World Academy of Science, Engineering and Technology
International Journal of Electronics and Communication Engineering
Vol:1, No:12, 2007

A New Block-based NLMS Algorithm and Its Realization in Block Floating Point Format

Abhijit Mitra

***Abstract*—** We propose a new normalized LMS (NLMS) algorithm, which gives satisfactory performance in certain applications in comaprison with conventional NLMS recursion. This new algorithm can be treated as a block based simplification of NLMS algorithm with significantly reduced number of multiply and accumulate as well as division operations. It is also shown that such a recursion can be easily implemented in block floating point (BFP) arithmetic, treating the implementational issues much efficiently. In particular, the core challenges of a BFP realization to such adaptive filters are mainly considered in this regard. A global upper bound on the step size control parameter of the new algorithm due to BFP implementation is also proposed to prevent overflow in filtering as well as weight updating operations jointly.

***Keywords*—** Adaptive algorithm, Block floating point arithmetic, Implementation issues, Normalized least mean square methods.

I. Introduction

THE normalized least mean square (NLMS) algorithm can be considered as a special case of the least mean square (LMS) recursion [1] which takes into account the variation in the signal level at the filter output and selects a normalized step size parameter, resulting in a stable as well as fast converging adaptive algorithm. For fast convergence properties, NLMS algorithm has found many applications where primarily the statistics of the input processes are unknown or changing with time that include adaptive equalization, adaptive noise cancellation, adaptive line enhancing, adaptive array processing etc [2]. Depending upon the application, NLMS adaptation teachnique thus has been developed from different viewpoints. Originally, Goodwin and Sin [3] formulated the NLMS algorithm as a constrained optimization problem. Later, Nitzberg [4] obtained the recursion by running the conventional LMS algorithm many times, for each new input sample. However, in the most common form of a length L NLMS based adaptive filter, the weight update equation is given by

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu(n)\mathbf{x}(n)e(n) \tag{1}$$

where $\mathbf{w}(n) = [w_0(n), w_1(n), ..., w_{L-1}(n)]^T$ is the tap weight vector at the nth index, $\mathbf{x}(n) = [x(n), x(n-1), ..., x(n-L+1)]^T$ is the tap input vector and $e(n) = d(n) - \mathbf{w}^T(n)\mathbf{x}(n)$ is the error signal with $d(n)$ being the desired response available during the initial training period. The variable $\mu(n)$ denotes the so-called time varying step-size paramemter and is taken as

$$\mu(n) = \frac{\tilde{\mu}}{(\sigma_n^2 + a)}, \tag{2}$$

where σ_n^2 is the squared Euclidean norm $\mathbf{x}^T(n)\mathbf{x}(n)$, $\tilde{\mu}$ denoted a step size control parameter, used to control the speed of convergence and chosen according to: $0 < \tilde{\mu} < 2$ for convergence [2] and a is an appropriate positive number introduced to avoid divide-by-zero like situations which may arise when σ_n^2 becomes very small. It may be noted that the parameter σ_n^2 is updated time-recursively in each nth index as

$$\sigma_{n+1}^2 = x^2(n+1) + \sigma_n^2 - x^2(n-L+1). \tag{3}$$

The only price paid by such a fast algorithm is certain additional computations over its LMS counterpart for calculating the step size in each iteration through eq. (2) and (3). However, due to its superior convergence behavior as well as capability to mitigate the gradient noise amplification by deploying normalized step size, usually NLMS is preferred over LMS recursion. In a recent paper [5], an effective scheme is also proposed to implement the NLMS algorithm in block floating point (BFP) format to check its suitability on digital arithmetic. The block floating point (BFP) format is chosen as that is a viable alternative to the fixed point (FxP) and the floating point (FP) systems, and has been used successfully in recent years in a wide class of signal processing applications [6]-[10] including different types of fixed coefficient filters. Although in that approach, care is taken to prevent overflow both in filtering and weight updating operations, the scheme is computationally complex as it is based on NLMS recursion. Threfore, from computational as well as convergence viewpoint, a new algorithm would be preferred if that one is (i) computationally simpler vis-à-vis NLMS algorithm, (ii) almost as fast as its NLMS counterpart, and (iii) easily realizable in digital arithmetic.

In this paper, we introduce such a new algorithm which would perform satisfactorily in many application areas in comaprison with NLMS recursion. This new algorithm can be perceived as a block based simplification of NLMS algorithm with significantly reduced number of *Multiply and Accumulate* (MAC) and division operations. As the proposed algorithm is block based, it is also shown that such a recursion can be easily implemented in BFP arithmetic and it handles implementational issues much efficiently than the approach proposed in [5]. In particular, the core challenges of a BFP realization to such adaptive filters are mainly treated here that are not encountered in the fixed coefficient case, namely, (a) unlike a fixed coefficient filter, the filter coefficients in an adaptive filter *cannot be represented* in the simpler fixed point form, as the coefficients in effect evolve from the data by a time update relation, and (b) the two principal operations in an adaptive filter, namely, filtering and weight updating, *are mutually coupled*, thus requiring an appropriate arrangement for joint prevention of overflow. To solve the first challenge, the proposed approach adopts appropriate BFP formats for the data and the filter coefficients separately and to treat the second condition, the scheme makes

Manuscript received February 22, 2004.

The author is with the Department of Electronics and Communication Engineering, Indian Institute of Technology (IIT) Guwahati, North Guwahati - 781039, India. E-mail: a.mitra@iitg.ernet.in.

necessary adjustments in both the filtering as well as weight update equations so as to sustain the adopted format and also to prevent overlow jointly in both these operations. Particularly, a global upper bound of the step size control parameter for the proposed new algorithm has been derived to prevent overflow in the aforesaid operations.

II. The Proposed Block-based NLMS Algorithm

In the proposed algorithm, we first partition the input data into non-overlapping blocks of size equal to the filter length and find out the maximum magnitude within each block to consider only that particular value to update $\mu(n)$ for the entire block of data. Thus, the weight update equation of this new block-based NLMS (BB-NLMS) recursion takes the following form:

$$\mathbf{w}(n+1) = \begin{cases} \mathbf{w}(n) + \frac{\tilde{\mu}}{x^2_{M_i}}\mathbf{x}(n)e(n), & \text{for } x_{M_i} \neq 0 \\ \mathbf{w}(n), & \text{for } x_{M_i} = 0 \end{cases} \quad (4)$$

where $\mathbf{w}(n)$, $\mathbf{x}(n)$, $e(n)$, $d(n)$ and $\tilde{\mu}$ carry their usual meaning as has been described by eq. (1) and $x_{M_i} = max\{|x(n)| \mid n \in Z_i'\}$, $Z_i' = \{iN, iN+1, ..., iN+L-1\}$, $i \in Z$. In other words, x_{M_i} corresponds to the maximum magnitude of the input data samples $x(n)$ for the ith block, and therefore the condition given by the latter half of R.H.S. of (4), i.e., $\mathbf{w}(n+1) = \mathbf{w}(n)$, would arise if and only if all the data samples of any ith block are 0, which is alike the NLMS algorithm. However, the main advantage of the above simpler algorithm stems from employing $x^2_{M_i}$ in the denominator instead of $(\sigma_n^2 + a)$ as is usual in NLMS, and thereby saving certain MAC operations while implementing the same in any finite precision digital arithmetic. The speed of convergence of BB-NLMS recursion is also expected to be higher than that of NLMS for a given step size control parameter as it is not overly restrictive like NLMS recursion. The range of convergence parameter of this newly introduced algorithm is briefly investigated in the following.

A. Convergence characteristics of BB-NLMS

The range of step size control parameter $\tilde{\mu}$ for NLMS is basically derived from the following relation

$$0 < \mu tr(\mathbf{R}) < 2 \quad (5)$$

which is true for any algorithm in LMS family [1]. Here, $tr(\mathbf{R}) = E[\mathbf{x}(n)\mathbf{x}^T(n)]$. As BB-NLMS can be considered as a special form of NLMS adaptation, therefore, putting $\mu(n)$ in the above equation as $\frac{\tilde{\mu}}{x^2_{M_i}}$ in this case, we get

$$0 < \tilde{\mu} < \frac{2x^2_{M_i}}{tr(\mathbf{R})}. \quad (6)$$

Note that, considering $tr(\mathbf{R}) \approx LE[x^2(n)]$ and $x^2_{M_i} \geq E[x^2(n)]$, eq. (6) can be simplified as

$$0 < \tilde{\mu} < \frac{2}{L}. \quad (7)$$

However, as we would see later, due to ertain implementational constraints, the actual range of $\tilde{\mu}$ in BFP format becomes almost half of this.

III. Implementation of the Proposed BB-NLMS Algorithm in BFP Arithmetic

A. BFP arithmetic

The BFP representation can be considered as a special case of FP format, where every block of L incoming data has a joint scaling factor corresponding to the largest magnitude data sample in the block. In other words, given a block $[x_1, ..., x_L]$, we represent it as

$$[x_1, ..., x_L] = [\overline{x}_1, ..., \overline{x}_L].2^{\gamma} \quad (8)$$

where $\overline{x}_l (= x_l.2^{-\gamma})$ represents the mantissa for $l = 1, 2, ..., L$ and the block exponent γ is defined as

$$\gamma = \lfloor log_2 Max \rfloor + 1 + S \quad (9)$$

where $Max = max(|x_1|, ..., |x_L|)$, '$\lfloor . \rfloor$' is the so-called floor function, meaning rounding down to the closest integer and the integer S is a scaling factor which is needed to prevent overflow during filtering operation. For the presence of S, the range of each mantissa is given as $|\overline{x}_l| \in [0, 2^{-S})$. The scaling factor S can be calculated from the inner product computation representing filtering operation. The filter output $y(n)$ at nth index, which is an inner product, is calculated in BFP format as

$$\begin{aligned} y(n) &= \langle \mathbf{w}, \mathbf{x}(n) \rangle \\ &= \mathbf{w}^T \mathbf{x}(n) \\ &= [w_0 \overline{x}(n) + ... + w_{L-1}\overline{x}(n-L+1)].2^{\gamma} \\ &= \overline{y}(n).2^{\gamma} \end{aligned} \quad (10)$$

where $\mathbf{w}$ is a length L fixed point filter coefficient vector and $\mathbf{x}(n)$ is the data vector at the nth index, represented in BFP format. For no overflow in $y(n)$, we need $|\overline{y}(n)| \leq 1$ at every time index, which can be satisfied by selecting [7]

$$S \geq S_{min} = \lceil log_2 (\sum_{k=0}^{L-1} |w_k|) \rceil \quad (11)$$

where '$\lceil . \rceil$' is the so-called ceiling function, meaning rounding up to the closest integer.

B. The proposed implementation

The proposed scheme consists of two simultaneous BFP representations, one for the filter coefficient vector $\mathbf{w}(n)$ and the other for the given data, namely, $x(n)$ and $d(n)$. These are as follows.

(a) **BFP representation of the filter coefficient vector**:
At each index of time, here we have a scaled representation of the filter coefficient vector as

$$\mathbf{w}(n) = \overline{\mathbf{w}}(n).2^{\psi_n} \quad (12)$$

where ψ_n is a time-varying block exponent that needs to be updated at each nth instant and is chosen to ensure that each $|\overline{w}_k(n)| < \frac{1}{2}$ for $k = 0, 1, ..., L-1$. If a data vector $\mathbf{x}(n)$ is given in the aforesaid BFP format as $\mathbf{x}(n) = \overline{\mathbf{x}}(n).2^{\gamma}$, where $\gamma = ex + S$, $ex = \lfloor log_2 M \rfloor + 1$, $M = max(|x(n-k)| \mid$

$k = 0, 1, ..., L-1$) and S is an appropriate scaling factor, then, the filter output $y(n)$ can be expressed as $y(n) = \overline{y}(n).2^{\gamma+\psi_n}$ with $\overline{y}(n) = \overline{\mathbf{w}}^T(n)\overline{\mathbf{x}}(n)$ denoting the output mantissa. To prevent overflow in $\overline{y}(n)$, the required condition is $|\overline{y}(n)| < 1$. However, in the proposed scheme, we confine $\overline{y}(n)$ to lie between $+\frac{1}{2}$ and $-\frac{1}{2}$, i.e., $|\overline{y}(n)| < \frac{1}{2}$ for reasons explained later. Since $|\overline{y}(n)| \leq \sum_{k=0}^{L-1} |\overline{w}_k(n)||\overline{x}(n-k)|$, $0 \leq |\overline{x}(n-k)| < 2^{-S}$ and $|\overline{w}_k(n)| < \frac{1}{2}$, this implies a lower limit of S as $S_{min} = \lceil log_2 L \rceil$.

(b) **BFP representation of the given data**:

The input data $x(n)$ and the desired response sequence $d(n)$ are first partitioned jointly into non-overlapping blocks of L samples each with the ith block ($i \in Z$) consisting of $x(n)$, $d(n)$ for $n \in Z_i' = \{iN, iN+1, ..., iN+L-1\}$. Further, both $x(n)$ and $d(n)$ are jointly scaled so as to have a common BFP representation within each block. This means that, for $n \in Z_i'$, $x(n)$ and $d(n)$ are expressed as

$$x(n) = \overline{x}(n).2^{\gamma_i}, \quad d(n) = \overline{d}(n).2^{\gamma_i} \tag{13}$$

where γ_i is the common block exponent for the ith block and is given as $\gamma_i = ex_i + S_i$ where $ex_i = \lfloor log_2 M_i \rfloor + 1$ and $M_i = max\{|x(n)|, |d(n)| \mid n \in Z_i'\}$. The block dependent scaling factor S_i plays a key role in terms of preventing overflow and is assigned carefully as per the following algorithm:

Algorithm: *Assign $S_{min} = \lceil log_2 L \rceil$ as the scaling factor to the first block and for any (i-1)-th block, assume $S_{i-1} \geq S_{min}$.*
Then, if $ex_i \geq ex_{i-1}$, choose $S_i = S_{min}$ (i.e., $\gamma_i = ex_i + S_{min}$)
else (i.e., $ex_i < ex_{i-1}$)
choose $S_i = (ex_{i-1} - ex_i + S_{min})$, s.t. $\gamma_i = ex_{i-1} + S_{min}$.

Note that when $ex_i \geq ex_{i-1}$, we can either have $ex_i + S_{min} \geq \gamma_{i-1}$ (Case A) implying $\gamma_i \geq \gamma_{i-1}$, or, $ex_i + S_{min} < \gamma_{i-1}$ (Case B) meaning $\gamma_i < \gamma_{i-1}$. However, for $ex_i < ex_{i-1}$ (Case C), we always have $\gamma_i \leq \gamma_{i-1}$.

The output error $e(n)$ is then evaluated as $e(n) = \overline{e}(n).2^{\gamma_i+\psi_n}$ where the mantissa $\overline{e}(n)$ is given as

$$\overline{e}(n) = \overline{d}(n).2^{-\psi_n} - \overline{y}(n). \tag{14}$$

Clearly, computation of $\overline{e}(n)$ involves an additional step of right-shift operation (frequently used in FP arithmetic) on $\overline{d}(n)$. However, since in an adaptive filter, filter coefficients are derived from data and thus can not be represented in the FxP format when data is given in a scaled form, such a step seems to be unavoidable. It is then easy to check that $|\overline{e}(n)| < 1$, since,

$$\begin{aligned} |\overline{e}(n)| &\leq |\overline{d}(n)|.2^{-\psi_n} + |\overline{y}(n)| \\ &< 2^{-(S_i+\psi_n)} + \frac{1}{2} \leq 2^{-S_i}(2^{-\psi_n} + \frac{L}{2}) \end{aligned} \tag{15}$$

as $2^{-S_i} \leq \frac{1}{L}$. Except for the case: $\psi_n = 0$ and $L = 1$, the R.H.S.≤ 1. For the above description of $e(n)$, $\mathbf{x}(n)$, $d(n)$ and $\mathbf{w}(n)$, the weight update equation (4) takes the form

$$\mathbf{w}(n+1) = \overline{\mathbf{u}}(n).2^{\psi_n} \tag{16}$$

where

$$\overline{\mathbf{u}}(n) = \overline{\mathbf{w}}(n) + \frac{\tilde{\mu}}{\overline{x}_{M_i}^2}\overline{\mathbf{x}}(n)\overline{e}(n), \tag{17}$$

with $\overline{x}_{M_i}^2 = x_{M_i}^2.2^{-2\gamma_i}$. To satisfy $|\overline{w}_k(n+1)| < \frac{1}{2}$ for $k = 0, 1, \cdots, L-1$, we first limit each $|\overline{u}_k(n)| < 1$, $k = 0, 1, \cdots, L-1$ with $\overline{u}_k(n)$ denoting the kth component of $\overline{\mathbf{u}}(n)$. Then, if each $\overline{u}_k(n)$ happens to be lying within $\pm\frac{1}{2}$, we make the assignments:

$$\overline{\mathbf{w}}(n+1) = \overline{\mathbf{u}}(n), \quad \psi_{n+1} = \psi_n. \tag{18}$$

Otherwise, we scale down $\overline{\mathbf{u}}(n)$ by 2, in which case,

$$\overline{\mathbf{w}}(n+1) = \frac{1}{2}\overline{\mathbf{u}}(n), \quad \psi_{n+1} = \psi_n + 1. \tag{19}$$

Since each $|\overline{w}_k(n)| < \frac{1}{2}$ for $k = 0, 1, \cdots, L-1$, it is sufficient to have

$$\frac{\tilde{\mu}}{\overline{x}_{M_i}^2}|\overline{x}(n-k)||\overline{e}(n)| < \frac{1}{2} \tag{20}$$

in order to satisfy the relation $|\overline{u}_k(n)| < 1$ for $k = 0, 1, \cdots, L-1$. The above equation thus gives a provision towards having an upper bound of $\tilde{\mu}$ in order to prevent overflow in filtering as well as weight updating operations.

C. A global upper bound on $\tilde{\mu}$

To find out any global upper bound, i.e., to ensure any variable of the form $\frac{|p|}{|q|}$ always lying within a certain limit B (where both p, q are variables and $p \neq \infty$, $q \neq 0$), it is necessary to meet the requirement $\frac{|p|_{max}}{|q|_{min}} \leq B$ in order to have a *worst-case* upper bound so that it works in all the situations. Therefore, for eq. (20), we need to find out a minimum value of the quantity $\overline{x}_{M_i}^2$, which comes as: $\frac{1}{4}.2^{-2S_i}$. The lower limit is obtained with the knowledge that at least the maximum magnitude data sample within the ith block in BFP format will be block normalized and therefore would be specified by the range $\frac{1}{2}.2^{-S_i} \leq |\overline{x}_{M_i}| < 2^{-S_i}$. In eq. (20), putting the above lower limit of $\overline{x}_{M_i}^2$ and the usual upper limit of $|\overline{e}(n)|$ as $2^{-S_i}(2^{-\psi_n} + \frac{L}{2})$ respectively, we get the following global upper bound on $\tilde{\mu}$:

$$\tilde{\mu} \leq \frac{1}{2(L+2)} \tag{21}$$

since $\psi_n \geq 0$ for all n. Note that in deriving the above upper bound, the value for $|\overline{x}(n-k)|$ used is $\frac{1}{2}2^{-S_i}$ as it is already taken into consideration that $|\overline{x}_{M_i}| \geq |\overline{x}(n-k)|$ for $n \in Z_i'$, $k = 0, 1, \cdots, L-1$. It is also interesting to note that the above global upper bound is somewhat more restrictive than the upper bound given in eq. (7) for algorithm convergence. The main reason for this is the difference between algorithmic and implementational point of view as the latter has to take care of no overflow in all situations and thus has to provide a worst-case upper bound, which, in this case, is given by eq. (21).

D. Complexity issues

The BFP implementation of the BB-NLMS algorithm, as proposed above and summarized in Table 1, relies mostly on FxP arithmetic and thus enjoys less processing time than its FP-based counterpart. It is also interesting to note that the BFP implementation of BB-NLMS recursion is faster than the BFP realization of conventional NLMS algorithm as the former

TABLE I

Summary of the BB-NLMS algorithm realized in BFP format (initial value of $\psi_n = 0$).

1. **Preprocessing:**
   Using the data for the i-th block, $x(n)$ and $d(n)$, $n \in Z_i'$ (stored during the processing of the $(i-1)$-th block),
(a) Evaluate block exponent γ_i as per the **Algorithm** of Section III and express $x(n)$, $d(n)$, $n \in Z_i'$ as $x(n) = \overline{x}(n).2^{\gamma_i}$, $d(n) = \overline{d}(n).2^{\gamma_i}$,
(b) Evaluate $\overline{x}_{M_i} = max\{|\overline{x}(n)| \mid n \in Z_i', i \in Z\}$.
2. **Processing for the i-th block:**
   For $n \in Z_i' = \{iN, iN+1, ..., iN+L-1\}$
(a) Filter ouput:
   $\overline{y}(n) = \overline{\mathbf{w}}^T(n)\overline{\mathbf{x}}(n)$,
   $\lambda(n) = \gamma_i + \psi_n$.
   ($\lambda(n)$ is the filter output exponent)
(b) Output error (mantissa) computation:
   $\overline{e}(n) = \overline{d}(n).2^{-\psi_n} - \overline{y}(n)$.
(c) Filter weight updating:
   $\overline{\mathbf{u}}(n) = \overline{\mathbf{w}}(n) + \frac{\tilde{\mu}}{\overline{x}^2_{M_i}}\overline{\mathbf{x}}(n)\overline{e}(n)$.
   If $|\overline{u}_k(n)| < \frac{1}{2}$ for all $k \in Z_L = \{0, 1, ..., L-1\}$
   then
   $\overline{\mathbf{w}}(n+1) = \overline{\mathbf{u}}(n)$,
   $\psi_{n+1} = \psi_n$,
   else
   $\overline{\mathbf{w}}(n+1) = \frac{1}{2}\overline{\mathbf{u}}(n)$,
   $\psi_{n+1} = \psi_n + 1$.
   end.
   $i \leftarrow i+1$.
   Go back to step 1.

doesn't require to update the step size parameter via eq. (2) and (3). For example, to calculate the step size mantissa $\frac{\tilde{\mu}}{\overline{x}^2_{M_i}}$ of BB-NLMS in BFP format, we need to employ 1 MAC operation in simple FxP format, and 1 division operation followed by 1 normalization, *only once per block of L samples* [Add to this a total of $(L-1)$ comparisons to check for x_{M_i}.]. This, in turn, saves a total of $(2L-1)$ MACs, $(L-1)$ divisions and L^2 additions over the BFP realization of conventional NLMS algorithm [5] for L number of iterations. The other computational accounts in this case, i.e., computations for filter output, output error calculation and weight updating, remain same in comparison with BFP implementation of conventional NLMS algorithm [5]. A summary of these operations have also been provided in Table 1. Table 2 provides a comparative account of the BFP-based realizations of BB-NLMS and NLMS algorithms in terms of number of MAC operations only, required per L iterations. It is clearly seen from the above computational accounts that though the BFP implementation of BB-NLMS algorithm reqires few comparison operations, but it mostly takes less computational figures in comparison with BFP realization of NLMS recursion, and therefore, is a better choice for high speed applications.

TABLE II

A comparison between the BFP-based realizations of the new BB-NLMS algorithm and the conventional NLMS algorithm in terms of only MAC operations required per L iterations for (a) calculating step-size mantissa, (b) weight updating, and (c) filtering ($y(n) = \mathbf{w}^T(n)\mathbf{x}(n)$).

	Operation (BB-NLMS)	No. of MAC	Operation (NLMS)	No. of MAC
(a)	$\frac{\tilde{\mu}}{\overline{x}^2_{M_i}}$	1	$\frac{\tilde{\mu}}{\overline{\sigma}^2_n + \overline{a}_i}$	$2L$
(b)	$\mathbf{w}(n+1)$	$L(L+1)$	$\mathbf{w}(n+1)$	$L(L+1)$
(c)	$y(n)$	L^2	$y(n)$	L^2

IV. Simulation Results and Discussions

The proposed algorithm has been simulated under finite precision like environment using MATLAB, in the context of equalization of channel effects and identification of several unknown systems. We show one such system identification problem below. A system output signal $y(n)$ was generated as: $y(n) = 0.7x(n) + 0.6x(n-1) + 0.2x(n-2) + \nu(n)$, with $x(n)$ and $\nu(n)$ representing the system input and the zero mean observation white noise respectively with the following variances: $\sigma_x^2 = 1$ and $\sigma_\nu^2 = 0.02$. Putting $L = 3$ in eq. (21), we get the global upper bound $\tilde{\mu} = 0.1$. With this upper bound, the new BB-NLMS algorithm was simulated in finite precision BFP format, choosing a block length of 3 and allocating 12 bits for the signed mantissa and 4 bits for the signed exponent for both the input data and the filter coefficients. Using the same finite precision set up as above and with the same value of $\tilde{\mu}$, the NLMS algorithm was simulated next in BFP format. The corresponding MSE characteristics for these two simulations are shown in Fig. 1(a) and Fig. 1(b) respectively. Further, the BB-NLMS and the original NLMS algorithms were simulated in infinite precision, using $\tilde{\mu} = 0.65$ for both the cases, thereby taking $\tilde{\mu}$ almost equal to its conventional upper bound for convergence ($\frac{2}{L} = \frac{2}{3} \approx 0.65$) in case of the BB-NLMS algorithm. We show the corresponding learning curves in Fig. 2(a) and Fig. 2(b) respectively. In both the infinite precision and finite precision cases, it is easily seen from the figures that the proposed BB-NLMS algorithm has the convergence speed almost equal, and sometimes better, to its respective NLMS counterpart. This new computationally efficient algorithm, therefore, can serve as a good alternative of the NLMS algorithm for block implementations in high speed application areas.

V. Conclusion

This paper has introduced an effective adaptive algorithm, BB-NLMS, and also presented a scheme for implementing the same in a BFP format. The main computations involved, as given by equations (14), (17), (18) and (19), are based on fixed point arithmetic only and thus can be realized using digital hardware that is simple, cheap and fast, with the advantage of floating point like wide dynamic range by means of a block exponent. The introduced algorithm provides high convergence speed with significantly reduced number of MAC as well as

division operations, in comparison with conventional NLMS recursion. The only price paid for this has been a reduced range for the step-size control parameter governed by a new upper limit, which is inversely proportional to the filter length. This, in turn, limits the application of this algorithm and its BFP implementation in adaptive noise canceling where the filter length may be very high. However, for high speed applications with relatively low filter length (e.g., in North American Digital Cellular standard IS-54, where the system coefficients should converge in the training mode roughly within first 50 training samples with high speed [11]), the BB-NLMS may be a viable alternative to conventional NLMS algorithm. Implementing such a less computationally complex algorithm on a real time processor can also be treated as an excellent future extension of the work to develop a general framework for different application areas as well as to study the optimal issues of realization. Efforts are now underway to carry out the same on a 16-bit TMS320C54X fixed point processor, with an investigation towards understanding the numerical error effects on the proposed implementation.

REFERENCES

[1] S. Haykin, *Adaptive Filter Theory*, 4th ed. Englewood Cliffs, NJ: Prentice-Hall, 2001.
[2] N. J. Bershad, "Analysis of the Normalized LMS Algorithm with Gaussian Inputs," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-34, no. 4, pp. 793-806, April 1986.
[3] G. C. Goodwin and K. S. Sin, *Adaptive Filtering, Prediction and Control*. Englewood Cliffs, NJ: Prentice-Hall, 1984.
[4] R. Nitzberg, "Application of the Normalized LMS Algorithm to MSLC," *IEEE Trans. Aerospace and Electronic Syst.*, vol. AES-21, no. 1, pp. 79-91, Jan. 1985.
[5] A. Mitra and M. Chakraborty, "The NLMS Algorithm in Block-Floating-Point Format," *IEEE Signal Processing Letters*, vol. 11, no. 3, pp. 301-304, March 2004.
[6] K. R. Ralev and P. H. Bauer, "Realization of Block Floating Point Digital Filters and Application to Block Implementations," *IEEE Trans. Signal Processing*, vol. 47, no. 4, pp. 1076-1086, April 1999.
[7] K. Kalliojarvi and J. Astola, "Roundoff Errors in Block-Floating-Point Systems," *IEEE Trans. Signal Processing*, vol. 44, no. 4, pp. 783-790, April 1996.
[8] A. Erickson and B. Fagin, "Calculating FHT in Hardware," *IEEE Trans. Signal Processing*, vol. 40, pp. 1341-1353, June 1992.
[9] S. Sridharan and D. Williamson, "Implementation of high order direct form digital filter structures," *IEEE Trans. Circuits Syst.*, vol. CAS-33, pp. 818-822, Aug. 1986.
[10] F. J. Taylor, "Block Floating Point Distributed Filters," *IEEE Trans. Circuits Syst.*, vol. CAS-31, pp. 300-304, Mar. 1984.
[11] R. D. Koilpillai, S. Chennakeshu and R. L. Toy, "Low Complexity Equalizers for U.S. Digital Cellular System," in *Proc. IEEE VTC*, May 1992. pp. 744-747.

Abhijit Mitra was born in Serampore, India, in 1975. He received the B.E.(Honors) degree from R. E. College, Durgapur, India, in 1997, M.E.Tel.E. degree from Jadavpur University, India, in 1999 and Ph.D. degree from Indian Institute of Technology, Kharagpur, India, in 2004, all in electronics and communication engineering. Since 2004, he is with Indian Institute of Technology, Guwahati, India, as an Assistant Professor. His research interests include finite wordlength digital signal processing, statistical signal processing, adaptive signal processing and wireless communications.
Dr. Mitra is a member of IEEE and also serves as a reviewer of IEEE Transactions on Signal Processing.

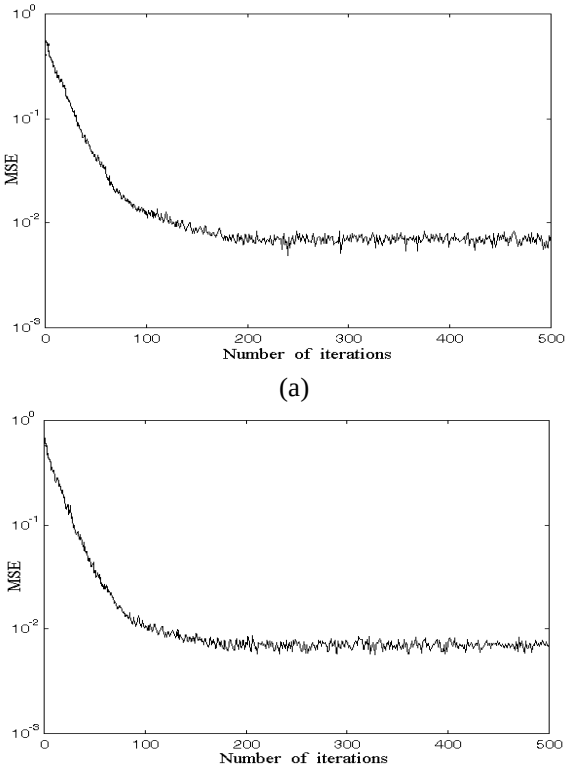

Fig. 1. MSE characteristics of the (a) BFP-based BB-NLMS algorithm and (b) BFP-based NLMS algorithm, both in finite precision with $\tilde{\mu} = 0.1$.

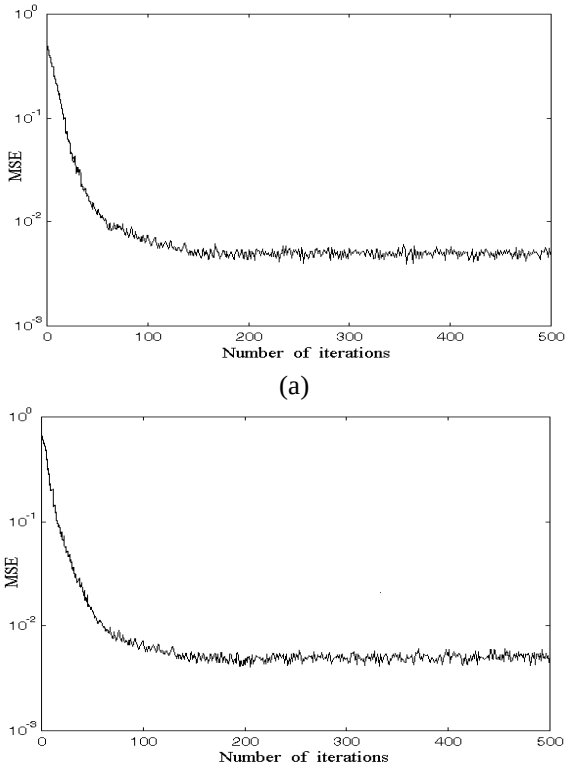

Fig. 2. MSE characteristics of the (a) BB-NLMS algorithm and (b) conventional NLMS algorithm, both in infinite precision with $\tilde{\mu} = 0.65$.