CHAPTER 1

REVIEW ON CHANNEL EQUALIZATION

1.1 INTRODUCTION

Communication systems comprise three fundamental elements: transmitter, channel and receiver. When signals are transmitted through a communications system, they are obstructed by some distortions which are mainly intersymbol interference (ISI) and noise. The transmitted signal is distorted by ISI which is caused by multipath effect in band limited (frequency selective) time dispersive channels and is the cause of bit errors on the receiver side. ISI is considered the main factor negatively affecting fast transmission of data over wireless channels. In order to eliminate or minimize these distortions, equalizers are employed in these systems. Equalization is the method of compensating for, eliminating or reducing the amplitude and phase distortion introduced by the transmission medium in communications systems. In a general meaning, the term *equalization* refers to any signal processing operation which minimizes ISI. An equalizing filter overcomes the ISI caused by individual received symbols of a transmitted data stream, as well as the crosstalk that for example occurs due to coupling of a transmitted pulse or that results from the capacitive coupling of the transmitted pulse on an outgoing pair interfering with the received pulse on an incoming pair. The task of equalizers is to provide efficient and error free communications by ensuring that signals transmitted through the channel are recovered as original at the end of the receiver that communications system has.

Distortions may be linear or nonlinear depending on the channel characteristics of channel. When transmitting information through a physical channel, various mechanisms distort the transmitted signal significantly, causing degradation or even failure in the communications. These mechanisms can be classified as additive thermal noise, man-made noise and atmospheric noise. In practice, many of the physical channels are characterized by various channel models. The most frequently encountered channel of communications is that with additive noise. An additive random noise process is involved in this channel model. The factors causing the additive noise process are amplifiers and electronic components on the

receiver side of the communications system the transmission's interference as radio signal transmission, for example. Thermal noise is the category of noise that electronic components and amplifiers cause. Statistically, that sort of noise gets classified as a random *Gaussian noise process* and modeling the channel in terms of mathematics is named the *additive Gaussian noise channel*. The mathematical model becomes an *additive white Gaussian noise* (AWGN) channel in the case of the random process being a *white-noise process.* The random process is a white-noise process when the power spectral density (PSD) is flat (constant) over all frequencies [1,2].

When compared with AWGN channels, mobile radio channel deficiencies make the signal on the receiver side greatly distorted or cause its significant fading. This fading is classified as a non-additive signal disturbance and appears as time variation in the signal amplitude. Some techniques are utilized to compensate for fading channel deficiencies. The main techniques used in compensating for fading channel impairments can be classified as equalization, channel coding and diversity that are employed to compensate for the signal distortions and improve the received signal quality [3]. This thesis concentrates on equalization technique.

Equalization techniques can be categorized into linear or nonlinear techniques depending on the way the output of an adaptive equalizer is used for subsequent control of the equalizer. The decision making device of the receiver processes the equalizer's output and determines the value of the digital data bit being received before applying a slicing or thresholding operation (a nonlinear operation) to determine the value of the reconstructed message data. If this data is not used in the feedback path for the adapting of the equalizer, it's a *linear* type of equalization, but on the other hand, if the decision making device feeds the reconstructed data back in order to alter the equalizer's subsequent outputs, the equalization is *nonlinear* [3]. If the used channels are nonlinear, linear equalizers cannot reconstruct the transmitted signal. There are various equalizer structures among which *linear transversal equalizer* (LTE) is the most common. The simplest LTE, whose transfer function is a polynomial, uses only feed forward taps and has many zeros but poles only at $z = 0$. This filter is called a *finite impulse response* (FIR) filter or simply a transversal filter. In this type of equalizer, the filter

coefficient linearly weights the received signal's current and past values before summing them to produce the output of the equalizer.

Besides, some applications employ nonlinear equalizers since linear equalizers cannot deal with high amount of channel distortion. The performance of linear equalizers on channels involving deep spectral nulls in the passband is not good and hence, linear equalizers enhance the noise present in the frequencies in which they place too much gain in attempting to compensate for the distortion. Nonlinear equalizers are superior in performance to linear equalizers because of these reasons. Three quite effective nonlinear methods which possess improvements over linear equalization methods and that are used in 2G and 3G systems are:

1. Decision Feedback Equalization (DFE)
2. Maximum Likelihood Symbol Detection (MLSD)
3. Maximum Likelihood Sequence Estimation (MLSE) [4]

There have been large amount of studies aimed at channel equalization using various methods, techniques and algorithms. Recently, neural network based fuzzy technology has been widely used as a powerful and significant tool in channel equalization of various types of signals. Experts have determined the fuzzy rules by utilizing the channel's input-output data pairs in this type of equalizers. Adaptive channel equalization based on neural networks and employing multilayer perceptron (MLP) has been developed as part of this thesis which has enabled the equalization of Quadrature Amplitude Modulation (QAM) type signals of various levels. This has been achieved for both linear and nonlinear channels using a Nonlinear Neuro-Fuzzy Equalizer (NNFE) at a relatively high adaptation speed and accurate equalizer output results which has proven to be quite effective and practical.

The changeable fuzzy IF-THEN rules which configure the fuzzy adaptive filter are formed by either human experts or the input-output pairs that are matched throughout a procedure of adaptation. In this study, neural networks and fuzzy technology are used for the development of a neuro-fuzzy equalizer for channel distortion of Quadrature Amplitude Modulation (QAM) signals. Even though the QAM signal has a complex form which is composed of real (in-phase) and imaginary (quadrature) parts, the complex signal is not directly applied to the

channel and equalizer since the used neuro-fuzzy filter is based on real values and best suits the signal processing that takes place in real multidimensional space. The modulation and demodulation of *M*-ary QAM (where *M*=4 & *M*=16) is accomplished by splitting the stream of data bit into the in-phase (I) and quadrature (Q) components. Gray coding is employed to map the I and Q components together. The significant feature of this thesis study is the application of 'normalization' method by which the modulated in-phase and quadrature QAM signal is normalized to a maximum of one. Consequently, each component of the complex signal attains values between 0 and 1 by first shifting the values such that the minimum value is zero and then scaling them such that the maximum value is 1. Each component then is input to the channel and equalizer separately and denormalized separately at the equalizer's output where they are recombined to form the final desired complex QAM scheme at the end. The normalization method provides better BER and convergence performance since it is stable in addition to more accurate equalizer output results with relatively small number of iterations before the minimum error is attained.

This thesis consists of five chapters where:

Chapter 1 presents an overview on channel equalization. The state of application of neuro-fuzzy system and fuzzy logic as well as their properties and features are explained.

Chapter 2 explains the channel equalization, the distortions and noise in the channel. Mathematical models and formulas representing the channels and nonlinear neuro-fuzzy equalizer used in the thesis together with its characteristics are described.

Chapter 3 outlines the architecture and operation principles of the nonlinear neuro-fuzzy network (NNFN). The used learning algorithm, the linguistic data about the target system and numerical input-output relationships of NNFN are explained in detail. Fuzzy rule-based fuzzy sets, the parameters and error calculations are analyzed.

Chapter 4 describes in detail the quadrature amplitude modulation (QAM) and its properties. The application of QAM on NNFN and the features of the thesis design are explained. The specific technique of normalization used in equalizing QAM signals and its mathematical implementation are described.

Chapter 5 illustrates the simulation results of the equalization system demonstrating graphically and statistically the performance of the equalization system. Bit error rate (BER) versus signal-to-noise ratio (SNR) analysis is made in tabulated and graphical forms proving the accuracy of the system. Comparisons between the channels and between the two constellations of QAM are made to illustrate the performance of the equalizer, as well. Conclusions are discussed at the end.

1.2 Overview

In order to accurately transmit the input signals from the transmitter to the receiver, minimization and thus equalization of distortions in the channel is critical. This can be successfully done by employing efficient equalization algorithms and techniques during the transmission of the signals from the transmitter to the receiver. This chapter considers methods used in channel equalization. Neural networks, fuzzy and neuro-fuzzy technologies which form the basis of the adaptive channel equalization are analyzed and discussed.

1.3 The State of Application of Channel Equalization

Linear and nonlinear distortions are the main obstacles in transmitting the input signals to the receiver of a communications system in their original state. These distortions, namely ISI and noise, are caused in the channel and channel equalization is needed in order to transmit the signals as accurately as possible. Even though both linear and nonlinear equalizers can be used for this purpose, nonlinear equalizers are more preferably used because they are capable of compensating both linear and nonlinear channel distortions effectively.

Two types of equalization are used which are sequence estimation and symbol detection. In this thesis, symbol detection technique is used to realize the adaptive channel equalization. This technique maps the input baseband signal of the input on top of a feature space that the representation of a learnt property of the transmitted signal determined. The symbols are separated by the usage of decision regions which function to classify the distorted signal.

The ISI problem which affects all digital communication systems is mainly caused by restricted bandwidth. The restricted bandwidth is caused by rectangular multilevel pulses

when they are filtered improperly as they pass through a communication system spreading in time, being smeared into adjacent time slots, causing ISI [2]. This ISI in turn causes errors when transmitting data over the channel. Additionally, channel characteristics have a significant role in causing distortions and the response of channel is time-variant meaning that channel characteristics are not known in advance. The time-variant channel response and the unknown channel characteristics obligates the equalizers to be designed to adjust themselves to the channel response and to adapt themselves to the variations of time in the response of channel so as to compensate for the channel characteristics' variations. Such equalizers are called *adaptive equalizers* and they have been receiving great attention because of their superior features. In practice, as an example, there are situations when the channel consists of dial-up telephone lines and the channel transfer function changes from call to call. In such a case, the equalizer should be an *adaptive filter*.

Adaptive equalizers are categorized as *supervised* and *unsupervised* equalizers. When it is necessary to use a training sequence because of the unpredictable channel characteristics in a communications system, supervised equalizers are employed. This is done in order for the channel response to be compared with the input to be able to update the parameters of the equalizer. On the other hand, some communications systems do not allow the use of training signals because the methods used to accomplish the equalization of channel do not allow the training sequence to be transmitted. This is when *unsupervised* equalization is employed. This equalization that involves a *self-recovery* method is also referred to as *blind equalization* [5].

Supervised equalization can be brought about by either *sequence estimation* or *symbol detection*. Sequence estimator's duty is to test the possible sequences of data instead of decoding every one of the received symbols on its own and then selecting the sequence of data that is most likely to be the output [4]. This sequence estimator is also referred to as maximum likelihood sequence estimator (MLSE).

Unsupervised or blind equalization is used when the signal has no memory i.e. the signals transmitted in successive symbol intervals are interdependent. In this case, each transmitted symbol is detected separately. The constant modulus algorithm (CMA), discovered by Godard [6] and Treichler [7] serves to be a highly significant algorithm for blind equalization. Its

robustness and capability of converging before phase recovery made this algorithm very successful [5]. Another algorithm called the multimodulus algorithm (MMA) [8,9] has improved performance over CMA since it provides low steady-state mean-squared error (MSE) in addition to cancelling the necessity for phase recovery in steady-state operation [9]. Additionally, *hybrid blind equalization algorithms* are different types of blind equalization algorithms known for combining or augmenting existing cost functions to attain improved performance [5].

Nonlinear equalizers are considered significant among signal processing techniques due to their both superior performance and improved features compared with linear equalizers, in addition to the wide variety they offer. One of those features is the ability to form nonlinear decision boundaries where the Bayesian equalizer determines the performance of these equalizers. Decision Feedback Equalizers (DFEs) are one class of nonlinear equalizers with relatively improved performance. Estimating and cancelling the ISI that an information symbol induces on future symbols after it has been detected and decided upon forms the basis of decision feedback equalization [4]. The DFE can possess two structures which are either direct transversal or lattice structures. The direct form is made up of a feed forward filter (FFF) and a feedback filter (FBF). The output of a detector located in between the FFF and FBF determines the decisions that will be input to the FBF, eventually adjusting the coefficients of the FBF to eliminate the current symbol's ISI caused by past detected symbols. The remarkable feature of the DFE is its superiority over *linear transversal equalizer* (LTE) which is the most common equalizer structure. This superiority is due to its smaller minimum mean square error (MMSE) than that of the LTE. This is caused by the severely distorted channel of the LTE or when it exhibits nulls in the spectrum causing the performance of an LTE to degrade and the minimum mean squared error (MMSE), which is the basic performance criterion of the DFE, to be quite better than that of the LTE.

The goal in designing a communications system is to transmit information to the receiver with as little deterioration as possible and at the meantime to satisfy design constraints of allowed signal bandwidth, transmitted energy and cost. In digital communications systems, the *probability of bit error* (P_e), which is named *bit error rate* (BER) is generally taken to be the

measure of degradation and performance. In analog communications systems, the signal-to-noise ratio (SNR) that is related with the end of the receiver is generally the performance criterion. It's important to attain a low mean square error (MSE) and high convergence rate beside a low BER in nonlinear channel equalization. Training sequences are also an important factor that determines the efficiency of a communications system. They are intended to be as short as possible which requires the adaptation process to end in as few iterations as possible.

The application of linear equalizers to nonlinear channels does not yield the desired BER performance since they are based on linear system theory and are used for equalization of linear channels. Recently, neural networks and fuzzy technology have evolved into a powerful tool in the equalization of nonlinear channel distortions.

1.4 State of Application of Neural Networks and Fuzzy Technologies for Channel Equalization

1.4.1 Design of neural network based equalizers

Nonlinear equalizers are capable of compensating for both nonlinear and linear channel distortion. Adaptive nonlinear equalizers that implemented neural network models were used extensively primarily for noise-cancellation in various applications. A multilayer perceptron (MLP) is one of the neural network structures which is used in neural network based equalizers. MLP networks consist of feedforward neural networks having one or more layers of neurons, known as hidden neurons that are between the input and output neurons.

Filtering is the process of changing the relative amplitudes of the frequency components in a signal or eliminating some frequency components completely in a variety of applications [10]. Assigning k information bits to the $M = 2^k$ possible signal amplitudes which can be carried out in a number of ways is called *mapping* or *transformation*. Generally, the nonlinear equalization includes a channel estimator since the channel information is not available at the receiver end [12]. Filtering comprise two estimation procedures, one of them being the mapping from the available samples and the other one being the estimation of the output of the filter from the input by the realization of this mapping [11]. The mapping is more difficult

for a nonlinear filter than for a linear filter but research still goes on to effectively realize the mapping of nonlinear filters.

1.4.2 Channel equalization by using fuzzy logic

Adaptive equalizers for nonlinear channels can be developed by a variety of effective ways. Baye's probability theory [13] is capable of bringing about the optimal solution for a symbol equalizer and is referred to as the Bayesian equalizer. Symbol decision equalizers are particularly simple and less complex in terms of computationality compared with the MLSE. A channel estimate is not always necessary for them. They function as inverse filters [14] and such algorithms as recursive least square (RLS) or least mean square (LMS) are employed to base an adaptive filter. The channel inverse is found by the adaptive filter where noise provides a linear decision boundary. In general, an optimal equalizer requires decision function that is naturally nonlinear. This equalization is usually thought to be a nonlinear problem of classification with this perspective and because of this reason, linear equalizers' performance is not good enough to be optimal. This is the reason search for nonlinear equalizes providing a nonlinear decision function has been undertaken. Nonlinear equalizers employing artificial neural networks (ANNs) [15], [16], [17] and radial basis function (RBF) networks [15], [18], [19] were successfully developed. Nonlinear equalizers using ANN and RBF networks were shown to provide superior performance to linear equalizers for channels corrupted with ISI and AWGN [20]. The ANN equalizers had some discrepancies due to poor convergence and RBF equalizers provided functional behavior which is localized and required by the optimal equalizer where it was difficult to train the centers. This, however, caused the examinations to find different nonlinear equalization techniques. A fuzzy adaptive filter forms the basis of a fuzzy equalizer and this fuzzy equalizer has been suggested in [21] as the result of examinations to find alternative nonlinear equalization techniques and a fuzzy system related equalizer is offered by [22]. It was found that these equalizers had good performance but the Bayesian equalizer decision function could not be found, in addition to the difficulty of demand by fuzzy adaptive filter based equalizer, for high computational complexity.

The fuzzy logic is based on fuzzy rules that use input-output data pairs of the channel. This type of adaptive equalizers operates by processing numerical data and linguistic information.

Fuzzy equalizer depends on fuzzy IF-THEN rules which are determined by human experts. These rules use the channel's input-output data pairs and carries out the construction of the filter for nonlinear channel. The bit error rate (BER) and adaptation speed can be improved by the linguistic and numerical information.

Digital communications involving quadrature amplitude modulation (QAM) can apply the fuzzy filter with both linear and nonlinear channel characteristics as has been achieved in this thesis. The present study proposes a complex fuzzy adaptive filter with changeable fuzzy IF-THEN rules, which is an extension of the real fuzzy filter. The filter inputs and outputs are all complex valued. However, the inputs of the channels are real reciprocals of the modulated complex transmitted inputs and the equalizer outputs are real reciprocal estimates of the reciprocal channel input signals. Afterwards, the reciprocal normalized equalizer outputs are denormalized to form the final complex-valued, equalized estimate outputs of the receiver. This technique which is primarily based on *normalization* and directly applied on the transmitter, on the whole presents a new method to successfully equalize complex-valued QAM signals which are severely distorted in both linear and hostile time-varying nonlinear channel environments, by using real-valued reciprocals of the signals in question. In addition to the methodology, the membership functions derived from the training data set and the gradient-descent learning algorithm which trains the data set, represent a significant element of the nonlinear neuro-fuzzy equalizer that is capable of this adaptive channel equalization. Its superiority relies not only on its high equalization performance but also on its capability of minimizing or eliminating the non-linear channel distortions that in general, linear equalizers are not capable of doing. In turn, the fuzzy logic based neuro-fuzzy equalization is proven to be an efficient equalizer on a complex scheme such as QAM with high approximation ability in nonlinear problems in addition to the linear ones.

A fuzzy adaptive filter is based on a set of fuzzy IF-THEN rules whose function is to change adaptively in order to minimize some criterion function as new information is available [35]. A recursive least squares (RLS) adaptation algorithm is used by a fuzzy adaptive filter.

The construction of RLS fuzzy adaptive filter is accomplished by the following four steps:

1) Defining fuzzy sets in the filter input space $U \epsilon R^n$ which has membership functions covering U.
2) Constructing a set of fuzzy IF-THEN rules that either human experts determine or the adaptation procedure determines by matching input-output data pairs;
3) Constructing a filter that is based on the set of rules; and,
4) Updating the filter's free parameters by utilizing the RLS algorithm.

The fuzzy adaptive filter's main advantage is the possibility of integrating linguistic information (in the shape of fuzzy IF-THEN rules) and numerical information (in the shape of input-output pairs) into the filter uniformly. At the end, when it's time to apply the fuzzy adaptive filter to equalization problems related with nonlinear communication channel, the following fundamental differences between RLS and LMS are reached:

1) The RLS algorithm is faster than that of the LMS algorithm.
2) Having, in fuzzy terms, incorporated some linguistic description about the channel into the fuzzy adaptive filter will extensively enhance the adaptation speed of RLS.
3) The fuzzy equalizer's bit error rate is quite approximate to the bit error rate of the optimal equalizer.
4) The excess mean-square error of the RLS algorithm is inclined towards zero as the number of iterations comes nearer to infinity.

Development of neuro-fuzzy system in order to equalize channel distortion includes the following steps:

-First, the methodologies utilized to equalize channel distortions are analyzed and state of application problems of neural and fuzzy technologies for the development of an equalizer is considered.

-Second, the data transmission structure is explained and the operation structure of adaptive channel equalization utilizing neuro-fuzzy network is presented.

-Third, the mathematical model of the neuro-fuzzy network for the development of equalization system for channel distortion is presented. The learning algorithm of neuro-fuzzy system is considered.

-Fourth, the development of the neuro-fuzzy equalizer for channel distortion is presented.

-Fifth, the QAM signaling is explained and its application on nonlinear neuro-fuzzy network is presented. The simulation results of the equalizer using QAM signals and analytical tables demonstrating the performance of the equalizer are presented. Additionally, tables comparing the different QAM constellations are presented.

1.5 Summary

In this chapter, the application of channel equalization is explained. The types of distortions in channels and the types of equalizers used to minimize them are explained with their classifications and properties. Performance criteria of equalizers, namely bit error rate (BER), signal-to-noise ratio (SNR) and convergence rate with their ideal indications are stated. Neural networks and fuzzy logic are particularly discussed and explained with their structures and features. The methods of equalization using neural networks, specifically filtering is described. Different types of algorithms, networks and equalizers used especially for difficult nonlinear channels are defined.

Fuzzy IF-THEN rules which constitute the basis of fuzzy logic are described to point out their significance in channel equalization. The steps of constructing a fuzzy adaptive filter using these rules are defined. The methods used in equalizing QAM signals applied on neuro-fuzzy network and the gradient-descent learning algorithm as part of the equalization system are described as well.

CHAPTER 2

STRUCTURE OF CHANNEL EQUALIZATION

2.1 Overview

All communications systems are composed of three fundamental subsystems which are transmitter, channel and receiver (Fig. 2.1). A transmitter's task is to transmit information signal through physical channel or transmission medium after converting it into a form which is convenient for transmission. The receiver's task, on the other hand, is to produce an accurate replica of the transmitted symbol sequence by recovering the message signal that the received signal contains. The communications channel acts as a connector between the transmitter and the receiver sending the electrical signal from the transmitter to the receiver. The unknown channel characteristics cause distortions to the transmitted signal before it reaches the receiver.

Figure 2.1 Basic components of a communications system

Digital communications systems are preferred more compared with the analog ones due to increasing demand for data communication and because digital transmission provides data processing options and flexibilities that analog transmission cannot offer. The distinguishing feature of a digital communications system is that it sends a waveform from a finite set of possible waveforms during a finite interval of time as opposed to an analog communication system that transmits a waveform from unlimited number of various waveforms which have theoretically infinite resolution. The message from the source which is represented by an information waveform is encoded before transmission so that transmission error can be detected and corrected by the receiver. At the receiver end, the message signal must be decoded before being used. The distortions preventing the correct transmission of signals are mainly intersymbol interference (ISI) and noise. Noise is meant to be unwanted electrical signals which exist in electrical systems. The equalization of channel is an efficient technique

employed to reduce or eliminate the obscuring effect of distortion caused in the channel. This chapter outlines the structure of data transmission system and the functions of its main components as well as the equalization of channel distortion.

2.2 Architecture of Data Transmission Systems

A communications channel is an electrical medium which connects the transmitter and the receiver, providing the data transmission from a source which generates the information to one or more destinations. In the analysis and design of communication systems, the characteristics of the physical channels through which the information is transmitted, are of particular importance. Wire lines or free space may be used in the communications path from the transmitter to the receiver. The examples for wire lines are coaxial cables, wire pairs and optical fibers. These are widely used in terrestrial telephone networks, even though infrared and optical free space links such as video, remote controls for TV and hi-fi equipment as well as some security systems may be used in different situations, as well. This point of transmission medium is where most of the attenuation and noise is observed [23].

The receiver functions to reverse the signal processing steps performed by the transmitter recovering the original message signal by compensating for any signal deteriorations caused by the channel. This involves amplification, filtering, demodulation and decoding and in general is a more complex task than the transmitting process.

There are many reasons as to why digital communication systems are preferred over analog systems. Digital communication systems (DCSs) represent an increase in complexity over the equivalent analog systems. The principal advantages and reasons of DCS's being the preferred option instead of analog communication systems can be listed as:

1. The ease with which digital signals, compared with analog signals, are regenerated.
2. Digital systems are not as prone to distortion and interference as analog systems.
3. Increased demand for data transmission.
4. Increased scale of integration, sophistication and reliability of digital electronics for signal processing, combined with decreased cost.
5. Facility to source code for data compression.

6. Possibility of channel coding (line and error control coding) to minimize the effects of noise and interference.
7. Ease with which bandwidth, power and time can be traded off in order to optimize the use of these limited resources.
8. Standardization of signals, irrespective of their type, origin or the services they support, leading to an integrated services digital network (ISDN)
9. Digital hardware can be implemented more flexibly than analog hardware.
10. Various types of digital signals such as data, telephone, TV and telegraph can be considered identical signals in transmission and switching [24].

Modulation, which is part of the transmission and equalization process, involves encoding information from a message source in a way that is convenient for transmission. It is accomplished by translating a baseband message signal to a bandpass signal at frequencies which are quite high when compared with the frequency of baseband. It is also referred to as the *mapping* of the baseband input information waveform into the bandpass signal. The bandpass signal is referred to as the *modulated* signal and the baseband message signal is referred to as the *modulating* signal. Modulation can be accomplished by varying the frequency, phase or amplitude of a high frequency carrier in conformity with the amplitude of the message signal. *Demodulation*, on the other hand, is the process of extracting the baseband message from the carrier in order to enable the aimed receiver (also known as the *sink*) to process and interpret it. In digital wireless communication systems, it's possible to represent the modulating signal as a time sequence of pulses or symbols, where each symbol has m finite states. The representation of n bits of information where $n = \log_2 m$ bits/symbol, is done by each symbol [4].

The block diagram illustrated in Fig. 2.2 can describe communications systems. The source of data is the signal generator that produces the information to be transmitted and modulated. This information is in the form of a message symbol that can consist of a single bit or a grouping of bits.

In order to make the transmission more efficient in terms of the time it takes and/or bandwidth it requires, encoder is employed as a signal processor that converts the sources of digital

information into binary form, i.e. each symbol is encoded as a binary word. Encoding is performed so as to enable the signal processor in the receiver to detect and correct errors which will provide the minimization and/or elimination of bit errors caused by noise in the channel.

The procedure used for detecting and correcting errors is called *coding*. Coding includes adding redundant (extra) bits to the stream of data. The redundant bits like parity bits are employed by the decoder and serve to correct errors at the receiver output even though a high degree of redundancy may increase the bandwidth of the encoded signal. Codes can be classified into two broad categories as *block codes* and *convolutional codes*. The main difference is that block coder is a *memoryless* device whereas a coder having a *memory* produces a convolutional coder. Hamming Codes, Golay Codes, Hadamard Codes, Cyclic Codes, BCH (Bose-Chaudhuri-Hocquenghem) Codes and Reed-Solomon Codes are some examples of block codes. In addition to block codes and convolutional codes, a new family of codes, called *turbo codes* is used recently and is being incorporated in 3G wireless standards. Turbo codes combine the capabilities of convolutional codes with channel estimation theory and can be thought of as nested or parallel convolutional codes. When implemented properly, turbo codes allow coding gains which are far superior to all previous error correcting codes and permit a link of wireless communications to come surprisingly near to realizing the Shannon capacity bound [4].

Each digital word has n binary digits and there are $M = 2^n$ unique code words which are possible where each code word corresponds to a certain amplitude level. However, each sample value from the analog signal could be any one of an infinitely high number of levels for the digital word which represents the amplitude closest to the actual sampled value to be utilized. That is known as *quantizing* [2]. Gray coding was used as the mapping of bits along the in-phase and quadrature axes of the QAM constellation as part of this thesis study. The Gray code has been selected since it has change of only one bit for each change of step in the quantized level. Multisymbol signaling can be thought of as a coding or bit mapping process

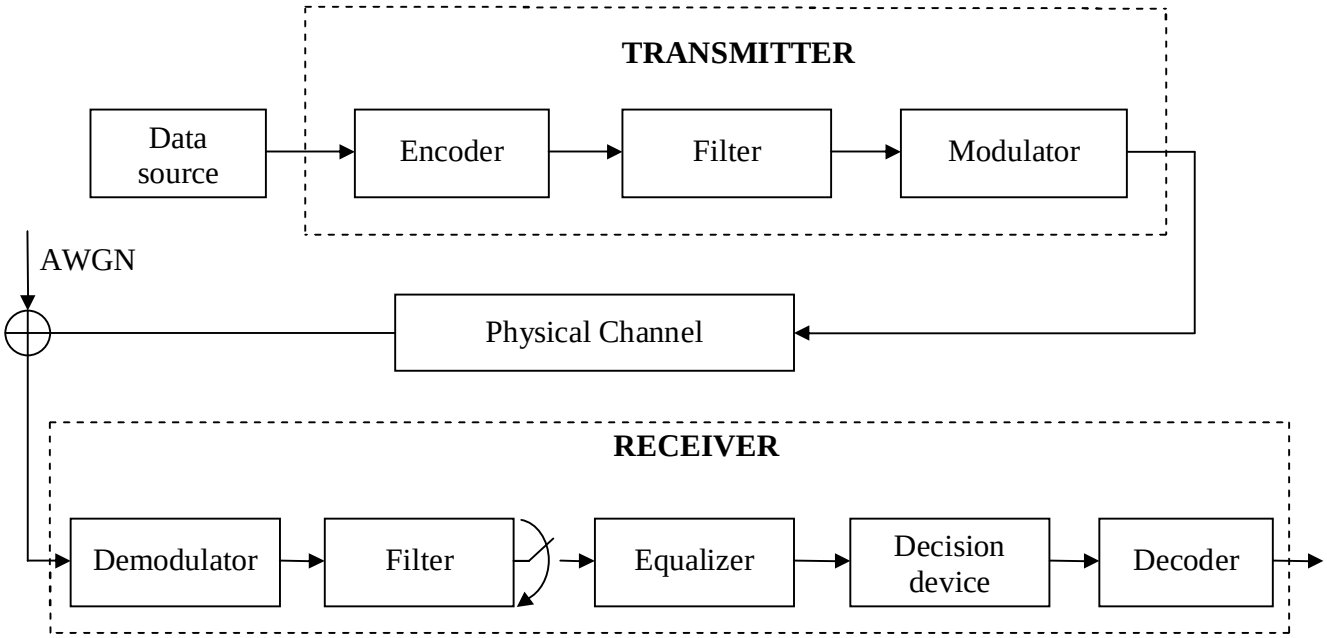


Figure 2.2 Architecture of a digital communications system [39]

in which *n* binary symbols (bits) are mapped into a single *M*-ary symbol. A detection error in a single symbol can therefore translate into several errors in the corresponding decoded bit sequence. The bit error rate (BER), therefore relies not only on the probability of symbol error and the symbol entropy but on the code or bit mapping used and the types of error which occur as well. If a Gray code is used to map binary symbols to phasor states, this type of error results in only a single decoded bit error [23]. Consequently, single errors in the receiver will cause minimal errors in the recovered level.

There are many criteria used in the evaluation of the performance of a communications system. The optimum system that is considered close to being ideal or perfect for digital systems is the one that minimizes the bit error rate (BER) at the receiver output subject to constraints on channel bandwidth and transmitted energy. This raises the matter of inventing a system with no bit error at the output even when there is noise in the channel. Shannon demonstrated in 1948 that it was possible to calculate a channel capacity *C* (bits/s) in the way that if the rate of information was less than *C*, the probability of error would approach to zero. In this case, the maximum possible bandwidth efficiency $\eta_{B\max}$, which is defined as the

capability of a modulation scheme to accommodate data within a limited bandwidth, is restricted by the channel noise and is stated by the channel capacity formula in Eq.2.1. Shannon's channel capacity formula is applicable to AWGN and is given by

$$\eta_{B\max} = \frac{C}{B} = \log_2\left(1+\frac{S}{N}\right)$$

or (2.1)

$$C = B\log_2\left(1+\frac{S}{N}\right)$$

in which C is the channel capacity (bits per second), B is the transmission bandwidth, S is the average power of the transmitted signal and N is the power spectral density of the white Gaussian noise. S/N is called the signal-to-noise ratio. Shannon also showed that errors that a noisy channel induces, could be decreased to any desired level by encoding the information properly, without sacrificing the rate of information transfer.

The physical medium or the channel that the message signal is transmitted through, induces distortions like intersymbol interference (ISI) and noise. The receiver, on the other hand is responsible for separating the source information from the received modulated signal which is distorted by noise that is usually random, additive white Gaussian noise (AWGN). The receiver's duty is to take the corrupted signal at the output of the channel and to convert it to a baseband signal that the baseband processor could handle. The baseband processor eliminates or minimizes this signal and distributes an estimate of the source information to the output of the communications system [2]. Demodulation process is employed at the receiver to the signal in order to recover the transmitted signal in its baseband form and make it ready to be processed by the receiver filter. At the end, the decision device reconstructs the encoded message signal depending on the decisions of the equalizer and the decoder reconstructs the sequence of transmitted signals by bringing about the reverse operation of the encoder.

2.3 Channel Characteristics

Channels must have appropriate frequency band for their transmission medium. The processed baseband signal is converted by the transmitter circuit into this frequency band. If the channel is a fiber-optic cable, the carrier circuits convert the baseband input to light frequencies and the transmitted signal is light.

Channels are classified as wire and wireless channels. Some examples of wire channels can be counted as coaxial cables, fiber-optic cables, twisted-pair telephone lines and waveguides whereas air, vacuum and seawater are examples of wireless channels.

The constraints channels may introduce are in favor of a particular type of signaling. Generally, the signal is attenuated by the channel so that the channel or the noise produced by an imperfect receiver deteriorates the delivered information from that of the source [2]. There are various sources that cause noise; those sources may be natural electrical disturbances such as lightning, artificial sources like ignition systems in cars, switching circuits in a digital computer or high voltage transmission lines. The channel is likely to involve amplifying devices such as satellite transponders in space communication systems or repeaters in telephone systems that help the signal to be above the noise level. In addition to noise, multiple paths that arise between the input and output of channel involve attenuation characteristics and time delays. The attenuation characteristics may vary with time, which makes the signal fade at the channel output. Fading of that type can be observed while listening to distant shortwave stations.

Another significant characteristic of channels is bandwidth. In general terms, bandwidth is defined to be the width of a positive frequency band of waveforms whose magnitude spectra are even about the origin $f = 0$. Bandwidth in a channel must be enough to accommodate the signal but reject the noise. High bandwidth allows more users to be assigned as well as more information to be transmitted. Some examples of band limited channels are telephone channels and digital microwave radio channels. When the channel is band limited to WHz, any frequency components above W will not be passed by the channel. In turn, the bandwidth of the transmitted signal will be limited to W Hz, as well. When the channel is not ideal (i.e.

$|f| \leq W$), signal transmission at a symbol rate equivalent of or exceeding W concludes as intersymbol interference (ISI) among a number of adjacent symbols. In addition to telephone channels, other physical channels which exhibit some form of time dispersion and thus introduce ISI, are also available. Radio channels like shortwave ionospheric propagation (HF) and tropospheric scatter are two examples of time-dispersive channels. In these channels, time dispersion and hence, ISI is the consequence of multiple propagation paths that have different path delays [1]. In addition to noise, multipath propagation and ISI, there are other impairments in the channels specifically nonlinear distortion, frequency offset and phase jitter. Channel impairments affect the transmission rate over the channel and the modulation technique to be used. Depending on the rates, bandwidth efficient modulation techniques are employed and some form of equalization is employed accordingly.

2.4 Channel Distortions

Channels which are used to transmit data distort signals in both amplitude and phase. In addition to the nature of the channel itself, other factors like linear distortion, nonlinear distortion and frequency offset are significant factors causing these distortions.

Linear distortion occurs in linear time-invariant systems in which channels are characterized as band-limited linear filters. Those channels like telephone channels are part of digital communications systems where distortionless transmission is highly desired. A linear time-invariant system will produce two types of linear distortion which are *amplitude distortion* and *phase distortion*. In order to have distortionless transmission with linear time-invariant systems, the first requirement is that the transfer function of the channel must be given by

$$H(f) = \frac{Y(f)}{X(f)} = Ae^{-j2\pi fT_d} \tag{2.2}$$

which means that in order to have no distortion at the system output, the following requirements have to be met:

1. Flat amplitude response. That is,

$$|H(f)| = \text{constant} = A \tag{2.3a}$$

2. The phase response that is a *linear* function of frequency. That is,

$$\theta(f) = \angle H(f) = -2\pi f T_d \tag{2.3b}$$

When the first condition is satisfied, no *amplitude distortion* exists and when the second condition is satisfied, no *phase distortion* exists. The second requirement is related with the *time delay* of the system and it is defined as

$$T_d(f) = -\frac{1}{2\pi f}\theta(f) = -\frac{1}{2\pi f}\angle H(f) \tag{2.4}$$

and it is compulsory that

$$T_d(f) = \text{constant} \tag{2.5}$$

for distortionless transmission. If $T_d(f)$ is not constant, there is phase distortion since the phase response, $\theta(f)$, is not a linear function of frequency.

Nonlinear distortion in telephone channels arises from nonlinearities in amplifiers and compandors used in the telephone system. This type of distortion is usually small and it is very difficult to correct [1]. There will be nonlinear distortion on the output signal if the voltage gain coefficients from the second order on, are not zero. There are three types of nonlinear distortions associated with the amplifiers which are *harmonic distortion*, *intermodulation distortion* (IMD) and *cross-modulation distortion*. Harmonic distortion occurs at the amplifier output and is caused by first and second order frequencies of the amplifier output. The intermodulation distortion is produced by cross-product term of the amplifier input-output equation whereas the cross-modulation distortion is caused by the third order distortion products of the amplifier output.

In addition to linear and nonlinear distortions, signals transmitted through telephone channels are subject to the impairment of frequency offset. A small frequency offset which is mostly less than 5 Hz, results from the use of carrier equipment in the telephone channel. High-speed digital transmission systems that use synchronous phase-coherent demodulation cannot

tolerate this type of offset. This offset is compensated for by the carrier recovery loop in the demodulator.

Phase jitter is basically a low-index frequency modulation of the transmitted signal with the low frequency harmonics of the power line frequency. Phase jitter poses a serious problem in digital transmission of high rates. Yet, it can be tracked and compensated for, to some extent, at the demodulator.

Distortion can occur within the transmitter, the receiver and the channel. As opposed to noise and interference, distortion appears when the signal is turned off.

2.4.1 Multipath propagation

Multipath fading occurs to varying extents in many different radio applications. It is caused whenever radio energy reaches the receiver by more than one path. Multiple paths may also occur due to ground reflections, reflections from stable tropospheric layers and refraction by tropospheric layers with extreme refractive index gradients [23]. Scattering obstacles also cause multipath propagation to some other systems like urban cellular radio systems.

There are two principal effects of multipath propagation on systems, their relative severity depending essentially on the relative bandwidth of the resulting channel compared with that of the signal being transmitted. The fading process is governed by changes in atmospheric conditions for fixed point systems such as the microwave radio relay network. The path delay spread often is adequately short for the channel frequency response to be essentially constant over its operating bandwidth. If that happens, fading is considered flat because all signal frequency components become prone to the same fade at any given instant. In the case of path delay spread being longer, the channel frequency response is likely to change rapidly on a frequency scale that can be compared with signal bandwidth. If that happens, the fading is considered frequency selective and the received signal is subject to severe amplitude and phase distortion. Adaptive equalizers may then be required to flatten and linearize the overall characteristics of channel. The flat fading effects can be combated by increasing transmitter power whilst the effects of frequency selective channel cannot. A fade margin is usually designed into the link budget to offset the expected multipath fades for microwave links

which are subject to flat fading. The magnitude of this margin depends on the required availability of the link.

Paths of multiple propagation that have different path delays cause time dispersion and ISI in time-dispersive channels. The reason for calling these channels *time-variant multipath channels* is that the relative time delays among the paths and the number of paths vary with time. Various frequency response characteristics are caused by the time-variant multipath conditions resulting in inappropriate frequency response characterization for time-variant multipath channels, which is used for telephone channels. Instead, scattering function statistically characterizes these radio channels. The scattering function is a two-dimensional representation of the average received signal power which depends on Doppler frequency and relative time delay.

2.4.2 Intersymbol interference

Rectangular pulse signaling, in principle, has a spectral efficiency of 0 bits/s/Hz since each rectangular pulse has infinite absolute bandwidth. In practice, of course, rectangular pulses can be transmitted over channels with finite bandwidth if a degree of distortion can be tolerated.

In digital communications, it might appear that distortion is unimportant since a receiver must only distinguish between pulses which have been distorted in the same way. If the pulses are filtered improperly as they pass through a communications system i.e. if the distortion is severe enough, they will spread in time. The decision instant voltage might then arise not only from the current symbol but also from one or more preceding pulses. *Intersymbol interference* (ISI) is caused when smearing the pulse for each symbol into adjacent time slots occurs. The pulses would have rounded tops instead of flat ones with a restricted bandwidth. What's important about ISI is the decision instant. The decision instant can be defined as the sampling instant (or sampling point) at which each time slot of the transmitted or received waveform begins. It is at this point that ISI occurs due to the smearing effect of the pulse. This smearing will cause unwanted contributions from the adjacent pulses that are likely to degrade bit error rate (BER) performance. The decision instant shows an important point: *The*

performance of digital communications systems is only related with decision instant ISI. If ISI occurs at times that are not decision instants, it does not matter [23].

If the signal pulses could be persuaded to pass through zero crossing point (of the time axis) at every decision instant (except one), then ISI would no longer be a problem. This suggests a definition for an ISI-free signal, i.e.: *If a signal passes through zero at all instants that are not one of the sampling instants, it's an ISI-free signal* [23].

While transmitting information with pulses over an analog channel, the original signal is a discrete time sequence (or an acceptable approximation); the received signal is a continuous time signal. The channel can be considered a low-pass analog filter, by that means, smearing or spreading the shape of the impulse train into a continuous signal with peaks that are related with the original pulses' amplitudes. Convolution of the pulse sequence by a continuous time channel response could describe the operation in terms of mathematics. The convolution integral is the beginning of the operation:

$$x(k) = h(k) * s(k) = \int_{-\infty}^{\infty} h(\tau)s(k-\tau)d\tau = \int_{-\infty}^{\infty} s(\tau)h(k-\tau)d\tau \tag{2.6}$$

where $x(k)$ denotes the received signal, $h(k)$ denotes the channel impulse response and $s(k)$ denotes the input signal. The second half on the right side of the above equation illustrates the commutativity property of the convolution operation.

Component $s(k)$ is the input pulse train that is comprised of periodically transmitted impulses of varying amplitudes, for that reason;

$$\begin{aligned} s(k) &= 0 \qquad \text{for } k \neq nT \\ s(k) &= S_n \qquad \text{for } k = nT \end{aligned} \tag{2.7}$$

where T is the symbol period. Here, it is meant that the only significant values of the variable of integration in the integral of equation (2.6), are those for which $k = nT$. A different value of k amounts to multiplication by 0 and for that reason, $x(k)$ can be stated as

$$x(k) = \sum_{n=-\infty}^{\infty} s_n h(k - nT) \tag{2.8}$$

The above equation that represents $x(k)$ is more similar to the convolution sum, however, it nevertheless is the description of a continuous time system. It illustrates that the received signal is comprised of the addition of a large number of shifted and scaled impulse responses of continuous time system. The amplitudes of the transmitted pulses of $x(k)$ scale the impulse responses.

The first term in Eq. 2.8 is the component of $x(k)$ because of the N^{th} symbol. The centre tap of the channel impulse response multiplies it. ISI terms are the other product terms in the summation. The appropriate samples in the tails of the channel impulse response scale the input pulses in the neighborhood of the N^{th} symbol.

2.4.3 Noise

In communications systems, the received waveform is usually classified as the desired part which contains the information and the extraneous or unwanted part. The desired part is the *signal* and the unwanted part is the *noise*. Noise limits our ability to communicate and causes more power consumption during the transmission of information. Minimizing the noise effects is achieved after enhancing the power amount in the transmitted signal. Yet, factors like equipment and various practical limitations restrict the level of power in the signal which is transmitted.

The most frequently encountered problem in the transmission of signals through any channel is additive noise that is generally generated internally at the receiver end by components like solid-state devices of a subsystem and resistors employed in the implementation of the communications system. That is at times referred to as *thermal noise*. Thermal noise is produced by the random motion of free charge carriers (usually electrons) in a resistive medium. Additive noise generated by the electronic components is usually found in a storage system's readback signal, as in the case of a radio or telephone communication system. When such noise occupies the same frequency band that the desired signal occupies, suitable design of the transmitted signal and its demodulator at the receiver can minimize its effect [23].

Another problem in transmission is the *non-thermal noise*, also known as the *shot noise*. Although the time averaged current flowing in a device may be constant, statistical fluctuations will be present if individual charge carriers have to pass through a potential barrier. The potential barrier may, for example, be the junction of a PN junction diode, the cathode of a vacuum tube or the emitter bus junction of a bipolar transistor. Such statistical fluctuations constitute shot noise.

Noise that arises from external sources can be coupled into a communication system by the receiving antenna. *Antenna noise* which is dominated by the broadband radiation produced in lightning discharges associated with thunderstorms, below 30 MHz originates from several different sources. This radiation is trapped by the ionosphere and propagates worldwide. Such noise is sometimes referred to as atmospheric noise.

Noise can be classified into categories as:

a. White noise: A stochastic process which has a flat power spectral density over the entirety of frequency range. It's not possible to express that sort of noise using quadrature components because of its wideband character. When problems tackling the narrowband signal demodulation in noise are in question, modeling the additive noise process as white and representing the noise using quadrature components is mathematically convenient. It's possible to accomplish this after putting forward that the signals and noise at the receiver managed to pass through an ideal bandpass filter, which has a passband including the spectrum of the signals but is a lot wider. The noise that is the result of passing the white noise process through a spectrally flat bandpass filter is referred to as *bandpass white noise*.

b. Electromagnetic Noise: Usually found in electrical devices like television and radio transmitters and receivers. They can be present at all frequencies.

c. Impulse Noise: An additive disturbance which arises primarily from the switching equipment in the telephone system. It is made up of short-duration pulses having random duration and amplitude.

d. Acoustic Noise: Present in almost all conversations and limit telecommunications environments such as telephone circuits and hands-free telephones. It may be

unnoticeable or distinct, depending on the time delay involved. If the delay between the speech and its echo (noise) is short, the noise is unnoticeable, but perceived as a form of spectral distortion referred to as *reverberation*. If, however, the delay exceeds a few tens of milliseconds, the noise is distinctly noticeable [25]. Background noise generated in a car cabin, air conditioners and computer fans represent some types of acoustic noise.

e. Processing Noise: Modeled as a zero-mean, white-noise process in data communication systems. It is the result of digital analog processing of signals, e.g. lost data packets in digital data communications systems or quantization noise in digital coding of image or speech.

f. Colored Noise: It's a Gaussian type noise which is part of wideband signal processes with non-constant spectrum. Autoregressive noise and brown noise are some examples of the non-white, colored noise.

Gaussian noise and specifically the additive white Gaussian noise (AWGN) is the most frequently encountered type of noise in communication systems. It represents the simplest mathematical model for a communication channel. Below are given a list of channel models in which the effects of noise on electrical communication and the most important characteristics of the transmission channels are investigated.

2.4.3.1 The additive noise channel

Contaminating noise in signal transmission usually has an additive effect in the sense that noise often adds to the information bearing signal at various points between the source and the destination. Random additive noise process $n(k)$ whose channel has a mathematical model as shown in Fig. 2.3, corrupts the transmitted signal $x(k)$. The additive noise becomes white when the random process has a power spectral density (PSD) which is constant over all frequencies and becomes the most often assumed model of additive white Gaussian noise (AWGN), when the noise has a Gaussian distribution. AWGN contains a uniform continuous frequency spectrum over a particular frequency band and the majority of physical communication channels implements this model since it is mathematically tractable.

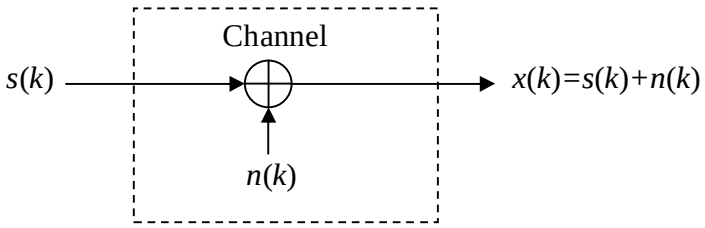


Figure 2.3 The additive Gaussian noise channel [39]

2.4.3.2 The linear filter channel

Filtering is an operation which includes extracting information about a quantity of interest from data with noise at time t by using measured data that includes t. A filter is considered linear when filtering, smoothing or predicting the amount at the filter output is done and this amount linearly depends on the observations applied to the filter input [25].

Linear filter channels are those that enable the transmitted signals to remain in specified bandwidth limitations without interfering with each other. The mathematical model including the additive noise is illustrated in Fig. 2.4 in which $s(k)$ is the channel input and the channel output is represented as

$$x(k) = s(k) * h(k) + n(k) = \int_{-\infty}^{+\infty} h(\tau)s(k-\tau)d\tau + n(k) \tag{2.9}$$

in which $h(\tau)$ is the linear filter impulse response and * denotes convolution.

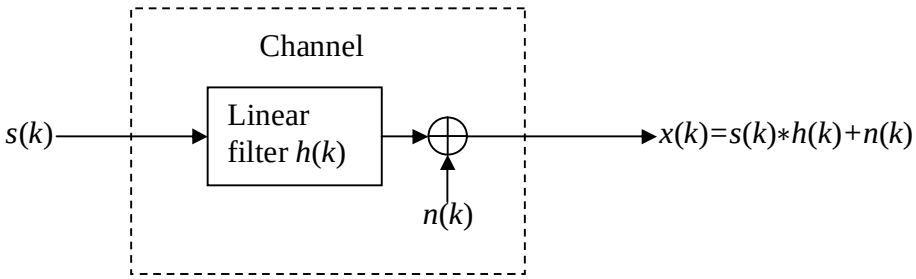


Figure 2.4 Linear filter channel with additive noise [39]

When attenuation is applied to the signal while being transmitted, the received signal becomes

$$x(k)=\alpha s(k)+n(k) \tag{2.10}$$

where α is the attenuation factor.

2.4.3.3 The linear time-variant filter channel

Mobile systems such as a moving vehicle and wireless channels such as radio channels cause multipath propagation resulting in time-varying fading signals because their frequency response characteristics are time-variant. The time-varying mobile channel characteristics necessitate using a channel equalizer which continuously adapts to these characteristics, effectively implementing a filter which is matched to these characteristics. A time-variant channel impulse response $h(\tau;k)$ is a characteristic of such time-variant linear filters. The channel response $h(\tau;k)$ contains an impulse applied at time k-τ where τ stands for the elapsed-time variable. The linear time-variant filter channel containing additive noise and the signal of channel output when $s(k)$ is the input, becomes

$$x(k) = s(k) * h(\tau;k) + n(k) = \int_{-\infty}^{+\infty} h(\tau;k)s(k-\tau)d\tau + n(t) \tag{2.11}$$

in which the time-variant impulse response has the following representation

$$h(\tau;k) = \sum_{n=1}^{L} a_n(k)\delta(k-\tau_k) \tag{2.12}$$

where the $\{a_n(k)\}$ denotes the possibly time-variant attenuation factors for the L multipath propagation paths. Substituting Eq. 2.12 into Eq. 2.11 makes the received signal

$$x(k) = \sum_{n=1}^{L} a_n(k)\delta(k-\tau_k) + n(k) \tag{2.13}$$

where each of the L multipath components is attenuated by $\{a_n(k)\}$ and delayed by $\{\tau(n)\}$.

A large majority of physical channels are formed by the three defined mathematical models and the communication systems are analyzed and designed based on these three channel models.

2.5 Summary

This chapter outlines the structure of channel equalization system. The factors causing distortions in the channel and their properties are explained and discussed in detail. The noise types and interferences are described in detail in addition to their effects on the channel and the ways of removing them from the channel.

The types of channels used within the data transmission system have been discussed. Mathematical models representing various types of channels have been outlined and described. Mathematical formulas representing the input, impulse response and the output of the channel have been explained beside the channel characteristics of each type.

CHAPTER 3

MATHEMATICAL BACKGROUND OF A NEURO-FUZZY EQUALIZER

3.1 Overview

When the channel distortion in communications applications is extreme and linear equalizers are not able to deal with them, nonlinear equalizers are employed instead. A linear equalizer doesn't have good performance on channels that have amplitude characteristics containing deep spectral nulls or on channels containing nonlinear distortions. In an effort to compensate for the channel distortion, the linear equalizer puts a vast gain in the vicinity of the spectral null for the channel distortion compensation and consequently increases the amount of additive noise the received signal has got.

Neural networks can be considered mathematical models of brain and mind activities. The main purpose of neural networks is to form the organization of numerous simple processing elements into layers for achieving tasks with higher level sophistication. High computation rates, high capability for nonlinear problems, massive parallelism and continuous adaptation are among the properties of neural networks. Those features turn neural networks into desired tools for different sorts of applications [28]. Neural networks have been put forward for equalization problems because of these attractive properties and their nonlinear capability.

On the other hand, neural networks have some weaknesses related with their individual models. Their computational power is low and learning capability is limited. At this point, the fuzzy systems have been considered to compensate these weaknesses with their capabilities of logically reaching conclusions on a more advanced (linguistic or semantic) level.

This chapter describes the synthesizing of fuzzy logic with neural networks, the operation and structure algorithms of neuro-fuzzy system as the channel equalization basis of QAM signals.

3.2 Neuro-Fuzzy System

Intelligent control is largely *rule based*, whereas classical control is rooted in the theory of linear differential equations, because the dependencies involved in its deployment are much

too complex to permit an analytical representation. In tackling such dependencies, it is expedient to use the mathematics of fuzzy systems and neural networks. The power of fuzzy systems lies in their ability to measure the quantity of linguistic inputs and to quickly provide a working approximation of complex and frequently unknown input-output rules of system. The power of neural networks is in their ability to *learn* from data. It's possible to combine neural networks and fuzzy logic in a number of ways and both have advantages that provide flexibility and effectiveness when combined. Fuzzy adaptive filters are effective because of their data approximation ability in nonlinear problems and therefore are widely used in signal processing problems. Fuzzy logic equalizers usually require fewer training samples than conventional equalizers, especially for linear channels. They are capable of yielding better error performance and also perform better in the presence of channel nonlinearities [29].

Neural networks supply algorithms for numeric classification, optimization and associative storage. When fuzzy logic and neural networks are integrated, the emerging neuro-fuzzy system becomes capable of training the network in a shorter time as a result of decreased number of nodes of the network. There is a natural synergy between neural networks and fuzzy systems that makes their hybridization a powerful tool for intelligent control and other applications.

3.3 Fuzzy Inference Systems

3.3.1 Architecture of fuzzy inference systems

Fuzzy Inference Systems (FIS) are one of the well known applications of fuzzy sets theory and fuzzy logic. They are used in achieving classification tasks, process control, offline process simulation and diagnosis and online decision support tools. The power of FIS depends on the twofold identity of both being capable of managing linguistic concepts and being universal approximators which are capable of performing nonlinear mappings between inputs and outputs.

FIS is often utilized for process simulation or control. Either expert knowledge or data can design them. Knowledge based FIS solely may suffer from a loss of accuracy, for complex

systems which is the most important motivation to use fuzzy rules concluded from data [30]. The functional blocks as explained below, comprise a fuzzy inference system (Figure 3.1).

- Determining a set of fuzzy IF-THEN rules. Fuzzy rules are composed of linguistic statements which describe the way the FIS makes a decision about the classification of an input or the controlling of an output.
- Fuzzifying the inputs, which involves transforming the crisp inputs into degrees to match with linguistic values, using the input membership functions defined by a data base.
- Combining the fuzzified inputs in accordance with the fuzzy rules to set up a rule strength (also called weight or fire strength).
- Determining the rule's consequence by putting together the rule strength and the membership function of the output.
- Combining the consequences so as to obtain an output distribution.
- Defuzzification of the output distribution which involves transforming the fuzzy rules of the inference into crisp output.

The operations upon fuzzy IF-THEN rules are explained in the steps below:

1. Mapping the inputs to membership values of each linguistic label, utilizing a set of input membership functions on the premise part (fuzzification process).
2. Computation of the rule strength by combining the fuzzified inputs (combining the membership values), by utilizing the process of the fuzzy combination. (the fuzzy combinations are also referred to as T-norms which are used in making a fuzzy rule and involve the operators of "and", "or" and sometimes "not")
3. Generating the qualified fuzzy or crisp consequent of each rule according to the rule strength.
4. Combining the entirety of the fuzzy rule outputs to attain one fuzzy output distribution and then aggregating the qualified consequent to produce a single crisp output (Defuzzification of output distribution).

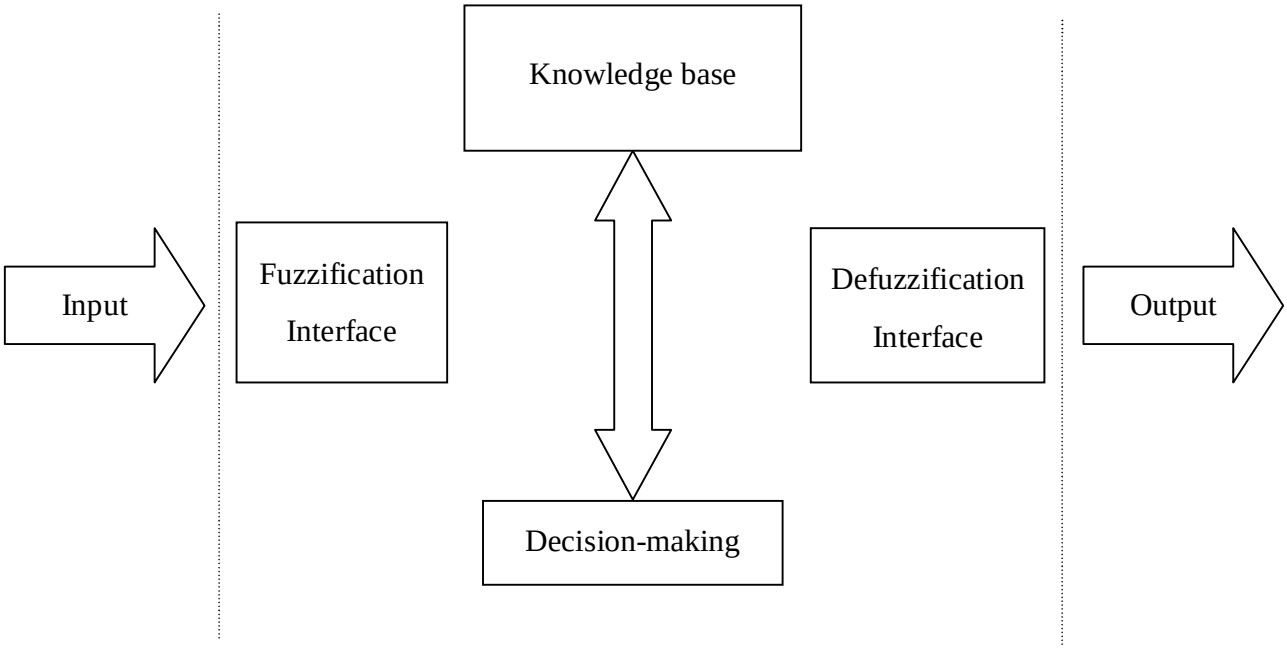


Figure 3.1 Structure of fuzzy inference system [39]

3.3.2 Rule base fuzzy if-then rule

The fuzzy knowledge base that includes a set of fuzzy IF-THEN rules forms one of the basic blocks of a fuzzy system. The following is the form of expression that represents fuzzy IF-THEN rules or fuzzy conditional statements [37].

$$\text{If } u \text{ is } A \text{ Then } y \text{ is } B \tag{3.1}$$

where u and y represent the input and output linguistic variables, A and B represent the labels of the fuzzy sets characterized by appropriate membership functions. A denotes the premise and B denotes the consequent part of the rule.

There are many forms representing IF-THEN rules among which Single Input Single Output (SISO), given by statement (3.1) is the simplest. Multi-Input Single Output (MISO) of the below given statements (3.2) and (3.3), are the other forms.

$$\text{If } u_1 \text{ is } A_1^j \text{ and } u_2 \text{ is } A_2^j \text{ and ,…., and } u_n \text{ is } A_n^l \text{ Then } y_q \text{ is } B_q^p \tag{3.2}$$

If u_1 is A_1^j and u_2 is A_2^j and ,…., and u_n is A_n^l Then y_1 is B_1^r and y_2 is B_2^s (3.3)

The membership functions describe the fuzzy values *A* and *B* and Figure 3.2 demonstrates the most widely used types of membership functions with their shapes.

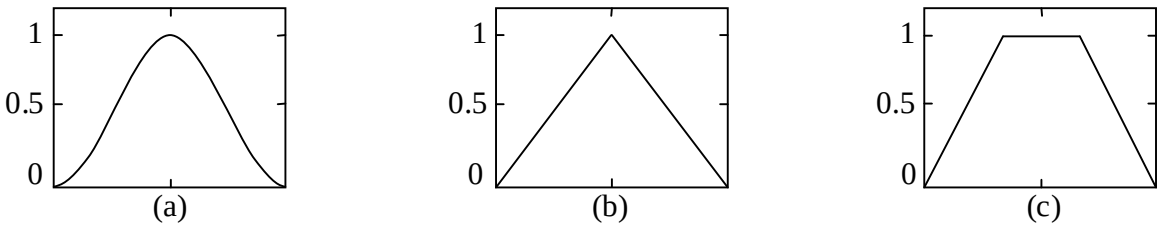


Figure 3.2 Examples of membership functions (a) bell, (b) triangular, (c) trapezoidal

The following exponential function is one representation of a decision function that produces a bell curve.

$$\mu(x) = \exp\left[\frac{-(x - x_0)^2}{2\sigma^2}\right] \tag{3.4}$$

where x is the independent variable on the universe, x_0 denotes the position of the peak relative to the universe and σ denotes the standard deviation.

The expressions (3.5) and (3.6) represent triangle and trapezoidal membership functions, respectively.

$$\mu(x) = \begin{cases} 1 - \frac{\overline{x} - x}{\overline{x} - x_l}, & x_l < x < \overline{x} \\ 1 - \frac{x - \overline{x}}{x_r - \overline{x}}, & \overline{x} < x < x_r \end{cases} \tag{3.5}$$

$$\mu(x) = \begin{cases} 1 - \frac{\overline{x}_l - x}{\overline{x}_l - x_l}, & x_l < x < \overline{x}_l \\ 1, & \overline{x}_l < x < \overline{x}_r \\ 1 - \frac{x - \overline{x}_r}{x_r - \overline{x}_r}, & \overline{x}_r < x < x_r \end{cases} \tag{3.6}$$

The following representation is the form of the types of rules, called Takagi and Sugeno fuzzy rules because the consequent part of the fuzzy rules is a mathematical function of the input variables.

$$\text{If } A1(x_1),\ A2(x_2),\ \ldots\ldots\ ,\ An(x_n) \text{ then } Y=f(x_1, x_2, \ldots, x_n) \tag{3.7}$$

where the premise part is fuzzy and the function f in the consequent part is usually a linear or quadratic mathematical function.

$$f = a_0 + a_1 \text{x } x_1 + a_2 \text{x } x_2 + \ldots + a_n \text{x } x_n \tag{3.8}$$

Fuzzy IF-THEN rules are widely used in modeling. They are considered the local description of the system being designed and form the basics of the fuzzy inference system (FIS).

Fuzzification**:** The aim of fuzzification is mapping the crisp input into a fuzzy set. This input can be from a set of sensors or features of those sensors like amplitude or frequency, and is mapped into fuzzy numbers of values from 0 to 1, using a set of input membership functions. The numeric inputs, $u_i \epsilon U_i$ are converted into fuzzy sets by the fuzzification process for the fuzzy system to use.

When U_i^* represents the set of all possible fuzzy sets which can be defined on U_i^* (given $u_i \epsilon U_i$), u_i is transformed to a fuzzy set denoted by A_i^{fuzz} that is defined on the universe of discourse U_i^*. The fuzzification operator F that produces this transformation is defined by

$$F: U_i \Rightarrow U_i^*$$

where

$$F(u_i) = A_i^{fuzz},$$

Frequently, “singleton fuzzification” is used. It produces a fuzzy set $A_i^{fuzz} \epsilon U_i^*$ with a membership function given by

$$\mu_{A_i^{fuzz}}(x) = \begin{cases} 1 & x = u_i \\ 0 & otherwise \end{cases}$$

Any fuzzy set with this form of membership function is termed “singleton”. Singleton fuzzification is the type for which the input fuzzy set has only a single point of nonzero membership and the number u_i is represented by the singleton fuzzy set. In implementations where singleton fuzzification is used, u_i only takes on its measured values without any noise involved. “Gaussian fuzzification” that uses bell type membership functions about input points, and triangular fuzzification using triangle shapes, are common examples [38].

3.3.3 Fuzzy inference mechanism

Designing a fuzzy inference system (FIS) from data can be separated into two principal stages: (1) automatic rule generation and (2) system optimization [30]. Rule generation is the guide to a fundamental system that has a given space partitioning and the set of rules that corresponds to it. System optimization is realized at different sorts of levels. Variable selection could be a comprehensive selection or is possibly handled rule by rule. The goal of rule base optimization is to choose the most efficient rules and to use rule conclusions in the best way. It’s possible to enhance space partitioning by adding or removing fuzzy sets and by tuning the parameters of membership function. Structure optimization has great significance: choosing variables, lessening the rule base and optimizing the number of fuzzy sets.

There are two main tasks associated with fuzzy inference mechanism:

1. Matching task which involves determining the degree of each rule’s being relevant to the current situation as marked by the inputs u_i, $i = 1,2, \dots, n$.
2. Inference step which involves reaching the conclusions from the current inputs u_i and the information in the rule-base.

When the fuzzy set representing the premise of the i^{th} rule is denoted by $A_1^j \times A_2^k \times \dots \times A_n^l$, there will be two steps for matching:

Step 1**:** Combining inputs with rule premises: This step is about finding fuzzy sets $\overline{A}_1^j$, $\overline{A}_2^k, \ldots, \overline{A}_n^l$, with membership functions.

$$\mu_{\overline{A}_1^j}(u_1) = \mu_{A_1^j}(u_1) * \mu_{\overline{A}_1^{fuzz}}(u_1)$$

$$\mu_{\overline{A}_2^k}(u_2) = \mu_{A_2^k}(u_2) * \mu_{\overline{A}_2^{fuzz}}(u_2)$$

.

.

$$\mu_{\overline{A}_n^j}(u_n) = \mu_{A_n^j}(u_n) * \mu_{\overline{A}_n^{fuzz}}(u_n)$$

(for all *j, k, … ,l*) combining the fuzzy sets from fuzzification with the fuzzy sets used in each of the terms in the rules' premises. When the singleton fuzzification is used, each of the input fuzzy sets has only a single point of nonzero membership function.

(e.g. $\mu_{\overline{A}_n^j}(\overline{u}_n) = \mu_{A_n^j}(u_n)$ for $\overline{u}_1 = u_1$ and $\mu_{\overline{A}_n^j}(\overline{u}_n) = 0$ for $\overline{u}_1 \neq u_1$)

To put it in another way, $\mu_{\overline{A}_n^{fuzz}}(u_i) = 1$, with singleton fuzzification, for all $i = 1,2, \ldots, n$ for the given u_i inputs resulting in

$$\mu_{\overline{A}_1^j}(u_1) = \mu_{A_1^j}(u_1)$$

$$\mu_{\overline{A}_2^k}(u_2) = \mu_{A_2^k}(u_2)$$

.

$$\mu_{\overline{A}_n^j}(\overline{u}_n) = \mu_{A_n^j}(u_n)$$

Step 2**:** Determining those rules that are on: In this step, membership values $\mu_i(u_1, u_2, \ldots, u_n)$ are determined for the premise of i^{th} rule which represents the certainty that each rule premise is consistent with the given inputs. Defining

$$\mu_i(u_1, u_2, \ldots, u_n) = \mu_{\overline{A}_i^j}(u_1)\mu_{\overline{A}_2^k}(u_2) \ldots \mu_{\overline{A}_n^l}$$

that is a function of the inputs u_i, $\mu_i(u_1, u_2, \ldots, u_n)$ represents the certainty that the antecedent of rule i matches the information in the case of singleton fuzzification use. The $\mu_i(u_1, u_2, \ldots, u_n)$ is a multidimensional certainty surface. It stands for the certainty of a premise of a rule and for the level to which a particular rule is consistent for a given set of inputs. The implied fuzzy set is determined by the inference step which is then taken by calculating the "implied fuzzy set" B_q, for the i^{th} rule, with the membership function

$$\mu_{\overline{B}_q^i}(y_q) = \mu_l(u_1, u_2, \ldots, u_n) * \mu_{B_q^p}(y_q) \tag{3.9}$$

The certainty level of the output's being a specific crisp output y_q within the universe of discourse y^q is specified by the implied fuzzy set $\overline{B}_q^l$ on considering simply rule I. The defuzzification that comes after the inference step is employed to aggregate the conclusions of all the rules which the implied fuzzy sets represent.

Defuzzification Methods**:** It is frequently important to find out a single crisp output from a FIS. For instance, in the case of one attempting to classify a letter drawn by hand on a drawing tablet, the FIS would be obliged to find out a crisp number to determine the letter that was drawn. A process called defuzzification is used to attain this crisp number. In other words, defuzzification means the way of extracting a crisp value from a fuzzy set as a representative value.

Two known methods can be used for defuzzifying:

Center of Gravity (COG): The method picks the output distribution and works to find its center of mass to produce one crisp number.

$$u = \frac{\sum_i \mu(x_i) x_i}{\sum_i \mu(x_i)} \tag{3.10}$$

where the crisp output value u is the abscissa (center of mass) under the center of gravity of the fuzzy set, $\mu(x_i)$ is the membership value in the membership function, x_i is a running point

in a discrete universe. This expression is also considered the weighted average of the elements in the support set.

The COG method for singletons attains the following expression

$$u = \frac{\sum_i \mu(s_i) s_i}{\sum_i \mu(s_i)} \tag{3.11}$$

where s_i is the position of singleton i in the universe and $\mu(s_i)$ represents the rule strength α_i of rule i. This technique has a good computational complexity and u is differentiable with respect to the singletons s_i, that is practical in neuro-fuzzy systems.

Center of Average (COA): In this widely used method, a crisp output y_q^{Crisp} is selected employing the centers of every one of the output membership functions and the highest certainty of every one of the conclusions the implied fuzzy sets represent, and is described as

$$y_q^{Crisp} = \frac{\sum_{i=1}^{R} b_i^q \sup_{yq}\{\mu B_q^i(y_q)\}}{\sum_{i=1}^{R} \sup_{yq}\{\mu B_q^i(y_q)\}} \tag{3.12}$$

here “sup” is the “supermum” (i.e., the least upper bound that is frequently regarded as maximum value). Therefore, $sup_x\{\mu(x)\}$ can simply be considered the highest value of $\mu(x)$.

Fig. 3.3 outlines the inference mechanisms on different types of fuzzy systems graphically. Most fuzzy inference systems can be categorized into three types depending on the types of fuzzy reasoning.

In Type 1 fuzzy systems, the defuzzifier puts together the output sets that correspond to the whole of the fired rules in a way to attain a single output set and afterwards comes up with a crisp number which represents this output set that is put together, e.g., the centroid defuzzifier comes up with the unity of the whole of the output sets and utilizes the centroid of the unity as the crisp output [31]. The weighted average of each rule’s crisp output introduced by rule’s weight and the output membership functions is the overall output.

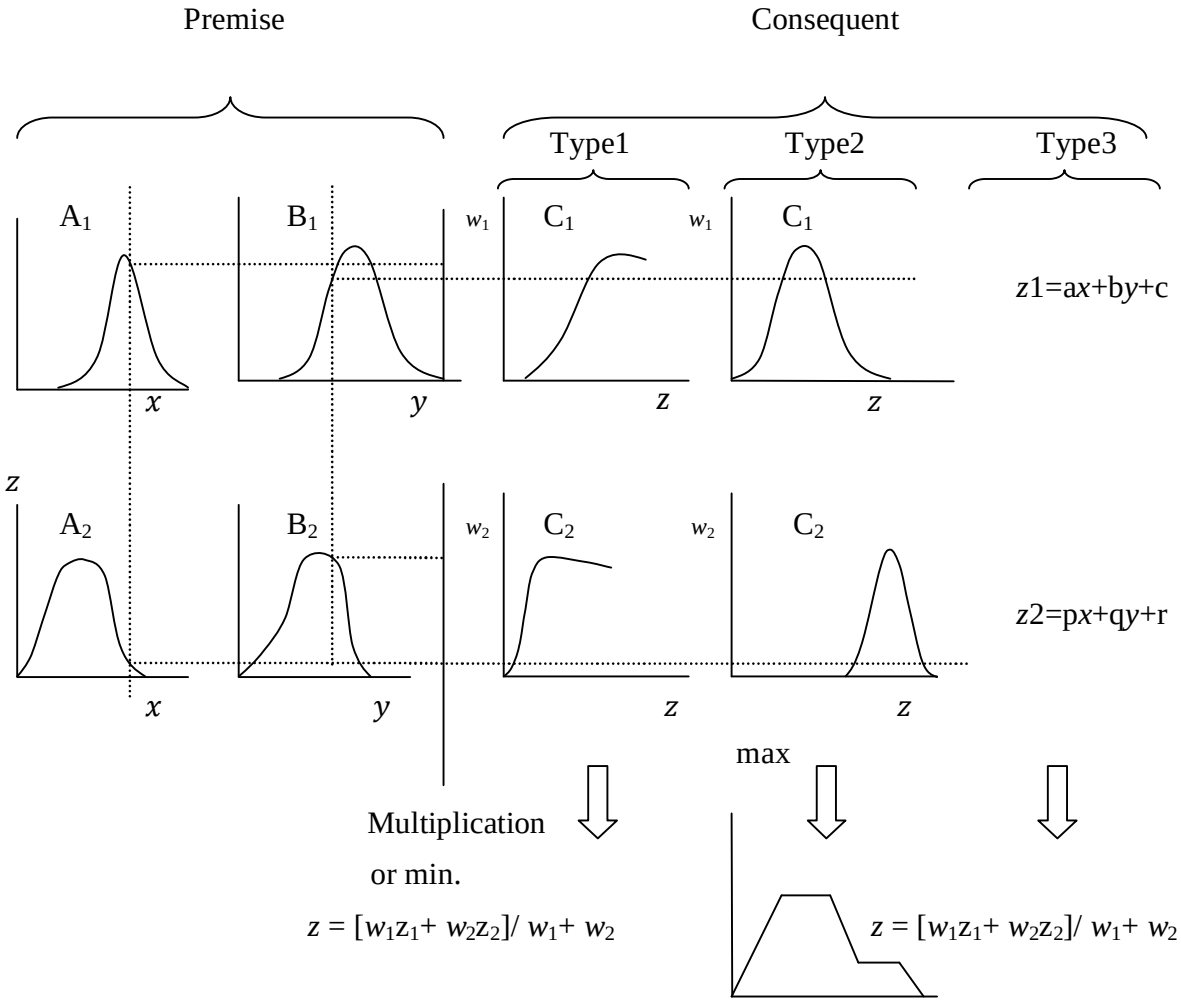


Figure 3.3 Types of fuzzy reasoning mechanisms [11]

In Type 2 fuzzy systems, fuzzy sets are quite helpful in conditions that make the determination of an exact membership function for a fuzzy set hard; for this reason, they are quite helpful in the incorporation of uncertainties. These uncertainties are caused by the knowledge employed in the construction of rules in a fuzzy logic system and lead to rules that have uncertain antecedents and/or consequents that are transformed in succession into uncertain antecedent and/or consequent membership functions [31]. The overall fuzzy output is attained after the application of 'max' operation to the fuzzy outputs that qualify. Every one of these outputs equals the minimum rule strength and each rule’s membership function.

Type 3 is Takagi and Sugeno's fuzzy IF-THEN rules. The output is a crisp number computed by the multiplication of every one of the inputs by a constant and summing the result afterwards. The weighted average of each rule's output is the output.

In Fig. 3.3, a fuzzy inference system with two rules and two inputs is used to demonstrate the different types of fuzzy rules and fuzzy reasoning described above.

3.4 Artificial Neural Networks

Recognizing that computing in the human brain takes place in a totally different manner from the traditional digital computer, has been the incentive for research into artificial neural networks, also known as "neural networks". The brain is extremely complex, nonlinear and parallel computing (information-processing system). It is capable of organizing its structural constituents, called *neurons*, in order to carry out some necessary computations (e.g. pattern recognition, perception and motor control) a lot more quickly than the highest speed digital computer of the present time [29].

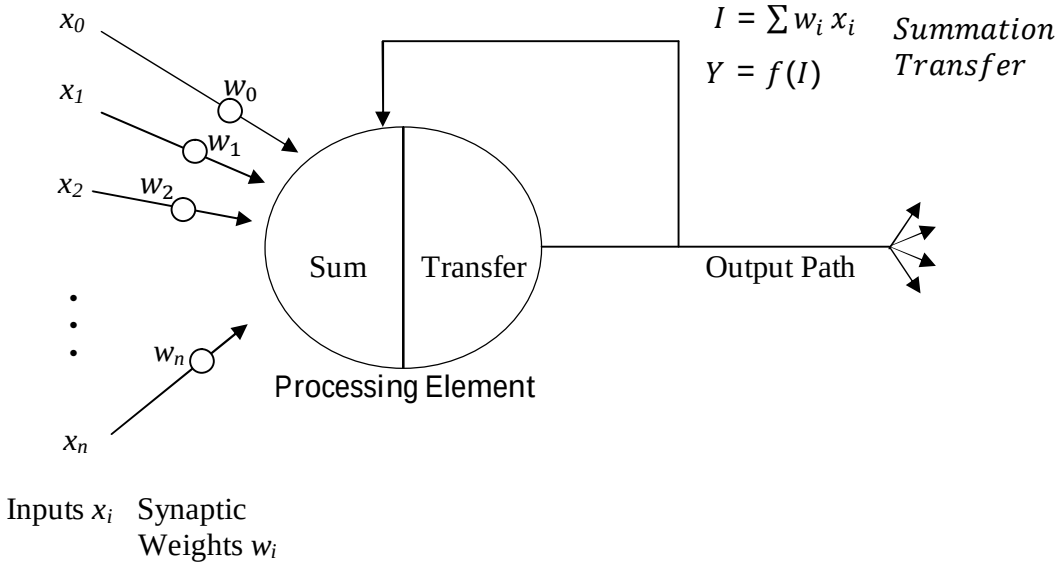


Figure 3.4 Artificial neuron

A *neuron* is a unit that processes information and is significant in a neural network's operation. The model of an artificial neuron that is fundamental in the design of artificial neural networks is demonstrated in the block diagram of Fig. 3.4.

A set of synapses, also called connecting links, are the foundation elements of the neuronal model. A weight or strength of its own characterizes each of these synapses. Specifically, a signal x_i for $i = 1,2,\dots,n$, at the input of synapse, connected to neuron k, is multiplied by the synaptic connection weight w_i, for $i = 1,2,\dots,n$. A result is generated by summing these products, feeding them through a transfer function and then outputting them.

The output of the artificial neuron displayed in Fig. 3.4 is calculated from

$$y_i = f(\sum_{j=1}^{n} w_{ij} x_j - \theta_i) \tag{3.13}$$

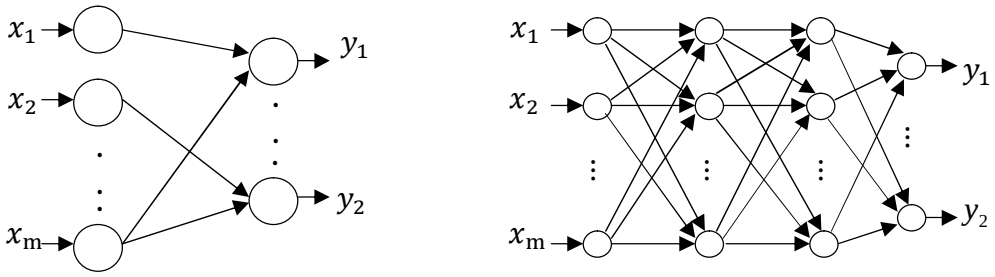
where x_i is the input, y_j is the output signal of the neuron, w_{ij} are the synaptic weight coefficients, θ_i denotes the bias and f is the activation function.

The activation function can be linear or nonlinear but a nonlinear sigmoid function is frequently utilized as the activation function (eq. 3.14).

$$y_j = \frac{1}{1+\exp[-(\sum_{j=1}^{n} w_{ij} x_i - \theta_j)]} \tag{3.14}$$

Neural networks are formed by a set of neurons in layer(s). The neurons are interconnected by weighted connections at certain connection points which are called *nodes*. The way of organization in a layered neural network is the layer formation. The least complicated formation of a network with layers uses an input layer of source nodes which projects onto an output layer of neurons (computation nodes) but not the other way round. This network is a *feedforward* or *acyclic* type of network. Neurons in the network act as processing elements which multiply an input by a set of weights and nonlinearly transform the result into an output value.

On the whole, three basically different architectural network classes can be defined which are single-layer feedforward (non-recurrent), multilayer feedforward and recurrent networks. The feedforward neural network structures are shown in Fig. 3.5.



(a) (b)

Figure 3.5 (a) A single layer network, (b) A simple multilayer network [11]

3.4.1 Neural network's learning

The most important specialty of a neural network is its capability of *learning* from its environment and *improving* its performance through learning. An interactive adjustment process applied to its synaptic weights and bias levels enables a neural network to learn about its environment. Every one of the iterations of the learning process makes the network well-informed of its environment. Learning in the circumstances of neural networks can be clearly stated to be a process by which the neural network's free parameters are adapted through a stimulation process by the environment where the network is embedded. The way that the parameter changes occur determines the type of learning [29].

A set of rules that are well determined and defined for the solution of a learning process is referred to as a learning algorithm. No learning algorithm that is the only one of its sort exists in the neural network design, as expected. The manner that the adjustment to a neuron's synaptic weight is clearly and exactly expressed, fundamentally cause the learning algorithms to differ from each other. Another factor that should be taken into consideration is the way

that a neural network (learning machine) which is comprised of a set of interconnected neurons, is related to its environment. In this latter context, a term is spoken as a learning paradigm that refers to a model of the environment in which the neural network operates [29]. There are two fundamental learning paradigms associated with neural networks: (1) Learning with a teacher (known as supervised learning) and (2) Learning without a teacher which is divided into two subdivisions that are unsupervised learning and reinforcement learning.

Supervised learning involves training with a teacher. The teacher can be thought of as a set of input-output examples representing the knowledge of the environment. Neural network, on the other hand, does not know the environment. Considering that a training vector drawn from the environment is applied to both the teacher and the neural network, the teacher is capable of supplying the neural network with a desired response for the training vector. The network parameters, i.e. the connection weights, are adjusted under the combined influence of the training vector and the error signal. The error signal is what makes the desired response differ from the actual response of the network. This adjustment is brought about in an iterative and step-by-step way aiming at eventually causing the neural network to *emulate* the teacher; this emulation is supposedly optimum in a statistical sense. This manner transfers the environment's knowledge that can be obtained by the teacher, to the neural network through learning as fully as possible. On reaching this condition, the teacher may be removed and the neural network copes with the environment entirely on its own.

The form of supervised learning just described, is the error correction learning which involves a closed-loop feedback system but the loop does not contain the unknown environment. The mean-square error or the sum of squared errors over the training samples that are in terms of the free parameters of the system constitutes the performance criterion for the system. This criterion may be visualized as a multidimensional error performance or simply error surface, with the parameters as coordinates. The true error surface is averaged over all possible input-output examples. It's a point on the error surface which represents any one of the system's operations that the teacher supervises. The operating point has to move down one after

another toward a minimum point of the error surface so that the system improves performance over time and thus learns from the teacher; it's possible for the minimum point to be a local minimum or a global minimum. A supervised learning system is capable of doing this using the helpful information it has about the gradient of the error surface that corresponds to the system's current behavior. The gradient of an error surface at any point is a vector which points in the direction of *steepest descent*. On providing an algorithm designed to minimize the cost function, a sufficient set of input-output examples and sufficient time allowed to carry out the training, a supervised learning system is generally capable of performing tasks like pattern classification and function approximation [29].

3.4.2 Multilayer perceptrons & backpropagation algorithm

Multilayer feedforward networks form a significant classification of neural networks. The network is characteristically comprised of a set of sensory units (source nodes) which establish the input layer, one or more hidden layers of computation nodes, and an output layer of computation nodes. The input signal propagates through the network in a forward direction, on a layer-by-layer basis. These neural networks are called *multilayer perceptrons* (MLP) that represent a generalization of a single-layer perceptron.

A widely used algorithm which is named the error back-propagation algorithm, trains the multilayer perceptrons in applications in order to successfully solve some challenging and diverse problems. Error correction learning rule forms the basis for this algorithm. It may be considered a generalization of an equally popular adaptive filtering algorithm: the least mean square (LMS) algorithm for the special case of a single layer neuron [29].

Error back-propagation learning is comprised of two passes through the different layers of the network which are a forward pass and a backward pass. The forward pass contains an activity pattern (input vector) whose effect propagates through the network one layer after another and is applied to the network's sensory nodes. Consequently, an output set is created as the real network response. In the duration of the forward pass, all of the synaptic weights of the

network are unchanging. In the duration of the backward pass, all the synaptic weights are adjusted according to an error correction rule. Particularly, the real network response is taken out of a desired (target) response to come up with an error signal. The error signal is propagated back through the network, in contrast to the direction of synaptic connections, thus the naming “error back-propagation”. The synaptic weights are adjusted such that the real network response moves nearer to the desired response statistically. The error back-propagation algorithm is known in the literature as the back-propagation algorithm, or simply, back-prop, as well. The learning process carried out with the algorithm is referred to as the back-propagation learning.

There are three distinguishing characteristics of a multilayer perceptron:

1. There is a nonlinear function involved in the model of each neuron. The nonlinearity mentioned here is smooth, in other words, differentiable everywhere. A generally employed nonlinearity form which is sufficient for this requirement is a sigmoidal nonlinearity that the following logistic function defines:

$$y_j = \frac{1}{1+\exp(-v_j)} \tag{3.15}$$

   where v_j is the induced local field (i.e. the weighted sum of all synaptic inputs plus the bias) of neuron j, and y_j is the output of the neuron.

2. One or more layers of hidden neurons which do not belong to the input or output of the network can be found in the network. The network is capable of learning complex duties by extracting increasingly significant specialties from the input patterns (vectors) due to these hidden neurons.

3. The network performs a high connectivity degree which is decided by the network synapses. A change in the network’s connectivity obligates a change in the population of synaptic connections or their weights.

The multilayer perceptron derives its computational power when these characteristics are combined with the capability of learning from experience through training. The back-propagation algorithm has great significance in neural networks since it supplies a computationally efficient method in order to train multilayer perceptrons.

Fig. 3.6 demonstrates the architectural graph of a multilayer perceptron with one hidden layer and an output layer. The illustrated network is fully connected meaning that a neuron in one layer of the network is connected to all the nodes/neurons in the previous layer. Signal flow through the network progresses in a forward direction, from left to right and on a layer-by-layer basis. The value of each neuron is computed by first summing the weighted sums and the bias and then applying f(sum) (the sigmoid function) to calculate the neuron's activation.

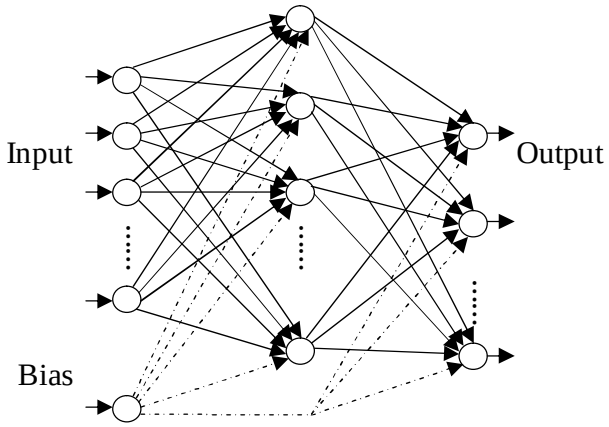


Figure 3.6 Multilayer feedforward network [11]

Next, the training processes of the three layer feedforward network will be analyzed. Firstly, three stages describing the feedforward phase in the network are: input (*I*), hidden (*H*) and output (*O*) layers.

- Input Layer (*I*): The input of the hidden layer is equal to the output of the input layer.

$$Output_I = Input_H$$

- Hidden Layer (*H)*: Summing the multiplication of all the input layer outputs by the corresponding weights connecting the two layers will be equal to the input of the hidden layer.

$$Input_H = \sum_i weight_{IHi} * output_I$$

The output of the hidden layer equals the result of the sigmoid transfer function of the hidden layer input.

$$Output_H = \frac{1}{1+e^{-Input_H}}$$

- Output Layer (*O*): Summing the multiplication of all the hidden layer outputs by the corresponding weights connecting the two layers will be equal to the inputs of the output layer.

$$Input_O = \sum_j weight_{HOj} * output_H$$

The output of the output layer equals the result of the sigmoid transfer function of the output layer input.

$$Output_O = \frac{1}{1+e^{-Input_O}}$$

At the end of the feedforward phase, the network error ($Error_O$) is found when the output of the output layer ($Output_O$) is compared with the target value of the neural network (Target).

$$Error_O = Target - Output_O$$

The aim of the back-propagation training is to minimize the error of all training patterns by adjusting the parameter weight values that involves updating the new value of the hidden-output layer weight in accordance with the following equation:

$$Nweight_{HO} = Oweight_{HO} + \eta * (error_O * Output_O * (1 - Output_O)) * Output_H$$

in which $Nweight_{HO}$ is the new hidden-output layer weight and $Oweight_{HO}$ is the old hidden-output layer weight, η is the rate of learning. The following equation updates the new weight of the hidden-input layer weight.

$$Nweight_{IH} = Oweight_{IH} + \eta * (error_H * Output_H * (1 - Output_H)) * Output_I$$

in which $Nweight_{IH}$ is the new hidden-input layer weight and $Oweight_{IH}$ is the old hidden-input layer weight.

The back-propagation training algorithm can be summarized as:

1. Performing the forward-propagation phase for an input pattern and calculating the output error.
2. Changing all weight values of each weight matrix by using the following formula:

$$Weight(old) + learning\ rate * output\ error * output(neuron\ i) * (1 - output(neuron\ i)) * output(neuron\ i - 1)$$

3. Going to step 1.
4. Ending the algorithm if all output patterns match their target pattern.

3.5 Neuro-Fuzzy Network Models

There are a lot of studies going on in the field of combining fuzzy systems and neural networks which are aimed at optimizing fuzzy systems. Fuzzy systems are capable of data approximation, handling rule uncertainties as well as performing high in the presence of channel nonlinearities. Beside associative storage and optimization, neural networks, on the other hand, have high learning capability and numerical accuracy.

It's obvious that instead of seeking solutions based on separate fuzzy logic or neural networks, it would be more useful to construct structural connectionist models or hybrid

systems which integrate them together. On doing so, the desirable features of robustness, uniformity and adaptivity inherent to neural networks can be combined with the inference, universality and representation features of fuzzy logic. Sequential hybrid systems and incorporated hybrid systems are the mostly used hybrid systems. A neuro-fuzzy system is an incorporated hybrid system that implements fuzzy inference system by using neural networks.

Neural networks are utilized to make changes in the linguistic terms' membership functions or to generate the linguistic rules themselves. Neuro-fuzzy networks are totally fuzzified multilayer feedforward networks. Fuzzy arithmetic is employed for the computation of the network output, therefore, addition, multiplication and the sigmoid function are fuzzified for this purpose [32].

In this thesis, a hybrid neuro-fuzzy system which implements a fuzzy inference system in neural network structure including nonlinear function for channel equalization is presented.

3.5.1 Nonlinear neuro-fuzzy network

3.5.1.1 Structure of the nonlinear neuro-fuzzy network

A fuzzy inference system (FIS) is designed from either expert knowledge or data. This study is concentrated on FIS that is based on automatic learning from data in which supervised learning is utilized and observed outputs belong to the training data. This type of FIS is constructed first by incorporating human expertise, which is also referred to as domain knowledge or linguistic data, about the target system. Secondly, fuzzy modeling is developed by interpreting the input-output data, also called the numerical data, of the target system. Takagi-Sugeno-Kang (TSK) type or Mamdani type fuzzy reasoning mechanisms are mostly implemented by neuro-fuzzy system structures. The purpose of the reasoning mechanisms is to be capable of expressing and solving a broad range of problems and problem types. Additionally, the mechanism has to be capable of determining those operations to be applied to a specific problem, when a problem's solution has been attained or when further work on the problem should be ended [29]. An adaptive neuro-fuzzy inference system (ANFIS) is a generic model of neuro-fuzzy network which is used in the implementation of TSK type fuzzy

system. ANFIS involves consequent parts which include linear functions enabling the TSK type neuro-fuzzy system to define the tackled problem using the combinations of linear functions. These systems may require extra rules in the course of modeling complex nonlinear processes for attaining the requested accuracy which eventually causes the quantity of neurons in the hidden layer to be increased.

In order to enhance the calculational power of the neuro-fuzzy system, the consequent section of every one of the rules employ nonlinear functions. The structure of the nonlinear neuro-fuzzy equalizer (NNFE) is based on these rules. NNFE network has enhanced computational power and is capable of describing nonlinear processes by using these nonlinear functions. In this thesis, NNFE is developed to successfully equalize both linear and nonlinear channel distortions, meanwhile, NNFE network is proven to yield better convergence rate and better BER results.

The fuzzy rules which have IF-THEN form and are composed by using nonlinear quadratic functions have been employed. These rules are in the following form:

$$\text{If } x_1 \text{ is } A_{j1} \text{ and } x_2 \text{ is } A_{j2} \text{ and } \ldots \text{ and } x_m \text{ is } A_{jm} \text{ then}$$

$$y_j = \sum_{i=1}^{m} (w1_{ij}\, x_i^2 + w2_{ij}\, x_i) + b_j \tag{3.16}$$

where $x_1, x_2, \ldots, x_m$ denote input variables, y_j $(j = 1, \ldots, n)$ denote output variables that are nonlinear quadratic functions, A_{ji} is a membership function for i-th rule of the j-th input defined as a Gaussian membership function. $w1_{ij}, w2_{ij}$ and b_j $(i = 1, \ldots, m,\ j = 1, \ldots, n)$ are network parameters.

The fuzzy model which is defined by IF-THEN rules could be attained after altering the conclusion parameters and antecedent section of the rules. This research work utilizes a gradient-descent method for the training of the parameters of the rules in the structure of the nonlinear neuro-fuzzy network. The nonlinear neuro-fuzzy network (NNFN) structure as demonstrated in Fig. 3.7 is proposed by using the fuzzy rules in equation 3.16.

The first layer of Fig. 3.7 demonstrates the nodes whose number is the equivalent of the number of input signals. The nodes are employed to distribute input signals.

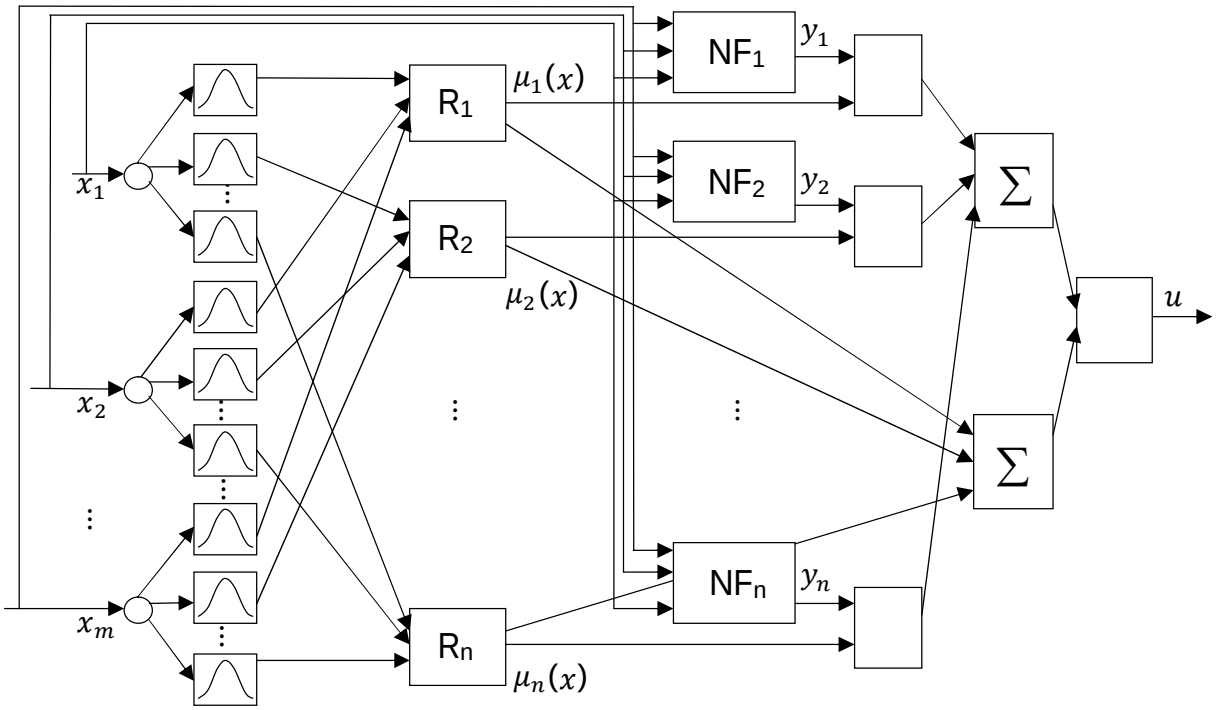


Figure 3.7 The NNFN architecture [39]

In the second layer, each node corresponds to one linguistic term. The membership degree, to which the input value belongs to in a fuzzy set, is calculated for each input signal that enters to the system. The following Gaussian membership function is used to describe the linguistic terms.

$$\mu 1_j(x_i) = e^{-\frac{(x_i - c_{ij})^2}{\sigma_{ij}^2}}, \; i = 1, \ldots, m, \quad j = 1,\ldots,J \tag{3.17}$$

where, m is the number of input signals, J is the number of linguistic terms that are assigned for external input signals x_i, c_{ij} and σ_{ij} are center and width of the Gaussian membership functions of the j-th term of i-th input variable, respectively. $\mu 1_j(x_i)$ denotes the membership function of i-th input variable for the j-th term.

In the third layer, the number of nodes is the corresponding of the number of the rules (R_1, R_2,….,R_n). One fuzzy rule is represented by each rule. The AND (min) operation is utilized when calculating the values of output signals. In formula (3.18), $\prod$ is the min operation.

$$\mu_l(x) = \prod_j \mu 1_j(x_i), \quad l = 1, \ldots, n, \quad j = 1, \ldots, J \tag{3.18}$$

The fourth layer represents the consequent layer which contains Nonlinear Functions (NF) that are denoted by NF_1, NF_2,…, NF_n. The following equation is employed to calculate the output of each nonlinear function in Fig.3.7.

$$y_j = \sum_{i=1}^{m} (w1_{ij} x_i^2 + w2_{ij} x_i) + b_j, \quad j = 1, \ldots, n \tag{3.19}$$

In the fifth layer, the output signals of third layer $\mu_l(x)$ are multiplied by the output signals of the nonlinear functions.

In the sixth and the seventh layers, the following output of the entire network is calculated by performing defuzzification.

$$u = \frac{\sum_{l=1}^{n} \mu_l(x) y_l}{\sum_{l=1}^{n} \mu_l(x)} \tag{3.20}$$

where y_l denote the outputs of the fourth layer which are nonlinear quadratic functions and u denotes the output of the entire network. The training process starts after calculating the output signal of the NNFN.

3.5.1.2 Learning of the nonlinear neuro-fuzzy network

The network parameter values which are c_{ij} and σ_{ij} $(i = 1, \dots, m,\ \ j = 1, \dots, n)$ in the second layer (antecedent part) and the parameter values of the nonlinear quadratic functions $w1_{ij}, w2_{ij}, b_j$ $(i = 1, \dots, m,\ \ j = 1, \dots, n)$ of the consequent part (fourth layer) are adjusted during the training process. The error value on the output of the network is calculated firstly.

$$E = \frac{1}{2}\sum_{i=1}^{O}(u_i^d - u_i)^2 \tag{3.21}$$

where O denotes the number of output signals of the network ($O = 1$ for the given case), u_i^d and u_i are the desired (target) and current output values of the network, respectively. The following formulas are used for adjusting the parameters $w1_{ij}, w2_{ij}, b_j$ $(i = 1, \dots, m,\ \ j = 1, \dots, n)$ and c_{ij} and σ_{ij} $(i = 1, \dots, m,\ \ j = 1, \dots, n)$:

$$w1_{ij}(t+1) = w1_{ij}(t) + \gamma\frac{\partial E}{\partial w1_{ij}} + \lambda(w1_{ij}(t) - w1_{ij}(t-1)) \tag{3.22}$$

$$w2_{ij}(t+1) = w2_{ij}(t) + \gamma\frac{\partial E}{\partial w2_{ij}} + \lambda(w2_{ij}(t) - w2_{ij}(t-1)) \tag{3.23}$$

$$b_j(t+1) = b_j(t) + \gamma\frac{\partial E}{\partial b_j} + \lambda(b_j(t) - b_j(t-1)) \tag{3.24}$$

$$c_{ij}(t+1) = c_{ij}(t) + \gamma\frac{\partial E}{\partial c_{ij}} \tag{3.25a}$$

$$\sigma_{ij}(t+1) = \sigma_{ij}(t) + \gamma\frac{\partial E}{\partial \sigma_{ij}} \tag{3.25b}$$

where γ is the learning rate, λ is the momentum, m is the number of input signals (input neurons) of the network and n denotes the number of rules (hidden neurons), $i = 1, \dots, m,\ \ j = 1, \dots, n$.

The derivative values involved in formulas (3.22-3.24) are calculated as follows:

$$\frac{\partial E}{\partial w1_{ij}} = (u(t) - u^d(t)) \cdot \frac{\mu_l}{\sum_{l=1}^{n} \mu_l} \cdot x_i^2$$

$$\frac{\partial E}{\partial w2_{ij}} = (u(t) - u^d(t)) \cdot \frac{\mu_l}{\sum_{l=1}^{n} \mu_l} \cdot x_i^2 \qquad (3.26)$$

$$\frac{\partial E}{\partial b_j} = (u(t) - u^d(t)) \cdot \frac{\mu_l}{\sum_{l=1}^{n} \mu_l}$$

The derivatives in Eq.(3.25) are calculated by the following formulas:

$$\frac{\partial E}{\partial c_{ij}} = \sum_j \frac{\partial E}{\partial u} \cdot \frac{\partial u}{\partial \mu_l} \cdot \frac{\partial \mu_l}{\partial c_{ij}} \quad , \qquad \frac{\partial E}{\partial \sigma_{ij}} = \sum_j \frac{\partial E}{\partial u} \cdot \frac{\partial u}{\partial \mu_l} \cdot \frac{\partial \mu_l}{\partial \sigma_{ij}} \qquad (3.27)$$

where

$$\frac{\partial E}{\partial u} = (u(t) - u^d(t)) \ , \quad \frac{\partial u}{\partial \mu_l} = \frac{y_l - u}{\sum_{l=1}^{L} \mu_l} \ , \quad i = 1, \dots, m, \ j = 1, \dots, n, \ l = 1, \dots n \qquad (3.28)$$

$$\frac{\partial \mu_l(x_j)}{\partial c_{ji}} = \begin{cases} \mu_l(x_j) \frac{2(x_j - c_{ji})}{\sigma_{ji}^2} & \text{if } j \text{ node is connected to rule node } l \\ 0, & \text{otherwise} \end{cases} \qquad (3.29)$$

$$\frac{\partial \mu_l(x_j)}{\partial \sigma_{ji}} = \begin{cases} \mu_l(x_j) \frac{2(x_j - c_{ji})^2}{\sigma_{ji}^3} & \text{if } j \text{ node is connected to rule node } l \\ 0, & \text{otherwise} \end{cases} \qquad (3.30)$$

Substituting the values found in formulas (3.28) through (3.30) into (3.27) and the values in (3.27) into (3.25), the learning of the NNFN parameters is brought about.

3.6 Summary

Neural networks and fuzzy logic are two effective systems, where neural networks have high learning capability, numerical accuracy in addition to robustness and adaptivity properties and fuzzy logic offers data approximation capability and high performance in the presence of channel nonlinearities and rule uncertainties. These desirable features of neural network and fuzzy logic are combined into neuro-fuzzy systems that yield fast and accurate results, by building connectionist hybrid systems. Fuzzy systems are designed from either expert knowledge or data, whereas the fuzzy inference system (FIS) is based on automatic learning from data through fuzzy rule-based fuzzy sets. The linguistic data about the target system and numerical input-output data are in FIS. Supervised learning which involves training with a set of input-output examples, is the model used in developing the NNFN which constitutes the basis of the equalizer in this study. This chapter outlines the architecture and operation principles of the nonlinear neuro-fuzzy system by describing the back-propagation learning algorithm which is used as part of the multilayer feedforward neural network employed to perform the adaptive channel equalization of quadrature amplitude modulation (QAM) signals.

CHAPTER 4

QUADRATURE AMPLITUDE MODULATION (QAM) APPLIED TO NON-LINEAR NEURO-FUZZY EQUALIZER (NNFE)

4.1 Analysis of QAM

Quadrature carrier signaling, as shown in Fig.4.1, is named *Quadrature Amplitude Modulation* (QAM). Generally, there is no restriction of having permitted signaling points only on a circle for QAM signal constellations. The general QAM signal is expressed as

$$s_m(n) = \mathrm{Re}\left[(A_{mc} + jA_{ms})g(n)e^{j2\pi f_c n}\right], \quad m = 1,2,\dots,M, \quad 0 \le n \le T \tag{4.1}$$

where A_{mc} and A_{ms} are the quadrature carrier signal amplitudes that bear information and $g(n)$ is the signal pulse.

The QAM signal waveforms may also be expressed as

$$\begin{aligned} s_m(n) &= \mathrm{Re}\left[V_m e^{j\theta_m} g(n) e^{j2\pi f_c n}\right] \\ &= V_m g(n)\cos(2\pi f_c n + \theta_m) \end{aligned} \tag{4.2}$$

where $V_m = \sqrt{A_{mc}^2 + A_{ms}^2}$ and $\theta_m = \tan^{-1}(A_{ms}/A_{mc})$. It is apparent, from this expression, that waveforms of QAM signal is also considered the combination of amplitude and phase modulation [1]. In this thesis, only the amplitude modulation is considered.

It's also possible that QAM signal waveforms are represented by a linear combination of two orthonormal signal waveforms, $f_1(n)$ and $f_2(n)$, i.e.,

$$s_m(n) = s_{m1} f_1(n) + s_{m2} f_2(n) \tag{4.3}$$

where

$$f_1(n) = \sqrt{\frac{2}{\xi_g}}\, g(n)\cos 2\pi f_c n \tag{4.4}$$

$$f_2(n) = -\sqrt{\frac{2}{\xi_g}}\, g(n)\sin 2\pi f_c n$$

and

$$\mathbf{s}_m = [s_{m1} + s_{m2}]$$

$$= [A_{mc}\sqrt{\tfrac{1}{2}\xi_g}\;\; A_{ms}\sqrt{\tfrac{1}{2}\xi_g}] \tag{4.5}$$

ξ_g is the energy of the signal pulse $g(n)$. The Euclidean distance between any pair of signal vectors, which is the determining factor of the probability of error, is

$$d_{mn}^{(e)} = |\mathbf{s}_m - \mathbf{s}_n|$$

$$= \sqrt{\tfrac{1}{2}\xi_g[(A_{mc} - A_{nc})^2 + (A_{ms} - A_{ns})^2]} \tag{4.6}$$

In the special case where the signal amplitudes take the set of discrete values $\{(2m - 1 - M)d,\ m = 1,2,\ldots,M\}$, the signal constellation diagram is rectangular and M-symbol (M = 4, 16 and 64 levels) QAM constellation is illustrated in Fig.4.1. If that happens, the Euclidean distance between adjacent points, i.e. the minimum distance, is

$$d_{min}^{(e)} = d\sqrt{2\xi_g} \tag{4.7}$$

Quadrature amplitude modulation which is also called quadrature carrier signaling is illustrated in Fig. 4.1. In addition to equations (4.1) and (4.2), the following form which involves a hybrid combination of amplitude and phase modulations, is the equivalent representation of the general QAM signal;

$$s(n) = x(n)\cos(\omega_c n) - y(n)\sin(\omega_c n) \tag{4.8}$$

where

$$g(n) = x(n) + jy(n) = |g(n)|e^{j\angle g(n)} \equiv R(n)e^{j\theta(n)} \tag{4.9}$$

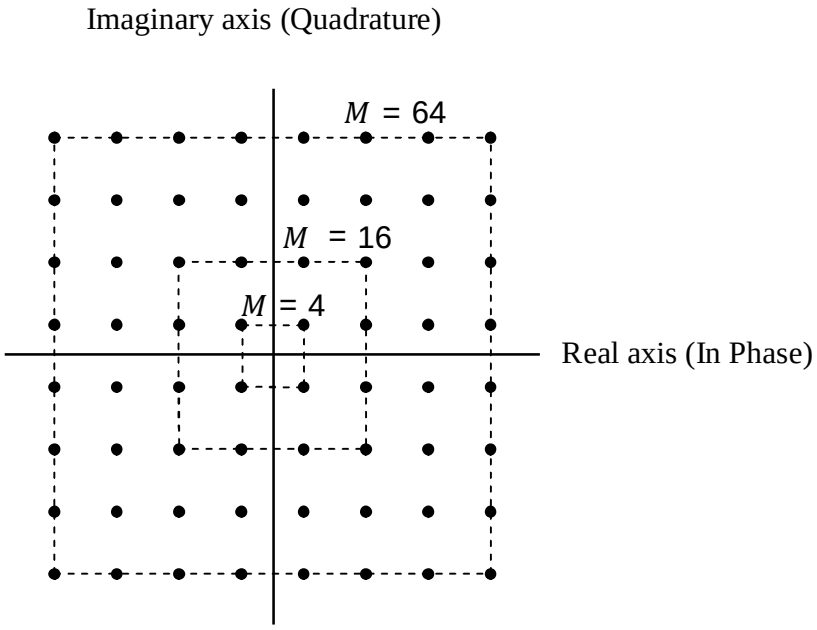


Figure 4.1 *M*-symbol QAM constellation ($\sqrt{M}$ amplitude levels per dimension) [1]

and $x(n) = Re\{g(n)\}$ & $y(n) = Im\{g(n)\}$. $x(n)$ is referred to as the *in-phase* modulation related to $s(n)$ and $y(n)$ is referred to as the *quadrature* modulation related to $s(n)$. $g(n)$ is called the complex envelope of $s(n)$ and f_c is the associated carrier frequency (in hertz) where $\omega_c = 2\pi f_c$.

When a complex baseband model of the channel is under consideration, beside the above defined transmitted data sequence $g(n)$, the channel impulse response h_n and the received signal $u(n)$ are complex valued and can be expressed as:

$$h_N = h_{I,n} + jh_{Q,n} \tag{4.10}$$

$$u(n) = u_I(n) + ju_Q(n) \tag{4.11}$$

where the subscripts I and Q refer to the in-phase (real) and quadrature (imaginary) components, respectively.

4.1.1 Significance of complex envelope and carrier frequency

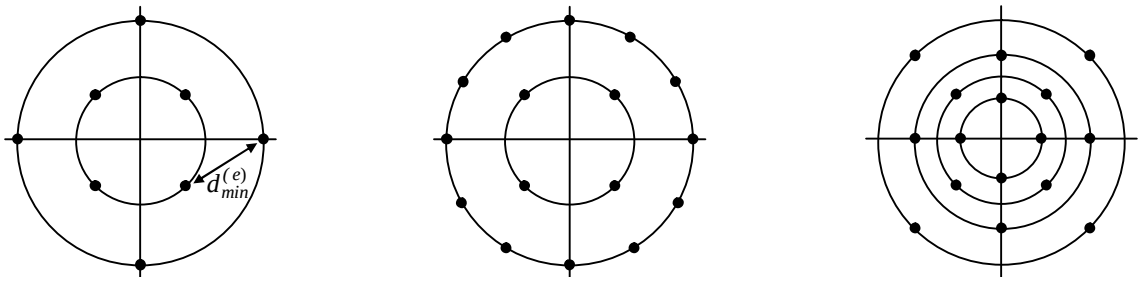
The carrier frequency f_c associated with the QAM signal $s(n)$ is a significant factor in communication problems. The *bandpass* waveform $s(n)$ has nonzero spectrum for frequencies in a band situated near $f = f_c$ where $f_c \gg 0$. The spectral magnitude (spectrum) can be neglected anywhere else. On the other hand, a *baseband* waveform (i.e. signal) contains a spectral magnitude which is nonzero for frequencies near the origin (i.e. $f = 0$) and can be neglected anywhere else. In communication problems, the information source signal is usually a baseband signal and transferring this signal to the desired destination in a communication system necessitates the use of a bandpass signal that has a bandpass spectrum concentrated at $\pm f_c$, and f_c is selected so that the bandpass signal $s(n)$ will propagate across a wire or a wireless communication channel. f_c is the frequency to which the spectrum of the baseband signal $g(n)$ is translated (shifted), by the $e^{j\omega_c n}$ factor in Eq. 4.1.

The baseband signal $g(n)$ is very significant because of its usefulness as the complex envelope representation for bandpass waveforms in modern communications systems. The complex envelope $g(n)$ is also employed as the modulating QAM baseband source in this thesis. It is the transmitted signal which is processed and equalized by the NNFE. *The complex envelope $g(n)$ operates as the baseband equivalent of the bandpass signal $s(n)$, instead of the bandpass signal itself.* When carrying out computer simulations of bandpass signals, it's possible to minimize the sampling rate of the simulation by using the complex envelope $g(n)$ rather than the bandpass signal $s(n)$.

4.1.2 Alternative implementations of QAM

Large numbers of possible QAM signal constellations that result in the best error performance of an average certain signal-to-noise ratio (SNR) have been examined by researchers. Fig. 4.2 illustrates some examples of symbol constellations that are also called signal space diagrams, representing alternative implementations of QAM signaling. The significance of a specific signal point constellation is in determining the probability of error and average power because the probability of error and average power are determined by the minimum distance between

pairs of signal points. Thus, the type of signal constellation is crucial since the probability of error and average power are the performance criteria of the system.



(a) (b) (c)

Figure 4.2 Three possible circular QAM signal constellations (a) 8-QAM (4,4), (b)16-QAM (4,12), (c) 16-QAM (4,4,4,4) [1]

In general, the constellation rule, known as the Campopiano-Glazer construction rule that yields optimum signal set performance can be summarized as follows: Select a closely packed subset of 2^l (where $l = \text{no. of bits}$) points as a signal constellation, from an infinite array of points closely packed in a regular array or lattice. In this case, "optimum" means minimum average power for a given probability. In a two-dimensional signal space, the optimum boundary surrounding an array of points tends toward a circle [24]. Therefore, better average power and error performance usually result from circular constellations. The signal constellation of Fig. 4.2(a) is recognized as the best eight-point QAM signal constellation since it requires the minimum average power for a given minimum distance $d_{\min}^{(e)}$ between signal points [1]. An additional example is that of 16-QAM in Fig. 4.2(b), where a better bit error rate (BER) performance can be gained with half the number of points in the inner circle, compared with another constellation with two amplitude rings and eight phase states on each ring, namely 16-QAM (8,8) that has eight points in the inner circle. The reason for better performance of 16-QAM (4,12) over 16-QAM (8,8) is that the constellation points are more evenly spaced over 12 distinct phases.

On the other hand, circular constellations do not always yield the best performance. The circular multiamplitude constellation for $M = 16$ as demonstrated in Fig. 4.2(c), is a generalization of the optimum 8-QAM constellation where the signal points at a given

amplitude level are phase-rotated by $\frac{1}{4}\pi$ relative to the signal points at adjacent amplitude levels. However, the circular 16-QAM constellation is not the best 16-point QAM signal constellation for the AWGN channel. This constitutes another cause for employing rectangular constellation in this research work.

4.2 Structure of Channel Equalization System

There are various types of distortions contaminating channels in digital communications. Linear and nonlinear equalizers are two main equalizer types used to clean up the distortions in the channels. Even though linear equalizers are widely and commonly used for the equalization of linearly distorted channels, they have been found to be ineffective on channels with time-varying characteristics and nonlinear distortion. On the other hand, nonlinear equalizers have proven better performance on channels that have severe nonlinear distortions caused by factors such as crosstalk and the nature of the channel itself. Nonlinear equalizers are also capable of eliminating linear distortions like intersymbol interference (ISI), in addition to nonlinear distortions.

Adaptive channel equalizers are used when the channel characteristics of the channel are not known in advance and most of the times the channel response is time-variant. These equalizers can adjust themselves to the channel response and for time-variant channels, can be adaptive to the time-variations in the channel response, due to their design. Adaptive equalizers are also capable of compensating for signal distortion coming from ISI that is a linear distortion which is the result of multipath inside time-dispersive channels [27,5].

A neural network is a parallel distributed processor built around a basic unit called a neuron. Neural networks' capability of approximating an unknown nonlinear input-output mapping by using supervised training motivates interest in neural networks. This property is particularly useful when the basis-forming physical mechanism that causes an input signal production is nonlinear in essence [25]. Neural networks can be feedforward, fully connected or partially connected type of networks. The logic forming the basis of a feedforward network is that input signals produce a network output response by propagating solely in the forward direction. In other words, no feedback exists in the network. A multilayer perceptron (MLP) is

one type of feedforward neural network which is discussed in this thesis and comprises an input layer of source nodes, computational nodes (neurons) containing one or more hidden layers and an output layer comprising computational nodes, as well. Another type of feedforward neural network is the radial basis function (RBF) network which has a single hidden layer, is more complex than MLP in terms of computational complexity but simpler in structure, and requires less time for training.

One widely used way for the development of adaptive equalizers for nonlinear channels is using fuzzy technology. The operation of fuzzy technology is based on processing numerical data and linguistic information in natural form. The fuzzy IF-THEN rules which are employed in the construction of the filter for the nonlinear channel are determined by human experts utilizing the channel's input-output information pairs. The speed of adaptation and the bit error rate (BER) are improved by the incorporation of linguistic and numerical information.

This thesis applies the structure of neuro-fuzzy network based equalization system implementing one of the two types of TSK or Mamdani fuzzy reasoning mechanisms. TSK stands for Takagi-Sugeno-Kang and fuzzy division of the input space forms the basis of this model's structure. Accordingly, a set of fuzzy membership functions $A_{k1}, A_{k2}, \dots A_{kJ}, (k = 1,2,3, \dots, K)$ that correspond to K number of fuzzy rules, are designed for J number of inputs [36]. This system is a nonlinear adaptive equalizer. An adaptive equalizer is a filter which is self-designing in that the adaptive filter depends for its operation on a recursive algorithm enabling the filter to provide satisfactory performance in an environment where total knowledge of the relevant signal characteristics can not be obtained. The starting point of the algorithm is an already found set of initial conditions representing all the information regarding the environment. Yet, within a motionless environment, successively iterating the algorithm makes it converge to the most favorable solution statistically. In an active environment, the algorithm presents a *tracking* ability in that it is capable of tracking time variations in the input data statistics, ensuring that variations are adequately slow [25].

Training and tracking are the common working manners of an adaptive equalizer. First, the transmitter sends a known, fixed-length training sequence to ensure that the receiver's equalizer adapts to a correct setting to detect the minimum BER. The training sequence is

characteristically a pseudorandom binary signal or an unchanging, prescribed bit pattern. After this training sequence, the user data (which may or may not include coding bits) is sent and the adaptive equalizer at the receiver uses a recursive algorithm for evaluating the channel and approximately calculating the filter coefficients to compensate for the distortion that the multipath in the channel creates. The training sequence is designed to allow an equalizer at the receiver to gain the appropriate filter coefficients in the worst possible channel conditions (e.g. fastest velocity, longest time delay spread, deepest fades, etc.) in order to ensure that, the filter coefficients are close to the optimal values for reception of user data when the training sequence is finished. The adaptive algorithm of the equalizer tracks the changing channel as user data are received [3]. Consequently, the adaptive equalizer frequently changes its filter characteristics over time. When an equalizer has been properly trained, it is said to have converged. The time span over which an equalizer converges is a function of the equalizer algorithm, the equalizer structure and the time rate of change of the multipath radio channel. Periodic retraining is required by equalizers in order to continue effective ISI cancellation and are commonly used in digital communication systems where user data is segmented into short time blocks or time slots. A time division multiple access (TDMA) wireless system is especially suitable for an equalizer. TDMA systems send data in fixed-length time blocks and the training sequence is usually sent at the beginning of a block. When a new data block is received, the equalizer is retrained using the same training sequence.

The adaptive equalizer processes three types of signals in order to determine the error signal which controls the adaptive algorithm. Comparison of the equalizer output with some signal $s_d(k)$ that is either a precise scaled replica of the transmitted signal $s(k)$ or is the representative of a known transmitted signal property is carried out to derive the error signal. The transmitted (input) signal $s(k)$ and the reference (desired) input signal $s_d(k)$ are required by the equalizer to determine the error signal. The desired signal is the synchronized version of the test signal that the receiver generates. The error signal is determined by first processing the input signal $s(k)$ to determine the equalizer output and then subtracting the desired signal $s_d(k)$ from this equalizer output. The adaptive algorithm uses this estimation error for the minimization of a *cost function* and for the updating of the equalizer weights to their optimum values in an iterative manner which decreases the cost function. For instance, the least mean

squares (LMS) algorithm seeks to find the optimum or almost optimum filter weights by applying the iterative operation:

$$\text{New weights} = \text{Previous weights} + (\text{constant}) \times (\text{Previous error}) \times (\text{Current input vector}) \quad (4.12a)$$

where

$$\text{Previous error} = \text{Previous desired output} - \text{Previous actual output} \quad (4.12b)$$

and the algorithm adjusts the constant for the controlling of the variation between filter weights on successive iterations. A programming loop repeats that series of operations quickly as the equalizer tries to *converge* and the error is minimized by numerous techniques like gradient or steepest descent algorithms. After reaching the convergence, the filter weights are frozen by the adaptive algorithm until an acceptable threshold is exceeded by the error signal or till another training sequence is started [3].

The channel equalization system structure is illustrated in Fig. 4.3 where $s(k)$ are binary transmitted input signals. The channel noise $n(k)$ is the additive Gaussian noise which distorts the input signals. The channel may be linear or nonlinear. In addition to noise, the intersymbol interference (ISI) that is caused by the spreading of symbol pulses into adjacent time slots of sampling signals, is the main reason for high SNR in such transmission systems.

The adaptive equalizer on the receiving side is applied to compensate for channel distortion by minimizing the effects of noise and ISI. Nonlinear neuro-fuzzy networks (NNFN), as part of this research work, can be successfully applied to adaptively equalize nonlinear communication systems. They are proven to reveal improved performance when little information about the channel is available.

4.3 Applications of QAM

Quadrature Amplitude Modulation (QAM) utilizes carrier phase shifting and synchronous detection to permit two double-sideband (DSB) signals to occupy the same frequency band.

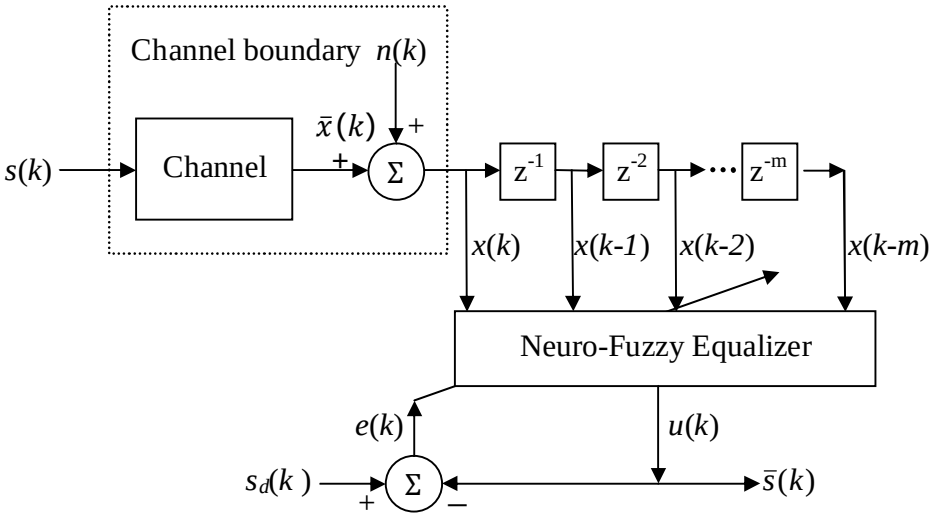


Figure 4.3 Structure of a neuro-fuzzy equalization system [11]

The transmitted signal has the following form:

$$x_c(n) = A_c[x_1(n)\cos(\omega_c n) \pm x_2(n)\sin(\omega_c n)] \tag{4.13}$$

This technique is more properly characterized as frequency-domain rather than frequency-division multiplexing, since the modulated spectra overlap each other [26]. QAM is limited to specialized applications, notably color television and digital data transmission.

Most communications technology applications such as radio and data communications often employ QAM whose various forms including *M*=16, *M*=32, *M*=64, *M*=128 and *M*=256 QAM are available. Each figure refers to the number of constellation points that is also the number of possible distinct states. The varying QAM levels are likely to be utilized in the case of a communications system requiring data rates beyond what 8-PSK (phase shift keying) can offer. The reason for this is that QAM supplies a wider distance between neighboring points in the I-Q plane since constellation points are spaced more evenly and hence, these points become more distinct and data errors are decreased [40].

QAM may exist as either analog or digital information. The analog formats of QAM are characteristically utilized such that multiple analog signals are passed on one carrier. For instance, it is utilized in PAL and NTSC systems in which various channels supplied by QAM ensure that it carries the elements of chroma or color data. In radio applications, a system

referred to as C-QUAM is employed for AM stereo where various channels ensure that the two signals necessary for stereo are passed on a single carrier [40].

Digital versions of QAM are frequently called “Quantized QAM” and are used more and more for data communications mostly inside radio communications systems applications. In case of being utilized for digital transmission for radio communications, QAM is capable of carrying higher data rates than ordinary amplitude and phase modulated schemes.

Many radio communications and data distribution applications use QAM. Yet, particular applications and standards employ some specific variants of QAM. In native broadcasting systems, for instance, 64-state and 256-state QAM are employed in digital cable television and cable modem systems. United Kingdom uses 16-state and 64-state QAM for digital terrestrial television utilizing DVB (Digital Video Broadcasting). United States made 64-state and 256-state QAM the compulsory modulation schemes that the standards determined for digital cable. Additionally, various QAM schemes are employed for numerous wireless and cellular technology applications, as well [40].

Multilevel QAM schemes are spectrum-efficient modulation schemes that are employed because of the growing request for high-speed multimedia services over the restricted radio spectrum [34].

4.4 Advantages and Disadvantages of QAM

4.4.1 Advantages of QAM

QAM is a modulation type that is frequently employed to modulate information signals upon a carrier utilized for radio communications. It is used frequently since it has got privileges compared with different types of data modulation like PSK, even though many forms of data modulation work side by side.

One advantage of using QAM comes from its being a higher order modulation form and consequently it is capable of carrying more information bits per symbol. A link’s data rate can be increased by choosing a higher order format of QAM.

The QAM signal constellation is equal to two Pulse Amplitude Modulation (PAM) signals on quadrature carriers, for rectangular signal constellations in which $M = 2^l$, where l is even, the two PAM signals have $\sqrt{M} = 2^{l/2}$ signal points. It's simple to calculate the probability of error for QAM using the probability of error for PAM because excellently separating the in-phase-quadrature signal components on the demodulating side is possible.

A clear privilege of rectangular QAM signal constellations is that it is easy to generate them as two Pulse Amplitude Modulation (PAM) signals impressed on phase-quadrature carriers; moreover, demodulating them is also easy. Although they are not the best *M*-ary QAM signal constellations for $M \geq 16$, the average transmitted power needed for achieving a given minimum distance is just a little more than the average power necessary for the best *M*-ary QAM signal constellation. Rectangular QAM signal constellation is easier to implement and has a slightly better BER performance. Rectangular *M*-ary QAM signals are widely used in practice for these reasons [1].

High spectral efficiency of QAM schemes make them potentially suitable for higher data rate transmissions in communication channel [27]. QAM and related quadrature-carrier methods offer increased modulation speed and are the modulation types that best suit digital transmission on telephone lines and other bandwidth-limited channels.

4.4.2 Disadvantages of QAM

Even though it seems that QAM increases the transmission efficiency for radio communications systems by utilizing both amplitude and phase variations, it contains some disadvantages. Firstly, it is more sensitive to noise since the states are compactly nearer causing a lower noise level to be necessary to take the signal to a different decision point. Receivers used in phase or frequency modulation are both capable of using limiting amplifiers which can take out any amplitude noise and enhance the noise reliance. That doesn't happen with QAM.

The second restriction is related with the amplitude component of the signal, as well. In the case of amplifying a phase or frequency modulated signal in a radio transmitter, it is not

necessary for the linear amplifiers to be used, however, if QAM containing an amplitude component is used, linearity has to be maintained. Unluckily, because of being less efficient and consuming more power, linear amplifiers are less appealing to mobile applications.

4.5 Design Features of *M*-QAM Applied to NNFE

4.5.1 Normalization

The most distinguishing feature of this thesis is *normalization*. Normalization is employed as the primary technique which enables the equalizer to process signals with values between 0 and 1 to minimize the cost function and thus the error at the maximum level, at the output. The main reason for using this technique is that the modulated QAM signals (with input values of $\pm 1, \pm 3$ for $M = 16$ as an example) are not directly accessible to the channel. In other words, the channel will distort the unchanged modulated QAM signal such that the equalizer will be unable to reduce the errors at the end to produce the equalized output. Therefore, it is inevitable to come up with a method that the equalizer can process the signal with. The motivation behind normalization technique is to ensure that the transmitted data with as small values as possible, are entered into the channel so that the distortions will be minimal and the nonlinear neuro-fuzzy equalizer (NNFE) will be able to minimize the error to the greatest possible extent.

Fig. 4.4 illustrates the block diagram of the system used in this thesis which involves the use of the normalizer. Normalizer shrinks the values of *M*-QAM modulated data values to values between 0 and 1 by:

1. Shifting the modulated values such that the minimum value will be 0. This is achieved by subtracting the minimum bit from each bit value in order to attain zeros as the minimum value in the modulated set of values.
2. Scaling the modulated values such that the maximum value will be 1. This is achieved by dividing each modulated value in the set (whose minimum is 0) by the maximum bit value in order to attain 1, as the maximum value in the set of values.

After normalization, the zeros in the transmitted set of values are replaced by -0.75 for M = 4 and -1 for M = 16, since the equalizer cannot process the 0 digit to yield its equalized counterpart. This is because there will be local errors causing large amounts of deviations at the output, as the numerous simulation tests have revealed. Consequently, normalization provides the channel with small valued numbers whose distortions are minimized as compared to those when not normalized. Additionally, the error will be further reduced and minimized by the Nonlinear Neuro-Fuzzy Equalizer (NNFE).

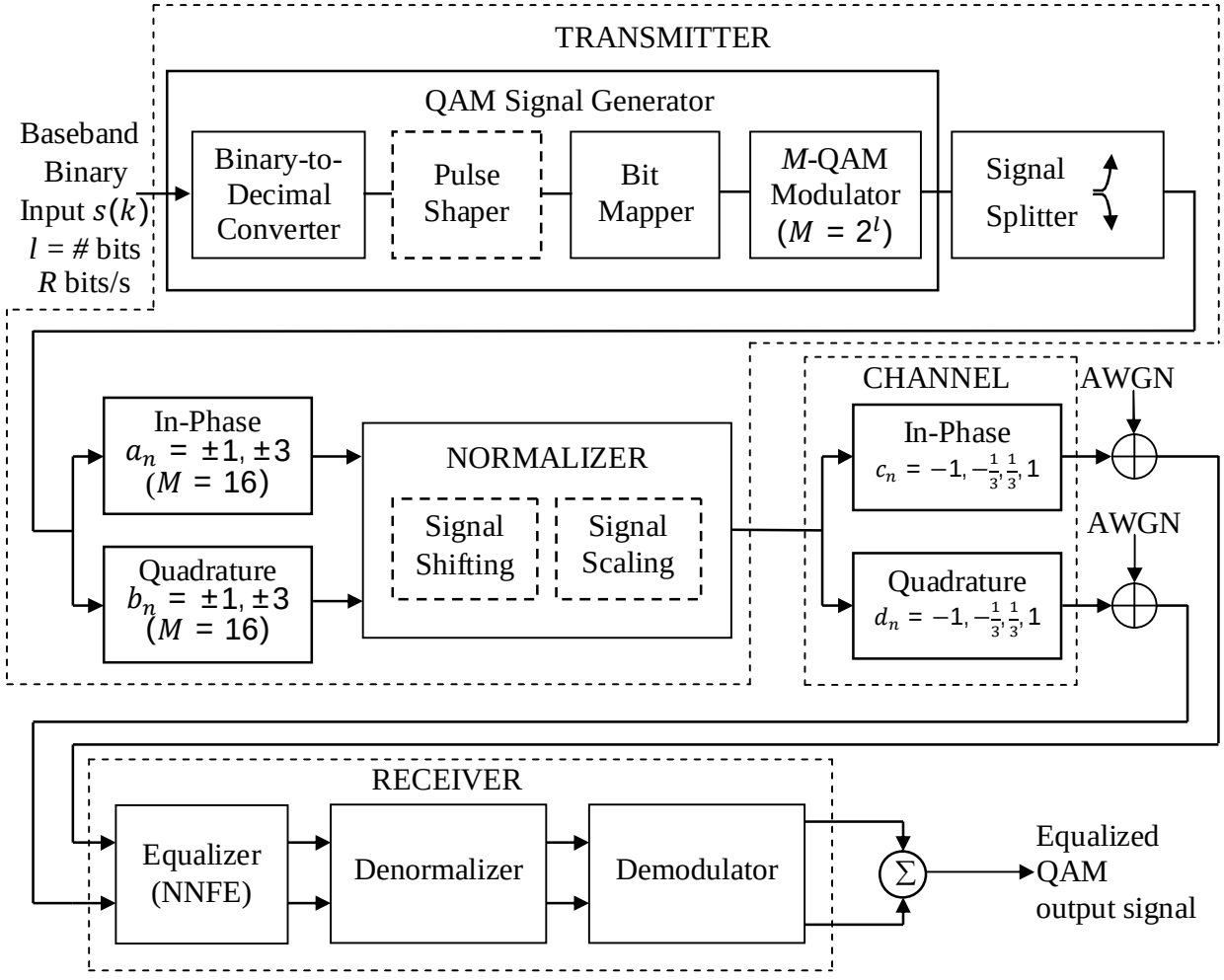


Figure 4.4 Block diagram of the normalizer-based M-QAM signal generating and equalizing communications system

4.5.2 Reciprocity

One of the significant features of the normalizer-based QAM equalizer is the reciprocity. The original transmitted symbol bits are not directly accessible to the receiver and this obligates the application of a different way of inputting the transmitted signal into the channel. On normalizing the modulated signal set, $M = 2^l$ number of reciprocal (corresponding) input values is entered into the channel and the same amount of noisy channel output values are received by the equalizer. Each of these corresponding bits actually represents its corresponding modulated counterpart since the original bits are not accessible to the channel and cannot be processed by the algorithm and the equalizer to yield the equalized output at the receiver end. At the end of the equalization process, each of the equalized signal bits that represents its corresponding transmitted counterpart is *denormalized* to yield its actual reciprocal value as its processed and equalized value. The denormalized corresponding equalizer output is calculated by:

$$x(k) = \pm a_n(k) \pm (percent\_dev) * (\pm) a_n(k) \tag{4.14}$$

and

$$percent\_dev = (eq(k) - c_n(k))/c_n(k) \tag{4.15}$$

where $a_n(k)$ denotes the original in-phase modulated input bit value, $eq(k)$ is the equalizer output value, $c_n(k)$ denotes the normalized transmitted bit value and $percent\_dev$ denotes the percent deviation of the equalizer output value from the normalized transmitted bit value (channel input). The similar calculation is applied for the quadrature part of the QAM signal.

Eventually, the in-phase and quadrature parts are combined by an adding device, to yield the equalized QAM symbol in complex form, at the end.

4.5.3 Complex representations of *M*-QAM constellations

The modulated transmitted symbols $g(n)$ and the normalized transmitted symbols $g(n)_{nor}$ of *M*-QAM can be represented, depending on the level of constellation, by *M* possible complex forms, as illustrated in Table 4.1:

Table 4.1 *M*-QAM transmitted symbols $g(n)$ and normalized transmitted symbols $g(n)_{nor}$

M	QAM ($g(n) = a_n + jb_n$)	Normalized QAM ($g(n)_{nor} = c_n + jd_n$)
4	$(1 + j), (1 - j)$	$(0.75 + j0.75), (0.75 - j0.75)$
	$(-1 + j), (-1 - j)$	$(-0.75 + 0.75j), (-0.75 - j0.75)$
16	$(1 + j), (1 - j)$	$(1/3 + j\,(1/3)), (1/3 - j\,(1/3))$
	$(-1 + j), (-1 - j)$	$(-1/3 + j(1/3)), (-1/3 - j\,(1/3))$
	$(1 + j3), (1 - j3)$	$(1/3 + j), (1/3 - j)$
	$(-1 + j3), (-1 - j3)$	$(-1/3 + j), (-1/3 - j)$
	$(3 + j), (3 - j)$	$(1 + j(1/3)), (1 - j(1/3))$
	$(-3 + j), (-3 - j)$	$(-1 + j(1/3)), (-1 - j(1/3))$
	$(3 + j3), (3 - j3)$	$(1 + j), (1 - j)$
	$(-3 + j3), (-3 - j3)$	$(-1 + j), (-1 - j)$

4.5.4 Multifunctionality

The normalizer-based QAM equalizer is capable of minimizing the error and producing accurate equalizer output results which are dependent on the *signal-to-noise ratio* (SNR), the *minimum error* and the *number of iterations* previously set. In other words, low SNR values will produce higher bit error rates (BER) since the AWGN added to the transmitted data is more dominant around low SNR values, especially around 0-4 dB, as shown in tabulations as a result of simulations applied on the system. Additionally, the equalizer output is calculated and corresponding equalizer outputs clearly demonstrate the relationship between SNR and the equalizer output for each constellation of the QAM modulation scheme (i.e. $M = 4$ and $M = 16$). The lower the SNRs, the more compact the equalizer output and the higher the SNRs, the more open the equalizer output is. In other words, at low SNRs, the equalized signal values are accumulated farther from the ideal (perfect) constellation points as opposed to more compact (closer) accumulation of the equalized signal values to the ideal constellation points, at high SNRs.

The same relationship of dependency of equalizer output to SNR is successfully achieved for the minimum error value that the NNFE operates to reach each time a simulation is executed. That is, lower set minimum error will enable the equalizer to yield more accurate output

results in terms of both lower BERs and more compact signal accumulation whereas higher minimum error will cause less accurate output results in terms of both higher BERs and less compact signal accumulation. Less compact signal accumulation means more scattering of the output signal bits away from the constellation points.

Lastly, normalizer-based QAM equalizer is also a function of the number of iterations as proven by the result of extensive computer simulations applied on the system. It is clearly observed and shown that equalization performance improves upon successive iterations. The less the number of iterations, the higher the BERs and the higher the number of iterations, the lower the BERs are, in addition to more open and clear equalizer output demonstrating highly accurate results.

4.5.5 Gray coding

Gray coding is the mapping technique used to map the transmitted data input that is in decimal form. In the case of *M*-QAM signaling, if, for example, a symbol (1010) in 16-QAM signaling is transmitted, it is clear that should an error occur, the transmitted signal will most likely be mistaken for one of its closest neighboring symbols, (1001) or (1011). The likelihood that (1010) would get mistaken for (1111) is relatively remote. If the assignment of bits to symbols follows the binary coding as illustrated by Fig. 4.5(a), some symbol errors will usually result in two or more bit errors, even with a large SNR, since two adjacent symbols in the complex field constellation, do not differ in only one bit. In the case of nonorthogonal schemes such as *M*-QAM signaling, one often uses a binary-to-*M*-ary code such that binary sequences that correspond to adjacent symbols differ in only one bit position as illustrated by Fig. 4.5(b); thus in the case of an *M*-ary symbol error occurring, it is more likely that only one of the *l* input bits will be in error. Gray code, as illustrated by Fig. 4.5(b), is a code that provides this desirable property [24].

The advantage brought by this technique is related with the fact that if an error occurs in the signal detection, it is more likely that a symbol is confused with one of its neighbors than with one of the others. Thus, the amount of wrong bits in the received signal sequence is minimized. This advantage is the reason for using Gray codes as part of this thesis.

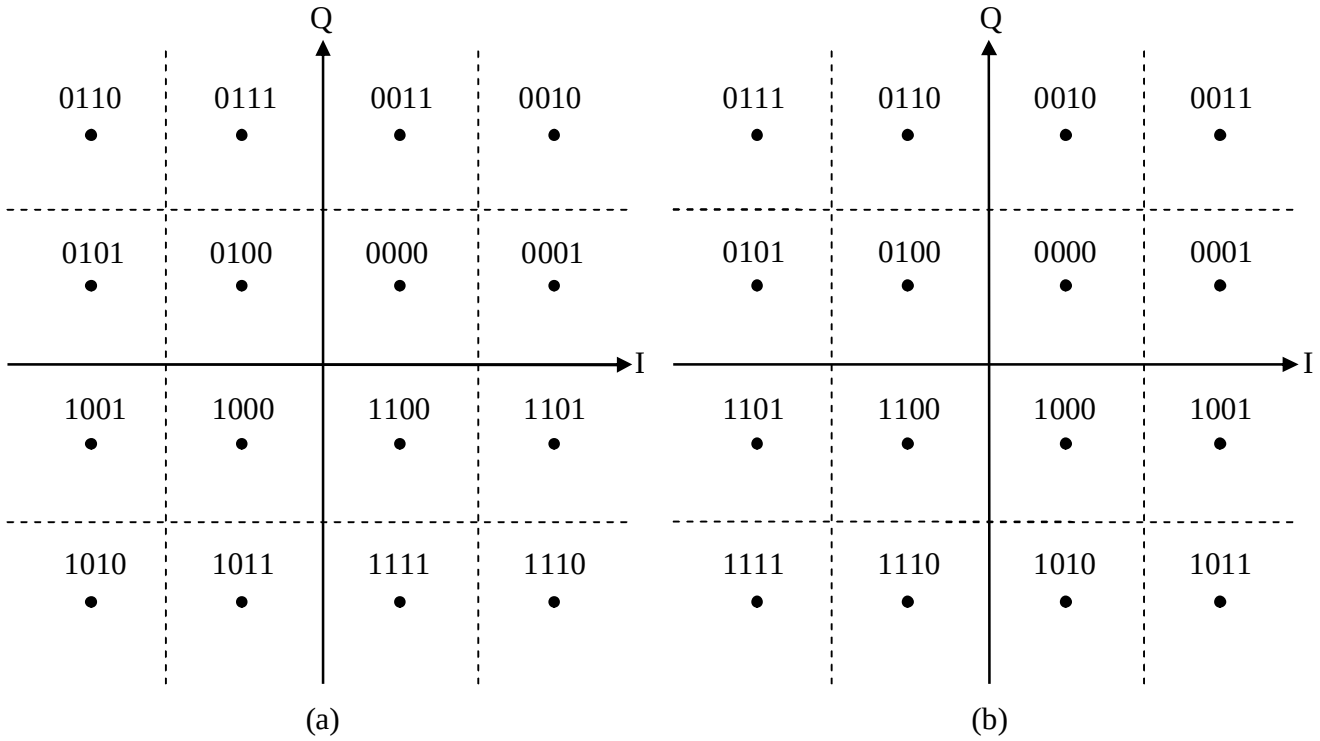


Figure 4.5 16-QAM constellation with (a) binary coding, (b) Gray coding

4.6 Summary

Quadrature amplitude modulation (QAM) and the way it is applied to nonlinear neuro-fuzzy equalizer (NNFE) are analyzed in this chapter. The complex envelope is used as the input QAM transmitted signal because it is the representative of bandpass waveforms and operates as the baseband equivalent of the actual bandpass signal.

The most distinguishing feature of the design of thesis is the normalizer. It enables the channel and the receiver to process the transmitted signals as minimized values between 0 and 1, thus minimizing the channel distortions and the estimation error (the difference between the desired response and the equalizer output). Other distinguishing features which are reciprocity and multifunctionality are also explained in detail enabling the comprehension of significant capabilities of the thesis design. Reciprocity basically points out the logic of the channel, algorithm and the NNFE processing the corresponding (representative) signals instead of the actual original transmitted signal since the original modulated transmitted symbol bits are not directly accessible to the channel and the receiver. Multifunctionality feature explains the

capability of the equalizer to process and equalize the transmitted data correctly and proportionally as a function of SNR, minimum error and the number of iterations. Additionally, Gray coding property with its implementation and advantages are explained, as well. Applications of QAM in practice are presented and its main advantages and disadvantages are also stated.

The structure of adaptive channel equalization and the type of equalizer which is mainly the nonlinear neuro-fuzzy equalizer are described. The operational properties of the nonlinear adaptive equalizer that involve basically the training and tracking properties are considered. The equalizer output is first computed from the transmitted test signal after a training process which involves the minimization of error to a level which enables the equalization of the channel. The tracking mode, through which the equalizer goes after the training enables the equalizer to track possible time variations which are typical characteristics of the nonlinear channel, by using a receiver estimate of the transmitted sequence as a desired response. Eventually, the compensation methods by using an adaptive nonlinear equalizer and the characteristics of the equalizer are understood.

CHAPTER 5

SIMULATION RESULTS AND ANALYSIS

5.1 Overview

This chapter is reserved for the in-depth analysis of the QAM signaling on normalizer-based nonlinear neuro-fuzzy equalizer (NNFE). The simulation results of two different constellation levels ($M = 4$ and $M = 16$) will be presented in tabulated forms. Analysis of the results of the simulations of each constellation in terms of bit error rate (BER) and signal-to-noise ratio (SNR) will be presented in tabulated forms, in order to illustrate the performance of the normalizer-based QAM equalization system. Actual channel and equalizer output figures demonstrating the state of the channel and the equalizer will be presented in order to demonstrate the accuracy of the system. All the statistical and test analysis results are presented for both linear and nonlinear channels and the accuracy and performance of the NNFE system in time-varying nonlinear channel conditions is presented.

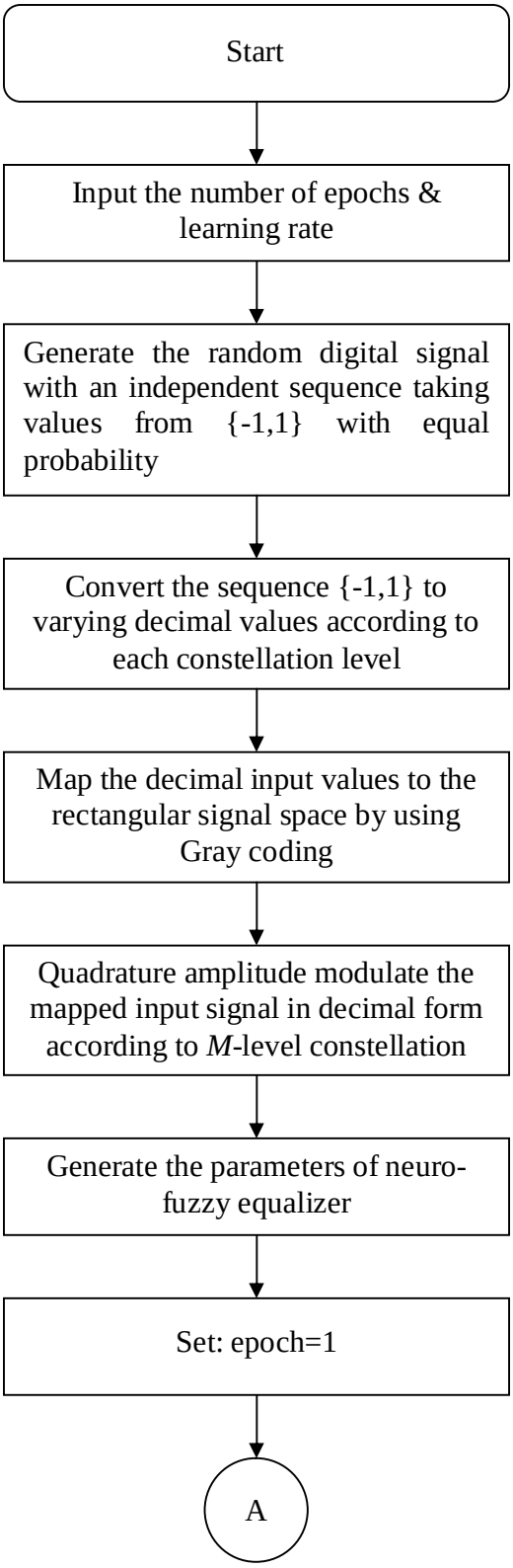
5.2 Development of Normalizer-based Nonlinear Neuro-Fuzzy Equalizer (NNFE) System

During the equalizer design, the equalizer current output signals (in normalized form) are compared with the input signals transmitted through the channel. When error is encountered, the learning of the neuro-fuzzy equalizer starts. The adjustment of the parameter values of the equalizer by utilizing formulas (3.22), (3.23), (3.24) and (3.25) is included in learning. Learning is continued until, for all input-output pairs, the value of the error will be an acceptable minimum value. During simulation, the transmitted signals $s(k)$ are normalized input samples that are the reciprocal (corresponding) signals of the transmitted signals with an equal probability of -1 and 1 and are corrupted by additive white Gaussian noise (AWGN). These corrupted (noisy) signals are the inputs for the equalizer.

5.3 Flowchart Diagram of the Normalizer-based Neuro-Fuzzy Equalization System

The flowchart diagram of the neuro-fuzzy equalization system with normalizer is given in Fig.5.1. The block scheme of realization of the normalizer-based neuro-fuzzy equalization system includes the following steps:

- Enter the number of epochs, learning rate
- Generate random digital input signal for channel with an independent sequence taking values from {-1,1} with equal probability
- Convert the sequence {-1,1} to decimal values according to each constellation level
- Map the decimal input values to the rectangular signal space by using Gray coding
- Quadrature amplitude modulate the mapped input signal in decimal form according to *M*-level constellation
- Generate the parameters of the neuro-fuzzy equalizer. Enter the number of neurons in input, hidden and output layers
- Set epoch number to 1
- Select input signal and send to the channel
- Add additive white Gaussian noise (AWGN) to the channel and calculate the output of the channel
- Define the input signals for the equalizer and send them to the equalizer
- Calculate the equalizer output
- Calculate the error of the equalizer output
- Test the value of error. If error is less than an acceptable minimum value, then take the next value of binary input signals and send it to the channel
- If error is more than an acceptable minimum value, then using the learning algorithm, train the parameters of the equalizer
- Take the next value of binary input signals and send it to the channel input
- Test the number of epochs. If it is more than the given number of epoch value, stop the training process
- If epoch number is less than the given number of epoch value, increment the current epoch value and send the first binary input signal to the channel.


Start
Input the number of epochs & learning rate
Generate the random digital signal with an independent sequence taking values from {-1,1} with equal probability
Convert the sequence {-1,1} to varying decimal values according to each constellation level
Map the decimal input values to the rectangular signal space by using Gray coding
Quadrature amplitude modulate the mapped input signal in decimal form according to *M*-level constellation
Generate the parameters of neuro-fuzzy equalizer
Set: epoch=1
A

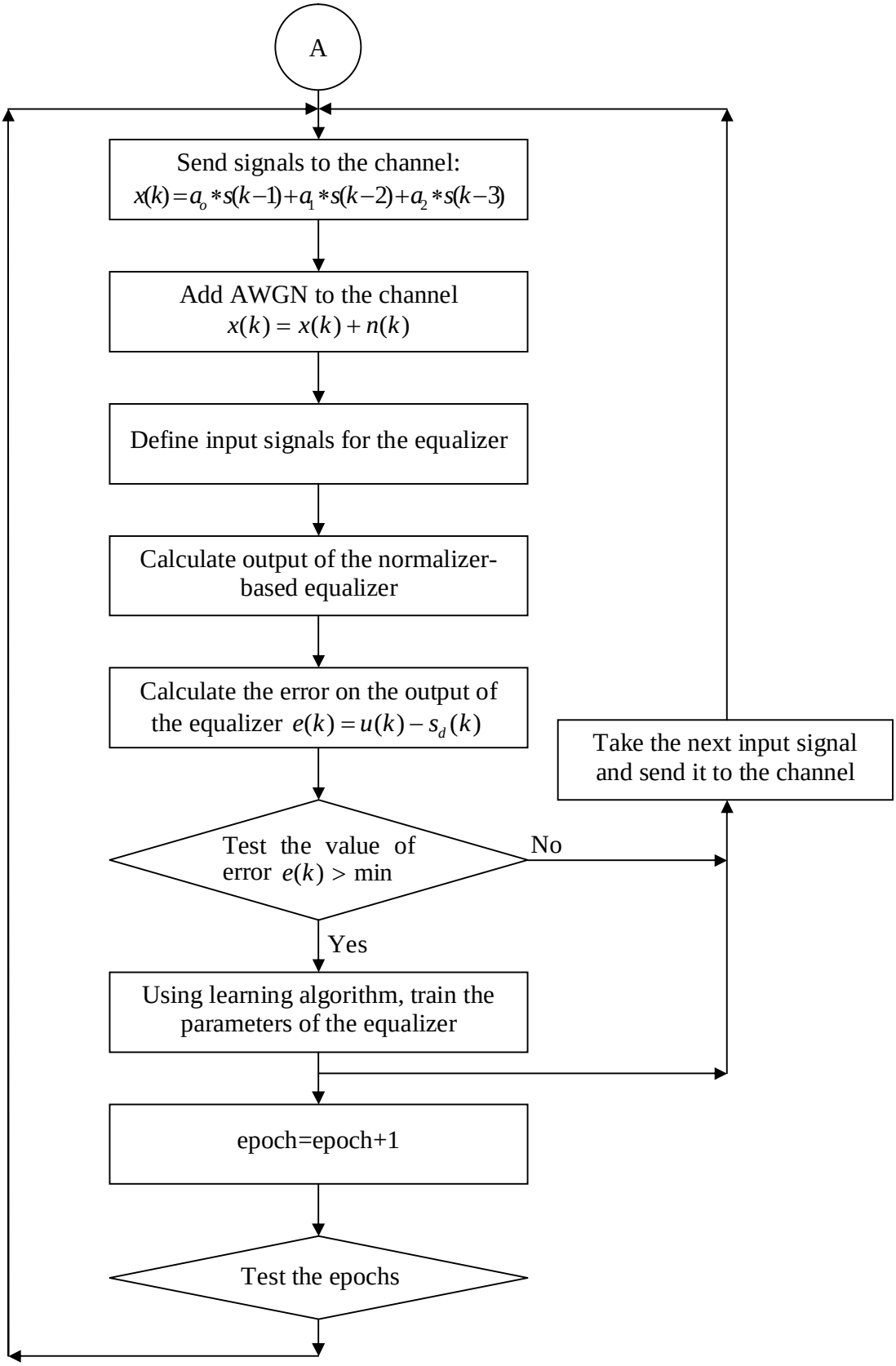


Figure 5.1 Flowchart diagram of normalizer-based neuro-fuzzy equalization system

The NNFE structure and its training algorithm are used for designing the equalizer. The network input signals $x(k)$ (see Fig. 3.7) are the channel output signals applied to the network at time $k, x_i(k-i), (i = 1 \ldots 4)$, the number of neurons in the input layer is equal to 27, the number of hidden neurons (rules) is equal to 27 and u is the output signal of the network.

During simulation, the input signals of the equalizer $x(k),\ x(k-1),\ x(k-2),\ x(k-3)$ are the output signals of the channel. The simulation on MATLAB of the channel equalization system has been executed during which 27 rules (hidden neurons) are used for the linear channel and 36 rules are used for the nonlinear channel. The learning of equalizers has been carried out for 2000 iterations and the parameter values of the NNFE have been determined.

5.4 Analysis of Bit Error Rate (BER) and Signal-to-Noise Ratio (SNR)

An equalizer's performance is evaluated by its probability of error P_e. This probability predicts the BER. In order to calculate the BER, the equalizer is tested with a statistically independent random set of 3000 symbols (samples) for both 4-QAM and for 16-QAM. The 3000 symbols correspond to 6000 bits for 4-QAM and 12000 bits for 16-QAM. The training and computation time of BER is approximately 25 minutes for 4-QAM and 50 minutes for 16-QAM on a CPU (central processing unit) with 1.5 GHz Intel Celeron processor. An error value $e_i(k)$ is generated for a range of noise variances.

The BER is plotted against the channel noise to determine and compare the equalizer performances. The BER is calculated from:

$$BER = \sum_{i=1}^{n} \frac{e_i(n)}{n} \tag{5.1}$$

where $e_i(n)$ is the number of bits in error and n is the number of total bits.

The channel noise is measured as a signal-to-noise ratio (SNR) given by:

$$SNR = 10\log_{10}\left(\frac{\sigma_s^2}{\sigma_n^2}\right) \text{dB} \tag{5.2}$$

where σ_s^2 and σ_n^2 are the signal and noise variances, respectively.

5.5 Simulation of the Normalizer-based NNFE System for Linear Channel

This thesis is based on processing and equalizing QAM signals for two constellation levels ($M = 4$ and $M = 16$) by using NNFN equalizer and its training algorithm. Extensive simulations have been executed for the equalization of signals transmitted through both linear and nonlinear channels.

In the first simulation where linear channel is employed, the following non-minimum-phase channel model is used:

$$x(k) = a_1(k)s(k) + a_2(k)s(k-1) + a_3(k)s(k-2) + n(k) \tag{5.3}$$

where $a_1(k) = 0.3482$, $a_2(k) = 0.8704$ and $a_3(k) = 0.3482$ and $n(k)$ is the additive white Gaussian noise (AWGN). This type of channel is widely used in real communications systems. Table 5.1 illustrates the BER comparison of the non-minimum-phase channel after training with noise. In Figures 5.5 and 5.12, the convergence curves of the NNFE of each constellation for 2000 learning iterations are illustrated. Figures 5.3 and 5.10 demonstrate the linear channel output states which are also the input (received) signal for the equalizer.

At the output of the equalization system, the deviation of the normalized transmitted signal from the normalized current equalizer output is determined. This deviation or error $e(k)$ of Eq. (5.4) is used in adjusting the network parameters. Training continues until the value of the error for all training sequence of signals is below a predetermined acceptable minimum value.

$$e(k) = u(k) - s_d(k) \tag{5.4}$$

where $e(k)$ is the minimum error, $u(k)$ is the equalizer output signal and $s_d(k)$ is the desired (reference) signal. The predetermined minimum error $e(k)$ set for the simulation of all QAM constellations (for both linear and nonlinear channels) is 0.0001.

5.6 Simulation of the NNFE System for Nonlinear Time-Varying Channel

In the second simulation, the processing and equalization of QAM signals by neuro-fuzzy equalization system for nonlinear, time-varying channel have been executed. The following nonlinear channel model is used for this simulation:

$$x(k) = a_1(k)s(k) + a_2(k)s(k-1) - 0.9(a_1(k)s(k) + a_2(k)s(k-1))^3 + n(k) \tag{5.5}$$

where $x(k)$ is the output of the channel, $s(k-1)$ is the time delay introduced by the channel and $a_1(k)$ and $a_2(k)$ are time varying channel coefficients with initial values $a_1(0) = 1$ and $a_2(0) = 0.5$. These channel coefficients are generated by using a second-order Markov model with 3rd order nonlinearity in the presence of AWGN filtered by a second-order Butterworth low-pass filter with normalized cut-off frequency 0.1 [31]. The colored Gaussian sequences that are used as time-varying coefficients a_i are generated with a standard deviation of 0.1. The time varying impulse response $h(k)$ of the channel model (5.5) is given by:

$$h(k) = \sum_{i=1}^{2} a_i(k)\delta(k-(i-1)) - 0.9\left[\sum_{i=1}^{2} a_i(k)\delta(k-(i-1))\right]^3 \tag{5.6}$$

where $\delta(k)$ is the unit impulse.

The simulations are performed using NNFE where 36 neurons are used in the hidden layer of the network. The transmitted signals are assumed to be normalized input samples that are the reciprocal (corresponding) signals of the independent sequence of transmitted signals with an equal probability of -1 and 1. On the output of the channel, the additive white Gaussian noise $n(k)$ is added to the transmitted signal. Figures 5.4 and 5.11 demonstrate the time-varying, nonlinear channel output states which are also the input (received) signal for the equalizer. Noise variation σ_n^2 that varies for each constellation and SNR are given respectively. In Figures 5.6 and 5.13, the convergence curves of the NNFE of each constellation for 2000 learning iterations are illustrated. Table 5.2 demonstrates the BER performance of the NNFE of the time-varying channel after equalization and the results are obtained when the equalizer is trained with noise.

5.7 Analysis of Simulations

The following presents the simulation analysis and results of *M*-QAM signals for $M = 4$ and $M = 16$ constellations transmitted through linear and nonlinear channel models of (5.3) and (5.5) respectively. Tables 5.1 and 5.2 show the BER performance of the linear and nonlinear channels after equalization for 4-QAM and 16-QAM respectively. The results are obtained when the equalizer has been trained with AWGN. Fig.5.3, 5.4, 5.10 and 5.11 clearly illustrate the channel output (receiver input) states for SNR=0dB, SNR=8dB and SNR=10dB. It can be clearly observed that at low SNR where the noise ratio is high, the signals are more distorted and scattered around the constellation points than the signals when SNR is high, since there is less noise around SNR=8dB and SNR=10dB. Figures 5.2 and 5.9 illustrate the effect of SNR on BER performance for both linear and nonlinear channels of 4-QAM and 16-QAM, respectively. Figures 5.7, 5.8, 5.14 and 5.15 clearly illustrate the equalizer output signal states of 4-QAM and 16-QAM constellations where the difference between severely noisy and less noisy conditions for both linear and nonlinear channels can be observed and realized.

5.7.1 Simulation results of 4-QAM

Table 5.1 BER performance of channel models (5.3) and (5.5) for 4-QAM

	LINEAR CHANNEL		NONLINEAR CHANNEL	
	SNR (dB)	**BER**	**SNR (dB)**	**BER**
4-QAM	8.049138	8.332915×10^{-4}	8.010341	9.161845×10^{-4}
	6.005757	7.711243×10^{-2}	6.031466	7.775319×10^{-2}
	4.007480	1.713182×10^{-1}	4.012408	1.867750×10^{-1}
	2.007692	4.217346×10^{-1}	2.016702	4.190250×10^{-1}
	0.058600	4.805201×10^{-1}	0.032696	4.825291×10^{-1}

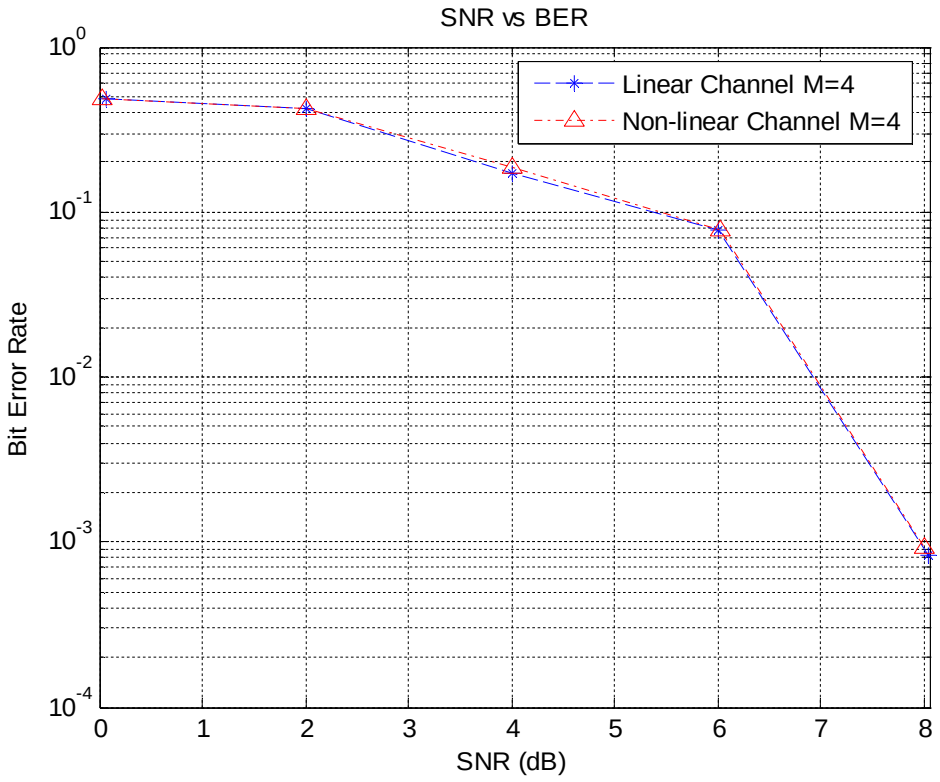
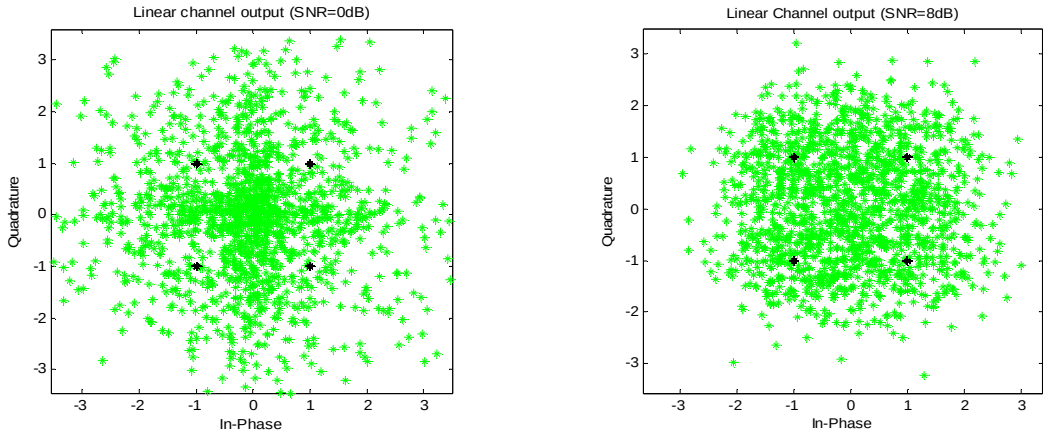
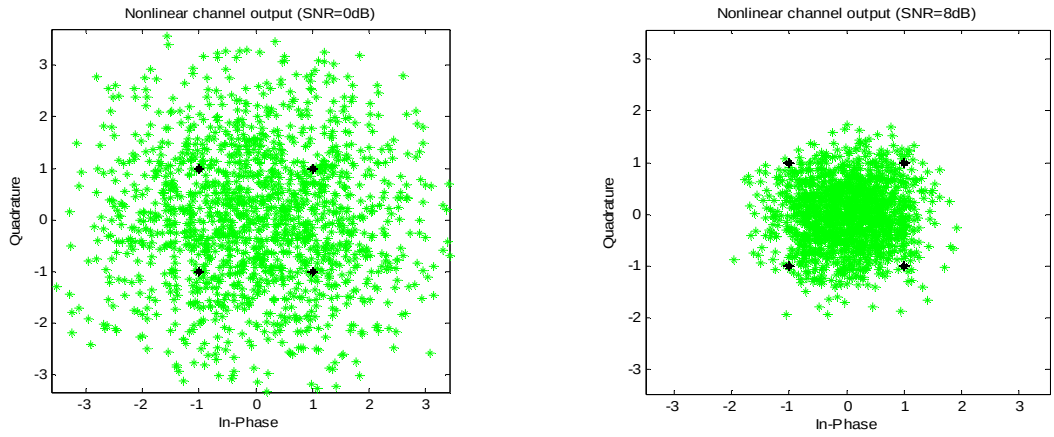


Figure 5.2 4-QAM BER performance of normalizer-based NNFE for linear channel (dashed line with '*') and nonlinear channel (dash-dotted line with triangles)



(a) (b)

Figure 5.3 Linear channel outputs of 4-QAM for (a) SNR=0dB, and (b) SNR=8dB,



(a) (b)

Figure 5.4 Nonlinear channel outputs of 4-QAM for (a) SNR=0dB, and (b) SNR=8dB,

The next figures illustrate the convergence curve of 4-QAM constellation using the linear channel.

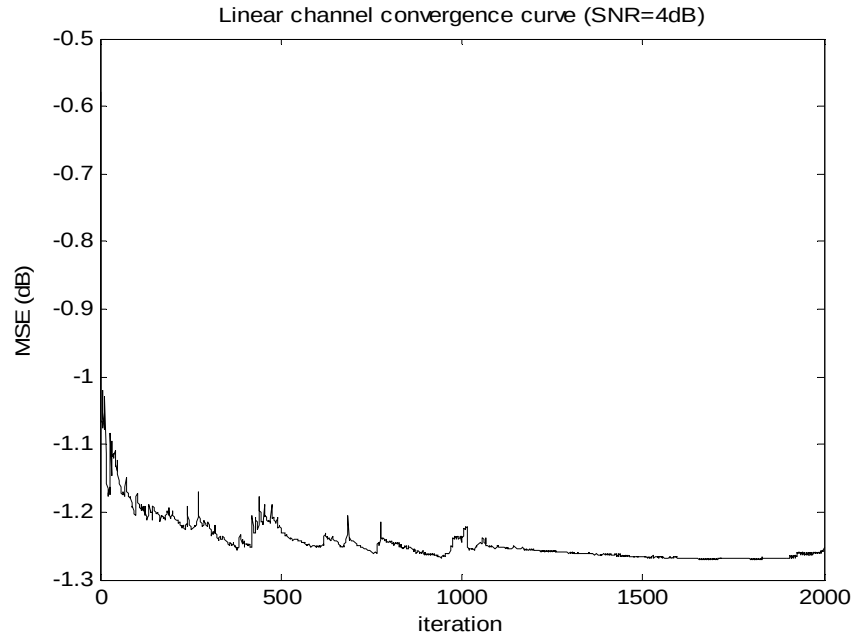


Figure 5.5 Linear channel convergence curve of 4-QAM at SNR=4dB and $\sigma_n^2 = 0.2$

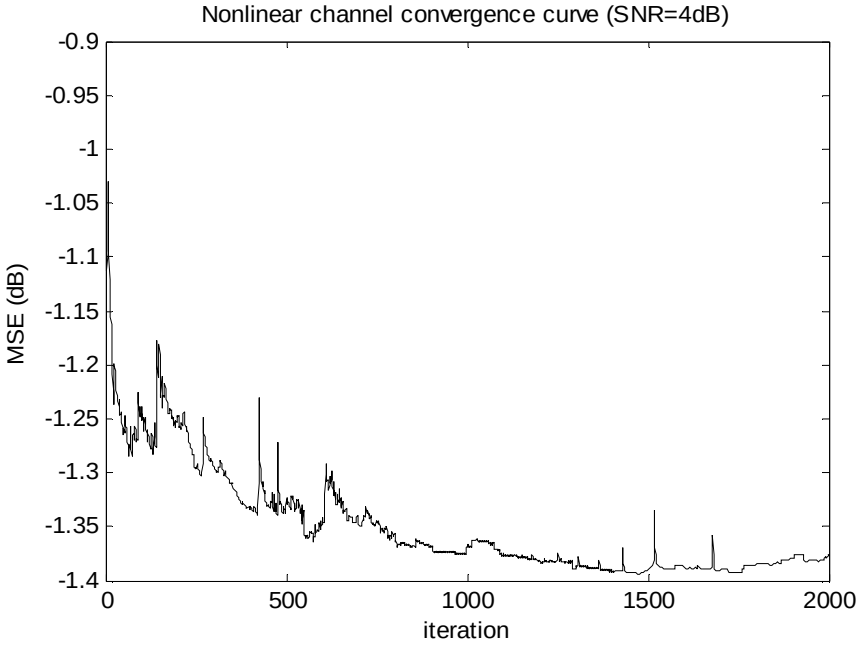
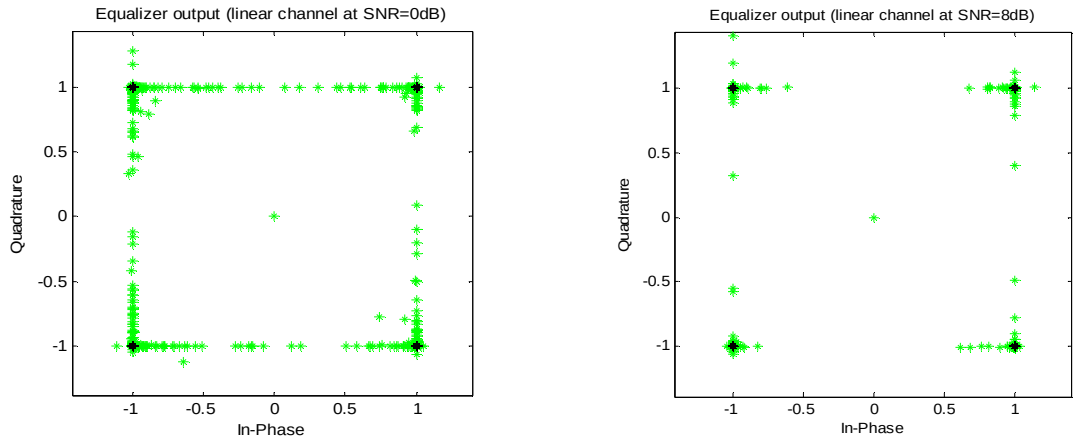


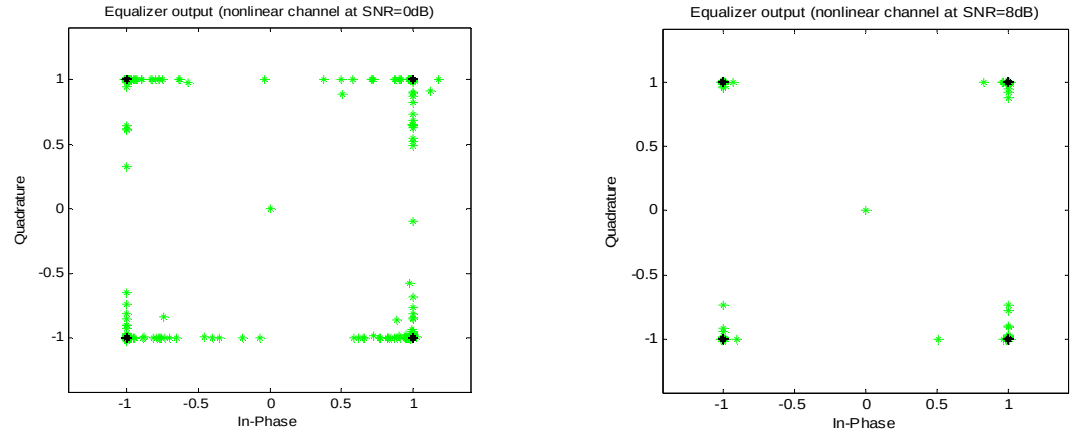
Figure 5.6 Nonlinear channel convergence curve of 4-QAM at SNR=4dB and $\sigma_n^2 = 0.028$

Figures 5.7 and 5.8 illustrate the equalizer output states of 4-QAM constellation where the difference between severely noisy and less noisy conditions can be realized.



(a) (b)

Figure 5.7 Equalizer outputs of 4-QAM for linear channel (a) SNR=0dB, and (b) SNR=8dB,



(a) (b)

Figure 5.8 Equalizer outputs of 4-QAM for nonlinear channel (a) SNR=0dB, and (b) SNR=8dB,

5.7.2 Simulation results of 16-QAM

Table 5.2 BER performance of channel models (5.3) and (5.5) for 16-QAM

	LINEAR CHANNEL		NONLINEAR CHANNEL	
	SNR (dB)	**BER**	**SNR (dB)**	**BER**
16-QAM	10.040336	2.254417×10^{-3}	10.031276	2.541667×10^{-3}
	8.010011	3.496667×10^{-2}	8.007868	3.801639×10^{-2}
	6.029870	8.228397×10^{-2}	6.012374	8.675261×10^{-2}
	4.009074	1.977416×10^{-1}	4.030092	1.983167×10^{-1}
	2.013060	4.223184×10^{-1}	2.004051	4.450448×10^{-1}
	0.068211	4.874667×10^{-1}	0.076989	4.907752×10^{-1}

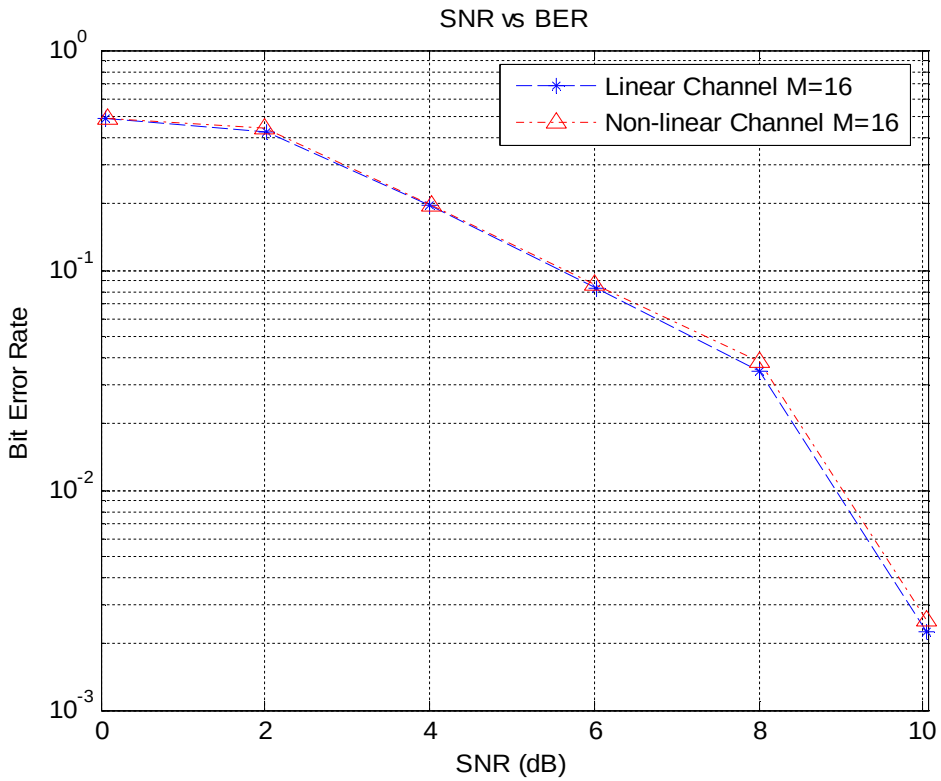
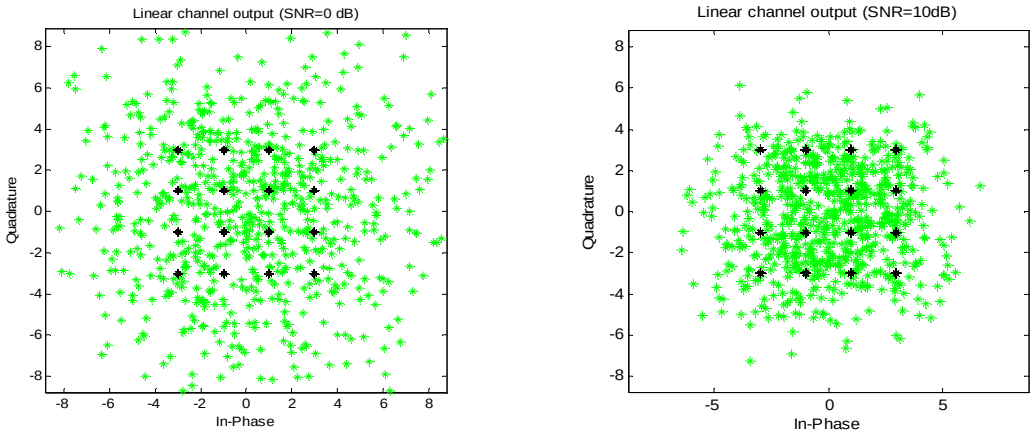
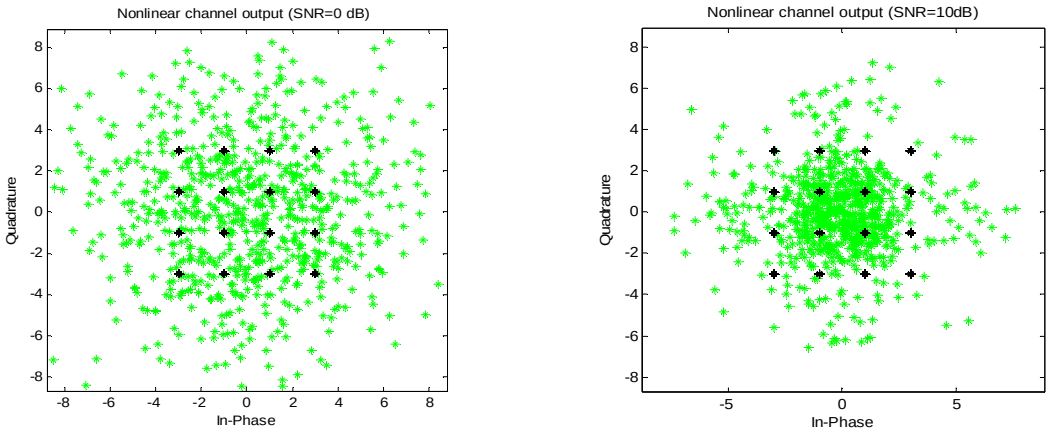


Figure 5.9 16-QAM BER performance of normalizer-based NNFE for linear channel (dashed line with '*') and nonlinear channel (dash-dotted line with triangles)



(a) (b)

Figure 5.10 Linear channel outputs of 16-QAM for (a) SNR=0dB, and (b) SNR=10dB,



(a) (b)

Figure 5.11 Nonlinear channel outputs of 16-QAM for (a) SNR=0dB, and (b) SNR=10dB,

Figures 5.12 and 5.13 illustrate the convergence curve of 16-QAM constellation. Figures 5.14 and 5.15 illustrate the equalizer output values of 16-QAM constellation where the difference between severely noisy and less noisy conditions can be realized.

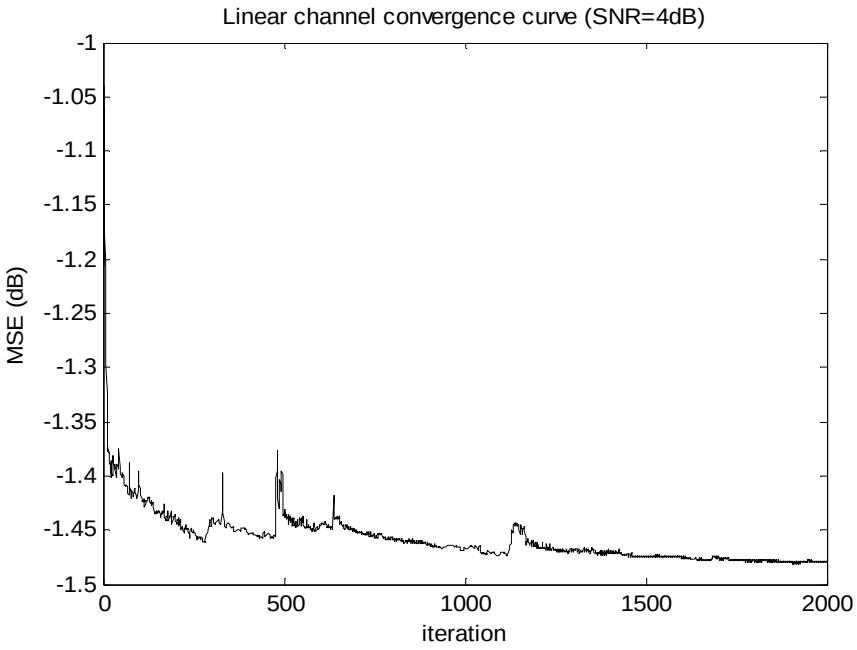


Figure 5.12 Linear channel convergence curve of 16-QAM at SNR=4dB and

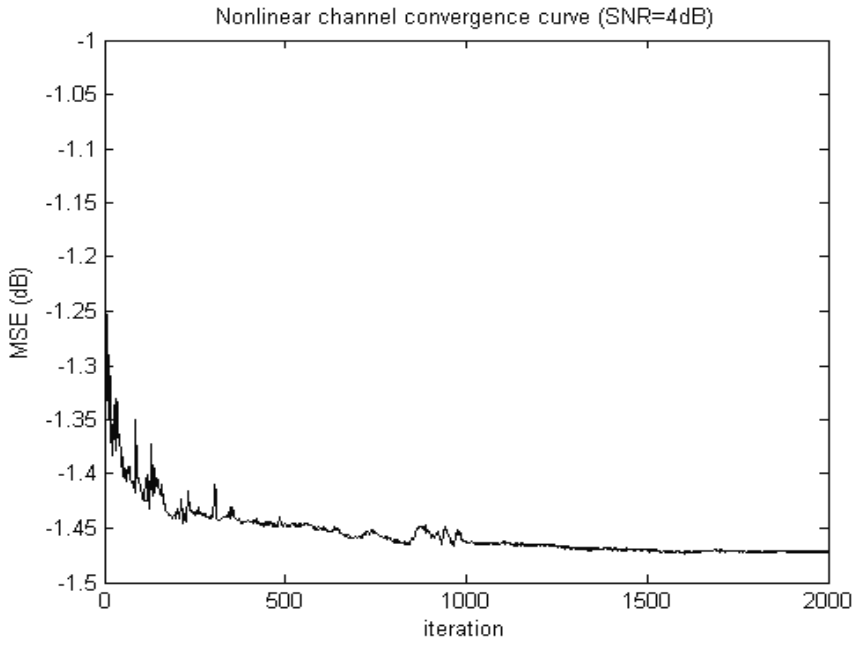
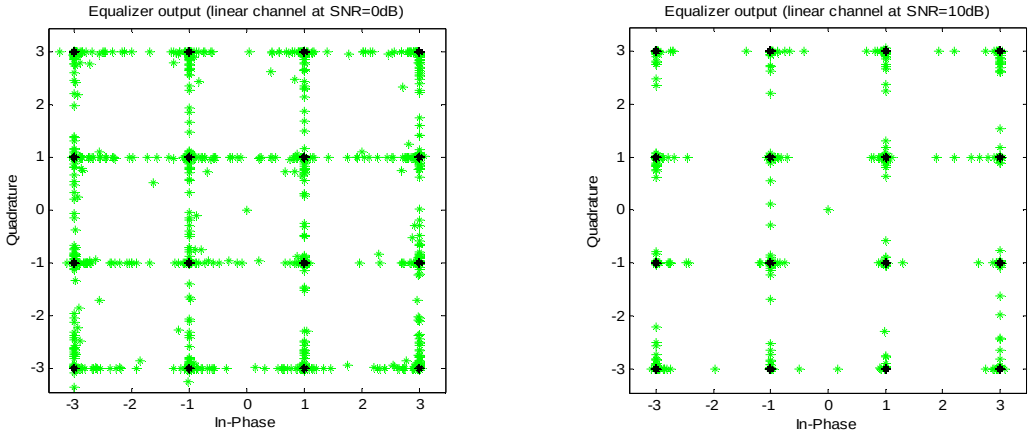
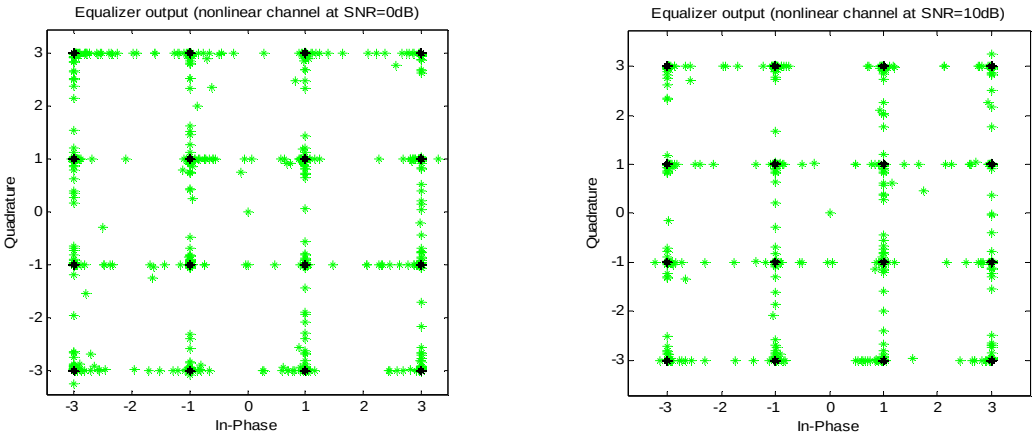


Figure 5.13 Nonlinear channel convergence curve of 16-QAM at SNR=4dB and



(a) (b)

Figure 5.14 Equalizer outputs of 16-QAM for linear channel (a) SNR=0dB, and (b) SNR=10dB,



(a) (b)

Figure 5.15 Equalizer outputs of 16-QAM for nonlinear channel (a) SNR=0dB, and (b) SNR=10dB,

5.8 Comparison Analysis

After the simulations of the QAM constellations, the results of the *M*-QAM equalization have been compared for linear and nonlinear channel models of the two constellations. Table 5.3 shows the comparison results of QAM constellation levels between linear and nonlinear channels and Fig. 5.16 illustrates the BER comparison of 4-QAM with 16-QAM.

Table 5.3 BER performance comparison of *M*-QAM between linear and nonlinear channels

SNR (dB)	4-QAM	16-QAM
	Linear vs. Nonlinear (%)	Linear vs. Nonlinear (%)
~10.00	N/A	12.74
~8.00	9.942	8.72
~6.00	0.831	5.431
~4.00	9.022	0.215
~2.00	-0.643	5.381
~0.00	0.418	0.679
Overall Average	+3.914%	+5.528%

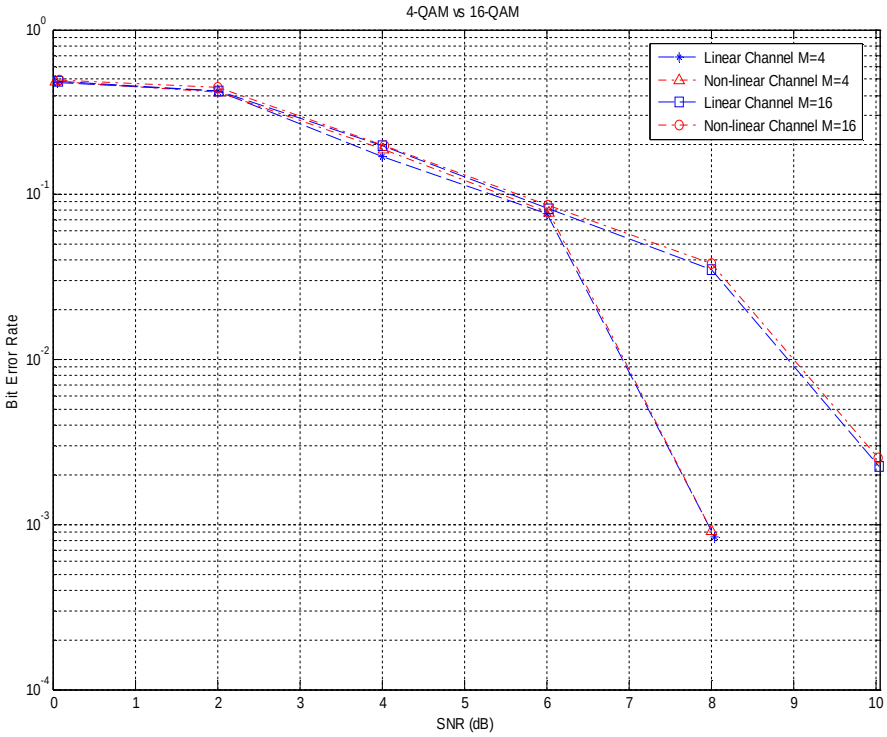


Fig. 5.16 BER comparison of 4-QAM with 16-QAM for both linear and nonlinear channels

Figure 5.17 illustrates the simulated BER performance of 4-QAM and 16-QAM together with the theoretical BER performance of 4-QAM and 16-QAM.

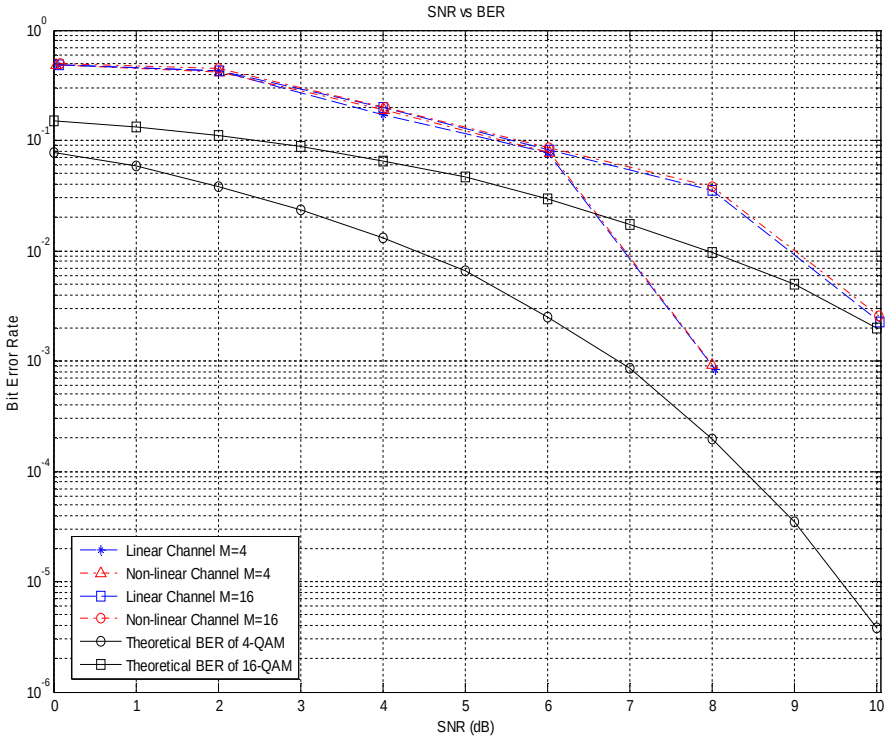


Fig. 5.17 Simulated and Theoretical Bit Error Rate of 4-QAM and 16-QAM [41,42]

CHAPTER 6

CONCLUSION

The MATLAB code which is based on gradient-descent learning algorithm has been designed and developed for NNFE equalization.

The technique of normalization has been introduced in this thesis as an efficient method which is applied directly at the transmitter of the communications system in order to further minimize the estimation error and thus the mean square error (MSE). The normalizer achieves this by reducing the transmitter input data in decimal form to numbers between 0 and 1 that vary according to the modulation scheme of multilevel quadrature amplitude modulation (QAM). The normalizer-based nonlinear neuro-fuzzy equalizer (NNFE) has effectively equalized the two different levels of QAM signals that are distorted in severely noisy channel conditions and has provided faster convergence rate and better BER performance. Comparison between linear and nonlinear channel models has revealed that NNFE performs slightly better equalization for linear channel than for the nonlinear channel model, since the average BER of linear channel is approximately 3.9% better than that of the nonlinear channel for 4-QAM. The same comparison between linear and nonlinear channel models of 16-QAM constellation has revealed that the average BER of linear channel is about 5.5% better than that of the nonlinear channel since it is more difficult to equalize the nonlinear channel because of the time-varying coefficients and the nonlinearities present in the nonlinear channel. The BER performances of multilevel QAM can also be observed from the graphical results of the simulations. The simulated bit error rates are quite high for small SNRs especially around SNR$\leq$4 dB due to high amount of noise, as can be observed from the graphs, but as the SNR increases, the BER values start decreasing more rapidly, as expected.

Overall, it can be concluded that the equalizer performs better for the linear channel in the case of both of the 4-QAM and 16-QAM constellations because the BER values of the linear channel have been tested to be less than that of the nonlinear channel as can be observed from the graphical BER performance of each channel for each constellation. Additionally, the equalizer has shown better BER performance in 4-QAM constellation when compared with

16-QAM constellation, which is an expected result. The simulation results obtained in this work demonstrate similarity with the theoretical BER performance for both 4-QAM constellation and 16-QAM constellation.

In this thesis, it has been proven that multilevel Quadrature Amplitude Modulation (QAM) signals can be successfully processed and accurately equalized in both linear channel conditions and nonlinear time-varying channel conditions by utilizing nonlinear neuro-fuzzy equalizer (NNFE).

FUTURE WORK

The suggestion for future work is the testing of QAM signal equalization on Adaptive Neuro-Fuzzy Inference System (ANFIS) and compare the performance of NNFE with that of ANFIS. Genetic algorithm instead of gradient-descent learning algorithm is suggested for the testing. Additionally, developing a system that is capable of the equalization of QAM involving phase modulation may be considered.

REFERENCES

[1] Proakis, J. (1995). *Digital Communications.* New York, McGraw-Hill.

[2] Couch, L.W. (2001). *Digital and Analog Communication Systems*, 6th edition. New Jersey, Prentice-Hall.

[3] Rappaport, T.S. (2002). *Wireless Communications*. New Jersey, Prentice-Hall.

[4] Proakis, J., "Adaptive Equalization for TDMA Digital Mobile Radio," *IEEE Transactions on Vehicular Technology,* Vol.40, No. 2, pp. 333-341, May 1991.

[5] Banovic, K., Khalid, M.A.S., Abdel-Raheem, E., "A Configurable Fractionally-spaced Blind Adaptive Equalizer for QAM Demodulators," *Science Direct*, pp. 1071-1088, November 2006.

[6] Godard, D.N., "Self-recovering Equalization and Carrier Tracking in Two-dimensional Data Communication Systems," *IEEE Transactions on Communications*, pp. 1867-1875, November 1980.

[7] Treichler, J.R., Agee, B.G., "A New Approach to Multipath Correction of Constant Modulus Signals," *IEEE Transactions on Acoustic Speech Signal Processing ASSP*, pp. 459-472, February 1983.

[8] Yang, J., Werner, J.J., Dumont G.A., "The Multimodulus Blind Equalization and Its Generalized Algorithms," *IEEE J. Select. Areas Communication*, pp.997-1015, May 2002.

[9] Yang, J., Werner, J.J., Dumont G.A., "The Multimodulus Blind Equalization Algorithm," *Proceeding in International Cenference on Digital Signal Processing*, pp. 127-130, 1997.

[10] Oppenheim, A.V., Willsky A.S. (1997). *Signals & Systems*, 2nd edition. New Jersey, Prentice-Hall.

[11] Abiyev, R., Mamedov, F., Alshanableh, T., "Equalization of Channel Distortion by Using Nonlinear Neuro-Fuzzy Network," *ISNN (2), Lecture Notes in Computer Science,* vol. 4492, pp.241-250, Springer, 2007.

[12] Patra, S.K., Mulgrew, B., "Efficient Architecture for Bayesian Equalization using Fuzzy Filters," *IEEE Transactions on Circuits and Systems-II:Analog and Digital Signal Processing*, Vol.45, No.7, pp.812-819, July 1998.

[13] Duda R.O., Hart, P.E. (1973) *Pattern Classification and Scene Analysis*, New York, Wiley.

[14] Quereshi, S.U.H., "Adaptive Equalization," *Proceedings IEEE*, vol.73, pp. 1349-1387, September 1985.

[15] Haykin, S. (1994). *Neural Networks-A Comprehensive Foundation*, Englewood Cliffs, New Jersey, Prentice Hall.

[16] Chen, S., Gibson, G.J., Cowan, C.F.N., Grant, P.M., "Adaptive Equalization of Finite Nonlinear Channels Using Multilayer Perceptron," *Signal Processing (EURASIP),* vol.20, pp.107-119, 1990.

[17] Meyer, M., Pfeiffer, G., "Multilayer Perceptron based Decision Feedback Equalizers for Channels with Intersymbol Interference," *Proceedings of Institute of Electrical Engineering*, Vol. 140, pp.420-424, December 1993.

[18] Moody, J., Darken, C.J., "Fast Learning in Networks for Locally Tuned Processing Unit," *Neural Computation*, Vol.1, No.2, pp. 281-294, 1989.

[19] Mulgrew, B., "Applying Radial Basis Functions," *IEEE Signal Processing Magazine,* Vol.13, pp.50-65, March 1996.

[20] Patra, S.K., Mulgrew, B., "Fuzzy Implementation of a Bayesian Equalizer in the Presence of Intersymbol and Co-Channel Interference," *IEE Proceedings of Communications,* Vol.145, No.5, October 1998.

[21] Wang, L.X., Mendel, J.M., "Fuzzy Adaptive Filters with Application to Nonlinear Channel Equalization," *IEEE Transactions on Fuzzy Systems,* Vol.1, pp.161-170, August 1993.

[22] Sarwal, P., Srinath, M.D., "A Fuzzy Logic System for Channel Equalization," *IEEE Transactions on Fuzzy Systems,* Vol.3, pp.246-249, May 1995.

[23] Glover, I.A., Grant, P.M. (2004). *Digital Communications,* 2nd edition. Essex, England. Pearson Prentice Hall.

[24] Sklar, B. (2001). *Digital Communications,* 2nd edition. New Jersey. Prentice Hall PTR.

[25] Haykin, S. (2002). *Adaptive Filter Theory,* 4th edition. New Jersey, Prentice Hall.

[26] Carlson, A.B., Crilly, P.B., Rutledge, J.C. (2002). *Communication Systems, An Introduction to Signals and Noise in Electrical Communication*. 4th edition. New York. McGraw Hill.

[27] Patra, J.C., Poh, W.B., Chaudhari, N.S., Das, A., "Nonlinear Channel Equalization with QAM Signal Using Chebyshev Artificial Neural Networks," *Proceedings of International Joint Conference on Neural Networks*, pp. 3214-3219, July 31-August 4, 2005.

[28] Peng, M., Chrysostomos, L., Nikias, L., Proakis, J., "Adaptive Equalization with Neural Networks: New Multilayer Perceptron Structures and Their Evaluation," *IEEE Transactions on Neural Networks*, pp.301-304.

[29] Haykin, S. (1999). *Neural Networks,* 2nd edition. New Jersey, Prentice Hall.

[30] Guillaume, S., "Designing Fuzzy Inference Systems from Data: An Interpretability-Oriented Review," *IEEE Transactions on Fuzzy Systems,* Vol.9, No.3, June 2001.

[31] Karnik, N., Mendel, J., Liang, Q., "Type-2 Fuzzy Logic Systems," *IEEE Transactions on Fuzzy Systems,* Vol.7, No.6, December 1999.

[32] Feuring, T., "Fuzzy Neural Networks are Overlapping," *IEEE Transactions on Fuzzy Systems,* pp.1154-1160.

[33] Tang, X., Alouini, M.S., Goldsmith, A.J., "Effect of Channel Estimation Error on M-QAM BER Performance in Rayleigh Fading," *IEEE Transactions on Communications,* Vol.47, No.12, December 1999.

[34] Ryu, J.H., Lee, Y.H., "Combined Equalization and Nonlinear Distortion Cancellation for Transmission of QAM Signal in Fixed Wireless Channel," *IEEE Transactions on Communications, ICASSP,* pp.489-492, 2003.

[35] Choi, J., Lima de C.A.C, Haykin S., "Kalman Filter-Trained Recurrent Neural Equalizers for Time-Varying Channels," *IEEE Transactions on Communications,* Vol.53, No.3, 2005.

[36] Siu, S., Ho, C.H., Lee, C.M., "TSK-based Decision Feedback Equalizer Using an Evolutionary Algorithm Applied to QAM Communication Systems," *IEEE Transactions on Circuits and Systems,* Vol.52, No.9, pp.596-600, September 2005.

[37] Abiyev, R.H., Kaynak, O., Alshanableh, T., Mamedov, F., "A Type-2 Neuro-Fuzzy System Based on Clustering and Gradient Techniques Applied to System Identification and Channel Equalization," *Applied Soft Computing,* Vol.11, Issue 1, pp. 1396-1406, Jan. 2011.

[38] Abiyev, R.H., Mamedov, F., Alshanableh, T., "Neuro-Fuzzy System for Channel Noise Equalization," *International Multi-Conference in Computer Science & Computer Engineering*. *International Conference on Artificial Intelligence*. IC-AI'04, Las Vegas, Nevada, USA, June 21-24, 2004.

[39] Abiyev, R.H., Mamedov, F., Alshanableh, T., "Nonlinear Neuro-Fuzzy Network for Channel Equalization," *Advances in Soft Computing 41*, Springer-Verlag, Berlin Heidelberg, pp. 327-336, 2007. IFSA 2007 World Congress, Cancum, Mexico, June 18-21, 2007.

[40] http://www.radio-electronics.com/info/rf-technology-design/pm-phase modulation/8qam-16qam-32qam-128-qam-256-qam-php

[41] Haykin, S. (2001). *Communication Systems,* 4th edition. Hoboken, New Jersey, John Wiley & Sons, Inc.

[42] Yoon, D., Kyongkuk, C., Jinsock, L., "Bit Error Probability of M-ary Quadrature Amplitude Modulation," *IEEE Transactions on Communications,* pp.2422-2427, VTC 2000.

APPENDIX

MATLAB Files

```
function [y,yc,signal_var]=chanel_model(x,r1,fk);
disp('1-M=4')
disp('2-M=16')
M_QAM=input('Select the QAM constellation:');
switch M_QAM
    case 1, operation=1;  %M=4 Constellation
d=0;   chan_mod=1;
n=6000;
fk=3000;

aa1=1; aa2=0.5; aa3=0; fpt = fopen('fchan1_2_coef500.dat','r')
      [a1,a2,a3]=chanel_coef(aa1,aa2,aa3,n);
[a1]=fscanf(fpt,'%f\n',[n]);
[a2]=fscanf(fpt,'%f\n',[n]);
[a3]=fscanf(fpt,'%f\n',[n]);  a3=0;
       fclose(fpt);

xin_1=2*round(rand(n,1))-1;
M=4; L=sqrt(M); nsamp=1; % Oversampling rate
kk = log2(M); % Number of bits per symbol
mapping = [0 1 3 2].'; %mapping by using Gray coding
for r=1:n
if xin_1(r)==-1
    xin_1(r)=xin_1(r)+1;
end
if r==n
xsym_1 = bi2de(reshape(xin_1,kk,length(xin_1)/kk).','left-msb');
xsym_1 = mapping(xsym_1+1);
y_qam_1 = qammod(xsym_1,M);
y_re_2=real(y_qam_1); y_im_2=imag(y_qam_1);
end
end
y_qam_1_out1 = real(y_qam_1)-min(real(y_qam_1)); %Shift real part of
              %y_qam_1 values such that minimum value is zero
y_qam_1_out1 = y_qam_1_out1/max(abs(y_qam_1_out1));%Scale such that
              %maximum real y_qam_1 value is one
y_qam_1_out2 = imag(y_qam_1)-min(imag(y_qam_1)); %Shift imaginary part of
              %y_qam_1 values such that minimum value is zero
y_qam_1_out2 = y_qam_1_out2/max(abs(y_qam_1_out2)); %Scale such that
              %maximum imaginary y_qam_1 value is one
y_re=y_qam_1_out1; y_im=y_qam_1_out2;
for k=1:length(xin_1)/kk;
if y_re(k)==0
   y_re(k)=-0.75;
end
if y_im(k)==0
   y_im(k)=-0.75;
end
if y_re(k)==1
   y_re(k)=0.75;
```

```
end
if y_im(k)==1
   y_im(k)=0.75;
end
end
y_re=y_re;
y_qam = y_re_2+(y_im_2)*i;
%-------------------------------------------------------------------------
    case 2, operation=2;  %M=16 Constellation
d=0;   chan_mod=3;
n=12000;
fk=3000;

aa1=1; aa2=0.5; aa3=0; fpt = fopen('fchan3_2_coef1000.dat','r')
      [a1,a2,a3]=chanel_coef(aa1,aa2,aa3,n);

[a1]=fscanf(fpt,'%f\n',[n]);
[a2]=fscanf(fpt,'%f\n',[n]);
[a3]=fscanf(fpt,'%f\n',[n]); a3=0;
       fclose(fpt);

xin_1=2*round(rand(n,1))-1;
M=16; L=sqrt(M); nsamp=1; % Oversampling rate
kk = log2(M); % Number of bits per symbol
mapping = [0 1 3 2 4 5 7 6 12 13 15 14 8 9 11 10].'; %mapping by using
            %Gray coding
for r=1:n
if xin_1(r)==-1
    xin_1(r)=xin_1(r)+1;
end
if r==n
xsym_1 = bi2de(reshape(xin_1,kk,length(xin_1)/kk).','left-msb');
xsym_1 = mapping(xsym_1+1);
y_qam_1 = qammod(xsym_1,M);
y_re_2=real(y_qam_1); y_im_2=imag(y_qam_1);
end
end
y_qam_1_out1 = real(y_qam_1)-min(real(y_qam_1)); %Shift real part
              %of y_qam_1 values such that minimum value is zero
y_qam_1_out1 = y_qam_1_out1/max(abs(y_qam_1_out1)); %Scale such that
              %maximum real %  y_qam_1 value is one
y_qam_1_out2 = imag(y_qam_1)-min(imag(y_qam_1)); %Shift imaginary part of
              %y_qam_1 %  values such that minimum value is zero
y_qam_1_out2 = y_qam_1_out2/max(abs(y_qam_1_out2)); %Scale such that
              %maximum imaginary y_qam_1 value is one
y_re=y_qam_1_out1; y_im=y_qam_1_out2;
for k=1:length(xin_1)/kk;
if y_re(k)==0
   y_re(k)=-1;
end
if y_re(k)==1/3
   y_re(k)=-1/3;
end
if y_re(k)==2/3
  y_re(k)=1/3;
```

```
end
if y_re(k)==1
  y_re(k)=1;
end
if y_im(k)==0
   y_im(k)=-1;
end
if y_im(k)==1/3
   y_im(k)=-1/3;
end
if y_im(k)==2/3
   y_im(k)=1/3;
end
if y_im(k)==1
   y_im(k)=1;
end
end
y_qam = y_re_2+(y_im_2)*i;
%-------------------------------------------------------------------------
    otherwise disp('unknown')
end
%================================CHANNEL=======================
for k=1:(fk)
       if chan_mod==1
           if k==1
            yc_1(k)=0.3482*y_re(k-d);
            yc_2(k)=0.3482*y_im(k-d);
           elseif k==2
            yc_1(k)=0.3482*y_re(k-d)+0.8704*y_re(k-1-d);
            yc_2(k)=0.3482*y_im(k-d)+0.8704*y_im(k-1-d);
           end
           if k>2
            yc_1(k)=0.3482*y_re(k-d)+0.8704*y_re(k-1-d)+0.3482*y_re(k-2-d);
            yc_2(k)=0.3482*y_im(k-d)+0.8704*y_im(k-1-d)+0.3482*y_im(k-2-d);
           end;
       end

       if chan_mod==3
             if k==1;
              yc_1(k)=a1(k)*y_re(k-d)-0.9*(a1(k)*y_re(k-d))^3;
              yc_2(k)=a1(k)*y_im(k-d)-0.9*(a1(k)*y_im(k-d))^3;
             elseif k>1
              yc_1(k)=a1(k)*y_re(k-d)+a2(k)*y_re(k-1-d)-0.9*(a1(k)*…
                      y_re(k-d)+a2(k)*y_re(k-1-d))^3;
              yc_2(k)=a1(k)*y_im(k-d)+a2(k)*y_im(k-1-d)-0.9*(a1(k)*…
                      y_im(k-d)+a2(k)*y_im(k-1-d))^3;
             end
        end
end
%================================END OF CHANNEL================
min_error=0.0001;
noise_var=0.3
mean_yc_1=sum(yc_1)/fk;
mean_yc_2=sum(yc_2)/fk;
signal_var_yc_1=sum((yc_1-mean_yc_1).^2)/fk
```

```
signal_var_yc_2=sum((yc_2-mean_yc_2).^2)/fk
SNR_yc_1 = 10*log10(signal_var_yc_1/noise_var)
SNR_yc_2 = 10*log10(signal_var_yc_2/noise_var)

for k=1:(fk)
y_1(k)=awgn(yc_1(k),SNR_yc_1);
y_2(k)=awgn(yc_2(k),SNR_yc_2);
y(k)=y_1(k)+y_2(k)*i;
end
%==============Calculating the Equivalent(Reciprocal) Channel Output=======
for k=1:(fk)
   if operation==1
        if y_re(k)==-0.75
        percent_dev_real(k) = ((y_1(k)-y_re(k))/y_re(k));
        y_1_ob(k) = -1+percent_dev_real(k)*(-1);
        end
        if y_re(k)==0.75
        percent_dev_real(k) = ((y_1(k)-y_re(k))/y_re(k));
        y_1_ob(k) = 1+percent_dev_real(k)*(1);
        end

        if y_im(k)==-0.75
        percent_dev_imag(k) = ((y_2(k)-y_im(k))/y_im(k));
        y_2_ob(k) = -1+percent_dev_imag(k)*(-1);
        end
        if y_im(k)==0.75
        percent_dev_imag(k) = ((y_2(k)-y_im(k))/y_im(k));
        y_2_ob(k) = 1+percent_dev_imag(k)*(1);
        end
   end
%-------------------------------------------------------
   if operation==2
        if y_re(k)==-1
        percent_dev_real(k) = ((y_1(k)-y_re(k))/y_re(k));
        y_1_ob(k) = -3+percent_dev_real(k)*(-3);
        end
        if y_re(k)==(-1/3)
        percent_dev_real(k) = ((y_1(k)-y_re(k))/y_re(k));
        y_1_ob(k) = -1+percent_dev_real(k)*(-1);
        end
        if y_re(k)==(1/3)
        percent_dev_real(k) = ((y_1(k)-y_re(k))/y_re(k));
        y_1_ob(k) = 1+percent_dev_real(k)*(1);
        end
        if y_re(k)==1
        percent_dev_real(k) = ((y_1(k)-y_re(k))/y_re(k));
        y_1_ob(k) = 3+percent_dev_real(k)*(3);
        end

        if y_im(k)==-1
        percent_dev_imag(k) = ((y_2(k)-y_im(k))/y_im(k));
        y_2_ob(k) = -3+percent_dev_imag(k)*(-3);
        end
        if y_im(k)==(-1/3)
        percent_dev_imag(k) = ((y_2(k)-y_im(k))/y_im(k));
```

```
        y_2_ob(k) = -1+percent_dev_imag(k)*(-1);
        end
        if y_im(k)==(1/3)
        percent_dev_imag(k) = ((y_2(k)-y_im(k))/y_im(k));
        y_2_ob(k) = 1+percent_dev_imag(k)*(1);
        end
        if y_im(k)==1
        percent_dev_imag(k) = ((y_2(k)-y_im(k))/y_im(k));
        y_2_ob(k) = 3+percent_dev_imag(k)*(3);
        end
   end
%-------------------------------------------------------
end
for k=1:fk
yy(k)=y_1_ob(k)+y_2_ob(k)*i;
end
%=================================================================
h = scatterplot(y_qam(1:nsamp*(fk)),nsamp,0,'go');
hold on;
scatterplot(y_qam_1(1:(fk/kk)),1,0,'k*',h);
title('Channel Input');
legend('Channel Input','Signal Constellation');
axis([-15 15 -15 15]); % Set axis ranges.
hold off;

h = scatterplot(yy(1:nsamp*(fk/kk)),nsamp,0,'g*');
hold on;
scatterplot(y_qam_1(1:(fk/kk)),1,0,'k*',h);
title('Received Signal');
legend('Received Signal','Signal Constellation');
axis([-15 15 -15 15]);
hold off;

N1=4; MM=1;
if chan_mod==1
    N2=27;
elseif chan_mod==3
    N2=36;
end

for t=(N1+1):(fk);
        Data2(t,:)=[y_re(t) y_re(t-1) y_re(t-2) y_re(t-3) y_re(t-4)];
        Data3(t,:)=[y_im(t) y_im(t-1) y_im(t-2) y_im(t-3) y_im(t-4)];
        Data4(t,:)=[y_1(t) y_1(t-1) y_1(t-2) y_1(t-3) y_1(t-4)];
        Data5(t,:)=[y_2(t) y_2(t-1) y_2(t-2) y_2(t-3) y_2(t-4)];
end
global xin r1 fk
if operation==1 %for M=4 constellation
n=3000;      %test all signals
fk=3000;     %training signals
elseif operation==2 %for M=16 constellation
n=3000;     %test all signals
fk=3000;     %training signals
end
```

```
disp('0- Parameters Unit  ')
disp('1- Learning         ')
disp('2- Reading          ')
disp('3- Saving           ')
disp('4- Without Learning ')
disp('5- Noise            ')
disp('6- Exit             ')
num=input('Enter number:');

switch num

    case 0, regim=0;
        xin_1=2*round(rand(n,1))-1;
        replay=input('Generate new random noise y/n :','s');
        if replay=='y'
            [r1,sigma]=noise(n,noise_var);
        else
            r1(1:n)=0;
        end
aa1=1; aa2=0.5; aa3=0; fpt = fopen('fchan3_2_coef1000.dat','r')
      [a1,a2,a3]=chanel_coef(aa1,aa2,aa3,n);
[a1]=fscanf(fpt,'%f\n',[n]);
[a2]=fscanf(fpt,'%f\n',[n]);
[a3]=fscanf(fpt,'%f\n',[n]); a3=0;
       fclose(fpt);

              SNR_yc_1 = 10*log10(signal_var_yc_1/noise_var);
              SNR_yc_2 = 10*log10(signal_var_yc_2/noise_var);
sprintf('signal_var_yc_1=%f noise_var=%f SNR=%f',signal_var_yc_1,…
                           noise_var,SNR_yc_1)
              plot(xin_1)
              title('Transmitted Signals');
              pause
              plot(y)
              title('Received Signals');
              pause
              l=1:1:n;
              plot(l,a1,'b',l,a2,'r')
              title('a1 % a2 coefficients');
              pause
              norm_1=minmax(yc_1)
              norm_2=minmax(yc_2)
              delta_1=(abs(norm_1(1))+abs(norm_1(2)))/(N2-1)
              delta_2=(abs(norm_2(1))+abs(norm_2(2)))/(N2-1)
              c_1(1:N1,1:N2)=0;
              c_2(1:N1,1:N2)=0;
              N22=ceil(N2/2);
                 ii=0; kc_1(1:N22)=0; kc_2(1:N22)=0;
                 for j=1:N22
                     kc_1(j)=delta_1*(j-1)/j;
                     kc_2(j)=delta_2*(j-1)/j;
                     c_1(1:N1,j)=0-ii*kc_1(j);
                     c_2(1:N1,j)=0-ii*kc_2(j);
                     o_1(1:N1,j)=0.1+0.01*j;
                     o_2(1:N1,j)=0.1+0.01*j;
```

```
                    if(j>1)
                    c_1(1:N1,N22+j-1)=ii*kc_1(j);
                    c_2(1:N1,N22+j-1)=ii*kc_2(j);
                    o_1(1:N1,N22+j-1)=0.1+0.01*j;
                    o_2(1:N1,N22+j-1)=0.1+0.01*j;
                    end
                    ii=ii+1;
                end
                if (mod(N2,2)==0)
                    c_1(1:N1,N2)=c_1(1:N1,N2-1)+delta_1;
                    c_2(1:N1,N2)=c_2(1:N1,N2-1)+delta_2;
                    o_1(1:N1,N2)=0.2;
                    o_2(1:N1,N2)=0.2;
                end
                o_1(1:N1,N2)=0.2; o_1(1:N1,N22)=0.2;
                o_2(1:N1,N2)=0.2; o_2(1:N1,N22)=0.2;
                w1_1=0.02*rand(N2,N1);
                w2_1=0.02*rand(N2,N1);
                b_1=0.02*rand(N2,1);
                w1_2=0.02*rand(N2,N1);
                w2_2=0.02*rand(N2,N1);
                b_2=0.02*rand(N2,1);
                [c,o,w1,w2,b,xin,y,yc,r1]=chanel_n1(N1,N2,M,c,o,w1,w2,b);

case 1, regim=1;
    norm_1=minmax(yc_1)
    norm_2=minmax(yc_2)
          delta_1=(abs(norm_1(1))+abs(norm_1(2)))/(N2-1)
          delta_2=(abs(norm_2(1))+abs(norm_2(2)))/(N2-1)
          c_1(1:N1,1:N2)=0;
          c_2(1:N1,1:N2)=0;
          N22=ceil(N2/2);
             ii=0; kc_1(1:N22)=0; kc_2(1:N22)=0;
             for j=1:N22
                 kc_1(j)=delta_1*(j-1)/j;
                 kc_2(j)=delta_2*(j-1)/j;
                 c_1(1:N1,j)=0-ii*kc_1(j);
                 c_2(1:N1,j)=0-ii*kc_2(j);
                 o_1(1:N1,j)=0.1+0.01*j;
                 o_2(1:N1,j)=0.1+0.01*j;
                 if(j>1)
                     c_1(1:N1,N22+j-1)=ii*kc_1(j);
                     c_2(1:N1,N22+j-1)=ii*kc_2(j);
                     o_1(1:N1,N22+j-1)=0.1+0.01*j;
                     o_2(1:N1,N22+j-1)=0.1+0.01*j;
                 end
                 ii=ii+1;
             end
             if (mod(N2,2)==0) % N2-n.*2 where n=floor(N2./2) if 2~=0
                 c_1(1:N1,N2)=c_1(1:N1,N2-1)+delta_1;
                 c_2(1:N1,N2)=c_2(1:N1,N2-1)+delta_2;
                 o_1(1:N1,N2)=0.2;
                 o_2(1:N1,N2)=0.2;
             end
             o_1(1:N1,N2)=0.2; o_1(1:N1,N22)=0.2;
```

```
                o_2(1:N1,N2)=0.2; o_2(1:N1,N22)=0.2;
                w1_1=0.02*rand(N2,N1);
                w2_1=0.02*rand(N2,N1);
                b_1=0.02*rand(N2,1);
                w1_2=0.02*rand(N2,N1);
                w2_2=0.02*rand(N2,N1);
                b_2=0.02*rand(N2,1);

    case 2, fpt=fopen(fname,'r');
        [c]=fscanf(fpt,'%f \n',[N1,N2]);
        [o]=fscanf(fpt,'%f \n',[N1,N2]);
        [w1]=fscanf(fpt,'%f \n',[N2,N1]);
        [w2]=fscanf(fpt,'%f \n',[N2,N1]);
        [b]=fscanf(fpt,'%f \n',[N2]);
        pause
    case 3, fpt = fopen(fname,'w');
        fprintf(fpt,'%f %f %f \n',c,o,w1,w2,b)
        fclose(fpt);
        pause
        [c,o,w1,w2,b,xin,y,yc,r1]=chanel_n1(N1,N2,MM,c,o,w1,w2,b);
    case 4, regim=4;
        chaneld_n1(N1,N2,MM,c,o,w1,w2,b,n,fk,xin,y,r1,regim)
        [c,o,w1,w2,b,xin,y,yc,r1]=chanel_n1(N1,N2,MM,c,o,w1,w2,b)
    case 5,
    case 6,
    otherwise disp('unknown')
end

global xin r1 fk
if regim==1
    a_1=input('Enter new learning rate:'); %for in-phase (real part)
    a_2=input('Enter new learning rate:'); %for quadrature (imaginary part)
end

for t=(N1):fk;
    Data(t,:)=[xin_1(t) xin_1(t-1) xin_1(t-2) xin_1(t-3)];
    Data1(t,:)=[yc_1(t-3) yc_1(t-3) yc_1(t-1) yc_1(t)];
    if(t>N1)
        tN1=10*t+N1;
        z1(tN1)=yc_1(t);  z2(tN1)=yc_1(t-1);
    end
end;
Data;
ind1 = find(xin_1(:,1) > 0);
ind2 = find(xin_1(:,1) < 0);
for t=2:fk
    zz1(t)=yc_1(t);   zz2(t)=yc_1(t-1);
end
plot(z2,z1,'*');
xlabel('x(k) no noise'); ylabel('x(k-1)');
pause
%=========================================================
for t=(N1):fk;
    Data(t,:)=[xin_1(t) xin_1(t-1) xin_1(t-2) xin_1(t-3)];
    if(t>N1)
```

```
        tN1=10*t+N1;
        z1(tN1)=y_1(t);  z2(tN1)=y_1(t-1);
    end
end;
Data;
ind1 = find(xin_1(:,1) > 0);
ind2 = find(xin_1(:,1) < 0);
for t=2:fk
    zz1(t)=y_1(t);   zz2(t)=y_1(t-1);
end
plot(z2,z1,'*');
xlabel('x(k) with noise'); ylabel('x(k-1)'); %channel state with noise
pause

if regim==1
[c,o,w1,w2,b,a]=nefuz_train2_n1(regim,N1,N2,MM,Data,fk,epoch,a_1,a_2,c_1,…
               o_1,w1_1,w2_1,b_1);
end
[row,col]=size(Data2)
[cfind,ofind,wfind1,wfind2,bfind,a_1,a_2]=nefuz_train2_dfen1(regim,M,N1,…
      N2,MM,Data,Data2,Data3,Data4,Data5,fk,n,epoch,a_1,a_2,c_1,o_1,c_2,…
      o_2,w1_1,w2_1,w1_2,w2_2,b_1,b_2,SNR_yc_1,SNR_yc_2,noise_var,xin_1,…
      y_qam_1,y_qam_1_out1,y_qam_1_out2,nsamp,y_re,y_im,y_re_2,y_im_2,…
      y_1,y_2,min_error,operation,chan_mod,mapping);
for t=(N1+1):n;
    Data4(t,:)=[y_1(t) y_1(t-1) y_1(t-2) y_1(t-3) y_1(t-4)].';
    Data5(t,:)=[y_2(t) y_2(t-1) y_2(t-2) y_2(t-3) y_2(t-4)].';
    for p=1:col-1
        x_1(p)=Data4(t,p);
        x_2(p)=Data5(t,p);
        out_1(t)=Data2(t,p);
        out_2(t)=Data3(t,p);
    end
end;

function [a1,a2,a3]=chanel_coef(aa1,aa2,aa3,n);
function [r1,sigma]=noise(n,noise_var);

function
[ys_1,ys_2,summin_1,summin_2,minm_1,minm_2,m_1,m_2,ym_1,ym_2,net_1,net_2]=…
        nefuz_n1(N1,N2,MM,x_1,x_2,c_1,o_1,c_2,o_2,w1_1,w2_1,w1_2,w2_2,b_1,…
        b_2,min_error);
m_1(1:N1,1:N2)=0; minm_1(1:N2)=0;
for p=1:N1
    for j=1:N2
        m_1(p,j)=exp(-(((x_1(p)-c_1(p,j))/o_1(p,j))^2));
    if 1<=j
    minm_1(j)=m_1(p,j)*m_1(p,j);
    end
    end
end
summin_1=0;
for j=1:N2
    for p=1:N1
        if m_1(p,j)<=0.1
```

```
           m_1(p,j)=0;
        end
        if m_1(p,j)~=0
            if minm_1(j)>m_1(p,j)
                minm_1(j)=m_1(p,j);
            end
        end
    end
    if minm_1(j)>=10
        minm_1(j)=0;
    end
end
summin_1=sum(minm_1);
for j=1:N2
    net_1(j)=0;
    for p=1:N1
        net_1(j)=net_1(j)+x_1(p)*x_1(p)*w1_1(j,p)+x_1(p)*w2_1(j,p);
    end
    net_1(j)=net_1(j)+b_1(j);
end
for p=1:MM
    ym_1(p)=0;
    for j=1:N2
        ym_1(p)=ym_1(p)+net_1(j)*minm_1(j);
    end
    if summin_1==0
        ys_1(p)=0;
    else
        ys_1(p)=ym_1(p)/summin_1;
    end
end
%-------------------------------------------------------------------------
for p=1:N1
    for j=1:N2
        m_2(p,j)=exp(-(((x_2(p)-c_2(p,j))/o_2(p,j))^2));

    if 1<=j
    minm_2(j)=m_2(p,j)*m_2(p,j);
    end
    end
end
summin_2=0;
for j=1:N2
    for p=1:N1
        if m_2(p,j)<=0.1
           m_2(p,j)=0;
        end
        if m_2(p,j)~=0
            if minm_2(j)>m_2(p,j)
                minm_2(j)=m_2(p,j);
            end
        end
    end
    if minm_2(j)>=10
        minm_2(j)=0;
```

```
    end
end
summin_2=sum(minm_2);
for j=1:N2
    net_2(j)=0;
    for p=1:N1
        net_2(j)=net_2(j)+x_2(p)*x_2(p)*w1_2(j,p)+x_2(p)*w2_2(j,p);
    end
    net_2(j)=net_2(j)+b_2(j);
end
for p=1:MM
    ym_2(p)=0;
    for j=1:N2
        ym_2(p)=ym_2(p)+net_2(j)*minm_2(j);
    end
    if summin_2==0
        ys_2(p)=0;
    else
        ys_2(p)=ym_2(p)/summin_2;
    end
end

function[c,o,w1,w2,b,a_1,a_2]=nefuz_train2_n1(regim,N1,N2,MM,Data,fk,…
         epoch,a_1,a_2,c,o,w1,w2,b);
function
[c,o,w1,w2,b]=chaneld_dfen1(N1,N2,MM,c,o,w1,w2,b,n,fk,xin,y,r1,regim);
function [cfind_1,cfind_2,ofind_1,ofind_2,wfind1_1,wfind1_2,wfind2_1,…
         wfind2_2,bfind_1,bfind_2,a_1,a_2] = nefuz_train2_dfen1(regim,M,…
         N1,N2,MM,Data,Data2,Data3,Data4,Data5,fk,n,epoch,a_1,a_2,c_1,…
         o_1,c_2,o_2,w1_1,w2_1,w1_2,w2_2,b_1,b_2,SNR_yc_1,SNR_yc_2,…
         noise_var,xin_1,y_qam_1,y_qam_1_out1,y_qam_1_out2,nsamp,y_re,…
         y_im,y_re_2,y_im_2,y_1,y_2,min_error,operation,chan_mod,mapping);
w1o_1=w1_1; w2o_1=w2_1; bo_1=b_1;co_1=c_1;oo_1=o_1;
w1o_2=w1_2; w2o_2=w2_2; bo_2=b_2;co_2=c_2;oo_2=o_2;
[row,col]=size(Data2);
time_begin = cputime;
%------------TESTING THE SIGNALS=> CALCULATING BER AND MSE, PLOTTING
%  THE CONVERGENCE CURVES---
epoc1=1;
epoch=2000;
while epoc1<=epoch;

    t_er_1(epoc1)=0;
    t_er_1_d(epoc1)=0;
    num_err(epoc1)=0;
    t_er_4(epoc1)=0;
    t_er_2(epoc1)=0;
    t_er_3(epoc1)=0;
    for t=(N1):n;
        for p=1:col-1
            x_1(p)=Data4(t,p);
            x_2(p)=Data5(t,p);
            out_1(t)=Data2(t,p);
            out_2(t)=Data3(t,p);
        end
```

```
[ys_1,ys_2,summin_1,summin_2,minm_1,minm_2,m_1,m_2,ym_1,ym_2,net_1,net_2]=…
            nefuz_n1(N1,N2,MM,x_1,x_2,c_1,o_1,c_2,o_2,w1_1,w2_1,w1_2,w2_2,…
            b_1,b_2,min_error);
        er_1(t)=out_1(t)-ys_1;
        er_1_d(t)=0;
        er_2(t)=out_2(t)-ys_2;
        erh_1=er_1(t);
        erh_2=er_2(t);

        if t<=fk
        if((summin_1~=0)&&(abs(er_1(t))>min_error))
        [c_1,o_1,w1_1,w2_1,b_1,w1o_1,w2o_1,bo_1] = trainf_n1_1(N1,N2,MM,…
             x_1,erh_1,minm_1,summin_1,m_1,ys_1,a_1,c_1,o_1,w1_1,w2_1,…
             b_1,ym_1,net_1,w1o_1,w2o_1,bo_1);
        er_3(t)=er_1(t);
        t_er_2(epoc1)=t_er_2(epoc1)+(er_3(t)*er_3(t))/2;
        end
        if((summin_2~=0)&&(abs(er_2(t))>min_error))
        [c_2,o_2,w1_2,w2_2,b_2,w1o_2,w2o_2,bo_2] = trainf_n1_2(N1,N2,MM,…
                 x_2,erh_2,minm_2,summin_2,m_2,ys_2,a_2,c_2,…
                 o_2,w1_2,w2_2,b_2,ym_2,net_2,w1o_2,w2o_2,bo_2);
        er_4(t)=er_2(t);
        t_er_3(epoc1)=t_er_3(epoc1)+(er_4(t)*er_4(t))/2;
        end
        end
        t_er_2_3_average(epoc1)=(t_er_2(epoc1)+t_er_3(epoc1))/2;
        t_er_1_4_average(epoc1)=(t_er_1(epoc1)+t_er_4(epoc1))/2;
        t_er_1(epoc1)=t_er_1(epoc1)+er_1(t);
        t_er_1_d(epoc1)=t_er_1_d(epoc1)+er_1_d(t);
        t_er_4(epoc1)=t_er_4(epoc1)+er_2(t);

    end;

    ser_1=t_er_1(epoc1);
    if epoc1==1
        ser0_1=ser_1;
    end
    decay_1=(ser0_1-ser_1)/ser0_1;
    if(decay_1<=0)
        decay0_1=-1;  decay1_1=-1;
    end
    if((mod(epoc1,5)==0))
        if(decay0_1>0)
            a_1=a_1*1.01;
        end
        if(decay0_1<0)
            a_1=a_1/1.011;
        end
        decay1_1=decay0_1;
        decay0_1=1;
    end
    ser0_1=ser_1;
sprintf('%i ser_1=%f decay_1=%f a_1=%f summin_1=%f',epoc1,ser_1,…
                  decay_1,a_1,summin_1)
```

```
    if epoc1==1
        serfind_1=ser_1;
    end
    if ser_1<serfind_1
        serfind_1=ser_1; cfind_1=c_1; ofind_1=o_1; wfind1_1=w1_1;…
                  wfind2_1=w2_1;  bfind_1=b_1;
    end
 %-------------------------------------------------------------------------
    ser_2=t_er_4(epoc1);
    if epoc1==1
        ser0_2=ser_2;
    end
    decay_2=(ser0_2-ser_2)/ser0_2;
    if(decay_2<=0)
        decay0_2=-1;  decay1_2=-1;
    end
    if((mod(epoc1,5)==0))
        if(decay0_2>0)
            a_2=a_2*1.01;
        end
        if(decay0_2<0)
            a_2=a_2/1.011;
        end
        decay1_2=decay0_2;
        decay0_2=1;
    end
    ser0_2=ser_2;
sprintf('%i ser_2=%f decay_2=%f a_2=%f summin_2=%f',epoc1,ser_2,…
                  decay_2,a_2,summin_2)
    if epoc1==1
        serfind_2=ser_2;
    end
    if ser_2<serfind_2
        serfind_2=ser_2; cfind_2=c_2; ofind_2=o_2;   wfind1_2=w1_2;…
                  wfind2_2=w2_2; bfind_2=b_2;
    end

    t_er_2_total(epoc1)=log10(t_er_2(epoc1)/fk);
    t_er_3_total(epoc1)=log10(t_er_3(epoc1)/fk);
    t_er_2_3_average_total(epoc1)=log10(t_er_2_3_average(epoc1)/fk);
    epoc1=epoc1+1;
end;

plot(t_er_2_total);
xlabel('k'); ylabel('error (dB)');

plot(t_er_3_total);
xlabel('k'); ylabel('error (dB)');
pause

plot(t_er_2_3_average_total);
xlabel('k'); ylabel('error (dB)');
pause
time_end=cputime-time_begin
%================================================================
```

```
epoch=300
for t=(1):n;
epoc1=1;

    while epoc1<=epoch;
    t_er_1(epoc1)=0;
    t_er_4(epoc1)=0;
    t_er_1_2(t,epoc1)=0;
    t_er_4_2(t,epoc1)=0;
        for p=1:col-1
            x_1(p)=Data4(t,p);
            x_2(p)=Data5(t,p);
            out_1(t)=Data2(t,p);
            out_2(t)=Data3(t,p);
        end

[ys_1,ys_2,summin_1,summin_2,minm_1,minm_2,m_1,m_2,ym_1,ym_2,net_1,net_2]=…
    nefuz_n1(N1,N2,MM,x_1,x_2,c_1,o_1,c_2,o_2,w1_1,w2_1,w1_2,w2_2,b_1,b_2);
        er_1(t,epoc1)=y_re(t)-ys_1;
        er_2(t,epoc1)=y_im(t)-ys_2;
        er_1_2(t)=y_re(t)-ys_1;
        er_2_2(t)=y_im(t)-ys_2;
        erh_1=er_1(t,epoc1);
        erh_2=er_2(t,epoc1);

       if t<=fk
        if((summin_1~=0)&&(abs(er_1(t,epoc1))>min_error))

[c_1,o_1,w1_1,w2_1,b_1,w1o_1,w2o_1,bo_1]=trainf_n1_1(N1,N2,MM,x_1,erh_1,…
                minm_1,summin_1,m_1,ys_1,a_1,c_1,o_1,w1_1,w2_1,b_1,ym_1,…
                net_1,w1o_1,w2o_1,bo_1);
        end

        if((summin_2~=0)&&(abs(er_2(t,epoc1))>min_error))
        [c_2,o_2,w1_2,w2_2,b_2,w1o_2,w2o_2,bo_2]=trainf_n1_2(N1,N2,MM,x_2,…
                    erh_2,minm_2,summin_2,m_2,ys_2,a_2,c_2,o_2,w1_2,w2_2,…
                    b_2,ym_2,net_2,w1o_2,w2o_2,bo_2);
        end
       end
        eq_1(t)=ys_1;
        eq_2(t)=ys_2;
        t_er_1(epoc1)=t_er_1(epoc1)+er_1_2(t);
        t_er_4(epoc1)=t_er_4(epoc1)+er_2_2(t);

    ser_1=t_er_1(epoc1);
    if epoc1==1
        ser0_1=ser_1;
    end
    decay_1=(ser0_1-ser_1)/ser0_1;
    if(decay_1<=0)
        decay0_1=-1;  decay1_1=-1;
    end
    if((mod(epoc1,5)==0))
        if(decay0_1>0)
            a_1=a_1*1.01;
```

```
        end
        if(decay0_1<0)
            a_1=a_1/1.011;
        end
        decay1_1=decay0_1;
        decay0_1=1;
    end
    ser0_1=ser_1;
sprintf('signal=%f %i ser_1=%f decay_1=%f a_1=%f summin_1=%f',…
               t,epoc1,ser_1,decay_1,a_1,summin_1)
    if epoc1==1
        serfind_1=ser_1;
    end
    if ser_1<serfind_1
        serfind_1=ser_1;  cfind_1=c_1;  ofind_1=o_1;  wfind1_1=…
                  w1_1;  wfind2_1=w2_1;  bfind_1=b_1;
    end
 %-------------------------------------------------------------------------
    ser_2=t_er_4(epoc1);
    if epoc1==1
        ser0_2=ser_2;
    end
    decay_2=(ser0_2-ser_2)/ser0_2;
    if(decay_2<=0)
        decay0_2=-1;  decay1_2=-1;
    end
    if((mod(epoc1,5)==0))
        if(decay0_2>0)
            a_2=a_2*1.01;
        end
        if(decay0_2<0)
            a_2=a_2/1.011;
        end
        decay1_2=decay0_2;
        decay0_2=1;
    end
    ser0_2=ser_2;
sprintf('signal=%f %i ser_2=%f decay_2=%f a_2=%f summin_2=%f',…
               t,epoc1,ser_2,decay_2,a_2,summin_2)
    if epoc1==1
        serfind_2=ser_2;
    end
    if ser_2<serfind_2
        serfind_2=ser_2;  cfind_2=c_2;  ofind_2=o_2;  wfind1_2=…
                  w1_2;  wfind2_2=w2_2;  bfind_2=b_2;
    end
%-------------------------------------------------------------------------
    epoc1=epoc1+1;
    end
end;

time_end=cputime-time_begin
%=================================================================
        if operation==1
          if chan_mod==1 %if channel is linear
```

```
           save Q4_linear1.txt BER_3 -append -ascii
           save Q4_linear2.txt SNR_yc_1 -append -ascii
           load Q4_linear1.txt; load Q4_linear2.txt;
           [r_Q4_lin_1,c_Q4_lin_1]=size(Q4_linear1);
           [r_Q4_lin_2,c_Q4_lin_2]=size(Q4_linear2);
          elseif chan_mod==3 %if channel is non-linear
           save Q4_nonlinear1.txt BER_3 -append -ascii
           save Q4_nonlinear2.txt SNR_yc_1 -append -ascii
           load Q4_nonlinear1.txt; load Q4_nonlinear2.txt;
           [r_Q4_nonlin_1,c_Q4_nonlin_1]=size(Q4_nonlinear1);
           [r_Q4_nonlin_2,c_Q4_nonlin_2]=size(Q4_nonlinear2);
          end
        end
        %-------------------------------------------------------
        if operation==2 %When M=16 Constellation
          if chan_mod==1 %if channel is linear
           save Q16_linear1.txt BER_3 -append -ascii
           save Q16_linear2.txt SNR_yc_1 -append -ascii
           load Q16_linear1.txt; load Q16_linear2.txt;
           [r_Q16_lin_1,c_Q16_lin_1]=size(Q16_linear1);
           [r_Q16_lin_2,c_Q16_lin_2]=size(Q16_linear2);
          elseif chan_mod==3 %if channel is non-linear
           save Q16_nonlinear1.txt BER_3 -append -ascii
           save Q16_nonlinear2.txt SNR_yc_1 -append -ascii
           load Q16_nonlinear1.txt; load Q16_nonlinear2.txt;
           [r_Q16_nonlin_1,c_Q16_nonlin_1]=size(Q16_nonlinear1);
           [r_Q16_nonlin_2,c_Q16_nonlin_2]=size(Q16_nonlinear2);
          end
        end
%-------------------------------------------------------------------------
for int=1:1:120
    if r_1==10*int
            for p=1:1:10
            data1_1(p)=data1(10*(int-1)+p);
            data2_2(p)=data2(10*(int-1)+p);
            end
        BERavg=sum(data1_1)/10;%average BER of 10 times(independent trials)
        SNRavg=sum(data2_2)/10;%average SNR of 10 times(independent trials)
        save data3.txt BERavg -append -ascii
        save data4.txt SNRavg -append -ascii

    elseif  r_1>10*int
            for p=1:1:10
            data1_1(p)=data1(10*(int-1)+p);
            data2_2(p)=data2(10*(int-1)+p);
            end
        BERavg=sum(data1_1)/10;%average BER of 10 times(independent trials)
        SNRavg=sum(data2_2)/10;%average SNR of 10 times(independent trials)
        load data3.txt; load data4.txt;
        [r_3,c_3]=size(data3); [r_4,c_4]=size(data4);
        equality_1=0; equality_2=0;
         for nn=1:1:r_3;
           if floor((10^6)*data3(nn))==floor((10^6)*BERavg)
               equality_1=1;
           end
```

```
        if floor((10^6)*data4(nn))==floor((10^6)*SNRavg)
            equality_2=1;
        end
       end

      if r_3<floor((r_1)/10) && equality_1~=1
         save data3.txt BERavg -append -ascii
      end
      if r_4<floor((r_2)/10) && equality_2~=1
         save data4.txt SNRavg -append -ascii
      end
      load data3.txt; load data4.txt;
      [r_3,c_3]=size(data3); [r_4,c_4]=size(data4);
  end
  %-------------------------------------------------------------------------
if operation==1 %When M=4 Constellation
 if chan_mod==1 %if channel is linear
  if r_Q4_lin_1==10*int
          for p=1:1:10
          Q4_linear1_1(p)=Q4_linear1(10*(int-1)+p);
          Q4_linear2_2(p)=Q4_linear2(10*(int-1)+p);
          end
      BERavg=sum(Q4_linear1_1)/10;
      SNRavg=sum(Q4_linear2_2)/10;
      save Q4_linear3.txt BERavg -append -ascii
      save Q4_linear4.txt SNRavg -append -ascii

  elseif  r_Q4_lin_1>10*int
          for p=1:1:10
          Q4_linear1_1(p)=Q4_linear1(10*(int-1)+p);
          Q4_linear2_2(p)=Q4_linear2(10*(int-1)+p);
          end
      BERavg=sum(Q4_linear1_1)/10;
      SNRavg=sum(Q4_linear2_2)/10;
      load Q4_linear3.txt; load Q4_linear4.txt;
      [r_Q4_lin_3,c_Q4_lin_3]=size(Q4_linear3);
      [r_Q4_lin_4,c_Q4_lin_4]=size(Q4_linear4);
      equality_1=0; equality_2=0;
       for nn=1:1:r_Q4_lin_3;
        if floor((10^6)*Q4_linear3(nn))==floor((10^6)*BERavg)
            equality_1=1;
        end
        if floor((10^6)*Q4_linear4(nn))==floor((10^6)*SNRavg)
            equality_2=1;
        end
       end

      if r_Q4_lin_3<floor((r_Q4_lin_1)/10) && equality_1~=1
         save Q4_linear3.txt BERavg -append -ascii
      end
      if r_Q4_lin_4<floor((r_Q4_lin_2)/10) && equality_2~=1
         save Q4_linear4.txt SNRavg -append -ascii
      end
      load Q4_linear3.txt; load Q4_linear4.txt;
      [r_Q4_lin_3,c_Q4_lin_3]=size(Q4_linear3);
```

```
        [r_Q4_lin_4,c_Q4_lin_4]=size(Q4_linear4);
     end
    end
    %-------------------------------------------------------------------
    if chan_mod==3 %if channel is non-linear
     if r_Q4_nonlin_1==10*int
             for p=1:1:10
             Q4_nonlinear1_1(p)=Q4_nonlinear1(10*(int-1)+p);
             Q4_nonlinear2_2(p)=Q4_nonlinear2(10*(int-1)+p);
             end
         BERavg=sum(Q4_nonlinear1_1)/10;
         SNRavg=sum(Q4_nonlinear2_2)/10;
         save Q4_nonlinear3.txt BERavg -append -ascii
         save Q4_nonlinear4.txt SNRavg -append -ascii

     elseif  r_Q4_nonlin_1>10*int
             for p=1:1:10
             Q4_nonlinear1_1(p)=Q4_nonlinear1(10*(int-1)+p);
             Q4_nonlinear2_2(p)=Q4_nonlinear2(10*(int-1)+p);
             end
         BERavg=sum(Q4_nonlinear1_1)/10;
         SNRavg=sum(Q4_nonlinear2_2)/10;
         load Q4_nonlinear3.txt; load Q4_nonlinear4.txt;
         [r_Q4_nonlin_3,c_Q4_nonlin_3]=size(Q4_nonlinear3);
         [r_Q4_nonlin_4,c_Q4_nonlin_4]=size(Q4_nonlinear4);
         equality_1=0; equality_2=0;
          for nn=1:1:r_Q4_nonlin_3;
            if floor((10^6)*Q4_nonlinear3(nn))==floor((10^6)*BERavg)
                equality_1=1;
            end
            if floor((10^6)*Q4_nonlinear4(nn))==floor((10^6)*SNRavg)
                equality_2=1;
            end
          end

         if r_Q4_nonlin_3<floor((r_Q4_nonlin_1)/10) && equality_1~=1
            save Q4_nonlinear3.txt BERavg -append -ascii
         end
         if r_Q4_nonlin_4<floor((r_Q4_nonlin_2)/10) && equality_2~=1
            save Q4_nonlinear4.txt SNRavg -append -ascii
         end
         load Q4_nonlinear3.txt; load Q4_nonlinear4.txt;
         [r_Q4_nonlin_3,c_Q4_nonlin_3]=size(Q4_nonlinear3);
         [r_Q4_nonlin_4,c_Q4_nonlin_4]=size(Q4_nonlinear4);
     end
    end
   end %ends operation==1
%-------------------------------------------------------------------------
   if operation==2 %When M=16 Constellation
    if chan_mod==1 %if channel is linear
     if r_Q16_lin_1==10*int
             for p=1:1:10
             Q16_linear1_1(p)=Q16_linear1(10*(int-1)+p);
             Q16_linear2_2(p)=Q16_linear2(10*(int-1)+p);
             end
```

```
    BERavg=sum(Q16_linear1_1)/10;
    SNRavg=sum(Q16_linear2_2)/10;
    save Q16_linear3.txt BERavg -append -ascii
    save Q16_linear4.txt SNRavg -append -ascii

 elseif  r_Q16_lin_1>10*int
        for p=1:1:10
        Q16_linear1_1(p)=Q16_linear1(10*(int-1)+p);
        Q16_linear2_2(p)=Q16_linear2(10*(int-1)+p);
        end
    BERavg=sum(Q16_linear1_1)/10;
    SNRavg=sum(Q16_linear2_2)/10;
    load Q16_linear3.txt; load Q16_linear4.txt;
    [r_Q16_lin_3,c_Q16_lin_3]=size(Q16_linear3);
    [r_Q16_lin_4,c_Q16_lin_4]=size(Q16_linear4);
    equality_1=0; equality_2=0;
     for nn=1:1:r_Q16_lin_3;
       if floor((10^6)*Q16_linear3(nn))==floor((10^6)*BERavg)
           equality_1=1;
       end
       if floor((10^6)*Q16_linear4(nn))==floor((10^6)*SNRavg)
           equality_2=1;
       end
     end

    if r_Q16_lin_3<floor((r_Q16_lin_1)/10) && equality_1~=1
       save Q16_linear3.txt BERavg -append -ascii
    end
    if r_Q16_lin_4<floor((r_Q16_lin_2)/10) && equality_2~=1
       save Q16_linear4.txt SNRavg -append -ascii
    end
    load Q16_linear3.txt; load Q16_linear4.txt;
    [r_Q16_lin_3,c_Q16_lin_3]=size(Q16_linear3);
    [r_Q16_lin_4,c_Q16_lin_4]=size(Q16_linear4);
 end
end
%-----------------------------------------------------------------------
if chan_mod==3 %if channel is non-linear
 if r_Q16_nonlin_1==10*int
        for p=1:1:10
        Q16_nonlinear1_1(p)=Q16_nonlinear1(10*(int-1)+p);
        Q16_nonlinear2_2(p)=Q16_nonlinear2(10*(int-1)+p);
        end
    BERavg=sum(Q16_nonlinear1_1)/10;%average BER of 10 times
    SNRavg=sum(Q16_nonlinear2_2)/10;%average SNR of 10 times
    save Q16_nonlinear3.txt BERavg -append -ascii
    save Q16_nonlinear4.txt SNRavg -append -ascii

 elseif  r_Q16_nonlin_1>10*int
        for p=1:1:10
        Q16_nonlinear1_1(p)=Q16_nonlinear1(10*(int-1)+p);
        Q16_nonlinear2_2(p)=Q16_nonlinear2(10*(int-1)+p);
        end
    BERavg=sum(Q16_nonlinear1_1)/10; %average BER of 10 times
    SNRavg=sum(Q16_nonlinear2_2)/10; %average SNR of 10 times
```

```
        load Q16_nonlinear3.txt; load Q16_nonlinear4.txt;
        [r_Q16_nonlin_3,c_Q16_nonlin_3]=size(Q16_nonlinear3);
        [r_Q16_nonlin_4,c_Q16_nonlin_4]=size(Q16_nonlinear4);
        equality_1=0; equality_2=0;
         for nn=1:1:r_Q16_nonlin_3;
           if floor((10^6)*Q16_nonlinear3(nn))==floor((10^6)*BERavg)
               equality_1=1;
           end
           if floor((10^6)*Q16_nonlinear4(nn))==floor((10^6)*SNRavg)
               equality_2=1;
           end
         end

        if r_Q16_nonlin_3<floor((r_Q16_nonlin_1)/10) && equality_1~=1
           save Q16_nonlinear3.txt BERavg -append -ascii
        end
        if r_Q16_nonlin_4<floor((r_Q16_nonlin_2)/10) && equality_2~=1
           save Q16_nonlinear4.txt SNRavg -append -ascii
        end
        load Q16_nonlinear3.txt; load Q16_nonlinear4.txt;
        [r_Q16_nonlin_3,c_Q16_nonlin_3]=size(Q16_nonlinear3);
        [r_Q16_nonlin_4,c_Q16_nonlin_4]=size(Q16_nonlinear4);
    end
   end
  end %ends operation==2
%-------------------------------------------------------------------------
end %ends int=1:1:120 loop
%-------------------------------------------------------------------------
     if operation==1
        load Q4_linear1.txt; load Q4_linear2.txt;
        [r_Q4_lin_1,c_Q4_lin_1]=size(Q4_linear1);
        [r_Q4_lin_2,c_Q4_lin_2]=size(Q4_linear2);
       if r_Q4_lin_1>=10
        load Q4_linear3.txt; load Q4_linear4.txt;
        fpt_Q4_linear3 = fopen('Q4_linear3.txt','r');
        fpt_Q4_linear4 = fopen('Q4_linear4.txt','r');
        [BER_Q4_linear3]=fscanf(fpt_Q4_linear3,'%f\n',[ran 3]);
        [SNR_Q4_linear4]=fscanf(fpt_Q4_linear4,'%f\n',[ran 3]);
        fclose(fpt_Q4_linear3); fclose(fpt_Q4_linear4);
        semilogy(SNR_Q4_linear4,BER_Q4_linear3,'*--')
        axis ([0 8.05 0.0001 1]) %axis tight
        grid on
        xlabel('SNR_Q4_linear (dB)'); ylabel('BER_Q4_linear ');
        title('SNR vs BER');
        legend('Linear Channel M=4',1);
        pause
        hold on;
       end
        load Q4_nonlinear1.txt; load Q4_nonlinear2.txt;
        [r_Q4_nonlin_1,c_Q4_nonlin_1]=size(Q4_nonlinear1);
        [r_Q4_nonlin_2,c_Q4_nonlin_2]=size(Q4_nonlinear2);
       if r_Q4_nonlin_1>=10
        load Q4_nonlinear3.txt; load Q4_nonlinear4.txt;
        fpt_Q4_nonlinear3 = fopen('Q4_nonlinear3.txt','r');
        fpt_Q4_nonlinear4 = fopen('Q4_nonlinear4.txt','r');
```

```
        [BER_Q4_nonlinear3]=fscanf(fpt_Q4_nonlinear3,'%f\n',[ran 3]);
        [SNR_Q4_nonlinear4]=fscanf(fpt_Q4_nonlinear4,'%f\n',[ran 3]);
        fclose(fpt_Q4_nonlinear3); fclose(fpt_Q4_nonlinear4);
        semilogy(SNR_Q4_nonlinear4,BER_Q4_nonlinear3,'^-.r')
        axis ([0 8.05 0.0001 1]) %axis tight
        grid on
        xlabel('SNR_Q4_nonlinear (dB)'); ylabel('BER_Q4_nonlinear ');
        title('SNR vs BER');
        legend('Linear Channel M=4','Non-linear Channel M=4',1);
        pause
      end
    end
%--------------------------------------------------------------------------
    if operation==2
       load Q16_linear1.txt; load Q16_linear2.txt;
       [r_Q16_lin_1,c_Q16_lin_1]=size(Q16_linear1);
       [r_Q16_lin_2,c_Q16_lin_2]=size(Q16_linear2);
      if r_Q16_lin_1>=10
       load Q16_linear3.txt; load Q16_linear4.txt;
       fpt_Q16_linear3 = fopen('Q16_linear3.txt','r');
       fpt_Q16_linear4 = fopen('Q16_linear4.txt','r');
       [BER_Q16_linear3]=fscanf(fpt_Q16_linear3,'%f\n',[ran 3]);
       [SNR_Q16_linear4]=fscanf(fpt_Q16_linear4,'%f\n',[ran 3]);
       fclose(fpt_Q16_linear3); fclose(fpt_Q16_linear4);
       semilogy(SNR_Q16_linear4,BER_Q16_linear3,'*--')
       axis ([0 10.05 0.001 1])
       grid on
       xlabel('SNR_Q16_linear (dB)'); ylabel('BER_Q16_linear ');
       title('SNR vs BER');
       legend('Linear Channel M=16',1);
       pause
       hold on;
      end
       load Q16_nonlinear1.txt; load Q16_nonlinear2.txt;
       [r_Q16_nonlin_1,c_Q16_nonlin_1]=size(Q16_nonlinear1);
       [r_Q16_nonlin_2,c_Q16_nonlin_2]=size(Q16_nonlinear2);
      if r_Q16_nonlin_1>=10
       load Q16_nonlinear3.txt; load Q16_nonlinear4.txt;
       fpt_Q16_nonlinear3 = fopen('Q16_nonlinear3.txt','r');
       fpt_Q16_nonlinear4 = fopen('Q16_nonlinear4.txt','r');
       [BER_Q16_nonlinear3]=fscanf(fpt_Q16_nonlinear3,'%f\n',[ran 3]);
       [SNR_Q16_nonlinear4]=fscanf(fpt_Q16_nonlinear4,'%f\n',[ran 3]);
       fclose(fpt_Q16_nonlinear3); fclose(fpt_Q16_nonlinear4);
       semilogy(SNR_Q16_nonlinear4,BER_Q16_nonlinear3,'^-.r')
       axis ([0 10.05 0.001 1])
       grid on
       xlabel('SNR_Q16_nonlinear (dB)'); ylabel('BER_Q16_nonlinear ');
       title('SNR vs BER');
       legend('Linear Channel M=16','Non-linear Channel M=16',1);
       pause
      end
    end
%--------------------------------------------------------------------------
 %Comparison of 4-QAM and 16-QAM with each other and with theoretical BER
       M_1=4; M_2=16;
```

```
        for th=0:1:10
Theoretical_BER_16(th+1)=(2/log2(M_2))*((sqrt(M_2)-1)/sqrt(M_2))*…
                        (1-erf(sqrt(3*log2(M_2)/(2*(M_2-1)))*sqrt(th)))
        end
        th=0:1:10
        lengthx=length(th)

        load Q16_linear3.txt; load Q16_linear4.txt;
        fpt_Q16_linear3 = fopen('Q16_linear3.txt','r');
        fpt_Q16_linear4 = fopen('Q16_linear4.txt','r');
        [BER_Q16_linear3]=fscanf(fpt_Q16_linear3,'%f\n',[ran 3]);
        [SNR_Q16_linear4]=fscanf(fpt_Q16_linear4,'%f\n',[ran 3]);
        fclose(fpt_Q16_linear3); fclose(fpt_Q16_linear4);
        semilogy(SNR_Q16_linear4,BER_Q16_linear3,'s--')
        axis ([0 10.05 0.000001 1])
        grid on
        xlabel('SNR_Q16_linear (dB)'); ylabel('BER_Q16_linear ');
        title('SNR vs BER');
        legend('Linear Channel M=16',1);
        pause
        hold on;

        load Q16_nonlinear3.txt; load Q16_nonlinear4.txt;
        fpt_Q16_nonlinear3 = fopen('Q16_nonlinear3.txt','r');
        fpt_Q16_nonlinear4 = fopen('Q16_nonlinear4.txt','r');
        [BER_Q16_nonlinear3]=fscanf(fpt_Q16_nonlinear3,'%f\n',[ran 3]);
        [SNR_Q16_nonlinear4]=fscanf(fpt_Q16_nonlinear4,'%f\n',[ran 3]);
        fclose(fpt_Q16_nonlinear3); fclose(fpt_Q16_nonlinear4);
        semilogy(SNR_Q16_nonlinear4,BER_Q16_nonlinear3,'o-.r')
        axis ([0 10.05 0.000001 1])
        grid on
        xlabel('SNR_Q16_nonlinear (dB)'); ylabel('BER_Q16_nonlinear ');
        title('SNR vs BER');
        legend('Linear Channel M=4','Non-linear Channel M=4',…
               'Linear Channel M=16','Non-linear Channel M=16',1);
        pause

        Theoretical_2nd_BER_4(1)=7.75*10^-2;
        Theoretical_2nd_BER_4(2)=5.85*10^-2;
        Theoretical_2nd_BER_4(3)=3.85*10^-2;
        Theoretical_2nd_BER_4(4)=2.35*10^-2;
        Theoretical_2nd_BER_4(5)=1.3*10^-2;
        Theoretical_2nd_BER_4(6)=6.5*10^-3;
        Theoretical_2nd_BER_4(7)=2.5*10^-3;
        Theoretical_2nd_BER_4(8)=8.5*10^-4;
        Theoretical_2nd_BER_4(9)=1.95*10^-4;
        Theoretical_2nd_BER_4(10)=3.45*10^-5;
        Theoretical_2nd_BER_4(11)=3.8*10^-6;
        semilogy(th,Theoretical_2nd_BER_4,'k-o')
        pause
        hold on;
        Theoretical_2nd_BER_16(1)=1.485*10^-1;
        Theoretical_2nd_BER_16(2)=1.308*10^-1;
        Theoretical_2nd_BER_16(3)=1.108*10^-1;
        Theoretical_2nd_BER_16(4)=8.738*10^-2;
```

```
        Theoretical_2nd_BER_16(5)=6.568*10^-2;
        Theoretical_2nd_BER_16(6)=4.635*10^-2;
        Theoretical_2nd_BER_16(7)=2.97*10^-2;
        Theoretical_2nd_BER_16(8)=1.75*10^-2;
        Theoretical_2nd_BER_16(9)=9.7*10^-3;
        Theoretical_2nd_BER_16(10)=4.94*10^-3;
        Theoretical_2nd_BER_16(11)=1.98*10^-3;
        semilogy(th,Theoretical_2nd_BER_16,'k-s')
        legend('Linear Channel M=4','Non-linear Channel M=4',…
               'Linear Channel M=16','Non-linear Channel M=16',…
               'Theoretical BER of 4-QAM','Theoretical BER of 16-QAM',3);
        pause
        hold on;
%=================================================================
if (epoc1==epoch || epoc1==epoch+1);
 %-------------Calculating the Equivalent(Reciprocal) Equalizer…
              Output (Denormalization)--------------------------
   for k=1:n
      if operation==1
        if y_re(k)==-0.75
        percent_dev_real(k) = ((eq_1(k)-y_re(k))/y_re(k));
        eq_1(k) = -1+percent_dev_real(k)*(-1);
        end
        if y_re(k)==0.75
        percent_dev_real(k) = ((eq_1(k)-y_re(k))/y_re(k));
        eq_1(k) = 1+percent_dev_real(k)*(1);
        end

        if y_im(k)==-0.75
        percent_dev_imag(k) = ((eq_2(k)-y_im(k))/y_im(k));
        eq_2(k) = -1+percent_dev_imag(k)*(-1);
        end
        if y_im(k)==0.75
        percent_dev_imag(k) = ((eq_2(k)-y_im(k))/y_im(k));
        eq_2(k) = 1+percent_dev_imag(k)*(1);
        end
     end
     %-----------------------------------------------------
     if operation==2
        if y_re(k)==-1
        percent_dev_real(k) = ((eq_1(k)-y_re(k))/y_re(k));
        eq_1(k) = -3+percent_dev_real(k)*(-3);
        end
        if y_re(k)==(-1/3)
        percent_dev_real(k) = ((eq_1(k)-y_re(k))/y_re(k));
        eq_1(k) = -1+percent_dev_real(k)*(-1);
        end
        if y_re(k)==(1/3)
        percent_dev_real(k) = ((eq_1(k)-y_re(k))/y_re(k));
        eq_1(k) = 1+percent_dev_real(k)*(1);
        end
        if y_re(k)==1
        percent_dev_real(k) = ((eq_1(k)-y_re(k))/y_re(k));
        eq_1(k) = 3+percent_dev_real(k)*(3);
        end
```

```
        if y_im(k)==-1
        percent_dev_imag(k) = ((eq_2(k)-y_im(k))/y_im(k));
        eq_2(k) = -3+percent_dev_imag(k)*(-3);
        end
        if y_im(k)==(-1/3)
        percent_dev_imag(k) = ((eq_2(k)-y_im(k))/y_im(k));
        eq_2(k) = -1+percent_dev_imag(k)*(-1);
        end
        if y_im(k)==(1/3)
        percent_dev_imag(k) = ((eq_2(k)-y_im(k))/y_im(k));
        eq_2(k) = 1+percent_dev_imag(k)*(1);
        end
        if y_im(k)==1
        percent_dev_imag(k) = ((eq_2(k)-y_im(k))/y_im(k));
        eq_2(k) = 3+percent_dev_imag(k)*(3);
        end
     end
     %-------------------------------------------------------
   end

    for t=(1):n;
                if(t>N1)
                tN1=10*t+N1;
            v1_1(t)=eq_1(t);
            v1_2(t)=eq_2(t);
                end
    end

v=v1_1+(v1_2)*i
h = scatterplot(v(1:nsamp*(fk)),nsamp,0,'g*');
hold on;
scatterplot(y_qam_1(1:(fk)),1,0,'k*',h);
title('Received Signal');
legend('Received Signal','Signal Constellation');
axis([-15 15 -15 15]); % Set axis ranges.
hold off;
pause

%% Demodulation
% Demodulate signal
zsym = qamdemod(v,M);

% A. Define a vector that inverts the mapping operation.
[dummy demapping] = sort(mapping);
% Initially, demapping has values between 1 and M.
% Subtract 1 to obtain values between 0 and M-1.
demapping = demapping - 1;

% B. Map between Gray and binary coding.
zsym = demapping(zsym+1);

% C. Do ordinary decimal-to-binary mapping.
z = de2bi(zsym,'left-msb');
% Convert z from a matrix to a vector.
z = reshape(z.',prod(size(z)),1);
```

```
%% BER Computation
% Compare x and z to obtain the number of errors and
% the bit error rate.
[number_of_errors,bit_error_rate] = biterr(xin_1,z)
pause

end
%=====================================================================
function[c_1,o_1,w1_1,w2_1,b_1,w1o_1,w2o_1,bo_1] = trainf_n1_1…
        (N1,N2,M,x_1,erh_1,minm_1,summin_1,m_1,ys_1,a_1,c_1,o_1,…
         w1_1,w2_1,b_1,ym_1,net_1,w1o_1,w2o_1,bo_1);
a2=0.2;   %a2 is the momentum
w2o1_1=w2_1;
w1o1_1=w1_1;
bo1_1=b_1;
co1_1=c_1;
oo1_1=o_1;
for j=1:N2
    for p=1:N1
        w1_1(j,p)=w1_1(j,p)+a_1*erh_1*x_1(p)*x_1(p)*…
                  minm_1(j)/summin_1+a2*(w1_1(j,p)-w1o_1(j,p));
        w2_1(j,p)=w2_1(j,p)+a_1*erh_1*x_1(p)*x_1(p)*…
                  minm_1(j)/summin_1+a2*(w2_1(j,p)-w2o_1(j,p));
    end
end

for j=1:N2
    b_1(j)=b_1(j)+a_1*erh_1*minm_1(j)/summin_1+a2*(b_1(j)-bo_1(j));
end
sec_1=1;
if sec_1==1
    for j=1:N2
        kw(j)=(net_1(j)-ys_1)/summin_1; %using defuzification
    end

    for p=1:N1
        for j=1:N2
            c_1(p,j)=c_1(p,j)+a_1*erh_1*kw(j)*minm_1(j)*2*…
                     (x_1(p)-c_1(p,j))/(o_1(p,j)^2);
            o_1(p,j)=o_1(p,j)+a_1*erh_1*kw(j)*minm_1(j)*2*…
                     ((x_1(p)-c_1(p,j))^2)/(o_1(p,j)^3);
        end
    end
    w2o_1=w2o1_1;
    w1o_1=w1o1_1;
    bo_1=bo1_1;
    co_1=co1_1;
    oo_1=oo1_1;
end
%---------------------------------------------------------------------------
function[c_2,o_2,w1_2,w2_2,b_2,w1o_2,w2o_2,bo_2] = trainf_n1_2…
      (N1,N2,M,x_2,erh_2,minm_2,summin_2,m_2,ys_2,a_2,c_2,o_2,w1_2,…
       w2_2,b_2,ym_2,net_2,w1o_2,w2o_2,bo_2);
a2=0.2;
```

```
w2o1_2=w2_2;
w1o1_2=w1_2;
bo1_2=b_2;
co1_2=c_2;
oo1_2=o_2;
for j=1:N2
    for p=1:N1
        w1_2(j,p)=w1_2(j,p)+a_2*erh_2*x_2(p)*x_2(p)*…
                  minm_2(j)/summin_2+a2*(w1_2(j,p)-w1o_2(j,p));
        w2_2(j,p)=w2_2(j,p)+a_2*erh_2*x_2(p)*x_2(p)*…
                  minm_2(j)/summin_2+a2*(w2_2(j,p)-w2o_2(j,p));
    end
end

for j=1:N2
    b_2(j)=b_2(j)+a_2*erh_2*minm_2(j)/summin_2+a2*(b_2(j)-bo_2(j));
end
sec_2=1;
if sec_2==1
    for j=1:N2
        kw(j)=(net_2(j)-ys_2)/summin_2; %using defuzification
    end

    for p=1:N1
        for j=1:N2
            c_2(p,j)=c_2(p,j)+a_2*erh_2*kw(j)*minm_2(j)*…
                     2*(x_2(p)-c_2(p,j))/(o_2(p,j)^2);
            o_2(p,j)=o_2(p,j)+a_2*erh_2*kw(j)*minm_2(j)*…
                     2*((x_2(p)-c_2(p,j))^2)/(o_2(p,j)^3);
        end
    end
    w2o_2=w2o1_2;
    w1o_2=w1o1_2;
    bo_2=bo1_2;
    co_2=co1_2;
    oo_2=oo1_2;
end
```