36th Asilomar Conference on Signals, Systems, and Computers, Pacific Grove, CA, USA, Nov. 3-6, 2002, Vol. II, pp. 1653–1658.

On frequency-domain implementations of filtered-gradient blind deconvolution algorithms

Marcel Joho[1] & Philip Schniter[2]

[1]Phonak Inc., Champaign, IL, USA, (joho@ieee.org)
[2]Dept. of EE, The Ohio State University, Columbus, OH, USA (schniter.1@osu.edu)

Abstract

This paper describes an efficient realization of an adaptive single-channel blind deconvolution algorithm. The algorithm uses fast convolution and correlation techniques, operates mainly in the frequency domain, and the adaptation of the deconvolution filter is based on the natural gradient learning algorithm. The proposed algorithm is compared to other methods via computational complexity analysis and simulations.

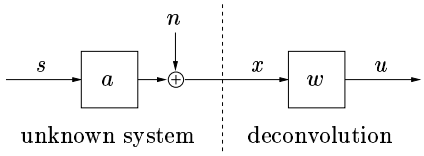

Figure 1: Setup of the blind deconvolution problem.

1 Introduction

1.1 Problem description

Blind deconvolution (BD) is a problem posed by many applications related to acoustics or communications, e.g., dereverberation of a speech signal or equalizing a dispersive communication channel. The general setup of the single-channel blind deconvolution problem is shown in Fig. 1. The *source signal* sequence $s \triangleq \{s_t\}$ is filtered by a causal filter $a \triangleq \{a_0, a_1, \ldots\}$. This process can be described by a convolutional sum

$$x_t = (a * s)_t + n_t = \sum\nolimits_{n=0}^{\infty} a_n s_{t-n} + n_t \qquad (1)$$

where $x \triangleq \{x_t\}$ is the sensor signal sequence and $n \triangleq \{n_t\}$ is the additive sensor noise sequence. In a blind setup, only x is accessible to the algorithm, whereas s, a, and n are unknown.

In a *blind deconvolution* problem we aim at finding a deconvolution filter $w \triangleq \{w_n\}$, such that the output of the deconvolution process

$$u_t = \sum_{n=-\infty}^{\infty} w_n x_{t-n} = \sum_{n=-\infty}^{\infty} (g_n s_{t-n} + w_n n_{t-n}) \qquad (2)$$

retrieves a waveform-preserving estimate of s, possibly delayed. Often either a *zero-forcing* or a MMSE deconvolution filter is the preferred solution. A zero-forcing filter forces the *global-system* $g \triangleq \{g_n\} = w * a$ to become a Kronecker delta, regardless of the sensor-noise amplification $w * n$ at the output u. On the other hand, the MMSE filter best reconstructs the source signal in the mean-square sense. For a real-time application, we require w to be causal and of finite length (FIR). Furthermore, we assume that s is a non-Gaussian distributed sequence.

1.2 Preliminary work

In communications, early publications in the field of *blind channel equalization* are [1–4]. In geophysics the term *blind deconvolution* is more common, as the interest mainly lies in obtaining a model of the system a or its stable inverse a^{-1}, whereas in data communications the term *blind equalization* is more commonly used, as the main interest lies in retrieving the transmitted data s. Most of the those algorithms are adaptive and often show a relatively slow convergence behavior, depending on the characteristics of the unknown channel a and also the degree of non-Gaussianity [5] of s. All of these algorithms operate in the time-domain and the gradient is usually derived by a correlation between the input signals and the output signals passed through a non-linearity.

Fundamental to all work for adaptive algorithms for complex signals and filter coefficients was the work by Widrow *et al.* [6]. Frequency-domain adaptive algorithms which made use of fast convolution techniques were first presented by Ferrara [7] and Clark *et al.* [8,9]. A good overview of frequency-domain adaptive filter algorithms was given by Shynk in [10].

Early implementations which applied frequency-domain adaptive-filter techniques to time-domain blind algorithms are presented in [11–15].

Later on, Amari *et al.* derived in [16] a time-domain blind-deconvolution algorithm based on the natural-gradient learning algorithm. In contrast to the blind-deconvolution algorithms previously mentioned, a *filtered version of the gradient* appears in the update equation. Originally, the natural gradient learning algorithm was applied first in blind source separation [17] and become very attractive because of having the so-called *equivariant property* as opposed to adaptive algorithms based on the regular gradient. For algorithms which enjoy the equivariant property, the convergence depends only on the initial global system $g(0) = w(0) * a$ where $w(0)$ denotes the initial deconvolution filter [18].

Efficient frequency-domain implementations of the natural gradient blind deconvolution algorithm were first derived by Lambert [19, 20]. Shortly afterwards, Douglas and Kung presented in [21] a different implementation of a blind deconvolu-

tion algorithm, also using the natural gradient for the update and operating in the frequency domain.

In the following, we present a different and novel implementation of a natural-gradient based blind deconvolution algorithm. In contrast to the algorithm proposed in [21], the presented algorithm uses fewer FFT / IFFT operations at the expense of larger vectors and, hence, FFT sizes. Moreover, the proposed algorithm does not reuses output samples from previous blocks, which were computed with a different filter, for the update. Therefore the algorithm does not relay on the assumption that the deconvolution filter changes only slowly between two consecutive blocks.

The outline of the paper is as follows: In Section 2 we describe the time domain algorithm, in Section 3 we describe the novel frequency domain algorithm, in Section 5 we give a complexity analysis, in Section 6 we give a simulation example, and finally in Section 7 conclusions are drawn. In Appendix A we summarize some properties of circulant matrices.

1.3 Notation

The notation used throughout this paper is the following: Vectors are written in lower case, matrices in upper case. Matrix and vector transpose, complex conjugation and Hermitian transpose are denoted by $(.)^T$, $(.)^*$, and $(.)^H = ((.)^*)^T$, respectively. The element-wise multiplication of two vectors or matrices is denoted by $\odot$. The identity matrix is denoted by $\mathbf{I}$, a vector or a matrix containing only zeros by $\mathbf{0}$. Vector or matrix dimensions are given in subscript. The DFT matrix $\mathbf{F}$ is defined as

$$[\mathbf{F}_C]_{mn} \triangleq e^{-j\frac{2\pi}{C}mn} \qquad (m,n = 0 \dots C-1) \tag{3}$$

$$\mathbf{F}_C^{-1} = 1/C \cdot \mathbf{F}_C^H = 1/C \cdot \mathbf{F}_C^* \tag{4}$$

where C is the DFT or FFT size. Circulant and diagonal matrices are denoted as $\tilde{\mathbf{A}}$ and $\bar{\mathbf{A}}$, respectively, and $\bar{\mathbf{a}} \triangleq \mathrm{diag}(\bar{\mathbf{A}})$ denotes a vector containing the diagonal elements of $\bar{\mathbf{A}}$. The operation $\tilde{\mathbf{A}} \triangleq \mathcal{C}(\tilde{\mathbf{a}})$ defines a circulant matrix with $\tilde{\mathbf{a}}$ in its first column and $\tilde{\mathbf{a}} \triangleq \mathcal{C}^{-1}(\tilde{\mathbf{A}})$ is the corresponding inverse operation (see Appendix A for more details).

The continuous time and the discrete sample time are denoted by t_c and $(t) \triangleq t \cdot T_s$, respectively, where T_s is the sampling period. The block index is denoted by $[k] \triangleq (k \cdot L)$ where L is the block length (block forward shift).

2 Time-domain implementation

2.1 Sample-wise filtering and update

We start with the single-channel *time-domain blind deconvolution* algorithm (TDBD) which was proposed by Amari *et al.* in [16]. This algorithm is based on the natural gradient learning method. At time sample t we have

$$u_t = \sum_{n=0}^{N} w_n(t)\, x_{t-n} \tag{5}$$

$$v_t = \sum_{n=0}^{N} w_{N-n}^*(t)\, u_{t-n} \tag{6}$$

$$y_t = g(u_t) \tag{7}$$

$$w_n(t+1) = w_n(t) + \mu\, \left(w_n(t) - y_{t-N} v_{t-n}^*\right), \quad n \in [0,N] \tag{8}$$

where $u \triangleq \{u_t\}$ is the output signal, and $\{v_t\}$ is an intermediate signal which is used for the adaptation. The *deconvolution filter* $\{w_n\}$ is an FIR filter of length $N+1$. The nonlinearity $g(.)$ depends on the pdf of the unknown source signal s. Based on a local convergence analysis, $g(.)$ is often suggested to be the *score function* of the pdf of s [22]. However, simulations have shown that the convergence behavior of the algorithm often is relatively robust to some variation of $g(.)$. Therefore, common choices are $g(u_t) \propto \mathrm{sign}(u_t)$ for *super-Gaussian* signals and $g(u_t) \propto u_t|u_t|^2$ for *sub-Gaussian* signals. For convenience, we require that $g(0) \equiv 0$. The algorithm (5) to (8) copes with *complex-valued signals and coefficients*.

2.2 Block-wise filtering and adaptation

Alternatively, we can carry out the filtering and the adaptation of the TDBD algorithm in a block-wise manner. At block k, $t = [kL - L + 1, kL]$, we have

$$u_t = \sum_{n=0}^{N} w_n[k]\, x_{t-n} \tag{9}$$

$$v_t = \sum_{n=0}^{N} w_{N-n}^*[k]\, u_{t-n} \tag{10}$$

$$y_t = g(u_t) \tag{11}$$

$$w_n[k+1] = (1+\mu)\, w_n[k] - \frac{\mu}{L} \sum_{t=kL-L+1}^{kL} y_{t-N} v_{t-n}^* \tag{12}$$

where L is the block size (block-wise forward shift). The block-wise update in (12) is equal to the average of the sample-wise update in (8) over an entire block of L samples. We will refer to the algorithm (9) to (12) as the *block time-domain blind deconvolution* algorithm (BTDBD).

In the following, we will restrict ourselves to the case where $N = L$, which will simplify the derivation. We rewrite (9), (10), and (12) for block k in matrix form:

$$\begin{bmatrix} u_{kL-3L+1} \\ \vdots \\ u_{kL} \end{bmatrix} = \begin{bmatrix} w_L & \cdots & w_0 \\ & \ddots & \\ & w_L & \cdots & w_0 \end{bmatrix} \cdot \begin{bmatrix} x_{kL-4L+1} \\ \vdots \\ x_{kL} \end{bmatrix} \tag{13}$$

$$\begin{bmatrix} v_{kL-2L+1} \\ \vdots \\ v_{kL} \end{bmatrix} = \begin{bmatrix} w_0 & \cdots & w_L \\ & \ddots & \\ & w_0 & \cdots & w_L \end{bmatrix}^* \cdot \begin{bmatrix} u_{kL-3L+1} \\ \vdots \\ u_{kL} \end{bmatrix} \tag{14}$$

$$\begin{bmatrix} w_0[k+1] \\ \vdots \\ w_L[k+1] \end{bmatrix} = (1+\mu) \begin{bmatrix} w_0[k] \\ \vdots \\ w_L[k] \end{bmatrix} - \frac{\mu}{L} \begin{bmatrix} v_{kL-L+1} & \cdots & v_{kL} \\ v_{kL-L} & \ddots & \vdots \\ \vdots & \ddots & v_{kL-L+1} \\ v_{kL-2L+1} & \cdots & v_{kL-L} \end{bmatrix}^* \cdot \begin{bmatrix} y_{kL-2L+1} \\ \vdots \\ y_{kL-L} \end{bmatrix}. \tag{15}$$

In (13) and (14) we have omitted the block index $[k]$ for the filter coefficients and, hence, w_n stands for $w_n[k]$ and

$$\mathbf{w}_k \triangleq (w_0[k], \dots, w_L[k])^T. \tag{16}$$

In block k the update equation (12) needs to be evaluated for $t = kL - L + 1, \ldots, kL$. This determines the dimensionality of (15). In (15) we need the latest $2L$ samples of v, i.e., $\{v_{kL-2L+1}, \ldots, v_{kL}\}$, which then determines the dimensionality of the LHS vector in (14). For the same reason, the LHS vector in (13) needs to be the same as the RHS vector in (14). Eq. (13) and (14) guarantee that all signal samples used for the update in (15) are derived from the current $\mathbf{w}_k$. The equations (13), (14), and (15), together with $y_t = g(u_t)$, yield the BTDBD algorithm in matrix form.

3 Frequency-domain implementation

In the following, we employ fast convolution techniques, to reduce the computationally complexity of the BTDBD algorithm. To this end, we define the following vectors of length L:

$$\mathbf{x}_k \triangleq (x_{kL-L+1}, \ldots, x_{kL})^T \tag{17}$$
$$\mathbf{u}_k \triangleq (u_{kL-L+1}, \ldots, u_{kL})^T \tag{18}$$
$$\mathbf{v}_k \triangleq (v_{kL-L+1}, \ldots, v_{kL})^T \tag{19}$$
$$\mathbf{y}_k \triangleq g(\mathbf{u}_k)\,. \tag{20}$$

where the nonlinearity function $g(.)$ is applied on each vector element. We also define the following vectors of length $4L$:

$$\tilde{\mathbf{w}}_k \triangleq (\mathbf{w}_k^T, \mathbf{0}_{3L-1}^T)^T \tag{21}$$
$$\tilde{\mathbf{x}}_k \triangleq (\mathbf{x}_{k-3}^T, \mathbf{x}_{k-2}^T, \mathbf{x}_{k-1}^T, \mathbf{x}_k^T)^T \tag{22}$$
$$\tilde{\mathbf{u}}_k \triangleq (\dot{\mathbf{u}}_{k-3}^T, \mathbf{u}_{k-2}^T, \mathbf{u}_{k-1}^T, \mathbf{u}_k^T)^T \tag{23}$$
$$\tilde{\mathbf{v}}_k \triangleq (\dot{\mathbf{v}}_{k-2}^T, \mathbf{v}_{k-1}^T, \mathbf{v}_k^T, \dot{\mathbf{v}}_{k-4}^T)^T \tag{24}$$
$$\tilde{\mathbf{y}}_k \triangleq (\mathbf{0}_L^T, \mathbf{0}_L^T, \mathbf{y}_{k-1}^T, \mathbf{0}_L^T)^T\,. \tag{25}$$

We denote a vector that can contain arbitrary elements with a dot e.g., $\dot{\mathbf{u}}_{k-3}$. Furthermore, we define the following circulant matrices

$$\tilde{\mathbf{W}}_k \triangleq \mathcal{C}(\tilde{\mathbf{w}}_k) \tag{26}$$
$$\tilde{\mathbf{V}}_k \triangleq \mathcal{C}(\tilde{\mathbf{v}}_k) \tag{27}$$

and the projection matrices

$$\mathbf{P}_\mathbf{w} \triangleq \begin{bmatrix} \mathbf{I}_{L+1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0}_{3L-1} \end{bmatrix} \tag{28}$$
$$\mathbf{P}_\mathbf{y} \triangleq \begin{bmatrix} \mathbf{0}_{2L} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_L & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0}_L \end{bmatrix}\,. \tag{29}$$

Since all the matrices involved in the equations (13) to (15) are Toeplitz, we can enlarge them to $4L \times 4L$ circulant matrices, such that we can embed (11) and (13) to (15) for block k in the following equations

$$\tilde{\mathbf{u}}_k = \tilde{\mathbf{W}}_k\, \tilde{\mathbf{x}}_k \tag{30}$$
$$\tilde{\mathbf{v}}_k = \tilde{\mathbf{W}}_k^H\, \tilde{\mathbf{u}}_k \tag{31}$$
$$\tilde{\mathbf{y}}_k = g(\mathbf{P}_\mathbf{y}\, \tilde{\mathbf{u}}_k) \tag{32}$$
$$\tilde{\mathbf{w}}_{k+1} = (1+\mu)\, \tilde{\mathbf{w}}_k - \frac{\mu}{L}\, \mathbf{P}_\mathbf{w}\, \tilde{\mathbf{V}}_k^H\, \tilde{\mathbf{y}}_k\,. \tag{33}$$

Recall that we required $g(0) \equiv 0$. Therefore $\tilde{\mathbf{y}}_k$ will have the zero padded structure as given in (25). Since (30), (31), and (33) describe now circular convolutions, fast convolution techniques can now be employed. Towards this end, we define the following vectors of length $C = 4L$:

$$\bar{\mathbf{w}}_k \triangleq \mathbf{F}_C\, \tilde{\mathbf{w}}_k = \mathrm{FFT}(\tilde{\mathbf{w}}_k) \tag{34}$$
$$\bar{\mathbf{x}}_k \triangleq \mathbf{F}_C\, \tilde{\mathbf{x}}_k = \mathrm{FFT}(\tilde{\mathbf{x}}_k) \tag{35}$$
$$\bar{\mathbf{u}}_k \triangleq \mathbf{F}_C\, \tilde{\mathbf{u}}_k = \mathrm{FFT}(\tilde{\mathbf{u}}_k) \tag{36}$$
$$\bar{\mathbf{v}}_k \triangleq \mathbf{F}_C\, \tilde{\mathbf{v}}_k = \mathrm{FFT}(\tilde{\mathbf{v}}_k) \tag{37}$$
$$\bar{\mathbf{y}}_k \triangleq \mathbf{F}_C\, \tilde{\mathbf{y}}_k = \mathrm{FFT}(\tilde{\mathbf{y}}_k)\,. \tag{38}$$

Consequently we have from (34)

$$\tilde{\mathbf{w}}_k \triangleq \mathbf{F}^{-1} \bar{\mathbf{w}}_k = \mathrm{IFFT}(\bar{\mathbf{w}}_k)\,. \tag{39}$$

The same is also true for reversing (35) to (38).

We now wish to transform (30), (31), and (33) into the frequency domain. To this end, we premultiply the equations on both side with the Fourier matrix $\mathbf{F}$. We begin with (30)

$$\bar{\mathbf{u}}_k = \mathbf{F}\, \tilde{\mathbf{W}}_k\, \mathbf{F}^{-1}\, \mathbf{F}\, \tilde{\mathbf{x}}_k = \bar{\mathbf{W}}_k\, \bar{\mathbf{x}}_k \tag{40}$$

where $\bar{\mathbf{W}}_k \triangleq \mathbf{F}\, \tilde{\mathbf{W}}_k\, \mathbf{F}^{-1}$. Note that from (56) we know that $\bar{\mathbf{W}}_k = \mathrm{diag}(\mathbf{F}\, \tilde{\mathbf{w}}_k)$ is a diagonal matrix. Applying similar steps to (31) and using (57) we get

$$\bar{\mathbf{v}}_k = \mathbf{F}\, \tilde{\mathbf{W}}_k^H\, \mathbf{F}^{-1}\, \mathbf{F}\, \tilde{\mathbf{u}}_k = \bar{\mathbf{W}}_k^*\, \bar{\mathbf{u}}_k\,. \tag{41}$$

Transforming (32) into the frequency domain gives

$$\bar{\mathbf{y}}_k = \mathbf{F}\, g\left(\mathbf{P}_\mathbf{y}\, \mathbf{F}^{-1}\, \bar{\mathbf{u}}_k\right) \tag{42}$$

and applying property (57) to (33) gives

$$\bar{\mathbf{w}}_{k+1} = (1+\mu)\, \bar{\mathbf{w}}_k - \frac{\mu}{L}\, \mathbf{F}\, \mathbf{P}_\mathbf{w}\, \mathbf{F}^{-1}\, \mathbf{F}\, \tilde{\mathbf{V}}_k^H\, \mathbf{F}^{-1}\, \mathbf{F}\, \tilde{\mathbf{y}}_k \tag{43}$$
$$= (1+\mu)\, \bar{\mathbf{w}}_k - \frac{\mu}{L}\, \mathbf{F}\, \mathbf{P}_\mathbf{w}\, \mathbf{F}^{-1}\, \bar{\mathbf{V}}_k^*\, \bar{\mathbf{y}}_k\,. \tag{44}$$

Equations (40), (41), and (44) can be rewritten as

$$\bar{\mathbf{u}}_k = \bar{\mathbf{w}}_k \odot \bar{\mathbf{x}}_k \tag{45}$$
$$\bar{\mathbf{v}}_k = \bar{\mathbf{w}}_k^* \odot \bar{\mathbf{u}}_k \tag{46}$$
$$\bar{\mathbf{w}}_{k+1} = (1+\mu)\, \bar{\mathbf{w}}_k - \frac{\mu}{L} \mathbf{F}\, \mathbf{P}_\mathbf{w} \mathbf{F}^{-1}\, (\bar{\mathbf{v}}_k^* \odot \bar{\mathbf{y}}_k)\,. \tag{47}$$

Alternatively, since $\mathbf{P}_\mathbf{w}\, \tilde{\mathbf{w}}_k = \tilde{\mathbf{w}}_k$ holds, we can reformulate (47) as

$$\bar{\mathbf{w}}_{k+1} = \mathbf{F}\, \mathbf{P}_\mathbf{w}\, \mathbf{F}^{-1}\, \left((1+\mu)\, \bar{\mathbf{w}}_k - \frac{\mu}{L}\, \bar{\mathbf{v}}_k^* \odot \bar{\mathbf{y}}_k\right)\,. \tag{48}$$

We will refere to (45), (46), (42), and (48) as the FDBD-I algorithm The complete implementation of the filter and adaptation equations is summarized in Fig. 2 and the block diagram is shown in Fig. 3. We have preferred to use (48) over (47), which has the advantage that wrap-around errors do not accumulate in $\tilde{\mathbf{w}}_k$ if the projection operation $\mathbf{F}\, \mathbf{P}_\mathbf{w}\, \mathbf{F}^{-1}$ is not carried out in each block (alternated filter projections [23]).

In the derivation of the proposed algorithm we have restricted ourselves to the case with $N = L$ and $C = 4L$. However, different choices for N are possible as long as $C \geq L + 3(N-1)$.

FDBD-I

Definitions:

$$\mathbf{P_w} \triangleq \begin{bmatrix} \mathbf{I}_{L+1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0}_{3L-1} \end{bmatrix}$$

$$\mathbf{P_y} \triangleq \begin{bmatrix} \mathbf{0}_{2L} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_L & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0}_L \end{bmatrix}$$

Initialization:

$$\mathbf{w}_0 \triangleq (w_0[0], \ldots, w_L[0])^T$$
$$\tilde{\mathbf{w}}_0 \triangleq (\mathbf{w}_0^T, \mathbf{0}_{3L-1}^T)^T$$
$$\bar{\mathbf{w}}_0 \triangleq \mathrm{FFT}(\tilde{\mathbf{w}}_0)$$

For each loop k do:

Filtering:

$$\mathbf{x}_k \triangleq (x_{kL-L+1}, \ldots, x_{kL})^T$$
$$\tilde{\mathbf{x}}_k \triangleq (\mathbf{x}_{k-3}^T, \mathbf{x}_{k-2}^T, \mathbf{x}_{k-1}^T, \mathbf{x}_k^T)^T$$
$$\bar{\mathbf{x}}_k = \mathrm{FFT}(\tilde{\mathbf{x}}_k)$$
$$\bar{\mathbf{u}}_k = \bar{\mathbf{w}}_k \odot \bar{\mathbf{x}}_k$$
$$\tilde{\mathbf{u}}_k \triangleq (\dot{\mathbf{u}}_{k-3}^T, \mathbf{u}_{k-2}^T, \mathbf{u}_{k-1}^T, \mathbf{u}_k^T)^T = \mathrm{IFFT}(\bar{\mathbf{u}}_k)$$

Adaptation:

$$\bar{\mathbf{v}}_k = \bar{\mathbf{w}}_k^* \odot \bar{\mathbf{u}}_k$$
$$\tilde{\mathbf{y}}_k \triangleq (\mathbf{0}_L^T, \mathbf{0}_L^T, \mathbf{y}_{k-1}^T, \mathbf{0}_L^T)^T = g(\mathbf{P_y}\,\tilde{\mathbf{u}}_k)$$
$$\bar{\mathbf{y}}_k = \mathrm{FFT}(\tilde{\mathbf{y}}_k)$$
$$\bar{\mathbf{w}}_{k+1} = \mathrm{FFT}\left(\mathbf{P_w}\ \mathrm{IFFT}\left((1+\mu)\,\bar{\mathbf{w}}_k - \frac{\mu}{L}\,\bar{\mathbf{y}}_k \odot \bar{\mathbf{v}}_k^*\right)\right)$$

Figure 2: FDBD-I : Algorithm (FFT size $C = 4L$).

4 FDBD-I: 25% overlap

The FDBD-I algorithm as proposed has a 75% overlap between the samples of two subsequent input vectors $\tilde{\mathbf{x}}_{k-1}$ and $\tilde{\mathbf{x}}_k$. Therefore, an output block $\mathbf{u}_k$ is computed, in fact, three times, namely in block k, $k+1$, and $k+2$. On the other hand, the vector $\mathbf{y}_{k-1}$, which is used for updating the filter coefficients, is computed only once, namely in block k. Now there is also the possibility to operate the FDBD-I with only 25% overlap between two subsequent input vectors, i.e. (22) is replaced by

$$\tilde{\mathbf{x}}_k \triangleq (\mathbf{x}_{3k-3}^T, \mathbf{x}_{3k-2}^T, \mathbf{x}_{3k-1}^T, \mathbf{x}_{3k}^T)^T \tag{49}$$

such that (23) becomes

$$\tilde{\mathbf{u}}_k \triangleq (\dot{\mathbf{u}}_{3k-3}^T, \mathbf{u}_{3k-2}^T, \mathbf{u}_{3k-1}^T, \mathbf{u}_{3k}^T)^T. \tag{50}$$

The $3L$ output samples $(\mathbf{u}_{3k-2}^T, \mathbf{u}_{3k-1}^T, \mathbf{u}_{3k}^T)^T$ from $\tilde{\mathbf{u}}_k$ are now computed once only. Essentially, this is equivalent to computing the filtering and adaptation steps only at blocks $\{\ldots, k-3, k, k+3, \ldots\}$. More details about when which block is computed is given in Fig. 4.

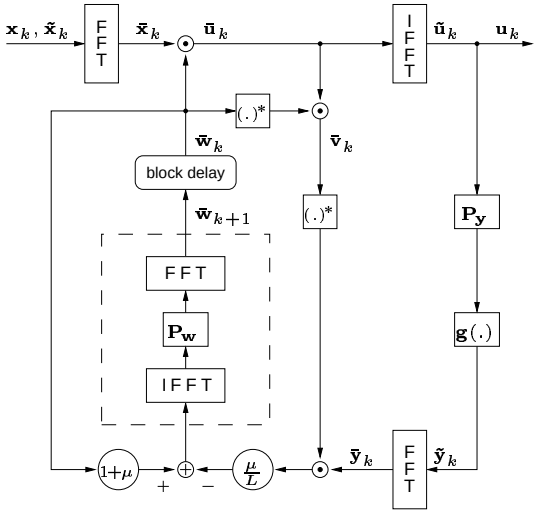

Figure 3: FDBD-I: Block diagram.

This simple trick reduces the computational complexity by a factor of three, as with the same number of operations we derive three times more output samples. One drawback is that only a third of the data is used now for the update, as $\mathbf{y}_{k-1}$ is not computed for every k. This might be a problem in the initial stage of the adaptation if fast convergence is required. Since the convergence rate depends primarily on the number of update iterations, the absolute convergence rate is about three times slower, which might still be fast enough to track a slowly time-varying system. Furthermore, the latency through the system, caused by processing delay, increases from $2L$ to $6L$ samples.

5 Complexity analysis

In this section we analyze and compare the computational complexity of the BTDBD, FDBD-DK, FDBD-I, and a FDBD-BG algorithm. The FDBD-BG is a frequency-domain implementation of a blind deconvolution based on a *Bussgang method*,

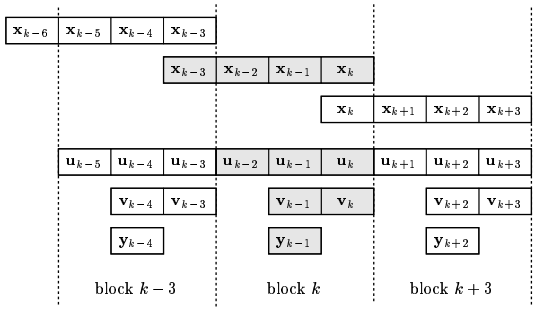

Figure 4: Signal computation of FDBD-I with 25% overlap between two subsequent input vectors $\tilde{\mathbf{x}}_{k-1}$ and $\tilde{\mathbf{x}}_k$.

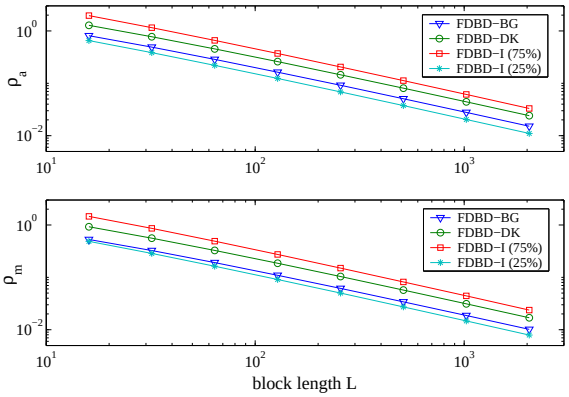

Figure 5: Relative computational complexity of the FDBD-BG, FDBD-DK, and FDBD-I algorithms compared to the BTDBD algorithm: Real additions (top) and real multiplications (bottom).

e.g., CMA. We assume having complex signals and also complex filter coefficients. We define the complexity as the number of real additions $\mathcal{A}_r$ and real multiplications $\mathcal{M}_r$. We do not encounter the complexity of the nonlinearity $g(.)$, as sometimes it can be implemented very efficient in hardware e.g. the $\mathrm{sign}(.)$-function for a super-Gaussian source signal. The computational complexity of different algorithms is given in Table 1. The complexity of one FFT or IFFT operation, $\mathcal{F}$, was counted as $\mathcal{F}=C\log C\mathcal{A}_c+C/2\log C\mathcal{M}_c=3C\log C\mathcal{A}_r+2C\log C\mathcal{M}_r$, where $\mathcal{A}_c$ and $\mathcal{M}_c$ denote one complex addition and multiplication, respectively.

To compare the computational complexities between the algorithms, we use the BTDBD as the reference. Then we calculate the ratios between the number of real additions and real multiplications to obtain L output samples, i.e.,

$$\rho_a = \frac{\mathcal{A}_r(\mathrm{FDBD})}{\mathcal{A}_r(\mathrm{BTDBD})} \tag{51}$$

$$\rho_m = \frac{\mathcal{M}_r(\mathrm{FDBD})}{\mathcal{M}_r(\mathrm{BTDBD})}\,. \tag{52}$$

The complexity reductions ρ_a and ρ_m are presented in Fig. 5 for different block sizes L. The complexity reduction for the FDBD-I with 25% overlap is by a factor 3 larger than for 75% overlap.

Table 1: Total computational complexity in real operations to obtain L output samples for each of the algorithms.

Algorithm	$\mathcal{A}_r$	$\mathcal{M}_r$
TDBD	$12L^2$ +8L	$16L^2$+16L
BTDBD	$12L^2$+8L	$12L^2$+12L+4N+4
FDBD-DK	48L log L + 64L	32L log L + 64L
FDBD-I (75%)	60L log L + 152L	40L log L + 144L
FDBD-I (25%)	20L log L + 152/3 L	40/3 L log L + 48L
FDBD-BG	30L log L + 42L	20L log L + 30L

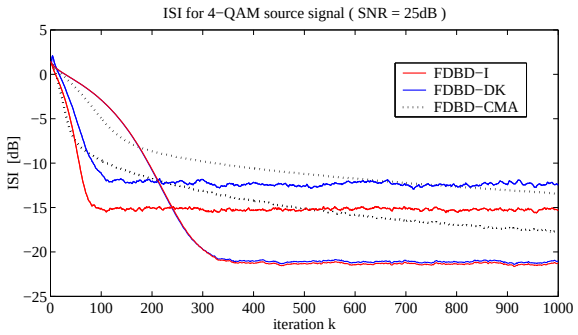

Figure 6: Convergence comparison between FDBD-I (75% overlap), FDBD-DK, and FDBD-CMA. The input SNR is 25 dB.

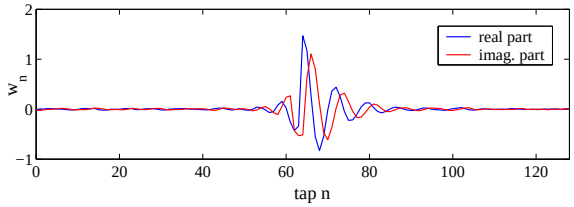

Figure 7: Real and imaginary part of w after convergence.

6 Simulation example

In the following, we give a simulation example to analyze the behavior of the algorithm proposed. The unknown non-minimum-phase channel is $a \propto \{1+4i, -1+5i, 11+2i, 17+11i, 1-20i\}$ and normalized to $\|a\|=1$. The source signal s is a 4-QAM signal, the input SNR is 25dB, the block size L is 128, and the nonlinearity is $g(u_t)=u_t\,|u_t|^2$. We compared the convergence behavior between the FDBD-I, the FDBD-DK and a FDBD-CMA algorithm. The FDBD-CMA is a frequency-domain implementation of the well-known *constant modulus algorithm*. The step sizes are $\mu=0.08$ and $\mu=0.02$ for the FDBD-I and FDBD-DK algorithm, and $\mu=0.2$ and $\mu=0.05$ for the FDBD-CMA algorithm. The larger step sizes are chosen to achieve a maximally fast initial convergence without becoming unstable in all trials. The performance criteria is the residual *intersymbol interference* (ISI), $\mathrm{ISI}(g) \triangleq (\sum_n |g_n|^2)/(\max_n |g_n|^2) - 1$, of the global system g. The curves shown in Fig. 6 are averages over 30 trials. The real and imaginary part of the deconvolution filter after convergence is shown in Fig. 7.

We observe that the FDBD-CMA has overall a slower convergence than FDBD-I and FDBD-DK algorithms. When comparing the FDBD-I with the FDBD-DK, we see that for small step sizes μ both have almost the same convergence behavior. However, for larger step sizes, the FDBD-DK becomes slightly slower than the FDBD-I algorithm and shows a higher residual ISI in the steady state. Other simulations with large step sizes have shown further, that the FDBD-I is less likely to become unstable than the FDBD-DK and FDBD-CMA algorithm for low SNR scenarios. Simulations with the FDBD-I and FDBD-DK algorithm for super-Gaussian source signals and $g(u_t)=\mathrm{sign}(u_t)$ have shown similar behaviors. Note, the constant modulus algorithm is unable to deconvolve the channel for super-Gaussian source signals.

7 Conclusions

We have presented a new way to implement the time-domain blind deconvolution algorithm from Amari *et al*. [16] efficiently in the frequency domain. The algorithm is based on the natural gradient learning method and, depending on the chosen nonlinearity, can handle either sub- or super-Gaussian source signals.

Furthermore, an analysis of the computational complexity has revealed that the Douglas-Kung algorithm [21] requires about 30% less operations than the proposed algorithm. This is because the Douglas-Kung algorithm makes the underlying assumption that the deconvolution filter changes only slowly from one block to the other, and therefore not all signal samples involved in the update are recomputed with the current filter. However, for large step sizes, where this assumption does not hold, the new algorithm showes a faster convergence and a more stable behavior. Possible applications of the algorithm are in acoustics, e.g. teleconferencing or hearing aids, as fast initial convergence and fast tracking is always of major importance.

References

[1] Y. Sato, "A method of self-recovering equalization for multilevel amplitude-modulation systems," *IEEE Trans. Computers*, pp. 679–682, June 1975.

[2] A. Benveniste, M. Goursat, and G. Ruget, "Robust identification of a nonminimum phase system: Blind adjustment of a linear equalizer in data communications," *IEEE Trans. Automat. Contr.*, vol. AC-25, no. 3, pp. 385–399, June 1980.

[3] D. N. Godard, "Self-recovering equalization and carrier tracking in two-dimensional data communication systems," *IEEE Trans. Commun.*, vol. COM-28, no. 11, pp. 1867–1875, Nov. 1980.

[4] J. R. Treichler and B. G. Agee, "A new approach to the multipath correction of constant modulus signals," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-31, no. 2, pp. 331–344, Apr. 1983.

[5] D. Donoho, "On minimum entropy deconvolution," *Applied Time Series Analysis II*, pp. 565–608, 1981.

[6] B. Widrow, J. McCool, and M. Ball, "The complex LMS algorithm," *Proc. IEEE*, pp. 719–720, Apr. 1975.

[7] E. R. Ferrara, "Fast implementations of LMS adaptive filters," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-28, no. 4, pp. 474–475, Aug. 1980.

[8] G. A. Clark, S. K. Mitra, and S. R. Parker, "Block implementation of adaptive digital filters," *IEEE Trans. Circuits Syst.*, vol. CAS-28, no. 6, pp. 584–592, June 1981.

[9] 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, no. 5, pp. 1073–83, Oct. 1983.

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

[11] C. K. Chan, M. R. Petraglia, and J. J. Shynk, "Frequency-domain implementations of the constant modulus algorithm," in *Proc. Asilomar Conf. Signals, Syst., Comput.*, Pacific Grove, CA, Oct. 30 – Nov. 1, 1989, vol. II, pp. 663–669.

[12] M. Ready, S. H. Goldberg, and R. Gooch, "Architecture considerations for frequency domain adaptive equalizers," in *Proc. Asilomar Conf. Signals, Syst., Comput.*, Pacific Grove, CA, Oct. 30 – Nov. 1, 1989, vol. II, pp. 663–669.

[13] J. J. Shynk, C. K. Chan, and M. R. Petraglia, "Blind adaptive filtering in the frequency domain," in *Proc. ISCAS*, New Orleans, LA, May 1–3, 1990, vol. I, pp. 275–278.

[14] J. J. Shynk, "Comparative performance study of several blind equalization algorithms," *Proc. SPIE*, vol. 1565, pp. 102–117, Apr. 3–4, 1991.

[15] J. Benesty and P. Duhamel, "Fast constant modulus adaptive algorithm," *IEE Proceedings-F*, vol. 138, no. 4, pp. 379–387, Aug. 1991.

[16] S.-I. Amari, S. C. Douglas, A. Cichocki, and H. H. Yang, "Novel on-line adaptive learning algorithms for blind deconvolution using the natural gradient approach," in *Proc. SYSID*, Kitakyushu, Japan, July 8–11, 1997, pp. 1057–1062.

[17] S.-I. Amari, A. Cichocki, and H. H. Yang, "A new learning algorithm for blind signal separation," *Advances in Neural Information Processing Systems*, vol. 8, pp. 757–763, 1996.

[18] S. C. Douglas, "On equivariant adaptation in blind deconvolution," in *Proc. Asilomar Conf. Signals, Syst., Comput.*, Pacific Grove, CA, Nov. 3–6, 2002.

[19] R. H. Lambert, *Multichannel Blind Deconvolution: FIR Matrix Algebra and Separation of Multipath Mixtures*, Ph.D. thesis, University of Southern California, 1996.

[20] R. H. Lambert and C. L. Nikias, "Blind deconvolution of multipath mixtures," in *Unsupervised Adaptive Filtering, Volume I: Blind Source Separation*, S. Haykin, Ed. 2000, pp. 377–436, John Wiley & Sons.

[21] S. C. Douglas and S.-Y. Kung, "Gradient adaptive algorithms for contrast-based blind deconvolution," *J. VLSI Signal Processing*, vol. 26, pp. 47–60, 2000.

[22] S. Amari and J. F. Cardoso, "Blind source separation—semiparametric statistical approach," *IEEE Trans. Signal Processing*, vol. 45, no. 11, pp. 2692–2700, 1997.

[23] M. Joho and G. S. Moschytz, "Connecting partitioned frequency-domain filters in parallel or in cascade," *IEEE Trans. Circuits Syst.–II*, vol. 47, no. 8, pp. 685–698, Aug. 2000.

[24] P. J. Davis, *Circulant Matrices*, John Wiley & Sons, 1979.

[25] R. M. Gray, *Toeplitz and Circulant Matrices: A review*, Stanford Electron. Lab., Tech. Rep. 6502-1, June 1971.

[26] M. Joho, *A Systematic Approach to Adaptive Algorithms for Multichannel System Identification, Inverse Modeling, and Blind Identification*, Ph.D. thesis, ETH Zürich, Dec. 2000.

A Circulant matrices and basic properties

Since there is a very close relationship between products of circulant matrices and circular convolutions, we recall some of the basic properties of circulant matrices. For a thorough description we refer to [24, 25] or [26, Chapter 3].

Let $\tilde{\mathbf{a}} = (a_1, \ldots, a_C)^T$. We define the corresponding circulant matrix $\tilde{\mathbf{A}}$ which has $\tilde{\mathbf{a}}$ as its first column as

$$\tilde{\mathbf{A}} \triangleq \mathcal{C}(\tilde{\mathbf{a}}) \triangleq \begin{bmatrix} a_1 & a_C & \ldots & a_2 \\ a_2 & a_1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & a_C \\ a_C & \ldots & a_2 & a_1 \end{bmatrix} . \tag{53}$$

The inverse operation $\tilde{\mathbf{a}} \triangleq \mathcal{C}^{-1}(\tilde{\mathbf{A}})$ returns the first column of $\tilde{\mathbf{A}}$. Furthermore, we define

$$\bar{\mathbf{a}} \triangleq \mathbf{F}\,\tilde{\mathbf{a}} \tag{54}$$

$$\bar{\mathbf{A}} \triangleq \mathbf{F}\,\tilde{\mathbf{A}}\,\mathbf{F}^{-1} . \tag{55}$$

Then (see [26])

$$\bar{\mathbf{A}} = \mathrm{diag}(\bar{\mathbf{a}}) \tag{56}$$

$$\mathbf{F}\,\tilde{\mathbf{A}}^H\,\mathbf{F}^{-1} = (\mathbf{F}\,\tilde{\mathbf{A}}\,\mathbf{F}^{-1})^H = \bar{\mathbf{A}}^H = \bar{\mathbf{A}}^* . \tag{57}$$

With (56) we see that the similarity transform (54) always diagonalizes any circulant matrix and therefore the eigenvalue decomposition (EVD) of a circulant matrix always has the form

$$\tilde{\mathbf{A}} = \mathbf{F}^{-1}\,\bar{\mathbf{A}}\,\mathbf{F} . \tag{58}$$