Nonlinear Adaptive Prediction of Complex-Valued Signals by Complex-Valued PRNN

Su Lee Goh, *Student Member, IEEE,* and Danilo P. Mandic, *Senior Member, IEEE*

***Abstract*—A complex-valued pipelined recurrent neural network (CPRNN) for nonlinear adaptive prediction of complex nonlinear and nonstationary signals is introduced. This architecture represents an extension of the recently proposed real-valued PRNN of Haykin and Li in 1995. To train the CPRNN, a complex-valued real time recurrent learning (CRTRL) algorithm is first derived for a single recurrent neural network (RNN). This algorithm is shown to be generic and applicable to general signals that have complex domain representations. The CRTRL is then extended to suit the modularity of the CPRNN architecture. Further, to cater to the possibly large dynamics of the input signals, a gradient adaptive amplitude of the nonlinearity within the neurons is introduced to give the adaptive amplitude CRTRL (AACRTRL). A comprehensive analysis of the architecture and associated learning algorithms is undertaken, including the role of the number of nested modules, number of neurons within the modules, and input memory of the CPRNN. Simulations on real-world and synthetic complex data support the proposed architecture and algorithms.**

***Index Terms*—Complex-valued analysis, multidimension forecasting, nonlinear adaptive prediction, RNNs.**

I. Introduction

REAL-world data are often subject to environmental noise and acquisition errors, which makes the application of standard linear modeling and adaptive filtering techniques difficult or inadequate. In addition, novel signal processing disciplines focus on classes of signals where nonlinearity and multimodality play a major role. Therefore, there is a need for advanced adaptive signal processing algorithms for all aspects of adaptive filtering applications. In this context, applications of adaptive prediction are manifold and include not only the common applications in signal processing but applications in the areas of biological and medical engineering, physics, and earth sciences as well [13], [28].

The theory of linear adaptive filters is already well established [2], [23], whereas architectures and algorithms for nonlinear adaptive filtering are still emerging [28]. Some recent results have shown that neural networks (NNs) are powerful tools for nonlinear adaptive filtering of real-world data [4], [6], [28], which is mainly due to their ability to uniformly approximate any continuous function on a compact domain [5], [7], [15], [20]. Given their ability to learn from examples, the application of neural networks in the area of nonlinear adaptive prediction and modeling offers potentially better performance compared with standard statistical and linear filtering methods [13], [28]. In this context, the so-called time delay neural networks (TDNNs) have been employed traditionally as nonlinear adaptive filters; however, due to the fact that they carry no memory, the memory of such a system is governed by the size of the time delay input line [4], [38]. This, in turn, restricts their practical applicability since, for quality performance, the length of the tap input line required increases with the complexity of a signal in hand. This represents a major obstacle for applications in real-time signal processing, since, for instance, the backpropagation algorithm requires a large number of training data and a great deal of training to converge [13], [33], [38].

Unlike "static" feedforward networks, recurrent neural networks (RNNs) possess rich internal nonlinear dynamics, which makes them capable of modeling more complex processes than their feedforward counterparts[1] [4], [29], [30]. Fully connected recurrent neural networks (FCRNNs) with internal feedback (memory) allow for modeling of complex dynamics [29] and, hence, have been recently considered as flexible tools for nonlinear adaptive filtering [28]. For real-time applications, the Real Time Recurrent Learning (RTRL) algorithm (see Williams and Zipser [40]) has been widely used to train FCRNNs. To process highly nonlinear real-valued nonstationary signals, Haykin and Li introduced the Pipelined Recurrent Neural Network (PRNN) [14]: a computationally efficient modular nonlinear adaptive filter. Based on a concatenation of M modules, each consisting of FCRNNs with N neurons, the PRNN was proven to possess improved capability of tracking nonlinearity as compared to single RNNs while maintaining low computational complexity ($\mathcal{O}(MN^4)$ for the PRNN with M modules, as compared to $\mathcal{O}((MN)^4)$ for the FCRNN). The PRNN architecture also helps to circumvent the problem of vanishing gradient, due to its spatial representation of a temporal pattern and feedback connections within the architecture [27], [36]. This architecture has been successfully employed for a variety of applications where complexity and nonlinearity pose major problems, including those speech processing [24], ATM traffic modeling [3], and communications [16], [34]. More insight into the PRNN performance is provided in [14], [26], and [27].

In modern disciplines, efficient data models are often complex-valued (communications, biomedical, radar), and consequently, adaptive filtering algorithms have been extended to process signals in the complex domain $\mathbb{C}$. Notice that prop-

Manuscript received August 14, 2003; revised June 9, 2004. The work of D. Mandic was supported in part by the Royal Society under Grant G503/24543/SM. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Tulay Adali.

The authors are with the Department of Electrical and Electronic Engineering, Imperial College London, London SW7 2AZ, U.K. (e-mail: su.goh@imperial.ac.uk; d.mandic@imperial.ac.uk).

Digital Object Identifier 10.1109/TSP.2005.845462

[1]Nonlinear autoregressive (NAR) processes can be modeled using feedforward networks, whereas nonlinear autogressive moving average (NARMA) processes can be represented using RNNs.

erties of complex signals are not only varying in terms of their statistical nature but also in terms of their bivariate or complex nature [8]. To process such signals, in 1975, Widrow *et al.* introduced the complex least mean square (CLMS) algorithm [39]. Later in the 1990s, to cater for the possibly nonlinear nature of complex signals, extensions of real-valued algorithms to the complex domain include the class of complex backpropagation algorithms [9], [18], [22]. For the case of RNNs, a complex variant of the RTRL algorithm has been introduced in [17]. Notice that these extensions are nontrivial, especially in the case of neural nonlinear adaptive filters, where one of the major issues to be solved is that of a suitable complex nonlinear activation function (see Kim and Adali [20]), [37]. According to Liouville's theorem, the only bounded and analytic function in $\mathbb{C}$ is constant [31], and to that cause, meromorphic functions[2] have been employed as complex nonlinear activation functions (due to their property that they are analytic everywhere except for a discrete subset of $\mathbb{C}$). At their singular points, these functions tend to infinity, thus removing the possibility of encountering essential singularities[3] [19], [20]. Due to these problems, for convenience, previous studies have mostly focused on the so-called split-complex activation functions[4] (AF) [1], [20]. The split-complex approach has been shown to yield reasonable performance for some applications in channel equalization [1], [17], [22], as well as for applications where there is no strong coupling between the real and imaginary part within the complex signal. However, for the common case where the inphase (I) and quadrature (Q) components are strongly correlated, algorithms employing the split-complex activation function tend to yield poor performance [8]. Notice that split-complex algorithms cannot calculate the true gradient unless the real and imaginary weight updates are mutually independent. Therefore, the problems encountered with split-complex learning algorithms for nonlinear adaptive filtering include the following: i) The solutions are not general since split-complex AFs are not universal approximators [20]; ii) split-complex AFs are not analytic, and hence, the Cauchy–Riemann equations do not apply [19], [37]; iii) split-complex algorithms are strictly speaking not "fully" complex [20], and such algorithms underperform in applications where complex signals exhibit strong component correlations [28]; iv) these algorithms do not have a generic form of their real-valued counterparts, and hence, their signal flow-graphs are fundamentally different [32].

Although there have been attempts to devise fully complex algorithms for RNNs, a general fully complex CRTRL has been lacking to date. To this cause, we first derive a CRTRL for a single recurrent neural network with a general "fully" complex activation function. This makes complex RNNs suitable for adaptive filtering of general complex-valued nonlinear and nonstationary signals. In addition, the derived CRTRL algorithm is generic and represents a natural extension of the real-valued RTRL. Next, for real-time applications, to be able to cope with unknown and large dynamics of the inputs, we introduce an adaptive amplitude into the nonlinear activation function of a neuron within an RNN. This way, following the approach from [10], [35], the adaptive amplitude CRTRL (AACRTRL) algorithm is derived. The complex PRNN (CPRNN) is then introduced as an extension of the real PRNN [14], and both the CRTRL and AACRTRL algorithms are modified to suit the CPRNN architecture. The analysis is supported by examples on prediction for several fundamental classes of complex-valued signals, including complex nonlinear, complex colored, and real-life complex-valued nonstationary signals.

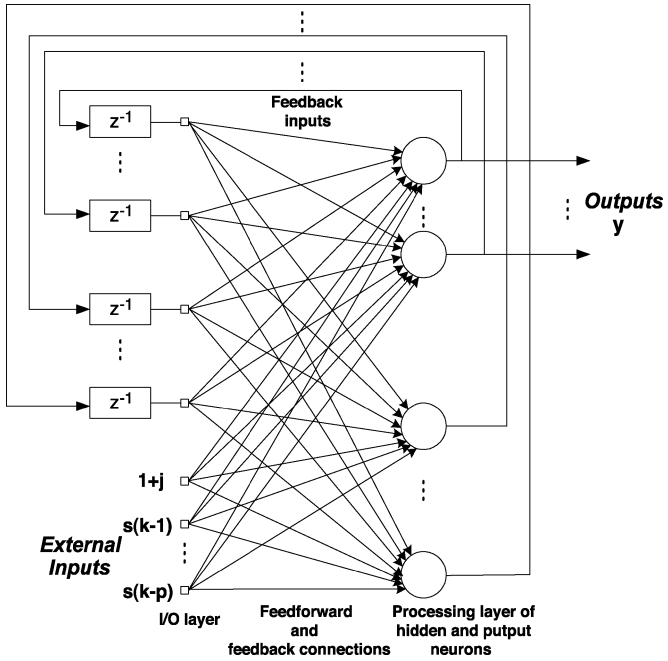


Fig. 1. Fully connected recurrent neural network for prediction.

The paper is organized in the following manner. In Section II, we present a general CRTRL algorithm for the FCRNN. In Section III, the CPRNN is introduced. In Section IV, an adaptive amplitude CRTRL algorithm for the CPRNN is derived. This followed by comprehensive simulations in Section V. Finally, the paper concludes in Section VI.

II. Complex-Valued Real-Time Recurrent Learning (CRTRL) Algorithm

A. Complex RNN

Fig. 1 shows an FCRNN, which consists of N neurons with p external inputs and N feedback connections. The network has two distinct layers, namely, the external input-feedback layer and a layer of processing elements. Let $y_l(k)$ denote the complex-valued output of a neuron $l = 1, \ldots, N$ at time index k and $\mathbf{s}(k)$ the $(1 \times p)$ external complex-valued input vector. The

[2]A meromorphic function is a single-valued function that is analytic in all but possibly a discrete subset of its domain, and at those singularities, it must approach infinity "like a polynomial" (these exceptional points must be poles and not essential singularities).

[3]A singularity that is neither removable nor isolated is known as essential singularity [31].

[4]In a split-complex AF, the real and imaginary component of the complex-valued input signal x are split and fed through the real-valued activation function $f_R(x) = f_I(x)$, $x \in \mathbb{R}$. The functional expression of the split-complex activation function is given by $f(x) = f_R(\mathrm{Re}(x)) + j f_I(\mathrm{Im}(x))$. Notice that this approach does not account for a "fully" complex signal, where the signal components are not independent.

overall input to the network $\mathbf{P}(k)$ then represents a concatenation of vectors $\mathbf{y}(k)$, $\mathbf{s}(k)$ and the bias input $(1+j)$ and is given by

$$\begin{aligned}\mathbf{P}(k) &= [s(k-1),\ldots,s(k-p),\\ &\qquad 1+j, y_1(k-1),\ldots,y_N(k-1)]^T\\ &= P_n^r(k)+jP_n^i(k),\quad n=1,\ldots,p+N+1 \end{aligned} \tag{1}$$

where $j=\sqrt{-1}$, $(\cdot)^T$ denotes the vector transpose operator, and superscripts $(\cdot)^r$ and $(\cdot)^i$ denote, respectively, the real and imaginary part of a complex number or complex vector.

For the lth neuron, its weights form a $(p+N+1)\times 1$-dimensional weight vector $\mathbf{w}_l^T = [w_{l,1},\ldots,w_{l,p+N+1}]$, $l=1,\ldots,N$, which are encompassed in the complex-valued weight matrix of the network $\mathbf{W}=[\mathbf{w}_1,\ldots,\mathbf{w}_N]$.

The output of every neuron can be expressed as

$$y_l(k)=\Phi\left(net_l(k)\right),\quad l=1,\ldots,N \tag{2}$$

where Φ is a complex nonlinear activation function of a neuron, and

$$\text{net}_l(k)=\sum_{n=1}^{p+N+1} w_{l,n}(k)P_n(k) \tag{3}$$

is the net input to lth node at time index k. For simplicity, we state that

$$y_l(k)=\Phi^r\left(net_l(k)\right)+j\Phi^i\left(net_l(k)\right)=u_l(k)+jv_l(k). \tag{4}$$

B. Complex-Valued RTRL Algorithm

For nonlinear adaptive prediction applications, there is only one neuron for which the output is defined by $y_1(k)$ The output error, which consists of its real $e^r(k)$ and imaginary $e^i(k)$ part, can be expressed as

$$e(k)=d(k)-y_1(k)=e^r(k)+je^i(k) \tag{5}$$

$$e^r(k)=d^r(k)-u_1(k),\quad e^i(k)=d^i(k)-v_1(k) \tag{6}$$

where $d(k)=d^r(k)+jd^i(k)$ is the teaching signal. For real-time applications and gradient descent algorithms, the cost function is given by [39]

$$E(k)=\frac{1}{2}|e(k)|^2=\frac{1}{2}e(k)e^*(k)=\frac{1}{2}\left[(e^r)^2+(e^i)^2\right] \tag{7}$$

where $(\cdot)^*$ denotes the complex conjugate operator. The CRTRL aims at minimizing the error by recursively altering the weight cofficients based on the gradient descent. Thus, for every weight $w_{l,n}\in\mathbf{W}$, $l=1,\ldots,N$, $n=1,\ldots,p+N+1$, we have

$$\begin{aligned} w_{l,n}(k+1) &= w_{l,n}(k)+\Delta w_{l,n}(k)\\ &= w_{l,n}(k)-\eta\nabla_{w_{l,n}}E(k)_{|w_{l,n}=w_{l,n}(k)} \end{aligned} \tag{8}$$

where η is the learning rate, which is typically a small positive constant. Notice that $E(k)$ is a real-valued function, and to calculate the gradient, we are required to derive partial derivates of $E(k)$ with respect to both the real and imaginary part of the weight coefficients separately, that is

$$\nabla_{w_{l,n}}E(k)=\frac{\partial E(k)}{\partial w_{l,n}^r}+j\frac{\partial E(k)}{\partial w_{l,n}^i}. \tag{9}$$

Calculating the gradient of the cost function with respect to the real part of the complex weight $\mathbf{w}^r$ gives[5]

$$\begin{aligned}\frac{\partial E(k)}{\partial w_{l,n}^r(k)}=\frac{\partial E}{\partial u_1}\left(\frac{\partial u_1(k)}{\partial w_{l,n}^r(k)}\right)+\frac{\partial E}{\partial v_1}\left(\frac{\partial v_1(k)}{\partial w_{l,n}^r(k)}\right)\\ l=1,\ldots,N\\ n=1,\ldots,p+N+1\end{aligned}. \tag{10}$$

Similarly, the partial derivative of the cost function with respect to the imaginary part of the complex weight $\mathbf{w}^i$ yields

$$\begin{aligned}\frac{\partial E(k)}{\partial w_{l,n}^i(k)}=\frac{\partial E}{\partial u_1}\left(\frac{\partial u_1(k)}{\partial w_{l,n}^i(k)}\right)+\frac{\partial E}{\partial v_1}\left(\frac{\partial v_1(k)}{\partial w_{l,n}^i(k)}\right)\\ l=1,\ldots,N\\ n=1,\ldots,p+N+1\end{aligned}. \tag{11}$$

The terms $\partial u_1(k)/\partial w_{l,n}^r(k)$, $\partial v_1(k)/\partial w_{l,n}^r(k)$, $\partial u_1(k)/\partial w_{l,n}^i(k)$, and $\partial v_1(k)/\partial w_{l,n}^i(k)$ are measures of the sensitivity of the output of the lth neuron at time instant k to a small variation in the value of $w_{l,n}(k)$. For convenience, we denote the above sensitivities as $\pi_{l,n}^{1,rr}(k)=\partial u_1(k)/\partial w_{l,n}^r(k)$, $\pi_{l,n}^{1,ir}(k)=\partial v_1(k)/\partial w_{l,n}^r(k)$, $\pi_{l,n}^{1,ri}(k)=\partial u_1(k)/\partial w_{l,n}^i(k)$, and $\pi_{l,n}^{1,ii}(k)=\partial v_1(k)/\partial w_{l,n}^i(k)$. For a gradient algorithm to be operating in the complex domain, we require a complex activation function to be analytic in $\mathbb{C}$, that is, it needs to satisfy the Cauchy–Riemann[6] equations. To make use of the Cauchy–Riemann equations, the partial derivatives of $E(k)$ (sensitivities) along the real and imaginary axes should be made equal, that is, for every neuron [19]

$$\pi_{l,n}^1(k)=\pi_{l,n}^{1,rr}(k)+j\pi_{l,n}^{1,ir}(k)=\pi_{l,n}^{1,ii}(k)-j\pi_{l,n}^{1,ri}(k). \tag{12}$$

Equating the real and imaginary parts on both sides of (12), we obtain

$$\pi_{l,n}^{1,rr}(k)=\pi_{l,n}^{1,ii}(k),\quad \pi_{l,n}^{1,ri}(k)=-\pi_{l,n}^{1,ir}(k). \tag{13}$$

A compact representation of $\nabla_{w_{l,n}}E(k)$ becomes

$$\nabla_{w_{l,n}}E(k)\equiv -e(k)\left(\pi_{l,n}^1(k)\right)^* \tag{14}$$

with the initial condition

$$\pi_{l,n}^1(0)=0. \tag{15}$$

[5]We derive the CRTRL for adaptive filtering applications (only one output y_1); however, the derivation is general enough to be straightforwardy extended to an RNN with more than one output.

[6]Cauchy–Riemann equations state that the partial derivatives of a function $f(z)=u(x,y)+jv(x,y)$ along the real and imaginary axes should be equal: $f'(z)=(\partial u/\partial x)+j(\partial v/\partial x)=(\partial v/\partial y)-j(\partial u/\partial y)$. Therefore, we obtain the Cauchy–Riemann equations as $(\partial u/\partial x)=(\partial v/\partial y)$, $(\partial v/\partial x)=-(\partial u/\partial y)$.

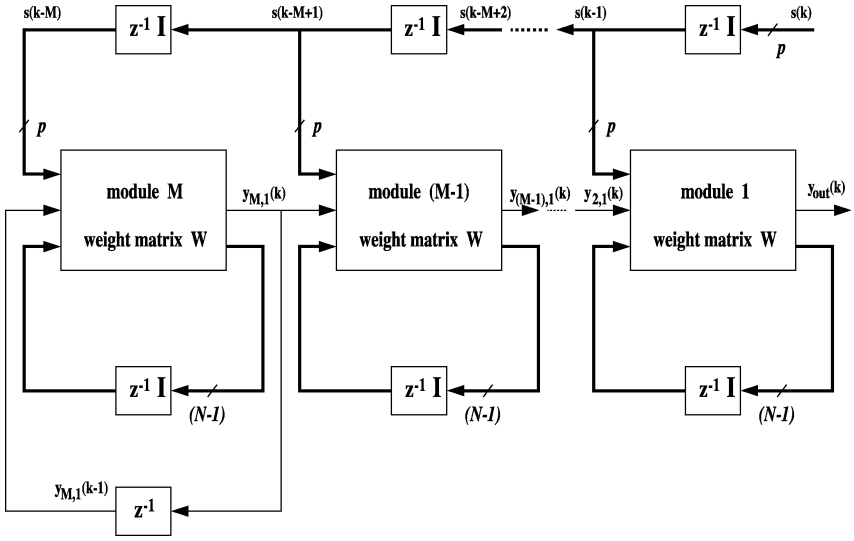


Fig. 2. Pipelined Recurrent Neural Network (PRNN).

Extending the approach from [40] to the complex-valued case, the update for the sensitivity $(\pi^1_{l,n}(k))^* = (\partial u_1(k)/\partial w^r_{l,n}(k)) - j(\partial v_1(k)/\partial w^r_{l,n}(k))$ can be derived as

$$\left(\pi^1_{l,n}(k)\right)^* = \{\Phi^*(k)\}' \times \left[\sum_{\alpha=1}^{N} w^*_{1,\alpha+p+1}(k)\left(\pi^\alpha_{l,n}(k-1)\right)^* + \delta_{ln}P^*_n(k)\right] \quad (16)$$

where

$$\delta_{ln} = \begin{cases} 1, & l = n \\ 0, & l \neq n \end{cases} \quad (17)$$

is the Kronecker delta. Finally, the total weight matrix update is given by

$$\mathbf{W}(k+1) = \mathbf{W}(k) + \eta e(k)\overline{\mathbf{\Pi}}^*(k) \quad (18)$$

where $\mathbf{\Pi}^*(k)$ are the matrix sensitivities $\pi^l_{l,n}(k), l = 1,\ldots,N$, $n = 1,\ldots,p+N+1$. This completes the derivation of the general "fully" complex RTRL. Notice that the derived algorithm has a generic form of the real-valued RTRL [40].

III. Complex-Valued Pipelined Recurrent Neural Network (CPRNN)

The nonlinear adaptive filtering architecture, as proposed by Haykin and Li, consists of two sections, namely, the nonlinear and linear one. The nonlinear section, which is called the pipelined recurrent neural network (PRNN), is essentially a modular RNN and performs nonlinear filtering, whereas the linear section represented by a finite impulse response (FIR) filter performs linear filtering of the signal. This cascaded combination of the PRNN and an FIR filter has been shown to be suitable for nonlinear prediction of real-valued nonstationary signals.

A. Complex PRNN

The CPRNN architecture contains M modules of FCRNN's connected in a nested manner, as shown in Fig. 2. The $(p \times 1)$-dimensional external complex-valued signal vector $\mathbf{s}^t(k) = [s(k-1),\ldots,s(k-p)]$ is delayed by m time steps $(z^{-m}\mathbf{I})$ before feeding the module m, where z^{-m}, $m = 1,\ldots,M$ denotes the m-step time delay operator, and $\mathbf{I}$ is the $(p \times p)$-dimensional identity matrix. The complex-valued weight vectors $\mathbf{w}_l$ are embodied in an $(p+N+1) \times N$ dimensional weight matrix $\mathbf{W} = [\mathbf{w}_1,\ldots,\mathbf{w}_N]$. All the modules operate using the same weight matrix $\mathbf{W}$ (a full mathematical description of the PRNN is given in [14] and [26]). The following equations provide a mathematical description of the CPRNN

$$y_{t,l}(k) = \Phi\left(net_{t,l}(k)\right), \quad t = 1,2,\ldots,M \quad (19)$$

$$net_{t,l}(k) = \sum_{n=1}^{p+N+1} w_{l,n}(k)P_{t,n}(k) \quad l = 1,\ldots,N \quad n = 1,\ldots,p+N+1 \quad (20)$$

$$\mathbf{P}^T_t(k) = [s(k-t),\ldots,s(k-t-p+1),1+j\, y_{t+1,1}(k-1),y_{t,2}(k-1),\ldots,y_{t,N}(k-1)] \quad \text{for } 1 \le t \le M-1 \quad (21)$$

$$\mathbf{P}^T_M(k) = [s(k-M),\ldots,s(k-M-p+1),1+j\, y_{M,1}(k-1),y_{M,2}(k-1),\ldots,y_{M,N}(k-1)] \quad \text{for } t = M. \quad (22)$$

For simplicity, we state that

$$y_{t,l}(k) = \Phi\left(net_{t,l}(k)\right) = \Phi^r\left(net_{t,l}(k)\right) + j\Phi^i\left(net_{t,l}(k)\right) = u_{t,l}(k) + jv_{t,l}(k). \quad (23)$$

The overall output of the CPRNN is $y_{1,1}(k)$, that is, the output of the first neuron of the first module. At every time instant k, for every module t, $t = 1,\ldots,M$, the one-step-ahead instantaneous prediction error $e_t(k)$ associated with a module is then defined as

$$e_t(k) = s(k-t+1) - y_{t,1}(k) = e^r_t(k) + je^i_t(k). \quad (24)$$

We can split the error term into its real and imaginary part, given by

$$e^r_t(k) = s^r(k-t+1) - u_{t,1}(k), \quad e^i_t(k) = s^i(k-t+1) - v_{t,1}(k). \quad (25)$$

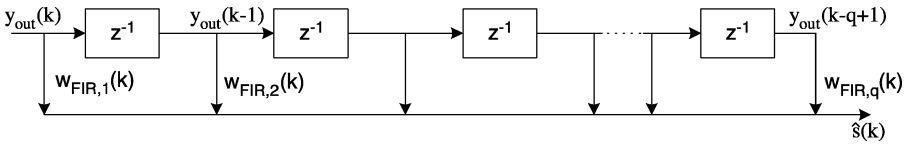


Fig. 3. An FIR filter.

Since the CPRNN consists of M modules, a total of M forward prediction error signals are calculated. The cost function of the PRNN introduced in [14] is now modified to suit processing in the complex domain and is given by

$$\begin{aligned} E(k) &= \sum_{t=1}^{M} \lambda^{t-1}(k) \left|e_t(k)\right|^2 \\ &= \sum_{t=1}^{M} \lambda^{t-1}(k) \left[e_t(k) e_t^*(k)\right] \\ &= \sum_{t=1}^{M} \lambda^{t-1}(k) \left[(e_t^r)^2 + \left(e_t^i\right)^2\right] \end{aligned} \tag{26}$$

which represent the weighted sum of instantaneous squared errors from outputs of the CPRNN modules, where $\lambda(k)$ is a (possible variable) forgetting factor. The forgetting factor $\lambda < 1$ plays a very important role in nonlinear adaptive filtering of nonstationary signals and is usually set to unity for stationary signals. Since, for gradient descent learning, we aim to minimize (26) along the entire CPRNN, the weight update for the nth weight at neuron l at the time instant k is calculated as

$$\Delta w_{l,n}(k) = -\eta \frac{\partial}{\partial w_{l,n}(k)} \left(\sum_{t=1}^{M} \lambda^{t-1}(k) \left|e_t(k)\right|^2 \right). \tag{27}$$

Following the derivation of the CRTRL, for $l = 1, \ldots, N$ and $n = 1, \ldots, p+N+1$, the weight update of every weight within the CPRNN can now be expressed as

$$\begin{aligned} & w_{l,n}(k+1) \\ &\quad = w_{l,n}(k) - \eta \nabla_{w_{l,n}} E(k)_{|w_{l,n}=w_{l,n}(k)} \\ &\quad = w_{l,n} + \eta \left(\sum_{t=1}^{M} \lambda^{t-1}(k) e_t(k) \left(\pi_{l,n}^{t,1}(k)\right)^* \right) \\ &\quad = w_{l,n}(k) + \eta \left(\sum_{t=1}^{M} \lambda^{t-1}(k) e_t(k) \left\{\Phi^*\left(net_{t,l}(k)\right)\right\}' \right. \\ &\qquad \times \left[\sum_{\alpha=1}^{N} w_{1,\alpha+p+1}^*(k) \left(\pi_{l,n}^{t,\alpha}(k-1)\right)^* \right. \\ &\qquad \left. \left. + \delta_{ln} P_{t,n}^*(k) \right] \right). \end{aligned} \tag{28}$$

Notice that this weight update has the same generic form as the weight update of the real-valued PRNN [14].

B. Linear Subsection

The linear subsection of the CPRNN consists of an FIR filter, which is shown in Fig. 3. The complex-valued least mean square (CLMS) algorithm is used to update the tap weights of this filter, for which the output is given by

$$\hat{s}(k) = \mathbf{w}_{\mathrm{FIR}}^T(k) \mathbf{y}_{\mathrm{out}}(k) \tag{29}$$

where $\mathbf{y}_{\mathrm{out}}(k) \triangleq [y_{\mathrm{out},1}(k), \ldots, y_{\mathrm{out},q}(k)]^T$ is the output from the first CPRNN module $(y_{1,1}(k))$, $\mathbf{w}_{\mathrm{FIR}}(k) \triangleq [w_{\mathrm{FIR},1}, \ldots, w_{\mathrm{FIR},q}(k)]^T$ the complex weight vector, and q the number of tap inputs. The output $\hat{s}(k)$ is a prediction of the sample $s(k)$ of the original signal. The error signal $e(k)$ required for adaptation of finite weights is obtained as the difference between the desired response $s(k)$ and the output of the filter $\hat{s}(k)$ and is given by

$$e(k) = s(k) - \hat{s}(k) = s(k) - \mathbf{w}_{\mathrm{FIR}}^T(k) \mathbf{y}_{\mathrm{out}}(k) \tag{30}$$

The weight update term for the CLMS algorithm is given by [39]

$$\mathbf{w}_{\mathrm{FIR}}(k+1) = \mathbf{w}_{\mathrm{FIR}}(k) + \mu e(k) \mathbf{y}_{\mathrm{out}}^*(k) \tag{31}$$

where μ is the learning rate.

IV. Adaptive Amplitude CRTRL (AACRTRL) Algorithm

Before performing the actual nonlinear filtering by neural nonlinear filters, a signal is usually first standardized to match the dynamical range of the nonlinearity within the neuron model.[7] Therefore, making the amplitude of the activation fuction adaptive might prove beneficial and help avoid some common problems experienced in neural network training, such as saturation, which dramatically slows down learning when the net input is mapped onto the tails of the sigmoid. In this section, we derive such a direct gradient descent algorithm for complex RNNs equipped with an adaptive amplitude of the activation function, which is called the adaptive amplitude CRTRL algorithm (AACRTRL).

The idea behind this learning algorithm is to obtain an activation function that adapts its range according to the dynamical changes of the input signal. Following the approach from [35], we can rewrite the nonlinear activation function Φ as

$$\Phi\left(\mathrm{net}_l(k)\right) = \psi_l(k) \overline{\Phi}\left(\mathrm{net}_l(k)\right) \tag{32}$$

where $\psi_l(k)$ is a variable that governs the amplitude of $\Phi(net_l(k))$, and $\overline{\Phi}(net_l(k))$ is the (referent) activation function with a unit amplitude. Thus, if $\psi_l(k) = 1$, then $\Phi(net_l(k)) = \overline{\Phi}(net_l(k))$. To make the amplitude $\psi_l(k)$ of

[7] For instance, if the activation function of a real-valued neuron is the logistic function $\Phi(x) = (1/1 + e^{-z})$ for which the mean is 1/2, then the range of the output is (0,1), and the inputs should be rescaled to match the mean and range of the logistic function. This causes problems in real-time signal processing when the range of the input is not known beforehand. Notice that such problems are not experienced with linear filters.

$\Phi(net_l(k))$ be time varying, we propose the update for the gradient adaptive amplitude as[8] [11], [21], [35]

$$\psi_l(k+1) = \psi_l(k) - \rho \nabla_{\psi_l} E(k)_{|\psi_l(k)} \tag{33}$$

where $\nabla_{\psi_l} E(k)$ denotes the gradient of the objective function $E(k)$ with respect to the amplitude of the activation function ψ_l, and ρ denotes the step size of the algorithm and is chosen to be a small positive constant. From (33), we have

$$\nabla_{\psi_l} E(k) = -\frac{\partial y_l(k)}{\partial \psi_l(k)} e(k) \tag{34}$$

where

$$y_l = \psi_l(k) \overline{\Phi}\left(net_l(k)\right). \tag{35}$$

The partial derivatives from (34) can be computed as

$$\begin{aligned} \frac{\partial y_l(k)}{\partial \psi_l(k)} &= \overline{\Phi}\left(net_l(k)\right) + \psi_l(k) \overline{\Phi}'\left(net_l(k)\right) \frac{\partial \left[net_l(k)\right]}{\partial \psi_l(k)} \\ &= \overline{\Phi}\left(net_l(k)\right) + \psi_l(k) \overline{\Phi}'\left(net_l(k)\right) \\ &\quad \times \frac{\partial}{\partial \psi_l(k)} \left(\sum_{n=1}^{p+N+1} w_{l,n}(k) P_n(k) \right). \end{aligned} \tag{36}$$

Since $(\partial \psi_l(k-1)/\partial \psi_l(k)) = 0$, the second term in (36) vanishes. This way, we have

$$\nabla_{\psi_l(k)} E(k) = \frac{\partial E(k)}{\partial \psi_l(k)} = -e(k) \overline{\Phi}\left(net_l(k)\right). \tag{37}$$

For simplicity, let all the neurons in the complex RNN share a common amplitude ψ, that is, $\psi_l(k) = \psi(k)$, to yield

$$y_l(k) = \Phi\left(net_l(k)\right) = \psi(k) \overline{\Phi}\left(net_l(k)\right). \tag{38}$$

This way, a simplified gradient update of the adaptive amplitude of the activation function of a neuron becomes

$$\begin{aligned} \psi(k+1) &= \psi(k) - \rho \nabla_{\psi(k)} E(k) \\ &= \psi(k) + \rho e(k) \overline{\Phi}\left(net_l(k)\right) \end{aligned} \tag{39}$$

which concludes the derivation of the AACRTRL.

V. Simulations

For the experiments, the nonlinearity at the neuron was chosen to be the complex logistic sigmoid function

$$\Phi(x) = \frac{1}{1 + e^{-\beta x}} \tag{40}$$

where $x \in \mathbb{C}$. The value of the slope of $\Phi(x)$ was $\beta = 1$, and the value of the learning rate for the CPRNN architecture was $\eta = 0.01$, whereas the learning rate for the FIR filter was $\mu = 0.3$. The forgetting factor for the CPRNN architecture was $\lambda = 0.8$

[8]Notice that this strategy can be applied for all the nonlinear functions for which $\Phi(\psi, x) = \psi \Phi(x)$, which is the case with the logistic, tanh, and many other commonly used activation functions.

TABLE I
Prediction Gain R_p Versus the Number of Modules M for Wind Signal, With $p = 1$ and $p = 4$

R_p(dB)	M=1	M=2	M=3	M=4	M=5	M=6	M=7	M=8	M=9	M=10
p=1	1.4209	6.129	8.7621	8.8148	8.8972	8.9214	9.4747	9.5727	9.6214	9.6895
p=4	1.7371	5.724	10.0744	10.0914	10.1152	10.1642	10.7420	10.7652	10.7771	10.7789

Complex-valued signals used for simulations were chosen to belong to different classes of signals, namely, nonlinear (42), colored (41), two complex-valued nonstationary (synthetic) speech signals $s1$, $s2$,[9] real-life wind data[10] (velocity and direction), and radar data.[11] The colored input signal was a stable complex $AR(4)$ process given by

$$\begin{aligned} r(k) = 1.79r(k-1) - 1.85r(k-2) \\ +1.27r(k-3) - 0.41r(k-4) + n(k) \end{aligned} \tag{41}$$

with complex white Gaussian noise (CWGN) $n(k) \sim \mathcal{N}(0,1)$ as the driving input. The real $n^r(k)$ and imaginary $n^i(k)$ component of the CWGN $n(k) = n^r(k) + jn^i(k)$ were mutually independent sequences having equal variances so that $\sigma_n^2 = \sigma_{n^r}^2 + \sigma_{n^i}^2$. The complex nonlinear input signal was [30]

$$z(k) = \frac{z(k-1)}{1 + z^2(k-1)} + r^3(k). \tag{42}$$

The measurement used to assess the performance was the prediction gain R_p given by [14]

$$R_p(k) \triangleq 10 log_{10} \left(\frac{\sigma_s^2}{\hat{\sigma}_e^2} \right) \text{ db} \tag{43}$$

where σ_s^2 denotes the variance of the input signal $s(k)$, whereas $\hat{\sigma}_e^2$ denotes the estimated variance of the forward prediction error $e(k)$. Following the approach from [14], the initialization procedure for all the experiments was epochwise, with 200 epochs consisting of 1000 samples each.

A. Effect of the Number of Modules and Tap Input Signals

To investigate the effect of the number of tap inputs p to the performance, the prediction gain R_p was calculated for the number of modules varying from $M = 1$ to $M = 10$ and for a fixed number of neurons within a module $N = 2$. Notice that $M = 1$ corresponds to the case of a single FCRNN. Table I shows the prediction gains for a complex real-world nonlinear and nonstationary wind signal for both $p = 1$ and $p = 4$ and all the other relevant parameters shared. The case with $p = 4$ achieved better performance as compared to the case with $p = 1$. The prediction gain increased with the number of modules of CPRNN. Notice that in both cases, R_p saturated after $M = 8$. Table II further illustrates the relationship between R_p and p with $M = 8$ (stable performance) for a complex wind signal. The results shows that R_p improves with increasing p

[9]Publicly available from "http://www.commsp.ee.ic.ac.uk/~mandic."

[10]Publicly available from "http://mesonet.agron.iastate.edu./request/awos/1min.php."

[11]Publicly available from "http://soma.ece.mcmaster.ca/ipix/."

TABLE II
PREDICTION GAIN R_p VERSUS THE NUMBER OF EXTERNAL INPUTS p FOR WIND SIGNAL

R_p(dB)	p=1	p=2	p=3	p=4	p=5	p=6	p=7	p=8	p=9	p=10
M=8	9.484	9.6457	10.5730	10.7652	10.8206	10.7865	10.6178	10.4337	10.2433	9.9869

TABLE III
PREDICTION GAIN R_p VERSUS THE NUMBER OF NEURONS N FOR WIND SIGNAL

R_p(dB)	N=1	N=2	N=3	N=4	N=5
M=8	10.8605	10.8578	10.8581	10.6326	9.93268

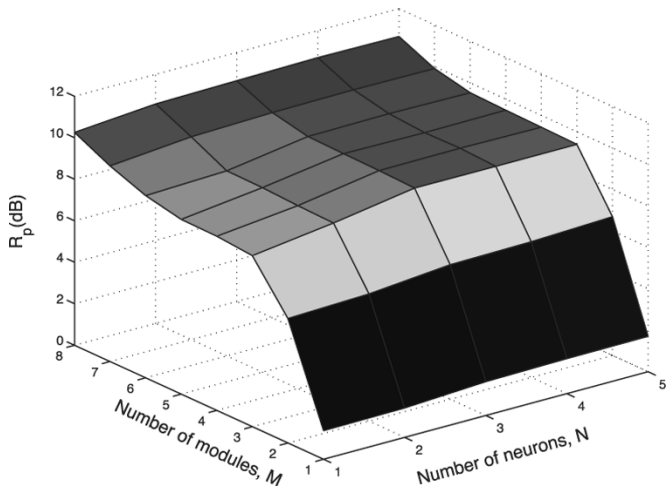


Fig. 4. Relationship between prediction gain R_p, number of modules M, and number of neurons N for nonlinear prediction of the complex wind signal.

from $p = 1$ to $p = 5$ but started to decrease from $p = 5$ onwards, which is attributed to overfitting. On the whole, the CPRNN showed a fairly consistent performance over a range of input sizes p.

B. Effect of the Number of Neurons Within a Module

To investigate the effect of the number of neurons within a module on the performance, prediction gain R_p was calculated for the number of neurons within each FCRNN, representing a module of the FCRNN varying from $N = 1$ to $N = 5$. From the previous experiments, the number of modules was set to $M = 8$, and the number of tap inputs $p = 5$. Table III shows the prediction gains for the complex wind signal and all the other learning parameters shared. From Table III, increasing the number of neurons N within a module did not improve the performance of the network. Fig. 4 illustrates the results shown in Tables I and III. From Fig. 4, CPRNN is robust to the change of the number of neurons within the modules and performs consistently for $M > 3$. There is a saturation in performance for large number of modules and neurons within a module.

C. Effect of Employing Adaptive Amplitude in the Nonlinearity

For the experiments in this section, the step size of the adaptive amplitude adaptation was set to $\rho = 0.15$. The number of modules was $M = 8$, the number of neurons within a module $N = 1$, and the number of tap inputs $p = 5$. For simulations, 100 iterations of independent trials were averaged on prediction of the colored (41) and nonlinear (42) input signals.

Sensitivity of CRTRL to the choice of initial amplitude $\psi(0)$ is illustrated in Table IV, where the influence of the initial amplitude $\psi(0)$ is depicted for both the colored (41) and nonlinear (42) stochastic inputs. From Table IV, the performance improved with the increase in $\psi(0)$. However, for initial values $\psi(0) > 3$, the algorithm did diverge. For further simulations, we chose $\psi = 2.75$. The reason for the algorithm to diverge for initial values $\phi(0) > 3$ is due to the $\beta \div \eta$ relationship in RNNs.[12] As the initial amplitude $\psi(0)$ increases, the effective steepness of Φ increases, and the sigmoid approaches a hard limiter, which is equivalent to an increase in the learning rate. The relationship between the slope β within a general activation function and the learning rate η in the RTRL-based learning of a general RNN and PRNN is thoroughly addressed in [12] and [25].

Table V shows a comparison of the prediction gains between the CRTRL and AACRTRL for various classes of signals. From the experiment, we can see that employing an adaptive amplitude in the nonlinearity improves the performance of the CPRNN architecture for all the input classes used at a cost of a small increase in computational complexity.

Fig. 5 shows the averaged performance curves over 100 iterations of independent trials for the complex-value real-time recurrent learning (CRTRL) algorithm and the adaptive amplitude complex-valued real-time recurrent learning (AACRTRL) algorithm on prediction of nonlinear (42) and colored (41) input by CPRNN. From Fig. 5, the AACRTRL exhibits a faster initial convergence than CRTRL, together with an improved performance.

To further depict the behavior of the gradient adaptive amplitude of the activation function, Fig. 6 illustrates a time variation of the adaptive amplitude $\psi(k)$ tested on a real-world speech recording that, for the purpose of this experiment, was made complex valued. The AACRTRL was clearly able to adapt the amplitude of the nonlinearity according to the changes in the dynamics of the input.

To verify the advantage of using the fully CRTRL over split CRTRL, we compared the performances of CPRNN trained with these algorithms in experiments on real-world radar and wind data. The results of simulations for radar data are shown in Fig. 7. Observe that the fully CRTRL algorithm was more stable and has exhibited better and more consistent performance than the split CRTRL algorithm. Fig. 8 shows the prediction performance of the CPRNN applied to the complex-valued real-world (velocity and angle components) wind signal in both the split and fully CRTRL case. The CPRNN was able to track the complex wind signal when the fully CRTRL algorithm was employed, which was not the case for split CPRNN.

[12]For example, consider functions $\Phi_1 = (1/1 + e^{-\beta x})$, $\Phi_2 = (3/1 + e^{-\beta x})$, and $\beta = 1$. The derivatives of those functions at the origin are $\Phi_1' = 1/4$ and $\Phi_2' = 3/4$, which is due to the difference in their amplitudes. Notice that the slope β in both functions was identical.

TABLE IV
PREDICTION GAIN R_p VERSUS INITIAL AMPLITUDE $\psi(0)$ FOR THE NONLINEAR (42) AND COLORED (41) SIGNAL

Rp(dB) / Initial amplitude	0.25	0.75	1.25	1.75	2.25	2.75	3.00	3.25
nonlinear signal	1.2314	1.6244	2.5328	3.1341	6.2208	12.4994	16.5161	NaN
coloured signal	0.81357	1.0379	2.2613	3.7972	4.7633	9.9086	10.9761	NaN

TABLE V
COMPARISON OF PREDICTION GAINS R_p BETWEEN THE FIXED AND TRAINABLE AMPLITUDE CRTRL

Rp(dB)	Nonlinear Input	Coloured Input	Speech s1	Speech s2	Wind	Radar
CRTRL with fixed amplitude	16.0543	11.1143	12.1592	13.3233	13.4802	8.1131
CRTRL with trainable amplitude	18.8711	13.9733	14.0783	14.7579	14.2618	10.0224

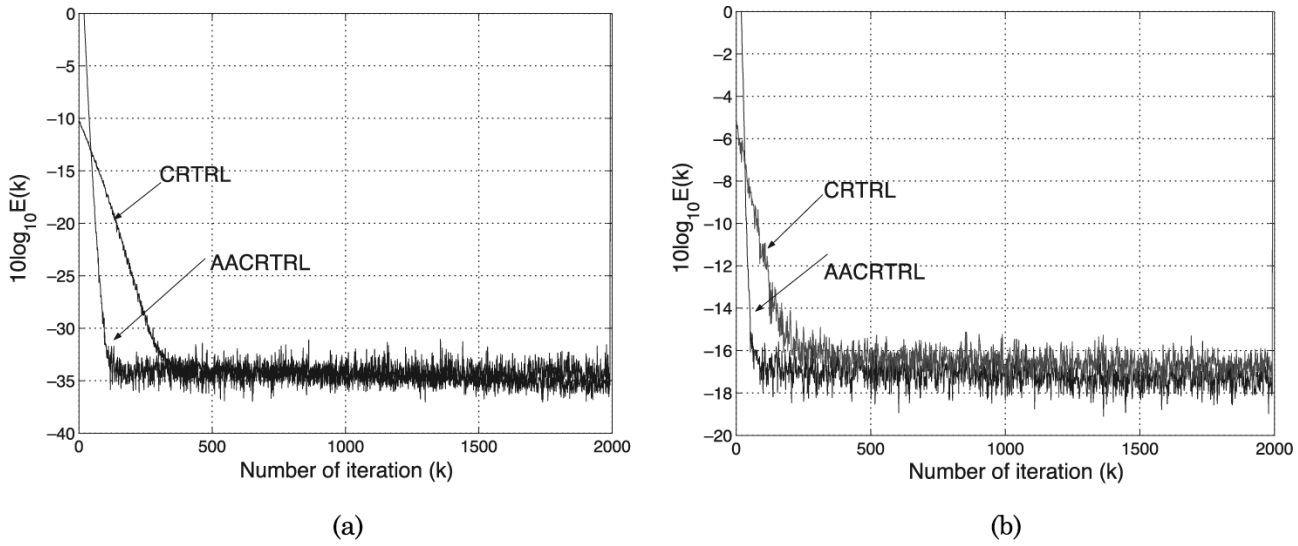


Fig. 5. Performance of CPRNN on a nonlinear and colored signals. (a) Performance of CPRNN on prediction of nonlinear input (42). (b) Performance of CPRNN on prediction of colored input (41).

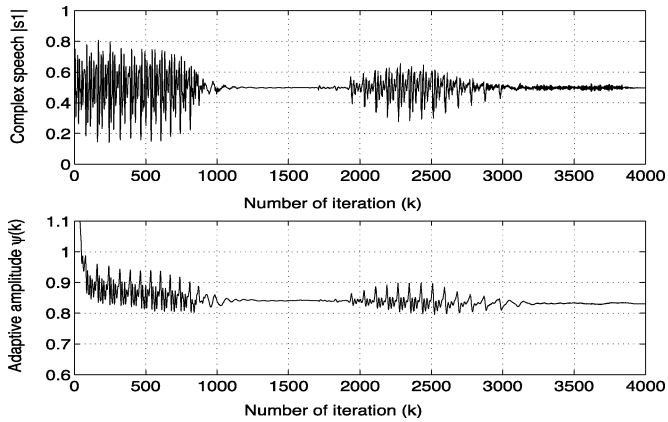


Fig. 6. Adaptive amplitudes for AACRTRL on complex speech signal s1.

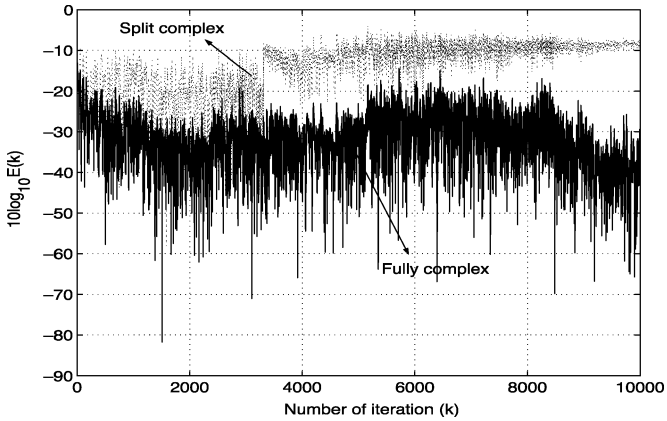


Fig. 7. Performance curve of fully CRTRL and split CRTRL for radar signal.

Table VI shows the comparison of average prediction gains (average over 100 independent trials) between the CPRNN + FIR and a single FCRNN for several general classes of complex signals. In all the cases, there was a significant improvement in the prediction gain when the CPRNN + FIR architecture was employed over the performance of a single module FCRNN.

VI. CONCLUSIONS

A complex-valued pipelined recurrent neural network (CPRNN) for prediction of nonlinear and nonstationary complex-valued signals has been presented. First, the complex-valued real-time recurrent learning (CRTRL) algorithm for

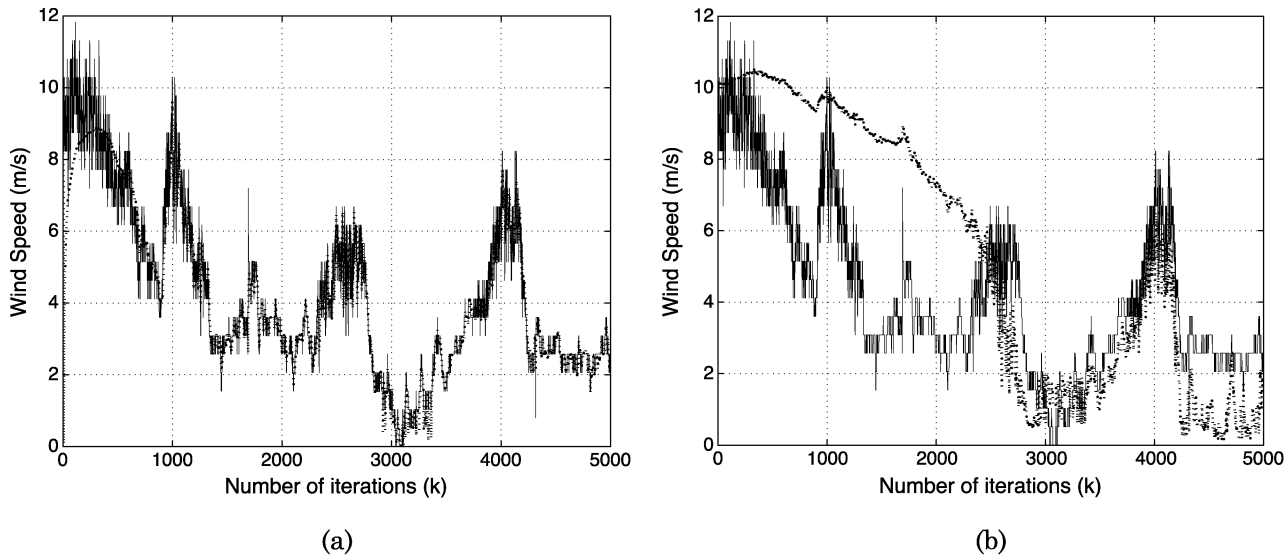


Fig. 8. Prediction of complex wind signal using CPRNN+FIR. Solid curve: actual wind signal. Dashed curve: nonlinear prediction. (a) Performance based on fully complex activation function; (b) performance based on split complex activation function.

TABLE VI
COMPARISON OF PREDICTION GAINS R_P BETWEEN THE CPRNN + FIR AND FCRNN SCHEME

Rp(dB)	Nonlinear Input	Coloured Input	Speech s1	Speech s2	Wind	Radar
CPRNN	19.9308	12.6789	14.1771	14.8341	14.3372	9.8235
single FCRNN	9.4781	5.2835	5.6379	6.4102	1.9064	3.0265

single fully connected complex-valued recurrent neural networks (FCRNNs) has been introduced, which has been derived for a general meromorphic complex activation function of a neuron. The proposed CRTRL algorithm has been shown to be generic and applicable for a variety of classes of complex signals, including those with strong component correlations. The CRTRL has then been modified to suit the CPRNN architecture. For nonlinear prediction of nonlinear and nonstationary complex-valued signals with an unknown dynamical range, a variant of the CRTRL algorithm with a gradient adaptive amplitude of the nonlinearity within the neurons (AACRTRL) has been derived. The performance of the CPRNN architecture and proposed algorithms has been evaluated on the classes of complex-valued nonlinear, colored, and real-life signals. Both synthetic and real-life complex-valued nonstationary signals have been used in the experiments to further demonstrate the performance improvement provided by the CPRNN over a single FCRNN.

ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers for their insightful comments.

REFERENCES

[1] N. Benvenuto and F. Piazza, "On the complex backpropagation algorithm," *IEEE Trans. Signal Process.*, vol. 40, no. 4, pp. 967–969, Apr. 1992.

[2] G. E. P. Box, G. M. Jenkins, and G. C. Reinsel, *Time Series Analysis: Forecasting and Control*. Englewood Cliffs, NJ: Prentice-Hall, 1994.

[3] 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.

[4] J. T. Connor, R. D. Martin, and L. E. Atlas, "Recurrent neural networks and robust time series prediction," *IEEE Trans. Neural Networks*, vol. 5, no. 2, pp. 240–254, Mar. 2001.

[5] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Math. Contr., Signals, Syst.*, vol. 2, pp. 303–314, 1989.

[6] R. Drossu and Z. Obradovic, "Rapid design of neural networks for time series prediction," *IEEE Computat. Sci. Eng.*, vol. 3, no. 2, pp. 78–89, Summer 1996.

[7] K. I. Funahashi, "On the approximate realization of continuous mappings by neural networks," *Neural Networks*, vol. 2, pp. 183–192, 1989.

[8] T. Gautama, D. P. Mandic, and M. M. Van Hulle, "A nonparametric test for detecting the complex-valued nature of time series," in *Proc. Knowledge-Based Intell. Inf. Eng. Syst.: Seventh Int. Conf.*, vol. 2773, Oxford, U.K., 2003, pp. 1364–1371.

[9] G. M. Georgiou and C. Koutsougeras, "Complex domain backpropagation," *IEEE Trans. Circuits Syst. II: Analog Digit. Signal Process.*, vol. 39, no. 5, pp. 330–334, May 1992.

[10] 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, 2003.

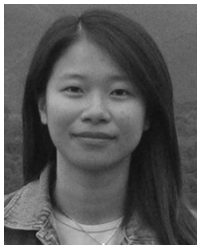
[11] A. I. Hanna and D. P. Mandic, "Nonlinear FIR adaptive filters with a gradient adaptive amplitude in the nonlinearity," *IEEE Signal Process. Lett.*, vol. 9, no. 8, pp. 253–255, Aug. 2002.

[12] ——, "A complex-valued nonlinear neural adaptive filter with a gradient adaptive amplitude of the activation function," *Neural Networks Lett.: Neural Netw.*, vol. 16, no. 2, pp. 155–159, 2003.

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

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

[15] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximations," *Neural Netw.*, vol. 2, pp. 359–366, 1989.
[16] Y. L. Hsieh, C. J. Chang, and Y. S. Chen, "A power control scheme with link gain prediction using PRNN/ERLS for DS-CDMA cellular mobile systems," in *Proc. IEEE Int. Conf. Commun.*, vol. 1, 2003, pp. 407–411.
[17] G. Kechriotis and E. S. Manolakos, "Training fully recurrent neural networks with complex weights," *IEEE Trans. Circuits Syst. II: Analog Digit. Signal Process.*, vol. 41, no. 3, pp. 235–238, Mar. 1994.
[18] T. Kim and T. Adali, "Approximation by fully complex MLP using elementary transcendental activation functions," in *Proc. XI IEEE Workshop Neural Networks Signal Process.*, 2001, pp. 203–212.
[19] ——, "Complex backpropagation neural network using elementary transcendental activation functions," in *Proc. Int. Conf. Acoust., Speech, Signal Process.*, vol. 2, 2001, pp. 1281–1284.
[20] ——, "Approximation by fully complex multilayer perceptrons," *Neural Comput.*, vol. 15, no. 7, pp. 1641–1666, 2003.
[21] I. R. Krcmar and D. P. Mandic, "A fully adaptive normalized nonlinear gradient descent algorithm for nonlinear system identification," in *Proc. Int. Conf. Acoust., Speech, Signal Process.*, vol. 6, 2001, pp. 3493–3496.
[22] H. Leung and S. Haykin, "The complex backpropagation algorithm," *IEEE Trans. Signal Process.*, vol. 39, no. 9, pp. 2101–2104, Sep. 1991.
[23] J. Makhoul, "Linear prediction: A tutorial review," *Proc. IEEE*, vol. 63, no. 4, pp. 140–148, Apr. 1975.
[24] D. Mandic, J. Baltersee, and J. Chambers, "Nonlinear adaptive prediction of speech with a pipelined recurrent neural network and advanced learning algorithms," in *Signal Analysis and Prediction*, A. Prochazka, J. Uhlir, P. W. Rayner, and N. G. Kingsbury, Eds. Boston, MA: Birkhauser, 1998.
[25] D. P. Mandic and J. A. Chambers, "Relating the slope of the activation function and the learning rate within a recurrent neural network," *Neural Comput.*, vol. 11, no. 5, pp. 1069–1077, 1999.
[26] ——, "Toward an optimal PRNN-based nonlinear predictor," *IEEE Trans. Neural Networks*, vol. 10, no. 6, pp. 1435–1442, Nov. 1999.
[27] ——, "On the choice of parameters of the cost function in nested modular RNNs," *IEEE Trans. Neural Networks*, vol. 11, no. 2, pp. 315–322, Mar. 2000.
[28] ——, *Recurrent Neural Networks for Prediction: Learning Algorithms, Architectures and Stability*. Chichester, U.K.: Wiley, 2001.
[29] L. R. Medsker and L. C. Jain, *Recurrent Neural Networks: Design and Applications*, ser. International Series on Computational Intelligence. Boca Raton, FL: CRC, 2000.
[30] K. S. Narendra and K. Parthasarathy, "Identification and control of dynamical systems using neural networks," *IEEE Trans. Neural Networks*, vol. 1, no. 1, pp. 4–27, Mar. 1990.
[31] T. Needham, *Visual Complex Analysis*. Oxford, U.K.: Oxford Univ. Press, 1997.
[32] O. Nerrand, P. Roussel-Ragot, L. Personnaz, and G. Dreyfus, "Neural networks and nonlinear adaptive filtering: unifying concepts and new algorithms," *Neural Comput.*, vol. 5, pp. 165–199, 1993.
[33] J. C. Principe, N. R. Euliano, and W. C. Lefebvre, *Neural and Adaptive Systems: Fundamentals Through Simulations*. New York: Wiley, 2000.
[34] F. C. Ren, C. J. Chang, and R. G. Cheng, "An intelligent transmission controller for TDMA/PRMA wireless multimedia communication systems," in *Proc. IEEE 50th Vehicular Technol. Conf.*, vol. 1, 1999, pp. 406–410.
[35] E. Trentin, "Networks with trainable amplitude of activation functions," *Neural Networks*, vol. 14, no. 5, pp. 471–493, 2001.
[36] L. Tsungnan, B. G. Horne, P. Tino, and C. L. Giles, "Learning long-term dependencies in NARX recurrent neural networks," *IEEE Trans. Neural Networks*, vol. 7, no. 6, pp. 1329–1338, Nov. 1996.
[37] F. Vitagliano, R. Parisi, and A. Uncini, "Generalized splitting 2D flexible activation function," in *Proc. 14th Italian Workshop Neural Nets*, 2003, pp. 85–95.
[38] B. Widrow and M. A. Lehr, "30 years of adaptive neural networks: perceptron, madaline, and backpropagation," *Proc. IEEE*, vol. 78, no. 9, pp. 1415–1442, Sep. 1990.
[39] B. Widrow, J. McCool, and M. Ball, "The complex LMS algorithm," *Proc. IEEE*, vol. 63, pp. 712–720, 1975.
[40] R. J. Williams and D. A. Zipser, "A learning algorithm for continually running fully recurrent neural networks," *Neural Comput.*, vol. 1, no. 2, pp. 270–280, 1989.



Su Lee Goh (S'04) received the B.Sc. (Hons) degree in electrical and electronic engineering from University Tenaga Nasional, Kajang, Malaysia, and the M.Sc. degree in communication and signal processing from Imperial College London, London, U.K. She is currently pursuing the Ph.D. degree in nonlinear adaptive signal processing at Imperial College London.

Her research interests include nonlinear signal processing, adaptive filters, complex-valued analysis, and forecasting.

Danilo P. Mandic (M'99–SM'03) received the B.Sc. Hons. degree in electrical and electronical engineering and the M.Sc. degree in signal processing from the University of Banja Luka, Banja Luka, Bosnia-Herzegovina, and the Ph.D. degree in nonlinear adaptive signal processing from the Imperial College London, London, U.K.

He is currently a Reader in signal processing with the Department of Electrical and Electronic Engineering, Imperial College London. His areas of interest include linear and nonlinear adaptive signal processing, system identification, blind source separation, and computer vision.

Dr. Mandic has received awards for his collaboration with industry and was also awarded the Nikola Tesla Medal for his innovative work.