International Journal of Neural Systems, Vol. 10, No. 2 (April, 2000) 95–106
© World Scientific Publishing Company

COMPLEX-VALUED MINIMAL RESOURCE ALLOCATION NETWORK FOR NONLINEAR SIGNAL PROCESSING

DENG JIANPING, N. SUNDARARAJAN and P. SARATCHANDRAN
School of Electrical and Electronic Engineering,
Nanyang Technological University, Singapore 639798
E-mail: ENSUNDARA@ntu.edu.sg

Received 2 November 1999
Revised 25 January 2000
Accepted 25 January 2000

This paper presents a sequential learning algorithm and evaluates its performance on complex valued signal processing problems. The algorithm is referred to as Complex Minimal Resource Allocation Network (CMRAN) algorithm and it is an extension of the MRAN algorithm originally developed for online learning in real valued 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 learning algorithm is illustrated using two applications from signal processing of communication systems. The first application considers identification of a nonlinear complex channel. The second application considers the use of CMRAN to QAM digital channel equalization problems. Simulation results presented clearly show that CMRAN is very effective in modeling and equalization with performance achieved often being superior to that of some of the well known methods.

1. Introduction

In the past decade, artificial neural networks have ever-increasingly been applied to many fields in signal processing including pattern classification and recognition, communications channel equalization and nonlinear systems modeling and identification.[1,2] Of the various types of networks being developed, Radial Basis Function (RBF) neural networks have been gaining special attention in these fields due to their simple topological structure and their ability to reveal how learning proceeds in an explicit manner.[3] Presently, most of the neural networks used are real-valued and are suitable for signal processing applications in real (multidimensional) space.

However, in some applications the signals are complex-valued and processing has to be done in complex multidimensional space, e.g., equalization of digital communications channels with Quadrature Amplitude Modulation (QAM) scheme. For these complex-valued signal processing problems, in order to preserve the concise formulation and elegant structure of the complex signals, complex neural networks are proposed. Some researchers have come up with a complex multilayer perceptron (MLP) and extended the backpropagation algorithm to a complex form.[4] Like its real counterpart, the complex multilayer perceptron is highly nonlinear in the parameters and suffers drawbacks of slow convergence and unpredictable solutions during learning. Chen *et al.*[5] and Inhyok *et al.*[6] have independently proposed the 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. The centers of hidden nodes are complex vectors in the input space and the nonlinearity function of hidden node is a real radially symmetric response around the hidden node center. The output layer consists of a set of linear

combiners with complex connection weights. It is known[5] that this kind of network can carry out an approximation for the nonlinear mapping from the complex multidimensional input space to the complex multidimensional output space. Several existing learning algorithms for the real RBF network have been extended to the complex RBF network.[5,6]

Recently, a new RBF network learning algorithm called Minimal Resource Allocation Network (MRAN) was developed by Yingwei *et al.*,[7–9] which is an improvement of the resource allocation Network (RAN) of Platt[10] and the RAN via Extended Kalman Filter (RANEKF) algorithm of Kadirkamanathan *et al.*[11] This sequential learning MRAN algorithm employs a scheme for adding and pruning RBF's hidden units, so as to achieve a parsimonious network structure. Other methods of updating the centers and the parameters of RBF network on-line have been presented by Luo and Billings in Refs. 12 and 13. However, in their method the initialization of the network centers is crucial for the performance and is done using "k-means clustering" algorithm.

A number of successful application of MRAN in areas such as nonlinear system identification, function approximation and time series prediction have been reported in Refs. 7–9. In this paper, the MRAN learning algorithm is extended to complex form (Complex MRAN or CMRAN) and the performance of the CMRAN algorithm is investigated for nonlinear signal processing problems. Simulation results suggest that CMRAN is very useful in these applications with performance achieved often being superior to that of some well-known existing methods.

The paper is organized as follows. Section 2 describes the Complex RBF network structure and the CMRAN algorithm. In Sec. 3, the performance of CMRAN algorithm is evaluated on the identification of a complex-valued nonlinear communication channel. The application of CMRAN for equalization problems is investigated in Sec. 4. Equalization results for one linear and two nonlinear complex channels with QAM signals are presented. Conclusions from this study are summarized in Sec. 5.

2. Complex Minimal Resource Allocation Network (CMRAN)

Figure 1 shows the structure of a complex-valued

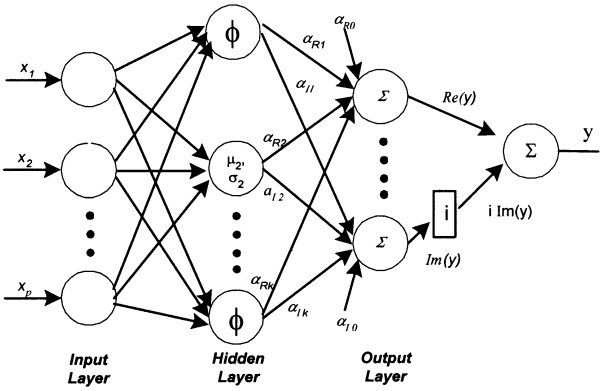

Fig. 1. Topology of complex RBF (CRBF) network.

RBF (CRBF) network with real Gaussian basis function. For simplicity of illustration of the algorithm, a single output (with multiple input) network is considered. For networks with multiple outputs, the algorithm can be extended in a straight forward manner.

Defining $\mathbf{x}$ as the p-dimensional complex input vector and $\boldsymbol{\mu}_k$ as the p-dimensional complex center vector for the kth hidden unit, the Euclidean distance $\| \bullet \|$ between the two complex-valued vectors is defined as[14]:

$$\begin{aligned}\|\mathbf{x}-\boldsymbol{\mu}_k\|^2 &= |x_1-\mu_{1k}|^2+|x_1-\mu_{2k}|^2+\cdots \\ &\quad +|x_p-\mu_{pk}|^2 \\ &= (\mathbf{x}-\boldsymbol{\mu}_k)^H(\mathbf{x}-\boldsymbol{\mu}_k) \end{aligned} \quad (1)$$

where $|\bullet|$ denotes the modulus of a complex number and H denotes the complex conjugate transposition (The operator $(\bullet)^H = ((\bullet)^T)^*$). Therefore, the response of hidden units to the network input vector $\mathbf{x}$ can be expressed as follows:

$$\phi_k(\mathbf{x}) = \exp\left(-\frac{1}{(\sigma_k)^2}(\mathbf{x}-\boldsymbol{\mu}_k)^H(\mathbf{x}-\boldsymbol{\mu}_k)\right), \quad (k=1,\ldots,h) \quad (2)$$

where $\boldsymbol{\sigma}_k$ 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 unit to output unit 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 $f : \mathbf{C}^{\mathrm{P}} \to \mathbf{C}$ given by

$$\begin{aligned} f(\mathbf{x}) &= \left(\alpha_{R0} + \sum_{k=1}^{h} \alpha_{Rk}\phi_k(\mathbf{x})\right) \\ &\quad + j\left(\alpha_{I0} + \sum_{k=1}^{h} \alpha_{Ik}\phi_k(\mathbf{x})\right) \\ &= \alpha_0 + \sum_{k=1}^{h} \alpha_k \phi_k(\mathbf{x}) \end{aligned} \tag{3}$$

where $\mathbf{x} \in \mathbf{C}^p$. When both network inputs and desired outputs are reduced to be real-valued, this complex RBF network degenerates naturally into a real RBF network. An examination of (2) and (3) reveals that CRBF network treats the real and imaginary parts of an input as if they were two separate real inputs. Inhyok *et al.*[6] have shown that CRBF with P complex inputs and a complex output can be viewed alternatively as a real RBF with 2P real inputs and two real outputs. It can approximate any continuous mapping in complex domain. Hence, it is seen that the Complex RBF network is a natural extension of the real RBF network. The CMRAN algorithm is also a natural extension of the real MRAN algorithm.

In the CMRAN algorithm, the network begins with no hidden units. As each input-output training data $(\mathbf{x}_n, y_n)$ is received, the error $e_n = y_n - f(\mathbf{x}_n)$ is calculated and the network is built up based on certain growth criteria. The algorithm adds hidden units, as well as adjusts the existing network, according to the data received. The criteria that must be met before a new hidden unit is added are:

$$(\mathbf{x}_n - \boldsymbol{\mu}_{nr})^H(\mathbf{x}_n - \boldsymbol{\mu}_{nr}) > \varepsilon_n \tag{4}$$

$$(y_n - f(\mathbf{x}_n))^H(y_n - f(\mathbf{x}_n)) > e_{\min} \tag{5}$$

$$e_{rmsn} = \sqrt{\sum_{i=n-(M-1)}^{n} \frac{e_i^* e_i}{M}} > e_{\min 1} \tag{6}$$

where $\boldsymbol{\mu}_{nr}$ is the center (of the hidden unit) which is closest to $\mathbf{x}_n$, the data that was just received. ε_n, $e_{\min}$ and $e_{\min 1}$ are thresholds to be selected appropriately. Equation (4) ensures that the new node to be added is sufficiently far from all the existing nodes. Equation (5) decides if the existing nodes are insufficient to obtain a network output that meets the error specification. Equation (6) checks whether the network has met the required sum squared error specification for the past M outputs of the network. Only when all these criteria are met, will a new hidden node be added to the network. Each new hidden node added to the network will have the following parameters associated with it:

$$\alpha_{h+1} = e_n\,, \quad \boldsymbol{\mu}_{h+1} = \mathbf{x}_n\,,$$
$$\sigma_{h+1}^2 = \kappa(\mathbf{x}_n - \boldsymbol{\mu}_{nr})^H(\mathbf{x}_n - \boldsymbol{\mu}_{nr})$$

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

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

are adapted using an Extended Kalman Filter (EKF) as follows:

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

$\mathrm{Re}(\bullet)$ and $\mathrm{Im}(\bullet)$ represent the real part and Imaginary part. In the EKF, a complex value is treated as a 2-dimensional real vector. $\mathbf{k}_n$ is the Kalman gain vector given by

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

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

$$\mathbf{a}_n = \begin{bmatrix} I_{2\times 2},\ \phi_1(\mathbf{x}_n)I_{2\times 2},\ \phi_1(\mathbf{x}_n)(2\boldsymbol{\beta}_1/\sigma_1^2)[\mathrm{Re}(\mathbf{x}_n - \boldsymbol{\mu}_1)^T, \mathrm{Im}(\mathbf{x}_n - \boldsymbol{\mu}_1)^T], \\ \phi_1(\mathbf{x}_n)(2\boldsymbol{\beta}_1/\sigma_1^3)(\mathbf{x}_n - \boldsymbol{\mu}_1)^{\mathbf{H}}(\mathbf{x}_n - \boldsymbol{\mu}_1), \ldots, \\ \phi_h(\mathbf{x}_n)I_{2\times 2},\ \phi_h(\mathbf{x}_n)(2\boldsymbol{\beta}_h/\sigma_h^2)[\mathrm{Re}(\mathbf{x}_n - \boldsymbol{\mu}_h)^T, \mathrm{Im}(\mathbf{x}_n - \boldsymbol{\mu}_h)^T], \\ \phi_h(\mathbf{x}_n)(2\boldsymbol{\beta}_h/\sigma_h^3)(\mathbf{x}_n - \boldsymbol{\mu}_h)^{\mathbf{H}}(\mathbf{x}_n - \boldsymbol{\mu}_h) \end{bmatrix}^T$$

here $\boldsymbol{\beta}_1 = [\alpha_{R1}, \alpha_{I1}]^T, \ldots, \boldsymbol{\beta}_h = [\alpha_{Rh}, \alpha_{Ih}]^T$, $\mathbf{R}_n$ is the variance of the measurement noise. $\mathbf{P}_n$ is the error covariance matrix which is updated by,

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

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 an $N \times N$ positive definite symmetric matrix. When a new hidden unit is allocated, the dimensionality of $\mathbf{P}_n$ increases to

$$\mathbf{P}_n = \begin{pmatrix} \mathbf{P}_{n-1} & 0 \\ 0 & P_0\mathbf{I}_0 \end{pmatrix}$$

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 unit.

The algorithm also incorporates a pruning strategy which is used to prune hidden nodes that do not contribute significantly to the output of the network. This is done by observing the output of each of the hidden nodes for a pre-defined period and then removing the node that has not been producing a significant output for that period. Consider the output o_k of the kth hidden unit:

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

If α_k or σ_k in the above equation is small, o_k might become small. Also, if $(\mathbf{x} - \boldsymbol{\mu}_k)^H(\mathbf{x} - \boldsymbol{\mu}_k)$ is large, the output will be small. This would mean that the input is far away from the center of this hidden unit. In any case, a small o_k means that its real part and imaginary part are both small. To reduce 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:

$$r_{Rk} = \frac{\|o_{Rk}\|}{\|o_{R\,\max}\|} \quad r_{Ik} = \frac{\|o_{Ik}\|}{\|o_{\mathrm{Im}\,ax}\|}. \tag{7}$$

$\|o_{R\,\max}\|$ is the largest absolute real value of hidden unit outputs. $\|o_{\mathrm{Im}\,ax}\|$ is the largest absolute imaginary value of hidden unit outputs. If both r_{Rk} and r_{Ik} of a normalized output $r_k (k \leq h)$ are less than a threshold, δ, for S_w consecutive observations, then it indicates that 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.

3. Identification of a Nonlinear Channel Model

In this section, the performance of the complex-valued MRAN is evaluated using an example of modeling a complex-valued nonlinear communication channel.

The example considered is same as the channel used by Chen *et al.* in Ref. 5. The transmitted digital signal $s(t) = s_R(t) + js_I(t)$ are 4-QAM symbols, $s_R(t)$ and $s_I(t)$ are the real and imaginary parts of $s(t)$. Both $s_R(t)$ and $s_I(t)$ can only take values from the symbol set $\{\pm 1\}$. The nonlinear channel model comprises three elements in cascade and the transfer functions of the elements are as follows:

$$u(t) = f_s(s(t)) = \frac{2s(t)}{1 + |s(t)|^2} \exp\left(j\frac{\pi}{3}\frac{|s(t)|}{1 + |s(t)|^2}\right)$$

$$A(z) = \frac{v(z)}{u(z)} = (0.3725 + j0.2172) \times (1 - (0.35 + j0.7)z^{-1}) \times (1 - (0.5 + j)z^{-1})$$

$$\hat{y}(t) = f_v(v(t)) = v(t) + 0.2v^2(t) - 0.1v^3(t)$$

$$y(t) = \hat{y}(t) + e(t); \quad e \sim N(0, 2\sigma_e^2).$$

The additive noise is $e(t) = e_R(t) + je_I(t)$, where both $e_R(t)$ and $e_I(t)$ are modeled as white Gaussian and have a same variance σ_e^2.

Two sets of 400 data samples with noise power $\sigma_e^2 = 0.1$ (i.e., -10 dB) and $\sigma_e^2 = 0.001$ (i.e., -30 dB) respectively were used to build-up/train the network using the CMRAN algorithm. The network input vector is defined as: $\mathbf{x}(t) = [s(t)\ s(t-1)\ s(t-2)]^T$ and the desired output is $y(t)$. The various threshold parameters required by the CMRAN algorithm are selected by tuning them (largely by trial and error) to get the best results on the training data. The tuned values are then used for the test data. For noise power of 0.1, the parameters are: $e_{\min} = 0.1$, $e_{\min 1} = 1$, $\varepsilon_{\max} = 4$, $\varepsilon_{\min} = 0.01$, $\gamma = 0.81$, the size of the two sliding windows M and Sw are 4 and 50 respectively, pruning threshold $\delta = 0.01$. For noise power of 0.001, the parameters are: $e_{\min} = 0.1$,

$e_{\min 1} = 0.5$, $\varepsilon_{\max} = 4$, $\varepsilon_{\min} = 0.01$, $\gamma = 0.98$, the size of the two sliding windows are both 70, pruning threshold $\delta = 0.01$.

Two criteria have been chosen to assess the performance of the algorithm. The first performance criterion is the mean square error (MSE) defined by

$$\hat{\sigma}_{\varepsilon}^2 = \frac{1}{2N}\sum_{t=1}^{N} |\varepsilon(t)|^2$$

$$\varepsilon(t) = y(t) - \hat{y}_m(t)$$

where $\hat{y}_m(t)$ is the CMRAN output.

The second criterion is the mean state error defined by

$$\hat{\sigma}_s^2 = \frac{1}{2n_s}\sum_{i=1}^{n_s} |\hat{y}_i - \hat{y}_{mi}|^2 .$$

For our example, there are $n_s = 4^3 = 64$ combinations of the input $\mathbf{x}(t)$ which gives rise to 64 points or states of noise-free channel output. The noise free channel output states are denoted as $\hat{y}_i (1 \le i \le n_s)$. Likewise the CMRAN output states are denoted as $\hat{y}_{mi} (1 \le i \le n_s)$.

The growth of hidden neurons in CMRAN and the evolution of the MSE are depicted in Fig. 2 and Fig. 3 respectively. Figures 4 and 5 display the constellation of the channel states with those produced by the CMRAN model for noise power 0.001 and 0.1. The corresponding state errors between the channel output states and the CMRAN output states are plotted in Figs. 6 and 7. As a comparison, Table 1 gives the network size and the values of $\hat{\sigma}_{\varepsilon}^2$

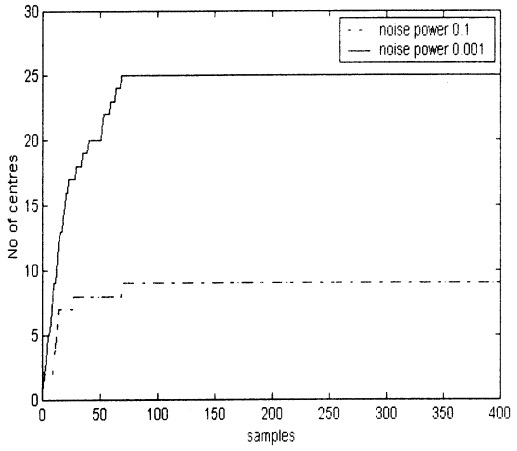

Fig. 2. Growth of hidden neurons for CMRAN network.

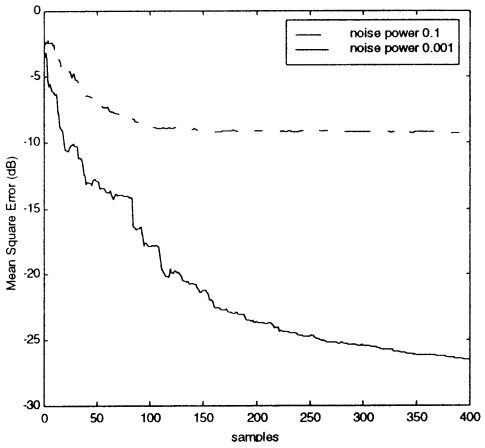

Fig. 3. Evolution of MSE for CMRAN network.

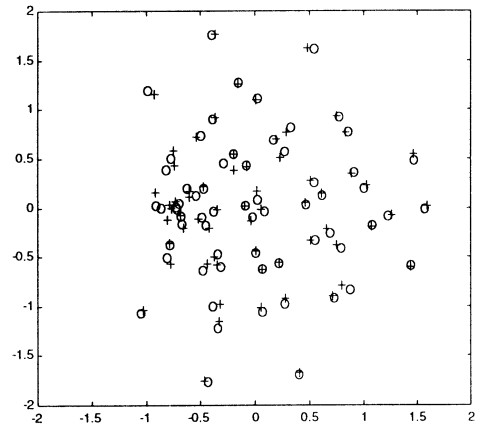

Fig. 4. State constellation. O channel, +CMRAN, noise power 0.001.

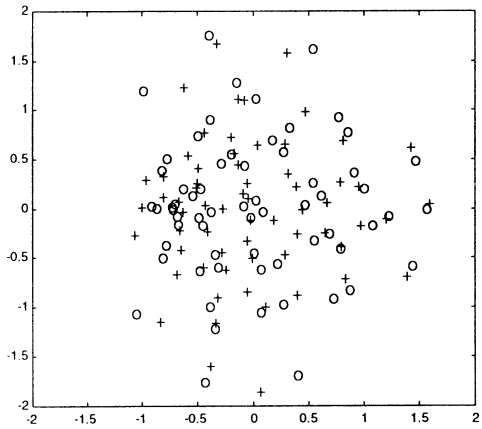

Fig. 5. State constellation, O channel, +CMRAN, noise power 0.1.

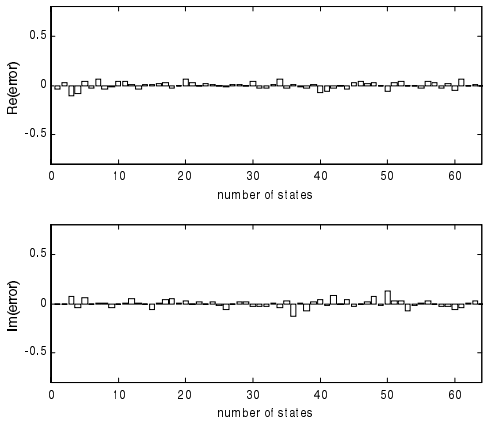

Fig. 6. State errors between channel and CMRAN model noise power 0.001.

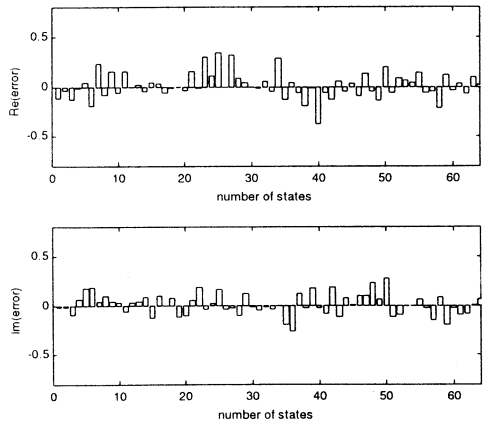

Fig. 7. State errors between channel and CMRAN model noise power 0.1.

Table 1. Performance comparison of CMRAN with well-known existing methods.

Noise Power		0.1	0.001
Mean square error $\hat{\sigma}_e^2$	CMRAN	0.1175	0.0023
	RBF with OLS	0.0942	0.0017
	RBF with hybrid	0.1078	0.0074
Mean state error $\hat{\sigma}_s^2$	CMRAN	0.0139	0.0014
	RBF with OLS	0.0149	0.0016
	RBF with hybrid	0.0174	0.0082
Number of hidden neurons	CMRAN	9	25
	RBF with OLS	30	30
	RBF with hybrid	30	30

and $\hat{\sigma}_s^2$ for CMRAN along with those of two well-known methods for the same example. The two methods are (i) the RBF with orthogonal least squares (OLS) algorithm and (ii) the RBF with the recursive hybrid clustering and LS algorithm.[5]

It is seen from the table that CMRAN is able to achieve similar errors as the other two methods but with much smaller networks than them. Further CMRAN is an online learning algorithm unlike the RBF with OLS method which is a batch learning algorithm.

4. Complex MRAN for Equalization Problems

In this section, we further demonstrate the usefulness of CMRAN by presenting its performance for linear and nonlinear channel equalization problems. Even though the main thrust of the paper is on nonlinear signal processing problems, we first present the results of CMRAN for a linear complex channel equalization problem. After that, the main results of using CMRAN for two nonlinear channel equalization problems are highlighted.

Consider the baseband discrete time model of a data transmission system given by

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

where a complex valued digital sequence $\{s_k\}$ is transmitted through a dispersive complex channel and the channel output is corrupted by an additive complex-valued noise $\{e_k\}$. In the above equation, $f_h(\cdot)$ is some complex valued function (which may be linear or nonlinear), n_h is the order of the channel. The task of the symbol decision equalizer is to reconstruct the transmitted symbols $s(k-\tau)$ based on noisy channel observation vector $\mathbf{y}(k) = [y(k) \cdots y(k-m+1)]^T$. τ is the equalizer decision delay, m is the equalizer dimension. It is known that Bayes decision theory[14] provides the optimal solution to the symbol decision problem.

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: $\mathbf{s}(k) = [s(k) \cdots s(k-m+1-n_h)]^T$

would result in n_s points or values of noise-free channel output vector $\hat{\mathbf{y}}(k) = [\hat{y}(k) \cdots \hat{y}(k-m+1)]^T$ where $n_s = 4^{n_h+m}$. These output vectors are referred to as desired channel states and are partitioned into four different classes, according to the value of $s(k-\tau)$ as follows:

$$Y_{m,\tau}^{(1)} = \{\hat{\mathbf{y}}(k) | s(k-\tau) = a + ja\},$$
$$Y_{m,\tau}^{(2)} = \{\hat{\mathbf{y}}(k) | s(k-\tau) = a - ja\},$$
$$Y_{m,\tau}^{(3)} = \{\hat{\mathbf{y}}(k) | s(k-\tau) = -a + ja\},$$
$$Y_{m,\tau}^{(4)} = \{\hat{\mathbf{y}}(k) | s(k-\tau) = -a - ja\}.$$

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 $\mathbf{y}(k) = [y(k) \cdots y(k-m+1)]^T$ is a random process with a conditional Gaussian density function centered at each of the desired channel states. The noisy observation vector is used as the input to the equalizer to determine the transmitted symbol $s(k-\tau)$.

Assume that the real and imaginary part of transmitted symbol $s(k)$ is equiprobable and independent sequences. Let λ be the *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 $\mathbf{y}(k)$ given $s(k-\tau) = \alpha^{(j)}$ is

$$f_B^{(j)}(\mathbf{y}(k)) = \lambda \sum_{i=1}^{n_s^j} \exp(-(\mathbf{y}-\mathbf{y}_i^{(j)})^H(\mathbf{y}-\mathbf{y}_i^{(j)})/2\sigma_e^2) \qquad 1 \le j \le 4.$$

The optimal Bayesian equalizer solution is defined as:

$$\hat{s}(k-\tau) = \alpha^{(j)}$$
$$\text{if } f_B^{(j)}(\mathbf{y}(k)) = \max\{f_B^{(j)}(\mathbf{y}(k)),\ 1 \le j \le 4\}.$$

This equation also defines the optimum decision boundaries for the partition of equalizer inputs sets. It is clear that these optimum decision boundaries are nonlinear hypersurfaces in the complex observation space and realizing such a nonlinear boundary will require the equalizer to have nonlinear mapping capabilities.

In recent investigations[1,16] Multilayer feed forward neural network and Radial Basis function (RBF) networks have been considered for use in channel equalization because of their ability to perform nonlinear classification. In Refs. 6 and 15 complex RBF equalizers for complex channels have been discussed. MRAN was first introduced in Ref. 17 to solve the equalization problems for real valued channels and signals. In this paper investigate the Complex-valued MRAN to implement complex channel equalization.

4.1. *CMRAN equalizer simulation results*

The CMRAN equalizer is shown in Fig. 8. Using the noisy channel output vector $\mathbf{y}(k)$ as the network input and $s(k-\tau)$ as the desired output, the CMRAN learning algorithm can readily be applied to train the complex RBF as shown by the dotted lines of Fig. 8. During the testing phase, the CMRAN output $f(\mathbf{y}(k))$ is fed into a nearest-neighbor decision device to give an estimate $\hat{s}(k-\tau)$ of the transmitted symbol $s(k-\tau)$, where τ is the equalizer decision delay.

4.1.1. *Linear channel equalization*

The following linear channel example[15] was chosen to graphically illustrate the decision boundary formed by CMRAN equalizer and the Bayesian equalizer. The equalizer dimension is chosen as $m = 1$. 4QAM signaling is used for the input sequence. The decision delay was chosen to be $\tau = 0$. The channel transfer function is given by

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

In this example, channel order $n_h = 2$ and the noise variance is $\sigma_e^2 = 0.06324$ (SNR = 15 dB). The real and imaginary part of input sequence were restricted to be from the symbol set $\{+1, -1\}$. Thus, there will be 64 output states for the channel. Figure 9 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.

One thousand and two hundred data samples at 15 dB SNR are used to build-up/train the CMRAN. The parameters of CMRAN are set as: $e_{\min} = 0.1$,

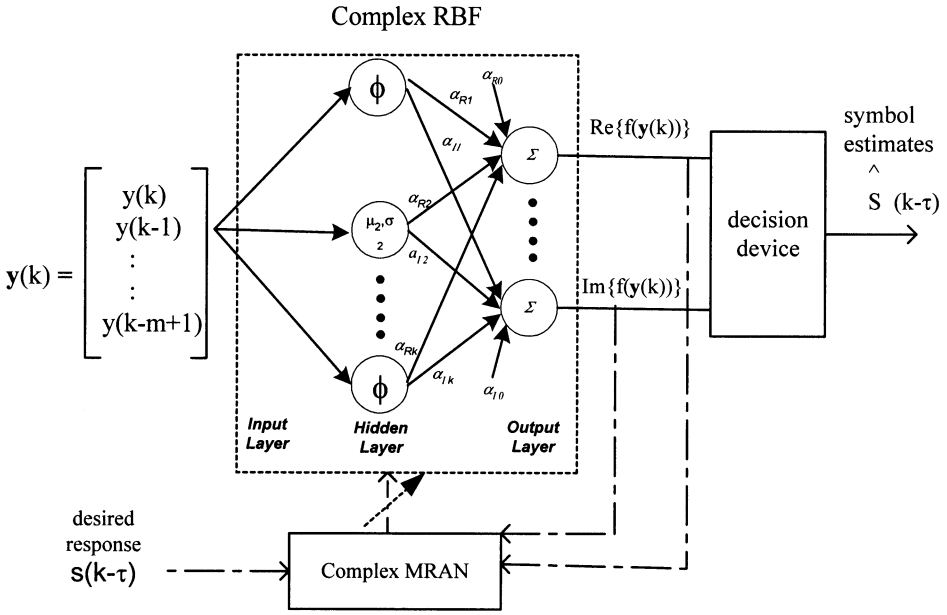

Fig. 8. CMRAN equalizer topology.

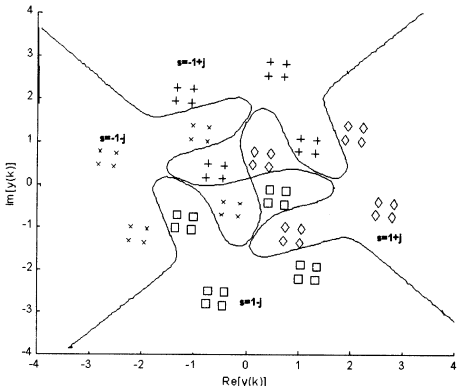

Fig. 9. Bayesian boundary (with 64 states).

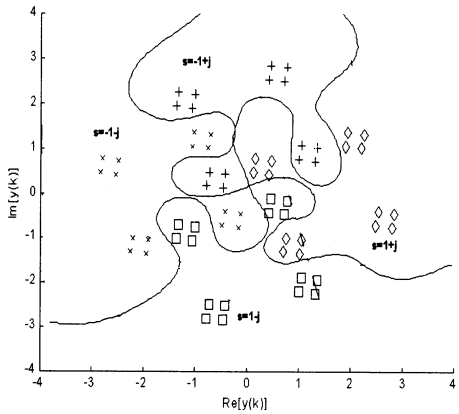

Fig. 10. Complex MRAN boundary (with 52 neurons).

$e_{\min 1} = 0.1$, $\varepsilon_{\max} = 0.25$, $\gamma = 1$. The size of the two sliding windows M and Sw are 10 and 80 respectively, and the pruning threshold $\delta = 0.01$. Figure 11 shows the growth of the hidden neurons in CMRAN as training progresses. The network has built up 52 hidden nodes at the end of the training. This is much less than the 64 hidden units if a hidden node is to be placed in each of the desired channel states. The decision boundary generated by CMRAN using only 52 hidden neurons is shown in Fig. 10. Comparing the CMRAN boundary with the Bayesian of Fig. 9, it can be seen that the Bayesian boundary is well-approximated at the critical region, which is the center portion of the figure. 10^5 test data of various SNR were used to test the resulting CMRAN equalizer. A comparison of their Symbol Error Rate (SER) curves in Fig. 12 show that CMRAN equalizer performs only slightly poorer than the ideal Bayesian equalizer.

4.1.2. *Nonlinear channel equalization*

In this section, we have considered two nonlinear channels, first one with a finite memory and next with an infinite memory.

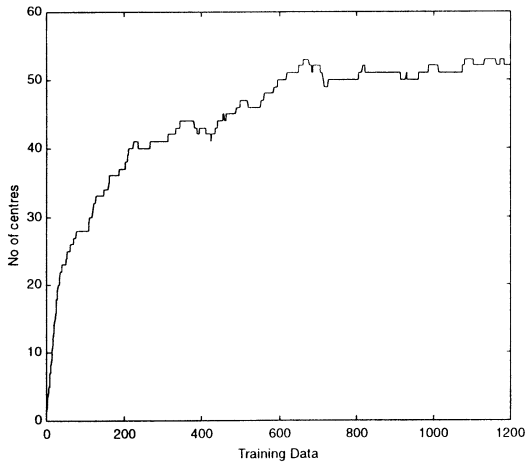

Fig. 11. Growth of hidden neurons as training progresses.

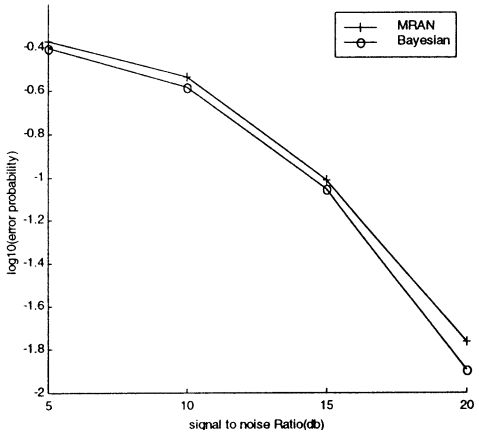

Fig. 12. Error curves for MRAN and Bayesian equalizer.

Finite memory channel:

In this example, we considered the performance of CMRAN equalizer in the presence of nonlinearity and noise for a finite memory nonlinear channel with 4-QAM signaling. The channel output[6] is given by

$$y_n = o_n + 0.2o_n^2 + 0.1o_n^3 + v_n\,; \quad v_n \sim N(0,\, 0.01)$$

$$o_n = (1 - j0.3434)s_n + (0.5 + j0.2912)s_{n-1}\,.$$

The equalizer dimension was set to $m = 2$ and the decision delay was $\tau = 0$. Samples of the training set and the test set were generated independently with both the real and imaginary parts of input sequence taking only values from the set ± 0.7. The parameters of CMRAN are set as: $e_{\min} = 0.2$, $e_{\min 1} = 0.8$, $\varepsilon_{\max} = 0.25$, $\gamma = 1$, the size of the two sliding windows M and Sw are 20 and 80 respectively, pruning threshold $\delta = 0.01$.

Figure 13 shows the SER values of the CMRAN equalizer as training progresses. Figure 14 shows the SER values of four other well-known methods from Ref. 6 for the same example. The four methods are: (i) FIR linear equalizers with LMS training (with 20 taps); (ii) Complex RBF network with Gaussian basis function built up by hybrid LMS algorithm (with 30 centers); (iii) Complex RBF network with TPS basis functions built up by Moody-Darken (MD) algorithm (with 30 centers); (iv) Complex RBF network with Gaussian

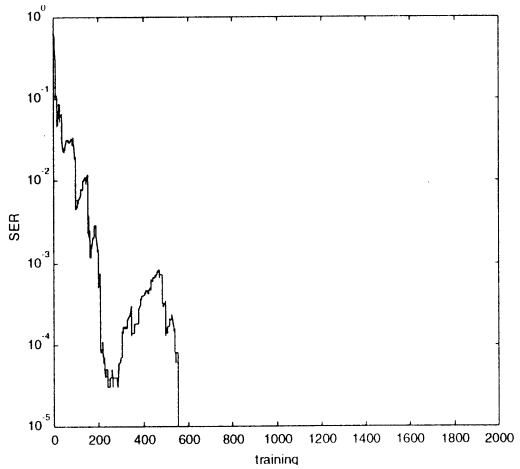

Fig. 13. SER values of CMRAN equalizer (14 centers).

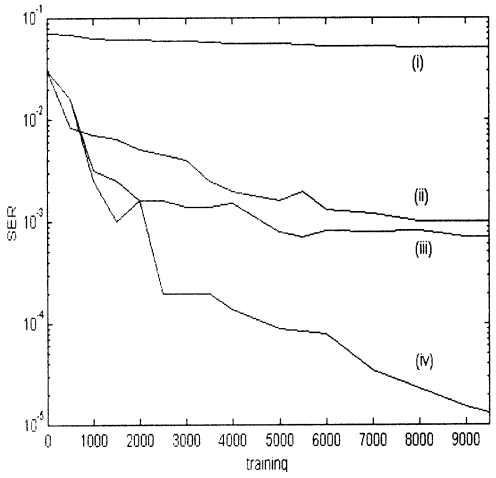

Fig. 14. SER values for (i) linear FIR (20), (ii) Gaussian/hybrid with 30 centers, (iii) TPS/MD with 30 centers, (iv) Gaussian/SG with 20 centers.

basis function built up by the stochastic gradient (SG) algorithm (with 20 centers). Comparing Figs. 13 and 14, it is clear that CMRAN produces lower SERs than the other methods of equalization. After only 600 training data, the CMRAN equalizer achieved a SER as low as 10^{-5} with 14 hidden units, while even the best among the four methods (the Gaussian SG) needed at least 9000 training data to achieve the same SER and that too with 20 hidden units. The Bayesian equalizer for this problem would use 64 channel states ($n_s = 4^3$).

Test sets with 100 000 samples at various SNR were used for the error probability evaluation. A comparison of probability of error curves is shown in Fig. 17. The CMRAN equalizer did not perform as well as Bayesian equalizer but that can be attributed to the compactness of the network realized. CMRAN used far fewer hidden nodes than the number of channel states. Figure 15 shows the distribution of channel output for the test data and Fig. 16 shows the CMRAN equalizer output distribution. It is clear from Fig. 16 that CMRAN is able to equalize the noisy channel output into four zones.

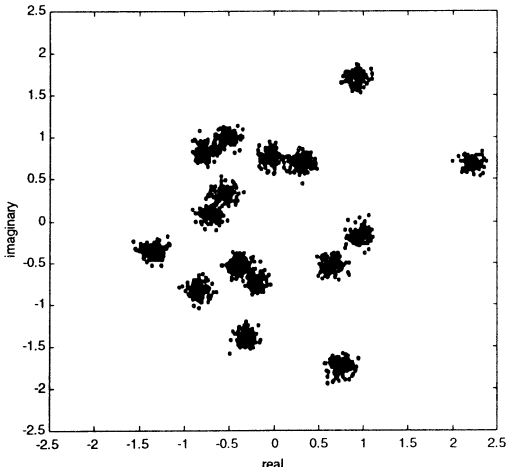

Fig. 15. Test set channel output distribution.

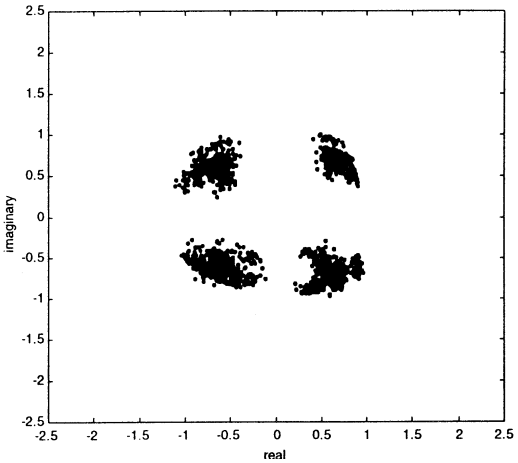

Fig. 16. CMRAN equalizer output sets (after 600 iterations).

Infinite memory channel:

In this example, a nonlinear channel with 16-QAM signaling[6] was used. The real and imaginary parts of input sequence takes values from the set $\{\pm 2.1, \pm 0.7\}$. The channel output is given by

$$y_n = -0.3y_{n-1} + o_n + 0.02o_n^2 + 0.01o_n^3 + v_n\,; \quad v_n \sim N(0,\, 0.01)$$

$$o_n = (1 - j0.3434)s_n + (0.5 + j0.2912)s_{n-1}\,.$$

Note that the channel has infinite memory due to feedback in the channel output equation. For the CMRAN equalizer, 20 000 training data were used. The parameters of CMRAN are: $e_{\min} = 0.01$, $e_{\min 1} = 0.8$, $\varepsilon_{\max} = 16$, $\varepsilon_{\min} = 0.16$, $\gamma = 0.9986$, the size of the two sliding windows M = 10 and Sw = 50, pruning threshold $\delta = 0.001$. The resulting CMRAN equalizer had 22 hidden units. Figures 18 shows the noisy channel output distribution. Figure 19 shows the CMRAN equalizer output distribution with 16 zones clearly separated out.

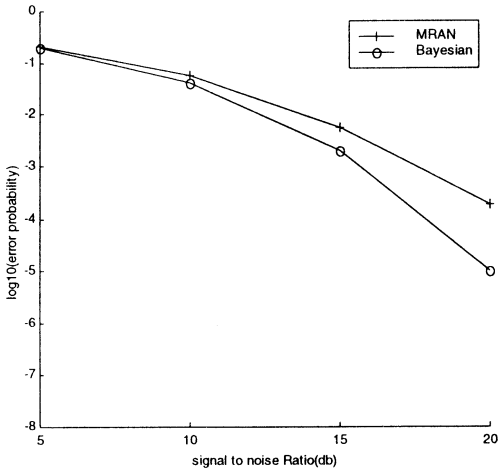

Fig. 17. Error curves for CMRAN and Bayesian equalizer.

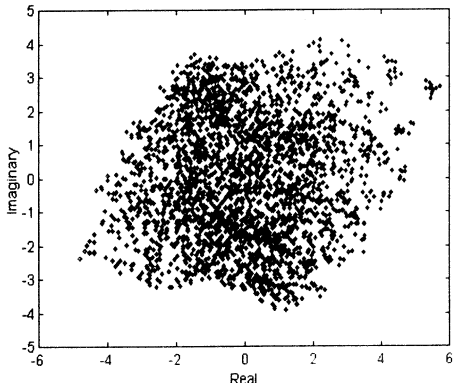

Fig. 18. Test set channel output distribution.

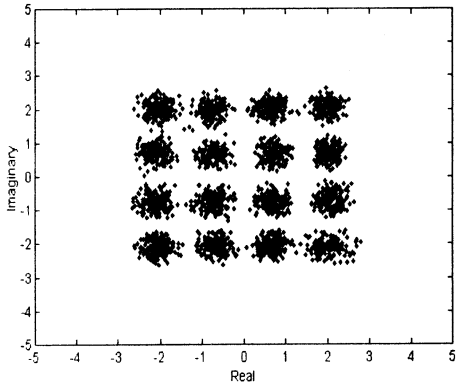

Fig. 19. Complex MRAN equalizer output (after 20 000 iterations).

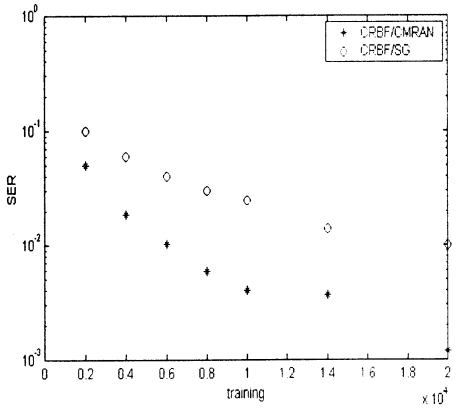

Fig. 20. SER values comparison.

Performance was measured by the symbol error rate (SER) as well as the normalized mean squared error (NMSE) (to be consistent with Ref. 6) and their values are shown in Figs. 20 and 21 respectively.

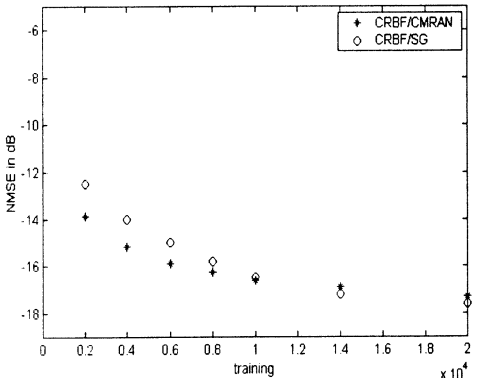

Fig. 21. NMSE values comparison.

These figures clearly show that CMRAN equalizer performs much better than the CRBF/SG equalizer[6] which used 30 hidden units.

5. Conclusions

In this paper, the recently developed minimal resource allocation network algorithm has been extended to a complex form (CMRAN) for operation on complex valued signal processing problems. The performance of the CMRAN algorithm is demonstrated using two applications from nonlinear signal processing of communication systems. The first case considers modeling a nonlinear complex channel. The study results show that CMRAN algorithm can produce more compact network than other well-known algorithms but with similar approximation error. In the second case, the potential of CMRAN for channel equalization with QAM signaling scheme is investigated. It has been shown that CMRAN network is well-suited for nonlinear channel equalization problems. Again, its performance is superior to those of several existing equalization methods. Because CMRAN has reduced network complexity as compared to the other methods, the time taken for equalization is also much less.

A way of achieving further reduction in network size may be to utilize decision-feedback (DFB) strategy in the CMRAN algorithm. If prior symbols are estimated correctly, DFB can achieve faster convergence and better performance. We are currently exploring the feasibility of combining DFB schemes with CMRAN.

References

1. S. Chen, B. Mulgrew and P. M. Grant 1993, "A clustering technique for digital communications channel equalisation using radial basis function networks," *IEEE Trans. Neural Networks* **4**(4), 570–578.
2. S. Chen, S. A. Billings and W. Luo 1989, "Orthogonal least squares methods and their application to non-linear system identification," *Internal. J. Control* **50**(5), 1873–1896.
3. K. Tao 1993, "A closet look at radial basis function (RBF) networks," in *Conference Record of 27th Asilomar Conference on Signal, System and Computers*, Pacific Grove, CA, USA, pp. 401–405.
4. T. L. Clarke 1990, "Generalization of neural networks to the complex plane," *Proc. Int. Joint Conf. Neural Networks*, San Diego, Vol. 2, pp. 435–440.
5. S. Chen, S. Mclaughlin and B. Mulgrew 1994, "Complex-valued radial basis function network. Part I: Network architecture and learning algorithms," *Signal Processing* **35**(1), 19–31.
6. I. Cha and S. Kassam 1995, "Channel equalisation using adaptive complex radial basis function networks," *IEEE Journal on Selected Areas in Communication* **13**(1), 122–131.
7. Y. Lu, N. Sundararajan and P. Saratchandran 1998, "Performance evaluation of a sequential minimal Radial Basis Function (RBF) neural network learning algorithm," *IEEE Transactions on Neural Networks* **9**(2), 308–318.
8. Y. Lu, N. Sundararajan and P. Saratchandran 1997, "A sequential learning scheme for function approximation using minimal radial basis function neural networks," *Neural Computation* **9**, 461–478.
9. Y. Lu, N. Sundararajan and P. Saratchandran 1997, "Identification of time-varying nonlinear systems using minimal radial basis function neural networks," *IEE Proceedings — Control Theory Application* **144**(2), 202–208.
10. J. C. Platt 1991, "A resource allocating network for function interpolation," *Neural Computation* **3**, 213–225.
11. V. Kadirkamanathan and M. Niranjan 1993, "A function estimation approach to sequential learning with neural networks," *Neural Computation* **5**, 954–975.
12. W. Luo, S. A. Billings 1998, "Structure selective updating for nonlinear models and radial basis function neural networks," *Int. J. Adaptive Control and Signal Processing* **12**, 325–345.
13. C. F. Fung, S. A. Billings, W. Luo 1996, "Online supervised adaptive training using radial basis function networks," *Neural Networks* **9**, 1597–1617.
14. J. G. Proakis 1983, *Digital Communication* (McGraw-Hill, New York).
15. S. Chen, S. Mclaughlin and B. Mulgrew 1994, "Complex-valued radial basis function network. Part II: Application to digital communications channel equalization," *Signal Processing* **36**, 175–188.
16. S. Chen, G. J. Gibson, C. F. N. Cowan and P. M. Grant 1990, "Adaptive equalisation of finite nonlinear channels using multilayer perceptrons," *Signal Processing* **20**, 107–119.
17. P. Chandra Kumar 1998, Radial basis function networks for communication channel equalisation," MEng thesis, School of Electrical and Electronic Eng., Nanyang Technological University, Singapore.