# Low-Complexity Nonlinear Adaptive Filter Based on a Pipelined Bilinear Recurrent Neural Network

Haiquan Zhao, *Member, IEEE*, Xiangping Zeng, and Zhengyou He, *Member, IEEE*

***Abstract*— To reduce the computational complexity of the bilinear recurrent neural network (BLRNN), a novel low-complexity nonlinear adaptive filter with a pipelined bilinear recurrent neural network (PBLRNN) is presented in this paper. The PBLRNN, inheriting the modular architectures of the pipelined RNN proposed by Haykin and Li, comprises a number of BLRNN modules that are cascaded in a chained form. Each module is implemented by a small-scale BLRNN with internal dynamics. Since those modules of the PBLRNN can be performed simultaneously in a pipelined parallelism fashion, it would result in a significant improvement of computational efficiency. Moreover, due to nesting module, the performance of the PBLRNN can be further improved. To suit for the modular architectures, a modified adaptive amplitude real-time recurrent learning algorithm is derived on the gradient descent approach. Extensive simulations are carried out to evaluate the performance of the PBLRNN on nonlinear system identification, nonlinear channel equalization, and chaotic time series prediction. Experimental results show that the PBLRNN provides considerably better performance compared to the single BLRNN and RNN models.**

***Index Terms*— Bilinear recurrent neural network, pipelined architecture, pipelined recurrent neural network, real-time recurrent learning, Volterra filter.**

## I. Introduction

IN NONLINEAR adaptive signal processing fields (including nonlinear system identification, nonlinear and nonstationary signal prediction, communication nonlinear channel equalization, and echo and noise cancelation), nonlinear adaptive filtering techniques have received considerable interest in recent years [1], [2]. In order to achieve certain predefined design goals, numerous researchers have contributed to its developments. Various kinds of nonlinear filter design approaches are proposed to model and approximate nonlinear systems [1], [2]. But, up to now, it is difficult to find a unified theory for accurately modeling and characterizing them. Previous nonlinear approaches can be generally classified into two categories in accordance to the employed techniques: Volterra filter [1] and neural network (NN) [2]. Due to the powerful capability to represent nonlinear systems, the well-known Volterra filter has become very popular recently. However, one of major problems of the Volterra filter is the heavy computational complexity of its implementation. Only its low order (such as second-order, third-order) is feasible in practical engineering [1].

In a wide range of engineering applications, due to the prime advantages (the capability to learn based on optimization of an appropriate error function and excellent performance for approximation of nonlinear functions [2]), many types of NNs [including multilayer percetron (MLP) [3], radius basis function (RBF) [4], functional linked artificial neural network [5], wavelet neural network (WNN) [6], [7], and recurrent neural network (RNN) [8]] have been successfully used for modeling complex nonlinear systems. Among them, research results show that the RNN can outperform feedforward neural networks (FNNs) such as MLP or RBF networks [9]. As infinite impulse response (IIR) filters with feedback, RNNs can yield smaller structures than FNNs in the same way that IIR filters can replace longer finite impulse response filters. Moreover, due to its feedbacks, RNN has dynamic characteristics. Therefore, these can enable it to acquire accurately nonlinear dynamic models, which are suitable for nonlinear prediction, nonlinear channel equalization [10], and modeling dynamical systems [9]. The most popular algorithms-based RNNs are the backpropagation through time (BPTT) [11], recurrent backpropagation [12], and real time recurrent learning (RTRL) [13] algorithms. The RTRL algorithm is attractive in that it is applicable to real time systems. But, it suffers from low convergence speed. In [14], Mak *et al.* reviewed various approaches to improve the RTRL algorithm and group them into common frameworks. Mandic *et al.* proposed a normalized real time recurrent learning (NRTRL) algorithm for RNNs [15], and this algorithm without significant demands on additional computational complexity was shown to impose additional stability and faster convergence to the RTRL. To further improve RNN training speed, Song *et al.* proposed a new robust adaptive gradient-descent (RAGD) algorithm of the RNN in terms of less discrete time steps of the transit and smaller steady-state error [16]. The weight convergence and $L_2$-stability of the RAGD algorithm were derived by the conic sector theorem. Furthermore, these works were extended to a class of multiple-input-multiple-output discrete time nonlinear systems [17]. In [18], a normalized adaptive recurrent learning to obtain a tradeoff between convergent speed and training error was proposed. Based on the RTRL algorithm, Liu *et al.* presented a framework to substantially reduce the

Manuscript received October 8, 2010; revised June 25, 2011; accepted June 27, 2011. Date of publication July 29, 2011; date of current version August 31, 2011. This work was supported in part by the National Science Foundation of China under Grant 61071183, by the Fundamental Research Funds for the Central Universities under Grant SWJTU11ZT07, and by the Doctoral Innovation Fund of Southwest Jiaotong University.

H. Zhao (corresponding author) is with the School of Electrical Engineering, Southwest Jiaotong University, Chengdu 610031, China (e-mail: hqzhao@home.swjtu.edu.cn).

X. Zeng is with the School of Information Science & Technology, Southwest Jiaotong University, Chengdu 610031, China.

Z. He is with the School of Electrical Engineering, Southwest Jiaotong University, Chengdu 610031, China.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNN.2011.2161330

resource requirement of learning in RNNs, while retaining high performance [19]. Recently, extended Kalman filter (EKF) and unscented Kalman filter (UKF) algorithms for training RNNs were presented in [20] and [21]. Fast convergence and good tracking performance are major advantages of the EKF and UKF algorithms. To capture the dynamic response of a system, combining the properties of attractor dynamics of the RNN and good convergence performance of the WNN, the recurrent wavelet neural network (RWNN) can cope with time-varying input or output through its own natural temporal operation due to the internal feedback neurons of a mother wavelet layer [22]. Chi-Huang Lu presented a design approach for stable predictive control of nonlinear discrete-time systems via RWNNs [23]. Although various algorithms and models based on the RNN have achieved high filter accuracy to a certain extent, the computational loads have largely increased.

Another disadvantage of a RNN is that it only utilizes linear input and first-order recurrent term while it fails to utilize the high-order terms of inputs, and nonlinear modeling capability of RNN is limited. In order to alleviate the inherent limitation of the RNN, the bilinear recurrent neural network (BLRNN) was proposed to solve the time series prediction problems by Park [24]. In fact, the BLRNN based on the bilinear polynomial model is regarded as a special case of the high-order neural networks (HONNs), and is capable of approximating arbitrary dynamical systems with great parsimony in the use of coefficient than other HONNs [25]. The BLRNN has been successfully applied in prediction of MPEG video traffic over asynchronous transfer mode (ATM) networks [26], short-term load forecasting [27], and nonlinear channel equalization [28]. To obtain faster convergence in training the BLRNN, a structure simplification was presented whereby the multiplications for bilinear components in the BLRNN are removed. Moreover, the computational burdens of the BLRNN are reduced to a certain extent [29]. Recently, a multiscale BLRNN, based on a wavelet-based NN architecture formulated by a combination of several individual BLRNN models, was proposed to speed up the convergence and improve the forecasting performance [30]. Even though the BLRNN shows promising results when applied to various problems, it still suffers from the heavy computational loads as the RNNs.

Consequently, we now focus on developing a computational efficient nonlinear adaptive filter-based BLRNN in this paper. In 1995, a computationally efficient modular nonlinear adaptive filter-based pipelined recurrent neural network (PRNN) was proposed to process highly nonlinear and nonstationary signals by Haykin and Li [31]. The design of the pipelined architecture follows the important engineering principle of divide and conquer and the biological principle of NN modules. Its significant merit is relatively low computational complexity. For a given number of neurons $N$, a PRNN with $M$ modules requires $O(MN^4)$ arithmetic operations, while an RNN with $MN$ neurons requires $O(M^4N^4)$ operations. Hence, various types of nonlinear filters-based PRNNs have been successfully used for a variety of applications where complexity and nonlinearity poses major problems, including speech processing [32]–[35], ATM, traffic modeling [36], nonlinear dynamic system [37], and communications [38]. Therefore, to reduce the computational complexity of the BLRNN, combining enhanced structure of the PRNN, a novel nonlinear adaptive filter-based pipelined bilinear recurrent neural network (PBLRNN) is proposed in this paper. The proposed nonlinear filter inherits the advantages of the PRNN with low computational complexity. Moreover, it has also some crucial properties such as decomposition of the complex problem, better spatial representation of the temporal information, and enriched dynamics, rendering them suitable for modeling nonlinear autoregressive moving average with exogenous inputs (NARMAX) processes.

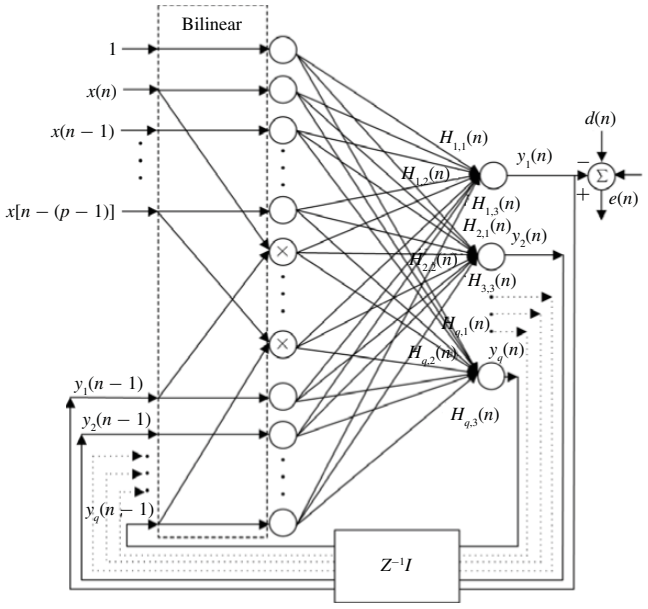


Fig. 1. BLRNN model with $p$ input and $q$ neurons.

The rest of this paper is organized as follows. Section II introduces a brief review of the BLRNN. The structure of the novel low complexity nonlinear adaptive filter is presented in Section III. In Section IV, a modified adaptive amplitude RTRL algorithm is derived by the gradient descent method. The effectiveness of the proposed nonlinear filter is illustrated by comparing with BLRNN and RNN filters for nonlinear dynamic system identification, nonlinear channel equalization, and nonlinear chaotic signals prediction in Section V. Finally, Section VI is devoted to a brief summary and discussion.

## II. Bilinear Recurrent Neural Network

It is well known that the BLRNN model is designed for an improvement of the traditional MLP and RNN models. In fact, in contrast to the RNN, the BLRNN is an extended version of the RNN. Without losing generality, a simple BLRNN with $p$ input and $q$ hidden neurons is depicted in Fig. 1.

In the network, its output depends not only on the current inputs, but also on recurrent signals. The extended input $[(1 + p + q + pq) \times 1]$ vector $\mathbf{X}(n)$ by the bilinear polynomial is defined by

$$\mathbf{X}(n) = \left[1, \mathbf{X}_E^T(n), \mathbf{X}_{EF}^T(n), \mathbf{X}_F^T(n)\right]^T \tag{1}$$

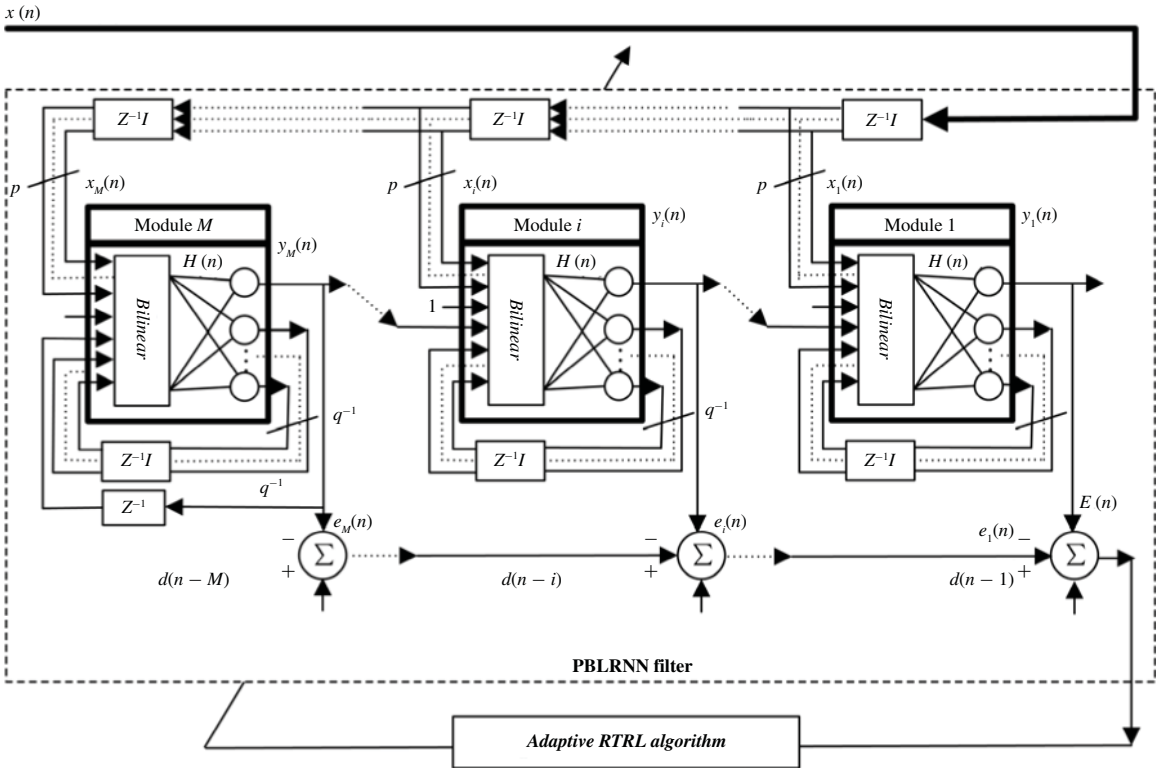


Fig. 2. PBLRNN with $M$ modules.

where $T$ denotes transpose of a vector or matrix, external input signal $(p \times 1)$ vector $\mathbf{X}_E(n)$, the recurrence signal $(q \times 1)$ vector $\mathbf{X}_F(n)$ and $(p \times q)$ vector $\mathbf{X}_{EF}(n)$ are respectively defined by

$$\mathbf{X}_E(n) = [x(n), \ldots, x(n-p+1)]^T \tag{2}$$

$$\mathbf{X}_F(n) = [y_1(n-1), \ldots, y_q(n-1)]^T \tag{3}$$

$$\mathbf{X}_{EF}(n) = [x(n)y_1(n-1), \ldots, x(n)y_q(n-1), \ldots, x(n-p+1)y_q(n-1)]^T. \tag{4}$$

Thus, the output $y_k(n)$ of the $k$th neuron in BLRNN model is computed by

$$y_k(n) = \varphi(U_k(n)) \quad k = 1, 2, \ldots, q \tag{5}$$

where $\varphi(\bullet)$ is a nonlinear activation function, and its input $U_k(n)$ is given by

$$\begin{aligned} U_k(n) &= \mathbf{X}^T(n)\mathbf{H}_k(n) \\ &= \left[1\mathbf{X}_E(n)^T\right]\mathbf{H}_{1,k}(n) + \mathbf{X}_{EF}^T(n)\mathbf{H}_{2,k}(n) \\ &\quad + \mathbf{X}_F^T(n)\mathbf{H}_{3,k}(n) \\ &= h_{1,k,0}(n) + \sum_{i=1}^{p} h_{1,k,i}(n)x(n-i+1) \\ &\quad + \sum_{i=1}^{p}\sum_{j=1}^{q} h_{2,k,q(i-1)+j}(n)x(n-i+1)y_j(n-1) \\ &\quad + \sum_{i=1}^{q} h_{3,k,i}(n)y_i(n-1) \end{aligned} \tag{6}$$

where the coefficient vectors $\mathbf{H}_k(n)$ are elements in a coefficient matrix $\mathbf{H}(n)$, which is defined by

$$\mathbf{H}(n) = \left[\mathbf{H}_1^T(n), \ldots, \mathbf{H}_k^T(n), \ldots, \mathbf{H}_q^T(n)\right] \tag{7}$$

and

$$\mathbf{H}_k(n) = \left[\mathbf{H}_{1,k}^T(n), \mathbf{H}_{2,k}^T(n), \mathbf{H}_{3,k}^T(n)\right]^T \tag{8}$$

where $\mathbf{H}_{1,k}(n)$, $\mathbf{H}_{2,k}(n)$, and $\mathbf{H}_{3,k}(n)$ are coefficients of $\mathbf{X}_E(n)$, $\mathbf{X}_{EF}(n)$, and $\mathbf{X}_F(n)$, respectively.

Finally, the output of the BLRNN is $y_1(n)$ where $k = 1$.

## III. Low-Complexity PBLRNN

Inspired by the pipelined architecture of the PRNN [31], the low-complexity PBLRNN is suggested as shown in Fig. 2, which is a modular network that consists of $M$ identical BLRNN modules cascaded in a nested manner. Moreover, the cascaded interconnection of the PBLRNN adheres to the principle of divide-and-conquer. That is, to cope with a complex modeling problem, it should be decomposed into several subtasks that are easier to handle. Moreover, due to the modules performed simultaneously in a pipelined parallelism fashion, it leads to a significant improvement in its total computational efficiency.

Each module of the PBLRNN has the same form of a BLRNN as depicted in Fig. 3, which describes the detailed structure of module $i$. At time $n$, the external input signal $\mathbf{X}_{E,i}(n)$ that is applied to module $i (i = 1, 2, \ldots, M)$ is a vector comprising $p$ past signal values, defined by

$$\mathbf{X}_{E,i}(n) = [x(n-i), x(n-(i+1)), \ldots, x(n-(i+p-1))]^T \tag{9}$$

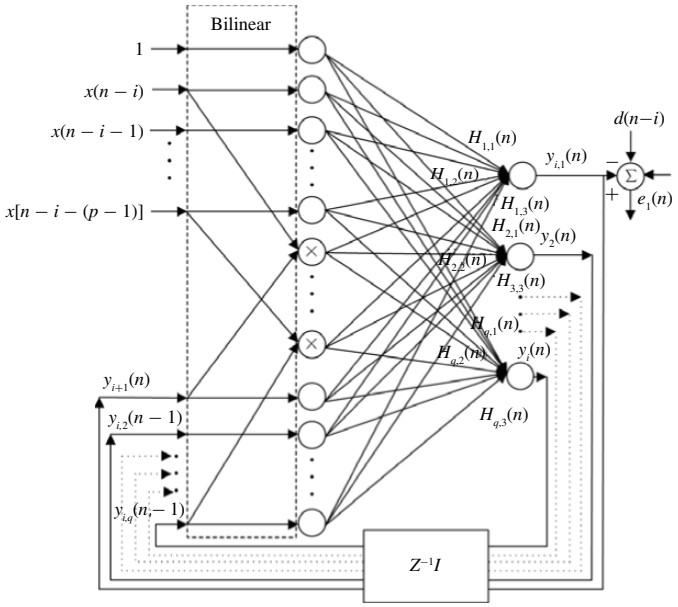


Fig. 3. Detailed architecture of module $i$ of the PBLRNN.

and is delayed by $Z^{-i}I$ at the input of the module $i$, where $Z^{-i}$ denotes the delay operator $i$ time units, and $I$ is the $(p \times p)$-dimensional identity matrix. Apart from the $p$ external inputs, each module of the PBLRNN has $q-1$ feedback inputs of its own, which are one-unit delayed of the internal signals. Thus, these signals applied to module $i$ constitute the vector $\mathbf{r}_i(n)$ which is the $q$-by-1 feedback vector as follows:

$$\mathbf{r}_i(n) = [y_{i+1,1}(n), \hat{\boldsymbol{r}}_i(n)]^T, \ i = 1, 2, \ldots, (q-1) \tag{10}$$

where $y_{i+1,1}(n)$ is the first neuron output in the adjacent module $i+1$. Vector $\mathbf{r}_i(n)$ is the one-step delayed output feedback signal, which originates from the module $i$ itself and is defined by

$$\hat{\mathbf{r}}_i(n) = [y_{i,2}(n-1), \ldots, y_{i,q}(n-1)]^T. \tag{11}$$

In particular, the uppermost module of the PBLRNN, namely, module $M$, operates as a standard fully connected BLRNN. The vector $\mathbf{r}_M(n)$ consists of the one-step delayed output signals in module $M$ that are fed back to itself, and is given by

$$\begin{aligned}\mathbf{r}_M(n) &= \left[y_{M,1}(n-1), \hat{\boldsymbol{r}}_M(n)\right]^T \\ &= \left[y_{M,1}(n-1), y_{M,2}(n-1), \ldots, y_{M,q}(n-1)\right]^T. \end{aligned}\tag{12}$$

Hence, the overall input $\hat{\boldsymbol{X}}_i(n)$ to module $i$ is a $[(p+q+1)\times 1]$ vector, including the external inputs $\mathbf{X}_{E,i}(n)$, The fixed input $+1$ included to accommodate a bias for each neuron and the feedback inputs $\mathbf{r}_i(n)$ are given by

$$\begin{aligned}\hat{\boldsymbol{X}}_i(n) &= \left[\hat{X}_{i,1}(n), \hat{X}_{i,2}(n), \ldots, \hat{X}_{i,p+q+1}(n)\right]^T \\ &= \left[\mathbf{X}_{E,i}^T(n), 1, \mathbf{r}_i^T(n)\right]^T. \end{aligned}\tag{13}$$

Moreover, it is clearly shown in Fig. 2 that the information flow into and out of the modules proceeds in a synchronized fashion. Consequently, all the modules in the PBLRNN operate in a similar fashion, exhibiting exactly the same number of external inputs and feedback signals, which are properly timed. Then, at the $n$th time point, input vector $\hat{\boldsymbol{X}}_i(n)$ is expanded to the vector $\mathbf{X}_i(n)$ by the bilinear polynomial, and is written as

$$\begin{aligned}\mathbf{X}_i(n) &= \left[X_{i,1}(n)X_{i,2}(n)\ldots, X_{i,L}(n)\right]^T \\ &= \big[1, x(n-i), x(n-i-1), \ldots, x(n-i-(p-1)), \\ &\quad x(n-i)r_{i,1}(n), \ldots, x(n-i-(p-1)r_{i,q}(n)), \\ &\quad r_{i,1}(n), r_{i,2}(n), \ldots, r_{i,q}(n)\big]^T \\ &= \left[\mathbf{V}_{i,1}(n)^T, \mathbf{V}_{i,2}(n)^T, \mathbf{V}_{i,3}(n)\right]^T \end{aligned}\tag{14}$$

where the length $L$ of $X_i(n)$ is defined by $L = 1 + p + q + (p \times q)$, the vectors $\mathbf{V}_{i,1}(n)$, $\mathbf{V}_{i,2}(n)$, and $\mathbf{V}_{i,3}(n)$ are written as respectively

$$\begin{aligned}\mathbf{V}_{i,1}(n) &= \left[1, x(n-i), x(n-i-1), \ldots, x(n-i-(p-1))\right]^T \\ \mathbf{V}_{i,2}(n) &= \left[x(n-i)r_{i,1}(n), \ldots, x(n-i-(p-1)r_{i,q}(n))\right]^T \\ \mathbf{V}_{i,3}(n) &= \left[r_{i,1}(n), r_{i,2}(n), \ldots, r_{i,q}(n)\right]^T. \end{aligned}\tag{15}$$

Without losing generality, all the modules are designed to have exactly the same $L$-by-$q$ synaptic weight matrix $\mathbf{H}(n)$ defined by

$$\mathbf{H}(n) = [\mathbf{H}_1(n), \ldots, \mathbf{H}_l, \ldots, \mathbf{H}_q(n)]^T \tag{16}$$

where $\mathbf{H}_l(n) = [\mathbf{H}_{l,1}(n), \mathbf{H}_{l,2}(n), \mathbf{H}_{l,3}(n)], l = 1, 2, \ldots, q$. And length $L_1$ of $\mathbf{H}_{l,1}(n)$ is same to $\mathbf{V}_{i,1}(n)$ and $L_1 = p+1$. Similarly, we define $L_2$ of $\mathbf{H}_{l,2}(n)$ as follows:

$$L_2 = p \times q \tag{17}$$

and $L_3$ of $\mathbf{H}_{l,3}(n)$is defined by $L_3 = q$.

Thus the $l$th neural unit output of module $i$ is written by

$$\begin{aligned}y_{i,l}(n) &= \varphi(U_{i,l}(n)) = \varphi(\mathbf{H}_l^T(n)\mathbf{X}_i(n)) \\ &= \varphi(\mathbf{H}_{l,1}^T(n)\mathbf{V}_{i,1}(n) + \mathbf{H}_{l,2}^T(n)\mathbf{V}_{i,2}(n) \\ &\quad + \mathbf{H}_{l,3}^T(n)\mathbf{V}_{i,3}(n)) \end{aligned}\tag{18}$$

where $U_{i,l}(n)$ denotes the net internal activity of neuron $l$ in module $i$ at the $n$th time point.

Then, the output $y_{i,1}(n)$ of the module $i$ can be interpreted as the estimate of desired signal $d(n-i)$. It is computed by a complete dependence form shown as follows:

$$\begin{aligned}y_{i,1}(n) &= \varphi\left(\mathrm{H}_1(n), \mathbf{X}_i(n), \mathbf{r}_i(n)\right) \\ &= \varphi\left[\mathbf{H}_{1,1}^T(n)\mathbf{V}_{i,1}(n) + \mathbf{H}_{1,2}^T(n)\mathbf{V}_{i,2}(n) + \mathbf{H}_{1,3}^T(n)\mathbf{V}_{i,3}(n)\right]. \end{aligned}\tag{19}$$

Finally, at time instant $n$, the output signal computed by the PBLRNN is defined by the first neuron of the first module as shown below

$$y(n) = y_{1,1}(n). \tag{20}$$

A key difference between the PBLRNN of Fig. 3 and the conventional BLRNN is that the PBLRNN is characterized by a nested nonlinearity. Owing to module nesting, the allover

output $y_{1,1}(n)$ of the PBLRNN can be expressed by

$$\begin{aligned} y(n) &= y_{1,1}(n) \\ &= \varphi(\mathbf{X}_1(n), \mathbf{r}_1(n), y_{2,1}(n)) \\ &= \varphi\{\mathbf{X}_1(n), \mathbf{r}_1(n), \varphi[\mathbf{X}_2(n), \mathbf{r}_2(n), y_{3,1}(n)]\} \\ &= \cdots \\ &= \varphi\{[\mathbf{X}_1(n), \mathbf{r}_1(n), \varphi(\mathbf{X}_2(n), \mathbf{r}_2(n), \ldots, \\ &\quad \varphi(\mathbf{X}_{M-1}(n), \mathbf{r}_{M-1}(n), y_{M,1}(n)), \ldots)]\}. \end{aligned} \tag{21}$$

Hence, it is obviously shown that the nonlinear computer power of the PBLRNN is enhanced by the nested nonlinearity [39]. But, the nesting process introduces a deteriorating effect in the relative amplitude of the output $y_{i,1}(n), i = 2, \ldots, M$ of a distant module $i$ [40]. It results in a degradation performance of nonlinear filter.

To effectively deal with the effects, according to the approaches in [41]–[43], we can rewrite the nonlinear activation function

$$\varphi(x) = \lambda(n)\varphi(x) = \frac{\lambda(n)}{\left[1 + \exp(-\beta x)\right]} \tag{22}$$

where $\lambda(n)$ is a variable that adjusts the amplitude of $\varphi(x)$, a slope $\beta$ is set to 1, and $\varphi(x)$ is the activation function with a unit amplitude. Thus, if $\lambda(n) = 1$, then $\varphi(x) = \varphi(x)$. Without loss of generality, let all the neurons in the PBLRNN share the same amplitude $\lambda(n)$ [41]–[43].

## IV. Adaptive Amplitude RTRL Algorithm

The RTRL algorithm, first proposed by Williams and Zipser [13], is one of the successful learning algorithms where the gradient of errors is propagated forward in time rather than backward in time. Moreover, researches have demonstrated that it is particularly suitable for online training of the PRNN [9]. However, slow convergence is one of its major disadvantages. Hence, to speed up convergence and overcome the effect of the nesting processing, keeping the view of the pipelined architecture, the modified adaptive amplitude RTRL algorithm is derived in this section.

In accordance with the RTRL algorithm rule [13], adjustments to the synaptic weight matrix $\mathbf{H}(n)$ is made to minimize the allover cost function $E(n)$. Thus, the value of the following $E(n)$ of the PBLRNN is calculated by

$$E(n) = \sum_{i=1}^{M} \varepsilon^{i-1} e_i^2(n) \tag{23}$$

where $\varepsilon$ is an exponential forgetting factor that lies in the range of $0 < \varepsilon \leq 1$, the inverse of $\varepsilon^{i-1}$ is a measure of the memory of the PBLRNN, and the error $e_i(n)$ of module $i$ is computed by

$$e_i(n) = d(n-i) - y_{i,1}(n). \tag{24}$$

Taking as objective the minimization of the cost function $E(n)$, the parameters $\mathbf{H}(n)$, $\lambda(n)$ of the PBLRNN are adjusted by the stochastic gradient estimation algorithm as the following formulas:

$$\begin{aligned} h_{l,1,j}(n) &= h_{l,1,j}(n-1) + \Delta h_{l,1,j}(n) + u_1 \Delta h_{l,1,j}(n-1) \\ &= h_{l,1,j}(n-1) - \frac{\eta_1}{2}\frac{\partial E(n)}{\partial h_{l,1,j}(n)} \\ &\quad + u_1\left[h_{l,1,j}(n) - h_{l,1,j}(n-1)\right] \end{aligned} \tag{25a}$$

$$\begin{aligned} h_{l,2,j}(n) &= h_{l,2,j}(n-1) + \Delta h_{l,2,j}(n) + u_2 \Delta h_{l,2,j}(n-1) \\ &= h_{l,2,j}(n-1) - \frac{\eta_2}{2}\frac{\partial E(n)}{\partial h_{l,2,j}(n)} \\ &\quad + u_2\left[h_{l,2,j}(n) - h_{l,2,j}(n-1)\right] \end{aligned} \tag{25b}$$

$$\begin{aligned} h_{l,3,j}(n) &= h_{l,3,j}(n-1) + \Delta h_{l,3,j}(n) + u_3 \Delta h_{l,3,j}(n-1) \\ &= h_{l,3,j}(n-1) - \frac{\eta_3}{2}\frac{\partial E(n)}{\partial h_{l,3,j}} \\ &\quad + u_3\left[h_{l,3,j}(n) - h_{l,3,j}(n-1)\right] \end{aligned} \tag{25c}$$

$$\begin{aligned} \lambda(n) &= \lambda(n-1) + \Delta\lambda(n) \\ &= \lambda(n-1) - \frac{\eta_4}{2}\frac{\partial E(n)}{\partial \lambda(n)} \end{aligned} \tag{26}$$

where $\eta_i (i = 1, 2, 3, 4)$ are learning rates, $u_i (i = 1, 2, 3)$ are the momentum to speed up convergence of the adaptive algorithm and $1 \leq j \leq L$.

Then, the partial derivates of $E(n)$ with respect to $h_{l,1,j}(n)$, $h_{l,2,j}(n)$, $h_{l,3,j}(n)$, and $\lambda(n)$ are respectively calculated by the following formulas:

$$\begin{aligned} \frac{\partial E(n)}{\partial h_{l,1,j}(n)} &= 2\sum_{i=1}^{M} \varepsilon^{i-1} e_i(n) \frac{\partial e_i(n)}{\partial h_{l,1,j}(n)} \\ &= -2\sum_{i=1}^{M} \varepsilon^{i-1} e_i(n) \frac{\partial y_{i,1}(n)}{\partial h_{l,1,j}(n)} \end{aligned} \tag{27a}$$

$$\begin{aligned} \frac{\partial E(n)}{\partial h_{l,2,j}(n)} &= 2\sum_{i=1}^{M} \varepsilon^{i-1} e_i(n) \frac{\partial e_i(n)}{\partial h_{l,2,j}(n)} \\ &= -2\sum_{i=1}^{M} \varepsilon^{i-1} e_i(n) \frac{\partial y_{i,1}(n)}{\partial h_{l,2,j}(n)} \end{aligned} \tag{27b}$$

$$\begin{aligned} \frac{\partial E(n)}{\partial h_{l,3,j}(n)} &= 2\sum_{i=1}^{M} \varepsilon^{i-1} e_i(n) \frac{\partial e_i(n)}{\partial h_{l,3,j}(n)} \\ &= -2\sum_{i=1}^{M} \varepsilon^{i-1} e_i(n) \frac{\partial y_{i,1}(n)}{\partial h_{l,3,j}(n)} \end{aligned} \tag{27c}$$

$$\begin{aligned} \frac{\partial E(n)}{\partial \lambda(n)} &= 2\sum_{i=1}^{M} \varepsilon^{i-1} e_i(n) \frac{\partial e_i(n)}{\partial \lambda(n)} \\ &= -2\sum_{i=1}^{M} \varepsilon^{i-1} e_i(n) \frac{\partial y_{i,1}(n)}{\partial \lambda(n)}. \end{aligned} \tag{28}$$

It is difficult to calculate $[\partial y_{i,1}(n)]/[\partial h_{l,1,j}(n)]$, $[\partial y_{i,1}(n)]/[\partial h_{l,2,j}(n)]$, and $[\partial y_{i,1}(n)]/[\partial h_{l,3,j}(n)]$ directly. According to the recursive algorithm presented in [44] and

[45], we introduce the sensitivity terms $Ph^{i,k}_{l,1,j}(n)$, $Ph^{i,k}_{l,2,j}(n)$, and $Ph^{i,k}_{l,3,j}(n)$ $(k = 1, 2, \ldots, q)$ respectively defined by

$$Ph^{i,k}_{l,1,j}(n) = \frac{\partial y_{i,k}(n)}{\partial h_{l,1,j}(n)} \tag{29a}$$

$$Ph^{i,k}_{l,2,j}(n) = \frac{\partial y_{i,k}(n)}{\partial h_{l,2,j}(n)} \tag{29b}$$

$$Ph^{i,k}_{l,3,j}(n) = \frac{\partial y_{i,k}(n)}{\partial h_{l,3,j}(n)} \tag{29c}$$

where $h_{l,1,j}(n)\,(j = 1, 2, \ldots, L_1)$, $h_{l,2,j}(n)(j = 1, 2, \ldots, L_2)$, and $h_{l,3,j}(n)(j = 1, 2, \ldots, L_3)$ are respectively elements of $H_{l,1}(n)$, $H_{l,2}(n)$, and $H_{l,3}(n)$. Note that $Ph^{i,1}_{l,1,j}(n) = \partial y_{i,1}(n)/\partial h_{l,1,j}(n)$, $Ph^{i,1}_{l,2,j}(n) = \partial y_{i,1}(n)/\partial h_{l,2,j}(n)$, and $Ph^{i,1}_{l,3,j}(n) = \partial y_{i,1}(n)/\partial h_{l,3,j}(n)$.

Then, according to the gradient descent rule, we can obtain the following equations (detailed derivation is provided in the Appendix):

$$Ph^{i,k}_{l,1,j}(n) = \lambda(n)\phi'\left(U_{i,k}(n)\right) \left\{\delta_{l,k} v_{k,1,j}(n) + \frac{\partial y_{i+1,1}(n)}{\partial h_{l,1,j}(n)} \sum_{m=1}^{p} h_{l,2,m}(n) v_{k,1,m}(n)\right\} \tag{30a}$$

$$Ph^{i,k}_{l,2,j}(n) = \lambda(n)\phi'\left(U_{i,k}(n)\right) \left\{\delta_{l,k} v_{k,2,j}(n) + \frac{\partial y_{i+1,1}(n)}{\partial h_{l,2,j}(n)} \sum_{m=1}^{p} h_{l,2,m}(n) v_{k,1,m}(n)\right\} \tag{30b}$$

$$Ph^{i,k}_{l,3,j}(n) = \lambda(n)\phi'\left(U_{i,k}(n)\right) \left\{\delta_{l,k} v_{k,3,j}(n) + \frac{\partial y_{i+1,1}(n)}{\partial h_{l,3,j}(n)} \sum_{m=1}^{p} h_{l,2,m}(n) v_{k,1,m}(n)\right\} \tag{30c}$$

and $1 \le i < M$.

For the case when $i = M$

$$Ph^{M,k}_{l,1,j}(n) = \lambda(n)\phi'\left(U_{M,k}(n)\right)\delta_{l,k} v_{k,1,j}(n) \tag{31a}$$

$$Ph^{M,k}_{l,2,j}(n) = \lambda(n)\phi'\left(U_{M,k}(n)\right)\delta_{l,k} v_{k,2,j}(n) \tag{31b}$$

$$Ph^{M,k}_{l,3,j}(n) = \lambda(n)\phi'\left(U_{M,k}(n)\right)\delta_{l,k} v_{k,3,j}(n). \tag{31c}$$

As a consequence, the updated weight equations are summarized as follows:

$$h_{l,1,j}(n) = h_{l,1,j}(n-1) + \eta_1 \sum_{i=1}^{M} \varepsilon^{i-1} e_i(n) Ph^{i,1}_{l,1,j}(n) + u_1\left(h_{l,1,j}(n) - h_{l,1,j}(n-1)\right) \tag{32a}$$

$$h_{l,2,j}(n) = h_{l,2,j}(n-1) + \eta_2 \sum_{i=1}^{M} \varepsilon^{i-1} e_i(n) Ph^{i,1}_{l,2,j}(n) + u_2\left(h_{l,2,j}(n) - h_{l,2,j}(n-1)\right) \tag{32b}$$

$$h_{l,3,j}(n) = h_{l,3,j}(n-1) + \eta_3 \sum_{i=1}^{M} \varepsilon^{i-1} e_i(n) Ph^{i,1}_{l,3,j}(n) + u_3\left(h_{l,3,j}(n) - h_{l,3,j}(n-1)\right). \tag{32c}$$

In addition, by the gradient algorithm rule, the amplitude parameter $\lambda(n)$ is adjusted by

$$\lambda(n) = \lambda(n-1) + \Delta\lambda = \lambda(n-1) + \eta_4 \sum_{i=1}^{M} \varepsilon^{i-1} e_i(n) \phi\left(U_{i,1}(n)\right). \tag{33}$$

Consequently, the total modified RTRL algorithm for the proposed PBLRNN is summarized in the following.

### A. Initialization of the Algorithm

1) The step sizes $\eta_1$, $\eta_2$, $\eta_3$, and $\eta_4$ are set as $0 < \eta_1, \eta_2, \eta_3, \eta_4 < 2$.
2) The weight vector $\mathbf{H}(n)$ of the PBLRNN are all initialized to small random values with $\mathbf{H}(n) < 10^{-3}$.
3) For all $i$, $k$, $l$, and $j$, recursive items in (A6) and (A7) (see the Appendix) are initialized to zeros with $Ph^{i,k}_{l,1,j}(n) = Ph^{i,k}_{l,2,j}(n) = Ph^{i,k}_{l,3,j}(n) = 0$.

### B. Forward Phase

For $i = 1{:}1{:}M$.

1) Compute the expanded vector $\mathbf{X}_i(n)$ in (14) using (9)–(12).
   For $l = 1{:}1{:}q$.
   a) Compute the output $y_{i,l}(n)$ of the $l$th neuron of $i$th module at time $n$ using (18).
   b) Compute (30) or (31).
   c) Compute the corrections $\Delta h_{l,1,j}(n)$, $\Delta h_{l,2,j}(n)$, and $\Delta h_{l,3,j}(n)$ using the second item of the right-hand side in (32).
   end.
2) Compute the error $e_i(n)$ of *ith* module at time $n$ using (24).
   end.

### C. Learning Phase

The instantaneous error $E(n)$ is given by (23) and the objective is to change the weights in the direction that minimizes $E(n)$.

Then the training phase involves the following steps.

1) Update the weights $h_{l,1,j}(n)$, $h_{l,2,j}(n)$, and $h_{l,3,j}(n)$ by (32).
2) Update the common amplitude $\lambda(n)$ of the activation function of every neuron by (33).

It is well known that one of the important issues in adaptive algorithms is the convergence. The convergence of the adaptive amplitude RTRL algorithm-based gradient descent rule depends on the selection of the initial values of the learning rate parameters and the momentum term. It is difficult that we obtain bounds on the step sizes $\eta_1$, $\eta_2$, $\eta_3$, and $\eta_4$ for stable operation of the proposed PBLRNN filter. However, we can argue heuristically that these values are selected in the range $0 < \eta_1, \eta_2, \eta_3, \eta_4 < 2$ [45]. During the training process, all these parameters start with a small value. Then, the values of the parameters are increased if the value of change of error $\Delta E(n) = E(n) - E(n-1)$ is negative, and

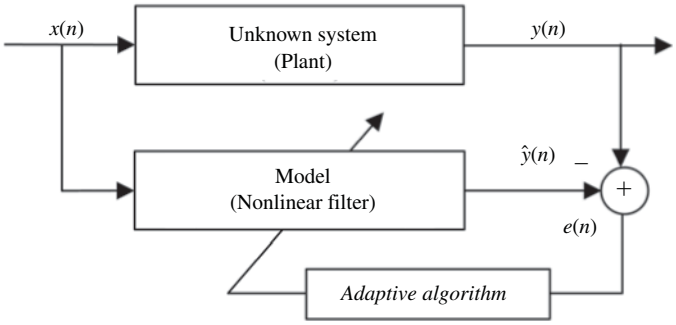


Fig. 4. Nonlinear dynamic identification scheme.

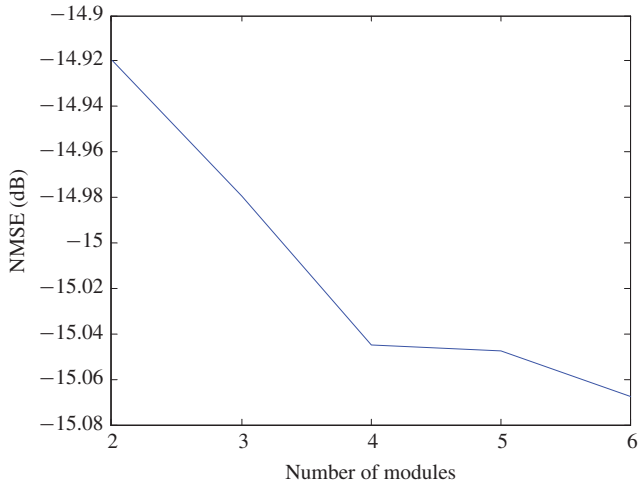


Fig. 5. NMSE versus the number of modules $M$ in the PBLRNN filter.

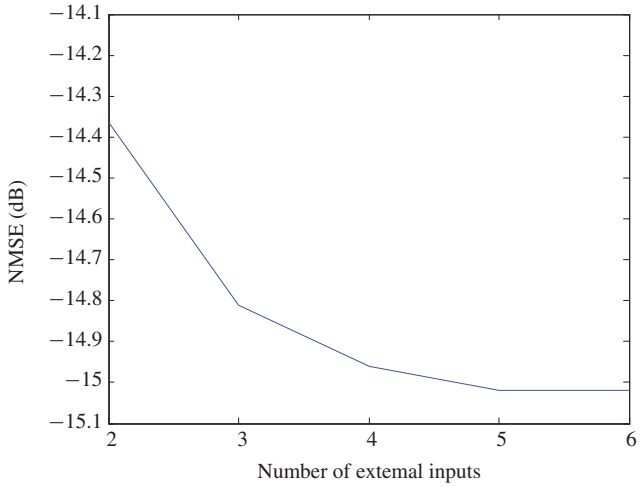


Fig. 6. NMSE versus the number of external inputs $p$.

TABLE I
PARAMETERS OF ALL THE NONLINEAR PREDICTORS

| Parameters | PBLRNN | PRNN | BLRNN | RNN |
|---|---|---|---|---|
| Learning rate | $\eta_1 = 0.2$ $\eta_2 = 0.2$ $\eta_3 = 0.2$ $\eta_4 = 0.1$ | $\eta_1 = 0.2$ $\eta_2 = 0.2$ $\eta_3 = 0.1$ | $\eta_1 = 0.07$ $\eta_2 = 0.07$ $\eta_3 = 0.07$ | $\eta = 0.05$ |
| Number of external inputs | 5 | 5 | 5 | 5 |
| Number of output layer neurons | 2 | 2 | 2 | 2 |
| Number of modules | 4 | 4 | – | – |

TABLE II
NMSEs OF ALL THE NONLINEAR FILTERS

| NN model | NMSE |
|---|---|
| **PBLRNN** | −15.9200 |
| **PRNN** | −12.7191 |
| **BLRNN** | −11.2419 |
| **RNN** | −9.4085 |

the learning rates are decreased if the change in the error $\Delta E(n) = E(n) - E(n-1)$ is positive. Therefore, this strategy leads to stable learning of the PBLRNN filter, guarantees the convergence, and speeds up the learning of the PBLRNN. Moreover, momentum terms are also employed to speed up the convergence.

## V. SIMULATION

In order to evaluate the performance of the proposed PBLRNN filter, a number of simulation studies are carried out for nonlinear dynamics systems identification, nonlinear channel equalization, and prediction of chaotic signals. Contrasting with PRNN, BLRNN, and RNN, a comparative analysis of the PBLRNN is presented in this section.

### A. Identification of Nonlinear Dynamic System

The identification problem of nonlinear dynamic system is aimed to find the nonlinear relation between the input and the output of the nonlinear dynamic system. In Fig. 4, the structure of the nonlinear dynamic identification system with the nonlinear filter is described. Here, $x(n)$ denotes the input of the system, $y(n)$ is plant output, and $\hat{y}(n)$ is output of the nonlinear filter.

In this experiment, the mathematical model of dynamic plant used in [3], [5], and [7] is described by the following difference equation:

$$y(n+1) = f\left(y(n), y(n-1), y(n-2), x(n), x(n-1)\right)$$
$$= \frac{y(n)y(n-1)y(n-2)x(n-1)\left[y(n-2)-1\right] + x(n)}{1 + y^2(n-1) + y^2(n-2)}. \tag{34}$$

The identification model of the plant is depicted by

$$\hat{y}(n+1) = NN[y(n), y(n-1), y(n-2), u(n), u(n-1)] \tag{35}$$

where $NN(.)$ denotes the neural networks used to identify the function $f$.

For the identification of the plant, an excitation sinusoidal signal is applied to the plant and the $NN$ model, and is expressed by

$$x(n) = \begin{cases} \sin\left(\frac{2\pi n}{250}\right) & \text{for } 0 < n \le 250 \\ 0.8\sin\left(\frac{2\pi n}{250}\right) + 0.2\sin\left(\frac{2\pi n}{25}\right) & \text{for } n > 250. \end{cases} \tag{36}$$

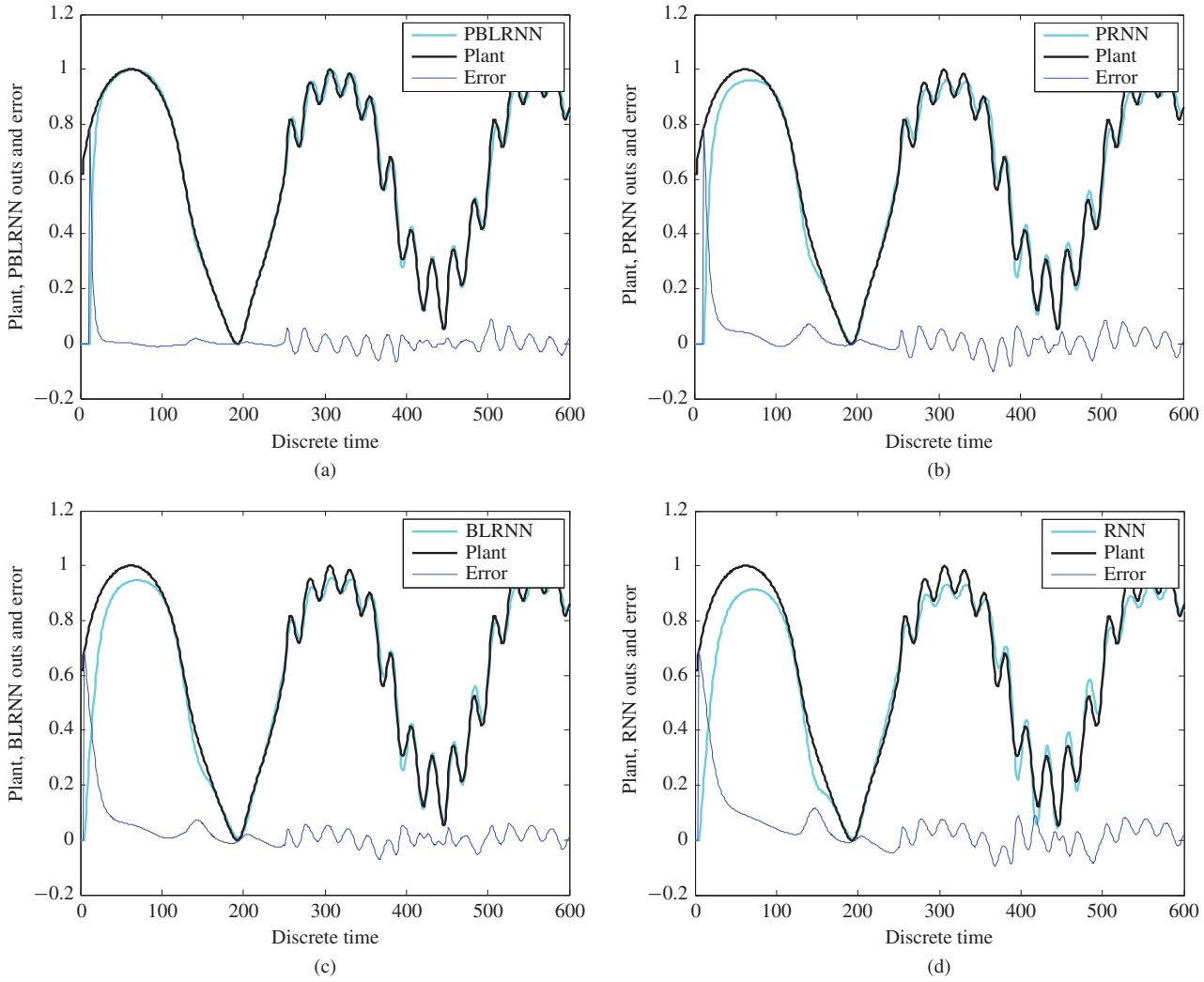


Fig. 7. Identification of the nonlinear dynamic plant with the test sinusoidal signal. (a) PBLRNN. (b) BLRNN. (c) PRNN. (d) RNN.

And the normalized mean square error (NMSE), a standard quantitative measure for performance evaluation, is defined as

$$\text{NMSE(dB)} = 10 \times \log 10 \left[ \frac{1}{\sigma^2 T_D} \sum_{n=1}^{T_D} \left[ y(n) - \hat{y}(n) \right]^2 \right] \quad (37)$$

where $\sigma^2$ denotes variance of the plant output sequence over the training duration $T_D = 600$.

*1) Effect of the Number of Modules:* Fig. 5 describes the influence of the number of modules in the PBLRNN filter on the NMSE performance, where NMSE is expressed versus $M$, for the PBLRNN models with external input $p = 5$. In all cases, the NMSE of the PBLRNN filter decreases gradually for moderate values of $M$ when $M \geq 4$. Moreover, the curve changes of NMSE shows that introducing additional modules in the PBLRNN does not markedly improve the performance further when is greater than $M = 4$. Therefore, in the simulation results that are presented in the sequel, optimal value $M$ of the number of modules for the PBLRNN is set to 4.

*2) Effect of the Number of External Inputs:* The NMSE with respect to the number of external inputs in the PBLRNN with $M = 4$ is shown in Fig. 6. For a particular level of $p$, Fig. 6 shows that the NMSE is slightly improved when $p > 5$. On the other hand, a large of number of external inputs increases the computational burdens of the PBLRNN filter. Therefore, in the following, the values $p$ are obtained by $p = 5$ for the PBLRNN filter.

According to the aforementioned discussion, the learning parameters of three NNs are chosen for several trials to obtain best results. All the parameters of the NNs are summarized in Table I, where the standard RTRL algorithms are employed for RNN, BLRNN, and PRNN, while the adjusted amplitude RTRL algorithm for the PBLRNN filter.

Fig. 7 depicts the results of the identification with the sinusoidal signal (36). We can observe that the identification of the plant is satisfactory for three networks, and also find that the estimation error of the PBLRNN filter is less than that of the BLRNN and RNN filters. To further depict the behavior of the adaptive amplitude of the activation function of the PBLRNN filter, a time variation of the adaptive amplitude is plotted in Fig. 8. It is clearly shown that adaptive amplitude RTRL algorithm is able to adapt the amplitude of the nonlinearity according to the changes in the dynamics of the input.

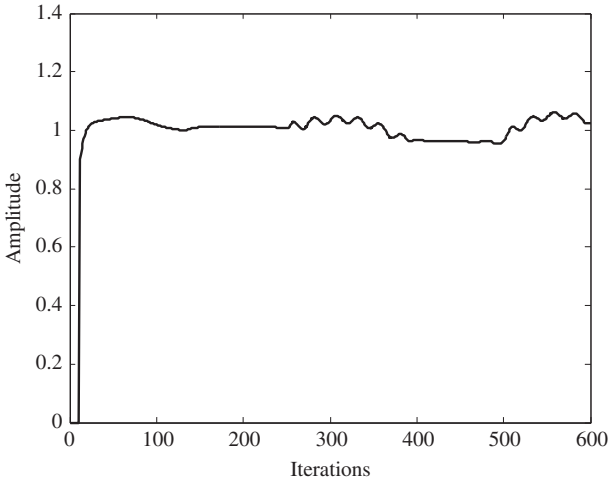


Fig. 8. Adaptive amplitudes for the PBLRNN filter.

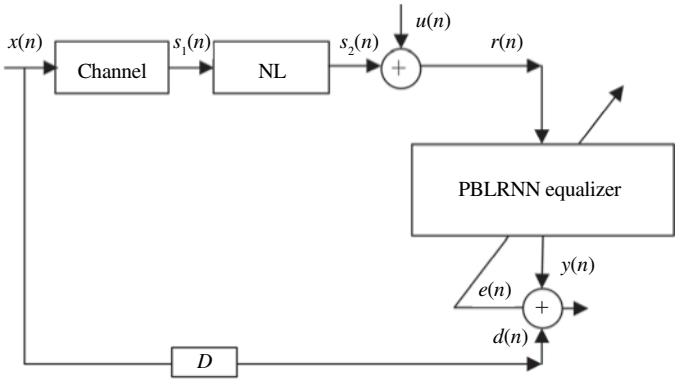


Fig. 9. Digital transmission system with the PBLRNN equalizer

TABLE III
PARAMETERS OF ALL THE NONLINEAR EQUALIZERS

| Parameters | PBLRNN | BLRNN | RNN |
|---|---|---|---|
| Learning rate | $\eta_1 = 0.1$ | $\eta_1 = 0.04$ | $\eta = 0.02$ |
| | $\eta_2 = 0.08$ | $\eta_2 = 0.03$ | |
| | $\eta_3 = 0.08$ | $\eta_3 = 0.03$ | |
| | $\eta_4 = 0.1$ | | |
| Number of external inputs | 5 | 5 | 5 |
| Number of output layer neurons | 2 | 2 | 2 |
| Number of modules | 4 | – | – |

To further evaluate the performance of the proposed filter, the corresponding NMSE values of all the examples are summarized in Table II. It is shown that the NMSE of the PBLRNN is lower than that of the PRNN, while PRNN over BLRNN and RNN.

### B. Nonlinear Channel Equalization

The performance of the PBLRNN as an equalizer is evaluated for nonlinear channel in wireless communication systems. The block diagram with the PBLRNN equalizer in a communication system is depicted in Fig. 9. The combined effect of the transmitter filter, the transmission medium, and other components are included in the "*Channel*," which can

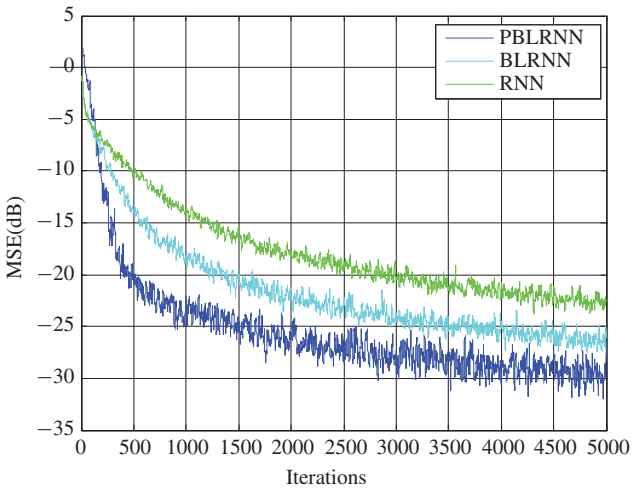


Fig. 10. Convergence properties of the equalizers under SNR = 16 dB for the nonlinear channel model.

TABLE IV
PARAMETERS OF ALL THE NONLINEAR PREDICTORS

| Parameters | PBLRNN | BLRNN | RNN |
|---|---|---|---|
| Learning rate | $\eta_1 = 0.8$ | $\eta_1 = 0.5$ | $\eta = 0.2$ |
| | $\eta_2 = 0.2$ | $\eta_2 = 0.1$ | |
| | $\eta_3 = 0.1$ | $\eta_3 = 0.1$ | |
| | $\eta_4 = 0.1$ | | |
| Number of external inputs | 5 | 5 | 5 |
| Number of output layer neurons | 2 | 2 | 2 |
| Number of modules | 4 | – | – |

TABLE V
COMPARISON OF THE RMSE FOR THREE NONLINEAR FILTERS

| Prediction model | RMSE |
|---|---|
| **PBLRNN** | 0.0357 |
| **BLRNN** | 0.0557 |
| **RNN** | 0.0752 |

be modeled as follows [10], [20], [21]:

$$s_1(n) = 0.3482x(n) + 0.8704x(n-1) + 0.3482x(n-2) \quad (38)$$

where the transmitted sequence $x(n)$ is with a 2-pulse amplitude modulation signal and in the form of $\{+1, -1\}$ in which each symbol is obtained from a uniform distribution. The "*NL*" block represents the nonlinear distortions of the symbols in the channel, its output may be expressed as

$$s_2(n) = s_1(n) + 0.2s_1^2(n) + 0.5s_1^3(n). \quad (39)$$

But, the output of the channel $s_2(n)$ is corrupted by noise $u(n)$, which is usually modeled as an additive white Gaussian noise process with a zero mean and variance $\sigma^2$. Then this corrupted signal is received at the receiver end and is given by $r(n) = s_2(n) + u(n)$. And where "$D$" is the transmission delay associated with the physical channel. $d(n)$ denotes the desired signal and is defined by $d(n) = x(n - D)$.

Convergence performance including the convergence speed and the steady-state error of the three equalizers averaged

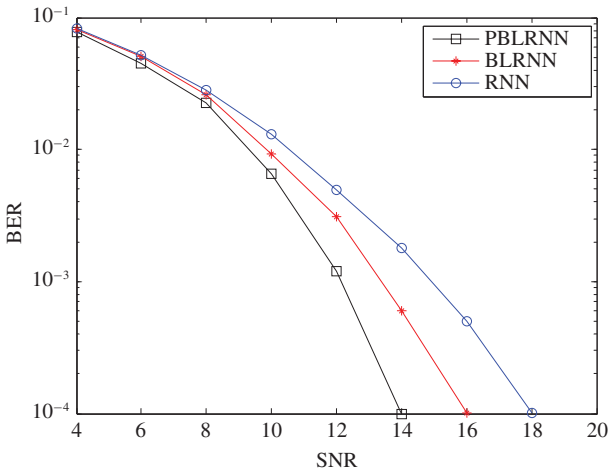


Fig. 11. BER performance of the equalizers for the nonlinear channel model.

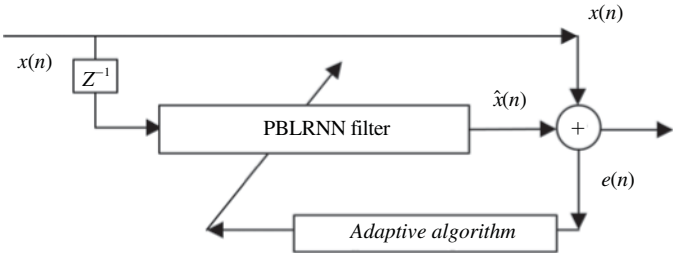


Fig. 12. Nonlinear predicting system with a PBLRNN filter.

over 100 independent experiments for the nonlinear channel is shown in Fig. 10. Each run has a different BPSK random sequence and random initialized coefficients of the equalizers, SNR of 16 dB is applied, and the learning rate parameters are set in Table III. We can clearly observe that the PBLRNN equalizer converges faster than BLRNN and RNN equalizers. Moreover, the MSE performance of the PBLRNN equalizer outperforms BLRNN and RNN equalizers. For instance, in Fig. 10, MSE value of the PBLRNN reaches $-28.5$ dB after 3000 training symbols, while MSE values of the BLRNN and RNN reach $-24$ and $-20$ dB, respectively.

Furthermore, we can run simulations for different SNR ranging from SNR $=$ 4 dB to SNR $=$ 20 dB at 2 dB intervals (4:2:20). BER performance comparisons are presented in Fig. 11. In each trial, the 5000 BPSK signals are used for training and next 10 000 signals are used for testing. The coefficient vectors of the equalizers are frozen after the training stage, and then the test is continued. It is seen that the PBLRNN equalizer shows better performance than the others for nonlinear channel model in wireless communication systems. In that respect, the pipelined architecture affords an efficient framework for improving further the nonlinear processing capabilities of the single BLRNN models.

### C. Prediction of Chaotic Time Series

To illustrate the effectiveness of the proposed filter for making nonlinear signal predictions, a nonlinear signal predicting

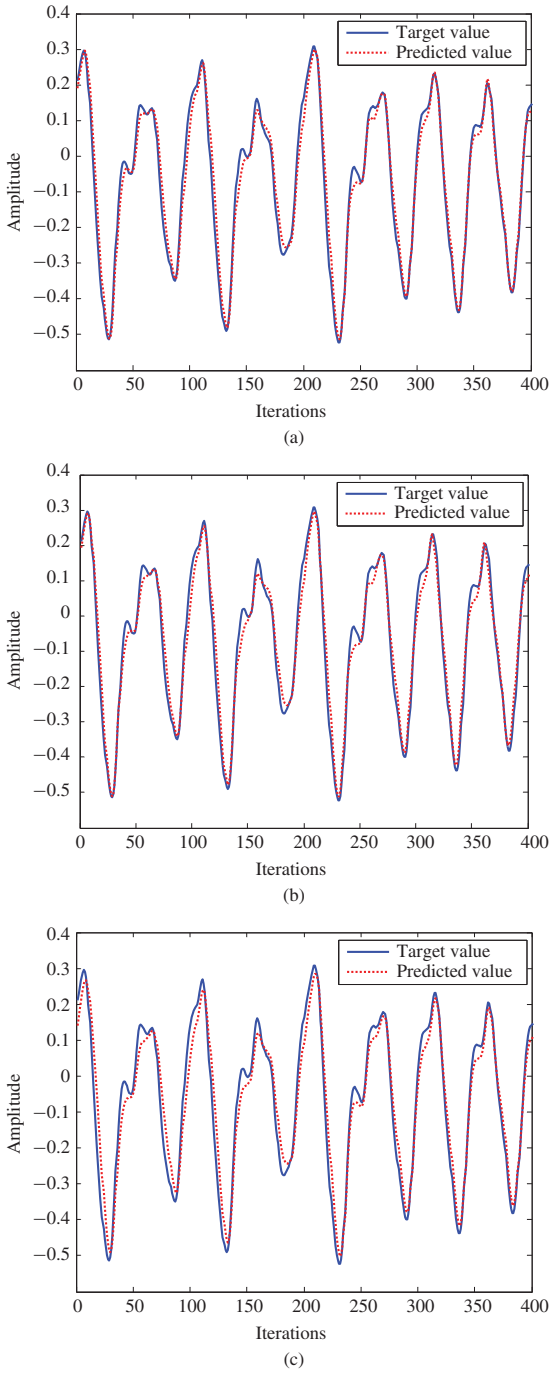


Fig. 13. Comparison of predicted values. (a) PBLRNN. (b) BLRNN. (c) RNN.

system with a PBLRNN filter is implemented in Fig. 12. And the Mackey-Glass chaotic time series often used as a benchmark for nonlinear predictors are chosen as input signals $x(n)$ of systems, which are generated by a delay differential equation [8], [22], [24], [25], and [46]

$$\frac{dy(t)}{dt} = \frac{\alpha y(t-\tau)}{1+y(t-\tau)^{10}} - by(t) \tag{40}$$

with values $\alpha = 0.2$, $b = 0.1$, and $\tau = 17$. The squared root of the mean square error (RMSE) is used to evaluate the

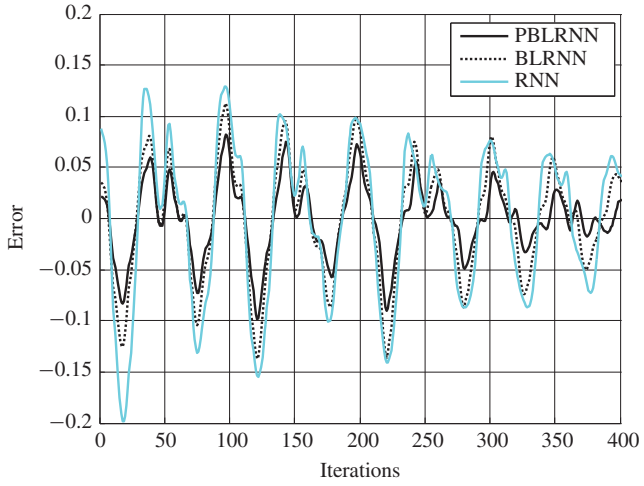


Fig. 14. Comparison of predicted errors.

prediction accuracy, and is defined by

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{t=1}^{n}\left(y(t)-\hat{y}(t)\right)^2} \tag{41}$$

where $\hat{y}(t)$ is the estimated value of $y(t)$ at time $t$.

In this experiment, the parameters of the PBLRNN, BLRNN, and RNN filters are set as shown in Table IV. During the training, the networks were tested on 400 samples of chaotic time series.

Table V depicts the comparative results of RMSE for the PBLRNN, BLRNN, and RNN filters. The RMSEs presented in Table V illustrate that the prediction accuracy of the PBLRNN is obviously superior over those of the BLRNN and RNN.

Figs. 13 and 14 show the predicted values and predicted errors, respectively, of the PBLRNN, BLRNN, and RNN filters. From these figures, it can be observed clearly that the predicted values of the PBLRNN agree better with the target value than those of the BLRNN and RNN filters, the predicted errors are found to be lower than those of BLRNN and RNN filters. Moreover, throughout the experiments, the results clearly reveal that the PBLRNN filter is capable of capturing the underlying dynamics from chaotic time series more efficiently than the BLRNN and RNN filters for this case.

As a consequence, the performance of the PBLRNN is significantly superior to that of the BLRNN and RNN filters in all cases. The improved performance of the proposed nonlinear filter is attributed to the following reasons: 1) each module of the PBLRNN filter utilizes the high-order terms of inputs; 2) the enhanced nesting module architecture; 3) the modified adaptive amplitude RTRL algorithm circumvents the nesting effect along the modules; and 4) the momentums speed up the convergence of the algorithm.

## VI. Conclusion

In this paper, we have proposed a nonlinear adaptive filter with a PBLRNN to reduce the computational burden of the BLRNN. The novel nonlinear filter consists of a number of modules-based BLRNNs that are interconnected in a chained form, and inherits the major characteristics (low computational complexity) of the pipelined architecture. The parameters that update the rules of the nonlinear adaptive filter are derived according to the adaptive amplitude RTRL algorithm. The performance of the filter presented in this paper has been assessed for nonlinear system identification, nonlinear channel equalization, nonlinear and nonstationary chaotic signal prediction, and compared with that of the BLRNN and RNN filters. Simulation results show that the proposed filter with lower computational complexity can converge faster, and outperform the single BLRNN model. It is also demonstrated that the PBLRNN filter is superior to the BLRNN- and RNN-based approaches. As suggested by the anonymous reviewers, further investigation on how to improve the convergence speed of the PBLRNN filter using the RTRL algorithm might be needed in the future work, while retaining high performance and low computational complexity in accordance with the methods in [17]–[19]. In addition, an alternative algorithm for RNNs in real-time implementation is a truncated BPTT($h$) in case of online update [47]. Researches show that the BPTT($h$) or any modified BPTT($h$) algorithms are much better than the RTRL algorithm in terms of convergence speed and computational complexity [11], [47]–[49]. Therefore, to further reduce computational burden and to improve the convergence performance of the PBLRNN, some novel adaptive algorithms-based BPTT($h$) rule of the proposed PBLRNN will be also discussed in the future work.

## Appendix

Substituting (15) and (18) into (29a–c), we have

$$\begin{aligned} Ph^{i,k}_{l,1,j}(n) = \lambda(n)\phi'(U_{i,k}(n))\Bigg\{&\delta_{l,k}v_{k,1,j}(n) \\ &+\frac{\partial y_{i+1,1}(n)}{\partial h_{l,1,j}(n)}\sum_{m=1}^{p}h_{l,2,m}(n)v_{k,1,m}(n) \\ &+\sum_{K=1}^{p}\sum_{m=2}^{q}h_{l,2,Kp+m}(n)v_{k,1,K}(n)\frac{\partial y_{i,m}(n-1)}{\partial h_{l,1,j}(n)} \\ &+\sum_{m=2}^{L_3}h_{l,3,m}(n)\frac{\partial y_{i,m}(n-1)}{\partial h_{l,1,j}(n)}\Bigg\} \end{aligned} \tag{A1a}$$

$$\begin{aligned} Ph^{i,k}_{l,2,j}(n) = \lambda(n)\phi'(U_{i,k}(n))\Bigg\{&\delta_{l,k}v_{k,2,j}(n) \\ &+\frac{\partial y_{i+1,1}(n)}{\partial h_{l,2,j}(n)}\sum_{m=1}^{p}h_{l,2,m}(n)v_{k,1,m}(n) \\ &+\sum_{K=1}^{p}\sum_{m=2}^{q}h_{l,2,Kp+m}(n)v_{k,1,K}(n)\frac{\partial y_{i,m}(n-1)}{\partial h_{l,2,j}(n)} \\ &+\sum_{m=2}^{L_3}h_{l,3,m}(n)\frac{\partial y_{i,m}(n-1)}{\partial h_{l,2,j}(n)}\Bigg\} \end{aligned} \tag{A1b}$$

$$Ph^{i,k}_{l,3,j}(n) = \lambda(n)\phi'(U_{i,k}(n))\Bigg\{\delta_{l,k}v_{k,3,j}(n) + \frac{\partial y_{i+1,1}(n)}{\partial h_{l,3,j}(n)}\sum_{m=1}^{p} h_{l,2,m}(n)v_{k,1,m}(n) + \sum_{K=1}^{p}\sum_{m=2}^{q} h_{l,2,Kp+m}(n)v_{k,1,K}(n)\frac{\partial y_{i,m}(n-1)}{\partial h_{l,3,j}(n)} + \sum_{m=2}^{L_3} h_{l,3,m}(n)\frac{\partial y_{i,m}(n-1)}{\partial h_{l,3,j}(n)}\Bigg\} \quad \text{(A1c)}$$

and $1 \le i < M$, where the notation $\delta_{l,k}$ denotes the Kronecker delta

$$\delta_{l,k} = \begin{cases} 1 & l = k \\ 0 & l \ne k \end{cases} \quad \text{(A2)}$$

and

$$\phi'(U_{i,k}(n)) = \phi(U_{i,k}(n))[1-\phi(U_{i,k}(n))],\ (1 \le i \le M,\ 1 \le k \le q) \quad \text{(A3)}$$

when $i = M$, we get

$$Ph^{M,k}_{l,1,j}(n) = \lambda(n)\phi'(U_{M,k}(n))\Bigg\{\delta_{l,k}v_{k,1,j}(n) + \sum_{K=1}^{p}\sum_{m=2}^{q} h_{l,2,Kp+m}(n)v_{k,1,K}(n)\frac{\partial y_{M,m}(n-1)}{\partial h_{l,1,j}(n)} + \sum_{m=2}^{L_3} h_{l,3,m}(n)\frac{\partial y_{M,m}(n-1)}{\partial h_{l,1,j}(n)}\Bigg\} \quad \text{(A4a)}$$

$$Ph^{M,k}_{l,2,j}(n) = \lambda(n)\phi'(U_{M,k}(n))\Bigg\{\delta_{l,k}v_{k,2,j}(n) + \sum_{K=1}^{p}\sum_{m=1}^{q} h_{l,2,Kp+m}(n)v_{k,1,K}(n)\frac{\partial y_{M,m}(n-1)}{\partial h_{l,2,j}(n)} + \sum_{m=1}^{L_3} h_{l,3,m}(n)\frac{\partial y_{M,m}(n-1)}{\partial h_{l,2,j}(n)}\Bigg\} \quad \text{(A4b)}$$

$$Ph^{M,k}_{l,3,j}(n) = \lambda(n)\phi'(U_{M,k}(n))\Bigg\{\delta_{l,k}v_{k,3,j}(n) + \sum_{K=1}^{p}\sum_{m=1}^{q} h_{l,2,Kp+m}(n)v_{k,1,K}(n)\frac{\partial y_{M,m}(n-1)}{\partial h_{l,3,j}(n)} + \sum_{m=1}^{L_3} h_{l,3,m}(n)\frac{\partial y_{M,m}(n-1)}{\partial h_{l,3,j}(n)}\Bigg\}. \quad \text{(A4c)}$$

Under the approximations that the learning rates $\eta_1, \eta_2$, and $\eta_3$ for the PBLRNN are chosen sufficiently small so that the weights are adapted slowly, then, the following approximations are computed by:

$$\frac{\partial y_{i,m}(n-1)}{\partial h_{l,1,j}(n)} \approx \frac{\partial y_{i,m}(n-1)}{\partial h_{l,1,j}(n-1)} = Ph^{i,m}_{l,1,j}(n-1) \quad \text{(A5a)}$$

$$\frac{\partial y_{i,m}(n-1)}{\partial h_{l,2,j}(n)} \approx \frac{\partial y_{i,m}(n-1)}{\partial h_{l,2,j}(n-1)} = Ph^{i,m}_{l,2,j}(n-1) \quad \text{(A5b)}$$

$$\frac{\partial y_{i,m}(n-1)}{\partial h_{l,3,j}(n)} \approx \frac{\partial y_{i,m}(n-1)}{\partial h_{l,3,j}(n-1)} = Ph^{i,m}_{l,3,j}(n-1). \quad \text{(A5c)}$$

Then (A4a), (A4b), and (A4c) can be respectively approximated by

$$Ph^{i,k}_{l,1,j}(n) \approx \lambda(n)\phi'(U_{i,k}(n))\Bigg\{\delta_{l,k}v_{k,1,j}(n) + Ph^{i+1,1}_{l,1,j}(n)\sum_{m=1}^{p} h_{l,2,m}(n)v_{k,1,m}(n) + \sum_{K=1}^{p}\sum_{m=2}^{q} h_{l,2,Kp+m}(n)v_{k,1,K}(n)Ph^{i,m}_{l,1,j}(n-1) + \sum_{m=2}^{L_3} h_{l,3,m}(n)Ph^{i,m}_{l,1,j}(n-1)\Bigg\} \quad \text{(A6a)}$$

$$Ph^{i,k}_{l,2,j}(n) \approx \lambda(n)\phi'(U_{i,k}(n))\Bigg\{\delta_{l,k}v_{k,2,j}(n) + Ph^{i+1,1}_{l,2,j}(n)\sum_{m=1}^{p} h_{l,2,m}(n)v_{k,1,m}(n) + \sum_{K=1}^{p}\sum_{m=2}^{q} h_{l,2,Kp+m}(n)v_{k,1,K}(n)Ph^{i,m}_{l,2,j}(n-1) + \sum_{m=1}^{L_3} h_{l,3,m}(n)Ph^{i,m}_{l,2,j}(n-1)\Bigg\} \quad \text{(A6b)}$$

$$Ph^{i,k}_{l,3,j}(n) \approx \lambda(n)\phi'(U_{i,k}(n))\Bigg\{\delta_{l,k}v_{k,3,j}(n) + Ph^{i+1,1}_{l,3,j}(n)\sum_{m=1}^{p} h_{l,2,m}(n)v_{k,1,m}(n) + \sum_{K=1}^{p}\sum_{m=2}^{q} h_{l,2,Kp+m}(n)v_{k,1,K}(n)Ph^{i,m}_{l,3,j}(n-1) + \sum_{m=1}^{L_3} h_{l,3,m}(n)Ph^{i,m}_{l,3,j}(n-1)\Bigg\}. \quad \text{(A6c)}$$

In a similar way, for the case when $i = M$, we have

$$Ph^{M,k}_{l,1,j}(n) \approx \lambda(n)\phi'(U_{M,k}(n))\Bigg\{\delta_{l,k}v_{k,1,j}(n) + \sum_{K=1}^{p}\sum_{m=2}^{q} h_{l,2,Kp+m}(n)v_{K,1,k}(n)Ph^{M,m}_{l,1,j}(n-1) + \sum_{m=2}^{L_3} h_{l,3,m}(n)Ph^{M,m}_{l,1,j}(n-1)\Bigg\} \quad \text{(A7a)}$$

$$Ph_{l,2,j}^{M,k}(n) \approx \lambda(n)\phi'(U_{M,k}(n))\Bigg\{\delta_{l,k}v_{k,2,j}(n) + \sum_{K=1}^{p}\sum_{m=1}^{q} h_{l,2,Kp+m}(n)v_{K,1,k}(n)Ph_{l,2,j}^{M,m}(n-1) + \sum_{m=1}^{L_3} h_{l,3,m}(n)Ph_{l,2,j}^{M,m}(n-1)\Bigg\} \quad \text{(A7b)}$$

$$Ph_{l,3,j}^{M,k}(n) \approx \lambda(n)\phi'(U_{M,k}(n))\Bigg\{\delta_{l,k}v_{k,3,j}(n) + \sum_{K=1}^{p}\sum_{m=1}^{q} h_{l,2,kp+m}(n)v_{k,1,K}(n)Ph_{l,3,j}^{M,m}(n-1) + \sum_{m=1}^{L_3} h_{l,3,m}(n)Ph_{l,3,j}^{M,m}(n-1)\Bigg\} \quad \text{(A7c)}$$

with initial conditions $Ph_{l,1,j}^{i,k}(n) = Ph_{l,2,j}^{i,k}(n) = Ph_{l,3,j}^{i,k}(n) = 0$.

But, the above-mentioned recursive equations are too complicated to implement. According to the method in [41], let us assume that the recursion is negligible

$$Ph_{l,1,j}^{i,k}(n-1) = Ph_{l,2,j}^{i,k}(n-1) = Ph_{l,3,j}^{i,k}(n-1) = 0 \quad \text{(A8)}$$

for all $i$, $k$, $l$, and $j$. Under this assumption, the coefficient update equations can be simplified to (30) and (31).

## ACKNOWLEDGMENT

The authors would like to thank the associate editor and the anonymous reviewers for their valuable comments and suggestions for improving the quality of this paper.

## REFERENCES

[1] V. J. Mathews and G. L. Sicuranza, *Polynomial Signal Processing*. New York: Wiley, 2001.

[2] S. Haykin, *Neural Networks: A Comprehensive Foundation*. Englewood Cliffs, NJ: Prentice-Hall, 1994.

[3] K. S. Narendra and K. Parthasarathy, "Identification and control of dynamical systems using neural networks," *IEEE Trans. Neural Netw.*, vol. 1, no. 1, pp. 4–27, Mar. 1990.

[4] J. Park and I. W. Sandberg, "Universal approximation using radial-basis-function networks," *J. Neural Comput.*, vol. 3, no. 2, pp. 246–257, Jun. 1991.

[5] J. C. Patra, R. N. Pal, B. N. Chatterji, and G. Panda, "Identification of nonlinear dynamic systems using functional link artificial neural networks," *IEEE Trans. Syst., Man, Cybern. B*, vol. 29, no. 2, pp. 254–262, Apr. 1999.

[6] Q. Zhang and A. Benviste, "Wavelet networks," *IEEE Trans. Neural Netw.*, vol. 3, no. 6, pp. 889–898, Nov. 1992.

[7] R. H. Abiyev and O. Kaynak, "Fuzzy wavelet neural networks for identification and control of dynamic plants—A novel structure and a comparative study," *IEEE Trans. Ind. Electron.*, vol. 55, no. 8, pp. 3133–3140, Aug. 2008.

[8] J. T. Connor, R. D. Martin, and L. E. Atlas, "Recurrent neural networks and robust time series prediction," *IEEE Trans. Neural Netw.*, vol. 5, no. 2, pp. 240–254, Mar. 1994.

[9] D. P. Mandic and J. A. Chambers, *Recurrent Neural Networks for Prediction*. Chichester, U.K.: Wiley, 2001.

[10] G. Kechriotis, E. Zervas, and E. S. Manolakos, "Using recurrent neural networks for adaptive communication channel equalizations," *IEEE Trans. Neural Netw.*, vol. 5, no. 2, pp. 267–278, Mar. 1994.

[11] F. J. Pineda, "Generalization of back-propagation to recurrent neural networks," *Phys. Rev. Lett.*, vol. 59, no. 19, pp. 2229–2232, 1987.

[12] P. J. Werbos, "Backpropagation through time: What it does and how to do it," *Proc. IEEE*, vol. 78, no. 10, pp. 1550–1560, Oct. 1990.

[13] R. J. Williams and D. Zipser, "A learning algorithm for continually running fully recurrent neural networks," *Neural Comput.*, vol. 1, no. 2, pp. 270–280, 1989.

[14] M. W. Mak, K.-W. Ku, and Y.-L. Lu, "On the improvement of the real time recurrent learning algorithm for recurrent neural networks," *Neurocomput.-IJON*, vol. 24, nos. 1–3, pp. 13–36, 1999.

[15] D. P. Mandic and J. A. Chambers, "A normalised real time recurrent learning algorithm," *Signal Process.*, vol. 80, no. 9, pp. 1909–1916, Sep. 2000.

[16] Q. Song, Y. Wu, and Y. C. Soh, "Robust adaptive gradient-descent training algorithm for recurrent neural networks in discrete time domain," *IEEE Trans. Neural Netw.*, vol. 19, no. 11, pp. 1841–1853, Nov. 2008.

[17] Z. Xu, Q. Song, and D. Wang, "Recurrent neural tracking control based on multivariable robust adaptive gradient-descent training algorithm," *Neural Comput. Appl.*, DOI 10.1007/s00521-011-0618-2, Jun. 2011.

[18] Y. Wu, Q. Song, and S. Liu, "A normalized adaptive training of recurrent neural networks with augmented error gradient," *IEEE Trans. Neural Netw.*, vol. 19, no. 2, pp. 351–356, Feb. 2008.

[19] Z. Liu and I. Elhanany, "A fast and scalable recurrent neural network based on stochastic meta descent," *IEEE Trans. Neural Netw.*, vol. 19, no. 9, pp. 1652–1658, Sep. 2008.

[20] J. Choi, A. C. C. Lima, and S. Haykin, "Kalman filter-trained recurrent neural equalizers for time-varying channels," *IEEE Trans. Commun.*, vol. 53, no. 3, pp. 472–480, Mar. 2005.

[21] J. Choi, M. Bouchard, and T. H. Yeap, "Decision feedback recurrent neural equalization with fast convergence rate," *IEEE Trans. Neural Netw.*, vol. 16, no. 3, pp. 699–708, May 2005.

[22] S. J. Yoo, J. B. Park, and Y. H. Choi, "Stable predictive control of chaotic systems using self-recurrent wavelet neural network," *Int. J. Control Autom. Syst.*, vol. 3, no. 1, pp. 43–55, Mar. 2005.

[23] C.-H. Lu, "Design and application of stable predictive controller using recurrent wavelet neural networks," *IEEE Trans. Ind. Electron.*, vol. 56, no. 9, pp. 3733–3742, Sep. 2009.

[24] D. C. Park and Y. Zhu, "Bilinear recurrent neural network," in *Proc. IEEE Int. Conf. Neural Netw.*, vol. 3. Orlando, FL, Jul. 1994, pp. 1459–1464.

[25] E. Kosmatopoulos and M. Polycarpou, "High-order neural network structures for identification of dynamical systems," *IEEE Trans. Neural Netw.*, vol. 6, no. 2, pp. 422–431, Mar. 1995.

[26] D.-C. Park, "Prediction of MPEG video traffic over ATM networks using dynamic bilinear recurrent neural network," *Appl. Math. Comput.*, vol. 205, no. 2, pp. 648–657, Nov. 2008.

[27] S. H. Shin and D. C. Park, "Short-term load forecasting using bilinear recurrent neural network," in *Proc. 4th Int. Symp. Neural Netw.*, vol. 3. 2007, pp. 111–116.

[28] D.-C. Park and T. K. Jeong, "Complex bilinear recurrent neural network for equalization of a satellite channel," *IEEE Trans. Neural Netw.*, vol. 13, no. 3, pp. 711–725, May 2002.

[29] D. C. Park, "Structure optimization of bilinear recurrent neural networks and its application to ethernet network traffic prediction," *Inf. Sci.*, 2009, to be published.

[30] D. C. Park, C. N. Tran, and Y. Lee, "Short-term load forecasting using multiscale bilinear recurrent neural network," in *Proc. 9th Pacific RIM Int. Conf. Artif. Intell.*, 2006, pp. 329–338.

[31] S. Haykin and L. Li, "Nonlinear adaptive prediction of nonstationary signals," *IEEE Trans. Signal Process.*, vol. 43, no. 2, pp. 526–535, Feb. 1995.

[32] J. Baltersee and J. Chambers, "Nonlinear adaptive prediction of speech with a pipelined recurrent neural network," *IEEE Trans. Signal Process.*, vol. 46, no. 8, pp. 2207–2216, Aug. 1998.

[33] D. G. Stavrakoudis and J. B. Theocharis, "Pipelined recurrent fuzzy neural networks for nonlinear adaptive speech prediction," *IEEE Trans. Syst., Man, Cybern., Part B: Cybern.*, vol. 37, no. 5, pp. 1305–1320, Oct. 2007.

[34] H. Zhao and J. Zhang, "A novel adaptive nonlinear filter-based pipelined feedforward second-order volterra architecture," *IEEE Trans. Signal Process.*, vol. 57, no. 1, pp. 237–246, Jan. 2009.

[35] H. Zhao and J. Zhang, "Pipelined Chebyshev functional link artificial recurrent neural network for nonlinear adaptive filter," *IEEE Trans. Syst., Man, Cybern., Part B: Cybern.*, vol. 40, no. 1, pp. 162–172, Feb. 2010.

[36] P. R. Chang and J. T. Hu, "Optimal nonlinear adaptive prediction and modeling of MPEG video in ATM networks using pipelined recurrent neural networks," *IEEE J. Sel. Areas Commun.*, vol. 15, no. 6, pp. 1087–1100, Aug. 1997.

[37] H. Q. Zhao and J. S. Zhang, "Nonlinear dynamic system identification using pipelined functional link artificial recurrent neural network," *Neurcomputing*, vol. 72, nos. 13–15, pp. 3046–3054, Aug. 2009.

[38] Y. S. Chen, C. J. Chang, and Y. L. Hsieh, "A channel effect prediction-based power control scheme using PRNN/ERLS for uplinks in DS-CDMA cellular mobile systems," *IEEE Trans. Wireless Commun.*, vol. 5, no. 1, pp. 23–27, Jan. 2006.

[39] D. P. Mandic and J. A. Chambers, "Toward an optimal PRNN-based nonlinear predictor," *IEEE Trans. Neural Netw.*, vol. 10, no. 6, pp. 1435–1442, Nov. 1999.

[40] D. P. Mandic and J. A. Chambers, "On the choice of parameters of the cost function in nested modular RNNs," *IEEE Trans. Neural Netw.*, vol. 11, no. 2, pp. 315–322, Mar. 2000.

[41] E. Trentin, "Networks with trainable amplitude of activation functions," *Neural Netw.*, vol. 14, nos. 4–5, pp. 471–493, May 2001.

[42] S. L. Goh and D. P. Mandic, "Recurrent neural networks with trainable amplitude of activation functions," *Neural Netw.*, vol. 16, no. 8, pp. 1095–1100, Oct. 2003.

[43] S. L. Goh and D. P. Mandic, "Nonlinear adaptive prediction of complex-valued signals by complex-valued PRNN," *IEEE Trans. Signal Process.*, vol. 53, no. 5, pp. 1827–1836, May 2005.

[44] P. L. Feintuch, "An adaptive recursive LMS filter," *Proc. IEEE*, vol. 64, no. 11, pp. 1622–1624, Nov. 1976.

[45] S. Haykin, *Adaptive Filter Theory*, 4th ed. Englewood Cliffs, NJ: Prentice-Hall, 2002.

[46] M. Z. Shi and M. Han, "Support vector echo-state machine for chaotic time-series prediction," *IEEE Trans. Neural Netw.*, vol. 18, no. 2, pp. 359–372, Mar. 2007.

[47] A. F. Atiya and A. G. Parlos, "New results on recurrent network training: Unifying the algorithms and accelerating convergence," *IEEE Trans. Neural Netw.*, vol. 11, no. 3, pp. 697–709, May 2000.

[48] O. D. Jesus and M. T. Hagan, "Backpropagation algorithms for a broad class of dynamic networks," *IEEE Trans. Neural Netw.*, vol. 18, no. 1, pp. 14–27, Jan. 2007.

[49] J. Mazumdar and R. G. Harley, "Recurrent neural networks trained with backpropagation through time algorithm to estimate nonlinear load harmonic currents," *IEEE Trans. Ind. Electron.*, vol. 55, no. 9, pp. 3484–3491, Sep. 2008.

**Haiquan Zhao** (M'10) received the B.S., M.S., and Ph.D. degrees in applied mathematics from the School of Information Science and Technology, Southwest Jiaotong University, Chengdu, China, in 1998, 2005, and 2011, respectively.

He is currently an Associate Professor with the School of Electrical Engineering, Southwest Jiaotong University. His current research interests include nonlinear signal processing for communication, nonlinear active noise control, nonlinear systems, and chaos.

Dr. Zhao has served as an active reviewer for several IEEE and other international journals.

**Xiangping Zeng** received the B.E. degree from the School of Electrical Engineering, Southwest Jiaotong University, Chengdu, China, in 1998, and the M.S. degree from the College of Information Engineering, Graduate School of the Chinese Academy of Sciences, Beijing, China, in 2006. She is currently pursuing the Doctorate degree in the field of signal and information processing with the School of Information Science and Technology, Southwest Jiaotong University.

Her current research interests include video image processing.

**Zhengyou He** (M'10) was born in Sichuan, China, in 1970. He received the B.Sc. and M.Sc. degrees from Chongqing University, Chongqing, China, in 1992 and 1995, respectively, and the Ph.D. degree from Southwest Jiaotong University, Chengdu, China, in 2001.

He has been a Professor with the Department of Electrical Engineering, Southwest Jiaotong University. He is the author or co-author of more than 100 journal papers and the owner of two invention patents. His current research interests include the area of signal processing and information theory and their applications in electrical power systems, and application of wavelet transforms in power systems.