Connecting Partitioned Frequency-Domain Filters in Parallel or in Cascade

Marcel Joho, *Student Member, IEEE* and George S. Moschytz, *Life Fellow, IEEE*

***Abstract*—The efficient implementation of connected filters is an important issue in signal processing. A typical example is the cascade of two filters, e.g., an adaptive filter with a time-invariant prefilter. The filtering and adaptation is carried out very efficiently in the frequency domain whenever filters with many coefficients are required. This is implemented as a block algorithm by using overlap-save or overlap-add techniques. However, in many real-time applications also, a short latency time through the system is required, which leads to a degradation of the computational efficiency. Partitioned frequency-domain adaptive filters, also known as multidelay adaptive filters, provide an efficient way for the filtering and adaptation with long filters maintaining short processing delays.**

This paper shows a computationally efficient way of implementing two or more partitioned frequency-domain filters in cascade or in parallel when their filter lengths are large. The methods presented require only one fast Fourier transform (FFT) and one inverse fast Fourier transform per input and output port, respectively. The FFT size can be even smaller than the length of the filters. The filters can be either time invariant or adaptive.

***Index Terms*—Cascade of filters, connecting filters, filters in parallel, frequency-domain adaptive filters, multidelay adaptive filters, partitioned frequency-domain adaptive filters.**

I. Introduction

A. Problem Description

ADAPTIVE algorithms are used nowadays in many applications related to acoustics or communications. In some cases, time-invariant or adaptive finite-impulse response (FIR) filters are required with many hundreds or even thousands of filter coefficients, e.g., echo cancelling [1], active noise control [2], or multichannel blind deconvolution [3], [4]. In such situations, the filtering and adaptation is usually carried out in the frequency-domain using block processing with overlap-save or overlap-add techniques [5]. Computationally efficient block algorithms use a large block size and avoid a large overlap between two succeeding input blocks. This results in a large processing delay through the system, which might be unacceptable for many real-time applications. One way to reduce the processing delay is to increase the overlap between two input vectors, but this leads to a degradation of the computational efficiency. *Partitioned frequency-domain filters*, also known as *multidelay filters*, provide a way out of this problem. They have a short system delay and can be implemented efficiently [6]–[9]. They can even be implemented for filter lengths exceeding the size of the fast Fourier transform (FFT).

Another typical situation occurring in adaptive filtering is when two FIR filters are connected in cascade: $h(q^{-1}) = a(q^{-1})b(q^{-1})$. One of them is time invariant and the other adaptive, which is the reason not to combine them into a single filter. This situation occurs, e.g., in adaptive beamforming [10], [11], where a time-invariant filter is used to calibrate a sensor and is followed by an adaptive filter. In other acoustical applications, e.g., teleconferencing systems, two adaptive systems are connected together, an echo canceller and a multichannel blind deconvolution stage [12]–[15]. Both contain adaptive filters but with different update strategies. This makes it difficult to combine the filters into a single adaptive filter with a joint update equation.

This paper presents a computationally efficient implementation of two or more long FIR filters in cascade or in parallel by using filter partitioning, for applications in which a small processing delay between the input and output of the system is required. Update equations are given for the case where the filters are adaptive. The outline of the paper is as follows: in Section II the frequency-domain least mean square (LMS) is described, in Section III filter partitioning and the partitioned frequency-domain LMS is introduced, Section IV contains the main contribution of this paper and describes how partitioned filters can be combined efficiently in cascade, parallel, or in a combined fashion. In Section V, the computational complexity of the proposed algorithms are investigated, and in Section VI a simulation example is given.

B. Notation

The notation used throughout this paper is the following. Vectors are underlined, matrices are boldfaced, vectors and matrices in the time domain are written in lower case, and vectors and matrices in the frequency domain are written in upper case. $\mathbf{P}$ is a projection matrix, $\mathbf{F}$ is the DFT matrix of dimension $C \times C$ where C is the size of the DFT or FFT, i.e., $(\mathbf{F})_{a,b} = \exp(-j(2\pi/C)ab)$ for $a, b = 0, \ldots, C-1$ and $j = \sqrt{-1}$. Matrix and vector transpose, complex conjugation, and Hermitian transpose are denoted by $(\cdot)^T$, $(\cdot)^*$ and $(\cdot)^H = ((\cdot)^*)^T$, respectively. $(\cdot)^{((-1))}$ denotes an element-wise inversion of a vector or a matrix. The element-wise multiplication of two vectors or matrices is denoted by $\odot$. The sample and block indices are denoted by (t) and $[k] \triangleq (kL)$, respectively, where L is the block length. The identity matrix is denoted by $\mathbf{I}$, a vector or a matrix containing only zeros are denoted by $\underline{0}$ and $\mathbf{0}$, respectively. Filter polynomials are given in the q-domain where q and q^{-1} are the sample forward-shift and sample delay operator, respectively

Manuscript received July 1999; revised March 2000. This work was supported in part by the Swiss Federal Institute of Technology, ETH Zurich, Switzerland. This paper was recommended by Associate Editor M. Ismail.

The authors are with the Signal and Information Processing Laboratory, Swiss Federal Institute of Technology, ETH Zurich, Switzerland.

Publisher Item Identifier S 1057-7130(00)06580-0.

[16], [17], e.g., $q^{-d}x(t) = x(t-d)$ and $q^{-dL}x[k] = x[k-d]$. $\deg(\cdot)$ gives the degree of a polynomial in q^{-1}. All filters considered in this paper are FIR. $E\{\cdot\}$ denotes the expectation operator. Rounding to the next smaller or larger integer is denoted by $\lfloor\cdot\rfloor$ and $\lceil\cdot\rceil$, respectively, and $\langle a\rangle_b = a - \lfloor a/b\rfloor \cdot b$ is the remainder when a is divided by the nonzero integer b. Vector or matrix dimensions are given in superscript.

C. Time-Domain Filtering

FIR filters are given as filter polynomials $h(q^{-1}) = h_0 + h_1 q^{-1} + \cdots + h_{N-1}q^{-N+1}$, where $\deg(h(q^{-1})) = N-1$, or as filter vectors $\underline{h} = (h_0, \ldots, h_{N-1})^T$. The two forms are related as follows:

$$h(q^{-1}) = \underline{h}^T \underline{q}. \tag{1}$$

N denotes the number of filter coefficients and

$$\underline{q} = (1, q^{-1}, \ldots, q^{-N+1})^T \tag{2}$$

is a vector containing increasing powers of the delay operator q^{-1} and has the appropriate dimension. In the following, (1) is used in both directions meaning that if a filter is given by a filter polynomial $h(q^{-1})$, the corresponding filter vector $\underline{h}$ is also uniquely defined by (1) and (2), and vice versa. Using (1) and (2), the filter operation in the time domain is defined as

$$\begin{aligned} y(t) &= h(q^{-1})x(t) && (3)\\ &= \underline{h}^T \underline{q} x(t) && (4)\\ &= \underline{h}^T (x(t), \ldots, x(t-N+1))^T. && (5) \end{aligned}$$

If $h(q^{-1})$ is adaptive, then the filter can be updated with a time-domain LMS algorithm

$$\begin{aligned} \underline{h}(t+1) &= \underline{h}(t) + \mu_0 e(t)(x(t), \ldots, x(t-N+1))^T && (6)\\ e(t) &= y_d(t) - y(t) && (7) \end{aligned}$$

where $y_d(t)$ is the desired output signal.

II. Frequency-Domain Filtering and Adaptation

A. Frequency-Domain Filtering

In some acoustical applications, FIR filters are used with many hundreds or even thousands of filter coefficients. In those cases the filtering as well as the adaptation are carried out in the frequency domain using a block algorithm with overlap-save or overlap-add techniques for reasons of computational efficiency [5]. For a block implementation of the filter operation in (3), the following vectors in the time domain are defined for the *overlap-save* method:

$$\begin{aligned} \underline{h} &= (h_0, \ldots, h_{N-1}, 0, \ldots, 0)^T && (8)\\ \underline{x}[k] &= (x(kL+L-C), \ldots, x(kL+L-1))^T && (9)\\ \underline{y}[k] &= (y(kL), \ldots, y(kL+L-1))^T. && (10) \end{aligned}$$

The vector $\underline{h}$ is padded with zeros to have length C. Next, the input vector $\underline{x}[k]$ and the filter coefficient vector $\underline{h}$ are transformed into the frequency domain, i.e.,

$$\underline{X}[k] = \mathbf{F}\underline{x}[k] \tag{11}$$

and $\underline{H} = \mathbf{F}\underline{h}$, respectively. Of course, the FFT is used here for a fast implementation. The linear convolution in the time domain is now replaced by an element-wise multiplication between the input and filter vector in the frequency domain

$$\begin{aligned} \underline{Y}[k] &= \underline{H} \odot \underline{X}[k] && (12)\\ \underline{y}[k] &= \mathbf{P}_{\underline{y}} \mathbf{F}^{-1} \underline{Y}[k] && (13)\\ \mathbf{P}_{\underline{y}} &= \begin{bmatrix} \mathbf{0}^{L\times(C-L)} & \mathbf{I}^{L\times L} \end{bmatrix}^{L\times C}. && (14) \end{aligned}$$

Strictly speaking, (12) performs a cyclic convolution between $\underline{h}$ and $\underline{x}[k]$ in the time domain. But as $\underline{h}$ is zero padded, at least L output samples belonging to the corresponding linear convolution between $\underline{h}$ and $\underline{x}[k]$ can be obtained from every block k. The *output projection matrix* $\mathbf{P}_{\underline{y}}$ in (13) is used to extract those L output samples. The FFT size C thereby has to fulfill [5]

$$C \geq L + N - 1. \tag{15}$$

Here, $C = N + L$ has been used in the derivation of the algorithm for simplicity.

B. Frequency-Domain Adaptation

In the case where $h(q^{-1})$ is an adaptive filter, the update of $\underline{h}[k]$ can be carried out in the frequency domain as well by using a *frequency-domain LMS* (FDLMS) [18]–[22]. To do so, the adaptation error is constructed in the time domain

$$\underline{e}[k] = \underline{y}_d[k] - \underline{y}[k] = (e(kL), \ldots, e(kL+L-1))^T \tag{16}$$

with $\underline{y}_d[k]$ being the vector containing the desired output samples in block k. The error signal is then transformed into the frequency domain, where it is used in the update equation of the FDLMS

$$\begin{aligned} \underline{E}[k] &= \mathbf{F}(\underline{0}^T \quad \underline{e}^T[k])^T && (17)\\ \underline{H}[k+1] &= \mathbf{P}_{\underline{H}}(\underline{H}[k] + \underline{\mu}_0[k] \odot \underline{X}^*[k] \odot \underline{E}[k]) && (18)\\ \mathbf{P}_{\underline{H}} &= \mathbf{F} \begin{bmatrix} \mathbf{I}^N & \mathbf{0} \\ \mathbf{0} & \mathbf{0}^{C-N} \end{bmatrix} \mathbf{F}^{-1}. && (19) \end{aligned}$$

Here, the *filter projection matrix* $\mathbf{P}_{\underline{H}}$ is used to guarantee that $\underline{h}[k+1]$ remains padded with zeros after every adaptation step in the time domain as in (8). If the premultiplication with $\mathbf{P}_{\underline{H}}$ is omitted in (18), the algorithm is known as the *unconstrained FDLMS* (UFDLMS) [23].

The step size of the adaptation can be adjusted in every frequency bin with $\underline{\mu}_0[k]$. Usually, a bin-wise step-size normalization is used to speed up the convergence rate [23]

$$\begin{aligned} \underline{\mu}_0[k] &= \mu_0 \cdot \underline{P}_{\underline{X}}^{((-1))}[k] && (20)\\ \underline{P}_{\underline{X}}[k] &= E\{\underline{X}^*[k] \odot \underline{X}[k]\} && (21)\\ &\cong \lambda \underline{P}_{\underline{X}}[k-1] + (1-\lambda)\underline{X}^*[k] \odot \underline{X}[k] && (22) \end{aligned}$$

where $\underline{P}_{\underline{X}}[k]$ is a vector containing the bin-wise power estimation of $\underline{X}[k]$ and λ is a forgetting factor.

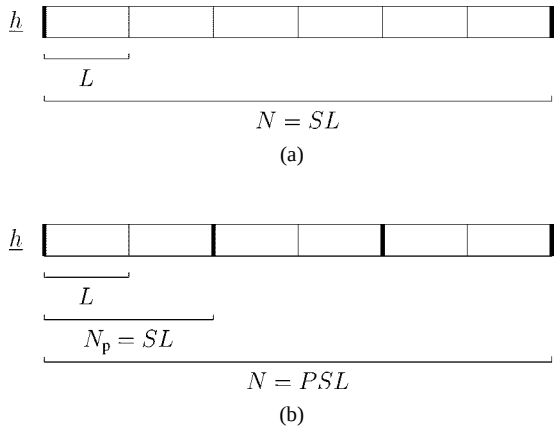

Fig. 1. Partitioning of the filter. (a) No partitioning: $P = 1, S = 6$. (b) With partitioning: $P = 3, S = 2$.

III. Partitioned Frequency-Domain Filtering and Adaptation

A. Partitioned Frequency-Domain Filtering

Instead of increasing the overlap between two succeeding input vectors $\underline{x}[k]$ and $\underline{x}[k+1]$ of the block algorithm, filter partitioning can be applied to reduce the processing delay. To do so, the number of filter coefficients of $\underline{h}$ must be a product

$$N = PSL \tag{23}$$

of three positive integers, where P is the number of *filter partitions*, S is the number of *filter segments* per filter partition, and L is the *block length* of the algorithm. Finding a suitable set of N, P, S, L, and C is a system design task, and an example is given in Section V.

Usually, N and L are specified only roughly and may have to be adjusted slightly to find a sensible choice of P and S, such that (23) is fulfilled. Often, the FFT size C is chosen as a power of two, and is sometimes given by hardware restrictions. The sampling period T_s and the maximum tolerable system delay $\tau_{\max}$ specify an upper bound on the block length L, given by $\tau = 2LT_s \leq \tau_{\max}$. Here, the assumption is made that the time of a whole block is needed for the computation of the filtering and adaptation. As $\underline{x}[k]$ has to be available at the beginning of the block k, and $\underline{y}[k]$ is not available until the end of block k, a second block is required to collect the input samples in $\underline{x}[k]$ in (9), and to simultaneously give out the output samples of $\underline{y}[k]$ in (10). Note, the total group delay of the system consists of the sum of the latency time τ of the system and the group delay of the filter $h(q^{-1})$. Methods to avoid the latency time through the system were described in [24], [25].

Next, the filter $h(q^{-1})$ is subdivided into P filter partitions, each of length $N_p = N/P = SL$, as seen in Fig. 1

$$h(q^{-1}) = \sum_{p=0}^{P-1} h_p(q^{-1}) q^{-pSL} \tag{24}$$

$$\deg(h_p(q^{-1})) \leq N/P - 1 = SL - 1, \quad p \in \{0, \ldots, P-1\} \tag{25}$$

$$\underline{h}_p = \left(h_{pSL}, \ldots, h_{(p+1)SL-1}, 0, \ldots, 0\right)^T \tag{26}$$

where $\underline{h}_p$ has length C. Note that with (25), the filter partitioning in (24) is uniquely defined. If (24) is used in (3) and the delay operator q^{-1} is applied to $x(t)$, then

$$y(t) = \sum_{p=0}^{P-1} h_p(q^{-1}) q^{-pSL} x(t) \tag{27}$$

$$= \sum_{p=0}^{P-1} h_p(q^{-1}) x(t - pSL) \tag{28}$$

$$= \sum_{p=0}^{P-1} y_p(t). \tag{29}$$

Equation (28) can be formulated as a block algorithm and transformed into the frequency domain in a way similar to that described in Section II. For the case where

$$\underline{H}_p[k] = \mathbf{F}\underline{h}_p[k] \tag{30}$$

is adaptive, (12) becomes

$$\underline{Y}[k] = \sum_{p=0}^{P-1} \underline{Y}_p[k] \tag{31}$$

$$= \sum_{p=0}^{P-1} \underline{H}_p[k] \odot \underline{X}[k - pS]. \tag{32}$$

From (28) to (32), we used the equivalence of $\underline{x}[k] \triangleq \underline{x}(kL)$ and as a consequence $\underline{x}[k - pS] = \underline{x}(kL - pSL)$. The need for the filter-partition length $N_p = SL$ to be a multiplicative of the block length L is now seen. The vector $\underline{X}[k-pS] = \mathbf{F}\underline{x}[k-pS]$ has already been computed pS blocks earlier. Therefore, only the latest input vector $\underline{X}[k]$ needs to be computed at block index k, while the remaining $P-1$ input vectors $\underline{X}[k-pS]$ in (32) are already available from previous blocks. If N_p cannot be written as SL, the terms $\underline{x}(kL-pN_p)$ will appear in (32) for an arbitrary N_p, whose FFT has not been computed previously.

Since the summation in (29) is now carried out in (31) in the frequency domain, only one inverse fast Fourier transform (IFFT) is required to obtain the output vector $\underline{y}[k]$ of the actual block k, by using (13). This is similar as without filter partitioning. The constraint on the minimum required FFT size now depends on the length of the filter partition N_p and not on the length of the filter N

$$C \geq L + N_p - 1 = L + SL - 1 = L + N/P - 1. \tag{33}$$

The minimum FFT size C required is now much smaller than that given by (15), where no filter partitioning is applied, especially in the case of a small block size L and a large filter length N. An interesting property of partitioned filters is that the FFT size C can be chosen even smaller than the length N of the filter $\underline{h}$. This is not possible for the conventional overlap-save or overlap-add algorithms, indicated by (15). This makes filter partitioning a powerful tool in the case where C is given or upper bounded, e.g., by hardware restrictions. The whole partitioned filter algorithm is now defined by (9)–(11), (13), (14), (26), (30),

(32), and (33), where $C = L+SL = L+N/P$ is used for simplicity. Fig. 2 graphically shows the derivation of the algorithm.

Note that choosing $P = 1$ results in the frequency-domain filter algorithm described in Section II, and $P = N$ corresponds to filtering in the time domain (5). This makes P a tradeoff parameter, dividing the computation of the filtering and adaptation between the time and frequency domain. Thus, filter partitioning can be seen as a natural extension to filtering in the frequency domain. In the case where *nonuniform filter partitioning* is applied [26], (23) changes to

$$N = \sum_{p=0}^{P-1} S_p L \tag{34}$$

where S_p is the number of filter segments of partition p.

B. Partitioned Frequency-Domain Adaptation

If the filter $h(q^{-1})$ is adaptive, the corresponding update equations for the filter partitions in the frequency domain are obtained by using the *partitioned frequency-domain LMS* (PFDLMS)

$$\underline{H}_p[k+1] = \mathbf{P}_{\underline{H}_p}(\underline{H}_p[k] + \underline{\mu}_p[k] \odot \underline{X}^*[k-pS] \odot \underline{E}[k]) \tag{35}$$

$$\mathbf{P}_{\underline{H}_p} = \mathbf{F}\begin{bmatrix} \mathbf{I}^{SL} & \mathbf{0} \\ \mathbf{0} & \mathbf{0}^{C-SL} \end{bmatrix}\mathbf{F}^{-1}. \tag{36}$$

The FFT size C used in (36) for the filter projection matrix $\mathbf{P}_{\underline{H}_p}$ is now also much smaller than that of $\mathbf{P}_{\underline{H}}$ in (19), i.e., without filter partitioning. However, more FFT and IFFT operations are required for the adaptation if filter partitioning is applied. This is because for every block k, each filter partition is transformed in (35) with $\mathbf{P}_{\underline{H}_p}$ into the time domain and back into the frequency domain, in order to zero pad the last $C - SL$ elements of $\underline{h}_p$. A possible reduction in the number of these operations is to omit the filter projection operations in certain blocks k, while accepting the fact that wrap-around effects of the overlap-save algorithm may slightly disturb the output signals. However, for small step sizes, the last elements of $\underline{h}_p$, which should ideally be zero, cannot deviate much from zero if the filter projection is left out in an update step. Taking this into account, the filter-projection operations can be organized in an alternating manner such that the premultiplication with $\mathbf{P}_{\underline{H}_p}$ in (35) is carried out for only one partition p in block k. The remaining $P-1$ filter partitions are updated with (35) without the premultiplication with $\mathbf{P}_{\underline{H}_p}$. To this end, $\mathbf{P}_{\underline{H}_p}$ is redefined as

$$\mathbf{P}_{\underline{H}_p}[k] = \begin{cases} \mathbf{F}\begin{bmatrix} \mathbf{I}^{SL} & \mathbf{0} \\ \mathbf{0} & \mathbf{0}^{C-SL} \end{bmatrix}\mathbf{F}^{-1}, & p = \langle k \rangle_P \\ \mathbf{I}^C, & \text{otherwise} \end{cases} \tag{37}$$

with $0 \le p \le P-1$. Now, with (37), only one FFT and IFFT are effectively required for the update (35) if *alternating filter projections* are applied, regardless of the number of filter partitions P (see [27], [28]).

Note that since $\mathbf{P}_{\underline{H}_p}\underline{H}_p[k] = \underline{H}_p[k]$ without alternating filter projections, alternatively $\underline{H}_p[k+1] = \underline{H}_p[k] + \mathbf{P}_{\underline{H}_p}(\underline{\mu}_p[k] \odot \underline{X}^*[k-pS] \odot \underline{E}[k])$ could be used instead of (35) for the update of the filter partitions $\underline{H}_p$, with the same computational complexity. However, if alternating filter projections are used, i.e.,

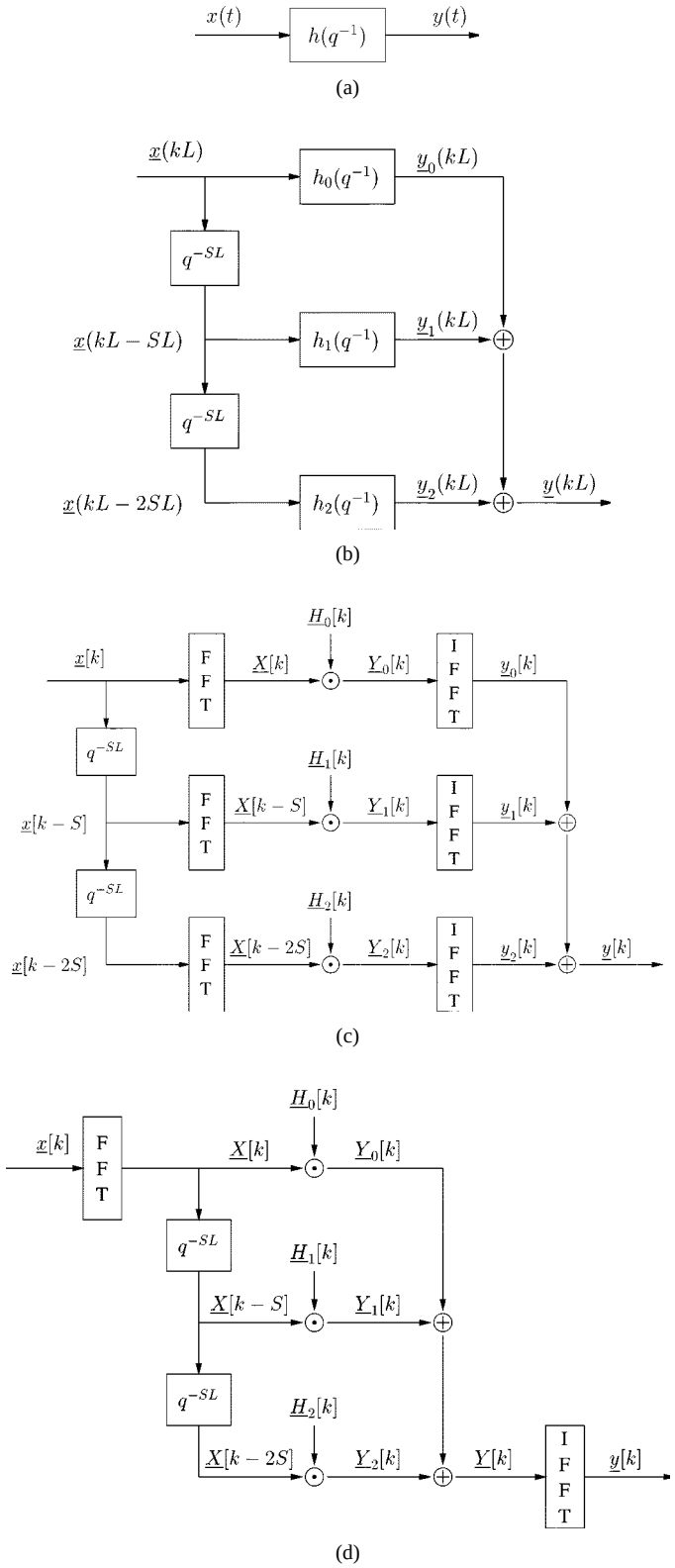

Fig. 2. Derivation of the filter partitioning procedure. (a) Filtering in the time domain. (b) Filter partitioning in the time domain. (c) Transforming each filter partition into the frequency domain and applying overlap-save or overlap-add techniques for the block filtering. (d) FFT's and the block delays q^{-SL} are exchanged at the input, and the IFFT's and summations are exchanged at the output.

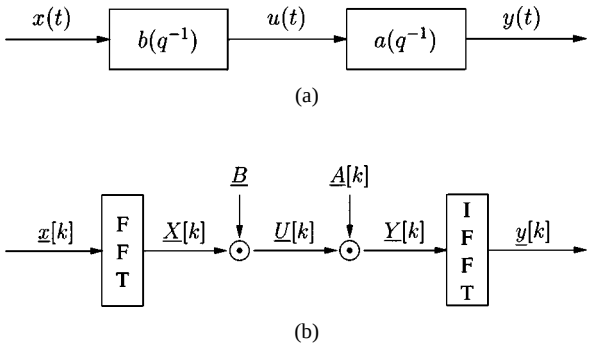

Fig. 3. Two FIR filters in cascade. (a) Time domain: $h(q^{-1}) = a(q^{-1})b(q^{-1})$. (b) Frequency domain: $\underline{H}[k] = \underline{A}[k] \odot \underline{B}$.

$\mathbf{P}_{\underline{H}_p}$ from (37) is used, (35) has the advantage that update errors caused by omitting the zero padding of $\underline{h}_p$ do not propagate by more than $P-1$ blocks.

To speed up the convergence rate of the adaptation, the bin-wise power normalization of the step size μ_0 can be extended to the PFDLMS with

$$\underline{\mu}_p[k] = \underline{\mu}_0[k - pS] \tag{38}$$

where $\underline{\mu}_0[k]$ is taken from (20) and (21), or if a different step size μ_p for the update of every filter partition is chosen

$$\underline{\mu}_p[k] = \mu_p \cdot \underline{P}_{\underline{X}}^{((-1))}[k - pS] \tag{39}$$

where $\underline{P}_{\underline{X}}[k]$ is defined in (21). Using (38) has the advantage that in (35) the terms $\underline{\mu}_0[k-pS] \odot \underline{X}^*[k-pS]$ have already been computed for $p = 1, \ldots, P-1$ and therefore only $\underline{\mu}_0[k] \odot \underline{X}^*[k]$ needs be computed in block k.

IV. Connecting Filters

A. Cascade of Two Filters

First, the case where two or more filters are connected in cascade is considered. A simple example is shown in Fig. 3 where, for some reason, two filters are separately connected in cascade. The concatenation of two filters in the time domain corresponds to a multiplication of their filter polynomials, i.e.,

$$h(q^{-1}) = a(q^{-1})b(q^{-1}) \tag{40}$$

where $\deg(h(q^{-1})) = N-1$, $\deg(a(q^{-1})) = N_a - 1$, $\deg(b(q^{-1})) = N_b - 1$, and

$$N = N_a + N_b - 1. \tag{41}$$

Obviously, both filters could be handled separately, e.g., if one of the filters is short and the other long, the longer one could be transformed into the frequency domain as described in Section II, while the shorter one would be left in the time domain. If both filters are long, both may be transformed into the frequency domain. Instead of computing the intermediate signal $u(t)$ in the time domain before passing it on to the second filter, as shown in Fig. 3(a), $\underline{U}[k]$ can be used as the input to the second filter as shown in Fig. 3(b). To this end, the filter vectors $\underline{a}$ and $\underline{b}$ have to be padded with zeros before being transformed into the frequency domain in order to have equal length C, similar to (8), i.e.,

$$\underline{A} = \mathbf{F}\underline{a} = \mathbf{F}(a_0, \ldots, a_{N_a-1}, 0, \ldots, 0)^T \tag{42}$$

$$\underline{B} = \mathbf{F}\underline{b} = \mathbf{F}(b_0, \ldots, b_{N_b-1}, 0, \ldots, 0)^T. \tag{43}$$

The convolution in (40) can be replaced by an element-wise multiplication of the filter vectors in the frequency domain

$$\underline{H} = \underline{A} \odot \underline{B}. \tag{44}$$

Replacing $\underline{H}$ in (12) with (44) leads to the new filter equation

$$\underline{Y}[k] = \underline{A} \odot \underline{B} \odot \underline{X}[k] = \underline{A} \odot \underline{U}[k] \tag{45}$$

as shown in Fig. 3(b). The filtered output samples in the time domain can be obtained with (13) and (14). Here, the constraint on the minimum FFT-size required for two frequency-domain filters in cascade is derived when (41) is used in (15)

$$C \geq L + N_a + N_b - 2 \tag{46}$$

which is exactly the same as in (15). Obviously, combining $\underline{A}$ and $\underline{B}$ to a single filter $\underline{H}$ with (44), and then using (12), is more efficient than applying (45) to every block k. However, in the case where, e.g., $\underline{A}[k]$ is adaptive, indicated by the temporal index $[k]$, (45) is calculated for every block k as $\underline{H}[k] = \underline{A}[k] \odot \underline{B}$. The update equations for $\underline{A}[k]$ are (16), (17), and

$$\underline{A}[k+1] = \mathbf{P}_{\underline{A}}(\underline{A}[k] + \underline{\mu}_0[k] \odot \underline{U}^*[k] \odot \underline{E}[k]) \tag{47}$$

$$\mathbf{P}_{\underline{A}} = \mathbf{F} \begin{bmatrix} \mathbf{I}^{N_a} & \mathbf{0} \\ \mathbf{0} & \mathbf{0}^{C-N_a} \end{bmatrix} \mathbf{F}^{-1}. \tag{48}$$

To speed up the convergence rate of the adaptation, the bin-wise normalization of the step size μ_0 now depends on the estimated power of $\underline{U}$ and not of $\underline{X}$,

$$\underline{\mu}_0[k] = \mu_0 \cdot \underline{P}_{\underline{U}}^{((-1))}[k] \tag{49}$$

$$\underline{P}_{\underline{U}}[k] = E\{\underline{U}^*[k] \odot \underline{U}[k]\} \tag{50}$$

$$\approx \lambda \underline{P}_{\underline{U}}[k-1] + (1-\lambda)\underline{U}^*[k] \odot \underline{U}[k]. \tag{51}$$

Obviously, the order in which the filters $\underline{A}$ and $\underline{B}$ are arranged can be interchanged for the filtering. However, the adaptation of $\underline{A}$ is preferable, because the intermediate signal $\underline{U}[k]$, which is used in (51) for the step-size normalization, is already available from the filtering (45). Note that the error signal is obtained from the output $y(t)$ and not from $u(t)$.

B. Cascade of Two Partitioned Filters

The filters $a(q^{-1})$ and $b(q^{-1})$ in (40) can be partitioned similarly to (23)–(25), with different numbers of filter partitions and filter segments per partition

$$h(q^{-1}) = \left(\sum_{p_a=0}^{P_a-1} a_{p_a}(q^{-1}) q^{-p_a S_a L} \right) \left(\sum_{p_b=0}^{P_b-1} b_{p_b}(q^{-1}) q^{-p_b S_b L} \right) \tag{52}$$

$$= \sum_{p_a=0}^{P_a-1} \sum_{p_b=0}^{P_b-1} a_{p_a}(q^{-1}) b_{p_b}(q^{-1}) q^{-(p_a S_a + p_b S_b)L}. \tag{53}$$

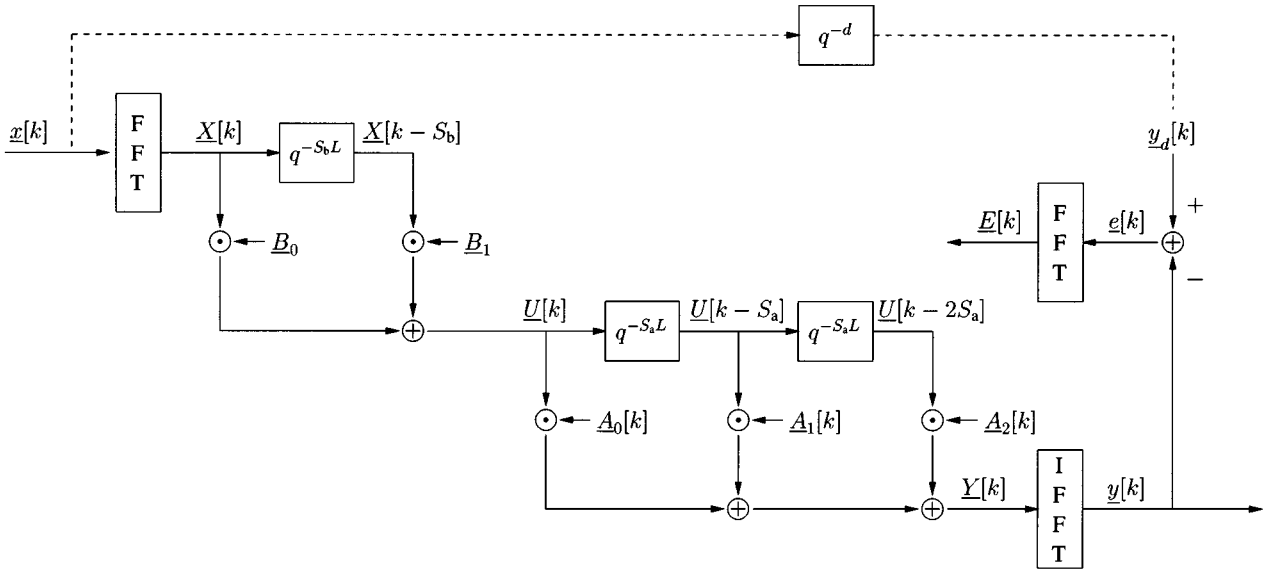

Fig. 4. SISO system with two partitioned frequency-domain filters in cascade: $b(q^{-1})$ is constant and $a(q^{-1})$ is adaptive. $a(q^{-1})$ and $b(q^{-1})$ are partitioned with $P_a = 2$ and $P_b = 3$, respectively. The error signal used for the adaptation of $\underline{A}_p[k]$ is derived in the time domain and then transformed into the frequency domain. The dashed lines show how the desired output signal is generated for the example in Section VI, such that $a(q^{-1})b(q^{-1})$ adapts toward q^{-d}, a simple delay of d time samples.

Here, the filter lengths of $a(q^{-1})$ and $b(q^{-1})$ are assumed to be multiplicatives of the block length L and can therefore be written as

$$N_a = P_a S_a L \tag{54}$$

$$N_b = P_b S_b L. \tag{55}$$

Again, the choice of P_a, S_a, P_b, S_b, and L, which then determine N_a and N_b, belongs to the system design.

The product of two filter partitions in (53) can now be handled similarly to (40). Using (53) in (3) and transforming the filtering into a block frequency-domain algorithm, similar to the steps from (24) to (32), we obtain

$$\underline{Y}[k] = \sum_{p_a=0}^{P_a-1} \sum_{p_b=0}^{P_b-1} \underline{A}_{p_a} \odot \underline{B}_{p_b} \odot \underline{X}[k - p_a S_a - p_b S_b]. \tag{56}$$

It is important to note that every term in the sum of (56) consists of an element-wise multiplication of two filter partitions. This reveals that in every signal path from $\underline{X}[k]$ to $\underline{Y}[k]$ there are, except for a delay, exactly two filter partitions in cascade, one from filter $a(q^{-1})$ and one from $b(q^{-1})$. This is the same situation as in (45), where two filters are placed in cascade. As the lengths of the filter partitions $a_{p_a}(q^{-1})$ and $b_{p_b}(q^{-1})$ are $N_{p_a} = S_a L$ and $N_{p_b} = S_b L$, respectively, the new constraint on the minimum FFT size required to get L output samples for every block k is obtained from (46), i.e.,

$$\begin{aligned} C \geq L + N_{p_a} + N_{p_b} - 2 &= L + S_a L + S_b L - 2 \\ &= L + N_a/P_a + N_b/P_b - 2. \end{aligned} \tag{57}$$

This is a weaker constraint than (46), especially if either N_a or N_b (or both) is much larger than the block length L.

A useful property of (56) is that only one FFT operation is required for every block k to obtain the latest input vector $\underline{X}[k]$, whereas the other input vectors have already been computed in previous blocks. Nevertheless, the summation in (56) can be implemented more efficiently if (52) is transformed directly into a block frequency-domain algorithm

$$\underline{U}[k] = \sum_{p_b=0}^{P_b-1} \underline{B}_{p_b} \odot \underline{X}[k - p_b S_b] \tag{58}$$

$$\underline{Y}[k] = \sum_{p_a=0}^{P_a-1} \underline{A}_{p_a} \odot \underline{U}[k - p_a S_a]. \tag{59}$$

With (58), the intermediate signal vector $\underline{U}[k]$ is defined. The filtering with (56) requires $P_a P_b - 1$ vector additions and $2P_a P_b$ element-wise vector multiplications, whereas with (58) and (59), the filtering requires $P_a + P_b - 2$ vector additions and $P_a + P_b$ element-wise vector multiplications. As $P_a P_b - 1 \geq P_a + P_b - 2$ ($\Leftrightarrow (P_a - 1)(P_b - 1) \geq 0$) with equality if $P_a = 1$ or $P_b = 1$ and $2P_a P_b \geq P_a + P_b$ ($\Leftrightarrow \; P_a(P_b - 1) + P_b(P_a - 1) \geq 0$) with equality if $P_a = P_b = 1$, (58) and (59) require fewer operations than (56) if at least one of the filters is partitioned. If, for some reason, more than two filters are connected in cascade, they can be partitioned in a straightforward manner according to the steps from (52) to (59).

If $a(q^{-1})$ is adaptive, then the filter partitions $\underline{A}_{p_a}[k]$ can be updated with the PFDLMS (16), (17), and

$$\begin{aligned} \underline{A}_{p_a}[k+1] = \mathbf{P}_{\underline{A}_{p_a}} \Big(&\underline{A}_{p_a}[k] + \underline{\mu}_{p_a}[k] \\ &\odot \underline{U}^*[k - p_a S_a] \odot \underline{E}[k] \Big) \end{aligned} \tag{60}$$

$$\mathbf{P}_{\underline{A}_{p_a}} = \mathbf{F} \begin{bmatrix} \mathbf{I}^{S_a L} & \mathbf{0} \\ \mathbf{0} & \mathbf{0}^{C - S_a L} \end{bmatrix} \mathbf{F}^{-1}. \tag{61}$$

Fig. 4 shows the signal-flow diagram of (58) and (59) in the case where $b(q^{-1})$ is constant and $a(q^{-1})$ is adaptive.

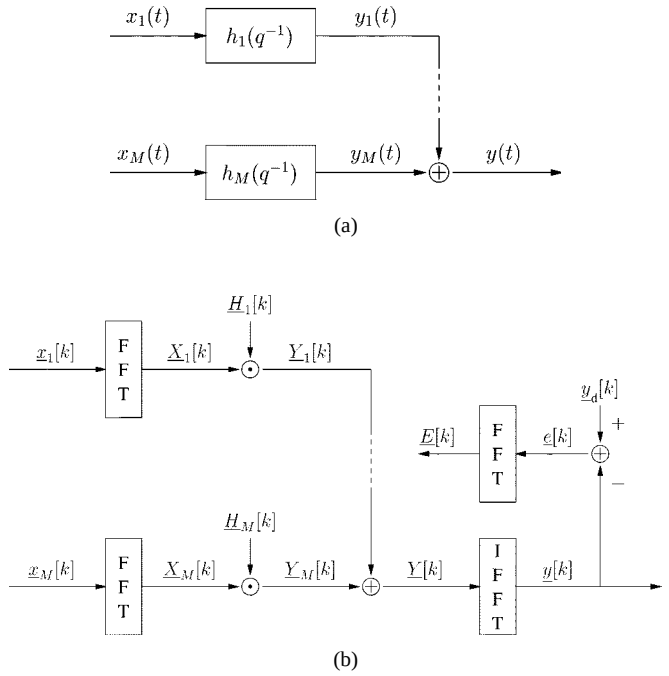

Fig. 5. FIR filters in parallel. (a) Time domain. (b) Frequency domain.

To speed up the convergence rate, a bin-wise step-size normalization with $\underline{\mu}_{p_a}[k] = \underline{\mu}_0[k - p_a S_a]$ and (49) can be used. For small step sizes, *alternating filter projections* can be used to reduce the computational complexity of the filter update.

C. Filters in Parallel

We now consider the case in which M filters are connected in parallel, see Fig. 5 (a). By *parallel*, we mean that there are M different input signals $x_m(t)$, each being filtered with a channel filter $h_m(q^{-1})$ of length N_m and the output of these filters is linearly combined into a single output signal $y(t)$

$$y(t) = \sum_{m=1}^{M} h_m(q^{-1}) x_m(t) \tag{62}$$

$$= \sum_{m=1}^{M} y_m(t) \tag{63}$$

with

$$\deg(h_m(q^{-1})) = N_m - 1, \quad m \in \{1, \ldots, M\}. \tag{64}$$

The filtering of all channels can be transformed into the frequency domain similarly as in the single input–single output (SISO) case described in Section II. To this end, the input signal vectors

$$\underline{x}_m[k] = (x_m(kL + L - C), \ldots, x_m(kL + L - 1))^T \tag{65}$$

are built and then transformed into the frequency domain, i.e.,

$$\underline{X}_m[k] = \mathbf{F}\underline{x}_m[k]. \tag{66}$$

The filter vectors are defined according to (8), and then padded with zeros so that they all have equal length C, i.e.,

$$\underline{h}_m = (h_{m,0}, \ldots, h_{m,N_m-1}, 0, \ldots, 0)^T. \tag{67}$$

$\underline{h}_m$ is the filter vector coresponding to $h_m(q^{-1})$ in (1) with length N_m. With $\underline{H}_m = \mathbf{F}\underline{h}_m$, the filters are transformed into the frequency domain. The filtering and summation in (62) and (63) are then both carried out in the frequency domain

$$\underline{Y}[k] = \sum_{m=1}^{M} \underline{Y}_m[k] \tag{68}$$

$$= \sum_{m=1}^{M} \underline{H}_m[k] \odot \underline{X}_m[k]. \tag{69}$$

This has the advantage that the output vector (10) can be obtained with (13) and (14) by using only one IFFT [see Fig. 5(b)].

In the above derivation, we assumed that every channel filter can have a different length N_m, but use the same block length L and FFT size C for the filtering in the frequency domain. Thus, the minimum required FFT size has to fulfill (15) for every filter length N_m

$$C \geq L + \max\{N_1, \ldots, N_M\} - 1. \tag{70}$$

In the case where the filters $h_m(q^{-1})$ are adaptive, they can be updated by a MISO-FDLMS. As in the SISO case, the error signal is built with (16) and (17). The update equation for the filter in channel m is

$$\underline{H}_m[k+1] = \mathbf{P}_{\underline{H}_m}(\underline{H}_m[k] + \underline{\mu}_m[k] \odot \underline{X}_m^*[k] \odot \underline{E}[k]) \tag{71}$$

where

$$\mathbf{P}_{\underline{H}_m} = \mathbf{F} \begin{bmatrix} \mathbf{I}^{N_m} & \mathbf{0} \\ \mathbf{0} & \mathbf{0}^{C-N_m} \end{bmatrix} \mathbf{F}^{-1} \tag{72}$$

is the filter projection matrix of channel m. Note that the same adaptation error vector $\underline{E}[k]$ is used for the update of all channel filters. Therefore, only one FFT is required here, regardless of the number of channels M. With the help of $\underline{\mu}_m[k]$, the step size of every frequency bin in each channel can be adjusted individually. A convenient choice is a bin-wise power normalization of the step size μ_0 for every channel with (20) and (21), e.g., $\underline{\mu}_m[k] = \mu_0 \cdot \underline{P}_{\underline{X}_m}^{(\langle -1 \rangle)}[k]$ with $\underline{P}_{\underline{X}_m}[k] = E\{\underline{X}_m^*[k] \odot \underline{X}_m[k]\}$. For small step sizes in $\underline{\mu}_m[k]$, the computational complexity can be reduced significantly by using alternating filter projections in (71). To this end, the premultiplication with $\mathbf{P}_{\underline{H}_m}$ in (71) is applied in block k only for one channel m in a cyclic manner. The other $M-1$ filter vectors are updated with (71) but without the premultiplication with $\mathbf{P}_{\underline{H}_m}$. To this end, $\mathbf{P}_{\underline{H}_m}$ is redefined as

$$\mathbf{P}_{\underline{H}_m}[k] = \begin{cases} \mathbf{F} \begin{bmatrix} \mathbf{I}^{N_m} & \mathbf{0} \\ \mathbf{0} & \mathbf{0}^{C-N_m} \end{bmatrix} \mathbf{F}^{-1}, & m = 1 + \langle k \rangle_M \\ \mathbf{I}^C, & \text{otherwise} \end{cases} \tag{73}$$

with $1 \leq m \leq M$. In doing so, only one FFT and IFFT are effectively required for the update in (71), regardless of the number of channels M.

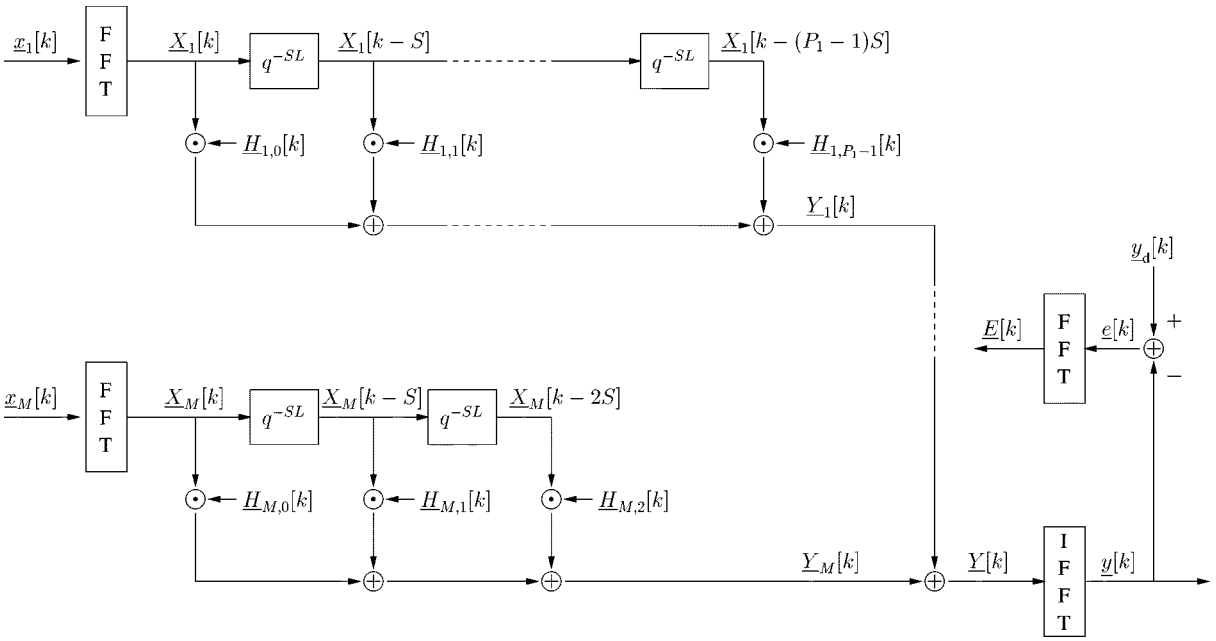

Fig. 6. MISO system with partitioned frequency-domain filters. Here, the case is shown where all channel filters $h_m(q^{-1})$ have a different length N_m and therefore a different P_m, but are partitioned with the same S and L in (23). The error signal used for the adaptation of $\underline{H}_{m,p}[k]$ is derived in the time domain and then transformed into the frequency domain.

D. Partitioned Filters in Parallel

The channel filters $h_m(q^{-1})$ can be partitioned individually following the steps in (23), (24) and (25). Since each filter can have a different length N_m, (23) now becomes

$$N_m = P_m S_m L. \tag{74}$$

The same block length L is taken here for all input channels in order to maintain a single block rate in the whole system. The choice of P_m and S_m is part of the system design. The partitioning of the filters then becomes

$$h_m(q^{-1}) = \sum_{p=0}^{P_m-1} h_{m,p}(q^{-1}) q^{-pS_mL} \tag{75}$$

$$\deg(h_{m,p}(q^{-1})) \le N_m/P_m - 1 = S_mL - 1 = N_{p_m} - 1,$$
$$m \in \{1,\ldots,M\}, \quad p \in \{0,\ldots,P_m-1\} \tag{76}$$

$$\underline{h}_{m,p} = (h_{m,pSL},\ldots,h_{m,(p+1)SL-1},0,\ldots,0)^T. \tag{77}$$

Next, the partitioned filters $h_m(q^{-1})$ in (75) are used in (62) and the delay operator q^{-1} is applied to $x_m(t)$

$$y(t) = \sum_{m=1}^{M} h_m(q^{-1}) x_m(t) \tag{78}$$

$$= \sum_{m=1}^{M} \sum_{p=0}^{P_m-1} h_{m,p}(q^{-1}) q^{-pS_mL} x_m(t) \tag{79}$$

$$= \sum_{m=1}^{M} \sum_{p=0}^{P_m-1} h_{m,p}(q^{-1}) x(t - pS_mL) \tag{80}$$

$$= \sum_{m=1}^{M} \sum_{p=0}^{P_m-1} y_{m,p}(t). \tag{81}$$

Equation (81) is then formulated as a block algorithm and transformed into the frequency domain similarly to the steps from (29) to (32), i.e.,

$$\underline{Y}[k] = \sum_{m=1}^{M} \sum_{p=0}^{P_m-1} \underline{Y}_{m,p}[k] \tag{82}$$

$$= \sum_{m=1}^{M} \sum_{p=0}^{P_m-1} \underline{H}_{m,p}[k] \odot \underline{X}_m[k - pS_m] \tag{83}$$

with $\underline{H}_{m,p}[k] = \mathbf{F}\underline{h}_{m,p}[k]$. The summations in (81) are now carried out in (83) in the frequency domain. Thus, the output of the system can be obtained with (13), using one IFFT only.

Since the same FFT size C is used for all channels, (33) must be satisfied for all filter-partition lengths N_{p_m}

$$C \ge L + \max\{N_{p_1},\ldots,N_{p_M}\} - 1 \tag{84}$$

$$= L + \max\{S_1,\ldots,S_M\}L - 1 \tag{85}$$

$$= L + \max\{N_1/P_1,\ldots,N_M/P_M\} - 1. \tag{86}$$

In this general form, each channel filter is partitioned individually, except that it has the same block length L and FFT size C. To obtain a more regular filtering in (83), the same number of filter segments S' can be chosen for the partitioning of each channel filter, e.g., $S' = \max\{S_1,\ldots,S_M\}$. The number of filter partitions P'_m of channel m is then chosen to be the *smallest* integer number, such that

$$N_m \le P'_m S' L. \tag{87}$$

The effective channel filter length $P'_m S' L$ can then become even larger than the specified length N_m. However, the efficiency of the algorithm, in terms of vector additions and element-wise

vector multiplications in (83) is at least as good as with different choices of S_m because

$$N_m = P_m S_m L \leq P'_m S' L \leq P_m S' L. \tag{88}$$

The left inequality holds because of (74) and (87), and the right inequality holds because we defined P'_m to be the smallest integer such that (87) holds. Thus $P'_m \leq P_m$ which indicates that fewer or the same number of vector additions and multiplications are required in (83) if the same number of filter segments S' is chosen in all channels. Fig. 6 shows a realization of (83), where all channel filters have equal length N and are partitioned with (23) ($N_m = N, P_m = P, S_m = S$ for $m = 1 \ldots M$).

For the case that the lengths of the channel filters N_m have a large variation, the filtering in the frequency domain can become inefficient for the channels with smaller filter lengths, because from (70) the largest channel filter determines the minimum required FFT size C. This is different when the channel filters are partitioned and have the same number of filter segments S', because then the minimum required size of the FFT only depends on S', see (85). If some channel filters are shorter than others, fewer vector additions and element-wise vector multiplications are needed in (83), because shorter filters have a smaller number of filter partitions P_m than longer filters.

In the case where the channel filters $h_m(q^{-1})$ are adaptive, the corresponding filter partitions can be updated by using a MISO-PFDLMS. The error signal $\underline{E}[k]$ is obtained from (16) and (17) and is used for the update of *all* filter partitions in *all* channels. Thus, also in the MISO case, only one FFT is required to transform the adaptation error vector $\underline{e}[k]$ into the frequency domain, regardless of the number of input channels M. The update equation for partition p in channel m is

$$\underline{H}_{m,p}[k+1] = \mathbf{P}_{\underline{H}_{m,p}}(\underline{H}_{m,p}[k] + \underline{\mu}_{m,p}[k] \odot \underline{X}^*_{m,p}[k - pS_m] \odot \underline{E}[k]) \tag{89}$$

$$\mathbf{P}_{\underline{H}_{m,p}} = \mathbf{F} \begin{bmatrix} \mathbf{I}^{S_m L} & \mathbf{0} \\ \mathbf{0} & \mathbf{0}^{C-S_m L} \end{bmatrix} \mathbf{F}^{-1}. \tag{90}$$

If the same number of filter segments is chosen in all channels, e.g., $S_m = S$, the filter projection matrix $\mathbf{P}_{\underline{H}_p}$ from (36) can be used in (89) for all $\mathbf{P}_{\underline{H}_{m,p}}$ instead of (90). The step sizes of the adaptation can be adjusted with $\underline{\mu}_{m,p}[k]$ in every frequency bin in partition p and channel m. A reasonable choice is a channel-wise power normalization with (20), (21), and (38), e.g., $\underline{\mu}_{m,0}[k] = \mu_0 \cdot \underline{P}^{((-1))}_{\underline{X}_m}[k]$ with $\underline{P}_{\underline{X}_m}[k] = E\{\underline{X}^*_m[k] \odot \underline{X}_m[k]\}$ and

$$\underline{\mu}_{m,p}[k] = \underline{\mu}_{m,0}[k - pS_m]. \tag{91}$$

The number of FFT's and IFFT's required in (89) can grow considerably for large M and P_m, due to the filter projection matrix $\mathbf{P}_{\underline{H}_{m,p}}$. As in (37) for the SISO-PFDLMS, and (73) for the MISO-FDLMS, *alternating filter projections* can be applied here also for the MISO-PFDLMS for small step sizes. In block k, the premultiplication with $\mathbf{P}_{\underline{H}_{m,p}}$ in (89) is then carried out only for one pair of m and p. In this case, $\mathbf{P}_{\underline{H}_{m,p}}$ is redefined, as in (92), shown at the bottom of the page, for $1 \leq m \leq M$ and $0 \leq p \leq P-1$. With this technique, only one FFT and one IFFT are effectively required for the update of all filter partitions, regardless of M and P. For simplicity, the assumption was made in (92) that each channel contains the same number of filter partitions P. For the general case, where each channel filter has a different number of partitions, (92) becomes slightly more complicated to describe.

An application of MISO filter partitioning was given in [28], where the MISO-PFDLMS was used to adapt the channel filters of an adaptive beamformer.

The extension to a multiple input–multiple output (MIMO) algorithm is straightforward for filtering. It can be handled similarly to having several MISO systems which share the same input signals. However, care must be taken for the adaptation. In applications where a desired output signal $y_{d,m}(t)$ is given for each output channel, e.g., stereophonic echo cancellation [29]; also, the adaptation can be handled as having several MISO systems in parallel. However, in other applications, e.g., multichannel blind deconvolution, where the desired output signal $y_{d,m}(t)$ is built from the output signal $y_m(t)$ itself, cross coupling between the channel outputs must be introduced for the adaptation, to prevent a separated source signal from appearing at more than one output.

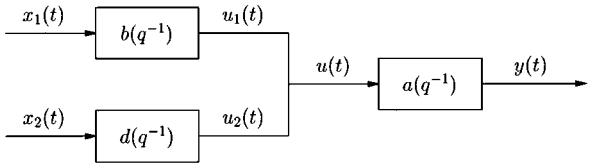

Fig. 7. Two parallel FIR filters in cascade with a third filter.

E. Combining Partitioned Filters in Parallel and in Cascade

Finally, a further combination of connected filters is shown in Fig. 7. The filters $b(q^{-1})$ and $d(q^{-1})$ are in parallel, followed by $a(q^{-1})$ in cascade. The filtering in the time domain is described as

$$y(t) = a(q^{-1})(b(q^{-1})x_1(t) + d(q^{-1})x_2(t)). \tag{93}$$

This structure is used if two filters in parallel have common roots. These are then placed into $a(q^{-1})$, which reduces the order of $b(q^{-1})$ and $d(q^{-1})$. Another situation occurs by combining noise cancelling and adaptive beamforming [10], [11], where $d(q^{-1})$ is a constant *target-signal filter* [30],

$$\mathbf{P}_{\underline{H}_{m,p}}[k] = \begin{cases} \mathbf{F} \begin{bmatrix} \mathbf{I}^{S_m L} & \mathbf{0} \\ \mathbf{0} & \mathbf{0}^{C-S_m L} \end{bmatrix} \mathbf{F}^{-1}, & pM + m = 1 + \langle k \rangle_{PM} \\ \mathbf{I}^C, & \text{otherwise} \end{cases} \tag{92}$$

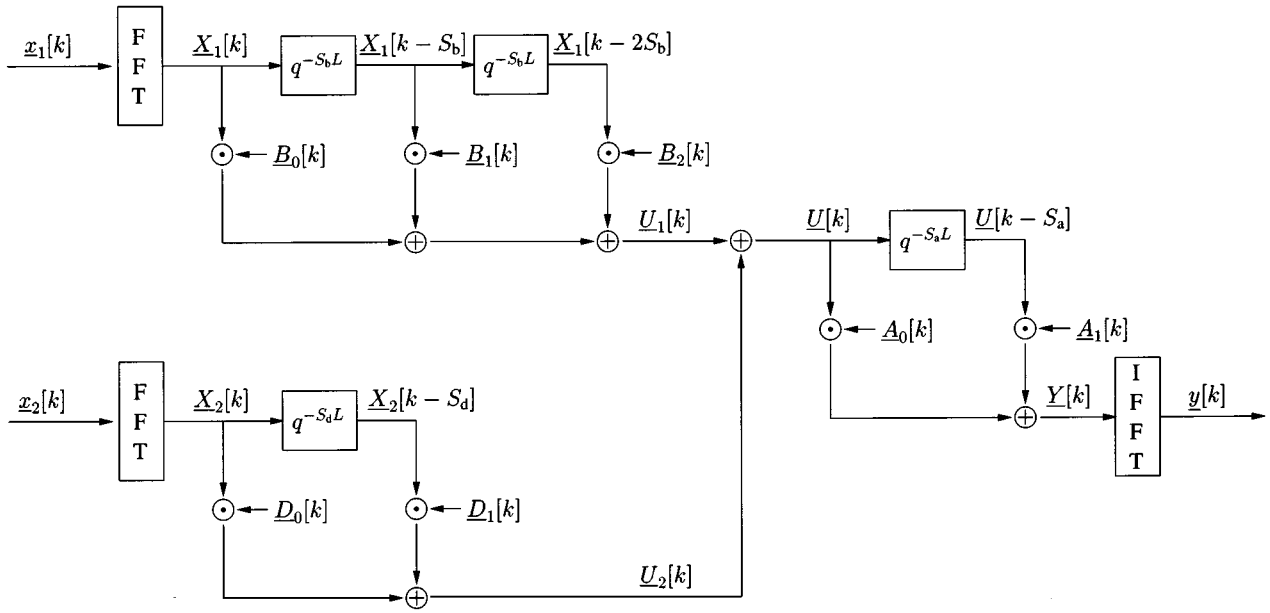

Fig. 8. Combination of three partitioned frequency-domain filters: $b(q^{-1})$ and $d(q^{-1})$ are in parallel and cascade with $a(q^{-1})$, as described in (93). $a(q^{-1})$, $b(q^{-1})$, and $d(q^{-1})$ are partitioned into $P_a = 2$, $P_b = 3$, and $P_d = 2$, respectively. All filters shown here are time variant.

TABLE I
EVALUATION OF THE COMPUTATIONAL COMPLEXITY OF THE MISO-PFDLMS AND MISO-LMS ALGORITHM

MISO-PFDLMS					
Eq.	$\mathcal{A}_r$	$\mathcal{M}_r$	$\mathcal{A}_c$	$\mathcal{M}_c$	$\mathcal{F}$
(66)					M
(83)			$(MP-1)C$	MPC	
(13)					1
(16)	L				
(17)					1
(89)		$2MC$	MPC	MPC	
(90)					$2MP$
(92)					2
MISO-LMS					
(5)	$MN-1$	MN			
(7)	1				
(6)	MN	$MN+1$			

$b(q^{-1})$ is an adaptive filter of the beamformer, and $a(q^{-1})$ is a postfilter which is used for dereverberation and noise reduction. As $a(q^{-1})$ and $b(q^{-1})$ are usually quite long filters, they can be implemented and combined efficiently with partitioned frequency-domain filters, as shown in Fig. 8. Transforming (93) into the partitioned frequency-domain yields

$$\underline{U}[k] = \underline{U}_1[k] + \underline{U}_2[k] \tag{94}$$

$$= \sum_{p=0}^{P_b-1} \underline{B}_p[k] \odot \underline{X}_1[k - pS_b] + \sum_{p=0}^{P_d-1} \underline{D}_p[k] \odot \underline{X}_2[k - pS_d]. \tag{95}$$

$\underline{Y}[k]$ is given by (59). The intermediate signal $\underline{U}[k]$ consists of the sum the output signals of the two parallel filters in the frequency domain and need not be transformed back into the time domain. The filter-partition length of $b(q^{-1})$ and $d(q^{-1})$, $N_b = S_bL$ and $N_d = S_dL$, respectively, can be different; however, both should be a multiple of the block length L. This is for the same reason as described before, namely that only one FFT operation per filter input is required for every block k, $\underline{X}_1[k] = \mathbf{F}\underline{x}_1[k]$ and $\underline{X}_2[k] = \mathbf{F}\underline{x}_2[k]$. The other input vectors in (95) are available from previous blocks. The constraint on the minimum FFT size C required to obtain L output samples at every block k becomes in this case

$$C \geq L + S_aL + \max\{S_b, S_d\}L - 2. \tag{96}$$

For efficiency reasons, S_b and S_d should be chosen to have the same value.

A complex network of several filters can be decomposed into smaller building blocks, each containing either filters in parallel or in cascade. These smaller blocks can then be partitioned by the methods given in this section.

V. COMPLEXITY ANALYSIS

In this section, the computational complexity of the MISO-PFDLMS algorithm is compared with that of the MISO-LMS. The complexity of the SISO-PFDLMS is obtained for $M = 1$. For $P = 1$, the PFDLMS degenerates to the FDLMS. The complexity is defined here as numbers of real additions $\mathcal{A}_r$ and numbers of real multiplications $\mathcal{M}_r$. In Table I, the complexity of the MISO-PFDLMS and MISO-LMS are given for the corresponding equations. A multiplication of a real and a complex number equals two real multiplications, e.g., $1\mathcal{M}_{rc} = 2\mathcal{M}_r$. $\mathcal{A}_c$ and $\mathcal{M}_c$ denote the number of complex additions and multiplications, respectively, and are related by $1\mathcal{A}_c = 2\mathcal{A}_r$ and $1\mathcal{M}_c = 2\mathcal{A}_r + 4\mathcal{M}_r$. Multiplications by zero, one, or minus one (negation) are not taken into account. Subtractions are counted as additions. An FFT or IFFT operation is denoted by $\mathcal{F}$ where $1\mathcal{F} = C\log_2 C\mathcal{A}_c + (C/2)\log_2 C\mathcal{M}_c$. The complexity of the step-size normalization is not taken into account. Equation (91) is used for the analysis of (89) and the fact that $\underline{\mu}_{m,p}[k]$ is a

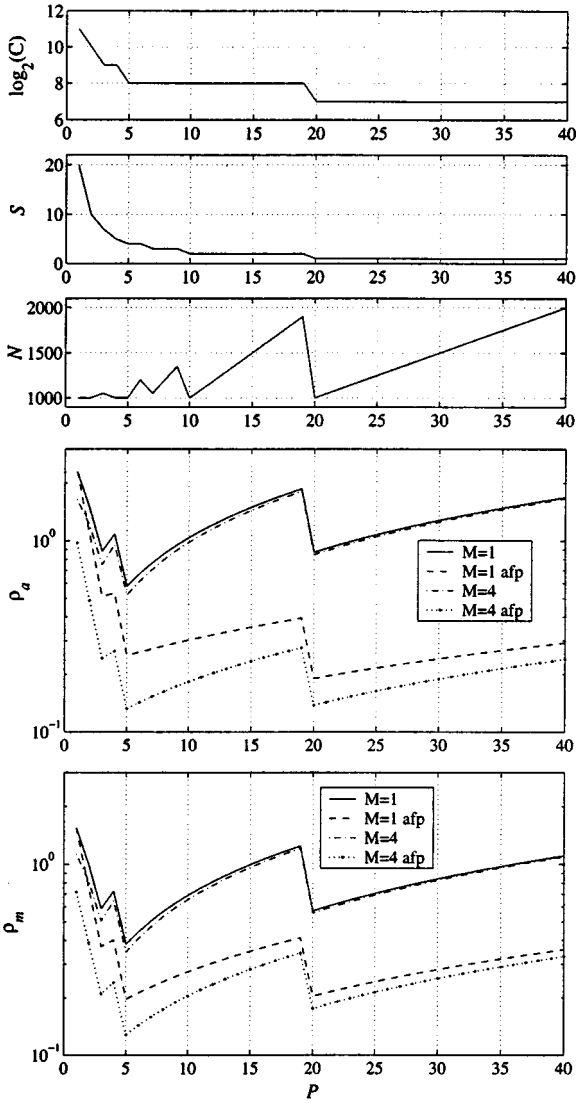

Fig. 9. Complexity comparison of the PFDLMS and LMS. The upper three plots show the partitioning parameters C, S and N versus the number of filter partitions P for $L = 50$. The lower two plots show the complexity ratios ρ_a and ρ_m for $M = 1, 4$, with and without alternating filter projections.

real-valued vector. Hence, to be fair, the PFDLMS is compared with an LMS and not a normalized LMS.

For the comparison, we calculate the ratios between the number of real additions and real multiplications to obtain L output samples, i.e.,

$$\rho_a = \frac{\mathcal{A}_r\text{PFDLMS}}{L \cdot \mathcal{A}_r\text{LMS}} \tag{97}$$

$$\rho_m = \frac{\mathcal{M}_r\text{PFDLMS}}{L \cdot \mathcal{M}_r\text{LMS}}. \tag{98}$$

As an example, we assume that we have the following specifications, which are typical for a real-time acoustical application: all channel filters have equal length N, where $N \geq N_{\min} = 1000$ and the maximal tolerable block length $L \leq L_{\max} = 50$.

The task is now to find a suitable parameter set of P, S, and C which fulfills $PSL \geq N_{\min}$, and yield a small ρ_a and ρ_m. To do so, ρ_a and ρ_m were evaluated for $M = 1, 4$ and $P = 1, \ldots, 40$.

TABLE II
ATTAINED COMPLEXITY REDUCTION FOR THE SISO CASE WITH AND WITHOUT ALTERNATING FILTER PROJECTIONS

Partitioning parameters					Eq. (90)		Eq. (92)	
C	L	S	P	N	ρ_a	ρ_m	ρ_a	ρ_m
256	50	4	5	1000	0.58	0.38	0.25	0.20
128	50	1	20	1000	0.87	0.57	0.19	0.20
128	43	2	12	1032	0.63	0.41	0.17	0.17

For every P, $S = \lceil N_{\min}/(PL) \rceil$ is chosen. The FFT size C is then chosen to be the smallest power of two which satisfied $C \geq (S+1)L - 1$. For a fair comparison, the PFDLMS is compared with an LMS of length $N = N_{\min}$.

Fig. 9 shows graphically the partitioning parameters C, S and N, and the obtained complexity ratios ρ_a and ρ_m, versus the number of filter partitions P for $L = 50$ and $M = 1, 4$. If no alternating filter projections are applied, the greatest complexity reduction is achieved with $P = 5$ filter partitions. However, with alternating filter projections, $P = 20$ gives the greatest reduction for the SISO case $(M = 1)$ (see also Table II). For $M > 1$, the complexity reduction is even larger for the case where alternating filter projections are applied. However, in the MISO case, each filter partition is padded with zeros only every $M \cdot P$ blocks, which makes it preferable to choose a partitioning with a small P, although there might exist a more efficient algorithm with a larger P.

In a second step, an exhaustive search is performed to find the smallest achievable ρ_a and ρ_m for the same specifications as above, allowing alternating filter projections. Surprisingly, the lowest values are obtained with a much smaller block length than $L_{\max}$, and also a smaller FFT size. With $P = 12$, $L = 43$ and $C = 128$, the computational complexity can be reduced to be only about 17% of the apropriate LMS (see Table II). In this analysis we do not make use of the fact that, for real-valued input data, all transforms are symmetric and require only about half of the additions and multiplications. In doing so, the computational complexity of the PFDLMS can be further reduced by approximately a factor of two. Note, the shorter the block length L, the harder it is to find an efficient algorithm for a given filter length N.

The FFT size C has a strong influence on the convergence behavior because of the decorrelation property of the DFT. If C is chosen too small, the DFT can not decorrelate the input signals enough, leading to a degradation of the convergence rate [1], [27]. A large C helps to obtain a fast convergence rate. Therefore, if different possible combinations of P, S, L, and C achieve about the same computational complexity reduction, the preferred combination is the one with the largest FFT size C. Furthermore, if alternating filter projections are used, a small P is also desirable, because then the filter projection operations are carried out more often for every filter partition.

VI. SIMULATION

In this section, we analyze the concepts of combining several frequency-domain filters presented in this paper. To this end, we set up a simulation example with two filters in cascade, as shown in Fig. 4. The constant filter $b(q^{-1})$ is chosen randomly

and $a(q^{-1})$ is adapted such as to converge toward a delayed inverse of $b(q^{-1})$. Thus, the overall transfer function after convergence becomes $h(q^{-1}) = a(q^{-1})b(q^{-1}) = q^{-d}$. A delay of d time samples is introduced to allow a noncausal expansion of $1/b(q^{-1})$ for the case where $b(q^{-1})$ is nonminimum phase.

The general setup is chosen as follows: the input signal sequence $x(\cdot)$ is white Gaussian noise with unity power. The time-invariant filter $b(q^{-1})$ is normalized such that $\sum |b_n|^2 = 1$. The adaptive filter is initialized to $a(q^{-1}) = 0$. The filter lengths are $N_b = 80$ and $N_a = 600$. The sampling frequency $f_s = 16$ kHz, $d = 200$ and the block length is chosen $L = 20$. The initial power estimation, used for the step-size normalization of μ_0 in (49), was set to $\underline{P_U}[0] = C \cdot \underline{1}$. No alternating filter projections are used for the adaptation of $a(q^{-1})$.

In order to analyze the behavior of the convergence depending on the filter partitioning chosen, the filters $a(q^{-1})$ and $b(q^{-1})$ are partitioned with (54) and (55) in several ways (see Table III). The FFT size C is chosen to be the smallest power of two which satisfies (57). For each partitioning parameter set $\mathcal{P}$, the step size μ_0 is chosen to achieve the fastest initial decay of the learning curve. The forgetting factor λ is used in (51) for the estimation of $\underline{P_U}[k]$. By setting $\lambda = 1$, we obtain a block LMS (BLMS), whose performance is used in this analysis as a reference, see partitioning (a). In this case no true step-size normalization is performed, as $\underline{P_U}[k] = \underline{P_U}[0] = C \cdot \underline{1}$.

The upper two plots in Fig. 10 show the impulse response of the constant filter $b(q^{-1})$ and the magnitude of the corresponding transfer function $|B(f)|$. Due to the large variation of $|B(f)|$, the intermediate signal $u(\cdot)$ is highly correlated, and therefore slows down the convergence rate of the adaptation of $a(q^{-1})$ with the BLMS. The lower two plots in Fig. 10 show the impulse response and the magnitude of the transfer function of the adaptive filter $a(q^{-1})$ after convergence. Since FIR filters can generate a transfer function with deep notches much better than with sharp peaks, the impulse response of $a(q^{-1})$ becomes much longer than the one of $b(q^{-1})$. It is clearly seen from the impulse response of $a(q^{-1})$, that the filter $b(q^{-1})$ must be nonminimum phase.

Fig. 11 shows the derived learning curves with the partitioning parameter sets given in Table III. With the partitioning (b), the adaptation initially converges faster than with (a). However, after a while, the algorithm gets unstable. The reason for this behavior is twofold. The first reason is due to the power estimation $\underline{P_U}[k]$, which is used in (49) to speed up the convergence. Since in (58) the term $\underline{B}_{p_b} \odot \underline{X}[k - p_b S_b]$ effectively performs a circular convolution in the time domain, the vector $\mathbf{F}^{-1}\underline{U}[k]$ contains elements which do not coincide with those of a linear convolution. As a consequence, the estimate $\underline{P_U}[k]$ is disturbed also by wrap-around effects; the larger N_{p_b} is compared to the FFT size C, the larger the disturbance. Note the difference between the convergence behavior of (c) and (d). The second one lies in the update equation for $\underline{A}_{p_a}[k+1]$. In (60), the term $\underline{\mu}_{p_a}[k] \odot \underline{U}^*[k - p_a S_a] \odot \underline{E}[k]$ effectively performs a circular convolution of *three* vectors or filters in the time domain. If no step-size normalization is applied, the filter $\mathbf{F}^{-1}\underline{\mu}_{p_a}[k]$ is simply a scaled impulse

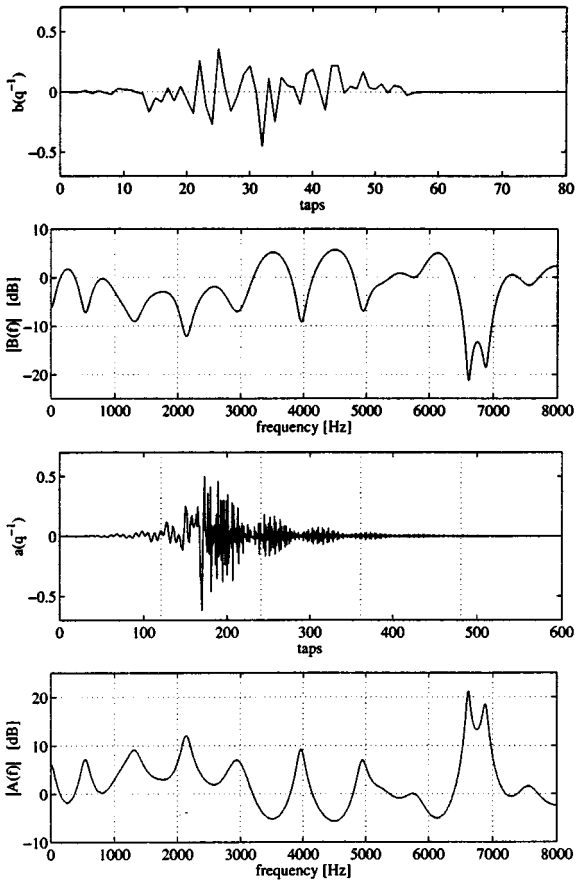

Fig. 10. Impulse response and transfer function of constant filter $b(q^{-1})$ and adaptive filter $a(q^{-1})$ after convergence. The case (d) is shown, where $a(q^{-1})$ is subdivided into five partitions, indicated by the dashed lines.

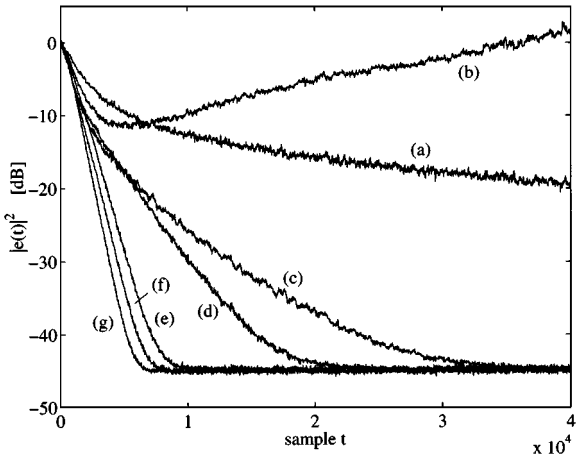

Fig. 11. Convergence behavior for the partitioning parameter sets given in Table III. The learning curves are averages over 100 runs.

TABLE III
PARTITIONING PARAMETER SETS

$\mathcal{P}$	L	C	S_a	P_a	S_b	P_b	λ	μ_0
(a)	20	64	1	30	1	4	1.0	0.06
(b)	20	64	1	30	1	4	0.9	0.06
(c)	20	128	3	10	2	2	0.9	0.25
(d)	20	128	3	10	1	4	0.9	0.25
(e)	20	256	6	5	4	1	0.9	0.25
(f)	20	512	15	2	4	1	0.9	0.5
(g)	20	1024	30	1	4	1	0.9	1.0

and no wrap-around effects harm the update of $a_{p_a}(q^{-1})$, as long as (57) is fulfilled. However, depending on the autocorrelation of the signal $u(\cdot)$, the impulse response of the filter $\mathbf{F}^{-1}\underline{\mu}_0[k] = \mu_0 \cdot \mathbf{F}^{-1}\underline{P}_U^{((-1))}[k]$, which is symmetric in the circular sense and therefore of linear phase, can be quite long. Thus, the FFT size C of the PFDLMS should be enlarged because of the adaptation, depending on the extent of the impulse response of $\mathbf{F}^{-1}\underline{P}_U^{((-1))}[k]$. This implies, that for highly correlated input signals, e.g., narrowband signals, the FFT size should be chosen much larger than $N_{p_a} + N_{p_b}$. This remark is also true for the FDLMS or the PFDLMS described in Sections II and III, respectively. Note, the output signal $y(\cdot)$ is unaffected by wrap around effects, as long as (57) is fulfilled. Increasing the FFT size C helps only to improve the adaptation.

The adaptation with the partitioning sets (c)–(g) is stable and achieves the same MSE in the steady state. Furthermore, the convergence rate is considerably faster than compared with (a). The influence of the FFT size C to the convergence rate is clearly seen. Increasing the FFT size C helps to speed up the convergence rate. However, once the DFT decorrelates the input signal of the adaptive filter, enlarging the DFT size C does not achieve any further improvement on the convergence rate. It is the decorrelation property of the DFT which helps the bin-wise step-size normalization to speed up the convergence rate. Pre-whitening the input signal of the adaptive filter has a similar effect.

VII. Summary

Filtering with long FIR-type filters is usually carried out in the frequency domain by using overlap-save or overlap-add techniques. However, in the case where only a small block-length is tolerable in order to guarantee a short processing delay through the system, these techniques become inefficient. In this case, partitioned frequency-domain filters provide an alternative. Such filters can be efficiently implemented, and require a smaller FFT size than comparable standard overlap-save or overlap-add techniques. The FFT size can be even smaller than the length of the filter, which is interesting if the FFT size is restricted by hardware considerations.

This paper has described efficient implementations of two or more filters in cascade or in parallel. Using the methods and the terminology introduced here, the extension to the case where several filters are arbitrarily combined in cascade and in parallel is straightforward. The efficiency of the proposed algorithm has been verified by a computational complexity analysis, and corresponding rules providing efficient filter partitioning have been derived. Furthermore, update equations in the frequency domain have been given for the case where the filters are adaptive. In a simulation example, the proposed algorithms have shown considerably faster convergence rates than a block LMS algorithm. Moreover, the influence of the filter partitioning on the convergence rate has been analyzed.

The methods described have great potential in signal processing applications which require very long FIR-type filters, but permit only a short processing delay through the system, e.g., multichannel echo cancelling, active noise control, multichannel blind deconvolution, adaptive beamforming, or any combination thereof.

References

[1] P. C. W. Sommen, "Adaptive Filtering Methods," Ph.D. dissertation, Tech. Univ. Eindhoven, Eindhoven, The Netherlands, 1992.

[2] S. M. Kuo and D. R. Morgan, "Active noise control: A tutorial review," *Proc. IEEE*, vol. 87, pp. 943–973, June 1999.

[3] R. H. Lambert, "Multichannel Blind Deconvolution: FIR Matrix Algebra and Separation of Multipath Mixtures," Ph.D. dissertation, Univ. Southern California, Los Angeles, CA, 1996.

[4] M. Joho, H. Mathis, and G. S. Moschytz, "An FFT-based algorithm for multichannel blind deconvolution," in *IEEE Int. Symp. Circuits and Systems*, Orlando, FL, May 30–June 2 1999, pp. III-203–206.

[5] A. V. Oppenheim and R. W. Schafer, *Discrete-Time Signal Processing*, 1st ed. Englewood Cliffs, NJ: Prentice-Hall, 1989.

[6] J. Soo and K. Pang, "A new structure for block FIR adaptive digital filters," *Proc. IREECON*, vol. 38, pp. 364–367, 1987.

[7] K. P. J. Soo, "Multidelay block frequency domain adaptive filters," *IEEE Trans. Acoust., Speech, and Signal Processing*, vol. 38, pp. 373–376, Feb. 1990.

[8] P. C. W. Sommen, "Partitioned frequency domain adaptive filters," in *Proc. Asilomar Conf. Signals, Systems, and Computers*, 1989, pp. 676–681.

[9] É. Moulines, O. A. Amrane, and Y. Grenier, "The generalized multidelay adaptive filter: Structure and convergence analysis," *IEEE Trans. Signal Processing*, vol. 43, pp. 14–28, Jan. 1995.

[10] O. L. Frost III, "An algorithm for linearly constraint adaptive array processing," *Proc. IEEE*, vol. 60, pp. 926–935, Aug. 1972.

[11] L. J. Griffiths and C. W. Jim, "An alternative approach to linearly constrained adaptive beamforming," *IEEE Trans. Antennas Propagat.*, vol. AP-30, pp. 27–34, Jan. 1982.

[12] D. W. E. Schobben and P. C. W. Sommen, "Transparent communication," in *IEEE Benelux Signal Processing Chapter Symp.*, Leuven, Belgium, Mar. 26–27, 1998, pp. 171–174.

[13] D. W. E. Schobben, "Efficient Adaptive multi-channel concepts in acoustics: Blind signal separation and echo cancellation," Ph.D. dissertation, Technical Univ. Eindhoven, Eindhoven, The Netherlands, 1999.

[14] W. Kellermann, "Echoes and noise with seamless acoustic man-machine interfaces—The challenge persists," in *Proc. Int. Workshop Acoustic Echo and Noise Control*, Pocono Manor, PA, Sept. 27–30, 1999, pp. 1–7.

[15] M. Joho and H. Mathis, "Performance comparison of combined blind/nonblind source separation algorithms," in *Proc. Int. Conf. Independent Component Analysis and Blind Signal Separation*, Aussois, France, Jan. 11–15, 1999, pp. 139–142.

[16] G. C. Goodwin and K. S. Sin, *Adaptive Filtering Prediction and Control*. Englewood Cliffs, NJ: Prentice-Hall, 1984.

[17] L. Ljung, *System Identification*. Englewood Cliffs, NJ: Prentice-Hall, 1987.

[18] E. R. Ferrara, "Fast implementations of LMS adaptive filters," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-28, pp. 474–475, Aug. 1980.

[19] G. A. Clark, S. K. Mitra, and S. R. Parker, "Block implementation of adaptive digital filters," *IEEE Trans. Circuits Syst.*, vol. CAS-28, pp. 584–592, June 1981.

[20] G. A. Clark, S. R. Parker, and S. K. Mitra, "A unified approach to time- and frequency-domain realization of FIR adaptive digital filters," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-31, pp. 1073–1083, Oct. 1983.

[21] E. R. Ferrara Jr., "Frequency-domain adaptive filtering," in *Adaptive Filters*, C. F. N. Cowan and P. M. Grant, Eds. Englewood Cliffs, NJ: Prentice-Hall, 1985, pp. 145–179.

[22] J. J. Shynk, "Frequency-domain and multirate adaptive filtering," *IEEE Signal Processing Mag.*, pp. 14–37, Jan. 1992.

[23] D. Mansour and A. H. Gray Jr., "Unconstrained frequency-domain adaptive filter," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-30, pp. 726–734, Oct. 1982.

[24] D. R. Morgan and J. C. Thi, "Delayless subband adaptive filter architecture," *IEEE Trans. Signal Processing*, vol. 43, pp. 1819–1830, Aug. 1995.

[25] J. D. Z. Chen, H. Bes, J. Vandewalle, I. Evers, and P. Janssens, "A zero-delay FFT-based subband acoustic echo canceller for teleconferencing and hands-free telephone systems," *IEEE Trans. Circuits Syst. II*, vol. 43, pp. 713–717, Oct. 1996.

[26] G. P. M. Egelmeers and P. C. W. Sommen, "A new method for efficient convolution in frequency domain by nununiform partitioning for adaptive filtering," *IEEE Trans. Signal Processing*, vol. 44, pp. 3123–3192, Dec. 1996.
[27] P. G. Estermann, "Adaptive Filter im Frequenzbereich: Analyze und Entwurfsstrategie," Ph.D. dissertation, Zürich, ETH Zürich, Switzerland, 1997.
[28] M. Joho and G. S. Moschytz, "Adaptive beamforming with partitioned frequency-domain filters," presented at the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics, New Paltz, NY, Oct. 19–22, 1997.
[29] M. M. Sondhi, D. R. Morgan, and J. L. Hall, "Stereophonic acoustic echo cancellation—An overview of the fundamental problem," *IEEE Signal Processing Lett.*, vol. 2, pp. 148–151, Aug. 1995.
[30] M. Joho and G. S. Moschytz, "On the design of the target-signal filter in adaptive beamforming," *IEEE Trans. Circuits Syst. II*, vol. 46, pp. 963–966, July 1999.
[31] B. Farhang-Boroujeny, "Analysis and efficient implementation of partitioned block LMS adaptive filters," *IEEE Trans. Signal Processing*, vol. 44, pp. 2865–2868, Nov. 1996.
[32] K. S. Chan and B. Farhang-Boroujeny, "Lattice PFBLMS: Fast converging structure for efficient implementation of frequency-domain adaptive filters," *Signal Processing*, vol. 78, pp. 79–89, July 1999.
[33] B. Widrow, J. R. Glover Jr., J. McCool, J. Kaunitz, C. S. Williams, R. H. Hearn, J. R. Zeidler, E. D. Dong Jr., and R. C. Goodlin, "Adaptive noise cancelling: Principles and applications," *Proc. IEEE*, vol. 63, pp. 1692–1719, Dec. 1975.
[34] P. J. Davis, *Circulant Matrices*. New York: Wiley, 1979.
[35] S. Haykin, *Adaptive Filter Theory*, 3rd ed. Englewood Cliffs, NJ: Prentice-Hall, 1996.

Marcel Joho (S'99) was born in Zurich, Switzerland, in 1967. He received the E.E. Diploma from the Swiss Federal Institute of Technology (ETH), Zurich, in 1993. He is currently working toward the Ph.D. degree at the Signal and Information Processing Laboratory (ISI), ETH.

Since 1993, he has been with the ISI, where he is currently a Senior Assistant. He spent three months at Phonak R&D, Stäfa, Switzerland, developing algorithms for adaptive beamforming and echo cancellation for hearing aids. His research interests include adaptive beamforming, echo cancellation, and blind source separation.

Mr. Joho received the SEV/IEEE Student Prize in 1993 for his diploma thesis "Adaptive Noise Suppression with Nonlinear Filters."

George. S. Moschytz (M'65–SM'77–F'78–LF'00) received the E.E. Diploma and the Ph.D. degree from the Swiss Federal Institute of Technology (ETH), Zurich, Switzerland.

Since 1973, he has been Professor and Director of the Institute for Signal and Information Processing, ETH. From 1963 to 1972, he was with Bell Telephone Labs, Holmdel, NJ, returning as Consultant in 1978, and annually since 1989. He is the author of *Linear Integrated Networks: Fundamentals* (New York: Van Nostrand, 1974), *Linear Integrated Networks: Design* (New York: Van Nostrand, 1975), co-author of *Active Filter Design Handbook* (New York: Wiley, 1981), and editor of *MOS Switched-Capacitor Filters: Analysis and Design* (New York: IEEE Press). He has authored many papers in the field of network theory, design, and sensitivity, and holds several patents in these areas. His present interests include digital, sampled-data and adaptive filters, neural networks for signal processing, and the application of signal processing techniques to biological signals.

Dr. Moschytz is a past President of the IEEE Circuits and Systems (CAS) Society and of the IEEE Swiss Chapter on Digital Communication Systems. From 1981 to 1982, he was President of the Swiss Section of the IEEE. He is presently on the Board of Directors of the IEEE CAS Society, and has held several terms the society's Adcom, as well as on the Editorial Board of the PROCEEDINGS OF THE IEEE. He has been an Associate Editor of the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS and of *Circuits and Systems Magazine*, as well as of several other technical journals. He is an elected member of the Swiss Academy of Engineering Sciences, winner of the Best Paper Award, a past member of the Swiss Electrotechnical Society, and member of the Eta Kappa Nu Honor Society.