SIAM J. MATRIX ANAL. APPL.
Vol. 12, No. 4, pp. 713–725, October 1991

009

REDUCING THE COMPUTATIONS OF THE SINGULAR VALUE DECOMPOSITION ARRAY GIVEN BY BRENT AND LUK*

B. YANG† AND J. F. BÖHME†

Abstract. A new, efficient, two-plane rotation (TPR) method for computing two-sided rotations involved in singular value decomposition (SVD) is presented. It is shown that a two-sided rotation can be evaluated by only two plane rotations and a few additions. This leads to significantly reduced computations. Moreover, if coordinate rotation digital computer (CORDIC) processors are used for realizing the processing elements (PEs) of the SVD array given by Brent and Luk, the computational overhead of the diagonal PEs due to angle calculations can be avoided. The resulting SVD array has a homogeneous structure with identical diagonal and off-diagonal PEs. Similar results can also be obtained if the TPR method is applied to Luk's triangular SVD array and to Stewart's Schur decomposition array.

Key words. singular value decomposition, systolic arrays, CORDIC, two-sided rotations, VLSI

AMS(MOS) subject classification. 15A18

1. Introduction. One important problem in linear algebra and digital signal processing is the singular value decomposition (SVD). Typical applications arise in beamforming and direction finding, spectrum analysis, digital image processing, etc. [1]. Recently, there has been a massive interest in parallel architectures for computing SVD because of the high computational complexity of SVD, the growing importance of real-time signal processing, and the rapid advances in very large scale integration (VLSI) that make low-cost, high-density and fast processing memory devices available.

There are different numerically stable methods for computing complete singular value and singular vector systems of dense matrices, for example, the Jacobi SVD method, the QR method, and the one-sided Hestenes method. For parallel implementations, the Jacobi SVD method is far superior in terms of simplicity, regularity, and local communications. Brent, Luk, and Van Loan have shown how the Jacobi SVD method with parallel ordering can be implemented by a two-dimensional systolic array [2], [3]. Various coordinate rotation digital computer (CORDIC) realizations of the SVD array have been reported by Cavallaro and Luk [4] and Delosme [5], [6].

The Jacobi SVD method is based on, as common for all two-sided approaches, applying a sequence of two-sided rotations to 2×2 submatrices of the original matrix. The computational complexity is thus determined by how to compute the two-sided rotations. In most previous works, a two-sided rotation is evaluated in a straightforward manner by four plane rotations, where two of them are applied from left to the two column vectors of the 2×2 submatrix and the other ones are applied from right to the row vectors, respectively. In the diagonal processing elements (PEs), additional operations for calculating rotation angles are required. This leads to an inhomogeneous array architecture containing two different types of PEs.

In this paper, we develop a two-plane rotation (TPR) method for computing two-sided rotations. We show that the above computational complexity can be reduced significantly because each two-sided rotation can be evaluated by only two plane rotations and a few additions. Moreover, the SVD array given by Brent and Luk becomes homogeneous with identical diagonal and off-diagonal PEs when CORDIC processors are

* Received by the editors September 28, 1989; accepted for publication (in revised form) August 2, 1990.
† Department of Electrical Engineering, Ruhr-Universität Bochum, 4630 Bochum, Germany.

used. In a recent work [6], Delosme has also indicated this possibility in connection with "rough rotations" independently. He has taken, however, a different approach that is based on encoding the rotation angles. He has still required four plane rotations on the off-diagonal PEs while diagonal and off-diagonal operations can be overlapped.

Our paper is organized as follows. In § 2, we briefly reexamine Jacobi's SVD method and Brent and Luk's SVD array. Then, we develop the TPR method in § 3. The CORDIC algorithm is described in § 4, where in particular CORDIC scaling correction techniques are discussed and examples of scaling-corrected CORDIC sequences are given. In § 5, a unified CORDIC SVD module for all PEs of the SVD array is presented. This module is compared to those proposed by Cavallaro, Luk, and Delosme in § 6. Finally, we stress the applicability of the TPR method to several other problems.

2. Jacobi SVD method. In this paper, we consider real, square, and nonsymmetric matrices. Let $M \in \mathbb{R}^{N\times N}$ be a matrix of dimension N. The SVD is given by

$$M = U\Sigma V^T, \tag{1}$$

where $U \in \mathbb{R}^{N\times N}$ and $V \in \mathbb{R}^{N\times N}$ are orthogonal matrices containing the left and right singular vectors, and $\Sigma \in \mathbb{R}^{N\times N}$ is a diagonal matrix of singular values, respectively. The superscript T denotes matrix transpose. Based on an extension of the Jacobi eigenvalue algorithm [7], Kogbetliantz [8] and Forsythe and Henrici [9] proposed to diagonalize M by a sequence of two-sided rotations,

$$M_0 = M, \qquad M_{k+1} = U_k^T M_k V_k \qquad (k = 0, 1, 2, \cdots). \tag{2}$$

U_k and V_k describe two rotations in the (i, j)-plane $(1 \leqq i < j \leqq N)$, where the rotation angles are chosen to annihilate the elements of M_k at the positions (i, j) and (j, i). Usually, several sweeps are necessary to complete the SVD, where a sweep is a sequence of $N(N-1)/2$ two-sided rotations according to a special ordering of the $N(N-1)/2$ different index pairs (i, j).

For sequential computing on a uniprocessor system, possibly the most frequently used orderings are the cyclic orderings, namely, the cyclic row ordering

$$(i,j) = (1,2),(1,3),\cdots,(1,N),(2,3),\cdots,(2,N),\cdots,(N-1,N) \tag{3}$$

or the equivalent cyclic column ordering. Sameh [10] and Schwiegelshohn and Thiele [11] have shown how to implement the cyclic row ordering on a ring-connected or a mesh-connected processor array. Recently, a variety of parallel orderings have been developed. Luk and Park [12] have shown that these parallel orderings are essentially equivalent to the cyclic orderings and thus share the same convergence properties.

Brent and Luk have suggested a particular parallel ordering and developed a square systolic array consisting of $\lceil N/2\rceil \times \lceil N/2\rceil$ PEs for implementing the Jacobi SVD method (Fig. 1). To do this, the matrix M is partitioned into 2×2 submatrices. Each PE contains one submatrix and performs a two-sided rotation

$$B = R(\theta_1)^T A R(\theta_2), \tag{4}$$

where

$$A = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \quad \text{and} \quad B = \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix} \tag{5}$$

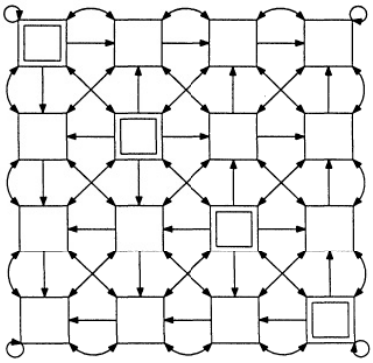

FIG. 1. *The* SVD *array given by Brent and Luk*.

denote the submatrix before and after the two-sided rotation, respectively, and

$$R(\theta)=\begin{pmatrix}\cos\theta & \sin\theta\\ -\sin\theta & \cos\theta\end{pmatrix} \tag{6}$$

describes a plane rotation through the angle θ. At first, the diagonal PEs (symbolized by a double square in Fig. 1) generate the rotation angles to diagonalize the 2×2 submatrices ($b_{12}=b_{21}=0$) stored in them. This means that θ_1 and θ_2 are first calculated from the elements of A and then relation (4) is used to compute b_{11} and b_{22}. We call this the generation mode. Then, the rotation angles are sent to all off-diagonal PEs in the following way: the angles associated to the left-side rotations propagate along the rows while the angles associated to the right-side rotations propagate along the columns. Once these angles are received, the off-diagonal PEs perform the two-sided rotations (4) on their stored data. We call this the rotation mode. Clearly, if we compute the rotation mode straightforwardly, we require four plane rotations. For the generation mode, additional operations for calculating θ_1 and θ_2 are required.

3. TPR method for computing two-sided rotations. In order to develop the TPR method for computing two-sided rotations more efficiently, we first discuss the commutative properties of two special types, the rotation-type and the reflection-type, of 2×2 matrices. We define

$$\mathcal{M}^{\mathrm{rot}}=\left\{\begin{pmatrix}x & y\\ -y & x\end{pmatrix}\middle| x,y\in\mathbb{R}\right\}\quad\text{and}\quad\mathcal{M}^{\mathrm{ref}}=\left\{\begin{pmatrix}x & y\\ y & -x\end{pmatrix}\middle| x,y\in\mathbb{R}\right\}. \tag{7}$$

The former is called rotation-type because it has the same matrix structure as a 2×2 plane rotation matrix. Similarly, the latter is called reflection-type because it has the same matrix structure as a 2×2 Givens reflection matrix [13]. Note that x and y must not be normalized to $x^2+y^2=1$. Using the above definitions, the following results can be shown by some elementary manipulations.

LEMMA 1. *If* $A_1\in\mathcal{M}^{\mathrm{rot}}$ *and* $A_2\in\mathcal{M}^{\mathrm{rot}}$, *then* $A_1A_2=A_2A_1\in\mathcal{M}^{\mathrm{rot}}$.

LEMMA 2. *If* $A_1\in\mathcal{M}^{\mathrm{ref}}$ *and* $A_2\in\mathcal{M}^{\mathrm{rot}}$, *then* $A_1A_2=A_2^TA_1\in\mathcal{M}^{\mathrm{ref}}$.

In particular, if we consider two plane rotations, we know the following.

LEMMA 3. *If* $R(\theta_1)$ *and* $R(\theta_2)$ *are plane rotations described by* (6), *then* $R(\theta_1)R(\theta_2)=R(\theta_1+\theta_2)$ *and* $R(\theta_1)^TR(\theta_2)=R(\theta_2-\theta_1)$.

Now, we give a theorem describing the rotation mode of the TPR method.

THEOREM. *If the* 2×2 *matrix* A *and the two rotation angles* θ_1 *and* θ_2 *are given, then the two-sided rotation* (4) *can be computed by two plane rotations, ten additions,*

and four scalings by $\frac{1}{2}$:

$$p_1 = (a_{22} + a_{11})/2, \qquad p_2 = (a_{22} - a_{11})/2, \tag{8}$$
$$q_1 = (a_{21} - a_{12})/2, \qquad q_2 = (a_{21} + a_{12})/2,$$

$$\theta_- = \theta_2 - \theta_1, \qquad \theta_+ = \theta_2 + \theta_1, \tag{9}$$

$$\begin{pmatrix} r_1 \\ t_1 \end{pmatrix} = R(\theta_-)\begin{pmatrix} p_1 \\ q_1 \end{pmatrix}, \qquad \begin{pmatrix} r_2 \\ t_2 \end{pmatrix} = R(\theta_+)\begin{pmatrix} p_2 \\ q_2 \end{pmatrix}, \tag{10}$$

$$b_{11} = r_1 - r_2, \qquad b_{12} = -t_1 + t_2, \tag{11}$$
$$b_{21} = t_1 + t_2, \qquad b_{22} = r_1 + r_2.$$

Proof. Using (8), the matrix A can be reformulated as

$$A = A_1 + A_2 = \begin{pmatrix} p_1 & -q_1 \\ q_1 & p_1 \end{pmatrix} + \begin{pmatrix} -p_2 & q_2 \\ q_2 & p_2 \end{pmatrix}.$$

Clearly, $R(\theta_1)$, $R(\theta_2)$ in (4) and A_1 are elements of $\mathcal{M}^{\text{rot}}$ while A_2 belongs to $\mathcal{M}^{\text{ref}}$. This leads to the following reformulation of the matrix B by using Lemmas 1–3:

$$\begin{aligned} B &= R(\theta_1)^T A R(\theta_2) \\ &= R(\theta_1)^T A_1 R(\theta_2) + R(\theta_1)^T A_2 R(\theta_2) \\ &= R(\theta_1)^T R(\theta_2) A_1 + R(\theta_1)^T R(\theta_2)^T A_2 \\ &= R(\theta_2 - \theta_1) A_1 + R(\theta_2 + \theta_1)^T A_2 \\ &= R(\theta_-)\begin{pmatrix} p_1 & -q_1 \\ q_1 & p_1 \end{pmatrix} + R(\theta_+)^T \begin{pmatrix} -p_2 & q_2 \\ q_2 & p_2 \end{pmatrix} \\ &= \begin{pmatrix} r_1 & -t_1 \\ t_1 & r_1 \end{pmatrix} + \begin{pmatrix} -r_2 & t_2 \\ t_2 & r_2 \end{pmatrix}. \end{aligned}$$

This completes the proof.

The generation mode of the TPR method follows directly from the above theorem.

COROLLARY. *If the* 2×2 *matrix* A *is given, we can diagonalize* A *and calculate the corresponding rotation angles* θ_1 *and* θ_2 *by two Cartesian-to-polar coordinates conversions, eight additions, and four scalings by* $\frac{1}{2}$:

$$p_1 = (a_{22} + a_{11})/2, \qquad p_2 = (a_{22} - a_{11})/2, \tag{12}$$
$$q_1 = (a_{21} - a_{12})/2, \qquad q_2 = (a_{21} + a_{12})/2,$$

$$r_1 = \text{sign}\,(p_1)\sqrt{p_1^2 + q_1^2}, \qquad r_2 = \text{sign}\,(p_2)\sqrt{p_2^2 + q_2^2}, \tag{13}$$
$$\theta_- = \arctan\,(q_1/p_1), \qquad \theta_+ = \arctan\,(q_2/p_2),$$

$$\theta_1 = (\theta_+ - \theta_-)/2, \qquad \theta_2 = (\theta_+ + \theta_-)/2, \tag{14}$$

$$b_{11} = r_1 - r_2, \qquad b_{22} = r_1 + r_2. \tag{15}$$

Proof. Regarding (11), $b_{12} = b_{21} = 0$ is equivalent to $t_1 = t_2 = 0$. Equation (13) follows then from (10). This completes the proof.

In equation (13), we choose the rotation through the smaller angle. All vectors lying in the first or the fourth quadrant are rotated onto the positive x-axis, and all vectors lying in the second and the third quadrant are rotated onto the negative x-axis. For vectors on the y-axis, the rotation direction is arbitrary. Thus, the generated rotation

angles θ_- and θ_+ satisfy $|\theta_-|, |\theta_+| \leqq 90°$. This results in

$$|\theta_1| \leqq 90° \quad \text{and} \quad |\theta_2| \leqq 90°, \tag{16}$$

due to (14).

Equation (16) is important with respect to the convergence of the Jacobi SVD method. Forsythe and Henrici [9] have proven the convergence for cyclic orderings if the rotation angles θ_1 and θ_2 are restricted to a closed interval inside the open interval $(-90°, 90°)$. They have also demonstrated that this condition may fail to hold, i.e., θ_1 and θ_2 may be $\pm 90°$, if the off-diagonal elements b_{12} and b_{21} in (5) have to be exactly annihilated. As a remedy, they suggested an under- or overrotation by computing the two-sided rotation (4) with angles $(1 - \gamma)\theta_1$ and $(1 - \gamma)\theta_2$ $(-1 < \gamma < 1)$ and proved its convergence. In practice, however, the finite machine accuracy in the real arithmetic allows only an approximative computation of the rotation angles and implies under- or overrotations. So the Jacobi SVD method converges without using under- or overrotations as shown by the experimental results of Brent, Luk, and Van Loan [3]. In case of CORDIC implementations, the effect of implicit under- or overrotations is more apparent. The angles $\pm 90°$ can never be exactly calculated because of the limited angle resolution $\arctan(2^{-p})$ of the CORDIC algorithm, where p denotes the mantissa length.

4. The CORDIC algorithm. In the previous section, we have seen that the main operations of the TPR-method are plane rotations and Cartesian-to-polar coordinates conversions. These operations can be carried out by multiplier–adder-based processors supported by software or special hardware units. An alternative approach is the use of dedicated processors that usually map algorithms more effectively to hardware. The CORDIC processor is such a powerful one for calculating trigonometric functions.

The CORDIC algorithm was originally designed by Volder [14] as an iterative procedure for computing plane rotations and Cartesian-to-polar coordinates conversions. It was later generalized and unified by Walther [15], enabling a CORDIC processor to calculate more functions, including hyperbolic functions, as well as multiplications and divisions. In the following, we consider Volder's CORDIC algorithm because only trigonometric functions are involved in SVD applications.

The CORDIC algorithm consists of iterative shift-add operations on a three-component vector,

$$\begin{pmatrix} x_{i+1} \\ y_{i+1} \end{pmatrix} = \begin{pmatrix} x_i - \sigma_i \delta_i y_i \\ y_i + \sigma_i \delta_i x_i \end{pmatrix} = \frac{1}{\cos(\alpha_i)} \begin{pmatrix} \cos(\alpha_i) & -\sigma_i \sin(\alpha_i) \\ \sigma_i \sin(\alpha_i) & \cos(\alpha_i) \end{pmatrix} \begin{pmatrix} x_i \\ y_i \end{pmatrix}, \tag{17}$$

$$z_{i+1} = z_i - \varepsilon \sigma_i \alpha_i \qquad (0 < \delta_i < 1;\ \sigma_i = \pm 1;\ \varepsilon = \pm 1;\ i = 0, 1, \cdots, n-1), \tag{18}$$

in which the iteration stepsize δ_i is defined by

$$\delta_i = \tan(\alpha_i) = 2^{-S(i)}. \tag{19}$$

The set of integers $\{S(i)\}$ parametrizing the iterations is called CORDIC sequence. Equation (17) can be interpreted, except for a scaling factor of

$$k_i = \frac{1}{\cos(\alpha_i)} = \sqrt{1 + \delta_i^2}, \tag{20}$$

as a rotation of $(x_i, y_i)^T$ through the angle α_i, where the sign $\sigma_i = \pm 1$ gives the rotation direction. After n iterations, the results are given by

$$\begin{pmatrix} x_n \\ y_n \end{pmatrix} = K \begin{pmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{pmatrix} \begin{pmatrix} x_0 \\ y_0 \end{pmatrix}, \tag{21}$$

$$z_n = z_0 - \varepsilon\alpha, \tag{22}$$

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.