Circuits Systems Signal Processing
Vol. 23, No. 5, 2004, Pp. 365–386

DOI: 10.1007/s00034-004-3080-8

A Simplicial Canonical Piecewise Linear Adaptive Filter*

J. L. Figueroa,[1] *J. E. Cousseau,*[1] *and*
R. J. P. de Figueiredo[2]

Abstract. A new adaptive nonlinear filter realization is presented based on a canonical piecewise linear (CPWL) approach. This filter realization consists of a linear part represented by a finite impulse response filter and a zero-memory nonlinear part implemented as a CPWL map. The resultant structure requires fewer parameters than other realizations found in the literature with comparable modeling capabilities. As a consequence, the proposed nonlinear filter, in addition to its simple description, has low computation and implementation costs. Some results related to convergence properties and implementation of the adaptive algorithm associated to this new realization are presented. The performance of the proposed filter is illustrated through simulation examples.
Key words: Adaptive filtering, nonlinear filtering, piecewise linear models.

1. Introduction

Classical adaptive filtering algorithms [10] consist of updating the coefficients of a linear filter in real time. These algorithms have applications in a large number of practical situations where the random signals measured in the environment can be well modeled as outputs of linear dynamical systems driven by white Gaussian noise. However, the increasing transmission power demand in high-speed communications and related applications, leads to the exploration of channel resources beyond their linear range.

A number of efforts have been made toward finding general and efficient algorithms and realizations for nonlinear adaptive filtering applications. One such

* Received August 8, 2003; revised April 7, 2004.

[1] CONICET and Department of Electrical and Computer Engineering, Universidad Nacional del Sur, Avda. Alem 1253, 8000 Bahia Blanca, Argentina. E-mail for Figueroa: figueroa@uns.edu.ar, E-mail for Cousseau: iecousse@criba.edu.ar

[2] Department of Electrical Engineering and Computer Science, Department of Biomedical Engineering, and the Department of Mathematics, University of California, Irvine, California 92697-2625, USA. E-mail: rui@uci.edu

general approach is non-parametric. It introduces a Reproducing Kernel Hilbert space F of Volterra series [6]–[9], [30], [31] to which the nonlinear adaptive filter is made to belong, and the optimal nonlinear filter structure and parameters are obtained by orthogonal projection in F calculated recursively from the data. The resultant adaptive filter appears naturally in the form of a multilayered neural network. In many other contributions, the least mean square (LMS) or the recursive least square (RLS) approaches are applied to truncated Wiener–Volterra series [22], [27], [32], or neural network models [26]. However, in all these approaches the computation and implementation costs increase sharply with the number of nonlinear elements present in the filter.

An intensively studied tool for nonlinear modeling is the piecewise linear (PWL) function. A PWL function is an approximate representation of a nonlinear function. It replaces the global nonlinear function by a series of linear subfunctions that are defined in properly partitioned subregions of the original definition region of the nonlinear function. Many results were obtained in the last three decades to find more efficient, and complete PWL representations [3]–[5], [21], [20], [24], [26], [32]–[34].

Classical expressions used to represent PWL functions, called *conventional* expressions [2], describe the function region by region and require a large number of parameters. An improvement to find a more compact expression for PWL functions was obtained for $\boldsymbol{R}^1$ domains in [5]. This realization proved to be *canonical*; in the sense that it has the minimum number of parameters for this mapping. On the other hand, as it fails to represent any arbitrary PWL map, it was extended in [20] not only for $\boldsymbol{R}^1$ but also for $\boldsymbol{R}^2$ domains. More evidence of the potential usefulness of PWL representations was obtained in [24], where the form of a PWL representation for a domain of arbitrary dimension was demonstrated. Also, in a series of articles [33]–[35], the PWL was studied in the context of nonlinear system identification.

More recently, in [19] [18], and [29], a systematic way of defining the PWL representation for arbitrary (continuous) domains in a more compact and efficient form was introduced. This PWL representation uses the concept of *simplicial partitions* of the domain of interest.

We are interested in this article in the particular PWL representation obtained in [24] only for the $\boldsymbol{R}^1$ domain in order to demonstrate the particular advantage in terms of computational complexity if compared with the other representations (conventional, canonical, etc.), with adaptive filtering applications in mind. Of particular interest is the fact that many physical systems present a special structure in their models: they can be represented as a linear dynamic filter followed by a static nonlinearity. This is a classical structure that in the systems theory literature is called the nonlinear Wiener model [1].

Related to this work, in [23] the researchers proposed the use of a canonical piecewise linear representation using a structure similar to that discussed in [4]. We call this realization the Lin–Unbehauen piecewise linear (LUPWL) represen-

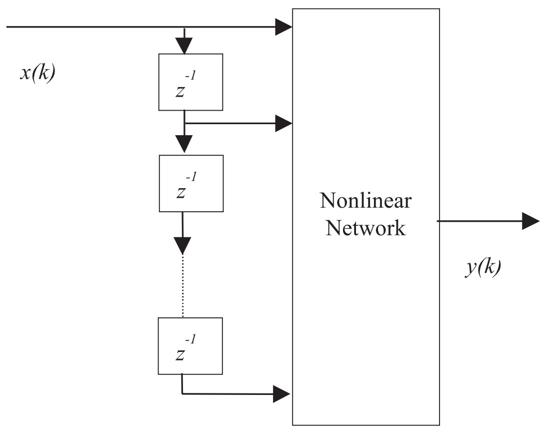

Figure 1. A general nonlinear filter.

tation. In addition, in [34] a direct-form PWL was demonstrated to be useful for nonlinear Wiener modeling, although the PWL function used is rather restricted and does not seem to be easily generalized for domains different from $\boldsymbol{R}^1$.

We present in this work a CPWL filter, the *simplicial canonical piecewise linear* (SPWL) filter, in the context of nonlinear Wiener models aiming to put in evidence the simple way in which this realization can describe these kind of models (if compared for example with [34]) and, also, to put in evidence the advantages of the resulting filter in terms of low complexity due to the reduced number of parameters (with respect to [23]).

The paper is organized as follows. In Section 2 a brief description of classical approximations for a nonlinearity is discussed, especially in terms of Volterra series and CPWL modeling approaches. In Section 3 the SPWL approximation for the Wiener model is presented and discussed. An LMS-based algorithm for coefficient updating of the CPWL realization is presented in Section 4. Some application examples are included in Section 5, mostly to compare the proposed nonlinear filter with other solutions available. Finally, the conclusions of this work are presented in Section 6. Due to limitations in space, we do not discuss the non-parametric approach, mentioned earlier, which allows a best approximation of a Volterra series without truncation. Details are given in [6]–[9], [31], [32].

2. Nonlinear filter modeling

Figure 1 depicts a general nonlinear filter with $x(k)$ as the input. Typically, the nonlinear network is approximated by a Volterra structure, a CPWL approximation, or a neural network. In this section, we discuss briefly the first two approaches in order to account for the different alternatives in nonlinear adaptive filtering.

2.1. Volterra model

The Volterra series model is the most complete model used for nonlinear systems. The Volterra series expansion of a nonlinear system consists of a nonrecursive series in which the output signal is related to the input signal as in the following expression:

$$y(k) = w_{o0} + \sum_{l_i=0}^{\infty} w_{o1}(l_1)x(k-l_1) + \sum_{l_i=0}^{\infty}\sum_{l_2=0}^{\infty} w_{o2}(l_1,l_2)x(k-l_1)x(k-l_2) \quad (1)$$
$$+ \cdots + \sum_{l_i=0}^{\infty}\sum_{l_2=0}^{\infty}\sum_{l_3=0}^{\infty} w_{o3}(l_1,l_2,l_3)x(k-l_1)\cdots x(k-l_3) + \cdots$$

It is usual to practice the truncation of this representation in two ways: (i) only N delayed inputs are considered, and (ii) only σ-order nonlinearity is considered. In this way, the structure given by (1) takes the form

$$y(k) = w_{o0} + \sum_{l_i=0}^{N} w_{o1}(l_1)x(k-l_1) + \sum_{l_i=0}^{N}\sum_{l_2=0}^{N} w_{o2}(l_1,l_2)x(k-l_1)x(k-l_2)$$
$$+ \cdots + \sum_{l_i=0}^{N}\sum_{l_2=0}^{N}\cdots\sum_{l_\sigma=0}^{N} w_{o\sigma}(l_1,l_2,\cdots,l_\sigma)x(k-l_1)\cdots x(k-l_\sigma),$$

where $w_{oi}(l_1,l_2,\dots,l_i)$ are the coefficients corresponding to σ-order nonlinearity modeled by the truncated Volterra series. These terms are also known as the Volterra kernels of the system.

One problem with truncated Volterra series is that high-order kernels can be ill conditioned, leading to stability problems if used as a realization for recursive identification. Another main drawback related to this model, in order to be applied in real time problems, is obviously the computational complexity related to the number of parameters involved. Although different approaches using this model with specific structures have been studied extensively (see for example [28]), complexity remains as an aspect to be improved.

To cope with specific modeling aspects, nonlinearities in the states or nonlinearities at the output have been used. They correspond to the known Hammerstein or Wiener nonlinear models, respectively. Specific aspects of recursive identification using a nonlinear Wiener model with a fixed (known) PWL nonlinearity were studied in [33]. In particular, parameter convergence can be obtained for IIR models in the linear part if the nonlinearity is strictly monotone (i.e., invertible). In addition, output error convergence for a FIR linear part can be obtained if the known nonlinearity is monotone [35].

2.2. *CPWL models*

The CPWL approach is an approximate representation of a nonlinear function. It replaces the global nonlinear function by a set of linear subfunctions which are defined in properly partitioned subregions of the original nonlinear function domain. A general CPWL function has a general expression [4] given by

$$f(\boldsymbol{x}(k)) = a + \boldsymbol{b}^T \boldsymbol{x}(k) + \sum_{i=1}^{\sigma} c_i \left| \boldsymbol{\alpha}_i^T \boldsymbol{x}(k) - \beta_i \right|, \tag{2}$$

where $\boldsymbol{b}$ and $\boldsymbol{\alpha}_i$ $(i = 1, \dots, \sigma)$ are N-dimensional weight vectors, a and β_i $(i = 1, \dots, \sigma)$ are scalar weights, and

$$\boldsymbol{x}(k) = [x(k), x(k-1), x(k-2), \dots, x(k-N+1)]^T \tag{3}$$

is the input vector. Geometrically, this function divides the input space into regions. For each region, the system is represented by a linear affine model. The boundaries of the partition of the linear subregions in the input space are determined by σ hyperplanes ($\boldsymbol{\alpha}_i^T \boldsymbol{x}(k) = \beta_i$ for $i = 1, \dots, \sigma$). The conditions for the uniqueness of a CPWL representation were discussed in [16]. This representation has found extensive use in the study of nonlinear circuits and systems, e.g., in the analysis and modeling of nonlinear devices and networks [13]. In spite of that, these structures can only represent a nonlinearity with domain in $\boldsymbol{R}^1$ [17].

In particular, the LUPWL uses an alternative canonical piecewise linear representation that can be described by the following expression:

$$f(\boldsymbol{x}(k)) = a + \boldsymbol{b}^T \boldsymbol{x}(k) + \sum_{i=1}^{\sigma} c_i \left(\left| \boldsymbol{\alpha}_i^T \boldsymbol{x}(k) - 1 \right| - \left| \boldsymbol{\alpha}_i^T \boldsymbol{x}(k) + 1 \right| \right). \tag{4}$$

It straightforward to verify that the basic modification with respect to the general CPWL models contemplates now a basis function with a saturation characteristic.

In [17], (2) was extended to a more general class of PWL descriptions. Using the concept of *simplicial partition* (a partition with a special structure), a basis for the representation of any memoryless function in $\boldsymbol{R}^N$ is defined in terms of N nested absolute values. In this form, considering a vector input as in (3), any nonlinear filter can be described as

$$y(\boldsymbol{x}(k)) = \boldsymbol{c}^T \boldsymbol{\Lambda}(\boldsymbol{x}(k)), \tag{5}$$

where the vector function $\boldsymbol{\Lambda}(\boldsymbol{x}(k))$ involves the computation of N nested absolute values that depend on the partition and the input vector. The particular definitions related to this model will be discussed in Section 3 in the context of a new nonlinear adaptive filter realization.

Although we are not specifically interested in this work in general $\boldsymbol{R}^N$ nonlinearities, this model appears particularly attractive due to its general properties in terms of function approximation [14] and, especially, for some constructive results available [18] in terms of computational complexity.

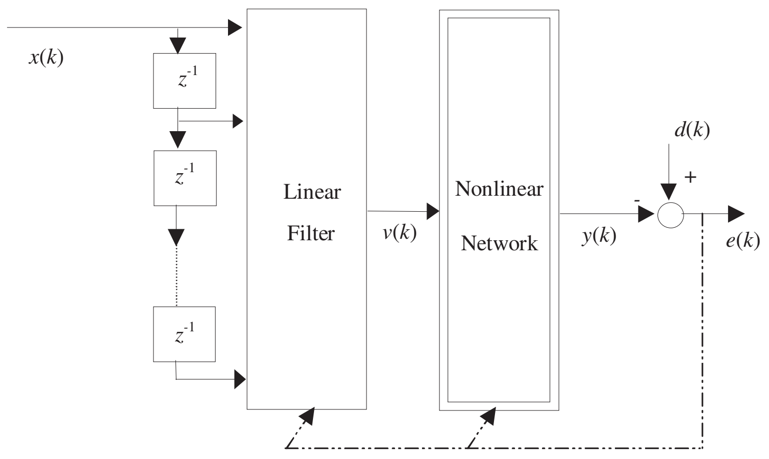

Figure 2. Adaptive nonlinear filter with nonlinear Wiener structure.

On the other hand, in some applications related to communications and signal processing, the system could be represented by a nonlinear Wiener model. This structure is defined by a dynamic linear filter followed by a static nonlinear gain (the see application examples in [10] and [23]). Using this structure, the complexity of the nonlinearity is strongly simplified. For example, in the Volterra series case, we should only consider the terms for which $l_1 = l_2 = l_3 = \cdots = l_N = 0$. In the following, we will consider the CPWL description for this application in particular.

3. Simplicial CPWL filtering

Figure 2 illustrates a nonlinear adaptive scheme where $d(k)$ is the desired output. We assume that $d(k)$ is an unbiased finite-memory nonlinear function of the input $x(k)$ and can be approximately represented by a linear filter followed by a static nonlinear gain function. For the linear part, it is possible to consider an N-order FIR realization given by

$$v(k) = \sum_{i=0}^{N-1} h_i x(k-i) = \boldsymbol{h}^T \boldsymbol{x}(k), \tag{6}$$

where $\boldsymbol{h} = [h_0\, h_1\, \cdots\, h_{N-1}]^T$.

In our approach, the CPWL model is the basic structure of the nonlinear filter realization. The static nonlinear function allows one to describe the output of the nonlinear filter as $y(k) = f(v(k)) : \boldsymbol{R} \longrightarrow \boldsymbol{R}$. The partition related to the CPWL approximation, defined by $\boldsymbol{\alpha}_j^T \boldsymbol{x} = \beta_j$, can be easily mapped to a simplicial partition given by $v = \beta_j$. Thus, the β_j parameters divide the domain in equal partitions, with $\beta_1 \leq \beta_2 \leq \cdots \leq \beta_\sigma$. With these assumptions in mind, we propose to use the following basis function [16]:

$$\Lambda_i(v(k)) = \begin{cases} \frac{1}{2}\left(v(k) - \beta_i + |v(k) - \beta_i|\right) & v \leq \beta_\sigma \\ \frac{1}{2}\left(\beta_\sigma - \beta_i + |\beta_\sigma - \beta_i|\right) & v > \beta_\sigma, \end{cases} \tag{7}$$

which results in an SPWL function with a special form given by

$$f(v(k)) = a + \sum_{i=1}^{\sigma} c_i \Lambda_i(v(k)) \tag{8}$$

or, in vector form,

$$f(v(k)) = \boldsymbol{c}^T \boldsymbol{\Lambda}(v), \tag{9}$$

where

$$\boldsymbol{c} = \begin{bmatrix} a & c_1 & c_2 & \ldots & c_\sigma \end{bmatrix}^T$$

and

$$\boldsymbol{\Lambda}(v) = \begin{bmatrix} 1 \\ \Lambda_1(v(k)) \\ \vdots \\ \Lambda_\sigma(v(k)) \end{bmatrix}. \tag{10}$$

From (7) it is clear that the nonlinear gain implicitly includes a saturation behavior.

Replacing (6) in (9), we obtain a closed expression for the proposed nonlinear filter

$$y(k) = f_s(\boldsymbol{x}(k)) = \boldsymbol{c}^T \boldsymbol{\Lambda}\left(\boldsymbol{h}^T \boldsymbol{x}(k)\right). \tag{11}$$

A realization of the SPWL filter is depicted in Figure 3. This realization can be compared with (4) for the LUPWL realization. It is evident from this comparison that the PWL introduced reduces the model complexity because only one FIR filter is required. In addition, the partitions are simplified because now the parameters (i.e., the β_i) are specified beforehand. Note that the number of free parameters in the proposed realization is $N + \sigma + 2$, which represents a small number compared with the $(N + 1)(\sigma + 1)$ involved in the LUPWL approach [23]. A typical LUPWL implementation is illustrated in Figure 4.

Also of particular interest for this work is the representation used in [34]. There, the nonlinear Wiener model was described by a more complete model for the linear part (an IIR model was used instead). Also, a direct-form PWL function was used to describe the static nonlinearity. The static nonlinearity was parameterized contemplating a fixed static gain in order to address possible adaptive control applications (this fixed static gain seems to be restrictive in order to model general nonlinear Wiener applications). The output of the nonlinear block is modeled as

$$y_{\boldsymbol{h},\boldsymbol{q}}(k) = f_w(\boldsymbol{q}, \hat{y}(k)),$$

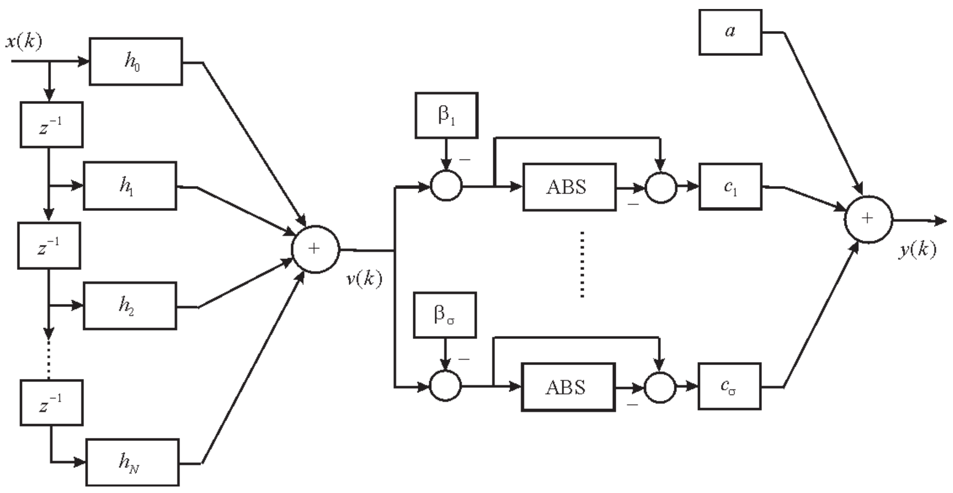

Figure 3. The Simplicial canonical piecewise linear (SPWL) adaptive filter.

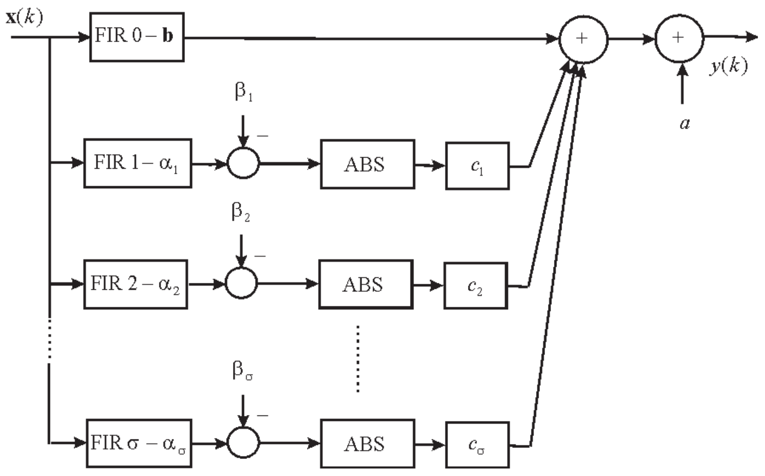

Figure 4. LUPWL realization.

where $\boldsymbol{h}$ and $\boldsymbol{q}$ refer to the parameters of the linear part and nonlinear part, respectively, and $\hat{y}(k)$ is the output of the linear part. In addition, $f_w(.,.)$ is a known function of $\boldsymbol{q}$ and $\hat{y}(k)$. The PWL function is designed fixing the slope of the static nonlinearity in one interval I_o of the partition, i.e.,

$$\frac{\partial}{\partial \hat{y}} f_w(\boldsymbol{q}, \hat{y}(k)) = k_o, \qquad \hat{y}(k) \in I_o,$$

where k_o is a constant. Note that $\frac{\partial}{\partial q} f_w(\boldsymbol{q}, \hat{y}(k)) = 0$ for $\hat{y}(k) \in I_o$. Also, bias is allowed where the gain is fixed, i.e.,

$$f_w(\boldsymbol{q}, \hat{y}(k)) = k_o \hat{y}(k) + f_{w,o}, \qquad \hat{y}(k) \in I_o.$$

To describe the model outside I_o, the complete $\boldsymbol{q}$ has the form

$$\boldsymbol{q} = [f_{w,-K}, \dots, f_{w,-1}, f_{w,o}, f_{w,1}, \dots, f_{w,K}]^T,$$

where $f_{w,i}$, $-K \le i \le K$ correspond to the values of $f_w(\boldsymbol{q}, \hat{y}(k))$ at the partition points. Although the expression for the piecewise approximation is direct, it must be constructed regarding each partition. This kind of parameterization of the nonlinearity is equivalent to our description in terms of number of parameters (i.e., the partition points are fixed). As can be observed, however, its implementation and description are not compact. It was addressed to certain feedback context applications where extensions to higher PWL dimensions were not the main concern. Despite that, this parameterization is useful to demonstrate certain properties related to the new SPWL realization.

4. LMS-based algorithm for the SPWL

In this section, we present basic results related to design of the updating equations and local convergence characterization of the proposed SPWL realization.

Because a close relationship exists between the proposed SPWL model and that studied in [34], convergence results will be based on this relationship. Specifically, the ordinary differential equation (ODE) method [25] can be used.

The objective of a typical LMS algorithm associated to the SPWL model is to choose the coefficients of the adaptive filter in such a way that the output signal, $y(k)$, during the period of observation, will match the desired signal as closely as possible in the LMS error sense (i.e., assuming a misadjustment at convergence). This minimization process can be used in the nonlinear adaptive filtering case by adequately reinterpreting the entries.

The standard approach to derive an LMS-based algorithm for the proposed nonlinear filter is to use the squared estimation error as an estimate of the mean-square error, i.e.,

$$J(e(k)) = e^2(k) = d^2(k) - 2d(k)y(k) + y^2(k),$$

where the adaptive filter output is given by

$$y(k) = \boldsymbol{c}^T \boldsymbol{\Lambda} \left(\boldsymbol{h}^T \boldsymbol{x}(k) \right).$$

An LMS-based algorithm can be used to minimize the objective function using the following equations:

$$\boldsymbol{h}(k+1) = \boldsymbol{h}(k) + \mu_h \psi_h(k) e(k) \tag{12}$$

Table 1. SPWL adaptive filter algorithm

```
Definitions:
        $y(k)$: SPWL filter output
        $e(k)$: error
        $v(k)$: linear filter output
Parameters:
        $M$ = number of $\boldsymbol{h}$ coefficients
        $N$ = number of $\boldsymbol{c}$ coefficients
        $\mu_h$ = step size related to $\boldsymbol{h}$ coefficients
        $\mu_c$ = step size related to $\boldsymbol{c}$ coefficients
Data: $\boldsymbol{x}(k)$ input vector ($M \times 1$)
        $d(k)$ desired response
Initialization
        $\boldsymbol{h}(0) = \mathbf{0}$
        $\boldsymbol{c}(0) = [-1\ 1\ 0\ \cdots\ 0\ -1]^T$
For each $k$, $k = 1, 2, \cdots$
                $v(k) = \boldsymbol{h}^T(k)\boldsymbol{x}(k)$
                $y(k) = \boldsymbol{c}^T(k)\boldsymbol{\Lambda}[v]$
                $e(k) = d(k) - y(k)$
                $\boldsymbol{h}(k+1) = \boldsymbol{h}(k) + \mu_h \boldsymbol{c}^T(k)\frac{\partial \boldsymbol{\Lambda}[v]}{\partial v}\boldsymbol{x}(k)e(k)$
                $\boldsymbol{c}(k+1) = \boldsymbol{c}(k) + \mu_c \boldsymbol{\Lambda}[v]e(k)$
```

$$\boldsymbol{c}(k+1) = \boldsymbol{c}(k) + \mu_c \boldsymbol{\psi}_c(k)e(k) \tag{13}$$

for $k = 0, 1, \ldots$, where $\boldsymbol{\psi}_c(k)$ and $\boldsymbol{\psi}_h(k)$ represent estimates of the gradient vector of the filter output with respect to the filter coefficients $\boldsymbol{c}$ and $\boldsymbol{h}$, respectively.

In particular, the respective gradients have the following form:

$$\boldsymbol{\psi}_c(k) = \boldsymbol{\Lambda}(v)$$

and

$$\boldsymbol{\psi}_h(k) = \left(\boldsymbol{c}^T \frac{\partial \boldsymbol{\Lambda}(v)}{\partial v}\right)\boldsymbol{x}(k),$$

where, if $[.]_j$ represents the jth component of the vector,

$$\begin{aligned} \left[\frac{\partial \boldsymbol{\Lambda}(v)}{\partial v}\right]_1 &= 0 \\ \left[\frac{\partial \boldsymbol{\Lambda}(v)}{\partial v}\right]_{j+1} &= \begin{cases} \frac{1}{2}\left(1 + \operatorname{sign}(v - \beta_j)\right) & v \le \beta_\sigma \\ 0 & v > \beta_\sigma \end{cases} \quad \text{for } j = 1, \ldots, \sigma. \end{aligned} \tag{14}$$

The algorithm obtained is summarized in Table 1. The SPWL algorithm can also be computed using (7) and (14).

Note here the direct relationship that exists between the parameters of the SPWL model proposed and those used in [34]. The model proposed in [34] lacks a compact form. Using $\mathbf{\Lambda}(v)$ to describe the PWL function, the relationship between both representations, based on (11), can be described by

$$y(k) = \left(\boldsymbol{G}^{-1}\boldsymbol{q}\right)^T \mathbf{\Lambda}\left(\boldsymbol{h}^T \boldsymbol{x}(k)\right), \tag{15}$$

where

$$\boldsymbol{G} = \begin{bmatrix} 1 & 1 & 1 & 1 & \cdots & 1 \\ 0 & \beta_2 - \beta_1 & \beta_3 - \beta_1 & & \cdots & \beta_\sigma - \beta_1 \\ 0 & 0 & \beta_3 - \beta_2 & & & \beta_\sigma - \beta_2 \\ & \vdots & & & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & \beta_\sigma - \beta_{\sigma-1} \end{bmatrix}^T$$

and $\boldsymbol{q}$ includes the PWL parameters of [34]. Clearly, the matrix $\boldsymbol{G}$, which depends on the partition, is lower triangular. In addition, because $\beta_i - \beta_{i-1} > 0$, matrix $\boldsymbol{G}$ is definite positive. Indeed, the nonlinear parameters for both descriptions $\boldsymbol{c}$ and $\boldsymbol{q}$ are related by $\boldsymbol{c} = \boldsymbol{G}^{-1}\boldsymbol{q}$. So, both models lead to the same description. Then, this close relationship indicates that convergence aspects of both algorithms are also related. Local convergence for general conditions were shown in [34], so similar results can be obtained for the proposed SPWL filter.

We emphasize here that the new SPWL filter was mainly addressed to allow the use of generalizations (description of nonlinearities not only in $\boldsymbol{R}^1$) related to the definition of the basis functions that describe the nonlinearity. A relative improvement with respect to the approach of [34] is the more compact way to describe the nonlinearity, even in $\boldsymbol{R}^1$.

4.1. Local convergence

A close relationship exists between the proposed SPWL model and that studied in [34], so convergence results will be based on this relationship. Specifically, the ODE method [25] can be used.

The ODE approach is quite general and comes from the field of stochastic approximation theory. The basic idea is to associate to a discrete parameter adaptation algorithm an ODE; in other words, to change the study of convergence of a stochastic nonlinear equation by the study of stability of the solutions of a deterministic differential equation. Following this methodology, the convergence properties of the discrete parameter adaptation algorithm are strongly related to the stability of the solutions of the differential equation. Two different kinds of algorithms can be studied in this form: *vanishing gain* algorithms (i.e., with the step size $\mu \to 0$), mostly oriented to estimation in a stationary environment, and *constant gain* algorithms (the kind of interest here), where the step size μ is left constant in order to contemplate tracking contexts. For this latter case, the ODE method guarantees that the adaptation algorithm converges in probability (not with probability one as in the vanishing gain algorithms).

Consider a generic stochastic gradient algorithm with the form

$$\phi(k+1) = \phi(k) + \mu e[\phi(k), \nabla(k, \phi(k))],$$

where $\phi(k)$ is the parameter vector, $e(k)$ is the error to be minimized, and $\nabla(k)$ is the estimated gradient of $e(k)$. The notation of $e(k)$ is addressed to put in evidence that it depends on $\phi(k)$ and $\nabla(k)$. The ODE associated has the form

$$\frac{\partial \varphi(t)}{\partial t} = f(\varphi), \tag{16}$$

where the form of $f(.)$ depends on the specific algorithm used. In order to guarantee the ODE association, some conditions must be verified. In our context, these conditions are related to the input and output signals, the specific algorithm, and the criteria to be used. Following [34], the input $\boldsymbol{x}(k)$ and output $y(k)$ signals are considered stochastic stationary processes, and the filters that generate the signals $e(k)$ and $\nabla(k)$ are considered exponentially stable. Then $f(.)$ represents an *average updating direction* that can be used to study the convergence properties of the algorithm. For example, stationary points, φ^*, of the generic algorithm can be obtained by means of the solutions of $f(\varphi^*) = 0$. In particular, if local convergence is addressed, we can use the indirect method of Liapunov (a linearization around a stationary point) to perform the study. Then based on a Taylor series expansion of (16), we construct a related linear differential equation of the form

$$\frac{\partial \varphi}{\partial t} = \left[\frac{\partial f(\varphi)}{\partial \varphi}\right]_{\varphi=\varphi^*} \varphi. \tag{17}$$

The point φ^* is a stable stationary point of (16) if and only if $\varphi = 0$ is an exponentially stable stationary point of (17). Then, this is equivalent to showing that the eigenvalues of the stability matrix $[\partial f/\partial \varphi]_{\varphi^*}$ have negative real parts. If the stability matrix is symmetric, its eigenvalues will be real. This is equivalent to saying that $[\partial f/\partial \varphi]_{\varphi^*}$ is a stability matrix if it is negative definite.

The ODE associated to the LMS-based algorithm presented is related to that obtained for the approach in [34], where local parameter convergence was stated for nonlinear Wiener models.

By defining $\boldsymbol{\theta}_s = [\boldsymbol{h}^T \ \boldsymbol{c}^T]^T$ and $\psi_s = [\psi_h^T \ \psi_c^T]^T$, the ODE associated to the CPWL filter is

$$\frac{\partial \boldsymbol{\theta}_s}{\partial t} = E[\psi_s e(k)]. \tag{18}$$

Under the basic assumption that the nonlinear model is described by the SPWL filter (this rather restrictive assumption is usually made in this context [23], [33]), except for a measurement noise $\nu(k)$, it can be easily verified that a stationary point $\boldsymbol{\theta}_s^*$ of the proposed algorithm corresponds to the solution of $E[\psi_s e(k)] = 0$.

With local convergence properties in mind, a linearization of (18) in a neighborhood of a stationary point $\boldsymbol{\theta}_s^*$ leads to the following expression:

$$\frac{\partial \boldsymbol{\theta}_s}{\partial t} \cong \left.\frac{\partial E[\boldsymbol{\psi}_s e(k)]}{\partial \boldsymbol{\theta}_s}\right|_{\boldsymbol{\theta}_s^*} (\boldsymbol{\theta}_s - \boldsymbol{\theta}_s^*). \tag{19}$$

In this way, by defining

$$\boldsymbol{P}_s(\boldsymbol{\theta}_s^*) = \left.\frac{\partial E[\boldsymbol{\psi}_s e(k)]}{\partial \boldsymbol{\theta}_s}\right|_{\boldsymbol{\theta}_s^*} = E\left.\left[\begin{array}{c} \left(\boldsymbol{c}^T \frac{\partial \boldsymbol{\Lambda}(v)}{\partial v}\right)\boldsymbol{x} \\ \boldsymbol{\Lambda}(v) \end{array}\right]\left[\begin{array}{c} \left(\boldsymbol{c}^T \frac{\partial \boldsymbol{\Lambda}(v)}{\partial v}\right)\boldsymbol{x} \\ \boldsymbol{\Lambda}(v) \end{array}\right]^T\right|_{\boldsymbol{\theta}_s^*},$$

local convergence can be guaranteed if eigenvalues of $\boldsymbol{P}_s(\boldsymbol{\theta}_s^*)$ have negative real part.

By defining now $\boldsymbol{\theta}_w = [\boldsymbol{h}\,\boldsymbol{q}]^T$ and $\boldsymbol{\psi}_w = [\boldsymbol{\psi}_h\,\boldsymbol{\psi}_q]^T$, the ODE associated to the stochastic gradient algorithm of [34] is given by

$$\frac{\partial \boldsymbol{\theta}_w}{\partial t} = E[\boldsymbol{\psi}_w e(k)], \tag{20}$$

which linearized in a neighborhood of a stationary point $\boldsymbol{\theta}_w^*$ can be rewritten as

$$\frac{\partial \boldsymbol{\theta}_w}{\partial t} \cong \left.\frac{\partial E[\boldsymbol{\psi}_w e(k)]}{\partial \boldsymbol{\theta}_w}\right|_{\boldsymbol{\theta}_w^*} (\boldsymbol{\theta}_w - \boldsymbol{\theta}_w^*), \tag{21}$$

where also, to guarantee local convergence, the following matrix was shown to have eigenvalues with negative real part [34]:

$$\boldsymbol{P}_w(\boldsymbol{\theta}_w^*) = \left.\frac{\partial E[\boldsymbol{\psi}_w e(k)]}{\partial \boldsymbol{\theta}_w}\right|_{\boldsymbol{\theta}_w^*}.$$

Using (15), the following mapping exists between $\boldsymbol{P}_w(\boldsymbol{\theta}_w^*)$ and $\boldsymbol{P}_s(\boldsymbol{\theta}_s^*)$:

$$\boldsymbol{P}_w(\boldsymbol{\theta}_w^*) = \begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{0} & (\boldsymbol{G}^T)^{-1} \end{bmatrix} \boldsymbol{P}_s(\boldsymbol{\theta}_s^*) \begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{G}^{-1} \end{bmatrix}.$$

For a given definition of the partition β_i, this defines a *congruence transformation* ([15], page 7) between both matrices, i.e., if $\boldsymbol{P}_w(\boldsymbol{\theta}_s^*)$ is negative definite, the same is verified by $\boldsymbol{P}_s(\boldsymbol{\theta}_s^*)$. As a consequence, the ODE associated to the SPWL adaptive filter converges, locally, to the stationary point $\boldsymbol{\theta}_s^*$.

Some remarks regarding local convergence aspects are the following:

- Because a bias in the input signal in that form could be contemplated, a known gain k_o for a specific sector I_o of the PWL description is important in [34]. Generically, using a fixed gain $k_o > 0$ in an arbitrary sector I_o, $\boldsymbol{P}_w(\boldsymbol{\theta}_w^*)$ can be written as the addition of two matrices, one with contributions of all sectors of the piecewise nonlinearity, and the other with the contribution associated to the constant sector with gain k_o, i.e.,

$$\boldsymbol{P}_w(\boldsymbol{\theta}_w^*) = \begin{bmatrix} \boldsymbol{A} & \boldsymbol{B} \\ \boldsymbol{B}^T & \boldsymbol{C} \end{bmatrix} + \begin{bmatrix} \tilde{\boldsymbol{A}} & \boldsymbol{0} \\ \boldsymbol{0}^T & \boldsymbol{0} \end{bmatrix},$$

where $\boldsymbol{A} = E\left[\frac{\partial f_w}{\partial \hat{y}}\boldsymbol{x}\boldsymbol{x}^T \frac{\partial f_w}{\partial \hat{y}}^T\right]$, $\boldsymbol{B} = E\left[\frac{\partial f_w}{\partial \hat{y}}\boldsymbol{x}\frac{\partial f_w}{\partial q}^T\right]$, $\boldsymbol{C} = E\left[\frac{\partial f_w}{\partial q}\frac{\partial f_w}{\partial q}^T\right]$ consider all sectors of the PWL function except I_o, and $\tilde{\boldsymbol{A}}$ considers matrix $\boldsymbol{A}$ in I_o, all evaluated at a stationary point. Note that both matrices are symmetric and $\boldsymbol{A} < 0$ is also symmetric.

- Consider the following lemma: We are given the symmetric matrix

$$\boldsymbol{S} = \begin{bmatrix} \boldsymbol{F} & \boldsymbol{G} \\ \boldsymbol{G}^T & \boldsymbol{H} \end{bmatrix},$$

  where $\boldsymbol{F}$ and $\boldsymbol{H}$ are negative definite. Then the following properties are equivalent: $\boldsymbol{S}$ is negative definite (semidefinite), and $\boldsymbol{F} - \boldsymbol{G}^T\boldsymbol{H}\boldsymbol{G}$ is negative definite (semidefinite).
- The lemma is verified assuming $\tilde{\boldsymbol{A}} < 0$ and $\boldsymbol{B} < 0$, because $\boldsymbol{A} < 0$ by construction, and $\boldsymbol{A} + \tilde{\boldsymbol{A}} < 0$, because both matrices are symmetric. This leads to $\boldsymbol{P}_w(\boldsymbol{\theta}_w^*) < 0$.
- The condition on $\tilde{\boldsymbol{A}}$ is satisfied if generic conditions of persistence of excitation of input signal [25] are verified. In this case we are working with an FIR model, so the persistence of excitation conditions implies a positive definite autocorrelation matrix of the input signal $\boldsymbol{R}_x = E[\boldsymbol{x}\boldsymbol{x}^T]$.
- $\boldsymbol{B}$ is related to the PWL description. The condition requires that there should be a signal energy somewhere in every subregion of the partition of the PWL model. In other words, the probability density function of the output must have a lower bound defined by a certain small, but positive, constant. This condition indicates that, for example, a pseudo random binary signal is unsuitable as an input signal due to its pointwise amplitude distribution.
- In our description, a constant k_o is equivalent to fix an element of $\boldsymbol{c}$. If $I_o \in (\beta_{i_o-1}, \beta_{i_o})$, then its corresponding gain is $k_0 = c(2) + c(3) + \cdots + c(i_o - 1)$. Then, fixing k_0 means setting $c(i_o - 1) = k_0 - c(2) - c(3) - \cdots - c(i_o - 2)$.
- It is clear that using an RLS updating equation, we must contemplate additional conditions in order to guarantee a suitable estimate of the inverse of the temporal covariance matrix [34].
- Additionally, a more complete linear part can be easily contemplated if Laguerre or Kautz bases, with fixed poles, are used to extend the linear FIR in our model [12].

4.2. SPWL filter implementation

The SPWL filter can be implemented directly using the expressions (6) and (9). The only particularity in the implementation is the evaluation of absolute values. A typical implementation is depicted in Figure 3. Note that it involves a lower number of parameters than the required in the LUPWL approach. This last realization is illustrated in its simplest form in Figure 4. This is an important difference because both realizations allow a description of the same systems. As previously stated, the main difference between both implementations are that in the proposed realization we need the implementation of a single FIR filter.

Other variants rather than an LMS-based algorithm can be used to implement the updating algorithm. The LMS-based algorithm was used mostly to exemplify the characteristics and convergence of the SPWL description.

A key aspect in the application of the SPWL algorithm is the selection of the partition β_i for $i = 1, \dots, \sigma$. Conceptually, the algorithm is formulated for a fixed set of grid points. First, the interval $[\beta_1, \beta_\sigma]$ must coincide well with the range of the signal $v(k)$, even though the linear part of the nonlinear Wiener model may be time varying. Other problems may arise with the interior grid point selection.

The first problem mentioned above is solved selecting the interval $[\beta_1, \beta_\sigma]$ wide enough, based on the excursion of the input signal and the expected variation of the linear filter parameters. However, due to the bounded behavior of the nonlinearity, we do not expect a malfunction if the input signal exceeds this range.

Having chosen $[\beta_1, \beta_\sigma]$, it remains to determine the interior points. As a general rule, it is clear that a higher density of points is needed in intervals where the slope of the nonlinearity changes significantly than in intervals where the behavior is close to linear.

With respect to the step size, and following the ideas of [23], it is possible to bound the parameters μ_h and μ_c as

$$\frac{1}{\left(\sum_{i=2}^{\sigma+1} c_i^2\right) \lambda_{\max}} > \mu_h > 0, \tag{22}$$

where $\lambda_{\max}$ is the maximum eigenvalue of $\boldsymbol{R}_x$, and

$$\frac{1}{\left(1 + \sum_{i=1}^{\sigma-1} (\beta_\sigma - \beta_i)^2\right)} > \mu_c > 0, \tag{23}$$

respectively. The simplest initial condition for the linear parameters $\boldsymbol{h}$ can be assumed to be a null vector and, on the other hand, for the nonlinear parameters $\boldsymbol{c}$, they can be chosen to define a unit gain at the static nonlinearity, i.e., $\boldsymbol{c}_1 = \beta_1$, $\boldsymbol{c}_2 = 1$, and $\boldsymbol{c}_i = 0$ for $i \neq 0, 1$.

Based on a fixed slope sector k_o, a simple choice of μ_c is related to that performed for an FIR model [10]. However, the choice of μ_h depends on the excursion of the signals related to the PWL model. For this reason, it is more related to the specific nonlinearity modeled. The examples presented in the following section are addressed to clarify this point.

5. Simulation examples

The performance of the SPWL adaptive filter discussed in the previous sections has been tested using computer simulations for different types of models. Some

Table 2. Total number of parameters for each algorithm used in the comparisons

Example	Volterra	FIR	LUPWL	SPWL
1	3	1	5	5
2	92	8	32	10 ($\sigma = 2$)
				16 ($\sigma = 8$)
3	25	5	15	8 ($\sigma = 2$)
				13 ($\sigma = 8$)

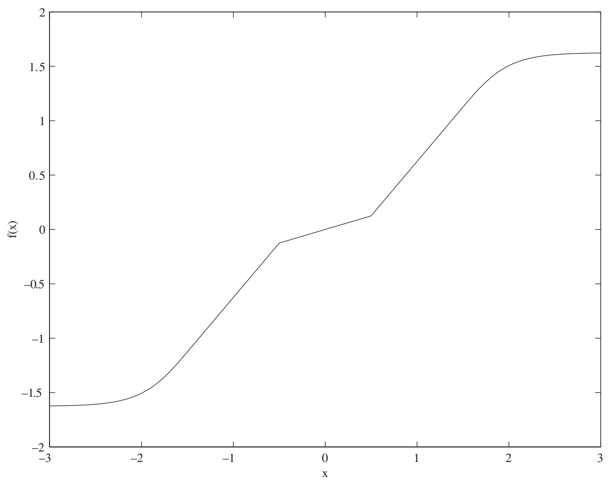

Figure 5. Input–output characteristic for example 1.

results are summarized in the following three examples, where the simplicial nonlinear adaptive filter proposed is compared with a linear filter, a Volterra approach, and the LUPWL realization. In all cases, the choice of the step size factors was performed to optimize convergence speed. For comparison purposes, the total number of parameters used in each example for the realizations evaluated is summarized in Table 2. The examples consider a white noise input, and the resulting learning curves are the result of 100 averaged squared error runs.

Example 1. The desired output is obtained from a memoryless nonlinear device with a response composed of an insensitive start region, a linear region, and a saturation region, as illustrated in Figure 5. The input signal is a Gaussian white random process with zero mean and variance $\sigma_x^2 = 2$. The step size parameters used for the SPWL in this example are $\mu_h = 0.005$ and $\mu_c = 0.05$.

This example illustrates the capability of each approach to approximate a given nonlinearity. Figure 6 depicts the mean-squared error (MSE) learning curves for

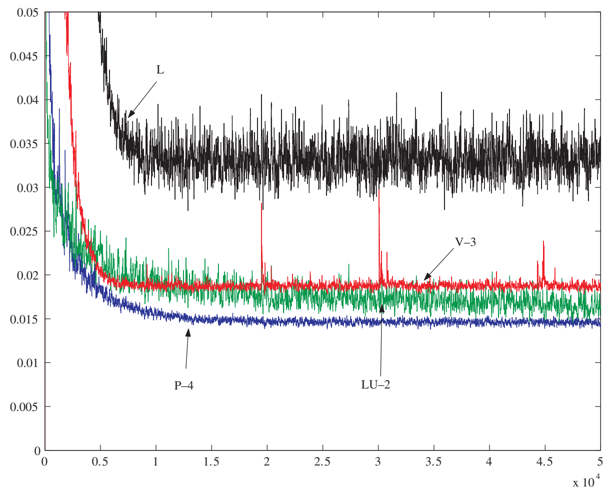

Figure 6. MSE learning curves for Example 1. $P-4$ SPWL realization proposed ($\sigma = 4$). $V-3$ 3rd-order Volterra realization. $LU-2$ LUPWL realization ($\sigma = 2$). L LMS-based FIR filter.

the proposed filter, a Volterra filter of order 3, and the LUPWL approach with $\sigma = 2$.

For this example, the number of parameters is not critical (a static gain). It could be expected for this example that, due to the lower number of parameters but similar modeling capabilities, the SPWL filter converges faster than the LUPWL realization. This behavior can be verified in the computer simulations, where convergence for the LUPWL realization was not fully achieved. Better modeling could be obtained with the Volterra filter, but ill-conditioning in this lower-order case is just apparent. Logically, the FIR model is not able to model the nonlinear input-output characteristic.

Example 2. In this example, the desired output was obtained using a nonlinear Wiener model, where [23]

$$\begin{aligned} v(k) &= 0.06 * x(k) + 0.1 * x(k-1) + 0.22 * x(k-2) \\ &\quad + x(k-3) + 0.27 * x(k-4) + 0.13 * x(k-5) \\ &\quad + 0.08 * x(k-6) + 0.07 * x(k-7). \end{aligned}$$

Also,

$$y(k) = \frac{e^{v/0.25} - 1}{e^{v/0.25} + 1},$$

and $x(n)$ is a unit variance white Gaussian random process. The step size parameters used for the SPWL filter in this example are $\mu_h = 0.005$ and $\mu_c = 0.01$.

Figures 7 and 8 depict the MSE learning curves for the proposed filter (for

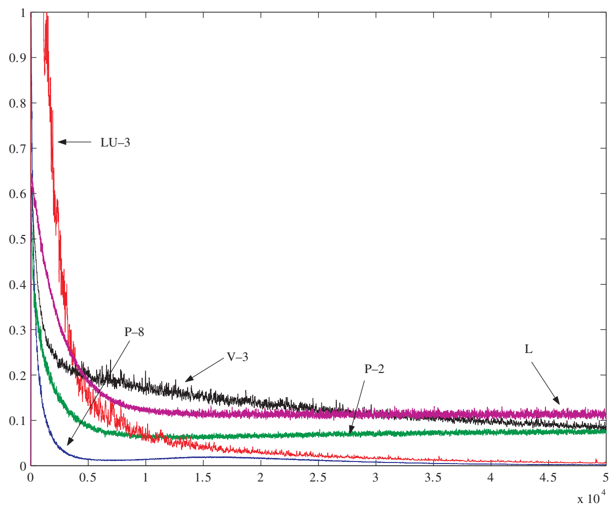

Figure 7. MSE learning curves for Example 2. $P - 8$ SPWL realization proposed ($\sigma = 8$). $P - 2$ SPWL realization proposed ($\sigma = 2$). $V - 3$ 3rd-order Volterra realization. $LU - 3$ LUPWL realization ($\sigma = 3$). L LMS-based FIR filter.

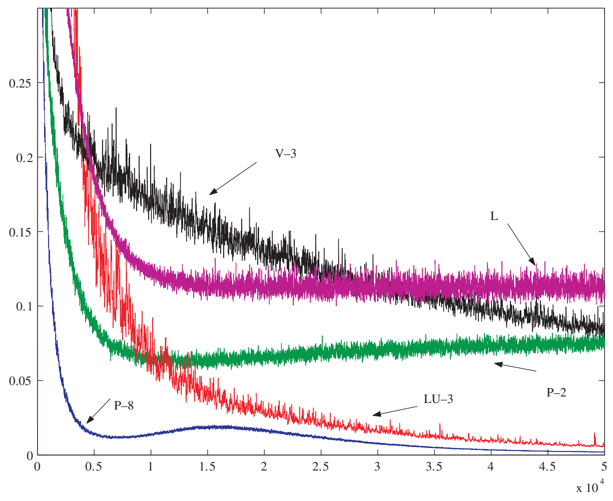

Figure 8. Detailed MSE learning curves for Example 2.

$\sigma = 2$ and $\sigma = 8$), an LMS-based linear filter, a Volterra filter of order 3, and the LUPWL realization with $\sigma = 3$.

In this case, the number of parameters is critical. The linear LMS-based filter

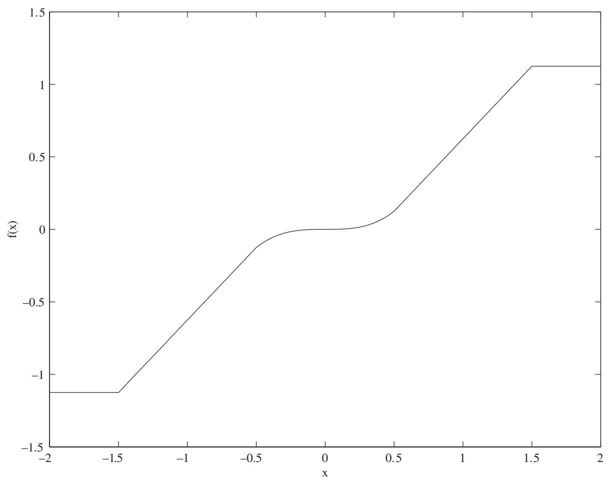

Figure 9. Input–output characteristic for example 3.

has 8 taps. The LUPWL filter uses 32 parameters. The Volterra filter uses 92 coefficients, and the proposed filter uses 16 parameters for the case of $\sigma = 8$.

For this complete model, it can be expected that the number of partitions necessary with the SPWL realization will be higher than with the LUPWL filter in order to obtain a similar convergence speed. This can be verified in the illustrations, where using only 2 partitions the SPWL does not converge to an MSE similar to the LUPWL realization. However, increasing the number of partitions (without increasing noticeably the number of parameters) in the SPWL realization, the convergence speed is even faster than for the LUPWL filter. In addition, the Volterra filter, due to the huge number of parameters necessary, has the lowest convergence speed, and the linear filter the highest MSE floor.

Example 3. Finally, we present the results of simulations with a Hammerstein model (a memoryless nonlinear subsystem followed by a linear system). The nonlinear subsystem is illustrated in Figure 9. The corresponding linear part to complete the model is given by the transfer function: $(1 - 0.7z^{-1})^2(1 + 0.7z^{-1})^2$.

Figure 10 depicts the MSE learning curves of the proposed filter with $\sigma = 8$, an LMS-based linear filter (with 5 taps), a Volterra filter of order 3, and the LUPWL approach with $\sigma = 2$. The step size parameters used for the SPWL filter in this example are $\mu_h = 0.005$ and $\mu_c = 0.05$. Note that in this case the system to be identified is not in the model set, and as a consequence the modeling capabilities of the SPWL realization and LUPWL realizations are not equivalent. Then, due to the higher number of parameters, we expect a better performance of the LUPWL filter. This can be verified by the illustration, where, even in this rather extreme case, the performance of the SPWL filter is yet reasonable, and the number of

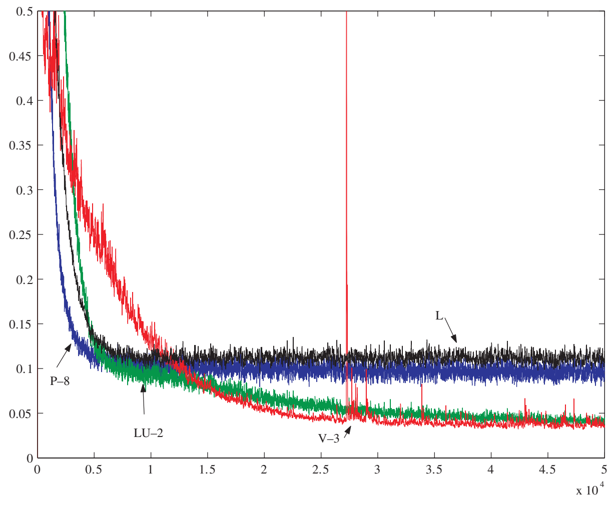

Figure 10. MSE learning curves for Example 3. $P-8$ SPWL realization proposed ($\sigma = 8$). $V-3$ 3rd-order Volterra realization. $LU-2$ LUPWL realization ($\sigma = 2$). L LMS-based FIR filter.

parameters is lower than for the other cases. The Volterra filter in this example shows a considerably ill-conditioned behavior.

6. Conclusions

A nonlinear Wiener structure for nonlinear adaptive filtering was studied in the context of an SPWL approach with a simplicial partition. The resulting structure requires fewer parameters than others found in the literature and presents an excellent convergence rate. The lower computational complexity is related to the particular partition of the domain chosen, solving the compromise of computational complexity by exchanging computational complexity (reducing a bank of FIR filters to only one FIR filter) for a higher number of parameters of design, i.e., the borders of the simplicial partitions. Although some basic results related to convergence and modeling capabilities are introduced, further research is being done to apply this model to more general problems other than the static nonlinearity following a linear filter considered. Neural networks models are of particular interest from an application point of view.

References

[1] S. Boyd and L. O. Chua, Fading memory and the problem of approximating nonlinear operators with Volterra series, *IEEE Trans. Circuits and Systems*, vol. 32, 1150–1161, 1985.

[2] M. Chien and E. Kuth, Solving nonlinear resistive networks using piecewise-linear analysis and simplicial subdivision, *IEEE Trans. Circuits and Systems Part I*, vol. 24, pp. 305–317, June 1997.

[3] L. O. Chua, Efficient computer algorithms for piecewise-linear analysis of resistive nonlinear networks, *IEEE Trans. Circuits Theory*, vol. 18, pp. 73–85, Jan. 1971.

[4] L. O. Chua and A. C. Deng, Canonical piecewise-linear representation, *IEEE Trans. Circuits and Systems,* vol. 35, pp. 101–111, 1988.

[5] L. O. Chua and S. Kang, Section wise piecewise-linear functions: Canonical representation, properties and applications, *Proc. IEEE*, vol. 65, pp. 915–929, 1977.

[6] R. J. P. de Figueiredo and T. A. Dwyer III, A best approximation framework and implementation for simulation of large-scale nonlinear systems, *IEEE Trans. on Circuits and Systems,* vol. CAS 27, No.11, pp.1005–1014, 1980.

[7] R. J. P. de Figueiredo, A new functional analytic framework for modeling artificial neural networks, *Proc. of the 1990 IEEE International Symposium on Circuits and Systems*, New Orleans, LA, pp. 723–726, May, 1990.

[8] R. J. P. de Figueiredo, A Reproducing Kernel Hilbert Space (RKHS) approach to the optimal modeling, identification, and design of nonlinear adaptive systems, Plenary Lecture, *Proc. of Adaptive Systems for Signal Processing, Communications, and Control Symposium 2000, AS-SPCC- The IEEE 2000*, 1-4, pp.42–47, October, 2000.

[9] R. J. P. de Figueiredo, Beyond Volterra and Wiener: Optimal modeling of nonlinear dynamical systems in a neural space for applications in computational intelligence, *Computational Intelligence: The Experts Speak*, edited by D.B. Fogel and C.J. Robinson, IEEE Press, and Wiley International, Piscataway, NJ, 2003.

[10] P. S. R. Diniz, *Adaptive Filtering: Algorithms and Practical Implementation,* Kluwer Academic Publishers, Norwell, MA, 1997.

[11] T. Eltoft and R.J.P. de Figueiredo, "A DCT-Based D-FANN for nonlinear adaptive time series prediction," *IEEE Trans. on Circuits and Systems, Part II*, pp. 1131-39, 2000.

[12] Z. Fejzo and H. Lev-Ari, Adaptive Laguerre-lattice filters, *IEEE Trans. Signal Process.*, vol. 45, no. 12, pp. 3006–3016, Dec. 1997.

[13] T. Fujisawa and E. S. Kuh, Piecewise-linear theory of nonlinear networks, *SIAM J. Appl. Math.,* vol. 22, pp. 307–328, 1972.

[14] F. Girosi, M. Jones, and T. Poggio, Regularization theory and neural networks architecture, *Neural Computation*, vol. 7, pp. 219–269, 1995.

[15] G. H. Golub and C. F. Van Loan, *Matrix Computations*, The John Hopkins University Press, Baltimore, MD, 1983.

[16] P. Julian, High level canonical piecewise linear representation: Theory and applications, Ph.D. thesis in Systems Control, Universidad Nacional del Sur, UMI Dissertation Services, 1999.

[17] P. Julian, A. Desages, and O.Agamennoni, High level canonical piecewise linear representation using a simplicial partition, *IEEE Trans. Circuits and Systems Part I*, vol. 46, pp. 463–480, 1999.

[18] P. Julian, A. Desages, and B. D'Amico, Orthonormal high-level canonical PWL functions with applications to model reduction, *IEEE Trans. Circuits and Systems Part I*, vol. 47, No. 5, pp. 702–712, May 2000.

[19] P. Julian, M. Jordan, and A. Desages, Canonical piecewise linear approximation of smooth funtions, *IEEE Trans. Circuits and Systems I*, vol. 45, pp. 567–571, May 1996.

[20] C. Kahlert and L. O. Chua, A generalized canonical piecewise-linear representation, *IEEE Trans. Circuits and Systems*, vol. 37, pp. 373–383, March 1990.

[21] J. Katzenelson, An algorithm for solving nonlinear networks, *Bell Syst. Tech. J.*, vol. 44, pp. 1605–1620, 1965.

[22] T. Koh and E. J. Powers, An adaptive nonlinear digital filter with lattice orthogonalization, *Proc. IEEE Int. Conf. on Acoust., Speech and Signal Process*, pp. 37–40, 1983.

[23] J. N. Lin and R. Unbehauen, Adaptive nonlinear digital filter with canonical piecewise-linear

structure, *IEEE Trans. Circuits and Systems*, vol. 37, pp. 347–353, 1990.

[24] J. N. Lin, H. Xu, and R. Unbehauen, A generalization of canonical piecewise-linear functions, *IEEE Trans. Circuits and Systems*, vol. 41, pp. 345–347, April 1994.

[25] L. Ljung and T. Soderstrom, *Theory and Practice of Recursive Identification*, MIT Press, Cambridge, MA, 1983.

[26] F. L. Luo and R. Unbehauen, *Applied Neural Networks for Signal Processing*, Cambridge University Press, Cambridge, U.K., 1996.

[27] D. M. Mansour and A. M. Gray, Jr., Frequency domain nonlinear filter, *Proc. IEEE Int. Conf. on Acoust., Speech and Signal Process.*, pp. 550–553, 1981.

[28] V. J. Mathews, Adaptive polinomial filters, *IEEE Signal Process. Mag.*, pp. 10–26, July 1991.

[29] P. Mundkur and R. J. P. de Figueiredo, Scaled simplicial approximation for inversion of Gaussian RBF expansions, *Proc. of the 2001 IEEE International Symposium on Circuits and Systems* 6–9, Vol. 3, pp. 89-92, May 2001.

[30] S. K. Sin and R. J. P. de Figueiredo, An evolution oriented algorithm for the optimal interpolative net, *IEEE Trans. Neural Network*, vol. 3, No. 2, pp. 315–323, 1992.

[31] S. K. Sin and R. J. P. de Figueiredo, Efficient learning procedures for optimal interpolating networks *Neural Network*, vol. 6, pp. 99–113, 1993.

[32] G. L. Sicuranza and G. R. Ramponi, Adaptive nonlinear digital filters using distributed arithmetic, *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 34, pp. 518–526, 1986.

[33] T. Wigren, Convergence analysis of recursive identification algorithms based on the nonlinear Wiener model, *IEEE Trans. Automat. Control*, vol. 38, No. 11, pp. 2191–2206, Nov. 1994.

[34] T. Wigren, Recursive prediction error identification using the nonlinear Wiener model, *Automatica*, vol. 29, no. 4, pp. 1011–1025, 1993.

[35] T. Wigren, Output error convergence of adaptive filters with compensation for output nonlinearities, *IEEE Trans. Automat. Control*, vol. 43, no. 7, pp. 975–978, July 1998.