# The NLMS Algorithm in Block Floating-Point Format

Abhijit Mitra, *Student Member, IEEE,* and Mrityunjoy Chakraborty, *Senior Member, IEEE*

***Abstract*—We present a novel scheme to implement the normalized least mean square algorithm in block floating-point (BFP) format, which permits processing of data over a wide dynamic range, at a cost significantly less than that of a floating-point processor. Appropriate BFP formats for both the data and the filter coefficients are adopted. Care is taken so that the chosen formats remain invariant to interblock transition and weight-updating operation, respectively. Care is also taken to prevent overflow during filtering, as well as weight-updating processes, by using a dynamic scaling of the data and a slightly reduced range for the step size control parameter, with the latter having negligible effect on convergence speed.**

***Index Terms*—Adaptive filters, block floating-point (BFP) arithmetic, normalized least mean square (NLMS) methods.**

## I. Introduction

THE BLOCK floating-point (BFP) format [1] is a viable alternative to the floating-point (FP) concept that has found wide usage in recent years for efficient realization of several signal processing algorithms [2]–[10], motivated primarily by demands from high-throughput and/or low-power applications. In this format, the incoming data are partitioned into nonoverlapping blocks, and depending upon the data sample with highest magnitude in each block, a common exponent is assigned for the block. This permits an overall FP-like representation of the data with fixed-point (FXP)-like computation within every block, thereby enabling the user to handle data over a wide dynamic range with temporal and/or spatial complexities comparable to that of FXP-based systems.

In the context of digital filters, the BFP method has been successfully used for efficient implementation of fixed coefficient filters [2], [3], [7]–[10]. Some studies [3]–[5] have also been made to investigate the associated numerical error behavior. However, to the best of our knowledge, no effort has so far been made to extend this treatment to adaptive filters that present more complex structures including error feedback. A BFP treatment to adaptive filters faces certain difficulties, not encountered in the fixed coefficient case, namely as follows.

- 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.
- The two principal operations in an adaptive filter—filtering and weight updating—are mutually coupled, thus requiring an appropriate arrangement for joint prevention of overflow.

In this letter, we present a novel scheme for BFP-based realization of the normalized least mean square (NLMS) algorithm [11], which enjoys superior convergence behavior over the LMS algorithm at the expense of certain additional computation. The proposed approach adopts appropriate BFP formats for the data and the filter coefficients separately, taking care so that: 1) the chosen format for the filter coefficients remains invariant to weight adjustment, and 2) a uniform BFP representation of the filter input vector is available during block-to-block transition phase when the data come from two adjacent blocks with two different exponents. Care is also taken to prevent overflow during filtering and weight updating processes by using a dynamic scaling of the data and a slightly reduced range for the step size control parameter that has negligible effect on convergence speed.

## II. BFP Arithmetic

As stated earlier, in BFP format, given a block $[x_1, \ldots, x_N]$, we represent it as $[x_1, \ldots, x_N] = [\overline{x}_1, \ldots, \overline{x}_N] \cdot 2^{\gamma}$ where $\overline{x}_k (= x_k \cdot 2^{-\gamma})$ represents the mantissa for $k = 1, 2, \ldots, N$ and the block exponent $\gamma$ is defined as $\gamma = \lfloor \log_2 \mathrm{Max} \rfloor + 1 + S$, where $\mathrm{Max} = \max(|x_1|, \ldots, |x_N|)$; $\lfloor . \rfloor$ is the so-called floor function, meaning rounding down to the closest integer, and the integer $S$ is a scaling factor needed to prevent overflow during filtering operation. Due to the presence of $S$, the range of each mantissa is given as $|\overline{x}_k| \in [0, 2^{-S})$. The scaling factor $S$ can be calculated from the inner product computation representing the filtering operation. An inner product is calculated in BFP arithmetic as

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

where $\mathbf{w}$ is a length $L$ fixed-point filter coefficient vector and $\mathbf{x}(n)$ is the data vector at the $n$th index, represented in BFP format. For no overflow in $y(n)$, we need $|\overline{y}(n)| < 1$ at every time index, which can be satisfied [3] by selecting $S \geq S_{\min} = \lceil \log_2 (\sum_{k=0}^{L-1} |w_k|) \rceil$ where $\lceil . \rceil$ is the so-called ceiling function, meaning rounding up to the closest integer.

Manuscript received May 6, 2003; revised June 6, 2003. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Richard C. Kavanagh.

The authors are with the Department of Electronics and Electrical Communication Engineering, Indian Institute of Technology (IIT), Kharagpur 721 302, India (e-mail: abhijit@ece.iitkgp.ernet.in; mrityun@ece.iitkgp.ernet.in).

Digital Object Identifier 10.1109/LSP.2003.822891

## III. Proposed Implementation

Consider a length $L$ NLMS-based adaptive filter that takes an input sequence $x(n)$ and updates the weights [11] as

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu(n)\mathbf{x}(n)e(n) \tag{2}$$

where $\mathbf{w}(n) = [w_0(n)w_1(n), \ldots, w_{L-1}(n)]^T$ is the tap weight vector at the $n$th index, $\mathbf{x}(n) = [x(n)x(n-1), \ldots, x(n-L+1)]^T$, $e(n) = d(n) - \mathbf{w}^T(n)\mathbf{x}(n)$ is the error signal, with $d(n)$ being the desired response available during the initial training period and $\mu(n)$ denoting the so-called time varying step size paramemter. In the most commonly used form [11] of the NLMS algorithm, $\mu(n)$ is taken as $\mu(n) = \tilde{\mu}/(\sigma_n^2 + \theta)$, where $\sigma_n^2 = \mathbf{x}^T(n)\mathbf{x}(n)$ , $\tilde{\mu}$ is a step size control parameter, used to control the speed of convergence and chosen according to $0 < \tilde{\mu} < 2$ for convergence and $\theta$, an appropriate positive number introduced to avoid divide-by-zero like situations that may arise when $\sigma_n^2$ becomes very small. It may be noted that the original NLMS algorithm is a special case of the above, corresponding to $\tilde{\mu} = 1$ and $\theta = 0$.

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

a) *BFP representation of the filter coefficient vector:* In this, at each index of time, we have a scaled representation of the filter coefficient vector as $\mathbf{w}(n) = \overline{\mathbf{w}}(n) \cdot 2^{\psi_n}$ where $\psi_n$ is a time-varying block exponent that needs to be updated at each index $n$ and is chosen to ensure that each $|\overline{w}_k(n)| < 1/2$ for $k \in Z_L = \{0, 1, \ldots, L-1\}$. If a data vector $\mathbf{x}(n)$ is given in the aforesaid BFP format as $\mathbf{x}(n) = \overline{\mathbf{x}}(n) \cdot 2^{\gamma}$, where $\gamma = ex + S$, $ex = \lfloor \log_2 M \rfloor + 1$, $M = \max(|x(n-k)| \mid k \in Z_L)$ and $S$ is an appropriate scaling factor, then, the filter output $y(n)$ can be expressed as $y(n) = \overline{y}(n) \cdot 2^{\gamma+\psi_n}$ with $\overline{y}(n) = \overline{\mathbf{w}}^T(n)\overline{\mathbf{x}}(n)$ denoting the output mantissa. To prevent overflow in $\overline{y}(n)$, it is required that $|\overline{y}(n)| < 1$. However, in the proposed scheme, we restrict $\overline{y}(n)$ to lie between $+1/2$ and $-1/2$, i.e., $|\overline{y}(n)| < 1/2$. 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 $|\overline{w}_k(n)| < 1/2$, this implies a lower limit of $S$ as $S_{\min} = \lceil \log_2 L \rceil$. The two conditions $|\overline{w}_k(n)| < 1/2$, $k \in Z_L$ and $|\overline{y}(n)| < 1/2$ ensure no overflow during updating of $\overline{\mathbf{w}}(n)$ and computation of output error mantissa, respectively, as shown later.

b) *BFP representation of the given data:* The input data $x(n)$ and the desired response sequence $d(n)$ are partitioned jointly into nonoverlapping blocks of $N$ samples each with the $i$th block ($i \in Z$) consisting of $x(n)$, $d(n)$ for $n \in Z_i' = \{iN, iN+1, \ldots, iN+N-1\}$. 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 $x(n) = \overline{x}(n) \cdot 2^{\gamma_i}$, $d(n) = \overline{d}(n) \cdot 2^{\gamma_i}$, where $\gamma_i$ is the common block exponent for the $i$th 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 scaling factor $S_i$ is assigned as per the following algorithm.

```
Algorithm
Assign S_min = ⌈log2 L⌉ as the scaling factor
to the first block and for any (i − 1)th block,
assume S_{i−1} ≥ S_min .
Then, if ex_i ≥ ex_{i−1}, choose S_i = S_min (i.e.,
γ_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. γ_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 $\gamma_i \leq \gamma_{i-1}$.

Additionally, we rescale the elements $\overline{x}(iN - L + 1), \ldots, \overline{x}(iN-1)$ by dividing by $2^{\Delta\gamma_i}$, where $\Delta\gamma_i = \gamma_i - \gamma_{i-1}$. Equivalently, for the elements $x(iN-L+1), \ldots, x(iN-1)$, we change $S_{i-1}$ to an effective scaling factor of $(S_{i-1} + \Delta\gamma_i)$. This permits a BFP representation of the data vector $\mathbf{x}(n)$ with common exponent $\gamma_i$ during block-to-block transition phase too, i.e., when part of $\mathbf{x}(n)$ comes from the $(i-1)$th block and part from the $i$th block. In practice, such rescaling is effected by passing each of the delayed terms $\overline{x}(n-j)$, $j = 1, \ldots, L-1$, through a rescaling unit that applies $\Delta\gamma_i$ number of right or left shifts on $\overline{x}(n-j)$, depending on whether $\Delta\gamma_i$ is positive or negative, respectively. This is, however, done only at the beginning of each block, i.e., at indexes $n = iN$, $i \in Z$. Also, note that though for Case A, $\Delta\gamma_i \geq 0$, for Cases B and C, however, $\Delta\gamma_i < 0$, meaning that in these cases, the aforementioned mantissas from the $(i-1)$th block are actually scaled up by $2^{-\Delta\gamma_i}$. It is, however, not difficult to see that the effective scaling factor $(S_{i-1} + \Delta\gamma_i)$ for the elements $x(iN-L+1), \ldots, x(iN-1)$ still remains lower bounded by $S_{\min}$, thus ensuring no overflow during the filtering operation.

The output error $e(n)$ is then evaluated as $e(n) = \overline{e}(n) \cdot 2^{\gamma_i + \psi_n}$ where the mantissa $\overline{e}(n)$ is given as $\overline{e}(n) = \overline{d}(n) \cdot 2^{-\psi_n} - \overline{y}(n)$. Clearly, computation of $\overline{e}(n)$ involves an additional step of right-shift operation on $\overline{d}(n)$—an operation that comes up frequently in FP arithmetic. However, since in an adaptive filter, filter coefficients are derived from data and thus cannot be represented in the FXP format when data are given in a scaled form, such a step seems to be unavoidable. It is easy to check that $|\overline{e}(n)| < 1$, since

$$\begin{aligned} |\overline{e}(n)| &\leq |\overline{d}(n)| \cdot 2^{-\psi_n} + |\overline{y}(n)| \\ &< 2^{-(S_i + \psi_n)} + \frac{1}{2} \leq \frac{2^{-\psi_n}}{L} + \frac{1}{2} \end{aligned} \tag{3}$$

as $2^{-S_i} \leq 1/L$. Except for the case $\psi_n = 0$ and $L = 1$, the RHS $\leq 1$. For the above description of $e(n)$, $\mathbf{x}(n)$, $d(n)$, and $\mathbf{w}(n)$, the weight update equation (2) can now be written as $\mathbf{w}(n+1) = \overline{\mathbf{v}}(n) \cdot 2^{\psi_n}$ where

$$\overline{\mathbf{v}}(n) = \overline{\mathbf{w}}(n) + \frac{\tilde{\mu}}{\overline{\sigma}_n^2 + \overline{\theta}_i}\overline{\mathbf{x}}(n)\overline{e}(n) \tag{4}$$

with $\overline{\sigma}_n^2 = \overline{\mathbf{x}}^T(n)\overline{\mathbf{x}}(n)$ and $\overline{\theta}_i = \theta 2^{-2\gamma_i}$, i.e., the mantissa of $\theta$ corresponding to the $i$th block.

To satisfy $|\overline{w}_k(n+1)| < 1/2$ for $k \in Z_L$, we first limit each $|\overline{v}_k(n)| < 1$, $k \in Z_L$, where $\overline{v}_k(n)$ denotes the $k$th component

TABLE I
SUMMARY OF THE 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 $\gamma_i$ as per the **Algorithm** of Section 3 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) Rescale the following elements of the $(i-1)$-th block: $\{\overline{x}(n) | n = iN - L + 1, ..., iN - 1\}$ as $\overline{x}(n) \rightarrow \overline{x}(n).2^{-\Delta\gamma_i}$, $\Delta\gamma_i = \gamma_i - \gamma_{i-1}$,
(c) Evaluate $\overline{\theta}_i = \overline{\theta}_{i-1}.2^{-2\Delta\gamma_i}$.
2. **Processing for the $i$-th block:**
   For $n \in Z_i' = \{iN, iN+1, ..., iN+N-1\}$
(a) $\overline{\sigma}_n^2$ estimation:
   If $n = iN$
   $\overline{\sigma}_n^2 = \overline{x}^2(n) + 2^{-2\Delta\gamma_i}[\overline{\sigma}_{n-1}^2 - \overline{x}^2(n-L)]$,
   else
   $\overline{\sigma}_n^2 = \overline{x}^2(n) + \overline{\sigma}_{n-1}^2 - \overline{x}^2(n-L)$.
(b) Filter ouput:
   $\overline{y}(n) = \overline{\mathbf{w}}^T(n)\overline{\mathbf{x}}(n)$,
   $ex\_out(n) = \gamma_i + \psi_n$.
   ($ex\_out(n)$ is the filter output exponent)
(c) Output error (mantissa) computation:
   $\overline{e}(n) = \overline{d}(n).2^{-\psi_n} - \overline{y}(n)$.
(d) Filter weight updating:
   $\overline{\mathbf{v}}(n) = \overline{\mathbf{w}}(n) + \frac{\tilde{\mu}}{\overline{\sigma}_n^2 + \overline{\theta}_i}\overline{\mathbf{x}}(n)\overline{e}(n)$.
   If $|\overline{v}_k(n)| < \frac{1}{2}$ for all $k \in Z_L = \{0, 1, ..., L-1\}$
   then
   $\overline{\mathbf{w}}(n+1) = \overline{\mathbf{v}}(n)$,
   $\psi_{n+1} = \psi_n$,
   else
   $\overline{\mathbf{w}}(n+1) = \frac{1}{2}\overline{\mathbf{v}}(n)$,
   $\psi_{n+1} = \psi_n + 1$.
   end.
   $i = i + 1$.
   Repeat steps 1 to 2.

of $\overline{\mathbf{v}}(n)$. Then, if each $\overline{v}_k(n)$ happens to be lying within $\pm 1/2$, we make the assignments: $\overline{\mathbf{w}}(n+1) = \overline{\mathbf{v}}(n)$, $\psi_{n+1} = \psi_n$. Otherwise, we scale down $\overline{\mathbf{v}}(n)$ by 2, in which case, $\overline{\mathbf{w}}(n+1) = \overline{\mathbf{v}}(n)/2, \psi_{n+1} = \psi_n + 1$. Since each $|\overline{w}_k(n)| < 1/2$ for $k \in Z_L$, it is sufficient to have $\tilde{\mu}/(\overline{\sigma}_n^2 + \overline{\theta}_i)|\overline{x}(n-k)||\overline{e}(n)| < 1/2$ in order to satisfy the relation $|\overline{v}_k(n)| < 1$, $k \in Z_L$. Noting that $|\overline{x}(n-k)|/(\overline{\sigma}_n^2 + \overline{\theta}_i) < 1$ and taking the upper limit of $|\overline{e}(n)|$ from (3) as $(2^{-\psi_n}/L + 1/2)$, it implies $\tilde{\mu} \leq L/(2^{-\psi_n+1} + L)$. Since $\psi_n \geq 0$ for all $n$, this then results in the following global upper bound: $\tilde{\mu} \leq L/(L+2)$.

It is also required that no overflow takes place during computation of $(\overline{\sigma}_n^2 + \overline{\theta}_i)$ in (4). Noting that $|\overline{x}(n-k)| < 2^{-S_{\min}} \leq 1/L$, we have, $\overline{\sigma}_n^2 + \overline{\theta}_i < L2^{-2S_{\min}} + \overline{\theta}_i \leq 1/L + \overline{\theta}_i$. Thus, for no overflow, it is sufficient to have $\overline{\theta}_i \leq (L-1)/L$ ($\approx 1$ for large $L$). In the proposed scheme, even though $\theta$ is fixed, $\overline{\theta}_i$ is to be updated once for each block by appropriate right/left shift operations on its previous value. Since both right and left shifts may be in operation, a practical approach will be to set its initial value by placing a binary 1 at the central bit location of the corresponding register, with all other bits set to binary 0. In that case, the initial value of $\gamma$ may be taken to be the average block exponent, say $\gamma_{\text{av}}$, which needs to be estimated beforehand heuristically from some *a priori* knowledge of input statistics and $S_{\min}$. Finally, we evaluate $\overline{\sigma}_n^2$ as

$$\overline{\sigma}_{n+1}^2 = \overline{x}^2(n+1) + \overline{\sigma}_n^2 - \overline{x}^2(n-L+1) \tag{5}$$

with a slight modification for the case when $\overline{x}(n+1)$ belongs to one block while all other entries of $\overline{\mathbf{x}}(n+1)$ to the previous block. In this case, the modified recursion is given by: $\overline{\sigma}_{n+1}^2 = \overline{x}^2(n+1) + 2^{-2\Delta\gamma_i}(\overline{\sigma}_n^2 - \overline{x}^2(n-L+1))$.

TABLE II
COMPARISON BETWEEN BFP VIS-A-VIS FP-BASED REALIZATION OF THE NLMS ALGORITHM. NUMBER OF OPERATIONS REQUIRED PER ITERATION FOR (a) CALCULATING STEP-SIZE (FOR BFP, ONLY STEP-SIZE MANTISSA), (b) WEIGHT UPDATING, AND (c) FILTERING ARE GIVEN. UNLESS SPECIFIED OTHERWISE, ALL THE GENERAL OPERATIONS INDICATE MANTISSA OPERATIONS. * IMPLIES THREE EXTRA SHIFT OPERATIONS THAT MAY BE REQUIRED AT THE STARTING INDEX OF EACH BLOCK

| 2(a) | MAC | Shift | Addition | Division | Exponent Comparison | Exponent Addition |
|---|---|---|---|---|---|---|
| BFP | 2 | Nil* | 1 | 1 | Nil | Nil |
| FP | 2 | 7 | 1 | 1 | 3 | 7 |

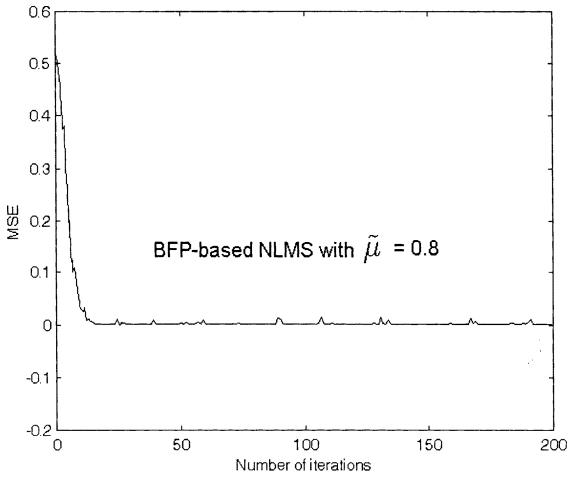
| 2(b) | MAC | Shift | Magnitude Check | Exponent Comparison | Exponent Addition |
|---|---|---|---|---|---|
| BFP | $(L+1)$ | $L$ | $L$ | Nil | 1 |
| FP | $(L+1)$ | $(2L+1)$ | Nil | $L$ | $(2L+2)$ |

| 2 (c) | MAC | Shift | Exponent Comparison | Exponent Addition |
|---|---|---|---|---|
| BFP | $L$ | Nil | Nil | 1 |
| FP | $L$ | $2L$ | $L$ | $2L$ |

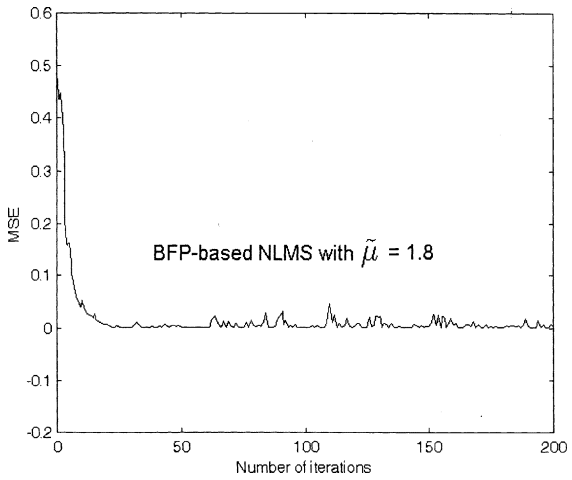
## IV. DISCUSSION AND CONCLUSION

The BFP-based implementation of the NLMS algorithm, as proposed above and summarized in Table I, relies mostly on FXP arithmetic and thus enjoys less processing time than its FP-based counterpart. For example, to compute filter output, the BFP method requires $L$ "Multiply and Accumulate (MAC)" operations (FXP) and, at the most, one exponent addition for each $n$. In FP, this, however, requires the following additional operations per sample: 1) $2L$ shifts (assuming availability of single-cycle barrel shifters); 2) $L$ exponent comparisons; and 3) $2L-1$ exponent additions. Similar advantages exist in weight updating also. In both cases, the number of additional operations required under FP-based realization increases linearly with filter length $L$. Table II provides a comparative account of the two approaches in terms of number of operations required. It is easily seen from this table that given a fixed-point processor with single-cycle MAC and barrel shifter units, the proposed scheme is about *three times faster* than an FP-based implementation. The storage requirement in the FP scheme is also higher as it needs to store several exponent values at each index.

The proposed algorithm has been implemented on the 16-bit TMS320C54X fixed-point processor. A system identification problem was considered where a unit variance random signal $x(n)$ was used to construct the plant output $d(n) = x(n) +$




(a)




(b)

Fig. 1. MSE characteristics of the BFP-based NLMS algorithm for (a) $\tilde{\mu} = 0.8$ and (b) $\tilde{\mu} = 1.8$ under the precision of 12 bits for mantissa and 4 bits for exponent.

$0.8x(n-1) + 0.64x(n-2) + z(n)$, $z(n)$ being the observation noise (white) with variance 0.01. Each sample of $x(n)$ and $d(n)$ was initially taken in FP form in a 16-bit word with upper 12 bits representing the signed mantissa and the lower four bits representing the signed exponent. Block formatting based on a block length of 10 was performed, and $\tilde{\mu}$ was chosen as 0.8. Details of the implementation are skipped here and are given in [12]. Preliminary investigations into an FP-based implementation on the same processor, however, confirm our assessment that the proposed scheme achieves a speedup of about three over its FP counterpart. Finally, simulations of the proposed scheme for two values of $\tilde{\mu}$, 0.8 and 1.8, result in the mean square error (MSE) characteristics shown in Fig. 1(a) and (b), respectively. It is clear from these figures that the restriction $\tilde{\mu} < 1$ does not alter the convergence speed in any significant way, but, as is expected, the smaller value of $\tilde{\mu}$ produces less excess mean square error in the steady state.

## ACKNOWLEDGMENT

The authors gratefully acknowledge certain useful suggestions by A. Polley that helped to improve the exponent assignment algorithm. The authors also convey their feeling of gratitude to the reviewers for their valuable comments.

## REFERENCES

[1] J. H. Wilkinson, *Rounding Errors in Algebraic Processes*. Englewood Cliffs, NJ: Prentice-Hall, 1963.
[2] 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, pp. 1076–1086, Apr. 1999.
[3] K. Kalliojärvi and J. Astola, "Roundoff errors in block-floating-point systems," *IEEE Trans. Signal Processing*, vol. 44, pp. 783–790, Apr. 1996.
[4] P. H. Bauer, "Absolute error bounds for block floating point direct form digital filters," *IEEE Trans. Signal Processing*, vol. 43, pp. 1994–1996, Aug. 1995.
[5] ——, "Asymptotic behavior of digital filters with block floating point arithmetic," in *Proc. ICASSP*, vol. III, Adelaide, Australia, 1994, pp. 609–612.
[6] A. Erickson and B. Fagin, "Calculating FHT in hardware," *IEEE Trans. Signal Processing*, vol. 40, pp. 1341–1353, June 1992.
[7] S. Sridharan and G. Dickman, "Block floating point implementation of digital filters using the DSP 56 000," *Microprocess. Microsyst.*, vol. 12, no. 6, pp. 299–308, July–Aug. 1988.
[8] 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.
[9] D. Williamson, S. Sridharan, and P. G. McCrea, "A new approach to block floating point arithmetic in recursive digital filters," *IEEE Trans. Circuits Syst.*, vol. CAS-32, pp. 719–722, July 1985.
[10] F. J. Taylor, "Block floating point distributed filters," *IEEE Trans. Circuits Syst.*, vol. CAS-31, pp. 300–304, Mar. 1984.
[11] S. Haykin, *Adaptive Filter Theory*. Englewood Cliffs, NJ: Prentice-Hall, 1986.
[12] M. Chakraborty, A. Mitra, and T. Krishnaswamy, "An implementation of the block floating point based NLMS algorithm on TMS320C54X processor," *Microprocess. Microsyst.*, submitted for publication.