Nonlinear channel equalization for wireless communication systems using Legendre neural networks

Jagdish C. Patra [a,], Pramod K. Meher [b], Goutam Chakraborty [c]*

[a] *School of Computer Engineering, Nanyang Technological University, Singapore*

[b] *Department of Communication Systems, Institute for Infocomm Research, Singapore*

[c] *Department of Software and Information Science, Iwate Prefectural University, Japan*

**Corresponding Author. E-mail : aspatra@ntu.edu.sg (J.C. Patra), pkmeher@i2r.a-s-tar.edu.sg (P.K. Meher), goutam@soft.iwate-pu.ac.jp (G. Chakraborty)*

Abstract

In this paper, we present a computationally efficient neural network (NN) for equalization of nonlinear communication channels with 4-QAM signal constellation. The functional link NN (FLANN) for nonlinear channel equalization which we had proposed earlier, offers faster mean square error (MSE) convergence and better bit error rate (BER) performance compared to multilayer perceptron (MLP). Here, we propose a Legendre NN (LeNN) model whose performance is better than the FLANN due to simple polynomial expansion of the input in contrast to the trigonometric expansion in the latter. We have compared the performance of LeNN-, FLANN- and MLP-based equalizers using several performance criteria and shown that the performance of LeNN is superior to that of MLP-based equalizer, in terms of MSE convergence rate, BER and computational complexity, especially, in case of highly nonlinear channels. LeNN-based equalizer has similar performance as FLANN in terms of BER and convergence rate but it provides significant computational advantage over the FLANN since the evaluation of Legendre functions involves less computation compared to trigonometric functions.

Keywords

Nonlinear channel equalization, Functional link neural network, Legendre neural network

1. Introduction

Due to wide applications of internet technology and multimedia digital communication, development of efficient and high-speed data transmission methods over wireless channels has gained high importance in recent years. In order to use the precious transmission bandwidth more efficiently, the digital signals are usually transmitted over band-limited channels. However, due to the dispersive nature of these channels, and multipath problems, the transmitted data get invariably distorted in amplitude and phase, causing the inter-symbol interference (ISI) at the receiver end [1,2].

The performance of linear finite impulse response (FIR) or lattice equalizers trained with least mean square (LMS) or recursive least square (RLS) algorithms is not satisfactory for highly nonlinear and dispersive channels. It is shown that neural network (NN)-based channel equalizers perform better than the linear channel equalizers [3–5]. Because of the nonlinear signal processing in the NN, these networks are capable of performing the nonlinear mapping between the high-dimensional input and output spaces. Multilayer perceptron (MLP) is one of the earliest proposed NN for channel equalization problem. Due to the multilayer architecture, the MLPs are inherently computationally intensive. Although the MLP provides robust solution, its excessive training time and high computational complexity appear as two major drawbacks of this approach.

In order to reduce the training time of the NN and to improve its convergence rate, an alternate NN structure called functional link NN (FLANN) was proposed by Pao [6]. The learning and generalization characteristics of FLANN have been studied further by Pao et al. [7,8]. The FLANN is basically a flat or single layer network in which the original input pattern is enhanced by increasing the pattern dimension by using trigonometric expansion. Patra and Pal [9,10] had suggested the application of FLANN for channel equalization in digital communication and have shown that the FLANN-based equalizers perform better than the MLP-based structures for nonlinear channel models. With 2-PAM signals, FLANN-based equalizers outperform the MLP- and polynomial perceptron (PPN)-based equalizers, in terms of bit error rate (BER) and mean square error (MSE) convergence. With extensive simulation studies Patra et al. [11] have shown that FLANN-based equalizers perform better than MLPand PPN-based equalizers for quadrature amplitude modulated (QAM) signals. The studies on FLANN-based equalizer are further extended by other researchers [12,13]. The superiority of FLANN-based equalizer for nonlinear channels with reduced decision feedback is shown in [12]. A field programmable gate array (FPGA) implementation of FLANN equalizer has been reported in [14]. Using 2-PAM signals, Zhao and Zhang [15] have suggested an improved FLANN structure for channel equalization in which they have cascaded the FLANN with Chebyshev polynomials. The main benefit of using FLANN is its fast convergence and less computational complexity over MLP. A fuzzy FLANN for effective control of nonlinear systems has also been reported recently [13]. Some of the other NN structures, e.g., radial basis function (RBF) networks [16,17] and recurrent NNs [18,19], have been successfully applied for channel equalization. These studies indicate the popularity and effectiveness of the FLANN for solving nonlinear problems.

In this paper, we propose a novel Legendre NN (LeNN) for nonlinear channel equalization problem with 4-QAM signal constellation. The structure of LeNN is similar to that of FLANN. The main novelty of LeNN is that it enhances the input pattern by using the orthogonal Legendre polynomials instead of using the trigonometric functions as is done in case of FLANN. Recently, we have reported a computationally efficient LeNN for nonlinear channel equalization [20]. With preliminary results, we showed the superior performance of LeNN equalizer over the MLP- and RBF-based equalizers for different nonlinear channel models. In this paper, with extensive simulation studies and considering several channels and nonlinear models, we have shown that the performance of LeNN is superior to that of an MLP-based equalizer in terms of BER, rate of convergence and computational complexity. We have also shown here that the computational complexity of LeNN is lower than the FLANN since the evaluation of Legendre polynomials involves less computation than the evaluation of trigonometric functions. For comparison purpose, we have also carried out the simulation with an LMS–FIR adaptive filter-based equalizer.

The rest of the paper is organized as follows. In Section 2, we have presented a general overview of channel equalization problem. In Section 3, we have briefly described the MLP, FLANN and LeNN structures which we have used in this study. A short description of generalized NN-based channel equalizer is provided in Section 4. The simulation studies are given in Section 5. Performance analysis of LeNN-based equalizer and its comparison with MLP and FLANN structures are discussed in Section 6. Finally, the conclusions of this study are presented in Section 7.

2. Communication channel equalization

Fig. 1 shows the schematic of a wireless digital communication system with an equalizer at the front-end of the receiver. The symbol $\{t_k\}$ denotes a sequence of T-spaced complex symbols of an L-QAM constellation in which both in-phase component $\{t_{k,I}\}$ and quadrature component $\{t_{k,Q}\}$ take one of the values $\{\pm 1, \pm 2, \ldots, \pm(\sqrt{L}-1)\}$, where $1/T$ denotes the symbol rate and k denotes the discrete time index. In a 4-QAM constellation, the unmodulated information sequence $\{t_k\}$ is given by

$$t_k = \pm 1 \pm j1, \tag{1}$$

where the symbols $\{1, -1\}$ are assumed to be statistically independent and equi-probable. In Fig. 1, the combined effect of transmitter-side filter and wireless transmission medium are included in the "channel". A widely used model for a linear dispersive channel is an FIR filter whose output at the kth instant is given by

$$a_k = \sum_{i=0}^{N_h-1} h_i t_{k-i}, \tag{2}$$

where h_i denotes the FIR filter weights and N_h denotes the filter order. Considering the channel to be a nonlinear one, the "NL" block introduces channel nonlinearity to the filter output. The discrete output of the nonlinear channel is given by

$$b_k = \psi\{a_k, a_{k-1}, a_{k-2}, \ldots, a_{k-N_h+1}; h_0, h_1, \ldots, h_{N_h-1}\}, \tag{3}$$

where $\psi\{\cdot\}$ is a nonlinear function generated by the "NL" block. The channel output is assumed to be corrupted with an additive Gaussian noise q_k with a variance of σ^2. The transmitted signal t_k after being passed through the nonlinear channel and added with the additive noise arrives at the receiver, which is denoted by r_k. The received signal at the kth time instant is given by $r_k = r_{k,I} + jr_{k,Q}$, where $r_{k,I}$ and $r_{k,Q}$ are the in-phase and quadrature components, respectively.

The purpose of equalizer attached at the receiver front-end is to recover the transmitted sequence t_k or its delayed version $t_{k-\tau}$, where τ is the propagation delay associated with the physical channel. In case of a linear channel, an adaptive equalizer (e.g., an adaptive FIR filter) can be used. During training period, the equalizer takes the corrupted sequence r_k and its delayed versions as input and produces an output y_k. With the knowledge of a desired (or target) output d_k ($d_k = t_{k-\tau}$), it updates the filter weights so as to minimize the error e_k ($e_k = d_k - y_k$),

using an adaptive algorithm (e.g., LMS algorithm). After completion of training the weights are freezed, and subsequently these weights are used to estimate the transmitted sequence. In this study, since we consider nonlinear channel models, we used NNs as equalizer in place of adaptive filter.

3. NN structures for channel equalization

In this section we briefly describe the MLP, FLANN and LeNN models as prospective approaches for implementation of nonlinear channel equalizer.

3.1. The MLP

The MLP-based neural network (shown in Fig. 2), is a multi-layered architecture in which each layer consists of certain number of nodes or processing units. In general, it consists of an input layer, one or more hidden layer(s) and one output layer. With enough number of nodes in a single hidden layer, any arbitrary mapping between input and output spaces is possible. In general, the input layer is the lowermost layer and the output layer is the uppermost layer. Usually, only one hidden layer is employed in many practical applications. A bias unit with a fixed output value of +1 is added to all layers except the output layer. A 2-layer MLP is denoted as $\{n_0, n_1 n_2\}$, where, n_0, n_1 and n_2, denote the number of nodes (excluding the bias unit) in the input, hidden and output layers, respectively. The number of nodes in the input and the output layers are equal to the dimension of the input and output patterns, respectively.

In a fully connected MLP, each neuron of a lower layer is connected to all nodes of the upper layer through a set of connecting weights. Since no processing is done in the input layer, the output of its nodes is equal to the input pattern itself. In the nodes of other layers, processing is carried out as follows. The weighted sum of outputs of the lower layer nodes is computed and passed thereafter through a continuous nonlinear function (usually a $\tanh(\cdot)$ or *sigmoid* function). During training phase, an input pattern is applied to the MLP, and the node-outputs of all the layers are computed. The MLP outputs (outputs of the nodes of output layer) are compared with the desired (or target) outputs to generate error signals e_k to update the weights of the MLP using a learning algorithm, e.g., back propagation (BP) or Levenberg–Marquardt (LM) algorithm [21]. This updating of connection weights is carried out iteratively until the MSE reaches a predefined small value.

3.2. The FLANN

The structure of an FLANN is shown in Fig. 3. FLANN is a single-layer flat structure where the hidden layers are eliminated by transforming the input pattern to a higher dimensional space such that in the projected higher dimensional space the patterns become linearly separable. Due to the absence of hidden layer, FLANN provides computational advantage over the MLP. Unlike the MLP, it performs pattern enhancement by using a set of orthogonal functions of either an element or the entire pattern. It is shown that the FLANN is capable of representing nonlinear mapping between the input and output spaces [6]. Usually, in an FLANN, the functional expansion is carried out by using trigonometric functions [10,11,22]. As an example, a 2-dimensional input

pattern given by $X = [x_1, x_2]^T$ is expanded to higher-dimensional pattern by trigonometric functions as $X^e = [1, x_1, \cos(\pi x_1), \sin(\pi x_1), \cos(2\pi x_1), \sin(2\pi x_1), \ldots; x_2, \cos(\pi x_2), \sin(\pi x_2), \cos(2\pi x_2), \sin(2\pi x_2), \ldots]^T$. Thus, the enhanced input pattern is expressed as $X^e = \Phi(X)$, where $\Phi = [\phi_1(X), \phi_2(X), \ldots, \phi_N(X)]^T$. $\{\phi_i\}_{i=1}^N$ is a set of basis functions.

In the FLANN structure of Fig. 3, the input vector *X* is transformed into an output vector *Y* given by $Y = f_W(X)$. The nonlinear function $f_W(\cdot)$ represents a set of the orthogonal basis function Φ, implemented in the "functional expansion" block, and a set of weights *W*. Here the *n*-dimensional input pattern *X* is enhanced to an *M*-dimensional enhanced pattern X^e. The FLANN architecture is represented by {*N*, *M*}, where *M* is the dimension of FLANN output. The learning theory and mathematical analysis behind FLANN have been studied and reported in [6,7,11,23]. Using Stone–Weierstrass theorem, Chen et al. [13] have shown that FLANN could be used as an universal approximator. The input–output relationship of FLANN is explained below.

Let $\mathcal{B} = \{\Phi(A)\}_{i=\mathcal{I}}$, $\mathcal{I} = \{1, 2, \ldots\}$ be a set of basis functions, where *A* is compact simply connected subset of $\mathcal{R}^n$. $\mathcal{B}$ has the following properties: (i) $\phi_1 = 1$, (ii) the subset $\mathcal{B}_j = \{\phi_i \in \mathcal{B}\}_{i=1}^N$ is linearly independent and (iii) $\sup_j[\sum_{i=1}^j \| \phi_i \|_A^2]^{1/2} < \infty$.

Let us consider a set of basis functions $\mathcal{B} = \{\phi_i\}_{i=1}^N$ as shown in Fig. 3. The *j*th linear sum of the FLANN is given by

$$s_j = \sum_{i=1}^{N} w_{ji} \cdot \phi_i(X), \tag{4}$$

where $X \in A \subset \mathcal{R}^n$, $X = [x_1, x_2, \ldots, x_n]^T$ is the input vector and $W_j = [w_{j1}, w_{j2}, \ldots, w_{jN}]^T$ is the weight vector of FLANN's *j*th output node. This can be expressed in a matrix form as

$$s_j = W_j^T \cdot \Phi, \tag{5}$$

where $\Phi = [\phi_1(X), \phi_1(X), \ldots, \phi_N(X)]^T$ is the basis vector, which is the output of the functional expansion block. The *M*-dimensional linear output is given by $S = W^T\Phi$, where $S = [s_1, s_2, \ldots, s_M]^T$, and $W = [W_1, W_2, \ldots, W_M]$ is a $N \times M$ weight matrix. The *j*th output of FLANN is given by

$$y_j = \rho(s_j), \tag{6}$$

where $\rho(\cdot) = \tanh(\cdot)$. Therefore, the *M*-dimensional output vector of FLANN is given by

$$Y = [y_1, y_2, \ldots, y_M]^T = f_W(X). \tag{7}$$

3.3. Learning of FLANN

The learning process involves updating of the weights of FLANN in order to minimize a given cost function. Gradient descent algorithm is used for learning where the gradient of a cost function with respect to the weights is determined and the weights are incremented by a fraction of the negative gradient at each iteration. The well known back propagation algorithm is used to update the weights of FLANN. Consider an FLANN with a single output node. The goal of the learning algorithm is to minimize E_k, the cost function at kth instant, given by

$$E_k = \tfrac{1}{2}[d_k - y_k]^2 = \tfrac{1}{2}e_k^2, \tag{8}$$

where d_k is the desired output at the kth instant, and e_k denotes the error term. In each iteration, an input pattern is applied, the output of the FLANN is computed and the error $e_k\ (=\ d_k - y_k)$ is obtained. The error value is used in the backpropagation algorithm to minimize the cost function until it reaches a pre-defined minimum value. The weights are updated as follows:

$$W_{k+1} = W_k + \Delta W_k = W_k + \left(-\alpha \frac{\partial E_k}{\partial W_k}\right), \tag{9}$$

where α is the learning parameter which is set between 0 and 1. The gradient of the cost function (8) is given by

$$\frac{\partial E_k}{\partial W} = e_k \frac{\partial y_k}{\partial W}. \tag{10}$$

The update rule for the weight w_{ji} is given by

$$w_{ji,k+1} = w_{ji,k} + e_{j,k} \frac{\partial y_{j,k}}{\partial w_{ji}}. \tag{11}$$

If nonlinear tanh(·) function is used at the output node, the update rule becomes

$$w_{ji,k+1} = w_{ji,k} + \alpha e_{j,k}(1 - y_{j,k})^2 \phi_i(X). \tag{12}$$

To improve convergence, a momentum term is added to the update rule as follows:

$$w_{ji,k+1} = w_{ji,k} + \alpha\, \Delta w_{ji,k} + \beta\, \Delta w_{ji,k-1}, \tag{13}$$

where β is the momentum factor which is set between 0 and 1.

3.4. The Legendre NN

Structure of the Legendre NN (LeNN) (shown in Fig. 4) is similar to FLANN. In contrast to FLANN, in which trigonometric functions are used in the functional expansion,

LeNN uses Legendre orthogonal functions. The major advantage of LeNN over FLANN is that the evaluation of Legendre polynomials involves less computations compared to that of the trigonometric functions. Therefore, LeNN offers faster training compared to FLANN. Some of the important properties of Legendre polynomials are that (i) they are orthogonal polynomials, (ii) they arise in numerous problems especially in those involving spheres or spherical coordinates or exhibiting spherical symmetry and (iii) in spherical polar coordinates, the angular dependence is always best handled by spherical harmonics that are defined in terms of Legendre functions [24].

The Legendre polynomials are denoted by $L_n(X)$, where n is the order and $-1< x <1$ is the argument of the polynomial. They constitute a set of orthogonal polynomials as solutions to the differential equation [24]:

$$\frac{d}{dx}\left[(1-x^2)\frac{dy}{dx}\right] + n(n+1)y = 0. \tag{14}$$

The zero and the first order Legendre polynomials are, respectively, given by $L_0(x) = 1$ and $L_1(x) = x$. The higher order polynomials are given by

$$\begin{aligned} L_2(x) &= \tfrac{1}{2}(3x^2 - 1), \\ L_3(x) &= \tfrac{1}{2}(5x^3 - 3x), \\ L_4(x) &= \tfrac{1}{8}(35x^4 - 30x^2 + 3). \end{aligned} \tag{15}$$

The recursive formula to generate higher order Legendre polynomials is expressed as

$$L_{n+1}(x) = \frac{1}{n+1}[(2n+1)xL_n(x) - nL_{n-1}(x)]. \tag{16}$$

As an example, the 2-dimensional input pattern $X = [x_1, x_2]^T$ is enhanced to a 7-dimensional pattern by Legendre functional expansion $X^e = [1, L_1(x_1), L_2(x_1), L_3(x_1), L_1(x_2), L_2(x_2)L_3(x_2)]^T$. The training of LeNN is carried out in the same manner as FLANN.

3.5. Computational complexity

For most of the applications, LeNN requires a functional expansion up to the fourth order Legendre polynomial. Evaluation of the Legendre polynomials of order 2, 3 and 4 requires three, five and six multiplications, respectively. Therefore, on average, five multiplications are needed to evaluate a Legendre polynomial up to fourth order. Similarly, the average number of additions required to evaluate Legendre polynomial up to fourth order is equal to 1. Whereas, in case of FLANN, the evaluation of trigonometric functions require much higher number of multiplications and additions, as it involves complex power series expansion. Therefore, LeNN has computational advantage over FLANN.

Let us consider an L-layer MLP with n_l number of nodes (excluding the threshold unit) in layer l, for $l = 0, 1, \ldots, L$, where n_0 and n_L are the number of nodes in the input and output layer, respectively. An L-layer ANN architecture may be represented by $\{n_0 - n_1 - \cdots - n_{L-1} - n_L\}$. Three basic computations, i.e., addition, multiplication and computation of tanh( ) are involved for updating weights of the ANN. The computations in the network are due to the following requirements:

(1) forward calculations to find the activation value of all the nodes of the entire network;
(2) back-error propagation for the calculation of square error derivatives and
(3) updating the weights of entire network.

Total number of weights to be updated in one iteration in an MLP is given by $\sum_{l=0}^{L-1}(n_l + 1)n_{l+1}$, whereas in case of an FLANN it is only $(n_0 + 1)n_1$. Since there is no hidden layer in FLANN, computational requirement is drastically reduced compared to that of an MLP. A comparison of the computational requirements in one iteration of training using the BP algorithm, for the three types of NNs, is provided in Table 1. In this table, the number of multiplications and additions required to evaluate the trigonometric or Legendre polynomials are not considered, and therefore, FLANN and LeNN are put in the same column.

4. Channel equalization using a generalized NN model

Fig. 5 depicts a schematic diagram of channel equalization for 4-QAM signals using NN. The in-phase component $r_{k,I}$ and quadrature component $r_{k,Q}$ at kth instant are passed through a delay line to obtain the current and past signals. The current and the delayed signal samples constitute the input signal vector to the equalizer and is given by $R_k = [r_{k,I}, r_{k,Q}, r_{k-1,I}, r_{k-1,Q}, \ldots]^T = [r_1, r_2, r_3, \ldots]^T$. During training phase, at kth instant, R_k is applied to the NN, and the NN produces an output $Y_k = [y_{k,I}, y_{k,Q}]^T$. The NN output is compared thereafter with the desired output $D_k = [d_{k,I}, d_{k,Q}]^T$ to produce an error signal. The error signal is used in the BP algorithm to update the weights.

The training process continues iteratively until the MSE reaches a predefined small value. Thereafter, the NN weights are freezed. During the test phase and actual use, the NN weights obtained after training are used for equalization purpose. In this study, we use three different NN structures, i.e., MLP, FLANN and proposed LeNN, for equalization of nonlinear channels. In addition, for comparison purpose, we have simulated a linear FIR-based adaptive equalizer trained with LMS algorithm. In the case of 4-QAM signal constellation, the channel equalization becomes a 4-class classification problem. The NN structures, basically, creates nonlinear decision boundaries in the input space by generating a discriminant function to classify the received signal into one of the four categories.

5. Simulation studies

Simulations were carried out extensively for the channel equalization problem, with several practical channels and nonlinear models, using three different NN structures and an FIR adaptive filter. The channel impulse response used for this study is given by [2]

$$h(i) = \frac{1}{2}\left[1 + \cos\left(\frac{2\pi}{\Lambda}(i-2)\right)\right] \quad \text{for } i = 1, 2 \text{ and } 3$$
$$= 0 \quad \text{otherwise.} \qquad (17)$$

The input correlation matrix is given by $\mathscr{R} = \mathrm{E}[R_k R_k^T]$, where E is the expectation operator. The eigenvalue ratio (EVR) of a channel is defined as $\lambda_{max}/\lambda_{min}$, where $\lambda_{max}$and λ_{min} are the largest and smallest eigenvalues of $\mathscr{R}$. Higher the value of EVR, worse the channel in terms of channel spread and is more difficult to equalize. In (17), the parameter Λ determines the EVR of the channel, larger the value of Λ, higher the EVR. In order to study the channels under different EVR conditions, Λ was varied between 2.9 and 3.5 in steps of 0.2. Thus, the values of EVR becomes 6.08, 11.12, 21.71 and 46.82 with Λ values of 2.9, 3.1, 3.3 and 3.5, respectively [2]. The corresponding normalized channel impulse responses in z-transform domain are given by

$$CH = 1, \quad \Lambda = 2.9 : 0.209 + 0.995z^{-1} + 0.209z^{-2},$$
$$CH = 2, \quad \Lambda = 3.1 : 0.260 + 0.930z^{-1} + 0.260z^{-2},$$
$$CH = 3, \quad \Lambda = 3.3 : 0.304 + 0.903z^{-1} + 0.304z^{-2},$$
$$CH = 4, \quad \Lambda = 3.5 : 0.341 + 0.876z^{-1} + 0.341z^{-2}. \qquad (18)$$

The transmitted message is 4-QAM signal constellation of the form $\pm 1 \pm j1$. Each symbol was drawn from a uniform distribution. A zero mean white Gaussian noise was added to the channel output. The received signal was normalized to unity so that the signal to noise ratio (SNR) becomes equal to the reciprocal of noise variance. The current received symbol r_k and past three symbols were used as input in FIR–LMS- and MLP-based equalizer. Whereas, for FLANN- and LeNN-based equalizers, the current symbol r_k and past two symbols were used as input (see Fig. 5). Thus, the FIR-based adaptive equalizer has 16 weights. (It is observed that the increase in the number of weights does not improve the equalizer performance.)

A number of experiments were carried out to determine the optimum NN architecture, the learning rate α and momentum parameter β. A 2-layer MLP with {8-8-2} architecture, i.e., with the number nodes in the input, hidden and output layer as 8, 8 and 2 (excluding the bias unit), respectively, is selected. The 6-dimensional input has been expanded to an 18-dimensional enhanced pattern, by using trigonometric and Legendre polynomials, for FLANN and LeNN, respectively. Thus, both FLANN and LeNN have an architecture of {18−2}. Back-propagation algorithm was used to train the NNs. Details of architecture of the four equalizers and the chosen learning parameters are provided in Table 2. Nonlinear tanh(·) function was used in all the nodes (except the input nodes) of the NNs. The delay parameter was selected as 1. The details of functional expansion are given in Table 3 and in the Appendix. The three nonlinear models and a linear model used in this study (see "NL" block of Fig. 1) are given by [10]

$$
\begin{aligned}
&NL = 0 : b_k = a_k, \\
&NL = 1 : b_k = \tanh(a_k), \\
&NL = 2 : b_k = a_k + 0.2a_k^2 - 0.1a_k^3, \\
&NL = 3 : b_k = a_k + 0.2a_k^2 - 0.1a_k^3 + 0.5\cos(\pi a_k).
\end{aligned} \tag{19}
$$

The linear channel model is represented by $NL = 0$. A nonlinear channel model that may occur due to the saturation of transmitter amplifier is represented by $NL = 1$. The nonlinear models $NL = 2$ and 3 denote two arbitrary nonlinear channels [10]. The main reason for using the channel models (18) and nonlinear models (19) is that these models have been widely used by other researchers [2,14,15,22].

6. Performance analysis

Here we discuss the performance analysis of NN-based equalizers in terms of computational complexity, MSE convergence, BER and signal constellation diagrams.

6.1. Computational complexity

The training time (in ms) of the different equalizers are shown in Table 2. This is the time taken to train the NN for 3000 iterations using a Pentium 4 CPU with 2.80 GHz clock and 1 GB RAM. It can be seen from Table 2 that the FIR-based equalizer takes the minimum time (6.15 ms) to train, whereas the MLP-based equalizer takes maximum training time (23.55 ms). The training time of FLANN- and LeNN-based equalizers were found to be 19.2 and 12.9 ms, respectively. Even though the architectures of LeNN and FLANN are similar, it takes more time to evaluate the trigonometric functions (used in FLANN) than to evaluate Legendre functions (used in LeNN).

To verify the computational difference between FLANN and LeNN, we simulated only the functional expansion part, using trigonometric and Legendre functions separately. The functional expansion, in which the 6-dimensional input pattern was expanded to 18-dimensional enhanced pattern, was carried out for 2×10^8 times. The time taken to evaluate the trigonometric functions and Legendre expansions were found to be 317 and 260s, respectively. This shows the computational efficiency of LeNN over FLANN.

6.2. MSE performance

To study the convergence characteristics and MSE performance of the equalizers, each equalizer was trained with 3000 iterations. To smooth out the randomness of the NN simulation, the MSE was averaged over 500 independent runs. The MSE characteristics for $CH = 2$ with 15 dB additive noise is shown in Fig. 6. It may be noticed that the MSE characteristics of LeNN and FLANN almost overlap each other.

It is clear that the performance FIR–LMS-based linear equalizer is the worst among the four equalizers. Its MSE settles between -7 and -10dB for the four NL models. In addition, it provided the slowest convergence rate. The MLP-based equalizer performs much better than the FIR

equalizer. Its MSE settles between −15 and −20 dB for the four NL models. The performances of LeNN and FLANN are found to be similar. The MSE floor for both LeNN- and FLANN-based equalizers is about −23dB for all the four NL models. The MSE convergence rate is also the fastest for LeNN- and FLANN-based equalizers. The MSE floor settles at about 1500 iterations for LeNN and FLANN equalizers, whereas in case of MLP, it takes about 3000 iterations. Similar performances are observed also for other channels with other values of additive noise.

6.3. BER performance

All the four equalizer structures were trained for 3000 iterations, and their weights are freezed and stored. Thereafter, to calculate the bit error rate the stored weights are loaded into the NN and new test symbols are transmitted. Based on the new received samples, the equalizer estimates the transmitted symbol. If there is a mismatch between the transmitted symbol (delayed by 1) and the NN equalizer output, it gives a bit error. The BER was computed with 2×10^6 test symbols. This process was repeated for different values of additive noise ranging from 10 to 20 dB with step increment of 1 dB. The BER performance of $CH = 2$ for the four NL models are shown in Fig. 7.

As expected, the BER decreases as the SNR increases. The $NL = 3$ case is the most severe nonlinear model. It can be seen that the NN-based equalizers perform much batter compared to FIR–LMS-based equalizer. Among the three NN-based equalizers, performance of MLP-based equalizer is inferior to other two. Interestingly, the performances of LeNN- and FLANN-based equalizers are quite similar. In case of MLP-based equalizer, for $NL = 3$, when SNR rises from 15 to 20 dB, the $\log_{10}(BER)$ falls from −1.68 to −3.22. In the same situation, in case of FLANN- and LeNN-based equalizers the BER fall is from −1.92 to −3.54 and from −1.92 to −3.47, respectively (see Fig. 7(d)). In case of more severe nonlinearity in the channel $CH = 3$, under similar situation, the BER fall for MLP-, FLANN- and LeNN-based equalizers are from −1.03 to −1.51, from −1.21 to −1.76 and from −1.22 to −1.75, respectively (due to space constraint the figure is not provided).

In order to show the performance of the equalizers under different channel nonlinearities, we have plotted BER with varying EVR at SNR = 15 dB, in Fig. 8. We have stated that $CH = 1$, $CH = 2$, $CH = 3$ and $CH = 4$ correspond to EVR values of 6.08, 11.12, 21.71 and 46.82, respectively. Higher the EVR, more difficult is the channel to equalize. It is observed that as EVR increases, the BER also rises. However, the rise of BER is less severe in case of FLANN- and LeNN-based equalizers compared to MLP-based equalizer. It can be seen that in case of $NL = 3$, as the EVR increases from 6.08 ($CH = 1$) to 46.82 ($CH = 4$), the rise of $\log_{10}(BER)$ for the MLP-, FLANN- and LeNN-based equalizers are given by from −2.77 to −0.69, from −3.06 to −0.85 and from −3.04 to −0.87, respectively (see Fig. 8(d)). Thus, LeNN- and FLANN-based equalizers perform better than MLP-based equalizer, as EVR increases.

It can be seen from Figs. 6–8 that the learning and performance curves of the LeNN- and FLANN-based equalizers almost overlap without clear distinction between the two. This is due to the fact that the performance of the LeNN- and FLANN-based equalizers are found to be statistically similar, which is determined by computing the confidence interval as explained below.

As the performance of ANN algorithms depends on the random initialization of the network parameters, it is important to analyze the statistical behavior of the algorithm by repeating the experiments several times. We need to calculate 95% or 99% *confidence interval* of the results, to ascertain that it is narrow enough so that we can use the algorithm reliably.

To calculate the confidence interval of our algorithm and to compare the results with other competitive algorithms, we selected two cases of BER experiments, for SNR = 12 and 16 dB (please refer to Fig. 7(d)). Out of the four algorithms, we choose our proposed LeNN algorithm and the FLANN algorithm. The FLANN algorithm was chosen as its performance is closely similar to that of LeNN. We repeated the experiments 20 times, starting from random initialization of parameters followed by training of the parameters and finally finding the BER. Thus, we obtain two sets of BER results, for LeNN and FLANN, at SNR = 12 dB. Let us name them as Γ_{L12} and Γ_{F12}. Similarly, experiments were performed for SNR = 16 dB, and the corresponding two sets of results are named as Γ_{L16} and $\Gamma_{F16,}$ for LeNN and FLANN algorithms, respectively.

To ascertain normal distribution of data, we have used the *normal probability* plot method. Considering n data points, this is done by sorting of data in ascending order in each set, assigning the index i to each data element, calculating $f_i = (i - 0.375)/(n + 0.25)$ corresponding to each data i, and finally plotting data value x_i versus f_i. An approximate straight line for each data set ensures the normal distribution of data. The mean and standard deviation for four data sets were $\mu_{L12} = -1.3557$, $\sigma_{L12} = 0.0075$; $\mu_{F12} = -1.3500$, $\sigma_{F12} = 0.0069$; $\mu_{L16} = -2.1749$, $\sigma_{L16} = 0.0169$ and $\mu_{F16} = -2.1733$, $\sigma_{F16} = 0.0159$ for four sets of data Γ_{L12}, Γ_{F12}, Γ_{L16} and $\Gamma_{F16,}$ respectively. Corresponding 99% confidence intervals are computed and obtained as ± 0.0043, ± 0.0040, ± 0.0097, ± 0.0091, respectively. Very low values of confidence interval ensure very high confidence in our algorithm. Though confidence interval for SNR = 16 dB is slightly more than that for SNR = 12 dB, their absolute values are still extremely small. Moreover the intervals for LeNN and FLANN are almost same.

6.4. Signal constellation diagrams

The signal constellation diagram provide a visual representation of the performance of the equalizer. To obtain signal constellation diagram, we trained the equalizers with 3000 samples. Thereafter, we fed 1000 new samples to the equalizer to obtain the equalizer outputs. The equalizer output for these 1000 samples are plotted in Fig. 9 for $CH = 4$ with additive noise of 15 dB and $NL = 3$. It can be seen that the signal constellation of the FIR–LMS-based equalizer is the most clumsy. The signal constellation of the MLP-based equalizer is less clean than the FLANN- and LeNN-based equalizers. However, it can be seen that the signal constellation produced by the LeNN- and FLANN-based equalizers are similar.

To demonstrate superior performance of LeNN over MLP, the signal constellation diagrams for a high EVR channel ($CH = 4$) with $NL = 2$ and 3 nonlinear models are shown in Fig. 10. It can be seen that the LeNN-based equalizer provides a much clean signal constellation than the MLP-based equalizer for high EVR channel with severe nonlinearity.

Generally the signal constellation diagram provides a qualitative assessment of the equalizer performance. However, in order to have a quantitative assessment of the constellation diagram, we carried out the computations as follows. After plotting the 1000 equalizer outputs as shown in Fig. 9 or 10, we counted the number of data points whose both in-phase and quadrature component values are greater than ±0.25 (i.e., the number of data points lying near the four corners of the square). Larger number of data points concentrating in these regions implies better performance of the equalizer. In Fig. 9, the number of such data points found to be 839, 968, 993 and 991 for the FIR–LMS-, MLP-, FLANN- and LeNN-based equalizers, respectively. The number of such data points in Fig. 10(a), (b), (c) and (d) are found to be 947, 803, 981 and 927, respectively. Higher number of data points concentrated about the four corners of the square indicates that performance of LeNN- and FLANN-based equalizers is similar but LeNN-based equalizer performs much better than MLP-based equalizer, especially for high-EVR channels with severe nonlinearities.

7. Conclusions

In this paper we propose a novel computationally efficient neural network, named Legendre neural network (LeNN), for nonlinear channel equalization in digital communication systems with 4-QAM signal constellation. Since LeNN is a single layer NN, it has lower computational complexity and simpler implementation compared to MLP. We have carried out extensive simulations with several channels and nonlinear models and found the superiority of LeNN-based equalizer over MLP-based equalizer in terms of MSE convergence rate, BER and computational complexity. The proposed LeNN-based equalizer has similar performance as FLANN-based equalizer, which we had proposed earlier. Besides, as the computation of Legendre functions is less complex than trigonometric functions, LeNN takes less training-time than FLANN. Because of lower computational requirement and ability to perform complex mapping between multi-dimensional input and output spaces, the LeNN has potential applications in other areas of science and engineering.

Acknowledgments

The authors wish to offer their sincere thanks to the anonymous reviewers whose positive and constructive comments helped us to enhance the quality of this paper, to a great extent.

Appendix

Let $r_k = [r_{k,I}, r_{k,Q}]$, $r_{k-1} = [r_{k-1,I}, r_{k-1,Q}]$ and $r_{k-2} = [r_{k-2,I}, r_{k-2,Q}]$. The input to the NN equalizer is $\mathscr{R}_k = [r_k; r_{k-1}; r_{k-2}]^T = [t_1, t_2, \dots, t_6]^T$. Therefore, $t_1 = r_{k,I}$, $t_2 = r_{k,Q}$, $t_3 = r_{k-1,I}$, $t_4 = r_{k-1,Q}$, $t_5 = r_{k-2,I}$ and $t_6 = r_{k-2,Q}$. Using functional expansion block, the 6-dimensional input pattern $\mathscr{R}_k$ is enhanced to 18-dimensional pattern X_k, where $X_k = [x_1, x_2, \dots, x_{18}]^T$. The expanded inputs for FLANN and LeNN are shown in Table 3.

References

[1] S. Haykin, Communication Systems, fourth ed., Wiley, New York, 2001.

[2] S. Haykin, Adaptive Filter Theory, third ed., Prentice-Hall, Upper Saddle River, NJ, 1996.

[3] S. Chen, G.J. Gibson, C.F.N. Cowan, Adaptive channel equalization using polynomial perceptron structure, Proc. IEE Part I 137 (October 1990) 257–264.

[4] G.J. Gibson, S. Siu, C.F.N. Cowan, The application of nonlinear structures to the reconstruction of binary signals, IEEE Trans. Signal Process. 39 (August 1991) 877–1884.

[5] M. Meyer, G. Pfeiffer, Multilayer perceptron based decision feedback equalizers for channels with intersymbol interference, Proc. IEE Part I 140 (December 1993) 420–424.

[6] Y.-H. Pao, Adaptive Pattern Recognition and Neural Networks, Addison-Wesley, Reading, MA, 1989.

[7] Y.-H. Pao, Y. Takefuji, Functional-link net computing: theory, system architecture, and functionalities, Computer 25 (5) (May 1992) 76–79.

[8] Y.-H. Pao, G.H. Park, D.J. Sobajic, Learning and generalization characteristics of random vector functional-link net, Neurocomputing 6 (2) (1994) 163–180.

[9] J.C. Patra, R.N. Pal, Functional link neural network-based adaptive equalization of nonlinear channels with QAM signal, in: Proceedings of IEEE International Conference on Systems, Man, Cybernetics (SMC1995), Vancouver, BC, Canada, October 1995, pp. 2081–2086.

[10] J.C. Patra, R.N. Pal, A functional link artificial neural network for adaptive channel equalization, Signal Process. 43 (2) (May 1995) 181–195.

[11] J.C. Patra, R.N. Pal, R. Baliarsingh, G. Panda, Nonlinear channel equalization for QAM signal constellation using artificial neural networks, IEEE Trans. Systems Man Cybern. Part B Cybern. 29 (2) (April 1999) 262–271.

[12] W.D. Weng, C.T. Yen, Reduced-decision feedback flann nonlinear channel equaliser for digital communication systems, Proc. IEE Commun. 151 (4) (2004) 305–311.

[13] C.-H. Chen, C.-J. Lin, C.-T. Lin, A functional-link-based neurofuzzy network for nonlinear system control, IEEE Trans. Fuzzy Syst. 16 (5) (October 2008) 1362–1378.

[14] C.T. Yen, W.D. Weng, FPGA realization of a neural-network-based nonlinear channel equalizer, IEEE Trans. Ind. Electron. 51 (2) (April 2004) 472–479.

[15] H. Zhao, J. Zhang, Functional link neural network cascaded with Chebyshev orthogonal polynomial for nonlinear channel equalization, Signal Process. 88 (2008) 1946–1957.

[16] S. Chen, B. Mulgrew, P.M. Grant, A clustering technique for digital communication

channel equalization using radial basis function networks, IEEE Trans. Neural Networks 4 (July 1993) 570–579.

[17] D. Jianping, N. Sundararajan, P. Saratchandran, Communication channel equalization using complex-valued minimal radial basis function neural networks, IEEE Trans. Neural Networks 13 (3) (May 2002) 687–696.

[18] G. Kechriotis, E. Zervas, E.S. Manolakos, Using recurrent neural networks for adaptive communication channel equalization, IEEE Trans. Neural Networks 5 (2) (March 1994) 267–278.

[19] J. Choi, M. Bouchard, T.H. Yeap, Decision feedback recurrent neural equalization with fast convergence rate, IEEE Trans. Neural Networks 16 (3) (May 2005) 699–708.

[20] J.C. Patra, W.C. Chin, P.K. Meher, G. Chakraborty, Legendre-FLANNbased nonlinear channel equalization in wireless communication systems, in: Proceedings of IEEE International Conference on Systems, Man, Cybernetics (SMC2008), Singapore, October 2008, pp. 1826–1831.

[21] S. Haykin, Neural Networks, second ed., Prentice-Hall, Upper Saddle River, NJ, 1999.

[22] W.D. Weng, C.S. Yang, R.C. Lin, A channel equalizer using reduced decision feedback Chebyshev functional link artificial neural networks, Inf. Sci. 177 (2007) 2642–2654.

[23] N. Sadegh, A perceptron network for functional identification and control of nonlinear systems, IEEE Trans. Neural Networks 4 (6) (November 1993) 982–988.

[24] R.E. Attar, Special Functions and Orthogonal Polynomials, Lulu Press, Morrisvelle, NC, 2006.

List of Tables

List of Figures

Number of	MLP	FLANN/LeNN
Weights	$\mathscr{S}_1$	$n_1\mathscr{N}_0$
Additions	$3\mathscr{S}_2 + 3n_L - n_0n_1$	$2n_1\mathscr{N}_0 + n_1$
Multiplications	$4\mathscr{S}_2 + 3\mathscr{S}_3 - n_0n_1 + 2n_L$	$3n_1\mathscr{N}_0 + n_0$
$\tanh(\cdot)$	$\mathscr{S}_3$	n_1
$\mathscr{S}_1 = \sum_{l=0}^{L-1}(n_l + 1)n_{l+1}, \quad \mathscr{N}_0 = n_0 + 1$		
$\mathscr{S}_2 = \sum_{l=0}^{L-1} n_l n_{l+1}, \quad \mathscr{S}_3 = \sum_{l=1}^{L} n_l$		

Table 1

Parameters	LMS	MLP	FLANN	LeNN
Architecture	{8−2}	{8−8−2}	{18−2}	{18−2}
No. of weights	16	90	38	38
Input symbols	$\{r_k, r_{k-1}, r_{k-2}, r_{k-3}\}$	$\{r_k, r_{k-1}, r_{k-2}, r_{k-3}\}$	$\{r_k, r_{k-1}, r_{k-2}\}$	$\{r_k, r_{k-1}, r_{k-2}\}$
Learning/momentum	0.01/0.01	0.05/0.05	0.7/0.5	0.7/0.5
Iterations	3000	3000	3000	3000
Training time (ms)	6.15	23.55	19.2	12.9

Table 2

x	FLANN	LeNN
x_1	t_1	t_1
x_2	$\cos(\pi t_1)$	$L_2(t_1)$
x_3	$\sin(\pi t_1)$	$L_3(t_1)$
x_4	t_2	t_2
x_5	$\cos(\pi t_2)$	$L_2(t_2)$
x_6	$\sin(\pi t_2)$	$L_3(t_2)$
x_7	t_3	t_3
x_8	$\cos(\pi t_3)$	$L_2(t_3)$
x_9	$\sin(\pi t_3)$	$L_3(t_3)$
x_{10}	t_4	t_4
x_{11}	$\cos(\pi t_4)$	$L_2(t_4)$
x_{12}	$\sin(\pi t_4)$	$L_3(t_4)$
x_{13}	t_5	t_5
x_{14}	$\cos(\pi t_5)$	$L_2(t_5)$
x_{15}	$\sin(\pi t_5)$	$L_3(t_5)$
x_{16}	t_6	t_6
x_{17}	$\cos(\pi t_6)$	$L_2(t_6)$
x_{18}	$\sin(\pi t_6)$	$L_3(t_6)$

Table 3

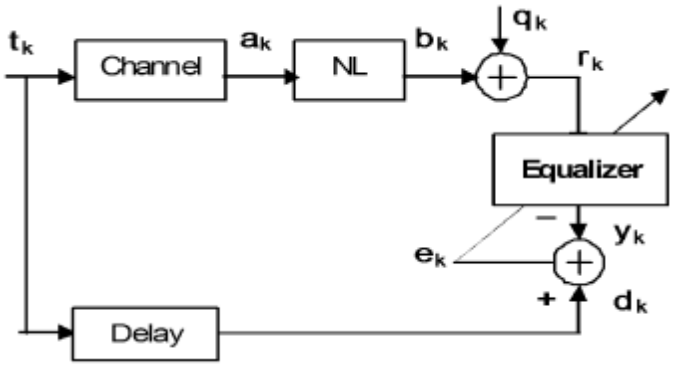

Fig. 1

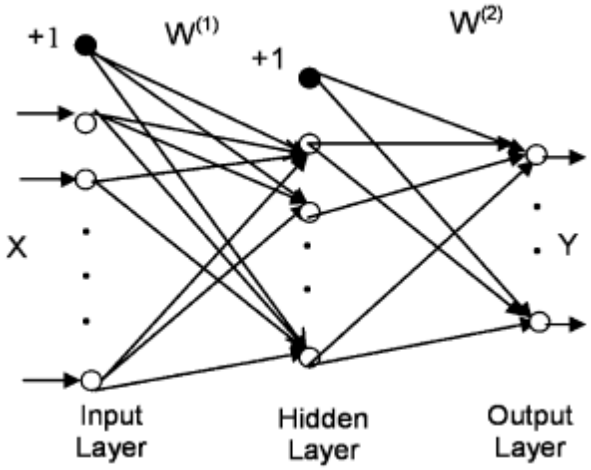

Fig. 2

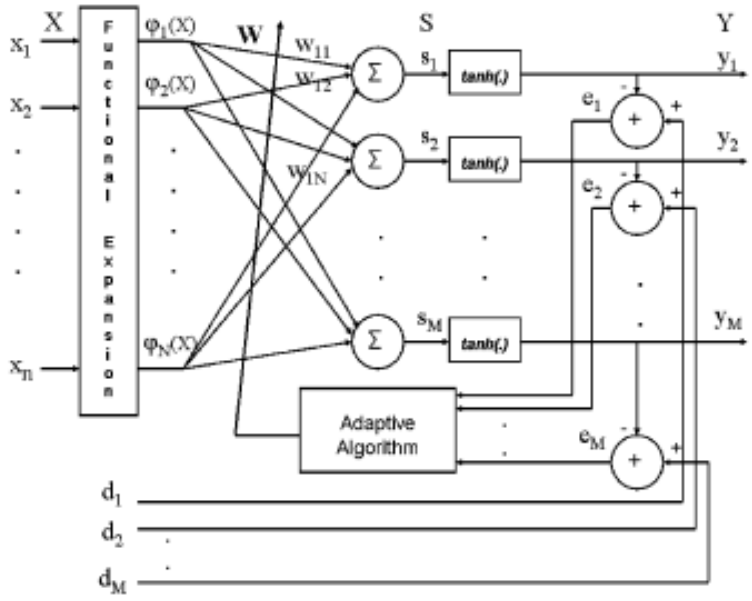

Fig. 3

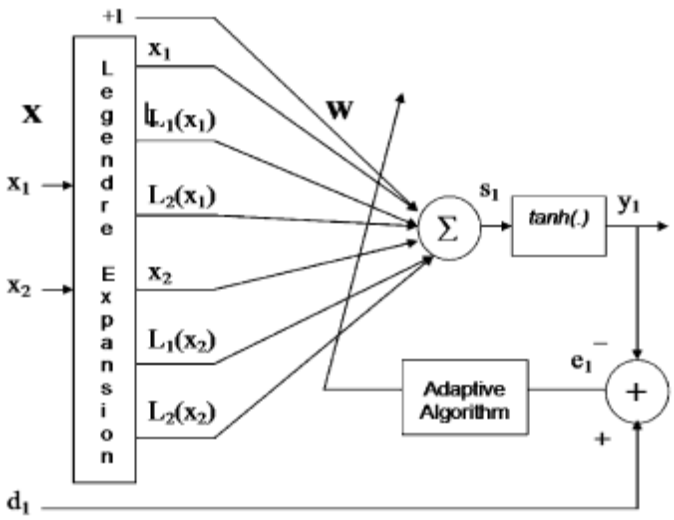
+1
X
x1
x2
Legendre Expansion
x1
L1(x1)
L2(x1)
x2
L1(x2)
L2(x2)
W
Σ
s1
tanh(.)
y1
e1
−
+
+
Adaptive Algorithm
d1

Fig. 4

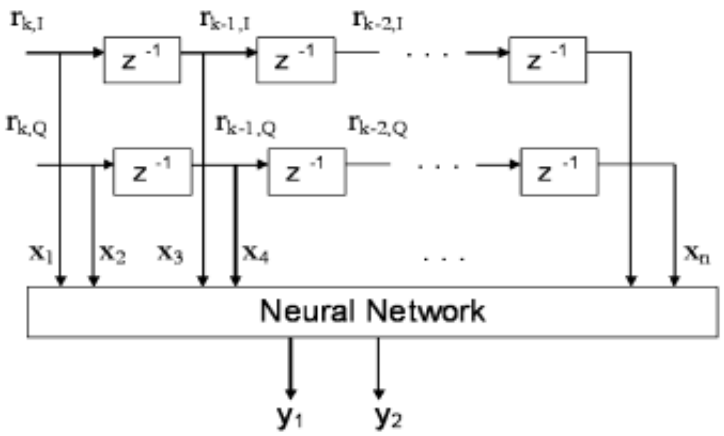
r k,I
r k-1,I
r k-2,I
z -1
z -1
z -1
r k,Q
r k-1,Q
r k-2,Q
z -1
z -1
z -1
x1
x2
x3
x4
xn
Neural Network
y1
y2

Fig. 5

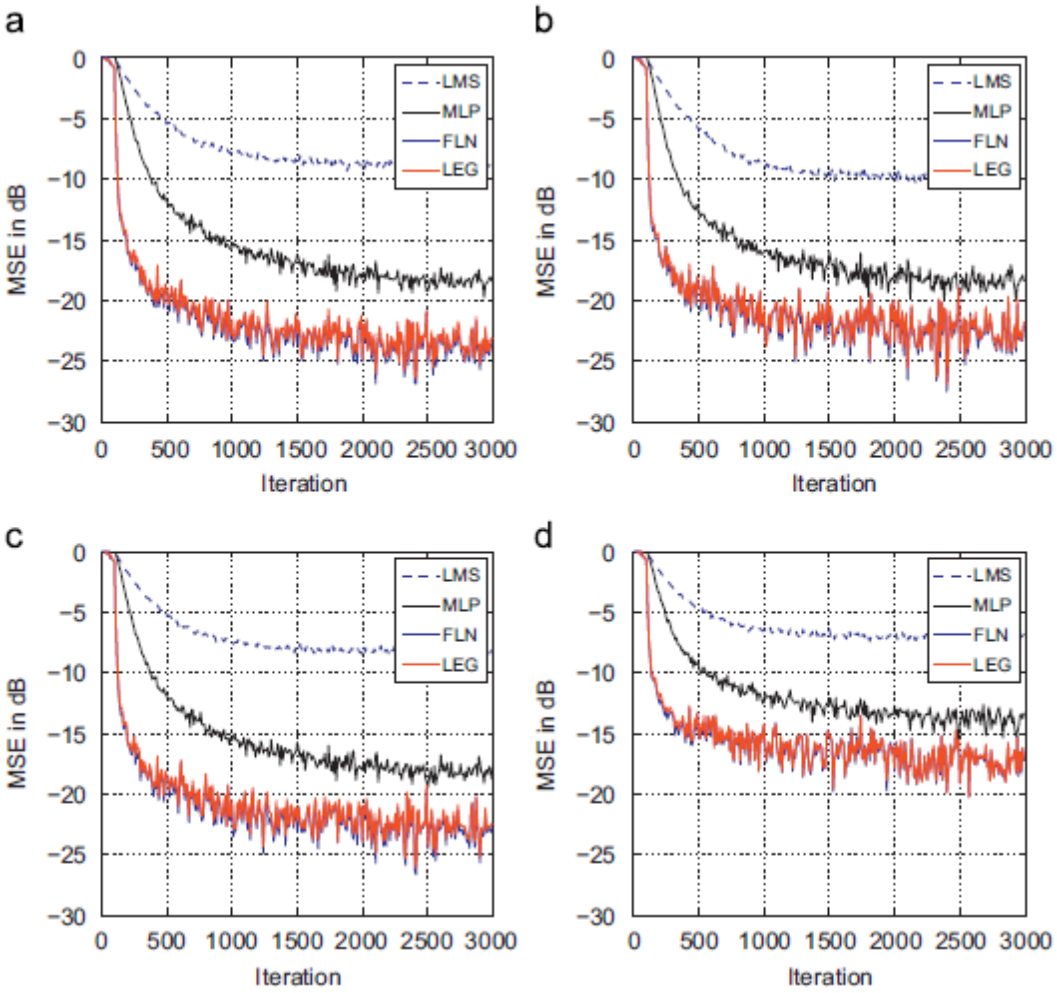
a
b
c
d
MSE in dB
Iteration
LMS
MLP
FLN
LEG
0
−5
−10
−15
−20
−25
−30
0
500
1000
1500
2000
2500
3000

Fig. 6

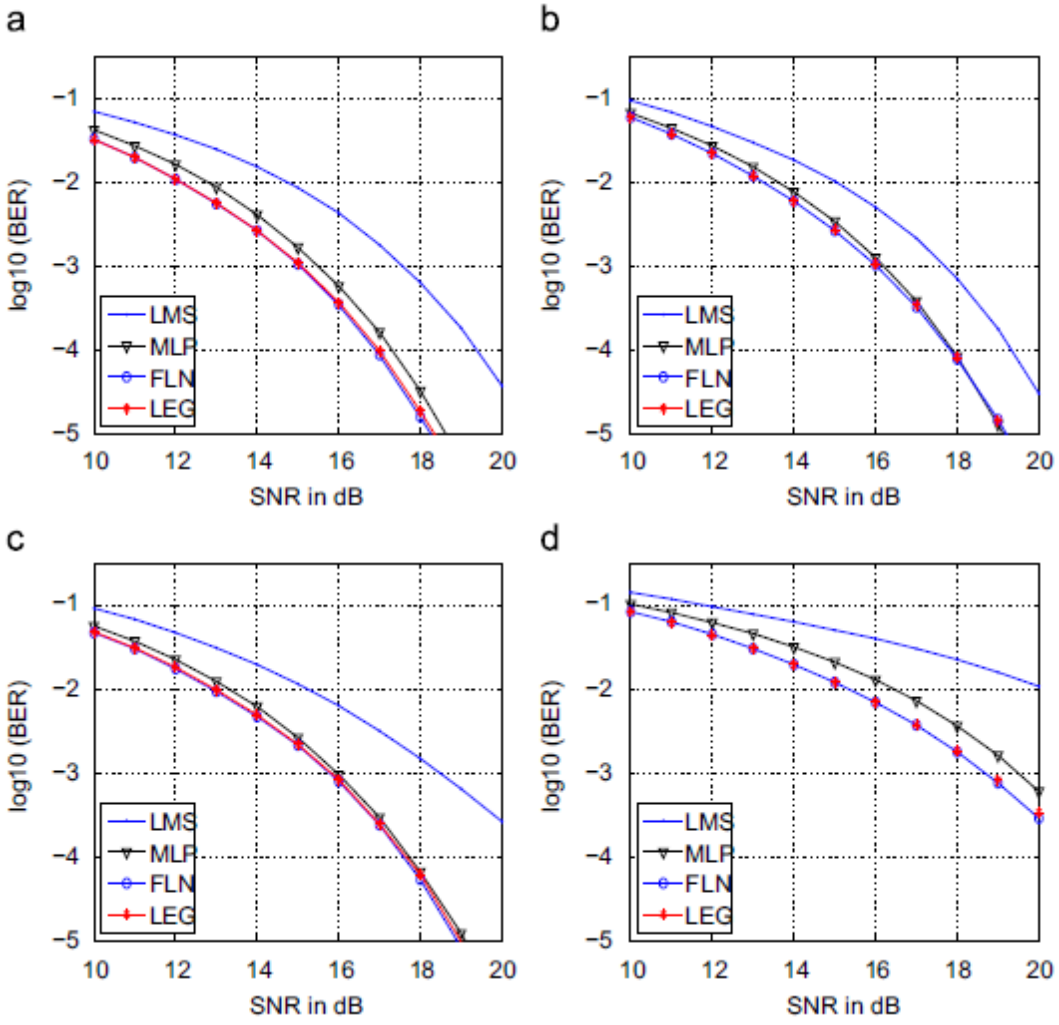

Fig. 7

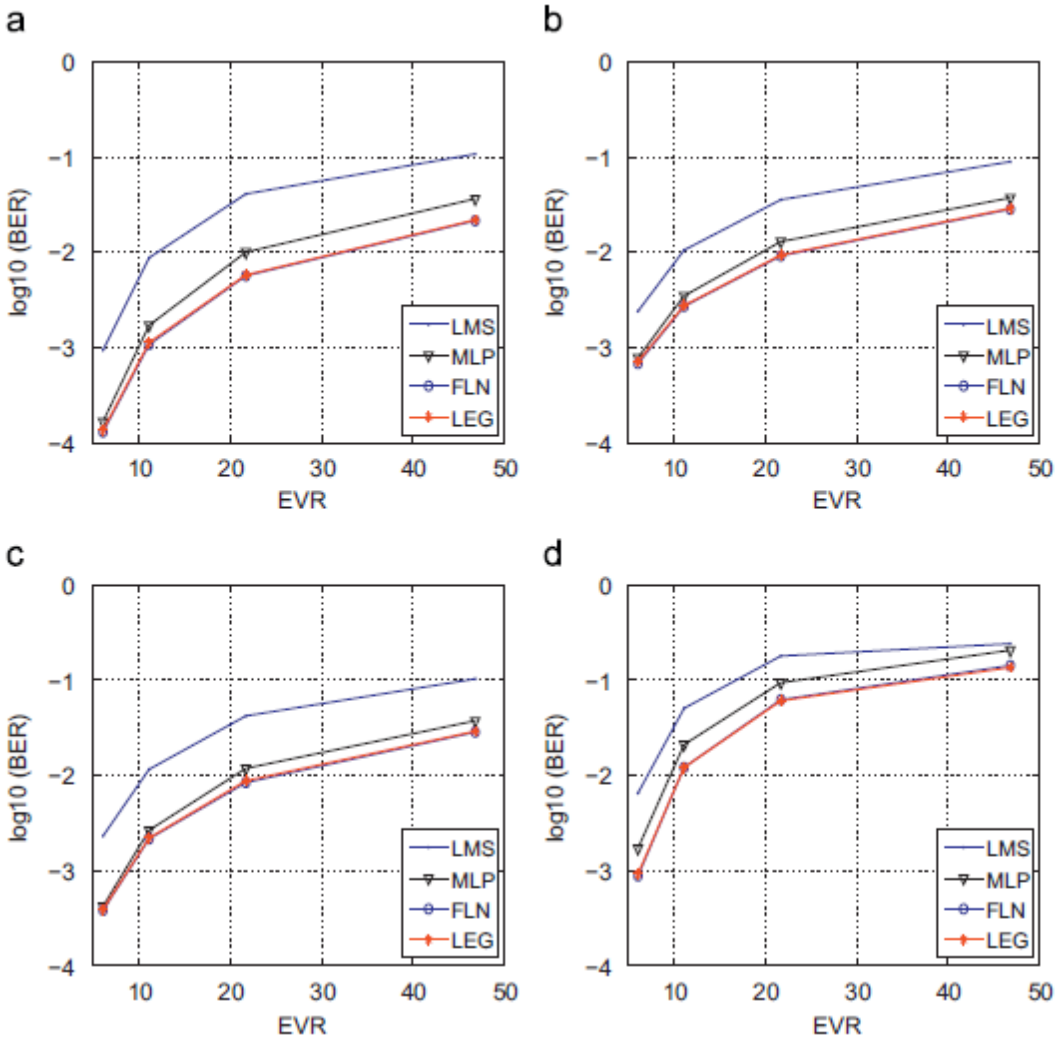

Fig. 8

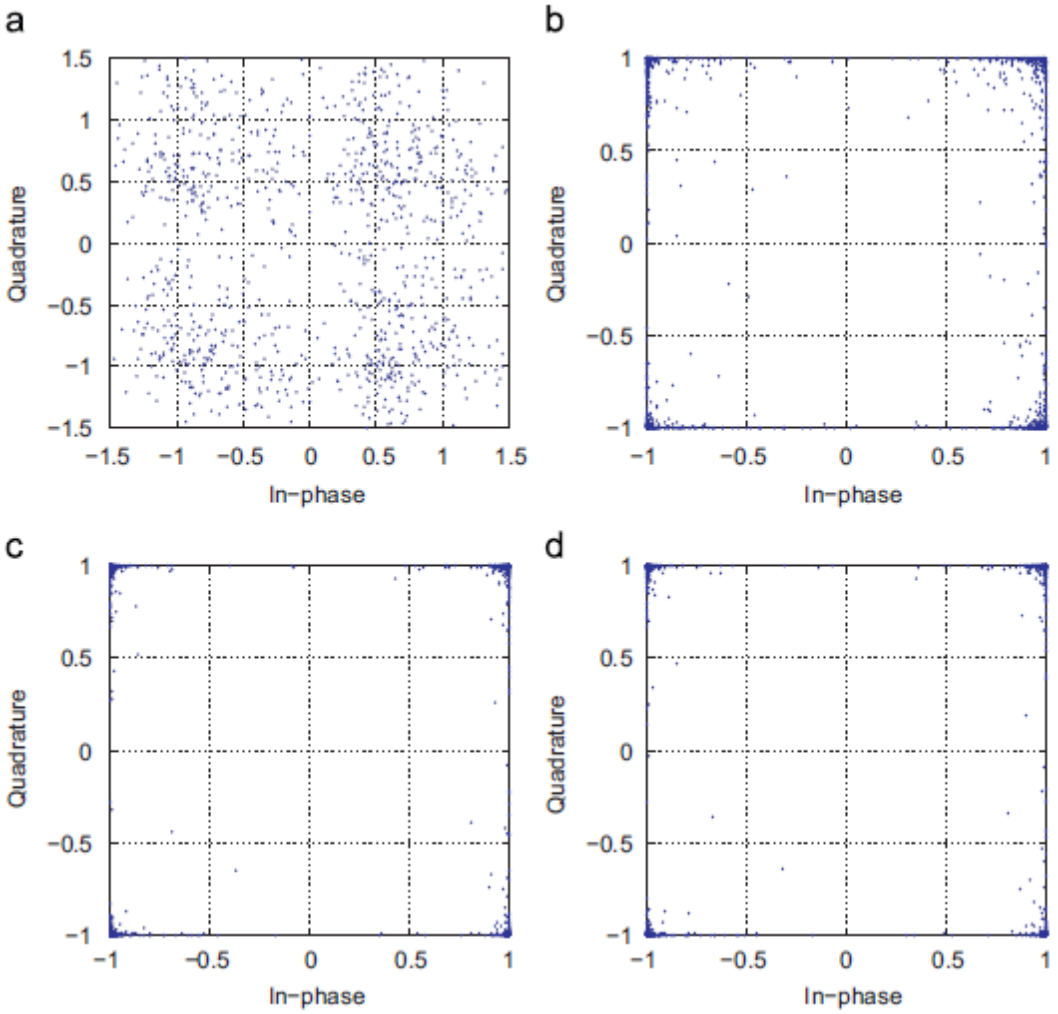

Fig. 9

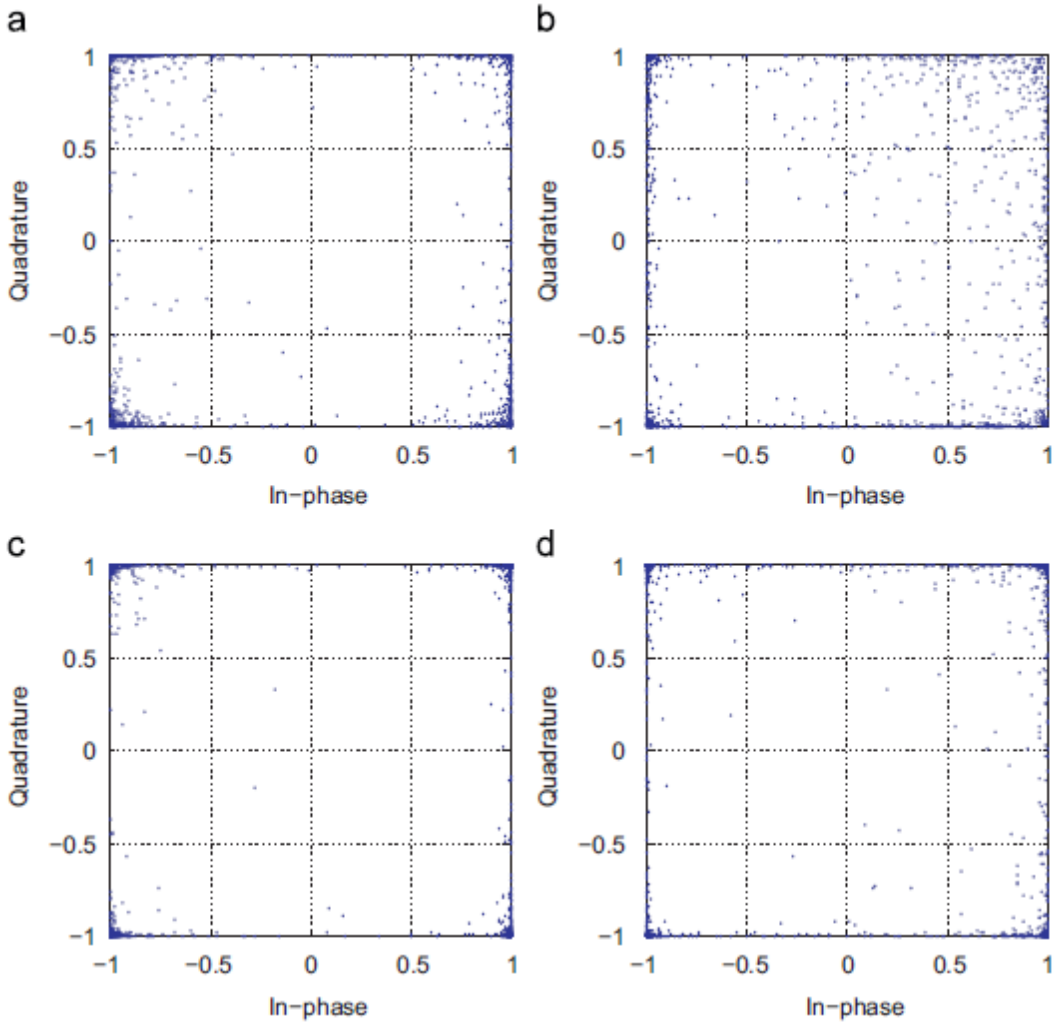

Fig. 10