

QR-RLS ADAPTATION OF MODULAR MULTICHANNEL LATTICE FILTERS

João Gomes Victor Barroso

Instituto Superior Técnico – Instituto de Sistemas e Robótica
 Av. Rovisco Pais, 1049-001 Lisboa, Portugal
 {jpg, vab}@isr.ist.utl.pt

ABSTRACT

This paper examines an extension of QR-RLS lattice filtering to the case of multiple input signals. The proposed adaptation algorithm is based on a compact square-root array formulation that is amenable to hardware implementation due to its scalar-only nature, parallelizability and numerical robustness. A technique that provides increased design flexibility by allowing unequal filter lengths to be specified for different input channels is also presented.

1. INTRODUCTION

Multichannel filtering is a topic of importance in applications where one seeks to combine observations arising from several sources to approximate a desired scalar signal. These include broadband beamforming, spectral analysis, and equalization of digital communication channels using fractional sampling and/or decision feedback. When implemented in lattice form, multichannel filters designed with a deterministic least-squares criterion offer many practical advantages, such as fast convergence for a broad range of input spectra, excellent robustness under limited numerical accuracy and high computational efficiency that stems from their inherently modular structure.

Starting from a modular decomposition approach developed in [1], this work addresses the problem of adapting the filter parameters using a square-root Recursive Least-Squares (RLS) algorithm, based on QR decomposition of the data matrix. These results are a rather straightforward multichannel extension of the array formulation of RLS filtering based on a fruitful link with Kalman filtering theory [2]. A similar processing structure is derived in [3] using signal flow graph manipulations.

The modular lattice structure naturally leads to filters that share the same order on all channels. Avoiding this constraint is useful, e.g., when designing decision-feedback equalizers, where filtering requirements are heterogeneous because one of the input channels actually consists of previous symbol decisions, whereas the remaining ones contain samples of received waveforms. This paper describes a technique for achieving unequal filter orders by incorporating input signals at different

points in the processing chain. Although these results are already known from [4], they are rederived here based on a simple geometric argument that fits nicely into the general framework of this work and [1].

2. BACKGROUND ON MODULAR DECOMPOSITION

This section reviews the modular decomposition approach developed in [1]. Vectors and matrices are represented by lowercase boldface and uppercase boldface letters, respectively. The notations $(\cdot)^T$, $(\cdot)^*$, $(\cdot)^H$ and $\langle \cdot, \cdot \rangle$ stand for transpose, complex conjugate, hermitian (conjugate transpose) and inner product. Modulo- L indexing is denoted by $(\cdot)_L$.

At time n , an L -channel FIR filter whose order is described by the multi-index $\mathbf{m} = [m_0 \dots m_{L-1}]$ acts on a vector of input samples $\mathbf{u}_{\mathbf{m}}(n) = [\mathbf{u}_{m_0}^{(0)T}(n) \dots \mathbf{u}_{m_{L-1}}^{(L-1)T}(n)]^T$, where $\mathbf{u}_{m_i}^{(i)}(n) = [u^{(i)}(n) \dots u^{(i)}(n - m_i + 1)]^T$ is the m_i -element data vector of channel i ($m_i \geq 1$, $0 \leq i \leq L - 1$).

A lattice filter is a cascade of several blocks which progressively increase the filter order from 0 to a desired value. If, up to a given block, the lattice filter optimizes a least-squares cost function for channel orders \mathbf{m} , then the next block will provide the optimal solution for orders $\mathbf{m} + L \triangleq [m_0 + 1 \dots m_{L-1} + 1]$. To avoid performing matrix operations associated with multichannel linear prediction when $L > 1$, a modular decomposition technique is used to update the order in L consecutive steps that yield scalar recursions [4, 1]. Let

$$\mathbf{u}_{i,\mathbf{m}}(n) = [\mathbf{u}_{m_0}^{(0)T}(n) \dots \mathbf{u}_{m_i}^{(i)T}(n) \mathbf{u}_{m_{i+1}}^{(i+1)T}(n-1) \dots \mathbf{u}_{m_{L-1}}^{(L-1)T}(n-1)]^T \quad (1)$$

be a time-updated input vector up to channel i . Sequentially increasing the filter memory by one sample in channels i , $(i-1)_L, \dots, (i-L+1)_L$, yields the extended vector $\mathbf{u}_{i,\mathbf{m}+L}(n)$ that underlies the next lattice block.

The internal structure of an L -channel block is an $L \times L$ array of scalar units, arranged into $l = 1, \dots, L$ sequential *stages* of L units each, thus forming L parallel processing *chains* [1]. Denoting by $\mathbf{u}_{i,l}(n)$ the original vector $\mathbf{u}_{i,\mathbf{m}}(n)$ augmented with past samples in channels $i, \dots, (i-l+1)_L$, the l -th unit of the i -th processing chain outputs forward and backward linear prediction errors based upon $\mathbf{u}_{i-1,l}(n)$ and $\mathbf{u}_{i,l}(n)$, respectively. These errors are recursively computed

This work was partially supported by the FCT *Programa Operacional Sociedade da Informação* (POSI) in the frame of QCA III, under contract PCTI/1999/CPS/33205.

from previous stages in an efficient manner that defines the pattern of interconnections between scalar units.

Forward *a priori* and *a posteriori* prediction errors are denoted by η and f , respectively, and the prediction coefficients by $\bar{\mathbf{a}}$. The equivalent quantities for backward prediction are β , b , and $\bar{\mathbf{c}}$. Specifically, for $1 \leq k \leq n$

$$\eta_{i,l}(k) = u^{(i)}(k) - \bar{\mathbf{a}}_{i,l}^H(n-1)\mathbf{u}_{i-1,l}(k) \quad (2)$$

$$f_{i,l}(k) = u^{(i)}(k) - \bar{\mathbf{a}}_{i,l}^H(n)\mathbf{u}_{i-1,l}(k) \quad (3)$$

$$\beta_{i,l}(k) = u^{(j)}(k - m_j - \delta) - \bar{\mathbf{c}}_{i,l}^H(n-1)\mathbf{u}_{i,l}(k) \quad (4)$$

$$b_{i,l}(k) = u^{(j)}(k - m_j - \delta) - \bar{\mathbf{c}}_{i,l}^H(n)\mathbf{u}_{i,l}(k), \quad (5)$$

where $j = (i - l)_L$ and the indicator function δ equals 1 when $l > i$ and 0 otherwise. Prediction coefficients are chosen to minimize an exponentially weighted sum of *a posteriori* errors with forgetting factor λ

$$F_{i,l}(n) = \sum_{k=1}^n \lambda^{n-k} |f_{i,l}(k)|^2, \quad B_{i,l}(n) = \sum_{k=1}^n \lambda^{n-k} |b_{i,l}(k)|^2. \quad (6)$$

Lattice filters take advantage of the fact that all samples in $\mathbf{u}_{i,l}(k)$ are contained in $\{\mathbf{u}_{i-1,l-1}(k), \mathbf{u}_{i,l-1}(k)\}$ to efficiently compute prediction errors in an order-recursive way. It may be shown that

$$f_{i,l}(k) = f_{i,l-1}(k) + \kappa_{i,l}^{f*}(n)b_{i-1,l-1}(k) \quad (7)$$

$$b_{i,l}(k) = b_{i-1,l-1}(k) + \kappa_{i,l}^{b*}(n)f_{i,l-1}(k), \quad (8)$$

where the forward reflection coefficient $\kappa_{i,l}^f(n)$ is obtained by projecting the vector of (weighted) forward errors $f_{i,l-1}(k)$ onto the vector of backward errors $b_{i-1,l-1}(k)$ [1]. Reciprocally, $\kappa_{i,l}^b(n)$ is related to the projection of $b_{i-1,l-1}(k)$ onto $f_{i,l-1}(k)$.

Finally, the filter output is calculated by cascading *ladder sections* that perform a linear combination of $b_{L-1,l}(n)$. Due to the exact decoupling property of backward errors, increasing the filter order simply amounts to adding more lattice blocks and the associated ladder coefficients, $\kappa_l(n)$, without disturbing the values computed for lower-order stages. Given a desired output, $d(n)$, *a posteriori* errors satisfy the recursion

$$e_l(k) = e_{l-1}(k) - \kappa_l^*(n)b_{L-1,l-1}(k), \quad (9)$$

allowing $\kappa_l(n)$ to be calculated by projecting $e_{l-1}(k)$ onto $b_{L-1,l-1}(k)$. Ladder *a priori* errors will be denoted by ξ .

3. UNEQUAL CHANNEL ORDERS

When several L -channel lattice blocks are cascaded, all channels are constrained to share the same equivalent filter order. Depending on the application, this may be a severe restriction that results in an unnecessarily high number of filter parameters, thus contributing to output jitter and increased computational complexity. This section describes a technique, developed in [4], which allows different orders to be specified

by cascading multichannel lattice units with increasing dimension. For ease of reference in the discussion, Figure 1a depicts the forward and backward prediction operations that are involved in the recursive update expressions (7)–(8) for $l = 1$ when all channel orders are equal.

Assume that the channel indices are initially ordered so that the corresponding desired filter orders are in descending order, $m_0 \geq m_1 \dots \geq m_{L-1}$. There will be a total of m_0 lattice blocks, labeled $0, \dots, m_0 - 1$, but some of the channels will only be effectively incorporated later in the processing chain. Channel i will enter a standard multichannel lattice unit at stage $m_0 - m_i$, so that the input signal $u^{(i)}(n)$ will contribute to a total of m_i units. In block s the lattice dimension L_s equals the number of channels that satisfy $m_i \geq m_0 - s$.

Suppose that a total of c new channels are incorporated into the lattice after block s , with indices $L_s, \dots, L_s + c - 1$. This implies that the underlying data vector for FIR prediction and filtering contains no samples for channels $L_s, \dots, L_s + c - 1$ at the input of lattice block $s + 1$. It is then straightforward to verify that, for $L_s \leq i \leq L_s + c - 1$, the residual $f_{i,0}(n)$ is just the prediction error of $u^{(i)}(n)$ given the fully-updated data vector at the output of the previous lattice block, denoted by $\mathbf{u}_{L_s-1,0}(n)$, regardless of the remaining newly-incorporated channels. Similarly, one can conclude that $b_{i,0}(n)$ for any of these channels is the prediction residual of $u^{(i)}(n)$ given $\mathbf{u}_{L_s-1,0}(n)$, and therefore coincides with $f_{i,0}(n)$.

Figures 1b–c show the samples that are (implicitly) involved in the forward/backward error computation of units $(0, 1)$ and $(i, 1)$, $i > L_s$, in the first stage of lattice block $s + 1$. They clearly illustrate that $\mathbf{u}_{L_s-1,0}(n)$ is indeed the only relevant data vector to consider when calculating residuals for the upper c channels. Noting that an entirely equivalent residual $e_{L_s}(n)$ is computed by the s -th ladder section for a reference signal $d(n)$ (through $e_0(n)$), one concludes that identical blocks should be locally replicated to process each channel that will be incorporated downstream. When $d(n)$ is replaced by $u^{(i)}(n)$, (9) still provides the order recursion that is needed to gradually propagate the prediction error between the input and output of a block. Figure 1d depicts the resulting configuration for orders $m_0 = 4$, $m_1 = 3$, $m_2 = 1$, where, similarly to [4], $L_s \times L_s$ lattice blocks and $1 \times L_s$ ladder sections are denoted by $\mathcal{L}[L_s]$ and $\mathcal{D}[L_s]$, respectively. Note that each ladder section is fed by a scalar prediction error and the L_s intermediate backward residuals $b_{L_s-1,l}(n)$, $0 \leq l \leq L_s - 1$ from its associated lattice block. The final output error, $e(n)$, is the difference between the reference symbol and the filter output obtained from the full set of desired input samples.

4. MULTICHANNEL QR-RLS ALGORITHM

Defining the *a posteriori* error vectors

$$\mathbf{f}_{i,l}(n) = [\lambda^{\frac{n-1}{2}} f_{i,l}^*(1) \dots f_{i,l}^*(n)]^T \quad (10)$$

$$\mathbf{b}_{i,l}(n) = [\lambda^{\frac{n-1}{2}} b_{i,l}^*(1) \dots b_{i,l}^*(n)]^T, \quad (11)$$

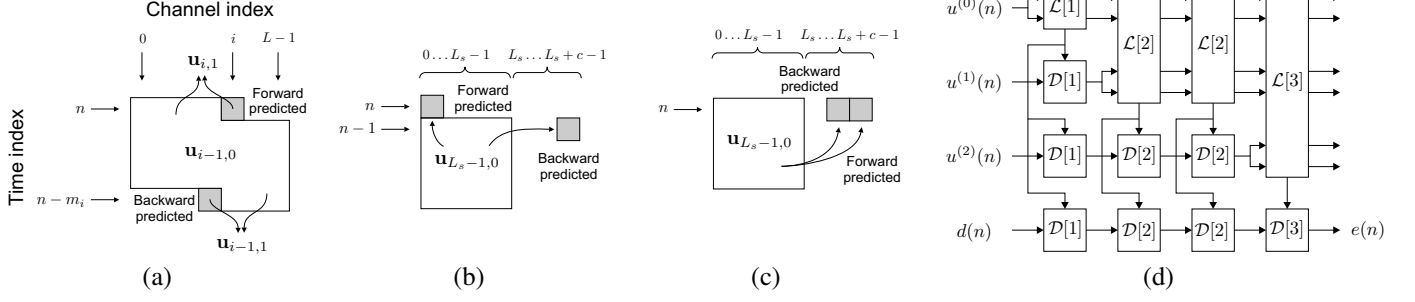


Fig. 1. Recursive prediction error updating in the first stage ($l = 1$) of a lattice block (a) Equal channel orders, unit ($i, 1$), $0 < i < L - 1$ (b) Incorporation of c new channels, unit ($0, 1$) (c) Unit ($i, 1$), $L_s < i \leq L_s + c - 1$ (d) Multichannel lattice filter structure for channel orders $m_0 = 4$, $m_1 = 3$, $m_2 = 1$

the optimal reflection coefficient in (7) minimizes the least-squares cost function

$$\kappa_{i,l}^f(n) = \arg \min_{\kappa} \|\mathbf{f}_{i,l-1}(n) + \kappa \mathbf{b}_{i-1,l-1}(n)\|^2, \quad (12)$$

and similarly for $\kappa_{i,l}^b(n)$ in (8) and $\kappa_l(n)$ in (9). If $\mathbf{f}_{i,l-1}(n)$ and $\mathbf{b}_{i-1,l-1}(n)$ were updated from time $n - 1$ to n simply by adding the current elements $f_{i,l-1}^*(n)$, $b_{i-1,l-1}^*(n)$ and multiplying all the previous ones by λ , then it would be possible to use an RLS-type algorithm to recursively calculate the reflection coefficients. This is not the case, as all forward/backward errors produced by an upstream lattice unit may change in response to the full multichannel input when reflection coefficients change. This problem can be circumvented by considering angle-normalized errors, as shown next.

In RLS filtering, *a priori* and *a posteriori* errors are related through a real conversion factor γ . In the context of single-channel lattice filters designed according to the least-squares criterion, conversion factors can be defined for the various types of error at a given stage. These conversion factors turn out to be identical for forward/backward prediction and ladder filtering, up to a unit delay [5]. As shown in [1], the same concept can be extended to the multichannel case, yielding for a given lattice block $f_{i,l}(n)/\eta_{i,l}(n) = \gamma_{i-1,l}(n)$, $b_{i,l}(n)/\beta_{i,l}(n) = \gamma_{i,l}(n)$ and $e_l(n)/\xi_l(n) = \gamma_{L-1,l}(n)$. It is then possible to define the following *angle-normalized* prediction errors

$$\epsilon_{i,l}^f(n) = \gamma_{i-1,l}^{1/2}(n) \eta_{i,l}(n) = \gamma_{i-1,l}^{-1/2}(n) f_{i,l}(n) \quad (13)$$

$$\epsilon_{i,l}^b(n) = \gamma_{i,l}^{1/2}(n) \beta_{i,l}(n) = \gamma_{i,l}^{-1/2}(n) b_{i,l}(n) \quad (14)$$

$$\epsilon_l(n) = \gamma_{L-1,l}^{1/2}(n) \xi_l(n) = \gamma_{L-1,l}^{-1/2}(n) e_l(n). \quad (15)$$

When written in terms of (13)–(15), time recursions for the input covariance, derived in [1], regain a conventional RLS form

$$F_{i,l}(n) = \lambda F_{i,l}(n-1) + |\epsilon_{i,l}^f(n)|^2 \quad (16)$$

$$B_{i,l}(n) = \lambda B_{i,l}(n-1) + |\epsilon_{i,l}^b(n)|^2, \quad (17)$$

and the same happens with other key variables, such as the crosscorrelation of forward/backward errors used in the computation of $\kappa_{i,l}^f(n)$, $\kappa_{i,l}^b(n)$. In fact, all time update expressions coincide with those that would be obtained if lattice unit

(i, l) were operating on a pair of fictitious signals $\epsilon_{i,l-1}^f(n)$, $\epsilon_{i-1,l-1}^b(n)$. For forward prediction

$$\kappa_{i,l}^f(n) = \arg \min_{\kappa} \|\epsilon_{i,l-1}^f(n) + \kappa \epsilon_{i-1,l-1}^b(n)\|^2, \quad (18)$$

and $\epsilon_{i,l-1}^f(n)$, $\epsilon_{i-1,l-1}^b(n)$ are defined similarly to (10)–(11) but exhibit time shift property required by RLS.

As shown in [2], the update expressions for RLS filtering are formally identical to the equations that describe a Kalman filter operating on a specific type of dynamic system. Through this direct correspondence, the vast literature on Kalman filtering can be brought to bear on the RLS problem. Specifically, the time update equations for $\kappa_{i,l}^f(n)$ derived from (18) coincide with those describing a Kalman filter for the following simple state-space model

$$\begin{aligned} x_1(n+1) &= \lambda^{-1/2} x_1(n) \\ y_1(n) &= \epsilon_{i-1,l-1}^{b*}(n) x_1(n) + \nu_1(n), \end{aligned} \quad (19)$$

where $x_1(n)$ is the state variable, $y_1(n) = \lambda^{-n/2} \epsilon_{i,l-1}^{f*}(n)$ is the observation and $\nu_1(n)$ is white noise with zero mean and unit variance. Table 1 lists the Kalman variables for the three state-space models that pertain to a given lattice unit. All lines of the table, except for the last two, follow directly from the equivalence between Kalman and RLS filtering. The expression for the *a priori* Kalman output error for forward prediction, $\alpha(n)$, can be verified as in [5], by using (13) and the top three lines of Table 1 to obtain

$$\alpha(n) = \lambda^{-n/2} \gamma_{i-1,l-1}^{1/2}(n) (\eta_{i,l-1}^*(n) + \kappa_{i,l}^f(n-1) \beta_{i-1,l-1}^*(n)).$$

The term inside parenthesis is computed by the (i, l) unit before the reflection coefficient is updated, and therefore equals the *a priori* error $\eta_{i,l}^*(n)$. The *a posteriori* Kalman output error, $e(n)$, is derived by first noting that the *a posteriori* state estimate is $-\lambda^{-n/2} \kappa_{i,l}^f(n)$, and then proceeding similarly to $\alpha(n)$ to obtain

$$\begin{aligned} e(n) &= \lambda^{-n/2} \gamma_{i-1,l-1}^{-1/2}(n) (f_{i,l-1}^*(n) + \kappa_{i,l}^f(n) b_{i-1,l-1}^*(n)) \\ &= \lambda^{-n/2} \gamma_{i-1,l-1}^{-1/2}(n) f_{i,l}^*(n). \end{aligned}$$

The Kalman conversion factor is defined as $\alpha(n)/e(n)$.

Kalman Variable	Forward Prediction	Backward Prediction	Ladder Filtering
Observation	$\lambda^{-n/2} \epsilon_{i,l-1}^{f*}(n)$	$\lambda^{-n/2} \epsilon_{i-1,l-1}^{b*}(n)$	$\lambda^{-n/2} \epsilon_{l-1}^*(n)$
Measurement vector	$\epsilon_{i-1,l-1}^{b*}(n)$	$\epsilon_{i,l-1}^{f*}(n)$	$\epsilon_{L-1,l-1}^{b*}(n)$
<i>A priori</i> state estimate	$-\lambda^{-n/2} \kappa_{i,l}^f(n-1)$	$-\lambda^{-n/2} \kappa_{i-1,l}^b(n-1)$	$\lambda^{-n/2} \kappa_l(n-1)$
State error covariance matrix	$\lambda^{-1} B_{i-1,l-1}^{-1}(n)$	$\lambda^{-1} F_{i,l-1}^{-1}(n)$	$\lambda^{-1} B_{L-1,l-1}^{-1}(n)$
<i>A priori</i> output error	$\lambda^{-n/2} \gamma_{i-1,l-1}^{1/2}(n) \eta_{i,l}^*(n)$	$\lambda^{-n/2} \gamma_{i-1,l-1}^{1/2}(n) \beta_{i,l}^*(n)$	$\lambda^{-n/2} \gamma_{L-1,l-1}^{1/2}(n) \xi_l^*(n)$
Conversion factor	$\frac{\gamma_{i-1,l-1}(n)}{\gamma_{i-1,l}(n)}$	$\frac{\gamma_{i-1,l-1}(n)}{\gamma_{i,l}(n)}$	$\frac{\gamma_{L-1,l-1}(n)}{\gamma_{L-1,l}(n)}$

Table 1. Kalman variables for normalized forward/backward prediction and ladder filtering

Among the variants of Kalman filtering algorithms, square-root formulations based on QR decomposition of the error covariance matrix or its inverse are of special interest due to their favorable numerical properties. These algorithms can be compactly expressed in array form, where the product of a prearray by a 2×2 unitary matrix Θ (Givens rotation) creates a postarray where some of the entries are nulled [2]. Specializing the so-called square-root information filter to the dynamical system (19) and using Table 1 yields

$$\begin{bmatrix} \lambda^{1/2} B_{i-1,l-1}^{1/2}(n-1) & \epsilon_{i-1,l-1}^b(n) \\ \lambda^{1/2} p_{i,l}^{f*}(n-1) & \epsilon_{i,l-1}^f(n) \\ 0 & \gamma_{i-1,l-1}^{1/2}(n) \end{bmatrix} \Theta_{i,l}^b = \begin{bmatrix} B_{i-1,l-1}^{1/2}(n) & 0 \\ p_{i,l}^{f*}(n) & \epsilon_{i,l}^f(n) \\ b_{i-1,l-1}^*(n) B_{i-1,l-1}^{-1/2}(n) & \gamma_{i-1,l}^{1/2}(n) \end{bmatrix}. \quad (20)$$

The array propagates $p_{i,l}^{f*}(n)$, from which the reflection coefficient can be obtained as $\kappa_{i,l}^f(n) = -p_{i,l}^f(n) B_{i-1,l-1}^{-1/2}(n)$. Note that the (3, 1) entry of the postarray need not be calculated, as it is not used in any other prearray. The arrays for backward prediction and ladder filtering are given by

$$\begin{bmatrix} \lambda^{1/2} F_{i,l-1}^{1/2}(n-1) & \epsilon_{i,l-1}^f(n) \\ \lambda^{1/2} p_{i,l}^{b*}(n-1) & \epsilon_{i-1,l-1}^b(n) \\ 0 & \gamma_{i-1,l-1}^{1/2}(n) \end{bmatrix} \Theta_{i,l}^f = \begin{bmatrix} F_{i,l-1}^{1/2}(n) & 0 \\ p_{i,l}^{b*}(n) & \epsilon_{i,l}^b(n) \\ f_{i,l-1}^*(n) F_{i,l-1}^{-1/2}(n) & \gamma_{i,l}^{1/2}(n) \end{bmatrix} \quad (21)$$

$$\begin{bmatrix} \lambda^{1/2} B_{L-1,l-1}^{1/2}(n-1) & \epsilon_{L-1,l-1}^b(n) \\ \lambda^{1/2} p_l^*(n-1) & \epsilon_{l-1}(n) \\ 0 & \gamma_{L-1,l-1}^{1/2}(n) \end{bmatrix} \Theta_l = \begin{bmatrix} B_{L-1,l-1}^{1/2}(n) & 0 \\ p_l^*(n) & \epsilon_l(n) \\ b_{L-1,l-1}^*(n) B_{L-1,l-1}^{-1/2}(n) & \gamma_{L-1,l}^{1/2}(n) \end{bmatrix}, \quad (22)$$

where $\kappa_{i,l}^b(n) = -p_{i,l}^b(n) F_{i,l-1}^{-1/2}(n)$ and the ladder coefficient is given by $\kappa_l(n) = p_l(n) B_{L-1,l-1}^{-1/2}(n)$. At each lattice stage, (20)–(22) can be evaluated in parallel for the L chains $i =$

$0, \dots, L-1$. At time n , the computations are sequentially performed for stages $l = 1, \dots, L$ of a lattice block, proceeding downstream until all blocks in the filter have been updated. For any unit (i, l) the time recursion is initialized with $p_{i,l}^f(0) = p_{i,l}^b(0) = p_l(0) = 0$ and $F_{i,l-1}(0) = B_{i-1,l-1}(0) = \mu$, where μ is a small positive constant. Moreover, in the first stage of the first block $\epsilon_{i,0}^f(n) = u^{(i)}(n)$, $\epsilon_{i-1,0}^b(n) = u^{(i-1)L}(n - \delta(i))$, $\epsilon_0(n) = d(n)$ and $\gamma_{i-1,0}(n) = 1$.

5. CONCLUSION

This paper presented an extension of QR-RLS adaptation for multichannel lattice filtering. Results available in the literature for the single-channel case were readily generalized to this new context by using the modular decomposition approach developed in [1], where multichannel lattice blocks are expressed as an interconnection of scalar units. Specifying unequal filter lengths basically requires trading off some of the internal units in lattice blocks for ladder sections, identical to the ones that generate the filter output. This elegant result preserves the modularity and regularity of the lattice filter.

Naturally, future work should assess the actual stability of the algorithm when implemented under relatively low fixed-point numerical precision, which is common in special-purpose signal processing hardware.

6. REFERENCES

- [1] J. Gomes, V. Barroso, "Multichannel lattice filtering based on parallel scalar modules," in *ConfTele'99*, (Sesimbra, Portugal), pp. 151–155, April 1999.
- [2] A. Sayed, T. Kailath, "A state-space approach to adaptive RLS filtering," *IEEE Sig. Proc. Magazine*, vol. 11, pp. 18–60, July 1994.
- [3] M. Harteneck *et al.*, "Algorithmically engineered fast multichannel adaptive filter based on QR-RLS," *IEE Proc. Vision, Image and Sig. Proc.*, vol. 146, pp. 7–13, Feb. 1999.
- [4] G. Glentis and N. Kalouptsidis, "A highly modular adaptive lattice algorithm for multichannel least squares filtering," *Sig. Proc.*, vol. 46, no. 1, pp. 47–55, 1995.
- [5] S. Haykin, *Adaptive Filter Theory*. Englewood Cliffs, NJ: Prentice-Hall, third ed., 1996.