V_{ij} and Q_{ij} denote $n \times n$ rotations in the (i,j)-plane (admittedly our notations are not completely satisfactory), we present Paige's algorithm:

Algorithm GSVD1.

do until convergence

for $i = 1, 2, \cdots, n-1$ do

for $j = i+1, i+2, \cdots, n$ do

begin

determine U_{ij} and V_{ij} to annihilate c_{ij} and c_{ji} ;

$A \leftarrow U_{ij}^T A; \quad B \leftarrow V_{ij}^T B;$

if $a_{ji} = b_{ji} = 0$

find Q_{ij} to zero out a_{ij} and b_{ij}

else $\{ a_{ij} = b_{ij} = 0 \}$

find Q_{ij} to zero out a_{ji} and b_{ji} ;

$A \leftarrow AQ_{ij}; \quad B \leftarrow BQ_{ij}$

end. □

By convergence it is meant that the rows of A and B have become parallel according to some predetermined measure.

A Parallel Implementation

In this section we modify Algorithm GSVD1 for parallel computations by adopting the "odd-even" ordering. An extra dividend is that the upper triangular structures of both A and B can be preserved. Now, if both A and B are upper triangular, then so are the matrices B^{-1} and $C \equiv AB^{-1}$. As such, the two enjoy these special relations:

$$(B^{-1})_{i,i+1} = (B_{i,i+1})^{-1},$$
$$C_{i,i+1} = A_{i,i+1}(B^{-1})_{i,i+1}.$$

Unlike Paige [11], here the nonsingularity of $B_{i,i+1}$ follows trivially from the nonsingularity and the upper triangular structure of B. We have thus proved

$$C_{i,i+1} = A_{i,i+1}(B_{i,i+1})^{-1}, \tag{12}$$

the key condition for an implicit application of Algorithm SVD2 to the upper triangular matrix C. We find rotations U and V for a 2×2 SVD:

$$U^T C_{i,i+1} V = S,$$

where S is diagonal. Then

$$U^T A_{i,i+1} = S \cdot V^T B_{i,i+1},$$

i.e., the two rows of $U^T A_{i,i+1}$ and $V^T B_{i,i+1}$ are parallel. We can thus find one rotation Q to (upper-)triangularize both matrices (cf. Paige [11]).

Let us study how the aforementioned transformations affect the two $n \times n$ upper triangular matrices A and B. We have

$$A \leftarrow U_{i,i+1}^T \, A \, Q_{i,i+1},$$
$$B \leftarrow V_{i,i+1}^T \, B \, Q_{i,i+1},$$

where $U_{i,i+1}$, $V_{i,i+1}$ and $Q_{i,i+1}$ denote appropriate $n \times n$ rotations in the $(i, i+1)$-plane. Note that both matrices $U_{i,i+1}^T A$ and $V_{i,i+1}^T B$ have only one non-zero subdiagonal element each, in the $(i+1, i)$-position. These two extraneous elements are annihilated by the same rotation $Q_{i,i+1}$, that restores both A and B to triangular forms. Here is our new GSVD algorithm for upper triangular A and B:

Algorithm GSVD2.

do until convergence

for $i = 1, 3, \cdots (i$ odd$), 2, 4, \cdots (i$ even$)$ do

begin

$\{ U_{i,i+1}$ and $V_{i,i+1}$ are "outer rotations" $\}$

determine $U_{i,i+1}$ and $V_{i,i+1}$ to annihilate $c_{i,i+1}$ and $c_{i+1,i}$;

$A \leftarrow U_{i,i+1}^T A; \quad B \leftarrow V_{i,i+1}^T B;$

find $Q_{i,i+1}$ to zero out $a_{i+1,i}$ and $b_{i+1,i}$;

$A \leftarrow AQ_{i,i+1}; \quad B \leftarrow BQ_{i,i+1}$

end. □

Algorithm GSVD2 is easily implementable on the triangular QRD-SVD array of Luk [9]. We compute initial QR-decompositions of both A and B as they are fed into the array. The SVD of $C_{i,i+1}$ and the triangularization of both $A_{i,i+1}$ and $B_{i,i+1}$ are performed in parallel on the processor array in a straightforward manner [9,10]. A significant fact is that one sweep of Algorithm GSVD2 can be completed in linear time.

An Example

We present an example, generated on a VAX-11/780 at Cornell University using MATLAB with an effective precision $\epsilon = 10^{-10}$. The initial matrices were

$$A = \begin{bmatrix} 1.30761 & -0.67658 & -0.84935 & -0.36462 \\ 0. & -0.58724 & 0.57963 & 0.66229 \\ 0. & 0. & 0.61666 & 0.00997 \\ 0. & 0. & 0. & -0.54019 \end{bmatrix}$$

and

$$B = \begin{bmatrix} 1.13645 & -0.73578 & 0.41857 & 0.03362 \\ 0. & -0.90705 & 0.00905 & -0.39844 \\ 0. & 0. & 0.30468 & 0.03115 \\ 0. & 0. & 0. & -0.05234 \end{bmatrix}.$$

Here the numbers are shown to only five decimal places due to a lack of space. To exhibit the quadratic rate of convergence, we show here the matrices $C \equiv AB^{-1}$, computed at the end of the 0th, 1st and 2nd sweeps, respectively:

$$\begin{bmatrix} 1.15060 & -0.18743 & -4.36279 & 6.53551 \\ 0. & 0.64742 & 1.88320 & -16.46078 \\ 0. & 0. & 2.02395 & 1.01400 \\ 0. & 0. & 0. & 10.32035 \end{bmatrix},$$

$$\begin{bmatrix} 20.46505 & 3.01675 & 0.08291 & -1.24184 \\ 0.00000 & 4.44366 & 0.00000 & 0.09526 \\ 0.00000 & 0.00000 & 0.39817 & -0.31057 \\ 0.00000 & 0.00000 & 0.00000 & 0.42972 \end{bmatrix},$$

$$\begin{bmatrix} 0.59714 & 0.00114 & -0.00002 & -0.00005 \\ 0.00000 & 0.28588 & -0.00000 & 0.00001 \\ -0.00000 & -0.00000 & 4.39602 & 0.00813 \\ -0.00000 & -0.00000 & 0.00000 & 20.73402 \end{bmatrix}.$$

After three sweeps, we got a diagonal matrix (to ten decimal places):

$$C = diag(20.73402, 4.39602, 0.28588, 0.59715),$$

and the two original matrices became

$$A = \begin{bmatrix} 0.96805 & 0.81599 & 0.19584 & 0.00235 \\ -0.00000 & 1.24946 & 0.05270 & -0.86169 \\ 0.00000 & 0.00000 & 0.23512 & -0.03943 \\ -0.00000 & 0.00000 & 0.00000 & 0.89943 \end{bmatrix}$$

and

$$B = \begin{bmatrix} 0.04669 & 0.03935 & 0.00945 & 0.00011 \\ 0.00000 & 0.28423 & 0.01199 & -0.19602 \\ -0.00000 & -0.00000 & 0.82246 & -0.13793 \\ 0.00000 & 0.00000 & -0.00000 & 1.50622 \end{bmatrix}.$$

Acknowledgements

The author would like to thank D. Boley and S. Qiao for many valuable discussions.

References

[1] R.P. Brent and F.T. Luk, *The solution of singular-value and symmetric eigenvalue problems on multiprocessor arrays*, SIAM J. Sci. Statist. Comput., 6 (1985), pp. 69-84.

[2] R.P. Brent, F.T. Luk, and C. Van Loan, *Computation of the generalized singular value decomposition using mesh-connected processors*, Proc. SPIE Vol. 431, Real Time Signal Processing VI (1983), pp. 66-71.

[3] R.P. Brent, F.T. Luk, and C. Van Loan, *Computation of the singular value decomposition using mesh-connected processors*, J. VLSI and Computer Systems, 1 (1985), to appear.

[4] K. Bromley and J.M. Speiser, *Signal Processing Algorithms, Architectures, and Applications*, Tutorial 31, SPIE 27th Annual Internat. Tech. Symp., San Diego, 1983.

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

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

[7] I.M. Kaplan and C. Van Loan, *On computing the CS decomposition with systolic arrays*, Tech. Report TR84-647, Computer Science Dept., Cornell Univ., 1984.

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

[9] F.T. Luk, *A triangular processor array for computing the singular value decomposition*, Tech. Report TR84-625, Computer Science Dept., Cornell Univ., 1984.

[10] F.T. Luk and S. Qiao, *A linear time method for computing the CS-decomposition*, Tech. Report EE-CEG-84-6, School of Electrical Engineering, Cornell Univ., 1984.

[11] C.C. Paige, *Computing the generalized singular value decomposition*, SIAM J. Sci. Statist. Comput., submitted for publication.

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

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

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

[15] G.W. Stewart, *A Jacobi-like algorithm for computing the Schur decomposition of a non-Hermitian matrix*, SIAM J. Sci. Statist. Comput., 6 (1985), to appear.

[16] C. Van Loan, *Analysis of some matrix problems using the CS decomposition*, Tech. Report TR84-603, Computer Science Dept., Cornell Univ., 1984.

[17] C. Van Loan, *Computing the CS and the generalized singular value decompositions*, Tech. Report TR84-614, Computer Science Dept., Cornell Univ., 1984.