Communication Channel Equalization Using Complex-Valued Minimal Radial Basis Function Neural Networks

Deng Jianping, Narasimhan Sundararajan, *Fellow, IEEE,* and P. Saratchandran, *Senior Member, IEEE*

***Abstract*—In this paper, a complex radial basis function neural network is proposed for equalization of quadrature amplitude modulation (QAM) signals in communication channels. The network utilizes a sequential learning algorithm referred to as complex minimal resource allocation network (CMRAN) and is an extension of the MRAN algorithm originally developed for online learning in real-valued radial basis function (RBF) networks. CMRAN has the ability to grow and prune the (complex) RBF network's hidden neurons to ensure a parsimonious network structure. The performance of the CMRAN equalizer for nonlinear channel equalization problems has been evaluated by comparing it with the functional link artificial neural network (FLANN) equalizer of Patra *et al.* and the Gaussian stochastic gradient (SG) RBF equalizer of Cha and Kassam. The results clearly show that CMRANs performance is superior in terms of symbol error rates and network complexity.**

***Index Terms*—Channel equalization, complex minimal resource allocation network (CMRAN), quadrature amplitude modulation (QAM), radial basis function (RBF) neural network.**

I. Introduction

With the growth of internet technologies, efficient high-speed data transmission techniques over communication channels have become an important topic for research. The channels used to transmit the data distort signals in both the amplitude and phase, causing what is known as intersymbol interference (ISI). This distortion causes the transmitted symbols to spread and overlap over successive time intervals, causing the information content to also spread among these symbols. Other factors like thermal noise, impulse noise, cross talk and the nature of the channel itself, cause further distortions to the received symbols. Signal processing techniques used at the receiver, to overcome these interferences, so as to restore the transmitted symbols and recover their information, are referred to as "equalization methods." Estimation theory suggests that the best performance for symbol detection is obtained by using a maximum likelihood sequence equalizer (MLSE) for the entire symbol sequence, which involves a batch-processing scheme. But the computational complexity of the MLSE is prohibitive and this has led to the popularity of equalizers that make decisions symbol-by-symbol as an alternative. The optimal solution for these symbol-decision equalizers has been approached using the Bayesian decision theory [1]. It can be seen from the Bayesian solution that the optimal solution corresponds to a nonlinear classification problem.

It is well known that neural networks are well suited for solving nonlinear classification problems. Because of this, multilayer feedforward neural networks, radial basis function (RBF) networks and recurrent neural networks have gained popularity in their use for equalization problems [2]–[4]. Artificial neural networks have also been applied for maximum likelihood sequence estimation (MLSE) [5], [6]. Initial work [7], [8] had demonstrated that multilayer perceptron (MLP) equalizers are superior to conventional transversal and decision feedback equalizers in terms of the equalizer performance metric, symbol error rate (SER). The problems, which severely restrict the practical implementations of MLPs, are their long training times and the lack of a methodology for the network architecture selection. Recently, Uncini *et al.* [9] have proposed a nonstandard MLP with an adaptable activation function using Catmull-Rom cubic splines to reduce the high complexity and long training time for equalizing a digital satellite radio link. RBF neural networks provide an attractive alternative to MLP for channel equalization problems because the structure of the RBF network has a close relationship to Bayesian methods for channel equalization and interference rejection problems [3]. RBF networks are presented with a training set of input–output pairs and use a training algorithm to learn the nonlinear mapping from input to output. Thus, they essentially carry out an approximation for the nonlinear mapping from the input to the output. In some applications, the signals are complex-valued and processing has to be done in a complex multidimensional space, e.g., equalization of digital communication channels with multilevel quadrature amplitude modulation (QAM) scheme. The modulation technique uses the data that have two components, amplitude and phase. Nonlinear characteristics of the channel cause spectral spreading, intersymbol interference and constellation warping on the QAM signals. Since the QAM links are very sensitive to nonlinear distortion, the equalization of QAM signals is a difficult problem to solve [9]. For these problems, in order to preserve the concise formulation and elegant structure of the complex signals, complex neural-network equalizers have been proposed.

Some researchers have come up with a complex MLP and extended the backpropagation algorithm to a complex form [10]. Patra *et al.* [11] have developed a functional link artificial neural network (FLANN) for equalization of QAM signals and have

Manuscript received November 29, 2000; revised November 12, 2001.

The authors are with the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore 639798 (e-mail: ensundara@ntu.edu.sg).

Publisher Item Identifier S 1045-9227(02)04443-0.

compared its performance with MLP and polynomial perceptron network (PPN) equalizers. FLANN has a single-layer ANN structure in which the nonlinearity is introduced by functional expansion of the input pattern by trigonometric polynomials. Because of the input pattern enhancement, FLANN is capable of forming arbitrarily nonlinear decision boundaries and can perform complex pattern classification tasks. They have shown that the FLANN equalizer's performance is better in terms of symbol error rate, mean square error and computational complexity based on studies on a number of linear and nonlinear real valued channel models.

Chen *et al.* [12] and Cha and Kassam [13] have independently proposed a complex radial basis function (CRBF) network, which is an extension of its real counterpart. The inputs and outputs of the network are both complex-valued while the radial basis function remains real. Cha and Kassam have used a stochastic gradient learning algorithm for their complex RBF network equalizer and have shown that its performance is superior to several existing algorithms based on a number of nonlinear complex channel models. The stochastic-gradient (SG) training algorithm (with Gaussian basis function) adapts all the free parameters of the network simultaneously by using stochastic gradient descent for error criterion. That is, the SG algorithm takes the instantaneous gradient of the squared error and moves the network parameters in the opposite direction of their respective gradients.

Recently, a new RBF network learning algorithm called minimal resource allocation network (MRAN) was developed by Lu Yingwei *et al.* [14], [15]. The sequential learning MRAN algorithm employs a scheme for adding and pruning RBFs hidden neurons, so as to achieve a parsimonious network structure. In MRAN algorithm, the network begins with no hidden neurons. As each training data pair (input and output) is received the network builds itself up based on three growth criteria. The algorithm adds new hidden neurons or adjusts the existing network parameters according to the training data received. The algorithm also incorporates a pruning strategy that is used to remove the hidden neurons that do not contribute significantly to the output. A number of successful application of MRAN in areas such as nonlinear system identification, function approximation, and time series prediction have been reported in [14], [15]. MRAN was first introduced in [16] to solve the communication equalization problems for real-valued channels and signals. In this paper, MRAN learning algorithm is extended to complex form (complex MRAN or CMRAN) and its application in complex channel equalization problems (QAM signals) is investigated. In order to compare the CMRANs performance with the earlier schemes, the same channel models from [11]–[13] have been used in this study.

The paper is organized as follows. Section II gives a brief introduction to nonlinear complex channel equalization problem. Section III describes the newly developed complex MRAN algorithm. The main results of the paper on the application of CMRAN for different nonlinear QAM channel equalization problems are presented in Section IV. This section also provides a comparison of performance with other methods. Conclusions based on the study are summarized in Section V.

II. Complex Channel Equalization Problem

Consider the base band discrete-time model of a data transmission system given by

$$\begin{aligned} y(n) &= \hat{y}(n) + e(n) \\ &= f_h(s(n), s(n-1), \ldots, s(n-n_h+1)) + e(n) \end{aligned} \quad (1)$$

where a complex valued digital sequence $s(n)$ is transmitted through a dispersive complex channel and the channel output is corrupted by an additive complex-valued noise $e(n)$. In the above equation $f_h(.)$ is some complex valued function (which may be linear or nonlinear), n_h is the length of the FIR channel. The task of the symbol decision equalizer is to reconstruct the transmitted symbols $s(n-\tau)$ based on noisy channel observation vector $\mathbf{y(n)} = [y(n)\cdots y(n-m+1)]^{\mathrm{T}}$ where τ is the equalizer decision delay and m is the equalizer dimension. It is well known that Bayes decision theory provides the optimal solution to the symbol decision problem [1].

Assume a 4-QAM input sequence with alphabet $\alpha = \{\alpha^{(1)}, \alpha^{(2)}, \alpha^{(3)}, \alpha^{(4)}\}$ where $\alpha^{(1)} = a + ja$, $\alpha^{(2)} = a - ja$, $\alpha^{(3)} = -a + ja$, $\alpha^{(4)} = -a - ja$ going through a noiseless channel of order n_h. The input sequence: $\mathbf{s(n)} = [s(n)\cdots s(n-m+2-n_h)]^{\mathrm{T}}$ would result in n_s points or values of noise-free channel output vector $\hat{\mathbf{y}}(\mathbf{n}) = [\hat{y}(n)\cdots\hat{y}(n-m+1)]^{\mathrm{T}}$ where $n_s = 4^{n_h+m-1}$. These output vectors are referred to as the desired channel states and are partitioned into four different classes $Y_{m,\tau}^{(j)}$, $(1 \le j \le 4)$, according to the value of $s(n-\tau)$. The number of states in $Y_{m,\tau}^{(1)}$, $Y_{m,\tau}^{(2)}$, $Y_{m,\tau}^{(3)}$ and $Y_{m,\tau}^{(4)}$ are denoted as n_s^1, n_s^2, n_s^3 and n_s^4 respectively. Due to the additive white Gaussian noise (AWGN), the channel outputs will form clusters around each of these desired channel states. Therefore, the noisy observation vector $\mathbf{y}(n) = [y(n)\cdots y(n-m+1)]^{\mathrm{T}}$ is a random process with a conditional Gaussian density function centered at each of the desired channel states [12]. The noisy observation vector is used as the input to the equalizer to determine the transmitted symbol $s(n-\tau)$.

Assume that the real and imaginary part of transmitted symbol $s(k)$ is equiprobable and independent sequences. Let λ be the *a priori* probability of $\mathbf{y}_i^{(j)}$, where $\mathbf{y}_i^{(j)} \in Y_{m,\tau}^{(j)}$, $(1 \le j \le 4)$. The conditional probability density function of $\mathbf{y(n)}$ given $s(n-\tau) = \alpha^{(j)}$ takes the following form [12]:

$$f_B^{(j)}(\mathbf{y}(n)) = \lambda \sum_{i=1}^{n_s^j} \exp\left(-\left(\mathbf{y}-\mathbf{y}_i^{(j)}\right)^{\mathrm{H}}\left(\mathbf{y}-\mathbf{y}_i^{(j)}\right)\Big/ 2\sigma_e^2\right) \quad 1 \le j \le 4 \quad (2)$$

where H (Hemitian) denotes the complex conjugate transposition. The optimal Bayesian equalizer solution is defined as

$$\begin{aligned} &\hat{s}(n-\tau) = \alpha^{(j)} \\ &\text{if } f_B^{(j)}(\mathbf{y}(n)) = \max\left\{f_B^{(j)}(\mathbf{y}(n)),\ 1 \le j \le 4\right\}. \end{aligned} \quad (3)$$

This equation also defines the optimum decision boundaries for the partition of equalizer inputs sets. It is clear that these

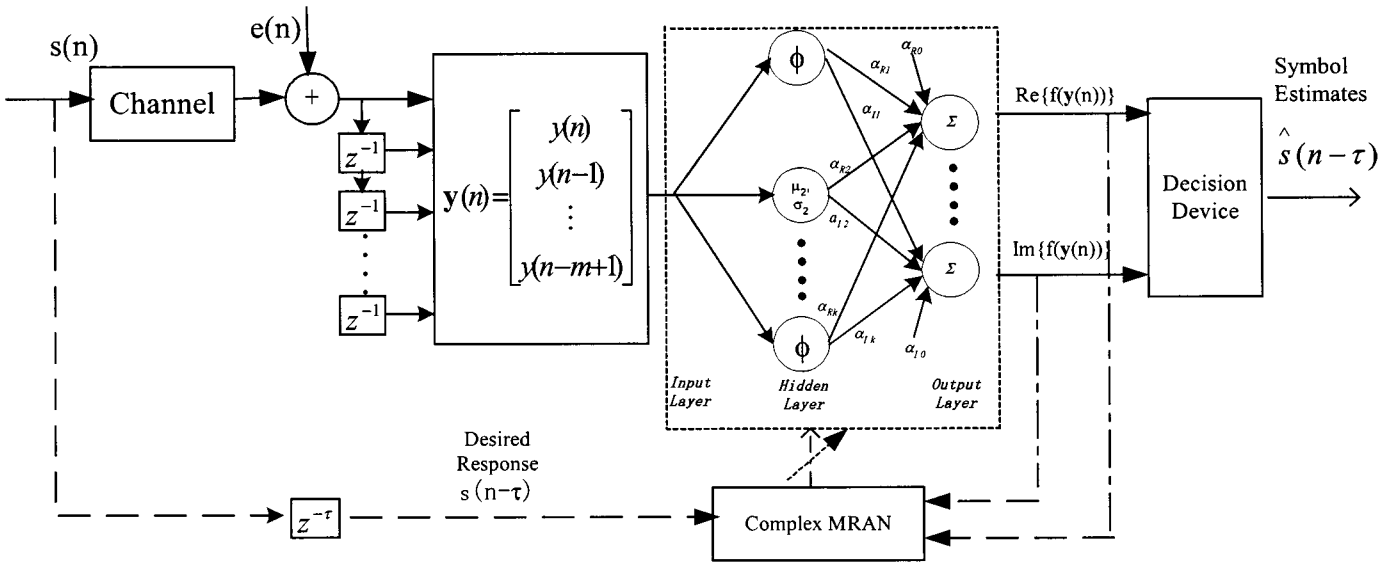

Fig. 1. CMRAN equalization scheme.

optimum decision boundaries are nonlinear hyper surfaces in the complex observation space and realizing such a nonlinear boundary will require the equalizer to have nonlinear mapping capabilities. Since RBF networks are known to naturally approximate the Bayesian equalizer [3], we have chosen a RBF network algorithm to build the equalizer. Since CMRAN is based on RBF network and produces a compact network structure, its application for QAM signal equalization is investigated in the following sections.

III. Complex Minimal Resource Allocation Network (CMRAN) Equalizer

The schematic diagram for the CMRAN equalizer is shown in Fig. 1. Using the noisy channel output vector $\mathbf{y}(n)$ as the network input and $s(n-\tau)$ as the desired output, CMRAN learning algorithm can readily be applied to train the complex RBF as shown by the dotted lines of Fig. 1. During the testing phase, the output $f(\mathbf{y}(n))$ of CMRAN network is fed into a nearest neighbor decision device to give an estimate $\hat{s}(n-\tau)$ of the transmitted symbol $s(n-\tau)$.

Defining $\mathbf{y}$ as the m-dimensional complex input vector and $\boldsymbol{\mu}_k$ as the m-dimensional complex center vector for the kth hidden neuron, the Euclidean distance $\|\bullet\|$ between the two complex-valued vectors is defined as

$$\begin{aligned}\|\mathbf{y}-\boldsymbol{\mu}_k\|^2 &= |y_1-\mu_{1k}|^2+|y_1-\mu_{2k}|^2+\cdots+|y_m-\mu_{mk}|^2\\ &= (\mathbf{y}-\boldsymbol{\mu}_k)^{\mathrm{H}}(\mathbf{y}-\boldsymbol{\mu}_k)\end{aligned} \tag{4}$$

where $|\bullet|$ denotes the modulus of a complex number. Therefore, the response of hidden neurons to the network input vector $\mathbf{y}$ can be expressed as follows:

$$\phi_k(\mathbf{y})=\exp\left(-\frac{1}{\sigma_k^2}(\mathbf{y}-\boldsymbol{\mu}_k)^{\mathrm{H}}(\mathbf{y}-\boldsymbol{\mu}_k)\right), \qquad (k=1,\ldots,h) \tag{5}$$

where σ_k is the (real valued) width of the Gaussian function and h indicates the total number of hidden neurons in the network.

The output layer of the RBF network is essentially a linear combiner. We define the coefficient α_k as the weight of the link connecting the kth hidden neuron to output neuron and α_0 as the bias term. In order to implement complex-valued outputs, we separate the complex valued α_k into α_{Rk} and α_{Ik} for the real and imaginary parts of the network output. Hence, the overall network response is a mapping $f\colon \mathbf{C}^P \to \mathbf{C}$ given by

$$\begin{aligned}f(\mathbf{y}) &= \left(\alpha_{R0}+\sum_{k=1}^{h}\alpha_{RK}\phi_k(\mathbf{y})\right)+j\left(\alpha_{I0}+\sum_{k=1}^{h}\alpha_{Ik}\phi_k(\mathbf{y})\right)\\ &= \alpha_0+\sum_{k=1}^{h}\alpha_k\phi_k(\mathbf{y})\end{aligned} \tag{6}$$

where $\mathbf{y} \in \mathbf{C}^p$. When both the network inputs and desired outputs are real-valued, this complex RBF network degenerates naturally into a real RBF network. An examination of (5) and (6) reveals that CRBF network treats the real and imaginary parts of an input as if they were two separate real inputs. Hence it is seen that the complex RBF network is a natural extension of the real RBF network which makes the CMRAN algorithm a natural extension of the MRAN algorithm.

The learning process of CMRAN involves allocation of new hidden neurons as well as adjusting network parameters, according to the data received. The network begins with no hidden neurons. As each input–output training data $(\mathbf{y}_n, s_{n-\tau})$ is received, the error $e_n = s_{n-\tau} - f(\mathbf{y}_n)$ is calculated and the network is built up based on three growth criteria described below. The criteria that must be met before a new hidden neuron is added are

$$(\mathbf{y}_n-\boldsymbol{\mu}_{nr})^{\mathrm{H}}(\mathbf{y}_n-\boldsymbol{\mu}_{nr}) > \varepsilon_n \tag{7}$$

$$(s_{n-\tau}-f(\mathbf{y}_n))^{\mathrm{H}}(s_{n-\tau}-f(\mathbf{y}_n)) > e_{\min} \tag{8}$$

where $\boldsymbol{\mu}_{nr}$ is the center (of the hidden neuron) which is closest to $\mathbf{y}_n$, the data that was just received. ε_n, $e_{\min}$ are thresholds

to be selected appropriately. Equation (7) ensures that the new neuron to be added is sufficiently far from all the existing neurons. Equation (8) decides if the existing neurons are insufficient to obtain a network output that meets the error specification. The algorithm begins with $\varepsilon_n = \varepsilon_{\max}$, the largest scale of interest, typically the size of the entire input space. The distance ε_n is decayed exponentially as

$$\varepsilon_n = \max\{\varepsilon_{\max}\gamma^n, \varepsilon_{\min}\}$$

where $0 < \gamma < 1$ is a decay constant. The value of ε_n is decayed until it reaches $\varepsilon_{\min}$. The exponential decaying of the distance criterion allows the algorithm to allocate fewer basis functions with larger widths (smoother basis functions) initially. With increasing number of observations, more basis functions with smaller widths are allocated to fine tune the approximation. In order to reduce the effect of noise in the network growth and also to make the transition of the number of the hidden neurons smooth, we include a third criterion, which uses the root mean square (rms) value of the output error over a sliding data window before adding a hidden neuron. The rms value of the network output error at nth observation e_{rmsn} is given by

$$e_{rmsn} = \sqrt{\sum_{i=n-(M-1)}^{n} \frac{e_i^* e_i}{M}}. \tag{9}$$

Then the extra growth criterion to be satisfied is

$$e_{rmsn} > e_{\min 1}. \tag{10}$$

It checks whether the network has met the required sum of squared error specification for the past M outputs of the network.

Only when all these three criteria are met, will a new hidden neuron be added to the network. Here $e_{\min 1}$ is a threshold value to be selected. When a new hidden neuron is added to the network, it will have the following parameters associated with it

$$\begin{aligned} \alpha_{h+1} &= e_n, \quad \boldsymbol{\mu}_{h+1} = \mathbf{y}_n \\ \sigma_{h+1}^2 &= \kappa(\mathbf{y}_n - \boldsymbol{\mu}_{nr})^{\mathrm{H}}(\mathbf{y}_n - \boldsymbol{\mu}_{nr}) \end{aligned} \tag{11}$$

where κ *is* a real valued overlap factor which determines the overlap of the responses of the hidden neurons in the input space. When an input to the network does not meet the criteria for adding a new hidden neuron, the network parameters

$$\begin{aligned} \mathbf{w} = [&\mathrm{Re}(\alpha_0), \mathrm{Im}(\alpha_0), \mathrm{Re}(\alpha_1), \mathrm{Im}(\alpha_1), \mathrm{Re}(\boldsymbol{\mu}_1^{\mathrm{T}}), \mathrm{Im}(\boldsymbol{\mu}_1^{\mathrm{T}}), \\ &\sigma_1, \ldots, \mathrm{Re}(\alpha_h), \mathrm{Im}(\alpha_h), \mathrm{Re}(\boldsymbol{\mu}_h^{\mathrm{T}}), \mathrm{Im}(\boldsymbol{\mu}_h^{\mathrm{T}}), \sigma_h]^{\mathrm{T}} \end{aligned} \tag{12}$$

are adapted using an extended Kalman filter (EKF) as follows:

$$\mathbf{w}_n = \mathbf{w}_{n-1} + \mathbf{k}_n \times [\mathrm{Re}(e_n), \mathrm{Im}(e_n)]^{\mathrm{T}}. \tag{13}$$

$\mathrm{Re}(\bullet)$ and $\mathrm{Im}(\bullet)$ represent the real part and imaginary part. In the EKF, a complex value is treated as a two-dimensional real vector. $\mathbf{k}_n$ is the Kalman gain vector given by

$$\mathbf{k}_n = \mathbf{P}_{n-1}\mathbf{a}_n[\mathbf{R}_n + \mathbf{a}_n^{\mathrm{T}}\mathbf{P}_{n-1}\mathbf{a}_n]^{-1}. \tag{14}$$

$\mathbf{a}_n$ is the gradient vector and has the following form:

$$\begin{aligned} \mathbf{a}_n = [&I_{2\times 2}, \phi_1(\mathbf{y}_n)I_{2\times 2}, \phi_1(\mathbf{y}_n)(2\beta_1/\sigma_1^2) \\ &\cdot [\mathrm{Re}(\mathbf{y}_n - \boldsymbol{\mu}_1)^{\mathrm{T}}, \mathrm{Im}(\mathbf{y}_n - \boldsymbol{\mu}_1)^{\mathrm{T}}] \\ &\cdot \phi_1(\mathbf{y}_n)(2\beta_1/\sigma_1^3)(\mathbf{y}_n - \boldsymbol{\mu}_1)^{\mathrm{H}}(\mathbf{y}_n - \boldsymbol{\mu}_1), \ldots, \\ &\phi_h(\mathbf{y}_n)I_{2\times 2}, \phi_h(\mathbf{y}_n)(2\beta_h/\sigma_h^2) \\ &\cdot [\mathrm{Re}(\mathbf{y}_n - \boldsymbol{\mu}_h)^{\mathrm{T}}, \mathrm{Im}(\mathbf{y}_n - \boldsymbol{\mu}_h)^{\mathrm{T}}] \\ &\cdot \phi_h(\mathbf{y}_n)(2\beta_h/\sigma_h^3)(\mathbf{y}_n - \boldsymbol{\mu}_h)^{\mathrm{H}}(\mathbf{y}_n - \boldsymbol{\mu}_h)]^{\mathrm{T}}. \end{aligned} \tag{15}$$

Here $\beta_1 = [\alpha_{R1}, \alpha_{I1}]^{\mathrm{T}}, \ldots, \beta_h = [\alpha_{Rh}, \alpha_{Ih}]^{\mathrm{T}}$, $\mathbf{R}_n$ is the covariance matrix of the measurement noise. $\mathbf{P}_n$ is the error covariance matrix which is updated by

$$\mathbf{P}_n = [\mathbf{I} - \mathbf{k}_n\mathbf{a}_n^{\mathrm{T}}]\mathbf{P}_{n-1} + Q\mathbf{I} \tag{16}$$

where Q is a scalar that determines the allowed random step in the direction of the gradient vector and $\mathbf{I}$ is an identity matrix. If the number of parameters to be adjusted is N then $\mathbf{P}_n$ is a $N \times N$ positive definite symmetric matrix. When a new hidden neuron is allocated, the dimensionality of $\mathbf{P}_n$ increases to

$$\mathbf{P}_n = \begin{pmatrix} \mathbf{P}_{n-1} & 0 \\ 0 & p_0\mathbf{I}_0 \end{pmatrix} \tag{17}$$

where p_0 initializes the new rows and columns. p_0 is an estimate of the uncertainty in the initial values assigned to the parameters. The dimension of the identity matrix $\mathbf{I}_0$ is equal to the number of new parameters introduced by the new hidden neuron.

The pruning strategy is also extended to a complex form in the algorithm. The hidden neurons that do not contribute significantly to the output of the network will be pruned. This is done by observing the output of each of the hidden neurons for a predefined period and then removing the neuron that has not been producing a significant output for that period. Consider the output o_k of the kth hidden neuron

$$o_k = \alpha_k \exp\left(-\frac{1}{(\sigma_k)^2}(\mathbf{y} - \boldsymbol{\mu}_k)^{\mathrm{H}}(\mathbf{y} - \boldsymbol{\mu}_k)\right). \tag{18}$$

If α_k or σ_k in the above equation is small, o_k might become small. Also, if $(\mathbf{y} - \boldsymbol{\mu}_k)^{\mathrm{H}}(\mathbf{y} - \boldsymbol{\mu}_k)$ is large, the output will be small. This would mean that the input is far away from the center of this hidden neuron. In any case a small o_k means that its real part and imaginary part are both small. To reduce the inconsistency caused by using the absolute values, both the real value and imaginary value of o_k are normalized with respect to the maximum (real and imaginary component of) output value among all the hidden neurons according to the following equation:

$$r_{Rk} = \frac{||o_{Rk}||}{||o_{R\max}||} \qquad r_{Ik} = \frac{||o_{Ik}||}{||o_{I\max}||}$$

where $||o_{R\max}||$ is the largest absolute real value of hidden neuron outputs. $||o_{I\max}||$ is the largest absolute imaginary value of hidden neuron outputs. If both r_{Rk} and r_{Ik} of a

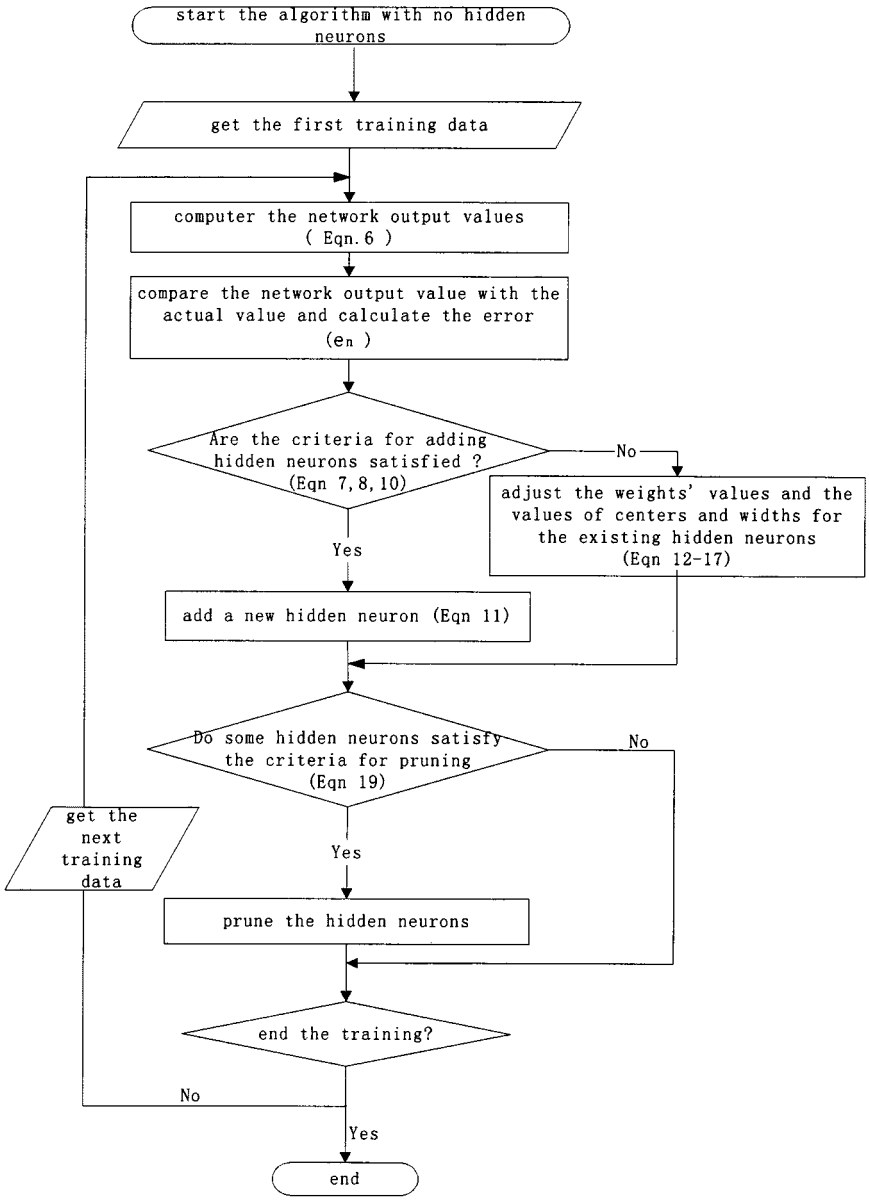

Fig. 2. Flow diagram for CMRAN learning algorithm.

normalized output $r_k(k \leq h)$ are less than a threshold, δ, for S_w consecutive observations

$$r_k < \delta \tag{19}$$

then the kth hidden neuron makes insignificant contribution to the network output and can be removed from the network. The dimensions of the EKF are then adjusted to suit the reduced network. Fig. 2 describes the steps involved in the CMRAN learning algorithm. The different threshold parameters of CMRAN are selected by tuning them largely by trial and error to get the best results for the training data. The tuned values are then used for the testing purposes.

IV. CMRAN Equalizer Performance Study

In this section we present the results of CMRAN on QAM channel equalization problems. The nonlinear channel models have been taken from [11]–[13]. Essentially, we have done a performance comparison on three channel model cases. The first case considered uses the nonlinear, nonminimum phase, real valued channel models from Patra *et al.* [11]. The second case considered is the linear, nonminimum phase, complex channel model, from Chen [12] whose decision boundaries are highly nonlinear. The third case is a nonlinear, nonminimum phase, complex channel model from Cha and Kassam [13]. Detailed performance studies using CMRAN for these models have been carried out and a comparison of CMRANs performance with the respective methods are analyzed.

A. Example 1: Real Nonlinear Channel (Patra's Model)

The normalized transfer function of the channel is given as

$$CH(z) = o(z)/s(z) = 0.447 + 0.894z^{-1}. \tag{20}$$

This is a nonminimum phase channel. Three different nonlinear distortions with increasing complexity with the following types of nonlinearities were introduced

$$NL=0: \quad y(k)=o(k) \tag{21}$$

$$NL=1: \quad y(k)=\tanh(o(k)) \tag{22}$$

$$NL=2: \quad y(k)=o(k)+0.2o^2(k)-0.1o^3(k) \tag{23}$$

$$NL=3: \quad y(k)=o(k)+0.2o^2(k)-0.1o^3(k)+0.5\cos(\pi(k)). \tag{24}$$

$NL=0$ corresponds to a linear channel model. $NL=1$ corresponds to a nonlinear channel which may occur in the channel due to saturation of amplifiers used in the transmitting system. $NL=2$ and $NL=3$ are two arbitrary nonlinear channel distortions.

A sequence of 4-QAM input symbols in the form $\{\pm 1 \pm j\}$ was generated, in which the real and imaginary part of the symbol were independently obtained from a uniform distribution. According to (20)–(24), the output sequences were generated. They are mixed with AWGN to get the desired SNR The equalizer order was three and the decision delay was set to two. For each case considered, 3000 training data samples at different SNRs were used to build up the CMRAN equalizers. After completion of the training, one million data samples with the same SNR as the training set were used to test the performance of the equalizers.

The values of the thresholds for the CMRAN algorithm were set as: $\varepsilon_{\max}=0.25$, $\gamma=1$, the size of the sliding window Sw is 40 and the pruning threshold $\delta=0.01$. The other parameters and the built up hidden neurons are as follows:

1) $NL=0$, $e_{\min}=0.6$, $e_{\min}1=1$, $M=40$; hidden neurons built up for SNR level of 10 dB, 12 dB, 14 dB, 15 dB are 8, 6, 7, 6, respectively.
2) $NL=1$, $e_{\min}=0.1$, $e_{\min}1=0.9$, $M=20$; hidden neurons built up for SNR level of 10 dB, 12 dB, 14 dB, 16 dB are 7, 8, 8, 8, respectively.
3) $NL=2$, $e_{\min}=0.1$, $e_{\min}1=0.8$, $M=20$; hidden neurons built up for SNR level of 10 dB, 12 dB, 14 dB, 16 dB are 6, 8, 6, 6, respectively.
4) $NL=3$, $e_{\min}=0.4$, $e_{\min}1=1$, $M=40$; hidden neurons built up for SNR level of 10 dB, 12 dB, 14 dB, 15 dB are 6, 8, 6, 5, respectively.

From the same example in [11], the performances of three popular ANN architectures i.e., an MLP, a polynomial perceptron network (PPN) and a functional link artificial neural network (FLANN) with BP algorithm, along with a linear LMS-based equalizer are reported. Out of them, FLANN equalizer, which has 19 and two neurons in the input and output layer, respectively, gave the best equalization results in terms of SER and MSE. The SER performance of CMRAN is plotted in Figs. 3–6 and compared with that of FLANN.

It is seen that CMRAN-based equalizers outperform FLANN-based equalizers for both linear and nonlinear channel models even when the nonlinearity becomes severe ($NL=3$).

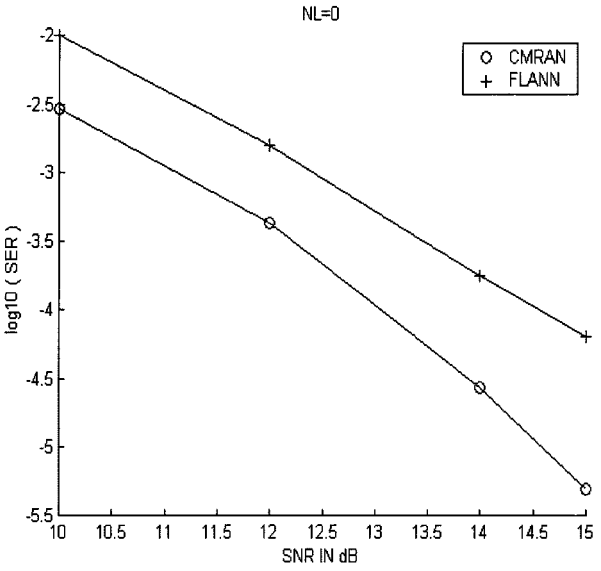

Fig. 3. SER performance, $NL=0$.

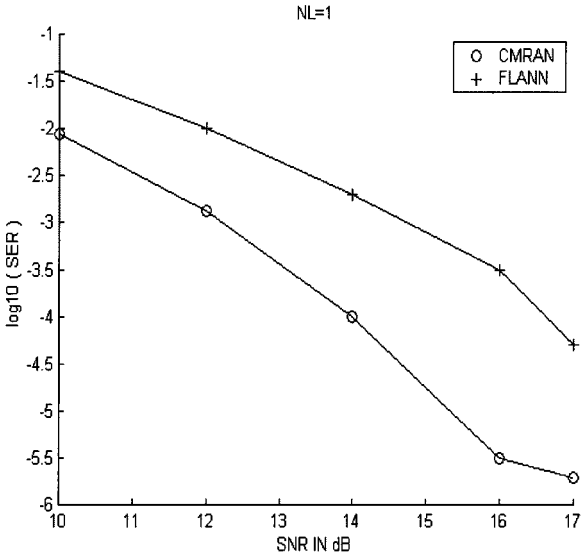

Fig. 4. SER performance, $NL=1$.

B. Example 2: Complex Linear Channel (Chen's Model)

This example [12] has been chosen because the linear channel is complex valued and the decision boundary is highly nonlinear compared to Example 1. The equalizer dimension is chosen as $m=1$. The decision delay was chosen to be $\tau=1$. The channel transfer function is given by

$$A(z)=(0.7409-j0.7406)(1-(0.2-j0.1)z^{-1})\cdot(1-(0.6-j0.3)z^{-1}). \tag{25}$$

In this example, channel order is $n_h=3$ and the noise variance is $\sigma_e^2=0.06324$ (SNR $=15$ dB). The real and imaginary part of the input sequence were restricted to be from the symbol set $\{+1,-1\}$. Thus, there will be 64 output states for the channel. Fig. 7 shows the channel states and the Bayesian decision boundary. It can be seen that Bayesian boundary partitions the observation space into four decision regions each corresponding to an input symbol.

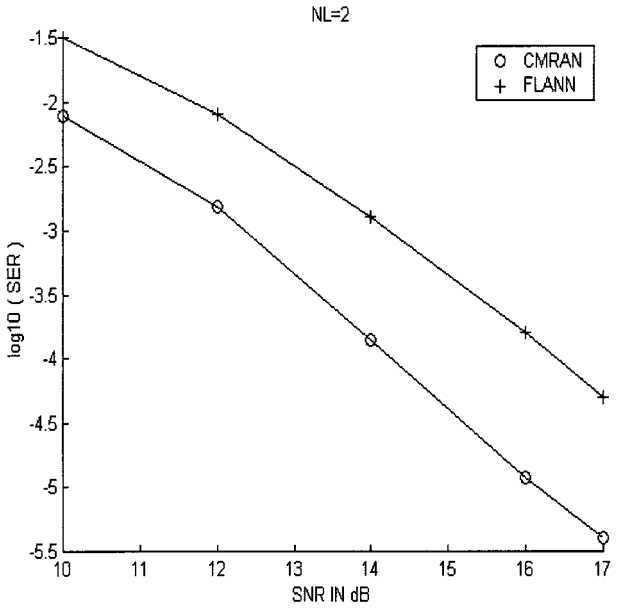

Fig. 5. SER performance, $NL = 2$.

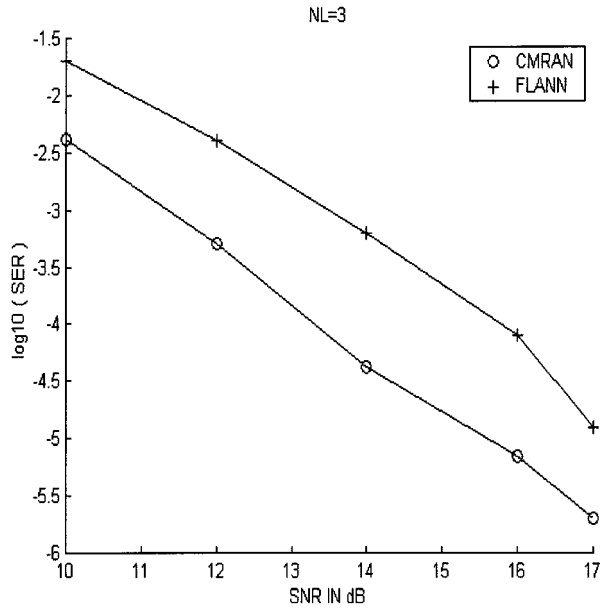

Fig. 6. SER performance, $NL = 3$.

1200 data samples at 15 dB SNR are used to train the CMRAN. The parameters of CMRAN are set as: $e_{\min} = 0.1, e_{\min 1} = 0.1, \varepsilon_{\max} = 0.25, \gamma = 1$. The size of the two sliding windows M and Sw are ten and 80 respectively, and the pruning threshold $\delta = 0.01$. The decision boundary generated by CMRAN using only 52 hidden neurons is shown in Fig. 8. Comparing the CMRAN boundary with the Bayesian of Fig. 7, it can be seen that the Bayesian boundary is well approximated at the critical region, which is the center portion of the figure. Fig. 9 shows the growth of the hidden neurons in CMRAN as training progresses. The network has built up 52 hidden neurons at the end of the training. This is much less than the 64 hidden neurons if a hidden neuron is to be placed in each of the desired channel states. 10^5 test data of various SNR were used to test the resulting CMRAN equalizer.

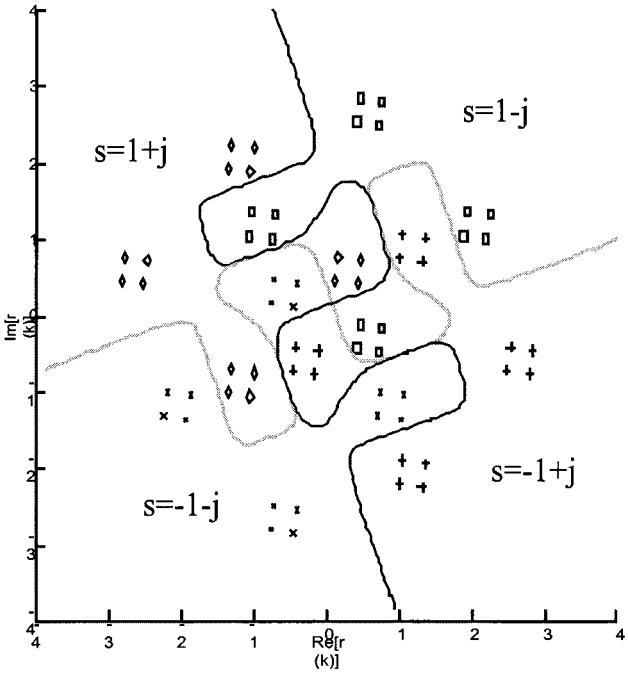

Fig. 7. Bayesian boundary (with 64 states).

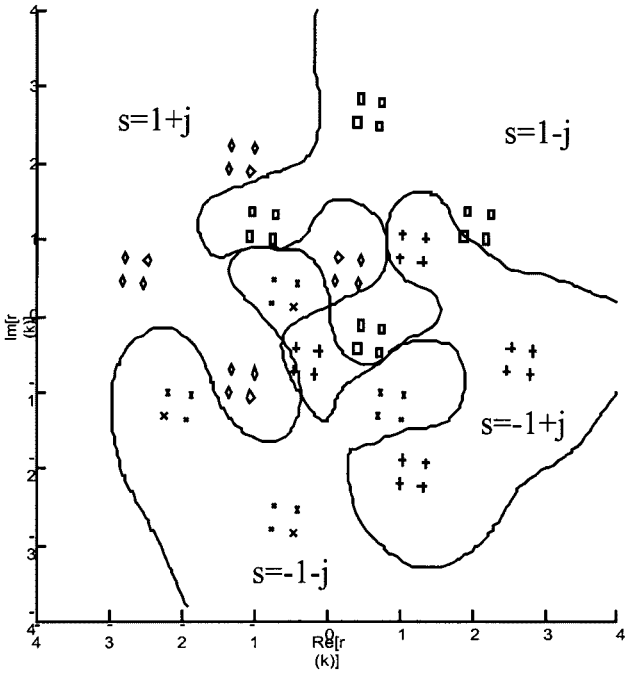

Fig. 8. Complex MRAN boundary (with 52 neurons).

A comparison of their symbol error rate (SER) curves in Fig. 10 show that CMRAN equalizer performs only slightly poorer than the ideal Bayesian equalizer.

An earlier study [17] has used the same channel model but with a equalizer delay of $\tau = 0$ which resulted in a different decision boundary. The same example is used here with a equalizer delay of $\tau = 1$ to illustrate the point that even with a simple linear model the decision boundary can be highly nonlinear and the CMRAN equalizer has done a good job in approximating the Bayesian decision boundary.

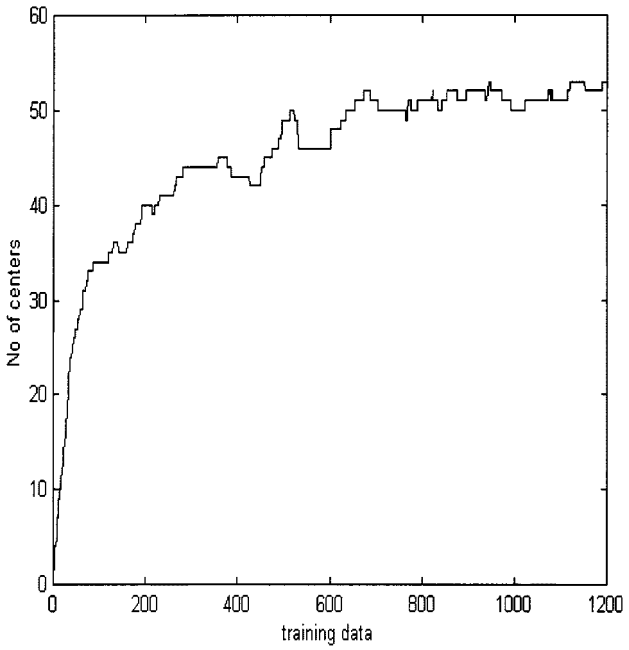

Fig. 9. Growth of hidden neurons as training processes.

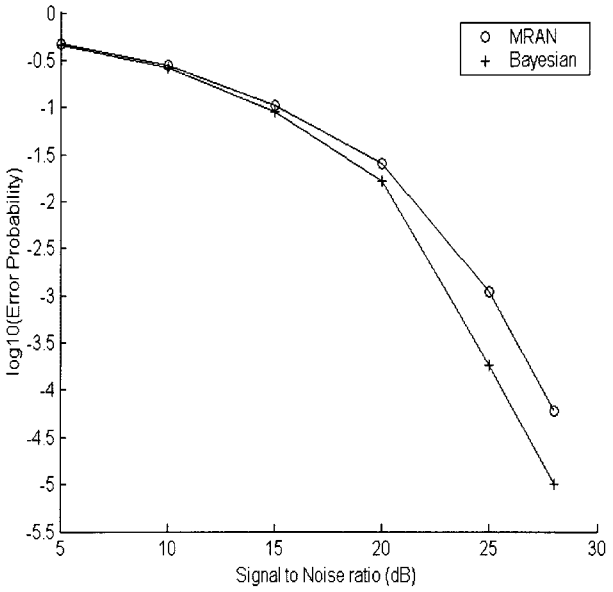

Fig. 10. Error curves for CMRAN and Bayesian equalizer.

C. Example 3: Complex Nonlinear Channel (Cha and Kassam's Model)

To evaluate CMRANs performance for a highly complex nonlinear channel the following model [13] is used

$$y_n = o_n + 0.1o_n^2 + 0.05o_n^3 + \nu_n; \qquad \nu_n \sim \mathrm{N}(0,\, 0.01) \quad (26)$$

$$o_n = (0.34 - j0.27)s_n + (0.87 + j0.43)s_{n-1} + (0.34 - j0.21)s_{n-2}. \quad (27)$$

The Gaussian/SG equalizer by Cha and Kassam had 30 hidden neurons. The equalizer order and delay were chosen to be $m = 3$ and $\tau = 1$, respectively. Both the real and imaginary parts of data in the training sequence are values from the set ± 0.7.

The CMRAN parameters are set as: $e_{\min} = 0.01$, $e_{\min 1} = 0.5$, $\varepsilon_{\max} = 0.16$, $\gamma = 1$, the size of the two sliding windows M and Sw are ten and 50, respectively, pruning threshold $\delta = 0.01$. After 4000 training samples, the resulting CMRAN equalizer had 18 hidden neurons. For this case the eye diagram and SER curves are presented for comparison. Fig. 11 shows the noisy channel output distribution. Fig. 12 shows the CMRAN equalizer output distribution. Fig. 13 shows the output of Cha and Kassam's equalizer, which had been built up by SG algorithm using 14 000 training samples. It is seen that CMRAN equalizer can clearly separate the initial data into four zones, while the SG equalizer has difficulty in separating the data in the right-hand zone. Also it should be noted that the separation of the clusters for CMRAN equalizer is shown after training with only 4000 samples whereas for SG equalizer it is shown after 14 000 samples.

Test sets with 100 000 samples at various SNRs were used for the error probability evaluation. A comparison of probability of error curves is shown in Fig. 14. With only 18 hidden neurons, the CMRAN equalizer performs better than the SG equalizer with 30 hidden neurons. Although the performance of CMRAN equalizer cannot match with that of the optimal Bayesian equalizer, which has 1024 channel states (which implies 1024 hidden neurons for the RBF network), the CMRAN equalizer structure with 18 hidden neurons is compact.

D. CMRAN Equalizer Complexity

After discussing the performance of CMRAN and SG equalizers in the earlier sections, a comparison of the training time and testing time for both the equalizers are in order to evaluate their complexities. Example 3 has been selected for this comparison.

1000 data symbols with a SNR equal to 21 dB were used to train the CMRAN equalizer. In order to achieve the same accuracy (in terms of SER) of the CMRAN equalizer, 14 000 samples were needed for the Gaussian/SG equalizer. Training time was obtained by averaging over three independent runs. In the testing process, the time for 100 000 data symbols were averaged to get the equalization time for each test data.

Table I shows the CPU time for training CMRAN and Gaussian/SG equalizers on a Pentium III/500 personal computer. It can be seen from the Table I that CMRAN takes 73% lesser training time compared to the SG scheme. It should be noted here that CMRAN takes more time to process each training data but requires less training samples for convergence.

Table II presents the equalization time for CMRAN, SG equalizer and Bayesian equalizer for each test data for the example 3. It can be seen from the table that CMRAN takes 90% less time than the Bayesian equalizer and 36% less time than the SG equalizer.

The use of EKF in CMRAN requires much computing power and memory. Although its use leads to faster convergence of network parameters, there is a trade-off involved in that; as the network size grows the computational time for processing each training data increases. This is due to the large size of the matrices that are manipulated. To overcome this problem, recently, EMRAN algorithm has been proposed as an extension to the MRAN algorithm [18].

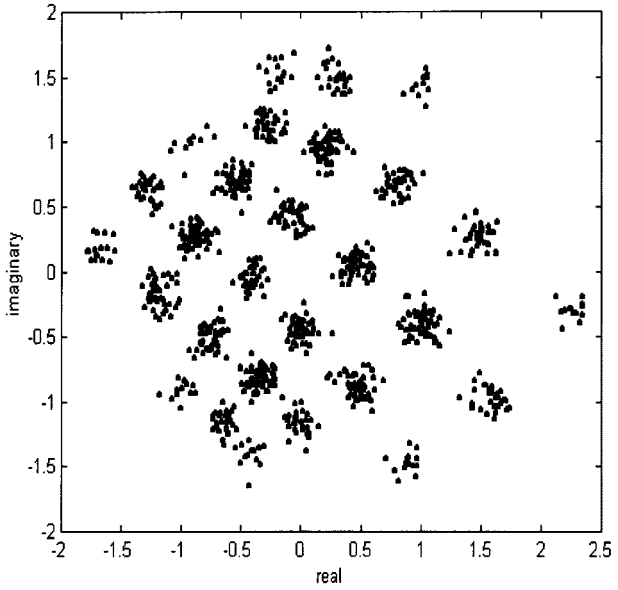

Fig. 11. Test set channel output distribution.

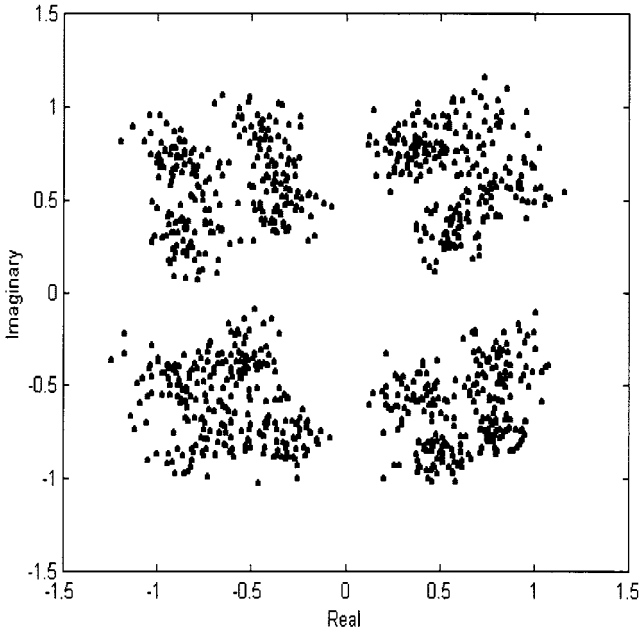

Fig. 12. CMRAN equalizer output (after 4000 training iterations).

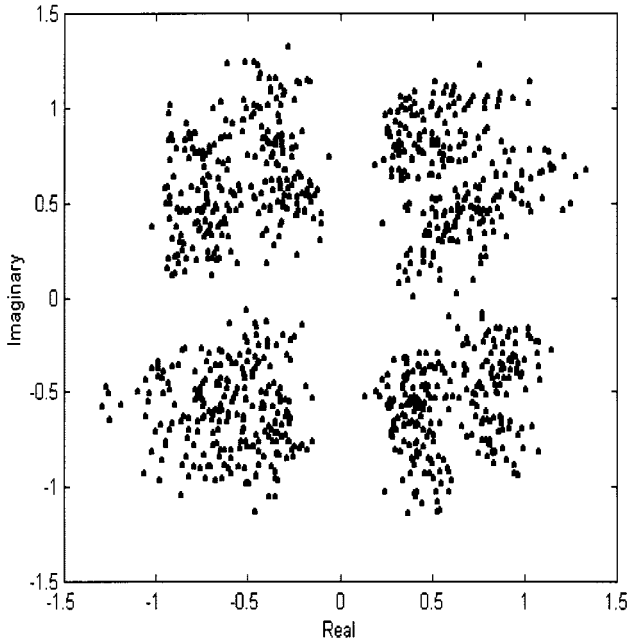

Fig. 13. CRBF/SG equalizer output (after 14 000 training iterations).

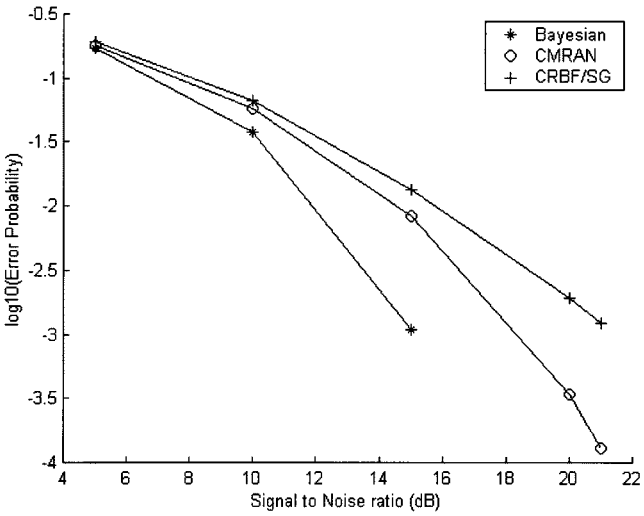

Fig. 14. Error curves for CMRAN, CRBF/SG, and Bayesian equalizer.

It is worth noting that the threshold values for CMRAN algorithm and also the data window size are to be chosen properly. As have been discussed in [15], when no measurement noise is assumed, the selection of the threshold is more critical. As in the most learning algorithms, we use some initial samples to obtain the information for selecting these parameters.

V. Conclusion

This paper has presented the performance of the recently developed CMRAN for the equalization of QAM signals. The equalizer's performance has been evaluated by using it to build up equalization networks for two complex-valued communication channels (linear and nonlinear) and a real-valued nonlinear channel. The CMRAN networks were then tested by comparing

TABLE I
CPU Time for CMRAN and SG Equalizers for Training

	MRAN (1000 training samples)	SG (14000 training samples)
Example 3	118 (sec)	443 (sec)

TABLE II
CPU Time for CMRAN, Gaussian/SG and Optimal Bayesian Equalizers for Each Test Data

	Hidden neurons	Test Time (msec)
CMRAN	18	0.725
Gaussian/SG	30	1.131
Bayesian	1024	7.684

their SER performance to that of the ideal Bayesian equalizer, and other existing equalizer networks. Simulation results presented clearly show that the CMRAN algorithm is able to build up a network that can perform equalization better than several existing methods. There is no need to estimate the exact number and locations of the noise-free equalizer input states, which is a prerequisite for Bayesian approach. Because CMRAN has reduced the network complexity compared to the other methods, the time taken for equalization is also lower.

ACKNOWLEDGMENT

The authors wish to express their thanks to Dr. J. Patra for many helpful discussions on Example 1 and also the reviewers for their critical comments.

REFERENCES

[1] J. G. Proakis, *Digital Communication*. New York: McGraw-Hill, 1983.
[2] S. Chen, G. J. Gibson, C. F. N. Cowan, and P. M. Grant, "Adaptive equalization of finite nonlinear channels using multilayer perceptrons," *Signal Processing*, vol. 20, pp. 107–119, 1990.
[3] S. Chen, B. Mulgrew, and P. M. Grant, "A clustering technique for digital communications channel equalization using radial basis function networks," *IEEE Trans. Neural Networks*, vol. 4, pp. 570–579, July 1993.
[4] G. Kechriotis, E. Zervas, and E. S. Manolakos, "Using recurrent neural networks for adaptive communication channel equalization," *IEEE Trans. Neural Networks*, vol. 5, Mar. 1994.
[5] J. D. Provence, "Neural network implementation for an adaptive maximum-likelihood receiver," in *Proc. IEEE Int. Symp. Circuits Syst.*, 1988, pp. 2381–2385.
[6] S. Bang, S. H. Sheu, and J. Bing, "Neural network for detection of signals in communication," *IEEE Trans. Circuits Syst. I*, pp. 644–655, Aug. 1996.
[7] S. Siu, G. J. Gibson, and C. F. N. Cowan, "Decision feedback equalization using neural network structures," in *Proc. 1st IEE Int. Conf. Artificial Neural Networks*, 1989, pp. 125–128.
[8] T. Fechner, "Nonlinear noise filtering with neural networks: Comparison with Weiner optimal filtering," in *Proc. 3rd IEE Int. Conf. Artificial Neural Networks*, 1993, No 373, pp. 143–147.
[9] A. Uncini, L. Vecci, P. Campolucci, and F. Piazza, "Complex-valued neural networks with adaptive spline activation function for digital radio links nonlinear equalization," *IEEE Trans. Signal Processing*, vol. 47, pp. 505–514, Feb. 1999.
[10] H. Leung and S. Haykin, "The complex backpropagation algorithm," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-39, pp. 2101–2104, Sept. 1991.
[11] J. C. Patra, R. N. Pal, R. Baliarsingh, and G. Panda, "Nonlinear channel equalization for QAM constellation using artificial neural networks," *IEEE Trans. Syst., Man, Cybern. B*, vol. 29, pp. 262–271, Apr. 1999.
[12] S. Chen, S. Mclaughlin, and B. Mulgrew, "Complex-valued radial basis function network, Part II: Application to digital communications channel equalization," *Signal Processing*, vol. 36, pp. 165–188, Feb. 1994.
[13] I. Cha and S. Kassam, "Channel equalization using adaptive complex radial basis function networks," *IEEE J. Select. Areas Commun.*, vol. 13, pp. 122–131, Jan. 1995.
[14] L. Yingwei, N. Sundararajan, and P. Saratchandran, "Performance evaluation of a sequential minimal Radial Basis Function (RBF) neural network learning algorithm," *IEEE Trans. Neural Networks*, vol. 9, pp. 308–318, Mar. 1998.
[15] ——, "A sequential learning scheme for function approximation using minimal radial basis function neural networks," *Neural Comput.*, vol. 9, pp. 461–478, Feb. 1997.
[16] C. P. Kumar, P. Saratchandran, and N. Sundararajan, "Nonlinear channel equalization using minimal radial basis function neural networks," *Proc. Inst. Elect. Eng. Vision, Image, Signal Processing*, vol. 147, no. 5, pp. 428–435, 2000.
[17] D. Jianping, N. Sundararajan, and P. Saratchandran, "Complex-valued minimal radial basis function neural network for nonlinear system processing," *Int. J. Neural Syst.*, vol. 10, no. 2, pp. 95–106, 2000.
[18] L. Yan, N. Sundararajan, and P. Saratchandran, "Analysis of minimal radial basis function network algorithm for real-time identification of nonlinear dynamic systems," *Proc. Inst. Elect. Eng. Contr. Theory Applicat.*, vol. 147, no. 4, pp. 476–484, July 2000.

Deng Jianping received the B.Eng. degree in 1993 from North China University of Electric Power, China, and the Master's degree in electrical and electronic engineering from Nanyang Technological University, Singapore, in 2001.

She is currently with the Center for Signal Processing at Nanyang Technological University.

Narasimhan Sundararajan (S'69–M'71–SM'84–F'96) received the B.E. degree in electrical engineering with First Class Honors from the University of Madras, India, in 1966, the M.Tech. degree from the Indian Institute of Technology, Madras, in 1968, and the Ph.D. degree in electrical engineering from the University of Illinois, Urbana-Champaign, in 1971.

From 1972 to 1991, he has worked at the Indian Space Research Organization and NASA on aerospace problems. Since February 1991, he has been with the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore, where currently he is a Professor.

Dr. Sundararajan is an Associate Fellow of AIAA. He is an Associate Editor for IEEE TRANSACTIONS ON CONTROL SYSTEMS TECHNOLOGY and *IFAC Journal on Control Engineering Practice* (CEP). His research interests are in the areas of neural networks and control with aerospace applications. He has published more than 100 papers and four books in the area of neural networks.

P. Saratchandran (M'87–SM'96) received the B.Sc.(Eng.) degree from Regional Engineering College, Calicut, India, and the M.Tech. degree from the Indian Institute of Technology, Kharagpur, both in electrical engineering. He received the M.Sc. degree in systems engineering from the City University, London, U.K., and the Ph.D. degree in the area of control engineering from Oxford University, Oxford, U.K.

He worked for two years as a scientist with the Indian Space Research Organization and spent five years as a Senior Design Engineer with the Hindustan Aeronautics Ltd.; India, designing defense related avionics systems. From 1984 to 1990, he worked in Australia for various defense industries as a Systems Consultant and Manager developing real-time software/systems in Ada for the Australian Defense Forces. During this period, he was also a Visiting Fellow at the Department of Mathematics and Computer Science at the Macquarie University in Sydney,Australia. Since 1990, he is with the Nanyang Technological University, where he is now an Associate Professor. He has several publications in refereed journals and has authored four books titled *Fully Tuned Radial Basis Function Neural Networks for Flight Control* (Boston, MA: Kluwer, 2001) *Radial Basis Function Neural Networks With Sequential Learning* (Singapore: World Scientific, 1999), *Parallel Architectures for Artificial Neural Networks* (Los Alamitos, CA: IEEE Computer Society Press, 1998) and *Parallel Implementations of Backpropagation Neural Networks* (Singapore: World Scientific, 1996). His interests are in neural networks, parallel computing and control.

Dr. Saratchandran is an Editor for the journal *Neural Parallel and Scientific Computations*. He is listed in the Marquis *Who's Who in the World* and in the *Leaders in the World* of the International Biographics Centre, Cambridge, U.K.