STOCHASTIC MODEL OF THE LMS VOLTERRA FILTER

Eduardo L. O. Batista, Orlando J. Tobias, and Rui Seara

LINSE – Circuits and Signal Processing Laboratory
Department of Electrical Engineering
Federal University of Santa Catarina
88040-900 – Florianópolis – SC – Brazil
E-mails: {dudu, orlando, seara}@linse.ufsc.br

ABSTRACT

This paper presents a statistical model for the LMS-Volterra filter, which is valid for truncated Volterra filters of any order with stationary input signals. Model expressions for the mean-weight behavior and learning curve are derived by considering a slow adaptation condition. The proposed model brings out and discusses the particular structure of the Volterra input autocorrelation matrix, focusing on its impact on the algorithm behavior. Simulation results illustrating the accuracy of the proposed analytical model are presented.

1. INTRODUCTION

In the last few years, nonlinear adaptive filters are becoming more and more feasible for using in practical applications. This is due to the increasing processing capacity of modern digital signal processors (DSPs), resulting in an increased research interest in this area. In this context, Volterra adaptive filters have become a feasible option for dealing with several nonlinear applications, such as active noise control [1], acoustic echo canceling [2], satellite-channel equalization [3], among others.

An interesting feature of Volterra filters, of nonlinear nature, is that they can be mathematically treated with the same procedures of conventional linear filters. This is due to the possibility of representing the output signal as a product of two vectors. Thus, an adaptive Volterra filter can make use of all available adaptive algorithms considered in linear applications. In this regard and because of its simplicity and robustness, we have considered here the LMS algorithm for adapting the weights of the Volterra filter.

Several works devoted to the analysis of adaptive Volterra filters are available in the open literature. One of the first works on that subject is presented in [4]. There, the discussed analytical results are restricted to second-order Volterra filters. Other works are concentrated on convergence properties and stability problems [5]-[7] as well as focused on filters considering input orthogonalization approaches [8]. Thus the modeling of generalized order Volterra filters is still open in the technical literature. Contributing in this scope, this paper aims to derive model expressions for the first and second order moments of the LMS-Volterra filter without restricting the filter order. Thus under slow adaptation conditions, new results are developed allowing to accurately model the behavior of the LMS-Volterra filter.

This paper is organized as follows. In Section 2, the standard Volterra filter is introduced. Section 3 derives the optimum weight vector. Section 4 is dedicated to the algorithm modeling, in which expressions for the mean-weight behavior, learning curve, and misadjustment are derived. Section 5 presents some numerical simulations aiming to assess the correctness of the proposed model. Finally, Section 6 provides the conclusions and remarks of the work in question.

2. VOLTERRA FILTER

A causal Volterra filter with a finite memory N and order P is described by [9]

$$y(n) = \sum_{p=1}^{P} y_p(n) \tag{1}$$

where $y_p(n)$ is written as

$$y_p(n) = \sum_{m_1=0}^{N-1} \sum_{m_2=m_1}^{N-1} \cdots \sum_{m_p=m_{p-1}}^{N-1} h_p(m_1, m_2, \ldots, m_p) \times \prod_{k=1}^{p} x(n-m_k) \; . \tag{2}$$

In (1) and (2), $x(n)$ and $y(n)$ represent the filter input and output signals, respectively, and $h_p(m_1, m_2, \ldots, m_p)$ denotes the p^{th}-order weight. Notice that the redundant terms in (2) were removed [1]. From [1], (2) can be rewritten as

$$y_p(n) = \mathbf{h}_p^{\mathrm{T}} \mathbf{x}_p(n) \tag{3}$$

where $\mathbf{h}_p$ is the p^{th}-order weight vector and $\mathbf{x}_p(n)$ is the p^{th}-order input vector, containing the cross products of the input vector samples. By defining the Volterra weight vector as

$$\mathbf{h}_{\mathrm{V}} = [\mathbf{h}_1^{\mathrm{T}}, \mathbf{h}_2^{\mathrm{T}}, \ldots, \mathbf{h}_P^{\mathrm{T}}]^{\mathrm{T}} \tag{4}$$

and the Volterra input vector as

$$\mathbf{x}_{\mathrm{V}}(n) = [\mathbf{x}_1^{\mathrm{T}}(n), \mathbf{x}_2^{\mathrm{T}}(n), \ldots, \mathbf{x}_P^{\mathrm{T}}(n)]^{\mathrm{T}} \tag{5}$$

from (1), one can write the input-output relationship of the Volterra filter as an inner product of two vectors. Thus,

$$y(n) = \mathbf{h}_{\mathrm{V}}^{\mathrm{T}} \mathbf{x}_{\mathrm{V}}(n) \; . \tag{6}$$

The vector description of the filtering operation (6) greatly facilitates the mathematical derivations for modeling a Volterra filter.

3. OPTIMUM WEIGHT VECTOR

Fig. 1 shows the block diagram of a Volterra filter aiming to estimate a signal $d(n)$, which is correlated with $x(n)$.

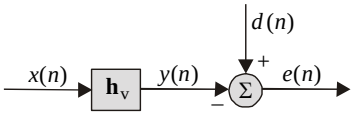

Figure 1 – Volterra filter applied to a signal estimation problem.

Orlando J. Tobias is also with the Electrical Engineering and Telecom. Dept., Regional University of Blumenau (FURB), Blumenau, SC, Brazil. This work was supported in part by the Brazilian National Research Council for Scientific and Technological Development (CNPq).

From Fig. 1, the instantaneous error signal is written as

$$e(n) = d(n) - y(n) = d(n) - \mathbf{h}_{\mathrm{V}}^{\mathrm{T}}\mathbf{x}_{\mathrm{V}}(n) . \quad (7)$$

According to the minimum mean-square error (MMSE) criterion [10], the optimum weight vector is obtained by minimizing the cost function $\xi = E[e^2(n)]$. Thus, by squaring (7) and taking the expected value of the resulting expression, we obtain

$$\xi = E[d^2(n)] - 2E[d(n)\mathbf{x}_{\mathrm{V}}^{\mathrm{T}}(n)]\mathbf{h}_{\mathrm{V}} + \mathbf{h}_{\mathrm{V}}^{\mathrm{T}}E[\mathbf{x}_{\mathrm{V}}(n)\mathbf{x}_{\mathrm{V}}^{\mathrm{T}}(n)]\mathbf{h}_{\mathrm{V}} . \quad (8)$$

Now, making $\nabla_{\mathbf{h}_{\mathrm{V}}}\xi = 0$, and defining the autocorrelation matrix of the input vector as $\mathbf{R}_{\mathrm{VV}} = E[\mathbf{x}_{\mathrm{V}}(n)\mathbf{x}_{\mathrm{V}}^{\mathrm{T}}(n)]$ and the cross-correlation vector between the desired and input signals as $\mathbf{p}_{\mathrm{V}} = E[d(n)\mathbf{x}_{\mathrm{V}}(n)]$, the optimum weight vector is given by

$$\mathbf{h}_{\mathrm{Vo}} = \mathbf{R}_{\mathrm{VV}}^{-1}\mathbf{p}_{\mathrm{V}} . \quad (9)$$

4. LMS-VOLTERRA ANALYSIS

4.1 Mean-weight behavior

By considering the LMS algorithm for weight updating, we have the following recursive expression [9]:

$$\mathbf{h}_{\mathrm{V}}(n+1) = \mathbf{h}_{\mathrm{V}}(n) + 2\mu e(n)\mathbf{x}_{\mathrm{V}}(n) \quad (10)$$

where μ is the step-size parameter. If different values of the step size are considered for each Volterra block, the LMS update expression is given by

$$\mathbf{h}_{\mathrm{V}}(n+1) = \mathbf{h}_{\mathrm{V}}(n) + 2\mathbf{M}e(n)\mathbf{x}_{\mathrm{V}}(n) \quad (11)$$

where $\mathbf{M}$ is a diagonal matrix containing the different step-size values for each weight. For the sake of simplicity, the development presented here considers the case of a single step-size value given by (10). Its extension to the multiple step-size case is straightforward. By adding a measurement noise $z(n)$ (zero-mean and uncorrelated with any other signals in the system) to $e(n)$ in (10) and taking the expected value of both sides of the resulting expression, we get

$$\begin{aligned} E[\mathbf{h}_{\mathrm{V}}(n+1)] = E[\mathbf{h}_{\mathrm{V}}(n)] + 2\mu E[d(n)\mathbf{x}_{\mathrm{V}}(n)] \\ +2\mu E[z(n)\mathbf{x}_{\mathrm{V}}(n)] - 2\mu E[\mathbf{x}_{\mathrm{V}}(n)\mathbf{x}_{\mathrm{V}}^{\mathrm{T}}(n)\mathbf{h}_{\mathrm{V}}(n)]. \end{aligned} \quad (12)$$

Now, assuming that the correlation between the input vectors is more important than the correlation between the weight vector and the input vector, we can approximate the expectation of the last term in the right hand side of (12) as follows:

$$E[\mathbf{x}_{\mathrm{V}}(n)\mathbf{x}_{\mathrm{V}}^{\mathrm{T}}(n)\mathbf{h}_{\mathrm{V}}(n)] \approx E[\mathbf{x}_{\mathrm{V}}(n)\mathbf{x}_{\mathrm{V}}^{\mathrm{T}}(n)]E[\mathbf{h}_{\mathrm{V}}(n)] . \quad (13)$$

Such an approximation is valid for small step-size values [10].
From (12), (13) and from the characteristics of $z(n)$, the recursive expression for the mean-weight behavior is given by

$$E[\mathbf{h}_{\mathrm{V}}(n+1)] = (\mathbf{I} - 2\mu\mathbf{R}_{\mathrm{VV}})E[\mathbf{h}_{\mathrm{V}}(n)] + 2\mu\mathbf{p}_{\mathrm{V}} . \quad (14)$$

By defining the weight-error vector as

$$\mathbf{v}(n) = \mathbf{h}_{\mathrm{V}}(n) - \mathbf{h}_{\mathrm{Vo}} \quad (15)$$

from (14), a recursive expression for $E[\mathbf{v}(n)]$ is easily obtained as

$$E[\mathbf{v}(n+1)] = (\mathbf{I} - 2\mu\mathbf{R}_{\mathrm{VV}})E[\mathbf{v}(n)] . \quad (16)$$

Note that (9), (14), and (16) have the same form as the corresponding expressions for a linear FIR filter [10]. However, the structure of matrix $\mathbf{R}_{\mathrm{VV}}$ and vector $\mathbf{p}_{\mathrm{V}}$ is quite different from those of the linear case. Such a fact impacts on the algorithm behavior.

4.2 Matrix $\mathbf{R}_{\mathrm{VV}}$

The knowledge of the structure of the input autocorrelation matrix for the LMS-Volterra filter plays an important role on the algorithm behavior. By recalling its definition

$$\mathbf{R}_{\mathrm{VV}} = E[\mathbf{x}_{\mathrm{V}}(n)\mathbf{x}_{\mathrm{V}}^{\mathrm{T}}(n)] \quad (17)$$

the structure of such a matrix considering a P^{th}-order Volterra filter, whose input vector is given by (5), can be alternatively expressed as

$$\mathbf{R}_{\mathrm{VV}} = \begin{bmatrix} \mathbf{R}_{11} & \mathbf{R}_{12} & \cdots & \mathbf{R}_{1P} \\ \mathbf{R}_{21} & \mathbf{R}_{22} & \cdots & \mathbf{R}_{2P} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{R}_{P1} & \mathbf{R}_{P2} & \cdots & \mathbf{R}_{PP} \end{bmatrix} \quad (18)$$

where each submatrix of $\mathbf{R}_{\mathrm{VV}}$ is given by $\mathbf{R}_{p_1p_2} = E[\mathbf{x}_{p_1}(n)\ \mathbf{x}_{p_2}^{\mathrm{T}}(n)]$, and $\mathbf{x}_{p_1}(n)$ and $\mathbf{x}_{p_2}(n)$ are, respectively, the p_1^{th}-order and p_2^{th}-order input vectors. The main result of the particular form of $\mathbf{R}_{\mathrm{VV}}$ is the coupling effect between the elements of the recursive expression (16).

4.3 Coupling effect of $\mathbf{R}_{\mathrm{VV}}$

Note from (14) and (16) that the mean-weight behavior depends on $\mathbf{R}_{\mathrm{VV}}$. By considering first a purely linear case, the input autocorrelation matrix is only composed of the submatrix $\mathbf{R}_{11}$, which for a white input signal with variance σ_x^2 is given by

$$\mathbf{R}_{11} = \begin{bmatrix} \sigma_x^2 & 0 & \cdots & 0 \\ 0 & \sigma_x^2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma_x^2 \end{bmatrix} \quad (19)$$

From (19), (16) results in a set of uncoupled equations, permitting to write a model expression for the i^{th} adaptive weight independently of all other weights as

$$E[v_i(n+1)] = (1 - 2\mu\sigma_x^2)E[v_i(n)] . \quad (20)$$

Now, by considering a second-order Volterra filter, matrix $\mathbf{R}_{\mathrm{VV}}$ is composed of the submatrices $\mathbf{R}_{11}$, $\mathbf{R}_{12}$, $\mathbf{R}_{21}$, and $\mathbf{R}_{22}$. By considering again a white input signal with variance σ_x^2, $\mathbf{R}_{11}$ is given by (19), $\mathbf{R}_{21} = \mathbf{R}_{12}^{\mathrm{T}} = \mathbf{0}$, and

$$\mathbf{R}_{22} = \begin{bmatrix} 3\sigma_x^4 & 0 & \cdots & 0 & \sigma_x^4 & 0 & \cdots & \sigma_x^4 \\ 0 & \sigma_x^4 & \cdots & 0 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma_x^4 & 0 & 0 & \cdots & 0 \\ \sigma_x^4 & 0 & \cdots & 0 & 3\sigma_x^4 & 0 & \cdots & \sigma_x^4 \\ 0 & 0 & \cdots & 0 & 0 & \sigma_x^4 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \sigma_x^4 & 0 & \cdots & 0 & \sigma_x^4 & 0 & \cdots & 3\sigma_x^4 \end{bmatrix} \quad (21)$$

Thus in this case, the structure of $\mathbf{R}_{\mathrm{VV}}$ is

$$\mathbf{R}_{VV} = \begin{bmatrix} \mathbf{R}_{11} & \mathbf{0} \\ \mathbf{0} & \mathbf{R}_{22} \end{bmatrix} \quad (22)$$

From (22), some interesting characteristics can be drawn. Since the submatrices $\mathbf{R}_{12}$ and $\mathbf{R}_{21}$ are null matrices, the first and second order blocks are not coupled. This fact is in agreement with the parallel block structure of a Volterra filter [9], also observed by making $\mathbf{R}_{21} = \mathbf{R}_{12}^{\mathrm{T}} = \mathbf{0}$ in (20). Thus,

$$E\left\{\begin{bmatrix} \mathbf{v}_1(n+1) \\ \mathbf{v}_2(n+1) \end{bmatrix}\right\} = \begin{Bmatrix} E[\mathbf{v}_1(n)] \\ E[\mathbf{v}_2(n)] \end{Bmatrix} - 2\mu \begin{Bmatrix} \mathbf{R}_{11} E[\mathbf{v}_1(n)] \\ \mathbf{R}_{22} E[\mathbf{v}_2(n)] \end{Bmatrix}. \quad (23)$$

Since components $\mathbf{v}_1(n)$ and $\mathbf{v}_2(n)$ are uncoupled, this permits the use of different step-size values for each block, improving the algorithm convergence characteristics. Also notice that due to the structure of $\mathbf{R}_{22}$, the recursion for $\mathbf{v}_2(n)$ is no longer an uncoupled system. By extending the results to a third-order Volterra filter, in which matrix $\mathbf{R}_{13}$ is not null, the first and third-order blocks are coupled.

4.4 Influence of $\mathbf{R}_{VV}$ on the step-size parameter

The structure of $\mathbf{R}_{VV}$ also presents a great influence on the maximum step-size value for algorithm convergence. As discussed in [10], the maximum step-size value is related to the trace of $\mathbf{R}_{VV}$, which from (18) is given by

$$\mathrm{tr}[\mathbf{R}_{VV}] = \mathrm{tr}[\mathbf{R}_{11}] + \mathrm{tr}[\mathbf{R}_{22}] + \cdots + \mathrm{tr}[\mathbf{R}_{PP}] \quad (24)$$

with tr[·] denoting the matrix trace operator. Since the correlation matrices are always positive definite, this upholds that

$$\mathrm{tr}[\mathbf{R}_{11}] < \mathrm{tr}[\mathbf{R}_{VV}]. \quad (25)$$

By considering (25), it is seen that the LMS-Volterra filter requires a maximum step-size value smaller than that of a linear filter for the same input signal, for maintaining algorithm convergence.

4.5 Learning curve

To determine the model expression for the learning curve, we use (7) along with the definition of the weight-error vector (15). Thus, the instantaneous error can be rewritten as

$$e(n) = d(n) - \mathbf{v}^{\mathrm{T}}(n)\mathbf{x}_V(n) - \mathbf{h}_{Vo}^{\mathrm{T}}\mathbf{x}_V(n). \quad (26)$$

By defining the optimum estimation error as $e_o(n) = d(n) - \mathbf{h}_{Vo}^{\mathrm{T}}\mathbf{x}_V(n)$, considering $\xi(n) = E[e^2(n)]$, squaring (26), and taking the expected value of both sides of the resulting expression, we obtain

$$\xi(n) = E[e_o^2(n)] + E\{[\mathbf{v}^{\mathrm{T}}(n)\mathbf{x}_V(n)]^2\} - E[e_o(n)\mathbf{v}^{\mathrm{T}}(n)\mathbf{x}_V(n)]. \quad (27)$$

By considering $\xi_{\min} = E[e_o^2(n)]$, defining the weight-error vector covariance matrix as $\mathbf{K}(n) = E[\mathbf{v}(n)\mathbf{v}^{\mathrm{T}}(n)]$, and using the Orthogonality Principle [10], the expression for the learning curve (27) can be rewritten as

$$\xi(n) = \xi_{\min} + \mathrm{tr}\{\mathbf{K}(n)\mathbf{R}_{VV}\}. \quad (28)$$

4.6 Weight-error correlation matrix

The learning curve (28) depends on the knowledge of $\mathbf{K}(n)$. A recursive expression for the latter parameter is obtained by expressing (10) as a function of the weight-error vector given by

$$\mathbf{v}(n+1) = \mathbf{v}(n) - 2\mu\mathbf{x}_V(n)\mathbf{x}_V^{\mathrm{T}}(n)\mathbf{v}(n) + 2\mu e_o(n)\mathbf{x}_V(n). \quad (29)$$

Now, making $\mathbf{v}(n+1)\mathbf{v}^{\mathrm{T}}(n+1)$ and taking the expected value of both sides of the resulting expression, we get

$$\begin{aligned} E[\mathbf{v}(n+1)\mathbf{v}^{\mathrm{T}}(n+1)] &= E[\mathbf{v}(n)\mathbf{v}^{\mathrm{T}}(n)] \\ &-2\mu E[\mathbf{v}(n)\mathbf{v}^{\mathrm{T}}(n)\mathbf{x}_V(n)\mathbf{x}_V^{\mathrm{T}}(n)] \\ &-2\mu E[\mathbf{x}_V(n)\mathbf{x}_V^{\mathrm{T}}(n)\mathbf{v}(n)\mathbf{v}^{\mathrm{T}}(n)] \\ &+4\mu^2 E[\mathbf{x}_V(n)\mathbf{x}_V^{\mathrm{T}}(n)\mathbf{v}(n)\mathbf{v}^{\mathrm{T}}(n)\mathbf{x}_V(n)\mathbf{x}_V^{\mathrm{T}}(n)] \\ &+4\mu^2 E[\mathbf{x}_V(n)e_o(n)e_o(n)\mathbf{x}_V^{\mathrm{T}}(n)] \\ &+2\mu E\{[\mathbf{I} - 2\mu\mathbf{x}_V(n)\mathbf{x}_V^{\mathrm{T}}(n)]\mathbf{v}(n)e_o(n)\mathbf{x}_V^{\mathrm{T}}(n)\} \\ &+2\mu E\{\mathbf{x}_V(n)e_o(n)\mathbf{v}^{\mathrm{T}}(n)[\mathbf{I} - 2\mu\mathbf{x}_V(n)\mathbf{x}_V^{\mathrm{T}}(n)]\}. \end{aligned} \quad (30)$$

From the Orthogonality Principle the last two right hand side terms in (30) are equal to zero. For the sake of mathematical simplicity, the fourth right hand side term of (30) is disregarded due to its small impact on the obtained recursive expression. Then determining the remaining terms in (30), the following recursive expression for the second-order moment is obtained:

$$\begin{aligned} \mathbf{K}(n+1) = \mathbf{K}(n) - 2\mu\left(\mathbf{K}(n)\mathbf{R}_{VV} + \mathbf{R}_{VV}\mathbf{K}(n)\right) \\ +4\mu^2\xi_{\min}\mathbf{R}_{VV}. \end{aligned} \quad (31)$$

Finally, with the use of (28) and (31), we find an expression for describing the learning curve of the LMS-Volterra algorithm.

4.7 Algorithm misadjustment

The misadjustment is defined as

$$\mathrm{M} = \xi_{\mathrm{excess}} / \xi_{\min} \quad (32)$$

where ξ_{excess} is the MSE in excess, i.e., the difference between the steady-state value of the MSE and its optimum value. Thus, subtracting $\xi_{\min}$ from (28), considering algorithm convergence, i.e., $\lim_{n\to\infty}\mathbf{K}(n+1) = \lim_{n\to\infty}\mathbf{K}(n) = \mathbf{K}_\infty$, and using (31), we get

$$\xi_{\mathrm{excess}} = \mathrm{tr}[\mathbf{K}_\infty\mathbf{R}_{VV}] = \mu\xi_{\min}\,\mathrm{tr}[\mathbf{R}_{VV}]. \quad (33)$$

Finally, substituting (33) into (32), we obtain

$$\mathrm{M} = \mu\,\mathrm{tr}[\mathbf{R}_{VV}]. \quad (34)$$

Note from the presented expressions that the particular structure of matrix $\mathbf{R}_{VV}$ has great impact on the behavior of the LMS-Volterra algorithm.

5. SIMULATION RESULTS

To verify the accuracy of the proposed model some numerical simulations are presented considering a system identification problem. In the given examples different step-size values, referred to its maximum step-size value $\mu_{\max}$ for algorithm convergence, are used. The measurement noise $z(n)$ is white and Gaussian with variance $\sigma_z^2 = 10^{-6}$.

Example 1: In this example, a second-order Volterra filter with memory equal to 7 is used. The input signal is white, Gaussian with unit variance. The plant is also a second-order Volterra filter given by

$$\begin{aligned} \mathbf{h}_1 &= [1.00\ 0.50\ 0.10\ -0.20\ -0.30\ -0.10\ 0.08]^{\mathrm{T}} \\ \mathbf{h}_2 &= [0.70\ 0.30\ 0.00\ -0.10\ 0.00\ 0.00\ 0.05\ 0.50\ 0.20\ 0.00\ 0.00 \\ &\quad -0.20\ 0.00\ -0.35\ 0.00\ -0.05\ 0.01\ 0.00\ -0.20\ 0.00\ 0.05 \\ &\quad 0.00\ 0.10\ 0.05\ 0.02\ -0.10\ 0.00\ 0.05]^{\mathrm{T}}. \end{aligned} \quad (35)$$

Fig. 2 shows the results obtained for the mean-weight behavior, whereas Fig. 3 presents the results for the learning curve, using $\mu = 0.5\mu_{max}$ with $\mu_{max} = 0.005$ (experimentally determined). In these figures, we verify the accuracy of the proposed model. A small mismatch during the transient phase of the learning curve is observed, due to the step-size value used ($\mu = 0.5\mu_{max}$), which violates the slow adaptation condition. For a smaller step-size value $\mu = 0.1\mu_{max}$ (Fig. 4), the learning curve obtained from Monte Carlo (MC) simulations presents a very good agreement with the proposed model.

Example 2: This example considers the same plant as in Example 1. Now, the input signal is a colored noise obtained from an AR(1) process, given by $x(n) = \alpha\, x(n-1) + \sqrt{1-\alpha^2}\, u(n)$, where $u(n)$ is a white noise process with unit variance and $\alpha = 0.5$. The step size used in this example is $0.25\mu_{max}$ with $\mu_{max} = 0.002$ (experimentally obtained). The learning curve results are shown in Fig. 5, where again a very good accuracy between the simulation and the model is observed.

Example 3: In this example, a third-order Volterra plant with memory equal to 8 is used. The adaptive filter also is a third-order Volterra filter with the same plant and memory size. The input signal is white noise with unit variance. The step size used is $0.2\mu_{max}$ with $\mu_{max} = 0.008$ (experimentally determined). Fig. 6 shows the results obtained for the learning curve (simulation and model), also confirming the model accuracy, now considering a higher-order Volterra filter.

Example 4: For this example, the input signal presents uniform probability distribution between -1 and 1. The plant and the modeling filter are the same as in Example 1. The step size is $0.1\mu_{max}$ with $\mu_{max} = 0.1$ (experimentally obtained). From Fig. 7, we again observe a very good agreement between the simulation and the analytical model results, now by considering an input signal with a non-Gaussian distribution.

Example 5: In this example, the plant is again the same as in Example 1 and the input signal presents a uniform probability distribution between 0 and 1. The step size is $0.2\mu_{max}$ with $\mu_{max} = 0.08$ (experimentally obtained). The results are shown in Fig. 8, from which we observe a very good accuracy between the simulation and the model.

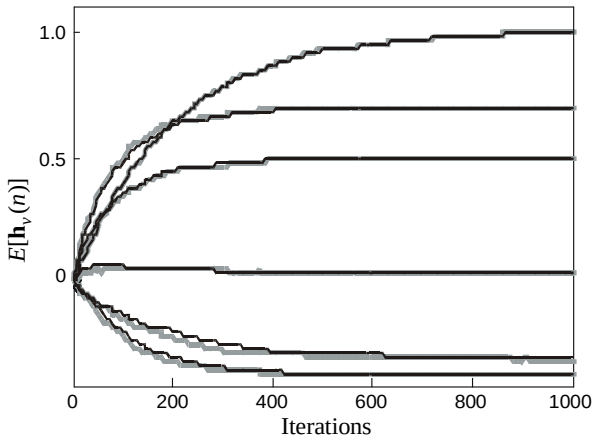

Figure 2 – Example 1. Evolution of $E[\mathbf{h}_V(n)]$ for $0.5\mu_{max}$. (Gray line) MC simulation (average of 200 independent runs). (Black line) analytical model Eq. (14).

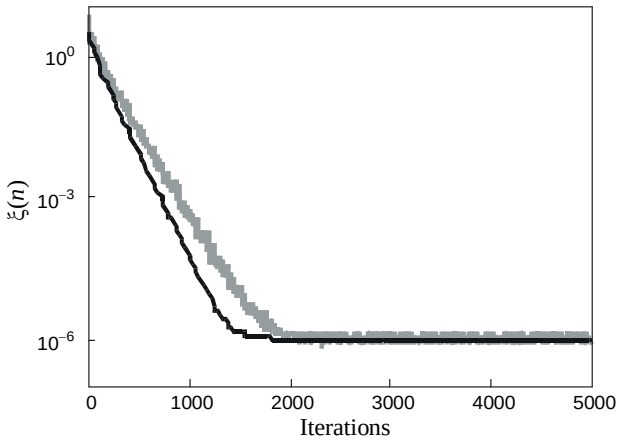

Figure 3 – Example 1. Evolution of $\xi(n)$ using $0.5\mu_{max}$. (Gray-ragged line) simulation (average of 200 independent runs). (Black line) analytical model Eqs. (28) and (31).

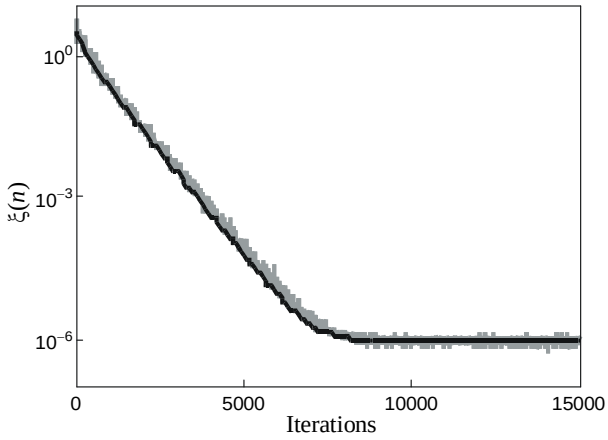

Figure 4 – Example 1. Evolution of $\xi(n)$ for $0.1\mu_{max}$. (Gray-ragged line) simulation (average of 200 independent runs). (Black line) analytical model Eqs. (28) and (31).

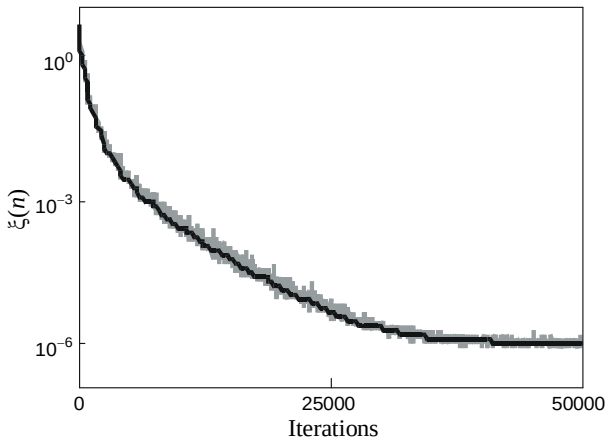

Figure 5 – Example 2. Evolution of $\xi(n)$ with $0.25\mu_{max}$. (Gray-ragged line) simulation (average of 200 independent runs). (Black line) analytical model Eqs. (28) and (31).

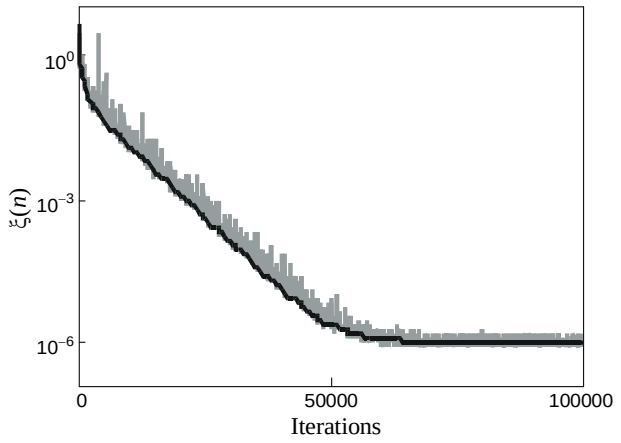

Figure 6 – Example 3. Evolution of $\xi(n)$ using $0.2\mu_{max}$. (Gray-ragged line) simulation (average of 200 independent runs). (Black line) analytical model Eqs. (28) and (31).

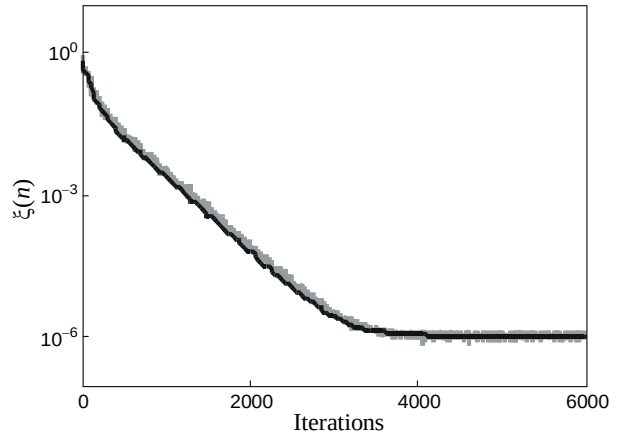

Figure 7 – Example 4. Evolution of $\xi(n)$ with $0.1\mu_{max}$. (Gray-ragged line) simulation (average of 200 independent runs). (Black line) analytical model Eqs. (28) and (31).

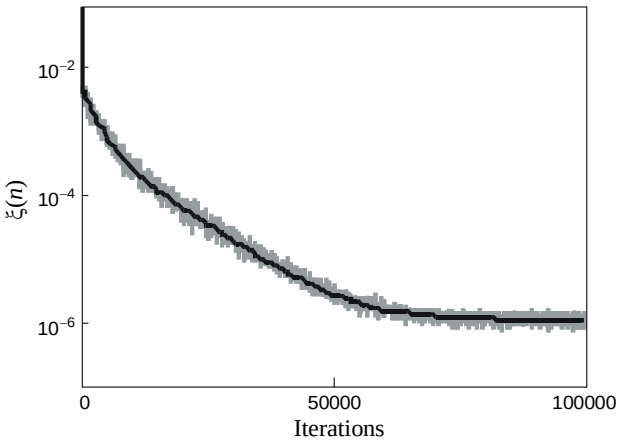

Figure 8 – Example 5. Evolution of $\xi(n)$ using $0.2\mu_{max}$. (Gray-ragged line) simulation (average of 200 independent runs). (Black line) analytical model Eqs. (28) and (31).

6. CONCLUSIONS AND REMARKS

In this work, a stochastic analysis of LMS-Volterra filters is presented. The vector form of the input-output relationship of the Volterra filter is exploited, permitting a tractable mathematics to derive the analytical expressions. In addition, the particular structure of the input autocorrelation matrix and its impact on the algorithm behavior are discussed. The developed model works well being independent of both the input signal statistics and Volterra filter order. The obtained results showed a very good agreement between the simulations and the proposed analytical model.

REFERENCES

[1] L. Tan and J. Jiang, "Adaptive Volterra filters for active control of nonlinear noise processes," *IEEE Trans. Signal Process.*, vol. 49, no. 8, pp. 1667-1676, Aug. 2001.

[2] A. Stenger, L. Trautmann, and R. Rabenstein, "Nonlinear acoustic echo cancellation with second order adaptive Volterra filters," in *Proc. IEEE Int. Conf. Acoustics, Speech, Signal Processing (ICASSP)*, Phoenix, AZ, vol. 2, Mar. 1999, pp. 877-880.

[3] A. Gutierrez and W. E. Ryan, "Performance of adaptive Volterra equalizers on nonlinear satellite channels," in *Proc. IEEE Int. Conf. Communications*, Seattle, WA, vol. 1, June 1995, pp. 488-492.

[4] T. Koh and E. Powers, "Second-order Volterra filtering and its application to nonlinear system identification," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 33, no. 6, pp. 1445-1455, Dec. 1985.

[5] P. M. Clarkson and M. V. Dokic, "Stability and convergence behavior of second order LMS Volterra filter," *Electronics Letters*, vol. 27, no. 5, pp. 441-443, Feb. 1991.

[6] M. V. Dokic and P. M. Clarkson, "On the performance of a second-order adaptive Volterra filter," *IEEE Trans. Signal Process.*, vol. 41, no. 5, pp. 1944-1947, May 1993.

[7] J. Chao and A. Inomata, "A convergence analysis of Volterra adaptive filters," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, Hong Kong, vol. 4, June 1997, pp. 2477-2480.

[8] T. Ogunfunmi and S. L. Chang, "Second order adaptive Volterra system identification based on discrete nonlinear Wiener model," *IEE Proc. - Vision, Image, Signal Process.*, vol. 148, no. 1, pp. 21-29, Feb. 2001.

[9] V. J. Mathews and G. L. Sicuranza, *Polynomial Signal Processing*, John Wiley & Sons Inc., 2000.

[10] S. Haykin, *Adaptive Filter Theory*, 4th ed., Prentice-Hall, 2002.