# Development Of Novel Neuro-Fuzzy Techniques

# For Adaptive Systems

A thesis submitted for the degree of

## Master of Technology (Research)


## T. Vamsi Krishna


**Department of Electrical Engineering**

**National Institute Of Technology**

**Rourkela, India**


**2006**

# Development Of Novel Neuro-Fuzzy Techniques For Adaptive Systems

## Abstract

**N**ovel approaches for designing adaptive schemes based on neuro-fuzzy platform have been developed. Two kinds of adaptive schemes namely, adaptive equalization and system identification are implemented using the developed proposed techniques. The Radial basis function (RBF) equalizer is chosen as a case study for adaptive equalization of the digital communication channels. An efficient method for reducing the centers of a RBF equalizer based on eigenvalue analysis is presented. The efficiency of the method is further verified for RBF equalizers with decision feedback for tackling channels with overlapping channel states. A comparative study between the proposed center reduction technique and other center reduction techniques for the RBF equalizer is discussed. In another breakthrough a parallel interpretation of the ANFIS (adaptive network based fuzzy inference systems) architecture is proposed. This approach helps to investigate the role of the fuzzy inference part and the sub-filter part of the ANFIS separately. The parallel interpretation of the ANFIS redefines the opinion reserved for the fuzzy inference system, thereby allowing it to be considered as a fuzzy weighted sub-filter network, with the weighting functions and the sub-filter units arranged parallely. This approach motivated in developing many novel schemes for designing adaptive systems with application to system identification problems. Finally, the limitations of the ANFIS architecture are discussed. These limitations are exploited to develop neuro-fuzzy models similar to the ANFIS with the objective of reducing the number of parameters in comparison to the ANFIS. The developed neuro-fuzzy models are compared to the ANFIS in terms of the time required for learning and number of parameters to be adapted.

# CERTIFICATE

This is to certify that the thesis work entitled "**Development Of Novel Neuro-Fuzzy Techniques For Adaptive Systems**", submitted by Mr. T. Vamsi Krishna is a record of bonafide work carried out under my supervision. This thesis fulfils all the requirements for the award of the degree Master of Technology (Research). The contents of this research work have not been submitted elsewhere for the award of any other degree to the best of my knowledge and belief.

**Dr. J. K. Satapathy, Ph.D. (Bradford)**
**Professor & Dean Admin**
Department of Electrical Engineering,
National Institute of Technology,
Rourkela-769008, India.

# Acknowledgements

I extend my sincere thanks to my thesis supervisor, Prof. J. K. Satapathy. Besides being my thesis supervisor, he has been a mentor and a guide. Working with him has been the most enriching experience of my life. I am very glad to have discussed with him a broad range of topics starting from my thesis to my career. This indeed has been the biggest motivating factor. I could always find a direction in my research work under his guidance. The best part in working with Prof. Satapathy has been that, as a student I never felt any pressure while speaking to him and could always express my opinions freely.

I would like to express my gratitude to Prof. P. K. Nanda, Prof. S. Patnaik, Prof. Susmita Das and Mr. S. Mohanty for teaching me the necessary course work. Their collective effort helped me to understand the concepts underlying my research work. I sincerely appreciate their invaluable suggestions and criticisms.

I would like to specially thank Ms. K. R. Subhashini for constantly encouraging me throughout my thesis. I would also like to thank her for the kindness and ever cheerful attitude.

I owe a lot of thanks to my friends Sunil, Saroj, Manas, Pankaj, R. P. Rana and many others for making my first ever hostel experience very memorable. I also thank my friends Netaji and Pavan for providing me good company during my work in the Soft Computing Lab.

**Dedicated to my parents**

# Contents

# LIST OF FIGURES

# LIST OF TABLES

# LIST OF SYMBOLS

| | |
|---|---|
| $n_a$ | Channel order |
| $m$ | Equalizer order or no. of equalizer inputs |
| $N$ | Possible number of channel states |
| $n_b$ | Feedback order |
| $d$ | Channel delay |
| $s(k)$ | Transmitted data signal at time instant $k$ |
| $y(k)$ | Noisy channel output at time instant $k$ |
| $n(k)$ | Additive white gaussian noise at time instant $k$ |
| $H(z)$ | Channel transfer function |
| $e(k)$ | Error or the difference between the desired signal and the output at time instant $k$ |
| $\hat{s}(k-d)$ | Detected data or equalizer output at time instant $k$ after a delay $d$ |
| $\hat{\boldsymbol{s}}(k-d)$ | Feedback vector based on the feedback order $n_b$ |
| $c_j$ | Center of a gaussian function |
| $\sigma_j^2$ | Noise variance |
| $(\lambda_{\max})_i$ | Maximum eigenvalue associated with a center $c_{i,j}$ |
| $N_{sub}$ | Number of subsets formed using the feedback vector $\hat{\boldsymbol{s}}(k-d)$ |
| $\mu(\cdot)$ | Membership function associated with a linguistic variable |
| $w_i$ | Weighting function or the fuzzy rule node output |
| $f_i$ | Sub-filter output |
| $\Pi$ | Product operator |
| $\min(\cdot)$ | Minimum operator |
| H | Hybrid rule node |
| $\Phi_i$ | Slope of the sigmoid function |

# LIST OF ABBREVIATIONS

| | |
|---|---|
| MLP | Multilayer Perceptron |
| RBF | Radial Basis Function |
| RNN | Recurrent Neural Network |
| DFE | Decision Feedback Equalizer |
| ANFIS | Adaptive network based Fuzzy Inference System |
| SF | Sub-Filter |
| FS | Fuzzy System |
| superSAB | (super) Self Adaptive Backpropagation |
| RBF-SF | Radial Basis Function weighted Sub-Filter network |
| MBF-SF | Modified Basis Function weighted Sub-Filter network |
| Sig1-SF | Logistic Sigmoid Function weighted Sub-Filter network |
| Sig2-SF | Hyperbolic Tangent Function weighted Sub-Filter network |
| M-ANFIS | Adaptive Network based Fuzzy Inference System with Modified rule nodes |
| ANFIS-2 | Adaptive Network based Fuzzy Inference System with 2 Fuzzy Inference Systems |

# Chapter 1

# Introduction

The purpose of this research is to explore two of the adaptive systems, namely, adaptive equalization and system identification from a neuro-fuzzy perspective. Therefore the research is presented in two parts. The first part of this work deals with adaptive equalization specifically pertaining to the simplification of the design of a Radial Basis Function Network (RBF) based equalizer. The second part of this research emphasizes the architecture of Adaptive Network Based Fuzzy Inference System (ANFIS) with an interpretation that leads to more interesting neuro-fuzzy models.

## 1.1 Adaptive Systems

In the recent years engineers are motivated to design adaptive systems which give a better performance amidst changing environment and system requirements. The adaptive systems (Fig 1.1) provide an optimal and robust solution when the system is subjected to a process called *learning*. The main advantage of the adaptive systems over the non-adaptive schemes lies in their self adjusting and time varying capabilities. Thus we find the application of adaptive systems in a range of applications like prediction, system identification (modeling), adaptive equalization of digital channels and interference cancellation. In most cases the system is modeled as a using a linear FIR filter (tapped delay line referred to as TDL) or a nonlinear filter (say, neural networks, fuzzy logics or a combination of both).

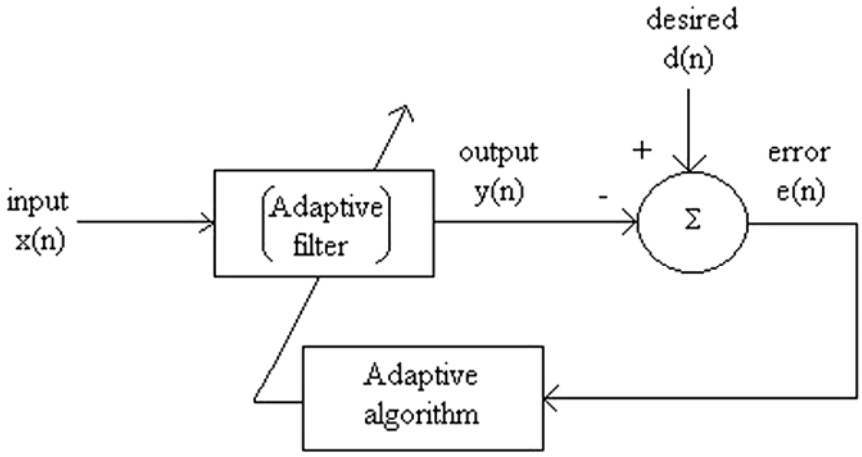



**Fig. 1.1: Adaptive system**

In each case the parameters of the adaptive filter are initialized to small random values and updated iteratively using an adaptive algorithm. The learning may be supervised (training data is known) or unsupervised (training data is not present). The learning process involves the minimization of a cost function with respect to the parameters of the adaptive filter. The cost function E is chosen to be the mean squared difference between the target value (desired output) $d(n)$ and the adaptive filter output $y(n)$. The learning may be facilitated by choosing an appropriate adaptive algorithm. Various adaptive algorithms like the least mean square (LMS) algorithm, recursive least squares (RLS) or the Kalman filter algorithm may be applied for learning. For instance the LMS algorithm provides robust performance by iteratively minimizing the mean square error in the direction opposite to the gradient of the cost function with respect to the parameters $w_i(n)$ of a TDL filter (see Fig.1.2). This can be expressed as

$$E = \sum_{n=1}^{P} e^2(n) \tag{1.1}$$

where the error over P ensembles of a training data set is given by

$$e(n) = d(n) - y(n) \tag{1.2}$$

If the output of the adaptive filter is given as

$$y(n) = \sum_i w_i(n) * x(n-i) \tag{1.3}$$

then the parameters $w_i(n)$ are updated using a learning rate $\eta$ as

$$w_i(n+1) = w_i(n) - \eta * \frac{\partial E}{\partial w_i(n)} \tag{1.4}$$

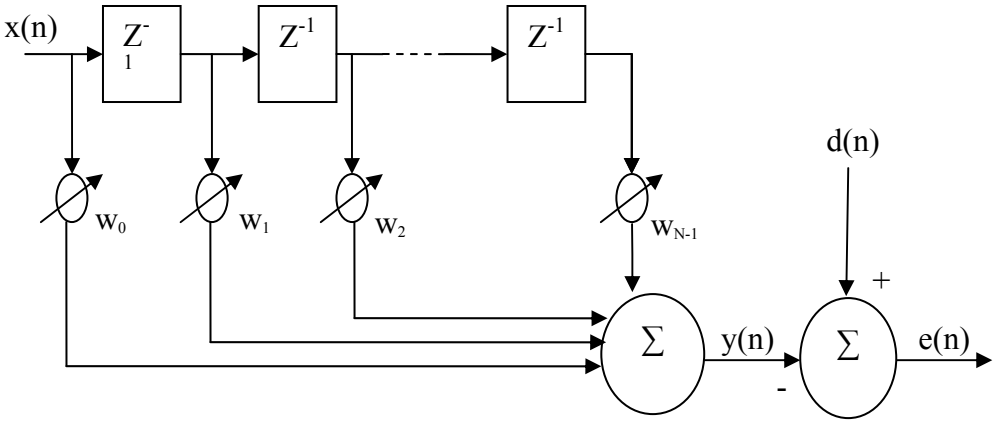



**Fig 1.2: Tapped Delay Line Filter**

## 1.2 Adaptive Equalization

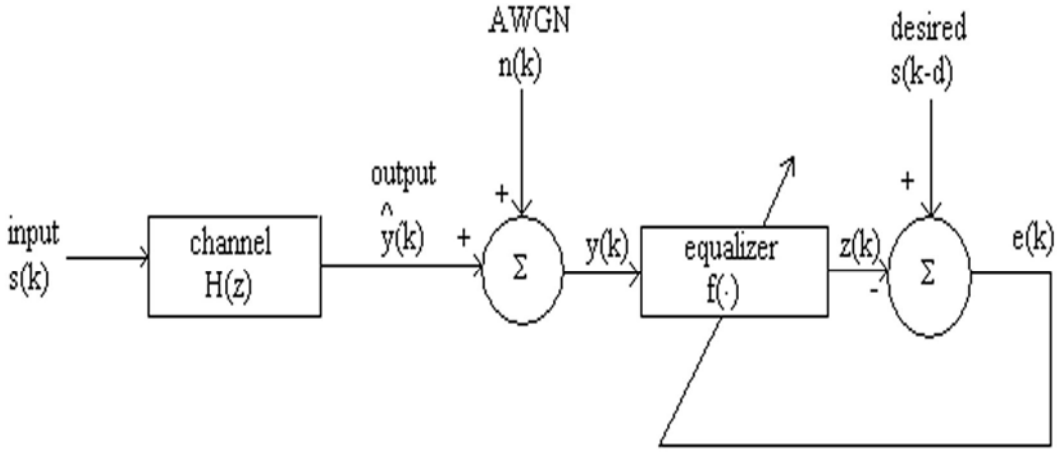



**Fig. 1.3: Adaptive Equalization**

This is otherwise refered to as nonlinear classification problem. In a dispersive medium (digital channel) an adaptive filter is placed in the receiving end to compensate the effects induced by the channel over the transmitted data. The digital channel introduces two impairments namely intersymbol interference (ISI) and additive gaussian noise (AWGN), $n(k)$. The transmitted data $s(k)$ is randomly generated, comprised of equiprobable and independent symbols. In case of a binary source the data is comprised of symbols -1 or +1.

The received signal $y(k)$ is represented as

$$y(k) = \sum_{i=0}^{n_a-1} h_i s(k-i) + n(k) \tag{1.5}$$

The channel can be modeled as a TDL with an order (number of coefficients) $n_a$.

$$H(z) = \sum_{i=0}^{n_a-1} h_i z^{-i} \tag{1.6}$$

Thus we can define adaptive equalization as a process that restores a time delayed version of the transmitted signal $\hat{s}(k-d)$ at the receiver, where $d$ is the channel delay. The equalizer training may be done using the LMS algorithm. The criterion is to achieve the minimum mean square error between the detected data $z(k)$ (equalizer output) and the desired data $s(k-d)$. This may be expressed as

$$e(k) = s(k-d) - z(k) \tag{1.7}$$

The equalizer operation may be expressed as a transformation $f(\cdot)$ equivalent to the inverse of the channel transfer function, applied on the received noisy data vector $\mathbf{y}(k)$. $\mathbf{y}(k) = [y(k), y(k-1),..., y(k-m+1)]$ where $m$ is the equalizer order. The number of possible combination of $m$ noiseless channel observations (also called as channel states) is given by $n_s = 2^{m+n_a-1}$.

The equalizer output is given by

$$z(k) = f(\mathbf{y}(k)) \tag{1.8}$$

A fully trained equalizer makes an estimate of the transmitted data (or bit) based on the received noisy observation vector $\mathbf{y}(k)$. A hard limiting operation is performed on the equalizer output $z(k)$ as follows

$$\hat{s}(k-d) = \text{sgn}\{z(k)\} \tag{1.9}$$

where the function $\text{sgn}\{\cdot\}$ is mathematically expressed as

$$\begin{aligned} \text{sgn}\{z(k)\} &= +1 \quad if \quad z(k) \geq 0 \\ \text{sgn}\{z(k)\} &= -1 \quad if \quad z(k) < 0 \end{aligned} \tag{1.10}$$

The above equation (1.10) clearly signifies equalization as a classification problem.

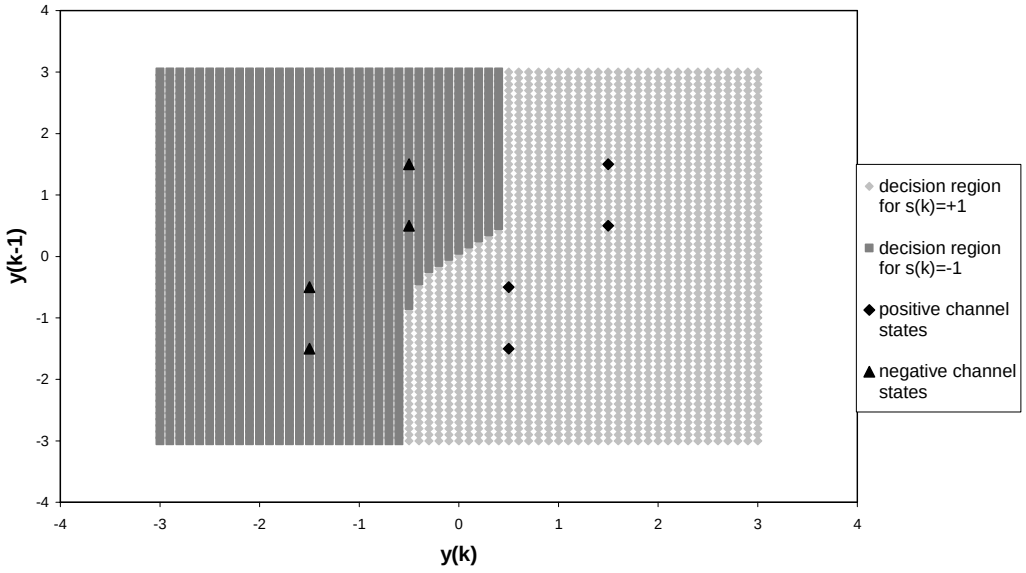
## 1.3 Linear & Nonlinear Equalizers

The equalizers may be classified as linear or nonlinear depending on the architecture. Linear equalizers are modeled on the tapped delay line filter. They are the simplest structures that can be realized. They require least computational complexity and training period. But such equalizers can never approach the optimal performance since they can at the best provide a linear classification of the received

data. This may be compensated by increasing the filter length and employing a nonlinear cost function as the criterion. In most cases it has been observed that an increase in the filter length (or order) enhances the additive white gaussian noise.

This leads to the definition of an optimal performance since equalization is a nonlinear process involving the construction of a nonlinear decision boundary (see Fig. 1.4) between the received data points (channel states) belonging to the various classes of data symbols used in transmission. Thus an optimal equalizer operates with the least number of misclassifications. Hence a nonlinear adaptive filter is ideally suited for adaptive equalization.

**Fig. 1.4: Nonlinear decision boundary for channel $H(z)=1+0.5z^{-1}$, m=2, d=0**




In the recent years, research in the field of neural networks has led to their application in adaptive equalization, control systems, system identification, etc. [1]-[5]. The processing units of a neural network introduce the nonlinearity required to implement a nonlinear task. The equalization using neural networks is viewed as a pattern classification task wherein the equalizer maps the estimated data symbol to the closest channel state. Equalizers using various neural network and fuzzy logics architectures have been reported [6]-[14]. Initial work by Cowan et al [6] proved that the multilayer perceptron (MLP) equalizer was far superior to the linear equalizers. But the multilayer perceptron equalizers require longer training period and the architecture selection is also debatable. The application of decision feedback enhances the performance of the MLP equalizer [7], [8]. The benefit of decision feedback lies in the fact that it improves the resolution in decision making, thereby reducing the

errors in the classification task. It also brings down the number of computing operations. The recurrent neural network (RNN) based equalizer is a very compact structure that can produce a low residual mean square error. Manolakos et al [9], [10] showed the application of RNN equalizer to adaptive equalization and blind equalization. However the highly nonlinear RNN equalizer suffers from the local minima problem during weight update. Radial Basis Function (RBF) network has been the most inspiring neural network due to its simple structure and training procedure. The RBF equalizer [12]-[14] has been reported to have given an optimal performance since it is modeled on the optimum Bayesian equalizer [19].

## 1.4 RBF EQUALIZER

An optimal Bayesian equalizer [19] gives the best classification of the estimated data symbols since it is based on the maximum a posteriori probability of occurrence of the channel states. It is assumed that the transmitted data $s(k)$ is drawn from a distribution that consists of independently and identically distributed symbols of -1 or +1. The received data $y(k)$ is observed in an additive white gaussian noise background. Thus an observation vector $\mathbf{y}(k)$ can appear in any of the $n_s$ noisy gaussian clusters.

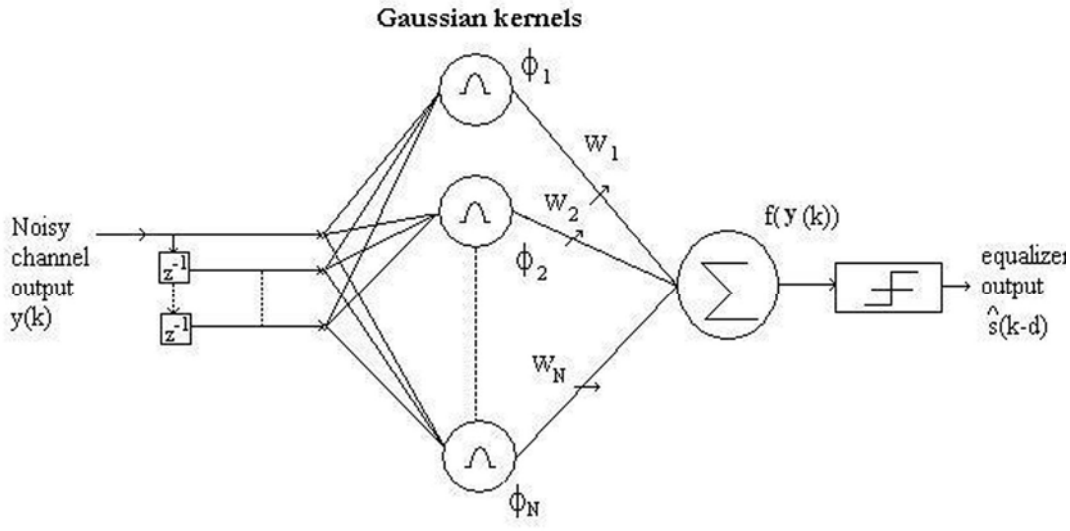



Fig. 1.5: RBF Equalizer

The noiseless channel state $\hat{\boldsymbol{y}}(k)$ is given by

$\hat{\boldsymbol{y}}(k) = [\hat{y}(k), \hat{y}(k-1), ..., \hat{y}(k-m+1)]$ where $m$ is the equalizer order.

The mean value of each cluster is equal to its respective noiseless channel state and the width of each cluster is equal to the noise variance $\sigma^2$. These $n_s$ noisy clusters can be classified into two classes according to the transmitted symbol $s(k-d)$ as

$$Y_{m,d}^{-} = \{\hat{\boldsymbol{y}}(k) \mid s(k-d) = -1\} \tag{1.11}$$

$$Y_{m,d}^{+} = \{\hat{\boldsymbol{y}}(k) \mid s(k-d) = +1\} \tag{1.12}$$

In the binary signalling scheme each desired channel state $\boldsymbol{y}_i^{+} \in Y_{m,d}^{+}$ and $\boldsymbol{y}_i^{-} \in Y_{m,d}^{-}$ occurs with the same apriori probability $p_i = 1/n_s = 1/2$. The number channel states in $Y_{m,d}^{+}$ and $Y_{m,d}^{-}$ can be represented as $n_s^{+}$ and $n_s^{-}$ respectively. The equalizer follows the decision rule

$$\begin{aligned} \hat{s}(k-d) &= \operatorname{sgn}\{f(\hat{\mathbf{y}}(k)\} = +1 \quad f(\mathbf{y}(k)) \geq 0 \\ \hat{s}(k-d) &= \operatorname{sgn}\{f(\hat{\mathbf{y}}(k)\} = -1 \quad f(\mathbf{y}(k)) < 0 \end{aligned} \tag{1.13}$$

The decision boundary is defined as the boundary that separates the above two classes of channel states or noisy gaussian clusters. It may be mathematically represented as

$$f(\mathbf{y}(k)) = \sum_{i=1}^{n_s^{+}} \exp\left(\frac{-\| \mathbf{y}(k) - \mathbf{y}_i^{+} \|^2}{2\sigma_n^2}\right) - \sum_{i=1}^{n_s^{-}} \exp\left(\frac{-\| \mathbf{y}(k) - \mathbf{y}_i^{-} \|^2}{2\sigma_n^2}\right) \tag{1.14}$$

The above equation can be simply represented as

$$\{\mathbf{y} \mid f(\mathbf{y})\} = 0 \tag{1.15}$$

The Radial Basis Function equalizer (Fig. 1.5) is modeled on the optimum Bayesian equalizer. The RBF equalizer structure is a two layered structure consisting of a hidden layer consisting of basis functions modeled using gaussian kernels and an output layer containing a summer which linearly combines the output of each basis function.

The RBF equalizer output is expressed as

$$f(\mathbf{y}(k)) = \sum_{j=1}^{n_s} w_j \exp\left(\frac{-\| \mathbf{y}(k) - c_j \|^2}{2\sigma_n^2}\right) \tag{1.16}$$

The centers of the gaussian kernel is equal to the desired channel state $c_j$ and the width of the gaussian kernel is equal to $2\sigma_n^2$.

## 1.5 System Identification

System identification (Fig. 1.6) otherwise called adaptive modeling is the process by which an adaptive filter is able to mimic the input-output relationship of an unknown system or plant (in control system terminology). The adaptive system might not possess the same transfer function of the unknown system, but it can be

functionally equivalent to the plant if we allow its output to be a best least squares fit to that of the unknown system [28].

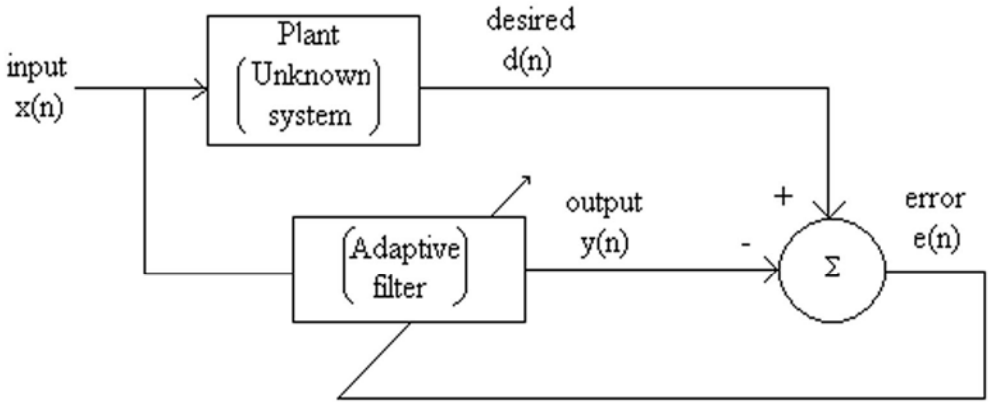



**Fig. 1.6 System Identification**

Various adaptive filter architectures have been reported in literatures which were either based on the neural networks, fuzzy logics or a combination of both [29], [30]. The neuro-fuzzy architecture has been of special interest, since the researchers have attempted to exploit the properties of neural networks and the filters based on fuzzy logics. The idea behind such an approach is to exploit the interpolation properties of the fuzzy logics filters and the rigid adaptive properties of the neural networks.

Most researchers have modeled neuro-fuzzy systems based on the Takagi-Sugeno model [31]. Linguistic variables are defined for each input to the plant. Rule nodes are formed by defining fuzzy IF-THEN statements using fuzzy conjunction operators. These rules are then combined using a function $f(\cdot)$ as follows

If $x$ is A and $y$ is B Then $z = f(x, y)$ (1.17)

Each rule has a crisp output and the output is obtained by weighted average of the crisp outputs.

$$z = w_1 z_1 + w_2 z_2 + \ldots \quad (1.18)$$

Adaptive Network Based Fuzzy Inference System (ANFIS) [32], is modeled on the Takagi-Sugeno model offers a wonderful neuro-fuzzy system for system identification. The ANFIS is a five layered structure with layer 1 and layer 4 containing adaptive parameters.

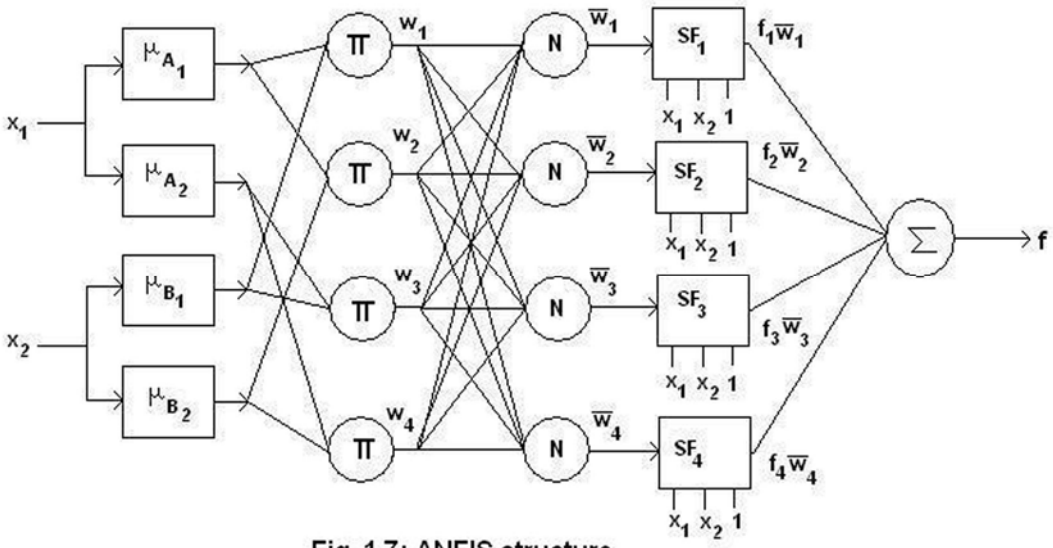



Fig. 1.7: ANFIS structure

The architecture of ANFIS (as in *fig. 1*.7) can be summarized as follows

**Layer 1:**

This layer contains nodes with adaptive fuzzy parameters. Each node outputs the degree of the membership function associated with it for an input $x$. The output of each node is represented as

$$O_i^1 = \mu_{A_i}(x) \tag{1.19}$$

where $A_i$ is the linguistic label associated to the node $i$. Bell function or gaussian function can be used for the membership function.

$$\mu_{A_i}(x) = \frac{1}{1+\left(\frac{x-c_i}{a_i}\right)^{2b_i}} \text{ for bell membership function} \tag{1.20}$$

$$\mu_{A_i}(x) = \exp\left(-\left(\frac{x-c_i}{a_i}\right)^2\right) \text{ for gaussian membership function}$$

The parameters of this layer $\{a_i, b_i, c_i\}$ are referred to as *premise* parameters

**Layer 2:**

The nodes of this layer are denoted by $\Pi$. The nodes in this layer multiply the incoming data and send out their product.

$$w_i = \mu_{A_i}(x) \cdot \mu_{B_i}(y) \text{ for } i = 1,2 \tag{1.21}$$

**Layer 3:**

The nodes of this layer are labelled as $\mathrm{N}$. Every node in this layer computes the ratio of the rule's firing strength to the sum of all rule's firing strengths.

$$\overline{w_i} = \frac{w_i}{\sum_i w_i} \text{ for } i = 1,2 \tag{1.22}$$

**Layer 4:**

The parameters $\{p_i, q_i, r_i\}$ of the nodes in this layer are adaptive. The output of this layer is given by

$$O_i^4 = f_i \cdot \overline{w_i} = (p_i x + q_i y + r_i) \cdot \overline{w_i} \tag{1.23}$$

The parameters of this layer are called as *consequent* parameters.

**Layer 5:**

The single node of this layer computes the output of the ANFIS by summing the incoming signals.

$$O_1^5 = f = \sum_i f_i \cdot \overline{w_i} = \frac{\sum_i f_i \cdot w_i}{\sum_i w_i} \tag{1.24}$$

The ANFIS is trained using a hybrid learning technique. The consequent parameters are updated using the recursive least squares, expressed as

$$\begin{aligned} X_{i+1} &= X_i + S_{i+1} a_{i+1} (b_{i+1}^T - a_{i+1}^T X_i) \\ S_{i+1} &= S_i - \frac{S_i a_{i+1} a_{i+1}^T S_i}{1 + a_{i+1}^T S_i a_{i+1}} \end{aligned} \tag{1.25}$$

where $X_i = [p_i \;\; q_i \;\; r_i]^T$ contains the consequent parameters, $a_{i+1} = [x \;\; y \;\; 1]^T$ is the training input vector, $b_i$ is the desired data and $S_i$ is the covariance matrix, with the initial value given by $S_0 = \gamma I$, where $\gamma$ is an large positive number and $I$ is an identity matrix.

## 1.6 Organization of the chapters

The subsequent chapters deal with the proposed work as follows

Chapter 2 elaborates the adaptive equalization implemented using Radial Basis Functions network and extends the concept to RBF equalizers with decision feedback. The implementation issues are briefed along with some popular architecture pruning strategies. A novel scheme for reducing centers in a RBF equalizer based on the eigenvalue analysis is presented. An interesting aspect relating the eigenvalue and the centers is highlighted. Chapter 3 deals with the interpretation of the structure of ANFIS and opens the prospect of realizing an ANFIS structure by selecting the sub-filters ahead of the fuzzy rule nodes. This idea is also applied in designing nonlinear

models that can perform in par with the ANFIS. Chapter 4 deals with an ANFIS structure with modified rule nodes. Also a scheme for reducing the ANFIS is proposed. Finally, chapter 5 concludes with the contribution of this thesis and provides an insight into the future of this research.

# Chapter 2

# RBF Equalizer

The Radial Basis Function (RBF) equalizer provides a superior performance since the structure is directly based on the optimum Bayesian classifier. The RBF network is a two layered structure comprised of a hidden layer consisting of gaussian basis functions (RBF nodes) and an output layer containing a summing unit. The output of the RBF network is a weighted sum of the responses of the RBF nodes. The desired channel states are selected as the centers of the RBF equalizer and the spread is chosen equal to the variance of the additive gaussian noise.

## 2.1 Training the RBF Equalizer

Learning is a process by which the free parameters of a neural network adapt to the changing environment and system requirements [1]. Learning can be classified as supervised or unsupervised. In supervised learning a neural network learns with respect to a known training data set (epoch). While in the unsupervised learning a training data set is not known. The unsupervised learning is referred to as blind equalization.

The RBF equalizer training can be considered in two parts. Firstly the number and position of the centers of the RBF nodes is ascertained (either by supervised or unsupervised manner). Secondly the weights of the RBF equalizer are adapted (again by supervised or unsupervised method).

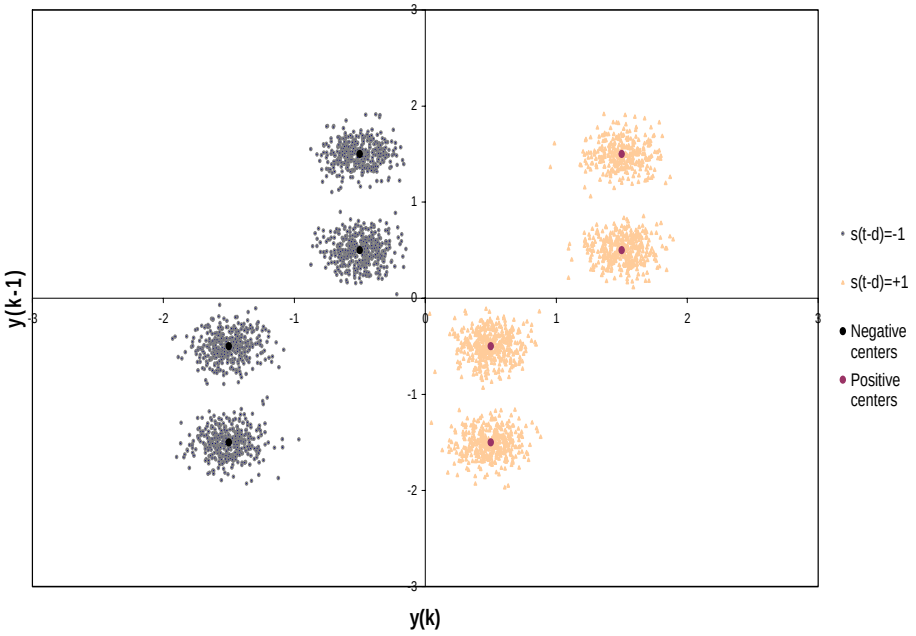
### 2.1.1 Estimation of RBF node parameters: Supervised learning

Various techniques have been proposed to estimate the centers of the RBF equalizer [12], [15]-[17]. These methods may be applied to the supervised and blind equalization schemes accordingly. The number of centers $N$ required to design a RBF equalizer is given by the expression

$$N = 2^n = 2^{m+na-1} \tag{2.1}$$

where $n = m + n_a - 1$ for a channel order $n_a$ and equalizer order and $m$.

Fig. 2.1 Channel states and noisy channel output
$H(z)=1+0.5z^{-1}$, m=2, d=0 and SNR=20 dB




The location of the centers can be found by observing the noisy channel observations (or channel states) by transmitting the training data signal (epoch) through the channel. The noisy channel outputs form clusters about the desired channel states with the width of the clusters equal to the variance of the additive white gaussian noise.

But the information about the channel order and channel coefficients is not known a priori and needs to be estimated. Lee et al [18] proposed a method to estimate the centers. Regression analysis is used to find the channel order $n_a$ and channel delay $d$. The digital channel is assumed to be a linearly dispersive and hence a regression model may be applied to estimate the channel coefficients $h_i$.The channel output can be expressed as

$$y(k) = \sum_{i=0}^{n_a-1} h_i s(k-i) + n(k) \qquad (2.2)$$

which can be expanded into

$$y(k) = h_0 s(k) + h_1 s(k-1) + \ldots + h_{n_a-1} s(k-n_a+1) + n(k) \qquad (2.3)$$

Thus the noise free channel output can be expressed as

$$\hat{y}(k) = \hat{h}_0 s(k) + \hat{h}_1 s(k-1) + \ldots + \hat{h}_{n_a-1} s(k-n_a+1) \qquad (2.3)$$

where $\hat{h}_i$ are the estimated channel coefficients with $0 \le i < n_a$, $\hat{y}(k)$ represents the estimated channel output with $0 \le k \le N$ and $s(k-i)$ is the transmitted binary symbol with $0 \le i < n_a$. In matrix form

$$\hat{\mathbf{Y}}=\mathbf{S}.\hat{\mathbf{H}}$$

The rows of matrix **S** are $N = 2^n$ possible combinations of the binary message symbols $s(k-i)$ that could be input through a channel of order $n_a$. The sum of squared errors (SSE) for the optimized $\hat{\mathrm{h}}_i$ is given by

$$SSE = \sum_{k=1}^{N} [y(k) - \hat{y}(k)]^2 = \mathbf{Y}^T \mathbf{Y} - \hat{\mathbf{H}}^T \mathbf{A}^T \mathbf{R} \qquad (2.4)$$

where

$$\hat{\mathbf{H}}=(\mathbf{S}^T \mathbf{S})^{-1} \mathbf{S}^T \mathbf{Y} \qquad (2.5)$$

The estimated channel order is the order of the model for which the SSE is minimum. The channel delay is estimated as the integer next larger than the channel order for which the SSE is maximum.

The number of centers is readily given from the estimated channel order by equation (2.1). The channel coefficients can be estimated from equation (2.5). The location of the centers (or the desired channel states) can be found by averaging the noisy channel outputs, by adaptive k-means algorithm [4], for each row of the input vector matrix **S** as follows

$$c_{i,j} = h_0 s_i(k-j) + h_1 s_i(k-j-1) + ... + h_{n_a-1} s_i(k-(n_a-1)-j) \qquad (2.6)$$

where $c_{i,j} = [c_{i,0}, c_{i,-1}, ..., c_{i,-m+1}]$ with $1 \le i \le N$ and $0 \le j \le m-1$.

The noise variance may be estimated as shown by Chen et al [2] as follows

$$\sigma^2 = E[\| \boldsymbol{y}(k) - c_{i,j} \|^2 / m] \qquad (2.7)$$

where $E$ is expectation operator and the channel output vector $\boldsymbol{y}(k)$ is given by $\boldsymbol{y}(k) = [y(k), y(k-1), ..., y(k-m+1)]$. If $\boldsymbol{s}(k) = \boldsymbol{s}_i$ at $k$ then the noise variance estimate is updated as

$$\hat{\sigma}^2(k) = \{(k-1)\hat{\sigma}^2(k-1) + \| \boldsymbol{y}(k) - c_{i,j} \|^2 / m\} / k \qquad (2.8)$$

### 2.1.2 Supervised learning for weights

The estimation of the centers and spread of the RBF nodes is followed by updating the weights connecting the RBF nodes to the output summing unit. If the transmitted data is equiprobable then half the weights are assigned +1 and the rest -1. But in practice the weights are initialized to small random values. They are updated by using the least mean square algorithm (LMS):

The cost function $E$ is chosen as the mean square error between the transmitted signal and the RBF equalizer output.

$$E = \frac{1}{2} e^2(k) \tag{2.9}$$

$$e(k) = s(k-d) - f(\boldsymbol{y}(k)) \tag{2.10}$$

where the RBF output is given by

$$f(\boldsymbol{y}(k)) = \sum_{i=1}^{N} w_i \Phi(\| \boldsymbol{y}(k) - c_{i,j} \|) \tag{2.11}$$

and $$\Phi(\| \boldsymbol{y}(k) - c_{i,j} \|) = \exp(\| \boldsymbol{y}(k) - c_{i,j} \|^2 / 2\sigma^2) \tag{2.12}$$

Thus the LMS update for the weights $w_i$ by a learning rate $\eta$ is given by

$$w_i(k+1) = w_i(k) - \eta \cdot \frac{\partial E}{\partial w_i(k)} \tag{2.13}$$

## 2.2 RBF equalizer with DFE

The performance of an equalizer is measured in terms of the bit error rate (BER). An equalizer is said to offer good performance if equalizer operates with a very small probability of misclassification error. The performance of an equalizer degrades when the desired channel states occur very close to one another. Chen et al [7], [19]-[20] show that the application of decision feedback improves the performance of the equalizer. An equalizer with decision feedback estimates the received data with the knowledge of previously detected outputs $\hat{\boldsymbol{s}}(k-d)$, where the feedback vector for a given feedback order $n_b$ is represented as

$$\hat{\boldsymbol{s}}(k-d) = [\ \hat{s}(k-d-1),\ \hat{s}(k-d-2),...,\ \hat{s}(k-d-n_b)] \tag{2.14}$$

When an RBF equalizer is trained using decision feedback a subset of channel states are considered based on the value of $\hat{\boldsymbol{s}}(k-d)$. In other words the number of channel

states involved in the classification problem are reduced by a factor $N_{sub} = N/2^{n_b}$. For instance the channel states shown in boldface in the *Table 2.1* represent the channel states involved when the feedback vector $\hat{\boldsymbol{s}}\ (k-d) = [-1,-1]$. The RBF equalizer with decision feedback can be considered as a network in which only the weights connected to the centers corresponding to the feedback vector $\hat{\boldsymbol{s}}(k-d)$ are updated while the weights connected to other centers are inactive at the instant $k$.

## 2.3 Implementation issues

The main problem encountered in the practical scenario of an RBF equalizer design is the size of the network which grows exponentially with increasing channel order $n_a$ and equalizer order $m$. Thus it may become practically impossible to train a network. In practice a node pruning rule may be incorporated as follows:

***RULE:***

The centers that are closest to the decision boundary play a more significant role in the decision making process while those lying farther away from the decision boundary contribute very little.

***INFERENCE:***

Thus the farthest centers form the decision boundary may be safely removed from the structure leaving a parsimonious RBF equalizer without performance degradation.

Many researchers have been motivated to exploit this rule to evolve reduced RBF equalizers [18], [21]-[23]. Lee et al [18] proposed a reduced RBF equalizer in which the centers were reduced by a factor of $2^d$.

For instance, consider the channel $h_1(z) = 0.2 + 0.3z^{-1} + 1.0z^{-2} + 0.3z^{-3}$ with channel order $n_a = 4$, equalizer order $m = 2$, and channel delay $d = 2$. The center reduction technique can be explained as follows

An input vector matrix of dimension $N \times n$ is generated as shown in *Table 2.2*. The rows contain all possible combinations of the input vector $\boldsymbol{s}(k)$, where $\boldsymbol{s}_i(k) = [s_i(k), s_i(k-1), ..., s_i(k-n+1)]$.

The centers (desired channel states) are found by inputting each input vector through the channel.

$$c_{i,j} = h_0 s_i(k-j) + h_1 s_i(k-1-j) + ... + h_{n_a-1} s_i(k-(n_a-1)-j) \qquad (2.15)$$

for $1 \le i \le N$ and $0 \le j \le m-1$.

The centers are divided into two sets by halving the *Table 2.2*. Then the centers in the two sets are grouped into subsets based on the vector $\boldsymbol{s}_i(k-d) = [s_i(k-1),...,s_i(k-d)]$

The subset distances are calculated between the centers belonging to every subset in the upper set having the vector $\boldsymbol{s}_i(k-d)$ to every center belonging to the subset in the lower set having the compliment of the vector $\boldsymbol{s}_i(k-d)$.

The pair of centers having the minimum subset distances are picked from the subsets. The equalizer is trained as explained above.

## 2.4 Center reduction based on eigenvalue analysis

A novel scheme for reducing the centers of a RBF equalizer was developed based on *eigenvalue* analysis.

### 2.4.1 Motivation behind the work

The convergence of the parameters of an adaptive filter to an optimal solution using LMS depends on the learning rate $\eta$. If a large $\eta$ is used for faster learning then the parameters will oscillate and the adaptive system will become unstable. On the other hand, if a very small $\eta$ is used then the parameters will take a long time to converge. Thus the selection of the learning rate $\eta$ becomes critical. A bound [24] can be stated for $\eta$ as below

$$0 \le \eta \le \frac{2}{\lambda_{\max}} \quad \text{if the cost function is chosen as } E = e^2(k) \qquad (2.16)$$

The above condition implies that the maximum eigenvalue $\lambda_{\max}$ plays an important role in the convergence of the parameters of any adaptive system to an optimal solution. The properties of $\lambda_{\max}$ are directly opposite to the those of $\eta$. Several training algorithms have been proposed to minimize the eigenvalue spread and

improve the convergence of adaptive filters [25], [26]. One way to update the parameters of the adaptive node using LMS may be to estimate the maximum eigenvalue $\lambda_{\max}$ from the autocorrelation matrix of the inputs and update the learning rate $\eta$. This may be accomplished by using a pseudo $\lambda_{\max}$ which is obtained from the autocorrelation of the past 8 channel observations at every iteration. Otherwise the autocorrelation has to be computed on the entire training set.

This led to the investigation of the role of $\lambda_{\max}$ in the distribution of the channel states. The noise component of the channel states plays a significant role in identifying the decision boundary between the channel states belonging to the two classes of symbols used. So the correlation properties of the channel states may be studied to get a clear idea on the role of every channel state in decision making.

### 2.4.2 Eigenvalue analysis and center pruning

Consider an input vector matrix $S_{i,j}$ where $1 \le i \le N$ and $0 \le j \le m-1$. The rows are represented by the vector $\boldsymbol{s}_i(k) = [s_i(k), s_i(k-1), ..., s_i(k-n+1)]$, composed of all the possible combinations of the symbols -1 and +1.

The columns of $S_{i,j}$ are shifted to the right in steps based on the channel delay $d$ in a cyclical manner.

For each vector $\boldsymbol{s}_i(k)$ the channel output is recorded as a center (desired channel state), this is given by $c_{i,j} = [c_{i,0}, c_{i,-1}, ..., c_{i,-m+1}]$ calculated as

$$c_{i,j} = h_0 s_i(k-j) + h_1 s_i(k-1-j) + ... + h_{n_a-1} s_i(k-(n_a-1)-j) \qquad (2.17)$$

The centers $c_{i,j}$ and their corresponding input vectors $\boldsymbol{s}_i(k)$ are tabulated as shown in *Table 2.3*.

The centers are grouped into $N_{sub} = 2^{n_b}$ subsets based on the feedback input vector $\hat{\boldsymbol{s}}_i(k-d) = [\hat{s}_i(k-d-1), ..., \hat{s}_i(k-d-n_b)]$.

The distances $d_{i,k}$ between each center $c_{i,j}$ and every other center $c_{k,j}$ within each subset are calculated, where $i \neq k$.

Autocorrelation matrix $A_i$ for the distances $d_{i,k}$ corresponding to each center $c_{i,j}$ is calculated.

The maximum eigenvalue $(\lambda_{\max})_i$ for each autocorrelation matrix $A_i$ corresponding to the center $c_{i,j}$ is calculated using QR algorithm [27].

The centers $c_{i,j}$ are arranged according to the ascending order of the maximum eigenvalue $(\lambda_{\max})_i$ as shown in *Table 2.4*.

The complimentary pair of centers or desired channel states with the smallest $(\lambda_{\max})_i$ is selected from each subset.

The RBF equalizer thus formed is trained as explained above and tested for bit error rate. If the performance is not satisfactory then the pair of complimentary centers with the next smallest $(\lambda_{\max})_i$ from each subset are added to the RBF equalizer. The training of the weights is repeated while as long as the performance stays close to the optimal performance.

### 2.4.3 Inference

It was observed that for the closest complimentary pair of centers or the desired channel states within each subset the maximum eigenvalue $(\lambda_{\max})_i$ was the smallest. This value of $(\lambda_{\max})_i$ increased as the complimentary pair of centers moved away from each other. In other words the value $(\lambda_{\max})_i$ is directly related to the distance between a pair of complimentary centers or desired channel states.

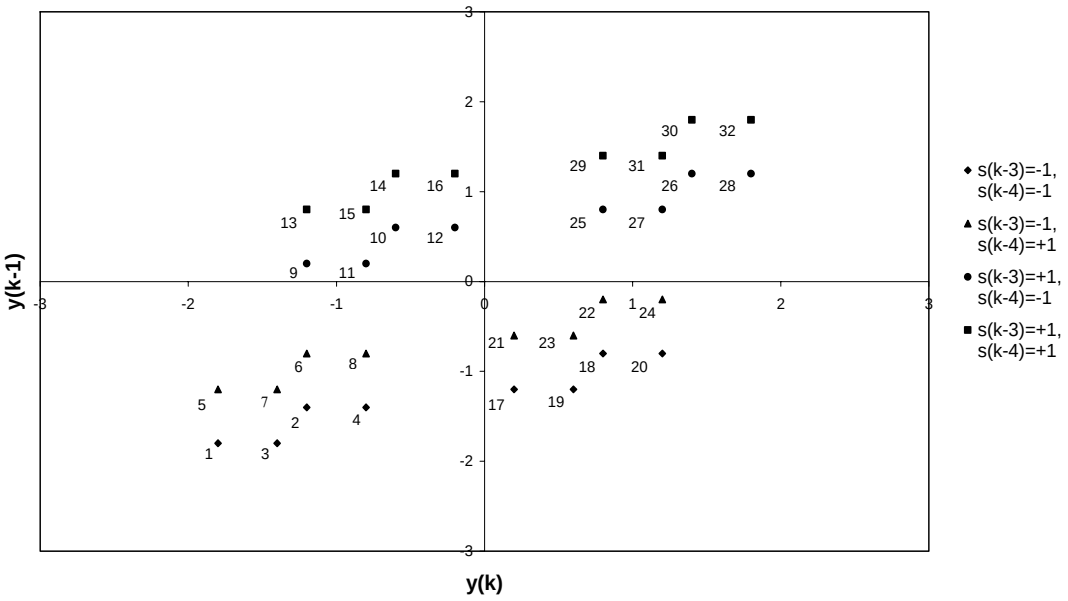
The above method is also applicable to a RBF equalizer using decision feed back. The center reduction per subset corresponding the feedback vector $\boldsymbol{s}_i(k-d)$ can be obtained through the eigenvalue analysis. The weights of the RBF equalizer with decision feedback are then trained using LMS. The criterion for adding centers to each subset is the same whether decision feedback is used or not.

## 2.5 Simulations

The simulations were carried out using $10^6$ randomly generated binary symbols [-1, +1]. The bit error rate (BER) or the ratio of the number of misclassified symbols to the total number of transmitted symbols is calculated to express the

performance measure. The RBF equalizer reduction is carried as long as the performance of the equalizer is comparable to the optimal BER. The performance of the RBF equalizer designed based on the eigenvalue analysis is compared to the equalizer designed based on the procedure reported by Lee at al. [18].

**Fig. 2.2: Centers grouped based on the sub vector *s* (k-d)**
**$H_1(z)=0.2+0.3z^{-1}+1.0z^{-2}+0.3z^{-3}$, m=2, d=2**




In the first example, the channel $H_1(z) = 0.2 + 0.3z^{-1} + 1.0z^{-2} + 0.3z^{-3}$ used in [18] is chosen. The channel $H_1(z)$ is of order $n_a = 3$ and has a channel delay $d = 2$. The equalizer order is chosen as $m = 2$ while the grouping factor is chosen as $n_b = 2$. The initial number of centers is calculated as $N = 32$. All possible centers are tabulated as in *Table 2.3*. The centers are grouped into $N_{sub} (= 2^{n_b} = 2^2 = 4)$ subsets based on the value of the sub vector $\hat{\mathbf{s}}_i(k-d)$. The eigenvalue analysis is applied over the subsets as explained above. The centers in every subset are arranged according to the ascending order of the maximum eigenvalue $(\lambda_{\max})_i$ as shown in *Table 2.4*. The figure *Fig. 2.2* shows the centers (4 & 17), (8 & 21), (12 & 25) and (16 & 29) are the closest to the decision boundary and are characterized by the smallest $(\lambda_{\max})_i$ within their respective subsets. And wisely these centers accomplish the reduced RBF equalizer as per the proposed technique. The resultant RBF equalizer

offers commendable performance when compared to its parent structure. The BER plot is depicted in *Fig 2.3*.

The second example is a magnification of the above problem. The equalizer order is increased to $m = 4$ and the grouping factor is chosen as $n_b = 4$. The total number of centers $N$ in this case is found to be 128. This structure is brought down to a RBF equalizer with 32 centers. The BER plot is shown in *Fig 2.4*.

In the third example a channel $H_2(z)$ of order $n_a = 4$ and channel delay of $d = 2$ is chosen. $H_2(z) = -0.2052 - 0.5131z^{-1} + 0.7183z^{-2} + 0.3695z^{-3} + 0.2052z^{-4}$. The equalizer order and grouping factor are given by $m = 3$ and $n_b = 4$ respectively. The centers were initially reduced from $N = 128$ to 32. But this resulted in a degradation of the performance, hence the centers with the next smallest $(\lambda_{\max})_i$ in every subset were added to the reduced structure. This resultant equalizer with 64 centers performed close to the full network. The BER plot is given in *Fig 2.5*.

In the subsequent simulations the case of a RBF equalizer with decision feedback is highlighted. The fourth example deals with a channel $H_1(z)$ having a delay $d = 2$. The equalizer order is chosen as $m = 2$ while the feedback order $n_b = 4$ takes the role of the grouping factor. The initial number of centers per subset is reduced from 8 to 2. The reduced RBF equalizer with decision feedback is more compact. Fig 2.6 shows the BER plot.

The fifth example takes the case of a channel $H_3(z)$ with overlapping centers which can only be trained using decision feedback. The channel order in this case is $n_a = 2$ and the delay is chosen as $d = 2$. The corresponding equalizer order and grouping factor (feedback order) are $m = 3$ and $n_b = 2$ respectively. Initial number of centers per subset is 8. The reduced RBF equalizer with DFE has 2 centers per subset. The BER plot is shown in *Fig 2.7*.

***Table No. 2.1:***

Channel states and symbols for $H_3(z) = 0.4084 + 0.8164z^{-1} + 0.4084z^{-2}$, $m = 2$, $d = 2$ and $n_b = 2$.

Boldface channel states are given for feedback vector $\boldsymbol{s}(k-d) = [-1,-1]$

| S(k-0) | s(k-1) | s(k-2) | s(k-3) | $c_{i,0}$ | $c_{i,1}$ |
|---|---|---|---|---|---|
| -1 | -1 | **-1** | **-1** | **-1.6332** | **-1.6332** |
| -1 | -1 | -1 | 1 | -1.6322 | -0.8164 |
| -1 | -1 | 1 | -1 | -0.8164 | -0.0004 |
| -1 | -1 | 1 | 1 | -0.8164 | 0.8164 |
| -1 | 1 | **-1** | **-1** | **-0.0004** | **-0.8164** |
| -1 | 1 | -1 | 1 | -0.0004 | 0.0004 |
| -1 | 1 | 1 | -1 | 0.8164 | 0.8164 |
| -1 | 1 | 1 | 1 | 0.8164 | 1.6332 |
| 1 | -1 | **-1** | **-1** | **-0.8164** | **-1.6332** |
| 1 | -1 | -1 | 1 | -0.8164 | -0.8164 |
| 1 | -1 | 1 | -1 | 0.0004 | -0.0004 |
| 1 | -1 | 1 | 1 | 0.0004 | 0.8164 |
| 1 | 1 | **-1** | **-1** | **0.8164** | **-0.8164** |
| 1 | 1 | -1 | 1 | 0.8164 | 0.0004 |
| 1 | 1 | 1 | -1 | 1.6332 | 0.8164 |
| 1 | 1 | 1 | 1 | 1.6332 | 1.6332 |

***Table No. 2.2:***

Channel inputs and resulting centers:

$H_1(z) = 0.2 + 0.3z^{-1} + 1.0z^{-2} + 0.3z^{-3}$,

$m = 2$

| i | $s_i(k-0)$ | $s_i(k-1)$ | $s_i(k-2)$ | $s_i(k-3)$ | $s_i(k-4)$ | $c_{i,0}$ | $c_{i,1}$ |
|---|---|---|---|---|---|---|---|
| 1 | -1 | -1 | -1 | -1 | -1 | -1.8 | -1.8 |
| 2 | -1 | -1 | -1 | -1 | 1 | -1.8 | -1.2 |
| 3 | -1 | -1 | -1 | 1 | -1 | -1.2 | 0.2 |
| 4 | -1 | -1 | -1 | 1 | 1 | -1.2 | 0.8 |
| 5 | -1 | -1 | 1 | -1 | -1 | 0.2 | -1.2 |
| 6 | -1 | -1 | 1 | -1 | 1 | 0.2 | -0.6 |
| 7 | -1 | -1 | 1 | 1 | -1 | 0.8 | 0.8 |
| 8 | -1 | -1 | 1 | 1 | 1 | 0.8 | 1.4 |
| 9 | -1 | 1 | -1 | -1 | -1 | -1.2 | -1.4 |
| 10 | -1 | 1 | -1 | -1 | 1 | -1.2 | -0.8 |
| 11 | -1 | 1 | -1 | 1 | -1 | -0.6 | 0.6 |
| 12 | -1 | 1 | -1 | 1 | 1 | -0.6 | 1.2 |
| 13 | -1 | 1 | 1 | -1 | -1 | 0.8 | -0.8 |
| 14 | -1 | 1 | 1 | -1 | 1 | 0.8 | -0.2 |
| 15 | -1 | 1 | 1 | 1 | -1 | 1.4 | 1.2 |
| 16 | -1 | 1 | 1 | 1 | 1 | 1.4 | 1.8 |
| 17 | 1 | -1 | -1 | -1 | -1 | -1.4 | -1.8 |
| 18 | 1 | -1 | -1 | -1 | 1 | -1.4 | -1.2 |
| 19 | 1 | -1 | -1 | 1 | -1 | -0.8 | 0.2 |
| 20 | 1 | -1 | -1 | 1 | 1 | -0.8 | 0.8 |
| 21 | 1 | -1 | 1 | -1 | -1 | 0.6 | -1.2 |
| 22 | 1 | -1 | 1 | -1 | 1 | 0.6 | -0.6 |
| 23 | 1 | -1 | 1 | 1 | -1 | 1.2 | 0.8 |
| 24 | 1 | -1 | 1 | 1 | 1 | 1.2 | 1.4 |
| 25 | 1 | 1 | -1 | -1 | -1 | -0.8 | -1.4 |
| 26 | 1 | 1 | -1 | -1 | 1 | -0.8 | -0.8 |
| 27 | 1 | 1 | -1 | 1 | -1 | -0.2 | 0.6 |
| 28 | 1 | 1 | -1 | 1 | 1 | -0.2 | 1.2 |
| 29 | 1 | 1 | 1 | -1 | -1 | 1.2 | -0.8 |
| 30 | 1 | 1 | 1 | -1 | 1 | 1.2 | -0.2 |
| 31 | 1 | 1 | 1 | 1 | -1 | 1.8 | 1.2 |
| 32 | 1 | 1 | 1 | 1 | 1 | 1.8 | 1.8 |

***Table No. 2.3:***

Channel inputs and resulting centers:

$H_1(z) = 0.2 + 0.3z^{-1} + 1.0z^{-2} + 0.3z^{-3}$,

$m = 2$

| i | $s_i(k-0)$ | $s_i(k-1)$ | $s_i(k-2)$ | $s_i(k-3)$ | $s_i(k-4)$ | $c_{i,0}$ | $c_{i,1}$ |
|---|---|---|---|---|---|---|---|
| 1 | -1 | -1 | -1 | -1 | -1 | -1.8 | -1.8 |
| 2 | -1 | 1 | -1 | -1 | -1 | -1.2 | -1.4 |
| 3 | 1 | -1 | -1 | -1 | -1 | -1.4 | -1.8 |
| 4 | 1 | 1 | -1 | -1 | -1 | -0.8 | -1.4 |
| 5 | -1 | -1 | -1 | -1 | 1 | -1.8 | -1.2 |
| 6 | -1 | 1 | -1 | -1 | 1 | -1.2 | -0.8 |
| 7 | 1 | -1 | -1 | -1 | 1 | -1.4 | -1.2 |
| 8 | 1 | 1 | -1 | -1 | 1 | -0.8 | -0.8 |
| 9 | -1 | -1 | -1 | 1 | -1 | -1.2 | 0.2 |
| 10 | -1 | 1 | -1 | 1 | -1 | -0.6 | 0.6 |
| 11 | 1 | -1 | -1 | 1 | -1 | -0.8 | 0.2 |
| 12 | 1 | 1 | -1 | 1 | -1 | -0.2 | 0.6 |
| 13 | -1 | -1 | -1 | 1 | 1 | -1.2 | 0.8 |
| 14 | -1 | 1 | -1 | 1 | 1 | -0.6 | 1.2 |
| 15 | 1 | -1 | -1 | 1 | 1 | -0.8 | 0.8 |
| 16 | 1 | 1 | -1 | 1 | 1 | -0.2 | 1.2 |
| 17 | -1 | -1 | 1 | -1 | -1 | 0.2 | -1.2 |
| 18 | -1 | 1 | 1 | -1 | -1 | 0.8 | -0.8 |
| 19 | 1 | -1 | 1 | -1 | -1 | 0.6 | -1.2 |
| 20 | 1 | 1 | 1 | -1 | -1 | 1.2 | -0.8 |
| 21 | -1 | -1 | 1 | -1 | 1 | 0.2 | -0.6 |
| 22 | -1 | 1 | 1 | -1 | 1 | 0.8 | -0.2 |
| 23 | 1 | -1 | 1 | -1 | 1 | 0.6 | -0.6 |
| 24 | 1 | 1 | 1 | -1 | 1 | 1.2 | -0.2 |
| 25 | -1 | -1 | 1 | 1 | -1 | 0.8 | 0.8 |
| 26 | -1 | 1 | 1 | 1 | -1 | 1.4 | 1.2 |
| 27 | 1 | -1 | 1 | 1 | -1 | 1.2 | 0.8 |
| 28 | 1 | 1 | 1 | 1 | -1 | 1.8 | 1.2 |
| 29 | -1 | -1 | 1 | 1 | 1 | 0.8 | 1.4 |
| 30 | -1 | 1 | 1 | 1 | 1 | 1.4 | 1.8 |
| 31 | 1 | -1 | 1 | 1 | 1 | 1.2 | 1.4 |
| 32 | 1 | 1 | 1 | 1 | 1 | 1.8 | 1.8 |

Table 2 differs from Table 1 as the columns are shifted based delay.

***Table No. 2.4:***

Subsets based on $\hat{\boldsymbol{s}}_i(k-d)$ for channel $H_1(z)$

a) Input vector: **-1, -1**

| i | $c_{i,0}$ | $c_{i,-1}$ | $(\lambda_{\max})_i$ |
|---|---|---|---|
| 4 | -0.8 | -1.4 | 7.00048 |
| 2 | -1.2 | -1.4 | 9.16614 |
| 3 | -1.4 | -1.8 | 12.1604 |
| 1 | -1.8 | -1.8 | 17.2778 |
| 17 | 0.2 | -1.2 | 7.0005 |
| 19 | 0.6 | -1.2 | 9.16614 |
| 18 | 0.8 | -0.8 | 12.1605 |
| 20 | 1.2 | -0.8 | 17.2778 |

b) Input vector: **-1, 1**

| i | $c_{i,0}$ | $c_{i,-1}$ | $(\lambda_{\max})_i$ |
|---|---|---|---|
| 8 | -0.8 | -0.8 | 7.00049 |
| 7 | -1.2 | -0.8 | 9.16614 |
| 6 | -1.4 | -1.2 | 12.1604 |
| 5 | -1.8 | -1.2 | 17.2779 |
| 21 | 0.2 | -0.6 | 7.00049 |
| 23 | 0.6 | -0.6 | 9.16618 |
| 22 | 0.8 | -0.2 | 12.1605 |
| 24 | 1.2 | -0.2 | 17.2779 |

c) Input vector: **1, -1**

| i | $c_{i,0}$ | $c_{i,-1}$ | $(\lambda_{\max})_i$ |
|---|---|---|---|
| 12 | -0.2 | 0.6 | 7.00048 |
| 10 | -0.6 | 0.6 | 9.16614 |
| 11 | -1.8 | 0.2 | 12.1605 |
| 9 | -1.2 | 0.2 | 17.2779 |
| 25 | 0.8 | 0.8 | 7.00049 |
| 27 | 1.2 | 0.8 | 9.16615 |
| 26 | 1.4 | 1.2 | 12.1605 |
| 28 | 1.8 | 1.2 | 17.2779 |

d) Input vector: **1, 1**

| i | $c_{i,0}$ | $c_{i,-1}$ | $(\lambda_{\max})_i$ |
|---|---|---|---|
| 16 | -0.2 | 1.2 | 7.00048 |
| 14 | -0.6 | 1.2 | 9.16614 |
| 15 | -1.8 | 0.8 | 12.1604 |
| 13 | -1.2 | 0.8 | 17.2779 |
| 29 | 0.8 | 1.4 | 7.00049 |
| 31 | 1.4 | 1.4 | 9.16614 |
| 30 | 1.2 | 1.8 | 12.1604 |
| 32 | 1.8 | 1.8 | 17.2779 |

Fig. 2.3: BER plot for $H_1(z)=0.2+0.3z^{-1}+1.0z^{-2}+0.3z^{-3}$, m=2, d=2

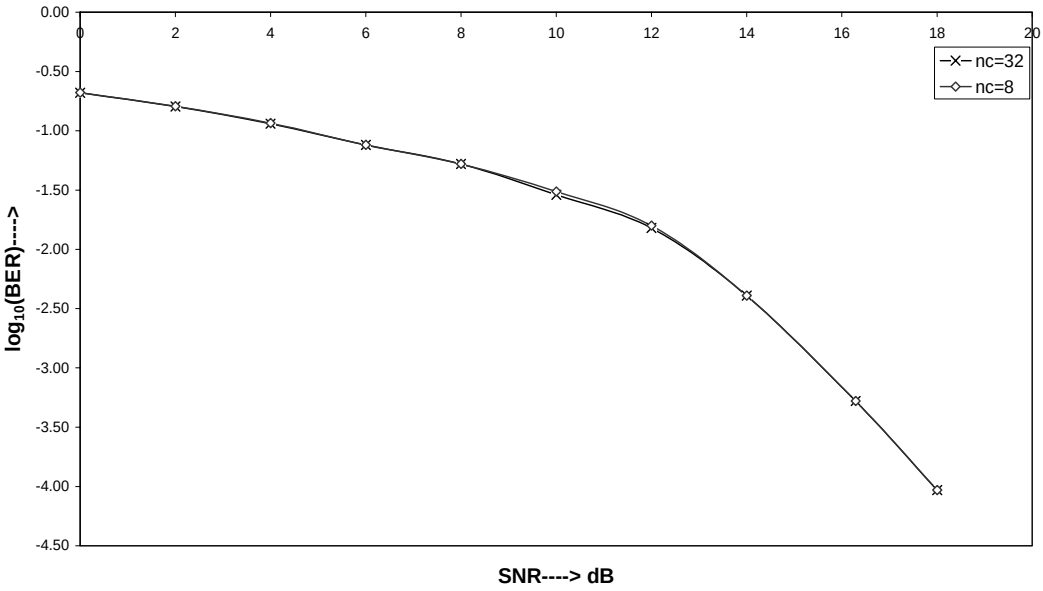



Fig. 2.4: BER plot for $H_1(z)=0.2+0.3z^{-1}+1.0z^{-2}+0.3z^{-3}$, m=4, d=2 and $n_b$=4

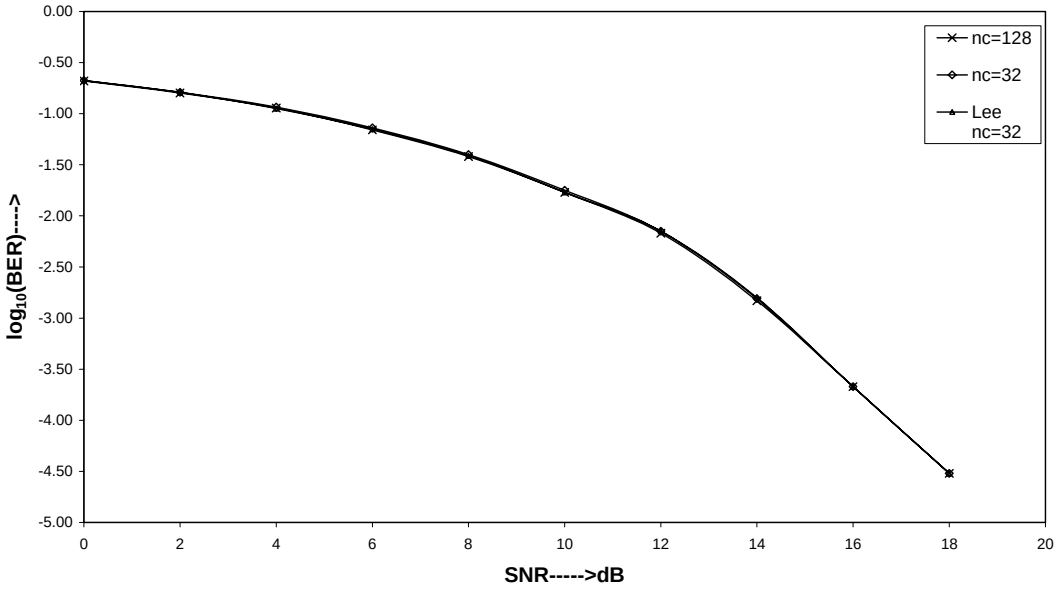


Fig. 2.5: BER plot for $H_2(z)=-0.2052-0.5131z^{-1}+0.7183z^{-2}+0.3695z^{-3}+0.2052z^{-4}$, m=3, d=2 and $n_b=4$

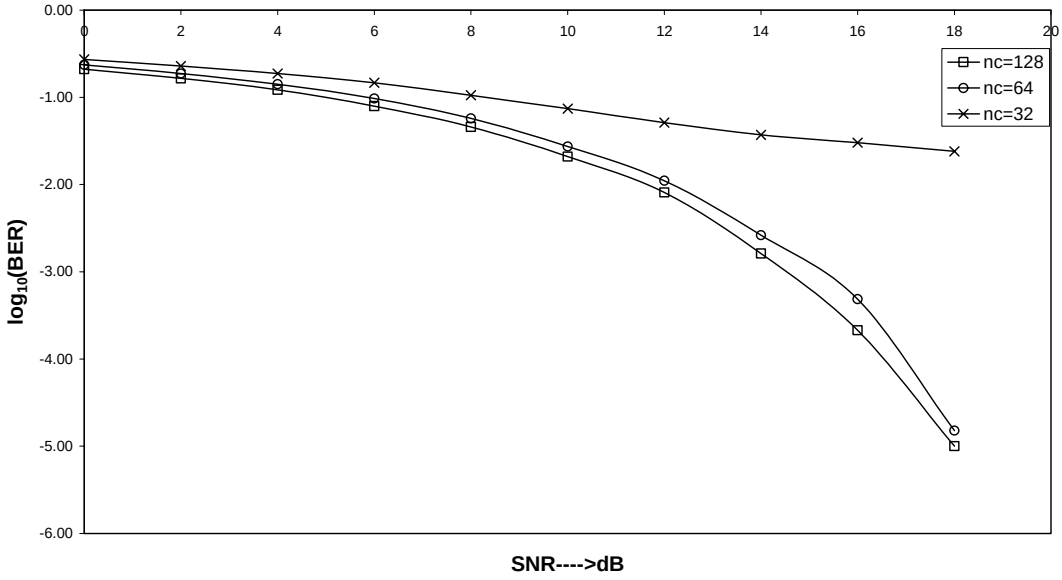



Fig. 2.6: BER plot for $H_1(z)=0.2+0.3z^{-1}+1.0z^{-2}+0.3z^{-3}$, m=4, d=2 and $n_b=4$ with DFE

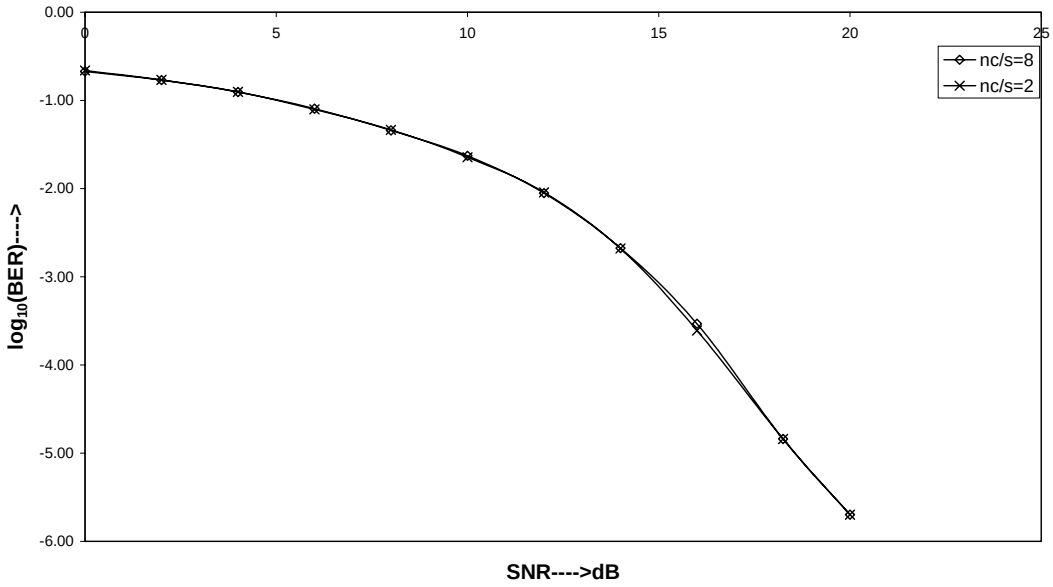
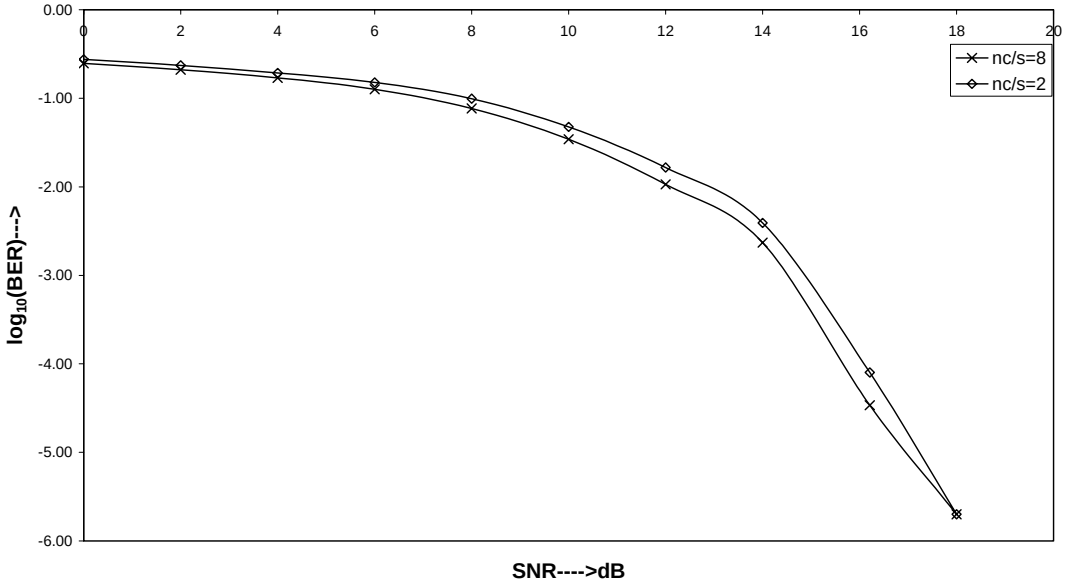


**Fig. 2.7: BER plot for $H_3(z)=0.4084+0.8164z^{-1}+0.4084z^{-2}$, m=3, d=2 and $n_b$=2 with DFE**



# Chapter 3

# System identification

System identification or system modeling deals with the design of an adaptive system or a mathematical model which can closely approximate the performance of an unknown system or a plant (as in control system terminology). In other words system modeling is the problem of estimating a system that transforms inputs into outputs given a set of examples of input-output pairs [5]. The free parameters of the adaptive system are adjusted till the mean square error between the plant output and the output of the adaptive system reaches a minimum. Most systems encountered in practice are characterized by nonlinear response and require nonlinear mapping. Hence a nonlinear model offers more degree of freedom and approaches an optimal solution. Mathematical modelling has been enhanced by the application of nonlinear models like neural networks (see [1], [3], [33]), fuzzy logics (see [30]-[31], [34]) or a combination of both (see [29], [30], [32] & [35]).

## 3.1 Neuro-Fuzzy systems

Neural networks provide an edge over other adaptive systems due to their nonlinear architecture and better learning capability. Further a system designed using neural networks is more robust to the time varying conditions. The fuzzy logic models provide more precision when used as interpolators. Further fuzzy logics allow the representation of the inputs to a system using the knowledge of the experts. The relation between the inputs and the output is defined using fuzzy IF-THEN rules, example, '*IF x is low THEN y is medium*'. Recently the fuzzy logic systems are being trained in the same manner as the neural networks [36].

This has motivated many researchers to develop neuro-fuzzy models for system identification. The basic idea underlying such an approach is to initially relate the inputs to the outputs in a comprehensive manner (using fuzzy rules) and then optimize the system using a network learning algorithm. Takagi et al [31] developed a fuzzy model called the TSK (Takagi-Sugeno-Kang) model that led to its wide practical application in control [30], [37] prediction and inference. The TSK model translates the premise part of the fuzzy IF-THEN rules into a nonfuzzy equation of the input variables, say, a weighted average of the rule nodes output thereby eliminating the need for defuzzification. This structure offers more simplicity and precision when there is an insufficient knowledge about the system output. The Adaptive Network

based Fuzzy Inference System (ANFIS), chiefly inspired by the TSK model, is an efficient architecture that offers a more transparent and efficient way of relating the input-output data to a system.

## 3.2 ANFIS

Jang proposed a benchmark neuro-fuzzy architecture [32] that effectively integrates the idea of the fuzzy systems and the neural networks. The ANFIS provides a representation of the prior knowledge into a set of constraints (network topology) to reduce the optimization search space based on the fuzzy systems. An adaptive scheme is used for the fuzzy controlled parametric tuning based on the back-propagation mechanism popular in neural networks. The main difference between the ANFIS and the TSK model lies in the fact that the premise part is adaptive and hence a more optimal solution can be reached. The architecture of the ANFIS is explained again with more emphasis laid on the contribution of the fuzzy system (FS) and the network (sub-filter or SF) part that is used to update the parameters of the FS part.

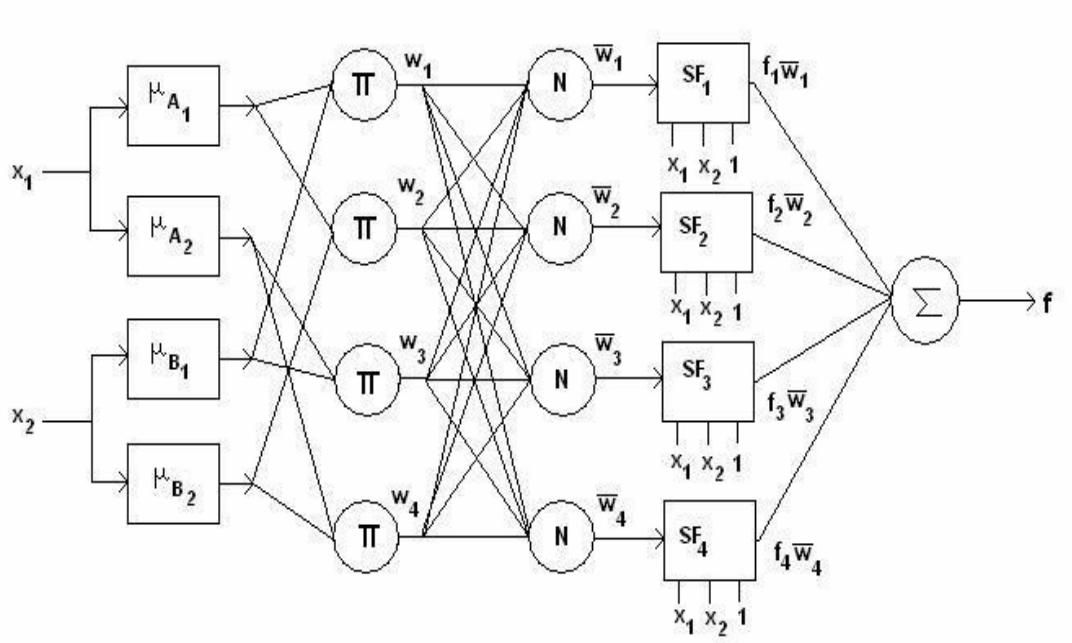



Fig. 3.1: ANFIS Structure

Consider an ANFIS structure with $n = 2$ inputs $[x_{1,k} \quad x_{2,k}]$ and $m = 2$ membership functions (bell type) for each input (*Fig 3.1*). Let $N = 2$ rule nodes be generated as depicted in the figure *Fig 3.1*. Hence there will be two sub-filter units in layer 4. Each sub-filter is composed of linear parameters $X_j = [p_j \quad q_j \quad r_j]$ with $1 \le j \le N$ . The output of the nodes in each layer may be given as follows

**Layer 1:**

The nodes of this layer contain the linguistic variables associated with the external input variables. The parameters of the membership functions (bell functions) representing the linguistic variables, $[a_{i,j} \;\; b_{i,j} \;\; c_{i,j}]$ are adaptive. The output of the nodes of this layer for an input $\mathbf{x}_k = [x_{k,1} \;\; x_{k,2}]$ is given as

$$O_j^1 = \mu_{A_k}(x_{k,j}) \tag{3.1}$$

$$\mu_{A_k}(x_{k,j}) = \frac{1}{1 + \left( \frac{x_{k,j} - c_{i,j}}{a_{i,j}} \right)^{2b_{i,j}}} \tag{3.2}$$

where $1 \le i \le m$, $j = 1,2$ and $k = 0,1,...,P-1$, for P patterns per epoch of training data.

**Layer 2:**

This layer contains the fuzzy rule nodes which are implemented using the fuzzy conjunction operator $\Pi$ or $\text{prod}(\cdot)$. The node output in this layer is represented as

$$O_j^2 = w_j = \mu_{A_i}(x_{k,1}) \cdot \mu_{B_i}(x_{k,2}) \tag{3.3}$$

**Layer 3:**

The nodes of this layer output the normalized output of the corresponding node in layer 2. This may be expressed as

$$O_j^3 = \overline{w}_j = \frac{w_j}{\sum_{j=1}^{N} w_j} \tag{3.4}$$

**Layer 4:**

This layer contains a sub-filter corresponding to each rule node. The sub-filter parameters $[p_i \;\; q_i \;\; r_i]$ are adaptive. The sub-filter operates as a linear combiner where the inputs are scaled by the parameters of the sub-filter and finally added to give an output. The output of each sub-filter is expressed as

$$O_i^4 = f_i \overline{w}_i = (\overline{w}_i x_{k,1}) \cdot p_i + (\overline{w}_i x_{k,2}) \cdot q_i + (\overline{w}_i) \cdot r_i \tag{3.5}$$

**Layer 5:**

This layer provides the network output which is expressed as the sum of the node outputs in layer 4.

$$O_1^5 = f = \sum_i f_i \cdot \overline{w_i} = \frac{\sum_i f_i \cdot w_i}{\sum_i w_i} \tag{3.6}$$

The ANFIS contains two layers (Layer 1 & Layer 4) of adaptive parameters. A hybrid learning rule is applied to train the ANFIS since it is composed of adaptive parameters belonging to two adaptive systems. The training is accomplished in two passes namely, the forward pass and the backward pass. In the forward pass the training data set (or epoch) is shown to the network while keeping the fuzzy parameters (otherwise called premise parameters) fixed. The error between the target value and the network output is calculated for each ensemble of the epoch and the parameters of the sub-filter (otherwise called consequent parameters) are updated recursively using the method of least squares. This may be reproduced as follows

$$\begin{aligned} X_{i+1} &= X_i + S_{i+1} a_{i+1} (b_{i+1}^T - a_{i+1}^T X_i) \\ S_{i+1} &= S_i - \frac{S_i a_{i+1} a_{i+1}^T S_i}{1 + a_{i+1}^T S_i a_{i+1}} \qquad \text{with } i = 0,1,..,P-1 \end{aligned} \tag{3.7}$$

$X_i = [p_i \;\; q_i \;\; r_i]^T$ represents the sub-filter parameters, $a_{i+1} = [x_{i+1,1} \;\; x_{i+1,2} \;\; 1]^T$ is the training input vector, $b_i$ is the desired data and $S_i$ is the covariance matrix for the $i^{th}$ pattern of the epoch, with the initial value given by $S_0 = \gamma I$, where $\gamma$ is an large positive number and $I$ is an identity matrix. In the backward pass, the parameters of the sub-filter are kept fixed while the parameters of the fuzzy membership functions are tuned using batch backpropagation method. This is done by calculating the network error for each pattern of the epoch and calculating the gradient of the output error square with respect to the parameters (using the chain rule) which are to be updated. When the entire epoch has been presented to the network, the average of the gradients is found over the entire epoch. Then the parameters are updated using the LMS algorithm.

### 3.2.1 Cascade Interpretation

The ANFIS can be interpreted as a combination of two adaptive systems, namely the fuzzy part (FS) and the network part (SF), in cascade trying to optimize the overall structure through a hybrid learning technique. The architecture of the

ANFIS (Fig. 3.1) shows that the network or sub-filter part is dependent on the fuzzy system part. In other words the number of sub-filter (SF) units is always dependent on the number of fuzzy IF-THEN rule nodes. This situation indicates that there is no freedom in the selection of the sub-filter units and this leads to an inflexible structure. The ANFIS output over an entire epoch may be expressed in matrix form as follows

$$A \cdot X = B \tag{3.8}$$

The equation (3.8) shows the relationship between the input matrix $A$, the parameter matrix $X$ and the desired output matrix $B$. This may be expanded as

$$\sum_{j=1}^{N} [w_j x_{i+1,1} \;\; w_j x_{i+1,2} \;\; w_j 1] \cdot [p_j \;\; q_j \;\; r_j]^T = b_{i+1} \tag{3.9}$$

The row vector of the matrix $A$ corresponding to equations (3.7) and (3.9) can be given as

$$a_{i+1} = [\mathrm{w}_j x_{i+1.1} \;\; w_j x_{i+1,2} \;\; w_j] \tag{3.10}$$

It may be observed in equation (3.5) that the external inputs $[x_{i+1,1} \;\; x_{i+1,2} \;\; 1]$ of the sub-filter units are pre-multiplied by the corresponding node outputs of layer 3 $(\overline{w}_i)$ before going to the sub-filter units. Hence the dependency of the sub-filter units on the fuzzy system cannot be disregarded. The equation (3.5) may be visualized as in figure *Fig 3.2*.

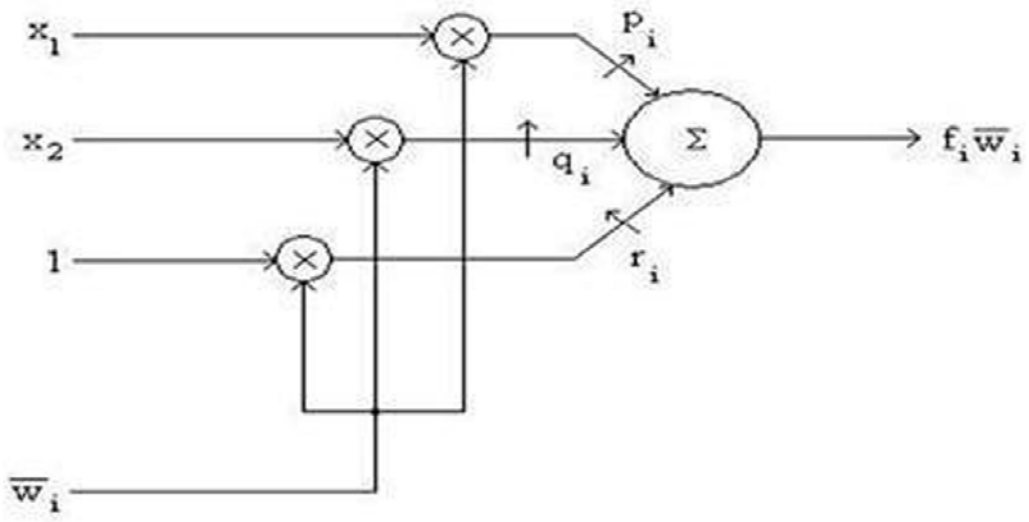



**Fig 3.2: Cascade interpretation**

Thus a constraint is placed in the selection of the sub-filter units subject to the number of fuzzy rule nodes $(w_i)$. But under the assumption that the ANFIS is composed of two independent adaptive systems it becomes inevitable that the choice of any number of elements in a structure can be made discretely. This can be achieved by the following interpretation.

### 3.2.2 Parallel Interpretation

The equation (3.5) may be rearranged as follows

$$O_i^4 = f_i \overline{w}_i = (p_i x_{k,1} + q_i x_{k,2} + r_i) \cdot \overline{w}_i \qquad (3.11)$$

The equation (3.11) is functionally equivalent to equation (3.5) though it leads to a different interpretation. It implies that the external inputs are applied to the sub-filter part and the fuzzy system separately. The outputs of both the systems are multiplied to give an output equivalent to those of the nodes in layer 4 of the ANFIS. In other words, the equation (3.11) suggests that the independent inputs are given to the sub-filter units and their outputs are post-multiplied by the corresponding outputs of layer 3 of the ANFIS. This may be illustrated as in the figure *Fig.3.3*. This kind of interpretation allows us to select the number of sub-filter units ahead of the number of rule nodes since the output of the sub-filter units is not bounded by the number of rule nodes of the fuzzy system.

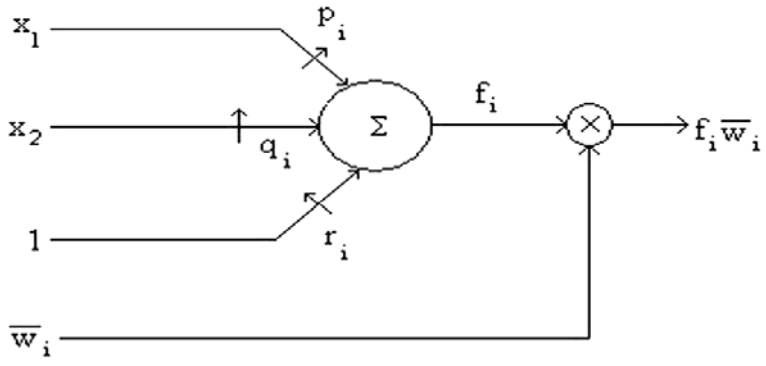



Fig 3.3: Parallel Interpretation

### 3.2.3 What is the difference?

The figures *Fig 3.2* and *Fig 3.3* are functionally equivalent to each other but provide interesting aspects of such representations when we analyze the role of the sub-filters and fuzzy system in the ANFIS with respect to the two interpretations. In the cascade interpretation, the ANFIS is purely a fuzzy inference system with the role of the sub-filters reduced to a set of adaptive parameters required for fine tuning the fuzzy parameters. Thus once the number of fuzzy IF-THEN rules is defined, a sub-filter unit is apparently chosen for each rule node. In practice the fuzzy membership functions are chosen by partitioning the input data range and symmetrically placing them while satisfying the condition for $\varepsilon$ completeness [38]. The ANFIS is usually composed of all possible rules that can be defined for a given linguistic set. Although an increase in the number of linguistic variables brings more clarity in defining the rules, it also leads to more structural complexity since as many sub-filters are to be included. For example, consider an ANFIS structure with 5 inputs and 3 membership functions per input. This leads to $N = 3^5 = 243$ rules. Obviously this is a large number to be considered, further there is a possibility of redundant and insignificant rule nodes.

The other interpretation for the ANFIS architecture lays more emphasizes on the role of the sub-filters when compared to the rule nodes of the fuzzy system. In fact the fuzzy rule nodes may be considered as scaling factors for the sub-filter units. Alternatively other type of weighting functions can be applied to the sub-filter units instead of a fuzzy weighting. The main requirement for such weighting functions is that their output should be in the range [0, 1].

ANFIS (or other neuro-fuzzy models) provide a grey-box modelling of an unknown system such that a partial transparency exists within the model due to the initial fuzzy inference provided by the rules. But as the fuzzy membership functions are adapted to obtain an optimal performance or approximation to the unknown system, the fuzzy rules lose their meaning and become simple weighting functions for the sub-filters. Finally the sub-filters assume the role of functional approximators. One advantage of the parallely weighted sub-filter approach is that it allows us to choose alternate weighting functions in situations where the fuzzy rules become ambiguous. Four adaptive networks have been developed based on the concept of

alternative weighting functions. These adaptive systems based on parellely weighted sub-filter approach perform close to the performance of the ANFIS.

## 3.3 Proposed Structures

The proposed structures called 'Parallely Weighted Sub-Filter Networks' (see Fig 3.4) perform in equivalence to the ANFIS though they do not contain a fuzzy inference part. These structures are motivated by the parallel interpretation of the ANFIS architecture. They utilize the concept of scaling the output of the sub-filters using normalized weighting functions $\overline{w}_i$. Thus the overall network output resembles the ANFIS output equation (3.6). Four such architectures are discussed below

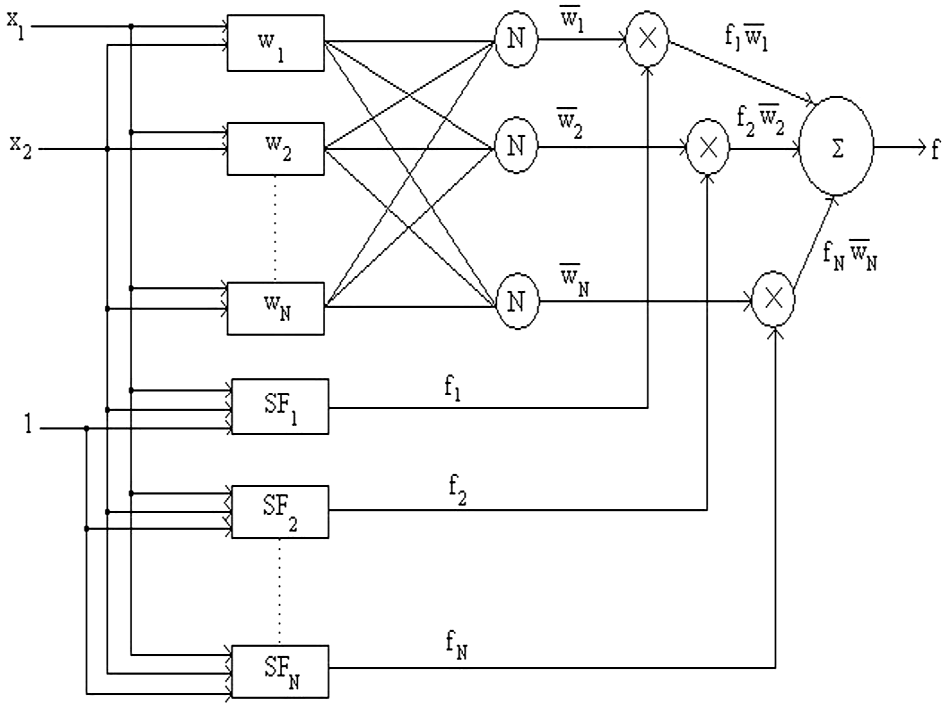



**Fig. 3.4: Parallely weighted sub-filter network**

*SF: Sub-Filter, w: Weighting Function

### 3.3.1 RBF weighted sub-filter approach (RBF-SF)

Jang et al [39] demonstrated the equivalence between the Radial Basis Function (RBF) network and a fuzzy inference system. When gaussian membership functions are used in the layer 1 of ANFIS then the rule nodes formed in layer 2 by the product of the gaussian membership functions resemble the gaussian kernels (RBF nodes). Thus the fuzzy part of the ANFIS may be replaced by the RBF nodes directly

by taking the center and width information for the RBF kernels from the rule nodes. Thus the sub-filters are now weighted by the normalized RBF node outputs. The overall network response for a two input network may be expressed as

$$f = \sum_{i=1}^{N} f_i \overline{w}_i = \frac{\sum_{i=1}^{N} f_i w_i}{\sum_{i=1}^{N} w_i} \tag{3.12}$$

where $w_i = \exp\left( \frac{\| x_{k+1,j} - c_{i,j} \|^2}{a_i^2} \right)$ (3.13)

and $f_i = (p_i x_{k+1,1} + q_i x_{k+1,2} + r_i)$ (3.14)

Here the parameters $c_{i,j}$ and $a_i^2$ are the center and spread of the RBF nodes respectively with $1 \le i \le N$, $1 \le j \le 2$ and $0 \le k \le P-1$. They are updated using batch backpropagation (superSAB see [36]), while the sub-filter parameters $X_i$ are updated using recursive least squares (RLS).

In the superSAB method, every parameter (say $p$) to be adapted has its own learning rate which is also updated. When the gradient of the error square with respect to the parameter to be updated, changes sign over two successive epochs then the learning rate of that parameter is decreased by a factor $\eta_d$ chosen between 0.6 and 0.9. If the gradient of the error square with respect to the parameter to be updated, does not change sign over two successive epochs then the learning rate is increased by a factor $\eta_u$ chosen between 1.05 and 1.3. This may be expressed as

$$\eta_p(t) = \eta_d \cdot \eta_p(t-1) \quad \text{if} \quad \nabla_p e(t) \cdot \nabla_p e(t-1) < 0 \tag{3.15}$$

$$\eta_p(t) = \eta_u \cdot \eta_p(t-1) \quad \text{if} \quad \nabla_p e(t) \cdot \nabla_p e(t-1) > 0 \wedge \nabla_p e(t-1) \cdot \nabla_p e(t-2) \ge 0$$

### 3.3.2 Modified basis function weighted sub-filter approach (MBF-SF)

The parameters of the weighting functions take credit for the error produced during the backward pass and hence are adjusted to minimize the mean square of the error. When there are more parameters available to take the credit for the error then the error can be dissipated over the parameters as much as possible. Consider a two input parallely weighted sub-filter network as defined by equations (3.12) and (3.13). The equation (3.13) may be expressed in the form as follows

$$w_i = \exp(-[A_i x_1^2 + B_i x_2^2 + C_i x_1 + D_i x_2 + E_i]) \tag{3.16}$$

The index k denoting the pattern number in the epoch can be dropped from the input for notational convenience. The adaptive parameters $A_i$, $B_i$, $C_i$, $D_i$ and $E_i$ are initially chosen as follows

$$A_i = \frac{1}{a_i^2},\ B_i = \frac{1}{a_i^2},\ C_i = \frac{-2c_{i,1}}{a_i^2},\ D_i = \frac{-2c_{i,2}}{a_i^2} \text{ and } E_i = \frac{(c_{i,1}^2 + c_{i,2}^2)}{a_i^2}$$

The parameters of the weighting function are trained using the batch backpropagation algorithm (superSAB). Though the initial shape of the weighting functions resemble the RBF kernels, this property does not hold after training.

**3.3.3 Sigmoid function weighted sub-filter approach**

The sigmoid function is the most popular nonlinear activation function used in multilayer perceptron (MLP) networks. The adaptive parameters $L = [l_{1,n}\ \ l_{2,n}\ ...\ \ l_{i,n}]$ of the sigmoid function are used to linearly combine the external inputs to give an output $V$. A nonlinear transformation is performed over the output $\Phi_i \cdot V_i$ to give the weighting function output. The parameter $\Phi_i$ called the slope of the activation function. The parameters of the weighting function are adapted using backpropagation technique (superSAB). The most popular sigmoid function is the logistic function (Fig 3.5) giving an output between 0 and 1. The logistic function may be expressed as

$$w_i = \frac{1}{1 + \exp(-\Phi_i V_i)} \tag{3.17}$$

where $V_i = (l_{i,1}x_1 + l_{i,2}x_2 + ... + l_{i,n}x_n)$ for an $n$ input parallely weighted sub-filter network. The overall network response is given as in (3.12).

Having observed the above weighting functions, it becomes clear that the scaling factors have always been assumed to be positive. The equation (3.11) may be rewritten as below

$$f_i\overline{w}_i = (p_i x_1 + q_i x_2 + r_i).\overline{w}_i = (p_i.\overline{w}_i)x_1 + (q_i.\overline{w}_i)x_2 + (r_i.\overline{w}_i)1 \tag{3.18}$$

The equation (3.18) may further be written as

$$f_i\overline{w}_i = P_i x_1 + Q_i x_2 + R_i \tag{3.19}$$

The above equation suggests that scalar products can be either positive or negative depending on the sign of the sub-filter parameters while the weighting function output has been restricted to be strictly positive. An attempt is made to give the weighting functions more degree of freedom by allowing it to be either positive or negative. Thus the hyperbolic tangent function (Fig 3.6) which gives an output in the range [-1,

+1] is chosen for the weighting function. It is observed that the adaptive network designed still approaches the minimum root mean square error (RMSE) criterion. The weighting function of such a network is given as

$$w_i = \frac{1-\exp(-\Phi_i V_i)}{1+\exp(-\Phi_i V_i)} \tag{3.20}$$

The network response is same as equation (3.12). Also an adaptive gain factor $G_i$ can be used to scale the weighting function so that it escapes the condition that $w_i$ should range between 0 and 1. The parameters of the weighting function are updated using the batch backpropagation algorithm (superSAB).

$$w_i = G_i \frac{1-\exp(-\Phi_i V_i)}{1+\exp(-\Phi_i V_i)} \tag{3.21}$$

The sub-filter network weighted by the logistic sigmoid function is denoted by (Sig1-SF) while the network weighted by hyperbolic tangent function is represented by (Sig2-SF).

## 3.4 Simulations

Various functions have been used for evaluating the proposed architectures. Four functions have been presented although other modelling problems can be dealt by the ANFIS and its contemporary structures. In each case the root mean square error of 0.01 was set as a good performance criterion. The simulations were carried out in the presence of measurement noise of variance 0.01.

### 3.4.1 Modeling a 2D sinc function

The function to be approximated is a two dimensional sinc function expressed as a function

$$f_1(x_1, x_2) = \sin c(x_1, x_2) = \frac{\sin(x_1)}{x_1} \cdot \frac{\sin(x_2)}{x_2} \tag{3.22}$$

where the external inputs $(x_1; x_2)$ are in the range [-10, 10] was used by Jang [32]. 250 epochs containing 121 of training data patterns are generated.

To model this function using the ANFIS, select a fuzzy system having two inputs and 4 bell membership functions per input. The membership functions per input are chosen such that they cover the entire data range and maintain $\varepsilon$ completeness. This would result in 16 rule nodes and hence 16 sub-filters. The ANFIS produced a root mean square error of 0.008344 at the end of 250 epochs of training.

The above function was approximated using a network of 13 sub-filters with each sub-filter weighted by a RBF node. To select the parameters of the RBF kernels a clustering technique like k-means method may be applied over the training data set or alternately by selecting the rule nodes of the ANFIS that would have been formed if each input data were classified using four gaussian membership functions, as the RBF nodes. The parameters of the RBF nodes were trained using the superSAB method while the parameters of the sub-filter are adapted using the RLS algorithm. The root mean square error at the end of 250 epochs was 0.008747.

The above function was approximated using a network of 13 sub-filters with each sub-filter weighted by a modified basis function. The initial parameters of the modified basis function were chosen based on the RBF weighting function. The root mean square error at the end of 250 epochs was observed to be 0.008238.

The sigmoid function described by equations (3.17) was applied as weighting functions for the sub-filters and architectures was chosen with 13 sub-filter units and their respective weighting functions. The initial parameters of the sigmoid functions were set to small random numbers. A root mean square error of 0.01026 was recorded for the architecture at the end of 250 epochs of training. The figure *Fig. 3.7* shows the root mean square error (RMSE) versus the epoch number for the above architectures.

### 3.4.2 Modeling a 3 input nonlinear function

The training data for this example is obtained from the nonlinear function described below

$$f_2(x_1, x_2, x_3) = (1 + x_1^{0.5} + x_2^{-1} + x_3^{-1.5})^2 \qquad (3.23)$$

The above function was used by Jang [32] and Takagi et al [40]. The independent inputs are in the range [1, 6]. ANFIS [8] uses two bell membership functions for each input generating 8 rules. Therefore 8 sub-filters are used. 216 training patterns are obtained from equation (3.23) and 200 epochs of training was carried out to achieve a RMSE of 0.007682. The architectures proposed contain 7 sub-filters units and weighting functions. The performance of the architectures is shown in figure *Fig. 3.8*. The root mean square error recorded for each structure at the end of 200 epochs of training are given below in *Table 3.1*

### 3.4.3 Modeling a radial function

This example was used by Maechler et al [41] to model PPL (Projection Pursuit Learning) networks.

$$f_3(x_1, x_2) = 24.234(r^2(0.75 - r^2)) \tag{3.24}$$

where $r^2 = \left(x_1 - \frac{1}{2}\right)^2 + \left(x_2 - \frac{1}{2}\right)^2$ (3.25)

The training data was generated from equation (3.24) in the range $x_1, x_2 \in [0,1]$. 100 epochs of data containing 36 training data patterns were used during training. The ANFIS required 4 bell membership functions for each input since it failed to train with 3 membership functions for each input. This lead to 16 rule nodes and hence 16 sub-filters were used for learning. The total number of adaptive parameters in ANFIS was equal to 72. The other architectures required just 8 sub-filters and hence 9 weighting functions. The RBF weighted sub-filter network recorded the lowest number of adaptive parameters equal to 40. The root mean square error (*Fig. 3.9*) at the end of 100 epochs of training for the various architectures is recorded in *Table 3.2*.

### 3.4.4 Modeling a complicated interaction function

This example was also used by Maechler et al [41] to model PPL (Projection Pursuit Learning) networks.

$$f_3(x_1, x_2) = 1.9(1.35 + e^{x_1}\sin(13(x_1 - 0.6)^2)e^{-x_2}\sin(7x_2)) \tag{3.26}$$

The training data was generated from equation (3.26) in the range $x_1, x_2 \in [0,1]$. 100 epochs of data containing 36 training data patterns was used during training. The ANFIS required 3 bell membership functions for each input. This lead to 9 rule nodes and hence 9 sub-filters were used for learning. The total number of adaptive parameters in ANFIS was equal to 45. The other architecture required 9 sub-filters and hence 9 weighting functions. The RBF weighted sub-filter network required 8 sub-filter units. This network contained 40 adaptive parameters. The root mean square error (*Fig. 3.10*) at the end of 100 epochs of training for the various architectures is recorded in *Table 3.3*.

***Table 3.1*:** Modeling a 2D sinc function

| Index | Architecture | RMSE | No. of Parameters |
|---|---|---|---|
| 1 | Adaptive Network based Fuzzy Inference System | 0.008344 | (8*3+16*4) **72** |
| 2 | RBF weighted sub-filter network | 0.008747 | (13*2+13*3) **65** |
| 3 | Modified basis function weighted sub-filter network | 0.008238 | (13*5+13*3) **104** |
| 4 | Logistic sigmoid function weighted sub-filter network (slope tuning) | 0.01206 | (13*3+13*3) **78** |

***Table 3.2*:** Modeling a 3 input nonlinear function

| Index | Architecture | RMSE | No. of Parameters |
|---|---|---|---|
| 1 | Adaptive Network based Fuzzy Inference System | 0.007682 | (6*3+8*4) **50** |
| 2 | RBF weighted sub-filter network | 0.008433 | (7*2+7*4) **42** |
| 3 | Logistic sigmoid function weighted sub-filter network (slope =1, no slope tuning) | 0.009368 | (8*2+8*4) **48** |
| 4 | Hyperbolic tangent function weighted sub-filter network | 0.00529 | (7*3+7*4) **49** |

***Table 3.3*:** Modeling a radial function

| Index | Architecture | RMSE | No. of adaptive parameters |
|---|---|---|---|
| 1 | Adaptive Network based Fuzzy Inference System | 0.000160 | (8*3+16*3) **72** |
| 2 | RBF weighted sub-filter network | 0.00154 | (8*2+8*3) **40** |
| 3 | Modified basis function weighted sub-filter network | 0.000144 | (8*5+8*3) **64** |

***Table 3.4:*** Modeling a complicated interaction function

| Index | Architecture | RMSE | No. of adaptive parameters |
|---|---|---|---|
| 1 | Adaptive Network based Fuzzy Inference System | 0.002116 | (6*3+9*3) **45** |
| 2 | RBF weighted sub-filter network | 0.005631 | (8*2+8*3) **40** |
| 3 | Modified basis function weighted sub-filter network | 0.001441 | (8*5+8*3) **64** |

**Fig. 3.5: Logistic sigmoid function**

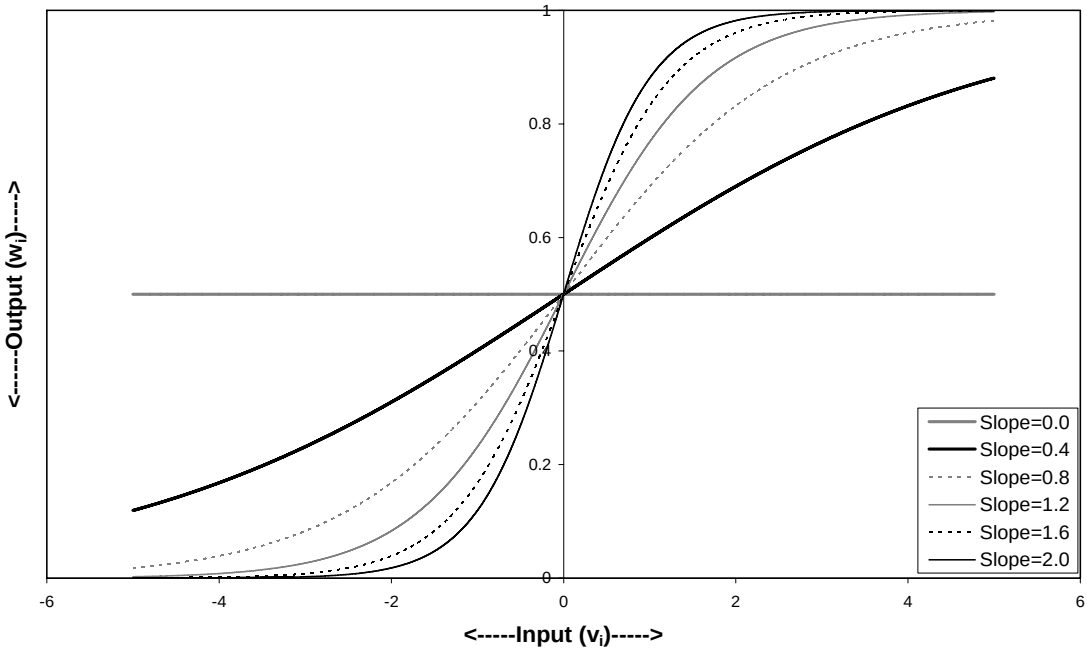



**Fig. 3.6: Hyperbolic tangent function**

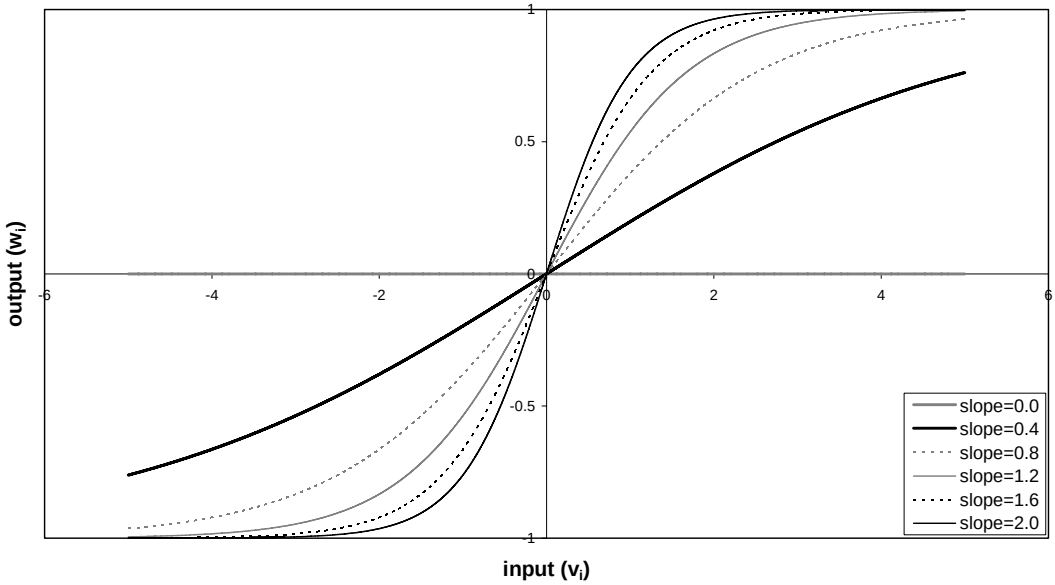


**Fig. 3.7: Modeling a 2D sinc function**

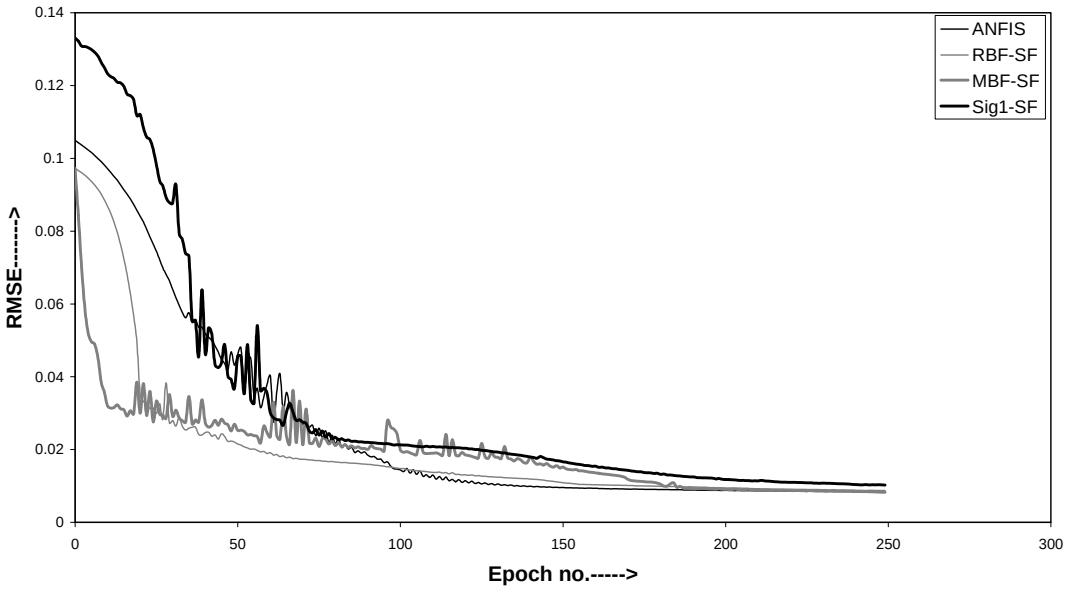



**Fig. 3.8: Modeling a 3 input nonlinear function**

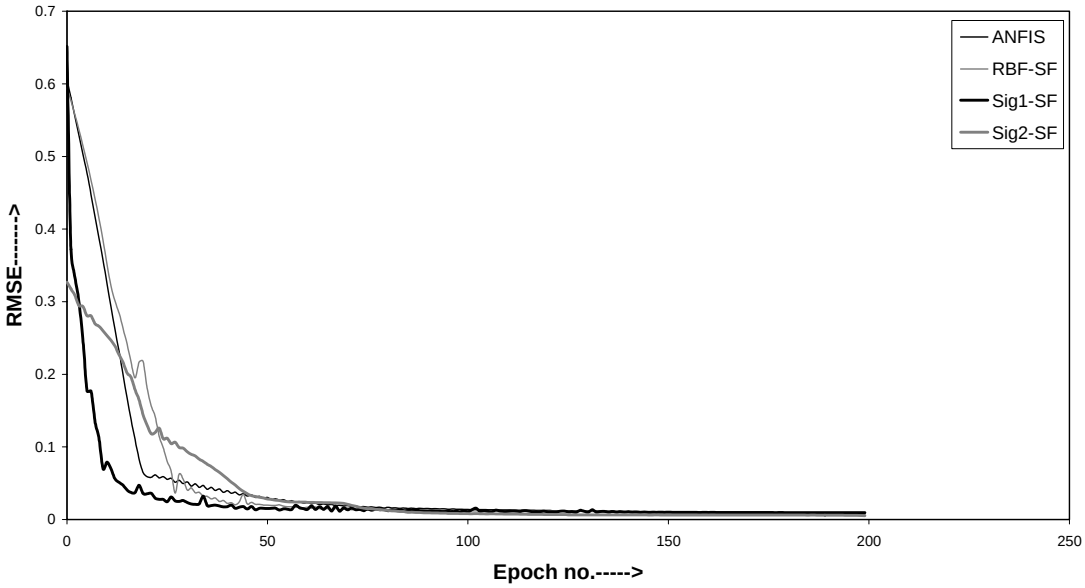


**Fig. 3.9: Modeling a Radial function**

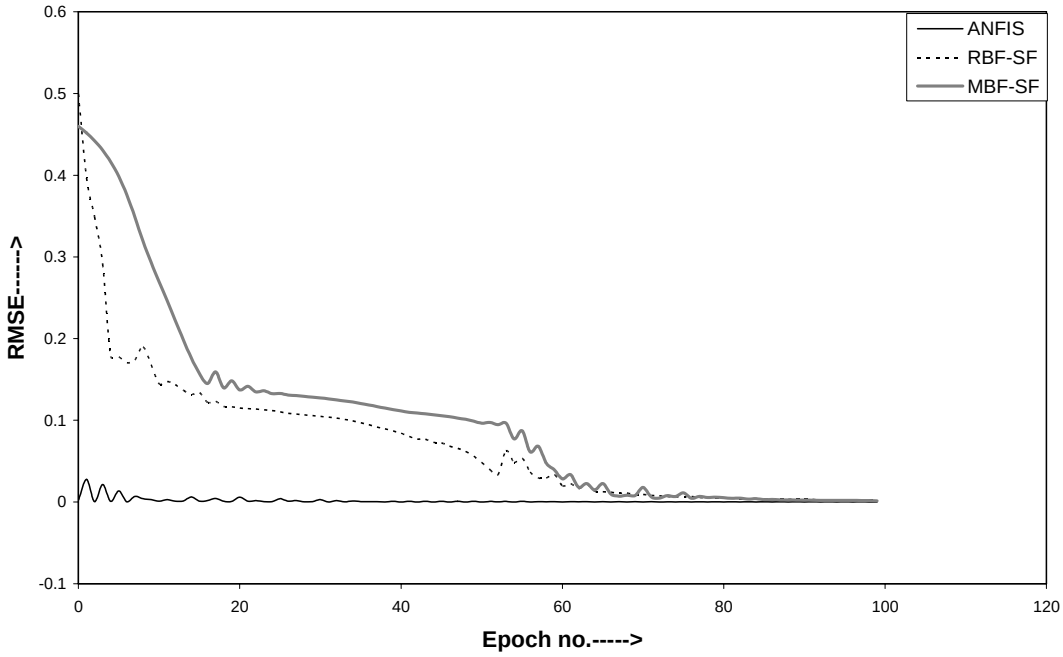
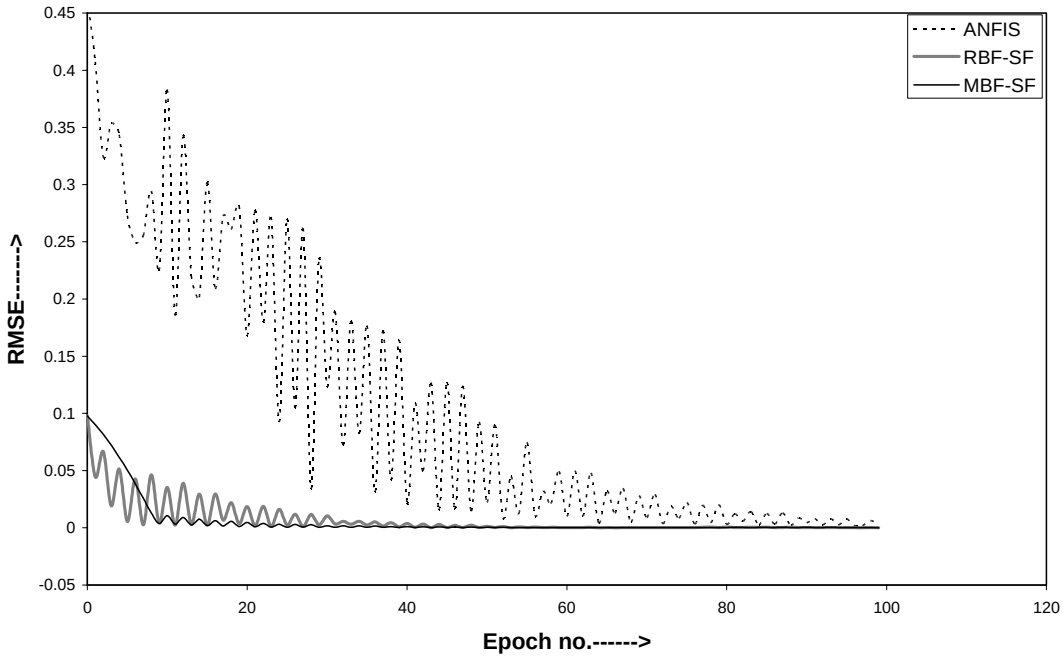



**Fig. 3.10: Modeling a complicated interaction function**



# Chapter 4

## Neuro-Fuzzy Models

The automation of the process in control and engineering areas has led to the design of several nonlinear system identification models that are used to improve the control performance and fault tolerance. The models that attract the most attention are the neuro-fuzzy models ([29], [35], [40] & [45]) for their accuracy and understandability. This is because of their ability to combine the principles of neural networks and fuzzy logics. They effectively utilize the interpolating capability of fuzzy systems and the adaptability of neural networks. Neuro-fuzzy models can be viewed as grey box models which draw a boundary between the neural network models and the fuzzy logic models. These models are basically fuzzy inference systems in which the rule nodes provide a functional mapping between the inputs and the outputs of the unknown system and the adaptive parameters are optimized using a suitable learning technique. Borgelt et al [36] showed how the adaptive techniques that are popular in neural networks training can be applied to fuzzy logic systems.

### 4.1 Adaptive network based fuzzy inference systems

Takagi et al [31] developed a fuzzy logic model (TSK model) with the fuzzy sets defined only in the premise part of the rule nodes while a non-fuzzy function is used in the consequent part of the rule nodes. The TSK model could give a reasonable performance when a weighted average operation was used for the non-fuzzy function. For instance a rule in a TSK model may be stated as

If $x_1$ is LOW and $x_2$ is LOW then $out = \overline{w}_1 x_1 + \overline{w}_2 x_2$ (4.1)

The adaptive network based fuzzy inference system (ANFIS) developed by Jang [32] stands as a landmark achievement in the neuro-fuzzy modeling. The main contribution of the ANFIS is the idea of expressing as network architecture, the main components of a fuzzy inference system: fuzzification, implication and defuzzification (if necessary). Nodes of the first hidden layer realize the linguistic terms of the linguistic variable associated to a given external input. The bell functions are used for expressing the linguistic variables. The nodes of the second layer called rule nodes define the conditions applied on the linguistic variables of the inputs.

These nodes use product operators (as the fuzzy conjunction operators). The third layer of the network includes nodes that normalize the fuzzy rule node outputs (outputs of nodes in the second layer). The fourth layer is composed of sub-filters that act as linear combiner of the external inputs. The number of sub-filters is directly obtained from the number of rule nodes. The fifth layer contains a single summing unit (or the network output) that combines the outputs of the nodes in preceding layer. The network is trained using a hybrid learning technique similar to the EM algorithm. Several ANFIS like architectures that follow the same strategy have been reported in literature. Even though these kind of systems have proven to deliver accurate performance, this is often achieved at the expense of the understandability of the fuzzy rules.

## 4.2 Limitations of ANFIS

Though the ANFIS is a milestone in the development of neuro-fuzzy models it has some limitations that motivate the design of alternate architectures. The first requirement of the ANFIS is that it needs an a priori decision on the number of linguistic variables for each external input. It roughly combines the linguistic terms associated with each input to give as many number of rules, hence the number of sub-filters associated with the rule nodes increases. This means that a large number of rules impose a burden on training the network and understanding the network. Secondly, the t-norm operator (fuzzy conjunction operator) used in the second layer should be a product operator since the gradient descent calculation used for updating the parameters of the linguistic terms requires that the nodes in every layer should be differentiable.

## 4.3 Alternate ANFIS structures

An attempt to develop alternate ANFIS architectures has been made with the idea of exploiting different kinds of t-norm operators and membership functions used in the first layer of ANFIS. A comparison between the architectures presented and the ANFIS is made in terms of performance, size of the network (pertaining to number of rule nodes and the number of adjustable parameters).

### 4.3.1 M-ANFIS

The ANFIS with modified rule nodes or M-ANFIS (*Fig. 4.1*) is an ANFIS structure which is composed of hybrid rule nodes. The ANFIS included only one differentiable t-norm operators for defining the rules. The product operator thus chosen is differentiable but imposed a constraint on the selection of the type of membership functions used to represent the linguistic terms associated with the input. The pretext in choosing a product operator was that when two membership functions whose output is in the range [0, 1] are multiplied, their product should also be in the range [0, 1]. This meant that only a gaussian or a bell function should be used as a membership function. Zhang et al [43] showed that the logical functions are continuous and differentiable. Hence the logical t-norm operator $\min(\cdot)$ has been used in the rule node definition.

The M-ANFIS is similar to the ANFIS network except that the second layer contains a modified rule node instead of the conjunction operator $prod(\cdot)$. Consider a two input M-ANFIS with $m$ linguistic terms for each external input $\mathbf{x}_k = [x_{k,1} \quad x_{k,2}]$. The type of membership function chosen is either bell function or gaussian function. The nodes of second layer, termed as rule nodes are given as follows

$$O_j^2 = w_j = \frac{\min(\mu_{A_i}(x_{k,1}), \mu_{B_i}(x_{k,2})) + prod(\mu_{A_i}(x_{k,1}), \mu_{B_i}(x_{k,2}))}{2} \qquad (4.2)$$

where $1 \le i \le m$, $j = 1,2$ and $k = 0,1,...,P-1$, for P patterns per epoch of training data. The $prod(\cdot)$ function computes the product of the input variables and the $\min(\cdot)$ function returns the minimum of the input variables. The nodes of this layer can be denoted by $H$, representing the hybrid rule node. A sub-filter is declared for each rule node as in the case of ANFIS. The learning mechanism for the M-ANFIS network is equivalent to that used in training ANFIS, the premise parameters are updated using gradient descent while keeping the parameters of the sub-filters fixed. The consequent parameters are updated using recursive least squares (RLS) while keeping the premise parameters fixed.

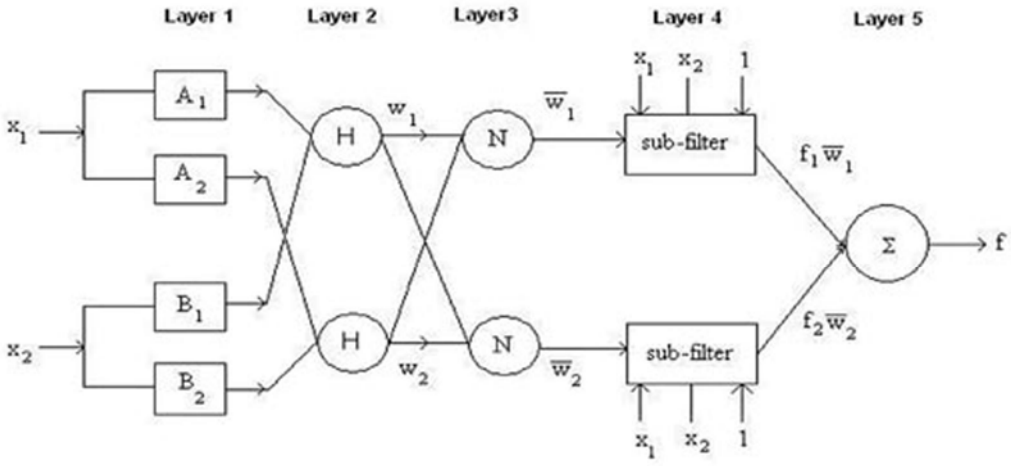



Fig . 4.1: M-ANFIS Architecture

## 4.3.2 ANFIS-2

The ANFIS-2 is an ANFIS network with two independent fuzzy inference systems. The architecture of the ANFIS-2 network is functionally equivalent to the original ANFIS structure. The structure of the ANFIS-2 (*Fig. 4.2*) network can be explained as follows

Consider a two input ANFIS-2 with two linguistic terms for each input. This structure contains two fuzzy inference systems and each fuzzy system is discussed separately

**Fuzzy inference system 1 or** $FIS_1$**:**

**Local Layer 1:**

The nodes of this layer contain the linguistic variables associated with the external input variables $\mathbf{x}_k = [x_{k,1} \;\; x_{k,2}]$ represented by bell membership function. The parameters $[a_{i,j} \;\; b_{i,j} \;\; c_{i,j}]$ of the bell membership function are adaptive. The output of the nodes of this layer are represented as

$$O_{j,i}^{11} = \mu_{j,A_i}^{1}(x_{k,j}) \tag{4.3}$$

where $i$ indicates the index of the linguistic term associated with the $j^{th}$ external input for the $k^{th}$ pattern of the epoch.

$$\mu_{j,A_i}^{1}(x_{k,j}) = \frac{1}{1+\left(\dfrac{x_{k,j}-c_{i,j}}{a_{i,j}}\right)^{2b_{i,j}}} \tag{4.4}$$

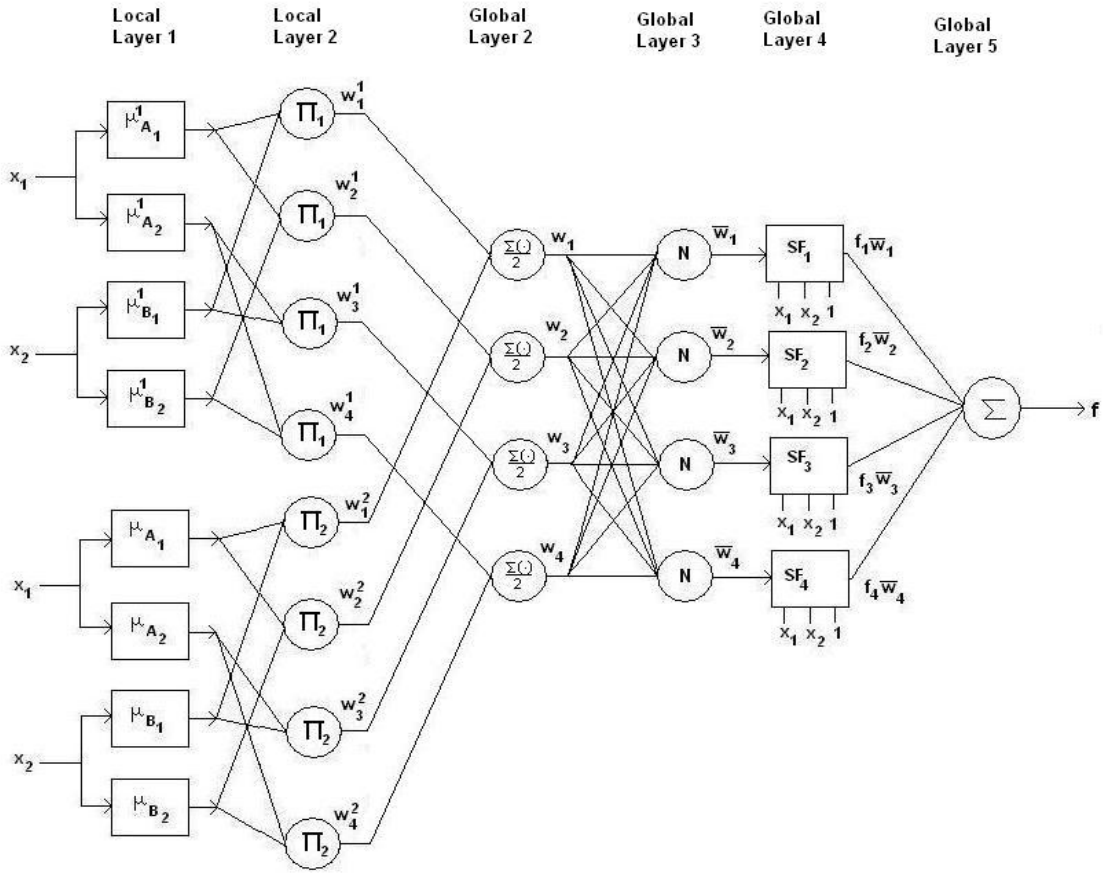



Fig 4.2: ANFIS-2 Structure

**Local Layer 2:**

This layer contains the local fuzzy rule nodes which are implemented using the fuzzy conjunction operator $\Pi_1$ or $\text{prod}(\cdot)_1$. The node output in this layer is represented as

$$O_j^{12} = w_j^1 = \mu_{1,A_i}^1(x_{k,1}) \cdot \mu_{2,A_i}^1(x_{k,2}) \tag{4.5}$$

**Fuzzy inference system 2 or $FIS_2$ :**

**Local Layer 1:**

The nodes of this layer contain the linguistic variables associated with the external input variables $\mathbf{x}_k = [x_{k,1} \;\; x_{k,2}]$ represented by gaussian membership function. The parameters $[\sigma_i \;\; C_{i,j}]$ of the gaussian membership function are adaptive. The output of the nodes of this layer are represented as

$$O_{j,i}^{21} = \mu_{j,A_i}^2(x_{k,j}) \tag{4.6}$$

$$\mu_{j,A_i}^2(x_{k,j}) = \exp\left(-\left(\frac{x_{k,j} - C_{i,j}}{\sigma_i}\right)^2\right) \tag{4.7}$$

**Local Layer 2:**

This layer contains the local fuzzy rule nodes which are implemented using the fuzzy conjunction operator $\Pi_2$ or $\text{prod}(\cdot)_2$. The node output in this layer is represented as

$$O_j^{22} = w_j^2 = \mu_{1,A_i}^2(x_{k,1}) \cdot \mu_{2,A_i}^2(x_{k,2}) \tag{4.8}$$

**Global Layer 2**:

This layer contains the global rule nodes which are denoted by $GR$. The node output of this layer are given by

$$O_j^2 = w_j = \left( \frac{w_j^1 + w_j^2}{2} \right) \tag{4.9}$$

**Global Layer 3:**

The nodes of this layer output the normalized output of the corresponding node in the global layer 2. This may be expressed as

$$O_j^3 = \overline{w}_j = \frac{w_j}{\sum_{j=1}^{N} w_j} \tag{4.10}$$

**Global Layer 4:**

This layer contains a sub-filter corresponding to each global rule node. The sub-filter parameters $[p_i \quad q_i \quad r_i]$ are adaptive. The sub-filter operates as a linear combiner where the inputs are scaled by the parameters of the sub-filter and finally added to give an output. The output of each sub-filter is expressed as

$$O_i^4 = f_i \overline{w}_i = (\overline{w}_i x_{k,1}) \cdot p_i + (\overline{w}_i x_{k,2}) \cdot q_i + (\overline{w}_i) \cdot r_i \tag{4.11}$$

**Global Layer 5:**

This layer provides the network output which is expressed as the sum of the node outputs in global layer 4.

$$O_1^5 = f = \sum_i f_i \cdot \overline{w_i} = \frac{\sum_i f_i \cdot w_i}{\sum_i w_i} \tag{4.12}$$

The ANFIS -2 is trained in the same manner as the ANFIS. The premise parameters which include the parameters of the gaussian membership function and those of the bell membership function are adjusted by gradient descent method while keeping the parameters of the sub-filters (consequent parameters) fixed. The consequent

parameters are updated using recursive least squares (RLS) while keeping the premise parameters fixed. The main advantage of the ANFIS-2 structure is that it can exhibit the same performance as the ANFIS using lesser number of sub-filter units. The total number of adaptive parameters in the ANFIS-2 is less than that of ANFIS.

### 4.3.3 Alternate ANFIS-2

An alternate ANFIS-2 structure can be constructed if the bell membership functions in the local layer 1 are replaced by triangular membership functions. It was observed that the alternate ANFIS-2 structure developed performed better than the original ANFIS. The number of free parameters of the alternate ANFIS-2 structure was still less compared to the ANFIS. The triangular membership function can be expressed for an external input $x_{k,j}$ as follows

$$\mu^{1}_{j,A_i}(x_{k,j}) = \max\left(\min\left(\left(\frac{x_{k,j}-a_{i,j}}{b_{i,j}-a_{i,j}}\right),\left(\frac{c_{i,j}-x_{k,j}}{c_{i,j}-b_{i,j}}\right)\right),0\right) \tag{4.13}$$

The derivative of the triangular membership function with respect to its parameters $[a_{i,j} \;\; b_{i,j} \;\; c_{i,j}]$ can be found using min-max differentiation [43].

## 4.4 Simulations

The same functions as used in the previous chapter are used to demonstrate the function approximation properties of the developed structures.

#### 4.4.1 Modeling a 2D sinc function

The function to be approximated is a two dimensional sinc function expressed as a function

$$f_1(x_1, x_2) = \sin c(x_1, x_2) = \frac{\sin(x_1)}{x_1} \cdot \frac{\sin(x_2)}{x_2} \tag{4.14}$$

where the external inputs $(x_1; x_2)$ are in the range [-10, 10] was used by Jang [32]. 250 epochs containing 121 of training data patterns are generated.

To model this function using the ANFIS required 4 bell membership functions per input. The membership functions per input are chosen such that they cover the entire data range and maintain $\varepsilon$ completeness. This would result in 16 rule nodes and hence 16 sub-filters. The RMSE was recorded as 0.008344 at the end of 250 epochs of training.

The M-ANFIS required the same number of parameters with only the rule definition of the ANFIS changed according to that of M-ANFIS. The root mean

square error (RMSE) of the M-ANFIS converged faster than the ANFIS. It was recorded as 0.00457 at the end of 250 epochs of training.

The ANFIS-2 required 3 bell membership functions for each input in $FIS_1$ and 3 gaussian membership functions for each input in $FIS_2$. Thus 9 rule nodes were extracted from each FIS and averaged to give 9 global rule nodes and so 9 sub-filters are chosen. It recorded an RMSE of 0.00597 at the end of 250 epochs of training.

The proposed structures converge faster than the original ANFIS. The performance of the three networks is given in *Fig. 4.3*. The structures are compared and tabulated in *Table 4.1*.

**4.4.2 Modeling a radial function**

This example was used by Maechler et al [41] to model PPL (Projection Pursuit Learning) networks.

$$f_3(x_1, x_2) = 24.234(r^2(0.75 - r^2)) \tag{4.15}$$

where $r^2 = \left(x_1 - \frac{1}{2}\right)^2 + \left(x_2 - \frac{1}{2}\right)^2$ (4.16)

The training data was generated from equation (4.15) in the range $x_1, x_2 \in [0,1]$. 100 epochs of data containing 36 training data patterns were used during training. The ANFIS required 4 bell membership functions for each input since it failed to train with 3 membership functions for each input. This lead to 16 rule nodes and hence 16 sub-filters were used for learning. The total number of adaptive parameters in ANFIS was equal to 72.

The M-ANFIS required 3 membership functions for each input. This resulted in 9 rule nodes and subsequently 9 sub-filters. The root mean square error (RMSE) of the ANFIS-2 converged faster than the ANFIS. The total number of adaptable parameters in this structure was equal to 45.

The ANFIS-2 required 3 membership functions for each input corresponding to its two fuzzy inference systems. Each fuzzy block produced 9 rules. Thus the global layer 2 contained 9 rules. Hence 9 sub-filters were chosen to approximate the above function. The alternate ANFIS-2 also produced performance comparable to the original ANFIS-2. The 3 bell membership functions in local layer 1 of the ANFIS-2 were replaced by 3 triangular membership functions. The performance of the three networks is given in *Fig. 4.4*. The structures are compared and tabulated in *Table 4.2*.

### 4.4.3 Modeling a complicated interaction function

This example was also used by Maechler et al [41] to model PPL (Projection Pursuit Learning) networks.

$$f_3(x_1, x_2) = 1.9(1.35 + e^{x_1}\sin(13(x_1 - 0.6)^2)e^{-x_2}\sin(7x_2)) \qquad (4.17)$$

The training data was generated from equation (4.17) in the range $x_1, x_2 \in [0,1]$. 100 epochs of data containing 36 training data patterns was used during training. The ANFIS required 3 bell membership functions for each input. This lead to 9 rule nodes and hence 9 sub-filters were used for learning. The total number of adaptive parameters in ANFIS was equal to 45.

The M-ANFIS required 4 membership functions for each input. This resulted in 16 rule nodes and subsequently 16 sub-filters. But the M-ANFIS trained faster than the ANFIS by several order of epochs. The total number of adaptable parameters in this structure was equal to 72.

The ANFIS-2 required 3 membership functions for each input corresponding to its two fuzzy inference systems. Each fuzzy block produced 9 rules. Thus the global layer 2 contained 9 rules. Hence 9 sub-filters were chosen to approximate the above function. The total number of adaptable parameters in this structure was equal to 45.

The alternate ANFIS-2 also produced performance comparable to the original ANFIS-2. The 3 bell membership functions in local layer 1 of the ANFIS-2 were replaced by 3 triangular membership functions. The performance of the three networks is given in *Fig. 4.5*. The structures are compared and tabulated in *Table 4.3*.

***Table 4.1*:** Modeling a 2D sinc function

| Index | Architecture | RMSE | No. of adaptive parameters |
|---|---|---|---|
| 1 | ANFIS | 0.008344 | (8*3+16*3) **72** |
| 2 | M-ANFIS | 0.004571 | (8*3+16*3) **72** |
| 3 | ANFIS-2 | 0.005967 | (6*3+6*2+9*3) **57** |

***Table 4.2:*** Modeling a radial function

| Index | Architecture | RMSE | No. of adaptive parameters |
|---|---|---|---|
| 1 | ANFIS | 0.000149 | (8*3+16*3) **72** |
| 2 | M-ANFIS | 0.000189 | (6*3+9*3) **45** |
| 3 | ANFIS-2 | 0.000235 | (6*3+6*2+9*3) **57** |
| 4 | Alternate ANFIS-2 | 0.000451 | (6*3+6*2+9*3) **57** |

***Table 4.3:*** Modeling a complicated interaction function

| Index | Architecture | RMSE | No. of adaptive parameters |
|---|---|---|---|
| 1 | ANFIS | 0.002116 | (6*3+9*3) **45** |
| 2 | M-ANFIS | 0.000078 | (8*3+16*3) **72** |
| 3 | ANFIS-2 | 0.00325 | (6*3+6*2+9*3) **57** |
| 4 | Alternate ANFIS-2 | 0.00241 | (6*3+6*2+9*3) **57** |

**Fig. 4.3: Modeling a 2D sinc function**

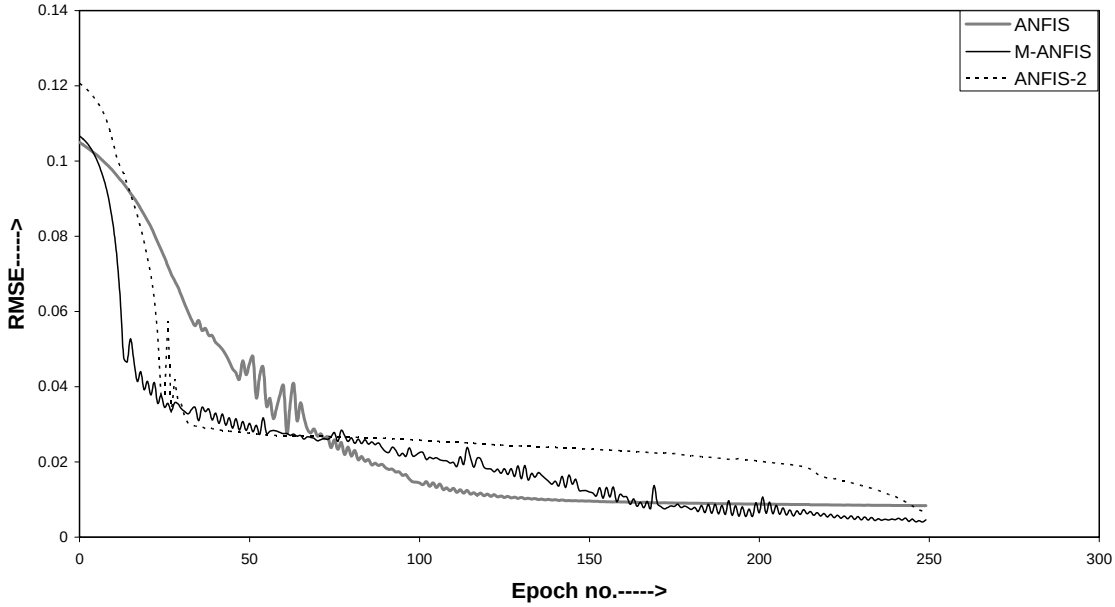


**Fig. 4.4: Modeling a radial function**

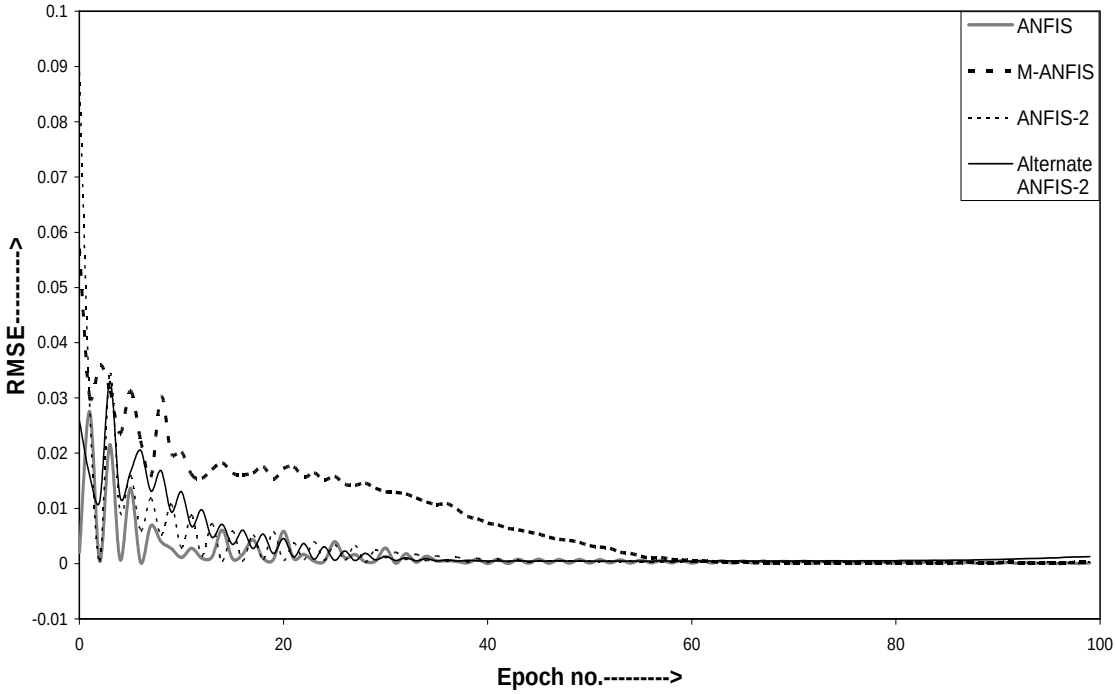
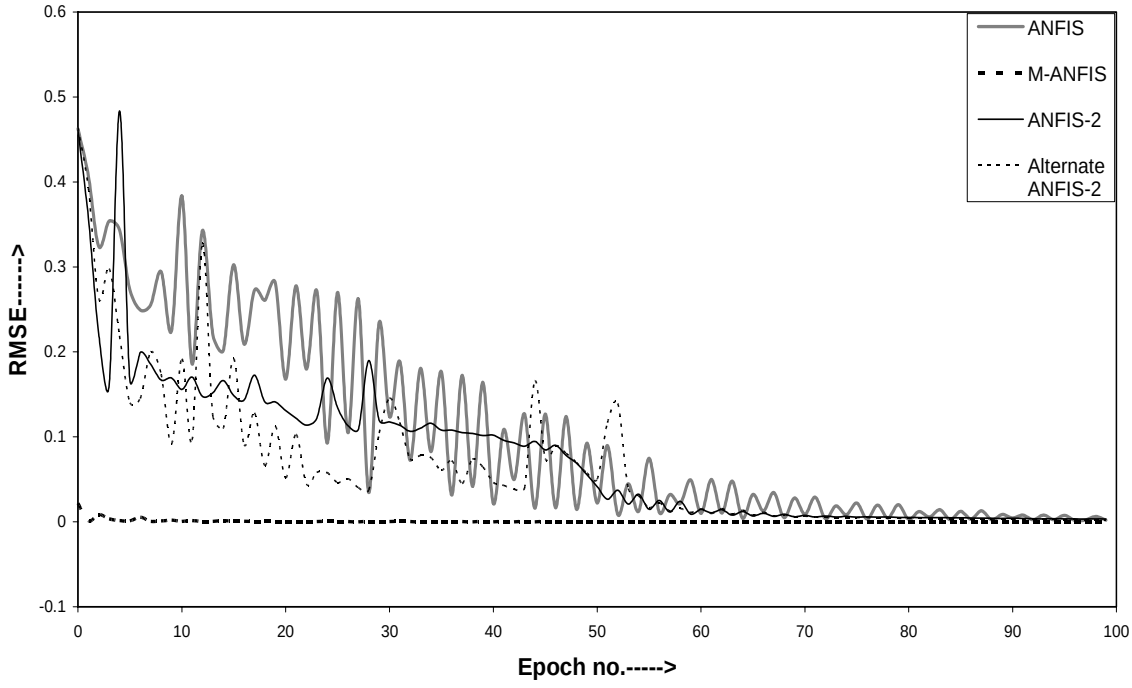



**Fig. 4.5: Modeling a complicated interaction function**



# Chapter 5

## Conclusion

This thesis was presented in two parts based on the two adaptive schemes that were discussed, namely, adaptive channel equalization and system identification. Thus the contribution of this research work is also discussed separately with respect to the two schemes.

Firstly, the need for adaptive equalization in digital communication channels was explained in the introduction chapter. It was suggested that the symbol by symbol classification or detection was the simplest way to realize the adaptive equalizers. It was noted that equalizers with nonlinear structure realize the optimal performance hence the neural networks based equalizers were strongly considered. The focus was mainly towards the design of radial basis function (RBF) equalizers since they are the ideal symbol by symbol detectors that could be realized. However, practical implementation of the RBF equalizer was hindered by issues relating to its structural complexity which grew along with the channel order or equalizer order or both. Hence various schemes for reducing the size of the RBF network were reviewed. This motivated the development of a novel center reduction technique for the RBF equalizer. Interesting insights were given regarding the role of the channel states with respect to their displacement from the decision boundary. A case study of the eigenvalue distribution corresponding to the channel states revealed that the channel states close to the decision boundary were characterized with the smallest maximum eigenvalue $(\lambda_{\max})_i$ while those away from the decision boundary were characterized with a larger maximum eigenvalue. Thus the maximum eigenvalue served as a criterion for selecting the centers that are absolutely necessary for decision making in the RBF equalizer.

A comparative study between the proposed center pruning technique and other center pruning techniques revealed that the developed technique has an edge over other schemes since it could be applied to channel characterized with overlapping channel states. Thus a reduced RBF equalizer with decision feedback could be designed.

The second contribution of this research work is to investigate the ANFIS architecture in regards to the problem of system identification. The interpretation of the ANFIS considered in commonplace was explained by terming it as a *cascade*

*interpretation*. A new kind of interpretation of the ANFIS architecture termed as the *parallel interpretation* was presented. The benefits of visualizing the ANFIS as a fuzzy weighted parallely implemented sub-filter network is that the fuzzy inference system can be considered to just scale the sub-filter outputs. Hence four different structures were developed motivated by such an interpretation which can be listed as follows

1. RBF weighted sub-filter approach (RBF-SF)
2. Modified basis function weighted sub-filter approach (MBF-SF)
3. Logistic sigmoid function weighted sub-filter approach (Sig1-SF)
4. Hyperbolic tangent function weighted sub-filter approach (Sig2-SF)

It was also explained that the weighting functions were bounded by the same range [0, 1], similar to the fuzzy rule nodes. Also it was hinted that a node reducing technique could be applied to realize a parsimonious structure.

Finally this research work dealt with the limitations of the ANFIS as explained in chapter 4. The logical t-norm function was used in defining the rule nodes. Also it was shown that a triangular membership function can be chosen to represent the linguistic terms associated with the external inputs. This resulted in novel ANFIS architectures since no attempt was made, so far, to use membership functions other than bell function or gaussian function. Three such network based adaptive fuzzy inference systems were developed which could be listed as follows

1. M-ANFIS or the ANFIS with modified rule nodes
2. ANFIS-2 or the ANFIS with two fuzzy inference systems
3. Alternate ANFIS-2 or the ANFIS with two fuzzy inference systems with bell membership function replaced by triangular membership function.

These architectures were applied to approximate functions used in the design of ANFIS and PPL (Projection Pursuit Learning) networks. It was shown that the ANFIS-2 structure not only realizes the performance of ANFIS but also attempts to reduce the number of rule nodes and subsequently the number of adaptive parameters in the network. It can be observed from the simulations that the developed structures outperform the ANFIS in terms of the required time for learning, minimum RMSE and have reduced number of adaptive parameters.

## Limitations and Future scope for research

As noted above the developed adaptive systems have been applied to only two types of adaptive schemes. The center reduction technique developed for the RBF equalizer may be applied to time varying channels. These ANFIS like structures have not been generalized. Especially, the parallely weighted sub-filter network has been applied to limited functions in the control system. A qualitative method of reducing the ANFIS and its contemporary structures may be developed.

The eigenvalue analysis may be applied to RBF networks used for system identification to realize parsimonious structures. The rule nodes of the ANFIS using gaussian membership function and product operator for defining rule nodes resembles RBF kernels hence the eigenvalue analysis may be applied to reduce the rule nodes of the ANFIS.

Further techniques like the Quick-Prop method or Resilient Backpropogation may be applied to adapt the parameters of the weighting functions or the fuzzy inference systems for improved performance.

# References


[1] Simon Haykin, "Neural Networks: A comprehensive foundation", Pearson Education, 2 Ed., 2004.

[2] K. Y. Lee, "Fuzzy adaptive decision feedback equalizer", Electronic Letters, Vol. 30, No. 10, pp. 749-751, May 1994.

[3] R. Langari and l. Wang, "A modified RBF network with application to system identification", 4th IEEE Proceedings on Control Applications, pp. 649-654, Sept 1995.

[4] C. F. Wong, T. L. Fine, "Adaptive blind equalization using artificial neural networks", IEEE International Conference on Neural Networks, Vol. 4, pp. 1974-1979, June 1996.

[5] T. Poggio and F. Girosi, "Networks for approximation and learning", Proceedings of IEEE, Vol. 78, No. 9, pp. 1481-1497, Sept. 1990.

[6] S. Siu, G. J. Gibson and C. F. N. Cowan, "Multilayer perceptron structures applied to adaptive equalisers for data communications", International Conference on Acoustics, Speech and Signal Processing, vol. 2, pp. 1183-1186, May 1989.

[7] S. Chen, B. Mulgrew, E. S. Chng, G. J. Gibson, "Space translation properties and the Minimum-BER linear-combiner DFE", IEE Proceedings on Communications, vol. 145, no. 5, pp. 316-322, Oct 1998.

[8] S. Siu, G. J. Gibson and C. F. N. Cowan, "Decision feedback equalisation using neural network structures and performance comparison with standard architecture", IEE Proceedings part I, vol. 137, no. 4, pp. 221-225, Aug 1990.

[9] G. Kechriotis, E. Zervas and E. S. Manolakos, "Using recurrent neural networks for adaptive communication channel equalization", IEEE Trans. On Neural Networks, vol. 5, no.2, March 1994.

[10] G. Kechriotis and E. S. Manolakos, "Training fully recurrent neural networks with complex weights", IEEE Trans. On Circuits and Systems, Vol. 41, No. 3, pp. 235-239, 1994.

[11] Li-Xin Wang and J. M. Mendel, "Fuzzy adaptive filters, with application to nonlinear channel equalization, IEEE Trans. on Fuzzy Systems, Vol. 1, No. 3, pp. 161-170, Aug. 1993.

[12] S. Chen, B. Mulgrew and P. M. Grant, “A clustering technique for digital communications channel equalization using radial basis function networks”, IEEE Trans. On Neural Networks, vol. 4, no. 4, July 1993.

[13] B. Mulgrew , “Applying radial basis functions”, IEEE Signal Processing Magazine, vol. 13, Issue 2, pp. 50-65, March 1996.

[14] J. Cid-Suerio and A. R. Fiqueiras-Vidal, “Recurrent radial basis function networks for optimal blind equalization”, III Proc. of IEEE on Neural Networks for Signal Processing, pp. 562-571, Sept 1993.

[15] S. Bouchired, M. Ibnkahla, D. Roviras and F. Castanie, “Equalization of satellite mobile communication channels using combined self-organizing maps and RBF networks”, Proc. of IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP, Vol. 6, pp. 3377-3379, May 1998.

[16] C. Chinrungrueng and C. H. Sequin, “Optimal adaptive K-means algorithm with dynamic adjustment of learning rate”, IEEE Trans. on Neural Networks, Vol. 6, No. 1, Jan 1995.

[17] S. Chen, C. F. N. Cowan and P. M. Grant, “Orthogonal least square algorithm for radial basis function networks”, IEEE Trans. on Neural Networks, Vol. II, No. 2, pp. 302-309, March 1991.

[18] J. Lee, C. Beach, N. Tepedelenlioglu, “A practical radial basis function equalizer”, IEEE Trans. Of Neural Networks, Vol. 10, No. 2, pp. 450-455, March 1999.

[19] S. Chen, B. Mulgrew and S. Mclaughlin, “Adaptive Bayesian equaliser with decision feedback”, IEEE Trans. Signal Processing, Vol. 41, No. 9, Sept 1993.

[20] S. Chen, B. Mulgrew and S. Mclaughlin, “Adaptive Bayesian decision feedback equaliser based on radial basis function network”, in Proc. ICC92, pp. 343.3.1-343.3.5.

[21] J. Gomes and V. Barroso, “Using a RBF network for blind equalization: Design and performance evaluation”, in Proc. of IEEE International Conference on Acoustics, Speech and Speech Processing, ICASSP’ 97, pp. 3285-3288, 1997.

[22] N. Sundararajan, P. Saratchandran, “Performance evaluation of a sequential minimal radial basis function (RBF) neural network learning algorithm”, IEEE Trans on Neural Networks, Vol. 9, No. 2, pp. 308-318, March 1998.

[23] Q. Gan, R. Subramanian, N. Sundararajan and P. Saratchandran, “Design for centers of RBF neural networks for fast time-varying channel equalization**”,** Electronics Letters, Vol. 32, No. 25, pp. 2333-2334, Dec 1996.

[24] Simon Haykin, “Adaptive filter theory”, Ed. 4, Pearson Education, 2002.

[25] F. Beaufays, “Transform-domain adaptive filters: An analytical approach”, IEEE Trans. On Signal Processing, Vol. 43, No. 2, Feb 1995.

[26] D. F. Marshall and W. K. Jenkins, “The use of orthogonal transforms for improving performances of adaptive filters”, IEEE Trans. On Circuits and Systems, Vol. 36, No. 4, pp. 474-484, April 1989.

[27] William H. Press, “Numerical Recipes in C”, Cambridge University Press, Ed.2.

[28] B. Widrow and S. D. Streams, “Adaptive Signal Processing”, Pearson Education, 2002.

[29] Chin-Teng Lin and Chia-Feng Juang, “An adaptive neural fuzzy filter and its applications”, IEEE Trans. on Syst., Man, Cybern.-Part B: Cybernetics, Vol. 27, No. 4, pp. 635-356, Aug 1997.

[30] J. –S. R. Jang, C. –T, Sun and E. Mizutani, “Neuro-Fuzzy and Soft Computing”, Prentice-Hall, 2003.

[31] T. Takagi and M. Sugeno, “Fuzzy identification of systems and its applications to modeling and control” IEEE Trans. on Syst., Man, Cybern., Vol. 16, pp. 116-132, 1985.

[32] J. –S, R. Jang, “ANFIS: Adaptive-Network-based Fuzzy Inference System”, IEEE Trans. on Syst., Man, Cybern., Vol. 23, No. 3, pp. 635-355, May/June 1993.

[33] H. Husain, M. Khalid and R. Yusof, “Nonlinear function approximation using Radial Basis Function neural networks”, IEEE Student Conference on Research and Development Proceedings, Shah Alam, Malaysia, pp. 326-329, 2002.

[34] J. M. Mendel, “Uncertainty, fuzzy logic, and signal processing”, Signal Processing, Vol. 80, pp. 913-933, 2000.

[35] R. Babuška, “Neuro-fuzzy methods for modeling and identification”, In A. Abraham, L. C. Jain and J. Kacprzyk, editors, Recent advances in Intelligent Paradigms and Applications, Springer-Verlag, Heidelberg, pp. 161-186, 2002.

[36] C. Borgelt and R. Kruse, " Speeding up fuzzy clustering with neural network techniques", Proc. 12th IEEE Conference on Fuzzy Systems, Vol. 2, pp. 852-856, May 2003.

[37] J. H. Yang, C. C. Yang and J. Y. Lai, "Adaptive sliding mode control for vehicle braking systems with TSK Model control method", Intelligent Systems and Control (ISC), 497-073, Cambridge, 2005.

[38] C. -C. Lee, "Fuzzy logic in control systems: Fuzzic logic controller-Part I", IEEE Trans. Syst., Man, Cybern., Vol. 20, pp. 419-435, 1990.

[39] J. -S. R. Jang and C. -T. Sun, "Functional equivalence between RBF networks and fuzzy inference systems", IEEE Trans. on Neural Networks, Vol. 4, pp. 156-159, Jan 1995.

[40] T. Takagi and I. Hayashi, "NN-driven fuzzy reasoning", Int. J. Approximate Reasoning, Vol. 5, No. 3, pp. 191-212, 1991.

[41] M. Maechler, D. Martin, D. Schimert, M. Csoppenszky and J. N. Hwang, "Projection pursuit learning networks for regression", Proc. of 2nd IEEE conference on Tools for Artificial Intelligence, pp. 350-358, Nov. 1990.

[42] J. -S. R. Jang and C. -T. Sun, "Neuro-fuzzy modeling and control", Proc. of IEEE, Vol. 83, No. 3, pp. 378-406, March 1995.

[43] X. Zhang, C. -C. Hang, S. Tan and P. -Z. Wang, "The min-max function differentiation and training of fuzzy neural networks", IEEE Trans. on Neural Networks, Vol. 7, No. 5, Sept. 1996.