A Class of Square Root and Division Free Algorithms and Architectures for QRD-Based Adaptive Signal Processing

by E.N. Frantzeskakis and K.J.R. Liu

T.R. 93-65

The Institute for Systems Research is supported by the National Science Foundation Engineering Research Center Program (NSFD CD 8803012), the University of Maryland, Harvard University, and Industry

A Class of Square Root and Division Free Algorithms and Architectures for QRD-Based Adaptive Signal Processing

E. N. Frantzeskakis and K.J.R. Liu

Electrical Engineering Department
and Institute for Systems Research
University of Maryland
College Park, MD 20742

Abstract

The least squares (LS) minimization problem constitutes the core of many real-time signal processing problems, such as adaptive filtering, system identification and adaptive beamforming. Recently efficient implementations of the recursive least squares (RLS) algorithm and the constrained recursive least squares (CRLS) algorithm based on the numerically stable QR decomposition (QRD) have been of great interest. Several papers have proposed modifications to the rotation algorithm that circumvent the square root operations and minimize the number of divisions that are involved in the Givens rotation. It has also been shown that all the known square root free algorithms are instances of one parametric algorithm. Recently, a square root free and division free algorithm has been proposed [4].

In this paper, we propose a family of square root and division free algorithms and examine its relationship with the square root free parametric family. We choose a specific instance for each one of the two parametric algorithms and make a comparative study of the systolic structures based on these two instances, as well as the standard Givens rotation. We consider the architectures for both the optimal residual computation and the optimal weight vector extraction.

The dynamic range of the newly proposed algorithm for QRD-RLS optimal residual computation and the wordlength lower bounds that guarantee no overflow are presented. The numerical stability of the algorithm is also considered. A number of obscure points relevant to the realization of the QRD-RLS and the QRD-CRLS algorithms are clarified. Some systolic structures that are described in this paper are very promising, since they require less computational complexity (in various aspects) than the structures known to date and they make the VLSI implementation easier.

SP EDICS:
5.2. Algorithms and Application Mappings
5.1. Architectures and VLSI Hardware

This work was supported in part by the ONR grant N00014-93-1-0566.

1 Introduction

The least squares (LS) minimization problem constitutes the core of many real-time signal processing problems, such as adaptive filtering, system identification and beamforming [6]. There are two common variations of the LS problem for adaptive signal processing:

1. Solve the minimization problem

$$w(n) = \arg\min_{w(n)} \| \mathcal{B}(n)(X(n)w(n) - y(n)) \|^2, \tag{1}$$

   where $X(n)$ is a matrix of size $n \times p$, $w(n)$ is a vector of length p, $y(n)$ is a vector of length n and $\mathcal{B}(n) = \mathrm{diag}\{\beta^{n-1}, \beta^{n-2}, \cdots, 1\}, 0 < \beta < 1$, that is, β is a forgetting factor.

2. Solve the minimization problem in (1) subject to the linear constraints

$$c^{iT} w(n) = r^i, i = 1, 2, \cdots, N, \tag{2}$$

   where c^i is a vector of length p and r^i is a scalar. In this paper, we consider only the special case for which $y(n) = 0$ for all n.

There are two different pieces of information that may be required as the result of this minimization [6]:

1. The optimizing weight vector $w(n)$ and/or

2. the optimal residual at the time instant n:

$$e(t_n) = X(t_n)w(n) - y(t_n), \tag{3}$$

   where $X(t_n)$ is the last row of the matrix $X(n)$ and $y(t_n)$ is the last element of the vector $y(n)$.

Efficient implementations of the recursive least squares (RLS) algorithms and the constrained recursive least squares (CRLS) algorithms based on the QR decomposition (QRD) were first introduced by McWhirter [14], [15]. A comprehensive description of the algorithms and the architectural implementations of these algorithms is given in [6, chap.14]. It has been proved that the QRD-based algorithms have good numerical properties [6]. However, they are not very appropriate for VLSI implementation, because of the square root and the division operations that are involved in the Givens rotation and the back-substitution required for the case of weight extraction.

Several papers have proposed modifications in order to reduce the computational load involved in the original Givens rotation [2, 5, 4, 8]. These rotation-based algorithms are not rotations any more, since they do not exhibit the normalization property of the Givens rotation. Nevertheless, they can substitute for the Givens rotation as the building block of the QRD algorithm and thus they can be treated as rotation algorithms in a wider sense:

Definition 1 *A Givens-rotation-based algorithm that can be used as the building block of the QRD algorithm will be called a* $\Re$otation algorithm.

A number of square-root-free $\Re$otations have appeared in the literature [2], [5], [8], [10]. It has been shown that a square-root-free and division-free $\Re$otation does exist [4]. Recently, a parametric family of square-root-free $\Re$otation algorithms was proposed in [8]; it was also shown that all the known square-root-free $\Re$otation algorithms belong to this family, which is called the "$\mu\nu$-family". In this paper we will refer to the $\mu\nu$-family of $\Re$otation algorithms with the name *parametric $\mu\nu$ $\Re$otation*. We will also say that a $\Re$otation algorithm is *a $\mu\nu$ $\Re$otation* if this algorithm belongs to the $\mu\nu$-family. Several QRD-based algorithms have made use of these $\Re$otation algorithms. McWhirter has been able to compute the optimal residual of the RLS algorithm without square root operations [14]. He also employed an argument for the similarity of the RLS and the CRLS algorithms to obtain a square-root-free computation for the optimal residual of the CRLS algorithm [15]. A fully-pipelined structure for weight extraction that circumvents the back-substitution divisions was also derived independently in [17] and in [19]. Finally, an algorithm for computing the RLS optimal residual based on the parametric $\mu\nu$ $\Re$otation was derived in [8].

In this paper, we introduce a parametric family of square-root-free and division-free $\Re$otations. We will refer to this family of algorithms with the name *parametric $\kappa\lambda$ $\Re$otation*. We will also say that a $\Re$otation algorithm is *a $\kappa\lambda$ $\Re$otation* if this algorithm is obtained by the parametric $\kappa\lambda$ $\Re$otation with a choice of specific values for the parameters κ and λ. We employ the arguments in [8], [14], [15] and [17] in order to design novel architectures for the RLS and the CRLS algorithms that have less computation and circuit complexity. Some systolic structures that are described here are very promising, since they require the minimum computational complexity (in various aspects) known to date, and they can be easily implemented in VLSI.

In Section 2, we introduce the parametric $\kappa\lambda$ $\Re$otation. In Section 3, we derive the RLS algorithms that are based on the parametric $\kappa\lambda$ $\Re$otation and we consider the architectural implementations for a specific $\kappa\lambda$ $\Re$otation. In Section 4, we follow the same procedure for the CRLS algorithms. In Section 5, we address the issues of dynamic range, lower bounds for the wordlength, stability and error bounds. We conclude with Section 6. In the Appendix we give the proofs of some lemmas that are stated in the course of the paper.

2 Square Root and Division free Algorithms

In this Section, we introduce a new parametric family of Givens-rotation-based algorithms that require neither square root nor division operations. This modification to the Givens rotation provides a better insight on the computational complexity optimization issues of the QR decomposition and makes the VLSI implementation easier.

2.1 The Parametric $\kappa\lambda$ $\Re$otation

The standard Givens rotation operates (for real-valued data) as follows:

$$\begin{bmatrix} r_1' & r_2' & \cdots & r_m' \\ 0 & x_2' & \cdots & x_m' \end{bmatrix} = \begin{bmatrix} c & s \\ -s & c \end{bmatrix} \begin{bmatrix} \beta r_1 & \beta r_2 & \cdots & \beta r_m \\ x_1 & x_2 & \cdots & x_m \end{bmatrix}, \tag{4}$$

where

$$c = \frac{\beta r_1}{\sqrt{\beta^2 r_1^2 + x_1^2}}, \quad s = \frac{x_1}{\sqrt{\beta^2 r_1^2 + x_1^2}} \tag{5}$$

$$r_1' = \sqrt{\beta^2 r_1^2 + x_1^2} \tag{6}$$

$$r_j' = c\beta r_j + s x_j, \quad j = 1, 2, \cdots, m \tag{7}$$

$$x_j' = -s\beta r_j + c x_j, \quad j = 2, 3, \cdots, m \ . \tag{8}$$

We introduce the following data transformation:

$$\begin{aligned} r_j &= \frac{1}{\sqrt{l_a}}a_j, \quad x_j = \frac{1}{\sqrt{l_b}}b_j, \quad r'_j = \frac{1}{\sqrt{l'_a}}a'_j, \quad j = 1,2,\cdots,m \\ x'_j &= \frac{1}{\sqrt{l'_b}}b'_j, \qquad\qquad\qquad\qquad j = 2,3,\cdots,m. \end{aligned} \tag{9}$$

We seek the square root and division-free expressions for the transformed data $a'_j, j = 1,2,\cdots,m$, b'_j, $j = 2,3,\cdots,m$, in (6) and solving for a'_1, we get

$$a'_1 = \sqrt{\frac{l'_a}{l_a l_b}(l_b\beta^2 a_1^2 + l_a b_1^2)}. \tag{10}$$

By substituting (5) and (9) in (7) and (8) and solving for a'_j and b'_j, we get

$$a'_j = \frac{l_b\beta^2 a_1 a_j + l_a b_1 b_j}{\sqrt{l_a l_b(l_b\beta^2 a_1^2 + l_a b_1^2)/l'_a}} \quad \text{and} \quad b'_j = \frac{-b_1\beta a_j + \beta a_1 b_j}{\sqrt{(l_b\beta^2 a_1^2 + l_a b_1^2)/l'_b}} \quad j = 2,3,\cdots,m \ . \tag{11}$$

We will let l'_a and l'_b be equal to

$$l'_a = l_a l_b(l_b\beta^2 a_1^2 + l_a b_1^2)\kappa^2, \quad l'_b = (l_b\beta^2 a_1^2 + l_a b_1^2)\lambda^2, \tag{12}$$

where κ and λ are two parameters. By substituting (12) in (10)-(11) we obtain the expressions

$$a'_1 = \kappa(l_b\beta^2 a_1^2 + l_a b_1^2) \tag{13}$$

$$a'_j = \kappa(l_b\beta^2 a_1 a_j + l_a b_1 b_j), \quad j = 2,3,\cdots,m \quad \text{and} \tag{14}$$

$$b'_j = \lambda\beta(-b_1 a_j + a_1 b_j), \quad j = 2,3,\cdots,m. \tag{15}$$

If the evaluation of the parameters κ and λ does not involve any square root or division operations, the update equations (12)-(15) will be square root and division-free. In other words, every such choice of the parameters κ and λ specifies a square root and division-free $\Re$otation algorithm.

Definition 2 *Equations (12)-(15) specify* the parametric $\kappa\lambda$ $\Re$otation algorithm. *Furthermore, a $\Re$otation algorithm will be called* a $\kappa\lambda$ $\Re$otation *if it is specified by (12)-(15) for specific square-root-free and division-free expressions of the parameters κ and λ.*

One can easily verify that the only one square root and division-free $\Re$otation in the literature to date [4] is a $\kappa\lambda$ $\Re$otation and is obtained by choosing $\kappa = \lambda = 1$.

2.2 The Relation between the Parametric $\kappa\lambda$ and the Parametric $\mu\nu$ $\Re$otation

Let

$$k_a = \frac{1}{l_a}, \quad k_b = \frac{1}{l_b}, \quad k'_a = \frac{1}{l'_a}, \quad k'_b = \frac{1}{l'_b}. \tag{16}$$

We can express k'_a and k'_b in terms of k_a and k_b as follows [8]:

$$k'_a = \left(k_a\beta^2 a_1^2 + k_b b_1^2\right)/\mu^2, \quad k'_b = \frac{k_a k_b}{\mu^2\nu^2}\frac{1}{k'_a}. \tag{17}$$

If we substitute (16) and (17) in (12) and solve for μ and ν we obtain

$$\mu = \frac{\kappa(k_a\beta^2 a_1^2 + k_b b_1^2)}{k_a k_b}, \quad \nu = \lambda. \tag{18}$$

The above provides a proof for the following Lemma:

Lemma 2.1 *For each square root and division-free pair of parameters (κ, λ) that specifies a $\kappa\lambda$ $\Re$otation algorithm A1, we can find square-root-free parameters $(\mu(\kappa), \nu(\lambda))$ with two properties: first, the pair $(\mu(\kappa), \nu(\lambda))$ specifies a $\mu\nu$ $\Re$otation algorithm A2 and second, both A1 and A2 are mathematically equivalent*[1]. □

Consequently, the set of $\kappa\lambda$ $\Re$otation algorithms can be thought of as a subset of the set of the $\mu\nu$ $\Re$otations. Furthermore, (18) provides a means of mapping a $\kappa\lambda$ $\Re$otation onto a $\mu\nu$ $\Re$otation. For example, one can verify that the square root and division-free algorithm in [4] is a $\mu\nu$ $\Re$otation and is obtained for

$$\mu = \frac{k_a\beta^2 a_1^2 + k_b b_1^2}{k_a k_b}, \quad \nu = 1.$$

In Fig. 1, we draw a graph that summarizes the relations among the classes of algorithms based on QR decomposition, a $\Re$otation algorithm, a $\mu\nu$ $\Re$otation and a $\kappa\lambda$ $\Re$otation.

[1]They evaluate logically equivalent equations.

3 RLS Algorithm and Architecture

In this Section, we consider the $\kappa\lambda$ $\Re$otation for optimal residual and weight extraction using systolic array implementation. Detailed comparisons with existing approaches are presented.

3.1 A Novel Fast Algorithm for the RLS Optimal Residual Computation

The QR-decomposition of the data at time instant n is as follows:

$$\begin{bmatrix} R(n) & u(n) \\ 0^T & v(t_n) \end{bmatrix} = T(n) \begin{bmatrix} \beta R(n-1) & \beta u(n-1) \\ X(t_n) & y(t_n) \end{bmatrix}, \tag{19}$$

where $T(n)$ is a unitary matrix of size $(p+1)\times(p+1)$ that performs a sequence of p Givens rotations. This can be written symbolically as

$$\begin{bmatrix} L(n)^{-\frac{1}{2}} & 0 \\ 0^T & l_q(n)^{-\frac{1}{2}} \end{bmatrix} \begin{bmatrix} \beta\bar{R}(n) & \beta\bar{u}(n) \\ \bar{X}(t_n) & \bar{y}(t_n) \end{bmatrix}$$

$$\xrightarrow{T(n)} \begin{bmatrix} L(n+1)^{-\frac{1}{2}} & 0 \\ 0^T & l_q(n+1)^{-\frac{1}{2}} \end{bmatrix} \begin{bmatrix} \beta\bar{R}(n+1) & \beta\bar{u}(n+1) \\ 0^T & b_{p+1}^{(p)} \end{bmatrix}, \tag{20}$$

where

$$\begin{aligned} L(\cdot)^{-\frac{1}{2}}\bar{R}(\cdot) &= R(\cdot), & L(\cdot)^{-\frac{1}{2}}\bar{u}(\cdot) &= u(\cdot), \\ l_q(n)^{-\frac{1}{2}}\bar{X}(t_n) &= X(t_n), & l_q(n)^{-\frac{1}{2}}\bar{y}(t_n) &= y(t_n) \end{aligned} \tag{21}$$

and

$$L(n) = \text{diag}\{l_1, l_2, \cdots, l_p\} \qquad L(n+1) = \text{diag}\{l'_1, l'_2, \cdots, l'_p\},$$

$$\bar{R}(n) = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1p} \\ & a_{22} & \cdots & a_{2p} \\ & & \ddots & \vdots \\ & & & a_{pp} \end{bmatrix} \qquad \bar{R}(n+1) = \begin{bmatrix} a'_{11} & a'_{12} & \cdots & a'_{1p} \\ & a'_{22} & \cdots & a'_{2p} \\ & & \ddots & \vdots \\ & & & a'_{pp} \end{bmatrix},$$

$$\bar{u}(n) = [a_{1,p+1}\ a_{1,p+1}\ \cdots\ a_{p,p+1}]^T \qquad \bar{u}(n+1) = \left[a'_{1,p+1}\ a'_{1,p+1}\ \cdots\ a'_{p,p+1}\right]^T,$$

$$[\bar{X}(t_n)\ \ \bar{y}(t_n)] = [b_1\ b_2\ \cdots\ b_p\ b_{p+1}].$$

Equations (12)-(15) imply that the i^{th} $\Re$otation is specified as follows:

$$l_i' = l_i l_q^{(i-1)} (l_q^{(i-1)} \beta^2 a_{ii}^2 + l_i b_i^{(i-1)^2}) \kappa_i^2 \tag{22}$$

$$l_q^{(i)} = (l_q^{(i-1)} \beta^2 a_{ii}^2 + l_i b_i^{(i-1)^2}) \lambda_i^2, \tag{23}$$

$$a_{ij}' = \kappa_i (l_q^{(i-1)} \beta^2 a_{ii} a_{ij} + l_i b_i^{(i-1)} b_j^{(i-1)}), \quad j = i, i+1, \cdots, p+1 \tag{24}$$

$$b_j^{(i)} = \lambda_i \beta (-b_i^{(i-1)} a_{ij} + a_{ii} b_j^{(i-1)}), \quad j = i+1, i+2, \cdots, p+1, \tag{25}$$

where $i = 1, 2, \cdots, p$, $b_j^{(0)} = b_j$, $j = 1, \cdots, p+1$ and $l_q^{(0)} = l_q$. For the optimal residual we have:

Lemma 3.1 *If the parametric $\kappa\lambda$ $\Re$otation is used in the QRD-RLS algorithm, the optimal residual is given by the expression*

$$e_{RLS}(t_n) = -\left(\prod_{i=1}^{p-1} \lambda_i \beta a_{ii}\right) \frac{\kappa_p \beta a_{pp}}{\lambda_p a_{pp}'} b_{p+1}^{(p)} v, \tag{26}$$

where $v = 1/\sqrt{l_q}$ if p is an even number and $v = \sqrt{l_q}$ if p is an odd number. □

The proof is given in the Appendix.

Here, l_q is a free variable. If we choose $l_q = 1$ we get $v = 1$ for both even and odd values of p and we can avoid the square root operation. We can see that for a recursive computation of (26) only one division operation is needed at the last step of the recursion. This compares very favorably with *the square root free fast algorithms* that require one division for every recursion step, as well as with the original approach (62), which involves one division and one square root operation for every recursion step.

The division operation in (26) cannot be avoided by proper choice of expressions for the parameters κ and λ. This is restated by the following Lemma, which is proved in the Appendix:

Lemma 3.2 *If a $\kappa\lambda$ $\Re$otation is used, the RLS optimal residual evaluation will require at least one division evaluation.* □

Note that the proper choice of the expression for the parameter λ_p, along with the rest of the parameters, is an open question, since the minimization of the multiplication operations, as well as communication and stability issues have to be considered.

3.2 A Systolic Architecture for the Optimal RLS Residual Evaluation

McWhirter has used a systolic architecture for the implementation of the QR decomposition [14]. This architecture is modified, so that equations (22)-(26) be evaluated for the special case of $\kappa_i = \lambda_i = 1, i = 1, 2, \cdots, p$ and $l_q = 1$. The systolic array, as well as the memory and the communication links of its components, are depicted in Fig. 2 [2]. The boundary cells (cell number 1) are responsible for evaluating (22) and (23), as well as the coefficients $\bar{c}_i = l_q^{(i-1)} a_{ii}$ and $\bar{s}_i = l_i b_i^{(i-1)}$ and the partial products $e_i = \prod_{j=1}^{i}(\beta a_{jj})$. The internal cells (cell number 2) are responsible for evaluating (24) and (25). Finally, the output cell (cell number 3) evaluates (26). The functionality of each one of the cells is described in Fig. 2. We will call this systolic array $S1.1$.

On Table 1, we collect some features of the systolic structure $S1.1$ and the two structures, $S1.2$ and $S1.3$, in [14] that are pertinent to the circuit complexity. The $S1.2$ implements the square-root-free QRD-RLS algorithm with $\mu = \nu = 1$, while $S1.3$ is the systolic implementation based on the original Givens rotation. In Table 1, the complexity per processor cell and the number of required processor cells are indicated for each one of the three different cells [3]. One can easily observe that $S1.1$ requires only one division operator and no square root operator, $S1.2$ requires p division operators and no square root operator, while $S1.3$ requires p division and p square root operators. This reduction of the complexity in terms of division and square root operators is penalized with the increase of the number of the multiplications and the communication links that are required.

Apart from the circuit complexity that is involved in the implementation of the systolic structures, another feature of the computational complexity is the number of *operations-per-cycle*. This number determines the minimum required delay between two consecutive sets of input data. For the structures $S1.2$ and $S1.3$ the boundary cell (cell number 1) constitutes the bottleneck of the computation and therefore it determines the operations-per-cycle that are shown on Table 5. For the structure $S1.1$ either the boundary cell or the output cell are the bottleneck of the computation.

[2]Note the aliases: $l_q^{(i-1)} \equiv \sigma_{in}, l_q^{(i)} \equiv \sigma_{out}, l_i \equiv l, a_{ij} \equiv r, b_j^{(i-1)} \equiv b_{in}, b_j^{(i)} \equiv b_{out}$, $e_{i-1} \equiv e_{in}, e_i \equiv e_{out}$.

[3]The multiplications with the constants β and β^2 are not encountered.

3.3 A Systolic Architecture for the Optimal RLS Weight Extraction

Shepherd et al. [17] and Tang et al. [19] have independently shown that the optimal weight vector can be evaluated in a recursive way. More specifically, one can compute recursively the term $R^{-H}(n)$ by

$$\begin{bmatrix} R^{-H}(n) \\ \# \end{bmatrix} = T(n) \begin{bmatrix} \frac{1}{\beta} R^{-H}(n-1) \\ 0^T \end{bmatrix} \tag{27}$$

and then use parallel multiplication for computing $w^T(n)$ by

$$w^T(n) = u^T(n) \left(R^{-H}(n) \right)^* . \tag{28}$$

The symbol # denotes a term of no interest. The above algorithm can be implemented by a fully pipelined systolic array that can operate in two distinct modes, 0 and 1. The initialization phase consists of $2p$ steps for each processor. During the first p steps the processors operate in mode 0 in order to calculate a full rank matrix R. During the following p steps, the processors operate in mode 1 in order to compute R^{-H}, by performing a task equivalent to forward substitution. After the initialization phase the processors operate in mode 0. In [17] one can find the systolic array implementations based both on the original Givens rotation and the Gentleman's variation of the square-root-free $\Re$otation, that is, the $\mu\nu$ $\Re$otation for $\mu = \nu = 1$. We will call these two structures $S2.3$ and $S2.2$ respectively.

In Fig. 3, we present the systolic structure $S2.1$ based on the $\kappa\lambda$ $\Re$otation with $\kappa_i = \lambda_i = 1, i = 1, 2, \cdots, p$. This is a square-root-free and division-free implementation. The boundary cells (cell number 1) are slightly simpler than the corresponding ones of the array $S1.1$. More specifically, they do not compute the partial products e_i. The internal cells (cell number 2), that compute the elements of the matrix R, are identical to the corresponding ones of the array $S1.1$. The cells that are responsible for computing the vector u (cell number 3) differ from the other internal cells only in the fact that they communicate their memory value with their right neighbors. The latter (cell number 4) are responsible for evaluating (28) and (27). The functionality of the processing cells, as well as their communication links and their memory contents, are given in Fig. 3. The mode of operation of each cell is controlled by the *mode bit* provided from the input. For a more detailed description of the operation of the mode bit one can see [15] and [17].

On Tables 2 and 5, we collect some computational complexity metrics for the systolic arrays $S2.1$, $S2.2$ and $S2.3$, when they operate in mode 0[4]. The conclusions we can draw are similar to the ones we had for the circuits that calculate the optimal residual: the square root operations and the division operations can be eliminated with the cost of an increased number of multiplication operations and communication links. We should also note that $S2.1$ does require the implementation of division operators in the boundary cells, since these operators are used during the initialization phase. Nevertheless, after the initialization phase the circuit will not suffer from any time delay caused by division operations. The computational bottleneck of all three structures, $S2.1$, $S2.2$ and $S2.3$, is the boundary cell, thus it determines the operations-per-cycle metric.

As a conclusion for the RLS architectures, we observe that the figures on Tables 1, 2 and 5 favor the architectures based on the $\kappa\lambda$ $\Re$otation, $\kappa = \lambda = 1$ versus the ones that are based on the $\mu\nu$ rotation with $\mu = \nu = 1$ and the standard Givens rotation. This claim is clearly substantiated by the delay times on Table 5, associated to the DSP implementation of the QRD-RLS algorithm. These delay times are calculated on the basis of the manufacturers benchmark speeds for floating point operations [18]. The readers may have to bear in mind that the weight extraction of [17] is not a good form due to the updating of R^{-1} if the weight vector at each time instant is required.

4 CRLS Algorithm and Architecture

The optimal weight vector $w^i(n)$ and the optimal residual $e^i_{CRLS}(t_n)$ that correspond to the i^{th} constraint vector c^i are given by the expressions [15]:

$$w^i(n) = \frac{r^i}{\parallel z^i(n) \parallel^2} R^{-1}(n) z^i(n) \tag{29}$$

and

$$e^i_{CRLS}(t_n) = \frac{r^i}{\parallel z^i(n) \parallel^2} \hat{e}^i_{CRLS}(t_n), \tag{30}$$

where

$$\hat{e}^i_{CRLS}(t_n) = X(t_n) R^{-1}(n) z^i(n). \tag{31}$$

[4]The multiplications with the constants $\beta, \beta^2, 1/\beta$ and $1/\beta^2$, as well as the communication links that drive the mode bit, are not encountered.

The term $z^i(n)$ is defined as follows

$$z^i(n) = R^{-H}(n)c^i \tag{32}$$

and it is computed with the recursion [15]

$$\begin{bmatrix} z^i(n) \\ \# \end{bmatrix} = T(n) \begin{bmatrix} \frac{1}{\beta} z^i(n-1) \\ 0^T \end{bmatrix}, \tag{33}$$

where the symbol # denotes a term of no interest. In this Section, we derive a variation of the recursion that is based on the parametric $\kappa\lambda$ $\Re$otation. Then, we design the systolic arrays that implement this recursion for $\kappa = \lambda = 1$. We also make a comparison of these systolic structures with those based on the Givens rotation and the $\mu\nu$ $\Re$otation introduced by Gentleman [6, 2, 15, 17].

From (32) and (21) we have $z^i(n) = \left(L(n)^{-1/2}\bar{R}(n)\right)^{-H} c^i$ and since $L(n)$ is a diagonal real valued matrix we get $z^i(n) = L(n)^{1/2}\bar{R}(n)^{-H}c^i$, where c^i is the constraint direction. If we let

$$\bar{z}^i(n) = L(n)\bar{R}(n)^{-H}c^i \tag{34}$$

we obtain

$$z^i(n) = L(n)^{-1/2}\bar{z}^i(n). \tag{35}$$

From (35) we get $\| z^i(n) \|^2 = \bar{z}^{i^H}(n)L^{-1}(n)\bar{z}^i(n)$. Also, from (21) and (35) we get $R^{-1}(n)z^i(n) = \bar{R}^{-1}(n)\bar{z}^i(n)$. Consequently, from (30), (31) and (29) we have

$$e^i_{CRLS}(n) = \frac{r^i}{\bar{z}^{i^H}(n)L^{-1}(n)\bar{z}^i(n)}\hat{e}^i_{CRLS}(n) \tag{36}$$

and

$$w^i(n) = \frac{r^i}{\bar{z}^{i^H}(n)L^{-1}(n)\bar{z}^i(n)}\bar{R}^{-1}(n)\bar{z}^i(n), \tag{37}$$

where

$$\hat{e}^i_{CRLS}(n) = X(n)\bar{R}^{-1}(n)\bar{z}^i(n). \tag{38}$$

Because of the similarity of (31) with (38) and (29) with (37) we are able to use a variation of the systolic arrays that are based on the Givens rotation [15, 17] in order to evaluate (36)-(37).

4.1 Systolic Architecture for the Optimal CRLS Residual Evaluation

From (26) and (36), if $l_q = 1$, we get the optimal residual

$$e^i_{CRLS}(n) = -\frac{r^i}{\bar{z}^{i^H}(n)L^{-1}(n)\bar{z}^i(n)} \left(\prod_{j=1}^{p-1} \lambda_j a_{jj}\right) \frac{\kappa_p a_{pp}}{\lambda_p a'_{pp}} b^{(p)}_{p+1}. \tag{39}$$

In Fig. 4, we present the systolic array $S3.1$, that evaluates the optimal residual for $\kappa_j = \lambda_j = 1, j = 1, 2, \cdots, p$, and the number of constraints is $N = 2$. This systolic array is based on the design proposed by McWhirter [15]. It operates in two modes and is in a way very similar to the operation of the systolic structure $S2.1$ (see Section 3). The recursive equations for the data of the matrix $\bar{R}$ are given in (22)-(25). They are evaluated by the boundary cells (cell number 1) and the internal cells (cell number 2). These internal cells are identical to the ones of the array $S2.1$. The boundary cells have a very important difference from the corresponding ones of $S2.1$: while they operate in mode 0, they make use of their division operators in order to evaluate the elements of the diagonal matrix $L^{-1}(n)$, i.e. the quantities $1/l_i, i = 1, 2, \cdots, p$. These quantities are needed for the evaluation of the term $\bar{z}^{i^H}(n)L(n)^{-1}\bar{z}^i(n)$ in (39). The elements of the vectors $\bar{z}^1$ and $\bar{z}^2$ are updated by a variation of (24) and (25), for which the constant β is replaced by $1/\beta$. The two columns of the internal cells (cell number 3) are responsible for these computations. They initialize their memory value during the second phase of the initialization (mode 1) according to (34). While they operate in mode 0, they are responsible for evaluating the partial sums

$$\eta_k = \sum_{j=1}^{k} \parallel \bar{z}^i_j \parallel^2 / l_j. \tag{40}$$

The output cells (cell number 4) are responsible for the final evaluation of the residual[5].

McWhirter has designed the systolic arrays that evaluate the optimal residual, based on either the Givens rotation or the square-root-free variation that was introduced by Gentleman [15, 2]. We will call these systolic arrays $S3.3$ and $S3.2$ respectively. On Tables 3 and 5 we collect some computational complexity metrics for the systolic arrays $S3.1$, $S3.2$ and $S3.3$, when they operate in mode 0 [6]. We observe that the $\mu\nu$ $\Re$otation-based $S3.2$, outperforms the $\kappa\lambda$ $\Re$otation-based $S3.1$.

[5]Note the alias $r^i \equiv r$.

[6]The multiplications with the constants $\beta, \beta^2, 1/\beta$ and $1/\beta^2$, as well as the communication links that drive the mode bit, are not encountered.

The two structures require the same number of division operators, while $S3.2$ needs less multipliers and also it has less communication overhead.

4.2 A Systolic Architecture for the Optimal CRLS Weight Vector Extraction

In Fig. 5, we present the systolic array that evaluates (37) for $\kappa_j = \lambda_j = 1, j = 1, 2, \cdots, p$ and the number of constraints equal to $N = 2$. This systolic array operates in two modes, just as the arrays $S2.1$ and $S3.1$ do. The boundary cell (cell number 1) is responsible for evaluating the diagonal elements of the matrices R and L, the variable l_q, as well as all the coefficients that will be needed in the computations of the internal cells. In mode 0 its operation is identical to the operation of the boundary cell in $S2.1$, while in mode 1 it behaves like the corresponding cell of $S3.1$. The internal cells in the left triangular part of the systolic structure (cell number 2) evaluate the non-diagonal elements of the matrix R and they are identical to the corresponding cells of $S3.1$. The remaining part of the systolic structure is a 2-layer array. The cells in the first column of each layer (cell number 3) are responsible for the calculation of the vector z^i and the partial summations (40). They also communicate their memory values to their right neighbors. The latter (cell number 4) evaluate the elements of the matrix R^{-H} and they are identical to the corresponding elements of $S2.1$. The output elements (cell number 5) are responsible for the normalization of the weight vectors and they compute the final result.

Shepherd et al. [17] and Tang et al. [19] have designed systolic structures for the weight vector extraction based on the Givens rotation and the square-root-free $\Re$otation of Gentleman [2]. We will call these two arrays $S4.3$ and $S4.2$ respectively. On Tables 4 and 5, we show the computational complexity metrics for the systolic arrays $S4.1$, $S4.2$ and $S4.3$, when they operate in mode 0. The observations we make are similar to the ones we have for the systolic arrays that evaluate the RLS weight vector (see Section 3).

Note that each part of the 2-layer structure computes the terms relevant to one of the two constraints. In the same way, a problem with N constraints will require an N-layer structure. With this arrangement of the multiple layers we obtain a unit time delay between the evaluation of the weight vectors for the different constraints. The price we have to pay is the global wiring for some of the communication links of cell 3. A different approach can also be considered: we may place the multiple layers side by side, one on the right of the other. In this way, not only the global wiring will be avoided, but also the number of communication links of cell 3, will be considerably reduced. The price we will pay with this approach is a time delay of p units between consequent evaluations of the weight vectors for different constraints.

As a conclusion for the CRLS architectures, we observe that the figures on Tables 3, 4 and 5 favor the architectures based on the $\mu\nu$ $\Re$otation, $\mu = \nu = 1$ versus the ones that are based on the $\kappa\lambda$ rotation with $\kappa = \lambda = 1$.

5 Dynamic Range, Stability, and Error Bounds

Both the $\kappa\lambda$ and $\mu\nu$ $\Re$otation algorithms enjoy computational complexity advantages compared to the standard Givens rotation with the cost of the denormalization of the latter. Consequently, the numerical stability of the QRD architectures based on these algorithms can be questioned. Furthermore, a crucial piece of information in the circuit design is the wordlength, that is the number of bits per word required to ensure correct operations of the algorithm without overflow. At the same time, the wordlength has large impact on the complexity and the speed of the hardware implementation. In this Section, we address issues on stability, error bounds and lower bounds for the wordlength by means of dynamic range analysis. We focus on the algorithm for RLS optimal residual extraction based on a $\kappa\lambda$ $\Re$otation. The dynamic range of the variables involved in the other newly introduced algorithms can be computed in a similar way.

In [13], Liu et al. study the dynamic range of the QRD-RLS algorithm that utilizes the standard Givens rotation. This study is based on the fact that the rotation parameters generated by the boundary cells of the systolic QRD-RLS structure eventually reach a *quasi-steady-state* regardless of the input data statistics, provided that the forgetting factor β is close to one. A worst case analysis of the steady state dynamic range reveals the bound [13]

$$\lim_{n\to\infty} |r_{ij}(n)| \leq \frac{(2\beta)^{i-1}}{\sqrt{1-\beta^2}} |x_{max}| \stackrel{\triangle}{=} \mathcal{R}_i^r, \quad j = i, i+1, \cdots, p+1 \tag{41}$$

for the contents of the processing elements of the i^{th} row in the systolic structure, $i = 1, 2, \cdots, p$, where $|x_{max}|$ is the largest value in the input data. Similarly, at the steady state the output of the

i^{th} row $x_j^{(i)}, j = i, i+1, \cdots, p+1$ is bounded by [13]

$$\lim_{n\to\infty} \left| x_j^{(i)}(n) \right| \leq (2\beta)^{i-1} \left| x_{max} \right| \triangleq \mathcal{R}_i^x, \quad j = i+1, i+2, \cdots, p+1. \tag{42}$$

Furthermore, the optimal residual e_{RLS} is bounded by [13]

$$\lim_{n\to\infty} \left| e_{RLS}(n) \right| \leq (2\beta)^{p-1} \left| x_{max} \right| \triangleq \mathcal{R}_i^{err}.$$

The latter is a BIBO stability result that applies also for the QRD-RLS algorithm based on a $\kappa\lambda$ $\Re$otation. Nevertheless, the internal stability is not guaranteed. More concretely, the terms involved in the QRD-RLS algorithm may not be upper bounded.

In view of the internal stability problem, a proper choice of the parameters κ and λ should be made. A correct choice will compensate for the denormalization of the type

$$r'_{ij} = \frac{1}{\sqrt{l'_i}} a'_{ij}, \quad x_j^{(i)} = \frac{1}{\sqrt{l_q^{(i)}}} b_j^{(i)}, \tag{43}$$

where l'_i and $l_q^{(i)}$ are given in (22) and (23) respectively. The terms κ_i^2 and λ_i^2 in (22) and (23) can be used as shift operators by choosing

$$\kappa_i = 2^{-\rho_i} \quad \text{and} \quad \lambda_i = 2^{-\tau_i}, \quad i = 1, 2, \cdots, p, \tag{44}$$

where ρ_i and τ_i take integer values. For instance, in (23), if $\tau_i > 0$ the effect of λ_i^2 on $(l_q^{(i-1)} \beta^2 a_{ii}^2 + l_i b_i^{(i-1)^2})$ will be a right shift of $2\tau_i$ bits. We can ensure that

$$0.5 \leq l'_i < 2 \quad \text{and} \quad 0.5 \leq l_q^{(i)} < 2, \quad i = 1, 2, \cdots, p \tag{45}$$

by forcing the most significant bit (MSB) of the binary representation to be either at position 2^0 or 2^{-1} after the shift operation. This normalization task has been introduced in [1] and further used in [4]. It can be described in analytic terms by the expression

$$\text{shift\_amount(unnormalized\_quantity)} = \lfloor \{\log_2 (\text{unnormalized\_quantity}) + 1\} /2 \rfloor$$

and it can be implemented very easily in hardware.

In the sequel, we consider the $\kappa\lambda$ $\Re$otation by choosing:

$$\begin{aligned} \rho_i &= \text{shift\_amount}\left[l_i l_q^{(i-1)}(l_q^{(i-1)}\beta^2 a_{ii}^2 + l_i b_i^{(i-1)^2})\right] \\ \tau_i &= \text{shift\_amount}\left[(l_q^{(i-1)}\beta^2 a_{ii}^2 + l_i b_i^{(i-1)^2})\right], \end{aligned} \tag{46}$$

for $i = 1, 2, \cdots, p$ [4]. Note that (46) along with (44) should precede (22)-(25) in the rotation algorithm. In conformity to [1] and [4] we will refer to the resulting rotation algorithm with the name *scaled rotation.*

The systolic array that implements the QRD-RLS algorithm for the optimal residual extraction is depicted in Fig. 6. A comparison of this systolic array with the one in Fig. 2 is summarized by the following points: The boundary cells generate the shift quantities ρ and τ associated with the parameters κ and λ respectively, and they communicate them horizontally with the internal cells. This yields two additional links for the boundary cells and four additional ones for the internal cells. In the dynamic range study that follows, we show that the number of bits these links occupy is close to the logarithm of the number of bits required by the rest of the links. The boundary cells are also responsible for computing the quantities $\prod_{i=1}^{p} \beta a_{ii}$ and $\prod_{i=1}^{p-1} \lambda_i$ in (26). In this case, λ_i is an exponential term according to (44), so the above product can be computed as the running sum of the exponents

$$g_i = \sum_{k=1}^{i} \tau_k, \qquad i = 1, 2, \cdots, p-1, \tag{47}$$

yielding an additional adder for the boundary cells. Finally, as far as the boundary cells are concerned, we observe that the cell at position (p, p) of the systolic array is not identical to the rest of the boundary cells. This is a direct consequence of (26). On the other hand, the shift operators constitute the only overhead of the internal and the output cells compared with the corresponding ones in Fig. 2. Overall, the computational complexity (in terms of operator counts) is slightly higher than that of the systolic array with $\kappa = \lambda = 1$.

Let us focus now on the dynamic range of the variables in the systolic array. By solving (43) for a'_{ij} and using (41) and (45) one can compute an upper bound for a_{ij} at the steady state, thus one can specify the dynamic range of the i^{th} row cell content. A similar result can be obtained for the output of the i^{th} row by using (42), (43) and (45). The results are summarized by the following

Lemma:

Lemma 5.1 *The steady state dynamic range of the cell content $\mathcal{R}_i^a$ and output range $\mathcal{R}_i^b$ in the i^{th} row are given by*

$$\lim_{n\to\infty} |a_{ij}(n)| \leq \mathcal{R}_i^a \triangleq \sqrt{2}\mathcal{R}_i^r \quad \textit{and} \quad \lim_{n\to\infty} \left|b_j^{(i)}(n)\right| \leq \mathcal{R}_i^b \triangleq \sqrt{2}\mathcal{R}_i^x \tag{48}$$

respectively.

The lower bounds in the wordlength come as a direct consequence of Lemma 5.1: The wordlength of the cell content $\mathcal{B}_i^a$ and output $\mathcal{B}_i^b$ in the i^{th} row must be lower bounded by

$$\mathcal{B}_i^a \geq \lceil \mathcal{B}_i^r + 0.5 \rceil \quad \text{and} \quad \mathcal{B}_i^b \geq \lceil \mathcal{B}_i^x + 0.5 \rceil \tag{49}$$

respectively, where $\mathcal{B}_i^r = \lceil \log_2 \mathcal{R}_i^r \rceil$ and $\mathcal{B}_i^x = \lceil \log_2 \mathcal{R}_i^x \rceil$ are the corresponding wordlength lower bounds for the QRD-RLS implementation based on the standard Givens rotation.

The parameters κ_i, λ_i are communicated via their exponents ρ_i and τ_i. The dynamic ranges of these exponents are given by Lemma 5.2 which is proved in the Appendix.

Lemma 5.2 *The steady state dynamic range of the terms ρ_i and τ_i at the i^{th} row $\mathcal{R}_i^\rho$ and $\mathcal{R}_i^\tau$ are given by*

$$\begin{aligned} \lim\nolimits_{n\to\infty} |\rho_i| \leq \mathcal{R}_i^\rho \triangleq \mathcal{B}_i^a + 2.5 \\ \lim\nolimits_{n\to\infty} |\tau_i| \leq \mathcal{R}_i^\tau \triangleq \mathcal{B}_i^a + 1.5 \end{aligned} \tag{50}$$

respectively[7].

Consequently, the lower bounds on the wordlength $\mathcal{B}_i^\rho$ and $\mathcal{B}_i^\tau$ of ρ_i and τ_i are

$$\mathcal{B}_i^\rho \geq \lceil \log(\mathcal{B}_i^a + 2.5) \rceil \quad \text{and} \quad \mathcal{B}_i^\tau \geq \lceil \log(\mathcal{B}_i^a + 1.5) \rceil \tag{51}$$

respectively.

For the computation of the optimal residual the boundary cells need to evaluate both the running product $e_i = \prod_{k=1}^{i} \beta a_{kk}$ and the running sum in (47). The dynamic ranges for these terms are given by the following Lemma:

[7]For the sake of simplicity in notation we have dropped the time parameter n from the expression in the limit argument.

Lemma 5.3 *The steady state dynamic range of the terms e_i and g_i at the i^{th} row $\mathcal{R}_i^e$ and $\mathcal{R}_i^g$ are given by*

$$\begin{array}{l} \lim_{n\to\infty} |e_i| \leq \mathcal{R}_i^e \triangleq \prod_{k=1}^{i} \mathcal{R}_k^a \\ \lim_{n\to\infty} |g_i| \leq \mathcal{R}_i^g \triangleq i\mathcal{B}_1^a + \frac{i(i+2)}{2} \end{array} \tag{52}$$

respectively.

The proof is given in the Appendix. With simple algebraic manipulations one can show that the corresponding lower bounds on wordlength $\mathcal{B}_i^e$ and $\mathcal{B}_i^g$ of e_i and g_i are

$$\mathcal{B}_i^e \geq \sum_{k=1}^{i} \mathcal{B}_k^a \quad \text{and} \quad \mathcal{B}_i^\tau \geq \max\{\lceil \log \mathcal{B}_1^a + \log i + 1\rceil, \lceil \log i + \log(i+2)\rceil\} \tag{53}$$

respectively.

Finally, consider the coefficients defined as

$$\begin{array}{ll} \bar{c}_i = l_q^{(i-1)}\beta^2 a_{ii} & \bar{s}_i = l_i b_i^{(i-1)} \\ \hat{c}_i = \beta a_{ii} & \hat{s}_i = \beta b_i^{(i-1)}, \end{array}$$

that describe the information exchanged by the remaining horizontal links in the systolic array (cf. Fig. 6). One can easily show that the steady state dynamic range of these coefficients, denoted by $\mathcal{R}_i^{\bar{c}}, \mathcal{R}_i^{\bar{s}}, \mathcal{R}_i^{\hat{c}}$ and $\mathcal{R}_i^{\hat{s}}$ respectively are

$$\begin{array}{ll} \lim_{n\to\infty} |\bar{c}_i| \leq \mathcal{R}_i^{\bar{c}} \triangleq 2\mathcal{R}_i^a & \lim_{n\to\infty} |\bar{s}_i| \leq \mathcal{R}_i^{\bar{s}} \triangleq 2\mathcal{R}_i^b \\ \lim_{n\to\infty} |\hat{c}_i| \leq \mathcal{R}_i^{\hat{c}} \triangleq \mathcal{R}_i^a & \lim_{n\to\infty} |\hat{s}_i| \leq \mathcal{R}_i^{\hat{s}} \triangleq \mathcal{R}_i^b. \end{array} \tag{54}$$

The implied wordlength lower bounds are $\mathcal{B}_i^{\bar{c}} \geq \mathcal{B}_i^a + 1$, $\mathcal{B}_i^{\bar{s}} \geq \mathcal{B}_i^b + 1, \mathcal{B}_i^{\hat{c}} \geq \mathcal{B}_i^a$ and $\mathcal{B}_i^{\hat{s}} \geq \mathcal{B}_i^b$ respectively.

In summary, (45), (48), (50), (52) and (54) show that all the internal parameters are bounded and therefore the algorithm is stable. Furthermore, the lower bounds on the wordlength provide the guidelines for an inexpensive, functionally correct realization.

The error bound of the whole QRD to a given matrix $A \in \Re^{m\times n}$ due to floating point operations is given by [1, 4]

$$||\delta A|| \leq \tau(m+n-3)(1+\tau)^{m+n-4}||A|| + O(\epsilon^2), \tag{55}$$

where τ is the upper bound and ϵ is the largest number such that $1+\epsilon$ is computed as 1. If (44) and (45) are satisfied, for $\kappa = \lambda = 1$, then it follows that $\tau = 6.5\epsilon$ [4]. This is fairly close to the standard Givens rotation which has $\tau = 6.0\epsilon$ [4].

6 Conclusion

We introduced *the parametric $\kappa\lambda$ $\Re$otation*, which is a square-root-free and division-free algorithm, and showed that the parametric $\kappa\lambda$ $\Re$otation describes a subset of the $\mu\nu$ $\Re$otation algorithms [8]. We then derived novel architectures based on the $\kappa\lambda$ $\Re$otation for $\kappa = \lambda = 1$ and made a comparative study with the standard Givens rotation and the $\mu\nu$ $\Re$otation with $\mu = \nu = 1$. Finally, a dynamic range study is pursued. It is observed that considerable improvements can be obtained for the implementation of some QRD-based algorithms.

We pointed out the trade-offs between the architectures based on the above $\Re$otations. Our analysis suggests the following decision rule for selecting between the architectures that are based on the $\mu\nu$ $\Re$otation and the $\kappa\lambda$ $\Re$otation : *Use the $\mu\nu$ $\Re$otation-based architectures, with $\mu = \nu = 1$, for the constrained minimization problems and the $\kappa\lambda$ $\Re$otation-based architectures, with $\kappa = \lambda = 1$, for the unconstrained minimization problems.* Table 5 shows the benchmark comparisons of different algorithms using different DSP processors and it confirms the properties claimed in this paper.

A number of obscure points relevant to the realization of the QRD-RLS and the QRD-CRLS algorithms are clarified. Some systolic structures that are described in this paper are very promising, since they require less computational complexity (in various aspects) from the structures known to date and they make the VLSI implementation easier.

A Appendix

Proof of Lemma 3.1:

First, we derive some equations that will be used in the course of the optimal residual computation.

If we solve (24), case $i = j = 1$, for $l_q\beta^2 a_{11}^2 + l_1 b_1^2$ and substitute in (22) we get

$$l_1' = l_1 l_q \frac{a_{11}'}{\kappa_1}\kappa_1^2$$

and therefore

$$\frac{l_1'}{l_1} = l_q a_{11}' \kappa_1. \tag{56}$$

If we solve (24), case $j = i$, for $l_q^{(i-1)}\beta^2 a_{ii}^2 + l_i b_i^{(i-1)^2}$ and substitute in (23) we get

$$l_q^{(i)} = \frac{\lambda_i^2 a_{ii}'}{\kappa_i}. \tag{57}$$

If we substitute the same expression in (22) we get

$$l_i' = l_i l_q^{(i-1)} a_{ii}' \kappa_i.$$

In the above expression we substitute $l_q^{(i-1)}$ from (57), and solve for l_i'/l_i to obtain

$$\frac{l_i'}{l_i} = \frac{\lambda_{i-1}^2 \kappa_i}{\kappa_{i-1}} a_{i-1,i-1}' a_{ii}'. \tag{58}$$

If we solve (22) for $l_q^{(i-1)}\beta^2 a_{ii}^2 + l_i b_i^{(i-1)^2}$ and substitute in (23) we get

$$l_q^{(i)} = \frac{\lambda_i^2}{\kappa_i^2} \frac{l_i'}{l_i} \frac{1}{l_q^{(i-1)}}. \tag{59}$$

Also, we note that (4) implies that

$$c_i = \beta r_{ii} / r_{ii}'$$

and by substituting (9) we obtain

$$c_i = \frac{\beta a_{ii}}{a_{ii}'} \sqrt{\frac{l_i'}{l_i}} \quad i = 1, 2, \cdots, p+1. \tag{60}$$

Similarly, from (4) and (9), we get

$$s_i = \frac{b_i^{(i-1)}}{a_{ii}'} \sqrt{\frac{l_i'}{l_q^{(i-1)}}} \quad i = 1, 2, \cdots, p+1. \tag{61}$$

The optimal residual for the RLS problem is [6]

$$e_{RLS}(t_n) = -\left(\prod_{k=1}^{p} c_k\right) v(t_n). \tag{62}$$

The expressions in (20) and (19) imply

$$v(t_n) = \frac{1}{\sqrt{l_q^{(p)}}} b_{p+1}^{(p)}.$$

If we substitute the above expressions of $v(t_n)$ and c_i in (62) we obtain

$$e_{RLS}(t_n) = -\prod_{i=1}^{p} \left(\frac{\beta a_{ii}}{a'_{ii}} \sqrt{\frac{l'_i}{l_i}} \right) \frac{1}{\sqrt{l_q^{(p)}}} b_{p+1}^{(p)}. \tag{63}$$

From (59) we get

$$l_q^{(p)} = \frac{\lambda_p^2}{\kappa_p^2} \frac{l'_p}{l_p} \frac{1}{l_q^{(p-1)}} = \frac{\lambda_p^2}{\kappa_p^2} \frac{l'_p}{l_p} \frac{\kappa_{p-1}^2}{\lambda_{p-1}^2} \frac{l_{p-1}}{l'_{p-1}} l_q^{(p-2)}$$

$$= \begin{cases} \prod_{j=1}^{k} \left(\frac{\lambda_{2j}^2}{\kappa_{2j}^2} \frac{l'_{2j}}{l_{2j}} \frac{\kappa_{2j-1}^2}{\lambda_{2j-1}^2} \frac{l_{2j-1}}{l'_{2j-1}} \right) l_q \quad , & p = 2k \\ \prod_{j=1}^{k-1} \left(\frac{\kappa_{2j}^2}{\lambda_{2j}^2} \frac{l_{2j}}{l'_{2j}} \frac{\lambda_{2j-1}^2}{\kappa_{2j-1}^2} \frac{l'_{2j-1}}{l_{2j-1}} \right) \frac{\lambda_p^2}{\kappa_p^2} \frac{l'_p}{l_p} \frac{1}{l_q} \quad , & p = 2k-1 \end{cases} . \tag{64}$$

Thus, from (63) and (64), for the case of $p = 2k$, we have

$$e_{RLS}(t_n) = -\prod_{j=1}^{k} \left(\frac{\beta a_{2j,2j}}{a'_{2j,2j}} \sqrt{\frac{l'_{2j}}{l_{2j}}} \frac{\beta a_{2j-1,2j-1}}{a'_{2j-1,2j-1}} \sqrt{\frac{l'_{2j-1}}{l_{2j-1}}} \left(\frac{\lambda_{2j}^2}{\kappa_{2j}^2} \frac{l'_{2j}}{l_{2j}} \frac{\kappa_{2j-1}^2}{\lambda_{2j-1}^2} \frac{l_{2j-1}}{l'_{2j-1}} \right)^{-\frac{1}{2}} \right) \cdot \frac{1}{\sqrt{l_q}} \cdot b_{p+1}^{(p)}.$$

By doing the appropriate term cancellations and by substituting the expressions of $l'_i/l_i, i = 1, 2, \cdots, 2k$ from (56) and (58) we obtain the expression (26) for the optimal residual. Similarly, for the case of $p = 2k-1$, from (63) and (64) we obtain

$$e_{RLS}(t_n) = -\prod_{j=1}^{k-1} \left(\frac{\beta a_{2j,2j}}{a'_{2j,2j}} \sqrt{\frac{l'_{2j}}{l_{2j}}} \frac{\beta a_{2j-1,2j-1}}{a'_{2j-1,2j-1}} \sqrt{\frac{l'_{2j-1}}{l_{2j-1}}} \left(\frac{\lambda_{2j-1}^2}{\kappa_{2j-1}^2} \frac{l'_{2j-1}}{l_{2j-1}} \frac{\kappa_{2j}^2}{\lambda_{2j}^2} \frac{l_{2j}}{l'_{2j}} \right)^{-\frac{1}{2}} \right)$$

$$\times \frac{\beta a_{pp}}{a'_{pp}} \sqrt{\frac{l'_p}{l_p}} \sqrt{\frac{\kappa_p^2 l_p l_q}{\lambda_p^2 l'_p}} b_{p+1}^{(p)}$$

and by substituting (58) we get (26).

Proof of Lemma 3.2:

The question is whether we can avoid the division in the evaluation of the residual. Obviously we should choose the expressions of the parameters κ_p and λ_p so that the equation $\kappa_p = \lambda_p a'_{pp}$ or

$$\lambda_p = \kappa_p / a'_{pp}$$

holds. But, from (24), for $j = i$, we get

$$\kappa_p / a'_{pp} = \frac{1}{l_q^{(p-1)} \beta^2 a_{pp}^2 + l_p b_p^{(p-1)^2}}.$$

Therefore, if we choose to avoid the division operation in the expression of the residual, we will need to perform another division in order to evaluate the parameter λ_p.

Proof of Lemma 5.2:

From (45) and the fact that $0 < \beta < 1$ we get

$$l_q^{(i-1)} \beta^2 a_{ii}^2 + l_i b_i^{(i-1)^2} < 2 a_{ii}(n)^2 + 2 b_i^{(i-1)^2}.$$

Consequently, at the steady state we have

$$\lim_{n \to \infty} \left| l_q^{(i-1)} \beta^2 a_{ii}^2 + l_i b_i^{(i-1)^2} \right| < 2 \left(\mathcal{R}_i^a \right)^2 + 2 \left(\mathcal{R}_i^b \right)^2.$$

Also, (41), (42) and (48) imply that $\mathcal{R}_i^a > \mathcal{R}_i^b$. Therefore, we obtain the bound

$$\lim_{n \to \infty} \left| l_q^{(i-1)} \beta^2 a_{ii}^2 + l_i b_i^{(i-1)^2} \right| < 4 \left(\mathcal{R}_i^a \right)^2$$

and by utilizing (23) and the fact that $l_q^{(i)} \geq 0.5$ we get

$$\lim_{n \to \infty} \left| \lambda_i^{-2} \right| \leq 2 \lim_{n \to \infty} \left| l_q^{(i-1)} \beta^2 a_{ii}^2 + l_i b_i^{(i-1)^2} \right| < 8 \left(\mathcal{R}_i^a \right)^2. \tag{65}$$

By substituting the expression $\lambda_i = 2^{\tau_i}$, using (65) and solving the resulted inequality for τ_i we obtain

$$\lim_{n \to \infty} |\tau_i| \leq \log \mathcal{R}_i^a + 1.5.$$

The expression for the dynamic range of τ_i in (50) is a direct consequence of the above inequality.

Similarly, for the computation of the dynamic range of the term ρ_i first one can prove that

$$\lim_{n\to\infty} \left| l_i l_q^{(i)} (l_q^{(i-1)} \beta^2 a_{ii}^2 + l_i b_i^{(i-1)^2}) \right| < 16\,(\mathcal{R}_i^a)^2$$

and then compute an upper bound for ρ_i at the steady state based on (22), (44) and the fact that $l_i' \geq 0.5$.

Proof of Lemma 5.3:

Since $0 < \beta < 1$, for the term e_i we have

$$\lim_{n\to\infty} |e_i| = \beta^i \prod_{k=1}^{i} \lim_{n\to\infty} |a_{kk}| \leq \prod_{k=1}^{i} \mathcal{R}_k^a.$$

Similarly, for the term g_i we have

$$\lim_{n\to\infty} |g_i| = \sum_{k=1}^{i} \lim_{n\to\infty} |\tau_k|$$

and from (50)

$$\lim_{n\to\infty} |e_i| \leq \sum_{k=1}^{i} \mathcal{R}_k^\tau = \sum_{k=1}^{i} (\mathcal{B}_k^a + 1.5)\,. \tag{66}$$

(41) implies that the wordlength for the variable r should satisfy the inequality

$$\mathcal{B}_i^r \geq \lceil (i-1)(1+\log\beta) + C \rceil,$$

where C is constant with respect to i. Since $\beta < 1$, it is sufficient to have

$$\mathcal{B}_i^r \geq \lceil i - 1 + C \rceil,$$

or

$$\mathcal{B}_i^r \geq i - 1 + \mathcal{B}_1^r. \tag{67}$$

A similar formula can be derived for the wordlength of the contents of the the array that utilizes the scaled rotation, based on (49) and (67). More specifically, we have

$$\mathcal{B}_i^a \geq \mathcal{B}_1^a + i - 1.$$

From this inequality and (66) we get

$$\lim_{n\to\infty} |e_i| \le i\mathcal{B}_i^a + \frac{i(i-1)}{2} + 1.5i.$$

The dynamic range expression in (52) follows directly.

References

[1] J.L. Barlow and I.C.F. Ipsen. Scaled Givens rotations for the solution of linear least squares problems on systolic arrays. *SIAM J. Sci. Stat. Comput.*, 8(5):716–733, Sept. 1987.

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

[3] W.M. Gentleman and H.T. Kung. Matrix Triangularization by Systolic Arrays. In *Proc. SPIE 298, Real-Time Signal Processing IV*, pages 19–26, 1981.

[4] J. Gotze and U. Schwiegelshohn. A Square Root and Division Free Givens Rotation for Solving Least Squares Problems on Systolic Arrays. *SIAM J. Scie. and Stat. Comput.*, 12(4):800–807, July 1991.

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

[6] S. Haykin. *Adaptive Filter Theory.* 2nd Ed.,Prentice Hall, 1991.

[7] S.F. Hsieh, K.J.R. Liu, and K. Yao. Dual-state systolic architectures for up/downdating rls adaptive filtering. *IEEE Trans. on Circuits and Systems II*, 39(6):382–385, June 1992.

[8] S.F. Hsieh, K.J.R. Liu, and K. Yao. A Unified Approach for QRD-Based Recursive Least-Squares Estimation without Square Routs. *To appear in IEEE Trans. on Signal Processing*, May 1993. (Also in Proc. IEEE/ICASSP, pp.1017-1020, Toronto, May 1991).

[9] S. Kalson and K. Yao. Systolic Array Processing for Order and Time Recursive Generalized Least-Squares Estimation. In *Proc. SPIE 564, Real-Time Signal Processing VIII*, pages 28–38, 1985.

[10] F. Ling. Efficient Least Squares Lattice Algorithm Based on Givens Rotations with Systolic Array Implementation. *IEEE Trans. Signal Processing*, 39:1541–1551, July 1991.

[11] F. Ling, D. Manolakis, and J.G. Proakis. A Recursive Modified Gram-Schmidt Algorithm for Least-Squares Estimation. *IEEE Trans. on Acous., Speech, and Signal Processing*, ASSP-34(4):829–836, Aug. 1986.

[12] K.J.R. Liu, S.F. Hsieh, and K. Yao. Systolic block Housholder transformation for RLS algorithm with two-level pipelined implementation. *IEEE Trans. on Signal Processing*, 40(4):946–958, April 1992.

[13] K.J.R. Liu, S.F. Hsieh, K. Yao, and C.T. Chiu. Dynamic Range, Stability and Fault-Tolerant Capability of Finite Precision RLS Systolic Array Based on Givens Rotation. *IEEE Trans. on Circuits and Systems*, 38(6):625–636, June 1991.

[14] J.G. McWhirter. Recursive Least-Squares Minimization using a Systolic Array. *Proc. of SPIE, Real Time Signal Processing VI*, 431:105–112, 1983.

[15] J.G. McWhirter and T.J. Shepherd. Systolic Array Processor for MVDR Beamforming. *IEE Prcceedings*, 136(2):75–80, April 1989. Pt.F.

[16] I.K. Proulder, J.G. McWhirter, and T.J. Shepherd. The QRD-based Least Squares Lattice Algorithm: Some Computer Simulations using Finite Wordlength. In *Proc. IEEE ISCAS*, pages 258–261, New Orleans, May 1990.

[17] T.J. Shepherd, J.G. McWhirter, and J.E. Hudson. Parallel Weight Extraction from a Systolic Adaptive Beamforming. *Mathematics in Signal Processing II*, 1990.

[18] R.W. Stewart, R. Chapman, and T.S. Durrani. Arithmetic Implementation of the Givens QR Ttiarray. In *Proc. of IEEE/ICASSP*, pages V–2405–2408, 1989.

[19] C.F.T. Tang and K.J.R. Liu. A VLSI Algorithm and Architecture for CRLS Adaptive Beamforming. In *Proc. of the 25th International Conference on Information Sciences and Systems*, pages 862–867, Baltimore, March 1991.

	$S1.1 : \kappa\lambda$			$S1.2 : \mu\nu$			$S1.3$: Givens rotation		
cell	1	2	3	1	2	3	1	2	3
number of	p	$\frac{p(p+1)}{2}$	1	p	$\frac{p(p+1)}{2}$	1	p	$\frac{p(p+1)}{2}$	1
sq.rt	-	-	-	-	-	-	1	-	-
div.	-	-	1	1	-	-	1	-	-
mult.	9	4	1	5	3	1	4	4	1
i/o	9	10	4	6	8	3	5	6	3

Table 1: RLS residual computational complexity.

	$S2.1 : \kappa\lambda$				$S2.2 : \mu\nu$				$S2.3$: Givens rotation			
cell	1	2	3	4	1	2	3	4	1	2	3	4
number of	p	$\frac{(p-1)p}{2}$	p	$\frac{p(p+1)}{2}$	p	$\frac{(p-1)p}{2}$	p	$\frac{p(p+1)}{2}$	p	$\frac{(p-1)p}{2}$	p	$\frac{p(p+1)}{2}$
sq.rt	-	-	-	-	-	-	-	-	1	-	-	-
div.	-	-	-	-	1	-	-	-	1	-	-	-
mult.	8	4	4	5	5	3	3	4	4	4	4	5
i/o	7	10	11	14	6	8	9	12	3	6	7	10

Table 2: RLS weight extraction computational complexity (mode 0).

	$S3.1 : \kappa\lambda$				$S3.2 : \mu\nu$				$S3.3$: Givens rotation			
cell	1	2	3	4	1	2	3	4	1	2	3	4
number of	p	$\frac{(p-1)p}{2}$	Np	N	p	$\frac{(p-1)p}{2}$	Np	N	p	$\frac{(p-1)p}{2}$	Np	N
sq.rt	-	-	-	-	-	-	-	-	1	-	-	-
div.	1	-	-	1	1	-	-	1	1	-	-	1
mult.	9	4	6	3	6	3	5	2	5	4	5	2
i/o	10	12	14	7	7	10	12	5	5	6	8	5

Table 3: CRLS optimal residual computational complexity (mode 0).

	$S4.1 : \kappa\lambda$					$S4.2 : \mu\nu$				
cell	1	2	3	4	5	1	2	3	4	5
number of	p	$\frac{(p-1)p}{2}$	Np	$\frac{Np(p+1)}{2}$	Np	p	$\frac{(p-1)p}{2}$	Np	$\frac{Np(p+1)}{2}$	Np
sq.rt	-	-	-	-	-	-	-	-	-	-
div.	1	-	-	-	1	1	-	-	-	1
mult.	8	4	6	5	1	5	3	5	4	-
i/o	8	12	19	14	4	6	8	14	10	4

	$S4.3$: Givens rotation				
cell	1	2	3	4	5
number of	p	$\frac{(p-1)p}{2}$	Np	$\frac{Np(p+1)}{2}$	Np
sq.rt	1	-	-	-	-
div.	1	-	-	-	1
mult.	4	4	5	5	-
i/o	4	8	13	10	4

Table 4: CRLS weight vector extraction comp. complexity (mode 0).

	operations-per-cycle	DSP96000 (ns)	IMS T800 (ns)	WEITEK 3164 (ns)	ADSP-3201/2 (ns)
S1.1	max{1 div. + 1 mult. , 9 mult. }	900	3150	1800	2700
S1.2	1 div. + 5 mult.	1020	2300	2700	3675
S1.3	1 sq.rt. + 1 div. + 4 mult.	1810	4500	5300	7175
S2.1	8 mult.	800	2800	1600	2400
S2.2	1 div. + 5 mult.	1020	2300	2700	3675
S2.3	1 sq.rt. + 1 div. + 4 mult.	1810	4500	5300	7175
S3.1	1 div. + 9 mult.	1420	3700	3500	4875
S3.2	1 div. + 6 mult.	1120	2650	2900	3975
S3.3	1 sq.rt. + 1 div. + 5 mult.	1810	4500	5300	7175
S4.1	1 div. + 8 mult.	1320	3350	3300	4575
S4.2	1 div. + 5 mult.	1020	2300	2700	3675
S4.3	1 sq.rt. + 1 div. + 4 mult.	1810	4500	5300	7175

Table 5: Minimum required delay between two consequent sets of input data.

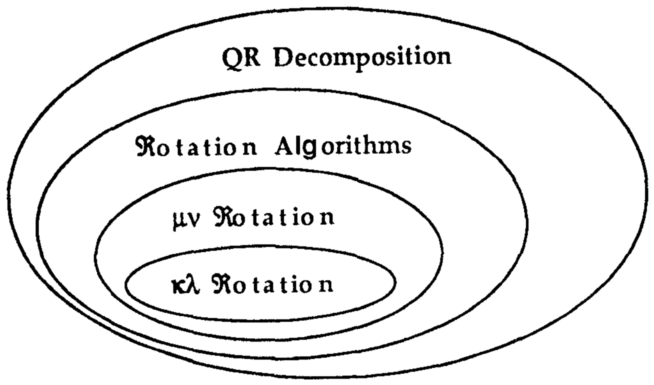


Figure 1: The relations among the classes of algorithms based on QR decomposition. a ℜotation algorithm, a $\mu\nu$ ℜotation and a $\kappa\lambda$ ℜotation.

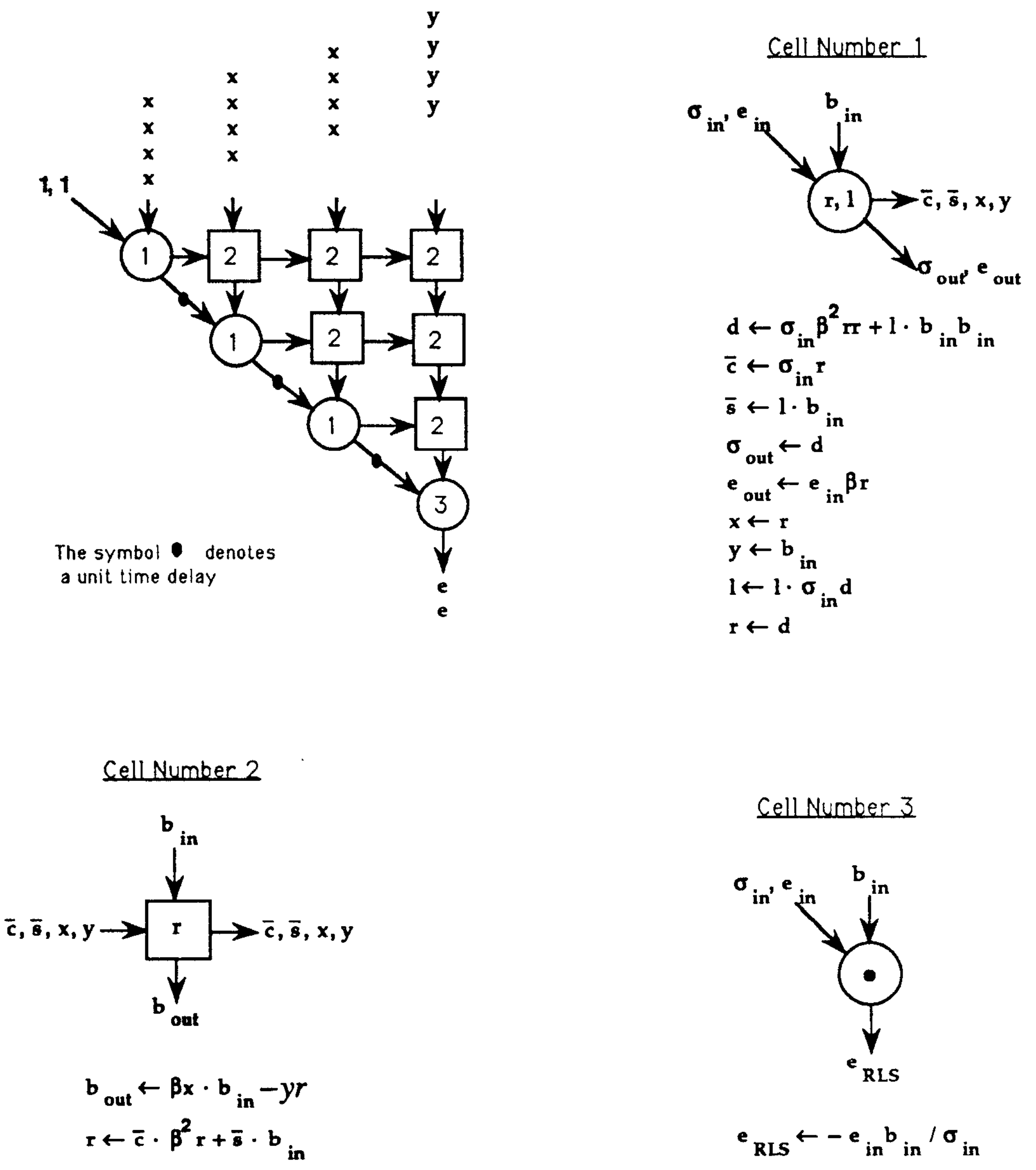
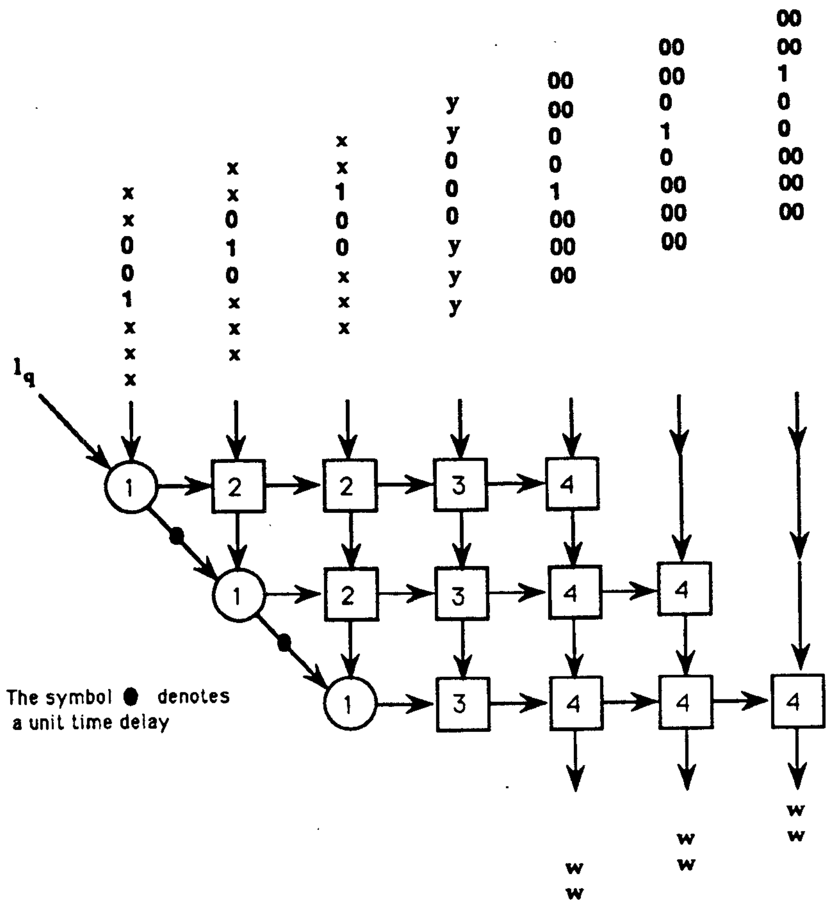
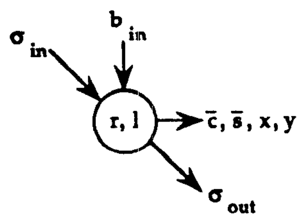

x x x x x
x x x x
x x x x
y y y y y
1, 1
1
2
3
The symbol ● denotes a unit time delay
e e
Cell Number 1
σ_{in}, e_{in}
b_{in}
r, l
$\bar{c}, \bar{s}, x, y$
σ_{out}, e_{out}
$d \leftarrow \sigma_{in} \beta^2 rr + l \cdot b_{in} b_{in}$
$\bar{c} \leftarrow \sigma_{in} r$
$\bar{s} \leftarrow l \cdot b_{in}$
$\sigma_{out} \leftarrow d$
$e_{out} \leftarrow e_{in} \beta r$
$x \leftarrow r$
$y \leftarrow b_{in}$
$l \leftarrow l \cdot \sigma_{in} d$
$r \leftarrow d$
Cell Number 2
b_{in}
$\bar{c}, \bar{s}, x, y$
r
$\bar{c}, \bar{s}, x, y$
b_{out}
$b_{out} \leftarrow \beta x \cdot b_{in} - yr$
$r \leftarrow \bar{c} \cdot \beta^2 r + \bar{s} \cdot b_{in}$
Cell Number 3
σ_{in}, e_{in}
b_{in}
e_{RLS}
$e_{RLS} \leftarrow -e_{in} b_{in} / \sigma_{in}$

Figure 2: $S1.1$: Systolic array that computes the RLS optimal residual. It implements the algorithm that is based on the $\kappa\lambda$ $\Re$otation for which $\kappa = \lambda = 1$.



Cell Number 1



mode 0 : $d \leftarrow \sigma_{in}\beta^2 rr + l \cdot b_{in} b_{in}$

$\bar{c} \leftarrow \sigma_{in} r$

$\bar{s} \leftarrow l \cdot b_{in}$

$\sigma_{out} \leftarrow d$

$x \leftarrow r$

$y \leftarrow b_{in}$

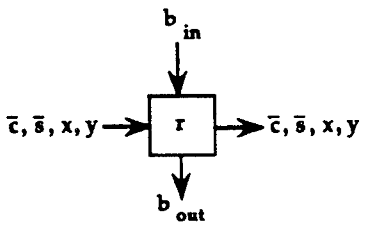
$l \leftarrow l \cdot \sigma_{in} d$

$r \leftarrow d$

mode 1: $x \leftarrow 1$

$y \leftarrow b / r$

Cell Number 2

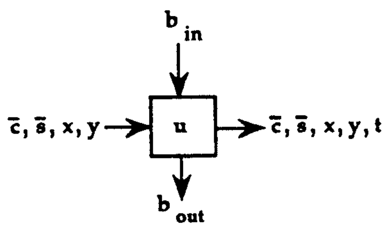


mode 0 : $b_{out} \leftarrow \beta x \cdot b_{in} - y \cdot \beta r$

$r \leftarrow \bar{c} \cdot \beta^2 r + \bar{s} \cdot b_{in}$

mode 1: $b_{out} \leftarrow x \cdot b_{in} - y \cdot r$

Cell Number 3

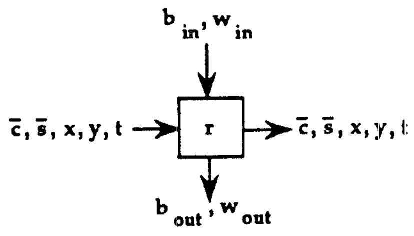


$b_{out} \leftarrow \beta x \cdot b_{in} - y \cdot \beta u$

$u \leftarrow \bar{c} \cdot \beta^2 u + \bar{s} \cdot b_{in}$

$t \leftarrow u$

Cell Number 4



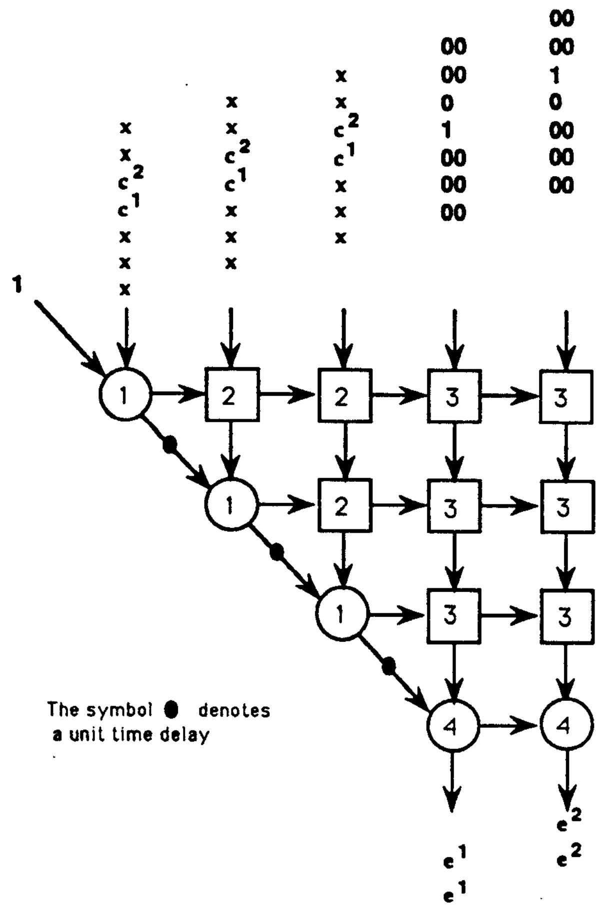
mode 0 : $b_{out} \leftarrow \frac{1}{\beta} x \cdot b_{in} - y \frac{1}{\beta} r$

$r \leftarrow \frac{1}{\beta^2} \bar{c} \cdot r + \bar{s} \cdot b_{in}$

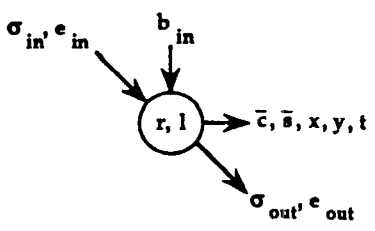
$w_{out} \leftarrow w_{in} + t \cdot r^*$

mode 1: if $b_{in} = 1$ then $r \leftarrow y$

Figure 3: $S2.1$: Systolic array that computes the RLS optimal weight vector. It implements the algorithm that is based on the $\kappa\lambda$ $\Re$otation for which $\kappa = \lambda = 1$.



Cell Number 1



mode 0 : $d \leftarrow \sigma_{in} \beta^2 rr + l \cdot b_{in} b_{in}$

$\bar{c} \leftarrow \sigma_{in} r$

$\bar{s} \leftarrow l \cdot b_{in}$

$\sigma_{out} \leftarrow d$

$e_{out} \leftarrow e_{in} \beta r$

$x \leftarrow r$

$y \leftarrow b_{in}$

$l \leftarrow l \cdot \sigma_{in} d$

$r \leftarrow d$

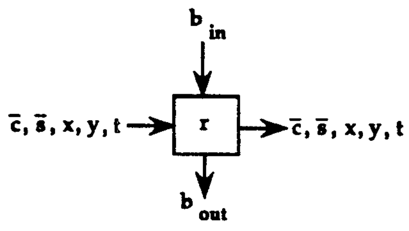
$t \leftarrow 1/l$

mode 1: $x \leftarrow 1$

$y \leftarrow b/r$

$t \leftarrow l$

Cell Number 2

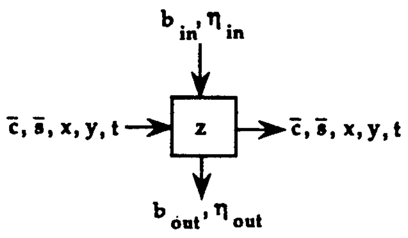


mode 0 : $b_{out} \leftarrow \beta x \cdot b_{in} - y \cdot \beta r$

$r \leftarrow \bar{c} \cdot \beta^2 r + \bar{s} \cdot b_{in}$

mode 1: $b_{out} \leftarrow x \cdot b_{in} - y \cdot r$

Cell Number 3



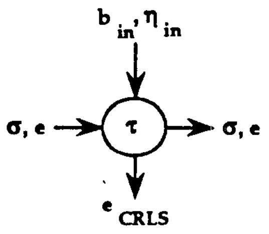
mode 0 : $b_{out} \leftarrow \frac{1}{\beta} x \cdot b_{in} - y \frac{1}{\beta} z$

$z \leftarrow \frac{1}{\beta^2} \bar{c} \cdot z + \bar{s} \cdot b_{in}$

$\eta_{out} \leftarrow \eta_{in} + t \cdot z^* z$

mode 1: if $b_{in} = 1$ then $z \leftarrow y \cdot t$

Cell Number 4

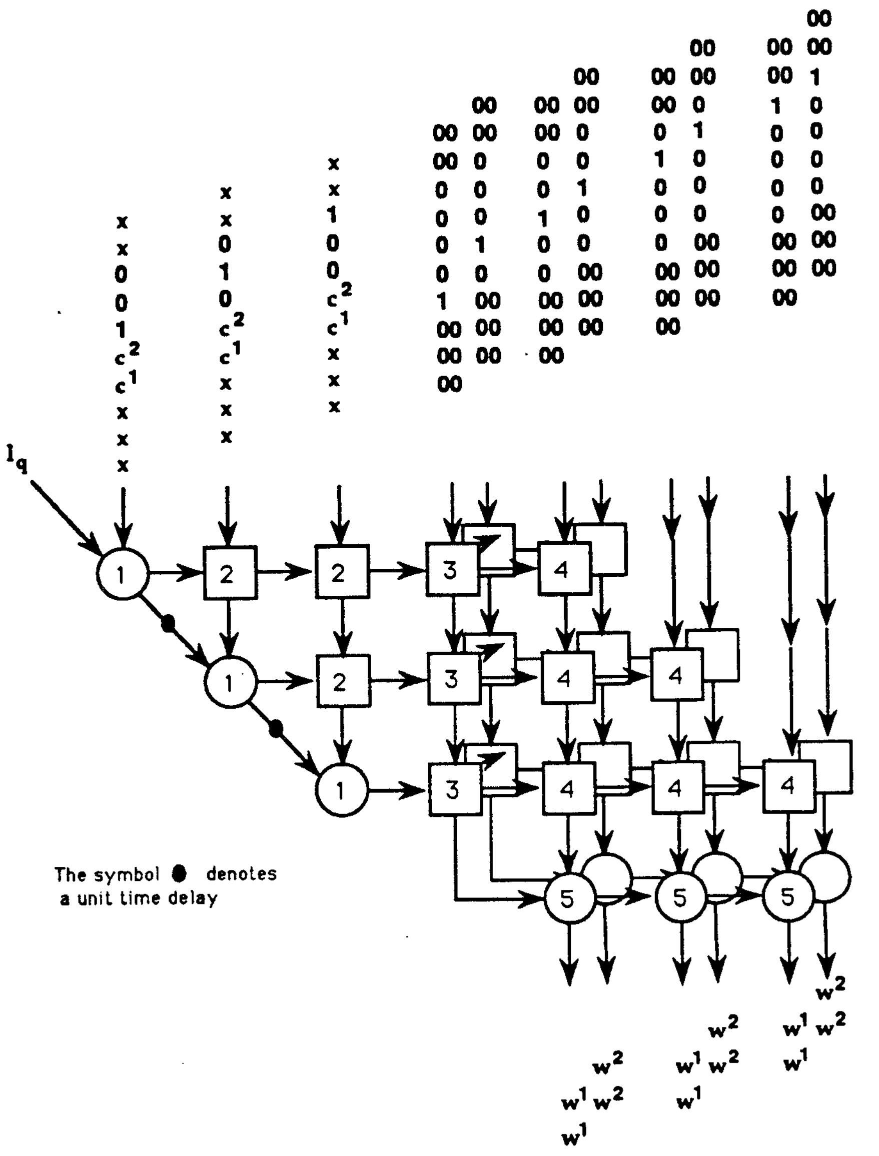


$t_1 \leftarrow \eta_{in} \sigma$

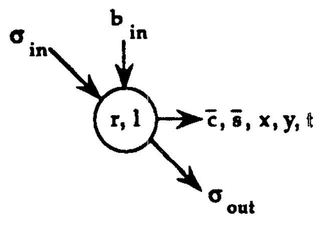
$t_2 \leftarrow \tau \cdot b_{in} e$

$e_{CRLS} \leftarrow - t_2 / t_1$

Figure 4: *S*3.1 : Systolic array that computes the CRLS optimal residual. It implements the algorithm that is based on the $\kappa\lambda$ $\Re$otations for which $\kappa = \lambda = 1$.



Cell Number 1



mode 0 : $d \leftarrow \sigma_{in}\beta^2 r r + l \cdot b_{in} b_{in}$

$\bar{c} \leftarrow \sigma_{in} r$

$\bar{s} \leftarrow l \cdot b_{in}$

$\sigma_{out} \leftarrow d$

$x \leftarrow r$

$y \leftarrow b_{in}$

$l \leftarrow l \cdot \sigma_{in} d$

$r \leftarrow d$

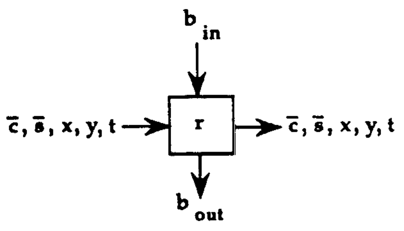
$t \leftarrow 1/l$

mode 1: $x \leftarrow 1$

$y \leftarrow b / r$

$t \leftarrow l$

Cell Number 2

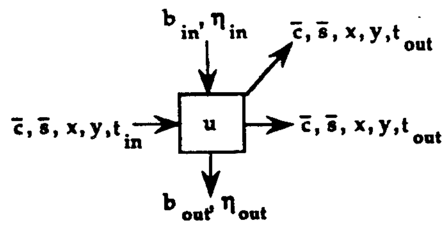


mode 0 : $b_{out} \leftarrow \beta x \cdot b_{in} - y \cdot \beta r$

$r \leftarrow \bar{c} \cdot \beta^2 r + \bar{s} \cdot b_{in}$

mode 1: $b_{out} \leftarrow x \cdot b_{in} - y \cdot r$

Cell Number 3



mode 0 : $b_{out} \leftarrow \frac{1}{\beta} x \cdot b_{in} - \frac{1}{\beta} y \cdot z$

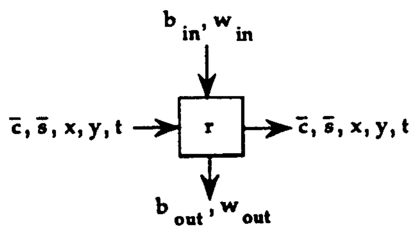
$z \leftarrow \frac{1}{\beta^2} \bar{c} \cdot z + \bar{s} \cdot b_{in}$

$\eta_{out} \leftarrow \eta_{in} + t_{in} z^* z$

$t_{out} \leftarrow z$

mode 1: if $b_{in} = 1$ then $r \leftarrow y \cdot t_{in}$

Cell Number 4



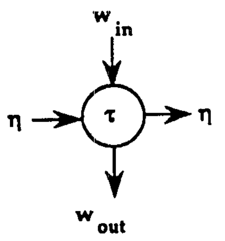
mode 0 : $b_{out} \leftarrow \frac{1}{\beta} x \cdot b_{in} - y \frac{1}{\beta} r$

$r \leftarrow \frac{1}{\beta^2} \bar{c} \cdot r + \bar{s} \cdot b_{in}$

$w_{out} \leftarrow w_{in} + t \cdot r^*$

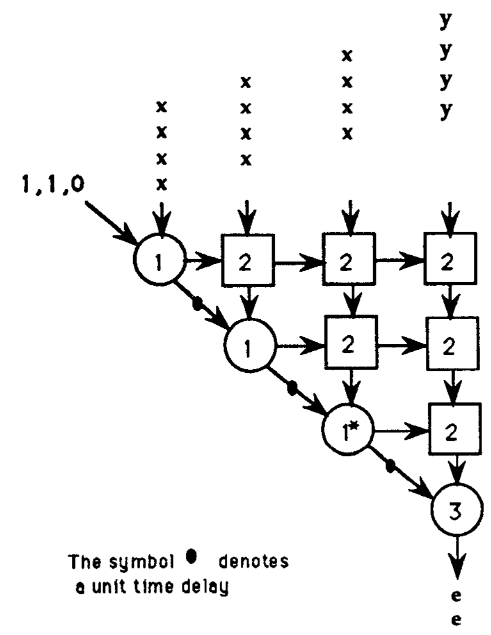
mode 1: if $b_{in} = 1$ then $r \leftarrow y$

Cell Number 5

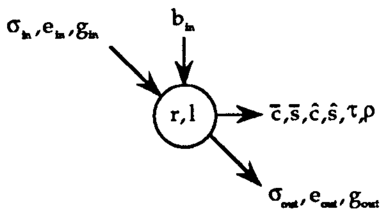


$w_{out} \leftarrow \tau \cdot w_{in} / \eta$

Figure 5: $S4.1$: Systolic array that computes the CRLS optimal weight vector. It implements the algorithm that is based on the $\kappa\lambda$ $\Re$otation for which $\kappa = \lambda = 1$.



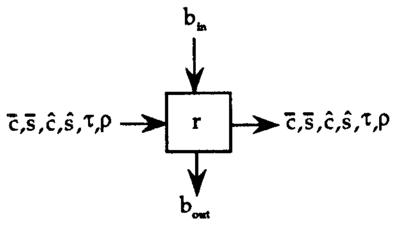
Cell Number 1



$\sigma_{in}, e_{in}, g_{in}$
b_{in}
r,l
$\bar{c},\bar{s},\hat{c},\hat{s},\tau,\rho$
$\sigma_{out}, e_{out}, g_{out}$

$d \leftarrow \sigma_{in} \cdot \beta^2 \cdot r \cdot r + l \cdot b_{in} \cdot b_{in}$
$\tau \leftarrow$ shift_amount(d)
$t \leftarrow l \cdot \sigma_{in} \cdot d$
$\rho \leftarrow$ shift_amount(t)
$\bar{c} \leftarrow \sigma_{in} \cdot r \cdot \beta^2$
$\bar{s} \leftarrow l \cdot b_{in}$
$\sigma_{out} \leftarrow d \cdot 2^{-2\tau}$; at (1*): $\sigma_{out} \leftarrow d$
$e_{out} \leftarrow e_{in} \cdot \beta \cdot r$
$g_{out} \leftarrow g_{in} + \tau$; at (1*): $g_{out} \leftarrow g_{in} - \tau$
$\hat{c} \leftarrow \beta \cdot r$
$\hat{s} \leftarrow \beta \cdot b_{in}$
$l \leftarrow t \cdot 2^{-2\rho}$
$r \leftarrow d \cdot 2^{-\rho}$

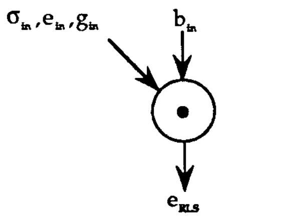
Cell Number 2



b_{in}
$\bar{c},\bar{s},\hat{c},\hat{s},\tau,\rho$
r
$\bar{c},\bar{s},\hat{c},\hat{s},\tau,\rho$
b_{out}

$b_{out} \leftarrow \hat{c} \cdot b_{in} \cdot 2^{-\tau} - \hat{s} \cdot r \cdot 2^{-\tau}$
$r \leftarrow \bar{c} \cdot r \cdot 2^{-\rho} + \bar{s}\ b_{in} \cdot 2^{-\rho}$

Cell Number 3



$e_{RLS} \leftarrow - e_{in} \cdot b_{in} \cdot 2^{-g_{in}} / \sigma_{in}$

Figure 6: Systolic array that computes the RLS optimal residual based on the scaled square root free and division free $\Re$otation.