A Complex-Valued RTRL Algorithm for Recurrent Neural Networks

Su Lee Goh
su.goh@imperial.ac.uk
Danilo P. Mandic
d.mandic@imperial.ac.uk
Department of Electrical and Electronic Engineering, Imperial College London, London SW7 2AZ, U.K.

A complex-valued real-time recurrent learning (CRTRL) algorithm for the class of nonlinear adaptive filters realized as fully connected recurrent neural networks is introduced. The proposed CRTRL is derived for a general complex activation function of a neuron, which makes it suitable for nonlinear adaptive filtering of complex-valued nonlinear and nonstationary signals and complex signals with strong component correlations. In addition, this algorithm is generic and represents a natural extension of the real-valued RTRL. Simulations on benchmark and real-world complex-valued signals support the approach.

1 Introduction

Recurrent neural networks (RNNs) are a widely used class of nonlinear adaptive filters with feedback, due to their ability to represent highly nonlinear dynamical systems (Elman, 1990; Tsoi & Back, 1994), attractor dynamics, and associative memories (Medsker & Jain, 2000; Principe, Euliano, & Lefebvre, 2000).[1] In principle, a static feedforward network can be transformed into a dynamic recurrent network by adding recurrent connections (Haykin, 1994; Puskorius & Feldkamp, 1994; Feldkamp & Puskorius, 1998). A very general class of networks that has both feedforward and feedback connections is the recurrent multilayer perceptron (RMLP) network (Puskorius & Feldkamp, 1994; Feldkamp & Puskorius, 1998), for which the representation capabilities have been shown to be considerably greater than those of static feedforward networks. Feedback neural networks have proven their potential in processing of nonlinear and nonstationary signals, with applications in signal modeling, system identification, time-series analysis, and prediction (Mandic & Chambers, 2001).

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

In 1989, Williams and Zipser proposed an on-line algorithm for training of RNNs, called the real-time recurrent learning (RTRL) algorithm (Williams & Zipser, 1989), which has since found a variety of signal processing applications (Haykin, 1994). This is a direct gradient algorithm, unlike the recurrent backpropagation used for the training of RMLPs. Despite its relative simplicity (as compared to recurrent backpropagation), it has been demonstrated that one of the difficulties of using direct gradient descent algorithms for training RNNs is the problem of vanishing gradient. Bengio, Simard, and Frasconi (1994) showed that the problem of vanishing gradients is the essential reason that gradient descent methods are often slow to converge. Several approaches have been proposed to circumvent this problem, which include both the new algorithms (such as extended Kalman filter, EKF) and new architectures, such as cascaded recurrent neural networks (Williams, 1992; Puskorius & Feldkamp, 1994; Haykin & Li, 1995).

In the fields of engineering and biomedicine, signals are typically nonlinear, nonstationary, and often complex valued. Properties of such signals vary not only in terms of their statistical nature but also in terms of their bivariate or complex-valued nature (Gautama, Mandic, & Hulle, 2003). As for the learning algorithms, in the area of linear filters (linear perceptron), the least mean square (LMS) algorithm was extended to the complex domain in 1975 (Widrow, McCool, & Ball, 1975). Subsequently, complex-valued backpropagation (CBP) was introduced by Leung and Haykin (1991) and Benvenuto and Piazza (1992). Thereby, a complex nonlinear activation function (AF) that separately processes the in-phase (I) and quadrature (Q) components of the weighted sum of input signals (net input) was employed. This way, the output from a complex AF takes two independent paths, which has since been referred to as the split-complex approach.[2] Although bounded, a split-complex AF cannot be analytic, and thus cannot cater to signals with strong coupling between the magnitude and phase (Kim & Adali, 2000).

In the 1990s, Georgiou and Koutsougeras (1992) and Hirose (1990) derived the CBP that uses nonlinear complex AFs that jointly process the I and Q components. However, these algorithms had difficulties when learning the nonlinear phase changes and the AF used in Georgiou and Koutsougeras (1992) was not analytic (Kim & Adali, 2000). In general, the split-complex approach has been shown to yield reasonable performance for some applications in channel equalization (Leung & Haykin, 1991; Benvenuto & Piazza, 1992; Kechriotis & Manolakos, 1994), and 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 quadra-

[2] In a split-complex AF, the real and imaginary components of the input signal x are separated and fed through the real-valued activation function $f_R(x) = f_I(x)$, $x \in \mathbb{R}$. A split-complex activation function is therefore given as $f(x) = f_R(Re(x)) + jf_I(Im(x))$, for example, $f(x) = \frac{1}{1+e^{-\beta(Re(x))}} + j\frac{1}{1+e^{-\beta(Im(x))}}$.

ture (Q) components are strongly correlated, algorithms employing split-complex activation function tend to yield poor performance (Gautama et al., 2003). Notice that split-complex algorithms cannot calculate the true gradient unless the real and imaginary weight updates are mutually independent.

Research toward a fully complex RNN has focused on the issue of analytical activation functions. A comprehensive account of elementary complex transcendental activation functions (ETFs) used as AFs is given in Kim and Adali (2002). They employed ETFs to derive a fully CBP algorithm using the Cauchy-Riemann[3] equations, which also helped to relax requirements on the properties of a fully complex activation function.[4]

For the case of RNNs, Kechriotis and Manalokas (1994) and Coelho (2001) introduced a complex-valued RTRL (CRTRL) algorithm. Both approaches, however, employed split complex AFs, thus restricting the domain of application. In addition, these algorithms did not follow the generic form of their real RTRL counterpart. There are other problems encountered with split-complex RTRL algorithms for nonlinear adaptive filtering: (1) the solutions are not general since split-complex AFs are not universal approximators (Leung & Haykin, 1991); (2) split-complex AFs are not analytic and hence the Cauchy-Riemann equations do not apply; (3) split-complex algorithms do not account for a "fully" complex nature of signal (Kim & Adali, 2003) and such algorithms therefore underperform in applications where complex signals exhibit strong component correlations (Mandic & Chambers, 2001); (4) these algorithms do not have a generic form of their real-valued counterparts, and hence their signal flow graphs are fundamentally different (Nerrand, Roussel-Ragot, Personnaz, & Dreyfus, 1993a).

Therefore, there is a need to address the problem of training a fully connected complex-valued recurrent neural network. Although there have been attempts to devise fully complex algorithms for online training of RNNs, a general fully complex CRTRL has been lacking to date. To this cause, we 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 complex-valued nonlinear and nonstationary signals. The fully connected recurrent neural network (FCRNN) used here is the canonical form of a feedback neural network (Nerrand, Roussel-Ragot, Personnaz, & Dreyfus, 1993b) that is general enough for the class of direct gradient-based CRTRL algorithms. The analysis is comprehensive and is supported by examples on benchmark complex-valued nonlinear and colored signals and together with simulations on real-world radar and environmental measurements.

---

[3] 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: $f'(z) = \frac{\partial u}{\partial x} + j\frac{\partial v}{\partial x} = \frac{\partial v}{\partial y} - j\frac{\partial u}{\partial y}$. This way $\frac{\partial u}{\partial x} = \frac{\partial v}{\partial y}$, $\frac{\partial v}{\partial x} = -\frac{\partial u}{\partial y}$.

[4] A fully complex activation function is analytic and bounded almost everywhere in $\mathbb{C}$.

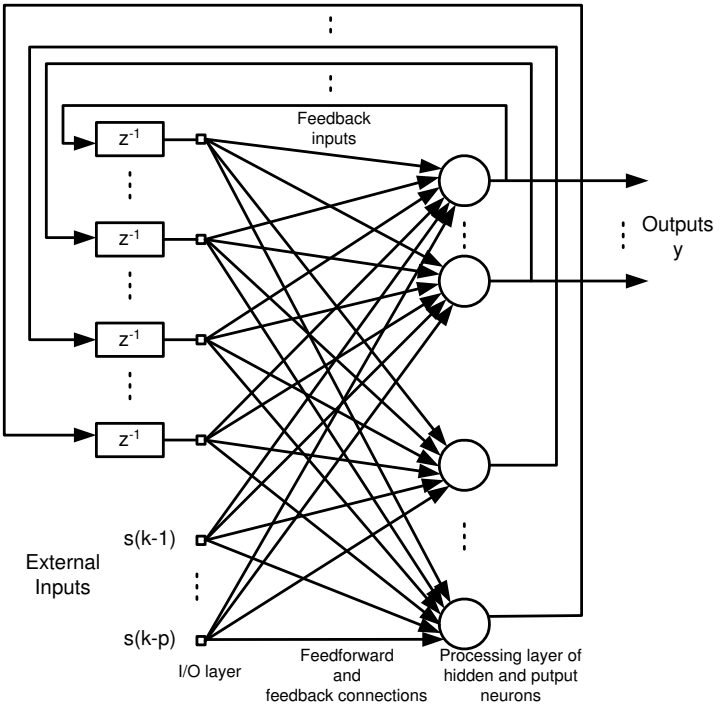

Figure 1: A fully connected recurrent neural network.

2 The Complex-Valued RTRL Algorithm

Figure 1 shows an FCRNN, which consists of N neurons with p external inputs. The network has two distinct layers consisting of the external input-feedback layer and a layer of processing elements. Let $y_l(k)$ denote the complex-valued output of each neuron, $l = 1, \ldots, N$ at time index k and $\mathbf{s}(k)$ the $(1 \times p)$ external complex-valued input vector. The overall input to the network $\mathbf{I}(k)$ represents the concatenation of vectors $\mathbf{y}(k)$, $\mathbf{s}(k)$ and the bias input $(1 + j)$, and is given by

$$\begin{aligned}\mathbf{I}(k) &= [s(k-1), \ldots, s(k-p), 1+j, y_1(k-1), \ldots, y_N(k-1)]^T \\ &= I_n^r(k) + jI_n^i(k), \quad n = 1, \ldots, p+N+1, \end{aligned} \tag{2.1}$$

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

A complex-valued weight matrix of the network is denoted by $\mathbf{W}$, where for the lth neuron, its weights form a $(p + F + 1) \times 1$ dimensional weight

vector $\mathbf{w}_l = [w_{l,1}, \ldots, w_{l,p+F+1}]^T$ where F is the number of feedback connections. The feedback connections represent the delayed output signals of the FCRNN. In the case of Figure 1, we have $F = N$.

The output of each neuron can be expressed as

$$y_l(k) = \Phi(net_l(k)), \quad l = 1, \ldots, N, \tag{2.2}$$

where

$$net_l(k) = \sum_{n=1}^{p+N+1} w_{l,n}(k) I_n(k) \tag{2.3}$$

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

$$y_l(k) = \Phi^r(net_l(k)) + j\Phi^i(net_l(k)) = u_l(k) + jv_l(k) \tag{2.4}$$

$$net_l(k) = \sigma_l(k) + j\tau_l(k) \tag{2.5}$$

where Φ is a complex nonlinear activation function.

3 The Derivation of the Complex-Valued RTRL

The output error consists of its real and imaginary parts and is defined as

$$e_l(k) = d(k) - y_l(k) = e_l^r(k) + je_l^i(k) \tag{3.1}$$

$$e_l^r(k) = d^r(k) - u_l(k), e_l^i(k) = d^i(k) - v_l(k), \tag{3.2}$$

where $d(k)$ is the teaching signal. For real-time applications the cost function of the recurrent network is given by (Widrow et al., 1975),

$$E(k) = \frac{1}{2}\sum_{l=1}^{N} |e_l(k)|^2 = \frac{1}{2}\sum_{l=1}^{N} e_l(k)e_l^*(k) = \frac{1}{2}\sum_{l=1}^{N}[(e_l^r)^2 + (e_l^i)^2], \tag{3.3}$$

where $(\cdot)^*$ denotes the complex conjugate. Notice that $E(k)$ is a real-valued function, and we are required to derive the gradient $E(k)$ with respect to both the real and imaginary part of the complex weights as

$$\nabla_{w_{s,t}} E(k) = \frac{\partial E(k)}{\partial w_{s,t}^r} + j\frac{\partial E(k)}{\partial w_{s,t}^i}, \ 1 \le l, \ s \le N, \ 1 \le t \le p + N + 1. \tag{3.4}$$

The CRTRL algorithm minimizes cost function $E(k)$ by recursively altering the weight coefficients based on gradient descent, given by

$$w_{s,t}(k+1) = w_{s,t}(k) + \Delta w_{s,t}(k) = w_{s,t}(k) - \eta \nabla_{w_{s,t}} E(k)_{|w_{s,t}=w_{s,t}(k)}, \tag{3.5}$$

where η is the learning rate, a small, positive constant. Calculating the gradient of the cost function with respect to the real part of the complex weight gives

$$\frac{\partial E(k)}{\partial w^r_{s,t}(k)} = \frac{\partial E}{\partial u_l}\left(\frac{\partial u_l(k)}{\partial w^r_{s,t}(k)}\right) + \frac{\partial E}{\partial v_l}\left(\frac{\partial v_l(k)}{\partial w^r_{s,t}(k)}\right), \quad \begin{matrix} 1 \le l,\ s \le N \\ 1 \le t \le p+N+1 \end{matrix}. \tag{3.6}$$

Similarly, the partial derivative of the cost function with respect to the imaginary part of the complex weight yields

$$\frac{\partial E(k)}{\partial w^i_{s,t}(k)} = \frac{\partial E}{\partial u_l}\left(\frac{\partial u_l(k)}{\partial w^i_{s,t}(k)}\right) + \frac{\partial E}{\partial v_l}\left(\frac{\partial v_l(k)}{\partial w^i_{s,t}(k)}\right), \quad \begin{matrix} 1 \le l,\ s \le N \\ 1 \le t \le p+N+1 \end{matrix}. \tag{3.7}$$

The factors $\frac{\partial y_l(k)}{\partial w^r_{s,t}(k)} = \frac{\partial u_l(k)}{\partial w^r_{s,t}(k)} + j\frac{\partial v_l(k)}{\partial w^r_{s,t}(k)}$ and $\frac{\partial y_l(k)}{\partial w^i_{s,t}(k)} = \frac{\partial u_l(k)}{\partial w^i_{s,t}(k)} + j\frac{\partial v_l(k)}{\partial w^i_{s,t}(k)}$ are measures of sensitivity of the output of the mth unit at time k to a small variation in the value of $w_{s,t}(k)$. These sensitivities can be evaluated as

$$\frac{\partial u_l(k)}{\partial w^r_{s,t}(k)} = \frac{\partial u_l}{\partial \sigma_l}\cdot\frac{\partial \sigma_l}{\partial w^r_{s,t}(k)} + \frac{\partial u_l}{\partial \tau_l}\cdot\frac{\partial \tau_l}{\partial w^r_{s,t}(k)} \tag{3.8}$$

$$\frac{\partial u_l(k)}{\partial w^i_{s,t}(k)} = \frac{\partial u_l}{\partial \sigma_l}\cdot\frac{\partial \sigma_l}{\partial w^i_{s,t}(k)} + \frac{\partial u_l}{\partial \tau_l}\cdot\frac{\partial \tau_l}{\partial w^i_{s,t}(k)} \tag{3.9}$$

$$\frac{\partial v_l(k)}{\partial w^r_{s,t}(k)} = \frac{\partial v_l}{\partial \sigma_l}\cdot\frac{\partial \sigma_l}{\partial w^r_{s,t}(k)} + \frac{\partial v_l}{\partial \tau_l}\cdot\frac{\partial \tau_l}{\partial w^r_{s,t}(k)} \tag{3.10}$$

$$\frac{\partial v_l(k)}{\partial w^i_{s,t}(k)} = \frac{\partial v_l}{\partial \sigma_l}\cdot\frac{\partial \sigma_l}{\partial w^i_{s,t}(k)} + \frac{\partial v_l}{\partial \tau_l}\cdot\frac{\partial \tau_l}{\partial w^i_{s,t}(k)} \tag{3.11}$$

Following the derivation of the real-valued RTRL (Williams & Zipser, 1989), to compute these sensitivities, start with differentiating equation 2.3 which yields

$$\frac{\partial \sigma_l(k)}{\partial w^r_{s,t}(k)} = \left[\sum_{q=1}^{N}\left(\frac{\partial u_q(k-1)}{\partial w^r_{s,t}(k)} w^r_{l,p+1+q}(k) - \frac{\partial v_q(k-1)}{\partial w^r_{s,t}(k)} w^i_{l,p+1+q}(k)\right)\right] + \delta_{sl} I^r_n(k)$$

$$\frac{\partial \tau_l(k)}{\partial w^r_{s,t}(k)} = \left[\sum_{q=1}^{N}\left(\frac{\partial v_q(k-1)}{\partial w^r_{s,t}(k)} w^r_{l,p+1+q}(k) + \frac{\partial u_q(k-1)}{\partial w^r_{s,t}(k)} w^i_{l,p+1+q}(k)\right)\right] + \delta_{sl} I^i_n(k)$$

$$\frac{\partial \sigma_l(k)}{\partial w^i_{s,t}(k)} = \left[\sum_{q=1}^{N}\left(\frac{\partial u_q(k-1)}{\partial w^i_{s,t}(k)} w^r_{l,p+1+q}(k) - \frac{\partial v_q(k-1)}{\partial w^i_{s,t}(k)} w^i_{l,p+1+q}(k)\right)\right] - \delta_{sl} I^i_n(k)$$

$$\frac{\partial \tau_l(k)}{\partial w^i_{s,t}(k)} = \left[\sum_{q=1}^{N}\left(\frac{\partial v_q(k-1)}{\partial w^i_{s,t}(k)} w^r_{l,p+1+q}(k) + \frac{\partial u_q(k-1)}{\partial w^i_{s,t}(k)} w^i_{l,p+1+q}(k)\right)\right] + \delta_{sl} I^r_n(k),$$

where

$$\delta_{sl} = \begin{cases} 1, & l = s \\ 0, & l \neq s \end{cases} \tag{3.12}$$

is the Kronecker delta. For a complex function to be analytic at a point in $\mathbb{C}$, it needs to satisfy the Cauchy-Riemann equations. To arrive at the Cauchy-Riemann equations, the partial derivatives (sensitivities) along the real and imaginary axes should be equal, that is,

$$\frac{\partial u_l(k)}{\partial w^r_{s,t}(k)} + j\frac{\partial v_l(k)}{\partial w^r_{s,t}(k)} = \frac{\partial v_l(k)}{\partial w^i_{s,t}(k)} - j\frac{\partial u_l(k)}{\partial w^i_{s,t}(k)}. \tag{3.13}$$

Equating the real and imaginary parts in equation 3.13, we obtain

$$\frac{\partial u_l(k)}{\partial w^r_{s,t}(k)} = \frac{\partial v_l(k)}{\partial w^i_{s,t}(k)} \tag{3.14}$$

$$\frac{\partial u_l(k)}{\partial w^i_{s,t}(k)} = -\frac{\partial v_l(k)}{\partial w^r_{s,t}(k)}. \tag{3.15}$$

For convenience, we denote the sensitivities as $\pi^{l,(rr)}_{s,t}(k) = \frac{\partial u_l(k)}{\partial w^r_{s,t}(k)}$, $\pi^{l,(ir)}_{s,t}(k) = \frac{\partial v_l(k)}{\partial w^r_{s,t}(k)}$, $\pi^{l,(ri)}_{s,t}(k) = \frac{\partial u_l(k)}{\partial w^i_{s,t}(k)}$ and $\pi^{l,(ii)}_{s,t}(k) = \frac{\partial v_l(k)}{\partial w^i_{s,t}(k)}$. By using the Cauchy-Riemann equations, a more compact representation of gradient $\nabla_{w_{s,t}} E(k)$ is given by

$$\begin{aligned}
\nabla_{w_{s,t}} E(k) &= \pi^{l,(rr)}_{s,t}(k)\frac{\partial E(k)}{\partial u_l(k)} + \pi^{l,(ir)}_{s,t}(k)\frac{\partial E(k)}{\partial v_l(k)} + j\pi^{l,(ri)}_{s,t}(k)\frac{\partial E(k)}{\partial u_l(k)} \\
&\quad + j\pi^{l,(ii)}_{s,t}(k)\frac{\partial E(k)}{\partial v_l(k)} \\
&= \left(\frac{\partial E(k)}{\partial u_l(k)} + j\frac{\partial E(k)}{\partial v_l(k)}\right)\left(\pi^{l,(rr)}_{s,t}(k) + j\pi^{l,(ri)}_{s,t}(k)\right) \\
&= \sum_{l=1}^{N} e_l(k)\left(\pi^{l,(rr)}_{s,t}(k) - j\pi^{l,(ir)}_{s,t}(k)\right) \\
&= \sum_{l=1}^{N} e_l(k)\left(\pi^{l}_{s,t}\right)^*(k).
\end{aligned} \tag{3.16}$$

The weight update is finally given by

$$\Delta w_{s,t}(k) = \eta \sum_{l=1}^{N} e_l(k)(\pi^l_{s,t})^*(k), \quad 1 \le l,\ s \le N,\ 1 \le t \le p+N+1, \tag{3.17}$$

with the initial condition

$$(\pi_{s,t}^{l})^{*}(0) = 0. \tag{3.18}$$

Under the assumption, also used in the RTRL algorithm (Williams & Zipser, 1989), that for a sufficiently small learning rate η, we have

$$\frac{\partial u_l(k-1)}{\partial w_{s,t}(k)} \approx \frac{\partial u_l(k-1)}{\partial w_{s,t}(k-1)}, \quad 1 \le l,\ s \le N,\ 1 \le t \le p+N+1 \tag{3.19}$$

$$\frac{\partial v_l(k-l)}{\partial w_{s,t}(k)} \approx \frac{\partial v_l(k-l)}{\partial w_{s,t}(k-l)}, \quad 1 \le l,\ s \le N,\ 1 \le t \le p+N+1, \tag{3.20}$$

the update for the sensitivities $(\pi_{s,t}^{l})^{*}(k)$ becomes

$$\begin{aligned}
(\pi_{s,t}^{l})^{*}(k) &= \frac{\partial u_l}{\partial \sigma_l}\Bigg[\delta_{sl} I_n^r(k) - j\delta_{sl} I_n^i(k) + \sum_{q=1}^{N}\Big[\Big(w_{l,p+1+q}^{r}(k) - jw_{l,p+1+q}^{i}(k)\Big) \\
&\qquad \pi_{s,t}^{q,(rr)}(k-1) + \Big(jw_{l,p+1+q}^{r}(k) + w_{l,p+1+q}^{i}(k)\Big)\pi_{s,t}^{q,(ri)}(k-1)\Big]\Bigg] \\
&\quad + \frac{\partial u_l}{\partial \tau_l}\Bigg[\delta_{sl} I_n^i(k) + j\delta_{sl} I_n^r(k) + \sum_{q=1}^{N}\Big[\Big(-w_{l,p+1+q}^{r}(k) + jw_{l,p+1+q}^{i}(k)\Big) \\
&\qquad \pi_{s,t}^{q,(ri)}(k-1) + \Big(w_{l,p+1+q}^{i}(k) + jw_{l,p+1+q}^{r}(k)\Big)\pi_{s,t}^{q,(rr)}(k-1)\Big]\Bigg] \\
&= \frac{\partial u_l}{\partial \sigma_l}\Bigg[\delta_{sl} I_n^*(k) + \sum_{q=1}^{N}\Big[w_{l,p+1+q}^{*}(k)\pi_{s,t}^{q,(rr)}(k-1) \ + \ jw_{l,p+1+q}^{*}(k)\pi_{s,t}^{q,(ri)}(k-1)\Big]\Bigg] \\
&\quad + \frac{\partial u_l}{\partial \tau_l}\Bigg[j\delta_{sl} I_n^*(k) + \sum_{q=1}^{N}\Big[-w_{l,p+1+q}^{*}(k)\pi_{s,t}^{q,(ri)}(k-1) \\
&\qquad + \ jw_{l,p+1+q}^{*}(k)\pi_{s,t}^{q,(rr)}(k-1)\Big]\Bigg] \\
&= \left(\frac{\partial u_l}{\partial \sigma_l} + j\frac{\partial u_l}{\partial \tau_l}\right)\Bigg[\delta_{sl} I_n^*(k) + \sum_{q=1}^{N} w_{l,p+1+q}^{*}(k)\left(\pi_{s,t}^{q,(rr)}(k-1) + j\pi_{s,t}^{q,(ri)}(k-1)\right)\Bigg] \\
&= \{\Phi'(k)\}^{*}\Bigg[\mathbf{w}_l^{H}(k)\boldsymbol{\pi}^{*}(k-1) + \delta_{sl} I_n^*(k)\Bigg],
\end{aligned} \tag{3.21}$$

where $(\cdot)^H$ denotes the Hermitian transpose operator. This completes the derivation of the fully complex RTRL (CRTRL).

3.1 Computational Requirements. The memory requirements and computational complexity of complex-valued neural networks differ from those of real-valued neural networks. A complex addition is roughly equivalent of two real additions, whereas a complex multiplication can be represented as four real multiplications and two additions or three multiplications and five additions.[5] The CRTRL computes sensitivities $(\pi^{l}_{s,t})^{*}(k) = \pi^{l,(ri)}_{s,t}(k) - j\pi^{l,(ir)}_{s,t}(k)$ at every iteration with respect to all the elements of the complex weight matrix, where every column corresponds to a neuron in the network. For a real-valued FCRNN, there are N^2 weights, N^3 gradients, and $\mathcal{O}(N^4)$ operations required for the gradient computation per sample. When the network is fully connected and all weights are adaptable, this algorithm has space complexity of $\mathcal{O}(N^3)$ (Williams & Zipser, 1989). Thus, the space complexity required by the complex-valued FCRNN becomes twice that of the real-valued case, whereas the computational complexity becomes approximately four times that of the real-valued RTRL.

4 Simulations

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

$$\Phi(x) = \frac{1}{1 + e^{-\beta x}}, \tag{4.1}$$

where x is complex valued. The slope was chosen to be $\beta = 1$ and learning rate $\eta = 0.01$. The architecture of the FCRNN (see Figure 1) consists of $N = 2$ neurons with the tap input length of $p = 4$.

Simulations were undertaken by averaging 100 iterations of independent trials on prediction of complex-valued signals. To support the analysis, we tested the CRTRL on a wide range of signals, including the complex linear, complex nonlinear, and chaotic (Ikeda map). To further illustrate the approach and verify the advantage of using the fully CRTRL (FCRTRL) over split CRTRL (SCRTRL), single-trial experiment were performed on real-world complex wind[6] and radar[7] data.

In the first experiment, the input signal was a complex linear $AR(4)$ process given by

$$r(k) = 1.79r(k-1) - 1.85r(k-2) + 1.27r(k-3) + 0.41r(k-4) + n(k), \tag{4.2}$$

[5] There are two ways to perform a complex multiplication: for the first case, $(a+ib)(c+id) = (ac-bd) + i(bc+ad)$, and for the second case, $(a+ib)(c+id) = a(c+d) - d(a+b) + i(a(c+d) + c(b-a))$ (publicly available online from http://mathworld.wolfram.com/). The latter form may be preferred when multiplication takes much more time than addition but can be less numerically stable.

[6] Publicly available online from http://mesonet.agron.iastate.edu/.

[7] Publicly available online from http://soma.ece.mcmaster.ca/ipix/.

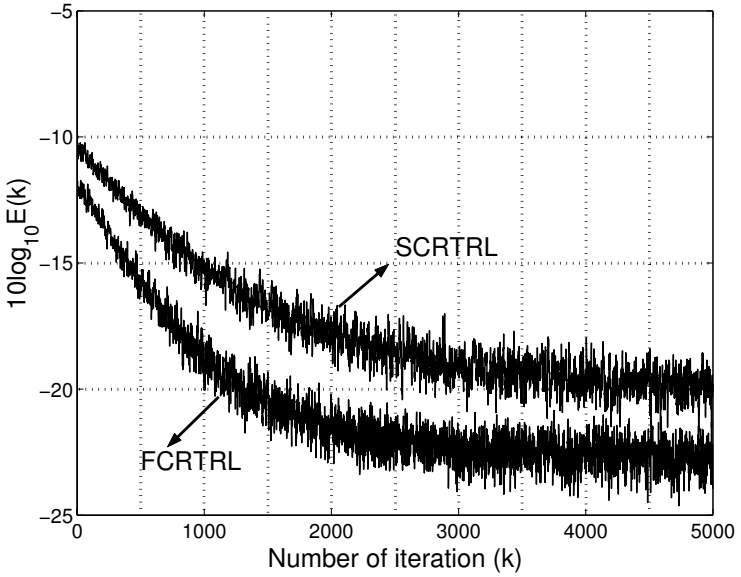

Figure 2: Performance curves for FCRTRL and SCRTRL on prediction of complex colored input, equation 4.2.

with complex white gaussian noise (CWGN) $n(k) \sim \mathcal{N}(0, 1)$ as the driving input. The CWGN can be expressed as $n(k) = n^r(k) + jn^i(k)$. The real and imaginary components of CWGN are mutually independent sequences having equal variances so that $\sigma_n^2 = \sigma_{n^r}^2 + \sigma_{n^i}^2$. For the second experiment, the complex benchmark nonlinear input signal was (Narendra & Parthasarathy, 1990):

$$z(k) = \frac{z(k-1)}{1 + z^2(k-1)} + r^3(k). \tag{4.3}$$

Figures 2 and 3 show, respectively, the performance curves for the FCRTRL and SCRTRL on complex colored (see equation 4.2) and nonlinear (see equation 4.3) signal. The proposed approach showed improved performance for both the complex colored and nonlinear input. The performance improvement was roughly 3 dB for the linear and 6 dB for the nonlinear signal. In addition, the proposed CRTRL exhibited faster convergence.

The simulations results for the Ikeda map are shown in Figure 4. Observe that the FCRTRL algorithm was more stable and has exhibited improved and more consistent performance over that of the SCRTRL algorithm.

Figure 5 shows the prediction performance of the FCRNN applied to the complex-valued radar signal in both the split and fully CRTRL case. The FCRNN was able to track the complex radar signal, which was not the case

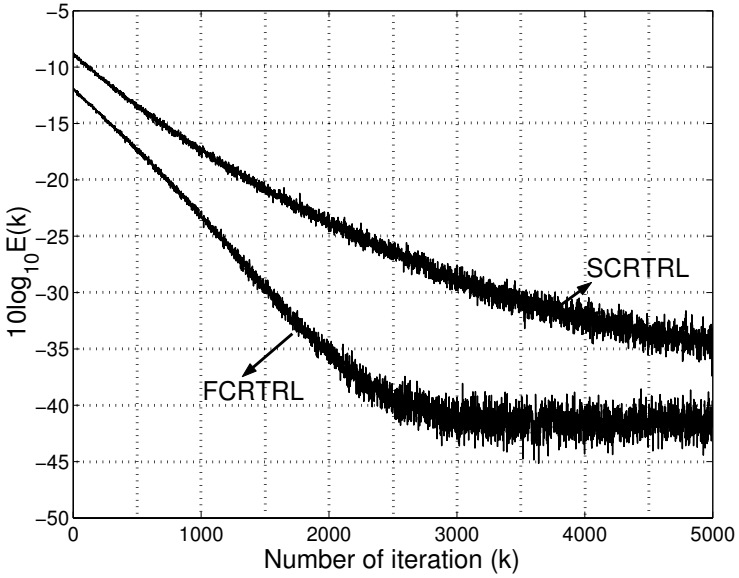

Figure 3: Performance curves for FCRTRL and SCRTRL on prediction of complex nonlinear input, equation 4.3.

for split CRTRL. For the next experiment, we compared the performance of FCRTRL and SCRTRL on wind data. Figures 6 and 7 show the results obtained for FCRTRL and SCRTRL. The FCRTRL achieved much improved prediction performance over the SCRTRL.

5 Conclusions

A fully complex real-time recurrent learning (CRTRL) algorithm for online training of fully connected recurrent neural networks (FCRNNs) has been proposed. The FCRTRL algorithm provides an essential tool for nonlinear adaptive filtering using FCRNNs in the complex domain and is derived for a general complex nonlinear activation function of a neuron. We have made use of the Cauchy-Riemann equations to obtain a generic form of the proposed algorithm. The performance of the CRTRL algorithm has been evaluated on benchmark complex-valued nonlinear and colored input signals, Ikeda map, and also on real-life complex-valued wind and radar signals. Experimental results have justified the potential of the CRTRL algorithm in nonlinear neural adaptive filtering applications. Expanding fully complex RNN into modular RNNs and developing extended Kalman filter techniques to enhance the estimation capability remain among the most important further challenges in the area of fully complex RNNs.

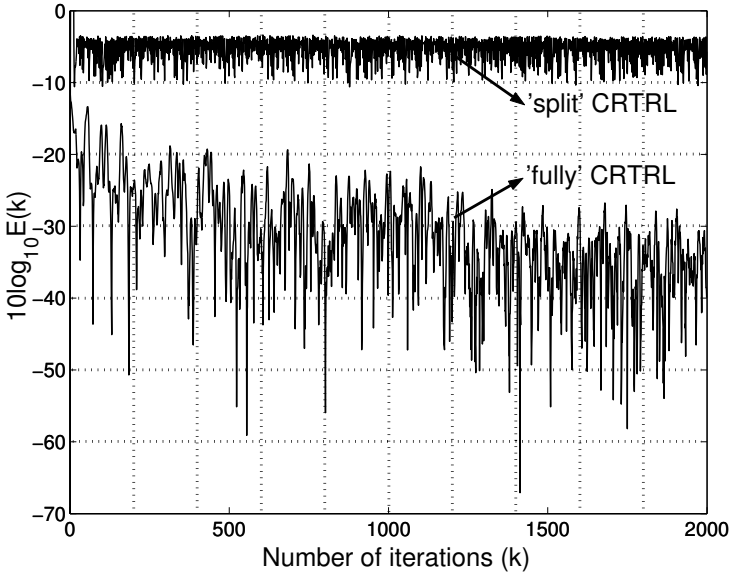

Figure 4: Performance curve of FCRTRL and SCRTRL for Ikeda map.

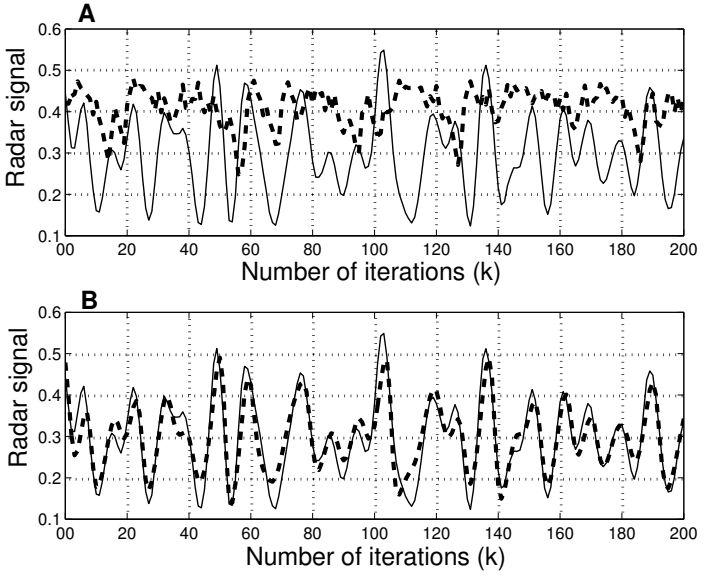

Figure 5: Prediction of complex radar signal based on fully complex and split complex activation function. (A) Split-complex CRTRL. (B) Fully complex CRTRL. Solid curve: actual radar signal. Dashed curve: nonlinear prediction.

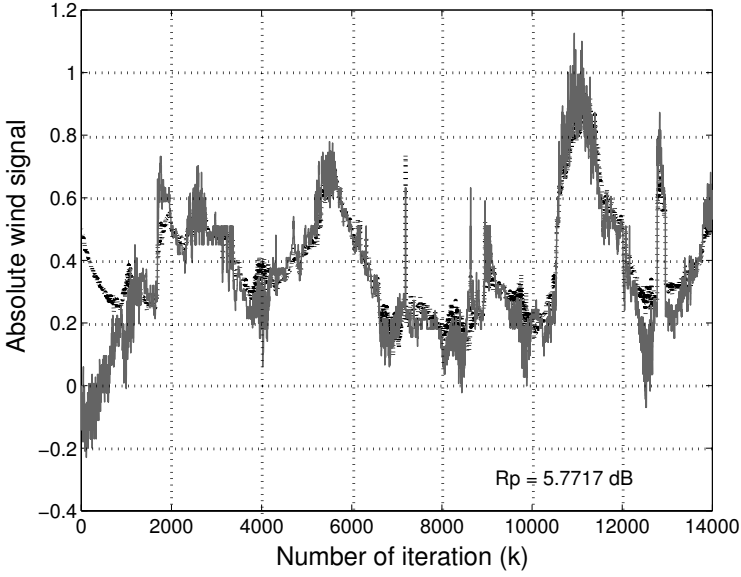

Figure 6: Prediction of complex wind signal using FCRTRL. Solid curve: Actual wind signal. Dotted curve: Predicted signal.

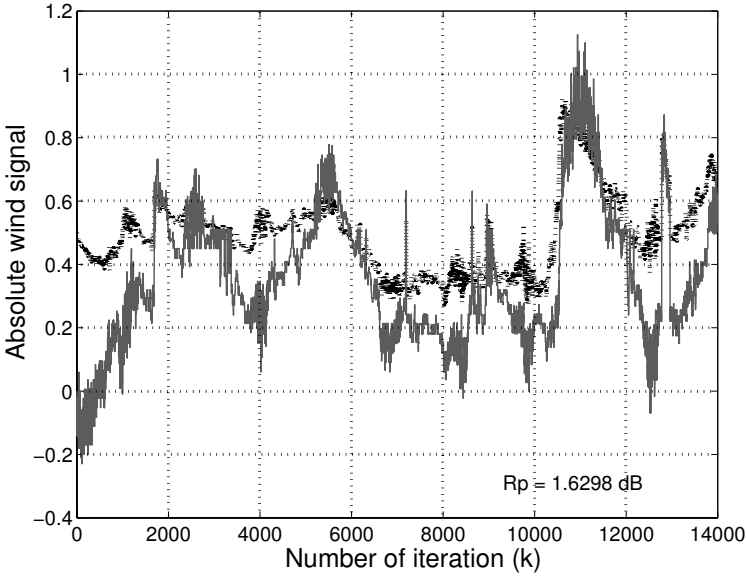

Figure 7: Prediction of complex wind signal using SCRTRL. Solid curve: Actual wind signal. Dotted curve: Predicted signal.

References

Bengio, Y., Simard, P., & Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks, 5*(2), 157–166.

Benvenuto, N., & Piazza, F. (1992). On the complex backpropagation algorithm. *IEEE Transactions on Signal Processing, 40*(4), 967–969.

Coelho, P. H. G. (2001). A complex EKF-RTRL neural network. *International Joint Conference on Neural Networks, 1*, 120–125.

Elman, J. L. (1990). Finding structure in time. *Cognitive Science, 14*, 179–211.

Feldkamp, L. A., & Puskorius, G. V. (1998). A signal processing framework based on dynamical neural networks with application to problems in adaptation, filtering and classification. *IEEE Transactions on Neural Networks, 86*(4), 2259–2277.

Gautama, T., Mandic, D. P., & Hulle, M. M. V. (2003). A non-parametric test for detecting the complex-valued nature of time series. In *Knowledge-Based Intelligent Information and Engineering Systems: 7th International Conference* (pp. 1364–1371). Berlin: Springer-Verlag.

Georgiou, G. M., & Koutsougeras, C. (1992). Complex domain backpropagation. *IEEE Transactions on Circuits and Systems—II: Analog and Digital Signal Processing, 39*(5), 330–334.

Haykin, S. (1994). *Neural networks: A comprehensive foundation*. Englewood Cliffs, NJ: Prentice Hall.

Haykin, S., & Li, L. (1995). Nonlinear adaptive prediction of nonstationary signals. *IEEE Transactions on Signal Processing, 43*(2), 526–535.

Hirose, A. (1990). Continuous complex-valued backpropagation learning. *Electronics Letters, 28*(20), 1854–1855.

Kechriotis, G., & Manolakos, E. S. (1994). Training fully recurrent neural networks with complex weights. *IEEE Transactions on Circuits and Systems—II: Analog and Digital Signal Processing, 41*(3), 235–238.

Kim, T., & Adali, T. (2000). Fully complex backpropagation for constant envelope signal processing. In *Neural Networks for Signal Processing X: Proceedings of the 2000 IEEE Signal Processing Society Workshop* (Vol. 1, pp. 231–240). Piscataway, NJ: IEEE.

Kim, T., & Adali, T. (2002). Universal approximation of fully complex feedforward neural networks. In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing, ICASSP* (Vol. 1, pp. 973–976). Piscataway, NJ: IEEE.

Kim, T., & Adali, T. (2003). Approximation by fully complex multilayer perceptrons. *Neural Computation, 15*(7), 1641–1666.

Leung, H., & Haykin, S. (1991). The complex backpropagation algorithm. *IEEE Transactions on Signal Processing, 3*(9), 2101–2104.

Mandic, D. P., & Chambers, J. A. (2001). *Recurrent neural networks for prediction: Learning algorithms, architectures and stability*. New York: Wiley.

Medsker, L. R., & Jain, L. C. (2000). *Recurrent neural networks: Design and applications*. Boca Raton, FL: CRC Press.

Narendra, K. S., & Parthasarathy, K. (1990). Identification and Control of Dynamical Systems Using Neural Networks. *IEEE Transactions on Neural Networks, 1*(1), 4–27.

Nerrand, O., Roussel-Ragot, P., Personnaz, L., & Dreyfus, G. (1993a). Neural networks and non-linear adaptive filtering: Unifying concepts and new algorithms. *Neural Computation, 5*, 165–199.

Nerrand, O., Roussel-Ragot, P., Personnaz, L., & Dreyfus, G. (1993b). Neural networks and non-linear adaptive filtering: Unifying concepts and new algorithms. *Neural Computation, 3*, 165–197.

Principe, J. C., Euliano, N. R., & Lefebvre, W. C. (2000). *Neural and adaptive systems: Fundamentals through simulations*. New York: Wiley.

Puskorius, G. V., & Feldkamp, L. A. (1994). Neurocontrol of Nonlinear Dynamical Systems with Kalman Filter Trained Recurrent Networks. *IEEE Transactions on Neural Networks, 5*(2), 279–297.

Tsoi, A. C., & Back, A. D. (1994). Locally recurrent globally feedforward networks: A critical review of architectures. *IEEE Transactions on Neural Networks, 5*(2), 229–239.

Widrow, B., McCool, J., & Ball, M. (1975). The complex LMS algorithm. *Proceedings of the IEEE, 63*, 712–720.

Williams, R. J. (1992). Training recurrent networks using the extended Kalman filter. In *Proceedings of the IJCNN'92 Baltimore, 4*, 241–246.

Williams, R. J., & Zipser, D. A. (1989). A learning algorithm for continually running fully recurrent neural networks. *Neural Computation, 1*(2), 270–280.

Received June 17, 2003; accepted May 5, 2004.