Nonlinear Stationary Channel Equalization of QAM Signals using Multiplicative Neuron Model

Kavita Burse[†] , R. N. Yadav[†] and S. C. Shrivastava[†]

Maulana Azad National Institute of Technology, Bhopal, India.

Summary
A novel feed forward multiplicative neural network architecture with optimum number of nodes is used for adaptive channel equalization in this paper.The replacement of summation at each node by multiplication results in more powerful mapping because of its capability of processing higher-order information from training data. Performance comparison with Chebyshev neural network show that the proposed equalizer provides satisfactory results in terms of mean square error convergence curves and bit error rate performance at various levels of signal to noise ratios.

Key words:
Channel equalization, 4-QAM signal, multiplicative neuron, feed forward neural network.

1. Introduction

As higher-level modulation becomes more desirable to cope with the need for high-speed data transmission, nonlinear distortion becomes a major factor, which limits the data carrying capacity of digital communication sytems. Thermal noise, impulse noise, cross talk and the nature of the channel itself distort the transmitted data in amplitude and phase due to which temporal spreading and consequent overlap of individual pulses occurs. The presence of inter symbol interference (ISI) in the system introduces errors in the decision device at the receiver output. Therefore, in the design of the transmitting and receiving filters, the objective is to minimize the effects of ISI, and thereby deliver the digital data to its destination with the smallest error possible. Equalizers modelled as adaptive digital filters which shape the receiver's transfer function are ubiquitous in todays signal processing applications to combat ISI in dispersive channels. Adaptive filters achieve desired spectral characteristics of a signal by altering the filter coefficients and thereby the filter response according to a recursive optimization algorithm. Adaptive coefficients are required since some parameters of the desired processing operation (for instance, the properties of some noise signal) are not known in advance [1].
When significant noise is added to the transmitted signal linear boundaries are not optimal. The received signal at each sample instant may be considered as a nonlinear function of the past values of the transmitted symbols. Further, since the nonlinear distortion varies with time and from place to place, effectively the overall channel response becomes a nonlinear dynamic mapping and the problem is tackled using classification techniques. As shown in a wide range of engineering applications, neural network (NN) has been successfully used for modeling complex nonlinear systems and forecasting signal with relatively simple architecture [2]-[4]. A wide range of neural architectures are available for modeling the nonlinear phenomenon of channel equalization. Feed forward networks like multilayer perceptron (MLP) which contain an input layer, an output layer and one or more hidden layers possess nonlinear processing capabilities and universal approximation characteristic and have been successfully implemented as channel equalizers [5]-[7]. The back propagation which is a supervised learning algorithm is used as a training algorithm [8]. These neuron models process the neural inputs using the summing operation.

Recently, higher-order networks have drawn great attention from researchers due to their superior performance in nonlinear input-output mapping, function approximation, and memory storage capacity. Some examples are Product unit neural network (PUNN), Sigma-Pi network (SPN), Pi-Sigma network (PSN) etc. They allow neural networks to learn multiplicative interactions of arbitrary degree. Multiplication plays an important role in neural modeling of biological behavior and in computing and learning with artificial neural networks. The multiplicative neuron contains units which multiply their inputs instead of summing them and thus allow inputs to interact nonlinearly. Multiplicative node functions allow direct computing of polynomials inputs and approximate higher order functions with fewer nodes. Thus they may present better approximation capability and faster learning times than the classical MLP because of their capability of processing higher-order information from training data [9]-[11]. The remaining of the paper is organized as follows: section 2 describes the basic adaptive channel equalizer

Manuscript received May 5, 2009
Manuscript revised May 20, 2009

scheme. In section 3 learning rule for multiplicative neuron is derived, section 4 overviews the Chebyshev functional link artificial neural network (CFLANN), section 5 provides the simulation and results and section 6 concludes the paper.

2. Adaptive Channel Equalization

The block diagram of adaptive equalization in figure 1 is described as follows. The external time dependant inputs consist of the sum of the desired signal *d(k)*, the channel nonlinearity NL and the interfering noise *v(k)*. The adaptive filter has a finite impulse response(FIR) structure. The impulse response is equal to the filter coefficients.The coefficients for a filter of order *p* are defined as

$$W_k = [w_k(0), w_k(1), ..., w_k(p)]^T \quad (1)$$

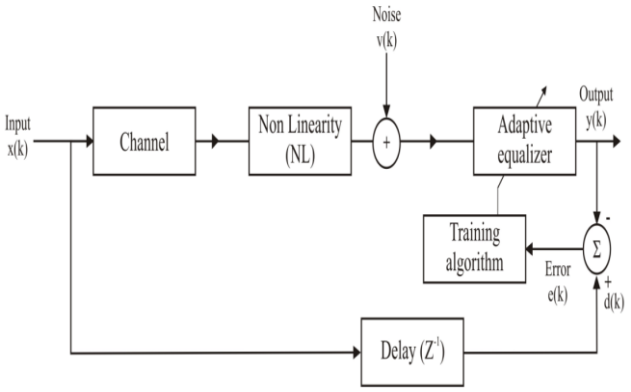


Fig. 1 Block diagram of an adaptive channel equalizer

A predefined delayed version of the original signal forms the training sequence to provide reference points for the adaptation process. The criterion for optimization is a cost function or the error signal which is the difference between the desired and the estimated signal given by

$$e(k) = d(k) - y(k) \quad (2)$$

The desired signal is estimated by convolving the input signal with the impulse response expressed as

$$d(k) = W_k^T x(k) \quad (3)$$

where, $x(k) = [x(k), x(k-1), ..., x(k-p)]^T$ is the input signal vector. The filter coefficients are updated at every time instant as

$$W_{k+1} = W_k + \Delta W_k \quad (4)$$

ΔW_k is a correction factor for the filter coefficients.

The optimization algorithm can be linear or nonlinear. Figure 2 shows a feed forward multiplicative neural network (MNN).

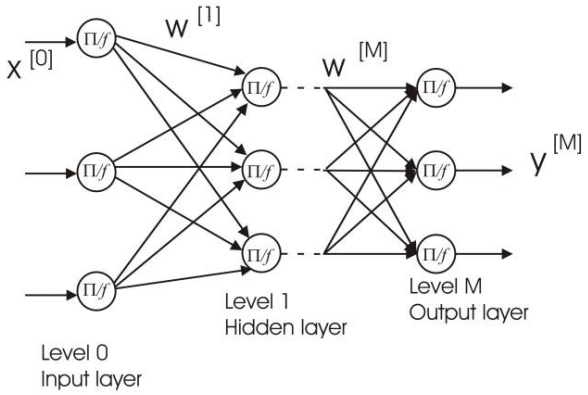


Fig. 2 Multiplicative neural network

The block diagram of a channel equalizer using MNN is shown in figure 3. The transmitter sends a known training sequence to the receiver. A sequence of 3000, equiprobable, 4-QAM complex valued symbol set, in which the input signal takes one of 4 different values given by all possible combinations of { -1, 1} + j*{ -1, 1}, where j = sqrt(-1) is generated. In the absence of the noise the output signal occupies well-defined M states of the M-QAM signal constellation. When the signal is passed through the nonlinear channel, it becomes a stochastic random process. Decision boundaries can be formed in the observed pattern space to classify the observed vectors between 4 classes. For equalization, the adaptive filter is used in series with the unknown system on the test signal $d(k)$ by minimizing the squared difference between the adaptive equalizer output and the delayed test signal. The task of the equalizer is to set its coefficients in such a way that the output $y(k)$ is a close estimate of the desired output $d(k)$. Depending on the value of the channel output vector, the equalizer tries to estimate an output, which is close to one of the transmitted values. The neural equalizer separately processes the real and imaginary part using the multiplicative, split complex, neural network model [12]-[13]. This can be viewed as 2 real valued activation functions for processing the in phase and quadrature component of the 4QAM signal. The split complex approach is generally used to avoid singular points and critical selection of network parameters like the weights, bias, learning rate and momentum factor.

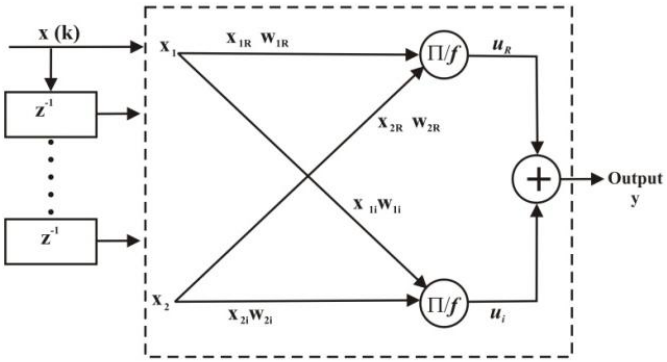


Fig. 3 Multiplicative neural network based channel equalizer

The real R and imaginary I parts of the input signal are split as

$$F(x(t)) = f(x_{1R}(t), x_{2R}(t)) + if(x_{1I}(t), x_{2I}(t)) \quad (5)$$

Where, the input $x_1(t) = x_{1R}(t) + ix_{1I}(t)$ and

$$x_2(t) = x_{2R}(t) + ix_{2I}(t) \quad (6)$$

3. Learning Rule for Multiplicative Neuron

An error back propagation (BP) based learning using a norm-squared error function is described as follows [14]-[15]. The algorithm is first developed for single hidden layer network which is then extended to multi layer NN architecture. The aggregation function is considered as a product of linear functions in different dimensions of space. A bipolar sigmoidal activation function is used at each node. This kind of neuron itself looks complex in the first instance but when used to solve a complicated problem needs less number of parameters as compared to the existing conventional models.

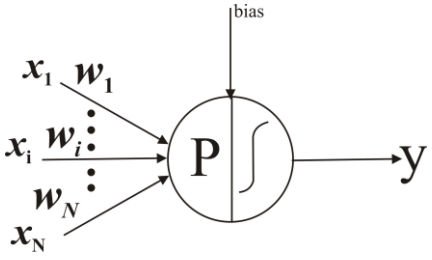


Fig.4 Structure of a single multiplicative neuron

Here the operator P is a multiplicative operation as given in equation 7. The aggregation u before applying activation function is given by:

$$u = \prod_{i=1}^{n} (w_i x_i + b_i) \quad (7)$$

The output at the node y is given by

$$y = f(u) = \frac{1 - e^{-u}}{1 + e^{-u}} \quad (8)$$

The mean square error is given by

$$E = \frac{1}{2N} \sum_{p=1}^{N} (y^p - y_d^p)^2 \quad (9)$$

where, p is the number of input patterns.

The weight update equation for the split complex back propagation algorithm is given by

$$\Delta w_i = -\eta \frac{dE}{d w_i}$$

$$= -\frac{1}{2}\eta(y-d)(1+y)(1-y)\frac{u}{(w_i x_i + b_i)} x_i \quad (10)$$

where, η is the learning rate and d is the desired signal. The bias is updated as

$$\Delta b_i = -\eta \frac{dE}{d b_i}$$

$$= -\frac{1}{2}\eta(y-d)(1+y)(1-y)\frac{u}{(w_i x_i + b_i)} \quad (11)$$

$$w_i^{new} = w_i^{old} + \Delta w_i \quad (12)$$

$$b_i^{new} = b_i^{old} + \Delta b_i \quad (13)$$

The weights are updated after the entire training sequence has been presented to the network once. This is called learning by epoch. The algorithm is extended to train multi layer multiplicative feed forward neural network as follows.

The symbols used are:

N_o is the number of inputs in the input layer.

n is the number of hidden layers in the FF network.

N_n is the number of neurons in the n^{th} hidden layer.

K is the number of outputs in the output layer.

j_n is the j^{th} neuron of the n^{th} hidden layer.

y_{jn}^{n} is the output of the j^{th} neuron of the n^{th} hidden layer.

y_{dk} is the desired output of the k^{th} neuron in the output layer.

y_k is the actual output of the k^{th} neuron in the output layer.

w_{jnjn-1} is the weight of the connection between j^{th} neuron of the $(n-1)^{th}$ layer and the j^{th} neuron of the n^{th} layer.

b_{jnjn-1} is the bias of the connection between j^{th} neuron of the $(n-1)^{th}$ layer and the j^{th} neuron of the n^{th} layer.

The output of the j^{th} neuron in the first hidden layer is given as

$$y_{j1}^{1} = f\left(\prod_{j0=1}^{N0}(w_{j1j0}x_{j0} + b_{j1j0})\right) \quad (14)$$

for j_1=1,2,….,N_1 and x_{j0} represents j^{th} input in the input layer and $f(.)$ is the activation function defined by

$$f(y) = \frac{1-e^{-y}}{1+e^{-y}} \quad (15)$$

The output of the j^{th} neuron in the second hidden layer is given as

$$y_{j2}^{2} = f\left(\prod_{j1=1}^{N1} w_{j2j1}y_{j1}^{1} + b_{j2j1})\right); \text{ for } j_2=1,2,\ldots,N_2 \quad (9)$$

The output of the j^{th} neuron in the n^{th} hidden layer is given as:

$$y_{jn}^{n} = f\left(\prod_{jn-1=1}^{Nn-1}(w_{jnjn-1}y_{jn-1}^{n-1} + b_{jnjn-1})\right) \quad ; \quad \text{for} \quad j_n=1,2,\ldots,N_n \quad (10)$$

The output of the k^{th} neuron in the output layer is given as

$$y_k = f\left(\prod_{jn=1}^{Nn-1}(w_{kjn}y_{jn}^{n} + b_{kjn})\right); \text{ for k=1,2,….,k} \quad (16)$$

A simple gradient descent rule, using a mean square error function is used for computation of weight update.

$$E_{MSE} = \frac{1}{2PK}\sum_{k=1}^{K}\sum_{p=1}^{P}(y_{dk}^{p} - y_k^{p})^2 \quad (17)$$

Where y_k^p and y_{dk}^p are the actual and desired values, respectively, of the output of the k^{th} neuron for the p^{th} pattern in the output layer. P is the number of training patterns in the input space. The weights are updated as below. Weights between output layer and the n^{th} hidden layer are given by:

$$\Delta w_{kjn} = -\eta\frac{\partial E_{MSE}}{\partial w_{kjn}} = \eta\delta_k\frac{\left[\prod_{jn=1}^{Nn}(w_{kjn}y_{jn}^{n} + b_{kjn})\right]}{(w_{kjn}y_{jn}^{n} + b_{kjn})}.y_{jn}^{n} \quad (18)$$

$$\delta_k = \frac{1}{PK}\left[\sum_{k=1}^{K}\sum_{p=1}^{P}(y_{dk}^{p} - y_k^{p}).\left[(1/2)(1+y_k^{p})(1-y_k^{p})\right]\right] \quad (14)$$

$$\Delta b_{kjn} = \eta\delta_k\frac{\left[\prod_{jn=1}^{Nn}(w_{kjn}y_{jn}^{n} + b_{kjn})\right]}{(w_{kjn}y_{jn}^{n} + b_{kjn})} = \frac{\Delta w_{kjn}}{y_{jn}^{n}} \quad (19)$$

Weights between n^{th} and $(n-1)^{th}$ hidden layer

$$\Delta w_{jnjn-1} = -\eta\frac{\partial E_{MSE}}{\partial w_{jnjn-1}} = \frac{\eta}{PK}\left[\sum_{k=1}^{K}\sum_{p=1}^{P}(y_{dk}^{p} - y_k^{p}).\frac{\partial y_k^{p}}{\partial y_{jn}^{n}}\right].\frac{\partial y_{jn}^{n}}{\partial w_{jnjn-1}}$$

$$= \eta\delta_k\frac{\left[\prod_{jn=1}^{Nn}(w_{kjn}y_{jn}^{n} + b_{kjn})\right]}{(w_{kjn}y_{jn}^{n} + b_{kjn})}.w_{kjn}.\frac{\partial y_{jn}^{n}}{\partial w_{jnjn-1}} \quad (20)$$

$$\Delta b_{jnjn-1} = \frac{\Delta w_{jnjn-1}}{y_{jn-1}^{n-1}} \quad (21)$$

Similarly, we can write equations for weight change between the hidden layer 1 and the input layer.

The weights and biases are updated as

$$w_i^{new} = w_i^{old} + \Delta w_i \quad (22)$$

$$b_i^{new} = b_i^{old} + \Delta b_i \quad (23)$$

4. Chebyshev Neural Network Architecture

In (CFLANN) polynomials are chosen to be the expanding nonlinear functions which map the input signal vector $x = [x_1, x_2 .....x_n]^T$ by N linearly independent functions $\phi = [\phi_1(x), \phi_2(x) .....\phi_N(x)]^T$. The linear combination of these function values can be presented in a

matrix form, that is, $S = W\phi$ where $S = [s_1, s_2 ..... s_m]^T$, and W is the $m \times N$ dimensional weighting matrix. The matrix S is fed into a bank of identical nonlinear functions to generate the equalized output

$Y = [y_1 y_2 .... y_m]^T$ where, $y_j = \rho(s_j)$, $j = 1,2,....m$.

CFLANN does not need hidden layers in its circuit structure. Being similar to MLP, the CFLANN also uses BP algorithm to train the neural networks [16]-[17]. The input signals to a CFLANN network are nonlinearly mapped into the output signal space, so the equalizer has also the ability to resolve the equalization problems for nonlinear channels.

5. Simulations and Results

To study the BER performances the equalizer structure was trained with 3000 iterations and tested over 10000 samples. A nonminimum phase stationary channel with the following transfer function is used:

CH1: 1.0

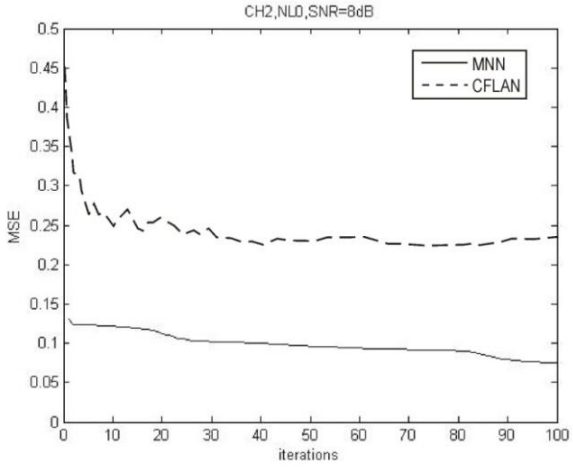
CH2: $0.447 + 0.894z^{-1}$

The nonlinearity introduced is

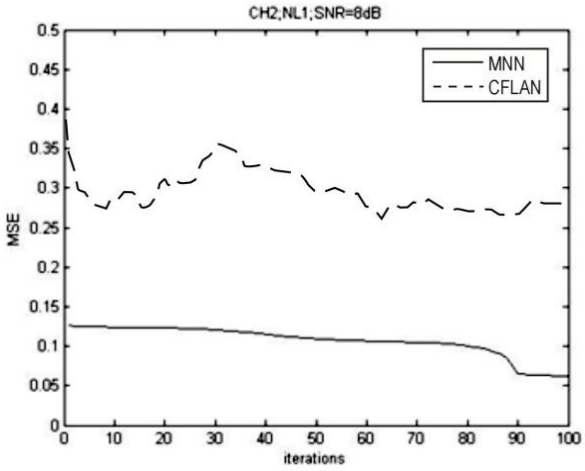
NL0: $b(k) = a(k)$

NL1: $b(k) = a(k) + 0.2a^2(k) - 0.1a^3(k)$

The data set has been pre-processed by normalizing them between 0.1 and 1. In all simulations, the results reported are the average of several runs in each case. The convergence characteristics of the MSE during the training mode for CH2 at 8 dB SNR are plotted in figure 5.



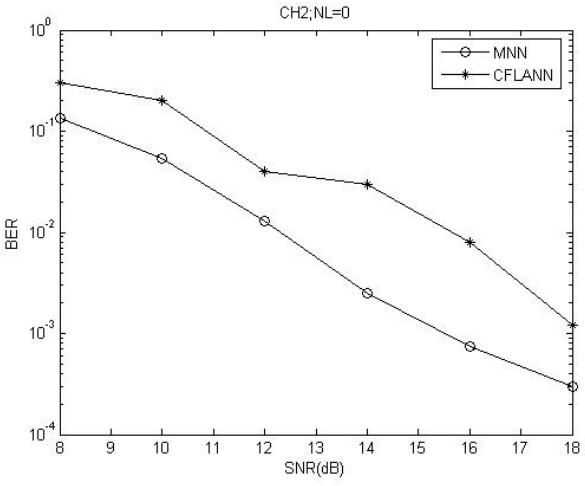
(a)



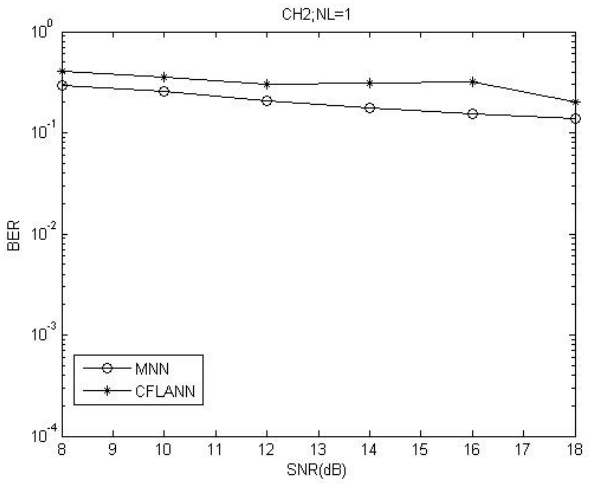
(b)

Fig. 5. Convergence curves of MSE for CH = 2 at SNR = 8 dB: (a) NL = 0, (b) NL = 1

The multiplicative neural network (MNN) equalizer has faster speed of convergence and smaller steady state MSE than CFLANN in either linear or nonlinear environment. In case of the CFLANN neural equalizer the input is expanded to 25 nodes and the number of output nodes is 2, in reduced decision feedback CFLANN the number of input nodes is 17 and the number of output node is 2 where as in case of MNN equalizer the structure is of the type $6 \times 6 \times 2$. The BER performance for various SNR is plotted in figure 6.



(a)



(b)

Fig. 6. BER vs. SNR, for CH = 2, (a) NL = 0, (b) NL = 1

4. Conclusion

A high order feed forward neural network equalizer with multiplicative neuron is proposed in this paper. Use of multiplication allows direct computing of polynomial inputs and approximation with fewer nodes. Performance comparison in terms of convergence rates and BER performance suggest the better classification capability of the proposed MNN equalizer over CFLANN.

References

[1] S. Haykin, *Adaptive Filter Theory*, Pearson Education, 2005, pp. 22-25.

[2] D.C. Park, M. A. El-Sharkawi, and R. J. Marks II, "Adaptively trained neural network," *IEEE Trans. Neural Networks*, vol. 2, pp. 334–345, May 1991.

[3] Pham DT, Liu X, *Neural networks for identification, prediction and control*, London: Springer, 1995.

[4] S. Bang, S. H. Sheu, and J. Bing, "Neural network for detection of signals in communication," *IEEE Trans. Circuits Syst. I*, vol. 43(8), pp. 644-655, Aug. 1996.

[5] S. Chen, G. Gibson, C. Cown, and P. Grant, "Adaptive equalization of finite nonlinear channels using multilayer perceptrons," *Signal Processing*, vol. 20, pp. 107-119, June 1990.

[6] G. Gibson, S. Siu, and C. Cowan, "Multilayer perceptron structures applied to adaptive equalizers for data communications," in *Proc. ICASSP*, May 1989, Glasgow, U.K., pp. 1183-1186

[7] T. Kim, T. Adali," Fully complex multi-layer perceptron network for nonlinear signal processing," *J. VLSI Signal Process.*, vol. 32, No. 1, pp. 29-43, 2002.

[8] Q. Zhang, "Adaptive equalization using the back propagation algorithm," *IEEE Trans. Circuits Syst.*, vol. 37, pp. 848–849, June 1990.

[9] C.L. Giles and T. Maxwell," Learning, invariance, and generalization in high-order neural networks," *Applied Optics*, vol. 26, no. 23, pp. 4972-4978, 1987.

[10] E.M. Iyoda, K. Hirota, and F. J. Von Zuben," Sigma-Pi cascade extended hybrid neural networks," *Journal of Advanced Computational Intelligence*, vol.6, no. 3, pp. 126-134, 2002.

[11] M Schmitt," On the complexity of computing and learning with multiplicative neurons," *Neural Comput.* vol. 14, no.2, pp. 241-301, Feb. 2002.

[12] A. Kantsila, M. Lehtokangas, J. Saarinen," Complex RPROP-algorithm for neural network equalization of GSM data bursts", *Neurocomputing*, vol. 61, pp. 339- 360, 2004.

[13] Kavita Burse, R.N. Yadav and S.C. Shrivastava," Complex Channel Equalization using Polynomial Neuron Model," in *Proc. IEEE 3rd Int. Symposium on Information Technology*, Kuala Lumpur, Malaysia, Aug. 26-29, 2008, pp. 771-775.

[14] R.N. Yadav, P.K. Kalra and J. John," Time series prediction with single multiplicative neuron model," *Applied soft computing*, vol 7, pp 1157-1163, 2007.

[15] Kavita Burse, R.N. Yadav, S.C. Shrivastava, Vishnu Pratap Singh Kirar," A compact pi network for reducing bit error rate in dispersive FIR channel noise model," to appear in *Proc. of WCSET:2009,World Congress on Science, Engineering and Technology*, Feb. 25-27, Penang, Malaysia.

[16] Jagdish Chandra Patra, Wei Beng Poh, Narendra S. Chaudhari and Amitabha Das "Nonlinear Channel Equalization with QAM Signal Using Chebyshev Artificial Neural Network," *Proceedings of International Joint Conference on Neural Networks*, Montreal, Canada, July 31 - August 4, 2005,pp.3214-3219.

[17] Wan-De Weng, Che-Shih Yang , Rui-Chang Lin "A channel equalizer using reduced decision feedback Chebyshev functional link artificial neural networks," *Information Sciences*, vol.177, no. 13, pp. 2642-2654, 2007.

Kavita Burse received her Bachelor of Engineering (Electronics and Communication) from Shri Govind Ram Seksaria Institute of Science and Technology, Indore, India in 1992 and M.Tech (Digital Communication) from Maulana Azad National Institute of Technology, Bhopal, India. Currently she is pursuing Ph.D. degree from the same Institute. She is faculty with the Department of Electronics and Communication at Truba Institute of Engineering and Technology, Bhopal, India. She is associate member CSI and life member ISTE.

R. N. Yadav received his Bachelor of Engineering degree from Motilal Nehru Regional Engineering College Allahabad, India, M.Tech degree from Maulana Azad College of Technology, Bhopal, India and P.hD. degree from Indian Institute of Technology, Kanpur, India in 1993, 1997 and 2005 respectively. Currently he is Assistant Professor in the Department of Electronics and Communication Engineering, Maulana Azad National Institute of Technology, Bhopal, India. He is a life member of IETE and IE(I). He has authored and reviewed more than twenty papers in international Journals and conferences of repute.

S.C. Shrivastava received his Bachelor and Master degree in Engineering from Government Engineering College, Jabalpur, India in 1968 and 1970 respectively and P.hD. degree in 1994 from Barkatullah University, Bhopal, India. He is a life member of IETE, IE(I) and ISTEE. Currently he is Professor and Head, Department of Computer Science and Engineering, Maulana Azad National Institute of Technology, Bhopal, India. He has authored several papers in national and international Journals and conferences.