Reducing the computations of the SVD array given by Brent and Luk

B. Yang and J.F. Böhme

Ruhr-Universität, Department of Electrical Engineering
4630 Bochum, West Germany

ABSTRACT

A new, efficient two plane rotations (TPR) method for computing two-sided rotations involved in singular value decomposition (SVD) is presented. By exploiting the commutative properties of some special types of 2x2 matrices, we show that a two-sided rotation can be computed by only two plane rotations and a few additions. Moreover, if we use coordinate rotation digital computer (CORDIC) processors to implement 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.

1. INTRODUCTION

One important problem in linear algebra and digital signal processing is the SVD. Particular applications arise in beamforming and direction finding, spectrum analysis, digital image processing etc.[1] Recently, there is a massive interest in parallel architectures for computing SVD due to the growing importance of real-time signal processing and advances in VLSI devices. Brent and Luk[2,3] have shown how a Jacobi method with parallel ordering can efficiently compute the SVD, and how the method can be implemented by a two-dimensional systolic array. The method is based on, as common for all two-sided approaches, applying a sequence of two-sided rotations to 2x2 submatrices of the original matrix. The computational complexity is thus determined by how to compute the two-sided rotations.

Usually, a two-sided rotation is realized by four plane rotations, where two of them are applied from left to the two column vectors of the 2x2 submatrix and the other ones are applied from right to the row vectors, respectively. For the diagonal PEs of the SVD array, additional operations for calculating the rotation angles are required. This leads to an inhomogeneous array architecture containing two different types of PEs. In this paper, we develop a new TPR method for computing two-sided rotations. We show that the above computational complexity is reduced significantly, and the SVD array becomes homogeneous when using CORDIC processors.

The paper is organized as follows. In section 2, we briefly re-examine the Jacobi SVD method and the SVD array. Then, we develop the TPR method in section 3. The CORDIC algorithm is described in section 4. Different techniques for scaling correction are discussed, and examples of scaling corrected CORDIC sequences for different data formats are given. In section 5, an unified CORDIC implementation of all PEs

of the SVD array is presented. Finally, some convergence aspects are discussed in section 6.

2. JACOBI SVD METHOD

The SVD of a matrix $M \in \mathbb{R}^{NxN}$ is given by

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

where $U \in \mathbb{R}^{NxN}$ and $V \in \mathbb{R}^{NxN}$ are orthogonal matrices and $\Sigma \in \mathbb{R}^{NxN}$ is a diagonal matrix of singular values. Based on an extension of the Jacobi eigenvalue algorithm, Kogbetliantz[4], Forsythe and Henrici[5] 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 represent rotations in the (i,j)-plane ($1 \le i < j \le N$). 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). Unfortunately, the best known cyclic orderings, namely, the cyclic row ordering

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

and the equivalent cyclic column ordering are not suitable for parallel computations.

Recently, Brent and Luk[2,3] suggested a parallel ordering allowing a high degree of parallelism. It enables the use of a square systolic array with $\lceil N/2 \rceil x \lceil N/2 \rceil$ PEs to implement the Jacobi SVD method (Figure 1). For doing this, the matrix M is partitioned into 2x2 submatrices. Each PE contains one submatrix and performs a two-sided rotation

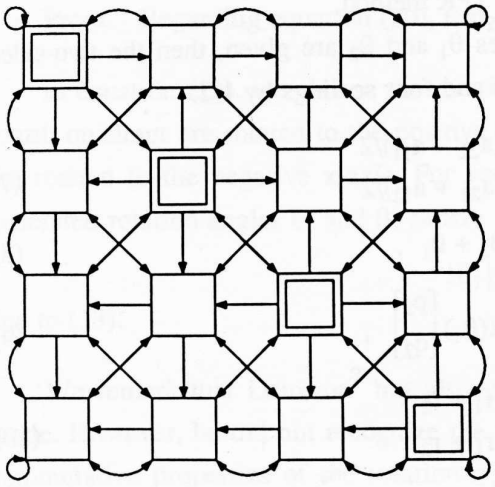

Figure 1. The SVD array given by Brent and Luk

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

where

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

denote the input and output matrices, and

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

describes a plane rotation through the angle θ, respectively. Notice that there are two different computation modes. In a diagonal PE, the rotation angles θ_1 and θ_2 are generated to diagonalize the 2x2 submatrix ($b_{12}=b_{21}=0$) stored. We call this the generation

mode. Then, the rotation angles are propagated to all off-diagonal PEs in the same row and the same column, in which pure rotations (4) with the received angles are performed. We call this the rotation mode. Clearly, if we directly compute (4) in the rotation mode, we require four plane rotations. For the generation mode, additional operations for calculating θ_1 and θ_2 are needed.

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 2x2 matrices, namely,

$$\mathcal{M}^{\text{rot}} = \left\{ \begin{bmatrix} c & s \\ -s & c \end{bmatrix} \;\middle|\; c,s\in\mathbb{R} \right\} \quad \text{and} \quad \mathcal{M}^{\text{ref}} = \left\{ \begin{bmatrix} c & s \\ s & -c \end{bmatrix} \;\middle|\; c,s\in\mathbb{R} \right\}.$$

Obviously, the plane rotation matrix and the Givens reflection matrix[6] with $c^2+s^2=1$ are two elements of the sets $\mathcal{M}^{\text{rot}}$ and $\mathcal{M}^{\text{ref}}$, respectively. The following results can be shown by elementary manipulations.

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

Lemma 2. If $A_1\in\mathcal{M}^{\text{ref}}$ and $A_2\in\mathcal{M}^{\text{rot}}$, then $A_1A_2=A_2^{\text{T}}A_1\in\mathcal{M}^{\text{ref}}$.

In particular, if we consider two plane rotations, we know,

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 $R(\theta_1)^{\text{T}}R(\theta_2)=R(\theta_2-\theta_1)$.

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

Theorem. If the 2x2 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 1/2:

$$\begin{cases} p_1 = (a_{22}+a_{11})/2 \\ q_1 = (a_{21}-a_{12})/2 \end{cases}, \qquad \begin{cases} p_2 = (a_{22}-a_{11})/2 \\ q_2 = (a_{21}+a_{12})/2 \end{cases}, \tag{7}$$

$$\theta_- = \theta_2-\theta_1\,, \qquad \theta_+ = \theta_2+\theta_1\,, \tag{8}$$

$$\begin{bmatrix} r_1 \\ t_1 \end{bmatrix} = R(\theta_-)\begin{bmatrix} p_1 \\ q_1 \end{bmatrix}, \qquad \begin{bmatrix} r_2 \\ t_2 \end{bmatrix} = R(\theta_+)\begin{bmatrix} p_2 \\ q_2 \end{bmatrix}, \tag{9}$$

$$\begin{cases} b_{11} = r_1-r_2 \\ b_{21} = t_1+t_2 \end{cases}, \qquad \begin{cases} b_{12} = -t_1+t_2 \\ b_{22} = r_1+r_2 \end{cases}. \tag{10}$$

Proof. Using (7), the input matrix A can be reformulated by

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

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 expression for the output matrix B by using the lemmas 1–3,

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

This completes the proof.

A direct consequence of the above theorem shows how a two-sided rotation in the generation mode can be computed in a similar way.

Corollary. If the 2x2 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 1/2:

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

$$\begin{cases} r_1 = \text{sign}(p_1)\sqrt{p_1^2 + q_1^2} \\ \theta_- = \arctan(q_1/p_1) \end{cases}, \qquad \begin{cases} r_2 = \text{sign}(p_2)\sqrt{p_2^2 + q_2^2} \\ \theta_+ = \arctan(q_2/p_2) \end{cases}, \tag{12}$$

$$\theta_1 = (\theta_+ - \theta_-)/2 , \qquad \theta_2 = (\theta_+ + \theta_-)/2 , \tag{13}$$

$$b_{11} = r_1 - r_2 , \qquad b_{22} = r_1 + r_2 . \tag{14}$$

Proof. Regarding equation (10), $t_1=t_2=0$ follows directly from $b_{12}=b_{21}=0$.

In equation (12), we choose the rotation through the smaller angle. All vectors lying in the first or the fourth quadrant are rotated to the positive x-axis, and all vectors lying in the second and the third quadrant are rotated to the negative x-axis. For vectors on the y-axis, the rotation direction is arbitrary. Thus, the generated rotation angles θ_- and θ_+ satisfy $|\theta_-|, |\theta_+| \le \pi/2$. This results in

$$|\theta_1| \le \pi/2 \quad \text{and} \quad |\theta_2| \le \pi/2 \tag{15}$$

due to (13).

We remark that Delosme[7] has also proposed to use (14) for computing b_{11} and b_{22} in the generation mode. However, he did not recognize the fact that the above approach is generally possible because of the commutative properties of the rotation-type and the reflection-type of matrices. So, he still requires four plane rotations for computing a two-sided rotation in the rotation mode.

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. Although these operations can be carried out by multiplier-

adder based processors supported by software or special hardware units, a simpler approach is the use of dedicated processors that map algorithms more effectively to hardware. The CORDIC processor is a powerful one for calculating these functions.

The CORDIC algorithm was originally designed by Volder[8] as an iterative procedure for computing plane rotations and Cartesian to polar coordinates conversions. It was later generalized and unified by Walther[9] enabling a CORDIC processor to calculate more functions, including hyperbolic functions as well as multiplications and divisions, in a similar way. In the following, only Volder's CORDIC algorithm is considered for the sake of simplicity because only trigonometric functions are involved in SVD applications.

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

$$\begin{bmatrix} x_{i+1} \\ y_{i+1} \end{bmatrix} = \begin{bmatrix} x_i - \sigma_i \delta_i y_i \\ y_i + \sigma_i \delta_i x_i \end{bmatrix} = \frac{1}{\cos(\alpha_i)} \begin{bmatrix} \cos(\alpha_i) & -\sigma_i \sin(\alpha_i) \\ \sigma_i \sin(\alpha_i) & \cos(\alpha_i) \end{bmatrix} \begin{bmatrix} x_i \\ y_i \end{bmatrix} \tag{16}$$

$$z_{i+1} = z_i - \varepsilon \sigma_i \alpha_i \qquad (i=0,1,\ldots,n-1)\ , \tag{17}$$

in which the iteration stepsize $0<\delta_i<1$ is defined by

$$\delta_i = \tan(\alpha_i) = 2^{-S(i)}\ . \tag{18}$$

The set of integers $\{S(i)\}$ is called CORDIC sequence. Equation (16) can be interpreted, except for a scaling factor of

$$k_i = \frac{1}{\cos(\alpha_i)} = \sqrt{1+\delta_i^2}\ , \tag{19}$$

as a rotation of $(x_i, y_i)^T$ through the angle α_i, where $\sigma_i=\pm 1$ gives the rotation direction. After n iterations (16) and (17), the resulting data are given by

$$\begin{bmatrix} x_n \\ y_n \end{bmatrix} = K \begin{bmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{bmatrix} \begin{bmatrix} x_0 \\ y_0 \end{bmatrix}\ , \tag{20}$$

$$z_n = z_0 - \varepsilon\alpha \tag{21}$$

with the overall scaling factor $K = \prod_i k_i$ and the cumulative rotation angle $\alpha = \sum_i \sigma_i \alpha_i$. Now, if the CORDIC sequence satisfies the following convergence condition

$$\alpha_i - \sum_{j=i+1}^{n-1} \alpha_j \le \alpha_{n-1} \qquad (i=0,1,\ldots,n-2)\ , \tag{22}$$

we can choose $\sigma_i=-\text{sign}(x_i y_i)$ or $\sigma_i=\text{sign}(\varepsilon z_i)$ to force y_n or z_n to zero, provided that the input data x_0, y_0 and z_0 lie in the convergence region

$$C = \sum_{i=0}^{n-1} \alpha_i \ge \begin{cases} |\arctan(y_0/x_0)| & \text{for } y_n \to 0 \\ |z_0| & \text{for } z_n \to 0 \end{cases}\ . \tag{23}$$

Both, plane rotations as well as Cartesian to polar coordinates conversions can be calculated in this way

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.