RADIAL BASIS FUNCTION EQUALIZATION USING COMPETITIVE LEARNING

CORINA BOTOCA[1], GEORGETA BUDURA, MICLĂU NICOLAE

Key words: radial basis function neural network, complex equalizer, competitive learning

In this paper the problem of equalization of multiple quadrature amplitude modulated signals, using a radial basis function (RBF) neural network, is studied. Because the equalizer performance is directly related to the estimations of the RBF centres, different competitive learning algorithms for the RBF centres are presented. A new competitive algorithm is introduced, the rival penalized competitive learning, which rewards the winner and penalises its first rival. Simulations results, performed in different conditions, are presented, showing that the performance of the RBF equalizer, if trained with this new algorithm is better comparative with other competitive algorithms.

1. **INTRODUCTION**

Communication channels introduce different perturbations in signals transmission, as example, nonlinear distortions, fading, intersymbol interference (ISI), adjacent channel interference and co channel interference. As effect, the channels characteristics are variable in amplitude and in phase. The technique of eliminating the undesired effects of communication channel is the equalization.

Traditional approaches to channel equalization are based on the inversion of the global channel response. In digital systems the complete channel inversion is neither required nor desirable. However, traditional techniques requiring the traffic model, are obsolete in the context of modern communications, where a mathematical model is not always possible to be drawn. Since transmitted symbols belong to a discrete alphabet, symbol demodulation can be recasted as a classification problem in the space of the received symbols. Neural networks (NN) are promising candidates, not only because they can learn an arbitrarily nonlinear input output function from examples, but also due to their adaptability, flexibility and speed. The studies performed during the last decade have already established

[1]) "Politehnica" University of Timisoara, Bd. V. Pârvan, CP 300223, E-mail: corina.botoca@etc.utt.ro

the superiority of neural equalizers comparative to the traditional equalizers, in conditions of high nonlinear distortions and rapidly varying signals.

During the last decade the research interest was focused on signals more efficient in transmission from the spectral point of view, as the multiple quadrature amplitude modulated (M-QAM) signals. But the M-QAM signals are severely affected by the nonlinear distortions, because they have a variable envelope modulation. To compensate these unwanted distortions equalizers for complex signals are necessary. The complex NN equalizers are straightforward extensions from the real counterparts [1], obtained by replacing the relevant parameters with complex values. Various neural equalizers have been developed, mostly combinations between a conventional linear transversal filter (LTF) and a neural network which may be a multilayer perceptron [1], [2], [3], [4], a radial basis function network (RBF) [5], [6], [7], [8], [9], [10], or a recurrent neural network [11], [12], [13], [14]. The LTF eliminates the linear distortions, such as ISI, so the NN has to compensate the nonlinearities. Many different nonlinear channels models have been introduced to simulate real situations, so a unitary comparison between all known equalizers is difficult to be done.

Recently, the RBF network received considerable attention, since the MLP network is plagued by long training times and may be trapped in bad local minima. The RBF network is able to approximate any arbitrary nonlinear function in the complex multi-dimensional space with a reduced calculus complexity comparative with other NN. RBF often provide a faster and more robust solution to the equalization problem [1], [7]. In addition, the RBF neural network has a structure similar to the optimal Bayesian symbol decision equalizer [6]. Therefore, the RBF is an ideal processing structure to implement the optimal Bayesian equalizer.

Several learning algorithms have been proposed to update the RBF parameters. Usually it is used an unsupervised learning algorithm to find the centres of the hidden neurons and a supervised learning algorithm for the weights of the output neurons. The competitive standard learning algorithm (CSL) [15] computes the squared distance between the input vector and the centres, chooses the winning centre, the one with the minimum distance, and moves it closer to the input vector. The major deficiency of the CSL algorithm is that it needs to know the exact number of clusters *k*, before performing data clustering. Otherwise, it will lead to a poor clustering performance. Unfortunately, it is often hard to determine *k* in advance in many practical problems. Another problem is that the classification depends on the initials centres values of the RBF, on the type of the chosen distance, on the number of classes. The CSL algorithm has also the "dead neurons" problem, which means that if a centre is inappropriately chosen, it may never be updated, thus it may never represent a class.

To circumvent the "dead neurons" problem it was proposed an extension of the CSL algorithm, named the frequency sensitive competitive learning (FSCL)

[15]. The FSCL algorithm reduces the learning rate of the frequent winners, so their chance to win, strategy called also "with conscience". Although it can almost successfully assign one or more seed points to a cluster without the "dead neurons" problem, it also needs knowing the exact number of clusters.

In this paper we introduce a new competitive method to update the RBF centres, the rival penalized competitive learning (RPCL) algorithm that performs appropriate clustering without knowing the clusters number, by automatically driving the extra number of seed points far away from the input data set. The RPCL algorithm rewards the winning neuron and penalizes with a de-learning rate the second winner, named rival. The algorithm is rather sensitive to the selection of the de-learning rate, but it is quite simple and provides a better convergence than the classic competitive learning and FSCL algorithms. It also eliminates the "dead neurons" problem.

2. **THE EQUALISATION PROBLEM**

The objective of equalization from the NN point of view is the separation of the received symbols in the output signal space, whose optimal decision region boundaries are generally highly nonlinear. Figure 1 represents a model of the communication system.

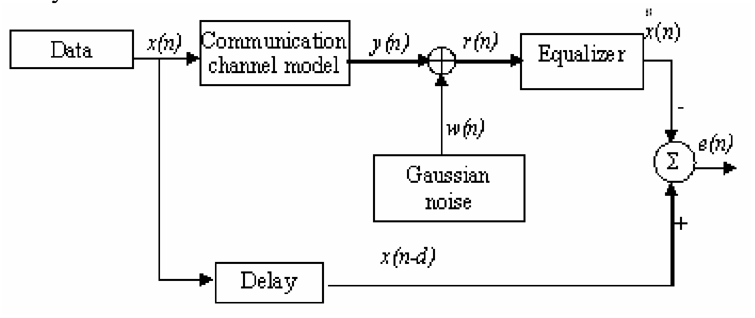

Fig. 1 A model of a communication system

The complex-valued digital sequence *x(n)* is transmitted through a dispersive complex channel of M order. If the input is a 4 QAM signal, its constellation is given by the following relation:

$$x(n) = x_R + jx_I = \begin{cases} x^{(1)} = \ \ 1 + j \\ x^{(2)} = -1 + j \\ x^{(3)} = \ \ 1 - j \\ x^{(4)} = -1 - j \end{cases} \tag{1}$$

The communication channel, that introduces linear and nonlinear distortions, may be modeled as presented in figure 2. Various models with different linearities (L) and nonlinerities (NL) are mentioned in literature. Most of the studies refer to the ones mentioned in that follows.

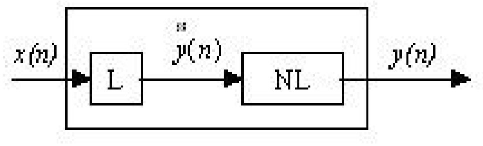

Fig2. The nonlinear channel model

The linear complex part of the channel is usually a transversal filter with finite impulse response, whose output is given by:

$$\overset{\approx}{y}(n) = \sum_{i=0}^{M-1} a_i x(n-i) \tag{2}$$

where a_i are the filter coefficients and M is the order of the filter.

The model suggested in [5] generates the output signal $\overset{\approx}{y}(n)$ according to the relation:

$$\overset{\approx}{y}(n) = (0.34 - 0.27j)x(n) + (0.87 + 0.43j)x(n-1) + (0.34 - 0.21j)x(n-2) \tag{3}$$

The nonlinear part of the channel is a very strong one and produces at the output:

$$y(n) = \overset{\approx}{y}(n) + 0.1[\overset{\approx}{y}(n)]^2 + 0.05[\overset{\approx}{y}(n)]^3 \tag{4}$$

Another model referred in [6] uses the following equations:

$$\begin{aligned} \overset{\approx}{y}(n) = &(0.7409 - 0.7406j)x(n) - (0.8890 - 0.2961j)x(n-1) + \\ &+ (0.1556 - 0.0223j)x(n-2) \end{aligned} \tag{5}$$

$$y(n) = \overset{\approx}{y}(n) - 0.055[\overset{\approx}{y}(n)]^2 + 0.14[\overset{\approx}{y}(n)]^3 \tag{6}$$

The channel output *y(n)* is corrupted by adding a complex-valued noise *w(n)*, usually a white Gaussian noise with a zero mean and a dispersion of σ_e^2, so the received signal is:

$$o(n) = y(n) + w(n) \tag{7}$$

We will consider the real and imaginary parts, $w_R(n)$ and $w_I(n)$ mutually independent sequences. Also, the signals *x(n)* and *w(n)* are assumed uncorrelated.

The task of the equalizer is to reconstruct the transmitted symbols as accurately as possible, producing an estimation $\overset{\approx}{x}(n)$, based on the noisy received signal *r(n)* and the desired delayed signal *x(n-d)*. The equalizer performance is evaluated with respect to the signal to noise ratio (SNR):

$$SNR = \frac{E\left[r^2(n)\right]}{E\left[w^2(n)\right]} = \frac{\sigma_s^2\left(\sum_{i=0}^{M-1} a_i^2\right)}{\sigma_e^2} = \frac{\sum_{i=0}^{M-1} a_i^2}{\sigma_e^2} \tag{8}$$

where E is the second moment, $\sigma_s^2=1$ is the transmitted symbol dispersion and σ_e^2 is the noise dispersion.

The channel output vector is passed through the RBF equalizer, consisting of a LTF of *m* order and a RBF NN, as presented in figure 3. The received signal ***r(n)*** applied to the RBF network input is the sequence ***r(n)=[r(n) r(n-1) ….r(n-m+1)]***^{T}. Because it involves *m* terms of the delayed version of the received signal, there are $N_S=4^{M+m-1}$ possible combinations of the channel input sequences, i.e. ***x(n)=[x(n) x(n-1) …x(n-m-M+2)]***^{T}. Correspondingly, the noise-free channel output vector is ***y(n)=[y(n) y(n-1) …y(n-m+1)]***^{T} and it has also N_S desired corresponding states.

From the NN point of view, the equalizer has to classify the received signal in one of the four possible classes $P_{m,d}$, according to the input delayed signal:

$$P_{m,d} = \bigcup_{1 \le l \le 4} P_{m,d}(l) \tag{9}$$

or:

$$P_{m,d}(l) = \left\{ y(n) \middle| x(n-d) = x^{(l)} \right\} \qquad 1 \le l \le 4 \tag{10}$$

3. **THE COMPLEX RADIAL BASIS FUNCTION NETWORK**

As depicted in figure 3, the RBF network has two layers, the hidden layer and the output layer. The RBF input and output are both complex and the nonlinearity of hidden neurons is a real function. The real part and the imaginary part of the signals are treated separately, in the same manner. The hidden layer is composed of an array of computing neurons, each one having a vector parameter $\boldsymbol{c_i}$, called centre. Each neuron computes a distance between its centre and the network input vector. This distance may be of different types and it is subsequently divided by a parameter ρ_i, called width, which is the spread of the corresponding centre. The result is passed through a real, nonlinear activation function $\phi_i(\bullet, \rho_i)$:

$$\phi_i = \phi\left(\left(\boldsymbol{r}(\boldsymbol{n}) - \boldsymbol{c_i}(\boldsymbol{n})\right)^{\boldsymbol{H}} \left(\boldsymbol{r}(\boldsymbol{n}) - \boldsymbol{c_i}(\boldsymbol{n})\right) / \rho\right) \quad 1 \le i \le n_h \tag{11}$$

where $\boldsymbol{r}$ is the complex input vector of n_h dimension , $\boldsymbol{c_i}$ is the centres vector of the radial basis functions, which is also a complex vector of n_h dimension, ρ_i is the centre spread parameter, n_h is the number of computing neurons. The operator $(\bullet)^H=((\bullet)^T)^*$, where $(\bullet)^T$ is the transposition operator and $(\bullet)^*$ is the complex conjugation operator.

The nonlinear output function is usually the Gaussian function:

$$\phi\left(\chi^2, \rho\right) = e^{-\frac{\chi^2}{\rho}} \tag{12}$$

The number of hidden neurons n_h is given by the number of possible states of the channel output N_S.

Similarity with the Bayesian equalizer imposes that the spread parameter $\rho=2\sigma_e^2$ [6] ,where σ_e^2 is the noise dispersion given by relation:

$$\sigma_e^2 = E\left\|\boldsymbol{r}(\boldsymbol{n}) - \boldsymbol{c_i}(\boldsymbol{n})\right\|^2 \tag{13}$$

The output layer of the network consists of eight neurons (two neurons for each class, one for the real part and the other for the imaginary part of each class) with a linear function:

$$f_{RBF}(r) = \sum_{i=1}^{n_h} \phi_i \boldsymbol{w_i} \tag{14}$$

where w_i are the complex weights.

According to the relation (12), f_{RBF} becomes:

$$f_{RBF}(r)=\sum_{i=1}^{n_h} w_i e^{-\frac{(r(n)-c_i(n))^H (r(n)-c_i(n))}{\rho_i}} \tag{15}$$

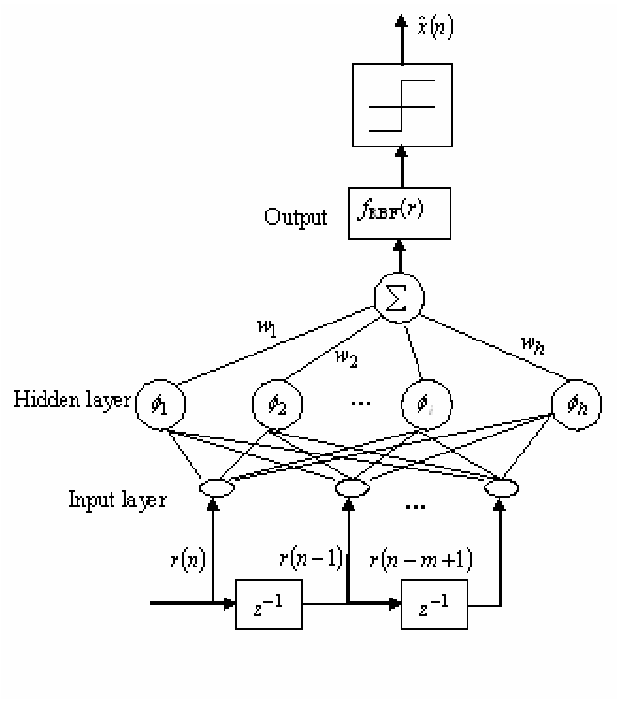

Fig3. The RBF neural network

4. COMPETITIVE LEARNING ALGORITHMS

4.1 THE COMPETITIVE STANDARD ALGORITHM

The competitive standard algorithm calculates the distance between the input vector and the RBF centres vector. The distance may be of different types, but usually the Euclidian norm is used:

$$\|r(n)-c_i(n)\|=\sqrt{|r(n)-c_i(n)|^2+.....+|r(n-m+1)-c_i(n-m+1)|^2} \tag{16}$$

The neuron *j* with a minimum distance is declared winner:

$$\mathrm{j} = \arg\min \left\| \boldsymbol{r(n)} - \boldsymbol{c}_i(\boldsymbol{n}) \right\|, \quad i = \overline{1, n_h} \tag{17}$$

The winning neuron centre is moved with a fraction η towards the input.

$$\boldsymbol{c_i}(\boldsymbol{n}+\boldsymbol{1}) = \boldsymbol{c_i}(\boldsymbol{n}) + \eta\left[\boldsymbol{r}(\boldsymbol{n}) - \boldsymbol{c_i}(\boldsymbol{n})\right] \tag{18}$$

The learning rate may be constant or descendant with a fraction, for example:

$$\eta(n+1) = \eta(n) - \frac{1}{n_h} \tag{19}$$

where n_h represents the number of neurons

The weights are randomly initialized, usually at the input vector values. Equations (17) and (18) are than applied iteratively until the algorithm converges or freezes as the number of iterations reaches a prespecified value, respectively the descending learning rate becomes zero or a very small value.

4.2 THE FREQUENCY SENSITIVE COMPETITIVE ALGORITHM

The simple classical CSL algorithm has the "dead neurons" problem. That is, if a neuron is initialized far away from the input data set in comparison with other neurons, it may never win the competition, so it may never learn, becoming a dead neuron. To solve this problem it has been introduced the so called "frequency sensitive competitive learning" algorithm or competitive algorithm "with conscience". Each centre counts the number of times when it has won the competition and reduces its learning rate consequently. If a neuron has won too often "it feels guilty" and it pulls itself out of the competition [15]. The FSCL algorithm is an extension of CSL, obtained by modifying relation (17) according to the following one:

$$j = \arg\min \gamma_i \left\| \boldsymbol{r}(\boldsymbol{n}) - \boldsymbol{c}_i(\boldsymbol{n}) \right\|, \quad i = \overline{1, n_h} \tag{20}$$

with the relative winning frequency γ_i of the centre c_i defined as:

$$\gamma_i = \frac{s_i}{\sum_{i=1}^{n_h} s_i} \tag{21}$$

where s_i is the number of times when the centre $\boldsymbol{c}_i$ was declared winner in the past. So the neurons that have won the competition during the past have a reduced chance to win again, proportional with the frequency term γ. After selecting out the winner, FSCL updates the winner with equation (18) in the same way as CSL, and meanwhile adjusting the corresponding s_i with the following relation:

$$s_i(n+1) = s_i(n) + 1 \tag{22}$$

FSCL can almost always successfully distribute the n_h centres into the input data set without the "dead neurons problem".

4.3 THE RIVAL PENALIZED COMPETITIVE ALGORITHM

The rival penalized competitive learning algorithm performs appropriate clustering without knowing the clusters number. It determines not only the winning neuron j but also the second winning neuron r, named rival:

$$r = \arg\min \left\| \boldsymbol{r}(\boldsymbol{n}) - \boldsymbol{c}_i(\boldsymbol{n}) \right\|, \quad i = \overline{1, n_h} \quad i \neq j \tag{23}$$

The second winning neuron will move away from the input its centre with a ratio β, called the de-learning rate. All the others neurons will not change their centres vector. So the learning law can be synthesized in the following relation:

$$\boldsymbol{c}_i(\boldsymbol{n+1}) = \begin{cases} \boldsymbol{c}_i(\boldsymbol{n}) + \eta\,[\boldsymbol{r}(\boldsymbol{n}) - \boldsymbol{c}_i(\boldsymbol{n})] & \text{if } i = j \\ \boldsymbol{c}_i(\boldsymbol{n}) - \beta\,[\boldsymbol{r}(\boldsymbol{n}) - \boldsymbol{c}_i(\boldsymbol{n})] & \text{if } i = r \\ \boldsymbol{c}_i(\boldsymbol{n}) \quad \text{if } i \neq j \quad \text{and} \quad i \neq r \end{cases} \tag{24}$$

If the learning speed η is chosen to be much greater than β, with at least one order of magnitude, the RBF network will automatically find the number of signal output classes. In other words, suppose that the number of classes is unknown and the number of hidden neurons n_h is greater than the number of the classes than the RBF centres will converge towards the centres of the input signals clusters. The penalizing competitive algorithm will move away the rival, in each iteration, converging much faster than the above mentioned algorithms. The number of extra seed points, respectively the difference between n_h and the number of classes will be driven away from the data set. If n_h is smaller than the number of the classes, than the network will oscillate during training, indicating that the number of hidden neurons must be increased.

5. **THE LEAST MEAN SQUARE ALGORITHM**

A supervised algorithm may be used to update the output neurons weights, for instance, the least mean square algorithm, given by the following relations:

$$\boldsymbol{w_i}(\boldsymbol{n}+\boldsymbol{1})=\boldsymbol{w_i}(\boldsymbol{n})+\alpha\, e(n)\phi(n) \tag{25}$$

where α is the learning constant and *e(n)* represents the complex error, determined with the relation:

$$e(n)=x(n-d)-f_{RBF}(r) \tag{27}$$

This algorithm minimizes the mean square error (MSE):

$$MSE=\frac{1}{N}\sum_{i=1}^{N}e_i^2(n) \tag{26}$$

where *N* is the number of input sequences.

6. **SIMULATION RESULTS**

There were generated 4-QAM signals, using an uniform distribution, independently the real part from the imaginary part. Simulations were done using the channel model introduced in reference [6] and presented in section 2. A white noise *w(n)* was generated and added to *y(n)*. The number of the hidden neurons, respectively of the desired centres, was chosen equal to N_S, the number of possible states of *y(n)*. For *M*=1, *m*=1 it results 64 hidden neurons. For the RBF output layer there were used 8 neurons, one for the real part and another for the imaginary part of each of the four possible states of the 4-QAM signal. The centres spread was chosen 0.28. There were applied *N*=7000 input signal sequences *x(n)*, *x(n)*=[*x(n) x(n-1) x(n-2)*] to train the RBF centres, in different conditions of noise, with all the three algorithms mentioned in section 4, CSL, FSCL and the new RPCL algorithm. The RBF centres were randomly initialized, around (5, 5j) as it can be seen in figures 4, 5 and 6. The delearning rate was chosen of a order of magnitude smaller than the learning rate, otherwise in these difficult conditions of nonlinearity and noise it may generate oscillations. The best results were obtained for the following learning constants: η=0.05, β=0.0001 and α=0.01. Figures 4 , 5 and 6 represent the output channel states y(n), the corrupted received signal r(n), the initial and final positions of the RBF centres in the case of a SNR=13dB (σ^2=0.05) after 100 training iterations using CSL, FSCL and RPCL algorithms.

The CSL algorithm failed to find the desired centres. Both other algorithms, FSCL and RPCL, succeed to orientate the RBF centres to the desired free of noise output channel states. In addition, the RPCL algorithm had driven away the extra number of seed points. As one can observe the RPCL algorithm could find closer positions of RBF centres to the desired output channel states than the FSCL algorithm, so its convergence is better.

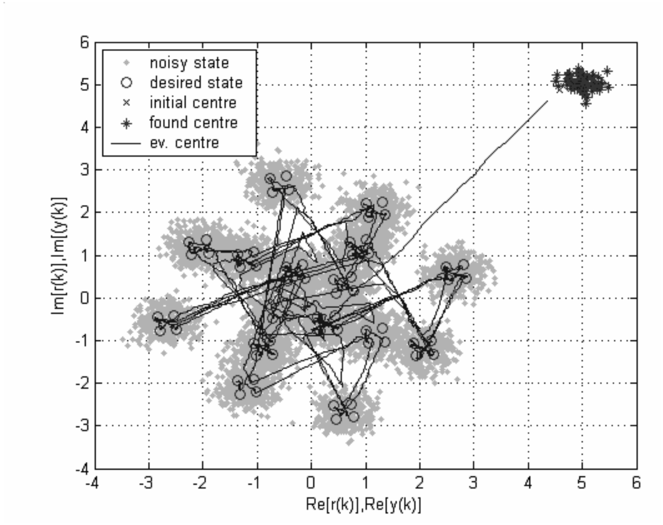

Fig.4 The output channel states *y(n)*, the corrupted received signal *r(n)*, the initial and final positions of the RBF centres in case of a *SNR*=13dB, after 100 training iterations, using CSL

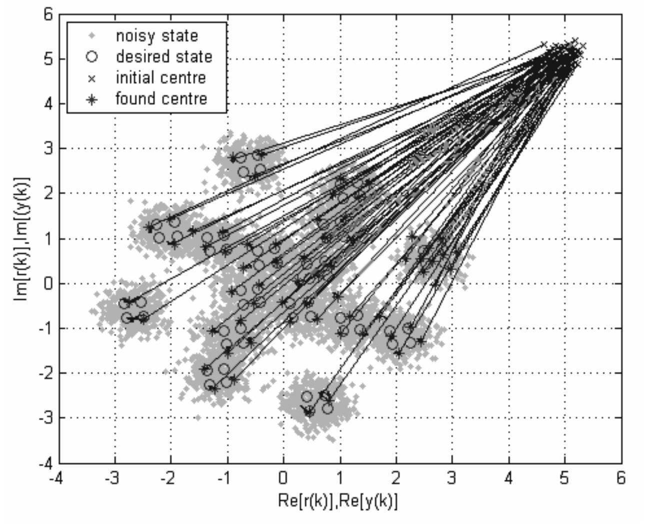

Fig.5 The output channel states *y(n)*, the corrupted received signal *r(n)*, the initial and final positions of the RBF centres in case of a *SNR*=13dB, after 100 training iterations, using FSCL

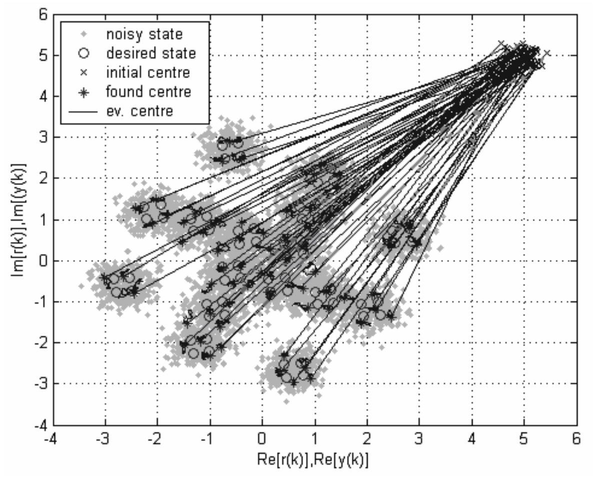

Fig.6 The output channel states *y(n)*, the corrupted received signal *r(n)*, the initial and final positions of the RBF centres in case of a *SNR*=13dB, after 100 training iterations, using RPCL

The RBF equalizer, using the FSCL and the RPCL algorithms, was tested in different conditions of noise, for different orders of the LTF and delays.

Figure 7 represents the MSE equalizer evolution , for the FSCL respectively RPCL algorithms, during 5000 epochs for a *SNR*=13dB and *m*=1.

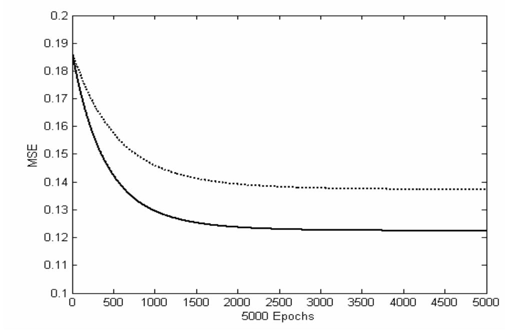

Fig.7 The comparative evolution of the MSE during 5000 epochs a for *SNR* =13 dB, *m*=1 (solid line - FSCL algorithm; dotted line - RPCL algorithm)

Figure 8 depicts the MSE evolution, using the new RPCL algorithm, during 3000 epochs for different signal to noise ratios (*SNR*=10dB and *SNR*=5 dB) and order *m* (*m*=1 and *m*=2) of the LTF and a delay of *d*=1. This performance is similar to the MLP NN equalizers, in conditions of a lower computational cost.

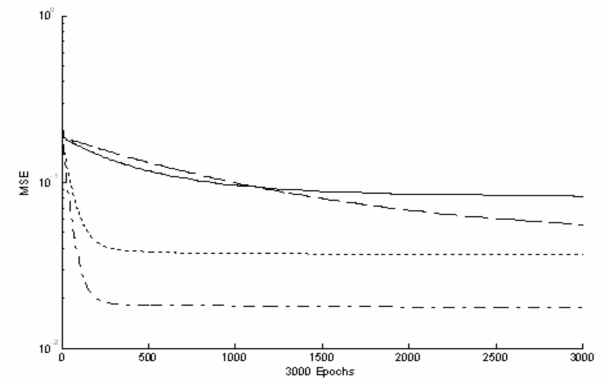

Fig.8 The evolution of MSE during 3000 epochs (solid line - *m*=1, *SNR*=5Db; dashed line - *m*=1, *SNR*=5dB; dotted line - *m*=1, *SNR*=10dB; dashed-dot line - *m*=2, *SNR*=10dB)

To represent the decision regions of the RBF complex equalizer, the complex output space was divided using a sampling step of δ=0.02 Figure 9 represents the output signals space partition, which has strong nonlinear decisions boundaries.

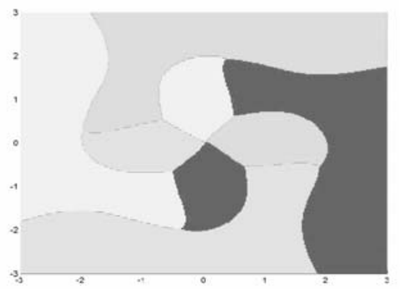

Fig.9 The output signals space partition

CONCLUSIONS

The main drawback of the neural network equalizers is the computational complexity and the extensive training. The proposed competitive algorithm, namely RPCL, while training the centres of the RBF network, rewards the winner and penalizes its closest rival. It is rather simple, generates strong nonlinear regions of decision in the signal space, yet having a better and faster convergence. In comparison with the clasic CSL algorithm, it doesn’t have the "dead neurons" problem. If compared with the FSCL algorithm, it has a better and faster convergence. So the RPCL algorithm is adequate to the adaptive equalization of fast varying signals corrupted with strong linear and nonlinear distortions. Because of its structure, similar to the Bayesian equalizer, the performance of the RBF equalizer is superior to the LTF and MLP equalizers. The MSE performance of the RPCL equalizer is similar to others RBF equalizers reported in literature, if tested in the same conditions. In order to improve the equalizers performances the order of the LTF filter coupled with the RBF neural network should be increased or the feedback should be introduced.

Received on (*month day, year*)

REFERENCES

[1] M. Ibnkahla, *Applications of Neural Networks to Digital Communication a Survey*, IEEE Signal Processing Magazine, November, pp.1186-1215, (1997)

[2] S.Bouchired, M. Ibnkahla, *Décision Neuronale Appliquée a L'égalisation De Canaux Satellitaires Mobiles*, Gretsi'99, Vannes, 13-17 Sept, Proceedings, pp.1125-1128, 1999

[3] T. Kim and T. Adali, *Fully complex multi-layer perceptron network for nonlinear signal processing*, Journal of VLSI Signal Processing, **32**, *1*, pp. 29-43, Aug./Sep. (2002)

[4] S.Sezer, Ph.Power, Neural Equalization Filtering for Highbandwidth QAM Channels, IEEE ICT 2001, Bucharest, 4-7 June, Proceedings, vol.**2**, pp.225-229, 2001

[5] I. Cha, S.Kassam, *Channel Equalization Using Complex-Valued Radial Basis Function Networks*, IEEE. Journal Select. Areas Commun,.**13**, *issue 1*, pp.122-131, (1995)

[6] S. Chen, S. McLaughlin, B. Mulgrew, *Complex-Valued Radial Basis Function Network. Network Architecture and Learning Algorithms, Signal Processing,* **35**, pp. 19-31, (1994)

[7] S. Chen, et al., *Complex-Valued Radial Basis Function Network. Part II: Application to Digital Communications Channel Equalization*, Signal Processing, **36**, pp. 175-188, (1994)

[8] Q. Gan, P. Saratchandran, N. Sundararajan, K. R. Subramanian, *A Complex Valued Radial Basis Function Network for Equalization of Fast Time Varying Channels*, IEEE Trans. On Neural Networks, **10**, *issue 4*, pp. 958-960, (1999)

[9] D. Jianping, N. Sundararajan, P. Saratchandran, *Communication Channel Equalization Using Complex-Valued Minimal Radial Basis Function Neural Network*, IEEE Trans. On Neural Networks,**13**, *No.3*, pp.687-696,(2002)

[10] J. Lee, C. D. Beach, N. Tepedelenlioglu, *Channel Equalization Using Radial Basis Function Network*, IEEE Int. Conf. On Neural Networks, vol. **4,** pp. 1924-1928, (1996)

[11] M.T.Madeira da Silva, M. Gerken, *A RNN-LC Hybrid Equalize*r, XI European Signal Processing Conference, Toulouse, 3-7 Sept, Proceedings, pp.341-344, 2002

[12] J.Choi, M. Bouchard, T.H.Yeap, *Decision Feedback Recurrent Neural Network Equalization with Fast Convergence Rate*, IEEE Trans. On Neural Networks, **20,** accept. for publication iul. (2005)

[13] G. Kechriotis, E. Zervas, E. S. Manolakos, *Using Recurrent Neural Networks for Adaptive Communication Channel Equalization*, IEEE Trans on Neural Networks, **5**, *2*, pp. 267-278, (1994).

[14] R. Parisi, E. Di Claudio, G. Orlandi, B. Rao, *Fast Adaptive Digital Equalization by Recurrent Neural Networks*, IEEE Trans. Signal Process., **45**, *Nov*., pp.2731-2739 , (1997)

[15] S.C.Ahalt, A.K. Krishnamurty, P.Chen,D.E.Melton, *Competitive Algorithms for Vector Quantization*, Neural Networks, **3**, pp.277-291, (1990)