On efficient implementations of the Set-Membership NLMS algorithm for real-time applications

José A. Apolinário Jr.[†] and Marcello L. R. de Campos[‡]

***Abstract*— This paper addresses two efficient implementations of the Set-Membership Normalized Least-Mean Squares (SM-NLMS) algorithm suited for real-time applications with strong limitation in computational complexity. It is shown how to take advantage of a particular property of set-membership algorithms, their data selective updating, to improve performance with no additional costs concerning computational complexity or hardware. Convergence speed and misadjustment of the proposed alternative implementations are compared to those of the conventional implementation of the SM-NLMS algorithm when the computational complexity per iteration is limited.**

***Index Terms*— Set-Membership adaptive filters, efficient implementations, Normalized LMS algorithm.**

I. Introduction

Set-Membership adaptive algorithms are considered attractive options for a wide range of applications because of their low average computational complexity, fast convergence, and good tracking capability. This paper presents two efficient implementations for the SM-NLMS algorithm [1] which take advantage of the fact that, in this algorithm, coefficient-vector updating is not performed unless the output error of the filter is larger than a certain threshold. The idea presented here avoids the typical peak complexity that the SM-NLMS algorithm imposes to the processor when updating is required. We investigate how simple implementation strategies can improve performance of the SM-NLMS algorithm in a scenario where computational resources are limited. Although presented for the SM-NLMS algorithm, the techniques can be extended easily to other SM algorithms, e.g., the Set-Membership Affine-Projections algorithm (SM-APA) [2].

This paper is organized as follows: In Section II, basic concepts concerning the SM-NLMS algorithm are reviewed. The two efficient implementations proposed in this work are introduced in Section III. Simulation results are detailed in Section IV and conclusions are summarized in Section V.

II. The Set-Membership NLMS Algorithm

Conventional adaptive filtering algorithms search, at time instant k, a coefficient vector $\boldsymbol{w}$ that minimizes the mean squared output estimation error (MSE), $E[e^2(k)]$, where

$$e(k) = d(k) - \boldsymbol{w}^T \boldsymbol{x}(k)$$

The authors are with the [†]Department of Electrical Engineering, Instituto Militar de Engenharia, Praça General Tibúrcio, 80, 22.290-270, Rio de Janeiro-RJ, Brazil, e-mail: apolin@ieee.org and [‡]Program of Electrical Engineering, COPPE/UFRJ, P. O. Box 68504, 21.945-970, Rio de Janeiro-RJ, Brazil, e-mail: campos@lps.ufrj.br.

is the output estimation error, and $d(k)$ and $\boldsymbol{x}(k)$ are the desired output signal and the input-signal vector, respectively.

In set-membership filtering (SMF), an upper bound of the output estimation error is specified such that all coefficient vectors satisfying the error constraint are considered feasible. The resulting adaptation algorithms are data-selective with a considerably reduced average computational complexity. Furthermore, the sparse updating usually results in lower misadjustment because the algorithms do not utilize the input-desired data pair if it does not imply innovation. In the SM-NLMS algorithm proposed in [1], the coefficient vector $\boldsymbol{w}(k-1)$ is updated to $\boldsymbol{w}(k)$ only if the *a priori* output error exceeds a certain threshold γ. Let $\mathcal{S}$ denote the space model, i.e., $(\boldsymbol{x}, d) \in \mathcal{S}$, and Θ the set of all possible vectors $\boldsymbol{w}$ that result in an error with a norm not exceeding γ. The *feasibility* set Θ is defined as the set of all filter vectors satisfying the error constraint for all possible input-desired data pairs and is given by

$$\Theta = \bigcap_{(\boldsymbol{x},d)\in\mathcal{S}} \left\{ \boldsymbol{w} \in \boldsymbol{R}^N \,:\, |d - \boldsymbol{w}^T \boldsymbol{x}| \leq \gamma \right\} \qquad (1)$$

The set of all $\boldsymbol{w}$ satisfying the error bound, obtained after training with k *input-desired* data pairs $\{\boldsymbol{x}, d\}$, denoted by $\mathcal{H}(k)$, is called the *constraint set* and can be expressed as

$$\mathcal{H}(k) = \left\{ \boldsymbol{w} \in \boldsymbol{R}^N \colon \left| d(k) - \boldsymbol{w}^T \boldsymbol{x}(k) \right| \leq \gamma \right\} \qquad (2)$$

The membership set $\psi(k) = \cap_{i=1}^{k} \mathcal{H}(i)$ is a subset of the feasibility set and is defined as the minimal set estimate for Θ at time k.

The recursion for the SM-NLMS algorithm can be summarized as follows:

$$\boldsymbol{w}(k) = \boldsymbol{w}(k-1) + \alpha(k) \frac{e(k)\boldsymbol{x}(k)}{\boldsymbol{x}^T(k)\boldsymbol{x}(k)} \qquad (3)$$

The step size, $\alpha(k)$, is computed at each iteration according to

$$\alpha(k) = \begin{cases} 1 - \frac{\gamma}{|e(k)|} & \text{if } |e(k)| > \gamma \\ 0, & \text{otherwise} \end{cases} \qquad (4)$$

The two implementations to be described in the next section are based on the fact that coefficient updates are carried out only when the output error is beyond a pre-established threshold γ.

III. The efficient implementations

A system with infinite impulse response (IIR), e.g., the one shown in Figure 1, is to be identified. Suppose the Digital Signal Processor (DSP) and the sampling frequency to be

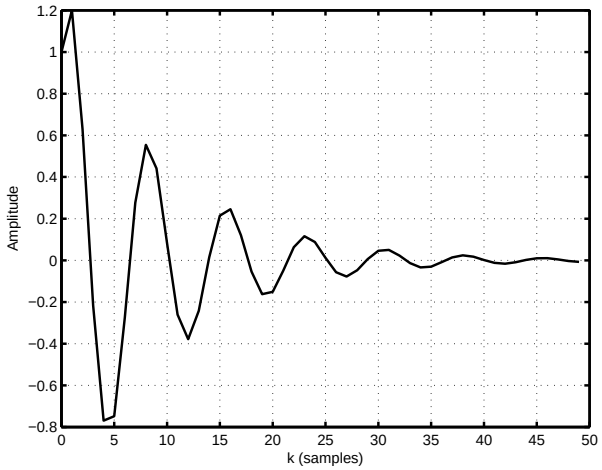

Fig. 1. IIR system impulse response.

used limit the order of a finite impulse-response (FIR) adaptive filter. In this case, performance is degraded due to insufficient modeling order.

A simple and natural way of obtaining a better FIR modeling of an IIR system is to increase the number of taps of the adaptive filter. In order to work with the same hardware at the same sampling frequency, one can take advantage of the basic feature of the SM-NLMS algorithm of not performing the parameter update at every iteration.

As observed in [3], [4], SM adaptive filtering (SM-AF) algorithms demand coefficient update only for a fraction of input data; after convergence, coefficient-update ratio can be as low as 10% when the environment is stationary and parameter γ is selected appropriately. In traditional SM-AF implementations, the DSP is usually left idle whenever coefficient update is not required, i.e., output error is smaller than the threshold. The advantages of set-membership adaptive filters in this case are possible lower misadjustment and lower power consumption when compared to conventional adaptive filters. However, the maximum computational complexity per iteration is similar to that of the NLMS algorithm.

The alternative implementation scheme suggested here takes advantage of processor idle time during iterations that do not require coefficient update. This idle time is used to extend computational capacity and, as a consequence, to be able to update longer filters than would be possible with the NLMS or the conventional SM-NLMS algorithms. The implementation schemes are based on a dissociation between the rate the system produces the output signal and the associated output error, from the rate the system updates the coefficient vector. In traditional implementations of adaptive filters, even for SM-AFs, these two rates are one and the same. In the implementation schemes proposed herein, system output and error are still produced at every iteration, but coefficient update can span two or more iterations. This is an extra degree of freedom left to the designer: longer filters take more iterations to be fully updated, but may perform better in steady state. Naturally, we assume that these larger filter orders require a computational capacity that is not available in the DSP in use and therefore are not an option for the conventional NLMS or SM-NLMS algorithm implementations.

In the following sub-sections, two alternative and efficient solutions to the implementation of SM-AFs will be described. In the first implementation scheme, a buffer is used to save the input-desired signal pair $(\mathbf{x}, d)$ in case the DSP is busy performing coefficient updates for previous data pairs. In the second implementation scheme, data pairs that are received while the DSP is busy updating previous data pairs are only used for calculating system output and error, and are discarded afterwards; they are not used for coefficient update.

A. Implementation I

This first implementation scheme does not discard the input data samples that are presented to the adaptive filter while the updating process has not been completed for previous iterations. Data are buffered and, as soon as the processor is free, they can be used to update the coefficients, depending on the threshold comparison test to be performed. The solely and obvious constraint that must be imposed in order to make this approach feasible is that the buffer must be of a limited size. As previously mentioned, input and desired signals received at any iteration are immediately used to generate an output signal and, optionally, the output error. The input-signal vector and desired signal are then stored in a buffer to be used later for coefficient update. Care is taken such that if there is no time to process the data pairs stored in the buffer or, in other words, whenever the buffer is full, the oldest samples are discarded.

B. Implementation II

In this second implementation scheme, if the computational complexity for a larger number of coefficients is too high to be performed in one single sample interval, all the rest of the computation needed for the coefficients update is carried out in the forthcoming iterations. The next test to decide if the estimate needs to be updated will be performed only after the previous update process has been completed; meanwhile, the filter runs with fixed coefficients such that the incoming samples are used only for the computation of the filter output. This solution, summarized in Table I, is very simple, does not need a buffer, and allows, with a slight decrease in the speed of convergence, the use of a larger number of coefficients with the same available hardware. Note in Table I that we initialize T such that $\hat{T} = T+1$ is the total number of iterations needed to update the coefficient vector. The DSP programmer would be in charge of optimizing the division of tasks to perform the updating process according to the processor capabilities.

IV. Simulation Results

In this section, an illustrative example is described. An IIR system, such as the one of Figure 1, having its system function given by

$$H(z) = \frac{1}{z^{-2} - 1.2z^{-1} + 0.81}$$

is to be identified using an FIR SM-NLMS adaptive filter. The signal-to-noise ration (SNR) was set to 40 dB and the input

TABLE I

AN EFFICIENT IMPLEMENTATION OF THE SM-NLMS ALGORITHM.

SM-NLMS: Implementation II

Initialization:
T is the number of extra iterations needed to update $\boldsymbol{w}$
$\hat{T} = T + 1$ and $\epsilon =$ small constant
for each k
{ $e(k) = d(k) - \boldsymbol{w}^T(k-1)\boldsymbol{x}(k)$;
if $(T == 0 \quad \& \quad |e(k)| > \gamma)$
{ $T = 0$ means the Conventional SM-NLMS algorithm:
$\alpha(k) = 1 - \gamma/|e(k)|$;
$\boldsymbol{w}(k) = \boldsymbol{w}(k-1) + \alpha(k)\frac{e(k)}{\epsilon + ||\boldsymbol{x}(k)||^2}\boldsymbol{x}(k)$;
}
elsif $(\hat{T} == T + 1 \quad \& \quad |e(k)| > \gamma)$
{ $\hat{k} = k$;
Compute the **first step** of the update:
$\begin{cases} \alpha(\hat{k}) = 1 - \gamma/|e(\hat{k})|; \\ \boldsymbol{w}(\hat{k}) = \boldsymbol{w}(\hat{k}-1) + \alpha(\hat{k})\frac{e(\hat{k})}{\epsilon + ||\boldsymbol{x}(\hat{k})||^2}\boldsymbol{x}(\hat{k}); \end{cases}$
$\hat{T} = 1$;
$\boldsymbol{w}(k) = \boldsymbol{w}(k-1)$;
}
else
{ $\boldsymbol{w}(k) = \boldsymbol{w}(k-1)$;
if $(\hat{T} < T + 1)$
{ $\hat{T} = \hat{T} + 1$;
Compute the $\hat{T}th$ **step** of the update above;
if $(\hat{T} == T + 1)$
{ $\boldsymbol{w}(k) = \boldsymbol{w}(\hat{k})$;
}
end
}
end
}
end
}
end

signal is colored noise with unit variance and an eigenvalue spread (of its autocorrelation matrix)close to 2000.

In order to investigate the performance of Implementation I, we have initially tested the above setup in 250 independent runs, with a large buffer which is never full and, therefore, no data pair is lost. The order of the SM-NLMS I algorithm was fixed in 63 (64 taps) and we varied the number of coefficients that the processor is able to update per iteration, we named it C. The result in terms of learning curves is depicted in Figure 2 where the curves of the conventional SM-NLMS with orders $N = 31$, 15, and 7 are shown along with the computational complexity equivalent algorithm-implementation described herein for $C = 32$, $C = 16$, and $C = 8$[1], respectively. For all algorithms in our simulations, we have used $\gamma = \sqrt{5}\sigma_n$, a typical value as noted in [5]. As shown in Figure 2, there is an influence of the number of taps updated per iteration with algorithm performance as C decreases. Nevertheless, compared to the conventional SM-NLMS algorithm with equivalent computational complexity, the results are very good, particularly after convergence, for the effects of undermodeling are severely reduced.

In this same experiment, we present in Figure 3 the buffer occupation for the particular case of $C = 32$. From this curve,

[1]Note that $C = 32, 16$, and 8, correspond to updating 64 in $2, 4$, and 8 iterations, respectively.

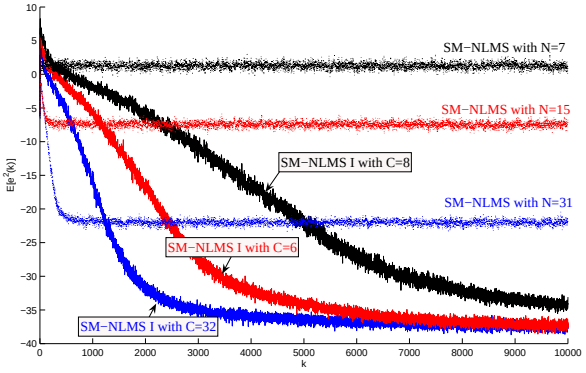

Fig. 2. Learning curves of SM-NLMS algorithms (original and Implementation I) with different computational complexities.

we see that the buffer is heavily used during convergence and is kept close to empty in steady state.

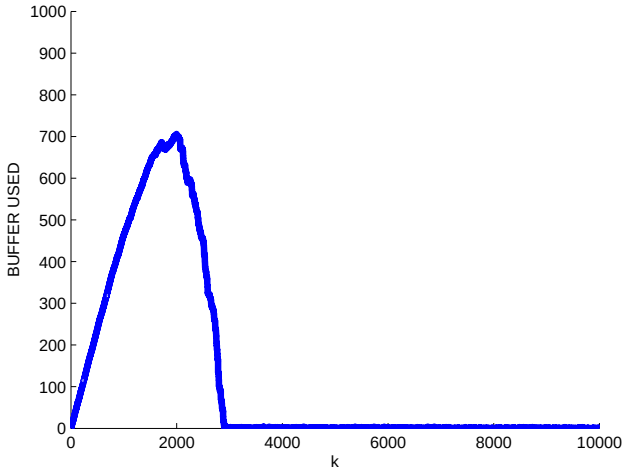

Fig. 3. Buffer occupation for Implementation I with $B = 1000$ and $C = 32$.

Yet, for this first implementation, we have analyzed the effect of the buffer-size, plotting in Figure 4 the learning curve for three different values: $B = 1000$, $B = 10$, and $B = 1$ while keeping C constant and equal to 32. The resulting curves show that the impact of not using all data pairs is not very significant. $B = 1$ corresponds to the Implementation II discussed in the previous section. As its performance is similar to that of the Implementation I, in terms of speed of convergence, and as it does not require buffering of data pairs, more attention is devoted to this implementation form in the remainder of this article.

We have assumed that a given processor when operating at a certain sampling frequency is able to perform coefficient updating for a maximum order $N = 18$ for the conventional SM-NLMS algorithm. Knowing that this value would result in an implementation that does not meet the desired performance due to undermodeling, the order was increased initially in 50% (27) and later doubled (36) for Implementation II described in

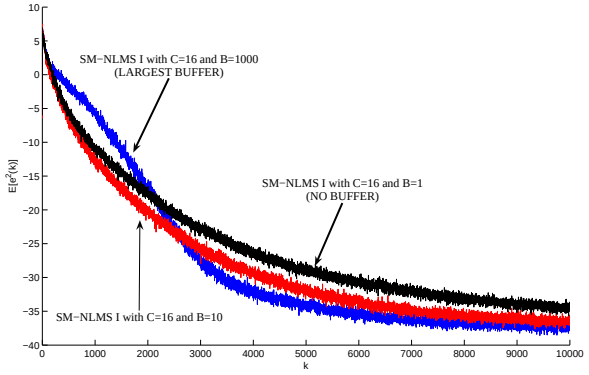

Fig. 4. Influence of buffer size in the MSE for Implementation I.

TABLE II
SUMMARY OF PEAK COMPUTATIONAL COMPLEXITY IN TERMS OF FLOATING POINT OPERATIONS (FPO).

Algorithm	FPO		
SM-NLMS	$3(N+1)+3$		
SM-NLMS Imp. II	$N+1+\frac{2N+5}{T+1}$		
Numerical example for $N=18$:			
Size of $\boldsymbol{w}$	SM-NLMS	Imp. II ($T=1$)	Imp. II ($T=2$)
$N+1=19$	**60**	39.5	32.67
$N+\frac{N}{2}+1=28$	87	**57.5**	47.67
$2N+1=37$	114	75.5	**62.67**

Table I. For this implementation, peak complexity per iteration is $(N+1)+(2N+5)/(T+1)^2$, not the $3(N+1)+3$ of the conventional SM-NLMS algorithm. T is the number of extra iterations required to complete coefficient updates.

The learning curves shown in Figure 5 were obtained with the error bound set to $\gamma = \sqrt{5}\sigma_n$, for an average of 1000 independent runs. The worst performance in steady-state presented in Figure 5 corresponds to the Conventional SM-NLMS algorithm with filter order $N = 18$. The other two MSE curves correspond to the SM-NLMS Implementation II with filter order equal to 27 and $T = 1$ extra iteration needed to update the coefficient vector, and to the SM-NLMS Implementation II with filter order equal to 36 and $T = 2$ extra iterations needed for the update of the coefficient vector.

For all three cases the computational complexities are similar, meaning we can use the same DSP at the same sampling frequency.

The corresponding learning curves show a clear advantage of the proposed implementation.

V. CONCLUSIONS

In this work, two optimized implementations—named Implementation I and II—for the set-membership Normalized LMS algorithm were discussed. The main idea behind these implementations is to avoid the peak complexity of SM adaptive filters, making them suited for real-time applications.

[2]Note that, if $T = 0$, the computational complexity corresponds to the conventional SM-NLMS algorithm; moreover, the first $N+1$ multiplications correspond to the fixed coefficient vector filtering operation.

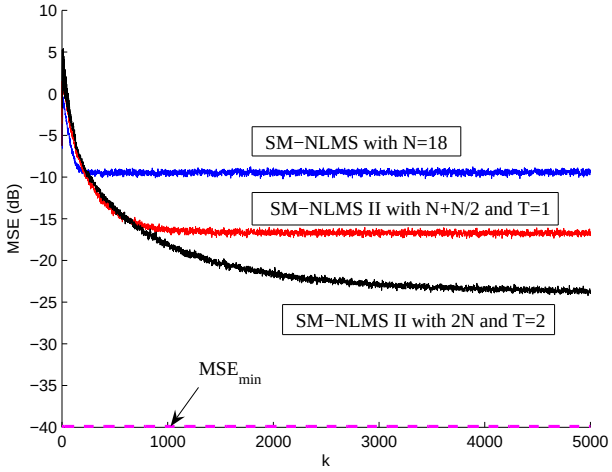

Fig. 5. Comparing the performance of the SM-NLMS Implementation II for different values of filter order and similar computational complexity.

Implementation I (SM-NLMS I) is based on buffering the input-desired data pair every iteration the processor is busy updating the coefficient vector. As seems through computer experiments, the scheme works well as the buffer is filled during learning and emptied in steady state. Nevertheless, it was also seem that the buffer size does not cause an important loss of performance when shortened. Implementation II follows as a better (although very simple) option for performing similarly without the need of using a buffer. It was shown, via computer simulations, that the proposed algorithm offers ways of improving the performance of an adaptive filter by allowing the increase of the filter order keeping exactly the same computational complexity. This feature might be very important in practical implementations when the DSP is operating close to its processing capacity.

ACKNOWLEDGMENTS

The authors thank CAPES, CNPq, and FAPERJ for partial funding of this work.

REFERENCES

[1] S. Gollamudi, S. Nagaraj, S. Kapoor, and Y. F. Huang, "Set-membership filtering and a set-membership normalized LMS algorithm with an adaptive step size," *IEEE Signal Processing Letters*, vol. 5, no. 5, pp. 111–114, May 1998.

[2] S. Werner and P. S. R. Diniz, "Set-membership affine projection algorithm," *IEEE Signal Processing Letters*, vol. 8, no. 8, pp. 231–235, Aug. 2005.

[3] S. Gollamudi, S. Kapoor, S. Nagaraj, and Y. F. Huang, "Set-membership adaptive equalization and an updator-shared implementation for multiple channel communications systems," *IEEE Trans. Signal Processing*, vol. 46, no. 9, pp. 2372–2385, Sept. 1998.

[4] S. Gollamudi and Y. F. Huang, "Updator-shared adaptive parallel equalization (u-SHAPE) using set-membership identification," in Proc. Int. Symp. Circuits and Systems (ISCAS), Atlanta, USA, May 1996.

[5] J. F. Galdino, J. A. A. Jr., and M. L. R. de Campos, "A set-membership NLMS algorithm with time-varying error bound," in Proc. Int. Symp. Circuits and Systems (ISCAS), Island of Kos, Greece, May 2006.