

The Design of an SCFNN Based Nonlinear Channel Equalizer

WAN-DE WENG^{1,2}, RUI-CHANG LIN^{1,3} AND CHUNG-TA HSUEH²

¹*Graduate School of Engineering Science and Technology*

²*Department of Electrical Engineering*

National Yunlin University of Science and Technology

Yunlin, 640 Taiwan

E-mail: {wengwd, g9212718}@yuntech.edu.tw

³*Department of Electronic Engineering*

Nan Kai Institute of Technology

Nantou, 542 Taiwan

E-mail: rclin@nkc.edu.tw

The design of a self-constructing fuzzy neural network (SCFNN)-based digital channel equalizer is proposed in this paper. We demonstrate that the SCFNN-based digital channel equalizer possesses the ability to recover the channel distortion effectively. The performance of SCFNN is compared with that of the adaptive-based-network fuzzy inference system (ANFIS) and the optimal Bayesian solution. Simulations were carried out in both real-valued and complex-valued nonlinear channels to demonstrate the flexibility of the proposed equalizer. The experimental results show that the performance of SCFNN can be close to that of the Bayesian optimal solution and ANFIS, while the hardware requirement of the trained SCFNN-based equalizer is much lower.

Keywords: SCFNN, ANFIS, Bayesian solution, channel equalizer, neural network

1. INTRODUCTION

Due to the increasing demand for high speed data transmission in communication systems, channels have inevitably become much noisier and more crowded than ever. This has consequently resulted in signal distortion at the receiver end. Generally, equalizers are used at the receiver end to recover distorted signals. Traditionally, the most commonly used strategy for reducing ISI is the linear finite-duration impulse response (FIR) filter based equalizer [1]. However, the performance of linear channel equalizers employing linear filters with an FIR or lattice structure is quite poor, especially when the channel encounters severe nonlinear distortion. In this case, we need to use nonlinear channel equalizers in order to attain a lower bit error rate (BER), lower mean squared error (MSE), and higher convergence rate than is possible with linear equalizers [2]. Since both fuzzy logic and artificial neural networks can perform complex mappings between their input and output spaces, their decision regions over the output spaces generally have nonlinear decision boundaries. Accordingly, it is believed that these two techniques are potentially suitable for nonlinear channel equalization. During the past 20 years, several algorithms for maximum-likelihood sequence estimation (MLSE) equalization have been proposed. For example, Chugg [3] applied MLSE to estimate unknown

Received June 30, 2004; revised November 29, 2004; accepted January 13, 2005.
Communicated by Liang-Gee Chen.

channels. Recently, MLSE was applied successfully to direct-sequence spread spectrum systems [4]. Although these papers showed that MLSE equalizers possess superior bit-error-rate performance, the high computational complexity could potentially restrict their application. Due to the theoretic importance of the Bayesian filter, most published papers about fuzzy filters have compared their system performance with that of the optimal Bayesian equalizer. The fuzzy nonlinear channel equalizer proposed in [5] can achieve performance very close to that of the optimal equalizer. Although the equalization performance we achieve is about 0.5 dB worse than the optimal solution, we attain large savings in the computational and hardware implementation costs. The trade-off between performance degradation and cost is obviously acceptable.

The self-constructing fuzzy neural network (SCFNN) is capable of constructing a simple network without the need for knowledge of the transmission channel. This is due to the SCFNN's ability to self-adjust the location of the input space fuzzy partition, so there is no need to estimate in advance the number and locations of equalizer input states. Thus, carefully setting conditions for the increased demand for fuzzy rules will make the architecture of the constructed SCFNN fairly simple. These advantages motivated us to build a channel equalizer that employs the SCFNN structure. The performance of an SCFNN based equalizer is compared with the popular adaptive based network fuzzy inference system (ANFIS) and the Bayesian optimal solution. An analysis of the characteristics of these fuzzy neural networks, such as the hardware structure, learning time, and output signal distributions, is presented. Our simulation results show that, when SCFNN is used as a digital channel equalizer, performance is very close to that of the Bayesian optimal solution and ANFIS. Moreover, since SCFNN is a self-constructed fuzzy neural network, it can adjust the membership function parameters of the fuzzy rules during the learning processes and will restrict the generation of fuzzy rules under stricter conditions. Therefore, the hardware structure of a well-trained SCFNN-based equalizer is simpler than that of the other two structures.

The rest of this paper is organized as follows: In section 2, the problem of complex channel equalization for a digital communication system is formulated. The structure, the inference output, the structure learning, and the parameter learning algorithms of SCFNN are described in section 3. In section 4, we present simulation results obtained under the assumption that the equalizer is operated over a real and complex nonlinear channel. Finally, section 5 summarizes this paper and indicates some of our future research directions.

2. COMPLEX CHANNEL EQUALIZATION PROBLEM

Consider the baseband discrete-time data transmission system in which 4-QAM modulated signals are transmitted. The original complex-valued message symbol at time instant kT is denoted by $t(k)$, where T is the symbol duration. The real part and the imaginary part of $t(k)$ are assumed to be independent and statistically independent, taking equiprobable values over $\{+1, -1\}$. The output of the linear dispersive FIR channel at time instant kT can be written as

$$a(k) = \sum_{i=0}^{n_h-1} h(i) \cdot t(k-1), \quad (1)$$

where $h(i)$, $i = 0, 1, \dots, n_h - 1$, are the channel tap values and n_h is the length of the FIR channel [2].

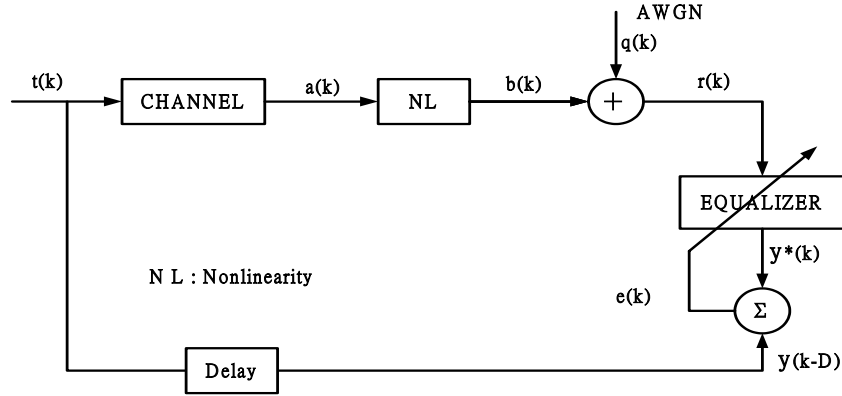


Fig. 1. Digital communication model.

Fig. 1 shows a block diagram of the discrete-time model of a digital transmission system with an equalizer [2]. The combined effect of the transmitter filter, the transmission medium, and any other components influencing the channel characteristics are included together in the “CHANNEL” block. The block labeled “NL” denotes the nonlinear distortion caused by the channel during transmission. The overall channel model can be easily fitted to either linear or nonlinear transmission environments, and its output can be written as

$$b(k) = f[a(k), a(k-1), \dots, a(k-n_h+1), h(0), h(1), \dots, h(n_h-1)], \quad (2)$$

where $f(\cdot)$ is the nonlinear function designated by the “NL” block. Assume further that the transmitted signal is corrupted by additive white Gaussian noise (AWGN) $q(k)$ of zero-mean and variance σ^2 . The signal $r(k)$ received at the receiver can, thus, be represented as its in-phase and quadrature components as $r(k) = r_{k,I} + jr_{k,Q}$. The purpose of the equalizer is to recover the transmitted symbol $t(k)$ or $t(k-\tau)$ from knowledge of the received signal samples with the least error rate, where τ is the transmission delay associated with the physical channel.

3. SELF-CONSTRUCTING FUZZY NEURAL NETWORK

3.1 The Structure and Inference Output of the SCFNN

The fuzzy logic rule with a constant consequence of the following form is adopted in SCFNN [6]:

$$R_j: \text{IF } r_1 \text{ is } A_{j1} \dots \text{ and } r_n \text{ is } A_{jn}, \text{ THEN } y_j = \omega_j, \quad (3)$$

where r_i , $i = 1, \dots, n$, and y_j denote the input and output variables, respectively, A_{ji} represents the linguistic term of the precondition part with membership function $\mu_{A_{ji}}$, ω_j denotes the constant consequent part, and n is the number of input variables.

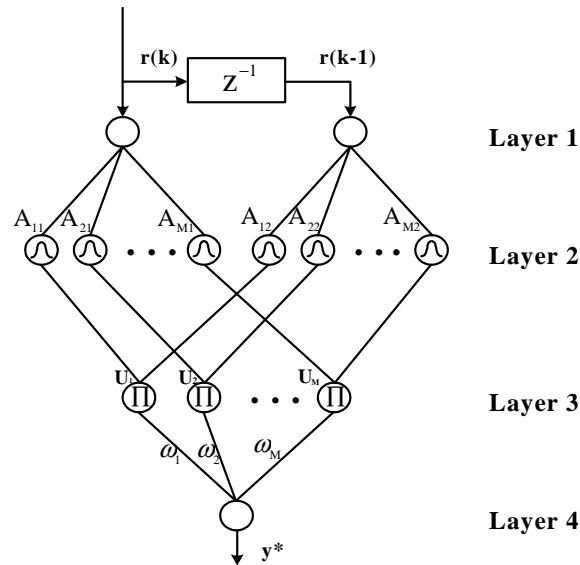


Fig. 2. Schematic diagram of SCFNN.

The structure of SCFNN is shown in Fig. 2, and the functions of the nodes in each layer are briefly described as follows:

Layer 1 This layer consists of one or more input nodes, each corresponding to one signal received from the channel. These nodes simply pass the input signal to the next layer. In our study, the input variables are the digital sequences $r(k)$ and $r(k-1)$.

Layer 2 Each node in this layer acts as a linguistic label of one of the input variables in Layer 1, i.e., the membership value specifying the degree to which an input value belongs to a fuzzy set is determined in this layer. The following Gaussian function is adopted as the membership function:

$$\mu_{A_{ji}} = \exp\left(-\frac{(r_i - m_{ji})^2}{\sigma_{ji}^2}\right), \quad (4)$$

where m_{ji} and σ_{ji} are its mean and standard deviation, respectively, of the Gaussian functions of the j -th term associated with the i -th input variable r_i .

Layer 3 Each node in this layer represents the precondition part of one fuzzy logic rule. At each of these nodes, the incoming signals are multiplied, and the output product term is used as the firing strength of the fuzzy rule. In mathematical form, we have, for the j -th rule node,

$$u_j = (\mu_{A_{j1}}(x_1)) \cdot (\mu_{A_{j2}}(x_2)) \dots (\mu_{A_{jn}}(x_n)) = \prod_{i=1}^n \mu_{A_{ji}}(x_i), \quad (5)$$

where u_j is the output of the j -th rule node.

Layer 4 This layer functions as a defuzzifier. The only node in this layer sums all the incoming signals to obtain the final inferred result, that is,

$$y^* = \sum_{j=1}^M u_j \omega_j, \quad (6)$$

where the link weight ω_j is the output action strength associated with the j -th rule and y^* is the output of SCFNN. The consequent part of one fuzzy logic rule is implicitly contained in ω_j .

3.2 Online Learning Algorithm for SCFNN

Two types of learning algorithms, namely, structure learning and parameter learning algorithms, are used to construct SCFNN. The structure learning algorithm is used to find the proper input space fuzzy partitions and fuzzy logic rules based on two goals: (i) to minimize the number of generated rules and (ii) to minimize the number of fuzzy sets on the universe of discourse of each input variable. The structure and parameter learning algorithms are described in the rest of this section in detail.

3.2.1 Structure learning

The main task in structure learning is to decide whether or not to add a new node (membership function) in Layer 2 and the associated fuzzy logic rule in Layer 3. Since one cluster formed in the input space corresponds to one potential fuzzy logic rule, the firing strength of a rule for each incoming piece of data r_i can be represented as the degree to which that the incoming piece of data belongs to the cluster. The firing strength obtained from Eq. (5) is used as the degree measures μ_j , $j = 1, \dots, M$, where M is the number of existing rules when the learning process is being performed. According to the degree measures, the criterion for generating a new fuzzy rule for the new incoming piece of data is, to the maximum degree,

$$\mu_{\max} = \max_{1 \leq j \leq M} \mu_j. \quad (7)$$

If $\mu_{\max} \leq \mu_{\min}$, then a new membership function will be generated, where $\mu_{\min} \in (0, 1)$

is a pre-specified threshold parameter that should be decayed during the learning process to limit the size of SCFNN. Next, the mean and standard deviations of the new membership function are assigned pre-specified values based on heuristic or prior knowledge. The mean and standard deviations of the new membership function are selected as

$$m_i^{new} = x_i, \quad (8)$$

$$\sigma_i^{new} = \sigma_i, \quad (9)$$

where x_i is the new incoming piece of data and σ_i is a pre-specified constant.

3.2.2 Parameter learning

The central task of the parameter learning algorithm for SCFNN is to obtain the adaptive rules that can be used to adjust the parameters of the network based on a given set of input-output pairs. If the parameters of the network are considered as elements of a parameter vector, then the learning process involves determining the vector which minimizes a given energy function. The gradient of the energy function with respect to the vector is computed, and the vector is adjusted along the negative gradient. This method is generally referred to as the back-propagation (BP) learning rule because the gradient vector is calculated in the direction opposite to the flow of the output of each node. To describe the online parameter learning algorithm of SCFNN using the supervised gradient descent method, we must first define the energy function as

$$E = \frac{1}{2}(y - y^*)^2, \quad (10)$$

where y is the desired output associated with the input pattern and y^* is the inferred output of SCFNN. The parameter learning algorithm based on back-propagation is described in the following.

Layer 4 The link weight ω_j is updated by the amount $\Delta\omega_j$:

$$\Delta\omega_j = \eta_\omega(y - y^*)u_j, \quad (11)$$

where factor η_ω is the learning-rate parameter of the link weight. The link weights in Layer 4 are updated according to the following equation:

$$\omega_j(N + 1) = \omega_j(N) + \Delta\omega_j, \quad (12)$$

where N denotes the iteration number of the j -th link.

Layer 3 In this layer the error terms associated with m_{ji} and σ_{ji} are calculated as

$$\Delta m_{ji} = \eta_m(y - y^*)\omega_j u_j \frac{2(x_i - m_{ji})}{\sigma_{ji}^2}, \quad (13)$$

$$\Delta\sigma_{ji} = \eta_{\sigma}(y - y^*)\omega_j u_j \frac{2(x_i - m_{ji})^2}{\sigma_{ji}^3}, \tag{14}$$

where η_m and η_{σ} are the learning-rate parameters of the mean and standard deviations of the Gaussian function, respectively. The mean value and standard deviations of the membership functions in this layer are updated as

$$m_{ji}(N + 1) = m_{ji}(N) + \Delta m_{ji}, \tag{15}$$

$$\sigma_{ji}(N + 1) = \sigma_{ji}(N) + \Delta\sigma_{ji}. \tag{16}$$

3.3 Computational Complexity Analysis of SCFNN

Recently, there has been considerable interest in the development of a dedicated hardware implementation [7, 8] which facilitates high speed processing. However, the main drawback of such an approach is that it is only cost effective for high-volume problems and requires the application of general-purpose or programmable hardware tools. Although the implementation of an SCFNN based equalizer will encounter the same problems mentioned above, the problems can be relieved through careful programming. For example, realizing the Gaussian functions in SCFNN requires a large amount of computational effort. Generally, look-up tables are the preferred approach to implementing this function. But this still requires a great deal of hardware. If hardware cost is a major concern in the design of the equalizer, a second-order approximating function with carefully selected coefficients is a reasonable choice.

In the following, we will use several tables to present the computation complexity of SCFNN at different stages of the learning process. Table 1 shows the whole computational burden when the process is calculating the inference output, where N_{hid} denotes the number of hidden nodes. Before the next training pattern enters SCFNN, the link weights and the membership function parameters must be adjusted. The computational requirements for adjusting ω_j , m_{ji} , and σ_{ji} are summarized in Tables 2, 3, and 4, respectively.

Table 1. Computational complexity of SCFNN when calculating the inference output.

Operation	Sub	Mul	Div	Exp	Min	Mul	Add
no. of operands	2	2	2	1	N_{hid}	2	N_{hid}
no. of times	1	3	2	1	1	N_{hid}	1

Table 2. Computational complexity of SCFNN when adjusting ω_j .

Operation	Sub	Mul	Add
no. of operands	2	3	2
no. of times	1	N_{hid}	N_{hid}

Table 3. Computational complexity of SCFNN when adjusting m_{ji} .

Operation	Sub	Mul	Div	Sub	Mul
no. of operands	2	2	2	2	5
no. of times	$2N_{hid}$	$2N_{hid}$	$2N_{hid}$	1	$2N_{hid}$

Table 4. Computational complexity of SCFNN when adjusting σ_{ji} .

Operation	Sub	Mul	Mul	Mul	Div	Sub	Mul
no. of operands	2	2	2	3	2	2	5
no. of times	$2N_{hid}$	$2N_{hid}$	$2N_{hid}$	$2N_{hid}$	$2N_{hid}$	1	$2N_{hid}$

4. SCFNN-BASED CHANNEL EQUALIZATION

The simulation results of applying the SCFNN structure to a 4-QAM channel equalization problem will be presented in this section. For the purpose of making a comparison with published results, we apply the nonlinear non-minimum phase and complex channel model proposed [2, 9-11]. In the following subsections, detailed performance studies on SCFNN under various models as well as the performance comparison of SCFNN with other structures will be presented.

ANFIS [12] uses a hybrid learning algorithm to identify the membership function parameters of single-output, Sugeno type fuzzy inference systems (FIS). A combination of the least-squares and back-propagation gradient descent methods is used to train of FIS membership function parameters. An ANFIS-based channel equalizer has been investigated under the same channel models for comparison purposes.

4.1 Real Nonlinear Channel

The normalized transfer function of the channel is given as in [2, 9]:

$$H_1(z) = 0.447 + 0.894z^{-1}, \quad (17)$$

which corresponds to a non-minimum phase channel. The following three different nonlinear distortions with increasing complexity are considered:

$$\begin{aligned} NL = 1, & \quad b(k) = \tanh(a(k)), \\ NL = 2, & \quad b(k) = a(k) + 0.2a^2(k) - 0.1a^3(k), \\ NL = 3, & \quad b(k) = a(k) + 0.2a^2(k) - 0.1a^3(k) + 0.5\cos(\pi a(k)), \end{aligned} \quad (18)$$

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

Assume that a sequence of 4-QAM modulated signals are transmitted through the channel, where the real and the imaginary parts of the transmitted symbols are inde-

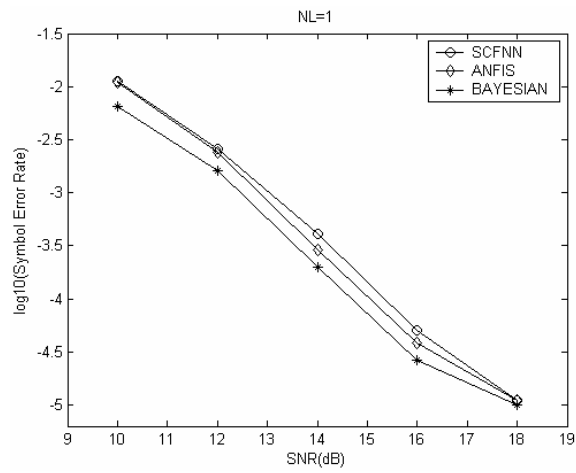
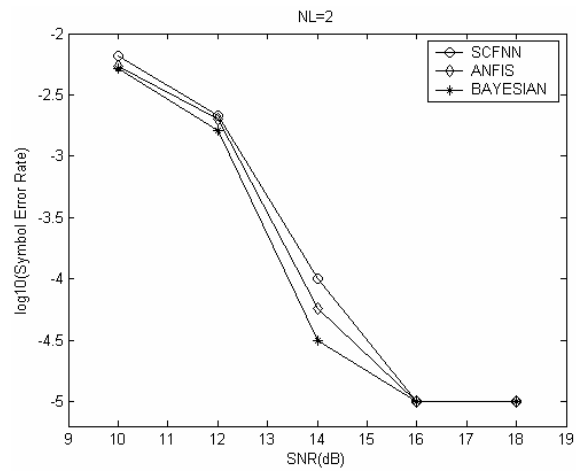
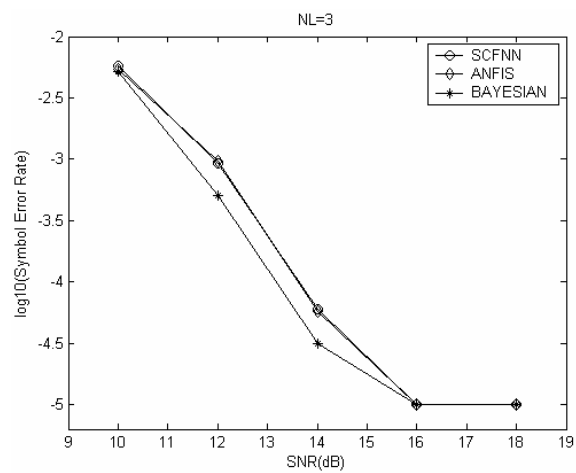
pendent and identically distributed discrete random processes, taking values that are equally likely to be $\{+1, -1\}$. Besides the intersymbol interference and nonlinear distortion introduced by the channel, the received signals are also assumed to be corrupted by additive white Gaussian noise (AWGN). In our design, we adopt a second-order equalizer with unit decision delay. When the circuit is operating in training mode, 1200 data samples are transmitted and used to build up the structure of the SCFNN-based equalizer at the receiver. When the training procedure is completed, the circuit is switched to testing mode. In our simulation, 100,000 data samples were tested to determine the error performance of the trained equalizers.

As a real-time operation requirement, the learning epoch was set 1 in the learning process. We set four membership functions for each of the variables; therefore, 16 rules would be created. The membership functions were bell-shaped curves. The initial choice of the threshold parameter μ_{\min} for SCFNN structure learning was empirically selected as 0.2, and its value was decreased as the learning process proceeds. A larger initial μ_{\min} would generally result in more fuzzy rules. Since a larger number of fuzzy rules usually implies a higher hardware implementation cost and a small number of rules results in a poor symbol error rate (BER), we determined a trade-off value for the initial μ_{\min} value during the simulation. Under the above conditions, the learning rate parameters in our circuit were chosen as $\eta_{\omega} = \eta_m = \eta_{\sigma} = 0.01$. Normally, these learning rate parameters are kept small in order to avoid possible oscillations of the learning processes. Another factor that affects the final number of rules is the initial deviation value of the Gaussian functions. In our simulation, we set this initial value to $\sigma_{new} = 0.75$. The numbers of rules generated under various simulation conditions by the time training finished are shown in Table 5. One can see that the numbers of rules required by the SCFNN based equalizers were kept satisfactorily low.

Table 5. Number of rules for training SCFNN.

SNR(dB) \ NL type	10	12	14	16	18
NL = 1	4	3	3	3	2
NL = 2	5	4	3	3	3
NL = 3	5	4	4	3	3

The symbol error rate (SER) performance of SCFNN compared with that of ANFIS and the Bayesian optimal solution is plotted in Figs. 3, 4, and 5. It can be seen that the capability of an SCFNN based equalizer to reconstruct the received destroyed symbols is close to that of ANFIS and the Bayesian optimal solution, even when the equalizer is operated under severe nonlinearity. The major advantage of SCFNN over the other methods is that it only takes one learning cycle to build a trained network during the training process. That is, SCFNN can be self-constructed in just 1 learning cycle. Thus, we can conclude that SCFNN is potentially more suitable for real-time symbol reconstruction.

Fig. 3. SNR vs. SER, $NL = 1$.Fig. 4. SNR vs. SER, $NL = 2$.Fig. 5. SNR vs. SER, $NL = 3$.

4.2 Complex Nonlinear Channel

To further evaluate the performance of SCFNN in the case of a highly complex nonlinear channel, we considered the following model [10, 11] with received signals:

$$\begin{aligned} \mu_n &= o_n + 0.2o_n^2 + 0.1o_n^3 + G, \\ o_n &= (1.0119 - j0.7589)s_n + (-0.3796 + j0.5059)s_{n-1}, \end{aligned} \tag{19}$$

where G is the Gaussian noise and o_n and s_n represent the channel output and the original 4-QAM modulated signals at time instant n , respectively. In this example, the channel order is 2, thus, there will be 16 different states for the channel output. The noise variance is assumed to be $\sigma_e^2 = 0.06324$; i.e., the signal-to-noise ratio SNR = 15 dB.

In the simulation, we used 2,000 data samples to train SCFNN. The initial threshold parameter μ_{\min} for structure learning was selected to be 0.05. The learning rate parameters were arbitrarily set to $\eta_\omega = \eta_m = \eta_\sigma = 0.01$. The deviation of a newly generated membership function was set to $\sigma_{new} = 0.5$. The numbers of rules required by the trained SCFNN network using the above parameters are shown in Table 6. In the same transmission environment, the ANFIS would need 16 rules in the structure, which is about twice as many as SCFNN requires.

Table 6. Number of rules for training SCFNN in complex channels.

SNR (dB)	5	10	15	20	25
Number of rules	9	9	8	7	6

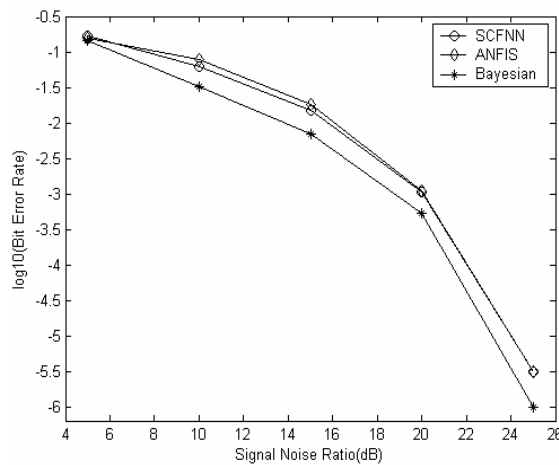


Fig. 6. Error curves for SCFNN, ANFIS and the Bayesian equalizer.

The error performance of the equalizers based on SCFNN, ANFIS and the Bayesian optimal solution is shown in Fig. 6. One can see that an SCFNN-based equalizer is only

about 1 dB worse than the optimal Bayesian equalizer when the SNR is high. But we need to note that in the Bayesian equalizer, there are 1,024 channel output states, which corresponds to a demand for 1,024 hidden neurons for the associated radial basis function (RBF) network. It is obvious that an SCFNN based equalizer is much more compact. In Fig. 7, we show a scatter diagram of the noisy channel output signals when SNR = 15 dB, where for the purpose of visualization, only 1,200 points are plotted. These signals are received at the receiver and are passed through the equalizer. In Figs. 8 (a) and (b), 100,000 points are depicted to exhibit the equalized output signal distributions from ANFIS and SCFNN equalizers, respectively. It can be seen that the SCFNN output distribution is more highly concentrated in the signal space than that of ANFIS is.

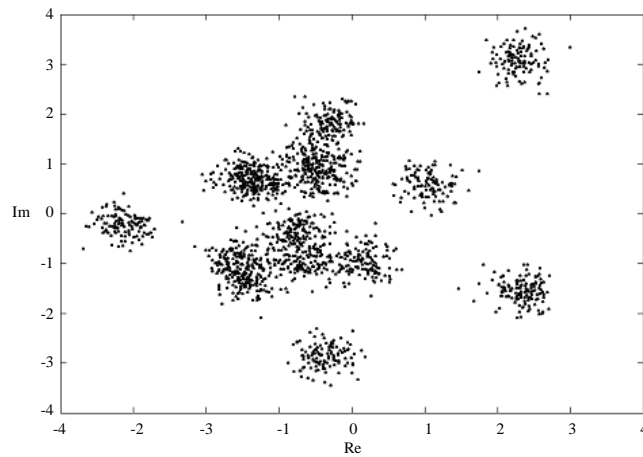
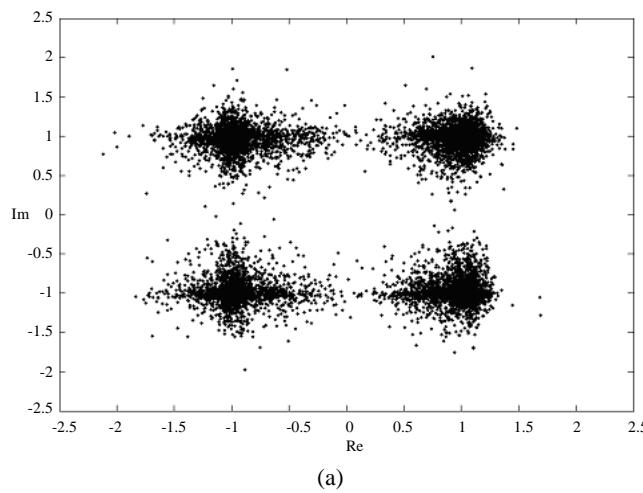


Fig. 7. Test channel symbol distribution.



(a)

Fig. 8. Equalizer output: (a) ANFIS, (b) SCFNN.

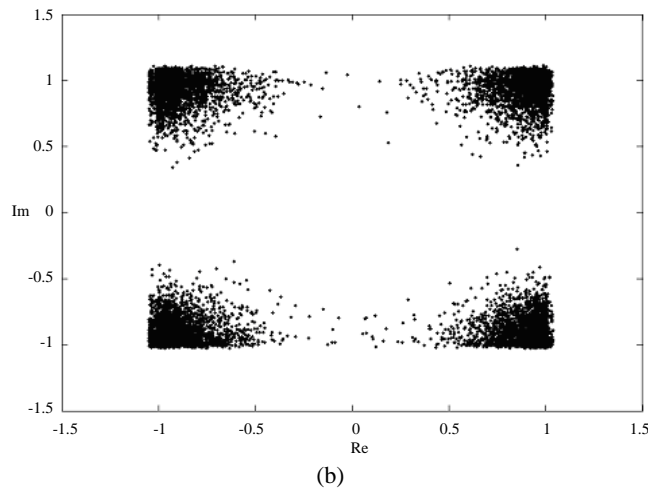


Fig. 8. (Cont'd) Equalizer output: (a) ANFIS, (b) SCFNN.

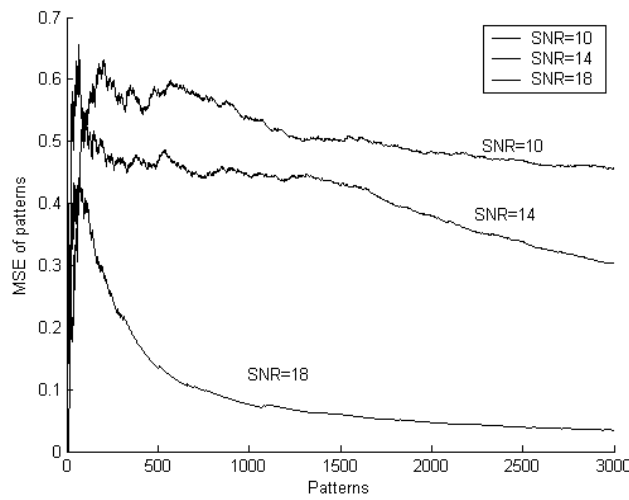


Fig. 9. Convergence curves for SCFNN in a learning cycle.

During the learning process for SCFNN, the parameters are adjusted every time a pattern has been dealt with. To achieve faster updating of the system parameters, we use only one cycle to train the circuit. Here, we define the mean-squared-error (MSE) between the equalizer input and output after the i -th pattern has been used for training as

$$MSE(i) = \frac{1}{i} \sum_{n=1}^i \sqrt{[y(n) - y^*(n)]^2}. \tag{20}$$

Fig. 9 depicts the convergence curves of SCFNN during one learning cycle. From the figure, one see that it takes only a few hundred patterns to achieve pretty good results. It

is worth noting that in a high SNR environment, SCFNN exhibits very a strong self-adjusting ability.

5. CONCLUSIONS

An SCFNN based nonlinear channel equalizer over real-valued and complex-valued nonlinear channels has been presented in this paper. The simulation results show that the SCFNN algorithm is capable of constructing a simple network whose performance is close to the optimal Bayesian solution. Due to the SCFNN's ability to self-adjust the location of the input space fuzzy partition, there is no need to estimate in advance the number and locations of the equalizer input states. Thus, we can set conditions for increasing the demand for fuzzy rules, which also makes the architecture of the constructed SCFNN fairly simple.

REFERENCES

1. J. G. Proakis, *Digital Communications*, 3rd ed., McGraw Hill, 1995.
2. J. C. Patra, R. N. Pal, R. Baliarsingh, and G. Panda, "Nonlinear channel equalization for QAM signal constellation using artificial neural network," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 29, 1999, pp. 262-271.
3. K. Chugg and A. Polydoros, "MLSE for unknown channels – part I: optimality considerations," *IEEE Transactions on Communications*, Vol. 44, 1996, pp. 836-846.
4. A. M. Al-Sanie and S. A. Alshebeili, "Blind channel estimation and data recovery in DS spread spectrum systems," *Signal Processing*, Vol. 81, 2001, pp. 1705-1714.
5. L. Wang and J. M. Mendel, "Fuzzy adaptive filters, with application to nonlinear channel equalization," *IEEE Transactions on Fuzzy Systems*, Vol. 1, 1993, pp. 161-170.
6. F. J. Lin, C. H. Lin, and P. H. Shen, "Self-constructing fuzzy neural network speed controller for permanent-magnet synchronous motor drive," *IEEE Transactions on Fuzzy System*, Vol. 9, 2001, pp. 751-759.
7. C. T. Yen, W. D. Weng, and Y. T. Lin, "FPGA realization of a neural-network-based nonlinear channel equalizer," *IEEE Transactions on Industrial Electronics*, Vol. 51, 2004, pp. 472-479.
8. J. J. Blake, L. P. Maguire, T. M. McGinnity, B. Roche, and L. J. McDAID, "The implementation of fuzzy systems, neural networks and fuzzy neural networks using FPGAs," *Information sciences*, Vol. 112, 1998, pp. 151-168.
9. D. Jianping, N. Sundararajan, and P. Saratchandran, "Communication channel equalization using complex-valued minimal radial basis function neural networks," *IEEE Transactions on Neural Networks*, Vol. 13, 2002, pp.687-696.
10. S. Chen, G. J. Gibson, C. F. N. Cowan, and P. M. Grant, "Reconstruction of binary signal using an adaptive radial-function equalizer," *Signal Processing*, Vol. 20, 1991, pp. 77-93.
11. I. Cha and S. A. Kassam, "Channel equalization using adaptive complex radial basis function networks," *IEEE Journal on Selected Areas in Communications*, Vol. 13,

1995, pp. 122-131.

12. J.-S. R. Jang, "ANFIS: adaptive network based fuzzy inference system," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 23, 1993, pp. 665-685.



Wan-De Weng (翁萬德) was born in Taipei, Taiwan, R.O.C., in 1963. He received the B.E. degree in Electrical Engineering from National Taiwan University, Taipei, Taiwan, R.O.C., the M.E. degree from the University of Utah, Salt Lake City, Utah, U.S.A., and the Ph.D. degree from Purdue University, West Lafayette, IN, U.S.A., in 1985, 1990, and 1993, respectively. In 1993, he joined the Department of Electrical Engineering, National Yunlin University of Science and Technology, Yunlin, Taiwan, R.O.C., as an Associate Professor. His research interests are signal processing in digital communication systems, VLSI design, and error control coding.



Rui-Chang Lin (林瑞昌) was born in Nantou, Taiwan, R.O.C., in 1955. He received the B.S. degree in Electrical Engineering from National Taiwan University of Science and Technology, Taipei, Taiwan, R.O.C., and the M.S. degree from the Taiwan Normal University, Taipei, in 1983, and 1990, respectively. Mr. Lin is currently a doctorate candidate in the Graduate School of Engineering Science and Technology, National Yunlin University of Science and Technology. In 1990, he joined the Department of Electronic Engineering, Nan kai Institute of Technology, Noutou, Taiwan, R.O.C., as an instructor. His research interests are neural network design and application in digital communication systems.



Chung-Ta Hsueh (薛仲達) was born in Taiwan R.O.C., in 1979. He received the B.S. degree from National Yunlin University of Science and Technology, Yunlin, Taiwan, R.O.C., in 2003. He is currently working toward the M.S. degree in the Graduate School of Electrical Engineering, National Yunlin University of Science and Technology, Yunlin, Taiwan, R.O.C. His current research focuses on applications of digital communication of artificial neural networks and their implementation on FPGA.