Adaptive Decision Feedback Equalization for Digital Satellite Channels Using Multilayer Neural Networks

Po-Rong Chang, *Member, IEEE,* and Bor-Chin Wang

Abstract—**This paper introduces an adaptive decision feedback equalization using the multilayer perceptron structure of an M-ary PSK signal through a TDMA satellite radio channel. The transmission is disturbed not only by intersymbol interference (ISI) and additive white Gaussian noise, but also by the nonlinearity of transmitter amplifiers. The conventional decision feedback equalizer (DFE) is not well-suited to detect the transmitted sequence, whereas the neural-based DFE is able to take into account the nonlinearities and therefore to detect the signal much better. Nevertheless, the applications of the traditional multilayer neural networks have been limited to real-valued signals. To overcome this difficulty, a neural-based DFE is proposed to deal with the complex PSK signal over the complex-valued nonlinear MPSK satellite channel without performing time-consuming complex-valued back-propagation training algorithms, while maintaining almost the same computational complexity as the original real-valued training algorithm. Moreover, a modified back-propagation algorithm with better convergence properties is derived on the basis of delta-bar-delta rule. Simulation results for the equalization of QPSK satellite channels show that the neural-based DFE provides a superior bit error rate performance relative to the conventional mean square DFE, especially in poor signal-to-noise ratio conditions.**

I. Introduction

An integrated digital land-mobile satellite system envisions the use of satellites to complement existing or planned terrestrial systems to provide mobile communications to thinly populated and/or large geographical areas. Digital satellite communication systems are frequently operated over nonlinear channels with memory. In fact, the satellite communication links are equipped with traveling wave tube (TWT) amplifiers at or near saturation for better efficiency. The TWT exhibits nonlinear distortion in both amplitude and phase conversions. In addition, at high transmission rates, the finite bandwidth of the channel causes a form of distortion known as intersymbol interference (ISI). In this paper, we will examine the problem of equalizing this type of nonlinear satellite communication link, where observed data are assumed to be corrupted by additive Gaussian white noise.

For digital satellite communication, it has been recognized that PSK/TDMA provides highly efficient use of power and bandwidth through the sharing of single transponder by several mobile units or earth stations accessing it. Satellite communication systems that use nonlinear TWT amplifiers require constant envelope signaling such as M-ary phase shift keying (MPSK). On the contrary, [4] showed the quadrature amplitude modulation (QAM) would not be feasible to do so on the TWT satellite link. References [1] and [2] showed that the coherent MPSK satellite channel can be characterized using a Volterra series expansion. Various techniques such as nonlinear Volterra-series-based transversal equalizer [2] and decision feedback equalizer [3] were developed to eliminate the undesired nonlinear distortion. Meanwhile, the Volterra series representation is too cumbersome and impractical for use in achieving the real-time satellite equalization involving a large number of computations. An alternative approach to nonlinear channel equalization is based on the multilayer perception (MLP).

Artificial neural networks are systems which use nonlinear computational elements to model the neural behavior of the biological nervous systems. The properties of neural networks include: massive parallelism, high computation rates, great capability for nonlinear problems, and ease for VLSI implementation, etc. All these properties make neural network attractive for various applications, such as image processing, pattern recognition, and digital signal processing, and neural networks have provided good solutions for these areas.

Recently, Chen *et al.* [6], and Siu *et al.* [5], have effectively utilized MLP neural networks as adaptive equalizers for several nonlinear channel models with additive colored noise. They demonstrated that the neural-based equalizer trained by the back-propagation algorithms showed superior performance over conventional decision feedback equalizer because of its capability to form complex decision regions with nonlinear boundaries. Nevertheless, their applications have been limited to real-valued baseband channel models and binary signals. However, for MPSK satellite communication, the channel models and the information bearing signals are complex-valued. So, there is a great need to develop a neural network equalizer that can deal with higher level signal constellations, such as M-ary PSK, as well as with complex-valued channel models. Chang and Chang [17] applied the method of splitting to separately treat the real and imaginary neuron inputs and outputs of the neural network equalizer over a complex-value QAM indoor radio fading channels in order to avoid complex-valued operations and to yield the better bit error rate performance. Here, we use the same concept to equalize QPSK satellite channels. Section II gives a brief description of channel modeling based on the Volterra series representation of nonlinear systems with memory. In Section III, we proposed a new neural-based decision-feedback equalizer to PSK systems

Manuscript received January 15, 1994; revised September 23, 1994. This work was supported in part by the National Science Council, Taiwan, R.O.C., under Contract NSC 84-2221-E009-016.

The authors are with the Department of Communication Engineering, National Chiao-Tung University, Hsin-Chu, Taiwan.

IEEE Log Number 9407497.

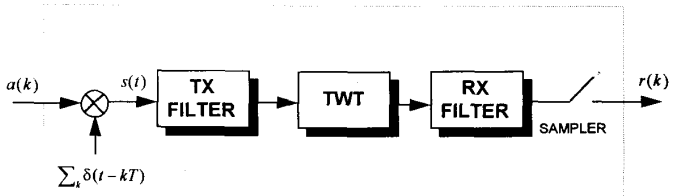

Fig. 1. A discrete-input discrete-output model of a nonlinear satellite channel.

over complex-valued channel without performing complex-valued back-propagation algorithms. A packet-wise neural equalization is introduced to track channel time variations and improve the system performance. Furthermore, it will be proven that the neural equalizer trained by packet-wise back-propagation algorithm approaches an ideal equalizer after receiving a sufficient number of packets. In Section IV, an algorithm based on a delta-bar-delta rule is proposed to improve the computational efficiency of the back propagation and provide faster network training of the neural equalizers. Computer simulations are presented in Section V.

II. Channel Modeling for Digital Satellite Links

Fig. 1 shows the block diagram of the baseband-equivalent model for a nonlinear satellite channel. Let its input be the sequence of M-ary information symbols $\{a_k\}$. Thus, the modulator output has the complex envelope

$$s(t) = \sum_{k=-\infty}^{\infty} a_k p(t - kT) \tag{1}$$

where T is the sampling period and $p(t)$ is the basic modulator waveform (typically, a rectangular with a T second duration). For an M-ary PSK, $a_k^M = 1$.

Although the traveling-wave tube (TWT) amplifier is modeled as a memoryless nonlinearity, it embedment between linear transmission (TX) and receiving (RX) filters leads to a nonlinear system with memory as a model for the overall channel. The TX filter includes the shaping filter and four-pole Butterworth filter with 3 dB bandwidth $1.7/T$. The RX filter is a two-pole Butterworth filter with 3 dB bandwidth $1.1/T$. A convenient tool to represent the overall channel is the Volterra series. Reference [2] showed that the symbol-rate sampling of receiving output can be described as

$$r(k) = \sum_{i=1}^{\infty} \sum_{k_1} \cdots \sum_{k_{2i-1}} a_{k-k_1} a_{k-1_2} \cdots a_{k-k_i} \cdot a^*_{k-k_{i+1}} \cdots a^*_{k-k_{2i-1}} H^{(2i-1)}_{k_i, \cdots, k_{2i-1}} + \nu(k) \tag{2}$$

where $\nu(n)$ is a complex Gaussian down-link noise, and the Volterra coefficients of the discrete-input discrete-output channel $H^{(1)}_k$, $H^{(3)}_{k_1, k_2, k_3}, \cdots$, are a set of complex numbers which describe the effect of the nonlinear channel on the symbol sequence $\{a_k\}$. The term with index $i = 1$ represents the linear part of the channel, i.e., it comprises the useful signal and linear ISI. The terms with index $i = 2$ corresponds to third-order distortion, and the terms with index $i = 3$ to fifth-order distortion, and so forth.

TABLE I
Reduced Volterra Coefficients

LINEAR PART
$H^{(1)}_0 = 1.22 + j0.646$
$H^{(1)}_1 = 0.063 - j0.001$
$H^{(1)}_2 = -0.024 - j0.014$
$H^{(1)}_3 = 0.036 + j0.031$
3RD ORDER NONLINEARITIES
$H^{(3)}_{002} = 0.039 - j0.022$
$H^{(3)}_{330} = 0.018 - j0.018$
$H^{(3)}_{001} = 0.035 - j0.035$
$H^{(3)}_{003} = -0.040 - j0.009$
$H^{(3)}_{110} = -0.01 - j0.017$
5TH ORDER NONLINEARITIES
$H^{(5)}_{00011} = 0.039 - j0.022$

Reference [1] showed that the symbol structure of PSK modulation results in insensitivity to certain kinds of nonlinearities. The Volterra coefficients $H^{(2i-1)}$ will induce nonlinearity of order less than $2i - 1$ when $a_k a_k^* = 1$, where $a_k \triangleq e^{j2\pi(k-1)/M}$, $k = 1, 2, \cdots, M$. For example, the channel nonlinearities reflected by the Volterra coefficients $H^{(3)}_{k_1 k_2 k_3}$ will not affect a PSK signal when $k_1 = k_3$ or $k_2 = k_3$. From [1] and [2], the computed Volterra coefficients for this channel, after reduction and deletion of the smallest, are shown in Table I. The thresholds used for the magnitude of Volterra coefficients are equal to 0.001 and 0.005 for the linear and nonlinear parts, respectively.

III. Neural-Based Decision Feedback Equalization for Nonlinear QPSK Channels

A feedforward neural network (shown in Fig. 2) is a layered network consisting of an input layer, an output layer, and at least one hidden layer of nonlinear processing elements. The nonlinear processing elements, which sum incoming signals and generate output signals according to some predefined function, are called neurons. In this paper, the function used by nonlinear neurons is called the sigmoidal function G defined by

$$G(x) = (1 - e^{-x})/(1 + e^{-x}) \tag{3}$$

where $G(x)$ lies in the interval $[-1, 1]$. The neurons are connected by terms with variable weights. The output of one neuron multiplied by a weight becomes the input of an adjacent neuron of the next layer.

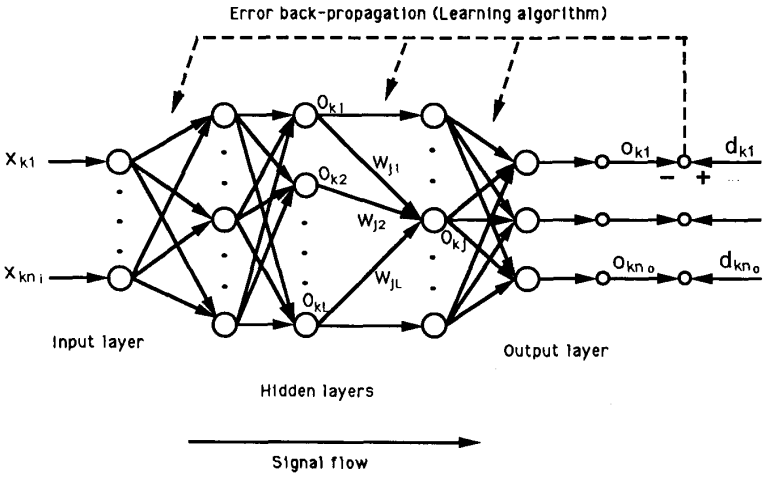

Fig. 2. Multilayer feedforward neural network.

$\boldsymbol{x}_k = [x_{k1}, x_{k2}, \cdots, x_{k,n_i}]^T$ denotes the input pattern at time instant k. $\boldsymbol{d}_k = [d_{k1}, d_{k2}, \cdots, d_{k,n_0}]^T$ is the desired output vector. The neuron j in layer m receives L_{m-1} inputs $o_{k1}^{(m-1)}, o_{k2}^{(m-1)}, \cdots, o_{kL_{m-1}}^{(m-1)}$ at time instant k, multiplies them by a set of weights $w_{j1}^{(m)}, w_{j2}^{(m)}, \cdots, w_{jL_{m-1}}^{(m)}$ and sums the resultant values. To this sum, a real threshold level $I_j^{(m)}$ is added. The output $o_{kj}^{(m)}$ of the neuron j at time instant k is given by evaluating the sigmoidal activation function

$$o_{kj}^{(m)} = G\left(\sum_{i=1}^{L_{m-1}} w_{ji}^{(m)} o_i^{(m-1)} + I_j^m\right). \tag{4}$$

The output value $o_{kj}^{(m)}$ serves as input to the $(m+1)$th layer to which the neuron is connected. Furthermore, an iterative learning algorithm, called back propagation, was suggested by Rumelhart *et al.* [12]. In back propagation, the output value of the output layer is compared to the desired output $\boldsymbol{d}_k$, resulting in error signal. The error signal is fed back through the network and weights are adjusted to minimize the error. For the simplicity of evaluating the backpropagation algorithm, the threshold level $I_j^{(m)}$ can be considered as a weight value from an additional virtual neuron with $o_{L_{m-1}^{(m-1)}} = 1$ to the jth neuron in the mth layer. More details of the back propagation will be discussed in Section IV.

Unfortunately, Birx and Pipenberg [19] showed that the continuity of the sigmoidal activation function of (3) is not valid on any complex domain that contains the point $e^{-x} = -1$. For example, when the imaginary component of x equals π or any multiple thereof, and the real component of x approaches zero, $G(x)$ will tend to infinity, where x belongs to the complex plane. In addition, the Cauchy–Riemann conditions of $G(x)$ are not satisfied on the same domain. This implies that the general expressions for the complex gradient of the error power with respect to hidden and output layer weights are not analytic on the domain. This complex behavior is troublesome for the back propagation. To overcome this difficulty, [19] suggests a "split" complex activation function that is created by modifying a classical sigmoidal activation function for both real and imaginary components of the input. In the next section, we use the concept of "splitting" to treat the real and imaginary neuron inputs/outputs separately as they are propagated back through the split complex activation function and its derivative, and then to avoid performing complex computations.

By applying the multilayer feedforward neural networks to the adaptive equalization problem, it is essential to establish their approximation capabilities to some arbitrary nonlinear real-vector-valued continuous mapping $\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{x})$: $D \subseteq R^{n_i} \rightarrow R^{n_0}$ from input/output data pairs $\{\boldsymbol{x}, \boldsymbol{y}\}$, where D is a compact set on R. Consider a feedforward network NN$(\boldsymbol{x}, \boldsymbol{w})$ with $\boldsymbol{x}$ as a vector representing inputs and $\boldsymbol{w}$ as a parameter weight vector that is updated by some learning rules. It is desired to train NN$(\boldsymbol{x}, \boldsymbol{w})$ to approximate the mapping $\boldsymbol{f}(\boldsymbol{x})$ as close as possible. The Stone–Weierstrass theorem [7] showed that for any continuous function $\boldsymbol{f} \in C^1(D)$ with respect to D, a compact metric space, an NN$(\boldsymbol{x}, \boldsymbol{w})$ with appropriate weight vector $\boldsymbol{w}$ can be found such that $\|\text{NN}(\boldsymbol{x}, \boldsymbol{w}) - \boldsymbol{f}(\boldsymbol{x})\|_{\boldsymbol{x}} < \epsilon$ for an arbitrary $\epsilon > 0$, where $\|\boldsymbol{e}\|_{\boldsymbol{x}}$ is the mean squared error defined by

$$\|\boldsymbol{e}\|_{\boldsymbol{x}} = \sqrt{\frac{1}{|D|}\sum_{x \in D} \|\boldsymbol{e}(\boldsymbol{x})\|^2} \tag{5}$$

where $\|\cdot\|$ is the vector norm and $|D|$ denotes the number of elements in D.

For neural network approximators, key questions are: how many layers of hidden units should be used, and how many units are required in each layer? Cybenco [8] has shown that the feedforward network with a single hidden layer can uniformly approximate any continuous function to an arbitrary degree of exactness provided that the hidden layer contains a sufficient number of units. However, it is not cost-effective for the practical implementation. Nevertheless, Chester [9] gave a theoretical support to the empirical observation that networks

with two hidden layers appear to provide high accuracy and better generalization than a single hidden layer network, and at a lower cost (i.e., fewer total processing units). Since, in general, there is no prior knowledge about the number of hidden units needed, a common practice is to start with a large number of hidden units and then prune the network whenever possible. Additionally, Huang and Huang [10] gave the lower bounds on the number of hidden units which can be used to estimate its order of magnitude.

As mentioned above, the feedforward neural network results in a static network which maps static input patterns to static output patterns. However, from (2), the satellite channel exhibits the temporal behavior where the output has a finite temporal dependence on the input. These temporal patterns in the input data are not recognizable by such a network. If the input signal is passed through a set of delay elements, the outputs of the delay elements can be used as the network inputs and temporal patterns can be trained with the standard learning algorithms of feedforward neural network. An architecture like this is often referred to as time delay neural network. It is capable of modeling dynamical systems where their input–output structure has finite temporal dependence.

Generally, the received signal transmitted over the digital satellite channels can be governed by the following discrete-time difference dynamic equation

$$r(k) = C(s(k), \cdots, s(k - n_D)) \tag{6}$$

where $r(k)$, $\boldsymbol{s}(k)$'s are the complex-valued received signal and the transmitted symbols, respectively; and n_D is the maximum lag involved in the satellite channel. The symbol $s(k)$ equals either 0 or 1 when the transmission is binary signaling. However, here, $\boldsymbol{s}(k)$'s are suggested to be in bipolar form $\{-1, 1\}$. In a general M-ary signaling system, the waveforms used to transmit the information are denoted by $\{a_k, k = 1, 2, \cdots, M\}$. It is possible to represent each symbol of the M-ary system by a $\log_2 M \times 1$ binary-state or bipolar-state vector, $\boldsymbol{s}(k)$. Here, we are interested in PSK systems. The constellations have their signal points on a circle, at $\{(\cos k\pi/2, \sin k\pi/2)\}_{k=0}^{3}$ for QPSK and $\{(\cos k\pi/4, \sin k\pi/4)\}_{k=0}^{7}$ for 8-PSK, $\cdots$ etc. The location of any signal point may be assigned to a particular bipolar-state vector $\boldsymbol{s}(k)$. The correspondence between signal location and the values of components in the bipolar-state vector is not unique. However, this correspondence is usually a one-to-one mapping. Reference [4] showed that the best choice for the code assignment is the Ungerboeck code for PSK signal in coded modulation. For example, the 8-PSK signal locations $(1, 0)$, $(-\sqrt{2}/2, \sqrt{2}/2)$, and $(-\sqrt{2}/2, -\sqrt{2}/2)$ may be assigned to $[-1, -1, -1]^T$, $[-1, 1, -1]^T$, and $[1, 1, 1]^T$ which correspond to the decimal representations "0," "3," and "5," respectively. Generally, the bipolar representation can be extended to any other M-ary systems, i.e., QAM, GMSK, and CPM.

Equation (6) becomes a weighted linear sum of transmitted symbols $\boldsymbol{s}(k)$'s when the satellite channel does not include the nonlinear TWT amplifier. Thus, the transfer function of the satellite channel between the transmitter and receiver is denoted as $H(z)$, which is an FIR system. Widrow [11] showed that a causal infinite impulse response (IIR) filter can achieve a delayed version of the system inverse to $H(z)$. The inverse or the equalizer filter for the general channel model can be governed by the following IIR-type dynamic equation

$$\hat{\boldsymbol{s}}(k) = \mathrm{EQ}\,(\boldsymbol{r}(k), \cdots, \boldsymbol{r}(k-n_f), \hat{\boldsymbol{s}}(k-1), \cdots, \hat{\boldsymbol{s}}(k-n_b)) \tag{7}$$

where $\hat{\boldsymbol{s}}(k)$ represents the equalized output signal or vector; n_f and n_b are maximum lags in the input and output, respectively. It should be noted that the responses $\hat{\boldsymbol{s}}(k)$ are identical to the transmitted symbols $\boldsymbol{s}(k)$ when the equalizer is a perfect and ideal channel inverse which can compensate the undesired nonlinear channel distortion completely. Moreover, to model the dynamics represented by (7), it is possible to convert the temporal sequence of radio frequency signal into a static pattern by unfolding the sequence over time and then use this pattern to train a static network. From a practical point of view, it is suggested to unfold the sequence over a finite period of time. This can be accomplished by feeding the input sequence into a tapped delay line of finite extent, then feeding the taps from the delay line into a static feedforward network. Thus, the channel inverse is achieved by training the static feedforward network. This can be referred to as inverse system identification. The basic configuration for achieving this is shown schematically in Fig. 3. The feedforward neural network-based decision feedback equalizer is placed behind the channel and receives both the channel outputs and detected symbols as its inputs. The network inputs at time k can be represented by $\boldsymbol{x}_k = [\boldsymbol{r}_k^T, \bar{\boldsymbol{s}}_k^T]^T$, where $\boldsymbol{r}_k = [\boldsymbol{r}^T(k), \cdots, \boldsymbol{r}^T(k - n_f)]^T$ and $\bar{\boldsymbol{s}}_k = [\bar{\boldsymbol{s}}^T(k-1), \cdots, \bar{\boldsymbol{s}}^T(k - n_b)]^T$. Notice that the received complex-valued signals $\boldsymbol{r}(k)$ should be represented by a 2×1 vector, i.e., $[r_I, r_Q]^T$, because the error back-propagation algorithms cannot be applied to the complex-valued inputs directly, where r_I and r_Q represent the in-phase (real) and quadrature (imaginary) components of $r(k)$. The detected symbols $\bar{\boldsymbol{s}}(k)$ are generated by feeding the static neural network outputs $\tilde{\boldsymbol{s}}(k)$ through a hardlimiter at time instant k and given by $\bar{\boldsymbol{s}}(k) = \mathbf{sign}(\tilde{\boldsymbol{s}}(k))$, where $\tilde{\boldsymbol{s}}(k) = NN(\boldsymbol{x}_k; \boldsymbol{w})$ and $\mathbf{sign}(v) = [sign(v_1), sign(v_2), \cdots, sign(v_n)]^T$, $\boldsymbol{v} = [v_1, v_2, \cdots, v_n]^T$. According to Fig. 3, the input–output relationship of the neural equalizer can be characterized by the function

$$\bar{\boldsymbol{s}}(k) = \mathrm{NNDFE}(\boldsymbol{x}_k; \boldsymbol{w}) = \mathbf{sign}(\mathrm{NN}(\boldsymbol{x}_k; \boldsymbol{w})) \tag{8}$$

where $\boldsymbol{w}$ is the weight vector of the feedforward network and $\bar{\boldsymbol{s}}(k)$ is the estimate of $\boldsymbol{s}(k)$. The training data involved in transmitted symbols provide the desired response of the static feedforward network, $\boldsymbol{d}_k(= \boldsymbol{s}(k))$ to train the network to approximate the perfect channel inverse or ideal equalizer $\mathrm{EQ}_{\mathrm{ideal}}(\cdot)$. Notice that a replica of the desired response is stored in the receiver. By the Stone–Weierstrass theorem, it is possible to find the appropriate weight vector $\boldsymbol{w}^*$ of the static feedforward network of the neural-based equalizer, such that

$$\|\mathrm{NN}(\boldsymbol{x}_k; \boldsymbol{w}^*) - \mathrm{EQ}_{\mathrm{ideal}}(\boldsymbol{x}_k)\|_{\boldsymbol{x}_k} < \epsilon \tag{9}$$

for an arbitrary $\epsilon > 0$ and all the $\boldsymbol{x}_k$ are in the region of interest.

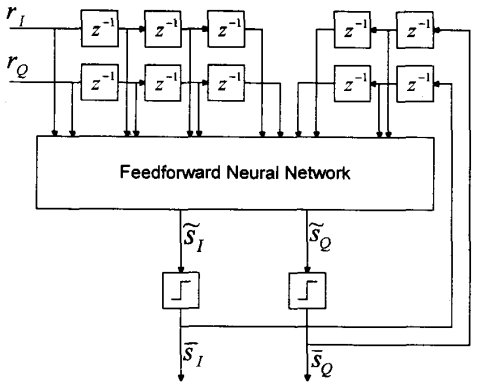

Fig. 3. Architecture of neural-based decision feedback equalizer for M-ary PSK system.

Since $\boldsymbol{d}_k$ is represented by a bipolar-state vector, each component of $\mathrm{NN}(\boldsymbol{x}_k; \boldsymbol{w}^*)$ becomes either -1 or 1 after a sufficiently long training period. This would imply that $\mathrm{NNDFE}(\boldsymbol{x}_k; \boldsymbol{w}^*) = \mathrm{NN}(\boldsymbol{x}_k; \boldsymbol{w}^*)$. From (9), we have

$$\|\mathrm{NNDFE}(\boldsymbol{x}_k; \boldsymbol{w}^*) - \mathrm{EQ}_{\mathrm{Ideal}}(\boldsymbol{x}_k)\|_{\boldsymbol{x}_k} < \epsilon. \tag{10}$$

For an M-ary PSK signaling communication system, $\boldsymbol{s}(k)$, $\tilde{\boldsymbol{s}}(k)$, and $\bar{\boldsymbol{s}}(k)$ should be represented by $\log_2 M \times 1$ vectors. Moreover, the received signal $r(k)$ is complex-valued and then can be expressed as a two-dimensional vector. Thus, the input layer of the network consists of two n_f-tap forward filters, and $\log_2 M$ n_b-tap feedback filters. As a result, the number of neurons in the input layer is given by

$$n_i = 2 \times (n_f + 1) + (\log_2 M) \times n_b. \tag{11}$$

In the output layer, the number of neurons is $n_o = \log_2 M$.

IV. Multilayer Neural Networks and Their Learning Rules

An iterative learning algorithm, called back propagation, was suggested by Rumelhart *et al.* [12]. This iterative learning algorithm is used to adjust the weights and threshold levels of the network in a pattern-wise manner. In a mathematical sense, the back-propagation learning rule is used to train the feedforward network $\mathrm{NN}(\boldsymbol{x}, \boldsymbol{w})$ to approximate a function $\boldsymbol{f}(\boldsymbol{x})$ from compact subset D of n_i-dimensional Euclidean space to a bounded subset $\boldsymbol{f}(D)$ of n_0-dimensional Euclidean space. Let $\boldsymbol{x}_k$ which belongs to D be the kth pattern or sample and selected randomly as the input of the neural network at time instant k, let $\mathrm{NN}(\boldsymbol{x}_k, \boldsymbol{w})$ $(= \boldsymbol{o}_k)$ be the output of the neural network, and let $\boldsymbol{f}(\boldsymbol{x}_k)$ $(= \boldsymbol{d}_k)$ which also belongs to $\boldsymbol{f}(D)$ be the desired output. This task is to adjust all the variable weights of the neural network such that the pattern-wise quadratic error E_k can be reduced, where E_k is defined as

$$E_k = \frac{1}{2}\|\mathrm{NN}(\boldsymbol{x}_k, \boldsymbol{w}) - \boldsymbol{f}(\boldsymbol{x}_k)\|^2 = \frac{1}{2}\sum_{j=1}^{n_0}(o_{kj} - d_{kj})^2 \tag{12}$$

where n_0 is the number of output nodes, o_{kj} and d_{kj} are the jth components of $\boldsymbol{o}_k$ and $\boldsymbol{d}_k$, respectively.

Here, we define the weighted sum of the output of the previous layer by the presentation of input pattern $\boldsymbol{x}_k$

$$net_{kj} = \sum_i w_{ji} o_{ki} \tag{13}$$

where w_{ji} is the weight which connects the output of the ith neuron in the previous layer with respect to the jth neuron, and o_{ki} is the output of the ith neuron. It should be noted that o_{ki} is equal to x_{ki} when the ith neuron is located in the input layer, where x_{ki} is the ith component of pattern $\boldsymbol{x}_k$. Notice that the threshold level variables are treated as the additional weight variables in the network. Using (3), the output of neuron j is

$$o_{kj} = \begin{cases} x_{kj}, \\ \quad \text{if the neuron } j \text{ belongs to the input layer} \\ G(net_{kj}), \\ \quad \text{otherwise.} \end{cases} \tag{14}$$

The pattern-wise or on-line back-propagation algorithm [12] minimizes the quadratic error E_k by recursively altering the connection weight vector at each pattern according to the expression

$$w_{ji}(k+1) = w_{ji}(k) - \eta \left.\frac{\partial E_k}{\partial w_{ji}}\right|_{w_{ji}=w_{ji}(k)} \tag{15}$$

where the learning rate η is usually set to be equal to a positive constant less than unity.

It is useful to see the partial derivative for pattern k, $\partial E_k / \partial w_{ji}$, as resulting from the product of two parts: one part reflecting the change in error to a function of the change in the network input to the neuron, and one part representing the effect of changing a particular weight on the network input

$$\frac{\partial E_k}{\partial w_{ji}} = \frac{\partial E_k}{\partial net_{kj}} \cdot \frac{\partial net_{kj}}{\partial w_{ji}}. \tag{16}$$

From (13), the second part becomes

$$\frac{\partial net_{kj}}{\partial w_{ji}} = \frac{\partial}{\partial w_{ji}}\left(\sum_i w_{ji} o_{ki}\right) = o_{ki}. \tag{17}$$

An error signal term δ called delta produced by the jth neuron is defined as follows

$$\delta_{kj} \triangleq -\frac{\partial E_k}{\partial (net_{kj})}. \tag{18}$$

Note that E is a composite function of net_{kj}, it can be expressed as follows

$$\begin{aligned} E_k &= E_k(o_{k1}, o_{k2}, \cdots, o_{kL}) \\ &= E_k(G(net_{k1}), G(net_{k2}), \cdots, G(net_{kL})) \end{aligned} \tag{19}$$

where L is the number of the neurons in the current layer. Thus, we have from (18)

$$\partial_{kj} = -\frac{\partial E_k}{\partial o_{kj}} \frac{\partial o_{kj}}{\partial net_{kj}}. \tag{20}$$

Denoting the second term in (20) as a derivative of the activation function

$$\frac{\partial o_{kj}}{\partial net_{kj}} = G'(net_{kj}). \tag{21}$$

However, to compute the first term, there are two cases. For the hidden-to-output connections, it follows the definition of E_k that

$$\frac{\partial E_k}{\partial o_{kj}} = -(d_{kj} - o_{kj}). \tag{22}$$

Substituting for the two terms in (20), we can get

$$\delta_{kj} = (d_{kj} - o_{kj})G'(net_{kj}). \tag{23}$$

Second, for hidden (or input)-to-hidden connection, the chain rule is used to write

$$\frac{\partial E_k}{\partial o_{kj}} = \sum_l \frac{\partial E_k}{\partial net_{kl}} \frac{\partial net_{kl}}{\partial o_{kj}} = -\sum_l \delta_{kl} w_{lj}. \tag{24}$$

Substituting into (20), it yields

$$\delta_{kj} = G'(net_{kj}) \sum_l \delta_{kl} w_{lj}. \tag{25}$$

Equations (23) and (25) give a recursive procedure for computing the δ's for all neurons in the network. In summary, the error signals δ's for all neurons in the network can be computed according to the following recursive procedure

$$\delta_{kj} = \begin{cases} (d_{kj} - o_{kj})G'(net_{kj}), \\ \quad \text{if neuron } j \text{ belongs to the output layer} \\ G'(net_{kj}) \sum_l \delta_{kl} w_{lj}, \\ \quad \text{otherwise.} \end{cases} \tag{26}$$

It should be mentioned that o_{ki} is equal to x_{ki} when neuron i belongs to the input layer. The expression of (26) is also called the generalized delta learning rule. Once those error signal terms have been determined, the partial derivatives for the quadratic error of the kth pattern can be computed directly by

$$\frac{\partial E_k}{\partial w_{ji}} = \frac{\partial E_k}{\partial net_{kj}} \frac{\partial net_{kj}}{\partial w_{ji}} = -\delta_{kj} o_{ki}. \tag{27}$$

Thus, the update rule of the on-line back-propagation algorithm is

$$w_{ji}(k+1) = w_{ji}(k) + \eta \delta_{kj} o_{ki}. \tag{28}$$

In the traditional equalizer, a replica of the desired response is stored in the receiver. Naturally, the generator of this stored reference has to be electronically synchronized with the known transmitted sequence. A widely used training signal consists of a pseudonoise (PN) sequence of length N_B. Moreover, the training signal can also be expressed as a collection of input–output data pairs, $\{\boldsymbol{x}_k, \boldsymbol{d}_k\}_{k=1}^{N_B}$. The weights of neural-based equalizer are updated by using the batch of these data pairs. Thus, the objective function should be modified in an expression of summation, $E = \sum_{k=1}^{N_B} E_k$, during the initial training. Thus, the update rule becomes

$$w_{ji}^{\text{new}} = w_{ji}^{\text{old}} - \eta \frac{\partial E}{\partial w_{ji}} = w_{ji}^{\text{old}} + \eta \sum_{k=1}^{N_B} \delta_{kj} o_{ki}. \tag{29}$$

Notice that the quantity $\boldsymbol{w}(k+1)$ is the updated weight vector after one pattern of learning; $\boldsymbol{w}^{\text{new}}$ is the updated weight vector after one batch of learning.

It is shown that the batch back-propagation learning algorithm is used to initialize the weight coefficients of the neural-based equalizer when the channel is unknown. Nevertheless, the on-line learning algorithm is used to adjust the weights to track channel time variations and said to be decision directed. However, from [13], the initial training can be executed by the on-line learning algorithm instead of batch learning since on-line learning is shown to approach batch learning provided that η is small. Initialization may be aided by the transmission of N_B known training symbols. The trained neural-based equalizer converges to the channel inverse when N_B is sufficiently large. It is known that the decision errors in equalizer tracking can lead directly to crashing of the equalizer, especially when the adaptation gain is high. Decision errors become more prevalent when the received signal-to-noise ratio is low, a condition that occurs unpredictably in nonlinear digital satellite channels. The susceptibility of adaptive equalizers or neural-based equalizers to crashes caused by propagation of decision errors implies that retraining procedures must be specific. For nonlinear satellite channel, periodic retraining is often used to improve reliability at some cost in throughput efficiency, via periodic insertion of training symbols into the data stream. More details about the packet adaptive equalization will be discussed in the next section.

A. Learning for Packet Adaptive Equalization

Packet equalization is a method that arises in TDMA communication systems, in which data are transmitted in fixed-length packets, rather than continuously [18]. It is usually assumed that the packet is largely self-contained for error detection, i.e., in terms of equalize initialization and at least fine synchronization. This overhead can achieve good performance with reasonable complexity. Packet equalization has some similarities with block-oriented methods for periodic training on continuous channels, although it is assumed that the channel state is independent from packet to packet. Frequency and packet synchronization are assumed to be maintained once initialized, but symbol timing and phase synchronization are assumed to be maintained once initialized, but symbol timing and phase synchronization typically need to be restored for each packet. It is known that the optimum approach to equalization is an off-line noncausal batch processing of the received signal with a large amount of training data. Such a formulation would be unfeasible and too complex to implement, but iterative approaches based on periodic training are possible. We will show that there is an equivalence of the off-line equalization and the packet-wise neural-based equalizer when the number of packets approaches infinity.

Considering the packet transmission, it is assumed that the length of the training data contained in the packet header and the total packet length are n_u and n_p, respectively. The main idea of the packet training scheme is used to train the neural-based equalizer with n_u training data for each packet. The neural-based equalizer can be retrained for every packet and thus track the time variations in the channel. This is quite similar to the on-line training for a sequence of packets.

The packet version of the back-propagation-based algorithm can be obtained by modifying (29) and given by

$$w_{ji}^{\text{new}} = w_{ji}^{\text{old}} + \eta \sum_{k=1}^{n_u} \delta_{kj} o_{ki}. \tag{30}$$

Furthermore, during data transmission after packet training period, the decision-directed adaptation is executed by (28). Similarly, the delta-bar-delta rule shown in the next section can also be applied to packet-wise back-propagation algorithm in order to increase the rate of convergence.

Furthermore, from the Stone–Weierstrass theorem, the exact channel inverse can be obtained by the batch learning when the length of training data is large enough and the maximum lags in both input and output of channel inverse are known. Especially, for time-varying channels, N_B would approach infinity. But it is unfeasible for any channel equalizer transmitting a large amount of training data continuously. Fortunately, [17] shows that the problem is solved by inserting a finite number of training data into the data stream periodically. This leads to the following theorem.

Theorem 1: The neural equalizer is guaranteed to be capable of converging to the channel inverse globally by performing the packet back propagation algorithms with a sufficient number of hidden nodes when the maximum lags in the input and output of channel inverse are known. From [17] and the Stone–Weierstrass theorem, a solution $\boldsymbol{w}^*$ can be generated by the packet-wise back-propagation algorithm such that

$$\|\text{NNDFE}(\boldsymbol{x}_k; \boldsymbol{w}^*) - \text{EQ}_{\text{ideal}}(\boldsymbol{x}_k)\|_{\boldsymbol{x}_k} < \epsilon \tag{31}$$

for an arbitrary $\epsilon > 0$, as $N \to \infty$, where N is the number of packets and $N_g = N \cdot n_u$.

B. Fast Learning Algorithms

Although the back-propagation algorithm is a useful method to find an optimal solution of a set of weight values of a network, many researchers [14] showed that the convergence rate of this algorithm is relatively slow. It is necessary to find a faster algorithm to improve the back-propagation algorithm and also to achieve better performance. In this subsection, a modified back-propagation algorithm with much faster convergence rate will be introduced to the neural equalizer. This faster algorithm is used to modify the negative gradient direction by adjusting the learning rate, and it is called the delta-bar-delta rule [14].

Since a large η corresponds to rapid learning but might also result in oscillations, [12] suggests that expression (15) might be modified to include a sort of momentum term in order to dampen oscillation. That is, the weight w_{ji} is updated at the k+1st iteration, according to the rule

$$\Delta w_{ji}(k+1) = -(1-\alpha)\eta(k+1)\frac{\partial E_k}{\partial w_{ji}} + \alpha\, \Delta w_{ji}(k) \tag{32}$$

where $\Delta w_{ji}(k+1)$ is the weight increment for the $k+1$st iteration, $\eta(k+1)$ is the learning rate value corresponding to $\Delta w(k+1)$ at time $k+1$, and α is the momentum rate.

However, Jacobs [14] showed that the momentum can cause the weight to be adjusted up the slope of the system error surface. This would decrease the performance of the learning algorithm. To overcome this difficulty, Jacobs [14] proposed a promising weight update algorithm based on the delta-bar-delta rule which consists of both a weight update rule and a learning rate update rule. The weight update rule is the same as the steepest descent algorithm and is given by (32). The delta-bar-delta learning rate update rule is described to follows

$$\Delta\eta(k+1) = \begin{cases} \kappa, & \text{if } \overline{\lambda}(k-1)\lambda(k) > 0 \\ -\phi\eta(k), & \text{if } \overline{\lambda}(k-1)\lambda(k) < 0 \\ 0, & \text{otherwise} \end{cases} \tag{33}$$

where $\lambda(t) = \partial E_k / \partial w_{ij}$ and $\overline{\lambda}(t) = (1-\theta)\lambda(t) + \theta\overline{\lambda}(t-1)$.

In these equations, $\lambda(k)$ is the partial derivative of the system error with respect w_{ij} at the kth iteration, and $\overline{\lambda}(k)$ is an exponential average of the current and past derivatives with θ as the base and index of iteration as the exponent. If the current derivative of a weight and the exponential average of the weight's previous derivatives possess the same sign, then the learning rate for that weight is incremented by a constant κ. The learning rate is decremented by a proportion ϕ of its current value when the current derivative of a weight and the exponential average of the weight's previous derivatives possess opposite signs.

From (33), it can be found that the learning rates of the delta-bar-delta algorithm are incremented linearly in order to prevent them from becoming too large too fast. The algorithm also decrements the learning rates exponentially. This ensures that the rates are always positive and allows them to be decreased rapidly. Jacobs [14] showed that a combination of the delta-bar-delta rule and momentum heuristics can achieve both the good performance and faster rate of convergence.

More recent work has produced improved learning strategies based on extended Kalman algorithm [15] and a recursive prediction error routine [16]. Although these two algorithms were each derived independently based on a different approach, they are actually equivalent. They both use the same search direction called the Gauss–Newton direction, for which the negative gradient is multiplied by the inverse of an approximate Hessian matrix of the given criterion. The computational complexity of both algorithms applied to the neural equalizer is examined in [17].

V. Simulation Results

The QPSK satellite channel model used in performance evaluation is based on the Volterra series expansion model developed as a result of measurement [1], [2]. The associated Volterra series coefficients are shown in Table I. Moreover, the channel output is corrupted by zero mean additive white Gaussian noise (AWGN). For convenience, the received signal could be normalized to unity. Then the received signal-to-noise ratio (SNR) becomes the reciprocal of the noise variance at the input of the equalizer. The bit error rates were determined by simulating the QPSK data transmission system and taking an average of 100 individual runs of 10^5 samples.

The complex-valued data are transmitted at a bit rate of 60 Mb/s over the Volterra-series-based QPSK satellite channel. The modulation scheme is QPSK with a symbol rate of 30

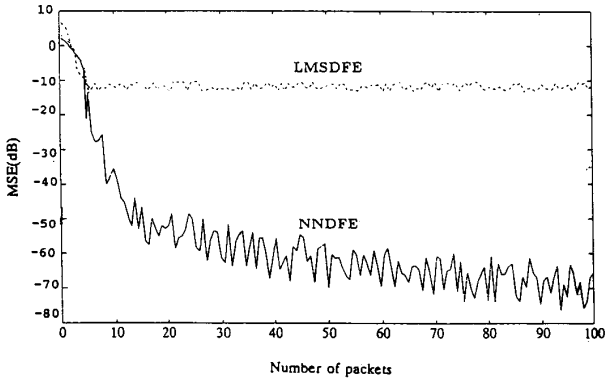

Fig. 4. Comparison of mean equare errors achieved by NNDFE and LMS-DFE when $n_u = 40$ and SNR = 10 dB.

M symbols/s and a symbol interval of 100 ns. The packet length is set to 400 symbols (800 bits). A 10% overhead would allow a maximum of 40 symbols for training, i.e., $n_u = 40$. The neural-based decision feedback equalizer includes a four-layer feedforward neural network. For simplicity, the neural equalizer is denoted by a short-hand notation NNDFE $((n_f, n_b), n_1, n_2, n_0)$, where n_f is the number of forward taps, n_b is the number of forward taps, n_1 is the number of neurons in hidden layer 1, n_2 is the number of neurons in hidden layer 2, and n_0 is the number of neurons in output layer. Similarly, traditional LMS decision feedback equalizer is denoted by LMSDFE (n_f, n_b). According to the suggestions of [3] and (11), n_f and n_b are set to be 3 and 2, respectively. Karam and Sari [3] showed that a two-stage feedback filter can cancel most third- and higher-order ISI terms resulted from the Volterra-series-based channel model. As discussed in Section II-A, n_i and n_0 can be found as 12 and 2, respectively, for QPSK system. According to Huang and Huang's [10] suggestions, it is possible to estimate the lower bounds on the numbers of neurons in both hidden layers. Thus, n_1 and n_2 can be chosen as 20 and 10, respectively. Fig. 4 illustrates MSE (mean square error) convergence of the packet-wise neural equalizer, NNDFE $((3, 2), 20, 10, 2)$, and LMSDEF $(3, 2)$ with the same initial learning rate $\eta(k)|_{k=0}$ 0.03 for training mode and 0.005 for decision-directed mode. The MSE is defined as follows

$$MSE = \sqrt{\frac{1}{N_B}\sum_{k=1}^{N_B} \|\bar{\boldsymbol{s}}(k) - \boldsymbol{d}_k\|^2} \tag{34}$$

where $\bar{\boldsymbol{s}}(k) = \text{NNDFE}(\boldsymbol{x}_k, \boldsymbol{w})$, $N_g = N \cdot n_u$ and $\boldsymbol{d}_k$ denotes the desired response. The values of parameters related to the delta-bar-delta rule are $\alpha = 0.8$, $\kappa = 0.00005$, $\phi = 0.4$, and $\theta = 0.3$. The NNDFE requires at least 50 packets to converge, while the LMSDFE converges in about 8 packets. The results also show that the steady-state value of averaged square error produced by the NNDFE converges to a value ($\leq$ –70 dB), which is much lower than the additive noise (–10 dB). This is a result of the approximation capability of packet back-propagation. Theorem 1 indicates that the approximation error approaches zero when the number of packets approaches infinity. The LMSDFE gives a steady value of averaged squared error at about –10 dB, which is around the noise floor. Fig. 5 compares the respective bit error rates (BER's) achieved by NNDFE $((3, 2), 20, 10, 2)$ and LMSDFE $(3, 2)$. It may be observed from Fig. 5 that the NNDFE attains about 3.2 dB improvement at BER = 10^{-4} relative to the LMSDFE having the same number of input samples.

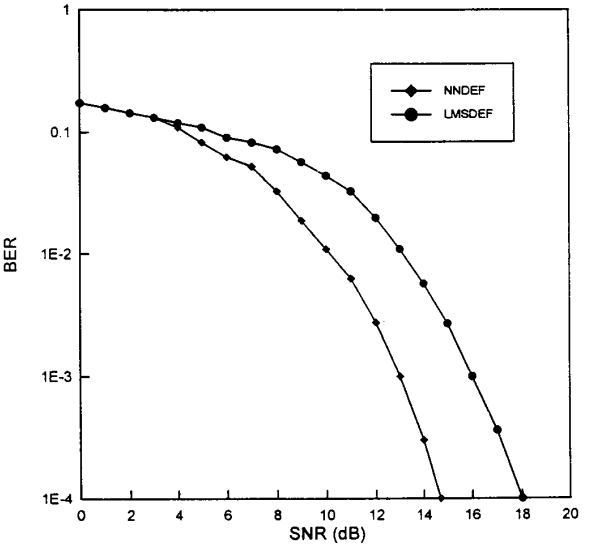

Fig. 5. Comparison of bit error rates achieved by NNDFE and LMSDFE when $n_u = 40$.

VI. Conclusion

This paper has introduced a four-layer neural-based adaptive decision feedback equalizer based on a concept of splitting which is capable of dealing with the M-ary PSK signals over the complex digital satellite channel by using cost-effective real-valued training algorithms, where the real and imaginary components of neuron outputs are separately propagated back through the split complex activation function and its derivative. The neural-based DFE offers a superior performance as a channel equalizer to that of the conventional LMS DFE, because of its ability to approximate arbitrary nonlinear mapping. For comparison of simulation results for the Volterra-series-based QPSK satellite channel, it can be seen that the neural-based DFE provides better BER performance, especially in high-noise conditions; also, the MSE of neural-based DFE converges to a value which is much lower than that of LMS DFE after receiving a sufficient number of packets. These results would be conducted to verify the performance and approximation capability of packet-wise back propagation.

References

[1] S. Benedetto, E. Biglieri, and R. Daffara, "Modeling and evaluation of nonlinear satellite links—A Volttera series approach," *IEEE Trans. Aerosp. Electron. Syst.*, vol. AES-15, pp. 494–506, July 1979.

[2] S. Benedetto and E. Biglieri, "Nonlinear equalization of digital satellite channels," *IEEE Trans. J. Select. Areas Commun.*, vol. SAC-1, pp. 57–62, Jan. 1983.
[3] G. Karam and H. Sari, "Analysis of predistortion, equalization, and ISI cancellation techniques in digital radio systems with nonlinear transmit amplifiers," *IEEE Trans. Commun.*, vol. 37, pp. 1245–1253, Dec. 1989.
[4] E. Biglieri *et al.*, *Introduction to a Trellis-Coded Modulation with Applications.* New York: Macmillan, 1992.
[5] S. Siu, G. J. Gibson, and C. F. N. Cowan, "Decision feedback equalization using neural network structures and performance comparison with standard architecture." *IEEE Proc.*, vol. 137, pt. I, no. 4, pp. 221–225, Aug. 1990.
[6] S. Chen *et al.*, "Adaptive equalization of finite nonlinear channels using multilayer perceprons," *Signal Processing*, vol. 20, pp. 107–119, 1990.
[7] M. E. Cotter, "The Stone–Weierstrass theorem and its application to neural nets," *IEEE Trans. Neural Networks,* vol. 1, pp. 290–295, 1990.
[8] G. Cybenko, "Approximations by superposition of a sigmoidal function," *Math. Contr. Syst. Signals,* vol. 2, no. 4, 1989.
[9] D. Chester, "Why two hidden layers are better than one," in *Proc. Int. Joint Conf. Neural Networks* (Washington, DC), June 1989, pp. I-613–I-618.
[10] S. C. Huang and Y. F. Huang, "Bounds on the number of hidden neurons in multilayer perceptrons," *IEEE Trans. Neural Networks,* vol. 2, pp. 47–55, Jan. 1991.
[11] B. Widrow and R. Winter, "Neural nets for adaptive filtering and adaptive pattern recognition," *IEEE Comput. Mag.*, vol. 21, pp. 25–39, Mar. 1988.
[12] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representation by error propagation," in *Parallel Distributed Processing: Explorations in the Microstructure of Congnition,* vol. 1. Cambridge, MA: MIT Press, 1986, ch. 8.
[13] S. Z. Qin, H. T. Su, and T. J. McAvoy, "Comparison of four neural net learning methods for dynamic system identification," *IEEE Trans. Neural Networks,* vol. 3, pp. 122–130, Jan. 1992.
[14] R. A. Jacobs, "Increased rates of convergence through learning rate adaption," *Neural Networks,* vol. 1, pp. 295–307, 1988.
[15] Y. Inguni, H. Sakai, and H. Tokumaru, "A real-time learning algorithm for a multilayered neural network based on the extended Kalman filter," *IEEE Trans. Signal Processing,* vol. 40, pp. 959–966, Apr. 1992.
[16] S. Chen *et al.*, "Parallel recursive prediction error algorithm for training layered neural networks," *Int. J. Contr.*, vol. 51, no. 6, pp. 1215–1228, 1990.
[17] P. R. Chang and C. C. Chang, "Adaptive packet equalization for indoor radio channel using multilayer neural networks," *IEEE Trans. Veh. Technol.*, vol. 43, Aug. 1994.
[18] J. G. Proakis, "Adaptive equalization for TDMA digital mobile radio," *IEEE Trans. Veh. Technol.,* vol. 40, pp. 333–341, May 1991.
[19] D. L. Birx and S. J. Pipenberg, "A complex mapping network for phase sensitive classification," *IEEE Trans. Neural Networks.,* vol. 4, pp. 127–135. Jan. 1993.
[20] H. Leung and S. Haykin, "The complex backpropagation algorithm," *IEEE Trans. Signal Processing,* vol. 39, no. 9, pp. 2101–2104, Sept. 1991.

Po-Rong Chang (M'87) received the B.S. degree in electrical engineering from the National Tsing-Hua University, Taiwan, in 1980; the M.S. degree in telecommunication engineering from National Chiao-Tung University, Hsinchu, Taiwan, in 1982; and the Ph.D. degree in electrical engineering from Purdue University, West Lafayette, IN, 1988.

From 1982 to 1984 he was a Lecturer in the Chinese Air Force Telecommunication and Electronics School for his two-year military service. From 1984 to 1985 he was an Instructor of electrical engineering at National Taiwan Institute of Technology, Taipei, Taiwan. From 1989 to 1990 he was a Project Leader in charge of SPARC chip design team at ERSO of Industrial Technology and Research Institute, Chu-Tung, Taiwan. Currently he is an Associate Professor of Communication Engineering at National Chiao-Tung University. His current interests include fuzzy neural network, wireless multimedia systems, and virtual reality.

Bor-Chin Wang was born in Tainan, Taiwan, in 1961. He received the B.S. and M.S. degrees in power mechanical engineering from the National Tsing-Hua University, Hsinchu, Taiwan, in 1983 and 1985, respectively. He is presently working toward the Ph.D. degree in communication engineering at National Chiao-Tung University, Hsinchu, Taiwan.

From 1985 to 1993 he was an Assistant Research Engineer at the Chung-Shan Institute of Science and Technology, Ministry of National Defense, Republic of China, where he worked on servo system design. His current research interests are in CDMA cellular system, wireless multimedia system, fuzzy neural network, and PCS systems.