New Algorithms for Improved Adaptive Convex Combination of LMS Transversal Filters

Jerónimo Arenas-García(*), *Member, IEEE*, Vanessa Gómez-Verdejo and Aníbal R. Figueiras-Vidal, *Senior Member, IEEE*

(*) Corresponding author address:
Dept. of Signal Theory and Communications, Universidad Carlos III de Madrid, 28911 Leganés-Madrid, Spain.
E-mail: jarenas@tsc.uc3m.es. http://www.tsc.uc3m.es
This work has been partly supported by CICYT grant TIC2002-03713.
A reduced version of this work has been previously presented at the 2003 IEEE Intl. Symp. on Intelligent Signal Processing.

Abstract

Among all adaptive filtering algorithms, Widrow and Hoff's Least Mean Square (LMS) has probably become the most popular because of its robustness, good tracking properties and simplicity. A drawback of LMS is that the step size implies a compromise between speed of convergence and final misadjustment. To combine different speed LMS filters serves to alleviate this compromise, as it was demonstrated by our studies on a two filter combination that we call Combination of LMS Filters (CLMS). Here, we extend this scheme in two directions. First, we propose a generalization to combine multiple LMS filters with different steps that provides the combination with better tracking capabilities. Second, we use a different mixing parameter for each weight of the filter in order to make independent their adaption speeds. Some simulation examples in plant identification and noise cancellation applications show the validity of the new schemes when compared to the CLMS filter and to other previous variable step approaches.

Index Terms

Least Mean Square (LMS), adaptive filtering, convex combination, plant identification, noise cancellation, transform-domain adaptive filters

I. INTRODUCTION

Information processing in variable and noisy environments is usually accomplished by means of adaptive filters. Among all adaptive algorithms, Widrow and Hoff's Least Mean Square (LMS) [1] has probably become the most popular for its robustness, good tracking capabilities and simplicity, both in terms of computational load and easiness of implementation [2], [3]. It has therefore been successfully applied to a wide variety of adaptive filtering problems, including plant identification [4] or noise cancellation applications [5], [6].

An important limitation regarding LMS filters is that the selection of a certain value for the step size implies a compromise between rate of convergence and final misadjustment (see [7], for instance). Many authors have proposed to alleviate this drawback by using a variable step size. Several batch algorithms can be found in [8]. When on-line learning is mandatory it is still possible to use algorithms that manage the adaption step in a stochastic manner [9], [10], [11]. These step size management methods have shown to be effective at improving LMS results, but they have the drawback of introducing some hyperparameters which are themselves subject to the same speed vs precision compromise. So, it is not easy to establish these hyperparameters, and optimal values are highly dependent on the dynamics of the scenario in which the filters are applied.

In [12] we proposed an alternative approach consisting on an adaptive convex combination of two LMS filters: one of them is fast, the other is slow (CLMS algorithm). These filters are combined in such a manner that the advantages of both component filters are kept: the rapid convergence from the fast filter, and the reduced steady state error from the slow filter. In [13], [14] we showed that, in steady state, CLMS performs at least as well as the best of the component filters. This scheme, that has also proven to outperform previous variable step approaches, is an analogy of a well-known neurological fact: human brains combine fast and coarse reactions against abrupt changes in the environment, with an early processing at the amygdala, and more elaborated but slower responses taken in the neocortex at a conscious level.

A similar combination scheme has been previously used by Jacobs et al. [15] in the field of neural networks. In [16], [17], a combination of multiple predictors of different orders is also proposed to obtain a universal linear predictor. The CLMS algorithm, however, is just aimed at improving the speed of convergence vs precision balance of LMS filters. So, the two component filters only differ in their step sizes.

In this paper we present two extensions of the CLMS algorithm. First, a generalization of CLMS to combine multiple LMS filters (M-CLMS algorithm [18]). As discussed in [7], in non stationary situations, the optimal value for the adaption step depends on the speed with which the Wiener solution changes. This way, each LMS filter in the M-CLMS scheme will be particularly good at tracking changes that occur at a certain speed, and the combination of all the filters will be able to effectively track most kinds of changes.

The second extension we describe in the paper, D-CLMS [19], modifies the combination scheme of CLMS, so that a different mixing parameter is used for each weight of the filter. The advantages of using different mixing parameters are similar to those of using a different step size for each weight of an LMS filter, as studied in [9]; i.e.,

- a reduced sensitivity to high eigenvalue spread in the autocorrelation matrix of the input process, and
- a lower tracking error when some coefficients of the Wiener solution remain unaltered.

The basic versions of all our combination schemes require just an extra parameter: the step size used to adapt the combination. Nevertheless, an improved performance can be obtained when using some additional mechanisms. Although these procedures introduce also some additional parameters, their selection is very easy and general settings exist that work well for most situations.

The rest of the paper is organized as follows. In the next section we briefly review the CLMS algorithm. Then, M-CLMS and D-CLMS filters, plus some procedures to improve their performances, are presented in Sections III and IV, respectively, and their computational complexity is analyzed in Section V. Section VI is devoted to several plant identification and noise cancellation experiments that show the advantages of the new algorithms both in comparison to CLMS and to previous variable step approaches. Finally, we present general conclusions and discuss some lines to extend this work.

II. THE CLMS ALGORITHM

Figure 1 shows the basic form of the discrete-time adaptive filtering problem. Different algorithms can be used to adapt weights $\mathbf{w}$ of the filter, most of them with the goal of minimizing an estimate of the expected quadratic error. The LMS algorithm adaption scheme [1] makes use of an instantaneous estimate of the gradient to search the minimum of the error surface

$$\mathbf{w}[k+1] = \mathbf{w}[k] + \mu e[k]\mathbf{x}[k] \tag{1}$$

where $\mathbf{x}[k]$ is the filter input at instant k, and $e[k]$ is the error incurred by the adaptive filter, i.e., $e[k] = d[k] - \mathbf{w}^T[k]\mathbf{x}[k]$, $d[k]$ being the desired output of the filter.

It is a well-known fact that the selection of a fixed step size μ implies a compromise between the rate of convergence to the optimal solution and the final misadjustment of the filter. Our proposal in [12], [13], [14] was to use an adaptive combination of two LMS filters, the first being a fast filter (i.e., with a high step size μ_1) and the second a slow filter (low adaption step μ_2). In principle both algorithms operate completely decoupled from each other, and they are conducted by their own errors $e_1[k]$ and $e_2[k]$, respectively.

The CLMS filter uses a convex combination of the weights of the two LMS filters

$$\mathbf{w}_{eq}[k] = v[k]\mathbf{w}_1[k] + (1 - v[k])\mathbf{w}_2[k] \tag{2}$$

where the mixing coefficient $v[k]$ is kept in the interval $(0, 1)$ by defining it as $v[k] = \mathrm{sgm}\,(a[k]) = 1/(1 + e^{-a[k]})$. Using this expression, it is obvious that both the output and the error of CLMS can be calculated using the same convex combination form:

$$\begin{aligned} y_{eq}[k] &= v[k]y_1[k] + (1 - v[k])y_2[k] && (3) \\ e_{eq}[k] &= v[k]e_1[k] + (1 - v[k])e_2[k] && (4) \end{aligned}$$

Combination parameter $a[k]$ is adapted to minimize the error of the overall adaptive filter, also using the LMS adaption rule:

$$a[k+1] = a[k] - \frac{\mu_a}{2}\frac{\partial e_{eq}^2[k]}{\partial a[k]} = a[k] - \mu_a e_{eq}[k]\left(e_1[k] - e_2[k]\right)v[k]\left(1 - v[k]\right) \tag{5}$$

This seems to be a good strategy because of the robustness of LMS schemes [20].

In equation (5), μ_a must be fixed to a very high value, so that the combination is adapted even faster than the fastest LMS filter. In the practice, we limit the values of $a[k]$ to the interval $[-4, 4]$, to prevent the algorithm from stopping because either $v[k]$ or $1 - v[k]$ is too close to 0.

The combination of LMS filters operates in a very intuitive way: when fast changes appear, the fast filter outperforms the slow one, making $v[k]$ evolve towards 1, and $\mathbf{w}_{eq}[k] \approx \mathbf{w}_1[k]$. On the contrary, in stationary situations, it is the slow filter which gets a lower quadratic error. So, $v[k]$ decreases near 0, and the combination achieves the low misadjustment of the slow LMS filter.

The performance of the basic combination algorithm, can be further improved if we take advantage of the faster convergence of the fast filter to speed up the convergence of the slow one. This can be done by step by step transferring a portion of weights $\mathbf{w}_1$ to $\mathbf{w}_2$. The modified adaption rule for $\mathbf{w}_2$ then becomes

$$\mathbf{w}_2[k+1] = \alpha\left(\mathbf{w}_2[k] + \mu_2 e_2[k]\mathbf{x}[k]\right) + (1 - \alpha)\mathbf{w}_1[k+1]. \tag{6}$$

This weight transfer procedure is only applied if the fast filter is significantly outperforming the slow one. More details on this will be given in the next section.

A second modification to the basic combination serves to reduce the pernicious effect of factor $e_1[k] - e_2[k]$ in equation (5) when both errors are similar. To alleviate this problem we can include a momentum term for adapting parameter $a[k]$:

$$a[k+1] = a[k] - \frac{\mu_a}{2}\frac{\partial e_{eq}^2[k]}{\partial a[k]} + \rho(a[k] - a[k-1]) \tag{7}$$

Although the CLMS algorithm requires the introduction of some extra parameters, we will see that their selection is very easy, and optimal values are not very dependent on the particular scenario in which the filter is being applied.

III. Combining Several LMS Filters: The M-CLMS Algorithm

Adaptive filters allow to track the Wiener solution in nonstationary situations. In these cases, the error incurred by an LMS filter consists of the sum of two terms [7]. The first term is related to the misadjustment of the filter, and it is minimized when using small step sizes. The second term comes from the lag in tracking the varying optimal solution, and it is therefore minimized with a high adaption step. As a consequence, the optimal step size for a tracking situation takes an intermediate value, which depends on the speed with which the optimal solution varies.

In this section we extend the ideas of CLMS to the combination of multiple LMS filters which differ in their adaption steps. We will call this scheme a Multiple CLMS (M-CLMS) filter. We expect that this M-CLMS filter improves the tracking properties of LMS by retaining the capabilities of all the component filters.

When combining L LMS filters, the weight vector of M-CLMS becomes

$$\mathbf{w}_{eq}[k] = \sum_{i=1}^{L} v_i[k]\mathbf{w}_i[k] \tag{8}$$

where $\mathbf{w}_i$ are the coefficients of the i-th filter in the combination, that has step size μ_i (as before, we consider $\mu_1 > \mu_2 > \cdots > \mu_L$). Again, all the component filters are LMS adapted using their own errors.

For a good performance of CLMS, it was very important to use a convex combination form and a sigmoidal activation function for $v[k]$. Similarly, in the M-CLMS filter, we have replaced the sigmoid with a softmax activation function to obtain the combination weights associated to each individual filter

$$v_i[k] = \frac{\exp(a_i[k])}{\sum_{j=1}^{L} \exp(a_j[k])}; \qquad i = 1, \ldots, L \tag{9}$$

which guarantees that $0 < v_i[k] < 1$ and $\sum_{i=1}^{L} v_i[k] = 1$. Besides, the softmax function stabilizes the combination when any filter is systematically outperforming the rest, while allowing an easy switching if the situation changes.

Premultiplying both sides of (8) by $\mathbf{x}^T[k]$, the output of the M-CLMS filter can be expressed as the convex combination of the outputs of all component filters. Using the fact that $e_{eq}[k] = d[k] - y_{eq}[k]$, a similar expression is derived for the error:

$$y_{eq}[k] = \sum_{i=1}^{L} v_i[k] y_i[k] \tag{10}$$

$$e_{eq}[k] = \sum_{i=1}^{L} v_i[k] e_i[k] \tag{11}$$

As for the CLMS filter, the mixing coefficients are updated using LMS rules with the objective to minimize the overall quadratic error

$$a_i[k+1] = a_i[k] - \frac{\mu_a}{2} \frac{\partial e_{eq}^2[k]}{\partial a_i[k]} = a_i[k] - \mu_a e_{eq}[k] \sum_{j=1}^{L} \frac{\partial v_j[k]}{\partial a_i[k]} e_j[k]; \qquad i = 1, \ldots, L \tag{12}$$

It is easy to show that the partial derivatives of $v_j[k]$ with respect to $a_i[k]$ are given by

$$\begin{aligned} \frac{\partial v_i[k]}{\partial a_i[k]} &= v_i[k] - v_i^2[k] \\ \frac{\partial v_j[k]}{\partial a_i[k]} &= -v_i[k] v_j[k], \qquad j \neq i \end{aligned} \tag{13}$$

Now, we just have to introduce (13) into (12) to get the final adaption rules for the mixing parameters

$$a_i[k+1] = a_i[k] - \mu_a e_{eq}[k] \left(e_i[k] - e_{eq}[k]\right) v_i[k]; \qquad i = 1, \ldots, L. \tag{14}$$

As for the CLMS algorithm, we must fix μ_a to a very high value to allow a very fast adaption of the combination. Furthermore, we do not let any $v_i[k]$ to increase above ϵ, to avoid that the other mixing parameters get too close to 0, what would cause the stopping of the corresponding learning rules[1]. It can be shown that $v_i[k] < \epsilon$ is satisfied if we keep $|a_i[k]| \leq \frac{1}{2} \ln \left(\frac{\epsilon(L-1)}{1-\epsilon} \right) = \epsilon'$. Obviously, as the number of component filters in the M-CLMS scheme increases, the selection of ϵ must be more restrictive. We have checked that $\epsilon = 1 - 0.01(L-1)$ is a reasonable setting in most situations.

In Table I we list the pseudocode for the M-CLMS algorithm as described up to this point. To get a more practical M-CLMS implementation, it is possible to devise certain procedures for the interaction among the component filters.

A. Speeding up the convergence of the slowest filters

When an abrupt change in the optimal solution occurs, all the filters in the multiple scheme show independent convergences towards the new minimum. It is possible to speed up the convergence of the slowest filters (and, consequently, that of the M-CLMS filter) by step by step transferring to them a portion of the weights of fastest filters. To do this, we transfer a part of $\mathbf{w}_{eq}[k]$ to all the filters that are significantly different and are performing worse than the complete combined scheme (note that in the transient situation that is being considered, $\mathbf{w}_{eq}[k]$ will be essentially similar to the weights of the fastest filters). To be more precise, we replace the adaption rule for the weights of the i-th filter to be

$$\mathbf{w}_i[k+1] = \alpha \left(\mathbf{w}_i[k] + \mu_i e_i[k] \mathbf{x}[k] \right) + (1-\alpha) \mathbf{w}_{eq}[k] \tag{15}$$

if $e^2_{i,f}[k]/e^2_{eq,f}[k] > \gamma$, where $\gamma > 1$ and $e^2_{i,f}[k]$ and $e^2_{eq,f}[k]$ are filtered versions of the corresponding quadratic errors, $e^2_i[k]$ and $e^2_{eq}[k]$:

$$\begin{aligned} e^2_{i,f}[k] &= 0.9\, e^2_{i,f}[k-1] + 0.1\, e^2_i[k] \\ e^2_{eq,f}[k] &= 0.9\, e^2_{eq,f}[k-1] + 0.1\, e^2_{eq}[k] \end{aligned} \tag{16}$$

The use of these short memory exponentially weighted averages, together with the condition $e^2_{i,f}[k]/e^2_{eq,f}[k] > \gamma$, guarantees that weight transfer is only applied to those filters whose current behavior is significantly worse than that of the overall combination.

[1] Imposing $v_i[k] < \epsilon$ plays a similar role to restriction $a[k] \in [-4, 4]$ for the CLMS case.

Regarding parameter α in (15), it indicates how much of filter $\mathbf{w}_{eq}[k]$ is transferred to each component filter. Consequently, the permissible values for this parameter are $0 < \alpha < 1$.

B. Speeding up the adaption of the mixing coefficients

A second improvement of the M-CLMS filter is to include a momentum term in equation (14) to speed up the learning of the $a_i[k]$ parameters

$$a_i[k+1] = a_i[k] - \frac{\mu_a}{2}\frac{\partial e_{eq}^2[k]}{\partial a_i[k]} + \rho\left(a_i[k] - a_i[k-1]\right) \tag{17}$$

This also reduces the negative effect produced by factor $e_i[k] - e_{eq}[k]$ in (14) being close to 0 (what usually occurs during long stationary intervals), that slows down the learning of the mixing parameters.

M-CLMS performance has shown to be quite insensitive to the selection of speeding up parameters. Their dependence on the particular scenario in which the filter is being applied is very low, and all of them have easy design rules. In general, $\alpha = 0.8$, $\gamma = 2$ and $\rho = 0.5$ are good choices, and these are the settings we have used in all the simulations in this paper.

IV. Using a Different Mixing Parameter for Each Weight: The D-CLMS Algorithm

One of the drawbacks of CLMS when compared to other proposals in the LMS literature [9] is that it uses a common mixing parameter for all the weights of the filter. The D-CLMS algorithm is an extension of CLMS that introduces a different combination parameter for each weight, thus allowing, if necessary, to have a quick adaption for some coefficients of the adaptive filter, while keeping a low step for the rest.

To decouple the combination of the different weights we transform equation (2) into

$$\mathbf{w}_{eq}[k] = \mathbf{V}[k]\mathbf{w}_1[k] + (\mathbf{I} - \mathbf{V}[k])\mathbf{w}_2[k] \tag{18}$$

where $\mathbf{I}$ is the identity matrix and $\mathbf{V}[k]$ is a diagonal matrix that includes the M coefficients $v_m[k]$:

$$\mathbf{V}[k] = \text{diag}\left\{v_1[k], v_2[k], \cdots, v_M[k]\right\} \tag{19}$$

As for the CLMS filter, $v_m[k]$ are activated using sigmoids, $v_m[k] = \text{sgm}(a_m[k])$.

The output and the error of the resulting D-CLMS filter are given by

$$y_{eq}[k] = \sum_{m=1}^{M} v_m[k] w_{1m}[k] x_m[k] + (1 - v_m[k]) w_{2m}[k] x_m[k] \tag{20}$$

$$e_{eq}[k] = d[k] - y_{eq}[k] \tag{21}$$

where $w_{1m}[k]$, $w_{2m}[k]$ and $x_m[k]$ are the m-th components of vectors $\mathbf{w}_1[k]$, $\mathbf{w}_2[k]$ and $\mathbf{x}[k]$, respectively.

The corresponding rules for the adaptation of the combination parameters result in this case:

$$a_m[k+1] = a_m[k] - \frac{\mu_a}{2} \frac{\partial e_{eq}^2[k]}{\partial a_m[k]} = a_m[k] + \mu_a e_{eq}[k] \left(w_{1m}[k] - w_{2m}[k] \right) x_m[k] v_m[k] (1 - v_m[k]), \; m = 1, \ldots, M \tag{22}$$

Usually, the step size μ_a needs to be adjusted to a value higher than that used by CLMS, to compensate for the smaller average value of these updates. It should be noted, however, that the learning of the mixing parameters is more noisy than in the CLMS case, due to the fact that less information is used to adapt each $a_m[k]$. Finally, we also restrict $a_m[k]$ values to be within $[-4, 4]$ to prevent the algorithm from paralysis.

The advantages of D-CLMS are similar to those of stochastic variable step algorithms that use a different adaption step for each coefficient:

- a lower tracking error if some coefficients of the optimal solution remain unchanged, and
- a reduced sensitivity to eigenvalue spread in matrix $\mathbf{R} = E\left\{\mathbf{x}[k]\mathbf{x}^T[k]\right\}$. Note, that, since D-CLMS does not intend to whiten the inputs to the filter, this advantage will be very limited unless $\mathbf{R}$ is a diagonal matrix.

The main steps of the D-CLMS algorithm have been itemized in Table II.

As we did for the CLMS and M-CLMS algorithms, to make D-CLMS fully effective it is convenient to include procedures for weight transfer and a momentum term for adapting $a_m[k]$. Inclusion of the momentum term is straightforward. Regarding weight transfer, it only makes sense to transfer a portion of the weights of the fast filter to the slow one. Besides, to get the maximum advantage from D-CLMS, it is interesting to have the possibility of transferring just some coefficients of the fast filter (those having their v_m close to one). Consequently, weight transfer is done from the combined filter

$$\mathbf{w}_2[k+1] = \alpha \left(\mathbf{w}_2[k] + \mu_2 e_2[k] \mathbf{x}[k] \right) + (1 - \alpha) \mathbf{w}_{eq}[k] \tag{23}$$

Again, some conditions must be satisfied so that the weight transfer makes sense. In particular, to prevent the slow filter from tracking the fast filter, weight transfer must only be applied if the error of the combination is significantly lower than that of the slow filter: if $e^2_{2,f}[k]/e^2_{eq,f}[k] > \gamma$, with $\gamma > 1$. These parameters have a similar interpretation to their M-CLMS counterparts and, consequently, we will use the same settings, $\alpha = 0.8$, $\gamma = 2$ and $\rho = 0.5$, for both algorithms.

V. COMPUTATIONAL COMPLEXITY OF THE PROPOSED ALGORITHMS

In this section we analyze the computational complexity of the proposed combination algorithms. Since the number of additions is, in all the cases, of the same order of magnitude than the multiplications, we will consider just the number of real multiplications and divisions.

Regarding the real LMS filter, it is well known that this algorithm requires $2M + 1$ multiplications to calculate the output of the filter and to update its weights, where M is the length of the adaptive filter. CLMS combines two LMS filters, thus initially requiring $4M + 2$ multiplications, and 6 more products are needed to compute the output of the filter and to update the mixing coefficients (Eqs. (3) and (5), respectively). Note that, for certain applications, it is enough to compute the output of the filter, while others may require to explicitly calculate $\mathbf{w}_{eq}[k]$, which, according to (2), needs $2M$ products. Weight transfer requires at each iteration the evaluation of averages of the quadratic errors of the slow and the overall filter (6 multiplications), as well as to compute $e^2_{2,f}[k]/e^2_{eq,f}[k]$. If conditions for weight transfer are satisfied (what occurs only in very particular situations), application of (6) needs $2M$ extra products. Finally, only one more multiplication is required by the momentum method. Obviously, the evaluation of the sigmoid activation function requires some extra effort, but it could be easily implemented in an efficient manner using a look-up table.

For a M-CLMS filter combining L filters, the number of multiplications required to adapt the components is $2LM + L$. M-CLMS output evaluation (Eq. (10)) and the update of the mixing coefficients (Eq. (14)) imply L and $2L + 1$ products, respectively, and LM more multiplications are needed if the weights are calculated explicitly. The weight transfer method needs $3(L + 1)$ products and L divisions at each iteration and, in the very unlikely case in which all the filters but one satisfy the conditions for weight transfer, it would require $2(L-1)M$ extra multiplications. One product is needed to apply the momentum term to each mixing coefficient, what gives a total of L. Again, the softmax evaluation is not well suited for a real time implementation, but it could be replaced by a look-up table containing the values of the exponential function. Even in this case, L divisions should be carried out.

Like CLMS, the D-CLMS filter combines just two LMS filters, but it has M mixing coefficients, one for each tap of the filter. The successive adaptation of the LMS components, and the application of (20) and (22) gives a total of $9M+3$ products. This number is increased to $10M+3$ if the explicit calculation of $\mathbf{w}_{eq}[k]$ is needed. Finally, $2M+6$ products and one division are needed for weight transfer, while the inclusion of a momentum term requires M more products.

We have summarized the above figures in Table III. The most important conclusions about the computational requirements of the proposed algorithms are:

- The complexity of all the algorithms increases linearly with the number of taps of the adaptive filter ($O(M)$), and for the basic CLMS, M-CLMS and D-CLMS algorithms, it is roughly twice, L times, and 4.5 times the computational complexity of a basic LMS filter.
- The weight transfer procedure may increase significantly the computational complexity of the proposed algorithms. Note, however, that this method was designed to accelerate the convergence of slow filters in a very particular situation (after abrupt changes in the solution), but this is not necessary in order to achieve the main goal of the combined schemes: to put together the fast convergence of filters with a high step size, and the low residual error of slow LMS filters. Furthermore, weight transfer, if used, will occur only in a very limited number of iterations, making the computational increase less important in the practice.

VI. EXPERIMENTS

We have carried out an extensive number of experiments with the combined algorithms presented in this paper. Although these algorithms could be used in any adaptive filtering application, we show here just plant identification and noise cancellation problems.

In plant identification [4], the adaptive filter is used to model an unknown plant $\mathbf{h}$ (see block diagram of Figure 2(a)) from some measurements at its inputs and outputs. The difference between the measured output (d) and the actual output of the filter (y) constitutes the error signal minimized by the filter. After optimization, the weight vector $\mathbf{w}$ serves to model the real plant $\mathbf{h}$. If the plant is non stationary and the adaptive filter is well designed, $\mathbf{w}$ is able to track the plant changes.

Figure 2(b) represents a widely used double sensor noise cancellation configuration. Signal d, received by the primary sensor, consists of an information s contaminated with a noise n, that is uncorrelated with s. A reference signal x, correlated with n, is also available, and it is used as the input to the adaptive filter. The output of the filter is subtracted from the primary signal to obtain the error used for the optimization of the filter. It is easy to show ([5], [6]) that the error signal is (ideally) a noise free version of s (i.e., $e = \hat{s}$). A real situation where this double sensor configuration is of application is the case of a speaker using a microphone. The microphone receives the speech signal (s), possibly contaminated with some noise. A second sensor can be placed away from the speaker, so that it only receives noise. Obviously, this is an adaptive problem since the speaker could very well vary the volume of the speech, move towards or away from the source of noise, the statistics of the noise may change, etc.

A. Plant Identification Experiments

In this subsection we use a plant identification example with artificial data to compare the performance of M-CLMS and D-CLMS filters versus CLMS and the previous variable step algorithm proposed in [9]: Variable-Step LMS (VS-LMS). We have carried out experiments with other stochastic variable step algorithms and have chosen VS-LMS as the representative of them because it has generally obtained the best performance provided that appropriate values are used for VS-LMS parameters. In addition, VS-LMS has the advantage of using a different step for each weight, what makes fair the comparison with D-CLMS.

VS-LMS proceeds by increasing/decreasing the value of the step sizes depending on the signs of each of the components of the gradient vector ($\nabla_i e^2[k] = e[k]x_i[k]$) during some of the last iterations. To be more specific, after m_0 alternate signs, the corresponding step size is divided by c ($c > 1$). On the contrary, if m_1 consecutive identical signs occur, it is multiplied by c. Besides, adaption steps are limited to the interval $[\mu_{\min}, \mu_{\max}]$, where $\mu_{\max}$ guarantees the stability of the algorithm (for a more detailed description of VS-LMS, please refer to [9]).

The plant in our experiments is a 16-tap transversal filter, with input $x[k]$ being a white, Gaussian, zero mean process, whose variance is selected to get $E\{\|\mathbf{x}[k]\|_2^2\} = 1$. The desired output of the adaptive filter is the actual output of the plant with added measurement noise $n[k]$ (see Fig. 2(a)), which is a process similar to $x[k]$ but with variance $\sigma_n^2 = 10^{-2}$.

The coefficients of the plant are initially selected as random values between -1 and 1, and changes are introduced at different speeds using the random-walk model:

$$\mathbf{h}[k+1] = \mathbf{h}[k] + \mathbf{q}[k] \tag{24}$$

where $\mathbf{q}[k]$ is an i.i.d. random zero-mean vector, with diagonal covariance matrix $\mathbf{Q} = E\{\mathbf{q}[k]\mathbf{q}^T[k]\}$, whose trace is related to the degree of non stationarity of the plant. When using this model, we will assume that changes affect only the first four coefficients of the plant, with the corresponding entries of $\mathbf{q}[k]$ being i.i.d. and Gaussian distributed.

We use the Mean Square Deviation between the real plant and the adaptive filter to measure the performances of the algorithms

$$\text{MSD}[k] = \|\mathbf{h}[k] - \mathbf{w}[k]\|_2^2 \tag{25}$$

All results are averaged MSDs over 1000 independent runs.

1) M-CLMS Performance: To study M-CLMS behavior we will select first an example where the plant changes with two different speeds, $\text{Tr}(\mathbf{Q}_1) = 10^{-6}$ and $\text{Tr}(\mathbf{Q}_2) = 10^{-5}$, during $25000 < k \leq 40000$ and $75000 < k \leq 90000$, respectively. Furthermore, at $k = 50000$ all the weights of the solution change abruptly (new random values are taken from $[-1, 1]$).

In this case, we combine four LMS filters with adaption steps $\mu_1 = 0.1$, $\mu_2 = 0.03$, $\mu_3 = 0.01$ and $\mu_4 = 0.003$. To adapt the combination we use $\mu_a = 100$, and we select "standard values" for the speeding up parameters, as discussed at the end of Section III. Finally, all weights of the LMS filters, as well as the mixing coefficients $a_i[k]$, have been initialized to zero.

Fig. 3(a) shows the behavior of the M-CLMS filter. We can see the fast convergence of this filter after abrupt transitions (at $k = 0$ and $k = 50000$), and how it combines the fast response of the μ_1 LMS filter with the low misadjustment of the μ_4 filter. This convergence would be slower if no speeding up procedures were applied. Looking at the values of the mixing parameters (Fig. 3(b)), we can also see how after $k = 0$ and $k = 50000$ the M-CLMS filter gradually changes from the fastest to the slowest filter. During tracking periods ($25000 < k \leq 40000$ and $75000 < k \leq 90000$) we can check (looking at either MSD or $v_i[k]$ graphics) that the M-CLMS filter operates at each moment like the best component LMS filter in the combination. So, we can conclude that M-CLMS puts together fast convergence, low misadjustment, and good tracking properties for changes at different speeds.

Fig. 5(a) compares the MSDs of the M-CLMS and VS-LMS filters. For VS-LMS we have used $\mu_{\min} = \mu_4$, $\mu_{\max} = 2\text{tr}\{\mathbf{R}\}/3 = 2/9$ (note that $\mu_{\max} > \mu_1$ what, in principle, could favor VS-LMS in fast changing situations) and $c = 1.1$. Regarding m_0 and m_1, the authors of [9] say that it is convenient to use $m_0 = m_1$ in non stationary environments. However, we observed that with this setting VS-LMS performed only slightly better than a fast LMS filter, achieving a MSD of around -20 dB during most of the example. As shown in the figure, selection of $m_1 = m_0 + 1 = 2$ favors the decreasing of the adaption steps and, after abrupt changes, VS-LMS converges slowly towards MSDs around -35 dB. During the tracking interval $75000 < k \leq 90000$ adaption steps remain too low and VS-LMS performance is much poorer than that of M-CLMS.

We can conclude that selection of VS-LMS parameters m_0 and m_1 is subject to a speed vs precision tradeoff similar to that of the step size in LMS filters, and optimal values are highly dependent on the scenario in which the filter is applied. M-CLMS is effective at breaking this compromise, and the selection of its parameters is easier.

2) D-CLMS Performance: We study next D-CLMS behavior using the same scenario considered in the previous subsection. In this case, we will combine only the fastest and the slowest LMS filters, with step sizes $\mu_1 = 0.1$ and $\mu_4 = 0.003$. The adaption constant for the mixing parameters is $\mu_a = 300$. Speeding up parameters have been fixed to the values recommended at the end of Section IV. Finally, LMS weights and the mixing coefficients have been initialized with zeros.

Fig. 4(a) represents MSD evolution for the two component filters and for their D-CLMS combination. D-CLMS performs, at every moment, at least as well as the best component filter, retaining the high adaption speed of the μ_1 filter, and the low MSD misadjustment of the slow filter in stationary situations. In addition, since during tracking intervals only four of the taps of the plant are affected by changes, D-CLMS is able to identify these weights (see in Fig. 4(b) the evolution of $v_1[k]$ and $v_5[k]$, corresponding to a changing and a stationary weights), using a fast adaptation to track them, while keeping the slow (and more precise) filter for the rest. As a result, D-CLMS in these situations achieves a lower MSD than any of the LMS filters. Finally, the beneficial effects of using speeding up procedures are well appreciated in the abrupt transitions at $k = 0$ and $k = 50000$, where D-CLMS achieves the MSD corresponding to a μ_4 LMS filter before this filter itself does.

Comparison with VS-LMS can be done in similar terms to those of the M-CLMS filter. Fig. 5(b) compares D-CLMS with VS-LMS for $\mu_{\min} = \mu_4$, $\mu_{\max} = 2/9$, $c = 1.1$, and $m_1 = m_0 + 1 = 2$. The discussion about VS-LMS performance that we did when comparing it to the M-CLMS filter, applies also to this case: unlike VS-LMS, D-CLMS breaks the speed vs precision tradeoff.

3) Tracking Performance of the Combined Filters: Recall that the main reason for proposing M-CLMS and D-CLMS was to improve the tracking capabilities of the CLMS scheme. Thus, in this subsection we will offer a detailed analysis of the tracking performance of the three filters. In Fig. 6(a) we have depicted the tracking errors of the three combined schemes for different degrees of non stationarity. As before, M-CLMS combines four LMS filters with step sizes $\mu_1 = 0.1$, $\mu_2 = 0.03$, $\mu_3 = 0.01$ and $\mu_4 = 0.003$, using $\mu_a = 100$ to adapt the combination. D-CLMS and CLMS combine just the fastest and the slowest of these LMS filters, adapting their mixing coefficients with $\mu_a = 300$ and $\mu_a = 100$, respectively. Since, in this case, we are just interested on analyzing steady-state tracking error, speeding up methods were not applied.

To measure tracking error, the random-walk model (24) was used permanently, averaging the residual error over 10000 iterations after the steady-state was achieved, and over 1000 independent runs. To get a more illustrative representation, all MSDs were normalized by the tracking error of the μ_2 LMS, so that a value below 0 dB implies that the combined schemes perform better than the μ_2 LMS.

First of all, we can see that all combined schemes generally offer better tracking properties than the LMS filter. In particular, CLMS outperforms the μ_2 LMS filter for most values of Tr$(\mathbf{Q})$, and in some cases by near 10 dB. This improvement is even more important for D-CLMS and M-CLMS. As explained in Section III, combined schemes retain the tracking capabilities of their components. Given that M-CLMS incorporates two more filters than CLMS, it achieves a lower tracking error in situations for which μ_1 is too high and μ_4 too low. Regarding D-CLMS, its advantage comes from the fact that only 4 of the 16 taps of the plant are suffering changes, a situation for which D-CLMS is specially suited. We have checked that when all the taps of the solution change at the same speed, D-CLMS and CLMS tracking properties are similar.

4) Combination of Transform-Domain Adaptive Filters: As we have seen, our combination schemes are able to improve the convergence rate vs residual misadjustment tradeoff of adaptive filters, as well as their tracking capabilities. However, if the component filters are sensitive to an eigenvalue spread in the autocorrelation matrix $\mathbf{R}$ (such as the basic LMS), their combination will also suffer the same problem. This drawback can be easily solved by recurring to more sophisticated filters such as transform-domain LMS or RLS, while keeping the advantages of the combination approach[2].

In this subsection we will consider the combination of transform-domain LMS filters [21], [22]. Transform-domain LMS filters apply an unitary transformation that is able to decorrelate to some extent the components of the input vector $\mathbf{x}[k]$, thus reducing the sensitivity to a non-diagonal $\mathbf{R}$. In particular, we will use the implementation of the Discrete Cosine Transform (DCT) LMS filter given in [3, p. 583]. It is important to remark that, although DCT LMS requires some extra computation for obtaining the decorrelated input, this calculation needs to be carried out only once by the combined schemes. Consequently, the computational complexity increase introduced by the combined schemes is less important in this case than when combining time-domain filters.

[2]Note that the combination schemes proposed in the paper could be used, with no essential modifications, with any other types of adaptive filters as the basic components.

Fig. 7 illustrates the convergence of two DCT LMS filters with step sizes $\mu_1 = 0.003$ and $\mu_4 = 0.0001$ in a stationary situation, for the 16-tap plant considered in previous subsections. In this case, the adaptive filter input is a colored process generated by passing $x[k]$ through a filter with transfer function $\sqrt{1-a^2}/(1-az^{-1})$, $a = 0.8$. Using time-domain LMS filters in this situation would result in a much slower convergence. If we use a combination of the two previous filters, that we call DCT CLMS (as usual, $\mu_a = 100$, $\alpha = 0.8$, $\gamma = 2$ and $\rho = 0.5$), we are again able to put together the faster convergence of the μ_1 DCT LMS filter and the smaller residual MSD of the slow filter. Furthermore, the weight transfer procedure is also effective at speeding up the convergence of the filter with step μ_4, what results in an improved convergence of the combination.

Finally, in Fig. 6(b) we analyze the tracking properties of the multiple and decoupled schemes in the DCT-domain (DCT M-CLMS and DCT D-CLMS), using standard values for their parameters, and including two additional DCT LMS filters with steps $\mu_2 = 0.001$ and $\mu_3 = 0.0003$ for the multiple scheme. Again, the three combined schemes show improved tracking properties over the μ_2 DCT LMS filter, with DCT M-CLMS and DCT D-CLMS obtaining the best results for a wide range of Tr$(\mathbf{Q})$.

B. Noise Cancellation Experiments

We present now several noise cancellation simulations with one synthetic and one real signal. We will give results only for the M-CLMS and CLMS algorithms. D-CLMS has been excluded from this problem because its performance was similar to that of the CLMS filter, but somehow more noisy.

In all the simulations, $n[k]$ is a Gaussian, white and zero mean noise with an average power of 0.5. $x[k]$, the reference signal, is obtained from $n[k]$ as the output of an all-pole filter $H(z)$ (see Fig. 2(b)). In the first half of the simulations ($1 \leq k \leq 10000$) the transfer function is

$$H(z) = \frac{1}{1 - 0.1485z^{-1} + 0.256z^{-2} - 0.1638z^{-3} + 0.1764z^{-4}}$$

Then, the coefficients of $H(z)$ linearly vary from $k = 10001$ to $k = 11000$, where the new filter, that is kept unchanged for the rest of the simulation, has only three poles

$$H(z) = \frac{1}{1 + 0.9z^{-1} - 0.4286z^{-2} - 0.192z^{-3}}$$

The primary signal is given by the sum of a carrier of information $s[k]$ of average power 0.5, contaminated with noise $n[k]$

$$d[k] = s[k] + \lambda[k]n[k] \tag{26}$$

where $\lambda[k] = 1$ (SNR $= 0$ dB) during all the example, except for $3000 < k \leq 6000$, when it linearly increases towards $\lambda[k] = \sqrt{10}$ (SNR $= -10$ dB) and then, instantaneously, it recovers its original value.

The performance of the adaptive filters will be measured as the square of the estimation error ($(s[k] - \hat{s}[k])^2 = (s[k] - e[k])^2$) averaged over 1000 independent realizations of the noise.

In these experiments M-CLMS combines three 5-tap LMS filters, with adaption steps $\mu_1 = 0.05$, $\mu_2 = 0.01$ and $\mu_3 = 0.005$. The learning rate for the mixing coefficients is $\mu_a = 2$. The reason for using a smaller step is that the values of $\frac{\partial e_{eq}^2[k]}{\partial a_i[k]}$, used to adapt the mixing parameters, are about two orders of magnitude greater than in the plant identification case. Speeding up parameters are not affected by this difference, and we can use again: $\alpha = 0.8$, $\gamma = 2$ and $\rho = 0.5$. CLMS combines the fastest and the slowest filters in the M-CLMS scheme, keeping the same settings for the rest of the parameters.

Fig. 8(a) and (b) represent the cancellation error when $s[k]$ is a sine wave: $s[k] = \sin(\pi k/10)$. In Fig. 8(a) we see that, following the SNR increment at $k = 6000$, both CLMS and M-CLMS very fast converge towards an error of -15 dB (corresponding to the μ_1 filter). Then, their errors continue decreasing, until they reach the final misadjustment of the μ_3 filter. During $3000 < k \leq 6000$, when the SNR in the primary signal decreases, M-CLMS clearly outperforms CLMS, as a consequence of the lower error achieved by the μ_2 filter, not present in the CLMS combination.

Fig. 8(b) represents the convergence after the change of $H(z)$. In this case the μ_2 filter is that offering a faster convergence, giving again some advantage to the M-CLMS filter. In any case, the convergence of the component filters after $k = 11000$ is very slow. The reason for this is that $x[k]$ is a colored process and the eigenvalue spread in matrix $\mathbf{R} = E\left\{\mathbf{x}[k]\mathbf{x}^T[k]\right\}$ is higher for the new $H(z)$.

Finally, we present results when $s(k)$ is a register of English speech sampled at 7,35 kHz and quantized with 16 bits. The signal has been rescaled to obtain the desired average power. Fig. 8(c) shows cancellation error for variable SNR in the primary signal. The behavior of the adaptive filters in this case is basically the same as for the sine wave, although some differences appear due to the distinct nature of $s[k]$. Fig. 8(d) depicts convergence after $k = 11000$. Again, M-CLMS has a faster convergence than CLMS.

VII. CONCLUSIONS

Following the fundamental idea of adaptively combining different adaptive filters to alleviate the speed vs precision compromise of LMS filters, in this paper we have presented two new algorithms that extend the ideas of the combination of one fast filter and one slow filter along two different directions. On the one hand, M-CLMS algorithm implements the combination of any number of LMS filters with different adaption steps, what improves the tracking properties of CLMS. On the other hand, D-CLMS uses a different mixing parameter for each coefficient of the filter, what gives some advantage when the optimal solution varies but some coefficients remain unaltered. Speeding up procedures have been designed to make the previous schemes fully effective. The computational demand of the new algorithms grows linearly with the number of filters and with their length.

Simulations have been carried out to illustrate the behavior of M-CLMS and D-CLMS, and their improved tracking properties over CLMS. Comparisons with previous variable step approaches have also shown the superior performance of the combined schemes. It has also been explained how the combination schemes could be used, with no essential modifications, to combine filters in transform-domain, getting a reduced sensitivity to colored inputs, while keeping their advantages in terms of improved rate of convergence vs misadjustment tradeoff and tracking capabilities.

Needless to say, the use of the proposed combination algorithms in other adaptive filtering applications represents an immediate possibility of getting more benefits from the above ideas. Work is also in progress to extend the methods we presented in this paper to other adaptive filter structures.

REFERENCES

[1] B. Widrow and M. E. Hoff, "Adaptive Switching Circuits," *IRE WESCON Conv. Rec.*, pp. 96–104, 1960.

[2] S. Haykin, *Adaptive Filter Theory (4th. ed.).* Upple Saddle River, NJ: Prentice-Hall, 2002.

[3] A. H. Sayed, *Fundamentals of Adaptive Filtering*. NY: Wiley, 2003.

[4] M. H. Hayes, *Statistical Digital Signal Processing and Modeling*. NY: Wiley, 1996.

[5] B. Widrow et al., "Adaptive Noise Cancelling: Principles and Applications," *Proc. IEEE*, vol. 63, pp. 1692–1716, 1975.

[6] M. B. Yeary and N. C. Griswold, "Adaptive IIR Filter Design for Single Sensor Applications," *IEEE Trans. on Instrumentation and Measurement*, vol. 51, pp. 259–267, 2002.

[7] B. Widrow, J. M. McCool, M. G. Larimore, and C. R. J. Jr., "Stationary and Nonstationary Learning Characteristics of the LMS Adaptive Filter," *Proc. IEEE*, vol. 64, pp. 1151–1162, 1976.

[8] A. Cichoki and R. Unbehauen, *Neural Networks for Optimization and Signal Processing*. West Sussex, UK: Wiley, 1993.

[9] R. W. Harris, D. M. Chabries, and F. A. Bishop, "Variable Step (VS) Adaptive Filter Algorithm," *IEEE Trans. Acoustics, Speech and Signal Proc.*, vol. 34, pp. 305–316, 1986.

[10] V. J. Mathews and Z. Xie, "A Stochastic Gradient Adaptive Filter with Gradient Adaptive Step Size," *IEEE Trans. on Signal Proc.*, vol. 41, pp. 2075–2087, 1993.

[11] K.-R. Müller, A. Ziehe, N. Murata, and S.-I. Amari, "On-line Learning in Switching and Drifting Environments with Application to Blind Source Separation," in *On-Line Learning in Neural Networks* (D. Saad, ed.), pp. 93–110. Cambridge, UK: Cambridge University Press, 1998.

[12] M. Martínez-Ramón, J. Arenas-García, A. Navia-Vázquez, and A. R. Figueiras-Vidal, "An Adaptive Combination of Adaptive Filters for Plant Identification," in *Proc. 14th Intl. Conf on DSP* (A. N. Skodras and A. G. Constantinides, eds.), pp. 1195–1198, Santorini, Greece, July 2002.

[13] J. Arenas-García, A. R. Figueiras-Vidal and A. H. Sayed, "Steady-state Performance of Convex Combinations of Adaptive Filters," in *Proc. ICASSP'05*, vol. IV, pp. 33–36, Philadelphia, PA, 2005.

[14] J. Arenas-García, A. R. Figueiras-Vidal, and A. H. Sayed, "Mean-square performance of a convex combination of two adaptive filters," to appear in *IEEE Trans. on Signal Proc.*

[15] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton, "Adaptive Mixtures of Local Experts," *Neural Computation*, vol. 3, no. 1, pp. 79–87, 1991.

[16] A. C. Singer and M. Feder, "Universal Linear Prediction by Model Order Weighting," *IEEE Trans. on Signal Proc.*, vol. 47, pp. 2685–2700, 1999.

[17] S. S. Kozat and A. C. Singer, "Multi-stage Adaptive Signal Processing Algorithms," in *Proc. 2000 IEEE Sensor Array and Multichannel Signal Processing Workshop*, pp. 380–384, Cambridge, MA, March 2000.

[18] J. Arenas-García, M. Martínez-Ramón, V. Gómez-Verdejo, and A. R. Figueiras-Vidal, "Multiple Plant Identifier via Adaptive LMS Convex Combination," in *Proc. 2003 IEEE Intl. Symp. on Intelligent Signal Proc.* (C. Molina et. al., eds.), pp. 137–142, Budapest, Hungary, Sep. 2003.

[19] J. Arenas-García, V. Gómez-Verdejo, M. Martínez-Ramón, and A. R. Figueiras-Vidal, "Separate-Variable Adaptive Com-

bination of LMS Adaptive Filters for Plant Identification," in *Proc. of the 13th IEEE Intl. Workshop on Neural Networks for Signal Proc.*, pp. 239–248, Toulouse, France, Sept. 2003.

[20] B. Hassibi, A. H. Sayed, and T. Kailath, "H^∞ Optimality of the LMS Algorithm," *IEEE Trans. on Signal Proc.*, vol. 44, pp. 267–280, 1996.

[21] S. S. Narayan, A. M. Peterson, and M. J. Narashima, "Transform domain LMS algorithm," *IEEE Trans. Acoust., Speech and Signal Proc.*, vol. 31, pp. 609–615, 1983.

[22] S. Florian and N. J. Bershad, "A weighted normalized frequency domain LMS adaptive algorithm," *IEEE Trans. Acoust., Speech and Signal Proc.*, vol. 38, pp. 788–798, 1988.

Jerónimo Arenas-García (S'00, M'04) was born in Seville, Spain, in 1977. He received the Telecommunication Engineer degree in 2000 from Universidad Politecnica de Madrid (ranked number 1; National Award to graduation), and his Ph.D. degree from Universidad Carlos III de Madrid, where he is now an Assistant Professor. His present research interests are focused in the fields of Adaptive Signal Processing, Machine Learning and their applications.

Vanessa Gómez-Verdejo was born in Madrid, Spain, in 1979. She received the Telecommunication Engineer degree in 2002 from Universidad Politécnica de Madrid, Madrid, Spain. Actually, she is pursuing the Ph.D. degree at the Department of Signal Theory and Communications, Universidad Carlos III de Madrid. Her present research interests are centered in the fields of adaptive signal processing and machine learning, mainly NN ensembles, and their applications.

Anibal R. Figueiras-Vidal (S'74, M'76, SM'84) received the Telecommunication Engineer degree from Universidad Politécnica de Madrid, Madrid, Spain, in 1973 (ranked number 1; National Award to graduation) and the Doctor degree (Honors) from Universidad Politécnica de Barcelona, Barcelona, Spain, in 1976. He is a Professor of signal theory and communications with Universidad Carlos III, Madrid. His research interests are digital signal processing, digital communications, neural networks, and learning theory. He has (co)authored more than 300 journal and conference papers in these areas. Dr. Figueiras received an "Honoris Causa" Doctorate degree in 1999 from Universidad de Vigo, Vigo, Spain. He is currently General Secretary of the Royal Academy of Engineering of Spain.

List of Tables

List of Figures

1.- Inputs: $\mu_a, \mu_i,\ \ i = 1, \ldots, L$

2.- Initialization: $a_i[0] = 0$, $v_i[0] = 1/L$, $\mathbf{w}_i[0] = \mathbf{0}$, $\ \ i = 1, \ldots, L$

3.- Loop $k = 0, 1, 2, \ldots$

$y_i[k] = \mathbf{w}_i^T[k]\mathbf{x}[k],\ \ i = 1, \ldots, L$

$e_i[k] = d[k] - y_i[k],\ i = 1, \ldots, L$

$y_{eq}[k] = \sum_{i=1}^{L} v_i[k] y_i[k],\ \ e_{eq} = d[k] - y[k]$

$\mathbf{w}_i[k+1] = \mathbf{w}_i[k] + \mu_i[k] e_i[k] \mathbf{x}[k],\ \ i = 1, \ldots, L$

$a_i[k+1] = a_i[k] - \mu_a e_{eq}[k] \left(e_i[k] - e_{eq}[k] \right) v_i[k] |_{-\epsilon'}^{\epsilon'},\ \ i = 1, \ldots, L$

$v_i[k+1] = \mathrm{softmax} \left(a_i[k+1] \right)$

$\left(\mathbf{w}_{eq}[k+1] = \sum_{i=1}^{L} v_i[k+1] \mathbf{w}_i[k+1] \quad \text{\% Only if needed by the application} \right)$

TABLE I

1.- Inputs: $\mu_a, \mu_i,\ \ i = 1, 2$

2.- Initialization: $\mathbf{w}_1[0] = \mathbf{w}_2[0] = \mathbf{0}$, $a_m[0] = 0$, $v_m[0] = 0.5, \quad m = 1, \ldots, M$

3.- Loop $k = 0, 1, 2, \ldots$

$y_i[k] = \mathbf{w}_i^T[k]\mathbf{x}[k],\ \ i = 1, 2$

$e_i[k] = d[k] - y_i[k],\ i = 1, 2$

$y_{eq}[k] = \mathbf{w}_{eq}^T[k]\mathbf{x}[k],\ \ e_{eq} = d[k] - y[k]$

$a_m[k+1] = a_m[k] + \mu_a e_{eq}[k] \left(w_{1m}[k] - w_{2m}[k] \right) x_m[k] v_m[k] (1 - v_m[k]) |_{-4}^{4},\ \ m = 1, \ldots, M$

$\mathbf{w}_i[k+1] = \mathbf{w}_i[k] + \mu_i[k] e_i[k] \mathbf{x}[k],\ \ i = 1, 2$

$v_m[k+1] = \mathrm{sgm} \left(a_m[k+1] \right),\ \ m = 1, \ldots, M$

$w_{eq,m}[k+1] = v_m[k+1] w_{1m}[k+1] + (1 - v_m[k+1]) w_{2m}[k+1],\ \ m = 1, \ldots, M$

TABLE II

	LMS	CLMS	M-CLMS	D-CLMS
Basic Algorithm	$2M+1$	$4M+8$	$2LM+4L+1$ $L\ \div$	$9M+3$
Explicit Weights Calculation	0	$2M$	LM	M
Weight Transfer	—	$2M+6$ $1\ \div$	$2(L-1)M+3(L+1)$ $L\ \div$	$2M+6$ $1\ \div$
Momentum	—	1	L	M

TABLE III

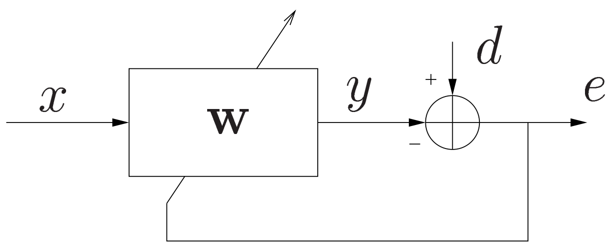
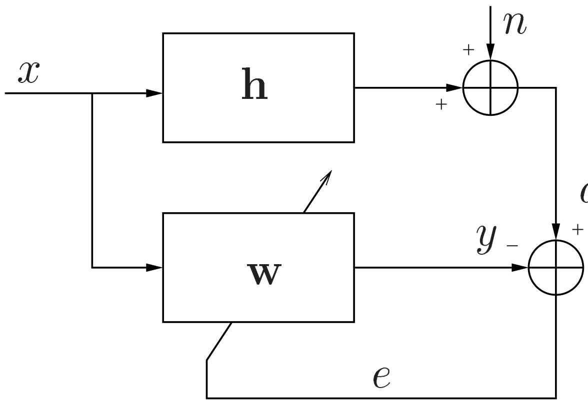
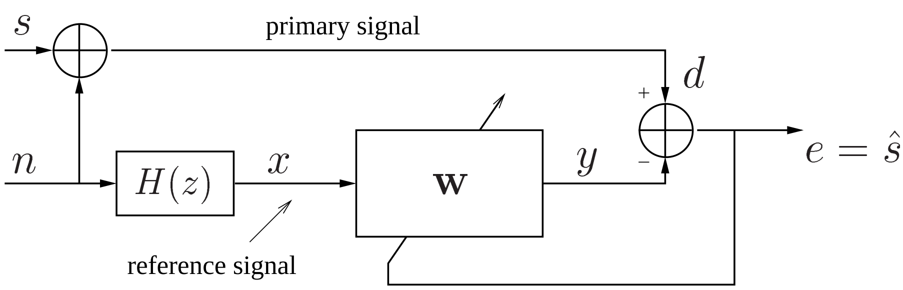


Fig. 1

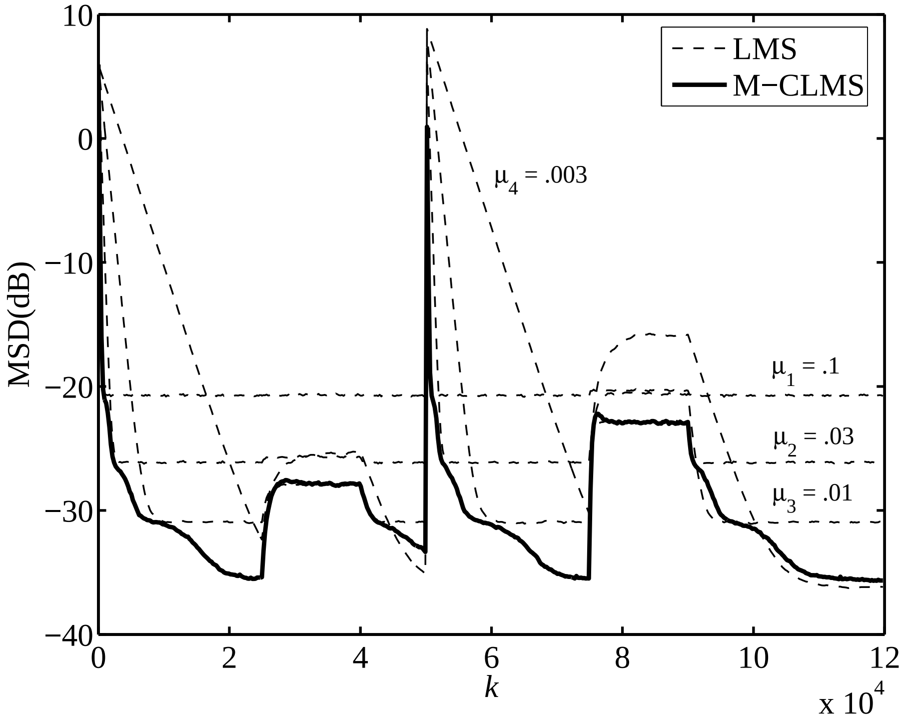


(a)



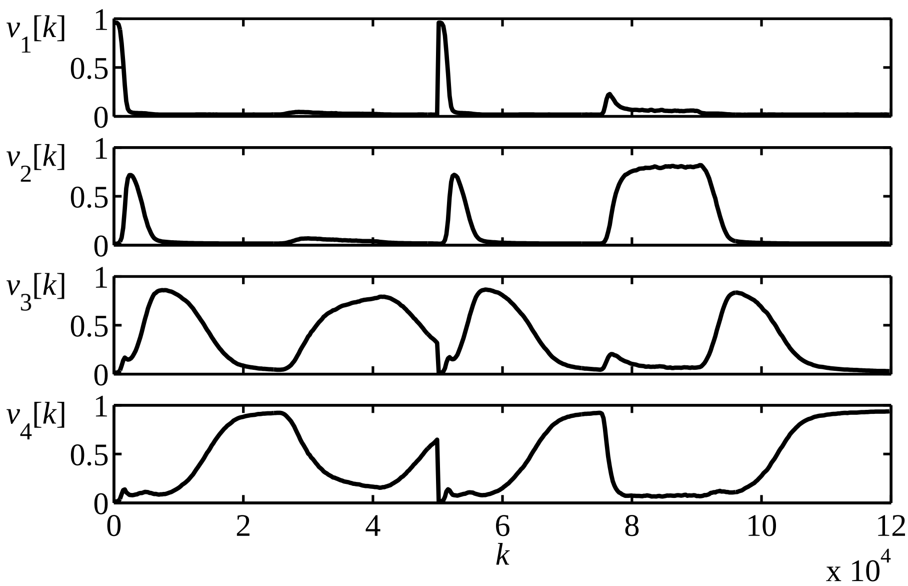
(b)

Fig. 2



LMS
M−CLMS
μ4 = .003
μ1 = .1
μ2 = .03
μ3 = .01
MSD(dB)
k
x 10^4

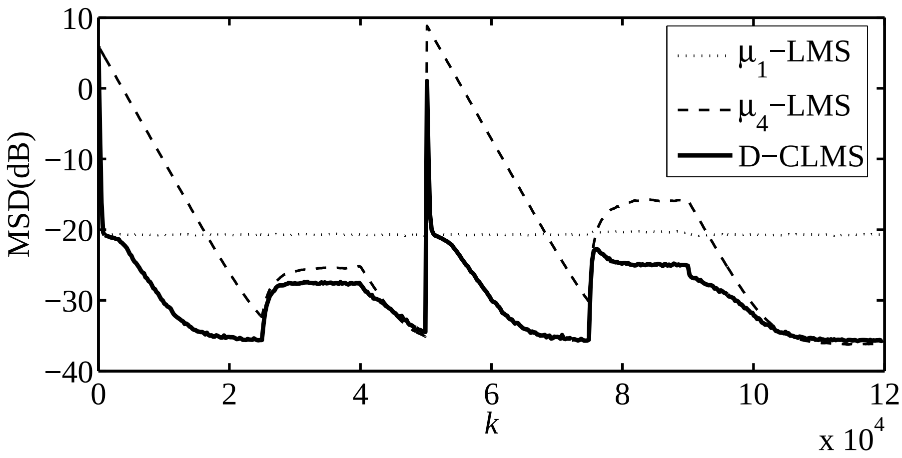
(a)



v1[k]
v2[k]
v3[k]
v4[k]
k
x 10^4

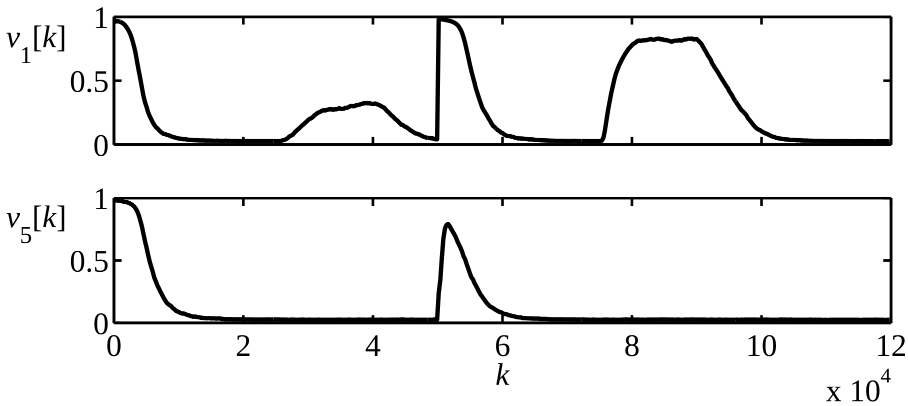
(b)

Fig. 3



μ1−LMS
μ4−LMS
D−CLMS
MSD(dB)
10
0
−10
−20
−30
−40
0
2
4
6
8
10
12
k
x 10^4

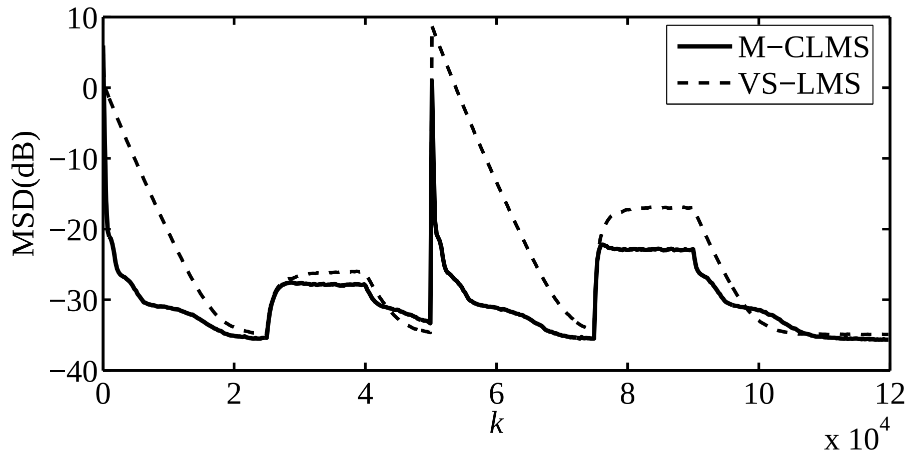
(a)



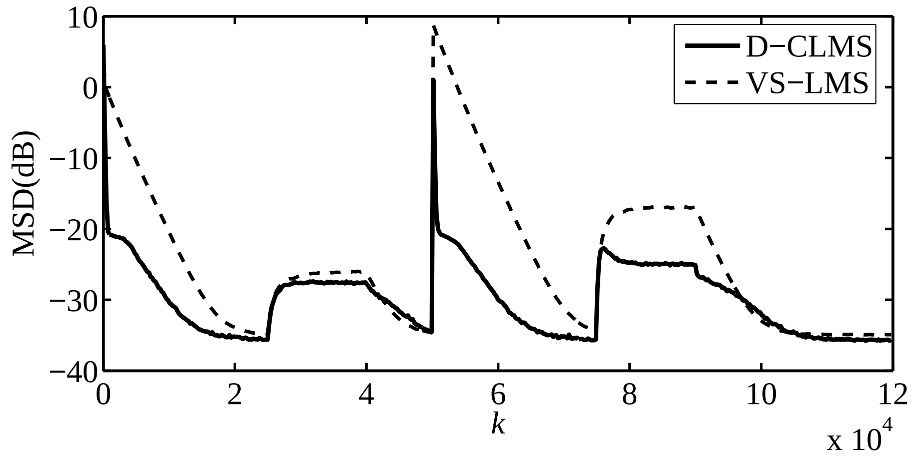
v1[k]
v5[k]
1
0.5
0
0
2
4
6
8
10
12
k
x 10^4

(b)

Fig. 4

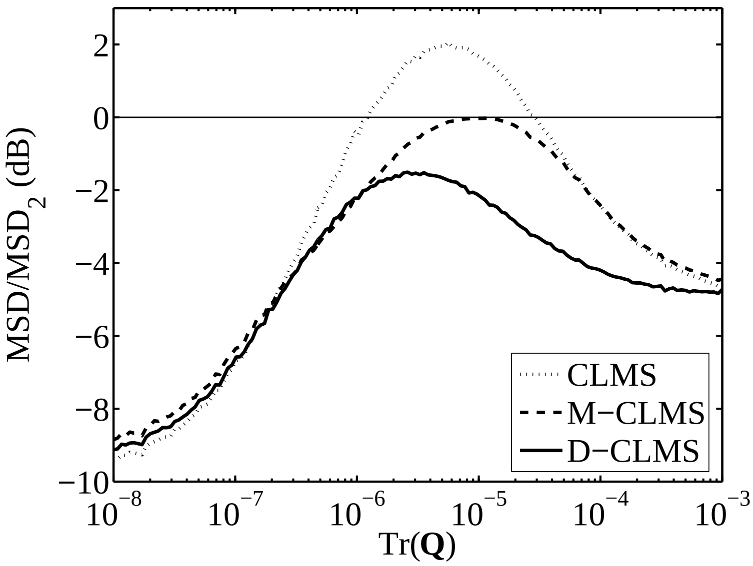

M−CLMS
VS−LMS
MSD(dB)
10
0
−10
−20
−30
−40
0
2
4
6
8
10
12
k
x 10^4

(a)

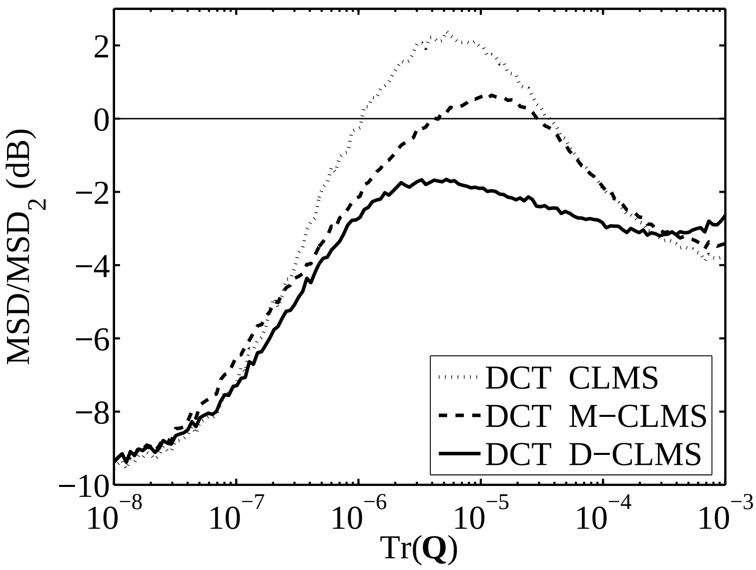

D−CLMS
VS−LMS
MSD(dB)
10
0
−10
−20
−30
−40
0
2
4
6
8
10
12
k
x 10^4

(b)

Fig. 5


MSD/MSD2 (dB)
2
0
−2
−4
−6
−8
−10
10−8
10−7
10−6
10−5
10−4
10−3
Tr(Q)
CLMS
M−CLMS
D−CLMS

(a)


MSD/MSD2 (dB)
2
0
−2
−4
−6
−8
−10
10−8
10−7
10−6
10−5
10−4
10−3
Tr(Q)
DCT CLMS
DCT M−CLMS
DCT D−CLMS

(b)

Fig. 6

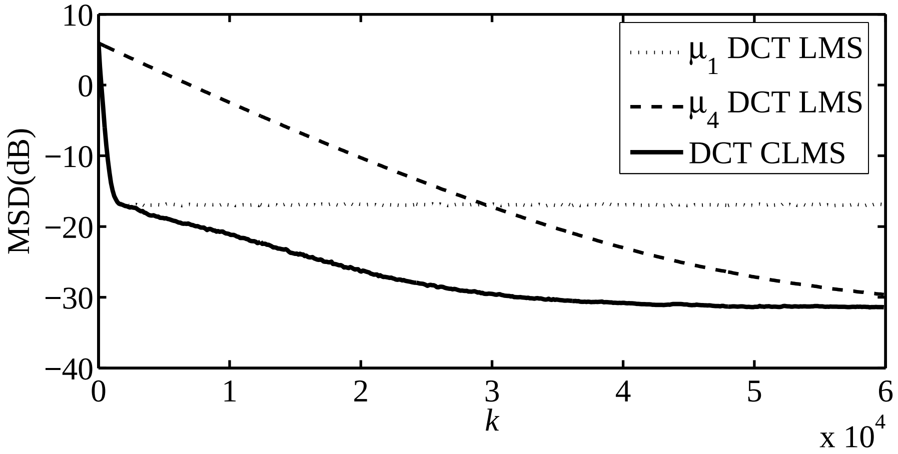
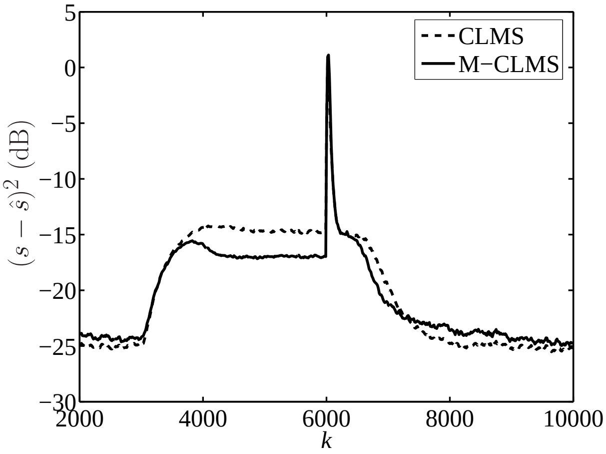
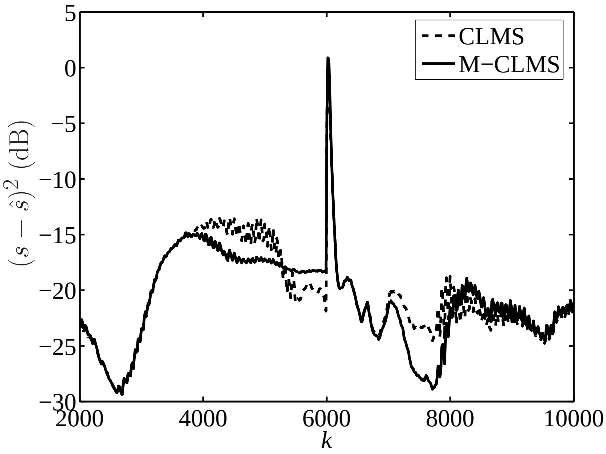

10
0
−10
−20
−30
−40
MSD(dB)
0
1
2
3
4
5
6
k
x 10^4
μ1 DCT LMS
μ4 DCT LMS
DCT CLMS

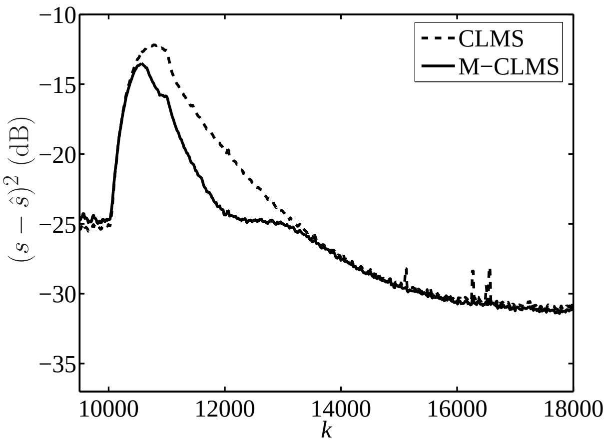
Fig. 7


CLMS
M−CLMS
$(s-\hat{s})^2$ (dB)
k

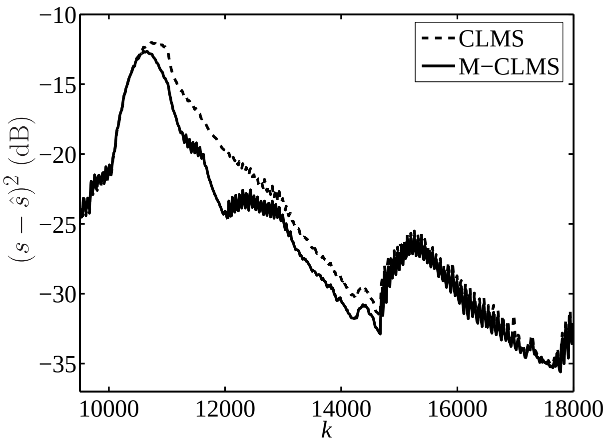
(a)


CLMS
M−CLMS
$(s-\hat{s})^2$ (dB)
k

(c)


CLMS
M−CLMS
$(s-\hat{s})^2$ (dB)
k

(b)


CLMS
M−CLMS
$(s-\hat{s})^2$ (dB)
k

(d)

Fig. 8