Iterative version of the QRD for adaptive RLS filtering

Jürgen Götze

Institute of Network Theory and Circuit Design
Technical University of Munich
Arcisstr. 21, 80290 Munich, Germany
email: jugo@nws.e-technik.tu-muenchen.de

Abstract

A modified version of the QR–decomposition (QRD) is presented. It uses approximate Givens rotations instead of exact Givens rotations, i.e., a matrix entry usually annihilated with an exact rotation by an angle ϕ is only reduced by using an approximate rotation by an angle $\tilde{\phi}$. The approximation of the rotations is based on the idea of CORDIC. Evaluating a CORDIC–based approximate rotation is to determine the angle $\tilde{\phi} = \phi_\ell = \arctan 2^{-\ell}$, which is closest to the exact rotation angle ϕ. This angle ϕ_ℓ is applied instead of ϕ. Using approximate rotations for computing the QRD results in an iterative version of the original QRD.

A recursive version of this QRD using CORDIC–based approximate rotations is applied to adaptive RLS filtering. Only a few angles of the CORDIC sequence, r say ($r \ll b$, where b is the word length), work as well as using exact rotations ($r = b$, original CORDIC). The misadjustment error decreases as r increases. The convergence of the QRD–RLS algorithm, however, is insensitive to the value of r. Adapting the approximation accuracy during the course of the QRD–RLS algorithm is also discussed. Simulations (channel equalization) confirm the results.

Keywords: QR–decomposition, CORDIC, approximate rotations, adaptive RLS filtering.

1 Introduction

In many signal processing applications (e.g. system identification, channel equalization) an adaptation algorithm is used to adapt the coefficients of the used filter to the instationarity of the underlying processes. The M coefficients $\mathbf{w}$ of the filter are adapted such that the difference between the filter output $y(t)$ and a given reference signal $\tilde{y}(t)$, i.e., the error signal $e(t)$, is minimized (Figure 1).

For the minimization of the error signal two different criteria are known – the least mean square (LMS) or the least squares (LS) minimization criteria. The algorithms derived from these minimization criteria are the LMS algorithm and the RLS algorithm, respectively [15]. The LMS algorithm is computationally inexpensive (O(M) per sample data vector, where M is the filter length) and easy to implement. The RLS algorithm on the other hand is computationally more expensive (O(M^2)) but its convergence is insensitive to the eigenvalue spread of the data covariance matrix.

Due to its computational expense the RLS algorithm does not comply with many real time applications and therefore parallel implementations have been developed. The method of choice for the fast parallel implementation of adaptive RLS filtering is based on the QR decomposition (QRD) $\mathbf{X} = \mathbf{QR}$ of the data matrix. The respective QRD–RLS

algorithm has better numerical properties than the covariance based RLS algorithm (no squaring of the condition number of the data matrix) and is highly suited for a parallel implementation.

The QRD–RLS algorithm can efficiently be implemented on the Gentleman–Kung array [7] (Figure 2). The main tasks of the processor cells of this multiprocessor array is the evaluation and the execution of plane rotations. In the diagonal cells the rotations are evaluated. These rotations are sent to the processor cells of the same row, where they are applied to the respective rows of the matrix. Finally, the processor array contains the upper triangular matrix $\mathbf{R}$ of the QRD $\mathbf{X} = \mathbf{QR}$ and $\mathbf{Q}$ is the procuct of all applied rotations. The upper triangular matrix can recursively be updated by simply feeding new data vectors (new rows of $\mathbf{X}$) into the processor array.

Usually for the implementation of the plane rotations additions, multiplications, divisions and square roots (or transcendental functions) are required. Therefore, the hardware requirements and the data pulse frequency of the processor array is determined by the complexity of the rotation evaluation. There are, however, different methods for modifying the plane rotations with respect to the efficient parallel implementation of rotation–based algorithms (QRD, QR–algorithm, Jacobi methods) on multiprocessor arrays: factorized/fast rotations [6, 14, 8, 20, 13], approximate rotations [19, 2], factorized and approximate rotations [10]. A further widely used method for a modified computation of plane rotations with respect to an efficient VLSI implementation is the CORDIC [21, 3, 5, 4]. CORDIC enables the evaluation and the execution of a plane rotation by a sequence of shift–and–add operations. Using the CORDIC algorithm a plane rotation is evaluated (and executed) by applying b elementary CORDIC angles $\phi_i = \arctan 2^{-i}$, where $i = 0, 1, 2, \ldots, b$ (b = word length).

In this paper the idea of using CORDIC–based approximate rotations, which was presented in [9, 12] for the Jacobi algorithm is applied to the Givens rotations of the QRD. Evaluating the CORDIC–based approximate Givens rotation is to determine the shift value ℓ corresponding to the CORDIC angle $\phi_\ell = \arctan 2^{-\ell}$ which is closest to the exact rotation angle ϕ (i.e. $\phi = \arctan y/x$ required to rotate $[x, y]^T$ to $[x', y']^T$ such that $y' = 0$). Using approximate rotations means no annihilation of the matrix entry but only a reduction of the respective matrix entry (i.e., $|y'| = |d| \cdot |y|$, where $0 \leq |d| < 1$). Thus, in order to obtain the final upper triangular matrix $\mathbf{R}$ the QRD with approximate rotations must be executed iteratively. Therefore, the use of approximate rotations for the QRD results in an iterative version of the QRD. One iteration step of this iterative QRD can be implemented on the Jacobi–type QRD array of Luk [18]. For signal processing applications it is, however, advantageous to use the QRD–RLS algorithm with approximate rotations in its standard form, i.e., as implemented on the Gentleman–Kung array. The matrix entry which is usually annihilated is only reduced. Instead of rotating this matrix entry to zero the remaining part of the matrix entry is neglected and, instead, it is dealt with the next data vector. An approximation accuracy of the rotation corresponding to a few angles of the CORDIC, r say, where $r \ll b$, is sufficient to obtain the same performance as using exact rotations ($r = b$). The misadjustment errror decreases as r increases. The speed of convergence, however, is insensitive to the value of r. Different approximation accuracies are considered and adapting the approximation accuracy during the course of the QRD–RLS algorithm is also discussed. Simulations of the QRD–RLS algorithm with CORDIC–based approximate rotations are presented for a channel equalization example.

In section 2 the QRD–RLS algorithm for adaptive filtering and its parallel implementation on the Gentleman–Kung array is reviewed. In section 3 the CORDIC–based approximate Givens rotation is presented. In section 4 the CORDIC–based approximate Givens rotations are used for adaptive QRD–RLS filtering. The performance for RLS filtering and the complexity of a parallel implementation are discussed. Section 5 concludes the paper.

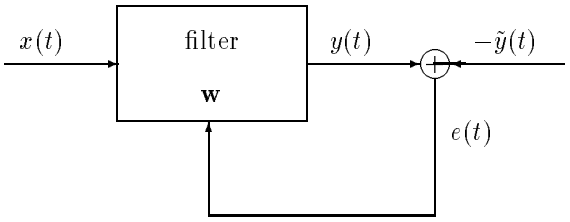

Figure 1: Adaptive filter

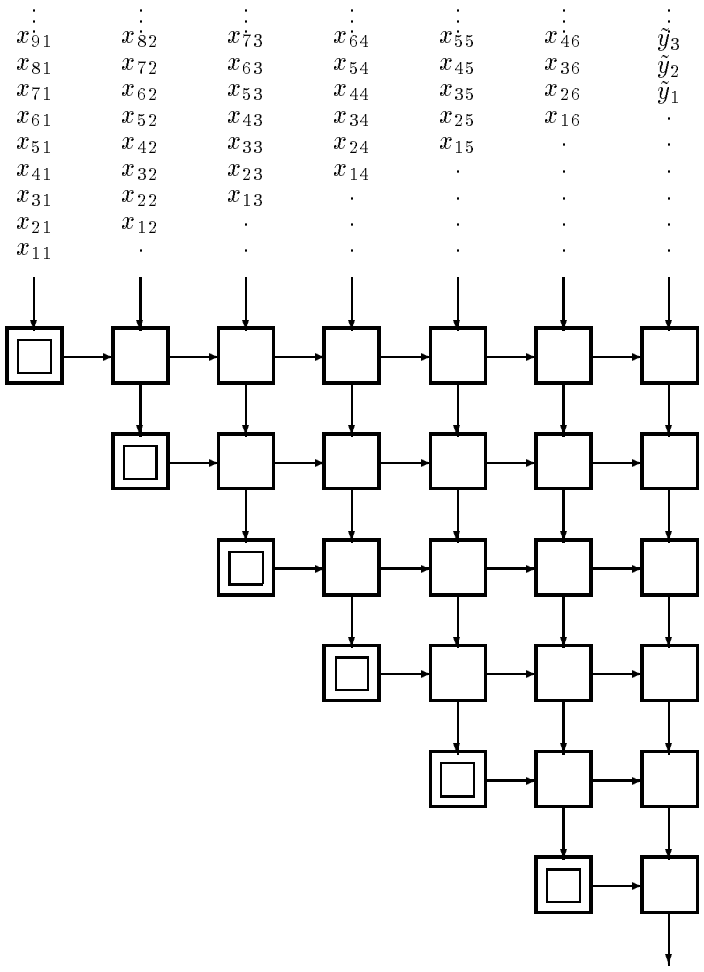

Figure 2: Gentleman–Kung array for QRD–RLS filtering ($M = 7$)

2 QRD–RLS Algotrithm

A Givens rotation $\mathbf{G}(\phi)$ is defined by the rotation angle ϕ as follows:

$$\mathbf{G}(\phi) = \frac{1}{\sqrt{1+\tan^2\phi}} \begin{bmatrix} 1 & \tan\phi \\ -\tan\phi & 1 \end{bmatrix} . \tag{1}$$

Usually the rotation angle ϕ is computed such that a vector $[x, y]^T$ is rotated by ϕ, i.e.

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \mathbf{G}(\phi) \begin{bmatrix} x \\ y \end{bmatrix} , \tag{2}$$

where ϕ is computed such that $y' = 0$. We denote the corresponding rotation $\mathbf{G}(\phi)$ as an exact rotation since it generates $y' = 0$ exactly. The rotation angle ϕ of the exact rotation is defined by

$$\phi = \arctan\frac{y}{x} . \tag{3}$$

Usually the QRD–RLS algorithm works as follows. Let

$$\mathbf{X}(t-1) = \mathbf{Q}(t-1)\mathbf{R}(t-1)$$

be the QRD of the $(t-1) \times M$ data matrix $\mathbf{X}(t-1)$ available at time step $t-1$. Given a new data vector $\mathbf{x}^T(t) = [x_1(t), \ldots, x_M(t)]^T$ the QRD of $\mathbf{X}(t) = \begin{bmatrix} \mathbf{X}(t-1) \\ \mathbf{x}^T(t) \end{bmatrix}$ is computed by appending the new data vector $\mathbf{x}^T(t)$ to $\mathbf{R}(t-1)$ and annihilating $\mathbf{x}^T(t)$ by a sequence of M Givens rotations. With λ as a forgetting factor ($\lambda = 1$ in order to find the QRD of $\mathbf{X}(t)$) we compute

$$\begin{bmatrix} \mathbf{R}(t) \\ \mathbf{0}^T \end{bmatrix} = \hat{\mathbf{Q}}^T(t) \begin{bmatrix} \lambda\mathbf{R}(t-1) \\ \mathbf{x}^T(t) \end{bmatrix} , \tag{4}$$

where $\hat{\mathbf{Q}}^T(t)$ is the product of the M Givens rotations required to annihilate $\mathbf{x}^T(t)$.

This QRD–RLS algorithm can efficiently be implemented on the Gentleman–Kung array [7] (Figure 2). Suppose that at time step $t-1$ the upper triangular matrix $\mathbf{R}(t-1)$ is stored in the upper triangular processor array. Now, the new data vector $\mathbf{x}^T(t)$ is feeded in the processor array as it is shown in Figure 2 for the first rows of $\mathbf{X}$. Let the index $[i]$ of $\mathbf{x}^T_{[i]}(t)$ denote the number of rotations already applied to the new data vector $\mathbf{x}^T_{[0]}(t) = [x_{[0]1}(t), \ldots, x_{[0]M}(t)]^T = \mathbf{x}^T(t)$. The i–th diagonal cell of the processor array computes the rotation $\mathbf{G}(\phi)$ required to annihilate the i–th component of $\mathbf{x}^T_{[i-1]}(t)$ with the i–th diagonal entry of $\mathbf{R}(t-1)$, i.e., $\mathbf{G}(\phi)$ is computed such that

$$\begin{bmatrix} r_{ii}(t) \\ 0 \end{bmatrix} = \mathbf{G}(\phi) \begin{bmatrix} \lambda r_{ii}(t-1) \\ x_{[i-1]i}(t) \end{bmatrix} .$$

Then the rotation $\mathbf{G}(\phi)$ is sent to the processor cells of the same row, where $\mathbf{G}(\phi)$ is applied to the i–th row of $\lambda\mathbf{R}(t-1)$ and the actual data vector $\mathbf{x}^T_{[i-1]}(t)$. The product of the M Givens rotations computed in the M diagonal cells and required to annihilate the M components of the data vector $\mathbf{x}^T(t)$ yields $\hat{\mathbf{Q}}^T(t)$.

3 CORDIC–Based Approximate Givens Rotation

An approximate rotation $\mathbf{G}(\tilde{\phi})$ is defined by an approximate rotation angle $\tilde{\phi}$ which guarantees

$$|y'| = |d| \cdot |y| \text{ with } 0 \leq |d| < 1 . \tag{5}$$

The reduction factor d depends on $\tau = x/y$ and $\tilde{t} = \tan\tilde{\phi}$ as follows:

$$d(\tau, t) = \frac{1 - \tau \cdot \tilde{t}}{\sqrt{1 + \tilde{t}^2}} \,. \tag{6}$$

Obviously, for the exact rotation where $\tilde{t} = \tan\phi = y/x$ one obtains $d = 0$.

At this point having defined an approximate rotation we make use of the idea of CORDIC [21], i.e. with respect to a simple implementation of the rotation we restrict ourselves to the set of approximate angles

$$\tilde{\phi} = \phi_i = \arctan 2^{-i} \,, \tag{7}$$

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

$$\mathbf{G}_u(i) = \begin{bmatrix} 1 & \sigma 2^{-i} \\ -\sigma 2^{-i} & 1 \end{bmatrix} \,. \tag{8}$$

An approximate unscaled rotation $\mathbf{G}_u(i) \cdot [x, y]^T$ (often denoted as a μ–rotation) can be executed by two shift–and–add operations. The orthonormal scaled rotation $\mathbf{G}_s(i) = K_i \mathbf{G}_u(i)$ is obtained by applying the scaling factor

$$K_i = \frac{1}{\sqrt{1 + 2^{-2i}}} \,. \tag{9}$$

The main difference to the original CORDIC is that only one specific rotation angle ϕ_i is chosen (the original CORDIC uses all ϕ_i for $i = 0, 1, \ldots, b$). Therefore, in contrary to the original CORDIC algorithm (constant scaling factor) the scaling factor K_i depends on the value of i.

The optimal CORDIC–based approximate rotation given by the optimal angle ϕ_ℓ is defined by the CORDIC angle ϕ_i ($i \in \mathcal{I}$) which is closest to the exact rotation angle ϕ, i.e.

$$|\phi_\ell - \phi| = \min_{i \in \mathcal{I}} |\phi_i - \phi| \,. \tag{10}$$

This optimal ϕ_ℓ can be determined following the ideas presented in [12, 11]. $\mathsf{man}(a)$ and $\exp(a)$, respectively, denote the mantissa and the exponent of a binary floating point number a.

- According to $\tan\phi = y/x \approx 2^{-\ell}$ an estimate for ℓ can be obtained by

$$\ell_e = \exp(y) - \exp(x) \,, \tag{11}$$

- Since $\mathsf{man}(y)/\mathsf{man}(x) \in [0.25, 1[$ one obtains

$$\ell \in \mathcal{J} = \{\ell_e, \ell_e + 1, \ell_e + 2\} \tag{12}$$

- By computing $[\hat{x}, \hat{y}]^T = \mathbf{G}_u(i) \cdot [x, y]^T$ with $i \in \mathcal{J}$ and checking the sign of $\hat{y}$ the optimal value for ℓ can be identified [11]. Note, that here only unscaled rotations are necessary.

This procedure guarantees $|d(\tau, \ell)| \leq 1/3$. $d(|\tau|, \ell)$ is shown in Figure 3 (solid line).

In virtue of an easy scaling factor compensation the approximate rotation is slightly changed by using a double rotation with $\ell = \ell + 1$ in order to avoid the square root in the scaling factor [3]. Therefore, the unscaled approximate rotation $\mathbf{G}_{du}(\ell)$ is

$$\mathbf{G}_{du}(\ell) = \begin{bmatrix} 1 - 2^{-2\ell} & \sigma 2^{-\ell+1} \\ -\sigma 2^{-\ell+1} & 1 - 2^{-2\ell} \end{bmatrix} \,. \tag{13}$$

The scaled rotation is $\mathbf{G}_{ds}(\ell) = K_\ell^2 \mathbf{G}_{du}(\ell)$, where $K_\ell^2 = 1/(1+2^{-2\ell})$ can recursively be computed by shift–and–add operations as follows [12]:

$$K_\ell^2 = (1-2^{-2\ell}) \prod_{i=1}^{w} (1+2^{-2^{i+1}\ell}) \text{ with } w = \log_2 \lfloor \frac{b}{2\ell} \rfloor . \tag{14}$$

The number of shift and add operations for scaling decreases for increasing values of ℓ (small angles), e.g., for $\ell > b/2$ no scaling is required at all.

The CORDIC–based approximate Givens rotation for rotating $[x, y]^T$ is summarized as follows:

- Determine the optimal ℓ by the procedure described above.
- Set $\ell = \ell + 1$ and compute $[\bar{x}, \bar{y}]^T = \mathbf{G}_{du}(\ell) \cdot [x, y]^T$.
- Execute $[x', y']^T = K_\ell^2 \cdot [\bar{x}, \bar{y}]^T$ according to (14).

This procedure slightly changes the original approximation and yields $d(|\tau|, \ell)$ as shown in Figure 3 (dash–dotted line). Here, we obtain $|d(|\tau|, \ell)| < 0.51$. Note, the fast convergence to the first approximation (solid line) for increasing ℓ.

The CORDIC–based approximate rotation can also be used to actually generate $y' = 0$ by applying the described procedure iteratively. Thereby, only the CORDIC angles, which are really necessary to generate $y' = 0$, are executed while the original CORDIC uses the complete sequence of CORDIC angles (suppose a small angle ϕ, then the large angles are also executed although they do not contribute to the reduction of y). The iterative application of the CORDIC–based approximate rotation to $[x, y]^T = [2, 1]^T$ is shown in Table 1 for a word length of $b = 16$. Obviously, only $r = 5$ CORDIC angles (r iterations of the above procedure) of the $b = 16$ CORDIC angles are really required to obtain $y' = 0$.

4 QRD–RLS Algorithm Using Approximate Rotations

Using CORDIC–based approximate rotations for the implementation of the Givens rotations it is possible to execute the rotations in the QRD–RLS algorithm with a chosen accuracy of the approximation, i.e. with a choosen number r of optimal CORDIC angles per rotation. As shown in Table 1 it is usually sufficient to execute $r \ll b$ optimal CORDIC angles to achieve an exact rotation. It will be shown in the sequel, however, that it is more efficient to work with approximate rotations for adaptive RLS filtering problems.

By using approximate rotations the new data vector is only reduced (instead of rotating it to zero as in (4)), i.e., we obtain

$$\begin{bmatrix} \tilde{\mathbf{R}}(t) \\ \tilde{\mathbf{x}}^T(t) \end{bmatrix} = \tilde{\mathbf{Q}}^T(t) \begin{bmatrix} \lambda \mathbf{R}(t-1) \\ \mathbf{x}^T(t) \end{bmatrix} , \tag{15}$$

where $\tilde{\mathbf{Q}}^T(t)$ is the product of the M approximate Givens rotations required to reduce the components of $\mathbf{x}^T(t)$. After this reduction $\tilde{\mathbf{x}}^T(t)$ is neglected, i.e., instead of reducing $\tilde{\mathbf{x}}^T(t)$ to zero by further μ–rotations (in order to obtain (4)) a new data vector $\mathbf{x}^T(t+1)$ is used and the next approximate QRD–update (15) is executed. Therefore, instead of executing n QRD–updates according to (4) one can roughly execute $\frac{b}{r}n$ approximate QRD–updates according to (15). One reason for the better performance of the approximate QRD–update algorithm is that it is more efficient to use the information provided by the new data vector approximately instead of spending the effort to actually reduce the current data vector exactly to zero (the main information is already included in $\tilde{\mathbf{R}}(t)$ since the greatest reduction of y is achieved by the first optimal CORDIC angle; compare Table 1).

Another way to analyze the better performance of the approximate QRD–updating algorithm is as follows. Suppose that the forgetting factor is $\lambda = 1$. The magnitude of the matrix elements of $\mathbf{R}(t)$ and $\tilde{\mathbf{R}}(t)$, respectively, will increase

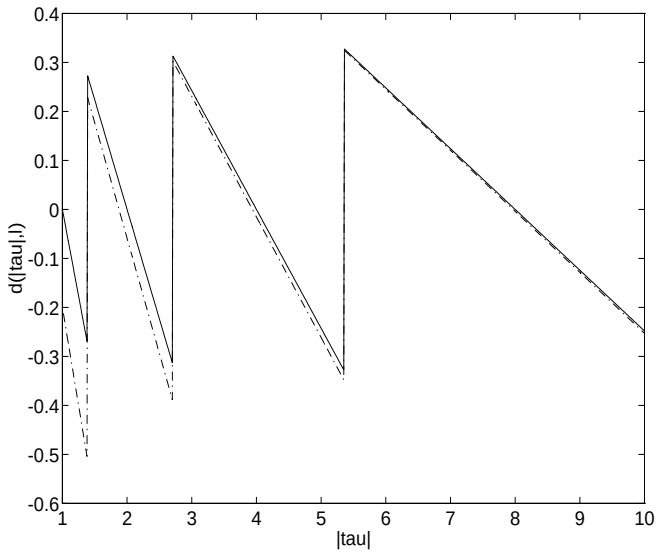

Figure 3: $d(|\tau|, \ell)$ for the CORDIC–based approximate Givens rotation (original approximation (solid line), double rotation method (dash–dotted line)).

			$[x, y]^T = [+2.0000, +1.0000]^T$
$r = 1:$	$\ell = 2$	$\sigma = -1$	$[x, y]^T = [+2.2352, -0.0588]^T$
$r = 2:$	$\ell = 6$	$\sigma = +1$	$[x, y]^T = [+2.2360, +0.0110]^T$
$r = 3:$	$\ell = 9$	$\sigma = -1$	$[x, y]^T = [+2.2361, +0.0023]^T$
$r = 4:$	$\ell = 11$	$\sigma = -1$	$[x, y]^T = [+2.2361, +0.0001]^T$
$r = 5:$	$\ell = 15$	$\sigma = -1$	$[x, y]^T = [+2.2361, -0.0000]^T$
$r = 6:$	$\ell = 18$	$> b = 16$	

Table 1: Iterative application of the CORDIC–based approximate rotation (double rotation method).

while the magnitude of the components of $\mathbf{x}^T(t)$ remains in the same range, i.e. the required angles will tend to zero for $t \rightarrow \infty$. Obviously, the original CORDIC does not take any advantage of this fact since it always executes the complete sequence of CORDIC angles. Therefore, a lot of unnecessary (large) CORDIC angles are executed until the range of the really required (small) CORDIC angles is reached. The presented CORDIC–based approximate rotation method takes full advantage of the decrease of the angles by only executing the CORDIC angles which are really necessary. This also holds for the usually applied forgetting factors $0 \ll \lambda < 1$ but to a smaller extend for smaller λ. Note, that smaller angles (ℓ large) also result in a reduced computational cost for scaling according to (14) (e.g. for $\ell > b/2$ no scaling is required at all). Furthermore, it is also possible to derive different types of μ–rotations adapted to the required rotation angle [11] such that the required amount of shift–and–add operations is reduced.

In essence it is sufficient to execute a small number r of optimal CORDIC angles per rotation as in (15) to obtain results comparable to the use of exact rotations as in (4). This is illustrated by the following simulations.

4.1 Simulations

The channel equalization computer experiment as given in chapter 5 of [15] is used to show the performance of the approximate QRD–updating algorithm (15) at different approximation levels (different values of r). Increasing the value W corresponds to an increased distortion of the channel which is equivalent to an increased eigenvalue spread. In Figure 4 ($W = 2.9$) and Figure 5 ($W = 3.5$) the ensamble average (30 independent runs) of the squared error $\epsilon^2(t)$ is shown for the LMS–algorithm and the QRD–RLS algorithm using different approximation levels ($r = 1$, $r = 2$, $r = 3$ CORDIC angles per rotation, and exact rotation $b = 32$). Obviously, using $r = 3$ optimal CORDIC angles per rotation works as well as using exact rotations. The misadjustment error decreases as r increases. The convergence of the approximate QRD–RLS does not depend on r.

Instead of working with a fixed number of CORDIC angles per rotation during the whole QRD–RLS algorithm it is also possible to adapt the number r of executed CORDIC angles per rotation during the course of the algorithm. Thereby, it is possible to reduce the required amount of μ–rotations. Usually the QRD–RLS algorithm requires approximately $2M$ steps (stationary processes) for convergence. In Figure 6 the ensamble average error is shown for working with $r = 3$ optimal CORDIC angles during the whole QRD–RLS algorithm and for working with the following adaptation

$$\begin{aligned} \textbf{A1:}\ r &= 1 \quad \text{for } t \leq M \\ r &= 2 \quad \text{for } M < t \leq 2M \\ r &= 3 \quad \text{for } t > 2M\,. \end{aligned}$$

Obviously, this adaptive scheme (dotted line) works as well as using $r = 3$ throughout the QRD–RLS algorithm (solid line). In Figure 7 the ensamble average error is shown for the QRD–RLS algorithm with $r = 3$ (solid line) and $r = 1$ (dashed line) optimal CORDIC angles during the whole algorithm and for a reduction of the approximation accuracy after the QRD–RLS algorithm has reached convergence (dotted line), i.e.,

$$\begin{aligned} \textbf{A2:}\ r &= 3 \quad \text{for } t < 50 \\ r &= 1 \quad \text{for } t \geq 50 \end{aligned}$$

Obviously, the approximation accuracy $r = 3$ is still nessecary when convergence has been reached. Otherwise, the ensamble average error increases to the level corresponding to $r = 1$.

4.2 Additional Remarks

Implementation: So far we have only compared the approximate QRD–updating (15) to the exact QRD–updating (4) from the algorithmic point of view. It is important to note that the implementation of the Gentleman–Kung

QRD–RLS array using CORDIC–based approximate rotations is much simpler than an implementation based on exact rotations (original CORDIC). Only one floating point adder per processor cell is required and can be used for the angle computation (determine optimal ℓ) and angle application (execution of μ–rotation and scaling). The results concerning the efficient implementation of floating point CORDIC–based approximate rotations derived for the Jacobi method [11] can be applied to the implementation of the QRD–RLS array with minor modifications.

Filter length and minimal error: Besides the performance and the simple implementation the presented algorithm provides another degree of freedom. The minimal reachable level of $E\{e^2(t)\}$ usually depends on the filter length M of the adaptive fiter. The minimum of $E\{e^2(t)\}$ decreases as M increases. Changing the filter length M, however, means a change of the implementation in that the Gentleman–Kung array requires M columns. Using the approximate QRD–updating algorithm corresponding to a given (long) filter length M enables the reduction of the minimal error level by increasing the approximation accuracy (i.e. the number of CORDIC angles per rotation r; see Figure 4 and 5). Using exact rotations results in the minimal possible error level while the use of approximate rotations yields error levels corresponding to smaller filter lengths $M_a < M$. Therefore, a reduced accuracy of the approximation (smaller r) is equivalent to use a shorter filter length and applying exact rotations. For this reason it is not necessary to change the architecture of the Gentleman–Kung array but only the value of r in order to realize different filter lengths.

Covariance estimation: It is also interesting to consider the approximate QRD–updating procedure as a method to obtain the square root (Cholesky factor) of the covariance matrix approximately. From this point of view the presented QRD–RLS algorithm using approximate rotations is an intermediate method between using all measured data to compute the covariance information (i.e. the QRD–updating using exact rotations) and the sign methods [16, 17] (see also [1] for an overview).

5 Conclusions

In this paper a QRD–RLS algorithm using CORDIC–based approximate rotations was presented. This algorithm can be efficiently implemented on the Gentleman–Kung processor array since only one shift–and–add operation is required in all processor cells to perform the most simple approximate rotation (one specific CORDIC angle). Using CORDIC–based approximate rotations with different approximation accuracies (different r) was discussed. The usual QRD–RLS algorithm is a special case of the presented algorithm using exact rotations, i.e., $r = b$. For computing the QRD the presented algorithm can be considered as an iterative version of the standard QRD (each iteration yields a further step to the final upper triangular matrix).

For QRD–RLS filtering the application of approximate rotations yields a better overall performance since an approximation accuracy corresponding to $r = 3$ optimal CORDIC angles per rotation works as well as using exact rotations ($b = 32$). Furthermore, no change of the architecture is required to realize different filter lengths since shorter filter lengths correspond to less accurate approximations of the rotations.

In the presented version the QRD–RLS algorithm is supposed to be as simple to implement as the LMS algorithm, however, still providing the advantages of the QRD–RLS algorithm compared to the LMS algorithm. A comparative study of the actual complexities of these implementations is a topic of further research.

References

[1] M.G. Bellanger. *Adaptive Digital Filters and Signal Analysis*. Marcel Dekker, Inc., 1987.

[2] J.P. Charlier, M. Vanbegin, and P. van Dooren. On Efficient Implementations of Kogbetliantz's Algorithm for Computing the Singular Value Decomposition. *Numer. Math.*, 52:279–300, 1988.

[3] 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.

[4] E.F. Deprettere, A.A.J. de Lange, and P. Dewilde. The Synthesis and Implementation of Signal processing Applications Specific VLSI CORDIC Arrays. In *Proc. IEEE Int. Conf. on Acoust., Speech, Signal Processing*, pages 974–977, Albuquerque (USA), 1990.

[5] M.D. Ercegovac and T. Lang. Redundant and On–Line CORDIC: Application to Matrix Triangularization and SVD. *IEEE Trans. on Computers*, 39:725–740, 1990.

[6] W.M. Gentleman. Least Squares Computations by Givens Rotations Without Square Roots. *J. Inst. Maths Applics*, 12:329–336, 1973.

[7] 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.

[8] G.H. Golub and C.F. van Loan. *Matrix Computations*. The John Hopkins University Press, second edition, 1989.

[9] J. Götze. Parallel Methods for Iterative Matrix Computations. In *Proc. IEEE Int. Symp. on Circuits and Systems*, pages 233–236, Singapore, 1991.

[10] J. Götze. On the Parallel Implementation of Jacobi's and Kogbetliantz's Algorithm. *To appear SIAM J. on Sci. Comput.*, 15, 1994.

[11] J. Götze and G.J. Hekstra. Adaptive Approximate Rotations for EVD and SVD. In M. Moonen and F. Catthoor, editors, *In Algorithms and Parallel VLSI Architectures*, Leuven, 1994.

[12] J. Götze, S. Paul, and M. Sauer. An Efficient Jacobi–Like Algorithm for Parallel Eigenvalue Computation. *IEEE Trans. on Computers*, 42:1058–1065, 1993.

[13] J. Götze and U. Schwiegelshohn. A Square Root and Division Free Givens Rotation for Solving Least Squares Problems on Systolic Arrays. *SIAM J. Sci Stat. Comput.*, 12:800–807, 1991.

[14] S. Hammarling. A Note on Modifications to the Givens Plane Rotation. *J. Inst. Maths Applics*, 13:215–218, 1974.

[15] S. Haykin. *Adaptive Filter Theory*. Prentice–Hall, 1986.

[16] D. Hertz. A Fast Digital Method for Estimating the Autokorrelation of a Gaussian Stationary Process. *IEEE Trans. on Acoust., Speech, Signal Process.*, 30:329–330, 1982.

[17] G. Jacovitti and R. Cusani. Performances of the Hybrid–Sign Correlation Coefficients Estimator for Gaussian Stationary Processes. *IEEE Trans. on Acoust., Speech, Signal Process.*, 33:731–733, 1985.

[18] F.T. Luk. A Rotation Method for Computing the QR-Decomposition. *SIAM J. Sci. Stat. Comput.*, 7:452–459, 1986.

[19] J.J. Modi and J.D. Pryce. Efficient Implementation of Jacobi's Diagonalization Method on the DAP. *Numer. Math.*, 46:443–454, 1985.

[20] W. Rath. Fast Givens Rotations for Orthogonal Similarity Transformations. *Numer. Math.*, 40:47–56, 1982.

[21] J.E. Volder. The CORDIC Trigonometric Computing Technique. *IRE Trans. Electronic Computers*, EC-8:330–334, 1959.

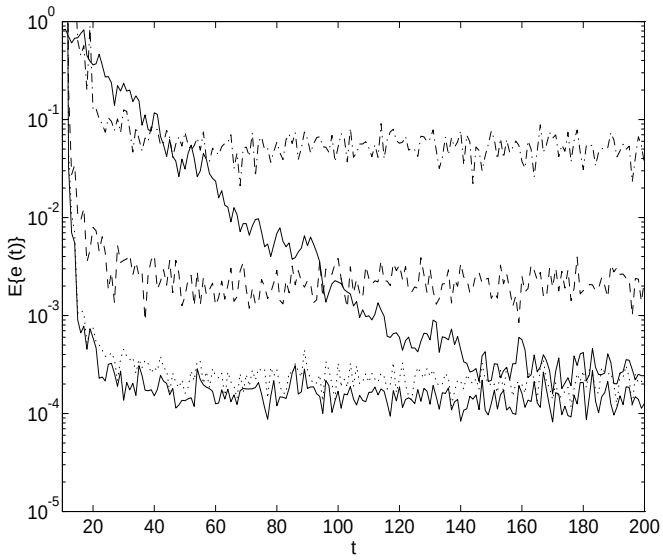

Figure 4: Ensamble average error of 30 independent runs of the channel equalization example (filter length $M = 10$) with $W = 2.9$ for the LMS algorithm and the QRD–RLS algorithm at different approximation levels of the rotations: LMS $\mu = 0.075$ (solid), QRD–RLS using exact rotations (solid), QRD–RLS with $r = 1$ (dash-dot), $r = 2$ (dashed), $r = 3$ (dotted) optimal CORDIC angles per rotation.

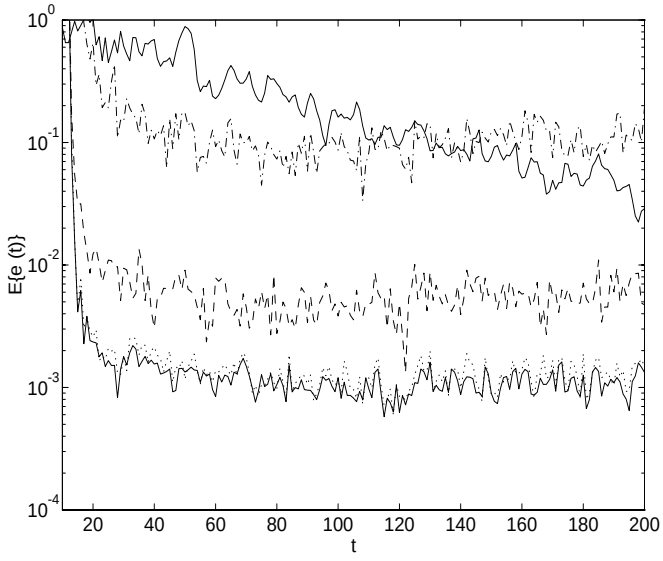

Figure 5: Ensamble average error of 30 independent runs of the channel equalization example (filter length $M = 10$) with $W = 3.5$ for the LMS algorithm and the QRD–RLS algorithm at different approximation levels of the rotations: LMS $\mu = 0.075$ (solid), QRD–RLS using exact rotations (solid), QRD–RLS with $r = 1$ (dash-dot), $r = 2$ (dashed), $r = 3$ (dotted) optimal CORDIC angles per rotation.

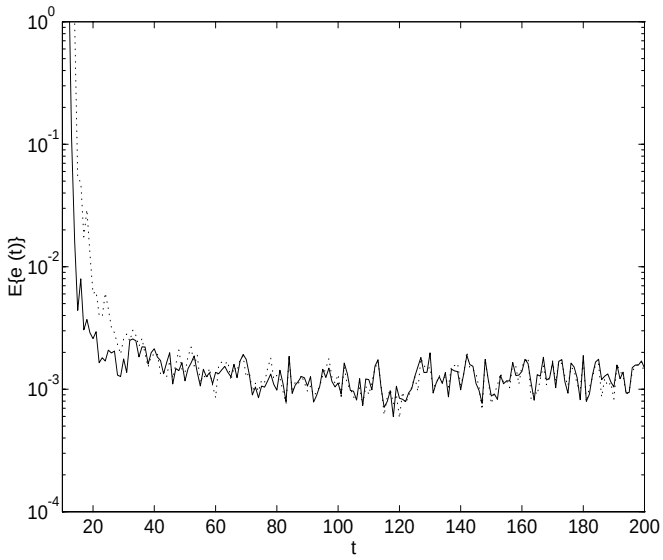

Figure 6: Ensamble average error of 30 independent runs of the channel equalization example (filter length $M = 10$) with $W = 3.5$ for the QRD–RLS algorithm with $r = 3$ (solid line) and the adaptive scheme according to adaptation **A1** (dotted line).

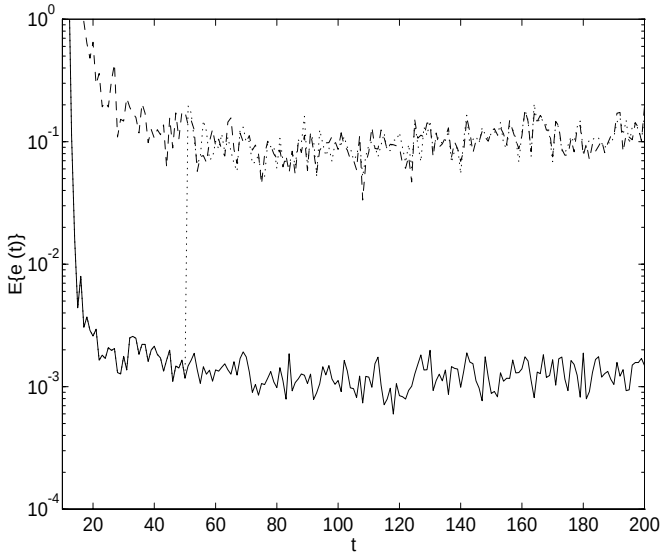

Figure 7: Ensamble average error of 30 independent runs of the channel equalization example (filter length $M = 10$) with $W = 3.5$ for the QRD–RLS algorithm with $r = 3$ (solid line), $r = 1$ (dashed line) and the adaptation of the accuracy according to adaptation **A2** (dotted line).