FAST ADAPTATION OF FREQUENCY-DOMAIN VOLTERRA FILTERS USING INHERENT RECURSIONS OF ITERATED COEFFICIENT UPDATES

Marcus Zeller and Walter Kellermann

Chair of Multimedia Communications and Signal Processing
University of Erlangen-Nuremberg
Cauerstr. 7, 91058 Erlangen, Germany
{zeller,wk}@LNT.de

ABSTRACT

Adaptive Volterra filters are a popular model for compensating distortions caused by nonlinear structures with memory such as low-quality loudspeakers. This paper proposes a fast version of the recently investigated repeated coefficient updates for the partitioned block frequency-domain adaptive Volterra filter. Exploiting inherent recursions of the iteration procedure yields an efficient realization with a very low additional complexity compared to the usual LMS adaptation. Experimental results for both noise and speech demonstrate a significant acceleration of the filter convergence and overall echo cancellation for realistic nonlinear AEC scenarios.

1. INTRODUCTION

Adaptive Volterra filters are a well-known model for identification and compensation of signal distortions caused by nonlinear structures with memory. A typical application scenario is given by nonlinear acoustic echo cancellation (NLAEC) as depicted in Fig. 1, where an adaptive second-order Volterra filter is used to compensate for the nonlinear echo $y(k)$ recorded by the microphone signal $d(k)$.

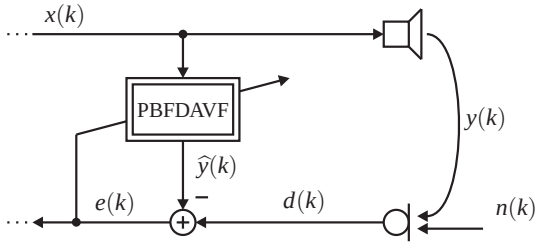

Figure 1: NLAEC scenario where the nonlinear echo $y(k)$ is to be compensated by an adaptive Volterra filter (PBFDAVF)

Commonly, standard NLMS updates are employed as adaptation algorithm in order to keep the costs for the adjustment of the filter within a tolerable range. However, since the nonlinear components are in general only weakly excited, the convergence of the corresponding coefficients is usually slowed down and thus not satisfactory. Therefore, an extension of iterated coefficient updates as they are known from linear adaptive filtering [1, 2, 3] has recently been investigated in [4]. There, an adaptive partitioned block frequency-domain Volterra filter (PBFDAVF) has been modified to incorporate such an iteration procedure and was shown to exhibit a significantly increased convergence with application to NLAEC.

This contribution proposes an efficient, fast version of the iterated coefficient update mechanism by exploiting recursive relations of the repeated filtering and update steps. First, Sec. 2 introduces the notation for this paper and presents the basic definitions of the PBFDAVF before the iteration algorithm is outlined in Sec. 3. The derivation of the essential recursion properties and the resulting fast realization is illustrated in Sec. 4 whereas Sec. 5 discusses the computational complexity of the regarded algorithms. Finally, experimental results for both noise and speech input can be found in Sec. 6.

2. PRELIMINARIES

In this section, we will briefly introduce the PBFDAVF in an explicit notation and a matrix notation. For more detailed information, the reader is referred to [5]. Note that all expressions using bold, underlined fonts denote **vectors** whereas bold fonts refer to **matrices**. Moreover, uppercase letters are reserved for DFT domain quantities and constants whereas lowercase letters denote time-domain signals.

The partitioned block second-order Volterra filtering in the DFT domain reads

$$\widehat{Y}_\nu(m) \;=\; \sum_{b=0}^{B_1-1} \widehat{Y}_{1,\nu,b}(m) + \sum_{b_1=0}^{B_2-1}\sum_{b_2=0}^{B_2-1} \widehat{Y}_{2,\nu,b_1,b_2}(m) \qquad (1)$$

where ν refers to the processed signal frame, m represents the frequency bin and B_1, B_2 denote the corresponding number of filter partitions [5]. The binwise contributions to the output spectra are given by

$$\widehat{Y}_{1,\nu,b}(m) \;=\; \widehat{H}_{1,\nu,b}(m)\, X_{\nu,b}(m) \qquad (2)$$

$$\widehat{Y}_{2,\nu,b_1,b_2}(m) \;=\; \frac{1}{M}\sum_{\widetilde{m}=0}^{M-1} \widehat{H}_{2,\nu,b_1,b_2}(\widetilde{m}, [m-\widetilde{m}]_M)\; X_{\nu,b_1}(\widetilde{m})\, X_{\nu,b_2}([m-\widetilde{m}]_M) \qquad (3)$$

where $\widehat{H}_{1,\nu,b}(m)$ and $\widehat{H}_{2,\nu,b_1,b_2}(m_1,m_2)$ specify the DFT coefficients of the linear and the quadratic kernel respectively. Due to the symmetry of the 2D-DFT, $[\ldots]_M$ denotes a modulo operation w.r.t. the DFT length M. Moreover, the spectra of the input frames correspond to

$$X_{\nu,b}(m) \;=\; \mathrm{DFT}_M\Big\{x_{\nu,b}(\kappa)\Big\} \qquad (4)$$

$$\text{where} \quad x_{\nu,b}(\kappa) \;:=\; x\big(\nu L + \kappa - (M-L) - bN\big) \qquad (5)$$

This work was supported by the Deutsche Forschungsgemeinschaft (DFG) under contract number KE890/2.

and where $\kappa \in \{0,...,M-1\}$. Here, $L \leq N$ is the shift of the overlap-save processing which relates to the partition size N via an overlap factor $\rho = \frac{N}{L}$.

Matrix Notation

For convenience, a matrix formulation of (1) to (5) will be employed in the following. Defining the vector of the time-domain input frames of the filter partitions as

$$\underline{\mathbf{x}}_{\nu,b} := \Big[x_{\nu,b}(0), \ldots, x_{\nu,b}(M-1)\Big]^{\mathrm{T}} \tag{6}$$

we may write the $M \times M$ matrix of the corresponding DFT spectra as

$$\mathbf{X}_{1,\nu,b} := \mathrm{diag}\Big\{\underline{\mathbf{X}}_{1,\nu,b}\Big\} \quad \text{where} \quad \underline{\mathbf{X}}_{1,\nu,b} = \mathbf{F}\,\underline{\mathbf{x}}_{\nu,b} \tag{7}$$

which utilizes the M-point DFT matrix $\mathbf{F}$. The $M \times M^2$ input matrices of the quadratic kernel are then given by

$$\mathbf{X}_{2,\nu,b_1,b_2} := \mathrm{diag}\Big\{\underline{\mathbf{X}}_{2,\nu,b_1,b_2}\Big\} \tag{8}$$

using the concatenated column vector

$$\underline{\mathbf{X}}_{2,\nu,b_1,b_2} := \frac{1}{M}\Big[\underline{\mathbf{X}}^{\mathrm{T}}_{2,\nu,b_1,b_2}(0), \ldots, \underline{\mathbf{X}}^{\mathrm{T}}_{2,\nu,b_1,b_2}(M-1)\Big]^{\mathrm{T}} \tag{9}$$

where the $\underline{\mathbf{X}}_{2,\nu,b_1,b_2}(m)$ comprise all DFT bin products of the input partitions that contribute to a particular output bin m in (3). Furthermore, the corresponding vectors of the adaptive Volterra filter are given as

$$\widehat{\underline{\mathbf{H}}}_{1,\nu,b} := \mathbf{F}\,\widehat{\underline{\mathbf{h}}}_{1,\nu,b} \tag{10}$$

$$\widehat{\underline{\mathbf{H}}}_{2,\nu,b_1,b_2} := \Big[\widehat{\underline{\mathbf{H}}}^{\mathrm{T}}_{2,\nu,b_1,b_2}(0), \ldots, \widehat{\underline{\mathbf{H}}}^{\mathrm{T}}_{2,\nu,b_1,b_2}(M-1)\Big]^{\mathrm{T}} \tag{11}$$

where the elements of $\widehat{\underline{\mathbf{H}}}_{2,\nu,b_1,b_2}(m)$ are arranged analogously to (9). Note that both the time-domain vectors $\widehat{\underline{\mathbf{h}}}_{1,\nu,b}$ of the linear kernel and the corresponding matrices of the quadratic kernel are zero-padded to the DFT length M in order to preserve the partitioned structure of the Volterra kernels [5].

Employing the above definitions, the filtering operations of (2), (3) may be expressed by inner products of the involved matrices. Thus, the Volterra filtering of (1) reads

$$\widehat{\underline{\mathbf{Y}}}_{\nu} = \sum_{b=0}^{B_1-1} \mathbf{X}_{1,\nu,b}\,\widehat{\underline{\mathbf{H}}}_{1,\nu,b} + \sum_{b_1=0}^{B_2-1}\sum_{b_2=0}^{B_2-1} \mathbf{X}_{2,\nu,b_1,b_2}\,\widehat{\underline{\mathbf{H}}}_{2,\nu,b_1,b_2} \tag{12}$$

and yields the resulting time-domain output frame as

$$\widehat{\underline{\mathbf{y}}}_{\nu} = \mathbf{F}^{-1}\,\widehat{\underline{\mathbf{Y}}}_{\nu}. \tag{13}$$

3. METHOD OF ITERATED ADAPTATION

Based on the provided matrix notation, we will now investigate the effect of iterated coefficient updates for the frequency-domain adaptation of the PBFDAVF according to (12). This means that the conventional LMS algorithm is repeated R times on the same frame data as proposed in [1, 2] for adaptive linear filtering scenarios and already outlined in [4] for adaptive Volterra filters.

Hence, the DFT coefficients in (12) are replaced by their iterated versions $\widehat{\underline{\mathbf{H}}}^{(r)}_{1,\nu,b}, \widehat{\underline{\mathbf{H}}}^{(r)}_{2,\nu,b_1,b_2}$ where the bracketed superscript indicates the current iteration number $0 \leq r \leq R-1$. As a consequence, the residual error in the r-th iteration is based on the corresponding filter output $\widehat{\underline{\mathbf{y}}}^{(r)}_{\nu}$ and can be calculated as

$$\underline{\mathbf{e}}^{(r)}_{\nu} = \mathbf{W}\left(\underline{\mathbf{d}}_{\nu} - \widehat{\underline{\mathbf{y}}}^{(r)}_{\nu}\right) \tag{14}$$

where the frame vector $\underline{\mathbf{d}}_{\nu}$ of the microphone reference is defined as in (5), (6) for $b = 0$ and contains the same data for all of the repetitions. Additionally, a windowing matrix

$$\mathbf{W} := \begin{bmatrix} \mathbf{0}_{M-L} & 0 \\ 0 & \mathbf{I}_L \end{bmatrix} \tag{15}$$

is introduced which accounts for the time-domain constraint of the overlap-save method [6]. Here, $\mathbf{I}_L$ denotes the $L \times L$ identity matrix whereas $\mathbf{0}_{M-L}$ specifies a square "all-zero" matrix of the given size.

Since the DFT-domain Volterra filter is implemented adaptively, we apply a standard LMS stochastic gradient algorithm [7] with a *separate normalization and joint iterations of both filter kernels (SNLMS-JI)*. The achieved filter updates are thus given by

$$\widehat{\underline{\mathbf{H}}}^{(r+1)}_{1,\nu,b} = \widehat{\underline{\mathbf{H}}}^{(r)}_{1,\nu,b} + \mathbf{C}_1\,\mathbf{\Psi}_1\,\mathbf{X}^{\mathrm{H}}_{1,\nu,b}\,\mathbf{F}\,\underline{\mathbf{e}}^{(r)}_{\nu} \tag{16}$$

$$\widehat{\underline{\mathbf{H}}}^{(r+1)}_{2,\nu,b_1,b_2} = \widehat{\underline{\mathbf{H}}}^{(r)}_{2,\nu,b_1,b_2} + \mathbf{C}_2\,\mathbf{\Psi}_2\,\mathbf{X}^{\mathrm{H}}_{2,\nu,b_1,b_2}\,\mathbf{F}\,\underline{\mathbf{e}}^{(r)}_{\nu} \tag{17}$$

for all partitions of the linear and the quadratic Volterra kernel respectively. Note that the superscript $^{\mathrm{H}}$ denotes a Hermitian conjugate, the diagonal matrices $\mathbf{\Psi}_1, \mathbf{\Psi}_2$ apply the kernel-dependent step sizes of the SNLMS algorithm [4] and the constraint matrices $\mathbf{C}_1, \mathbf{C}_2$ are used to enforce the zero-padding of the time-domain kernel partitions. For the linear case, the $M \times M$ matrix $\mathbf{C}_1$ reads

$$\mathbf{C}_1 := \mathbf{F}\begin{bmatrix} \mathbf{I}_N & 0 \\ 0 & \mathbf{0}_{M-N} \end{bmatrix}\mathbf{F}^{-1} \tag{18}$$

whereas for the quadratic case, the corresponding $M^2 \times M^2$ constraint matrix $\mathbf{C}_2$ has a more sophisticated structure which is due to the construction of the input vectors by (8), (9). Hence, the definition analogously to (18)

$$\mathbf{C}_2 := \mathbf{F}_{2\mathrm{D,s}} \cdot \mathbf{C}_{2\mathrm{D,s}} \cdot \mathbf{F}^{-1}_{2\mathrm{D,s}} \tag{19}$$

requires the formulation of 2D-DFT matrices $\mathbf{F}_{2\mathrm{D,s}}$ with *scrambled* elements in each row in order to provide the appropriate frequency transform along with an intermediate 2D windowing operation $\mathbf{C}_{2\mathrm{D,s}}$ according to the time-domain kernel partitioning [5]. We will, however, skip the explicit definition of these matrices for reasons of compactness, although their derivation is straightforward.

Note that a continuous usage of the filter coefficients requires a defined "hand-over", i.e.

$$\widehat{\underline{\mathbf{H}}}^{(0)}_{1,\nu+1,b} = \widehat{\underline{\mathbf{H}}}^{(R)}_{1,\nu,b} \tag{20}$$

$$\widehat{\underline{\mathbf{H}}}^{(0)}_{2,\nu+1,b_1,b_2} = \widehat{\underline{\mathbf{H}}}^{(R)}_{2,\nu,b_1,b_2} \tag{21}$$

holds throughout all processed frames. Obviously, using the SNLMS-JI algorithm with only $R = 1$ iterations corresponds to a standard SNLMS algorithm as given in [4, 5].

4. FAST ITERATION ALGORITHM

Although the outlined iteration method provides significant improvements in the convergence speed of adaptive Volterra filters [4], it is highly desirable to develop a more efficient version of the straightforward iteration approach of Sec. 3. In particular, it is imperative to avoid the costs of repeated quadratic filtering and updating due to the large number of second-order kernel coefficients. In the following, we will thus utilize the matrix formulation of the PBFDAVF to exploit the inherent recursions of the iterated update procedure.

Starting from the a-priori error frame $\underline{\mathbf{e}}_\nu^{(0)}$, any further iteration tends to improve the minimization of the residual error block w.r.t. the current input frame and the reference data [1, 4]. All subsequent filter outputs for $1 \leq r \leq R-1$ are therefore characterized by

$$\underline{\widehat{\mathbf{y}}}_\nu^{(r)} \;=\; \underline{\widehat{\mathbf{y}}}_\nu^{(r-1)} + \Delta\underline{\widehat{\mathbf{y}}}_\nu^{(r)} \tag{22}$$

with $\Delta\underline{\widehat{\mathbf{y}}}_\nu^{(r)}$ as the refinement of the filter output due to the already updated Volterra coefficients from the preceding iteration. Using (12), (13) and taking (16), (17) into account, this refinement reads

$$\Delta\underline{\widehat{\mathbf{y}}}_\nu^{(r)} \;=\; \mathbf{G}_\nu\,\underline{\mathbf{e}}_\nu^{(r-1)} \tag{23}$$

where $\mathbf{G}_\nu$ denotes an $M \times M$ iteration gain matrix

$$\mathbf{G}_\nu \;:=\; \mathbf{F}^{-1}\left(\sum_{b=0}^{B_1-1}\mathbf{X}_{1,\nu,b}\,\mathbf{C}_1\,\boldsymbol{\Psi}_1\,\mathbf{X}_{1,\nu,b}^{\mathrm{H}} + \sum_{b_1=0}^{B_2-1}\sum_{b_2=0}^{B_2-1}\mathbf{X}_{2,\nu,b_1,b_2}\,\mathbf{C}_2\,\boldsymbol{\Psi}_2\,\mathbf{X}_{2,\nu,b_1,b_2}^{\mathrm{H}}\right)\mathbf{F} \tag{24}$$

which is constant throughout all performed iterations for a given input frame.

Furthermore, inserting (22) into (14), we observe a recursive relation for the residual error block

$$\underline{\mathbf{e}}_\nu^{(r)} \;=\; \mathbf{W}\left(\underline{\mathbf{d}}_\nu - \underline{\widehat{\mathbf{y}}}_\nu^{(r-1)} - \Delta\underline{\widehat{\mathbf{y}}}_\nu^{(r)}\right) \;=\; \underline{\mathbf{e}}_\nu^{(r-1)} - \Delta\underline{\mathbf{e}}_\nu^{(r)} \tag{25}$$

where the additional error reduction is given by

$$\Delta\underline{\mathbf{e}}_\nu^{(r)} \;:=\; \mathbf{W}\,\Delta\underline{\widehat{\mathbf{y}}}_\nu^{(r)} \;=\; \mathbf{W}\,\mathbf{G}_\nu\,\underline{\mathbf{e}}_\nu^{(r-1)}. \tag{26}$$

Considering (26) reveals that both right-hand side terms of (25) depend on the error block of the previous iteration $r-1$. Since this recursion may be traced back for all previous iteration steps as well, the a-posteriori error of any iteration is yielded by

$$\underline{\mathbf{e}}_\nu^{(r)} \;=\; \left(\mathbf{I}_M - \mathbf{W}\,\mathbf{G}_\nu\right)^r\,\underline{\mathbf{e}}_\nu^{(0)} \tag{27}$$

which merely relies on the a-priori error itself. On the other hand, the complete evolution of the DFT-domain partitions of the linear Volterra kernel for all iterations $0 \leq r \leq R-1$ can be expressed recursively as

$$\begin{aligned}\underline{\widehat{\mathbf{H}}}_{1,\nu,b}^{(r+1)} &= \underline{\widehat{\mathbf{H}}}_{1,\nu,b}^{(r)} + \mathbf{C}_1\,\boldsymbol{\Psi}_1\,\mathbf{X}_{1,\nu,b}^{\mathrm{H}}\,\mathbf{F}\,\underline{\mathbf{e}}_\nu^{(r)} \\ &= \underline{\widehat{\mathbf{H}}}_{1,\nu,b}^{(r-1)} + \mathbf{C}_1\,\boldsymbol{\Psi}_1\,\mathbf{X}_{1,\nu,b}^{\mathrm{H}}\,\mathbf{F}\left(\underline{\mathbf{e}}_\nu^{(r-1)} + \underline{\mathbf{e}}_\nu^{(r)}\right) \\ &\;\;\vdots \\ &= \underline{\widehat{\mathbf{H}}}_{1,\nu,b}^{(0)} + \mathbf{C}_1\,\boldsymbol{\Psi}_1\,\mathbf{X}_{1,\nu,b}^{\mathrm{H}}\,\mathbf{F}\cdot\sum_{i=0}^{r}\underline{\mathbf{e}}_\nu^{(i)}.\end{aligned} \tag{28}$$

Analogously,

$$\underline{\widehat{\mathbf{H}}}_{2,\nu,b_1,b_2}^{(r+1)} \;=\; \underline{\widehat{\mathbf{H}}}_{2,\nu,b_1,b_2}^{(0)} + \mathbf{C}_2\,\boldsymbol{\Psi}_2\,\mathbf{X}_{2,\nu,b_1,b_2}^{\mathrm{H}}\,\mathbf{F}\cdot\sum_{i=0}^{r}\underline{\mathbf{e}}_\nu^{(i)} \tag{29}$$

holds for the partitions of the quadratic DFT-domain kernel.

Efficient Realization

Having developed these recursive formulations for the residual error block and the update of the DFT domain Volterra coefficients, we seek an implementation which allows for an efficient calculation of (28), (29). However, as the DFT matrices $\mathbf{F}$ in (24) are likely to be evaluated by FFT operations in practice, the matrix-based recursion of (27) cannot be applied straightforwardly. Regarding (23), we nevertheless find that any additional refinement of the filter output can be obtained by multiplication of the recent error block with the gain matrix $\mathbf{G}_\nu$, rather than by a complete filtering in all Volterra kernels and partitions. Note that in this case, the involved DFTs may be replaced efficiently by appropriate FFT algorithms.

Furthermore, the computational demands of the iteration method are greatly reduced by allowing for unconstrained updates for all intermediate coefficient adjustments, i.e. if

$$\mathbf{C}_1 \;=\; \mathbf{I}_M \quad\text{and}\quad \mathbf{C}_2 \;=\; \mathbf{I}_{M^2} \tag{30}$$

holds for the constraint matrices. Despite the huge savings in complexity, this simplification still yields attractive convergence behaviour as will be demonstrated by the results of Sec. 6. Additionally regarding the kernel-dependent subband normalization of the applied DFT-domain LMS algorithm, the step size matrices are represented by

$$\boldsymbol{\Psi}_1 \;=\; \mathrm{diag}\left\{\underline{\mu}_{1,\nu}\right\} \quad\text{and}\quad \boldsymbol{\Psi}_2 \;=\; \mathrm{diag}\left\{\underline{\mu}_{2,\nu,\mathrm{aug}}\right\} \tag{31}$$

where

$$\underline{\mu}_{1,\nu} \;:=\; \Big[\mu_{2,\nu}(0),\,\ldots,\,\mu_{2,\nu}(M-1)\Big]^{\mathrm{T}} \tag{32}$$

$$\underline{\mu}_{2,\nu,\mathrm{aug}} \;:=\; \Big[\mu_{2,\nu}(0)\,\underline{\mathbf{1}}_M^{\mathrm{T}},\,\ldots,\,\mu_{2,\nu}(M-1)\,\underline{\mathbf{1}}_M^{\mathrm{T}}\Big]^{\mathrm{T}}. \tag{33}$$

Note that the $\underline{\mathbf{1}}_M$ denote "all-one" vectors of length M, as required for the definition of the augmented step size vector for the quadratic filter partitions. The scalar step sizes for both Volterra kernels $p = 1, 2$ are moreover given by

$$\mu_{p,\nu}(m) \;:=\; \frac{\alpha_p}{S_{p,\nu}(m) + \delta_p} \tag{34}$$

where the α_p represent user-specified control parameters, the δ_p denote regularization constants and the $S_{p,\nu}(m)$ refer to the recursively averaged spectral powers of the kernel inputs as outlined in [4, 5].

Exploiting the above modifications, the iteration gain matrix is reduced to the form

$$\mathbf{G}_\nu \;=\; \mathbf{F}^{-1}\,\mathbf{P}_\nu\,\mathbf{F} \tag{35}$$

where the power matrix

$$\mathbf{P}_\nu \;:=\; \sum_{b=0}^{B_1-1}\mathbf{X}_{1,\nu,b}\,\boldsymbol{\Psi}_1\,\mathbf{X}_{1,\nu,b}^{\mathrm{H}} + \sum_{b_1=0}^{B_2-1}\sum_{b_2=0}^{B_2-1}\mathbf{X}_{2,\nu,b_1,b_2}\,\boldsymbol{\Psi}_2\,\mathbf{X}_{2,\nu,b_1,b_2}^{\mathrm{H}} \tag{36}$$

collects the weighted energy of all kernels and partitions contributing to each DFT bin. Due to the matrices involved in (36), the resulting $\mathbf{P}_\nu$ exhibits an $M \times M$ diagonal structure as well and is constant for a given frame ν. Hence, it may be computed framewise prior to the repeated adaptation with an arbitrary number of iterations.

Consequently, the influence of all iterations $1 \leq r \leq R-1$ on the a-posteriori error can be computed successively by

$$\underline{\mathbf{e}}_\nu^{(r)} = \underline{\mathbf{e}}_\nu^{(r-1)} - \mathbf{W}\,\mathbf{G}_\nu\,\underline{\mathbf{e}}_\nu^{(r)} \quad (37)$$

$$\underline{\mathbf{a}}_\nu^{(r)} = \underline{\mathbf{a}}_\nu^{(r-1)} + \underline{\mathbf{e}}_\nu^{(r)} \quad (38)$$

where $\underline{\mathbf{a}}_\nu^{(r)}$ denotes the corresponding accumulated error block and is initialized by an "all-zero" vector of length M, i.e. $\underline{\mathbf{a}}_\nu^{(0)} := \underline{\mathbf{0}}_M$. Thus, the effective adaptation of the DFT-domain Volterra coefficients can be carried out in a single, direct update and yields

$$\widehat{\underline{\mathbf{H}}}_{1,\nu,b}^{(R)} = \widehat{\underline{\mathbf{H}}}_{1,\nu,b}^{(0)} + \mathbf{C}_1\,\mathbf{\Psi}_1\,\mathbf{X}_{1,\nu,b}^{\mathrm{H}}\,\mathbf{F}\,\underline{\mathbf{a}}_\nu^{(R-1)} \quad (39)$$

$$\widehat{\underline{\mathbf{H}}}_{2,\nu,b_1,b_2}^{(R)} = \widehat{\underline{\mathbf{H}}}_{2,\nu,b_1,b_2}^{(0)} + \mathbf{C}_2\,\mathbf{\Psi}_2\,\mathbf{X}_{2,\nu,b_1,b_2}^{\mathrm{H}}\,\mathbf{F}\,\underline{\mathbf{a}}_\nu^{(R-1)} \quad (40)$$

which is governed by the total accumulation of all intermediate errors. Accordingly, this algorithm is referred to as *fast iterated adaptation (SNLMS-FI)* due to its remarkable computational savings. It should be mentioned here that for the coefficient adjustments of (39), (40), the constraint matrices $\mathbf{C}_1, \mathbf{C}_2$ are not restricted as given by (30). This implies that the actually performed filter update may as well fulfill the zero-padding constraint of the time-domain partitioning [5].

5. COMPLEXITY CONSIDERATIONS

In this section, we present a short analysis of the proposed algorithm in terms of complex multiplications (CMUL) and compare this to the computational burden of a straightforward iteration method.

For these evaluations we investigate a non-overlapping processing with $L \equiv N$, which requires only one transform for the new input data $\underline{\mathbf{x}}_{\nu,0}$ of the first partition as all other block spectra may be acquired by shifting previous input spectra by one partition. A typical implementation is considered, which exploits the symmetries of the second-order Volterra kernel and thus the number of necessary quadratic filter coefficients is roughly halvened [5]. Moreover, we neglect the efforts for computing the power estimates contained in the normalized step sizes (34) and restrict ourselves to unconstrained versions of the adaptation according to (30), in order to cover only the core calculations of these algorithms.

After the input frame has been transformed, the complexity of a straightforward SNLMS-JI algorithm [4] is determined by the loop of R iterations in total, each of which contributes $B_1 M$ and $\frac{1}{2} B_2^2 M (2M+1)$ CMUL by filtering, 2 FFTs by overlap-save processing and error calculation and another $2B_1 M$ and $B_2^2 M^2$ CMUL by updating all kernel partitions. Thus it can be seen that the overall complexity of the straightforward iterated coefficients updates has approximately R times the complexity of the single-update case where $R = 1$.

Up to the calculation of the a-priori error block, the corresponding fast implementation (SNLMS-FI) requires the same operations as the straightforward approach. However, having calculated the power matrix with $2B_1 M$ and $B_2^2 M^2$ CMUL for each frame, there is only need for another 2 FFTs and a weighting of all M DFT bins in order to obtain the output refinements and the accumulation $\underline{\mathbf{a}}_\nu^{(r)}$ throughout each of the remaining $R-1$ iterations. Finally, we have only one direct update of the Volterra coefficients, requiring the same computations as each of the R updates for the SNLMS-JI.

Assuming a standard radix-2 implementation, we have a complexity of $\frac{1}{2} M \,\mathrm{ld}(M)$ CMUL per FFT operation[1]. Thus, the aforementioned calculations yield a total of

$$\left[3RB_1 + \frac{R}{2} B_2^2 + \frac{2R+1}{2}\,\mathrm{ld}(M)\right] M + 2RB_2^2 M^2$$

CMUL for the straightforward SNLMS-JI algorithm and

$$\left[5B_1 + \frac{1}{2} B_2^2 + R\,\mathrm{ld}(M) + (R-1)\right] M + 3B_2^2 M^2$$

complex multiplications for its fast version. As can be seen by these expressions, the term governed by M^2 is scaled by R for the straightforward approach whereas it is fixed at $3B_2^2 M^2$ for the fast SNLMS-FI adaptation method. This reflects the fact that an increasing refinement and iterated error calculation as outlined by (35) to (40) can be achieved without using a repeated quadratic filtering of considerable complexity. In particular, this is due to the mapping of the M^2 complexity of an explicit second-order kernel filtering (3) to a complexity of order M by multiplication with the diagonal power matrix (36).

To provide a more illustrative example, we evaluate these algorithmic demands for a scenario of nonlinear AEC where $N = 64$, $M = 128$, the lengths of the Volterra kernels are specified such that $B_1 = 5$, $B_2 = 1$ and an adaptation with $R = 4$ is performed. The number of operations for overlapping frames ($\rho = 4$) is also given, in order to complete the range of comparable processing methods [4]. In this case, the effort for transforming the input data is noticeably increased, since no DFT spectra from previous frames can be re-used and hence the computational demands are slightly disproportionate. Comparing the corresponding CMUL with those of a non-overlapping SNLMS, the resulting workload w.r.t. complex multiplications per second is listed in Table 1.

Algorithm	**CMUL / sec**	**add. effort**
SNMLS ($\rho = 4$)	18,944,000	+319.9 %
SNMLS-JI ($R = 4$)	17,880,000	+296.3 %
SNLMS-FI ($R = 4$)	7,048,000	+56.2 %
SNLMS ($\rho = 1$)	4,512,000	n/a

Table 1: Comparison of workload for several algorithms

From these figures, the benefit of the SNLMS-FI in terms of computational complexity is obvious, since this fast version consumes only approximately 50% more calculations for $R = 4$ iterations per update step. On the other hand, the demands of the straightforward SNLMS-JI are roughly proportional to the number of iterations.

[1] ld(...) denotes the logarithmus dualis (i.e. base-2 logarithm)

6. ADAPTATION BEHAVIOUR

Contemporary to the analysis of the computational complexity we now demonstrate the effectiveness of this mechanism w.r.t. to an acceleration of the adaptation for frequency-domain Volterra filters. Therefore, an NLAEC scenario as depicted in Fig. 1 is evaluated, where the nonlinear path of the acoustic echo $y(k)$ is given by a second-order Volterra filter with memory lengths according to $N = 64$ and $B_1 = 5, B_2 = 1$ and a corresponding power ratio of 20 dB for the linear-to-nonlinear signal components in $y(k)$. For the task of echo cancellation, a PBFDAVF of the same size is utilized which applies an FFT size $M = 128$ and a frame shift according to the specified overlap factor. All adaptations are performed using the parameters $\alpha_p = 0.3$ and $\delta_p = 0.001$. Moreover, the generated echo is subjected to additive white Gaussian noise $n(k)$ such that an SNR of 30 dB is obtained for the microphone signal in order to investigate a realistic single-talk situation for NLAEC in a noisy environment.

At first, the ERLE measure [5] w.r.t. the reference $d(k)$ and the residual error $e(k)$ is evaluated for a speech-like coloured, Laplacian noise input. As illustrated in Fig. 2, there is a significant increase in convergence speed for an adaptation with iterated coefficient updates compared to the single-update case using either non-overlapping frames or processing with $\rho = 4$. Furthermore it can be seen that this enhancement is also maintained by the SNLMS-FI algorithm, although its derivation implies the use of unconstrained updates for the intermediate iterations. The missing constraint, however, allows for a greater degree of freedom in accelerating an adaptation by iterated coefficient updates and thus the fast algorithm in fact yields a slightly better echo cancellation than the straightforward algorithm. Note that this effect is in accordance with experiments where the SNLMS-JI has been applied with only one final constraint after all R iterations.

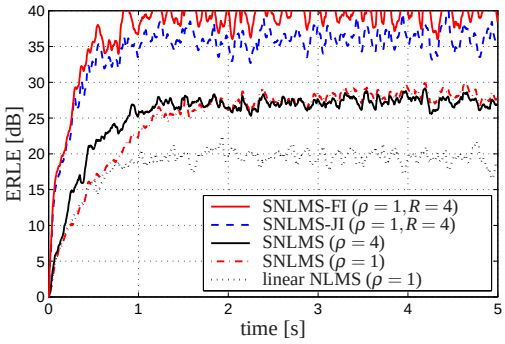

Figure 2: ERLE results for noise input (various algorithms)

As can be seen from Fig. 3, these observations also hold for experiments with real speech signals, as both the SNLMS-JI and the SNLMS-FI yield a significant acceleration of the adaptation speed here as well. Although the computational demands are kept at an acceptable level, the SNLMS-FI is shown to be well capable of obtaining an additional ERLE gain of typically more than 10 dB compared to a conventional SNLMS algorithm – regardless of the frame overlap. Note that in both cases, the achieved steady state gain of the iteration algorithms is above the given SNR, which is due to an exploitation of the short-time signal characteristics and beneficial for noisy AEC scenarios [1, 4].

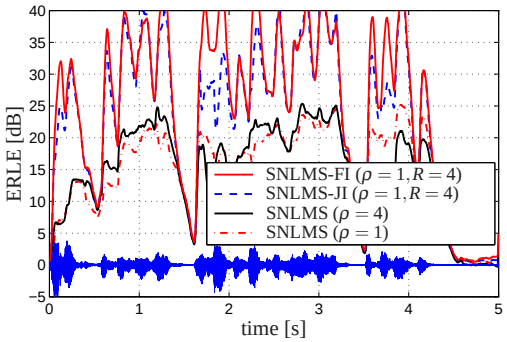

Figure 3: ERLE results for speech input (various algorithms)

7. CONCLUSIONS

We have presented a fast version of the iterated NLMS adaptation for DFT-domain Volterra filters. By exploiting the recursive relations inherent to repeated Volterra filtering and coefficient updates, this method has been shown to yield significant computational savings which are even more apparent than in the case of linear filtering. Nevertheless, our results for both noise and speech inputs demonstrate the same gain in convergence speed and steady state performance as for the straightforward iteration procedure which readily justifies the moderate increase in complexity over a conventional single-update adaptation. Although the derivation of this fast iteration algorithm is based on second-order Volterra filters for presentational convenience, the resulting approach is easily extensible to higher order structures as well.

REFERENCES

[1] K. Eneman and M. Moonen, "Iterated Partitioned Block Frequency-Domain Adaptive Filtering for Acoustic Echo Cancellation", *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 11, no. 2, pp. 143–158, Mar. 2003.

[2] K. Eneman and M. Moonen, "On Iterating the Partitioned Block Frequency-Domain Adaptive Filter", *Technical Report 00-127*, Katholieke Universiteit Leuven, Dec. 2000.

[3] J. Benesty and T. Gänsler, "On Data-Reuse Adaptive Algorithms", *Proc. Int. Workshop on Acoustic Echo and Noise Control (IWAENC)*, Kyoto, Japan, Sept. 2003.

[4] M. Zeller and W. Kellermann, "Iterated Coefficient Updates of Partitioned Block Frequency Domain Second-Order Volterra Filters for Nonlinear AEC", in *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, Honolulu, Hawaii, USA, Apr. 2007.

[5] F. Küch and W. Kellermann, "Partitioned Block Frequency-Domain Adaptive Second-Order Volterra Filter", *IEEE Transactions on Signal Processing*, vol. 53, pp. 564–575, Feb. 2005.

[6] J. G. Proakis and D. G. Manolakis, *Digital Signal Processing: Principles, Algorithms and Applications*, New Jersey: Prentice Hall, 2006 (Fourth Edition).

[7] S. Haykin, *Adaptive Filter Theory*, New Jersey: Prentice Hall, 2002 (Fourth Edition).