Journal of VLSI Signal Processing, 2, 5–15 (1996)

SVD–Updating Using Orthonormal μ–Rotations

JÜRGEN GÖTZE* **AND PETER RIEDER**

Institute of Network Theory and Circuit Design, Technical University of Munich, D–80290 Munich, Germany

GERBEN J. HEKSTRA

Department of Electrical Engineering, Delft University of Technology, NL–2628 CD Delft, The Netherlands

Received September 30, 1995. Revised April 15, 1996.

Editors: Kung Yao and Flavio Lorenzelli

Abstract. In this paper the implementation of the SVD–updating algorithm using orthonormal μ–rotations is presented. An orthonormal μ–rotation is a rotation by an angle of a given set of μ–rotation angles (e.g. the angles $\Phi_i = \arctan 2^{-i}$) which are choosen such that the rotation can be implemented by a small amount of shift–add operations. A version of the SVD–updating algorithm is used where all computations are entirely based on the evaluation and application of orthonormal rotations. Therefore, in this form the SVD–updating algorithm is amenable to an implementation using orthonormal μ–rotations, i.e., each rotation executed in the SVD–updating algorithm will be approximated by orthonormal μ–rotations. For all the approximations the same accuracy is used, i.e., only $r \ll w$ (w: wordlength) orthonormal μ–rotations are used to approximate the exact rotation. The rotation evaluation can also be performed by the execution of μ–rotations such that the complete SVD–updating algorithm can be expressed in terms of orthonormal μ–rotations. Simulations show the efficiency of the SVD–updating algorithm based on orthonormal μ–rotations.

Keywords: singular value decomposition (SVD), SVD–updating, CORDIC, orthonormal μ–rotations.

1. Introduction

By computing the singular value decomposition (SVD) of an $m \times n$ data matrix it is possible to extract the signal and noise subspaces of the data. The knowledge of these subspaces is essential in many applications, e.g. DOA–estimation [22], state–space system identification [19], communication [1]. In practice, where the problems are usually time varying, it is important to be able to track these subspaces. Therefore, in recent years various subspace tracking algorithms have been proposed. These algorithms are based on rank revealing decompositions [4], [23], the Lanczos algorithm [5] or the SVD–updating algorithm [20].

* This work was done while with Rice University, Houston, Texas supported by the Alexander von Humbodt Foundation and Texas Advanced Technology Program

The SVD–updating algorithm incorporates a new data vector by a matrix vector multiplication, a QRD–updating step and a fraction of a sweep of Kogbetliantz's SVD algorithm [20], [9]. These types of computations are well suited for parallel implementations and it has been shown in [21], that all the computations can nicely be combined resulting in a systolic algorithm and architecture for SVD–updating.

However, this original version of the SVD–updating algorithm [20] exhibits numerical problems because round–off error accumulation destroys the orthogonality of the singular vectors. Reorthogonalization steps can avoid this problem but are not appropriate for a systolic implementation. In [25] this problem was solved by using the parameterization of the orthonormal matrix of right singular vectors by $n(n \Leftrightarrow 1)/2$ plane rotations and updating the respective rotation angles. In this form the SVD–updating algorithm is en-

tirely based on the evaluation and application of plane rotations.

These plane rotations can be implemented in the standard way using square roots and divisions or transcendental functions. In order to simplify the implementation various possibilities for the modification of orthogonal plane rotations have been presented, e.g., square root free or square root and division free rotations (see e.g. [17]). With respect to an efficient ASIC (application specific integrated circuit) implementation another widely used method is the CORDIC algorithm [26], [27], i.e., representing the rotation angle Φ in the basis "$\arctan 2^{-i}$" (we assume a finite wordlength of w bits):

$$\Phi = \sum_{i=0}^{w} \tau_i \Phi_i \qquad (1)$$

where $\Phi_i = \arctan 2^{-i}$ form the basis angles and $\tau_i \in \{+1, \Leftrightarrow 1\}$ are the digits of the representation. The CORDIC rotation by an angle Φ can be executed by $w+1$ recursions each consisting of two shift–add operations and a scaling procedure (this scaling can also be executed by shift–add operations).

It has been shown in [15], [11] that the CORDIC idea can be applied to obtain approximate rotations, i.e. a rotation by an approximate rotation angle $\tilde{\Phi} \approx \Phi$. Let $\Phi_i = \arctan 2^{-i}$, $i \in \mathcal{I} = \{0, 1, \ldots, w\}$, be the set of possible rotation angles (a rotation by one specific Φ_i is defined as an orthonormal μ–rotation [13], [14]), an r–level approximation of the exact rotation angle Φ is $\tilde{\Phi} = \Phi_{i_1} + \Phi_{i_2} + \ldots + \Phi_{i_r}$, where $i_1 > i_2 > \ldots > i_r$ and $i_s \in \mathcal{I}$. This corresponds to a representation of the approximate rotation angle $\tilde{\Phi}$ as follows:

$$\tilde{\Phi} = \sum_{s=1}^{r} \tau_{i_s} \Phi_{i_s} \quad \text{where } \tau_{i_s} \in \{+1, \Leftrightarrow 1\}. \qquad (2)$$

This representation only consists of specific angles of the complete CORDIC sequence ($i = 0, 1, \ldots, w$). Therefore in contrast to the CORDIC algorithm, the specific i_s must be determined. But on the other hand only the really necessary rotations of the CORDIC sequence are used (compare for example the two representations (1) and (2) for a small angle Φ). In [13] a method was derived that allows the evaluation of the optimal μ–rotation angle (i.e. $\tilde{\Phi} = \Phi_{i_1}$) using μ–rotations as well. The μ–rotation angles $i_2, i_3, \ldots$ can be determined by an iterative application of the same procedure [15]. An elementary architecture for evaluating and applying the orthonormal μ–rotations was presented in [14].

It has been shown in [12] that the use of approximate rotations is worthwile in order to avoid square root computations or square root and division computations without degrading the performance of the SVD–updating algorithm. In this paper we demonstrate the efficiency of approximate rotations based on orthonormal μ–rotations for the SVD–updating algorithm. This requires the extension of the ideas presented in [15] to the SVD, i.e., the use of orthonormal μ–rotation is discussed for Kogbetliantz's SVD algorithm and applied to the SVD–updating algorithm as given in [25]. Note, that only this numerically stable version of the SVD–updating algorithm is appropriate with respect to an implementation of the entire algorithm based on orthonormal μ–rotations, since only this version is entirely based on the evaluation and application of plane rotations. Simulations show that very coarse approximations, i.e., using $r \ll w$ orthonormal μ–rotations (throughout our examples we use $r = 1$) per plane rotation works as well as using exact rotations (i.e. $r = w$ for the exact CORDIC).

In section 2 we present some preliminaries. First of all the definition of orthogonal plane rotations and their implementation using CORDIC is given. Then, the linear algebra algorithms composing the SVD–updating algorithm are reviewed, i.e., the QRD–updating and the SVD using Kogbetliantz's algorithm. Section 3 presents the SVD–updating algorithm as given in [25] requiring only applications and evaluations of plane rotations throughout the algorithm. In section 4 we describe the computation of the orthonormal μ–rotations for the 2×1 QRD and the 2×2 SVD subproblems. It is shown how the evaluation of the optimal orthonormal μ–rotations for a QRD subproblem can be referred to the execution of μ–rotations. The orthonormal μ–rotations for the SVD subproblem can be evaluated using the orthonormal μ–rotations determined for two inde-

pendent QRD subproblems. The presented procedure for evaluating the orthonormal μ–rotations improves the methods presented in [11], [16] significantly. Approximate rotations based on orthonormal μ–rotations are applied to the SVD–updating algorithm in section 5. In section 6 simulations show the efficiency of the presented algorithm and section 7 concludes the paper.

2. Preliminaries

In this section the matrix decompositions (QRD,SVD) required for the SVD–updating algorithm are defined and their computation as appropriate for the SVD–updating algorithm (QRD-updating and computation of the SVD using Kogbetliantz's algorithm) are reviewed. The discussed versions of these algorithms are entirely based on the evaluation and application of orthonormal plane rotations. The implementation of these orthonormal plane rotations using CORDIC is also discussed.

2.1. Orthonormal Rotations

DEFINITION 1 *[Orthonormal Plane Rotation] An orthonormal plane rotation (Givens rotation)* $\boldsymbol{G}_{pq}(\Phi) \in R^{n\times n}$ *is defined by the rotation angle* Φ *and the* (p,q)*–plane in which the rotation takes place, i.e., the embedding of* $\cos\Phi$ *and* $\sin\Phi$ *in the* (pp, pq, qp, qq) *positions of a* $n \times n$ *identity matrix.* $\boldsymbol{G}_{pq}(\Phi)$ *is an orthonormal rotation, since* $\boldsymbol{G}_{pq}^T(\Phi)\boldsymbol{G}_{pq}(\Phi) = \boldsymbol{I}$.

Without loss of generality we will only consider in detail the evaluation and application of 2×2 orthonormal rotations.

$$\begin{aligned}\boldsymbol{G}(\Phi) &= \begin{bmatrix} \cos\Phi & \sin\Phi \\ \Leftrightarrow\sin\Phi & \cos\Phi \end{bmatrix} = \\ &= \frac{1}{\sqrt{1+\tan^2\Phi}} \begin{bmatrix} 1 & \tan\Phi \\ \Leftrightarrow\tan\Phi & 1 \end{bmatrix} \end{aligned} \quad (3)$$

in the following.

2.2. CORDIC

The CORDIC procedure [26], [27] uses the representation (1) for the rotation angle Φ. Therefore, $\tan\Phi_i = 2^{-i}$ holds for the basis angles such that with (3) one obtains the CORDIC rotation

$$\boldsymbol{G}(\Phi) = \frac{1}{K_w} \prod_{i=0}^{w} \begin{bmatrix} 1 & \tau_i 2^{-k} \\ \Leftrightarrow\tau_i 2^{-k} & 1 \end{bmatrix} \quad (4)$$

where the scaling factor $\frac{1}{K_w}$ is independent of the rotation angle:

$$\frac{1}{K_w} = \prod_{i=0}^{w} \frac{1}{\sqrt{1+2^{-2i}}}\,. \quad (5)$$

There have been different efforts to eliminate the scaling factor or at least to bring it into a simple binary representation. Delosme [6] proposed a method for computing a variable scaling factor on–line. This case arises for variable iteration bounds in (4). Instead of working with the basis angles Φ_i the basis angles are obtained by twice executing a rotation by Φ_{i+1}. The respective double rotation $\boldsymbol{G}_d(\bar{\Phi})$ is given by

$$\boldsymbol{G}_d(\bar{\Phi}) = \frac{1}{K_w^2} \prod_{i=1}^{w+1} \begin{bmatrix} 1 \Leftrightarrow 2^{-2i} & \tau_i 2^{-i+1} \\ \Leftrightarrow\tau_i 2^{-i+1} & 1 \Leftrightarrow 2^{-2i} \end{bmatrix} \quad (6)$$

The basis angles of the double rotation are given by

$$\bar{\Phi}_i = \arctan\frac{2^{-i+1}}{1 \Leftrightarrow 2^{-2i}} = 2\cdot\Phi_{i+1}\,.$$

Now, four (instead of two) shift–add operations are required per recursion step but the scaling factor is square root free. To avoid the division, in the scaling factor, too, the following simple identity can be used:

$$\begin{aligned}&\frac{1}{1+2^{-2k}} = \\ &\quad = (1 \Leftrightarrow 2^{-2k})(1+2^{-4k})(1+2^{-8k})\cdots\end{aligned} \quad (7)$$

We will elaborate this further when approximate rotations are discussed.

2.3. QR–Decomposition

DEFINITION 2 *[QR–decomposition] The QR-decomposition of a matrix* $\boldsymbol{X} \in R^{m\times n}(m \geq n)$ *is defined by*

$$\boldsymbol{X} = \boldsymbol{Q}\begin{bmatrix}\boldsymbol{R}\\ \boldsymbol{O}\end{bmatrix}, \tag{8}$$

where $\boldsymbol{Q} \in R^{m\times m}$ *is orthonormal* ($\boldsymbol{Q}^T\boldsymbol{Q} = \boldsymbol{I}$) *and* $\boldsymbol{R} \in R^{n\times n}$ *is upper triangular.*

2×1 QRD subproblem: A vector $[x, y]^T$ is rotated by the angle Φ using

$$\begin{bmatrix}x'\\ y'\end{bmatrix} = \boldsymbol{G}(\Phi)\begin{bmatrix}x\\ y\end{bmatrix}. \tag{9}$$

Computing Φ such that $y' = 0$ solves the 2×1 QRD subproblem, i.e., compute

$$\Phi = \arctan\frac{y}{x}. \tag{10}$$

Computing the QRD: The triangular matrix $\boldsymbol{R}$ is obtained by solving a sequence of 2×1 QRD subproblems, i.e., applying a sequence of orthonormal rotations $\boldsymbol{G}_{pq}(\Phi)$ to the matrices $\boldsymbol{X}$ (original $\boldsymbol{X}$ overwritten), where $\boldsymbol{G}_{pq}(\Phi)$ annihilates the instantaneous x_{pq} element for $1 < q \leq n$ and $q + 1 < p \leq m$, i.e.,

$$\boldsymbol{X} \leftarrow \boldsymbol{G}_{pq}(\Phi) \cdot \boldsymbol{X}, \tag{11}$$

where $\Phi = \arctan(x_{qp}/x_{pp})$, such that $\boldsymbol{Q} = \prod_{p,q}\boldsymbol{G}_{pq}(\Phi)$ and $\boldsymbol{X}$ is overwritten by $\boldsymbol{R}$.

QRD–updating: An alternative for triangularizing $\boldsymbol{X}$ columnwise is to perform the triangularization row by row. This yields the recursive QRD–updating. Let $\boldsymbol{X}_{[k-1]}$ be the $k \Leftrightarrow 1 \times n$ data matrix available at time step $k \Leftrightarrow 1$ and $\boldsymbol{x}_{[k]}^T$ be the new data vector measured at time step k one obtains

$$\boldsymbol{X}_{[k]} = \begin{bmatrix}\lambda\boldsymbol{X}_{[k-1]}\\ \boldsymbol{x}_{[k]}^T\end{bmatrix}, \tag{12}$$

where λ is the forgetting factor.

Given the QRD of $\boldsymbol{X}_{[k-1]}$

$$\boldsymbol{X}_{[k-1]} = \boldsymbol{Q}_{[k-1]}\begin{bmatrix}\boldsymbol{R}_{[k-1]}\\ \boldsymbol{O}\end{bmatrix}, \tag{13}$$

the upper triangular factor $\boldsymbol{R}_{[k]}$ is obtained by appending the new data vector $\boldsymbol{x}_{[k]}^T$ to the weighted matrix $\lambda\boldsymbol{R}_{[k-1]}$ and using a sequence of Givens rotations $\boldsymbol{G}_{pq}(\Phi)$ $(p = k; 1 \leq q \leq n)$ to annihilate the appended row, i.e.,

$$\begin{bmatrix}\boldsymbol{R}_{[k]}\\ \boldsymbol{0}^T\end{bmatrix} \leftarrow \prod_{q=1}^{n}\boldsymbol{G}_{kq}(\Phi)\begin{bmatrix}\lambda\boldsymbol{R}_{[k-1]}\\ \boldsymbol{x}_{[k]}^T\end{bmatrix}. \tag{14}$$

2.4. Singular Value Decomposition

DEFINITION 3 *[SVD] The SVD of a matrix* $\boldsymbol{X} \in R^{m\times n}$ *is defined by*

$$\boldsymbol{X} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T \tag{15}$$

where $\boldsymbol{U} \in R^{m\times m}$ *and* $\boldsymbol{V} \in R^{n\times n}$ *are orthonormal matrices* ($\boldsymbol{U}^T\boldsymbol{U} = \boldsymbol{I}$, $\boldsymbol{V}^T\boldsymbol{V} = \boldsymbol{I}$) *and* $\boldsymbol{\Sigma} = diag(\sigma_1, \ldots, \sigma_n)$ *is an* $m \times n$ *diagonal matrix containing the singular values* σ_i.

2×2 SVD subproblem: Given a 2×2 matrix $\boldsymbol{A} = \begin{bmatrix}a_{11} & a_{12}\\ a_{21} & a_{22}\end{bmatrix}$ the rotations $\boldsymbol{G}(\Phi^U)$ and $\boldsymbol{G}(\Phi^V)$ are applied to the left and right of $\boldsymbol{A}$:

$$\begin{bmatrix}a'_{11} & a'_{12}\\ a'_{21} & a'_{22}\end{bmatrix} = \boldsymbol{G}(\Phi^U)^T\begin{bmatrix}a_{11} & a_{12}\\ a_{21} & a_{22}\end{bmatrix}\boldsymbol{G}(\Phi^V). \tag{16}$$

Computing Φ^U and Φ^V such that $a'_{12} = 0$ and $a'_{21} = 0$ holds, solves the 2×2 SVD subproblem. It has been shown in [28], [3], [7] that the angles Φ^U and Φ^V can be determined from two angles Φ^R and Φ^S that can be computed independently by solving two 2×1 QRD subproblems. With

$$\begin{aligned} x_1 &= (a_{22} + a_{11})/2;\ \ x_2 = (a_{22} \Leftrightarrow a_{11})/2;\\ y_1 &= (a_{21} \Leftrightarrow a_{12})/2;\ \ y_2 = (a_{21} + a_{12})/2; \end{aligned} \tag{17}$$

we determine the two rotations $\boldsymbol{G}(\Phi^R)$ and $\boldsymbol{G}(\Phi^S)$ such that

$$\begin{bmatrix}x'_1\\ y'_1\end{bmatrix} = \boldsymbol{G}(\Phi^R)\begin{bmatrix}x_1\\ y_1\end{bmatrix} \tag{18}$$

$$\begin{bmatrix}x'_2\\ y'_2\end{bmatrix} = \boldsymbol{G}(\Phi^S)\begin{bmatrix}x_2\\ y_2\end{bmatrix}; \tag{19}$$

yields $y'_1 = 0$ $(\Phi^R = \arctan(y_1/x_1))$ and $y'_2 = 0$ $(\Phi^S = \arctan(y_2/x_2))$. Then using

$$\begin{aligned} \Phi^U &= \frac{1}{2}(\Phi^S \Leftrightarrow \Phi^R) \Leftrightarrow\\ &\Leftrightarrow \boldsymbol{G}(\Phi^U) = \boldsymbol{G}(\Leftrightarrow\Phi^R/2)\boldsymbol{G}(\Phi^S/2) \end{aligned} \tag{20}$$

$$\Phi^V = \frac{1}{2}(\Phi^S + \Phi^R) \Leftrightarrow \\ \Leftrightarrow \boldsymbol{G}(\Phi^V) = \boldsymbol{G}(\Phi^R/2)\boldsymbol{G}(\Phi^S/2) \tag{21}$$

in (16) yields $a'_{12} = a'_{21} = 0$.

Kogbetliantz's SVD algorithm: Without restriction of generality we will only consider the Kogbetliantz SVD algorithm for a square matrix $\boldsymbol{A} \in R^{n\times n}$ (it is well known that it is advantageous to apply Kogbetliantz's SVD algorithm to the upper triangular matrix $\boldsymbol{R}$ obtained by a preparatory QRD). The Kogbetliantz SVD algorithm is given as follows:
$\boldsymbol{V} = \boldsymbol{I}$; $\boldsymbol{U} = \boldsymbol{I}$;
for $l = 0, 1, 2, \ldots$
for all index pairs (p,q)

$$\begin{aligned} \boldsymbol{A} &\leftarrow \boldsymbol{G}^T_{p,q,l}(\Phi^U) \cdot \boldsymbol{A} \cdot \boldsymbol{G}^T_{p,q,l}(\Phi^V) \\ \boldsymbol{V} &\leftarrow \boldsymbol{V} \cdot \boldsymbol{G}_{p,q,l}(\Phi^V) \\ \boldsymbol{U} &\leftarrow \boldsymbol{U} \cdot \boldsymbol{G}_{p,q,l}(\Phi^U) \end{aligned} \tag{22}$$

where $\boldsymbol{G}_{p,q,l}(\Phi^U)$ and $\boldsymbol{G}_{p,q,l}(\Phi^V)$ are the plane rotations in the (p,q)–plane of the l–th iteration. For the index pairs (p,q) a cyclic–by–row ordering scheme is used, i.e.,

$$\begin{aligned} &(p,q) = \\ &= (1,2),(1,3),\ldots,(1,n)(2,3),\ldots,(n \Leftrightarrow 1,n). \end{aligned} \tag{23}$$

This ordering scheme can be mapped to a parallel ordering scheme making the Kogbetliantz algorithm highly suited for parallel implementation [18]. The execution of all $n(n \Leftrightarrow 1)/2$ pairs of (23) is called a sweep (l–th sweep).

The plane rotations $\boldsymbol{G}_{p,q,l}(\Phi^U)$ and $\boldsymbol{G}_{p,q,l}(\Phi^V)$ are obtained by solving the respective (p,q) 2×2 SVD subproblem for each transformation (22). Therefore, the off–diagonal quantity

$$S = \sqrt{\|\boldsymbol{A}\|_F^2 \Leftrightarrow \sum_{i=1}^{n} a_{ii}^2} \tag{24}$$

is reduced by each transformation (22) such that the matrix $\boldsymbol{A}$ converges to a diagonal matrix containing the singular values of $\boldsymbol{A}$ (i.e. $\boldsymbol{\Sigma}$).

2.5. Remarks on Parallel Implementations

The discussed algorithms (QRD, Kogbetliantz's SVD algorithm) are highly suited for parallel implementation. This is essentially because of the representation of the orthogonal matrices in terms of orthonormal plane rotations. This parametrization (representation) of the orthogonal matrices enables the parallel implementation of the algorithms on systolic arrays [10], [2] and also the use of CORDIC arithmetic for the implementation of the processor elements [28] (it is also the essential idea behind avoiding the reorthogonalization steps in the SVD–updating algorithm that is discussed in the next section). It is interesting to note that this parametrization is a result of Euler (1770) (can be found in [8]) at a time when neither parallel implementations nor computer arithmetic was an issue.

3. SVD–Updating Algorithm

The important result of Moonen et. al. [20], [21] was to realize that the parallel implementation of the basic algorithms (QRD–updating, Kogbetliantz's algorithm) can be nicely combined to the SVD–updating algorithm.

The SVD–updating algorithm is based on a matrix vector multiplication, a QRD-updating step, and the computation of the SVD by the Kogbetliantz algorithm. Let $\boldsymbol{X}_{[k-1]} = \boldsymbol{U}_{[k-1]}\boldsymbol{\Sigma}_{[k-1]}\boldsymbol{V}_{[k-1]}$ be the SVD of $\boldsymbol{X}_{[k-1]}$ at time step $k \Leftrightarrow 1$ and $\boldsymbol{x}^T_{[k]}$ be the new data vector. In order to put the QRD–updating and the SVD–computation together it is necessary to project the new data vector $\boldsymbol{x}^T_{[k]}$ to the already computed matrix of right singular vectors $\boldsymbol{V}_{[k-1]}$:

$$\tilde{\boldsymbol{x}}^T_{[k]} \leftarrow \boldsymbol{x}^T_{[k]}\boldsymbol{V}_{[k-1]}\,. \tag{25}$$

Then, the QRD–updating is executed using $\tilde{\boldsymbol{x}}^T_{[k]}$ as the appended vector:

$$\begin{bmatrix} \tilde{\boldsymbol{R}}_{[k]} \\ \boldsymbol{0}^T \end{bmatrix} \leftarrow \prod_{q=1}^{n} \boldsymbol{G}_{kq}(\Phi) \begin{bmatrix} \lambda \boldsymbol{R}_{(k-1)} \\ \tilde{\boldsymbol{x}}^T_{[k]} \end{bmatrix} \tag{26}$$

Now, the SVD of $\tilde{\boldsymbol{R}}_{[k]}$ is computed using Kogbetliantz's algorithm. In order to reduce the complexity of the Kogbetliantz's algorithm it was shown

in [9], [20] that one sweep or even a fraction of a sweep of Kogbetliantz's SVD algorithm is sufficient to track the subspace of sufficiently slow time varying processes. Annihilating only the matrix elements $\tilde{r}_{i,i+1} (i = 1, \ldots, n \Leftrightarrow 1)$ after each update, i.e.

$$\boldsymbol{R}_{[k]} \leftarrow \prod_{i=1}^{n-1} \boldsymbol{G}^T_{i,i+1,[k]}(\Phi^U) \cdot \tilde{\boldsymbol{R}}_{[k]} \cdot \prod_{i=1}^{n-1} \boldsymbol{G}_{i,i+1,[k]}(\Phi^V) \quad (27)$$

$$\boldsymbol{V}_{[k]} \leftarrow \boldsymbol{V}_{[k-1]} \cdot \prod_{i=1}^{n-1} \boldsymbol{G}_{i,i+1,[k]}(\Phi^V) \quad (28)$$

also enables a regular implementation of the SVD–updating on a systolic array [21]. In this form ((25),(26),(27),(28)) the SVD–updating algorithm requires a reorthogonalization of the matrices $\boldsymbol{V}_{[k]}$. This reorthogonalization can be avoided by parameterizing the $\boldsymbol{V}_{[k]}$ in terms of $n(n \Leftrightarrow 1)/2$ orthogonal rotations

$$\boldsymbol{V}_{[k-1]} = \prod_{i=1}^{n} \prod_{j=i+1}^{n} \boldsymbol{G}_{i,j,[k-1]}(\Phi) \quad (29)$$

(e.g. a QRD of $\boldsymbol{V}_{[k-1]}$ yields this factorization) and updating the respective rotation angles by applying the rotations $\prod_{i=1}^{n-1} \boldsymbol{G}_{i,i+1,[k]}(\Phi^V)$ to this factorization [25]. Now, the matrix vector multiplication (25) can be also executed by the application of rotations. Therefore, this form of the SVD–updating algorithm is completely based on the evaluation and application of orthonormal rotations.

4. Orthonormal μ–Rotations

All rotation computations in the SVD–updating algorithm are replaced by approximate rotations in the following, where orthonormal μ–rotations are used as the set of available approximate rotations.

4.1. QRD subproblem

While the execution of an exact rotation as described in (9), (10) guaranees $y' = 0$, an approximate rotation $\boldsymbol{G}(\tilde{\Phi})$ defined by an approximate rotation angle $\tilde{\Phi} \approx \Phi$ only ensures

$$| y' | = | d | \cdot | y | \quad (30)$$

with $0 \leq | d | < 1$.

Suppose we have used an approximate angle $\tilde{\Phi}$, equation (9) yields:

$$y' = \Leftrightarrow \sin\tilde{\Phi} \cdot x + \cos\tilde{\Phi} \cdot y = = (\Leftrightarrow \sin\tilde{\Phi} \cdot \frac{x}{y} + \cos\tilde{\Phi}) \cdot y\,. \quad (31)$$

Representing this equation (31) according to (30) using $\tan\Phi = y/x$ one obtains

$$d(\Phi, \tilde{\Phi}) = \Leftrightarrow \sin\tilde{\Phi} \cdot \frac{1}{\tan\Phi} + \cos\tilde{\Phi}\,. \quad (32)$$

Obviously, for the exact rotation where $\tilde{\Phi} = \Phi$ one obtains $d = 0$.

At this point having defined an approximate rotation we make use of the idea of CORDIC [26], i.e. with respect to a simple implementation of the rotation we restrict ourselves to the set of approximate angles

$$\tilde{\Phi} = \Phi_i = \arctan 2^{-i}, \quad (33)$$

where $i \in \mathcal{I} = \{0, 1, 2, \ldots, w\}$. Therefore, we only allow rotations of the form

$$\boldsymbol{G}(\Phi_i) = \frac{1}{K_i} \boldsymbol{G}_u(\Phi_i) = = \frac{1}{\sqrt{1 + 2^{-2i}}} \begin{bmatrix} 1 & \tau_i 2^{-i} \\ \Leftrightarrow \tau_i 2^{-i} & 1 \end{bmatrix}, \quad (34)$$

where $1/K_i = 1/\sqrt{1 + 2^{-2i}}$ is the scaling factor and $\boldsymbol{G}_u(\Phi_i)$ is an (unscaled) μ–rotation. $\boldsymbol{G}(\Phi_i)$ is called an orthonormal μ–rotation.

Since an orthonormal μ–rotation $\boldsymbol{G}(\Phi_i)$ is defined by the angle index i computing the optimal orthonormal μ–rotation $\boldsymbol{G}(\Phi_{i_o})$ corresponds to finding the CORDIC angle Φ_i $(i \in \mathcal{I})$ which is closest to the exact rotation angle $\Phi = \arctan(y/x)$, i.e.

$$| \Phi_{i_o} \Leftrightarrow \Phi | = \min_{i \in \mathcal{I}} | \Phi_i \Leftrightarrow \Phi | \quad \Rightarrow \quad \tilde{\Phi} = \Phi_{i_o} . \tag{35}$$

The direction of the rotation τ_i is determined by $\tau_i = \text{sign}(x)\text{sign}(y)$. Therefore, having determined τ_i we can work with $x > 0$ ($x \leftarrow |x|$) and $y > 0$ ($y \leftarrow |y|$) to evaluate Φ_{i_o}.

Choosing the optimal approximate angle according to (35) is equivalent to

$$\min_{i \in \mathcal{I}} |d(\Phi, \Phi_i)| \tag{36}$$

This minimization can be done by determing the angle intervals $[\Phi_{gi}, \Phi_{gi+1}]$ such that $\tilde{\Phi} = \Phi_i$ is the optimal angle whenever $\Phi \in [\Phi_{gi}, \Phi_{gi+1}[$. The limits of the intervals Φ_{gi} follow from the solution of

$$d(\Phi_{gi}, \Phi_i) = \Leftrightarrow d(\Phi_{gi}, \Phi_{i+1}) , \tag{37}$$

i.e. the angle Φ_{gi} where choosing Φ_i leads to the same reduction factor $|d|$ as choosing Φ_{i+1}. Solving (37) yields:

$$\tan \Phi_{gi} = \frac{\sin \Phi_i + \sin \Phi_{i+1}}{\cos \Phi_i + \cos \Phi_{i+1}} = = \tan \left(\frac{\Phi_i + \Phi_{i+1}}{2} \right) . \tag{38}$$

As the orthonormal μ–rotation $\boldsymbol{G}(\Phi_i)$ in (34) is defined by one specific recursion step of the original CORDIC sequence (4) we define an orthonormal double μ–rotation by one specific rotation step of the double rotation sequence (6):

$$\boldsymbol{G}_d(\bar{\Phi}_i) = \frac{1}{K_i^2} \boldsymbol{G}_{du}(\bar{\Phi}_i) = = \frac{1}{1 + 2^{-2i}} \begin{bmatrix} 1 \Leftrightarrow 2^{-2i} & \sigma 2^{-i+1} \\ \Leftrightarrow \sigma 2^{-i+1} & 1 \Leftrightarrow 2^{-2i} \end{bmatrix} \tag{39}$$

where $1/K_i^2 = 1/(1 + 2^{-2i})$ is the scaling factor and $\boldsymbol{G}_{du}(i)$ is the (unscaled) double μ–rotation. The scaling factor can be recursively computed by shift–and–add operations [15] (see (7)):

$$1/K_i^2 = (1 \Leftrightarrow 2^{-2i}) \prod_{s=1}^{b} (1 + 2^{-2^{s+1} i}) \tag{40}$$

with $b = \log_2 \lceil \frac{w}{2i} \rceil$. Since the orthonormal double μ–rotations enable an easy scaling factor compensation (40) and since the limits of the intervals can be easily determined for the set of orthonormal double μ–rotations we restrict our set of approximate rotations to the orthonormal double μ–rotations from now on. The limits of the intervals $[\bar{\Phi}_{gi}, \bar{\Phi}_{gi+1}[$ for the choice of the optimal double μ–rotation angle are now given by

$$\tan \bar{\Phi}_{gi} = \tan \left(\frac{\bar{\Phi}_i + \bar{\Phi}_{i+1}}{2} \right) . \tag{41}$$

Therefore, given the vector $\boldsymbol{v} = [x, y]^T$ (i.e. $\tan \Phi = y/x$) we use $\bar{\Phi}_i$ if $\Phi > \bar{\Phi}_{gi}$ and $\bar{\Phi}_{i+1}$ if $\Phi \leq \bar{\Phi}_{gi}$. Since $\bar{\Phi}_{gi} = \bar{\Phi}_i/2 + \bar{\Phi}_{i+1}/2 = \Phi_{i+1} + \Phi_{i+2}$ this decision can be made by using two unscaled μ–rotations $\boldsymbol{G}_u(\Phi_{i+1})$ and $\boldsymbol{G}_u(\Phi_{i+2})$, i.e. compute

$$[x_r, y_r]^T = \boldsymbol{G}_u(\Phi_{i+1}) \boldsymbol{G}_u(\Phi_{i+2}) [x, y]^T . \tag{42}$$

Thereby, we obtain

$$\begin{aligned} y_r > 0 \Leftrightarrow \Phi > \bar{\Phi}_{gi} \quad &\Rightarrow \quad \text{use } \bar{\Phi}_i \\ y_r \leq 0 \Leftrightarrow \Phi \leq \bar{\Phi}_{gi} \quad &\Rightarrow \quad \text{use } \bar{\Phi}_{i+1} . \end{aligned}$$

In order to find the optimal i_o one more unscaled μ–rotation is required. Let $\text{man}(a)$ and $\exp(a)$, respectively, denote the mantissa and the exponent of a binary floating point number a. Since we can obtain an estimate for the optimal i_o by computing $i_e = \exp(y) \Leftrightarrow \exp(x)$ and since $\text{man}(y)/\text{man}(x) \in [0.25, 1[$ one obtains $i_o \in \mathcal{J} = \{i_e, i_e + 1, i_e + 2\}$. Therefore, it is possible to determine the optimal $\bar{\Phi}_{i_o}$ ($i_o \in \mathcal{J}$) as follows. Compute:

$$\begin{aligned} \boldsymbol{v}_0 &= \boldsymbol{G}_u(\Phi_{i+2}) \boldsymbol{v} \\ \boldsymbol{v}_1 &= \boldsymbol{G}_u(\Phi_{i+1}) \boldsymbol{v}_0 \\ \boldsymbol{v}_2 &= \boldsymbol{G}_u(\Phi_{i+3}) \boldsymbol{v}_0 . \end{aligned}$$

Then

$$\bar{\Phi}_{i_o} = \begin{cases} \bar{\Phi}_{i_e} & \text{if } \boldsymbol{v}_1(2) > 0 \\ \bar{\Phi}_{i_e+1} & \text{if } \boldsymbol{v}_2(2) > 0 \\ \bar{\Phi}_{i_e+2} & \text{otherwise} \end{cases}$$

This procedure yields the optimal orthonormal double μ–rotation $\boldsymbol{G}_d(\bar{\Phi}_{i_o})$ such that $y' = d(\Phi, \bar{\Phi}_{i_o}) \cdot y$ with $|d(\Phi, \bar{\Phi}_{i_o})| \leq 1/3$.

4.2. SVD subproblem

The optimal orthonormal double μ–rotations for the two QRD subproblems (18) and (19), i.e. $\boldsymbol{G}_d(\bar{\Phi}^R_{i_o})$ and $\boldsymbol{G}_d(\bar{\Phi}^S_{i_o})$, can be determined by the procedure described above. Given these rotations the SVD rotations are obtained according to (20) and (21). Since $\bar{\Phi}_i \approx 2\bar{\Phi}_{i+1}$ we obtain the approximate rotations for the SVD subproblem as follows:

$$\boldsymbol{G}(\tilde{\Phi}^U) = \boldsymbol{G}_d(\Leftrightarrow\bar{\Phi}^R_{i_o+1})\boldsymbol{G}_d(\bar{\Phi}^S_{i_o+1}) \quad (43)$$

$$\boldsymbol{G}(\tilde{\Phi}^V) = \boldsymbol{G}_d(\bar{\Phi}^R_{i_o+1})\boldsymbol{G}_d(\bar{\Phi}^S_{i_o+1}) \quad (44)$$

By using i_o+1 instead of i_o for the SVD rotations we actually use $\tilde{\Phi}^R = 2\bar{\Phi}^R_{i_o+1}$ and $\tilde{\Phi}^S = 2\bar{\Phi}^S_{i_o+1}$ as the approximate angles for the QRD subproblem (instead of $\tilde{\Phi}^R = \bar{\Phi}^R_{i_o}$ and $\tilde{\Phi}^S = \bar{\Phi}^S_{i_o}$). Therefore, the appximation of the QRD subproblem changes correspondingly and we obtain $y' = d(\Phi, \tilde{\Phi}) \cdot y$ with $|d(\Phi, \tilde{\Phi})| < d_{max} = 0.42$.

It remains to show that $a'^2_{12} + a'^2_{21} \leq d^2_{SVD}(a^2_{12} + a^2_{21})$ with $0 \leq d^2_{SVD} < 1$ is guaranteed for each subproblem, in order to meet the requirements for the convergence of the Kogbetliantz algorithm [12]. For the two QRD subproblems (18) and (19) one obtains

$$y'_1 = d_1 y_1 \qquad \text{where } 0 \leq |d_1| < d_{max} \quad (45)$$

$$y'_2 = d_2 y_2 \qquad \text{where } 0 \leq |d_2| < d_{max} \quad (46)$$

such that [28]

$$a'_{12} = \Leftrightarrow y'_1 + y'_2; \qquad a'_{21} = y'_1 + y'_2. \quad (47)$$

Therefore, with (17) one obtains

$$\begin{aligned} a'^2_{12} + a'^2_{21} &= 2(y'^2_1 + y'^2_2) \\ &< 2d^2_{max}(y^2_1 + y^2_2) \\ &= d^2_{max}(a^2_{12} + a^2_{21}) \end{aligned}$$

such that

$$d^2_{SVD} \leq d^2_{max} = 0.17\,. \quad (48)$$

Furthermore, instead of $\bar{\Phi}_1 = 53.1301^\circ$ we use $90^\circ \Leftrightarrow \bar{\Phi}_1 = 36.8699^\circ$ that guarantees $|\tilde{\Phi}^U|, |\tilde{\Phi}^V| < 90^\circ$ and also leads to an improved approximation of the exact angles. Note that this method improves the reduction factors (and therefore the performance of the entire algorithms) given in [15], [11], [16].

As discussed in [15], [13] it is always possible to improve the accuracy of the approximate rotations by using $r > 1$ orthonormal double μ–rotations to approximate the exact rotation.

5. SVD–Updating Using Orthonormal μ–Rotations

Now, we examine the SVD–updating algorithm for its performance using approximate rotations. Throughout our discussion we assume that only one orthonormal double μ–rotation is used to approximate the exact rotations.

QRD–updating: The use of approximate rotations results in an iterative version of the QR–decomposition [11]. Several sweeps are necessary to achieve a sufficient approximation of the QR–decomposition. Another possibility is the improvement of the accuracy of the orthogonal plane rotation by increasing the number r of orthogonal μ–rotations per plane rotation. Thereby, the number of required sweeps is reduced. In the case of QRD–updating, however, the accuracy of the approximate rotations is not as essential [11]. In the case of the SVD–updating algorithm, where the new data vector is projected to the orthogonal basis $\boldsymbol{V}_{[k-1]}$ (25) the requirements on the accuracy of the rotations is decreased since the projection already maps $\boldsymbol{x}_{[k]}$ approximately into the new basis $\boldsymbol{V}_{[k]}$ (for sufficiently slow time varying processes). This is also the reason for the success of the approximations to the Kogbetliantz algorithm discussed in the next paragraph. In our examples (see section 6) we will use only one orthonormal double μ–rotation to approximate the exact rotations of the QRD–updating. Note, however, that if the time variance increases, the accuracy of the approximate rotations (i.e. of the QRD–updating) can be increased by choosing $r > 1$.

Approximations of Kogbetliantz's SVD algorithm: In [20], [21], [25] only a fraction of one sweep of Kogbetliantz's SVD algorithm is executed per QRD–update without significant deterioration of the tracking performance. Executing only a fraction of one sweep ($n \Leftrightarrow 1$ rotations) al-

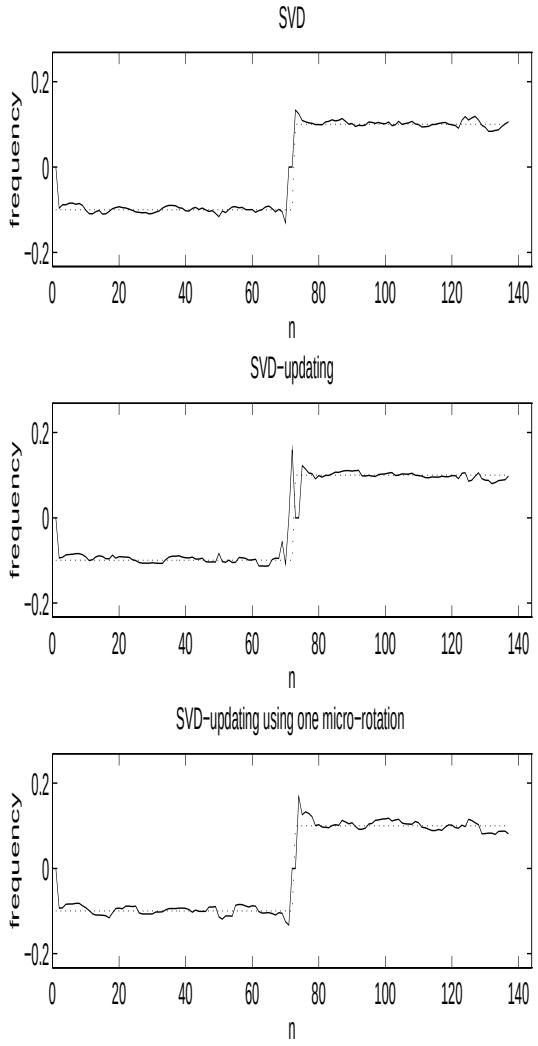

Fig. 1. Frequency estimation for $s_1(t)$ using the exact SVD (top), using the SVD–updating algorithm with exact rotations (middle), and the SVD–updating algorithm based on one orthonormal double μ–rotation (bottom).

ready is a very coarse approximation of the Kogbetliantz algorithm. Using approximate rotations, i.e., annihilating the off–diagonal elements by not completely setting them to zero but only reducing them, is a further degree of approximation brought into the Kogbetliantz algorithm. Compared to the coarse approximation one obtains by executing only a fraction of a sweep, however, the additional degree of approximation added by using approximate rotations is negligible (especially for $d_{SVD}^2 < 0.17 \ll 1$).

Representation and updating of singular vectors: At time step $k \Leftrightarrow 1$ $\boldsymbol{V}_{[k-1]}$ is described by the $n(n \Leftrightarrow 1)/2$ rotations parametrizing the orthonormal matrix $\boldsymbol{V}_{[k-1]}$. In order to get $\boldsymbol{V}_{[k]}$, $\boldsymbol{V}_{[k-1]}$ is updated with the $n \Leftrightarrow 1$ rotations $\prod \boldsymbol{G}_{i,i+1,[k]}(\Phi^V)$, i.e. the rotation angles of the parametrization must be updated. In [25] a method for executing this updating is described. This also enables the execution of the matrix vector multiplication by the application of rotations. Storing each angle in the basis $\bar{\Phi}_i$ according to (2), this angle updating can be easily extended to the use of approximate rotations.

Implementation: One of the strengths of the SVD–updating algorithm is the possibility of implementing it on a systolic array [21]. As shown in [24] there are many equivalent implementations of the SVD–updating algorithm enabling an arbitrary degree of throughput. Using orthonormal μ–rotations instead of exact rotations reduces the overall amount of required shift–add operations (usually $r \ll w$ is sufficient).

6. Simulations

The performance of the algorithms is analysed for two different kinds of signals:

- $s_1(t)$ is a signal where the frequency jumps suddenly
- $s_2(t)$ is a frequency modulated signal

In both cases SNR= 10dB holds. $m = 8$ data samples are used per time step. $n = 140$ sample points of the signals are taken. The frequencies are estimated in each time step using the ESPRIT algorithm based on the significant subspaces as obtained from the singular vectors of $\boldsymbol{V}_{[k]}$ belonging to the dominant singular values. Figure 1 (jump of frequency) and Figure 2 (modulation of frequency) compare the SVD, the SVD–updating, and the SVD–updating using orthonormal μ–rotations. The figures are organized as follows. The dotted lines in each subplot show the exact frequencies at each time step. The solid lines show the estimated frequencies. The subplots at the top of each figure show the estimated frequencies if a complete SVD is executed for each time step (i.e. call MATLAB svd($\boldsymbol{X}_{[k]}$) for each k). The subplots in the middle of the figures show the estimated frequencies using the SVD–updating algorithm with exact rotations. The subplots at

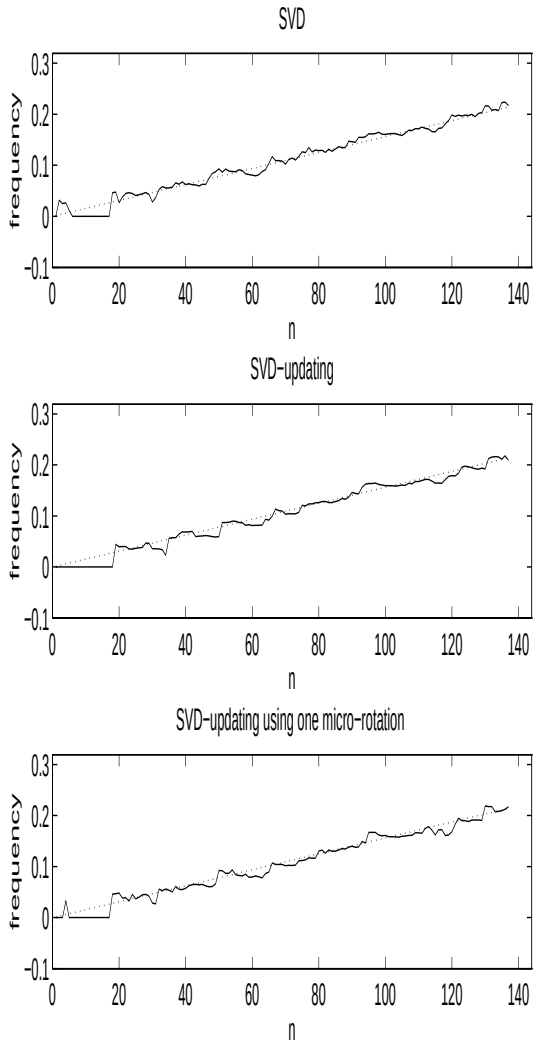

Fig. 2. Frequency estimation for $s_2(t)$ using the exact SVD (top), using the SVD–updating algorithm with exact rotations (middle), and the SVD–updating algorithm based on one orthonormal double μ–rotation (bottom).

the bottom of each figure show the estimated frequencies if the SVD–updating algorithm uses only one orthonormal double μ–rotation to approximate the exact rotations of the QRD subproblems (the approximate SVD rotations are obtained as products of these two orthonormal double μ–rotations, see (43),(44)).

7. Conclusions

In this paper it was shown that the use of approximate rotations based on orthonormal μ–rotations is particularly well suited for the SVD–updating algorithm. For example, slowly time varying processes correspond to small rotation angles which is especially advantageous for approximate rotations. Assuming a small angle the original CORDIC would execute the entire sequence of μ–rotation angles increasing the angle in the beginning of the sequence before decreasing it at the end of the sequence. Therefore, compared to the original CORDIC (exact rotation) only a fraction of the shift–add operations is required for the orthonormal μ–rotation based algorithms in order to obtain essentially the same performance.

References

1. S.B. Bensley and B. Aazhang. Subspace–Based Channel Estimation for Code Division Multiple Access Communication Systems. *IEEE Trans. on Communications (submitted)*, 1994.
2. R.P. Brent and F.T. Luk. The Solution of Singular Value and Symmetric Eigenvalue Problems on Multiprocessor Arrays. *SIAM J. Sci. Stat. Comput.*, 6:69–84, 1985.
3. J.R. Cavallaro and F.T. Luk. CORDIC Arithmetic for an SVD Processor. *J. Parallel & Distributed Computing*, 5:271–290, 1988.
4. T.F. Chan. Rank Revealing QR Factorization. *Linear Algebra and Its Applications*, 89:67–82, 1987.
5. P. Comon and G.H. Golub. Tracking a Few Extreme Singular Values and Vectors in Signal Processing. *Proceedings of the IEEE*, 78:1327–1343, 1990.
6. J.-M. Delosme. CORDIC Algorithms: Theory and Extensions. In *Proc. SPIE Advanced Alg. and Arch. for Signal Processing IV*, volume 1152, pages 131–145, 1989.
7. J.-M. Delosme. Bit-Level Systolic Algorithm for the Symmetric Eigenvalue Problem. In *Proc. Int. Conf. on Application Specific Array Processors*, pages 771–781, Princeton (USA), 1990.
8. L. Euler. *Opra Omnia I*, 6:283–315, 1921.
9. W. Ferzali and J.G. Proakis. Adaptive SVD Algorithm for Covariance Matrix Eigenstructure Computation. In *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, pages 2615–2618, Toronto (Canada), 1990.
10. W.M. Gentleman and H.T. Kung. Matrix Triangularization by Systolic Arrays. In *SPIE Real–Time Signal Processing IV 298*, pages 19–26, San Diego (USA), 1981.
11. J. Götze. An Iterative Version of the QRD for Adaptive RLS Filtering. In *Proc. SPIE's Int. Conf. on Advanced Signal Processing: Algorithms, Architectures and Implementations*, San Diego (USA), 1994.
12. J. Götze. On the Parallel Implementation of Jacobi and Kogbetliantz Algorithms. *SIAM J. on Sci. Comput.*, 15:1331–1348, 1994.

13. J. Götze and G.J. Hekstra. Adaptive Approximate Rotations for EVD. In M. Moonen and F. Catthoor, editors, *In Algorithms and Parallel VLSI Architectures*, Leuven, 1994.
14. J. Götze and G.J. Hekstra. An Algorithm and Architecture based on Orthonormal μ–Rotations for Computing the Symmetric EVD. *INTEGRATION – The VLSI Journal. Special issue on algorithms and parallel VLSI architectures*, pages 21–40, 1995.
15. J. Götze, S. Paul, and M. Sauer. An Efficient Jacobi–Like Algorithm for Parallel Eigenvalue Computation. *IEEE Trans. on Computers*, 42:1058–1065, 1993.
16. J. Götze, P. Rieder, and J.A. Nossek. Parallel SVD–Updating Using Approximate Rotations. In *SPIE Advanced Signal Processing: Algorithms, Architectures and Implementations VI*, pages 242–252, San Diego (USA), 1995.
17. S.F. Hsieh, K.J.R. Liu, and K. Yao. A Unified Square–Root–Free Approach for QRD–Based Recursive Least Squares Estimation. *IEEE Trans. on Signal Processing*, 41:1405–1409, 1993.
18. F.T. Luk and H. Park. On Parallel Jacobi Orderings. *SIAM J. Sci. Stat. Comput.*, 10:18–26, 1989.
19. M. Moonen, B. De Moor, L. Vandenberghe, and J. Vandewalle. On– and Off–Line Identification of Linear State–Space Models. *Int. J. Control*, 49:219–232, 1989.
20. M. Moonen, P. van Dooren, and J. Vandewalle. A Singular Value Decomposition Updating Algorithm for Subspace Tracking. *SIAM J. Matrix Anal. Appl.*, 13:1015–1038, 1992.
21. M. Moonen, P. van Dooren, and F. Vanpoucke. On the QR Algorithm and Updating the SVD and the URV Decomposition in Parallel. *Linear Algebra and Its Applications*, 188:549–568, 1993.
22. A. Paulraj, R. Roy, and T. Kailath. A Subspace Approach to Signal Parameter Estimation. *Proceedings of the IEEE*, 74:1044–1045, 1986.
23. G.W. Stewart. An Updating Algorithm for Subspace Tracking. *IEEE Trans. on Signal Processing*, 40:1535–1541, 1992.
24. H.W. van Dijk and E.F. Deprettere. Transformational Reasoning on Time–Adaptive Jacobi–Type Algorithms. In *3rd Int. Workshop on SVD and Signal Processing*, Leuven (Belgium), 1994.
25. F. Vanpoucke, M. Moonen, and E. Deprettere. A Numerically Stable Jacobi Array for Parallel SVD Updating. In *SPIE Advanced Signal Processing: Algorithms, Architectures and Implementations V*, volume 2296, pages 403–412, San Diego (USA), 1994.
26. J.E. Volder. The CORDIC Trigonometric Computing Technique. *IRE Trans. Electronic Computers*, EC-8:330–334, 1959.
27. J.S. Walter. An Unified Algorithm for Elementary Functions. In *Proc. Spring Joint Computer Conference*, volume 38, page 397. AFIPS press, 1971.
28. B. Yang and J.F. Böhme. Reducing the Computations of the Singular Value Decomposition Array Given by Brent and Luk. *SIAM J. Matrix Anal. Appl.*, 12:713–725, 1991.

Jürgen Götze received his Dipl.–Ing. and Dr.–Ing. degrees in Electrical Engineering from the Technical University of Munich in 1987 and 1990, respectively. From 1987 to 1990 he was with the Institute of Network Theory and Circuit Design, Technical University of Munich. From 1991 to 1992 he was with the Dept. of Computer Science, Yale University, supported by a research fellowship of the German National Science Foundation. From 1992 to 1994 he was with the Dept. of Electrical Engineering, Technical University of Munich. In 1995 he was with the Department of Electrical and Computer Engineering, Rice University, supported by the Alexander von Humboldt foundation. Since 1996 he is again with the Technical University of Munich. His research interests include numerical linear algebra, parallel algorithms, signal and array processing, approximation theory, wavelets.

Peter Rieder was born in Germany on April 4, 1967. He recieved his Dipl.–Ing. degree in Electrical Engineering from the Technical University of Munich in 1993. Since 1993, he is with the Institute of Network Theory and Circuit Design at the Technical University of Munich, working on his Ph.D. degree. His research interests include parallel algorithms, signal processing, especially multirate filterbanks and wavelets.

Gerben J. Hekstra was born in Rotterdam, The Netherlands, on December 9, 1965. He received the M.Sc. degree in Informatics in 1990 from the Delft University of Technology. Since 1991 he has been at the Network Theory section of the Electrical Engineering Department working on his Ph.D. degree. He is currently working on the realization of a high–speed graphics engine for photo–realistic rendering of artificial scenes. His research interests include computer arithmetic, parallel algorithms and architectures, and efficient VLSI architectures for high–throughput computations..

Table of contents:

Contact address:

Jürgen Götze
Network Theory & Circuit Design
TU Munich, Arcisstr. 21
D–80290 Munich, Germany
Tel.: +49 89 2105 8508
Fax.: +49 89 2105 8504
email: jugo@nws.e-technik.tu-muenchen.de

Title of the Special Issue: "Array Optimization and Adaptive Tracking Algorithms"