Variable-Block-Size Lapped Transforms

Ricardo L. de Queiroz and K. R. Rao

***Abstract*—A structure for implementing lapped transforms with time-varying block sizes that allows full orthogonality of the transient transforms is presented. The formulation is based on a factorization of the transfer matrix into orthogonal factors. Such an approach can be viewed as a sequence of stages with variable-block-size transforms separated by sample-shuffling (delay) stages. Details and design examples for a first-order system are presented.**

I. Introduction

The choice of a block size in transform coding is a tradeoff between time and frequency resolutions. Although not always explicitly pointed out, a variable block size is a local adaptation aiming to obtain a better projection of the signal on the time-frequency plane. The block size dictates the tradeoff. Larger blocks mean coarser time resolution and narrower subbands, whereas small blocks mean better time localization and worse frequency resolution. Block transforms of variable sizes can be easily applied to images as the problem is simplified to a tiling of the image into rectangular regions. We address the issues of perfect reconstruction (PR) and orthogonality using a lapped transform (LT) [1], i.e., a paraunitary uniform FIR filter bank (PUFB) [2] instead of a block transform. A number of recent papers deal with time-varying filter banks [3]–[6]. However, few address a general framework to change the number of channels with orthogonality in the transitions, and often, the mechanism for change is applied to the filter coefficients. Here, as in [3], the change occurs in the structural factors that compose the filter bank. Hence, the transforms and the transitions are inherently orthogonal. We do not use boundary filters nor do we segment the signal into independent sections.

II. Lapped Transforms

M-channel LT's can be described through their respective polyphase transfer matrices (PTM) [2] (the transfer matrix relating the M polyphase components of the input signal to the M subbands). The block size is M, and the length of the filters is assumed to be $L = NM$. The entries of the PTM have terms of order $N-1$, i.e., polynomials up to z^{-N+1}. The PTM $\mathbf{E}(z)$ is assumed paraunitary such that $\mathbf{E}^{-1}(z) = \mathbf{E}^T(z^{-1})$. In other words, it is unitary in the unit circle ($z = e^{j\omega}$), and the analysis system leading the polyphase components to the subbands is lossless. For $N = 1$ (and real coefficients), it is clear that the PTM is a regular orthogonal matrix. The PTM can be parameterized using a cascade of delays and orthogonal stages as

$$\mathbf{E}(z) = \mathbf{B}_0 \prod_{i=1}^{N_s-1} (\mathbf{\Lambda}(z)\mathbf{B}_i) \tag{1}$$

where all stages $\mathbf{B}_i$ are allowed to be arbitrary $M \times M$ orthogonal matrices, N_s is the number of stages, and $\mathbf{\Lambda}(z)$ is a paraunitary matrix containing only delays. We assume that $\mathbf{\Lambda}(z)$ is a full-rank matrix with only one nonzero element per row or column, where each nonzero element can be 1 or z^{-1}. The general factorization of all PUFB's is the one where N_s is the McMillan degree of $\mathbf{E}(z)$ (i.e., the order of the determinant of $\mathbf{E}(z)$) and where $\mathbf{\Lambda}(z) = \text{diag}\{z^{-1}, 1, 1, \ldots, 1\}$ [2]. The symmetric delay factorization (SDF) [7] is the case where M is even, $N_s = N$, and

$$\mathbf{\Lambda}(z) = \begin{bmatrix} z^{-1}\mathbf{I}_{M/2} & 0 \\ 0 & \mathbf{I}_{M/2} \end{bmatrix} \tag{2}$$

where $\mathbf{I}_n$ is the $n \times n$ identity matrix.

III. Time-Varying Lapped Transforms

For time-varying filters, however, the notation may not be that simple. For an invariant filter, if $y_i(m) = \sum_{n=0}^{q-1} h_{ij}(n)\, x_j(m-n)$, then the q-tap filter $h_{ij}(n)$ has z transform $H_{ij}(z) = \sum_{n=0}^{q-1} h_{ij}(n)z^{-n}$, and $Y_i(z) = H_{ij}(z)X_j(z)$. Let the filter have time-varying coefficients so that its input-output relation is

$$y_i(m) = \sum_{n=0}^{q-1} h_{ij}(m,n)x_j(m-n). \tag{3}$$

We cannot say that $Y_i(z) = H_{ij}(z,m)X_j(z)$ by using $h_{ij}(m,n)$ to replace the coefficients of $H_{ij}(z,m)$. However, because of the usual assumption of invariant filters, $H_{ij}(z)$ is often viewed as a description of an implementation algorithm, and we simplify the notation by using $H_{ij}(z,m)$ to represent the filter coefficients at instant m. Therefore, z^{-1} may be viewed in a strictly systemic approach, meaning a delay of the input, i.e., retrieve output from buffer and place input on buffer.

The key to obtaining time-varying orthogonal filter banks [3] is to change the matrices $\mathbf{B}_i$ along the time index while keeping them as orthogonal matrices at all times. Thus, for PR orthogonal time-varying LT's (PUFB's), we can rewrite (1) as

$$\mathbf{E}(z,m) = \mathbf{B}_0(m) \prod_{i=1}^{N_s-1} (\mathbf{\Lambda}(z)\mathbf{B}_i(m)) \tag{4}$$

where $\mathbf{B}_i(m)$ changes with time but remains an $M \times M$ orthogonal matrix. $\mathbf{E}(z,m)$ remains paraunitary for all m since

$$\mathbf{E}^T(z^{-1},m)\mathbf{E}(z,m) = \mathbf{\Lambda}^T(z^{-1})\mathbf{\Lambda}(z) = \mathbf{I}_M \tag{5}$$

i.e., $\mathbf{E}(z,m)$ is paraunitary because $\mathbf{B}_i(m)$ and $\mathbf{\Lambda}(z)$ (as we have defined it) are also paraunitary. From the above discussion on the z transform of a time-varying filter, we apply the term *instantaneously paraunitary* [3], [7] to describe the property of $\mathbf{E}(z,m)$. Furthermore, the linear transform (not its state space representation) that leads all input samples into subbands is an orthogonal transform [1], [3], [7].

IV. Changing the Block Size

With time-varying filter banks as described, the overall transform is orthogonal. From a transform point of view, we can think of this as a linear (orthogonal) transform $\mathbf{T}$ leading all input samples in vector $\mathbf{x}$ to the subband samples in vector $\mathbf{y}$. Thus

$$\mathbf{y} = \mathbf{T}\mathbf{x} \quad \text{and} \quad \mathbf{x} = \mathbf{T}^T\mathbf{y} \tag{6}$$

where

$$\mathbf{T} = \tilde{\mathbf{B}}_0 \prod_{i=1}^{N-1} \mathbf{W}_i\tilde{\mathbf{B}}_i \tag{7}$$

$$\tilde{\mathbf{B}}_i = \text{diag}\{\cdots, \mathbf{B}_i(m-1), \mathbf{B}_i(m), \mathbf{B}_i(m+1), \ldots\} \tag{8}$$

Manuscript received July 12, 1995; revised July 2, 1996. The associate editor coordinating the review of this paper and approving it for publication was Prof. Truong Q. Nguyen.

R. L. de Queiroz was with the University of Texas at Arlington, Arlington, TX 76019 USA. He is now with Xerox Corporation, Webster, NY 14580 USA (e-mail: queiroz@wrc.xerox.com).

K. R. Rao is with the University of Texas at Arlington, Arlington, TX 76019 USA (e-mail: rao@ee.uta.edu).

Publisher Item Identifier S 1053-587X(96)09056-3.

and $\mathbf{W}_i$ is a permutation matrix representing all the delays in a stage. As $\mathbf{B}_i(m)$ is orthogonal, so is $\tilde{\mathbf{B}}_i$, and $\mathbf{T}$ is orthogonal if and only if $\mathbf{W}_i$ is also orthogonal. As long as $\mathbf{W}_i$ implies simple permutations, it is clearly an orthogonal matrix. Therefore, the dimensions of $\mathbf{B}_i(m)$ have no effect on the orthogonality of $\mathbf{T}$ as long as $\mathbf{B}_i(m)$ and $\mathbf{W}_i$ remain orthogonal. Clearly, we can change the number of channels of the filter bank by changing the sizes of $\mathbf{B}_i(m)$. Therefore, a PUFB with a variable number of channels can be implemented through a sequence of stages with variable-block-size orthogonal transforms. These stages are intermediated by shuffling (delay) stages. It is, in other words, a variable-block-size lapped transform. In a state-space approach, we can rewrite (4) as

$$\mathbf{E}(z,m) = \mathbf{B}_0(m) \prod_{i=1}^{N_s-1} (\mathbf{\Lambda}(z,i,m)\mathbf{B}_i(m)) \tag{9}$$

where $\mathbf{\Lambda}(z,i,m)$ is the delay block at stage i and instant m. As in the case of (1), we also consider it a full-rank matrix with only one nonzero element per row or column. An entry z^{-1} in $\mathbf{\Lambda}(z,i,m)$ at the kth row and lth column means that the kth element of $\mathbf{B}_{i-1}(m)$'s input vector is the lth element of $\mathbf{B}_i(m-1)$'s output vector. Similarly, the same occurs to a "1" entry in relation to $\mathbf{B}_i(m)$'s output vector. For the class of delay matrices described, in order to obtain orthogonality, it is necessary and sufficient to make $\mathbf{W}_i$ a permutation matrix. It is necessary because otherwise, $\mathbf{W}_i$ would be rank deficient, and it is sufficient because if $\mathbf{W}_i$ is orthogonal, then $\mathbf{T}$ is orthogonal, as we saw. Now, let us examine the translation of $\mathbf{W}_i$ (as a permutation matrix) to the system described in (9). It is clear that the set of columns of $\mathbf{\Lambda}(z,i,m)$ with the entry z^{-1} has to match the set of all columns of $\mathbf{\Lambda}(z,i,m-1)$ that are all 0's as well as those that contain a z^{-1} entry. Let

$$\mathbf{E}(z,m) = \mathbf{E}_0(z,m)\mathbf{\Lambda}(z,i,m)\mathbf{E}_1(z,m) \tag{10}$$

where $\mathbf{E}_0(z,m)$ is a $k \times k$ paraunitary matrix while $\mathbf{E}_1(z,m)$ is an $l \times l$ one. Obviously, $\mathbf{\Lambda}(z,i,m)$ is a $k \times l$ matrix. If $k > l$,

$$\mathbf{\Lambda}^T(z^{-1},i,m)\mathbf{\Lambda}(z,i,m) = \mathbf{I}_k \tag{11}$$

$$\mathbf{E}^T(z^{-1},m)\mathbf{E}(z,m) = \mathbf{I}_k. \tag{12}$$

If $k < l$, on the other hand, these relations do not hold. The stationary relations do not hold because the sizes of the matrices involved also change. Assume a transition from M_1 to M_2 channels (block size changes from M_1 to M_2 samples). With a quick transition in mind, we assume that $\mathbf{E}_0(z,m-1)$ and $\mathbf{E}_1(z,m-1)$ are paraunitary matrices of size $M_1 \times M_1$, whereas $\mathbf{E}_0(z,m+1)$ and $\mathbf{E}_1(z,m+1)$ are paraunitary matrices of size $M_2 \times M_2$. Let d'_m be the number of delays in $\mathbf{\Lambda}(z,i,m)$ and d_m be the respective number of "1" entries. Thus

$$\begin{aligned} d_{m-1} + d'_{m-1} &= M_1 \\ d_m + d'_m &= k \\ d_{m+1} + d'_{m+1} &= M_2 \\ d_m + d'_{m+1} &= l. \end{aligned} \tag{13}$$

With the assumption that

$$d_{m+1} - d_{m-1} = d'_{m+1} - d'_{m-1} \tag{14}$$

we can show that

$$l - k = (M_2 - M_1)/2. \tag{15}$$

Hence, we see that $k \neq l$ is a necessary restriction, as long as $M_1 \neq M_2$. Furthermore, we cannot make l and k equal to M_1 or M_2. In other words, when changing the number of channels from M_1 to M_2, one cannot implement the transition using only block transforms of size M_1 or M_2. The process has to step over intermediary sizes for $\mathbf{B}_i(m)$. The same conclusion can be reached using the general factorization, i.e., using $d'_{m-1} = d'_{m+1}$ instead of (14).

V. Example: First-Order System

For a first-order algorithm using SDF, we have

$$\mathbf{E}(z) = \mathbf{B}_0(m)\mathbf{\Lambda}(z)\mathbf{B}_1(m) \tag{16}$$

where $\mathbf{\Lambda}(z)$ is as in (2). The filters' length is twice the block size. Assume that in a stationary state, the proposed filter bank with M channels is factorized as

$$\mathbf{E}(z) = \mathbf{C}_M\mathbf{\Lambda}(z)\mathbf{D}_M. \tag{17}$$

Then, we change the block size (number of channels) from $M_1 = 2u$ to $M_2 = 2v$. As we discussed, the transition must contain blocks with size different than M_1 or M_2. We present two distinct switching methods in Fig. 1. In this figure, the number of samples carried in each branch is indicated. Fig. 1(a) shows the original configuration at instant m for the M_1-channel case. Fig. 1(b) shows the desired configuration (the M_2-channel case). In the hard switch method shown in Fig. 1(c), the lapped transform is quickly changed to an M_2-channel output. The price paid for that is that we must have two transient filter banks: one with $M_1 = 2u$ channels and the other with $M_2 = 2v$ channels. This is because we used a block transform of size $(u+v) \times (u+v)$ as $\mathbf{B}_1(m+1)$. It does affect the filter's length but does not affect the number of channels. The filters have length $3u + v$ (instant $m + 1$) and $3v + u$ (instant $m + 2$). The second alternative is shown in Fig. 1(d). It is a soft switch stepping through an intermediate filter bank of $u + v$ channels with filters of $2(u + v)$ taps. This would be the only transitory filter bank. In Fig. 1(c), the matrices left to be optimized as free parameters are $\mathbf{B}_1(m+1)$, $\mathbf{B}_0(m+1)$, and $\mathbf{B}_0(m+2)$. In Fig. 1(d), the matrix to be optimized is $\mathbf{B}_0(m+1)$. To summarize, the difference between the soft and hard switch methods resides on the choice of which factor ($\mathbf{B}_0(m+1)$ or $\mathbf{B}_1(m+1)$) will be a matrix of different size (in this case, of size $(u+v) \times (u+v)$).

The framework in Fig. 1 is general for lapped transforms with $L = 2N$. In a design example, we use (as stationary filter bank) the modulated lapped transform (MLT) [1]. The MLT is a special cosine-modulated filter bank possessing filters with good stopband attenuation and that allows fast implementation [1]. An algorithm for a variable-block-size MLT was presented in [4]. The approach in [4] requires explicit changes in the basis functions by changing the modulating window size. Our approach is based on the design of the orthogonal factors composing a lapped transform [3], [7]. Hence, we work directly with a fast implementation algorithm, which is maintained. The MLT is simply an example, and the generality of the framework allows us to replace it with any other LT.

The design examples in Fig. 2 show the transition filters for a switch from a four-channel MLT to an eight-channel MLT. In the hard switch method, there are two transition filter banks: one with four channels and 10 taps and the other with eight channels and 14 taps. If the soft switch is applied, there is only one transition filter bank with six channels and 12 taps. In both cases, the transition filters were optimized for maximum coding gain of the instantaneous filter bank [3], [7] for an AR(1) spectrum with correlation coefficient 0.95.[1] Note that the transitory LT's are not required to be cosine-modulated filter banks. However, the lower frequency basis functions of these transforms resemble the bases of the MLT, whereas the higher frequency bases (which have less significance to the coding gain) do not.

[1]The optimization was carried by factorizing each orthogonal matrix as a product of plane rotations and by applying a nonlinear unconstrained optimization to the degrees of freedom (rotation angles). The optimization algorithm selected uses simplex search and is provided by MATLAB 4.2c.

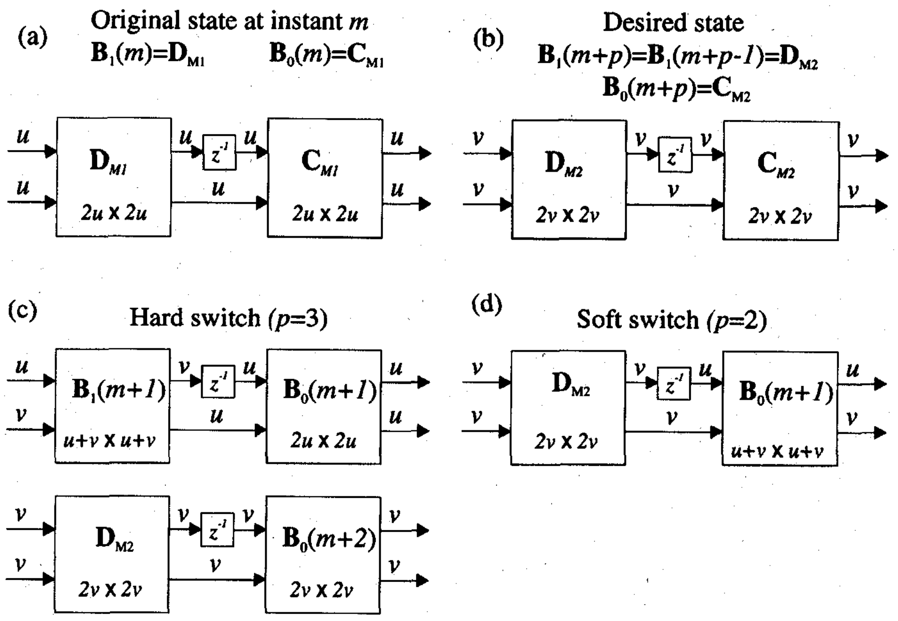

Fig. 1. Flowgraph for variable-block-size lapped transform switching from $M_1 = 2u$ channels to $M_2 = 2v$ channels: (a) Original state at instant m. (b) Desired state after p transform instants. (c) Hard switch method, switching abruptly from M_1 to M_2 channels. This will imply two transitory states. (d) Soft switch method, stepping through an intermediate filter bank of $u + v$ channels.

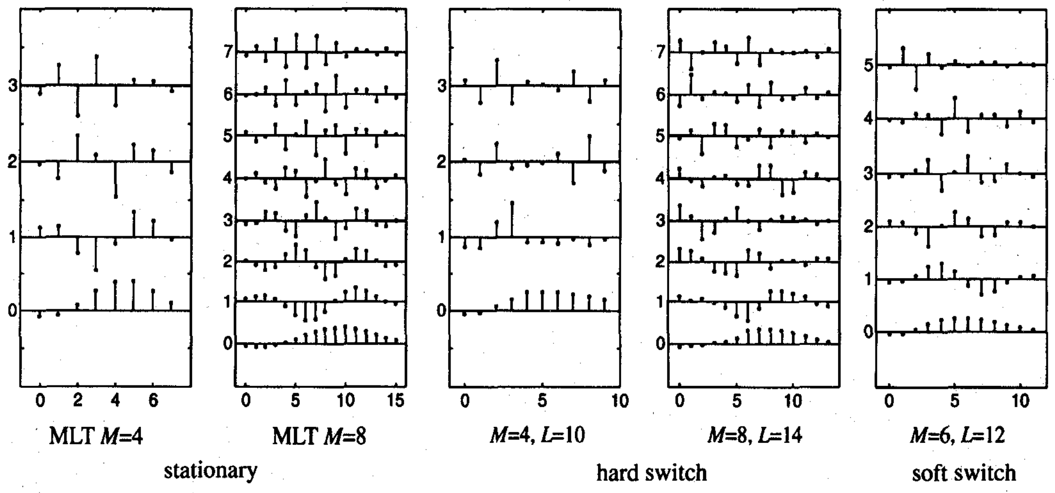

Fig. 2. Impulse response of the filters of the time-varying MLT, switching from four to eight channels. The normal MLT's for $M = 4$ ($L = 8$) and $M = 8$ ($L = 16$), the two transition filter banks for the hard switch, and the six-channel filter bank in the case of a soft switch are shown. The transition filters were optimized for maximum coding gain. L is the length (in taps) of the filters.

VI. Remarks

Filter banks can be associated in such a way as to construct discrete wavelet and wavelet packets transforms. The association of these filter banks follows the paths of a tree. By dynamically pruning or expanding branches of the tree, one can continuously reshape the tree paths. Such an approach is a time-varying wavelet packet transform. In this fashion, one can obtain a better tiling of the time-frequency plane. Often, the process is applied to a fixed binary tree. Each node is either a leaf, or it is split into two branches generating two child nodes. The extension to M-ary trees is trivial, but the problem is still reduced to an on–off decision (prune or expand branches). Pruning a branch can be viewed as applying a one-channel filter bank. Using LT's with time-varying block size (number of channels), we can achieve a much-enhanced tiling of the time-frequency plane. This is done by not only deciding on pruning a branch or not but also by deciding the number of channels to assign to each node in the tree (including the one-channel case). As the decision is no longer a binary problem, an enhanced tiling can be achieved.

In a one-node version of the time-varying wavelet-packet, we can use a single variable size transform for image compression [8], [9]. Variable-block-size coding and quadtrees have been successfully used to encode images [8]–[11]. The approach described here can be readily applied to replace the DCT in [8] and [9]. In addition, it can be combined with the approaches described in [10] and [11] by encoding the variable-block-size partitions of the image in the LT domain.

References

[1] H. S. Malvar, *Signal Processing with Lapped Transforms*. Norwood, MA: Artech House, 1992.

[2] P. P. Vaidyanathan, *Multirate Systems and Filter Banks*. Englewood Cliffs, NJ: Prentice-Hall, 1993.

[3] R. L. de Queiroz and K. R. Rao, "Time-varying lapped transforms and wavelet packets," *IEEE Trans. Signal Processing*, vol. 41, pp. 3293–3305, Dec. 1993.

[4] C. Herley, J. Kovacevic, K. Ramchandran, and M. Vetterli, "Tilings of the time-frequency plane: Construction of arbitrary orthogonal bases and fast tiling algorithms," *IEEE Trans. Signal Processing*, vol. 41, pp. 3341–3359, Dec. 1993.

[5] I. Sodagar, K. Nayebi, and T. P. Barnwell, "Time-varying filter banks and wavelets," *IEEE Trans. Signal Processing*, vol. 42, pp. 2983–2996, Nov. 1994.

[6] R. Gopinath and C. S. Burrus, "Factorization approach to unitary time-varying filter banks trees and wavelets," *IEEE Trans. Signal Processing*, vol. 43, pp. 666–680, Mar. 1995.

[7] R. L. de Queiroz, "On lapped transforms," Ph.D. Dissertation, Univ. of Texas at Arlington, Aug. 1994.

[8] C. Chen, "Adaptive transform coding via quadtree-based variable block-size DCT," in *Proc. ICASSP*, May 1989.

[9] K. Ramchandran and M. Vetterli, "Best wavelet packet bases in a rate distortion sense," *IEEE Trans. Image Processing*, vol. 2, pp. 160–175, Apr. 1993.

[10] G. Sullivan and R. Baker, "Efficient quadtree coding of images and video," *IEEE Trans. Image Processing*, vol. 3, pp. 327–331, May 1994.

[11] J. Vaisey and A. Gersho, "Image compression with variable block size segmentation," *IEEE Trans. Signal Processing*, vol. 40, pp. 2040–2060, Aug. 1992.

Transform Domain Adaptive Linear Phase Filter

Soo-Chang Pei and Chien-Cheng Tseng

Abstract—**This correspondence describes a new adaptive linear-phase filter whose weights are updated by normalized least-mean-square (LMS) algorithm in transform domain. This algorithm provides faster convergence rate compared with the time domain linear phase LMS algorithm. Various real-valued orthogonal transforms are investigated such as discrete cosine transform (DCT), discrete Hartley transform (DHT), and power of two (PO2) transform, etc. By using the symmetry property of transform matrix, an efficient implementation structure is proposed. A system identification example is presented to demonstrate its performance.**

I. Introduction

In recent years, adaptive filtering has been widely used in various applications such as equalization, noise cancelation, and system identification [1], [2]. Until now, one of the popular adaptive algorithms is the Widrow–Hoff least-mean-square (LMS) algorithm. This is due to its simple realization and well-documented performance analysis. However, a major drawback of the LMS algorithm is its slow convergence rate when a long impulse response FIR filter is employed. One of the methods to achieve higher convergence speed is to implement adaptive filters in the frequency domain, i.e., the discrete Fourier transform (DFT) domain [3]. The DFT is later extended to other orthogonal transforms such as the discrete cosine transform (DCT), the power of two (PO2) transform, etc., [4], [5]. On the other hand, considerable effort has been spent in developing the recursive least squares (RLS) method for adaptive filtering [2]. The RLS algorithm gives considerable improvement in convergence rate over the LMS algorithm, but it requires higher storage requirements and is computationally intensive over LMS. In order to overcome the limitations of RLS method, the fast RLS algorithm has been suggested [12]. Although the computation complexity of the fast RLS algorithm is proportional to the filter length, it usually requires a more sophisticated program to implement it.

Manuscript received April 27, 1995; revised June 12, 1996. The associate editor coordinating the review of this paper and approving it for publication was Dr. Andreas Spanias.

The authors are with the Department of Electrical Engineering, National Taiwan University, Taipei 10764, Taiwan, ROC.

Publisher Item Identifier S 1053-587X(96)09065-4.

In many applications, it is desirable to adaptively perform linear-phase filtering to prevent any phase distortion in the observed data. The first example is that the signal passing through a nonlinear phase equalizer will suffer an increasing intersymbol interference level and degrade detectability; therefore, the adaptive linear-phase equalizer is preferred [6], [11]. The second example is that the pseudonoise (PN) spread-spectrum system can be further improved, with respect to its immunity to narrowband interference, by incorporating an adaptive linear-phase filter before despreading operation takes place [7]. The third example is that exact linear-phase processing is useful in data communication and speech processing where precise time alignment is essential [8]. Other typical applications include image processing [9], system identification, adaptive noise cancelation [10], etc. Despite so many adaptive linear phase filters being developed, all of them are performed in time domain. If an LMS algorithm is applied to update filter parameters, it certainly has a slow convergence rate. Thus, it is interesting to perform adaptive linear phase filtering in transform domain. As for adaptive nonlinear phase filtering, a faster convergence rate is undoubtedly expected.

II. Time Domain Adaptive Linear-Phase Filter

The output sequence y_n of an N length FIR filter is obtained by

$$y_n = \mathbf{a}_n^t \mathbf{x}_n \tag{1}$$

where the input signal vector $\mathbf{x}_n$ and the weight vector $\mathbf{a}_n$ are given by

$$\mathbf{x}_n = [x_n \quad x_{n-1} \cdots x_{n-(N-1)}]^t \tag{2}$$

$$\mathbf{a}_n = [a_{n0} \quad a_{n1} \cdots a_{n(N-1)}]^t. \tag{3}$$

The FIR filter is called a linear-phase filter if the following symmetric condition is satisfied:

$$\mathbf{a}_n = \pm \mathbf{J} \mathbf{a}_n \tag{4}$$

where the exchange matrix $\mathbf{J}$ is defined by

$$\mathbf{J} = \begin{bmatrix} 0 & & 1 \\ & \cdot^{\cdot^{\cdot}} & \\ 1 & & 0 \end{bmatrix}. \tag{5}$$

Using the linear-phase condition, the filter output is rewritten as

$$y_n = \hat{\mathbf{a}}_n^t \mathbf{M} \mathbf{x}_n \tag{6}$$

where reduced parameter vector $\hat{\mathbf{a}}_n$ and matrix $\mathbf{M}$ are defined in Table I. By minimizing the mean square error (MSE) between the filter output y_n and the desired signal d_n, i.e.,

$$E[(d_n - \hat{\mathbf{a}}_n^t \mathbf{M} \mathbf{x}_n)^2] \tag{7}$$

we obtain an LMS adaptive algorithm as follows:

$$\begin{aligned} e_n &= d_n - \hat{\mathbf{a}}_n^t \mathbf{M} \mathbf{x}_n \\ \hat{\mathbf{a}}_{n+1} &= \hat{\mathbf{a}}_n + 2\mu e_n \mathbf{M} \mathbf{x}_n \end{aligned} \tag{8}$$

where μ is the step size. Let correlation matrix $\mathbf{R}_{xx} = E(\mathbf{x}_n \mathbf{x}_n^t)$ and cross correlation vector $\mathbf{p}_{xd} = E(\mathbf{x}_n d_n)$; then, it is easy to show that this linear-phase LMS algorithm converges to its optimum solution $(\mathbf{M}\mathbf{R}_{xx}\mathbf{M}^t)^{-1}\mathbf{M}\mathbf{p}_{xd}$ in the mean square sense if step size satisfies condition

$$0 < \mu < \frac{1}{\lambda_{\max}(\mathbf{M}\mathbf{R}_{xx}\mathbf{M}^t)} \tag{9}$$