NOVEL ARTIFICIAL NEURAL NETWORK ARCHITECTURES AND ALGORITHMS FOR NON-LINEAR DYNAMICAL SYSTEM MODELING AND DIGITAL COMMUNICATIONS APPLICATIONS

A DISSERTATION

SUBMITTED TO THE SIGNAL PROCESSING DIVISION,

DEPARTMENT OF ELECTRONIC AND ELECTRICAL ENGINEERING

AND THE COMMITTEE FOR POSTGRADUATE STUDIES

OF THE UNIVERSITY OF STRATHCLYDE

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

By

Amir Hussain

September 1996

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of Doctor of Philosophy.

(Principal Adviser)

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of Doctor of Philosophy.

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of Doctor of Philosophy.

Approved for the University Committee on Graduate Studies:

Dean of Graduate Studies & Research

Declaration

I declare that the Thesis embodies my own research work and that it is composed by myself. Where appropriate, I have made acknowledgement to the work of others.

Acknowledgements

I am deeply indebted to Dr. John James Soraghan and Professor Tariq Salim Durrani, my supervisors, for providing excellent guidance and technical support throughout the course of the PhD research.

I also gratefully acknowledge the financial support provided for the research work undertaken in this thesis by the University of Strathclyde Post-graduate Research Scholarship, the Department of Electronic and Electrical Engineering Maintenance Award and the David Livingston Bursary.

Finally, I would like to especially thank Prof. Tariq Durrani for providing the financial support I needed to attend various International Workshops and Conferences.

Read in the name of thy Lord ... who taught man that which he knew not.
The Holy Quran

Abstract

This thesis presents novel Artificial Neural Network (ANN) based architectures and algorithms for solving two important signal processing applications.

Firstly, in the context of non-linear dynamical system modeling applications, a new two-layer linear-in-the-parameters, feedforward ANN structure is developed, which is termed the Feedforward Functionally Expanded Neural Network (FFENN), in order to efficiently model chaotic and equation-error type non-linear dynamical processes. A general design strategy is presented for specifying the type and number of non-linear basis functions within the FFENN's single hidden layer, for an arbitrary number of network inputs. The FFENN structure employing the proposed basis functions in its hidden layer is essentially a hybrid neural network incorporating to a variable extent, the combined modeling capabilities of the conventional Multi-Layered Perceptron (MLP), Radial Basis Function (RBF) and Volterra Neural Networks (VNN). Its output mean squared error surface is shown to be uni-modal allowing high speed single-run least squares based learning. A new pruning strategy based on an iterative pruning-retraining scheme coupled with statistical model validation tests, is also devised in order to optimise the size of FFENN structures for non-linear dynamical system modeling applications. Numerous case studies are performed using simulated chaotic, equation-error and a variety of real-world noisy, non-stationary time series processes which show that the new FFENN based predictor models consistently outperform other recently reported, feedforward and recurrent ANN structures, both in terms of non-linear prediction ability and relative computational complexity requirements. To enable efficient modeling of a more general class of non-linear dynamical systems, a new computationally efficient Recurrent Neural Network (RNN) structure is also developed, which is termed the Recurrent Functionally Expanded Neural Network (RFENN). Its learning

algorithm is derived and a pruning strategy proposed. The development of the RFENN is shown to result in a new class of computationally efficient RNNs incorporating all linear-in-the-parameters feedforward ANNs (such as the RBF and VNN) adapted to employ local output feedback. Various case studies are performed using simulated chaotic, output-error and real-world noisy times series processes, which show that the RFENN based predictor models can significantly outperform the corresponding feedforward FENN and other ANN predictors in the modeling of certain types of non-linear dynamical processes. The FFENN and RFENN are also successfully applied to the task of real-time adaptive non-linear prediction of real non-stationary signals. A new hybrid RFENN-FIR adaptive structure comprising the non-linear RFENN subsection feeding into a linear Finite Impulse Response (FIR) subsection is also developed and shown to outperform the stand-alone FFENN and RFENN based adaptive predictors.

Secondly, in the context of digital communications applications, two new adaptive non-linear Decision Feedback Equalizer (DFE) structures are developed, which are termed: the Decision Feedback Functional-Link Equalizer (DFFLE) with Expanded Feedback Terms (DFFLE-EFT); and the DFFLE with Unexpanded Feedback Terms (DFFLE-UFT). The DFFLE-UFT employs the recently reported non-linear-in-the-parameters Feedforward Functional-Link Equalizer (FFLE) as its feedforward filter and a linear feedback filter. In contrast, the novel DFFLE-EFT structure non-linearly combines both the equalizer input and decision feedback samples. Learning algorithms and general design strategies are presented for both the structures. Pruning techniques for optimizing the sizes of the FFLE and DFFLE structures are also proposed. In the first digital communications application considered in the thesis, the new structures are employed for the equalization of linear and non-linear communication channels in the presence of ISI and additive (uncorrelated and correlated) noise. In the second digital communications application considered in the thesis, the FFLE and DFFLE structures are proposed as a novel solution to the problem of overcoming co-channel interference in digital communications systems. Various simulation case studies are performed for both applications, which show that the new DFFLE-EFT is a viable alternative to the optimal symbol Bayesian Transversal Equalizer (TE) and all other ANN based TE structures (which have been reported to date for approximating the Bayesian TE).

Notation and Acronyms

ADPCM Adaptive Differential Pulse Code Modulation

ANN Artificial Neural Network

APP Adaptive Polynomial Perceptron

BER Bit-Error Rate

BM Boltzmann Machine

BP Back Propagation

BPTT Back Propagation Through Time

CDMA Code Division Multiple Access

DDM Decision Directed Mode

DFE Decision Feedback Equalizer

DFFLE Decision Feedback Functional-Link Equalizer

DFFLE-EFT DFFLE with Expanded Feedback Terms

DFFLE-UFT DFFLE with Unexpanded Feedback Terms

DR Delta Rule

EKF Extended Kalman Filter

FANN Feedforward Artificial Neural Networks

FENN Functionally Expanded Neural Network

FFENN Feedforward Functionally Expanded Neural Network

FFLE Feedforward Functional-Link Equalizer

FIR Finite Impulse Response

FLNN Functional-Link Neural Network

FRLS Fast Recursive Least Squares

GDR Generalized Delta Rule

HM Hopfield Model

HOS Higher Order Statistics

IIR Infinite Impulse Response

ISI Inter-Symbol Interference

LMS Least Mean Squares

LRGFN Locally Recurrent Globally Feedforward Networks

MA Moving Average

MAP Maximum Aposteriori Probability

MIMO Multiple Input Multiple Output

MLP Multi-Layered Perceptron

MLSE Maximum Likelihood Sequence Estimator

MLVA Maximum Likelihood Viterbi Algorithm

MNN Memory Neural Networks

MSE Mean Squared Error

NAR Non-linear Auto-Regressive

NARMA Non-linear Auto-Regressive Moving Average

NARMAX NARMA with eXogenous inputs

NARX Non-linear Auto-Regressive with eXogenous inputs

OLS Orthogonal Least Squares

PAM Pulse Amplitude Modulation

QAM Quadrature Amplitude Modulation

RANN Recurrent Artificial Neural Network

RBF Radial Basis Function

RFENN Recurrent Functionally Expanded Neural Network

RFENN-FIR RFENN - Finite Impulse Response

RLS Recursive Least Squares

RNN Recurrent Neural Network

RPE Recursive Prediction Error

RTRL Real Time Recurrent Learning

RTRN Real Time Recurrent Network

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Introduction

In the past decade and a half, there has been a strong resurgence in the field of Artificial Neural Networks (ANNs) involving researchers from many diverse disciplines. This renewed interest is primarily because of new network topologies, improved learning algorithms, improved theoretical foundations, greatly enhanced computer systems for simulation, and emerging analogue VLSI implementation techniques [1] [19] [30] [24] [45] [56, 57]. ANNs are useful in that they learn knowledge without the need of a priori specification of a representation scheme. This is most useful for problems in which either the objective cannot be expressed precisely in terms of measurable parameters, or the set of parameters is poorly defined.

ANNs have been found to be useful in many applications ranging from non-linear controllers, CAM, optimization, constraint satisfaction, speech processing, robotics, automatic-target recognition, linear and non-linear adaptive filtering, character recognition, dimensionality reductions and pattern classification [56]. However, many outstanding problems need to be solved before the ANNs can be applied widely in practice. Primarily, the general use of ANNs is hampered by

their inabilities to generalise to new problems and to scale to larger problems; and in particular, their high learning computational requirements [28]. Hence, the need for development of new computationally efficient neural network structures that can be used for solving complex real world applications. In this thesis, we develop new ANN based architectures and algorithms for solving two important signal processing applications:

- Identification of non-linear dynamical systems: Modeling of both simulated and real world non-linear dynamical processes has been investigated using the new ANN models.
- Equalization of digital communications systems: The use of new ANN based structures in equalization of linear and non-linear dispersive channels in the presence of noise and co-channel interference is investigated.

1.2 Motivation of the Thesis

The motivation for investigating **Artificial Neural Network Architectures and Algorithms for Non-linear Dynamical System Modeling and Digital Communications Applications** is two-fold:

The first is to develop new Feedforward and Recurrent Artificial Neural Network structures which can provide a viable alternative to the conventional computationally expensive, multi-layered neural networks that have been employed to date for the modeling of non-linear dynamical systems. The conventional ANNs employed for non-linear dynamical system modeling applications are generally highly non-linear in the parameters, complex models that require computationally expensive, non-linear learning algorithms in order to approximate the dynamic system's input-output behaviour [34] [181] [182] [191] [36]. Hence, development of new parsimonious, computationally efficient ANN based models that could additionally, also reveal useful insights into the physical mechanism of the non-linear

system, would be highly desirable.

Secondly, to exploit the use of new ANN structures as non-linear adaptive filters in the telecommunications industry. With the present great demand for data communication services, bit rates and symbol rates are being pushed towards their theoretical limits. Consequently, communication channel impairments that previously went unnoticed can now be particularly problematic [165]. For example, when transmitting data over the PSTN (Public Switched Telephone Network) at moderate bit rates, the channel can be considered to be linear; however, at high bit rates, the non-linearities introduced by the network elements such as the coupling transformers, codecs and amplifiers cannot be ignored, and must be compensated for by the use of appropriate non-linear signal processing techniques [184] [165]. Conventional neural network based adaptive non-linear equalizers have excessive computational requirements and require relatively large training periods in order to realise the optimal equalization performance [172] [183] [184] [166]. Hence, new faster and computationally efficient neural network equalizers need to be developed which can better compensate for not only the linear and non-linear communication channel distortion, but additionally also be able to suppress other significant interference factors such as co-channel interference effects, encountered in many digital communications systems (for example, digital cellular radio) [190].

1.3 Contribution of Thesis

The five main contributions of this thesis are now identified:

1. A new linear in the parameters, two-layer feedforward ANN architecture has been developed which is termed the Feedforward Functionally Expanded Neural Network (FFENN), for efficient modeling of chaotic and equation-error type non-linear dynamical systems [175]. The new structure alleviates the non-linear learning difficulties associated with conventional

multi-layered ANNs by employing least squares based learning algorithms in its output layer. New non-linear basis functions are proposed for the single hidden-layer of the FFENN that emulate other universal approximators namely, the squashing type sigmoidal activation functions employed by the conventional feedforward Multi-Layered Perceptron (MLP), the Gaussian and multi-quadratic type activation functions employed by the conventional feedforward RBF networks, and polynomial-subset activation functions employed by the conventional Volterra Neural Network (VNN). The new FFENN structure employing the proposed basis functions in its hidden layer is essentially a *hybrid* neural network incorporating to a variable extent, the rich modeling capabilities of the conventional MLP, RBF and VNN structures.

A general design strategy for specifying the type and number of basis functions within the FFENN's hidden layer for an arbitrary number of network inputs is also presented. A new pruning strategy based on an iterative pruning-retraining scheme coupled with correlation and chi-squared statistic based model validity tests [109] is also devised in order to optimise the size of FFENN structures for non-linear dynamical system modeling applications. Numerous case studies are carried out using simulated chaotic, equation-error type and a variety of real-world noisy, non-linear time series processes to compare the modeling and prediction performance of the FFENN with various other recently reported feedforward and recurrent neural network based predictor models. The new FFENN based predictor models are shown to consistently outperform the other techniques both in terms of non-linear prediction ability and relative computational complexity requirements. The respective contributions of the various proposed non-linear basis functions responsible for the superior FFENN performance are also illustrated in the various case studies.

2. A new Recurrent ANN architecture termed the Recurrent Functionally Expanded Neural Network (RFENN) is developed to enable efficient modeling of a more general class of non-linear dynamical systems [176] [175]. The RFENN is based on the linear-in-the-parameters FFENN employing local output feedback. Its associated learning algorithm is derived and key structural and learning computational complexity comparisons are made between the new RFENN architecture and the conventional Recurrent Neural Network (RNN) structure, which show a significantly simpler computational requirement of the RFENN. The reduction in learning computational complexity requirements of the RFENN is in fact, due to the employment of non-linear basis functions at its input single hidden layer only, whereas, in conventional multi-layered RNN structures non-linear basis functions are employed at both the hidden and output layers. The development of the RFENN is thus shown to result in a new class of computationally efficient RNNs incorporating all linear-in-the-parameters feedforward neural networks (such as the RBF and VNN) adapted to employ local output feedback. Various case studies using simulated chaotic, output-error and real-world noisy, non-linear time series processes have been used to show that the RFENN can outperform the FFENN and other recently reported neural network based predictor models in the modeling of certain types of non-linear dynamical processes [52].

3. The application of the new RFENN and FFENN structures to real-time (on-line) adaptive non-linear prediction of non-stationary time series processes has also been proposed and investigated [175]. Both are shown to significantly outperform the conventional linear filtering approaches in the adaptive modeling of real world, highly non-stationary signals including real NH_3 laser and actual speech data. A new hybrid RFENN-FIR adaptive structure comprising the non-linear RFENN subsection feeding into a linear Finite Impulse Response (FIR) subsection has also been developed [175] and shown to outperform both the stand-alone FFENN and RFENN

based predictor models.

4. Two novel adaptive non-linear Decision Feedback Equalizer (DFE) structures have been developed [177] [178] [179] termed, the Decision Feedback Functional-Link Equalizer (DFFLE) with Unexpanded Feedback Terms (DFFLE-UFT), and DFFLE with Expanded Feedback Terms (DFFLE-EFT). The DFFLE-UFT employs the conventional, recently developed non-linear-in-the-parameters Feedforward Functional-Link Equalizer (FFLE) [163] [164] as its feedforward filter and a linear feedback filter. In contrast, the novel DFFLE-EFT structure non-linearly combines both the equalizer input and decision feedback samples.

   Learning algorithms are presented for both the structures along with their design strategies. Key structural and computational complexity comparisons are made between the new DFFLEs and the conventional FFLE, which show significantly simpler computational requirements of the DFFLEs. The Extended Kalman Filter (EKF) algorithm [171] has also been applied to both the FFLE and the DFFLE structures in order to enhance their speed of error convergence characteristics. A new general design strategy has also been presented for the conventional FFLE and shown to give new insights into its computational requirements with increasing input dimensions for $2-ary$ PAM based systems. Pruning techniques for optimizing the sizes of the FFLE and DFFLE structures are also proposed. Various simulation case studies have been carried out on the application of DFFLE structures to equalization of linear and non-linear communications channels in the presence of Inter-Symbol Interference (ISI), and both, additive white and coloured noises. The new DFFLE-EFT employing a novel functional-link model non-linearly combining both the equalizer input and decision feedback samples, is shown to consistently outperform the optimal symbol Bayesian Transversal Equalizer (TE) and various other recently reported ANN based non-linear TE and DFEs, both in terms of the Bit-Error Rate

performance characteristics and relative computational requirements; yielding a much closer approximation to the optimal sequence Maximum Likelihood Viterbi Algorithm (MLVA) based equalizer. An error-propagation analysis has also been performed for the DFFLEs using simulations, which show a very small resulting performance degradation.

5. A novel solution to the problem of co-channel interference supression in digital communications systems has been proposed based on the new DFFLE and FFLE structures [179]. A realistic co-channel system is used as an example to show that, relative to other recently reported non-linear ANN based equalizers (namely the FFLE and RBF equalizers), the DFFLE-EFT structure is significantly more effective in dealing with the co-channel interference effects and also has the minimal relative computational complexity requirements. Simulations are also used to show that error propagation in the DFFLE results only in a very small performance degradation.

1.4 Thesis Overview

This thesis is organized into six main chapters as follows:

Chapter 1 gives the motivation and contribution of this thesis.

Chapter 2 reviews the main ANN paradigms namely the feedforward and recurrent neural network structures. It also describes a general framework incorporating models based on the feedforward and recurrent ANNs, for efficient modeling of non-linear dynamical systems. Feedforward ANNs are described in detail in Appendix A.

Chapter 3 discusses two important problems encountered in digital communications systems; firstly, the equalization of finite linear and non-linear communication channels in the presence of Inter-Symbol Interference (ISI) and additive noise; and secondly, the suppression of co-channel interference. It investigates the

structures, learning algorithms and the relative advantages and disadvantages of various linear and ANN based adaptive non-linear equalizer structures that have been proposed to date to solving these problems.

Chapter 4 describes a new feedforward ANN architecture for efficient modeling of equation-error type non-linear dynamical systems, namely the Feedforward Functionally Expanded Neural Network (FFENN). Its learning algorithm is derived and a formal design strategy presented, together with a discussion on the choice of the proposed hidden layer's functional expansion model. A pruning strategy has also been proposed for optimising the size of the FFENN for non-linear dynamical system modeling applications. Several case studies using a variety of simulated equation-error, chaotic, and real-world non-linear time series processes are carried out in order to compare the modeling performance of the new FFENN structure with other recently reported ANN based predictor models.

Chapter 5 describes a new recurrent ANN structure termed the Recurrent Functionally Expanded Neural Network (RFENN) for efficient modeling of the more general output-error type non-linear dynamical systems. Its learning algorithm is derived and a pruning strategy presented. Several case studies are carried out using simulated output-error, chaotic and real-world noisy time series processes, in order to compare the modeling capability of the RFENN with the FFENN and other recently reported feedforward and recurrent ANN predictor models. Chapter 5 finally, also presents an investigation into the use of the FFENN and RFENN structures for adaptive (on-line) non-linear prediction of real world, highly non-stationary time series processes. Real world NH_3 laser time series and an actual speech signal are used as two studies. A new hybrid adaptive RFENN-FIR predictor is also devised and its performance compared with the stand-alone FFENN, RFENN and the conventional linear predictors.

Chapter 6 describes two new Decision Feedback Equalizer (DFE) structures

namely, the DFFLE-EFT and the DFFLE-UFT structures. Their learning algorithms are presented along with formal design strategies assuming $2 - ary$ PAM signalling schemes. Key structural and computational complexity comparisons are made between the new structures and the conventional recently reported Feedforward Functional-Link Equalizer (FFLE). A formal design strategy is also presented for the FFLE. An Extended Kalman Filter (EKF) algorithm is also applied to the FFLE and DFFLEs for updating their output layer weights. Pruning strategies are also proposed for optimising the sizes of the FFLE and DFFLE structures. The new DFFLE based structures are applied to solving two important problems encountered in digital communications: namely, co-channel interference supression and the equalization of linear and non-linear digital communications channels in the presence of Inter-Symbol Interference (ISI) and both, additive uncorrelated and correlated Gaussian noise signals. For both applications, the performance of the new structures has been evaluated against other ANN based equalizers that have been proposed to-date to approximate the underlying optimal symbol Bayesian and optimal sequence Maximum Likelihood Viterbi Algorithm (MLVA) based equalizers. Error propagation effects in the DFFLEs have also been investigated for both applications.

Chapter 7 finally concludes this thesis with a critical overview of the various ANN issues, characteristics and limitations that have been tackled in this thesis and discusses relevant future work proposals.

Appendix A reviews various multi-layered and single-hidden layered feedforward ANNs. It also discusses the general issues and limitations that relate to feedforward ANNs.

Appendix B gives a listing of the Real Time Recurrent Learning (RTRL) algorithm employed for updating conventional Recurrent Neural Network (RNN) structures.

Appendix C gives a listing of the author's publications.

Chapter 2

Artificial Neural Networks (ANNs) For Non-Linear Dynamical System Modeling

2.1 Introduction

The human brain is the most complex computing device known to man and its powerful thinking, remembering and problem-solving capabilities have inspired scientists to attempt computer modelling of its operation. Artificial Neural Networks (ANNs) are human attempts to emulate the functionality of the human brain in order to solve difficult scientific problems.

The origin of ANN can be traced back to the early twentieth century when psychologists attempted to identify the neural basis of intelligence [61]. Today ANN have matured into an attractive alternative for solving problems involving learning, due to sustained efforts of many researchers in the last thirty years.

An ANN typically consists of many simple computational elements or neurons arranged in layers and operating in parallel [19]. The weights which define the strength of connection between the neurons (also called nodes) are adapted during operation to improve performance. The power and complexity of learning in

an ANN are highly dependent on the operations performed by its nodes, its topology, the learning algorithm used, and most importantly, the nature of the application. The type of functions computable by an ANN are strongly affected by the various properties of the nodal activation function (that is, whether it is linear or nonlinear, thresholded or smooth, deterministic or stochastic *etc.*). The interconnection pattern of the ANN (whether feedforward or feedback) dictates its dynamic behaviour on whether the activation can sustain, and whether or not its output evolves over time. To summarize, ANNs are therefore specified by the network architecture, node characteristics and learning algorithms.

While much of the recent research in ANNs has been directed towards the development of new architectures for pattern classification problems [21] [25] [24] [31], there has been some considerable work in developing and applying ANN structures to the identification, prediction and control of non-linear dynamical systems [33] [34] [27].

This chapter firstly reviews the two main categories of ANNs namely, static feedforward and dynamic recurrent ANNs in sections 2.2 and 2.3 respectively. Finally in section 2.4, models based on Feedforward and Recurrent ANNs for the identification of non-linear dynamical systems are investigated.

2.2 Feedforward ANNs

In general, the neurons in an ANN are arranged in one of three layers:

- input layer
- one or more intermediate (or *hidden*) layers
- output layer

A class of neural networks, where the inputs feed through the network layers to the output, is referred to as the Feedforward ANN (FANN). FANNs are basically static networks which implement non-linear transformations of the form

$$\mathbf{y} = G(\mathbf{x})$$

where, typically, the network inputs $\mathbf{x}$ belong to R^n and the network outputs $\mathbf{y}$ belong to R^m; where n and m are integers that represent the dimensions of $\mathbf{x}$ and $\mathbf{y}$ respectively. These static FANNs, which are characterized by node equations that are memoryless, that is, their output is a function only of the current input, not of past or future inputs or outputs; are useful in a variety of applications such as approximation of logic functions, pattern recognition and functional approximation [24]. It is important to realize that all FANNs can also be easily adapted to act as dynamic networks, where the node equations are described by differential or difference equations [24]. This issue will be addressed in section 2.4, in which the problem of non-linear dynamical system modelling using ANNs is investigated.

FANNs can be classified into two types namely, single-(hidden)-layered FANNs and multi-(hidden)-layered FANNs. Single-hidden layered FANNs have also been called two-layered FANNs in literature by counting the output layer as an additional layer [24].
A detailed review of various FANNs is given in Appendix A. The multi-layered FANNs that are discussed in Appendix A include the conventional and most popular Multi-Layered Perceptron (MLP) and its extensions. The single-layered FANNs that are discussed in Appendix A include the linear-in-the-parameters Radial Basis Function (RBF) and Volterra Neural Networks (VNN) and the conventional non-linear-in-the-parameters Functional Link Neural Network (FLNN). Finally, Appendix A discusses the general issues and limitations that relate to FANNs.

In the next section, the other paradigm of ANNs namely, the Recurrent ANNs are discussed.

2.3 Recurrent ANNs

Recurrent ANNs (RANNs) belong to the class of dynamic networks. The node equations are described by differential or difference equations. Dynamic RNNs can be classified into two types: RNNs with output feedback, and RNNs with state feedback.

2.3.1 RNNs: Networks with Output Feedback

This architecture which is illustrated in Figure 2.1, was introduced by Narendra *et al* [34] [35] who used it primarily for non-linear identification and control problems.

However, this structure is very general in that the MLP can be replaced by any of the FANNs discussed in Appendix A. In chapter 5, a new Recurrent Neural Network structure termed the Recurrent Functionally Expanded Neural Network (RFENN) is developed by incorporating local output feedback in a newly developed Feedforward Functionally Expanded Neural Network (FFENN) reported in chapter 4.

2.3.2 RNNs: Networks with State Feedback: The Hopfield Network, Boltzmann Machine, Real Time Recurrent Networks and Locally Recurrent Globally Feedforward Networks (LRGFN)

In this class of Recurrent Neural Networks (RNNs), state feedback is employed. These networks are typically single layered networks with feedback connections between nodes. In the most general case depicted in Figure 2.2, all nodes are fully interconnected, that is, every node is connected to every other network node including itself [118]. Each node contributes one component to the state vector. The output of the RNN can be viewed to be the output of any or all of the network nodes. Additionally, external inputs may be applied to any or all these

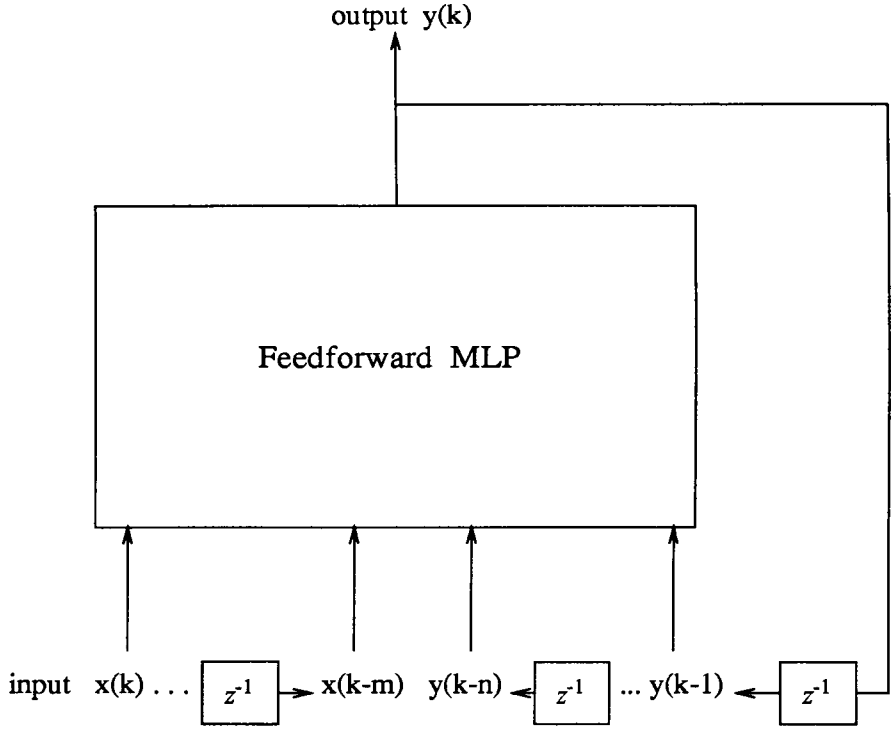

Figure 2.1: Recurrent Networks with output feedback

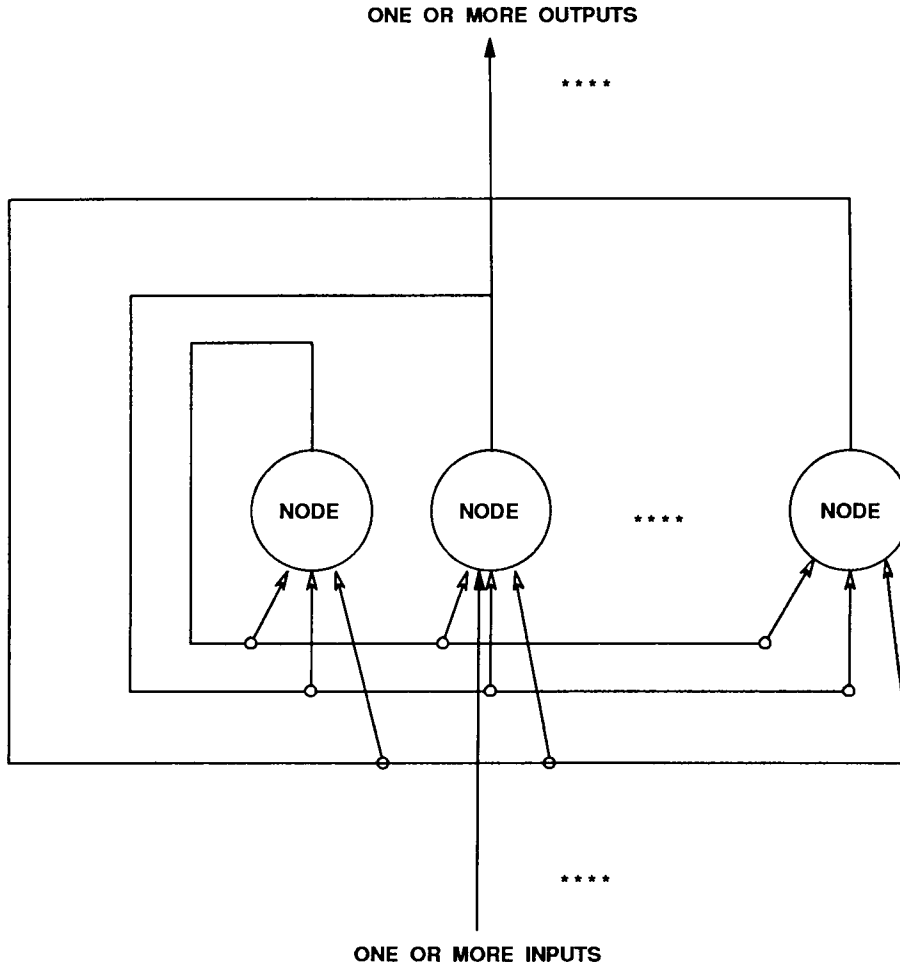

Figure 2.2: Recurrent Networks with state feedback

nodes as well. This class of RNNs is the most general and encompasses all other Recurrent ANNs. The FANN structures discussed in Appendix A can also be viewed as simplifications of this class of RNNs. Some of the important Recurrent ANN structures are now reviewed, namely the Hopfield Model Neural Network, the Boltzmann Machine, the Real Time Recurrent Networks and the recently reported class of Locally Recurrent Globally Feedforward Networks (LRGFN).

The Hopfield Network

The Hopfield Model (HM) network is probably the best known dynamic recurrent neural network model and comprises a single layer of fully connected and symmetrically weighted McCulloch-Pitts neurons [61]. Hopfield [76] defined two types of neurons:

- the two state or digital neurons which constitute the Digital HM.
- the graded response or analogue neurons which constitute the Analogue HM.

Hopfield has shown that an energy function exists for the network, and that the HM implements a gradient descent algorithm. The operation of the HM can be summarised as follows:
After the network is initialized , the unknown input pattern is presented and the network then iterates to convergence. The presentation of a corrupt input pattern results in the reproduction of the perfect pattern as the output -the network therefore acts as a *content-addressable* or an associative memory [76].

The HM in fact modifies another recurrent ANN termed the Brain-State in a Box (BSB) network that was developed earlier by Anderson [2] which is a positive feedback system with amplitude limitation, and comprises a set of highly interconnected neurons that self feed back. The problem with the HM network (and the BSB network) is that for optimisation problems involving multi-minima cost functions, it is liable to find the local minima instead of the global minimum solution [102]. As a content addressable memory, the HM is capable of storing

no more than $0.138N$ random patterns, where N is the number of neurons in the network. Various methods have been proposed for enhancing the storage capacity of the HM to approach the theoretical upper limit of $2N$ random patterns [56]. The main limitation of the HM however, is that, it lacks hidden neurons; which are known to learn internal representations of training patterns, thereby enhancing the performance of the neural network. The Hopfield Network has however, been found to be useful in solving a number of applications, in particular the bearing estimation problem reported by Jha [102] and Hussain [174].

The Boltzmann Machine is a generalisation of the HM, which uses both hidden and visible neurons that are in the form of stochastic, binary state units, and is discussed next.

The Boltzmann Machine

Ackney, Hinton and Sejnowski [4] developed the Boltzmann Machine. It is a generalization of the Hopfield network, and modifies it in two significant ways:

- A stochastic update rule is used during recall (which allows the system to escape from local minima in the energy surface).
- For both, the learning and recall phases, the BM uses simulated annealing (in analogy to the metallurgical term annealing in which low energy states of a metal are achieved by first raising the metal to a very high temperature and then gradually cooling it). This assists convergence to the global minimum solution.

A BM operating at high temperatures behaves much like a random model and at low temperatures, much like a deterministic model. At intermediate temperatures it has a stochastic component whose magnitude depends upon the temperature parameter. The BM has a probabilistic component in that the individual neurons or nodes, are stochastic rather than deterministic. A delta energy level is computed for the two possible states of a node and then the node is active with

a probability given by the Boltzmann distribution, hence the name Boltzmann Machine. The probability of a node being on is given by:

$$p = 1/(1 + E^{-\Delta E/T})$$

where E is the energy of the node, ΔE is the delta energy level, and T is the temperature.

Because of this stochastic component, a node can sometimes assume a new state value that increases instead of decreasing the overall energy of the system. This mimics physical annealing and assists in escaping local local minima and moving towards a global minimum [102].

All states of the system are possible at thermal equilibrium (although some are more likely than others) and their distribution can be computed from the Boltzmann distribution. If the probabilities of the system states are close to the environment states, then the BM accurately models the environment.

The learning rule in BM is based on comparing the results of a free running system to a system whose inputs and outputs (or either) are forced to remain at an appropriate value i.e. clamped. The probabilities of the states are compared for the two runs and weights locally adjusted so as to improve the accuracy with which the BM models the environment.

The learning algorithm in BM was the first model to propose solutions using hidden units in a neural network, introducing the notion of internal representation of features of the problem. However, the computing power necessary to implement a BM is very large; a single step in the learning process necessitates the estimation of a probability distribution. Traditionally, the BM are viewed as a class of binary stochastic recurrent networks. Recently however, a polytomous (multi-category) BM has been developed by Anderson *et al* [3] which employs neurons with polytomous responses rather than simple binary responses. The BM can solve constraint satisfaction problems, but has so far been little used in real applications.

Real-Time Recurrent Networks

The Real-Time Recurrent Neural Network (RTRNN) [118] (or more generally, the fully Recurrent Neural Network (RNN) [67]) differs from the Hopfield network in two ways [56]:

- The network contains hidden nodes;
- The network has arbitrary dynamics.

The structure of a general (m,n)RTRNN with m external inputs and n fully interconnected nodes, comprises two layers namely, a concatenated input-out layer (comprising the m term external input vector and the one-step delayed n term output vector); and a processing layer comprising the n hidden nodes. The activation of all or some of the hidden nodes can be considered as the output of the network and all the nodes can be trained to produce desired outputs [56]. The network is fully inter-connected in that there are a total of mn forward connections and n^2 feedback connections (of which n are self-feedback connections).
The dynamics of the RTRNN can be described by the following set of equations:

$$s_i(k+1) = \sum_{j=1}^{n} w_{i,j}(k) y_j(k) + \sum_{j=1}^{m} w_{i,j+n}(k) x_j(k) \tag{2.1}$$

$$y_i(k+1) = f(s_i(k+1)) \quad \text{for} \quad i = 1, \ldots, n \tag{2.2}$$

where $w_{i,j}(k)$ represents the weight of the connection from the j^{th} to the i^{th} node at time k, and the activation function $f(.)$ can be any real function differentiable with respect to its argument (typically chosen to be the $tanh(.)$ function).

Several algorithms have been proposed for training of RNN structures represented by the above dynamics. Two of the more well known include the Back-Propagation Through Time and the Real Time Recurrent Learning algorithms which are summarized below:

- **Back-Propagation Through Time (BPTT) Learning:**

  In this approach, the idea is to convert the network from a feedback system into a purely feedforward system by unfolding the network over time. This

is derived from the fact, that if the system processes a signal that is n time steps long, then n copies of the network can be created. The feedback connections are modified and become feedforward connections from one network to the subsequent network. The resulting system can then be trained by the standard BP by treating it as one large feedforward network with the copied weights being treated as shared weights [69]. This approach is known as the BPTT learning. A variant of this approach known as the *Truncated Back Propagation Through Time* has also been proposed [119] which tries to approximate the true gradient by unfolding the network over the last p time steps (less than n). In this case, only p copies of the weights are made and normal BP with weight sharing is used as before. Obviously, if critical information for a problem occurs more than p time steps into the past then performance degradation will result. The main problem with these approaches is the large memory cost required to maintain several copies of the network. This can be overcome by using the other approach discussed below:

- **Real Time Recurrent Learning (RTRL):**

  In this popular approach [118], the gradient is calculated recursively, and its use in fact gives the RNN its name of Real-Time Recurrent Neural Network (RTRNN) [56]. Although,this approach is more memory efficient than the Back-Propagation Through Time, the recursive process nevertheless incurs a significant computational requirement - the order of $O(n^4)$ where n is the number of network nodes. The complete RTRL algorithm is derived in Appendix B.

  Recently, Srinivasan *et al* [38] have presented formal convergence proofs for both the Back-Propagation Through Time, and the Real Time Recurrent Learning algorithms. They have also shown that Truncated Back Propagation Through Time is sufficient for convergence.

Locally Recurrent Globally Feedforward Networks (LRGFN)

Very recently, a number of researchers have experimented with a class of architectures that fall in between feedforward only MLP type architectures, and fully recurrent network architectures (e.g., the Real Time Recurrent Network described above). This class of architectures has been called Locally Recurrent Globally Feedforward Networks (LRGFN) by Tsoi and Back [82], who after a careful review identified the following three major LRGFN architectures that have been reported to date:

1. *LRGF Networks with Local Synapse Feedback:*

   These architectures are exactly the same as the the Feedforward ANNs, except that the synapses can be expressed as a linear transfer function (to give them dynamics) instead of just constants. Included in this class of architectures, are the purely feedforward MLP structure employing the McCulloch-Pitts based neurons reported by Lapedes-Farber [33] for non-linear time series prediction; the Back-Tsoi Infinite-Impulse-Response (IIR) synapse MLP architecture [7] in which each synapse incorporates feedback in the form of an IIR filter; and the De Vries-Principe architecture [112] in which they have generalized the time-delay synapses into a type of an IIR structure by the introduction of a Gamma Operator in place of the usual z^{-1} operator. This class of structures includes a special case of the LRGF networks with local activation feedback, which are described next.

2. *LRGF Networks with Local Activation Feedback:*

   In this case, the feedback is taken before the activation signal (which is the linear sum of the weighted inputs) enters the non-linear activation function. The transfer function of the feedback path may in general, comprise both zeros and poles.

3. *LRGF Networks with Local Output Feedback:*

In these networks, as the name suggests, the feedback path is taken around the non-linear activation function, that is, after the signal has passed through the non-linearity. This feedback can take the form of a simple all pole transfer function, or a more general linear transfer function. Included in this category are the Frasconi-Gori-Soda architecture [49], and the Poddar-Unnikrishnan architecture also known as Memory Neural Networks (MNNs) [37], which are essentially feedforward networks with each neuron having a feedback transfer function comprising of one pole. The MNNs can be considered to be a special case of the generalized Frasconi-Gori-Soda architecture whose feedback transfer function can have both poles and zeros, and have recently been successfully employed in the identification and control of noiseless non-linear dynamical systems [37].

Note that in these architectures, feedback is only allowed from the output of the neuron itself hence resulting in the name local output feedback, and not from other neurons, in which case a globally recurrent network results, such as the RTRNN discussed above. The synapses of the LRGF networks with local output feedback are simply constants.

Note that another important category of ANNs namely Unsupervised ANNs (such as Kohonen's Self-Organizing Map (SOM) [72] and its variants [56]) have not been addressed in this thesis. A good review of their structures and learning algorithms can be found in [56]. In the next section, the use of the Feedforward and Recurrent ANNs as models of non-linear dynamical processes is investigated.

2.4 ANN based models for Identification of Non-linear Dynamical Systems

Much success has been achieved in the use of feedforward neural networks for modelling static non-linear maps [24]. It is now known that such networks are in fact capable of approximating not only any continuous map arbitrarily closely

[26], but also the derivatives of such maps [13]. There is a growing interest in extending this performance to the dynamic case - modelling non-linear dynamical systems using feedforward and recurrent ANNs.

2.4.1 The Problem

Assume that a set of measurements can be carried out on a non-linear dynamical process. From this data, a predictor model must be derived whose dynamical behaviour is as close as possible to that of the process. The identification of the non-linear process is synonymous with the estimation of the parameters of the predictor model, based on the available data. If the predictor model is implemented as an ANN, then identification corresponds to successful training of the neural network [86].

In the following sections 2.4.2 to 2.4.4, we use non-linear generalizations of three popular non-linear dynamical system models namely, the output-error, the NARMAX and the NARX (equation-error) models; corresponding to three different assumptions on the noise [79]. Single Input Single Output Systems (SISO) have been considered to clarify the concepts. The predictor associated to each model is described. In each case, we show which class of neural networks (that is, feedforward or recurrent) is most appropriate when used to implement the non-linear predictor.

Finally in sections 2.4.5 and 2.4.6, two other important classes of non-linear dynamical systems namely the NAR and chaotic time series processes are described along with their corresponding ANN based predictors.

2.4.2 The Output-Error Model and Recurrent ANNs

In the output-error model, it is assumed that the process (measured) output $y_p(k)$ obeys the following equations:

$$\begin{aligned} x_p(k) &= \phi[\mathbf{x_p}(k-1), \mathbf{u}(k-1)] \\ y_p(k) &= x_p(k) + e(k) \end{aligned}$$

where

$$\mathbf{u}(k-1) = u(k-1), \ldots, u(k-m)$$

are the lagged external inputs of the process, and

$$\mathbf{x_p}(k-1) = x_p(k-1), \ldots, x_p(k-n)$$

are the lagged internal states of the process, $e(k)$ is a white noise sequence and $\phi(.)$ is some continuous non-linear function. Hence, an output-error model describes a process with additive output noise.

The output of the associated predictor (or modeller) $y(k)$ illustrated in Figure 2.3, such that $y_p(k) - y(k) = e(k)$ is given by:

$$y(k) = \phi[\mathbf{y}(k-1), \mathbf{u}(k-1)] \tag{2.3}$$

where $\mathbf{y}(k-1) = y(k-1), \ldots, y(k-n)$.

Therefore, as can be seen from the above equation 2.3, the associated predictor of the output-error process is recurrent of order n and has to possess non-linear modelling capabilities in order to approximate the non-linear function $\phi(.)$. Hence, to implement the above predictor as a neural network, a feedforward neural network will obviously not be appropriate as it lacks the auto-regressive component that is required by the above output-error model predictor. Consequently, a dynamic Recurrent Neural Network type structure (such as the Real Time Recurrent Neural Network (RTRNN) described in section 2.3.2), trained by an appropriate algorithm (such as the RTRL) will have to be employed for effective approximation of the above $\phi(.)$ non-linear continuous function. A major problem with the conventional RNN structures however, as discussed in section 2.3.2, is the high computational complexity associated with their learning algorithms.

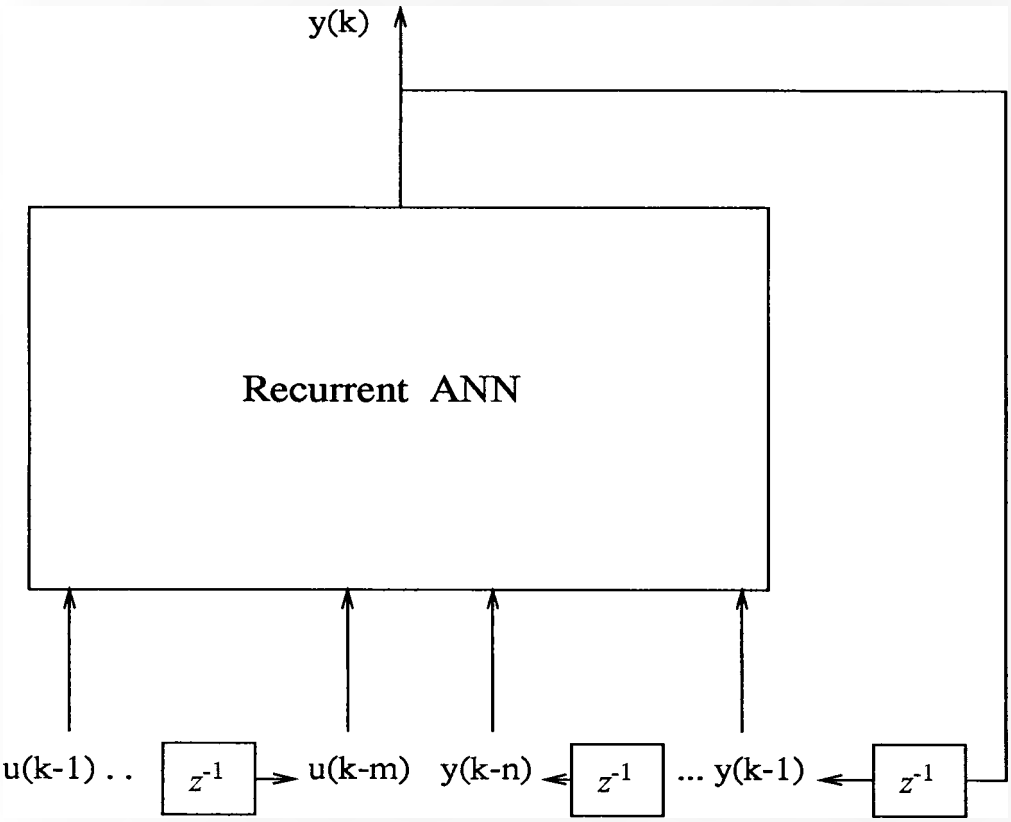

Figure 2.3: Associated Neural Predictor of the output-error process

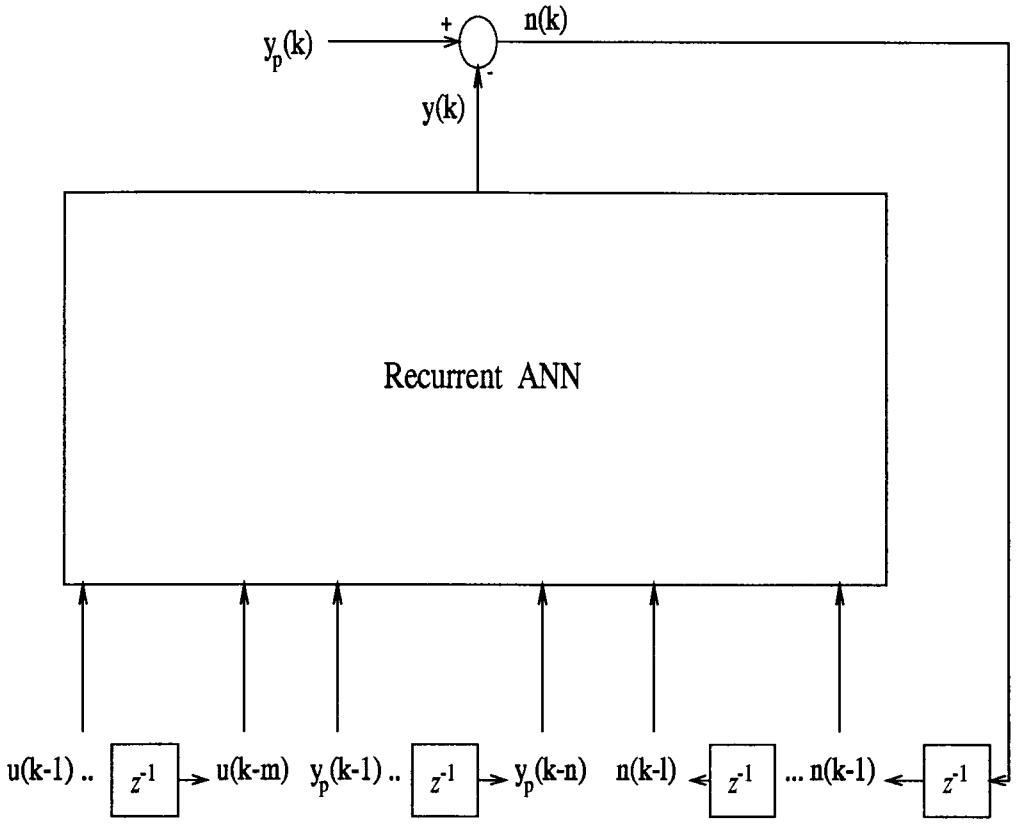

Figure 2.4: Associated Neural Predictor of the NARMAX process

2.4.3 NARMAX model and Recurrent ANNs

A wide class of discrete time non-linear dynamic systems can be represented by the Non-linear Auto-Regressive Moving Average with eXogenous inputs (NARMAX) model [180]. The NARMAX model describes a description of the system of some non-linear functional expansion of lagged inputs, outputs and prediction errors as follows:

$$\begin{aligned} x_p(k) &= \phi[\mathbf{x_p}(k-1), \mathbf{u}(k-1), \mathbf{e}(k-1)] + e(k) \\ y_p(k) &= x_p(k) \end{aligned}$$

where $\mathbf{e}(k-1) = e(k-1), \ldots, e(k-l)$ and $\mathbf{x_p}(k-1) = x_p(k-1), \ldots, x_p(k-n)$.

The associated neural predictor illustrated in Figure 2.4 for the NARMAX

model has an output $y(k)$ defined by [52]:

$$y(k) = \phi[\mathbf{y_p}(k-1), \mathbf{u}(k-1), \mathbf{n}(k-1)] \tag{2.4}$$

where $\mathbf{y_p}(k-1) = y_p(k-1), \ldots, y_p(k-n)$ and

$\mathbf{n}(k-1) = n(k-1), \ldots, n(k-l)$ where $n(k) = y_p(k) - y(k)$.

Thus, the predictor of the NARMAX model is also recurrent of order l, and as before any dynamic RNN based predictor can be used to approximate the non-linear function $\phi(.)$.

2.4.4 NARX (Equation-Error) Model and Feedforward ANNs

A special case of the NARMAX model is referred to as the Non-linear Auto-Regressive with eXogenous input (NARX) model (or the equation error model) and is assumed to obey the following equations [180]:

$$\begin{aligned} x_p(k) &= \phi[\mathbf{x_p}(k-1), \mathbf{u}(k-1)] + e(k) \\ y_p(k) &= x_p(k); \end{aligned}$$

As can be seen, the above equation-error system (which describes a process with additive state noise) is a simplified form of the NARMAX system, as the white noise source is now assumed to enter the system additively. In general however, the noise source may be correlated and can enter the system in a more complicated manner such as in the manner accommodated by the NARMAX system representation.

The associated predictor of the NARX process illustrated in Figure 2.5, has an output $y(k)$ given by:

$$y(k) = \phi[\mathbf{y_p}(k-1), \mathbf{u}(k-1)] \tag{2.5}$$

where $\mathbf{y_p}(k-1) = y_p(k-1), \ldots, y_p(k-n)$ are the measured process outputs.

As can be seen from equation 2.5 above, the predictor of the equation-error process is non-recursive and *feedforward* of order $(n+m)$ with inputs comprising $\mathbf{y_p}(k-1)$, and $\mathbf{u}(k-1)$ (the external inputs of the process). Therefore, the

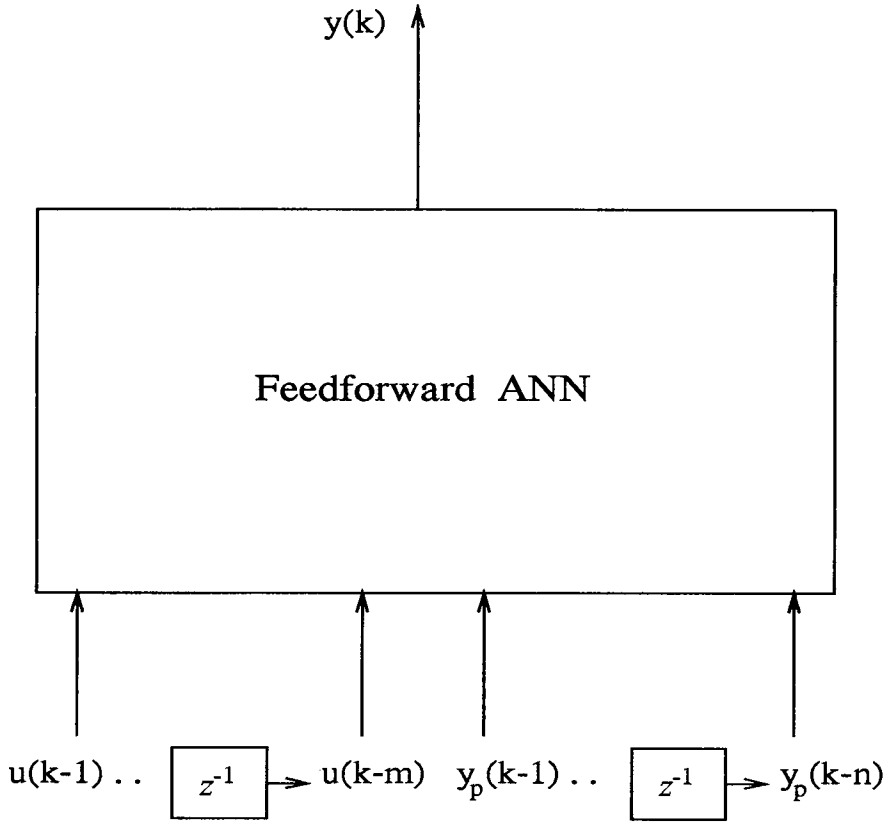

Figure 2.5: Associated Neural Predictor of the equation-error (NARX) process

NARX predictor can be implemented by a feedforward ANN. It is important to understand how the above dynamic system whose output is a non-linear function of the past inputs and outputs, can be modelled by feedforward ANNs which are versatile static maps. Since the output of a feedforward ANN is a non-linear function of its current inputs only, it should presumably model only memoryless transformations as discussed earlier. However, the use of a tapped delay line can provide a means for all the necessary past inputs and outputs of the system being modelled, to be fed as explicit inputs to the network. Hence, the non-linear transformation $\phi(.)$ that captures the dependence of the output of the NARX system on the specified m past inputs and n past outputs can be captured by any of the feedforward ANNs discussed in Appendix A, such as has been demonstrated for the RBF networks [187] [192] [20] [18] [22], and the MLP [182] [34] [110] [36], and its variant termed the Time-Delay Neural Network (TDNN) (also known as the FIR-MLP) [113], in which each MLP synapse is represented by a Finite-Impulse Response (FIR) filter to give it dynamics.

However, in the *highly non-linear in the parameters* multi-layer feedforward MLP type structures, that have been employed to-date for approximating $\phi(.)$ [182] [34] [36], nonlinear transformations are performed on the data at both input and output layers thus requiring the use of complex non-linear learning algorithms. As discussed in Appendix A.6, this leads to problems such as slow convergence and multi-minimum error surfaces. Furthermore, a theoretical analysis of such multi-layered networks is also difficult. Optimisation techniques such as genetic algorithm, learning automata and simulated annealing, although capable of achieving the global minimum also require extensive computation [66]. Another major problem associated with multilayer structures as discussed in Appendix A.6 is the determination of the optimum network size (number of hidden layers, nodes etc) [24].

The recently proposed single-hidden layered (or two-layered) linear-in-the-parameters, feedforward ANNs namely, the Radial Basis Function (RBF) [12] and the Volterra Neural Networks [40] [63] (discussed in Appendix A.4 and Appendix

A.5.1) can also approximate $\phi(.)$, and alleviate the need of non-linear learning algorithms in the output layer by employing linear weight updating rules. In the case of the RBF however, it has been shown in a study on modeling of non-linear dynamical systems by Weigend and Rumelhart *et al* [115], that a prohibitively large number of basis functions may be needed to cover high dimensional input spaces along with expensive computations required by the additional learning algorithms needed for updating the position and spread of basis functions in the input hidden layer. In a recent analysis by Narendra [68], the advantages of the RBF networks over MLP structures are firstly noted namely, rapid training and ease of analysis. However, the main downside of the RBF networks has also been demonstrated: namely, an exponential growth in complexity with the number of network inputs. On the other hand, the Volterra Neural Network (VNN) based structures suffer from slow convergence [164] and exponentially increasing hidden layer size with increasing network input dimensions [138].

In chapter 4 of this thesis, a new linear-in-the-parameters Feedforward Functionally Expanded Neural Network (FFENN) has been proposed as an alternative two-layer feedforward ANN for modelling the $\phi(.)$ function in the equation-error model above. It has also been employed to model two other important classes of non-linear dynamical systems namely, the Non-linear AR (NAR) [33] and chaotic time series processes [15], which are described next.

2.4.5 NAR model and Feedforward ANNs

The Non-linear Auto-Regressive (NAR) model is an extension of the linear Auto-Regressive (AR) model used for time series modelling [52] and can be expressed as:

$$y(k) = \phi(\mathbf{y}(k-1)) + e(k) \tag{2.6}$$

where $\mathbf{y}(k-1) = y(k-1), \ldots, y(k-n)$ and $e(k)$ is uncorrelated white noise of zero mean and assumed to have a finite variance σ^2. The associated **optimal**

predictor has an output $\hat{y}(k)$ given by [52]:

$$\hat{y}(k) = \phi(\mathbf{y}(k-1)) \tag{2.7}$$

This predictor can be seen to be similar to the predictor for the NARX model, that is non-recursive feedforward, but of order n rather than $(n+m)$. Hence, for this predictor, Feedforward ANNs (FANNs) would be the appropriate choice as has been demonstrated by Lapedes *et al* [33], who were the first to propose use of FANNs as a NAR model for time series prediction. A feedforward ANN provides a non-linear approximation to $\phi(.)$ given by

$$\hat{y}(k) = \hat{\phi}(\mathbf{y}(k-1)) \tag{2.8}$$

where $\mathbf{y}(k-1) = y(k-1), \ldots, y(k-n)$ are the inputs to the feedforward ANN predictor.

2.4.6 Chaotic Processes and Feedforward ANNs

Chaos is the very complex behavior of a dynamical system that is both non-linear and deterministic [103] [104]. It represents a powerful notion permitting the use of a deterministic system to explain highly irregular fluctuations exhibited by many physical phenomena encountered in nature [57]. In practice, a discrete-time, deterministic, non-linear dynamical system is said to be chaotic if it has the property of sensitive dependence on initial conditions and has a relatively small number of degrees of freedom [19]. Increasingly though, the term chaotic is also used for stochastic systems exhibiting a sensitive dependence for all possible initial conditions [19]. Alternatively said, deterministic chaos is characterized by an exponential divergence of nearby trajectories. For the problem of time series prediction, which is synonymous with modelling of the underlying physical mechanism responsible for its generation; there are two related consequences: Firstly, since the uncertainty of the prediction increases exponentially with time, chaos precludes any long-term predictability. Secondly, on the other hand, it allows for short term predictability, that is, a random looking time series might

have been produced by a deterministic system and actually be predictable in the short run. A prediction algorithm for chaotic systems thus has to capture the short term structure of the time series. The short-range structure of chaotic behaviour can be captured by expressing the present value of the chaotic time series sample $y(k)$ as a function of the previous d values of the time series:

$$y(k) = \phi(\mathbf{y}(k-1)) \tag{2.9}$$

where the vector $(\mathbf{y}(k-1) = y(k-1), \ldots, y(k-d))$ lies in the d-dimensional state space.

An efficient method of fitting the non-linear function $\phi(.)$ is to use a feedforward neural network predictor with a single output [15] [22] [19]; the inputs to the network being the observations $(y(k-1), y(k-2), \ldots, y(k-d))$. The output of the feedforward ANN predictor is given by (as per the NAR predictor):

$$\hat{y}(k) = \hat{\phi}(\mathbf{y}(k-1)) \tag{2.10}$$

In real-world chaotic processes, intrinsic noise $e(k)$ will be present and the task of the neural network predictor will be to reconstruct $f(.)$ without modelling the noise, as in the case of the NAR predictor.

The goal of identification of noisy non-linear dynamical systems such as the output-error system (with additive output noise) and the equation error system (with additive state noise), as well as NAR and noisy chaotic time series, is to find a neural network predictor that implements a function as close as possible to $\phi(.)$ in a bounded domain of state space. Therefore, the prediction error $(y(k) - \hat{y}(k))$ should be as close as possible to the noise $e(k)$ once the training is completed.

2.5 Conclusions

The two main ANN paradigms namely, the Feedforward and Recurrent ANNs have been discussed. Traditionally used structures belonging to each category, have been described with particular emphasis on the relative advantages and disadvantages of their structural and computational requirements. Their application

to modelling of various classes of discrete-time non-linear dynamical systems has also been illustrated. It has been shown that the choice of the correct ANN model (that is, whether feedforward or recurrent) is crucial to effective modelling of a particular class of the non-linear dynamical system.

In the next chapter, the application of conventionally used linear and ANN based non-linear adaptive equalizers to two important digital communications applications is investigated.

Chapter 3

ANNs for Digital Communications Applications

3.1 Introduction

In recent years, non-linear adaptive filters based on Artificial Neural Network models have been used successfully for system identification and noise-cancellation in a wide class of applications [56]. This chapter reviews the application of ANNs to two important problems encountered in digital communications namely,

1. *Digital Communications Channel Equalization:* This problem is generally concerned with the task of reconstructing digital signals that have been passed through communication channels in the presence of both linear and non-linear distortion and further corrupted by additive noise.

2. *Overcoming Co-channel interference in Digital Communications Systems:* The performance of many digital communications systems employing frequency reuse such as cellular radio, is also significantly impaired by the problem of co-channel interference. Efficient equalization schemes are required in these communications systems in order to achieve an acceptable error-rate performance.

Sections 3.2 and 3.3 investigate the various linear and non-linear approaches proposed to date for solving the above problems.

3.2 Digital Communications Channel Equalization

The demand for very high speed efficient data transmission over physical communication channels has been substantially increased over the last decade and a half [55]. Telephone channels, radio channels, and even fibre optic channels often have non-flat frequency responses and non-linear phase responses in the signal passband [71]. Sending digital data at high speeds through these channels often results in a linear distortion phenomenon called Inter-Symbol Interference (ISI), caused by signal pulse smearing in the dispersive medium. As a result of ISI, the transmitted symbols are spread and overlapped over successive time intervals [149]. In addition to linear distortion, the transmitted symbols are also subject to other impairments such as thermal noise, impulse noise and $non-linear$ distortion arising from the modulation-demodulation process, crosstalk interference, the use of amplifiers and converters, and the nature of the channel itself [55]. The above factors limit the maximum attainable data-rate in digital communications systems. All the signal processing techniques employed at the receiver's end to combat the introduced channel (linear and non-linear) distortion and recover the transmitted symbols are referred to as equalization schemes. Adaptive equalization is in general, characterized by the structure of the equalizer, its adaptation algorithm and the use or not of training sequences [153].

Two basic categories of adaptive equalizers exist, namely the sequence estimation and symbol-decision equalizers [194]. The optimal sequence estimation equalizer is the adaptive Maximum Likelihood Sequence Estimator (MLSE) [123] which is known to provide the best attainable performance in combating channel ISI in the presence of additive white Gaussian noise. The adaptive MLSE is implemented in the form of a channel estimator and a Viterbi algorithm. However,

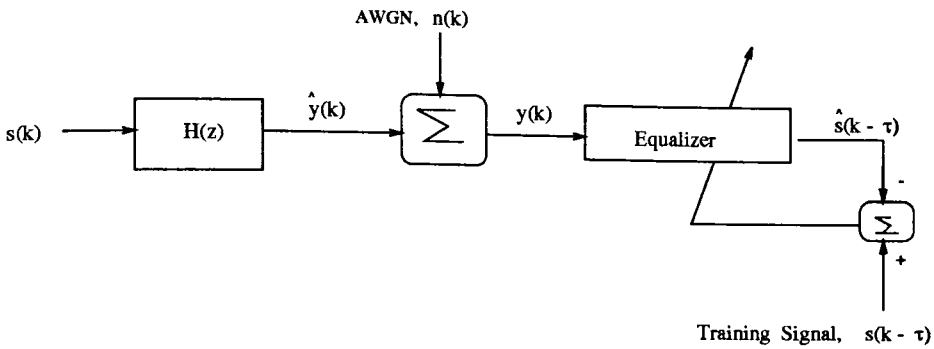

Figure 3.1: Schematic of Data-Transmission System

the high computational complexity and deferring decisions associated with the MLSE are often unacceptable in many practical communications systems [188]. Most of the equalization applications today therefore employ equalizers that operate symbol-by-symbol [172]. Symbol decision equalizers can be classified into two categories namely, the direct-modeling equalizers in which the channel model is identified explicitly, and the indirect-modeling equalizers which recover the transmitted symbols by directly filtering the channel observations, usually using an adaptive linear filter (the Linear Transversal Equalizer (LTE) [149]), without estimating a channel model explicitly. The indirect-modeling approach is by far most widely used and it is considered in this thesis. A discussion of the indirect-modeling linear and non-linear neural network based equalizer structures for the equalization of linear and non-linear communications channels in the presence of ISI and AWGN is reported later in this chapter.

3.2.1 Problem Statement

A general discrete-time model of the digital communications system is depicted in Figure 3.1, where the channel includes the effects of the transmitter filter, the transmission medium, the receiver matched filter and other components [154].

A widely used model for a linear dispersive channel is the FIR model [149],

which can be expressed as:

$$\hat{y}(k) = \sum_{i=0}^{l} a_i s(k-i) + n(k) \tag{3.1}$$

where $l+1$ is the length of the channel impulse response and a_i are the channel taps. In general, both $s(k)$ and a_i can be complex valued. In this thesis, channels and symbols are restricted to be real valued which corresponds to the use of multi-level Pulse Amplitude Modulation (M-ary PAM) [148], with a symbol constellation defined by

$$s_i = 2i - M - 1, \;\; 1 \leq i \leq M \tag{3.2}$$

The real case has been used so that the basic principles and concepts can be highlighted. In particular, the case of binary signals ($M = 2$) provides a very useful geometric visualization of the equalization process. All the results presented in this thesis for PAM can be extended to the general case of complex valued channels with Quadrature Amplitude Modulation (QAM) scheme [153]. The symbols $s(k)$ are further assumed to be equi-probable and independent with the following properties:

$$E(s(k)) = 0$$

$$E[s(k_1)s(k_2)] = \sigma_s^2 \delta(k_1 - k_2)$$

where $E[.]$ denotes the expectation operator, σ_s^2 is the symbol variance, and $\delta(k)$ is the Dirac delta function [154]. The channel output $\hat{y}(k)$ is further assumed to be corrupted by Additive White Gaussian Noise (AWGN) $n(k)$ specified by:

$$E(n(k)) = 0$$

$$E[n(k_1)n(k_2)] = \sigma_n^2 \delta(k_1 - k_2)$$

where σ_n^2 is the noise variance. Also, $s(k)$ and $n(k)$ are assumed to be uncorrelated. The above channel model has been used extensively to model a variety of communications systems, such as the HF communications channel [152] [153].

If non-linear distortion is taken into account, which is now a significant factor hindering further increase in the maximum attainable data rate [184], the general channel model will be non-linear [161]. Although sources of channel non-linearity such as non-linearity in data converters may be regarded as memoryless, these non-linear components are connected to or embedded in a linear dynamic network and, consequently the overall channel response is a non-linear dynamic mapping. That is, the received signal at each sample instant is a non-linear function of the past values of the transmitted symbols [184]. The general non-linear channel model can be represented as:

$$\hat{y}(k) = f(s(k), \ldots, s(k-l); A) + n(k)$$

where f is some non-linear function, A is the channel parameter vector and $n(k)$ is the additive noise which can be non-Gaussian and correlated.

The task of any indirect-modeling equalizer is: given the overall channel observation $y(k)$, estimate the transmitted data $s(k)$. The symbol decision equalizer at any sample instant k processes the m most recent channel observations, and makes a decision $\hat{s}(k-\tau)$ regarding the symbol transmitted at $k-\tau$, where the integers m and τ are referred to as the equalizer order and delay respectively. How the m channel observations are processed determines the performance and complexity of the equalizer. Depending on whether the equalizer knows the originally transmitted symbols or not, it is characterized as trained adaptation or blind equalizer respectively. In blind equalization, the equalizer has no exact knowledge of the transmitted sequence. The adaptation of the equalizer is attempted in a way to match the statistics of the output of the equalizer to those of the transmitted sequence. A good review of blind equalization techniques can be found in [162]. Recently, a new class of blind equalizers based on Higher Order Statistics (HOS) have been reported by Hatzinakos *et al.* [6] and Alkulaibi and Soraghan *et al* [5]. In this thesis we shall restrict our discussion to trained adaptation equalization schemes.

During the training period of most adaptive equalization systems (including

cellular mobile radio), the reference desired signal $s(k-\tau)$ which is to be reconstructed, is available. After the training period, the equalizer can use its own decision $\hat{s}(k-\tau)$ for continuous fine-tuning of the weights. This type of weight update is referred to as Decision Directed Mode (DDM) [142]. In normal operation, the equalizer's decisions after sufficient training are correct with a high probability thus enabling the adaptive equalizer to maintain precise equalization [148]. Moreover, a decision-directed adaptive equalizer can track slow variations in the channel characteristics or linear perturbations in the receiver front end, such as slow jitter in the sampler phase [149].

3.2.2 Adaptive Equalizers

This section introduces the two main categories of adaptive equalizers namely, the symbol decision and sequence estimation equalizers. Two well known symbol decision structures are the Transversal Equalizer (TE) (also called Feedforward Equalizer) and the Decision Feedback Equalizer (DFE). The task of a TE is essentially to use the information in the observed channel output vector

$$\mathbf{y}(k) = [y(k) \ldots y(k-m+1)]^T$$

to produce an estimate $\hat{s}(k-\tau)$ of the actual transmitted symbol $s(k-\tau)$, where m and τ are as defined above. The conventional TE is the Linear TE (LTE) which is discussed in the next section. It has been shown in [172] [173] [188] that channel equalization is inherently a non-linear problem and non-linear equalizer structures are therefore required to achieve fully or near optimal equalization performance. Recently, non-linear adaptive TE filters based on fuzzy logic and trained by LMS and RLS algorithms have been proposed by Wang *et al* [114] to provide near-optimal performance for the case of second order fuzzy equalizers. However, in a new study by Ralph and Soraghan [81], it has been shown that the performance of the fuzzy adaptive filters deteriorates significantly from the optimal equalization solution if their order is increased beyond two. A review of the various non-linear neural network based structures reported to date is given

in section 3.2.5 of this chapter.

A powerful technique to improve the equalization performance is to employ decision feedback [149]. The advantage of the DFE (which is a simple non-linear structure) [9] [128] is that ISI is eliminated without enhancement of noise by using past decisions, and a disadvantage is the possible error propagation caused by decision errors. The operation of the conventional DFE is based on the m most recent channel observations and p past decisions. The conventional DFE consists of two linear finite filters known as feedforward and feedback filters; and produces an output given by:

$$o(k) = \hat{f}(\mathbf{y}(k), \hat{\mathbf{s}}(k-\tau)) = \mathbf{b}^T\mathbf{y}(k)+\mathbf{c}^T\hat{\mathbf{s}}(k-\tau) = \sum_{i=1}^{m} b_i y(k-i+1)+\sum_{i=1}^{p} c_i \hat{s}(k-i-\tau)$$

where b_i and c_i are the coefficients of the feedforward and feedback filters respectively; with the integers m and p being referred to as the feedforward and feedback filter orders respectively. The conventional DFE uses linear algorithms such as the LMS or RLS for its weight updates. It is important to note that the feedforward filter within the conventional DFE is a linear TE (LTE) whose performance is limited by hyperplanes. The linearity of these decision regions in the input signal space has been shown to limit the performance of the conventional DFE [185]. Siu *et al* [185] proposed a new approach for the DFE using the Multi-Layered Perceptron (MLP) as the feedforward filter, and demonstrated its superior performance over the conventional DFE both in terms of equalizer convergence times and tolerance to noise. Chen *et al* [193] and more recently Theodoridis *et al* [160] also applied the Radial Basis Function (RBF) neural network to the DFE to enhance equalizer performance and reduce computational complexity.

In chapter 6 of this thesis we propose a new approach for the DFE based on the conventional Functional-Link Neural Network (FLNN) and compare its performance with other recently reported Feedforward (TE) and DFE structures.

The following sections focus on the three main classes of equalizers proposed to date for solving the Digital Communications Channel Equalization problem

stated in section 3.2.1; namely:

1. Optimal Sequence Equalizers (section 3.2.3)
2. Linear Symbol-Decision Equalizers (section 3.2.4)
3. Nonlinear Symbol-Decision Equalizers (section 3.2.5)

3.2.3 Optimal Sequence Equalizers: The MLSE

The optimal solution to the equalization of a FIR channel model in the presence of ISI and additive white Gaussian noise can be derived by using the principle of Maximum Likelihood detection of the entire transmitted sequence. The resulting structure is known as the Maximum likelihood Sequence Estimator which is discussed below.

The optimal Maximum Likelihood Sequence Estimator (MLSE)

The objective of the MLSE is to determine the one transmitted sequence out of all possible transmitted sequences that is closest (according to a specific probability measure) to the actual received sequence. The optimal solution for this class of equalizers is the Maximum Likelihood Viterbi Algorithm [143] [154], which determines the estimated symbol sequence $(\hat{s}(1), \ldots, \hat{s}(k), \ldots)$ by minimizing the cost

$$J = \sum_{k=1}^{\inf} (y(k) - \sum_{i=0}^{l} a_i \hat{s}(k-i))^2$$

The MLVA provides the lowest error-rate attainable for any FIR channel in the presence of additive white noise, provided the channel is known. When the channel response is unknown, a channel estimator can be employed to provide a channel estimate [123] and this gives rise to an adaptive MLVA [194]. Deferring decisions are essential in such an architecture, and in practice a final decision is made after some fixed delay which must be chosen large enough so that the performance degradation due to pre-mature decisions is negligible. Such a long

delay is not welcome in many practical applications. As the operation is based on a large segment of the received data sequence, the computation and memory requirements of the adaptive MLVA are far more than those of the simple architectures based on symbol by symbol decisions. The Viterbi Algorithm is also sensitive to the assumption of AWGN. The assumption of AWGN in digital communications channels is only an approximation because, although the channel noise itself is generally Gaussian and white, the matched filter at the receiver will colour the noise [154]. Consequently, for coloured noise, performance loss will occur. Additionally, the adaptive MLVA is sensitive to the channel estimate, that is, if the channel estimator does not provide a reasonably good estimate of the true channel model, then significant performance degradation is expected. For stationary channels, a fairly accurate estimate of the channel model can be made and consequently the MLVA performance degradation is not significant. However, it has been recently shown in a new study by Chen, McLaughlin and Grant [198] that performance degradation of the adaptive MLVA becomes serious for highly non-stationary channels, and this degradation is inherent in the structure of the MLVA.

3.2.4 Linear Symbol-Decision Equalizers

The indirect modeling equalization approach is sometimes referred to as inverse modeling, because traditionally, the equalization problem is viewed as an inverse filtering problem in which the equalizer forms an approximation to the inverse of the distorting communications channel [149]. From this viewpoint, the filter within the general TE architecture becomes linear and, the resulting equalizer is called a Linear Transversal Equalizer (LTE) [148]. An LTE of feedforward order m produces an output at time k given by:

$$o(k) = \hat{f}(\mathbf{y}(k)) = \mathbf{b}^T\mathbf{y}(k) = \sum_{i=1}^{m} b_i y(k-i+1)$$

where $\mathbf{b} = [b_1 \dots b_m]^T$ are the coefficients or weights of the m-th order feedforward LTE. A decision slicer then quantizes the LTE's output $o(k)$ to produce an

estimate $\hat{s}(k-\tau)$ of the actual transmitted symbol $s(k-\tau)$.

Considerable research has been performed on the criterion for adapting the LTE filter's weights [153]. Two criteria that have found widespread use in optimizing the linear equalizer weights are the Peak Distortion criterion [142] [154] and the Mean-Square-Error (MSE) criterion [154]. Minimization of the Peak Distortion (defined as the worst case ISI at the equalizer output) results in the Zero-Forcing (ZF) algorithm for adapting the LTE weights. However the ZF algorithm does not take account of the channel noise and is only guaranteed to minimize the worst case ISI if the peak distortion before equalization is less than 100%; a condition which is often not met at high speeds on channels with high eigen value ratios [149]. These limitations do not exist if the MSE criterion is used in the adjustment of the equalizer coefficients. Today, most of the adaptive algorithms for equalizers are based on the criterion of minimizing the MSE between the desired equalizer output and its actual output, that is, the learning algorithms adjust the filter parameters to achieve a minimum of the criterion (where $\mathbf{E}$ is the statistical operator) :

$$\mathbf{E}[(s(k-\tau) - \hat{f}(\mathbf{y}(k)))^2]$$

For the LTE, the optimum weight vector which will yield the minimum MSE is readily given by the Weiner solution [153]:

$$\mathbf{b}_{opt} = (\mathbf{E}[\mathbf{y}(k)\mathbf{y}^T(k)])^{-1}\mathbf{E}[\mathbf{y}(k)s(k-\tau)]$$

However, the solution for $\mathbf{b}_{opt}$ involves inverting the co-variance matrix of the equalizer input vector $\mathbf{y}(k)$ which is computationally expensive. Alternatively, an iterative procedure such as the LMS algorithm can be used, which is based on the method of steepest descent and is robust and computationally efficient. As such the LTE employing the LMS updating algorithm is quite widely used today in most high speed modems [71]. The LMS updating algorithm for the LTE can be written as:

$$\mathbf{b}(k+1) = \mathbf{b}(k) + \mu e(k+1)\mathbf{b}(k+1)$$

where μ is the convergence factor and $e(k)$ is the error signal given by $e(k) = (s(k-\tau) - o(k))$ where $s(k-\tau)$ is the training signal. Alternatively, the RLS algorithm [56] can also be used to achieve faster weight updates but at the expense of additional computations. After the initial training period, the adaptive LTE weights should have converged to b_{opt}. During data transmission, the LTE weights can be continued to be adapted (fine-tuned) in a Decision Directed manner, with the error signal now given by $e(k) = (s(k-\tau) - \hat{s}(k-\tau))$. Several variations to the LTE have been proposed including the Fractionally Spaced TE [144], new adaptive lattice equalizers reported by Grant [146] [147], and frequency domain equalizers [145]. Kalman Filter based equalizers have also been reported in [127], and shown to approximate the Infinite Impulse Response (IIR) Weiner filter solution.

The adaptive LTE inspite of its computational simplicity and robustness has certain shortcomings [194]. Primarily, from the inverse filtering viewpoint, increasing the order of the LTE should lead to a more accurate approximation and better equalization performance. This however is not true due to the noise enhancement phenomenon [184]. It has been shown in [172] [184] [190] that the LTE does not achieve the full performance potential of the symbol decision TE structure, and better performance can be achieved from the same information contained in the equalizer inputs, if more complex (non-linear) filtering methods are employed [184]. Furthermore, for the case of equalization of non-linear communication channels, the LTE will only be able to compensate for the linear distortion [55].

3.2.5 Non-linear Symbol-Decision Equalizers: The Optimal Bayesian and ANN based Equalizers

This section derives the optimal Bayesian or Maximum A posteriori Probability (MAP) symbol decision equalizer structure for the Digital Communications Equalization problem stated in section 3.2.1. This is followed by a review of

the various ANN based adaptive non-linear equalizer structures that have been reported to date to approximate the optimal MAP symbol decision equalizer namely, the MLP, Radial Basis Function (RBF), Volterra Neural Network (VNN) and Functional-Link Neural Network (FLNN) based equalizers.

The Optimal Maximum Aposteriori Probability (MAP) or Bayesian Symbol Equalizer

The optimal solution for the symbol-decision Transversal Equalizer (TE) structure is known as the MAP symbol-decision equalizer [139] [140] [141] and is derived using Bayes decision theory. This Bayesian TE is summarised below for the digital communications system depicted in Figure 3.1, assuming $2-ary$ PAM signalling, which implies that $s(k)$ is now an independent identically distributed $(i.i.d)$ sequence taking values from $+1, -1$ with an equal probablity of 0.5. The above equalization process can be viewed as a classification problem, which seeks to classify the received symbol $y(k)$ into one of the two possible classes, $y(k) = +1$ or $y(k) = -1$ (or, one of the symbol points $s_i, 1 \leq i \leq M$ for $M-ary$ PAM). The general symbol-decision equalizer structure is characterized by the equalizer order m and delay τ. For the general channel of $l+1$ taps given in equation 3.1, there are $r_s = 2^{l+m}$ combinations of the channel input sequence:

$$\mathbf{s}(k) = [s(k) \ldots s(k-m+1-l)]^T$$

This gives rise to r_s points or values of the noise-free channel output vector:

$$\hat{\mathbf{y}}(k) = [\hat{y}(k) \ldots \hat{y}(k-m+1)]^T$$

These points which will be referred to as the desired channel states, can be partitioned into $M = 2$ classes denoted by $Y^+_{m,\tau}$ and $Y^-_{m,\tau}$ depending on the value of the desired transmitted signal $s(k-\tau)$ (which is to be reconstructed):

$$Y^+_{m,\tau} = \hat{\mathbf{y}}(\mathbf{k}) | \mathbf{s}(\mathbf{k}-\tau) = +\mathbf{1},$$

$$Y^-_{m,\tau} = \hat{\mathbf{y}}(\mathbf{k}) | \mathbf{s}(\mathbf{k}-\tau) = -\mathbf{1}$$

Each desired state $\mathbf{y}_i^+ \sum Y_{m,\tau}^+$ or $\mathbf{y}_i^- \sum Y_{m,\tau}^-$ has a priori probability of appearance p_i. Since all the desired channel states are assumed to be equi-probable, all the p_i are equal to $p = 1/r_s$. The number of states in $Y_{m,\tau}^+$ and $Y_{m,\tau}^-$ are denoted as r_s^+ and r_s^- respectively with the values of both being equal to $r_s^+ = r_s^- = r_s/2$. Because of the additive Gaussian noise, the observation vector $\mathbf{y}(k) = [y(k) \ldots y(k - m + 1)]^T$ is a stochastic process having conditional Gaussian density functions centered at each of the desired channel states. It is thus apparent that the channel observations form clusters and the means of these data clusters are the desired states. Determination of the value of $s(k - \tau)$ based on the observation vector $\mathbf{y}(k)$ is a decision problem. Bayes decision theory [139] [141] provides the optimal solution to the general decision problem and is applicable here. Computing the 2 Bayesian decision variables for $2 - ary$ PAM [188]

$$\eta_{+1}(k) = \sum_{i=1}^{r_s^+} p_i p_n(\mathbf{y}(k) - \mathbf{y}_i^+)$$

and

$$\eta_{-1}(k) = \sum_{i=1}^{r_s^-} p_i p_n(\mathbf{y}(k) - \mathbf{y}_i^-)$$

where $\eta_{+1}(k)$ and $\eta_{-1}(k)$ are the conditional probability density functions (pdf)s of $\mathbf{y}(k)$ given that $s(k - \tau) = 1$ and $s(k - \tau) = -1$ respectively for $2 - ary$ PAM, and $p_n(.)$ is the pdf of the noise vector $\mathbf{n}(k) = n(k) \ldots n(k - n + 1)$. The minimum error probability decision is defined by:

$$\hat{s}(k - \tau) = sgn(f(\mathbf{y}(k)) = sgn(\eta_{+1}(k) - \eta_{-1}(k))$$

where sgn is the signum function and $f(.)$ is called the optimal Bayesian decision function. For the case of a Gaussian noise distribution, the following explicit expression results:

$$\hat{s}(k - \tau) = sgn\left(\sum_{i=1}^{r_s^+} \alpha e^{(-\|\mathbf{y}(k) - \mathbf{y}_i^+\|^2/2\sigma_n^2)} - \sum_{j=1}^{r_s^-} \alpha e^{(-\|\mathbf{y}(k) - \mathbf{y}_j^-\|^2/2\sigma_n^2)}\right)$$

where α is $p_i(2\pi\sigma_n^2)^{-m/2}$ multiplied by an arbitrary positive constant, and the first sum is over $\mathbf{y}_i^+ \sum Y_{m,\tau}^+$, the second sum is over $\mathbf{y}_i^- \sum Y_{m,\tau}^-$; σ_n^2 is the noise

variance and $||.||$ denotes the Euclidian norm. The optimal equalizer solution can be seen to clearly depend on the noise distribution as well as the desired channel states. Under the previous assumptions on the digital data symbols $s(k)$, all the coefficients α are equal, and multiplying $f(\mathbf{y}(k))$ by a positive constant does not alter the optimality and all the α can therefore be set to 1. Note however, that if the assumption of equiprobable symbols is violated, each desired state $\mathbf{y}_i^+ \sum Y_{m,\tau}^+$ or $\mathbf{y}_i^- \sum Y_{m,\tau}^-$ can have different a priori probability of appearance p_i, which specifies the corresponding α.

The set of points $\mathbf{y}$ that satisfy:

$$f(\mathbf{y}) = 0$$

is referred to as the optimal decision boundary, which partitions the m-dimensional observation space into two decision regions corresponding to the equalizer decisions $\hat{s}(k-\tau) = 1$ and $\hat{s}(k-\tau) = -1$. In general, for $M-ary$ PAM baseband systems, the Bayesian decision procedure partitions the m-dimensional observation space into M decision regions. Because the optimal Bayesian decision function $f(.)$ defined above is non-linear, the optimal decision boundary is a hypersurface in the observation space and can be highly non-linear [188]. The boundary of any linear equalizer such as the LTE is only a hyperplane in the observation space, and hence it can be concluded that linear equalizer structures are inherently sub-optimal and a considerable performance gap will always exist between them and the optimal equalizer. However, it should be noted that although the MAP, also known as the Bayesian TE [194], offers significant performance improvement over the LTE it does so at the expense of a considerable increase in the computational complexity [172] [194] [164]. The dominant factor is that r_s, the number of combinations of the channel input sequence, can be quite large for practical channels. This fact has motivated the investigation of reduced complexity non-linear equalizer structures capable of realizing highly non-linear decision boundaries. Recently a Bayesian solution has been derived for the DFE structure [195] and shown to offer improved equalization performance compared to the Bayesian-TE

with a reduced computational requirement. The adaptive Bayesian DFE provides the optimal solution for the symbol decision DFE structure. However, both the adaptive Bayesian TE and Bayesian DFE structures assume perfect a priori knowledge of the communications channels, since the realization of the Bayesian equalization performance depends on knowing the channel states, and hence a performance degradation will result if an accurate channel estimate is not available. Two adaptive schemes are currently available to obtain the vector states [193] [195] which will further add to the required computational load. The first method estimates the channel model explicitly based on, for example the LMS algorithm and uses the channel estimate to calculate the channel states. The second approach estimates the channel states directly based on a clustering algorithm. The first method requires a shorter training period whilst the second scheme offers greater immunity to non-linear channel distortion. Note that in general, both the Bayesian TE and Bayesian DFE cannot achieve the theoretical best performance bound set by the adaptive MLVA since they are only symbol-decision equalizers. However, in a new study Chen, McLaughlin and Grant [198] have shown that for highly non-stationary channels such as multi-path mobile radio fading channels, the adaptive Bayesian DFE is actually superior to the adaptive MLVA.

Several other reduced complexity indirect modeling non-linear TE and DFE structures have been proposed to approximate the optimal Bayesian decision function $f(.)$, a good overview of which can be found in [185],[164] and [129].

Note that the optimal Bayesian TE solution was derived above for the linear channel equalization case in the presence of AWGN. For the general case of equalization of finite non-linear channels in the presence of coloured additive noise, the optimal Bayesian solution was investigated by Chen and Grant *et al* [184]. They showed by the use of simple examples, that non-linearities and coloured noise only change the location of the classification (or decision) boundary, and do not fundamentally alter the nature of the equalization problem. The attraction of

the neural network based equalizers is their ability to adaptively form the *general* Bayesian solution for the symbol-decision structure and therefore to provide significant performance gains over the conventional linear filter approach, as will be illustrated through simulation results in chapter 6.

Some of the recently proposed more popular neural network based non-linear equalizers including the MLP [172], the VNN based Adaptive Polynomial-Perceptron (APP) [183], the RBF [188] [194] and the FLNN based equalizers [163] [164] will now be reviewed.

The Multi-Layered Perceptron (MLP) based Adaptive Non-linear Equalizer

A neural network may simply be considered as a non-linear mapping between input and output. Gibson *et al* [172] proposed a non-linear TE structure based on the conventional MLP and demonstrated its superior performance over the LTE. The equalizer uses a layered feedforward structure with input, output and hidden layers. The hidden layers provide the capability by use of the non-linear sigmoidal function, to create intricately curved partitioning of the input signal space to create highly non-linear decision regions. Theoretically, a MLP equalizer of sufficient size (even one hidden layer with sufficient number of nodes) [33] can realise the optimal Bayesian decision function $f(.)$. To date, the MLP based equalizer has been employed for the equalization of both linear and non-linear channel models with a good degree of success [184].

There are however many practical difficulties associated with this structure that require further investigation. The selection of the architecture and parameter values for the MLP is mainly by trial and error which is an extremely time consuming process. The training of the MLP equalizer is based on a gradient descent algorithm known as the Back Propagation (BP) algorithm [75] which is discussed in Appendix A.3, and learning times are typically very long because of the highly non-linear in the parameters MLP structure. Furthermore, convergence to which local minimum of the MSE surface depends upon the initial choice

of the equalizer parameters and a theoretical analysis of the training algorithm is very difficult. Although the use of recursive Gauss-Newton algorithms proposed in [181] [171] can be applied to improve the convergence properties of the MLP equalizer, these algorithms require significantly more computations at each recursion, and may have difficulties in meeting the real-time requirements of high speed data transmission where adaptive equalization is mainly needed [71].

The Volterra Neural Network (VNN) based Adaptive Non-linear Equalizer

An alternative non-linear adaptive TE structure was proposed by Chen, Gibson and Cowan [183] based on the classical Volterra Neural Network (VNN), which is discussed in Appendix A.5.1. The output of the VNN based equalizer was related to its inputs through a Truncated Volterra Series (TVS) expansion as follows [164]:

$$\begin{aligned} y(k) &= w_0 + \sum_{i_0=1}^{m-1} w_1(i_1)x(k-i_1) + \sum_{i_0=1}^{m-1}\sum_{i_2=0}^{m-1} w_2(i_1,i_2)x(k-i_1)x(k-i_2) + \dots \\ &\quad + \sum_{i_1=0}^{m-1}\sum_{i_2=0}^{m-1}\dots\sum_{i_q=0}^{m-1} w_q(i_1,i_2,\dots,i_q)x(k-i_1)x(k-i_2)\dots x(k-i_q) \end{aligned}$$

The above system can be represented as a (m,q)VNN (or TVS) equalizer, where m denotes the feedforward order of the equalizer, and q represents the degree of expansion of the Volterra Series expansion.

A modified two-layer polynomial-perceptron structure employing the TVS expansion of the inputs at the hidden layer and a non-linear sigmoid activation function at the output layer was proposed in [183] and shown to offer superior performance to the conventional LTE at the expense of increased computational complexity. The complexity of the Adaptive Polynomial-Perceptron (APP) equalizer was determined by two structure parameters, namely the equalizer feedforward order m and the polynomial degree of expansion q of the equalizer input symbols as shown in the TVS expansion above. However, the APP equalizer structure suffers a drawback of exponentially increasing filter dimensions [194],

which makes its use impractical for realistic channels of even moderate orders.

The Radial Basis Function (RBF) based Adaptive Non-linear Equalizer

The two-layer linear-in-the-parameters Radial Basis Function neural network was proposed as a non-linear adaptive equalizer by Chen, Gibson and Grant [188]. As discussed in Appendix A.4, a key feature of the RBF networks is their ability to provide non-linear approximation ability and yet possess a linear in the parameters structure. In the hidden layer, the Euclidean distance from the current equalizer input vector to a number of centres is calculated. The Euclidean distances are then passed through a non-linearity which is usually a Gaussian function of the form

$$f(\mathbf{y}) = exp(-\mathbf{y}^2/\sigma^2)$$

where σ is real constant. As discussed in Appendix A.4, other choices for the non-linearity $f(.)$ can typically include the thin-plate-spline function, the multi-quadratic and inverse multi-quadratic functions. In the output layer, the outputs from the non-linear functions are combined by a weighted linear combiner to form the equalizer output.

By appropriate choice of a sufficiently large number of centres, the RBF equalizer has been shown to approximate both the adaptive optimal symbol decision Bayesian TE and Bayesian DFE structures [194]. The gradient descent LMS algorithm can readily be used to train the RBF equalizer and the training will be guaranteed to converge to the single global minimum of the MSE surface [188]. However, the ability of the adaptive RBF equalizer to provide the optimal equalization performance is crucially dependent upon whether its centers can be positioned at the desired channel states [194]. A supervised clustering algorithm has been proposed [194] for updating the position of the basis functions in the input hidden layer which in turn, increases the number of computations required for efficient training of the RBF equalizer in order to realize the optimal performance. The computational cost required by the RBF for fully implementing the optimal Bayesian TE is also often unacceptably high [16], and the RBF equalizer

has to be operated with fewer centres and consequently sub-optimal performance. Furthermore, complete specification of the RBF equalizer namely, the number of basis functions (centres) and their spread required, for optimal performance requires a priori knowledge of the additive noise variance and the channel order (which determines the number of desired channel states), the estimation of which will further add to the computational load.

The Functional-Link Neural Network (FLNN) based Adaptive Non-linear Equalizer

The conventional non-linear-in-the-parameters Functional-Link Neural Network (FLNN) discussed in Appendix A.5.2 was proposed as a non-linear adaptive TE structure by Gan, Soraghan and Durrani [163] [164] and shown to offer superior performance (in terms of equalizer convergence speed and Bit Error Rate characteristics) compared to both the MLP and the APP based equalizers, for equalizing communications channels in the presence of ISI and additive white Gaussian noise. The structure of the Feedforward Functional-Link Equalizer (FFLE) consists of two layers:

- **Functional-Link Expander Input Layer:** It performs a non-linear transformation which maps the input space onto a new larger dimensional output space. Actual choice of functions was discussed in [164]. The purpose of these functions is to extract certain useful features of the input data which render easier separation of the input classes.

- **Output Layer comprising a Linear Combiner and a Sigmoidal Thresholder:** At this stage the weighted values of the enhanced input functions $F(k)$ are linearly combined before being fed into a sigmoidal unit. The weights $w_i(k)$ where $i = 0, 1, \ldots, N$ are updated using the Delta Rule (DR) algorithm.

The FFLE has been shown to be capable of equalizing both linear and non-linear channel models in the presence of ISI additive noise [164], on account of its

ability to form highly non-linear decision regions. However, the Feedforward Functional Link Equalizer (FFLE) requires a large number of functional-link terms for near-optimal equalization performance, and like the MLP and VNN based adaptive equalizers lacks a general design (specification) algorithm.

In chapter 6 of this thesis, a new procedure for the design of FFLE of an arbitrary order is presented, which is shown to give highly useful insights into its computational requirements. Two novel approaches for the DFE structures based on the FFLE are also introduced along with their design strategies, and their performance compared to the FFLE and other neural network based adaptive feedforward (TE) and DFE structures.

Other ANN based equalizers include the Self-Organizing Map (SOM) based equalizer [54], which operates as an efficient blind equalizer capable of cancelling both the linear and non-linear channel distortions. However, this approach exhibits poor performance in severely amplitude distorted channels. Another ANN based equalizer structure has been recently reported based on the Recurrent Neural Network (RNN) [55], and shown to consistently outperform the LTE and MLP based equalizers in the equalization of both linear and non-linear dispersive channels. However, its performance was not compared with the optimal Bayesian TE or MLVA structures, and employed the computationally expensive Real Time Recurrent Learning (RTRL) algorithm for its weight updates.

A classification of various trained adaptation equalizers described in this chapter, in terms of types, structures and algorithms is given in Figure 3.**2**. This chart extends that reported by Proakis [155] and Gan [164].

The next section introduces the second digital communications application considered in this thesis namely, the problem of co-channel interference suppression in digital communications systems. Following a formulation of the problem, a review of the various equalizer solutions reported to date is given.

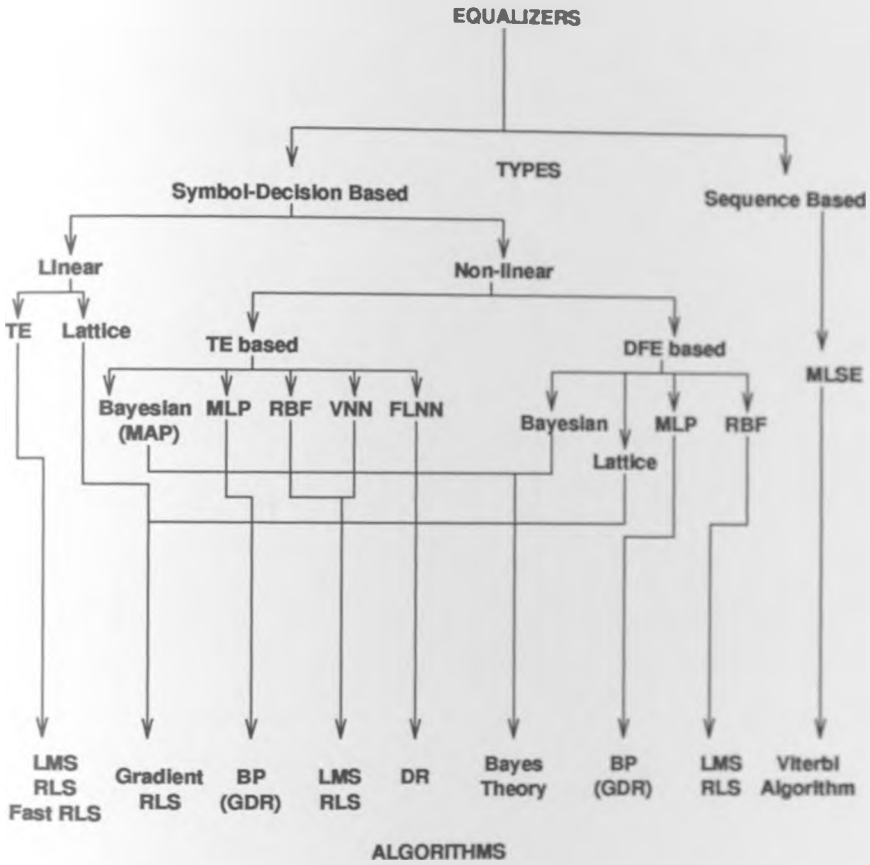

Figure 3.2: Classification of various Equalizer Types, and Algorithms

3.3 Co-channel Interference Supression in Digital Communications Systems

Adaptive equalization is known be an important technique for combating distortion and inter-symbol interference in communications channels. However, many communications systems are also impaired by what is known as co-channel interference. Many digital communications systems such as cellular radio and dual polarised micro-wave radio, for example, employ frequency reusage and often exhibit performance limitation due to co-channel interference [167]. Frequency reuse is referred to the employment of radio channels on the same same carrier frequency to cover different areas or cells situated sufficiently apart from one another, and allows cellular radio systems to handle far more simultaneous calls than the total number of allocated channel frequencies. Signals from co-channel cells (i.e. cells of the same channel frequency) will however interfere with each other. The degradation in quality due to co-channel interference is often more severe than that caused by additive noise or ISI [170]. In land mobile radio systems for instance, geographical frequency reuse is used to provide a system with a high traffic carrying capacity, using a limited amount of the radio spectrum. The extent to which frequencies can be reused is limited by the tolerance of the receiver to co-channel interference. The traffic capacity of the system is directly linked to the extent of frequency re-use, and consequently to a receiver's ability to combat co-channel interference. The optimal solution to the problem assumes perfect knowledge of the mobile environment [131]. As the mobile environment is unknown and can change, adaptive equalizers are therefore required in these communications systems in order to achieve an acceptable error-rate performance [131] [190]. Other examples of co-channel interference can be found in [167].

3.3.1 Problem Statement

The discrete time model of the data transmission system considered in this thesis is shown in Figure 3.3. In this model, $H_0(z)$ is the dispersive channel transfer

function and $H_1(z) \ldots H_\gamma(z)$ represent the interfering co-channels. All channels are modelled by Finite Impulse Response (FIR) filters as:

$$H_i(z) = \sum_{j=0}^{l_i} h_{ij} z^{-j} \quad i = 0, 1, \ldots, \gamma$$

In Figure 3.3, $s_0(k)$ represents the transmitted data (which is known during the equalizer training mode) and $s_i(k)$, i= $1, \ldots, \gamma$ are the unknown interfering data sequences. All s_i i= $0, \ldots, \gamma$ are assumed to be equiprobable and bipolar independent identically distributed (iid) binary sequences (i.e. taking values $+1$ or -1). The dispersive channel output $\hat{y}(k)$ and the output from the co-channels $c(k)$ are corrupted by Additive White Gaussian Noise (AWGN) $n(k)$ of zero mean and variance σ_n^2. All $s_i(k)$ i= $0, 1, \ldots, \gamma$ are assumed to be uncorrelated with $n(k)$. The overall channel observation can thus be written as:

$$y(k) = \hat{y}(k) + c(k) + n(k)$$

where

$$\hat{y}(k) = \sum_{j=0}^{l_0} h_{0j} s_0(k-j)$$

with l_0 being the order of the distorting channel; and

$$c(k) = \sum_{i=1}^{\gamma} \sum_{j=0}^{l_i} h_{ij} s_i(k-j)$$

where l_i, $(i = 1, \ldots, \gamma)$ is the order of the $i-th$ interfering co-channel.

If $E[\hat{y}^2(k)] = \sigma_s^2$ (where $E[.]$ is the expectation operator) and $E[c^2(k)] = \sigma_c^2$; then the following expressions can be defined:

The Signal to Noise Ratio (SNR) is given by:

$$SNR = \frac{\sigma_s^2}{\sigma_n^2}$$

The Signal to Interference Ratio (SIR) is defined as:

$$SIR = \frac{\sigma_s^2}{\sigma_c^2}$$

And the Signal to Interference plus Noise Ratio (SINR) is given by:

$$SINR = \frac{\sigma_s^2}{(\sigma_c^2 + \sigma_n^2)}$$

The task of any indirect-modeling co-channel equalizer is: given the overall channel observation $y(k)$, estimate the transmitted data $s_0(k)$. The symbol decision equalizer at any sample instant k processes the m most recent channel observations, and makes a decision $\hat{s}_0(k-\tau)$ regarding the symbol transmitted at $k-\tau$, where the integers m and τ are referred to as the equalizer order and delay respectively. During the training period of most adaptive equalization systems (including cellular mobile radio), the reference desired signal $s_0(k-\tau)$ which is to be re-constructed, is available, whereas the other interfering signals $s_i(k), i = 1, \ldots, \gamma$ are not known.

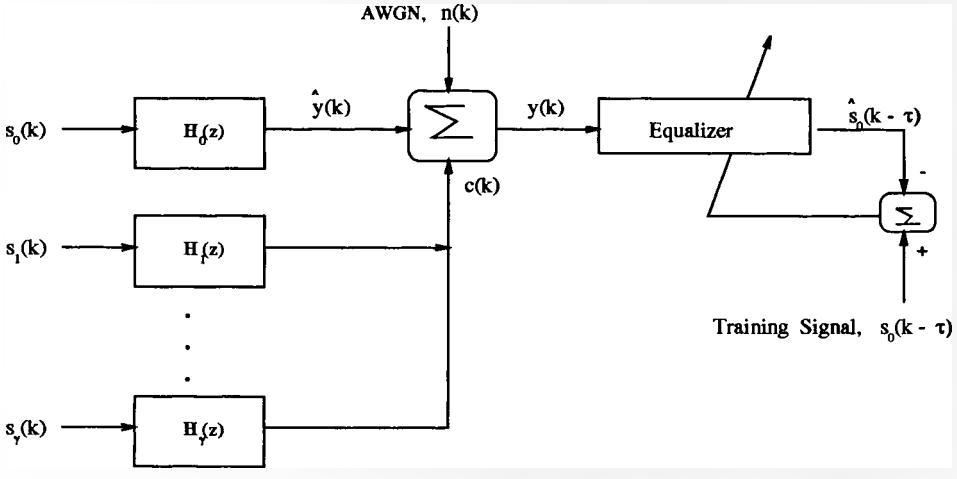

Figure 3.3: Data Transmission System involving co-channel interference

3.3.2 Existing Algorithms

For equalization of channels in the presence of ISI, AWGN and co-channel interference, the following developments have been recently reported. In [29] Wales presented a reduced complexity receiver structure for improving the tolerance of a receiver to the presence of like modulated co-channel interference. Performance

improvements were demonstrated over a conventional receiver for static channels using an analytical technique, and for Rayleigh fading channels using Monte Carlo simulation techniques. However, for Rayleigh fading channels, the performance of the receiver structure was seen to degrade at high signal to interference ratios [29]. In [169] an equalizer structure for combating co-channel interference was described that relied upon the stronger of the wanted and interfering signals to capture a conventional BPSK demodulator. The detected signal was then re-modulated and cancelled from the received signal to allow detection of the weaker signal. This approach has a low complexity but performance improvements are strictly limited to situations where the interfering signal level exceeds that of the wanted signal and the technique has yet to be extended to time dispersive channels. In [170] Giridhar, Shynk and Gooch presented non-linear co-channel demodulation techniques based on the MLSE and the Maximum Aposteriori Probability (MAP) symbol detection for joint recovery of both the desired and the interfering signals. However the algorithms are computationally expensive and assume perfect a priori knowledge of both the primary and interfering channels. Also they provide optimal performance only for the case of comparable energies of the desired and interfering signals. In [190] Chen and Mulgrew employed a non-linear adaptive equalizer based on the Radial Basis Function (RBF) neural network to overcome co-channel interference. A complex two-stage learning strategy was employed to model the effects of the channel ISI and co-channel interference and thus provide the optimal Bayesian equalization solution. Performance improvements were reported over the LTE for moderate to high signal to interference ratios. However, even for small orders of the primary and interfering channels, computing the optimal equalizer performance using the feedforward RBF is extremely costly and impractical as a prohibitively large number of centres are needed to represent an equal number of the noise-free observation states. Determination of the number of noise free observation states (which set the required number of RBF centers for optimal performance) also requires apriori knowledge of the orders of both the distorting channel and the unknown interfering co-channels.

In chapter 6 of this thesis, we investigate the use of the conventional FFLE and new DFE structures in overcoming co-channel interference in digital communications systems.

3.4 Conclusions

This chapter has reviewed two important digital communications applications namely, the problem of linear and non-linear communications channel equalization in the presence of ISI and additive noise; and the task of overcoming co-channel interference in digital communications systems. Various approaches reported to date to solving the above problems have been discussed with particular emphasis on the ANN based adaptive non-linear equalizer solutions. It has been shown through the use of the optimal Bayesian TE solution that the equalization of communications channels in the presence of ISI and additive noise, is inherently a non-linear problem and the optimal equalizer has to possess non-linear decision making capabilities. For the additional case of co-channel interference supression, the optimal Bayesian solutions derived in [190] [196] have also concluded the need of non-linear adaptive equalizer based solutions. These facts have in recent years, motivated the design of ANN based equalizers.

In the next chapter, a new Feedforward ANN structure is proposed for efficient modeling of non-linear dynamical systems.

Chapter 4

New Feedforward Functionally Expanded Neural Network For Non-Linear Dynamical System Modeling Applications

4.1 Introduction

Feedforward neural networks capable of learning complex input-output mappings were discussed in chapter 2. It was shown how, given a set of inputs and desired outputs, an adequately chosen neural network can capture the underlying dynamics of the non-linear system (that generated the data set) through an appropriate learning mechanism.

In section 4.2 of this chapter, a new two-layer linear-in-the-parameters Feedforward Functionally Expanded Neural Network (FFENN) structure [175] is presented for efficient modeling of chaotic and equation-error type non-linear dynamical systems. New non-linear basis functions are proposed for the FFENN's single hidden layer and the rationale behind their choice is also discussed. The

new FFENN structure employing the proposed terms in its hidden layer is essentially a hybrid neural network incorporating to a variable extent, the combined modeling capabilities of the conventional MLP, RBF and VNN structures. A general design strategy for specifying the functional expansion model at the hidden layer of the FFENN for an arbitrary number of network inputs is also presented. The error-surface of the FFENN is shown to be uni-modal allowing high speed single-run learning. A least squares based learning algorithm is derived for the FFENN thereby alleviating the non-linear learning difficulties associated with conventional multi-layered feedforward ANNs. A new pruning technique based on an iterative pruning re-training strategy coupled with correlation and chi-squared statistic based model validity tests [109] is also devised in order to optimise the size of the FFENN structures for non-linear dynamical system modeling applications.

In section 4.3, various case studies using simulated chaotic, NARX, NAR and real world noisy, non-linear time series processes are used to compare the modeling and prediction performance of the FFENN with other recently reported feedforward and recurrent neural network based predictor models [42]. The respective contributions of the various proposed basis functions within the FFENN hidden layer are also illustrated in the various case studies.

4.2 The new Feedforward Functionally Expanded Neural Network FFENN structure

The complete two-layer FFENN is illustrated in Figure 4.1. It comprises a functional expander within its single hidden layer and an output layer. The functional expander performs a non-linear transformation which maps the input space onto a new larger dimensional non-linear hidden space. The choice of basis functions employed in the functional expander discussed in the next sections 4.2.1, 4.2.2 and 4.2.3.

The output layer of the FFENN shown in Figure 4.1 comprises a set of linear

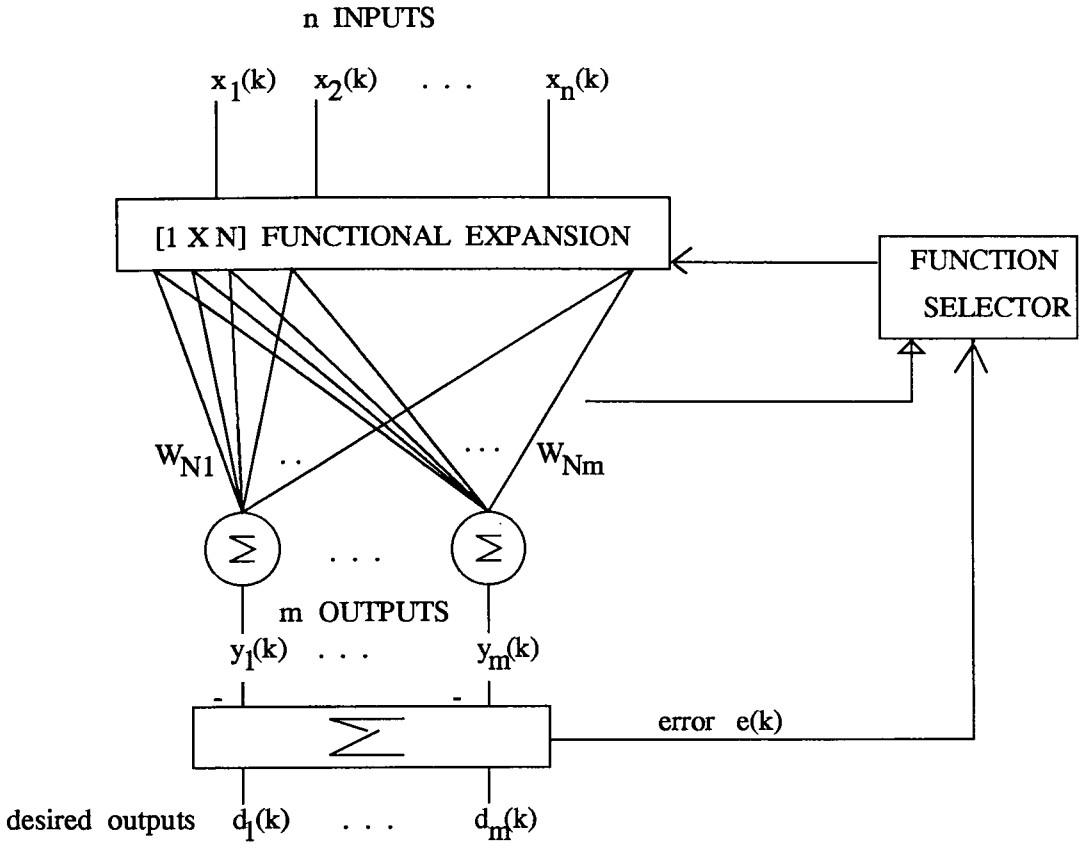

Figure 4.1: The Feedforward Functionally Expanded Neural Network

combiners. Thus the overall FFENN structure can be seen to possess non-linear approximation ability by virtue of the non-linear functional expander employed at its hidden layer and yet, learning of its output layer weights is a linear problem. It is this latter characteristic of the FFENN that provides the motivation for exploiting its use in complex real world non-linear system modeling applications. Also illustrated in Figure 4.1 is a function selection (pruning) process which will be discussed in section 4.2.5.

In the next section 4.2.1, a formal scheme is presented which enables the design of a functional expander for any number of FFENN inputs. New functional terms are proposed for the FFENN hidden layer's functional expansion model in order to enhance its non-linear approximation ability, and the rationale behind their choice is discussed in sections 4.2.2 and 4.2.3.

4.2.1 Design Strategy for the FFENN

For any number n of FFENN inputs all normalized to within the range $(+1, -1)$, expand the inputs using the following functional expansion model $F(k)$:

$F(k)$ = sum of the following (linear and non-linear) N basis functions:

1. zero-order (dc) term (resulting in 1 term).

2. original input terms (resulting in n terms). These terms will enable modeling of linear systems.

3. sine expansion of the n inputs, comprising $\sin(x_i), \sin(2x_i)$ and $\sin(3x_i)$ terms for $i = 1, \ldots, n$ (resulting in a total of $3n$ terms).

4. cosine expansion of the n inputs comprising $\cos(x_i), \cos(2x_i)$ and $\cos(3x_i)$ terms for $i = 1, \ldots, n$) (resulting in a total of $3n$ terms)

5. product of each input with the sine and cosine activation functions of *other* inputs comprising $x_i \sin(x_j)$ and $x_i \cos(x_j)$ terms (for $i \neq j$ $i, j = 1, \ldots, n$) resulting in a total of $2n(n-1)$ terms.

6. Outer-product expansion of the n inputs resulting in $((P_2^n + P_3^n +, \ldots, + P_{n-1}^n) +$ 1) terms for n greater than two inputs, where $P_j^n = \frac{n!}{(n-j)!j!}$ where ! denotes factorial. Note that for $n = 2$ inputs the outer-product expansion will result in 1 term, and for $n = 1$ there will be no outer product terms).

The design strategy for the FFENN described above will now be illustrated by examples.

For a single input FFENN(1,N), the expansion model $F(k)$ will comprise the following $N = 8$ terms (assuming x(k-1) represents the single time series sample used as input):

$$\begin{aligned} F(k) \;=\; & x(k-1), \sin(x(k-1)), \sin(2x(k-1)), \sin(3x(k-1)), \cos(x(k-1)), \\ & \cos(2x(k-1)), \cos(3x(k-1)), 1 \end{aligned} \tag{4.1}$$

For a 2-input FFENN(2,N), the $N = 20$ term expansion model $F(k)$ is devised as follows (assuming x(k-1) and x(k-2) represent the two time series samples used as inputs):

$$\begin{aligned} F(k) \;=\; & 1, x(k-1), x(k-2), \sin(x(k-1)), \sin(2x(k-1)), \sin(3x(k-1)), \\ & \sin(x(k-2)), \sin(2x(k-2)), \sin(3x(k-2)), \cos(x(k-1)), \\ & \cos(2x(k-1)), \cos(3x(k-1)), \cos(x(k-2)), \cos(2x(k-2)), \\ & \cos(3x(k-2)), x(k-1)\sin(x(k-2)), x(k-1)\cos(x(k-2)), \\ & x(k-2)\sin(x(k-1)), x(k-2)\cos(x(k-1)), x(k-1)x(k-2) \end{aligned} \tag{4.2}$$

For a 3-input FFENN(3,N), $F(k)$ will now comprise the following $N = 38$ terms:

$$\begin{aligned} F(k) \;=\; & 1, x(k-1), x(k-2), x(k-3), \sin(x(k-1)), \sin(2x(k-1)), \\ & \sin(3x(k-1)), \sin(x(k-2)), \sin(2x(k-2)), \sin(3x(k-2)), \\ & \sin(x(k-3)), \sin(2x(k-3)), \sin(3x(k-3)), \cos(x(k-1)), \\ & \cos(2x(k-1)), \cos(3x(k-1)), \cos(x(k-2)), \cos(2x(k-2)), \\ & \cos(3x(k-2)), \cos(x(k-3)), \cos(2x(k-3)), \cos(3x(k-3)), \\ & x(k-1)\sin(x(k-2), x(k-1)\sin(k-3), x(k-1)\cos(x(k-2), \\ & x(k-1)\cos(x(k-3)), x(k-2)\sin(x(k-1)), x(k-2)\sin(x(k-3)), \\ & x(k-2)\cos(x(k-1)), x(k-2)\cos(x(k-3))x(k-3)\sin(x(k-1)), \\ & x(k-3)\sin(x(k-2)), x(k-3)\cos(x(k-1)), x(k-3)\cos(x(k-2)), \\ & x(k-1)x(k-2), x(k-1)x(k-3), x(k-2)x(k-3), \\ & x(k-1)x(k-2)x(k-3) \end{aligned} \tag{4.3}$$

For a 4-input FFENN(4,N), $F(k)$ comprises $N = 64$ terms illustrated below:

$$\begin{aligned} F(k) \;=\; & [1, x(k-i),\ \sin(jx(k-i)),\ \cos(jx(k-i)),\ i = 1,\ldots,4\ j = 1,\ldots,3] \\ & [x(k-i)\sin(x(k-j)),\ x(k-i)\cos(x(k-j))\ i \neq j\ i,j = 1,\ldots,4] \\ & x(k-1)x(k-2),\ x(k-1)x(k-3),\ x(k-1)x(k-4),\ x(k-2)x(k-3) \end{aligned}$$

n	N
1	8
2	20
3	38
4	64
5	102
6	160
7	254
8	416
9	710
10	1264
11	2334
12	4432
n	$1 + 2n^2 + 4n + \sum_{i=1}^{n} P_i^n$

Table 4.1: General Design Strategy for (n, N)FFENN

$$x(k-2)x(k-4),\ x(k-3)x(k-4),\ x(k-1)x(k-2)x(k-3),$$
$$x(k-1)x(k-2)x(k-4),\ x(k-1)x(k-3)x(k-4),$$
$$x(k-2)x(k-3)x(k-4),\ x(k-1)x(k-2)x(k-3)x(k-4) \qquad (4.4)$$

Note that in the above examples, past delayed values of the same input $(x(k) = x(k-1), \ldots, x(k-n))$ have been used as explicit inputs to the FFENN. The above expansion models also apply to the case of several distinct inputs $x_1(k), x_2(k), \ldots, x_n(k)$ being used as FFENN inputs.

If the above expansion models are scaled to larger input dimensions, then Table 4.1 can be constructed relating the number of FFENN inputs n to the number of terms N in the functional expansion model $F(k)$. It is interesting to note that the RBF network with fixed non-linear hidden layer basis functions or centres (and widths) can be regarded as a FFENN. The conventional Functional Link Neural Network (FLNN) discussed in the Appendix A.5.2 also resembles the above FFENN in respect of employing a functional expansion of the inputs. However, the FLNN possesses a non-linear in the parameters structure (as its output layer is essentially a perceptron requiring the non-linear Delta Rule (DR)

for its weight updates); and most importantly the nature of its functionally expanded (trigonometric) input terms also differ. The functional expansion model employed in the feedforward FENN is unique as specified in the design strategy above and discussed in the next section. The VNN which employs a purely polynomial expansion of its inputs can also be considered to be a special case of the FFENN, in which the number of polynomial expansion terms grow exponentially with increasing input dimensions. As can be seen from Table 4.1 however, an increase in the number of the inputs does not lead to an exponentional increase in the size of the FFENN hidden layer (expansion model), unlike the VNN and RBF structures. However, it can also be seen from Table 4.1 that for inputs greater than 10, the computational requirements of the FFENN hidden layer become excessive and practically difficult to realize, unless an effective pruning strategy is employed. The pruning of the FFENN is discussed in section 4.2.5.

The non-linear approximation ability of the FFENN resulting from use of the above expansion models will be illustrated through various simulated and real-world non-linear time series modeling applications in section 4.3 of this chapter. In the next sections 4.2.2 and 4.2.3, the rationale behind the choice of the proposed non-linear basis functions in the design strategy above is discussed in terms of the modeling capabilities possessed by their underlying structures.

4.2.2 Discussion on Choice of Non-linear Basis Functions

For the input x normalized to within the range $(+1, -1)$, plots of the cosine activation functions $\cos(x)$, $\cos(2x)$ and $\cos(3x)$ are shown (superimposed) in Figure 4.2. Plots of the sine activation functions $\sin(x)$, $\sin(2x)$ and $\sin(3x)$ are illustrated in Figure 4.3. As can be seen from Figure 4.2, the cosine basis functions in fact emulate the Gaussian bell shaped functions similar to those employed in the RBF network [187] (see Appendix A.4); whereas the sine basis functions in Figure 4.3 simulate the squashing sigmoidal shaped activation functions similar to those employed in the MLP network.

The other proposed non-linear basis functions comprising the outer-product

expansion of the FFENN inputs can in fact be considered to be a polynomial expansion of the inputs without the $n-th$ power of the inputs (see Appendix A.5.1).

And finally, the other non-linear basis functions proposed for the FFENN hidden layer's functional expansion model include terms comprising the product of each FFENN input with the sine and cosine activation functions of *other* FFENN inputs. Comparative plots of $x\sin(x)$ and $x\cos(x)$ are illustrated in Figures 4.4 and 4.5 respectively. As can be seen from Figure 4.4, the $x\sin(x)$ activation function emulates a multi-quadratic type non-linearity which is also commonly employed in the RBF network (see Appendix A.4); whereas the $x\cos(x)$ basis function illustrated in Figure 4.5 approximates a squashing type sigmoidal activation function commonly employed in the MLP network. These higher order cross-terms in the FFENN expansion model can be considered to bridge the approximation abilities provided by the stand-alone polynomial type outer-product terms and the orthonormal sine and cosine basis functions.

Hence the new FFENN structure employing the above proposed non-linear basis functions within its single hidden layer can be considered to be a hybrid neural network incorporating to a variable extent, the rich modeling capabilities of the MLP, RBF and VNN structures.

4.2.3 Approximation Ability of the FFENN

Recalling that polynomials can be used to approximate any non-linear continuous function $\phi(.) : R^n \rightarrow R^m$ to an arbitrary degree of accuracy, it is interesting to note that the sigmoidal activation function employed in each node of the conventional MLP (which is known be a universal approximator), can be expressed as an inverted polynomial series:

$$f(x) = (1 + e^{-x})^{-1} = \left(1 + \sum_{i=0}^{\infty} \frac{(-x)^i}{i!}\right)^{-1}$$

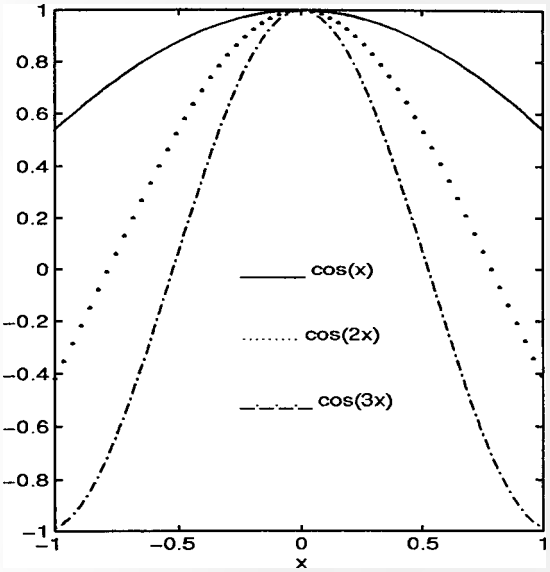

Figure 4.2: Plots of $\cos(x)$, $\cos(2x)$ and $\cos(3x)$

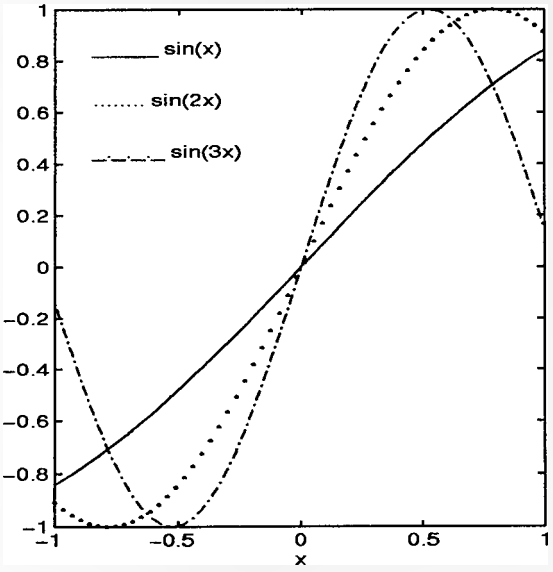

Figure 4.3: Plots of $\sin(x)$, $\sin(2x)$ and $\sin(3x)$

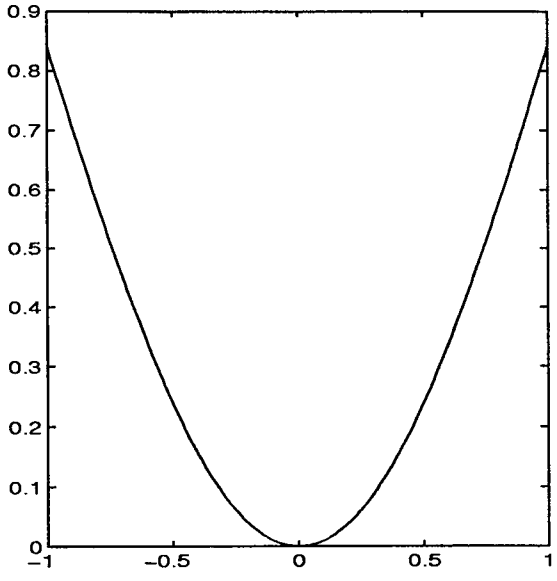

Figure 4.4: Plot of $x\sin(x)$

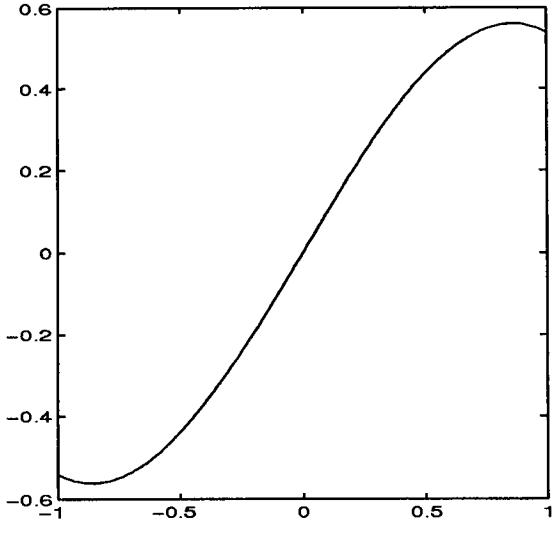

Figure 4.5: Plot of $x\cos(x)$

Thus it can be confidently said that if a sufficient number of such higher order non-linear mapping functions are employed in the FFENN hidden layer's functional expansion model set, such as the orthonormal basis functions that can be expressed as polynomial series: namely,
the sine activation functions

$$\sin(\alpha x_i) = (\alpha x_i) - (\alpha x_i)^3/3! + (\alpha x_i)^5/5! - , \ldots \text{ for } \alpha = 1,2,3 \text{ for } i = 1,\ldots,n$$

which comprise an infinite series of odd power expansion of the n inputs,
and the cosine activation functions

$$\cos(\alpha x_i) = 1 - (\alpha x_i)^2/2! + (\alpha x_i)^4/4! - , \ldots \text{ for } \alpha = 1,2,3 \text{ for } i = 1,\ldots,n$$

which comprise an infinite series of even power expansion of the n inputs;
and the higher-order odd cross-terms

$$x_i \sin(x_j) = x_i x_j - x_i(x_j^3/3!) + x_i(x_j^5/5!) - , \ldots \text{ for } i \neq j, \ i,j = 1,\ldots,n$$

which in fact comprise higher order cross-products between each FFENN input with an infinite series of odd power expansion of the other FFENN inputs;
and finally the higher-order even cross-terms

$$x_i \cos(x_j) = x_i - x_i(x_j^2/2!) + x_i(x_j^4/4!) - , \ldots \text{ for } i \neq j, \ i,j = 1,\ldots,n$$

which in fact comprise higher order cross-products between each FFENN input with an infinite series of even power expansion of the other FFENN inputs; then the FFENN can be confidently said to uniformly approximate $\phi(.)$ to within an arbitrary accuracy. Thus, employment of the above proposed basis functions in conjunction with the polynomial-subset outer-product (joint activation) terms can be seen to provide a very rich FFENN input functional expansion model, which may additionally (like the Volterra Neural Network) also provide highly useful insights into the physical composition of the underlying non-linear system dynamics.

It is important to note that the design procedure presented in section 4.2.1 above provides a useful starting point. Nevertheless, the functional expansion

model of the FFENN is extremely flexible in that, virtually *any* function of the input such as $tanh(.)$, $exp(.)$, *etc.* could also be employed. In practice some physical knowledge of the non-linear system to be identified can also be incorporated into the functional expander. On the other hand, if no a priori system knowledge is available and a more enhanced FFENN approximation to the underlying system is required than that provided by use of the above expansion models, then one can easily include for instance, additional higher order polynomial terms (from the Volterra Series expansion of the inputs) in the FFENN functional expansion model; as the Volterra Series model is well known to be able to approximate any non-linear continuous function $\phi(.)$ to within an arbitrary accuracy [63].

4.2.4 Derivation of the FFENN Learning Algorithm

The output weights for a *general* Multiple Input Multiple Output (MIMO) (n,N;m)FFENN (with n inputs, a N term functionally expanded input model and m outputs) are updated as follows:

Defining the hidden layer (with a [1xN] non-linear expansion model of the input) vector at time k as:

$$F(k) = [f_1(k) \dots f_N(k)]^T$$

And the weight vector of the ith output node as:

$$W_i(k) = [w_{1i}(k) \dots w_{Ni}(k)]^T \text{ for } i = 1, \dots, m \text{ outputs}$$

where the superscript T denotes vector (or matrix) transpose.

1. Compute the m FFENN outputs:

$$y_i(k) = F^T(k)W_i(k-1) \text{ for } i = 1, ..., m \text{ outputs} \tag{4.5}$$

where $F(k)$ are the functionally expanded input terms which transform the input space R^n of n inputs onto a new non-linear hidden (or intermediate) space of dimension R^N. The overall input-output mapping of the FFENN is thus $R^n \rightarrow R^N \rightarrow R^m$.

2. Compute (prediction) error for each output (where d_i is the i^{th} desired output):

$$\epsilon_i(k) = d_i(k) - y_i(k) \tag{4.6}$$

The mean squared error (MSE) is therefore:

$$E[\epsilon_i^2] = E(d_i(k)^2) - 2W_i^T(k-1)E(d_i(k)F(k)) + W_i^T(k-1)E(F(k)F^T(k))W_i(k-1) \tag{4.7}$$

where E is the statistical expectation operator, $E(F(k)F^T(k))$ is an $[N \times N]$ auto-correlation matrix of the functionally expanded input vector $F(k)$; and $E(d_i(k)F(k))$ is an N element cross-correlation vector between the desired signal $d_i(k)$ and $F(k)$. The minimum MSE is obtained by setting the weight derivative of the MSE cost function to zero as follows:

$$\frac{\partial E[\epsilon_i^2]}{\partial W_i(k-1)} = 0$$

which results in:

$$-2E(d_i(k)F(k)) + 2E(F(k)F^T(k))W_i(k-1) = 0$$

The optimum FFENN weight coefficients $W_{i_{opt}}(k-1)$, which minimize the MSE are thus the solution to the following set of N *extended* Wiener-Hopf equations:

$$E(F(k)F^T(k))W_{i_{opt}}(k-1) = E(d_i(k)F(k)) \quad \text{for } i = 1, \ldots, m \tag{4.8}$$

Assuming that $E(F(k)F^T(k))$ is non-singular, then the optimum FFENN weight vector will be unique given by:

$$W_{i_{opt}}(k-1) = E(F(k)F^T(k))^{-1}E(d_i(k)F(k)) \tag{4.9}$$

where the subscript (-1) denotes matrix inversion. The corresponding minimum MSE (MMSE) for the FFENN is thus obtained by directly substituting for $W_{i_{opt}}(k-1)$ into equation 4.7 above, resulting in:

$$MMSE = E(d^2(k)) - W_{i_{opt}}^T(k-1)E(d_i(k)F(k)) \tag{4.10}$$

which includes the best linear (Wiener) MMSE for the case of $F(k) = [x_1(k) \dots x_n(k)]$, that is, without a non-linear functional expansion of the inputs. The advantage of this particular FFENN structure is that linear adaptive filter theory can be readily applied for on-line adaptation.

Note that for the case of FFENN inputs $x_i(k)$, $i = 1, \dots, n$ and desired outputs $d_i(k)$, $i = 1, \dots, m$ both being wide-sense stationary ergodic signals, the correlation matrices, $E(F(k)F^T(k))$ and $E(d_i(k)F(k))$ are time-invariant, and hence $W_{i_{opt}}(k-1)$ will also be time-invariant. However, as can be seen from equation 4.9, computation of $W_{i_{opt}}(k-1)$ requires matrix inversion and is therefore computationally expensive, especially if the data are time-varying and a new optimum weight vector must be calculated at each iteration. Note also that the quadratic form of the mean squared error expression (equation 4.7 above) with respect to the FFENN weights, shows that there will be no local minima in the error surface. In practice, fast and certain convergence may be obtained by use of the following *recursive* weight update, which overcomes the need for computation of matrix inversion:

3. Update FFENN weights for each $i = 1, \dots, m$ outputs using:

Exponentially Weighted Recursive Least Squares (RLS) Update:

The exponentially weighted Recursive Least squares RLS (Kalman) estimator can be derived by minimising the following cost function with respect to the weight coefficient vector $W_i(k)$

$$J_i = \sum_{t=1}^{k} \lambda^{k-t} \epsilon_i(t)^2$$

where the ensemble averages have now been replaced by time averages, and λ represents a weighting or forgetting factor $\in (0, 1)$. Thus exponential weighting into past data has been introduced in order to allow modeling of time-variant systems. The optimum weight vector for each output is now readily given by the extended Wiener solution:

$$W_{i_{opt}}(k) = R^{-1}(k)\theta_i(k) \tag{4.11}$$

where the auto-correlation matrix $R(k)$ is now defined as:

$$R(k) = \sum_{t=1}^{k} \lambda^{k-t} F(t) F^T(t) \tag{4.12}$$

and the cross-correlation vector $\theta_i(k)$ is:

$$\theta_i(k) = \sum_{t=1}^{k} \lambda^{k-t} d_i(t) F(t) \tag{4.13}$$

Note that $R(k)$ in equation 4.12 above, can also be written recursively as:

$$R(k) = \lambda R(k-1) + F(k) F^T(k) \tag{4.14}$$

The inverse of $R(k)$ can now be estimated recursively using the matrix inversion lemma [78] as follows (assuming $P(k)$ represents $R^{-1}(k)$):

$$P(k) = \frac{1}{\lambda}[P(k-1) - \frac{P(k-1)F(k)F^T(k)P(k-1)}{\lambda + F^T(k)P(k-1)F(k)}] \tag{4.15}$$

The cross-correlation vector $\theta_i(k)$ in equation 4.13 can also be written recursively as:

$$\theta_i(k) = \lambda \theta_i(k-1) + d_i(k) F(k) \tag{4.16}$$

Substituting for $\theta_i(k)$ from equation 4.16 above into equation 4.11, yields:

$$W_{i_{opt}}(k) = \lambda P(k) \theta_i(k-1) + d_i(k) P(k) F(k) \tag{4.17}$$

Substituting for $P(k)$ from equation 4.15 into equation 4.17 above, finally results in a recursive update for the FFENN weights $W_i(k)$ for each output (noting that $P(k-1)\theta_i(k-1) = W_i(k-1)$:

$$W_i(k) = W_i(k-1) + P(k) F(k) \epsilon_i(k) \quad \text{for } i = 1, \ldots, m \text{ outputs} \tag{4.18}$$

where the prediction error $\epsilon_i(k)$ is as defined in equation 4.6 above. Therefore, to summarize, the complete learning algorithm for the FFENN is summarized by equations 4.5, 4.6, 4.15 and 4.18 respectively. Numerically robust versions of RLS can be used instead of the above, such as the Givens Least Squares algorithm [192] or the Square Root RLS [85]. For fixed functionally expanded input terms $F(k)$

at the FFENN hidden layer, the output mean squared error surface was shown to be quadratic (in equation 4.7 above) and the RLS algorithm will therefore guarantee convergence to the single global minimum. The simpler Least Mean Squares (LMS) algorithm which is a stochastic gradient algorithm can also be used for updating the FFENN output layer weights as follows:

$$W_i(k) = W_i(k-1) + \mu\epsilon_i(k)F(k) \tag{4.19}$$

where μ controls the convergence rate. However, the rate of convergence of the LMS algorithm is dependent on the spread of the eigenvalues of the input expansion model's autocorrelation matrix, $E[F(k)F^T(k)]$, with a large eigenvalue spread dictating a significantly slower convergence rate [128].

On the other hand, the Least Squares criterion based RLS algorithm will converge more rapidly compared to the LMS but at the expense of an increased computational complexity, $O(N^2)$ compared to $O(N)$. The convergence properties of the RLS and LMS are well established [56]. Various *Fast* RLS (FRLS) algorithms have also been recently proposed to reduce the complexity of the RLS from $O(N^2)$ to $O(N)$ [93], as have a new class of algorithms linking the normalized LMS and RLS algorithms [159], whose listings can be found in [164] and [159] respectively; and can also be readily employed to train the above FFENN structure.

Thus, once the full expansion model at the hidden layer of the FFENN is specified (using the design strategy of section 4.2.1), the exponentionally weighted RLS algorithm can then be used to provide an efficient means for real time adaptation of the network weights. This will give the FFENN a significant advantage over the multi-layered neural network structures such as the MLP in recursive identification applications.

4.2.5 Pruning of the Fully Expanded FFENN: Use of Model Validation Tests

Problem of Overfitting

A serious issue in the application of any neural network to a problem domain is the size of the network as measured by the number of free parameters of the network. As for other methods of functional approximation, such as polynomial, too many free parameters will allow the network to fit the training data arbitrarily closely, but will not necessarily lead to optimal generalization [115]. As discussed in Appendix A.6, although there is no general method to determine the optimal size of the network for a particular task, there are statistical arguments which suggest that the number of training parameters required to fully determine the weights in a network are proportional to the number of weights in the network [24]. A rule of thumb often cited is that the number of weights be less than one-tenth of the number of training patterns [115]. Commonly employed methods for dealing with this problem in highly non-linear-in-the-parameters multi-layered neural networks such as the MLP, are pruning using computationally expensive techniques such as optimal brain damage, optimal brain surgeon, weight sharing, weight decay, weight elimination and soft weight sharing strategies [24]. For the linear in the parameters FFENN, whose response is a linear function of its weights, a computationally efficient pruning strategy is employed, as discussed below:

The idea in building a parsimonous FFENN model is to include only those functionally expanded input terms f_i which have a significant contribution to the output in the model. In other words, we want to select the dominant or most significant terms from the expanded input model set $f_i, i = 1, \ldots, N$ to construct the parsimonous model. We employ an iterative pruning-retraining strategy whereby: the fully expanded FFENN structure (resulting from the design strategy described above) is initially trained on the training data set and the output MSE value on the training set computed. The insignificant functions in the expansion model set with the smallest weights (relative to the largest most significant function's weight) are then successively pruned one by one starting with the least significant one. Basis functions with small weights tend to contribute less to the overall computation of the FFENN output, and thus are promising pruning candidates.

After each insignificant function is pruned, the resulting pruned FFENN structure is re-trained on the same train set in order to determine the optimal weights for the remaining unpruned functions, and the resulting output MSE value computed at each stage. The pruning process is stopped at the stage when a pruned FFENN is found to be incapable of reducing the output MSE on the training set to the desired level or when the *model validation tests* are found not to be satisfied. The model validation tests are discussed later in this section.

The final weights resulting from the above pruning re-training process are then fixed and the FFENN's generalization ability is tested (by performing adaptive or iterated predictions) on the previously unseen test data. Extensive simulation results will demonstrate the effectiveness of the above pruning strategy in the modeling of both simulated and real-world non-linear time series processes.

Note that the FFENN pruning scheme described above has an off-line training requirement, similar to the traditional method of supervised learning employed in conventional neural network structures. If however, the pruning strategy is omitted, then the fully expanded FFENN can also be employed to learn in an on-line fashion similar to Haykin *et al*'s recently reported architecture [90], that is, it can learn continuously (using the exponentionally weighted RLS algorithm described above) to adapt to statistical variations of the incoming non-stationary time series while performing its filtering role at the same time. This capability of the FFENN (and its recurrent counter-part) is investigated in chapter 5, when it is applied to the problem of on-line adaptive non-linear prediction of highly non-stationary signals.

Model Validity Tests

For modeling of non-linear dynamical systems, we propose that the above pruning process for the FFENN be used in conjunction with *model validity tests* [109] in order to detect any sort of inadequacy in the modeling process. The need of these tests, which have been shown to be a very powerful aid in neural network modeling [110], arises owing to the fact that for some non-linear system modeling

applications, a detailed knowledge of the non-linear system may be needed in order to decide which of the FFENN's expanded input functions are significant and which can be ignored for the application at hand. Thus if a pruned FFENN's input expansion model set does not include all the significant terms constituting the underlying system, then the pruned FFENN could still yield a satisfactory MSE performance on the training set, but will perform poorly on the test set, that is, the pruned FFENN will not be a good representation of the non-linear system. Computationally simple correlation based model validation tests employed at the end of each training phase should indicate when such a deficient situation arises, that is, whether the pruned FFENN model has indeed successfully modeled the underlying non-linear system dynamics in order to perform successfully on the test set.

The correlation based model validity tests are devised on the principle that, if the pruned FFENN model's structure (represented by number of selected functionally expanded input terms) and parameter (weighting coefficient) values are correct, then the network's output prediction errors should be unpredictable from (that is, uncorrelated with) all linear and non-linear combinations of past inputs and outputs. The correlation based model validity tests for MIMO non-linear systems (with n inputs and m outputs) are given below:

If the identified (that is, trained) FFENN model of the non-linear MIMO dynamical system is adequate, then it can be shown that [109] the prediction errors $e_u(k) = e_1(k), \ldots, e_m(k)$ (corresponding to each of the m outputs should satisfy the following conditions (for FFENN distinct inputs represented by $x_i(k)$ for $i = 1, \ldots, n$):

$$
\begin{aligned}
R_{e_u e_v}(\tau) &= \sigma(\tau) \text{ (an impulse) } i = 1, \ldots, m \;\; v = 1, \ldots, m \\
R_{x_u e_v}(\tau) &= 0 \text{ for all } \tau \;\; u = 1, \ldots, n \;\; v = 1, \ldots, m \\
R_{e_u (e_v x_w)}(\tau) &= 0 \text{ for } \tau \geq 0 \;\; u = 1, \ldots, m \;\; v = 1, \ldots, m \;\; w = 1, \ldots, n \\
R_{(x_u x_v)' e_w}(\tau) &= 0 \text{ for all } \tau \;\; u = 1, \ldots, n \; v = 1, \ldots, n \;\; w = 1, \ldots, m \\
R_{(x_u x_v)' (e_w e_z)}(\tau) &= 0 \text{for all } \tau \; u = 1, \ldots, n \; v = 1, \ldots, n \; w = 1, \ldots, m \; z = 1, \ldots, m
\end{aligned}
$$

where $R_{ab}(\tau)$ indicates the cross-correlation function between $a(k)$ and $b(k)$ at lag τ, $e_v(k)x_w(k) = e_v(k+1)x_w(k+1)$, $(x_u(k)x_v(k))' = x_u(k)x_v(k) - \overline{x_u(k)x_v(k)}$ where $\overline{x_u(k)x_v(k)}$ represents the time average or mean value of $x_u(k)x_v(k)$. In practice, the normalized correlations are computed. Note that if several lagged values of the same single network input $x_u(k)$ are used, that is if $x_u(k-1), x_u(k-2), \ldots, x_u(k-d)$ are used as explicit inputs, such as for modeling of dynamical systems, then the above tests will reduce to the case for identification of Single Input Single Output (SISO) non-linear systems, which are summarized below (assuming the input variable is $x(k)$ whose lagged values $(x(k-1), \ldots, x(k-n))$ are used as inputs):

$$\begin{aligned} R_{ee}(\tau) &= \sigma(\tau) \text{ an impulse} \\ R_{xe}(\tau) &= 0 \text{ for all } \tau \\ R_{e(ex)}(\tau) &= 0 \text{ for } \tau \geq 0 \\ R_{(x^2)'e}(\tau) &= 0 \text{ for all } \tau \\ R_{(x^2)'(e^2)}(\tau) &= 0 \text{ for all } \tau \end{aligned}$$

where $e(k)x(k) = e(k+1)x(k+1)$, $(x^2)'(k) = x^2(k) - \overline{x^2(k)}$, and $\overline{x^2(k)}$ is the time averaged or mean value of $x^2(k)$. The sampled normalised correlation function between two sequences $a(k)$ and $b(k)$ is given by:

$$R_{ab}(\tau) = \frac{\sum_{k=1}^{S-\tau} a(k)b(k+\tau)}{(\sum_{k=1}^{S} a^2(k) \sum_{k=1}^{S} b^2(k))^{1/2}}$$

Normalization ensures that the correlation functions lie in the range $(+1, -1)$ irrespective of the signal strengths. The criteria used for model validation is that: if all the above correlation functions fall within the (95%) confidence intervals $(+1.96/\sqrt{S}, -1.96/\sqrt{S})$ (where S represents the number of system observations or output samples used in the modeling (training) process), then the FFENN model is regarded as adequate. In general, one can allow for a few data samples to lie slightly outside the confidence bands if the system data being modeled is real world and the identification procedure is a recursive one [192].

As an alternative to the above correlation tests, the Chi-squared statistical tests introduced by Bohlin [10] and adapted to the non-linear case by Leontaritis and Billings [111] can also be used to validate an identified (trained) model. If we define $\Omega(k)$ to be the following s-dimensional vector:

$$\Omega(k) = [r(k)r(k-1)\dots r(k-s+1)]^T$$

where $r(k)$ is some chosen non-linear function of the past inputs, outputs and prediction errors, then the Chi-squared statistic η is computed using the formula:

$$\eta = S\mu^T(R^TR)^{-1}\mu \tag{4.20}$$

where

$$R^TR = \frac{1}{S}\sum_{k=1}^{S}\Omega(k)\Omega^T(k)$$

and

$$\mu = \frac{1}{S}\sum_{k=1}^{S}\Omega(k)e(k)/\sigma_e$$

where σ_e^2 is the variance of the residuals or prediction error $e(k)$. Under the null hypothesis that the input-output data are generated by the model, the statistic η is asymptotically chi-squared distributed with s degrees of freedom. Thus, if the values of η for several different choices of $r(k)$ are within the acceptance region (95%), that is

$$\eta \leq X_s^2(\alpha)$$

the identified model can be regarded as adequate, where $X_s^2(\alpha)$ is the critical value of the chi-squared distribution with s degrees of freedom for the given significance level $\alpha(0.05)$ [192].

The power of the above tests in the selection of an appropriate identified FFENN predictor model will be demonstrated through the identification of numerous simulated and real world dynamical non-linear time series processes in the next section. Note that to date, the above tests have not been shown to be applicable to the validation of models of chaotic processes.

Next are presented extensive simulation results illustrating the application of the new FFENN structure to the modeling of a large class of both simulated and real-world non-linear dynamical processes.

4.3 Non-linear Dynamical System Modeling Using the FFENN: Application Examples and Comparative Performance Analysis

The rationale of the FFENN, like the RBF, is that a feedforward neural network may be viewed as performing a simple curve-fitting operation in a high dimensional space [187]. According to this viewpoint, whenever a sufficient number of data points are available, the data should be divided into a fitting (training) set and testing set. The former is used in the FFENN expansion model's selection procedure and the latter is used to validate the selected network. This provides an interpretation for the two fundamental concepts in neural networks, namely learning and generalization. Learning can be viewed as producing a surface in multi-dimensional space that fits the set of data in some sense (in the present case, Least Squares sense), and generalization is then equivalent to interpolating the test data set on this fitting surface [187]. This approach has been adopted in the present study.

Specifically, we consider the problem of re-constructing the *generator* of the data, where the data is a single or multi-dimensional time series obtained from either simulated chaotic maps, equation-error, output-error processes (see chapter 2 for details) or real world noisy dynamical processes. The aim in all cases is to use the deterministic Feedforward FENN (and the Recurrent FENN developed in chapter 5) to attempt a reconstruction of the generator of the data.

Note that as discussed in chapter 2, time series prediction is synonymous with modeling of the underlying physical mechanism responsible for its generation [90]; and consequently, for the case of chaotic data, it will only be able to capture the

short term structure of the time series.

In the following sections, numerous simulation case studies are performed using a variety of both simulated and real-world noisy non-linear time series processes. The modeling capability of the FFENN has also been compared with various other conventional and recently reported feedforward and recurrent ANN based predictor models.

4.3.1 CASE I: Modeling of simulated Chaotic Time Series: Logistic Map, Henon Map and the Mackey Glass Equation

The 1st Order Chaotic Logistic Map modeling

For the first application we have used the iterated quadratic or logistic map

$$y(k) = 4y(k-1)(1-y(k-1))$$

on the unit interval to serve as an example of deterministic chaos. A single step predictor based on a fully expanded (1,8;1)FFENN structure of the form:

$$\hat{y}(k) = \sum_{j=1}^{8} w_j(k-1) f_j(k)$$

was postulated comprising eight functionally expanded terms ($N = 8$) of the single past input $y(k-1)$ as illustrated in equation 4.1 of section 4.2.1. The single output of the FFENN provided an estimate of the next step prediction $\hat{y}(k)$. The fully expanded network was evaluated on 200 logistic map samples generated from an initial condition $y(0) = 0.2$. The RLS updating algorithm with an exponential weighting factor selected to equal $\lambda = 0.92$ (as it gave the lowest training Mean Squared Error (MSE) value of 0.0046), was employed which resulted in the following expansion model at the end of training:

$$\begin{aligned}\hat{y}(k) \;=\; & -1.246y(k-1) + 6.963\sin(y(k-1)) - 0.897\sin(2y(k-1)) \\ & +0.025\sin(3y(k-1)) + 8.83\cos(y(k-1)) - 0.125\cos(2y(k-1)) \\ & -0.041\cos(3y(k-1)) - 8.667 \qquad (4.21)\end{aligned}$$

It can be seen that after just a single pass through the training set (additional passes were not found to yield any significant performance gains), the FFENN based predictor has identified the significant functional components (basis functions) representing the underlying logistic map time series, whilst attenuating the other insignificant expansion terms. To illustrate the advantage of using a linear-in-the-parameters FFENN-RLS structure trained by a fast least squares based RLS algorithm, an equivalent non-linear-in-the-output-layer weights (1,8;1)FFENN-DR predictor was devised by incorporating a sigmoidal activation function (the $tanh(.)$ basis function) at its output layer following the linear combiner. The non-linear Delta Rule algorithm (described in Appendix A.3) was thus employed for its weight updates (with an optimally chosen step-size to achieve the fastest convergence), and the FFENN-DR predictor was evaluated on the same 200 sample training set. However, the FFENN-DR failed to model the logmap (the output error didnot converge to zero) with the above number of training samples, thus illustrating the learning difficulty inherent in the conventional non-linear in the parameters neural network based predictor models. The squared error curves of the FFENN-RLS and the FFENN-DR structures are illustrated in Figures 4.6 and 4.7 respectively.

The pruning strategy described in section 4.2.5 was used to prune the above fully expanded FFENN-RLS predictor model. An optimal (1,3;1)FFENN structure resulted after selecting the three most significant terms from the expansion model of equation 4.21 above (that is, the third, sixth and last terms on the right hand side of equation 4.21), and then re-training the pruned FFENN using another single pass through the same training set which resulted in a training MSE of 0.0057. The final (1,3;1)FFENN structure resulting from the pruning re-training process comprised the following terms (which represent a one-step ahead FFENN based predictor model for the chaotic time series):

$$\hat{y}(k) = 3.914 \sin(y(k-1)) + 7.167 \cos(y(k-1)) - 7.163 \qquad (4.22)$$

Further pruning resulted in a (1,2;1)FFENN structure which was found to

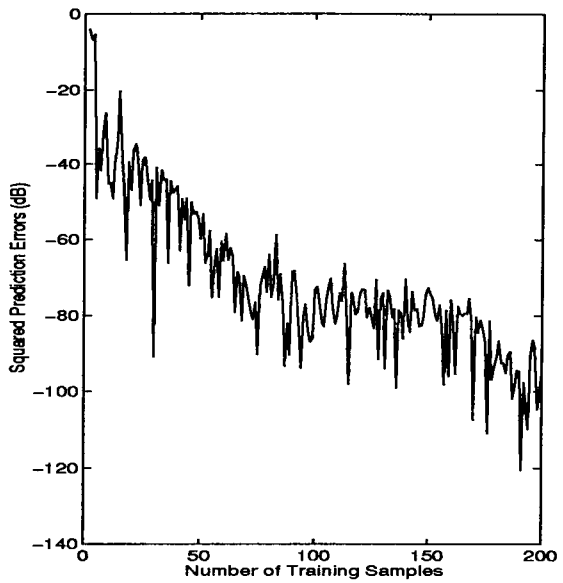

Figure 4.6: Output Error Squared of the new linear-in-the-output-layer weights FFENN-RLS one-step predictor on 200 Logmap Training Samples

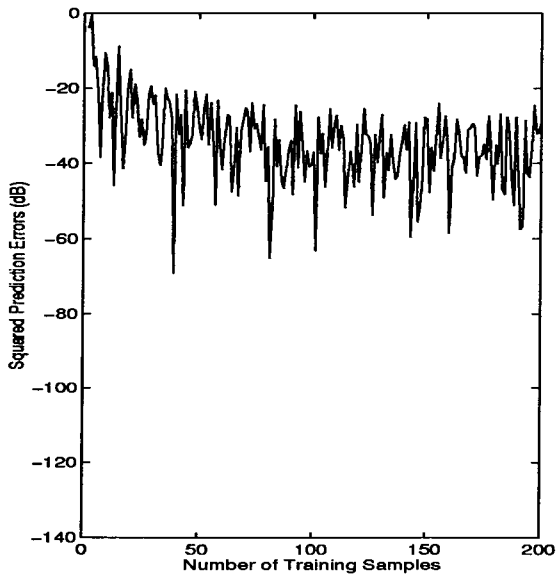

Figure 4.7: Output Error Squared of non-linear-in-the-output-layer weights FFENN-DR one-step predictor on 200 Logmap Training Samples

be incapable of reducing the training set error to near zero (giving a significantly large training set MSE of 0.1375).

To enable comparison with other recently reported neural network predictors [42], the generalisation performance of the above trained, fully expanded and pruned FFENN predictor models with fixed weights was evaluated (tested) on a further 100 point test sequence generated with same initial condition $x(0) = 0.2$. The comparison measure used was the output test MSE computed according to:

$$MSE = \frac{\sum_{k=1}^{100}(y(k) - \hat{y}(k))^2}{100}$$

FFENN		McDonnel *et al*		
(1,8;1)	(1,3;1)	(1,2;1)	(1,10;1) MLP	(1,15;1) MLP
3.5×10^{-10}	4.5×10^{-6}	4×10^{-4}	1.7×10^{-3}	2×10^{-4}

Table 4.2: MSE Performance Comparison of Single Step Predictors on the Logistic map for 100 samples.

The results summarised in Table 4.2 are reported for the fully expanded (1,8;1)FFENN and the pruned (1,3;1)FFENN structures. The output test MSE values produced by these predictor models are compared with the published MSE results of the (1,2;1) predictor model reported by McDonnel *et al* [42], which evolved from a parent recurrent IIR perceptron after 5000 generations of an optimisation process incorporating a complex multi-agent stochastic search technique. Two other Back Propagation based (1,10;1) and (1,15;1) feedforward MLP predictors comprising sigmoidal hidden units (also reported in [42]) have also been included in the Table 4.2 for comparison. As can be seen from Table 4.2, both the fully expanded and the pruned FFENN-RLS predictor models give a MSE significantly lower than the other recently reported structures. Weigend *et al* [115] report using three Radial Basis Function (RBF) nodes to model the logmap without further elaboration. A significant advantage of the FFENN based predictor model over McDonnel *et al*'s [42] evolved (1,2;1) feedforward structure is that

the significant functional terms constituting the pruned (1,3;1)FFENN structure, were directly identified from the fully expanded linear-in-the-parameters FFENN's expansion model which evolved after just just a single pass through the 200 sample training set. On the other hand, many thousands of generations were required by McDonnel *et al*'s evolutionary search technique in order to yield an adequate transversal filter predictor of the correct model order and weighting coefficients. Compared to the MLP based non-linear predictors from [42], it can be seen that:

- FFENN predictors offer significantly lower MSEs.
- The pruned FFENN predictor employs less than half the number of hidden units.
- The linear-in-the-parameters FFENNs employ least squares based learning algorithms which guarantee fast convergence to the global minimum solution, whereas the highly non-linear-in-the-parameters MLP structure employs the computationally expensive non-linear Delta Rule based Back Propagation (BP) algorithm which requires long training times and also suffers from the problem of converging to local minimum solutions (that is, can give unpredictable solutions as discussed in Appendix A).

Figure 4.8 gives the state space plot for the optimal 3 term (1,3;1)FFENN predictor model outputs and the actual quadratic mapping function. As can be seen, the FFENN predictor model illustrated in equation 4.22 employing a weighted squashing sigmoidal like basis function (approximated by the $\sin(y(k-1))$) and a Gaussian like activation function (approximated by the $\cos(y(k-1))$), has clearly produced an interpolation surface to the logistic map which generated the data. Note that it does not interpolate the waveform itself which is a fruitless exercise as the waveform is chaotic.

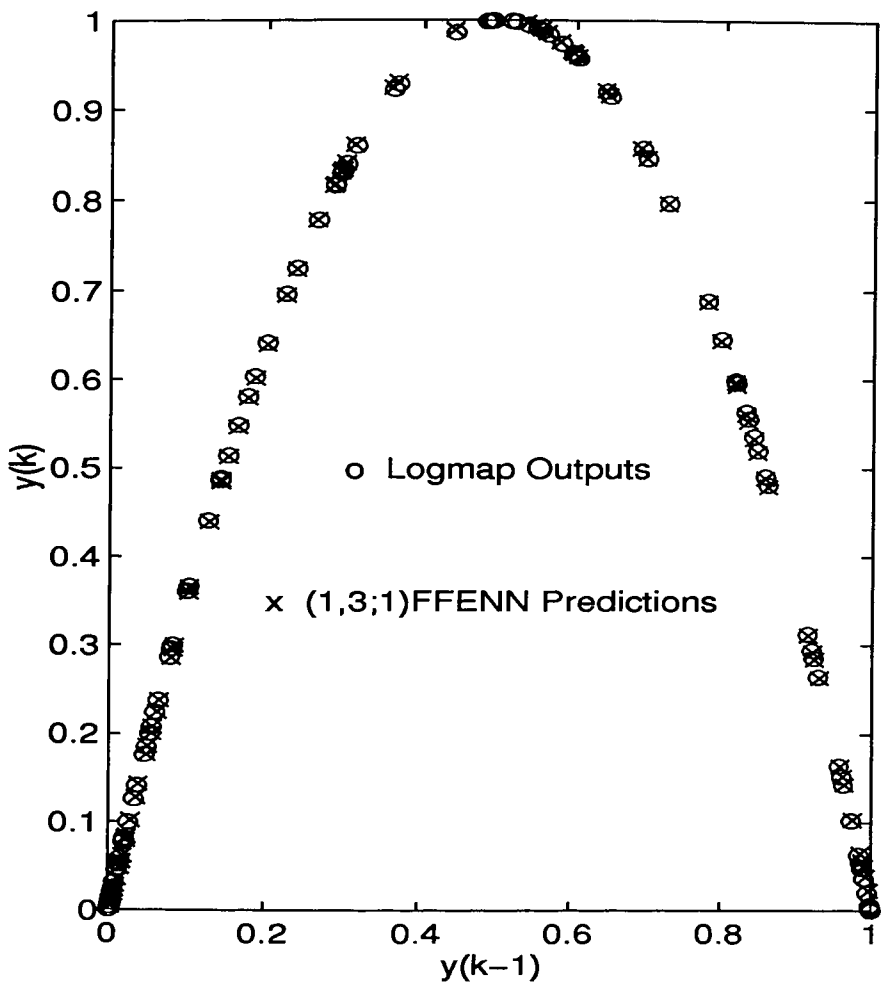

Figure 4.8: State Space plot for logistic map outputs and the (1,3;1)FFENN one-step predictions

Modeling of 2nd Order Chaotic Henon Time Series

In this case, the Henon time series represented by the following equation (which provides chaotic behaviour):

$$y(k+1) = 1 - 1.4y(k)^2 + 0.3y(k-1)$$

with initial conditions $y(-1) = y(0) = 0$ was modeled by a 2-input (2,20;1)FFENN structure employing the $N = 20$ term expansion model illustrated in equation 4.2. The output of the FFENN provided the one-step ahead prediction estimate $\hat{y}(k+1)$.

The FFENN was initially trained on the first 100 samples of the Henon time series using the RLS algorithm with λ set to 0.88 as it gave the lowest training set MSE of 0.010. Higher values of λ were found to result in higher training set MSEs. The resulting FFENN's expansion model was then optimized by successively pruning off the insignificant functions as discussed in the pruning strategy of section 4.2.5.At each pruning step, the pruned FFENN structure was re-trained on the same training set in order to obtain the optimum weight coefficients for the selected most significant functions. For comparison, a 2-input Volterra Neural Network (VNN) employing a 24 term truncated Volterra series expansion model (with up to fourth order outer-product expansion of the inputs) also trained using the RLS algorithm on the first 100 Henon time series samples with λ set to 0.88. The (2,24;1)VNN produced an MSE of 0.014 on the training set and employed the following 24 term expansion model:

$$\begin{aligned} F(k) \;=\; & y(k-1), y(k-2), y(k-1)^2, y(k-2)^2, y(k-1)^3, y(k-2)^3, y(k-1)^4 \\ & y(k-2)^4, y(k-1)y(k-2), y(k-1)^2y(k-2), y(k-1)y(k-2)^2 \\ & y(k-1)^3y(k-2), y(k-1)y(k-2)^3, y(k-1)^4y(k-2) \\ & y(k-1)y(k-2)^4, y(k-1)^2y(k-2)^2, y(k-1)^2y(k-2)^3 \\ & y(k-1)^2y(k-2)^4, y(k-1)^3y(k-2)^2, y(k-1)^3y(k-2)^3 \\ & y(k-1)^3y(k-2)^4, y(k-1)^4y(k-2)^2, y(k-1)^4y(k-2)^3 \\ & y(k-1)^4y(k-2)^4 \end{aligned} \tag{4.23}$$

FFENN			VNN
(2,20;1)	(2,6;1)	(2,4;1)	(2,24;1)
2.43×10^{-8}	3.3×10^{-7}	1.5×10^{-4}	1.44×10^{-7}

Table 4.3: Test MSE Performance Comparison of Single Step Predictors on the Henon map for 500 samples.

The output test MSEs of the fully expanded FFENN, its pruned versions, and the VNN on 500 test samples are compared in Table 4.3. As can be seen, the fully expanded (2,20;1)FFENN employing the rich expansion model combining (to a variable extent) the modeling capabilities of the RBF, MLP and VNN, offers a significantly superior output test MSE performance compared to the (2,24;1)VNN employing a purely polynomial expansion model. The pruned (2,6;1)FFENN predictor model (which produced a training set MSE = 0.012) and the (2,4;1)FFENN predictor model (which produced a training set MSE= 0.011) are seen to offer comparable performance to the (2,24;1)VNN. Note that further pruning of the (2,4;1)FFENN resulted in the (2,3;1)FFENN which produced a significantly large training set MSE of 0.075. Hence, the trained (2,4;1)FFENN was concluded to be the optimally pruned FFENN one-step ahead predictor model for the Henon time series, and its structure which evolved at the end of training and generated the one-step predictions on the test data, is illustrated below:

$$\hat{y}(k) = 0.33\sin(y(k-2)) + 5.36\cos(y(k-1)) - 0.48\cos(2y(k-1)) - 4.11$$

A state-space plot for the actual Henon time series and the (2,4;1)FFENN predictor's one-step ahead predictions is shown in Figure 4.9. As can be seen, the 4−term FFENN predictor model comprising a combination of weighted sigmoidal and Gaussian shaped basis functions has successfully produced an interpolation surface to the 2nd order Henon map which generated the chaotic time series.

The FFENN and the VNN structures were then employed to perform 2-step ahead predictions on the Henon time series. Both structures were first trained on the first 100 samples using the RLS algorithm (with optimal values of $\lambda = 0.999$ for the FFENN and $\lambda = 0.95$ for the VNN); and then tested for generalization

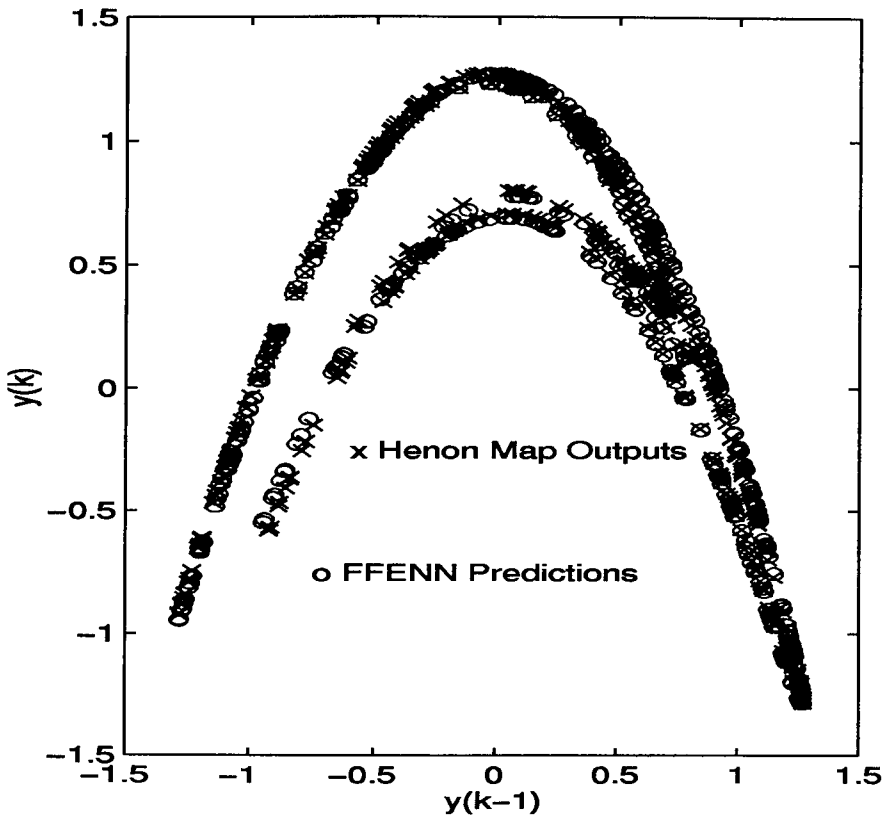

Figure 4.9: State Space plot for Henon map outputs and the (2,4;1)FFENN one-step predictions.

FFENN		VNN
(2,20;1)	(2,5;1)	(2,24;1)
0.0404	0.0463	0.0647

Table 4.4: Test MSE Performance Comparison of 2-Step Predictors on the Henon map for 500 samples.

that is, used to perform two-step ahead predictions with fixed weights on the next 500 samples of the Henon time series. As can be seen from the output MSE test values shown in Table 4.4, both the fully expanded (2,20;1)FFENN (which produced a training set MSE of 0.219) and the pruned (2,5;1)FFENN (which gave a lower training set MSE of 0.087) based 2-step predictors outperform the more complex 24 term (2,24;1)VNN based 2-step predictor by producing significantly lower test MSEs. This indicates that the FFENN's richer non-linear expansion model is more effective in modeling the underlying dynamics of the 2nd order Henon map which generated the chaotic time series. Further pruning of the (2,5;1)FFENN resulted in a (2,4;1)FFENN structure which produced a significantly large training set MSE of 0.684, on account of its inability to capture the underlying system representation. Hence the pruned (2,5;1)FFENN structure is concluded to be the optimally pruned FFENN 2-step predictor model.

The structure of the optimally pruned (2,5;1)FFENN-RLS based 2-step ahead predictor model for the Henon map which evolved at the end of training and was evaluated on the test data, is illustrated below:

$$\begin{aligned}\hat{y}(k+1) &= -0.329y(k-1) + 0.852\cos(2y(k-1)) - 0.015\hat{y}(k)\cos(y(k-1)) \\ &\quad +0.252y(k-1)\cos(\hat{y}(k)) - 0.461\hat{y}(k)y(k-1)\end{aligned}$$

Modeling of Chaotic Mackey-Glass Time Series

The Mackey-Glass equation has been used to represent a model for white blood cell production in leukemia patients [39]. The model is complicated by the addition of a time delay τ in the non-linear differential equation as follows:

$$\frac{dy(k)}{dt} = \frac{ay(k-\tau)}{1+y^c(k-\tau)} - by(k) \tag{4.24}$$

Consistent with McDonnel *et al* [42] and as discussed in Jones *et al* [43], the free parameters in the above model were selected as $a = 0.2$, $b = 0.1$, $c = 10$ and $\tau = 30$. This time series was recently modeled by McDonnel *et al* using a one step ahead (4-5-2-1)Recurrent predictor model comprising 17 parameters, which evolved from a parent Recurrent IIR Perceptron after 5000 generations of a training process incorporating a complex evolutionary search technique. The one-step ahead predictions of their recurrent model on a 500 sample training set, and a test set comprising the subsequent 500 time series samples were evaluated. The one-step prediction ability of the evolved model was measured in terms of the MSE and the *a*verage *r*elative *v*ariance (arv), which for a set of data samples S is defined as [42]:

$$arv(S) = \frac{\sum_{k=1}^{S}(y(k)-\hat{y}(k))^2}{\sum_{k=1}^{S}(y(k)-mean)^2} = \frac{MSE}{\sigma^2}$$

The use of the Mean (average) of the Squared Error makes the arv measure independent of the size of S. Additionally, normalization (division by σ^2, the estimated variance of data set) removes the dependence on the dynamic range of the data. This normalization implies that if the estimated mean of the data ($mean$) is used as a predictor, then a $arv = 1.0$ is obtained. The MSE for the training set was obtained from the last 450 points of the training set by McDonnel *et al* to allow for the transient effects.

In this section we investigate a new approach for modeling of the Mackey-Glass equation using the feedforward FENN. A fully expanded two-input (2,20;1)FFENN predictor model comprising a 20 term functional expansion model illustrated in equation 4.2 was postulated. Two previous time series samples $(y(k-1), y(k-2))$

FFENN			McDonnel *et al*
20 terms	4 terms	3 terms	17 terms
$0.0007(2.2 \times 10^{-5})$	$0.0023(7.2 \times 10^{-5})$	$0.0266(8.4 \times 10^{-4})$	$0.0014(5 \times 10^{-5})$

Table 4.5: Training Performance Comparison: *arv* and training set MSE (in parenthesis) measures of FFENN and other published Single Step Non-linear Predictors on the Mackey Glass Chaotic Time Series.

were used as inputs to the FFENN whose output thus provided the one-step ahead prediction $\hat{y}$ of the chaotic time series. The exponentionally weighted RLS algorithm with a forgetting factor of $\lambda = 0.97$ was employed for the recursive training. The MSE and *arv* measures of the fully expanded (2,20;1)FFENN on the last 450 samples of the 500 sample training set are listed in Table 4.5. Also shown are the performance measures achieved by the recurrent predictor model of McDonnel *et al* on the training set. Three and four input (3,38;1)FFENN and (4,64;1)FFENN models were found to give lower training set MSE values relative to the (2,20;1)FFENN, but were not employed in an attempt to find the non-linear FFENN predictor with the minimal complexity.

As can be seen from Table 4.5, a comparison of the performance measures on the training set shows that the 20 parameter FFENN predictor model outperforms the corresponding 17 parameter Recurrent predictor model reported by McDonnel *et al*. After iterative pruning-retraining of the (2,20;1)FFENN, an optimal 4 term (2,4;1)FFENN structure evolved which is illustrated below:

$$\begin{aligned}\hat{y}(k) &= 1.991y(k-1) - 0.877\sin(y(k-2)) + 0.738y(k-1)\sin(y(k-2)) \\ &\quad -1.045y(k-2)\sin(y(k-1)) \end{aligned} \tag{4.25}$$

Further pruning of the (2,4;1)FFENN resulted in a (2,3;1)FFENN which produced a significantly larger MSE (and *arv* measures) on the training set, thus reflecting its inability in capturing the underlying dynamics of the chaotic time series. The performance measures of the above 4 term FFENN one-step predictor model on the training set are also given in Table 4.5. The performance measures of the further pruned (2,3;1)FFENN are also shown for comparison. As can be seen from Table 4.5, the 4 term FFENN predictor model gives similar prediction

	(FFENN-RLS		McDonnel *et al*
Parameters	20	4	17
arv (test MSE)	$0.00057(1.8 \times 10^{-5})$	$0.0012(3.9 \times 10^{-5})$	$0.0025(9 \times 10^{-5})$

Table 4.6: Test Performance Comparison: *arv* and test set MSE (in parenthesis) measures of the Single Step Non-linear Predictors on the Mackey Glass Chaotic Time Series

performance compared to the fully expanded (2,20;1)FFENN and McDonnel *et al*'s recurrent predictor models on the training set, whereas the 3 term FFENN model is comparatively incapable of capturing the system representation.

The test MSE and *arv* performance measures of the optimally pruned and trained (2,4;1)FFENN predictor model illustrated in equation 4.25 and its parent (2,20;1)FFENN model on the 500 sample test set are compared in Table 4.6, along with corresponding measures achieved by the the Recurrent predictor model of McDonnel *et al*. As can be seen, the simple 4 term FFENN one-step predictor model comprising the weighted linear and non-linear functional terms (namely a combination of weighted, sigmoidal shaped activation function approximated by the sin(.) term, and multi-quadratic shaped activation functions approximated by $y(k-i)\sin(y(k-j)$ functions) illustrated in equation 4.25 above, gives significantly superior test performance measures compared to the more complex 17 term optimal Recurrent Predictor model reported in [42]. A sample of the actual Mackey Glass time series and the corresponding one-step predictions generated by the 4-term FFENN model illustrated in equation 4.25 above, are shown superimposed in Figure 4.10. State space plots for the actual Mackey Glass time series and the (2,4;1)FFENN one-step predictor model are also shown in Figures 4.11 and 4.12 respectively, both obtained from 900 samples. As can be seen by comparing the two Figures 4.11 and 4.12, the simple 4−term FFENN one-step predictor has successfully produced an interpolation surface to the complex Mackey Glass Equation which generated the chaotic time series.

McDonnel *et al* also report on a two-step ahead predictor model yielding

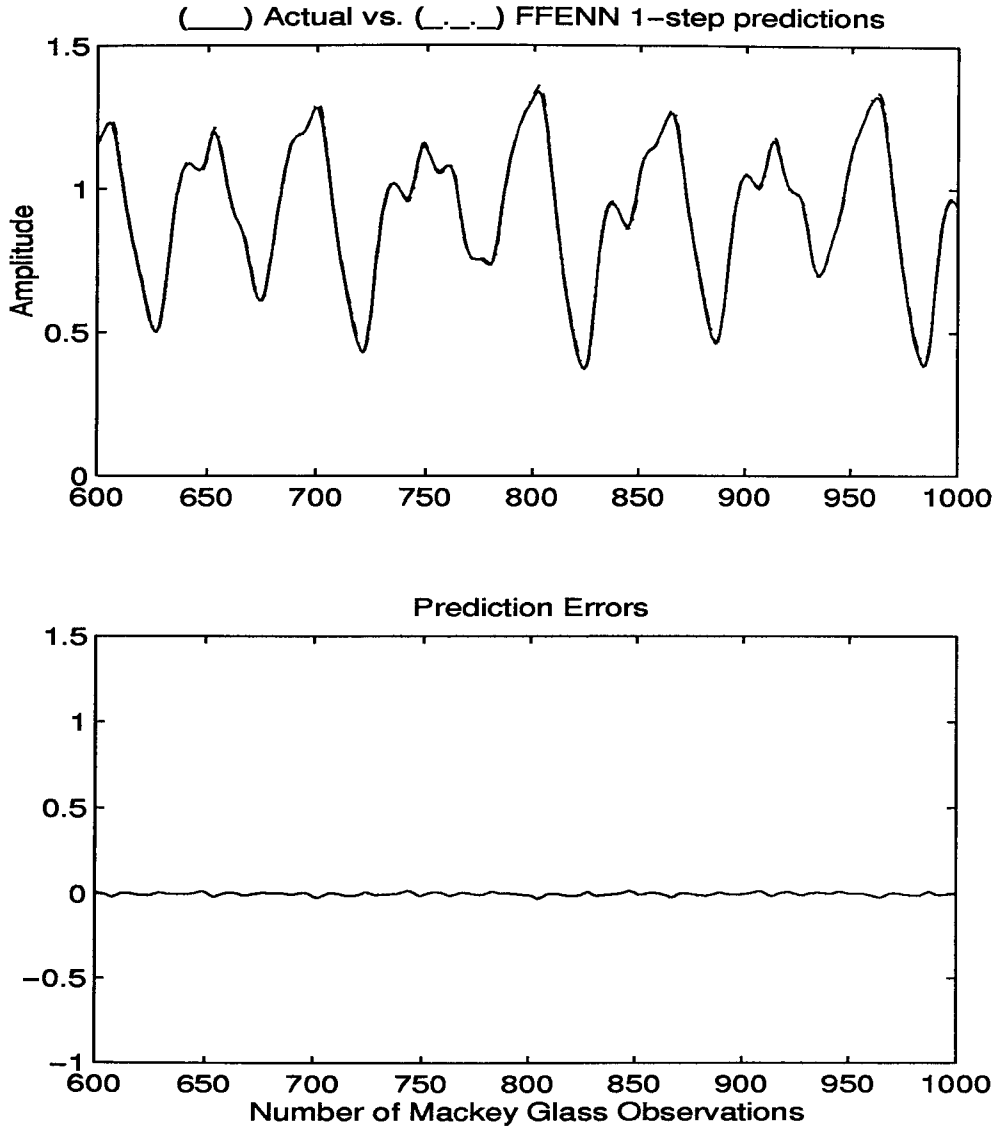

Figure 4.10: Comparison of evolved (2,4;1)FFENN model's one step predictions with actual Mackey Glass test Data.

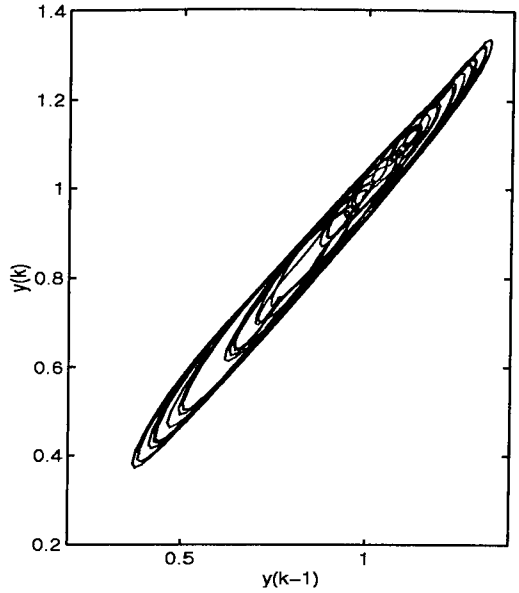

Figure 4.11: State Space plot for the actual Mackey Glass Time Series.

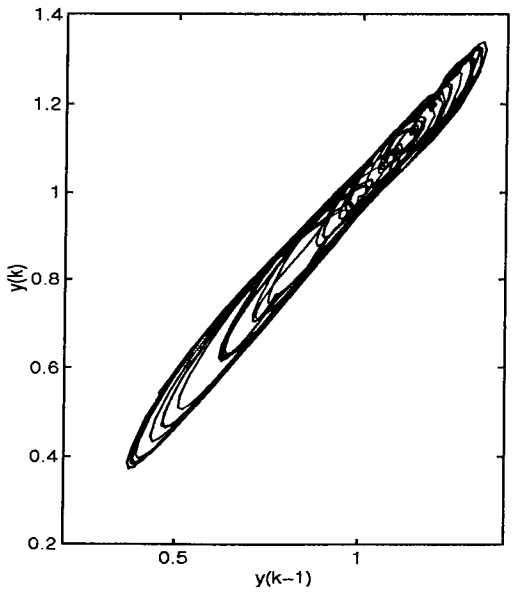

Figure 4.12: State Space plot for the (2,4;1)FFENN one-step predictions.

	(FFENN	McDonnel *et al.*
Parameters	7	17
test MSE	0.0016	0.0070

Table 4.7: Test Performance Comparison: test set MSE of Two-Step Non-linear Predictors on the Mackey Glass Chaotic Time Series.

an $MSE = 0.0070$ on the test set. A corresponding (2,20;1)FFENN two-step ahead predictor model was devised and upon pruning, an optimal (2,7;1)FFENN model evolved which was found to produce an $MSE = 0.0016$ on the test set. The comparative results are summarised in Table 4.7. The final structure of the (2,7;1)FFENN two-step predictor model which evolved at the end of training and was evaluated on the test set, is illustrated below:

$$\begin{aligned}\hat{y}(k) &= 0.957\hat{y}(k-1) - 0.819\sin(2\hat{y}(k-1)) + 0.5626\sin(2y(k-2)) \\ &\quad +0.0689\cos(2\hat{y}(k-1)) - 0.948\hat{y}(k-1)\cos(y(k-2)) \\ &\quad +0.045y(k-2)\sin(\hat{y}(k-1)) + 1.418y(k-2)\cos(\hat{y}(k-1)) \quad (4.26)\end{aligned}$$

Hence, it can be concluded that the proposed richer functional expansion models have enabled the the new FFENN one-step and two-predictors (which are both feedforward structures), in better capturing the underlying chaotic system representation compared to the corresponding more complex, recurrent predictors of McDonnel *et al.*

In their paper, McDonnel *et al* have also shown their recurrent predictor to be more effective than its feedforward counterpart in modeling of the above Mackey-Glass time series. Therefore, it is expected that a recurrent FENN based predictor would out-perform the above Feedforward FENN predictor. This assertion is investigated in chapter 5.

4.3.2 Case II: modeling of simulated NARX and NAR type non-linear Dynamical Systems (including MIMO NAR)

NARX and NAR type dynamical systems were discussed in chapter 2. In this section, the application of the FFENN to modeling of simulated Single Input Single Output (SISO) NARX, NAR and Multi-Input Multi-Output (MIMO) NAR time series processes is investigated.

Modeling of a simulated NARX Process

Consider the following simulated NARX time series model [182]:

$$\begin{aligned} y(k) &= (0.8 - 0.5exp(-y^2(k-1)))y(k-1) - (0.3 + \\ &\quad 0.9exp(-y^2(k-1)))y(k-2) + x(k-1) \\ &\quad +0.2x(k-2) + 0.1x(k-1)x(k-2) + e(k) \end{aligned} \tag{4.27}$$

where the additive system noise $e(k)$ is a Gaussian white noise sequence with zero mean and variance $E[e^2(k)] = 0.04$, and the system input $x(k)$ is an independent sequence of uniform distribution with zero mean and variance 1.0. A (4,64;1)FFENN structure comprising the normalized input vector $[x(k-1)x(k-2)y(k-1)y(k-2)]$ and employing the 64 term expansion model illustrated in equation 4.4 was fitted to 500 data points generated from the NARX system equation 4.27 above. The fully expanded (4,64;1)FFENN structure was trained by the RLS algorithm with λ set to 0.99 (as it gave the lowest training set MSE of 0.0238) with the FFENN output providing a one step-ahead prediction $\hat{y}(k)$ of the actual system output $y(k)$. The squared prediction errors are plotted in Figure 4.13.

The model validity tests for the (4,64;1)FFENN are illustrated in Figure 4.14, which show the same order of magnitude as those reported in [182] and confirm that the fully expanded FFENN network is an adequate model for the NARX system as all the correlation tests are satisfied.

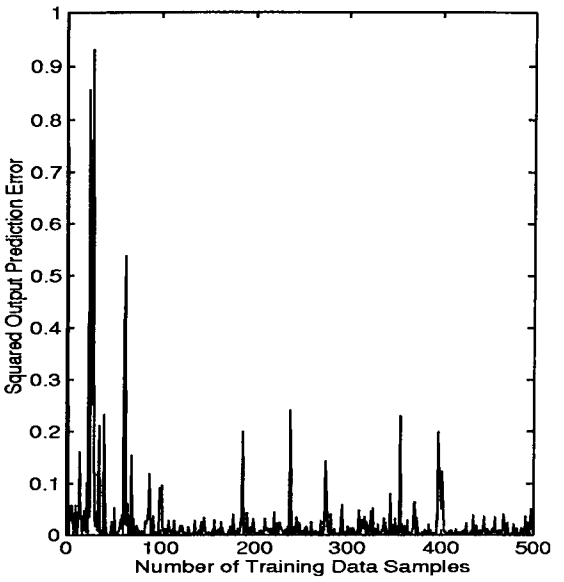

Figure 4.13: FFENN Output Prediction Errors (squared) on the 500 sample NARX training set.

In an attempt to optimize the size of the (4,64;1)FFENN, the iterative pruning-retraining strategy coupled with model validity tests (described in section 4.2.5) was employed. An optimal sized (4,15;1)FFENN predictor model evolved which produced a training set MSE of 0.0239. Its model validity tests were also found to be satisfied which confirmed that the (4,15;1)FFENN network is an adequate model of the NARX system. Further pruning resulted in a (4,14;1)FFENN network which produced a significantly larger training set MSE of 0.0740 and its model validity tests illustrated in Figure 4.15 show that the system inputs are highly correlated with the prediction errors, thus confirming that the 14 term (4,14;1)FFENN predictor model is not an adequate representation of the underlying NARX system.

Hence the pruned and trained (4,15;1)FFENN can be concluded to be the optimal sized FFENN model for the NARX system, and its structure is illustrated below (which represents a one-step ahead FFENN predictor model):

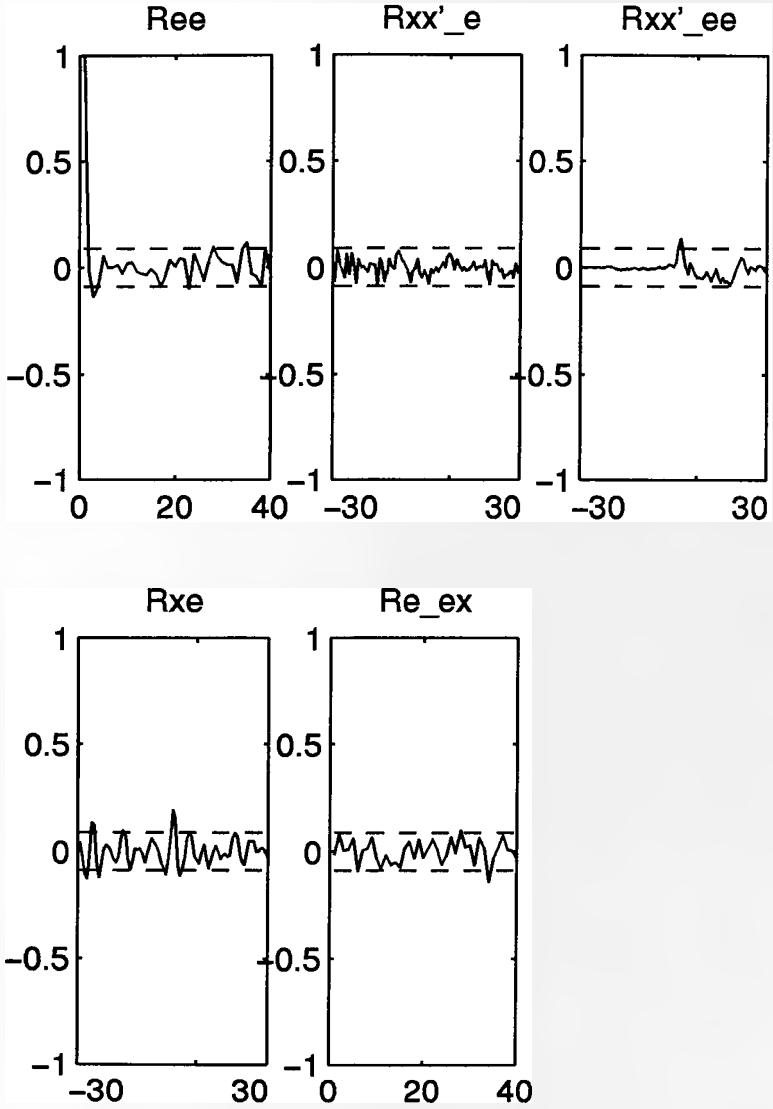

Figure 4.14: Correlation Based Model Validity Tests for the fully expanded (4,64;1)FFENN one-step predictor. Top left plot is $R_{e^2}(\tau)$, Top middle plot: $R_{x^2'e}(\tau)$, Top right plot: $R_{x^2'e^2}(\tau)$, Bottom left plot: $R_{xe}(\tau)$, Bottom middle plot: $R_{e(ex)}(\tau)$. The dashed lines represent the 95% confidence bands, and the x-axis of each plot denotes the lag τ values.

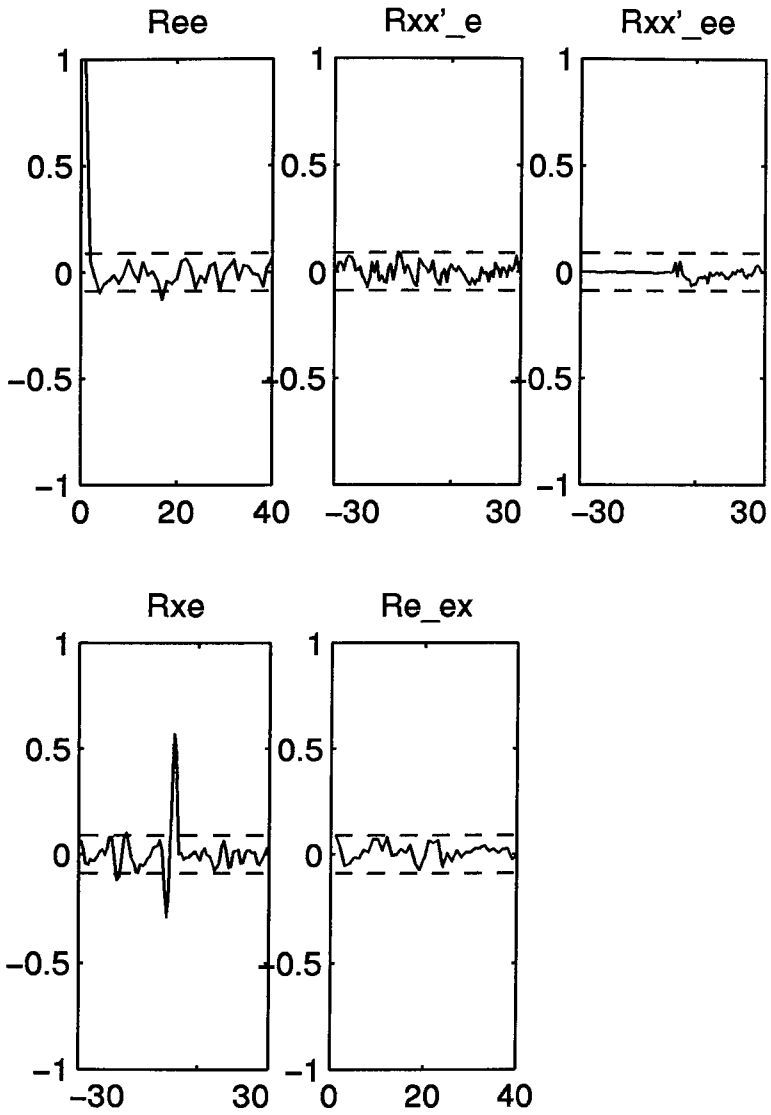

Figure 4.15: Correlation Based Model Validity Tests for the pruned (4,13;1)FFENN one-step predictor. Top left plot is $R_{e^2}(\tau)$, Top middle plot: $R_{x^2'e}(\tau)$, Top right plot: $R_{x^2'e^2}(\tau)$, Bottom left plot: $R_{xe}(\tau)$, Bottom middle plot: $R_{e(ex)}(\tau)$. The dashed lines represent the 95% confidence bands, and the x-axis of each plot denotes the lag τ values.

$$\begin{aligned}\hat{y}(k) = {} & 0.361x(k-1) + 1.057\sin(y(k-1)) + 0.107\sin(x(k-2)) \\ & -0.077\sin(2y(k-1)) - 0.166\sin(2y(k-2)) - 0.09\cos(y(k-1)) \\ & -0.406\cos(y(k-2)) + 0.385\cos(x(k-1)) + 0.229\cos(2y(k-2)) \\ & -0.117\cos(2x(k-1)) + 0.036x(k-1)\sin(y(k-1)) \\ & -0.398y(k-2)\cos(y(k-1)) + 0.152y(k-1)y(k-2) \\ & -0.012y(k-2)x(k-2) + 0.267\end{aligned}$$

After the identification procedure, an extra 500 test data points were generated from the NARX system equation 4.27 by using a different input sequence to that used to produce the training set. The measure of predictive capability of the fitted (trained) FFENN model over the test set was now chosen to be the FFENN model predicted outputs (with fixed weights), rather than the previously employed one-step ahead predictions (also computed with fixed FFENN weights). The model predicted outputs which are often a better metric of the trained model's predictive accuracy or generalization (for the case of non-chaotic systems) are defined by [110]:

$$\hat{y}(k) = F(x(k-1), x(k-2), \hat{y}(k-1), \hat{y}(k-2))$$

where $F(.)$ for the case of the FFENN, denotes the hidden layer functional expansion model. Note that the model predicted outputs are obtained by replacing the FFENN input vector $[y(k-1)y(k-2)x(k-1)x(k-2)]$ comprising the actual system outputs and inputs, by the input vector $[x(k-1)x(k-2)\hat{y}(k-1)\hat{y}(k-2)]$ consisting of the NARX system inputs and the model's own past predicted outputs.

The model predicted outputs of the above pruned (4,15;1)FFENN model as well as the parent fully expanded (4,64;1)FFENN model (with fixed weights and the system outputs replaced by the models own past predictions), were evaluated on the next 500 sample test set and are illustrated in Figures 4.16 and 4.17 respectively. In Figure 4.16, a few particularly large errors can be seen around

the two hundred and seventieth samples (suggesting amplification of the model predictions due its own past outputs) whereas, most of the other prediction errors can be seen to arise mainly at the peaks of the data.

As can be seen by comparing Figures 4.16 and 4.17, the optimally pruned and trained (2,15;1)FFENN network's model predicted outputs are significantly closer to the actual system outputs over the same test data as compared to the fully expanded (4,64;1)FFENN model, with the 15 term FFENN predictor model yielding an overall test MSE of 0.0229 (Figure 4.17) compared to 0.0331 (Figure 4.16) produced by the fully expanded FFENN model. This indicates that the generalization ability of the FFENN model has been significantly enhanced for this application by the use of the pruning strategy.

Figure 4.18 illustrates a general architecture for modeling of any equation-error (NARX) type non-linear dynamical system using a FFEN network. Note that for the non-linear system, it is assumed that the additive system noise is uncorrelated (as in equation 4.27 above). However, for the case of additive correlated noise sources, the prediction errors will also be correlated [110] (which can be readily detected by the model validity tests), and the corresponding FFENN model will consequently be a biased estimator of the underlying non-linear system being modeled. The bias can only be eliminated if the prediction errors (residuals) become uncorrelated with past measurements. One way to achieve this is to model the noise. For the case of additive coloured noise sources, a simple linear noise model can be used during the training period; that is, past noise samples can be fed as explicit inputs into the FFENN and their weighted values linearly combined at the output layer without undergoing any non-linear functional transformation. For more complex noise sources however, a detailed analysis needs to be carried out.

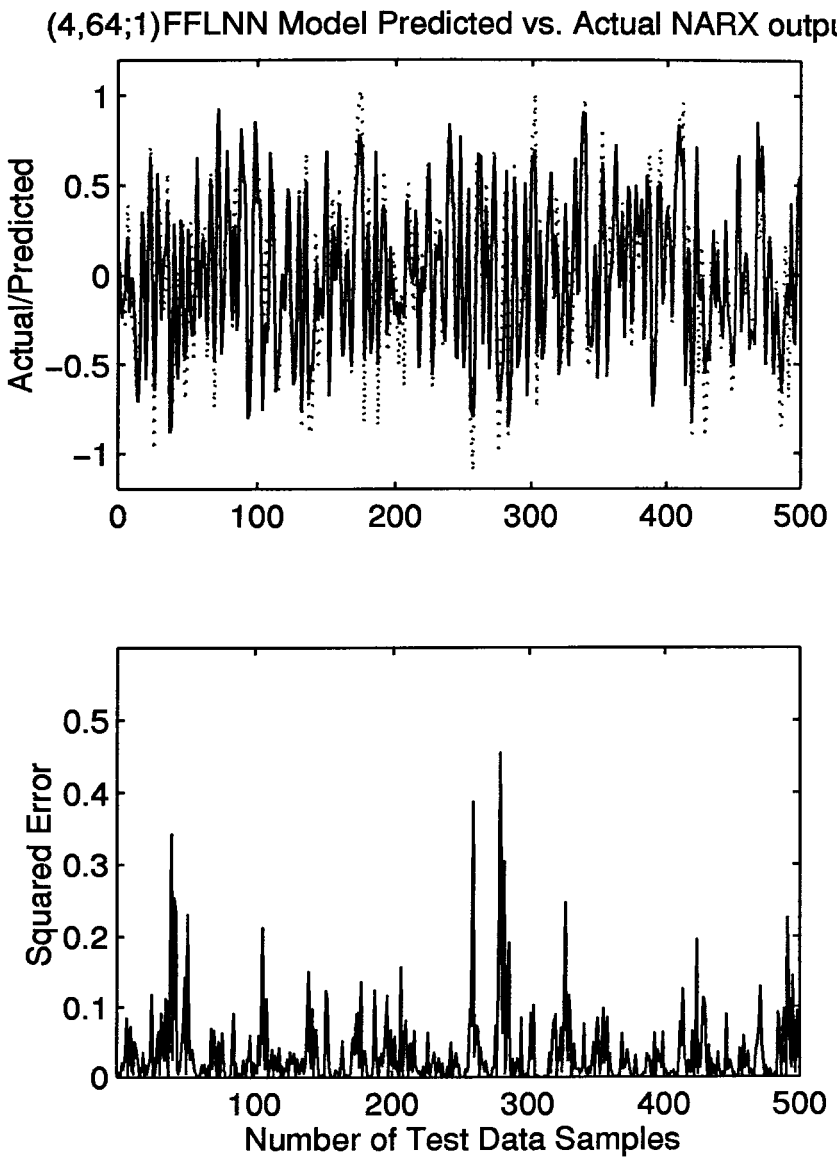

Figure 4.16: Comparison of the Actual NARX test outputs (solid line) with the fully expanded (4,64;1)FFENN model predicted outputs (dotted line).

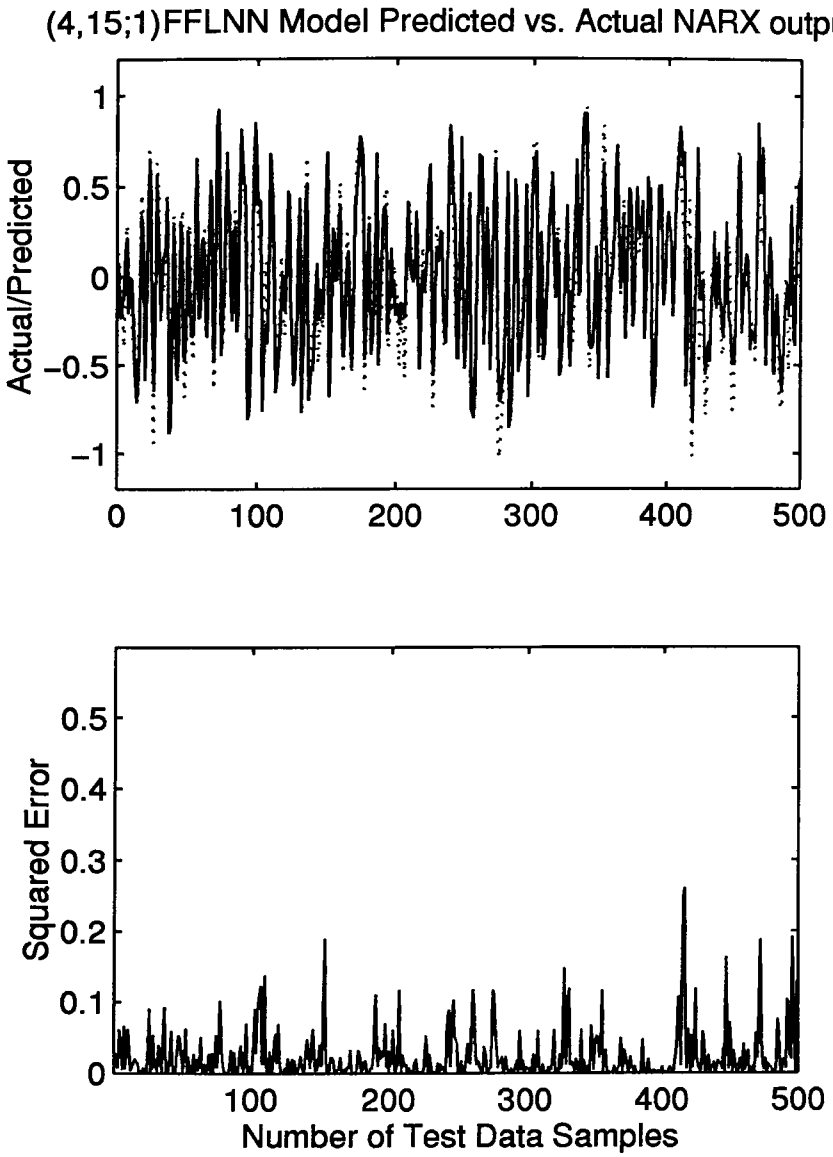

Figure 4.17: Comparison of the Actual NARX test outputs (solid line) with the optimally pruned (4,15;1)FFENN model predicted outputs (dotted line).

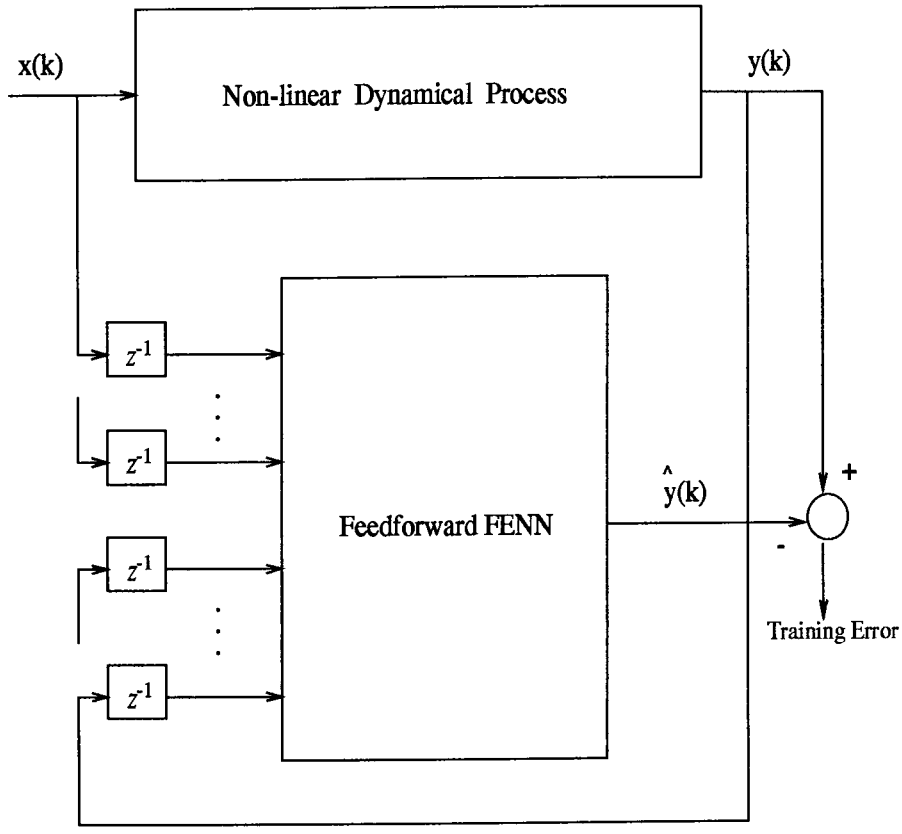

Figure 4.18: General equation-error Dynamical System Modeling using the FFENN

Modeling of a simulated NAR Process

Next, consider the following simulated NAR time series [191] generated by:

$$\begin{aligned} y(k) &= (0.8 - 0.5exp(-y^2(k-1)))y(k-1) - (0.3 + \\ &\quad 0.9exp(-y^2(k-1)))y(k-2) + 0.1\sin(\pi y(k-1)) + e(k) \quad (4.28) \end{aligned}$$

where $e(k)$ is a Gaussian white noise sequence with zero mean and variance 0.01. A two input FFENN employing the input vector $[y(k-1)y(k-2)]$ and the $N = 20$ term expansion model illustrated in equation 4.2 was fitted to the first 800 sample (training) set of a total of 1500 observation samples generated from the NAR system equation 4.28 above. The RLS algorithm with λ set to 1.0 produced the lowest training set MSE of 0.0112.

The structure of the trained fully expanded FFENN model which evolved at the end of training is illustrated below:

$$\begin{aligned} \hat{y}(k) &= 0.732y(k-1) + 0.929y(k-2) + 0.452\sin(y(k-1)) \\ &\quad +0.686\sin(y(k-2)) - 0.266\sin(2y(k-1)) + 0.103\sin(2y(k-2)) \\ &\quad +0.051\sin(3y(k-1)) - 0.086\sin(3y(k-2)) + 0.207\cos(y(k-1)) \\ &\quad -0.699\cos(y(k-2)) + 0.041\cos(2y(k-1)) + 0.413\cos(2y(k-2)) \\ &\quad -0.064\cos(3y(k-1)) - 0.131\cos(3y(k-2)) \\ &\quad +0.152y(k-1)\sin(y(k-2)) - 0.238y(k-1)\cos(y(k-2)) \\ &\quad +0.183y(k-2)\sin(y(k-1)) - 2.764y(k-2)\cos(y(k-1)) \\ &\quad -0.356y(k-1)y(k-2) + 0.215 \end{aligned}$$

Several chi-squared tests for the above (2,20;1)FFENN network were computed and all were found to be within the 95% confidence band. Four typical chi-squared tests and the auto-correlations of the prediction error $e(k)$ are shown in Figures 4.19 and 4.20. These model validity tests confirm that this FFENN network is an adequate model for the NAR system.

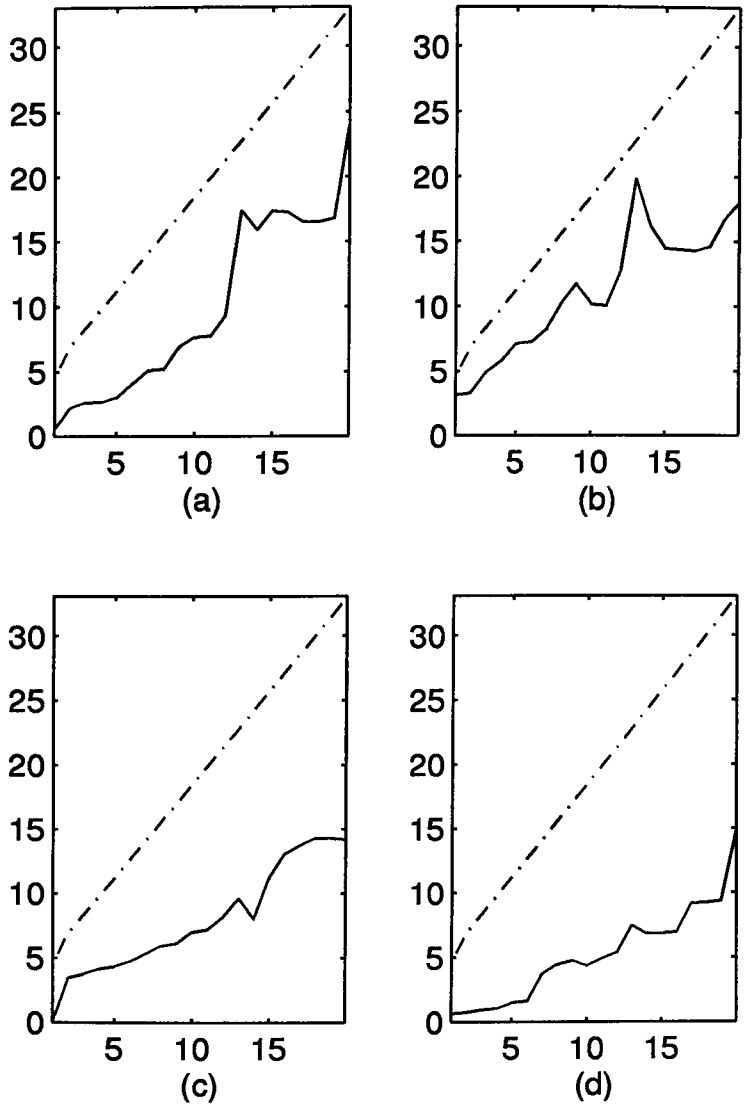

Figure 4.19: Chi-squared Statistic η Tests for the fully expanded (2,20;1)FFENN model: (a) $r(k) = y(k-1)$ (b) $r(k) = e(k-1)y(k-1)$ (c) $r(k) = e(k-1)y(k-1)^2$ (d) $r(k) = y(k-1)^2$. The dotted lines indicate 95% confidence limits and the x-axis of each plot denote the lags.

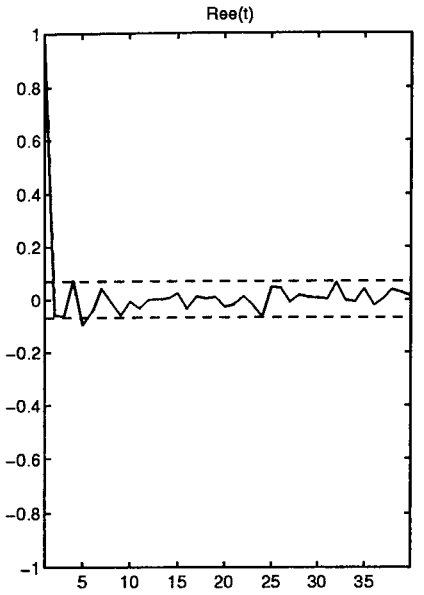

Figure 4.20: Auto-correlations of residuals $R_{e^2}(\tau)$ for the fully expanded (2,20;1)FFENN model.

Application of the pruning strategy of section 4.2.5 to the above fully expanded network model resulted in a pruned 3 term (2,3;1)FFENN model comprising the three most significant terms, which produced a training set MSE of 0.0116. The structure of the identified (2,3;1)FFENN predictor model is illustrated below:

$$\hat{y}(k) = 0.604y(k-1) + 1.696y(k-2) - 2.864y(k-2)\cos(y(k-1)) \qquad (4.29)$$

Several chi-squared tests for the above model were computed and all were within the 95% confidence bands. The auto-correlations of the prediction error $e(k)$ for this model are shown in Figure 4.21, and four typical chi-squared tests are illustrated in Figure 4.22. The model validity tests confirm that this FFENN structure is an adequate model for the time series.

Further pruning resulted in a (2,2;1)FFENN model comprising just the two most significant terms from the fully expanded FFENN's expansion model. However, the (2,2;1)FFENN model upon re-training produced a significantly larger

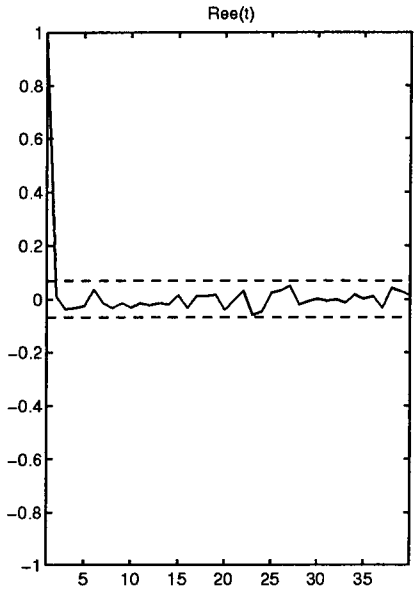

Figure 4.21: Auto-correlations of residuals $R_{e^2}(\tau)$ for pruned (2,3;1)FFENN model.

training set MSE of 0.161. The auto-correlations of its prediction errors are illustrated in Figure 4.23, which confirm that this model is not an adequate representation of the NAR time series system. Hence, the identified 3 term (2,3;1)FFENN illustrated in equation 4.29 above, can be concluded to be the optimal sized FFENN model for the NAR time series.

In order to best illustrate how accurately the identified (trained) (2,3;1)FFENN represents the NAR system, an examination of the autonomous NAR system response and the iterative FFENN model response is now made. The iterative (2,3;1)FFEN network response is generated from equation 4.29 above (representing the trained optimal FFENN model) by replacing the inputs $[y(k-1)y(k-2)]$ by the network's own past predictions $[\hat{y}(k-1)\hat{y}(k-2)]$, that is, the iterative FFENN model response is generated (without any knowledge of the actual time series values) from

$$\hat{y}(k) = 0.604\hat{y}(k-1) + 1.696\hat{y}(k-2) - 2.864\hat{y}(k-2)\cos(\hat{y}(k-1)) \qquad (4.30)$$

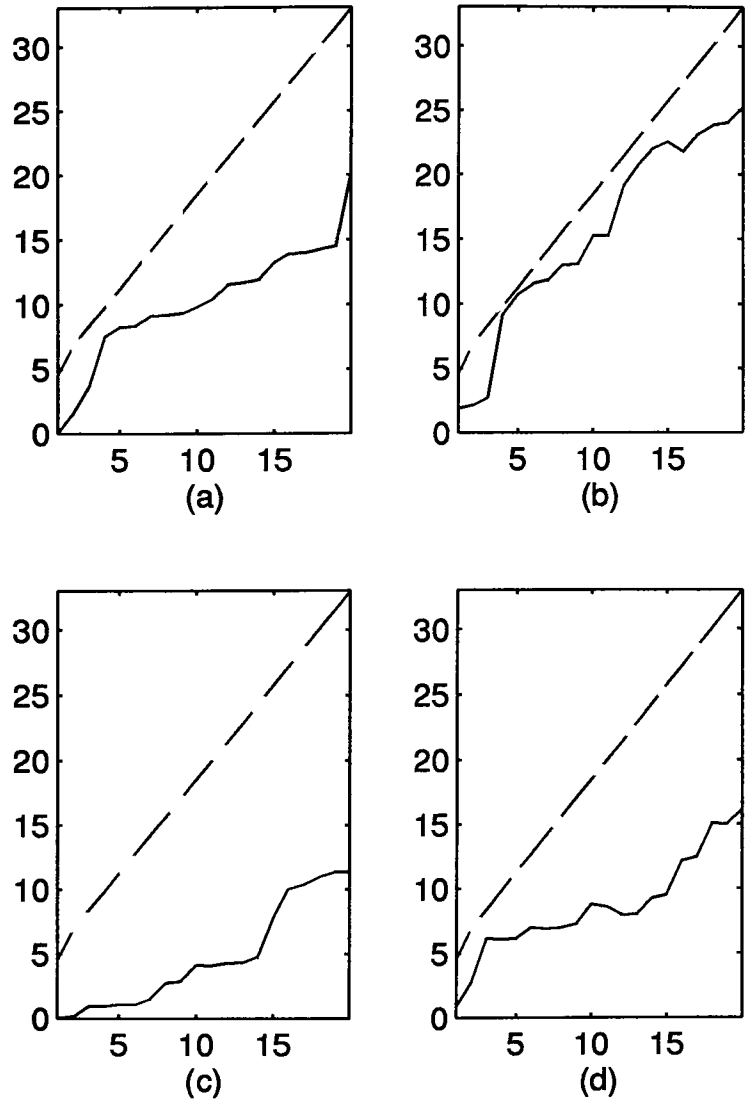

Figure 4.22: Chi-squared Statistic η Tests for the pruned (2,3;1)FFENN model: (a) $r(k) = y(k-1)$ (b) $r(k) = e(k-1)y(k-1)$ (c) $r(k) = e(k-1)y(k-1)^2$ (d) $r(k) = e(k-1)^2y(k-1)$. The dotted lines indicate 95% confidence limits and the x-axis of each plot denote the lags.

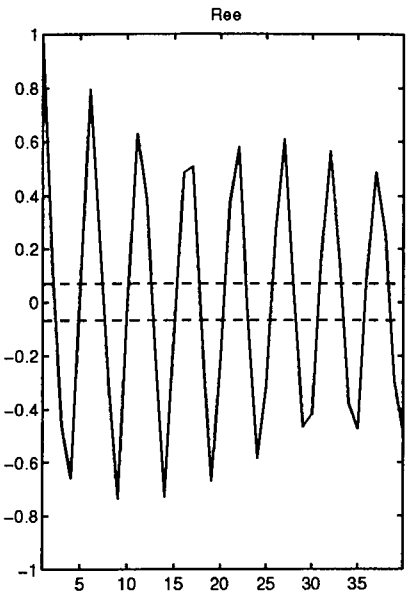

Figure 4.23: Auto-correlations of residuals $R_{e^2}(\tau)$ for (2,2;1)FFENN model.

It can easily be proved that without the noise $e(k)$, the simulated NAR system (equation 4.28) generates a stable limit cycle as shown in Figure 4.24, where the last 900 of a total of 1500 noise-free NAR system observation samples have been plotted. The iterative (2,3;1)FFENN response also produces a very similar limit cycle as can be seen from the same Figure 4.24, where 700 iterative model responses have been plotted. Both the limit cycles have approximately a period of 5, in the sense that every five samples approximately complete a circle (2π phase angle) in the state space [191].

Figure 4.25 shows that even though the (2,3;1)FFENN was identified (trained) using the noisy NAR system observations, the iterative FFEN network response closely matches the response of the autonomous system ($e(k) = 0$). This demonstrates that the identified 3 term FFENN model comprising weighted values of its past two predicted outputs ($\hat{y}(k-1), \hat{y}(k-2)$) and a weighted higher order sigmoidal-shaped activation function ($\hat{y}(k-2)\cos(\hat{y}(k-1))$, has indeed captured the underlying dynamics of the simulated non-linear system.

Note that the above NAR system was also modeled by Chen and Billings

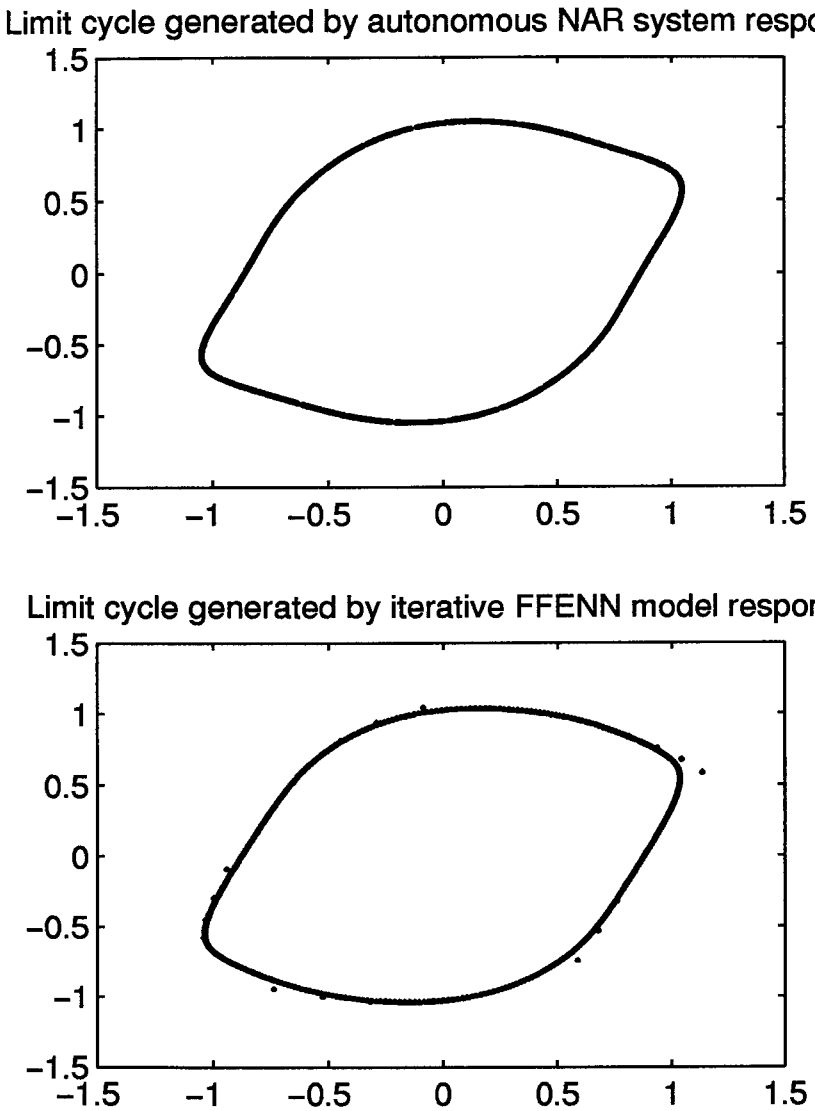

Figure 4.24: Response of Autonomous NAR system (900 samples) and Iterative 3-term FFENN model (700 samples). Projection to two subspaces (Limit Cycles).

[191] using the RBF and a linear-in-the-parameters Functional-Link Neural Networks (FLNN). In the case of the RBF, the computationally expensive Orthogonal Least Squares (OLS) learning algorithm was used to identify an optimal RBF model with 30 centres selected from a total of 1000 system observations. The non-linearity was chosen to be the thin plate spline activation function for all the centres. In the case of the FLNN, the OLS algorithm identified an optimal network comprising 9 terms selected from a 7-th order purely polynomial expansion model of the inputs. Both the 9 term FLNN which is in fact, strictly a Volterra Neural Network (VNN), and the 30 term RBF models provided almost identical performance to that provided by the new FFENN model reported above. However, the difference in the relative computational complexities of the structures is significant, with the FFENN employing just 3 basis functions compared to 30 and 9 employed by the RBF and FLNN models respectively. Also, note that the new FFENN model was identified above using a much simpler strategy; namely, by starting off with the newly proposed 20 term expansion model, the fully expanded (2,20;1)FFENN was initially trained using the fast exponentionally weighted RLS algorithm. From the resulting trained FFENN model, the insignificant functional terms were successively pruned, and at each stage the pruned FFENN model re-trained on the same training set in order to obtain the optimal weighting coefficients for the selected functions. The validity of the identified pruned FFENN model was verified using simple correlation and Chi-squared statistic based model validity tests.

Extension to modeling of MIMO NAR System

1500 samples of simulated time series were generated using the multi-dimensional non-linear model [189]:

$$\begin{aligned} y_1(k) &= (0.8 - 0.5exp(-y_1^2(k-1)))y_1(k-1) \\ &\quad (0.3 + 0.9exp(-y_1^2(k-1)))y_1(k-1) \\ &\quad +0.1\sin(y_2(k-1)) + e_1(k) \end{aligned}$$

$$\begin{aligned} y_2(k) &= 0.6y_2(k-1) + 0.2y_2(k-1)y_2(k-2) \\ &\quad 1.2\tanh(y_1(k-2)) + e_2(k) \end{aligned}$$

where $e_1(k)$ and $e_2(k)$ are Gaussian random variables of zero mean and covariance:

$$\begin{bmatrix} 0.01 & 0 \\ 0 & 0.01 \end{bmatrix}$$

A four-input two-output (4,64;2)FFENN structure comprising the input vector $[y_1(k-1)y_1(k-2)y_2(k-1)y_2(k-2)]$ and the 64 term expansion model illustrated in equation 4.4 of the design strategy of section 4.2.1, was trained on the first 800 sample training set using the RLS algorithm with $\lambda = 1$. The outputs of the FFENN provided the one-step ahead predictions $[\hat{y}_1(k)\hat{y}_2(k)]$. The pruning strategy of section 4.2.5 identified a 25 term FFENN predictor model from the parent fully expanded (4,64,2)FFENN. As before, an examination of the autonomous NAR system response and the iterative FFENN model reponse is now made, in order to best illustrate how accurately the identified (trained) 25 term FFENN represents the actual NAR system.

The autonomous ($e_1(k) = e_2(k) = 0$) NAR system generates stable limit cycles as shown in Figure 4.25, where the last 900 of the 1500 noise-free NAR system observation samples have been plotted. The iterative FFENN response (generated by replacing the inputs $[y_1(k-1)y_1(k-2)y_2(k-1)y_2(k-2)]$ by FFENN's own past predictions $[\hat{y}_1(k-1)\hat{y}_1(k-2)\hat{y}_2(k-1)\hat{y}_2(k-2)]$) also produces

almost identical limit cycles as can be seen from the same Figure 4.25, where 700 iterative model responses have been plotted. Figure 4.25 shows that even though the FFENN model was identified using noisy observations, its iterative outputs closely match to the outputs from the autonomous system. This confirms that the selected FFENN model has indeed captured the underlying dynamics of the MIMO NAR system.

Note that the above MIMO NAR system has also been efficiently modeled by Chen, Grant and Cowan [189] who proposed use of a two-output RBF network to capture the underlying system dynamics. However, their selected RBF network (which resulted from use of the Orthogonal Least Squares (OLS) algorithm) comprised a total of 50 Gaussian basis functions, which are exactly twice the number of basis functions employed by the above FFENN model. Furthermore, the performance of the 50 term RBF network, which was also evaluated in terms of its iterative response in [189], is seen to be no better than that attained by the 25 term FFENN model above (which comprises a combination of weighted sigmoidal shaped, Gaussian and multi-quadratic shaped activation functions).

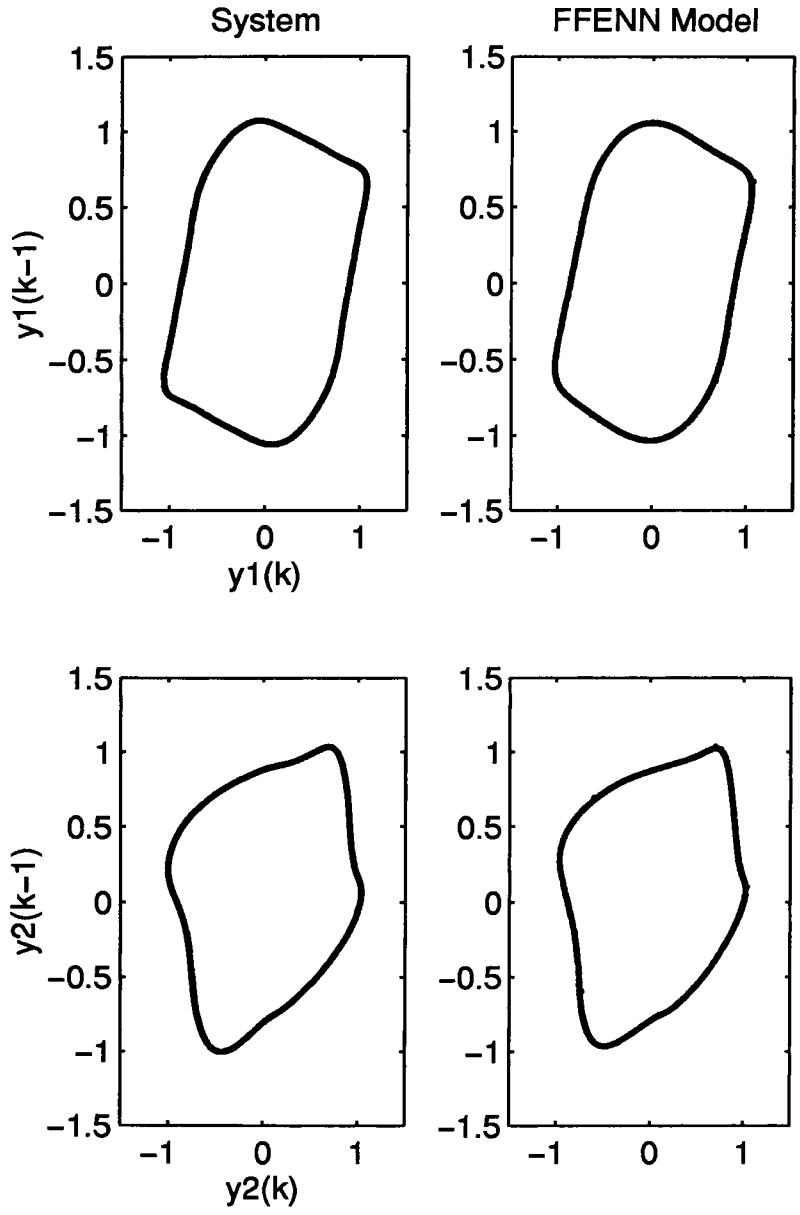

Figure 4.25: Response of Autonomous NAR system (900 samples) and Iterative 3-term FFENN model (700 samples). Projection to two subspaces (Limit Cycles).

4.3.3 CASE III: Modeling of Real-World Data: Stock Market and Sunspots

The illustrative examples considered so far have been ideal in the sense that we had apriori knowledge regarding the system order. However, even if the model order was assumed to be unknown, use of the model validation tests have proved to be useful in determination of the correct model order for modeling of simulated non-linear systems [110].

In real physical situations however, the unknown system noise can have the effect of forcing the system's actual low dimensional behaviour into a high dimensional space. This implies that the apparent order of the data is higher than it needs to be, which thus complicates the task of fitting an appropriate model order capable of capturing the evolution of the system. The major advances in dynamical systems practice to-date, have been in devising regularised approaches to deal with noise problems to determine how to project the data into subspaces in which the variance of the noise is minimsed [11] [22]. Currently, in our knowledge there are no equivalent regularisation schemes to deal with these problems using neural networks, and this is a potential area for future research.

In the following sections, we investigate the modeling of real world noisy possibly chaotic time series processes using the new feedforward FENN based approach.

Modeling of real Stock Market Data

In the first real application, real stock market data from the S & P weekly [120] was modeled using the FFENN structure. The available weekly price data comprised of a total of 156 samples and was divided into a fitting set comprising the first 110 samples, and the remaining 46 samples were selected to constitute the test data for evaluating the model's predictions. As there are not sufficient samples the model order has to be decided on the basis of the training set alone. This and the next application (involving modeling of real sunspots) are deliberately

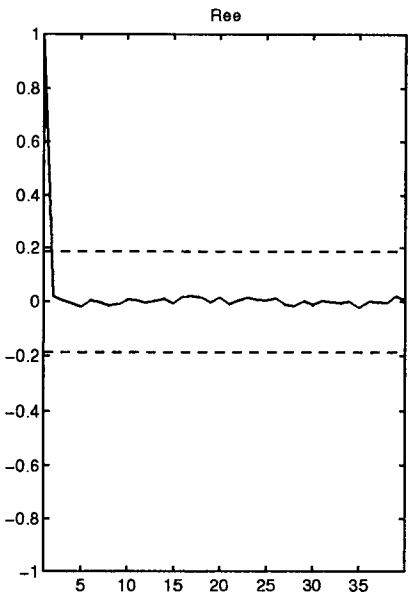

Figure 4.26: Auto-correlations of residuals $R_{e^2}(\tau)$ for fully expanded (unpruned) (1,8;1)FFENN model.

chosen examples which have a minimum of information upon which a decision has to be made. A single input (1,8;1)FFENN employing the 8−term expansion model illustrated in section 4.2.1, was initially fitted to the training data using the exponentionally weighted RLS algorithm to provide one-step (corresponding to one week) ahead predictions $\hat{y}(k)$ using just the single previous (week's) value $y(k-1)$. With $\lambda = 0.999$, an optimal training set MSE value of 0.0095 resulted. The prediction error auto-correlation and several chi-squared tests were computed for the final (1,8;1)FFENN model and all were found to be within the 95% confidence bands. These model validation tests which are illustrated in Figures 4.26 and 4.27, show that the fully expanded 8 term FFENN model is an adequate representation of the highly non-linear stock market time series data.

Use of two and three input fully expanded (2,20;1)FFENN and (3,38;1)FFENN structures whilst satisfying all the model validity tests, didnot give any improvement on the training set MSE value achieved by use of the (1,8;1)FFENN model.

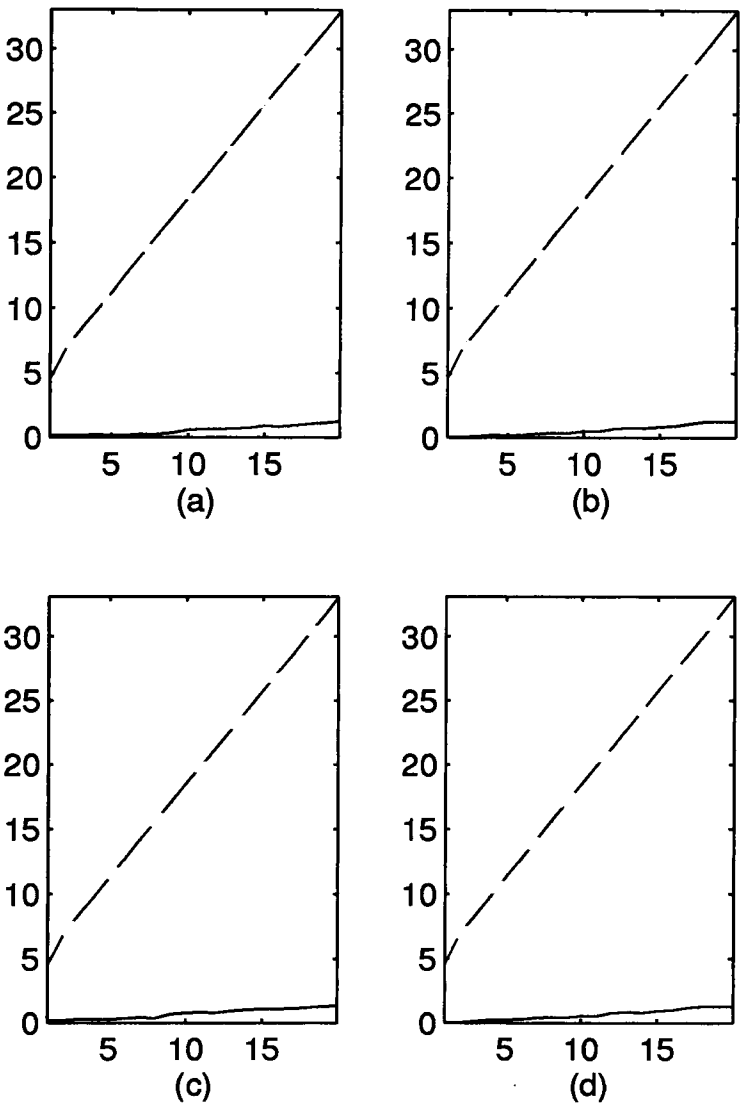

Figure 4.27: Chi-squared Statistic η Tests for the fully expanded (1,8;1)FFENN model: (a) $r(k) = y(k-1)e(k-1)$ (b) $r(k) = e(k-1)^2y(k-1)$ (c) $r(k) = e(k-1)y(k-1)^2$ (d) $r(k) = e(k-1)^2y(k-1)^2$. The dotted lines indicate 95% confidence limits and the x-axis of each plot denote the lags.

The structure of the (1,8;1)FFENN which evolved at the end of training is illustrated below:

$$\begin{aligned}\hat{y}(k) &= 0.525y(k-1) + 0.416\sin(y(k-1)) + 0.328\sin(2y(k-1)) \\ &\quad -0.127\sin(3y(k-1)) - 0.083\cos(y(k-1)) - 0.316\cos(2y(k-1)) \\ &\quad +0.264\cos(3y(k-1)) + 0.118 \qquad (4.31)\end{aligned}$$

As can be seen, weighted values of the previous (week's) stock market sample $y(k-1)$ and two sigmoidal shaped activation functions (approximated by the $\sin(y(k-1))$ and $\sin(2y(k-1))$ terms) and a Gaussian shaped term (approximated by the $\cos(2y(k-1))$ function) are the most significant functions which have successfully captured the short term (weekly) generation of all the non-linear trends within the stock market data set.

In order to optimize the size of the above fully expanded (1,8;1)FFENN model, the insignificant functions were successively pruned to yield an optimal (1,2;1)FFENN model comprising just the two most significant terms from the expansion model of equation 4.31 above. The pruned (1,2;1)FFENN upon training, produced a training set MSE of 0.0095 which is equal to that produced by the fully expanded (1,8;)FFENN over the same data set, and comprised the following weighted terms (which represents a one-step ahead predictor model):

$$\hat{y}(k) = 0.71y(k-1) + 0.314\sin(y(k-1)) \qquad (4.32)$$

Prediction error auto-correlation and various chi-squared based model validity tests for the above (1,2;1)FFENN model illustrated in Figures 4.28 and 4.29, confirm its validity over the training set.

The final 2 term model with fixed weights, was then used to perform one-step ahead (weekly) predictions over the 46 sample test set. A comparison of the actual and model one-step ahead predictions is illustrated in Figure 4.30, from which it can be seen that the predictions are very close to the actual stock prices, with an overall test MSE value calculated to equal 0.0028 . For comparison, a linear 10−input Moving Average (MA) model which averaged the 10 previous

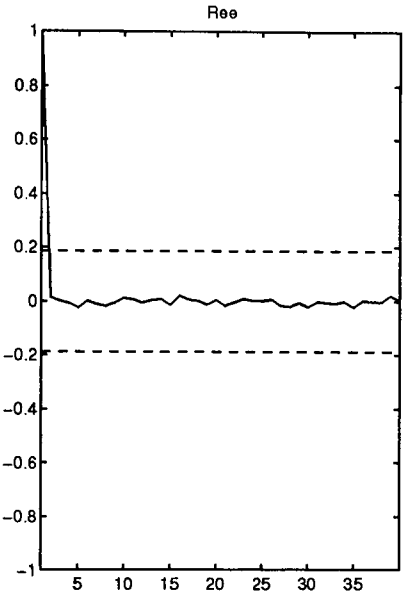

Figure 4.28: Auto-correlations of residuals $R_{e^2}(\tau)$ for final (1,2;1)FFENN model.

prices to predict the next price, is also shown superimposed on the two curves. As can be seen from Figure 4.30, the non-linear 2−term FFENN predictor totally outclasses the linear predictor.

To compare the performance of the FFENN model against other non-linear neural network predictors, a two input Volterra Neural Network (2,24;1)VNN model comprising the Volterra series expansion model employed in section 4.3.1, was also simulated under identical conditions. The trained (2,24;1)VNN model produced a test MSE value of 0.0110. The actual stock market data and VNN one-step ahead predictions are illustrated in Figure 4.31, along with the 10−input MA model's predictions. The prediction errors of the (1,2;1)FFENN and (2,24;1)VNN over the training and test sets are plotted in Figure 4.32, which shows that the 2−term FFENN not only better captures the coarse details of the stock market data, but also the fine details, such as, at the point when the stock market prices exhibit the greatest fluctuation, the FFENN model predicts the fluctuation with a prediction error of −0.28, whereas the 24−term VNN model can only predict the fluctuation with an error of −0.37.

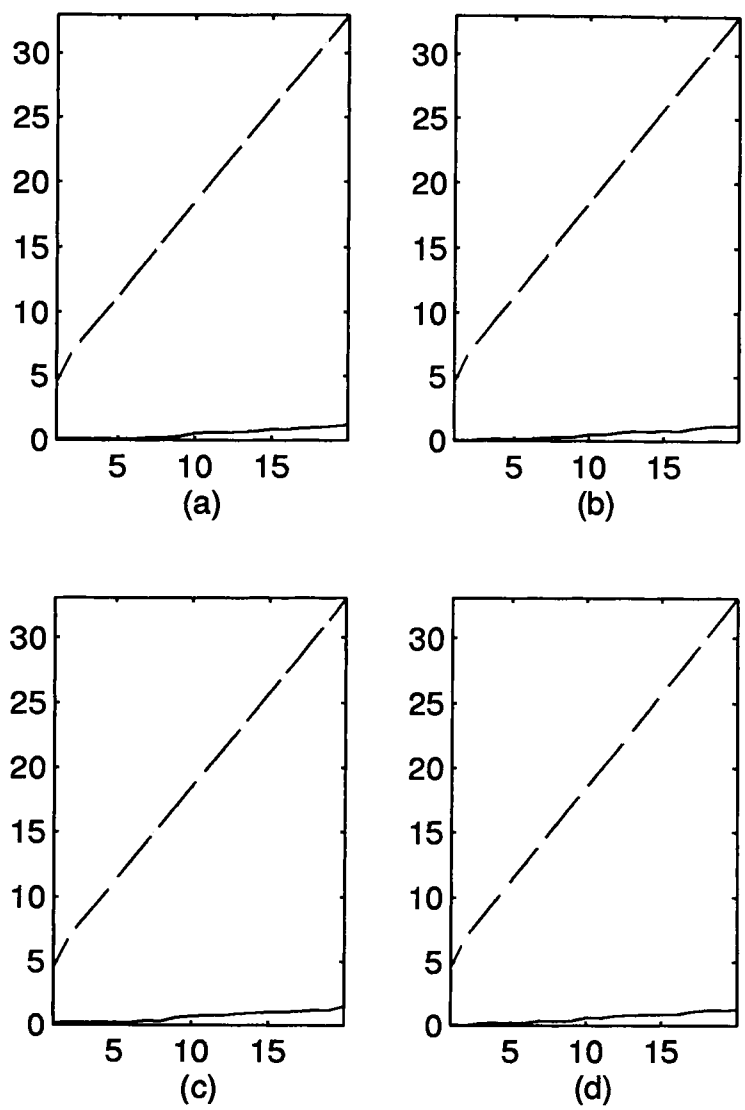

Figure 4.29: Chi-squared statistic η Tests for final (1,2;1)FFENN model: (a) $r(k) = y(k-1)e(k-1)$ (b) $r(k) = e(k-1)^2 y(k-1)$ (c) $r(k) = e(k-1)y(k-1)^2$ (d) $r(k) = e(k-1)^2 y(k-1)^2$. The dotted lines indicate 95% confidence limits and the x-axis of each plot denote the lags.

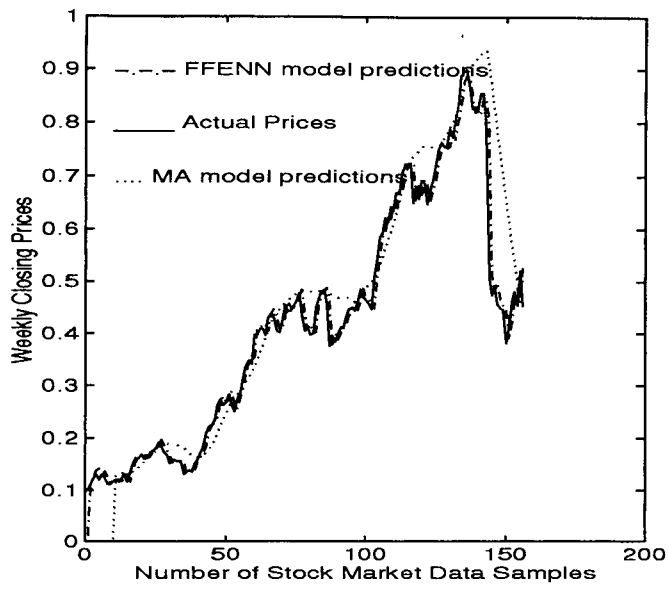

Figure 4.30: Comparison of optimally pruned first order (1,2;1)FFENN model one-step predictions with 10-th order linear MA model one-step predictions.

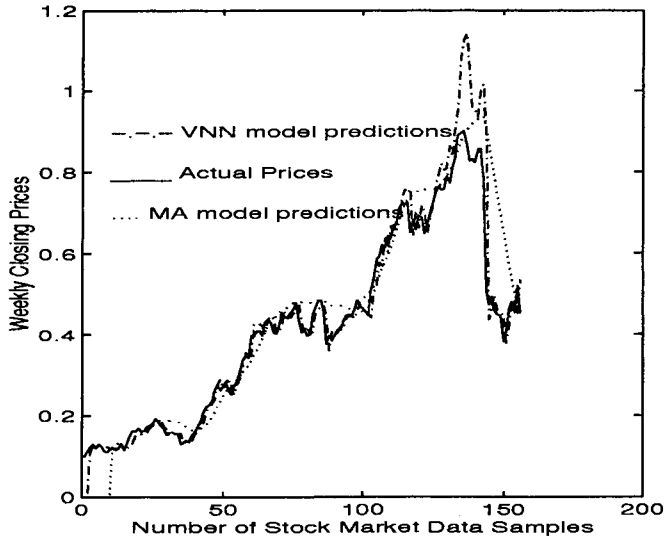

Figure 4.31: Comparison of 2nd order (2,24;1)VNN model one-step predictions with 10-th order linear MA model one-step predictions.

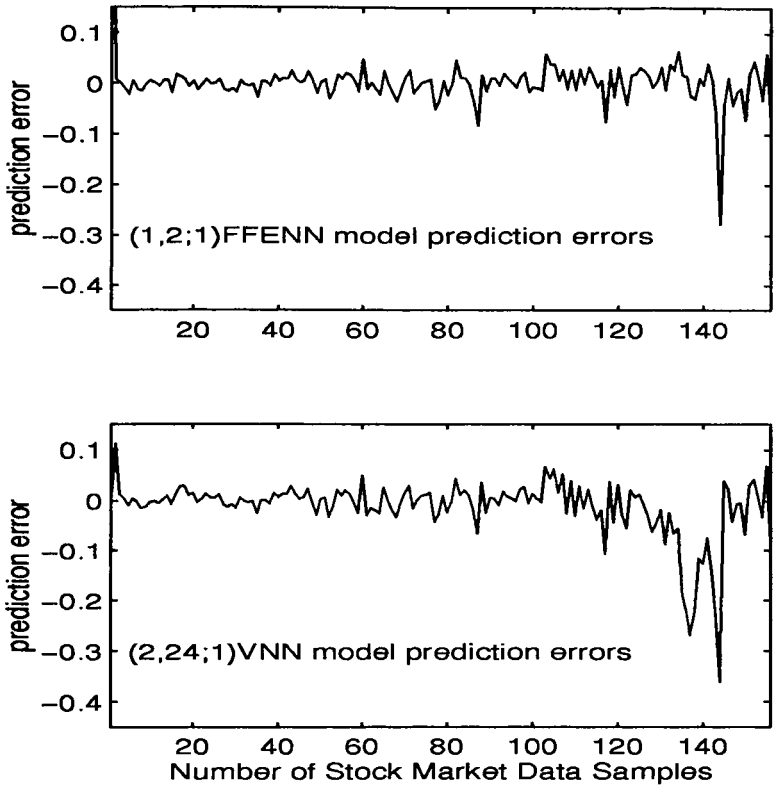

Figure 4.32: Comparison of prediction errors: 2 term FFENN one-step predictor versus the 24 term VNN one-step predictor models.

Note that the above stock market data was also modeled by NeuralWare [120] using a complex ten input MLP structure comprising two hidden layers and trained by the computationally expensive Back Propagation (BP) algorithm. Their MLP structure processed the 10 previous weeks prices to predict the next week's stock price. The first hidden layer comprised twenty nodes whereas the second hidden layer comprised ten nodes each possessing the sigmoidal activation function. After hundreds of presentations of the entire data set, the BP learning algorithm finally converged. The trained MLP model with fixed weights was then used to produce one-step (week) ahead predictions on the same entire data set. The performance of the complex MLP model was no better than that achieved by the new 2−term FFENN predictor model illustrated in equation 4.32 above. Note

that the use of a first order FFENN predictor model in performing highly efficient one-step predictions on the real stock market data comes as no surprise, as many researchers have actually shown that most of the reported highly complex non-linear one-step predictor models of stock market data can be easily outperformed by a simple first order linear predictor which simply substitutes the current stock market value for the next step's (week's, month's *etc*) stock market value [22] [19]. The above FFENN predictor model which can be seen from equation 4.32 to employ a linearly weighted value of the current stock market value together with a higher order non-linear sigmoidal shaped basis function of it, (the $sin(.)$ function which, in fact utilizes an infinite series of the odd-power expansion of the current input as discussed in section 4.2.3), in order to more effectively predict the next week's value.

Modeling of real Sunspots

In this application, real sunspot data collected annually over the years 1700-1987 [137] were modeled using the the FFENN. The sunspots, often larger in diameter than the earth, are dark blotches on the sun. They were first observed around 1610, shortly after the invention of the telescope [48]. Yearly averages have been recorded since 1700 and are plotted shown in Figure 4.33. Note that the sunspot series exhibits a maxima approximately every 7 to 15 years. The underlying mechanism for sunspot appearances is not exactly known, and the sunspot series has served as a benchmark, particularly in the statistics literature for time-series modeling [115] [59], where the objective is to generate a single-step prediction based on past observations [42]. The sunspot series was first studied by Yule [199] using a linear Auto-Regressive (AR) model. In a recent evaluation of different models, Priestley [137] favoured the Threshold AR (TAR) model of Tong and Lim [95]. The maximum order of their TAR model was fixed to 20 (corresponding to a total of 43 parameters), but their optimal model used information from the previous 12 years.

The sunspots have to-date also been modeled using a variety of non-linear

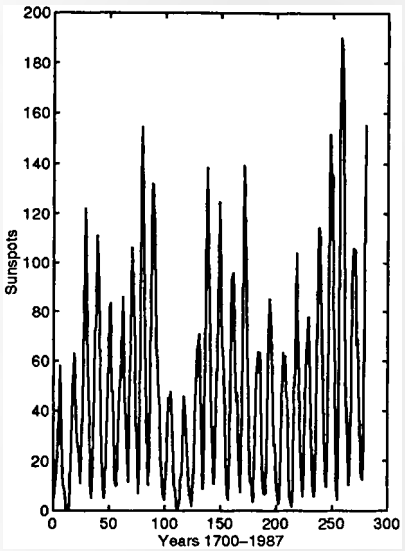

Figure 4.33: Annual Sunspot Time Series Data over the years 1700-1987.

complex neural network based predictor models [115] [42]. More recently, McDonnel and Wagen [42] employed a Transversal (feedforward) single hidden layered Perceptron network model comprising a total of 14 parameters, which evolved from parent recurrent IIR perceptron after a highly computationally expensive optimisation process incorporating a complex multi-agent stochastic search. McDonnel *et al* also devised other Recurrent network models which gave inferior performance relative to their feedforward model. Previously, Weigend, Rumelhart and Huberman [115] employed a fully connected 12−input and 8−sigmoidal hidden unit (12,8;1)MLP structure comprising a total of 113 parameters. This (12,8;1)MLP model was then pruned using a weight elimination technique to yield an optimal (12,3;1)MLP structure comprising a total of 43 parameters. Svaver *et al* [59] employed the Optimal Brain Damage (OBD) method of Le Cun *et al* [107] to generate a pruned network with 5 inputs that were not fully-connected to 3 hidden units in a two-layer feedforward neural network [42].

We now investigate a new FFENN based approach for modeling of the sunspot time series. Consistent with Tong [95], Weigend *et al* [115], Svaver *et al* [59] and

FFENN-RLS			
(1,8;1)	(2,20;1)	(3,38;1)	(4,64;1)
0.0026(0.0189)	0.0016(0.0114)	0.0016(0.0117)	0.0018(0.0126)

Table 4.8: Training Performance Comparison: *arv* and training set MSE (in parenthesis) measures of various FFENN based Single Step Predictors on the Sunspot series for years 1700-1920.

McDonnel *et al* [42], we use the sunspot time series from 1700 through 1920 for training, and the data from 1921 to 1979 for evaluation of the prediction. The measure of performance employed for the evolved FFENN models, is also the same as used by the above authors, namely the *a*verage *r*elative *v*ariance (*arv*) measure which was discussed in an earlier application in section 4.3.1.

The performance attained by fitting each of the fully expanded (1,8;1)FFENN, (2,20;1)FFENN, (3,38;1)FFENN and (4,64;1)FFENN models on the training set is illustrated in Table 4.8. The results for all the fitted structures were obtained by use of the exponentionally weighted RLS algorithm with the forgetting factor λ selected to give the lowest training set MSE and $arv(train)$ values for each model. As can be seen from the Table 4.8, the fully expanded 2—input (2,20;1)FFENN structure gives the lowest training set MSE and $arv(train)$ values. The underlying structure of the (2,20;1)FFENN which evolved at the end of training is illustrated below:

$$\begin{aligned}\hat{y}(k) \;=\; & 1.159y(k-1) + 1.552y(k-2) + 0.276\sin(y(k-1)) \\ & -1.904\sin(2y(k-1)) + 1.515\sin(3y(k-1)) + 1.275\sin(y(k-2)) \\ & +1.023\sin(2y(k-2)) - 0.559\sin(3y(k-2)) + 0.081\cos(y(k-1)) \\ & -1.693\cos(2y(k-1)) + 0.045\cos(3y(k-1)) + 0.817\cos(y(k-2)) \\ & -1.255\cos(2y(k-2)) + 0.151\cos(3y(k-2)) \\ & -1.151y(k-1)\sin(y(k-2)) - 0.771y(k-1)\cos(y(k-2)) \\ & -2.854y(k-2)\sin(y(k-1)) - 4.06y(k-2)\cos(y(k-1)) \\ & +0.139y(k-1)y(k-2) + 1.922 \end{aligned} \tag{4.33}$$

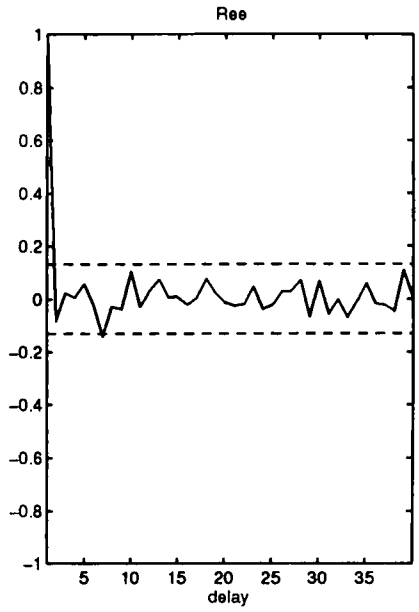

Figure 4.34: Auto-correlations of residuals $R_{e^2}(\tau)$ for fully expanded (unpruned) (2,20;1)FFENN model.

As can be seen from equation 4.33 above, the FFENN structure which represents a one-step (year) ahead predictor model for the sunspot time series, reveals the significant terms which are responsible for the short term (yearly) generation of the sunspots. Several Chi-squared tests were computed for the above identified (trained) (2,20;1)FFENN model, all of which were found to be satisfied. Figure 4.34 illustrates four typical chi-squared tests computed and Figure 4.35 shows the auto-correlation plot of the prediction errors. These tests confirm that the identified 20 term FFENN predictor model is an adequate representation of the sunspot time series.

To further optimise the size of the identified (2,20;1)FFENN model, the pruning strategy discussed in section 4.2.5 was employed. An optimal (2,14;1)FFENN model resulted which produced a training set MSE of 0.0116 and $arv = 0.0016$. Its output error auto-correlation and typical chi-squared based tests are illustrated in Figures 4.36 and 4.37 respectively, which can be seen to be satisfied. significantly large training set MSE of 0.0140 and an $arv = 0.0020$ and failed to

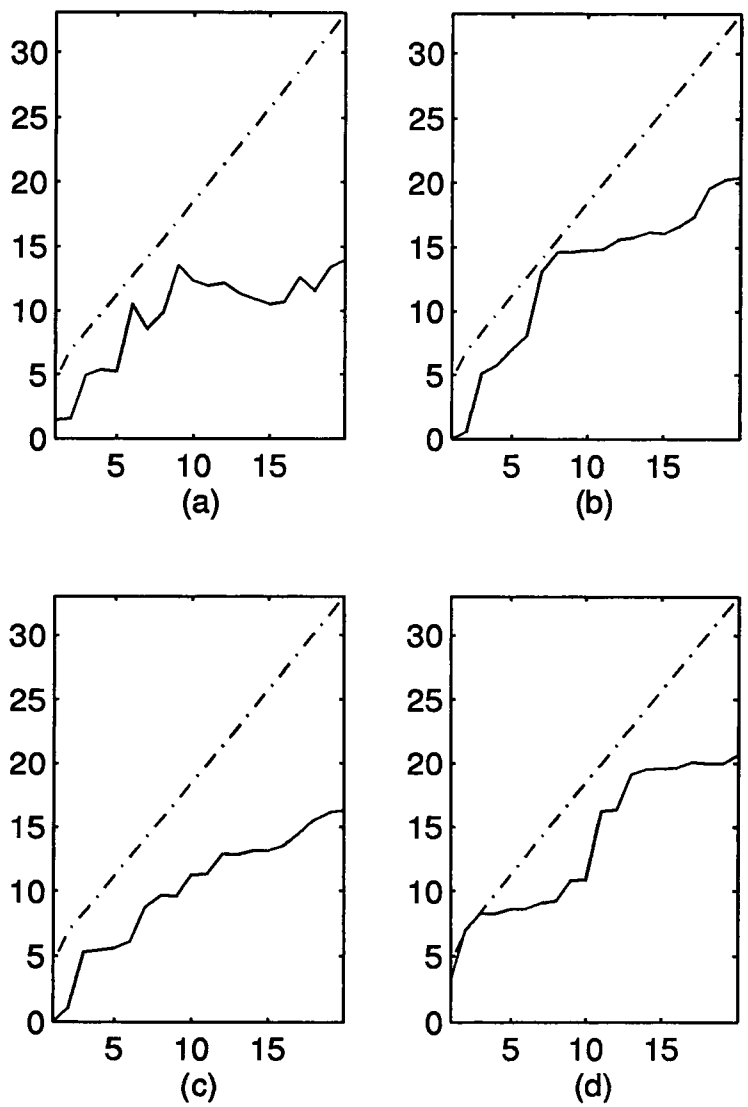

Figure 4.35: Chi-squared Statistic η Tests for the fully expanded (2,20;1)FFENN model: (a) $r(k) = e(k-1)$ (b) $r(k) = y(k-1)$ (c) $r(k) = y(k-1)^2$ (d) $r(k) = e(k-1)^2 y(k-1)^2$. The dotted lines indicate 95% confidence limits and the x-axis denote the lags.

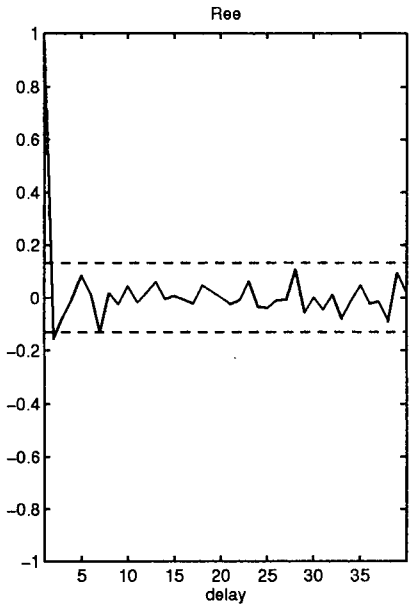

Figure 4.36: Auto-correlations of residuals $R_{e^2}(\tau)$ for final identified (2,14;1)FFENN model.

satisfy all the model validity tests. The structure of the trained optimal 14–term (2,14;1)FFENN model is illustrated below:

$$\begin{aligned}\hat{y}(k) = \; & 1.01y(k-1) + 1.155y(k-2) + 1.231\sin(y(k-2)) \\ & -2.06\sin(2y(k-1)) + 1.984\sin(2y(k-2)) - 1.139\cos(y(k-2)) \\ & -1.49\cos(2y(k-1)) + 1.539\sin(3y(k-1)) - 1.096\sin(3y(k-2)) \\ & -0.662y(k-1)\sin(y(k-2)) - 0.111y(k-1)\cos(y(k-2)) \\ & -2.778y(k-2)\sin(y(k-1)) - 3.809y(k-2)\cos(y(k-1)) \\ & +2.693 \end{aligned} \tag{4.34}$$

The optimal 14 term FFENN model illustrated in equation 4.34 above, was then used to perform one-step ahead predictions on the test set, and the $arv(predict)$ measure computed. Table 4.9 compares the average relative variance (arv) measure of the training set and Table 4.10 compares the arv measure of the test sets achieved by the the above 14–term (2,14;1)FFENN predictor model and other

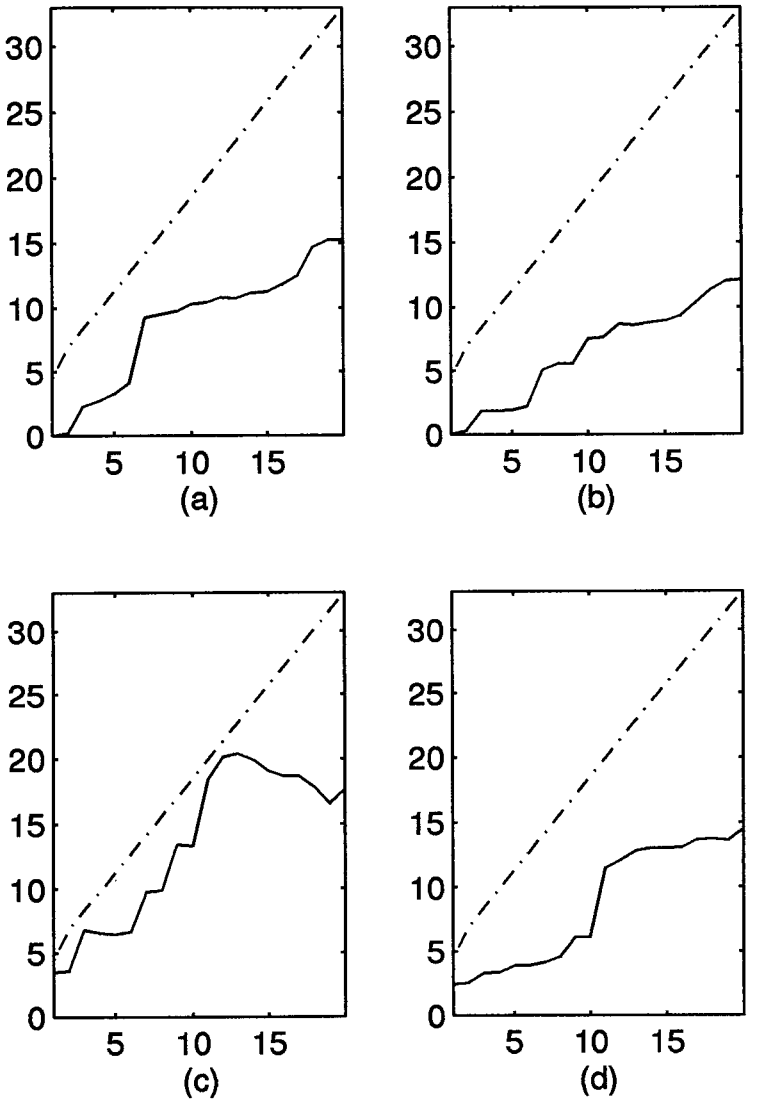

Figure 4.37: Chi-squared Statistic η Tests for final identified (2,14;1)FFENN model: (a) $r(k) = y(k-1)$ (b) $r(k) = y(k-1)^2$ (c) $r(k) = e(k-1)^2 y(k-1)$ (d) $r(k) = e(k-1)^2 y(k-1)^2$. The dotted lines indicate 95% confidence limits and the x-axis denote the lags.

FFENN	Tong *etal.*	Weigend *etal.*	Svaver *etal.*	McDonnel *etal.*
arv(*train*)	arv(*train*)	arv(*train*)	arv(*train*)	arv(*train*)
0.170	0.097	0.082	0.090	0.101

Table 4.9: *arv* of Training Set : Performance Comparison of various Models on the Sunspot series for years 1700-1920

	FFENN	Tong	Weigend	Svaver	McDonnel
no. of parameters	14	16	43	16	23
No. of Inputs	2	12	12	5	8
arv(*predict*)$_{1921-55}$	0.108	0.097	0.086	0.082	0.0971
arv(*predict*)$_{1956-79}$	0.18	0.28	0.35	0.35	0.37
arv(*predict*)$_{1921-79}$	0.1270	0.1733	0.2045	0.2031	0.2183

Table 4.10: *arv* of Test Sets: Performance Comparison of various Single Step Predictor Models on the Sunspot series for years 1921-1979.

recently reported predictor models, namely the TAR model reported by Tong *et al* [95], the pruned (12,3;1)MLP model reported by Weigend *et al* [115], the pruned two-layer feedforward model of Svaver *et al* [59], and the evolved feed-forward single-layered Perceptron model reported by McDonnel *et al* [42] which were all simulated under similar conditions.

As can be seen from Table 4.10, the 14−term FFENN single-step predictor model gives similar performance on the test set from 1921 − 1955 compared to other non-linear predictors of a greater computational complexity. However, it gives the best performance on the range of test set from 1956 − 1979, which is known to be atypical of the entire time-series [115] and the most difficult sunspot time-series set to model on account of its highly non-stationary nature. If the test *arv* measures are weighted and averaged across both the ranges of test sets 1921 − 1955 and 1956 − 79 for all the models, then the FFENN model still significantly outperforms all the other more complex models reported to date by least a 26% lower test *arv* measure, as shown in Table 4.10. The actual sunspot time series data and the FFENN model's one-step predictions are shown

superimposed in Figure 4.38, along with the prediction errors. Another important point to note is that the FFENN model uses only two past observations to make its single step predictions, whereas all other models required information from at least the last 5 sunspot observations. This shows that non-linear predictor models of smaller degrees of freedom can be more effective in the short-term prediction of the sunspot data than corresponding non-linear predictors of larger degrees of freedom. Note that additionally, an examination of the evolved FFENN predictor model illustrated in equation 4.34 above, can also provide highly useful insights into the physics of the underlying sunspot time series generating mechanism. Recently, the above sunspot data has also been modeled by Chen [197] using an RBF network based predictor model, which also gave similar performance to that achieved by the new FFENN model reported in this chapter, but at the expense of significantly greater computational complexity requirements.

Note that two-year and three year ahead predictor models based on the FFENN can also be easily developed, as previously demonstrated for the case of modeling of the chaotic Henon and Mackey-Glass time series. This was not attempted for this case as there were no comparative results available from other researchers for a performance comparison. The actual periodogram of the annual sunspot data and the periodogram estimated by the optimal 14 term FFENN one-step predictor model are shown superimposed in Figure 4.39, which can be seen to be closely matched. Also note that unlike other complex sunspot predictor models reported to date, an examination of the FFENN one-step predictor model illustrated in equation 4.34 gives highly useful insights into the physics of the underlying annual sunspot generating mechanism.

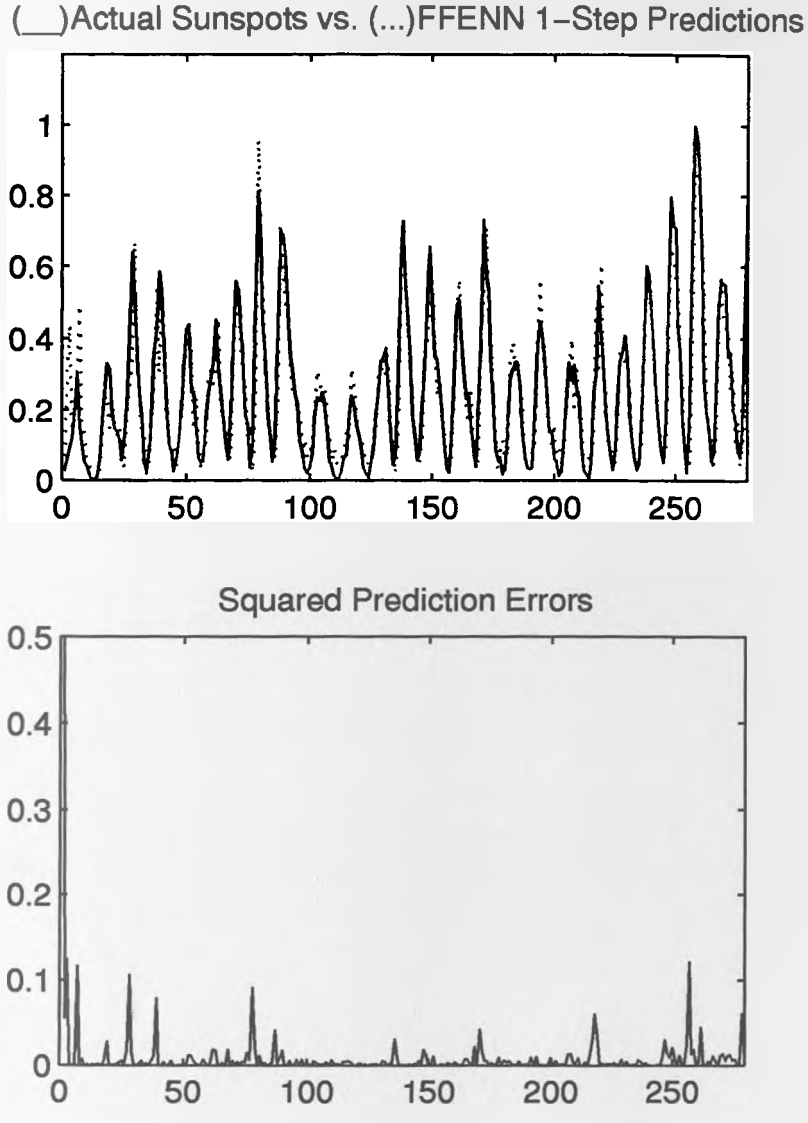

Figure 4.38: Upper Plot: Actual sunspot time series (years 1700-1987) versus FFENN model one-step predictions. Lower Plot: Prediction errors squared.

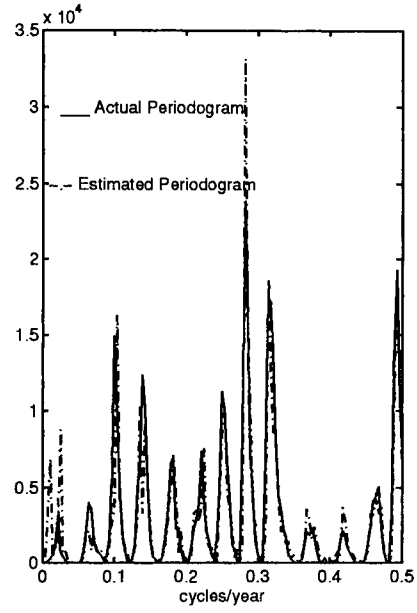

Figure 4.39: Comparison of the FFENN predictor model estimated Periodogram with Actual.

4.4 Conclusions

In this chapter, a new Feedforward Artificial Neural Network (FANN) architecture has been presented termed the FFENN, and its learning algorithm has been derived. New basis functions have been proposed for the FFENN hidden layer's functional expansion model which emulate other universal approximators namely, the squashing type sigmoidal shaped activation functions employed in the MLP network, Gaussain and multi-quadratic shaped activation functions employed in the RBF network, and polynomial subset outer-product terms employed in the VNN. The new FFENN structure employing the proposed terms in its hidden layer is essentially a hybrid neural network incorporating to a variable extent, the rich modeling capabilities of the conventional MLP, RBF and VNN structures. A general design strategy has been proposed for specifying the type and number of basis functions within the hidden layer of the FFENN for an arbitrary

number of inputs. The non-linear approximation ability of the FFEN resulting from use of the non-linear basis functions has also been discussed. A least squares based exponentionally weighted recursive algorithm has also been derived for adapting the FFENN output layer weights. A general pruning strategy based on an iterative pruning-re-training scheme coupled with model validation tests has also been proposed for the new structure. It is based on a simple heuristic that assesses the relevance of the functional terms in the input hidden layer, according to the magnitude of their weights. Functions (or hidden neurons) with small weights tend to contribute less to the overall computation of the FFENN output, and thus are promising pruning candidates. The pruned network then needs to be retrained to achieve the desired performance prior to the next pruning step. The shrinking network size in fact can make the subsequent training relatively faster. This simple pruning strategy coupled with model validity tests has been shown in this chapter through the use of numerous case studies, to consistently result in parsimonious FFENN predictor models of complex non-linear dynamical systems. As illustrated in the various simulation case studies, the linear-in-the-parameters FFENN predictor models can also additionally provide highly useful insights into the physical composition of the underlying non-linear dynamical systems. Note also that for the range of simulated and real-world time series processes considered in this chapter, upto a maximum of fourth order FFENN predictor models have been found to be sufficient for capturing the underlying non-linear dynamical systems representations; and shown to consistently outperform other more complex non-linear ANN based predictors both in terms of relative computational requirements and non-linear modeling capability. The relative contributions of the proposed non-linear basis functions responsible for the superior FFENN performance, were also illustrated in the various case studies. Several other non-linear and chaotic time series processes have also been highly efficiently modeled by the FFENN based predictor models, which have not been included in the thesis, including the Van-Der-Pol Oscillator [22], the chaotic Lorenz Map [41] and real currency exchange rates.

In the next chapter, a new Recurrent ANN structure termed the Recurrent Functionally Expanded Neural Network (RFENN) is developed and applied to the above task of modeling both simulated and real-world non-linear time series processes. The FFENN and RFENN structures are also applied to the task of real-time adaptive non-linear prediction of real-world non-stationary signals.

Chapter 5

New Recurrent Functionally Expanded Neural Network For Non-Linear Dynamical System Modeling Applications

5.1 Introduction

It was shown in chapter 2 that for the identification of *general* non-linear dynamic systems such as output error or NARMAX models, recurrent structures are more appropriate than the conventional feedforward neural network based structures [38].A major problem with the conventional RNN structures however, as discussed in chapter 2, is the high computational complexity associated with their learning algorithms such as the widely used RTRL. In section 5.2 of this chapter, a new Recurrent Functionally Expanded Neural Network (RFENN) structure [176] [175] is presented and its associated learning algorithm derived. Key structural and learning computational complexity comparisons are made between the new RFENN architecture and the conventional RNN structure. A pruning strategy for the RFENN is also presented.

In section 5.3, various case studies using a simulated output-error process, simulated chaotic time series and real world noisy non-linear time series processes are used to compare the performance of the RFENN with the FFENN and other recently reported feedforward ANN and RNN predictor models [52].

Finally in section 5.4, the application of the new FFENN and RFENN structures to on-line non-linear adaptive prediction of non-stationary signals is investigated. A new hybrid RFENN-FIR adaptive structure [175] comprising the non-linear RFENN subsection feeding into a linear FIR subsection is also developed. Real world laser time series data and an actual speech signal are used as two case studies to compare the adaptive modeling performance of the new structures with the conventionally employed linear filtering approaches.

5.2 The new Recurrent Functionally Expanded Neural Network (RFENN) structure

The Multiple Input Multiple Output (MIMO) RFENN structure illustrated in Figure 5.1 is basically a two-layer Feedforward FENN employing local output feedback. The input single hidden layer comprises a functional expander which performs a non-linear transformation that maps the input space onto a new larger dimensional non-linear hidden space. The approximation ability of the RFENN relies mainly on the type of the functional expansion model employed; for which the actual choice of functions and a general design strategy has been presented in section 4.2.1 of chapter 4.

The output layer of the RFENN comprises a set of linear combiners corresponding to the number of desired outputs. Each RFENN output is the result of summation of the weighted values of the functionally expanded input terms and its own past values. Note that each RFENN output layer node has feedback from the weighted past values of its own output only, not from the outputs of other nodes. In this context, the RFENN employs local output feedback similar

to Frasconi *et al*'s architecture [49], as opposed to global output feedback employed in the fully recurrent structures such as the Real-Time Recurrent Network (RTRN) [118] discussed in chapter 2.

5.2.1 Derivation of RFENN's learning algorithm: The Real-Time Recursive Update (RTRU) Algorithm

The weights for a general MIMO RFENN are updated as follows:

The j-th output of the RFENN (for a [1xN] functional expansion of the n inputs and feedback of M_j past values of the j-th output) can be written as:

$$y_j(k) = \sum_{i=1}^{M_j} a_{ij} y_j(k-i) + \sum_{l=1}^{N} b_{lj} f_l(k) \text{ for } j = 1, \ldots, m \text{ outputs}$$

where a_{ij} and b_{lj} are the feedback and feedforward weights respectively for the j-th output, $f_l(k)$ $l = 1, \ldots, N$ are the functionally expanded input terms which transform the input space R^n of n inputs $(x_1(k), \ldots, x_n(k))$onto a new non-linear hidden space of increased dimension R^N; and $y_j(k-i)$, $i = 1, \ldots, M_j$ are the M_j past values of the j-th output $y_j(k)$. The above RFENN is said to be recurrent of order M_j for the j-th output.

Defining at time k:

$$\Theta_j(k) = [a_{1j}(k) \ldots a_{M_j j}(k) \;\; b_{1j}(k) \ldots b_{Nj}(k)]^T$$

and

$$X_j(k) = [y_j(k-1) \ldots y_j(k-M_j) \;\; f_1(k) \ldots f_N(k)]^T$$

where ^{T} denotes transpose, gives:

$$y_j(k) = \Theta_j^T(k) X_j(k) \text{ for } j = 1, \ldots, m \text{ outputs} \tag{5.1}$$

$\Theta_j(k)$ is now chosen to minimise the Mean Squared Error (MSE) cost function J defined as follows:

$$J = (1/2) E(e_j(k)^2)$$

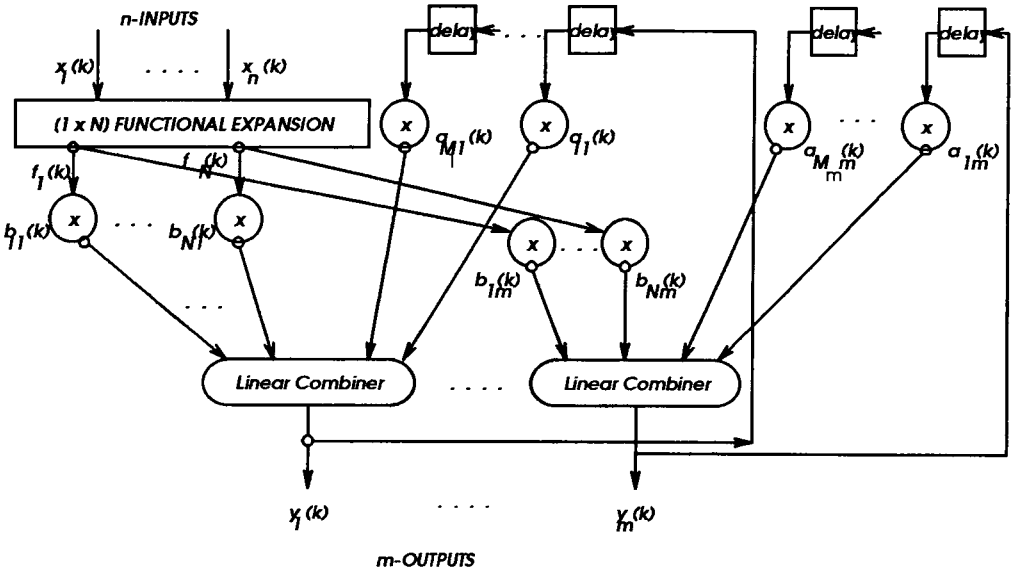

Figure 5.1: The Recurrent Functionally Expanded Neural Network (RFENN)

where $E(.)$ is the expectation operator and $e_j(k)$ is the j-th output error defined by:

$$e_j(k) = d_j(k) - y_j(k)$$

with $d_j(k)$ being the j-th desired output (which is available during the RFENN training mode).

The optimal value for $\Theta_j(k) = \Theta_{j(opt)}$ for the j-th output is found when

$$\frac{\partial J}{\partial \Theta_j(k)} = 0$$

Thus,

$$\frac{\partial J}{\partial \Theta_j(k)} = \frac{1}{2} E\left(\frac{\partial e_j^2}{\partial \Theta_j(k)}\right) = E\left(e_j(k)\frac{\partial e_j(k)}{\partial \Theta_j(k)}\right) = 0 \quad \text{for } \Theta_j(k) = \Theta_{j(opt)}$$

giving

$$\frac{\partial J}{\partial \theta_{ij}(k)} = -E\left((d_j(k) - y_j(k))\frac{\partial y_j(k)}{\partial \theta_{ij}(k)}\right) = 0 \quad \text{for } i = 1, \ldots, (M_j + N)$$

The last expression can be decomposed into: (assuming $a_{ij}(k)$ and $b_{lj}(k)$ are independent and change slowly with time)

$$\frac{\partial J}{\partial a_{ij}(k)} = E\left[e_j(k)\left(y_j(k-i) + \sum_{p=1}^{M_j} a_{pj}(k)\frac{\partial y_j(k-p)}{\partial a_{pj}(k-p)}\right)\right] = 0 \quad \text{for } i = 1, \ldots, M_j$$

$$\frac{\partial J}{\partial b_{lj}(k)} = E\left[e_j(k)\left(f_l(k) + \sum_{i=1}^{M_j} a_{ij}(k)\frac{\partial y_j(k-i)}{\partial b_{lj}(k-i)}\right)\right] = 0 \quad \text{for } l = 1, \ldots, N$$

From the above two gradient expressions, a recursive estimator for the j-th output gradient with respect to all its feedforward and feedback weight coefficients can be identified as:

$$\frac{\partial y_j(k)}{\partial \Theta_j(k)} = X_j(k) + \sum_{i=1}^{M_j} a_{ij}(k)\frac{\partial y_j(k-i)}{\partial \Theta_j(k-i)} \quad \text{for } j = 1, \ldots, m \text{ outputs} \qquad (5.2)$$

where the recursive gradient estimator for each output is the following column vector:

$$\frac{\partial y_j(k)}{\partial \Theta_j(k)} = \left[\frac{\partial y_j(k)}{\partial a_{1j}(k)}, \ldots, \frac{\partial y_j(k)}{\partial a_{M_j j}(k)}, \frac{\partial y_j(k)}{\partial b_{1j}(k)}, \ldots, \frac{\partial y_j(k)}{\partial b_{Nj}(k)}\right]^T$$

The gradient expressions derived above are exact and gradient search iterative techniques can now be used to approximate $\Theta_{j(opt)}$. Assuming the instantaneous gradient estimate:

$$\frac{\partial J}{\partial \Theta_j(k)} = E\left(e_j(k)\frac{\partial y_j(k)}{\partial \Theta_j(k)}\right) = \left(e_j(k)\frac{\partial y_j(k)}{\partial \Theta_j(k)}\right) \text{ for } j = 1, \ldots, m \text{ outputs}$$

a Real-Time Recursive Update (RTRU) algorithm for a MIMO $(n, N; m, (M_1, \ldots, M_m))$RFENN (with n and m denoting the number of inputs and outputs respectively and $M_1, \ldots, M_m$ representing the past M_j values of each of the $y_j(k-i)$ $i = 1, \ldots, M_j$ $j = 1, \ldots, m$ outputs fed back) can now be written as follows:

(1) Compute the RFENN's m outputs $y_j(k)$, for $j = 1, \ldots, m$, using Equation 5.1 above.

(2) Compute the m output gradients' estimates using Equation 5.2 above.

(3) Update the RFENN weight vector for each output using a recursive Gauss-Newton update (which is known to converge to at least a local minimum of an error surface [181] [58]) as follows:

$$\Theta_j(k+1) = \Theta_j(k) + P_j(k+1)\Delta_j(k+1) \text{ for } j = 1, \ldots, m \tag{5.3}$$

where $\Delta_j(k+1)$ is chosen to be a smoothed recursive estimator for the output gradient [181], and is updated as:

$$\Delta_j(k+1) = \gamma_m \Delta_j(k) + \gamma_g e_j(k+1)\frac{\partial y_j(k+1)}{\partial \Theta_j(k+1)} \tag{5.4}$$

where γ_g and γ_m are the adaptive gain and momentum (smoothing) parameters respectively; and $P_j(k+1)$ is the inverse of the Hessian matrix $R_j(k+1) = (\partial y_j(k+1)/\partial \Theta_j(k+1))(\partial y_j(k+1)/\partial \Theta_j(k+1))^T$, which it recursively approximates as [110] (for each of the $j = 1, \ldots, m$ outputs):

$$P_j(k+1) = \frac{1}{\lambda}\left(P_j(k) - \frac{P_j(k)(\partial y_j(k+1)/\partial \Theta_j(k+1))(\partial y_j(k+1)/\partial \Theta_j(k+1))^T P_j(k)}{\lambda + (\partial y_j(k+1)/\partial \Theta_j(k+1))^T P_j(k)(\partial y_j(k+1)/\partial \Theta_j(k+1))}\right) \tag{5.5}$$

where λ is the forgetting factor, which for adaptive applications $\in (0,1)$ [181]. Note that for the case of the functionally expanded input terms $f_l(k)\ \ l = 1,\ldots,N$ comprising just the network inputs $x_i(k)\ \ l = 1,\ldots,n$, that is for no non-linear transformation of the inputs, the extended vector $X_j(k)$ becomes

$$X_j(k) = [y_j(k-1)\ldots y_j(k-M_j)\ \ x_1(k)\ldots x_n(k)]^T$$

and the structure of the above RFENN will reduce to that of a conventional Infinite Impulse Response (IIR) filter; and the RTRU learning algorithm described above will become similar to the Recursive Prediction Error (RPE) algorithm used for updating the IIR weights [58].

Note that in order for the RFENN weight update equation 5.3 above to converge, it is important that the hessian matrix $R_j(k+1)$ always be positive definite (so that it is invertible), and that the poles of the RFENN system equation 5.1 always lie inside the unit circle (so that the RFEN network is stable). The poles of RFENN can be obtained by re-writing equation 5.1 as

$$y_j(k) = \sum_{i=1}^{M_j} a_{ij} y_j(k-i) + \sum_{l=1}^{N} b_{lj} f_l(k) \ \text{ for } j = 1,\ldots,m \text{ outputs}$$

Taking $z-$transforms on both sides readily gives (for each output)

$$Y_j(z) = [\frac{\sum_{l=1}^{N} b_{lj}}{1 - A_j(z)}] F_l(z) \ \text{ for } j = 1,\ldots,m \text{ outputs}$$

where $Y_j(z)$ is the z-transform of the $i-$th RFENN output; $F_l(z)$ denotes the z-transform of the $l-$th functionally expanded input term $f_l(k)$; and $A_j(z) = \sum_{i=1}^{M_j} a_{ij} z^{-i}$. A simple effective test of stability (for small M_j) would be check after each update of the RTRU algorithm that the sum of the RFENN feedback coefficients $\sum_{i=1}^{M_j} |a_{ij}|$ is less than 1 [58]. Other approaches to monitoring stability of pole-polynomials in conventional IIR filters have also been suggested which can also be readily employed for the RFENN, but they are either computationally expensive or non-robust. The problem is largely still an ongoing area of research [58].

Also note that if the matrix $P_j(k)$ is implemented in its basic form as shown in equation 5.5 above it can become fairly large, a phenomenon referred to as covariance explosive growth in conventional RPE algorithms. Covariance explosion can occur as a result of: the system parametrization not being unique [158], which is possible for the RFENN if two values of Θ_j result in the same input-output relationship defined by equation 5.1; Additionally, covariance explosion may also occur if the input signal excitation is poor [157]. Many numerical measures have been developed to overcome this problem [156]. A simple technique often used is the constant trace adjustment [182], in which $P_j(k)$ is adjusted in such a way that its trace remains constant, *viz*

$$\bar{P}_j(k+1) = \frac{1}{\lambda}\left(P_j(k) - \frac{P_j(k)(\partial y_j(k+1)/\partial\Theta_j(k+1))(\partial y_j(k+1)/\partial\Theta_j(k+1))^T P_j(k)}{\lambda + (\partial y_j(k+1)/\partial\Theta_j(k+1))^T P_j(k)(\partial y_j(k+1)/\partial\Theta_j(k+1))}\right)$$

$$P_j(k+1) = \frac{K_0}{\text{trace}[\bar{P}_j(k+1)]}\bar{P}_j(k+1) \text{ for } j = 1, \ldots, m \text{ outputs}$$

where K_0) is a arbitrary positive constant. The above will thus set an upper bound for the eigenvalues of the $P_j(k)$ matrix. A more sophisticated technique called exponential resetting and forgetting reported by Salgado and Goodwin *et al* [156] can also be employed.

5.2.2 Computational Considerations and Comparison with the conventional RTRL Algorithm

The Hessian matrix is incorporated in order to improve the RTRU algorithm's convergence rate but at the expense of an increase in the computational complexity. Otherwise, setting $P(k) = I$, the identity matrix yields the stochastic gradient LMS type algorithm as follows:
RFENN-stochastic gradient

$$\Theta(k+1) = \Theta(k) + \mu e(k+1)\frac{\partial y(k+1)}{\partial\Theta(k+1)}$$

where μ is the convergence factor. This algorithm requires only the order of $(M+N)$ operations but has a much slower convergence. Conventional RNNs are

trained by the RTRL algorithm [56] which is a temporal supervised learning algorithm based on an approximation to the method of steepest descent. The RTRL is *non-local* in the sense that each weight much have access to the complete weight matrix and the complete error vector. At any time k it requires consideration of a total of $(M^3 + NM^2)$ values of each dynamic output gradient computed with respect to all the feedback and feedforward weights [56]. On the contrary, the RFENN's learning algorithm RTRU is *local* and requires just $(M^2 + NM)$ operations for the estimation of each of its output gradient (assuming the same number of hidden layer nodes N in both the structures).

Note that the above reduction in the relative complexity of the RFENN's learning algorithms has been achieved by employing non-linear basis functions only at the input single hidden layer of the RFENN, whereas in conventional multi-layered RNN structures non-linear basis functions are employed at both the hidden and output layers. Therefore, the RFENN's structure is similar to the simple IIR type filter with a functionally expanded input layer, whereas on the contrary the RNNs are highly non-linear in the parameters Multi-Layered Perceptron (MLP) based structures with full interconnections (feedforward and feedback) between all nodes [56].

Note that the above RTRU algorithm can also be readily employed to train for example, RBF and VNN based Recurrent structures in which the RFENN hidden layer's functional expansion model $F(k)$ is replaced by Gaussian and Polynomial expansion models respectively.

5.2.3 Variations of the RFENN and Comparison with Other Recurrent Architectures

The above RFENN architecture is similar to a single-hidden layered RNN with output feedback which is restricted to be local (see chapter 2 section 2.3.1). It can also be considered to be similar to the Locally Recurrent Globally Feedforward (LRGF) network with local activation feedback (see chapter 2, section 2.3.2),

which is a special case of the LRGF with local synapse feedback such as the Back-Tsoi IIR MLP architecture [7], except that in the case of the RFENN, the output after feedback has not been passed through a non-linear activation function; and the RFENN is essentially a single IIR-MLP node with a non-linear input pre-processor. On the other hand, one may easily incorporate a non-linear activation function at the output of the RFENN after the feedback, without effecting its learning algorithm, giving the following new output:

$$z(k) = f(y(k)) \tag{5.6}$$

where $f(.)$ could be taken to be the $tanh(.)$ non-linearity.

Alternatively, by placing the above non-linear activation function prior to feeding back of the RFENN output $y(k)$, an architecture similar to the LRGF network with Local Output Feedback (such as the Frasconi-Gori-Soda architecture [49]) would be obtained. However, derivation of the corresponding new learning algorithm for the RFENN would incur a large computational cost similar to that of the RTRL.

Note that the architecture of the RFENN depicted in Figure 5.1 is in fact completely general [176] and encompasses Recurrent Neural Networks (RNNs) based on all other linear-in-the-parameters, feedforward neural networks such as the RBF and VNN structures, adapted to employ local output feedback thereby resulting in RVNN and RRBF structures.

5.2.4 Pruning Strategy for the RFENN

Unlike the feedforward ANNs, the pruning of conventional RNN structures has not been studied extensively to date. Recently, Giles *et al* [51] proposed a computationally expensive pruning strategy for conventional RNNs based on a pruning-retraining method, similar to the one we proposed for the FFENN structure in chapter 4, section 4.2.5. In order to obtain an optimally pruned RFENN predictor, we introduce the following pruning strategy:

Although the pruning strategy proposed for the FFENN in chapter 4 is general,

and can be readily applied to the RFENN, we propose that for any non-linear system modeling application, the FFENN be applied first to find a computationally efficient solution. If further performance improvement is required (or is expected, if the underlying system is known to be output-error type for example), then the computationally more expensive RFENN can be employed. This approach is in line with that recently proposed by Piche [150] who also suggested that feedforward predictors requiring computationally simpler learning algorithms should always be first tried to model complex non-linear dynamical systems, before computationally expensive recurrent predictor models are applied. Following this, pruned RFENN models are obtained as follows:
After applying the pruning strategy of section 4.2.5 to the FFENN, the resulting optimally pruned expansion model $F(k)$ can readily replace the hidden layer expansion model $F(k)$ of the RFENN structure. This approach will therefore save the pruning process being carried out a second time or exhaustively for the RFENN and will quickly reveal the benefits (if any) that result from the use of the recursive predictor model.

In the next section, the RFENN is employed in various simulation case studies involving modeling of both simulated and real-world non-linear time series processes, and its performance compared with the FFENN and other conventional ANN based predictor models.

5.3 Application of the RFENN to Non-linear Dynamical System Modeling and Comparative Performance Analysis

As discussed in chapter 2, Recurrent ANNs (RANNs) will be more effective than Feedforward ANNs (FANNs) in modeling certain classes of non-linear dynamical processes. Specifically, the RANNs will be able to model both the underlying poles and zeros of the non-linear systems, whereas the corresponding FANNs

whilst being able to readily model the underlying system poles, will need to be of a relatively large order for effectively modeling the system zeros. Alternatively stated, for the modeling of NARMA output-error type processes, RANN based non-linear predictors will be more effective than corresponding FANN based predictors. On the other hand, RANNs will not give any advantage over FANNs in the modeling of NAR type time series processes [52].

In the following sections 5.3.1 to 5.3.3, several case studies are carried out using both simulated and real data in order to investigate the modeling capability of the new RFENN based predictor model, and compare its performance with the FFENN and other neural network models.

5.3.1 CASE I: modeling of simulated NARMA output error type Non-linear Dynamical Systems

A bilinear NARMA (1,1) process [52] was generated as follows:

$$y(k) = e(k) + 0.5e(k-1)y(k-1)$$

where $e(k)$ represents a normally distributed noise sequence of zero mean and unity variance $N(0,1)$. A single step predictor based on a (1,8;1,1)RFENN structure of the form: $\hat{y}(k) = a_1\hat{y}(k-1) + \sum_{l=1}^{8} b_l f_l(k)$ was postulated comprising $(N = 8)$ functionally expanded terms $f_l(k)$ of the single past NARMA input $y(k-1)$ as illustrated in equation 4.1 of section 4.2.1. The single current output $(m = 1)$ of the RFENN (with its single past output $(M = 1)$ fed back) thus provided an estimate of the next step prediction on the NARMA time series.

The RFENN with the above expansion model was trained on the first 500 observations of a time series generated from the above NARMA(1,1) model using the RTRU learning algorithm with $\gamma_g = 1$, $\gamma_m = 0$, $\lambda = 0.999$ found to give the lowest training set MSE. Note that with these values of the RTRU parameters, the recursive Gauss-Newton update employed in the RTRU algorithm becomes similar to the exponentially weighted RLS update employed for the FFENN in chapter 4. To enable comparison with other recently reported neural network

		Connor *et al*		
Recurrent	Feedforward	Feedforward	Recurrent	Fully
RFENN	FFENN	(MLP) NAR	NARMA	Recurrent
RTRU (1,7;1,1)	RLS (1,7;1)	BP (1,7;1)	RTRL (1,5;1,1)	RTRL (1,5;1,1)
1.29(0.045)	1.43(0.05)	1.48(0.028)	1.28(0.025)	1.11(0.022)

Table 5.1: MSE (variance MSE) Performance Comparison of Single Step Predictors on the NARMA(1,1) Process for 10,000 samples.

structures [52], the trained RFENN with fixed weights was tested (that is, used to perform adaptive one-step ahead predictions) on another time series of 10,000 observations. The testing set Mean Squared Error is used as the basis of model comparison. The results are summarised in Table 5.1. Note that for a fair comparison, the dc bias term in the 8 term functional expansion model of the RFENN has not been counted, thus giving a (1,7;1,1)RFENN structure and a (1,7;1)FFENN. Standard errors of all the estimated MSEs are given in parenthesis. Note that the theoretical optimal NARMA predictor would have an MSE of 1.00 [52].

Table 5.1 shows that the RFENN outperforms all feedforward neural network structures including the FFENN(1,7;1) based NAR model trained by the RLS algorithm and the MLP(1,7;1) based NAR model (reported in [52]) trained using the Back Propagation (BP) algorithm, in modeling the NARMA(1,1) process; and also offers close performance to both the Recurrent NARMA(1,5;1,1) and fully Recurrent Neural Network (1,5;1,1) based structures (also reported in [52]) and trained by variations of the conventional RTRL. Note that the bilinear NARMA process has only been used as a case study; in general, the RFENN can be readily applied to model any NARMA output error type process.

	FFENN-RLS	RFENN-RTRU	McDonnel *et al*
No. parameters	20	21	17
arv (train MSE)	$0.0007(2.2 \times 10^{-5})$	$0.0009(2.9 \times 10^{-5})$	$0.0014(5 \times 10^{-5})$
arv (test MSE)	$0.00057(1.8 \times 10^{-5})$	$0.00038(1.2 \times 10^{-5})$	$0.0025(9 \times 10^{-5})$

Table 5.2: Test Performance Comparison: *arv* and training set MSE (in parenthesis) measures of the Single Step Non-linear fully expanded FFENN and RFENN based Predictors on the Mackey Glass Chaotic Time Series.

5.3.2 CASE II: modeling of simulated Chaotic Data: The Mackey Glass Time Series

As discussed in section 4.3.1, it is expected from the work of [42] that a recurrent predictor model would be more effective in modeling of the Mackey-Glass equation than a corresponding feedforward predictor. The Feedforward FENN predictor was shown to outperform the complex recurrent Perceptron predictor reported in [42]. In this section, a corresponding (2,20;1,1)RFENN one-step predictor was postulated and simulated under identical conditions on the same training set (with the RTRU parameters set to $\gamma_g = 1$, $\gamma_m = 0$, $\lambda = 0.995$ as they resulted in the lowest training set MSE) and test sets. The MSE and *arv* performance measures achieved by the (2,20;1,1)RFENN, the (2,20;1)FFENN and McDonnel *et al*'s Recurrent Perceptron based one-step predictors are illustrated in Table 5.2, on both the training and test sets.

Table 5.2 shows that, as expected from the work of [42], the RFENN model is indeed a more efficient one-step predictor compared to its feedforward version (namely the FFENN model) as well as the Recurrent Perceptron model of [42].

In order to obtain an optimally pruned RFENN predictor, the strategy proposed in section 5.2.4 was employed as follows: We devised a pruned (2,4;1,1)RFENN predictor comprising the 4 terms which constituted the optimal (2,4;1)FFENN predictor (that evolved from the pruning of the fully expanded (2,20;1)FFENN in

	(2,4;1)FFENN	(2,4;1,1)RFENN	McDonnel *et al*
No. parameters	4	5	17
arv (test MSE)	$0.0012(3.9 \times 10^{-5})$	$0.0006(1.96 \times 10^{-5})$	$0.0025(9 \times 10^{-5})$

Table 5.3: Test Performance Comparison: *arv* and test set MSE (in parenthesis) measures of the Single Step Non-linear pruned FFENN and RFENN Predictors on the Mackey Glass Chaotic Time Series.

section 4.3.1 of chapter4). The (2,4;1,1)RFENN produced lower *arv* and MSE values on the same training set compared to the (2,4;1)FFENN (with $arv = 0.0014$ and train MSE= 5×10^{-5} for the RFENN compared to $arv = 0.0020$ and train MSE= 6.1×10^{-5} produced by the corresponding FFENN).

The test MSE and *arv* performance measures of the (2,4;1,1)RFENN and its corresponding (2,4;1)FFENN one-step predictors on the test sets are illustrated in Table 5.3. As can be seen from Table 5.3, the 5 term RFENN predictor model significantly outperforms the equivalent 4 term FFENN based predictor and the more complex 17 term Recurrent Perceptron model of McDonnel *et al* [42], producing significantly lower MSE and *arv* values on the test data sets (exactly half of those produced by the FFENN and a quarter of those produced by the Recurrent Perceptron model. In order to test the optimality of the (2,4;1,1)RFENN predictor (and establish the effectiveness of this proposed pruning technique for the RFENN), a (2,3;1,1)RFENN predictor comprising the 3 terms constituting the further pruned (2,3;1)FFENN predictor of section 4.3.1, was postulated and simulated under identical conditions. As expected, similar to the case of the (2,3;1)FFENN, the pruned (2,3;1,1)RFENN predictor gave a significantly large MSE(= 0.0354) and *arv* (= 1.125) on the training set, thus confirming its inability to capture the underlying system's representation. Hence, for the modeling of the Mackey-Glass equation, the (2,4;1,1)RFENN model can be concluded to be the optimal predictor; whose final structure that evolved at the end of training is illustrated below:

$$\begin{aligned}\hat{y}(k) &= 2.349y(k-1) - 0.817\sin(y(k-2)) + 0.849y(k-1)\sin(y(k-2)) \\ &\quad -1.115y(k-2)\sin(y(k-1)) - 0.431\hat{y}(k-1) \qquad (5.7)\end{aligned}$$

Note that comparing the structure of the above (2,4;1,1)RFENN one-step predictor model (illustrated in the above equation 5.7) with the (2,4;1)FFENN one-step predictor model of equation 4.25, it can be seen that the final weight coefficients of the first four terms are almost equal, with the improvement in test MSE (and *arv*) performance attained by the above (2,4;1,1)RFENN predictor being due to the addition of the weighted recurrent term $(-0.431\hat{y}(k-1))$.

5.3.3 CASE III: modeling of Real World Data: Stock Market Data, Sunspots

The same weekly S & P stock market data that was effectively modeled by the (1,2;1)FFENN previously previously in chapter 4 (section 4.3.3) was employed for this case study. In chapter 4, the 2 term FFENN weekly predictor model was shown to outperform more complex tenth order linear and non-linear 24 term Volterra Neural Network (VNN) predictors. In this section, in order to investigate whether the use of recurrent predictors would give any performance improvements, a (1,8;1,1)RFENN one-step predictor model comprising the 8 term expansion model of the fully expanded (1,8;1,1)FFENN was devised. It was trained (by the RTRU algorithm) and tested on the same data sets. The resulting MSE measures achieved by the RFENN on the training and test sets are listed in Table 5.4, and can be seen to be almost identical to those achieved by use of the FFENN one-step predictor.

	(1,2;1)FFENN	(1,8;1,1)RFENN
Train Set MSE	0.0095	0.0096
Test Set MSE	0.0028	0.0029

Table 5.4: Performance Comparison of FFENN and RFENN Single Step Predictors on weekly S & P Stock Market Data.

Hence, it can be concluded that for this time series data, no additional benefit may be obtained by use of the RFENN predictor over the FFENN. From this result, it can therefore be deduced that the underlying weekly generator of this particular stock market data set can be best approximated by a NAR model.

The RFENN was also used to model the real sunspot data, that was previously efficiently modeled by the FFENN predictor model in chapter (section 4.3.3). However, the (2,20;1,1)RFENN predictor model, when simulated under identical conditions, was found to give inferior performance on the training and test sets. These findings are again consistent with those of McDonnel *et al* [42] who also found their evolved feedforward predictor model to perform better than other evolved recurrent predictor models.

In the next section, the use of the FFENN and RFENN predictors in performing adaptive (on-line) non-linear prediction of highly non-stationary signals is investigated.

5.4 FFENN and RFENN Structures for Adaptive (On-Line) Non-linear Prediction of Non-stationary Time Series

Many physical signals encountered in practice, for example speech signals, are generated by non-linear dynamical processes that exhibit two distinct characteristics namely: non-linearity and non-stationarity. Prediction is known to play a

key role in the modeling and coding of speech such as in the Adaptive Differential Pulse Code Modulation (ADPCM) of speech signals, where the error signal rather than the signal itself is transmitted. The non-stationary nature of speech has conventionally been dealt with by the use of adaptive filtering. However, traditional techniques employed to deal with the non-linear nature of speech to-date have mostly focused on the use of linear adaptive filtering schemes [62]. Only very recently, non-linear adaptive predictors based on the Volterra Neural Network (VNN) [85] and a novel Pipelined Recurrent Neural Network (PRNN) [90] have been proposed and shown to outperform the traditional linear adaptive schemes in the one-step prediction of speech signals. In this section, we investigate the use of the FFENN and RFENN structures as adaptive non-linear predictors and demonstrate their application to actual speech and laser data. Note that the case of laser data and speech signals have merely been chosen as case studies.

The traditional method of supervised learning employed by the conventional ANNs including the new FFENN and RFENN structures described above, is unsuitable for the adaptive non-linear prediction of non-stationary signals because of its off-line requirement. What is needed is a neural network that can learn on-line, that is, the network continuously learns to adapt to the statistical variations of the incoming time series whilst performing its filtering role at the same time. The FFENN and RFENN structures described above can be readily adapted to perform the above role by simply omitting their off-line pruning strategies and making the effective training period of the networks learning algorithms equal to *infinity*, that is, the training period is equated to the length of the incoming time series.

5.4.1 CASE IV: Adaptive Non-linear Prediction of Real NH_3 chaotic Laser Data

For the first case, a second order (2,20;1)FFENN was used to perform adaptive one-step predictions on real highly non-stationary, noisy, chaotic NH_3 laser data

FFENN-RLS	RFENN-RTRU	FIR-RLS
(2,20;1)	(2,20;1,2)	(20;1)
0.0034	0.0032	0.0327

Table 5.5: Performance Comparison of non-linear second order FFENN, RFENN and linear 20-th order FIR based Single Step Predictors on Laser Data.

acquired from the Santa Fe Institute Database [113]. The 1100 sample actual time series data and the corresponding FFENN one-step predictions obtained by use of the exponentially weighted RLS algorithm with $\lambda = 0.99$, are illustrated in Figure 5.2. The overall MSE was computed to equal 0.0034.

A recurrent (2,20;1,2)RFENN one-step predictor model was also devised and used to perform adaptive one-step predictions on the same data set, using the RTRU algorithm. Its performance is illustrated in Figure 5.3. The MSE figure for the 22 term RFENN was calculated to equal 0.0032, which is slightly better than that achieved by the corresponding 20 term FFENN predictor model.

For comparison a linear 20^{th} order FIR one-step predictor model was devised and used to perform adaptive one-step predictions on the same data, also using the RLS with $\lambda = 1$. The results for the linear predictor are illustrated in Figure 5.4, and its overall MSE performance measure was calculated to equal 0.0327, which is about 10 times greater than that achieved by the non-linear FFENN and RFENN predictors of similar complexity. The MSE performance measures of all the single-step predictors are listed in Table 5.5.

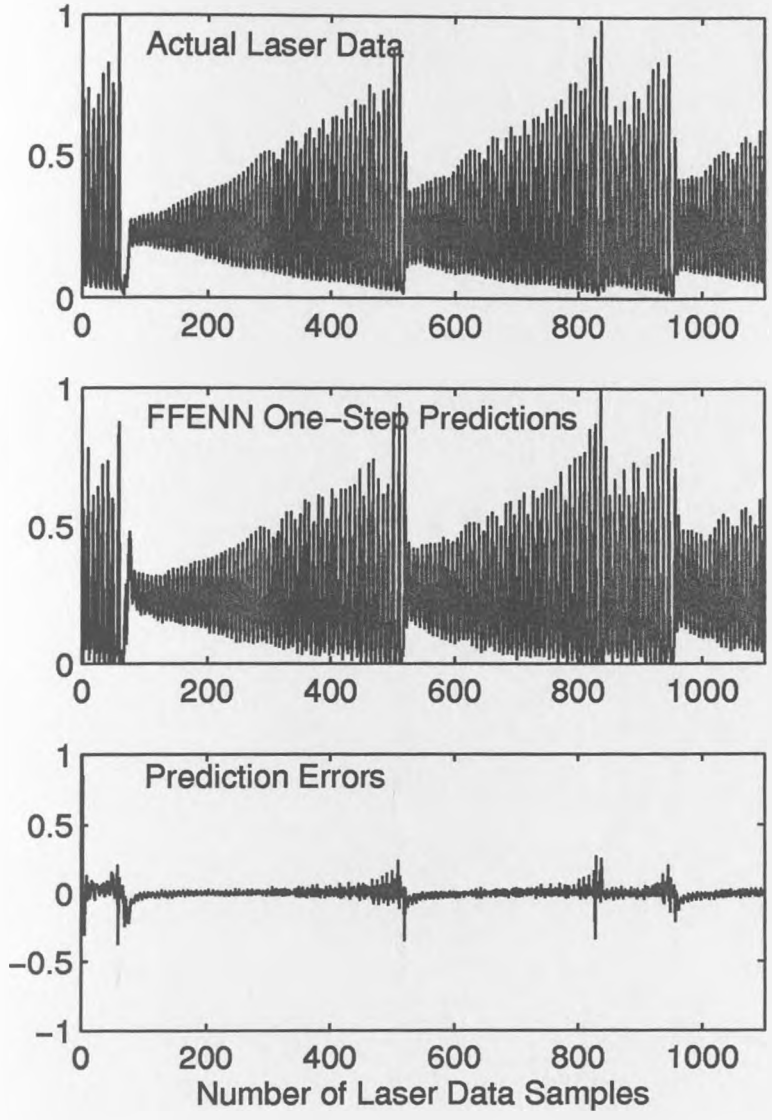

Figure 5.2: Performance of 2nd order (2,20;1)FFENN-RLS based one-step predictor model on the laser time series data.

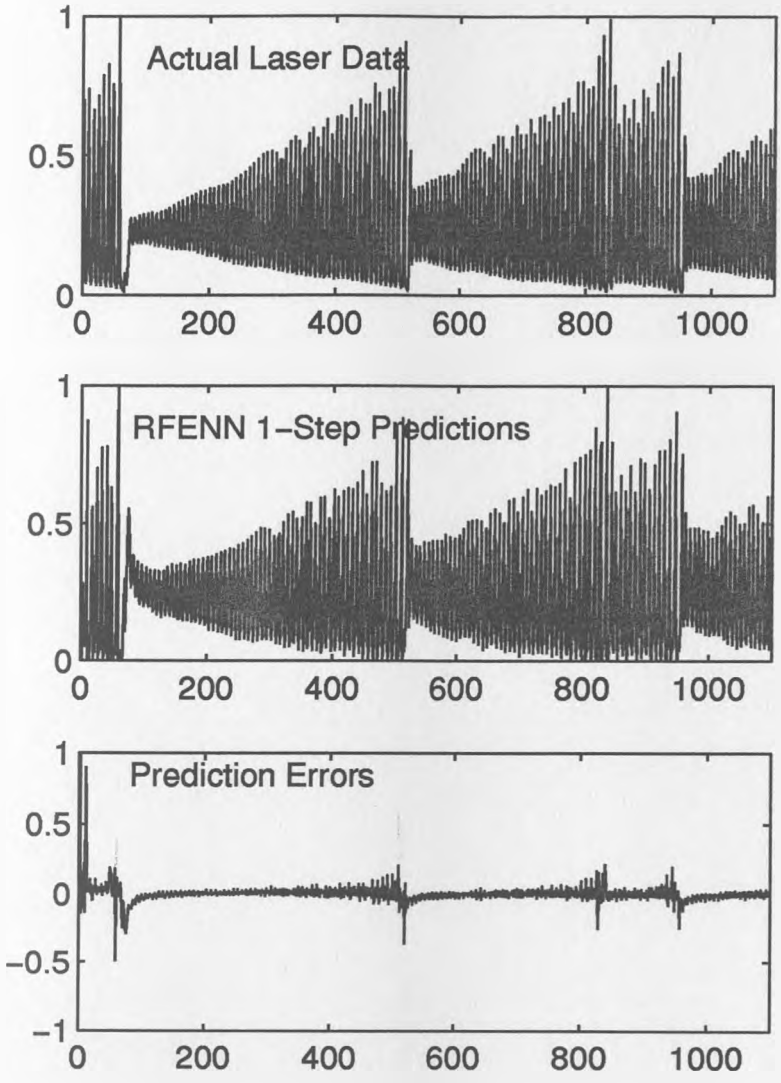

Figure 5.3: Performance of 2nd order (2,20;1,2)RFENN-RTRU based one-step predictor model on the laser time series data.

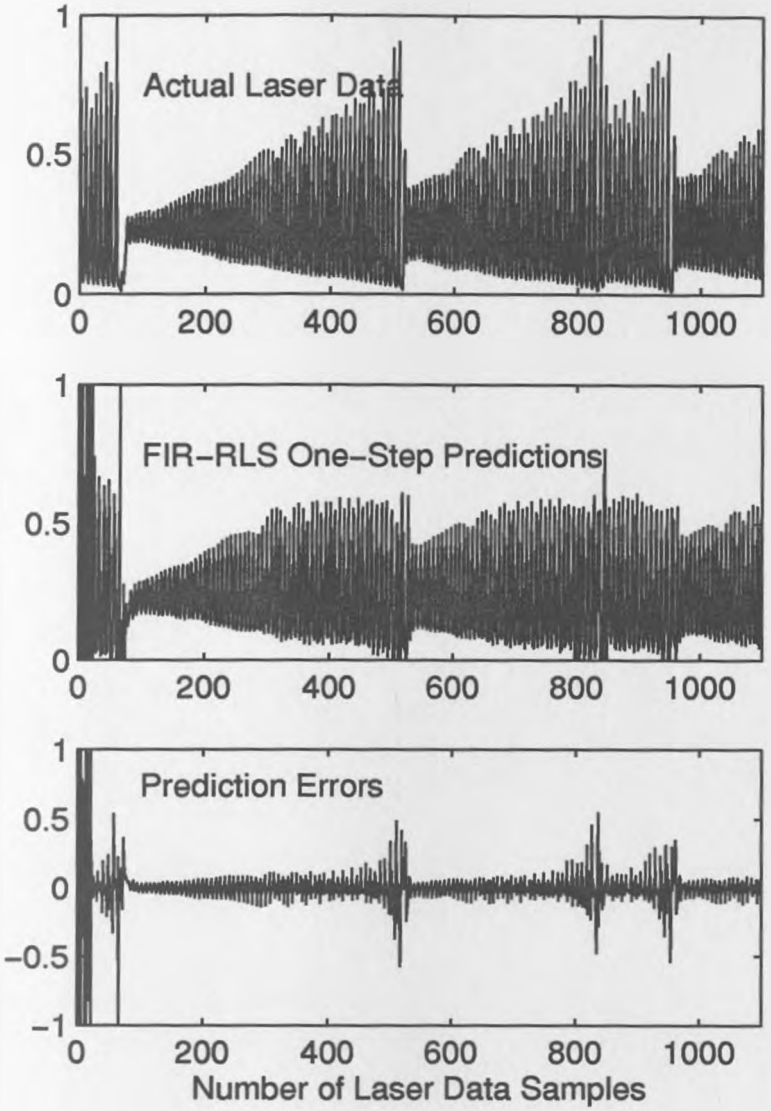

Figure 5.4: Performance of 20th order (20;1)FIR-RLS based one-step predictor model on the laser time series data.

5.4.2 CASE V: Adaptive Non-linear Prediction of a Real Speech Signal

For a real speech signal, the adaptive one-step predictions performed by a (2,20)FFENN, a(2,20;1,1)RFENN and a linear second order FIR-RLS are illustrated in Figures 5.5, 5.6 and 5.7 respectively. The overall MSEs computed for each predictor are compared in Table 5.6. As can be seen, both the non-linear FENN based predictors outperform the linear one (which can only capture the underlying linear dynamics of the speech generating process). Of the non-linear predictors, the RFENN offers the best performance showing that benefits may be obtained in the modeling of speech by the use of recurrent predictor models over feedforward non-linear predictors.

FFENN	RFENN	FIR
0.0264	0.0252	0.0310

Table 5.6: MSE Performance Comparison of Single-Step Predictors on real Speech Signal.

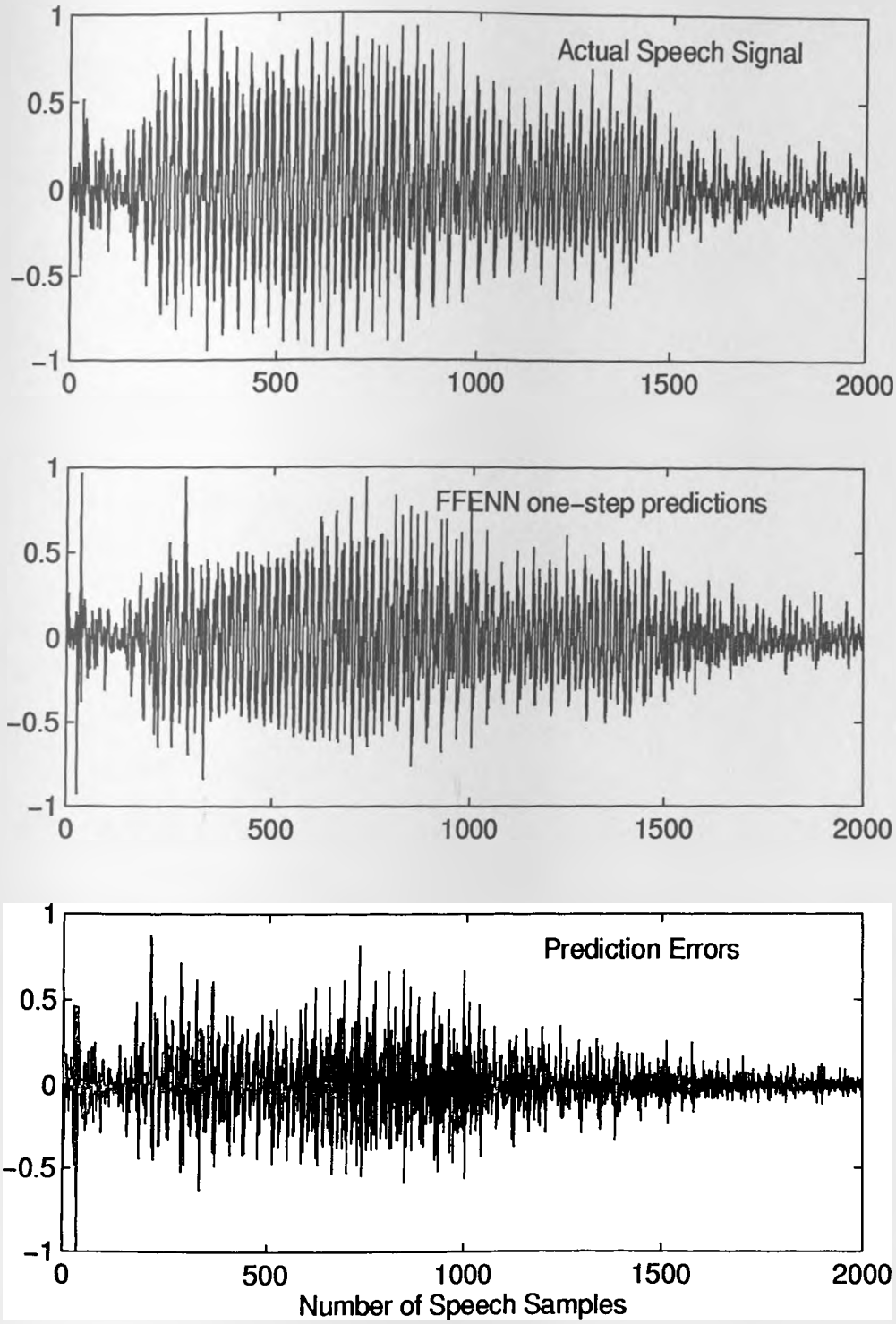

Figure 5.5: Single-Step Predictions of (2,20;1)FFENN predictor on real Speech Signal.

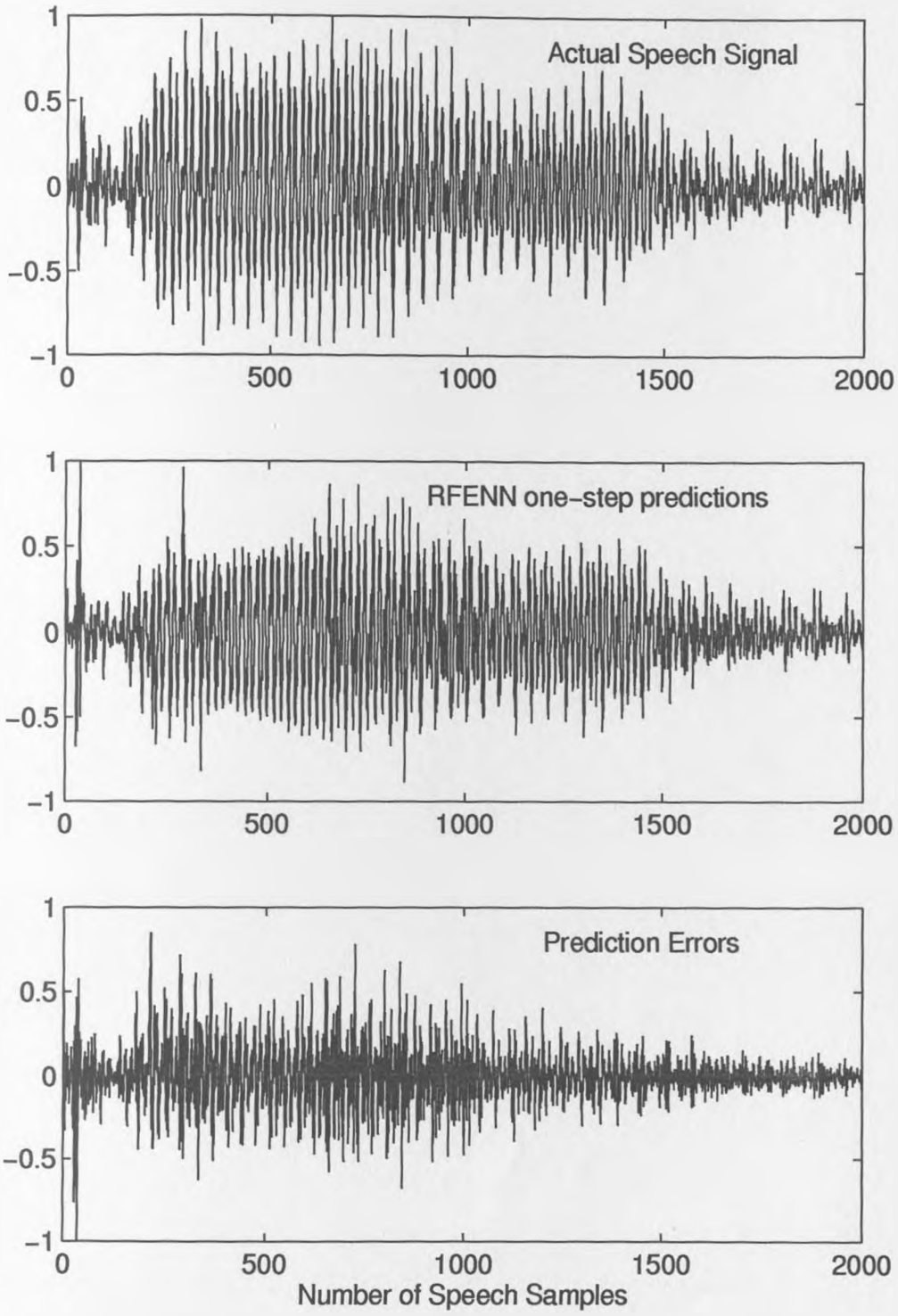

Figure 5.6: Single-Step Predictions of (2,20;1)RFENN predictor on real Speech Signal.

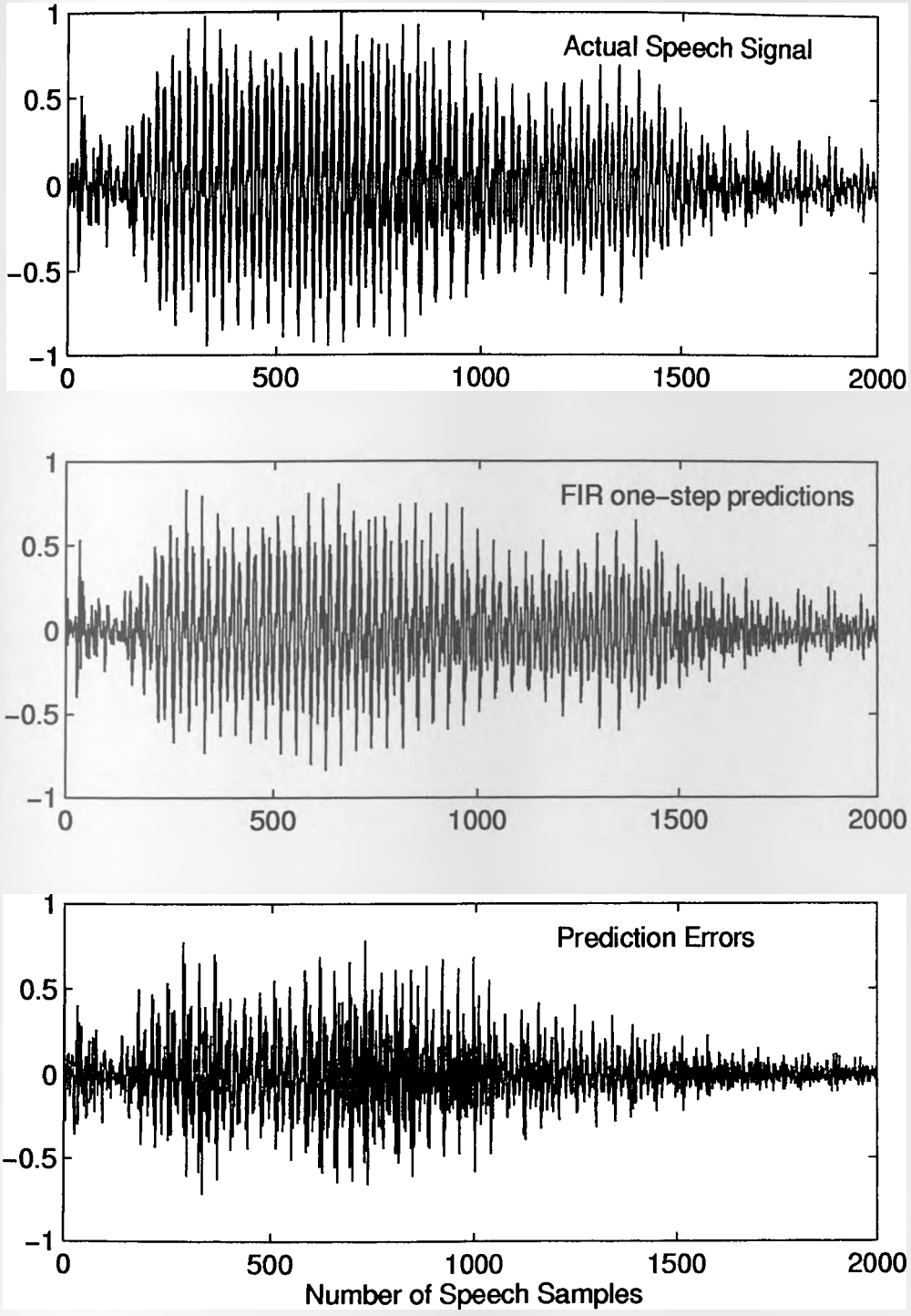

Figure 5.7: Single-Step Predictions of FIR predictor on real Speech Signal.

In [90], Haykin *et al* showed that additional benefits may be obtained in the modeling of speech signals, if a Recurrent Neural Network based predictor is used in conjunction with a linear FIR filter connected to its output. They argued that the prediction ability of the hybrid structure would be enhanced by:

- The RNN structure performing a non-linear mapping from the input space to an intermediate space with the aim of linearizing the input signal.
- And secondly, the FIR filter performing a linear mapping from the new intermediate space to the output space.

In order to investigate this, a new hybrid RFENN-FIR predictor [175] was devised comprising of a (2,20;1,1)RFENN subsection (trained as before using the RTRU), with its output feeding into a 16 tap linear FIR subsection, whose weights were updated by use of the LMS algorithm with a step size of 0.1. Note that the new structure adapted by simultaneous use of the RTRU and LMS algorithms, was found to significantly improve on the MSE performance measure attained by the stand alone RFENN predictor on the same speech signal, as can be seen in Table 5.7. The adaptive one-step predictions performed by the hybrid RFENN-FIR predictor model are illustrated in Figure 5.8. The optimal FIR tap and LMS step-size values were determined as per Haykin *et al*'s case, by trial and error.

FFENN	RFENN	RFENN-FIR	FIR
0.0264	0.0252	0.0170	0.0310

Table 5.7: MSE Performance Comparison of Single-Step Non-linear RFENN based Predictors on real Speech Signal.

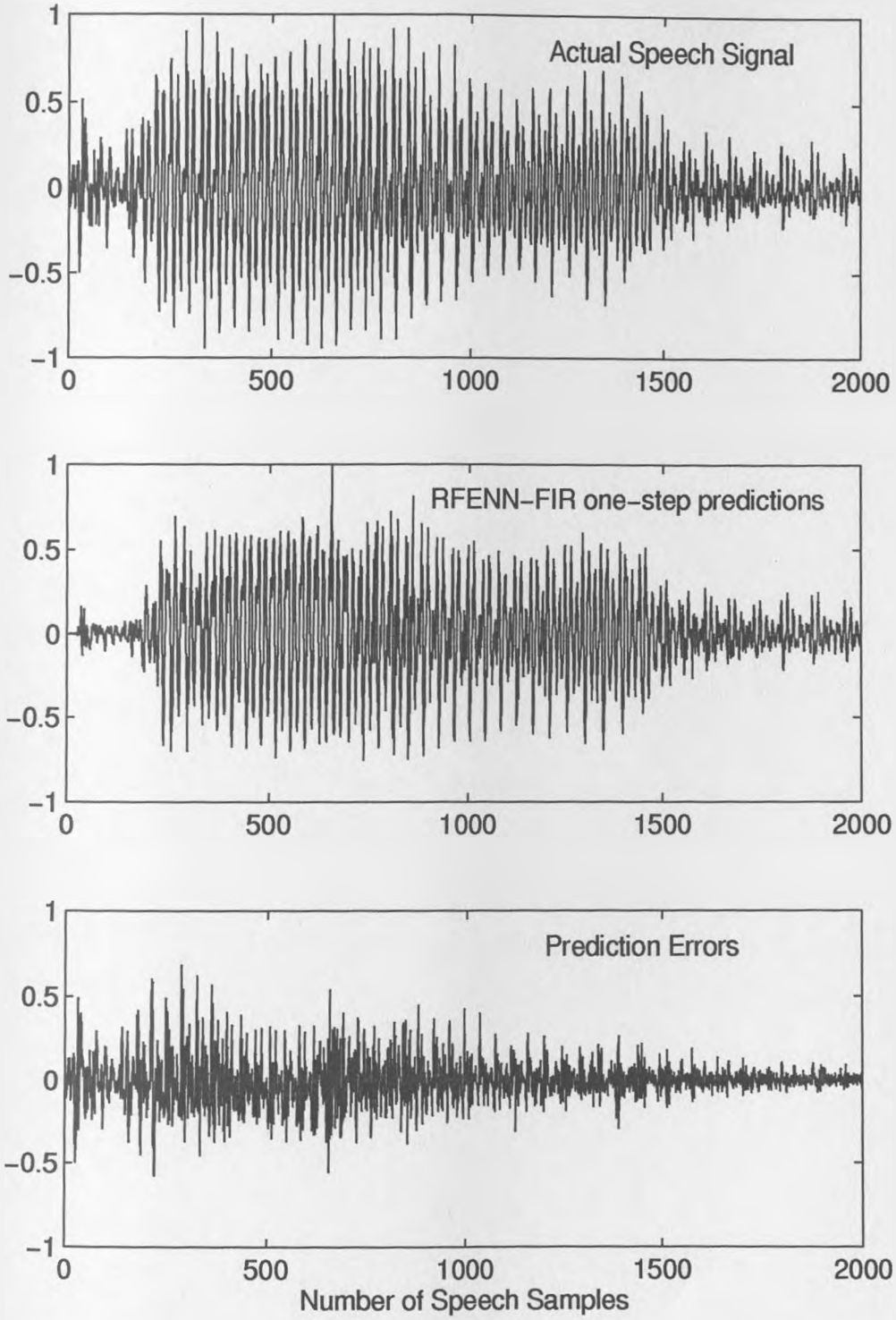

Figure 5.8: Single-Step Predictions of hybrid RFENN-FIR predictor on real Speech Signal.

5.5 Conclusions

A new RFENN structure has been presented based on the FFENN employing local output feedback, and its associated learning algorithm has been derived. Its structural and learning computational requirements have been shown to be significantly simpler compared to those of the conventional Recurrent Neural Networks (RNNs). The reduction in complexity has been achieved by employing non-linear basis functions (functional expansion model) only at its input single hidden layer, whereas in conventional multi-layered RNN structures non-linear basis functions are employed at both the hidden and output layers. This novel approach is in fact completely general and incorporates Recurrent Neural Networks based on all other linear-in-the-parameters, feedforward neural networks such as the RBF and VNN structures, which are adapted to employ local output feedback. This has resulted in a new class of computationally efficient RNNs [176]. A pruning strategy was also proposed for the RFENN structure in this chapter and shown to result in parsimonious RFENN predictor models of complex non-linear dynamical systems. It was shown to outperform the FFENN and other recently reported ANN structures in the modeling of certain types of non-linear dynamical systems.

The new FFENN and RFENN structures were also shown to be capable of performing adaptive (on-line) non-linear prediction of highly non-stationary signals more efficiently than the conventionally used linear adaptive predictors. A new hybrid RFENN-FIR adaptive structure was also devised comprising a non-linear RFENN subsection feeding into a linear FIR subsection. It was shown to outperform both the stand-alone FFENN and RFENN predictor models in the modeling of a real speech signal. Note that no stability problems were encountered with the RTRU learning algorithm of the RFENN in any of the simulation case studies carried out in this chapter. We conjecture that this could be the result of using a relatively small number of feedback samples M_j.

In the next chapter, new adaptive non-linear equalizers are presented and applied to two important digital communications applications.

Chapter 6

New Adaptive Non-linear Equalizers for Digital Communications Applications

6.1 Introduction

As discussed in chapter 3, the attraction of the neural network based equalizers is their ability to adaptively form the general optimal Bayesian solution for the symbol-decision structure and therefore to provide considerable performance gains over the conventional linear filter approach. A non-linear adaptive TE based on the conventional non-linear-in-the-parameters Functional-Link Neural Network (FLNN) has been previously shown [164] to offer superior speed of error convergence and BER performance characteristics compared to the LTE and the MLP and VNN based equalizer structures in the adaptive equalization of dispersive communications channels in the presence of additive uncorrelated noise. The superior performance of this Feedforward Functional-Link Equalizer (FFLE) is in fact due its ability to form highly non-linear decision regions [164] – an essential property for realization of the optimal equalization solution, as discussed in chapter 3. However, as with the MLP and VNN based equalizers, the main drawback

of the FFLE is the lack of a general design (specification) algorithm as discussed in chapter 3. Indeed it has been the lack of a formal design strategy that has led to the previous conjecture [164] that the size of the FFLE's hidden layer in theory, will grow exponentially with increasing input dimensions for $2-ary$ PAM signalling. In section 6.2 of this chapter, we first devise a general framework for the design of FFLE of an *arbitrary* order. The new design strategy is shown to give highly useful insights into the computational complexity requirements of the FFLE with its increasing input orders. The Extended Kalman Filter (EKF) algorithm is also applied to the FFLE in an attempt to enhance its speed of error convergence characteristics.

In section 6.3 of this chapter, two novel DFE architectures are presented [177] [178] [179], termed the Decision Feedback Functional-Link Equalizer with Unexpanded Feedback Terms (DFFLE-UFT) and a DFFLE with Expanded Feedback Terms (DFFLE-EFT). The DFFLE-UFT employs the non-linear FFLE as its feedforward filter, whereas the feedback filter is linear. In contrast, the DFFLE-EFT structure non-linearly combines both the equalizer input and decision feedback samples. Learning algorithms are presented for both the structures along with their design strategies. Key structural and computational complexity comparisons are made between the new DFFLE structures and the FFLE. The EKF algorithm is also applied to both the DFFLE structures to enhance their speed of error convergence characteristics. Pruning techniques for optimizing the sizes of the FFLE and DFFLE structures are also proposed in section 6.4.

Section 6.5 presents several case studies on a comparative performance evaluation of the new DFFLE structures with the FFLE and other recently reported non-linear TE and DFE structures. The optimal MLVA and the Bayesian TE are used as two benchmarks to assess the performance of the non-linear equalizer structures.

In the first application (section 6.5.1), the problem of equalizing linear and non-linear communications channels in the presence of ISI and both additive uncorrelated and additive correlated noise sequences is considered.

In the second application (section 6.5.2), a novel solution to the problem of co-channel interference suppression in digital communications systems is proposed based on the FFLE and DFFLE structures [179]. A comparative performance analysis is carried out between the FFLE, DFFLEs and other non-linear equalizers recently proposed for solving the same problem.

6.2 Analysis of conventional Feedforward Functional Link Equalizer (FFLE) Structure

As discussed in chapter 3, the general FFLE structure illustrated in Figure 6.1 consists of two layers:

- **Functional-Link Expander Input Layer:** It performs a non-linear transformation which maps the input space onto a new larger dimensional output space. Actual choice of functions was discussed in [31] [164]. The purpose of these functional-link terms was shown in [164] to be the extraction of certain useful features of the input data which render easier separation of the transmitted input classes. It was shown that a second order FFLE(2,M) expands the two tapped-input data (y_k, y_{k-1}) (that is, the noisy channel observations) into $M = 21$ functionally expanded terms $F(k) = f_0, f_1, \ldots, f_M$ for $2-ary$ PAM, as follows (assuming the bias input term $f_0 = 1$ is already included):

$$\begin{aligned} F(k) \;=\; & y_k, y_{k-1}, y_k y_{k-1}, \sin(n\pi y_k), \\ & \sin(n\pi y_{k-1}), \cos(n\pi y_k), \cos(n\pi y_{k-1}), \\ & y_k \sin(\pi y_{k-1}), y_k \cos(\pi y_{k-1}), \\ & y_{k-1} \sin(\pi y_k), y_{k-1} \cos(\pi y_k), \\ & sgn(y_k), sgn(y_{k-1}) \quad for \;\; n = 1, 2, 3 \end{aligned} \tag{6.1}$$

- **Output Layer comprising a Linear Combiner and a Sigmoidal Thresholder:** At this stage the weighted values of the enhanced input

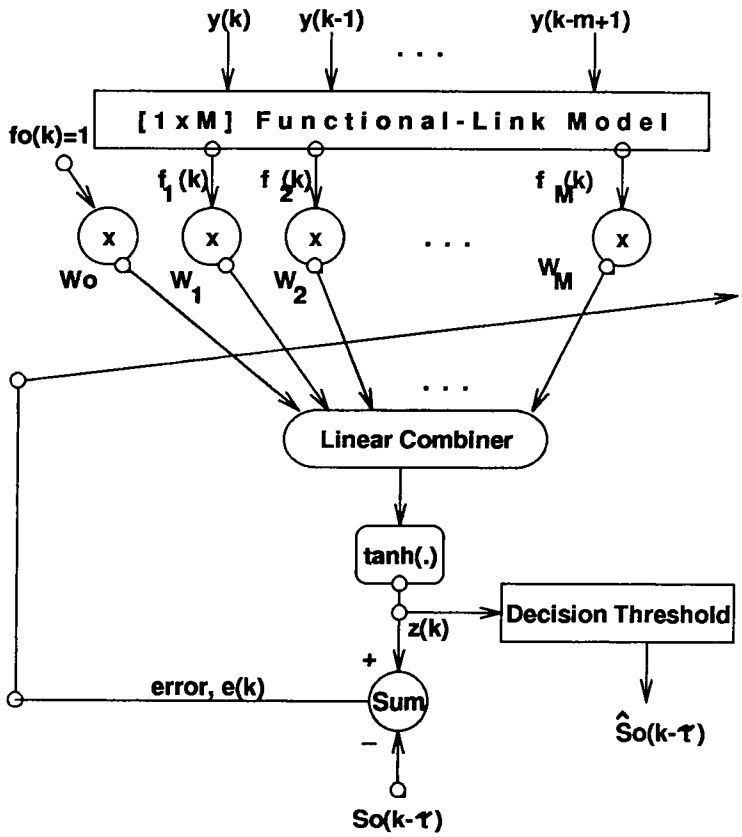

Figure 6.1: The conventional Feedforward Functional-Link Equalizer (FFLE)

functions $F(k)$ are linearly combined before being fed into a sigmoidal unit. The weights $w_i(k)$ where $i = 0, 1, \ldots, M$ are updated usually using the stochastic gradient Delta Rule (DR) algorithm which takes into account the output sigmoidal non-linearity as follows:

$$w_i(k+1) = w_i(k) + \mu e(k)(1 - z(k)^2) f_i(k)$$

Where the error $e(k) = s(k-\tau) - z(k)$ with $s(k-\tau)$ being the desired response and $z(k) = \tanh(\sum_{i=0}^{N} f_i(k) w_i(k))$ is the output from the sigmoidal function. Also note that μ is the step size.

The significance of the sigmoidal activation function employed in the output layer of the non-linear-in-the-parameters FFLE for the problem of channel equalization was not addressed in [164]. We present the following intuitive explanation: It has been shown for the case of the VNN based Adaptive Polynomial-Perceptron (APP) equalizer structure by Chen *et al* [183], that although the employment of the sigmoid necessitates the use of non-linear learning algorithms in the output layer (such as the DR which also forms the basis of the BP algorithm) and consequently complicates the error surface (that is, makes it multi-minima as discussed in Appendix A), it nevertheless provides an attractive feature for the problem of channel equalization, namely it improves the flexibility (classification ability) of the non-linear equalizer that is, enhances its ability to form highly non-linear decision boundaries – an essential property for realization of the optimal equalization solution, as discussed in chapter 3. As the structure of the APP is identical to that of the FFLE above, apart from the difference in the input functional-link expansion models employed in the two, with the APP employing a purely polynomial (Truncated Volterra Series) expansion of the equalizer inputs; therefore, the use of the sigmoid in the FFLE can also be concluded to be beneficial for the digital communications channel equalization problem.

6.2.1 Design Strategy for the FFLE

The trigonometric and joint-activation terms used in $F(k)$ (equation 6.1 above) for a second order (2,21)FFLE were shown in [164] through the use of a linear algebraic approach and various simulation case studies to be *useful* for $2-ary$ PAM signalling, that is, they assist the equalizer in classifying the input data with low output Mean Squared Error (MSE). It was shown that if the above functions, in particular the odd symmetry functions (such as the $sgn(y_k)$ and $\sin(\pi y_k)$)are augmented into the FFFLE structure to enhance the input representation, then the equalizer performs very well with near 100% classification and a very small output MSE (near 0) for equalizing both Minimum Phase and Non-Minimum Phase channels. This is because these functions were shown to extract certain useful features of the input data which render easier separation of the input classes. In this section, the useful expansion model illustrated in equation 6.1 is generalized for an arbitrary number of FFLE inputs, thereby yielding the following design strategy:

A general library of user defined functional-link expansion model $F(k) = f_0, \ldots, f_M$ for any (m,M)FFLE structure of feedforward order m (y_{k-i} $i = 0, \ldots, m-1$) and M expansion terms comprises the following functional-link terms for $2-ary$ PAM signalling (assuming the bias input term $f_0 = 1$ is already included):

1. original FFLE input samples, y_{k-i} for $i = 1, \ldots, m$ (resulting in m terms).

2. trigonometric functional expansion model comprising sum of the following components

   (a) orthogonal trigonometric functions of the m FFLE input samples, namely $\sin(a\pi y_{k-i})$ and $\cos(a\pi y_{k-i})$ for $a = 1, 2, 3$ (resulting in a total of $6m$ terms).

   (b) product of each FFLE input sample with the trigonometric sine and cosine functions of other FFLE input samples, that is, $y_{k-i}\sin(\pi y_{k-j})$

and $y_{k-i}\cos(\pi y_{k-j})$, for $i \neq j$, $i,j = 0,\ldots,m-1$ (resulting in a total of $2m(m-1)$ terms).

3. signum function of each FFLE input sample, $sgn(y_{k-i})$ (resulting in m terms)

4. outer-product (joint activation) expansion of the m FFLE inputs (resulting in $((P_2^m + P_3^m + , \ldots, + P_{m-1}^m) + 1)$ terms for m greater than two inputs, where $P_i^m = \frac{m!}{(m-i)!i!}$ where ! denotes factorial.

The above design strategy can best be illustrated by examples:

A single input (1,M)FFLE will comprise the following $M = 8$ terms for its expansion model:

$$\begin{aligned} F(k) &= y_k, \sin(n\pi y_k), \cos(n\pi y_k), sgn(y_k) \\ &\quad \text{for } n = 1,2,3 \end{aligned} \tag{6.2}$$

A second order (2,M)FFLE comprises the expansion model $F(k)$ listed in equation 1 above.

For a third order (3,M)FFLE, $F(k)$ comprises the following $M = 40$ terms:

$$\begin{aligned} F(k) = {} & y_k, y_{k-1}, y_{k-2}, y_k y_{k-1}, y_k y_{k-2}, y_{k-1} y_{k-2}, y_k y_{k-1} y_{k-2} \\ & \sin(n\pi y_k), \sin(n\pi y_{k-1}), \sin(n\pi y_{k-2}), \\ & \cos(n\pi y_k), \cos(n\pi y_{k-1}), \cos(n\pi y_{k-2}), \\ & y_k \sin(\pi y_{k-1}), y_k \sin(\pi y_{k-2}), y_k \cos(\pi y_{k-1}), \\ & y_k \cos(\pi y_{k-2}), y_{k-1} \sin(\pi y_k), y_{k-1} \sin(\pi y_{k-2}), \\ & y_{k-1} \cos(\pi y_k), y_{k-1} \cos(\pi y_{k-2}), y_{k-2} \sin(\pi y_k), \\ & y_{k-2} \sin(\pi y_{k-1}), y_{k-2} \cos(\pi y_k), y_{k-2} \cos(\pi y_{k-1}), \\ & sgn(y_k), sgn(y_{k-1}), sgn(y_{k-2}) \quad for \ n = 1,2,3 \end{aligned} \tag{6.3}$$

For a fourth order (4,M)FFLE, $F(k)$ similarly comprises the following $M = 67$ terms:

$$F(k) = y_k, y_{k-1}, y_{k-2}, y_{k-3}, y_k y_{k-1}, y_k y_{k-2}, y_k y_{k-2},$$

$$
\begin{aligned}
&y_{k-1}y_{k-2}, y_{k-1}y_{k-3}, y_{k-2}y_{k-3}, y_k y_{k-1}y_{k-2}y_{k-3} \\
&y_k y_{k-1}y_{k-2}, y_k y_{k-2}y_{k-3}, y_{k-1}y_{k-2}y_{k-3} \\
&\sin(n\pi y_k), \sin(n\pi y_{k-1}), \sin(n\pi y_{k-2}), \\
&\sin(n\pi y_{k-3}), \cos(n\pi y_k), \cos(n\pi y_{k-1}), \\
&\cos(n\pi y_{k-2}), \cos(n\pi y_{k-3}), \\
&y_k \sin(\pi y_{k-1}), y_k \sin(\pi y_{k-2}), y_k \sin(\pi y_{k-3}), \\
&y_k \cos(\pi y_{k-1}), y_k \cos(\pi y_{k-2}), y_k \cos(\pi y_{k-3}) \\
&y_{k-1} \sin(\pi y_k), y_{k-1} \sin(\pi y_{k-2}), y_{k-1} \sin(\pi y_{k-3}), \\
&y_{k-1} \cos(\pi y_k), y_{k-1} \cos(\pi y_{k-2}), y_{k-1} \cos(\pi y_{k-3}) \\
&y_{k-2} \sin(\pi y_k), y_{k-2} \sin(\pi y_{k-1}), y_{k-2} \sin(\pi y_{k-3}) \\
&y_{k-2} \cos(\pi y_k), y_{k-2} \cos(\pi y_{k-1}), y_{k-2} \cos(\pi y_{k-3}), \\
&y_{k-3} \sin(\pi y_k), y_{k-3} \sin(\pi y_{k-1}), y_{k-3} \sin(\pi y_{k-2}) \\
&y_{k-3} \cos(\pi y_k), y_{k-3} \cos(\pi y_{k-1}), y_{k-3} \cos(\pi y_{k-2}), \\
&sgn(y_k), sgn(y_{k-1}), sgn(y_{k-2}), sgn(y_{k-3}) \ \ for \ \ n = 1, 2, 3
\end{aligned}
\tag{6.4}
$$

Hence, a general Table 6.1 can be constructed relating the number of inputs n to the number of terms M in the input functional-link expansion model $F(k)$. From Table 6.1, it can be deduced that unlike the APP equalizer, an increase in the number of the inputs does not lead to an exponentional increase in the size of the FFLE expansion model.

Thus, the above proposed general design strategy has given a new insight into the computational requirements of the FFLE for an arbitrary number m of received noisy channel observations $(y_k, y_{k-1}, \ldots, y_{k-m+1})$. It is however, evident from the above Table 6.1 that even for a moderate number of channel observations (equalizer inputs), for example $m = 4$ requires a large number of corresponding functional-links ($M = 67$) for efficient equalization performance. This in turn, can degrade the FFLE's generalization capability, especially in high noise conditions. Hence, the development of alternative reduced complexity FFLE based architectures is highly desirable. In section 6.3 a novel approach for the DFE

is proposed based on the FFLE which is shown to, in addition to enhancing its equalization capability, result in a significant reduction in its relative computational complexity requirements.

m	M
1	8
2	21
3	40
4	67
5	106
6	165
7	260
8	423
m	$2m^2 + 5m + \sum_{i=1}^{m} P_i^m$

Table 6.1: General Design Strategy for (m,M)FFLE

In the next section, the Extended Kalman Filter (EKF) algorithm is proposed for enhancing the FFLE's speed of error convergence characteristics, which are crucial for tracking time-varying communication channels.

6.2.2 Application of the Extended Kalman Filter (EKF) to the FFLE

The non-linear Delta Rule used for updating the FFLE weights, being a stochastic gradient algorithm consequently suffers from the drawback of slow convergence. In this section, the Extended Kalman Filter [171] algorithm is proposed for enhancing the speed of error convergence of the FFLE. Defining

$$\mathbf{w} = [w_0 w_1 \dots w_M]^T$$

$$\mathbf{f} = [f_0 f_1 \dots f_M]$$

Hence, the FFLE output can be written as:

$$z(k) = \phi(\mathbf{fw})$$

where $\phi(.)$ is the sigmoidal activation function assumed to be the $tanh(.)$ function for the present case (which is obviously differentiable). The error signal $e(k)$ is defined as before:

$$e(k) = s(k-\tau) - z(k)$$

The above network can be represented as a non-linear dynamical system:

$$\mathbf{w}(k+1) = \mathbf{w}(k) \quad \text{state equation} \tag{6.5}$$

$$s(k-\tau) = \phi(\mathbf{fw}) + e(k) \quad \text{measurement equation} \tag{6.6}$$

The problem is to estimate $\mathbf{w}(k) \rightarrow \hat{\mathbf{w}}(k)$

Hence, the Taylor series expansion can be used to expand the measurement equation about the current weight estimate $\hat{\mathbf{w}}(k)$ thereby linearizing the activation function as follows [56]:

$$\phi(\mathbf{fw}) = [\frac{\delta\phi(\mathbf{fw})}{\delta\mathbf{w}}]_{\mathbf{w}=\hat{\mathbf{w}}(k)}\mathbf{w}(k) + [\phi(\mathbf{f}\hat{\mathbf{w}}) - [\frac{\delta\phi(\mathbf{fw})}{\delta\mathbf{w}}]_{\mathbf{w}=\hat{\mathbf{w}}(k)}\hat{\mathbf{w}}(k)] \tag{6.7}$$

The first term on the right hand side of the above equation 6.7 is the linear term, and the second term on the right hand side is termed the modeling error term. Now $\frac{\delta\phi(\mathbf{fw})}{\delta\mathbf{w}}$ needs to be determined. For $\phi(.)$ chosen to be the $tanh(.)$ activation function, the FFLE output becomes:

$$z(k) = tanh(\mathbf{fw}) = \frac{2}{1+e^{-\mathbf{fw}}} - 1 \tag{6.8}$$

Which gives:

$$\frac{\delta(z(k))}{\delta\mathbf{w}} = \frac{\delta\phi(\mathbf{fw})}{\delta\mathbf{w}} = \frac{\mathbf{f}2e^{-\mathbf{fw}}}{(1+e^{-\mathbf{fw}})^2} \tag{6.9}$$

Re-arranging equation 6.8 above gives:

$$\frac{1}{(1+e^{-\mathbf{fw}})^2} = \frac{4}{(1+z(k))^2}$$

Substituting into equation 6.9 above gives:

$$\frac{\delta\phi(\mathbf{fw})}{\delta\mathbf{w}} = \frac{\mathbf{f}e^{-\mathbf{fw}}(1+z(k))^2}{2} \tag{6.10}$$

Re-arranging equation 6.8 above again for $e^{-\mathbf{fw}}$ gives:

$$e^{-\mathbf{fw}} = \frac{1-z(k)}{1+z(k)}$$

Substituting into equation 6.10 above finally gives:

$$\frac{\delta\phi(\mathbf{fw})}{\delta\mathbf{w}} = \frac{\mathbf{f}(1-z(k)^2)}{2} \tag{6.11}$$

Now at $\mathbf{w} = \hat{\mathbf{w}}$, the FFLE output becomes

$$z(k) = \phi(\mathbf{f}\hat{\mathbf{w}}) = \hat{z}(k)$$

Thus, giving (for the case of the $tanh(.)$ sigmoidal non-linearity):

$$[\frac{\delta\phi(\mathbf{fw})}{\delta\mathbf{w}}]_{\mathbf{w}=\hat{\mathbf{w}}(k)} = \frac{\mathbf{f}(1-\hat{z}(k)^2)}{2} \tag{6.12}$$

Ignoring the modelling error term on the right hand side of equation 6.7 above, and substituting for the $\phi(\mathbf{fw})$ term from equation 6.7 into the measurement equation 6.6 above, gives a new measurement equation:

$$s(k-\tau) = [\frac{\delta\phi(\mathbf{fw})}{\delta\mathbf{w}}]_{\mathbf{w}=\hat{\mathbf{w}}(k)}\mathbf{w}(k) + e(k) \tag{6.13}$$

with the state equation as before:

$$\mathbf{w}(k+1) = \mathbf{w}(k) \quad \text{state equation} \tag{6.14}$$

Now the exponentially weighted RLS algorithm can be applied to the above system to estimate $\mathbf{w}(k)$ as follows:
Letting $\mathbf{q}(k) = \frac{\mathbf{f}(1-\hat{z}(k)^2)}{2}$, and $d(k) = s(k-\tau)$ gives the dynamical system:

$$\mathbf{w}(k+1) = \mathbf{w}(k) \tag{6.15}$$

$$d(k) = \mathbf{q}(k)\mathbf{w}(k) + e(k) \tag{6.16}$$

Defining the least squares cost function:

$$J(k) = \sum_{n=1}^{k} \lambda^{k-n} e^2(n)$$

where $(0 \leq \lambda \leq 1)$ is the exponentional forgetting factor, and $e(k) = d(k) - \mathbf{q}(k)\mathbf{w}(k)$ from the measurement equation 6.16 above. Hence, minimising $J(k)$ with respect to the weights $\mathbf{w}$ and equating the gradient to zero gives the following expression for the optimal weight vector:

$$\mathbf{w_{opt}}(k) = \mathbf{Q^{-1}}(k)\theta(k)$$

where $\mathbf{Q^{-1}}(k)$ is the inverse of the auto-correlation matrix $\mathbf{Q}(k)$ given by:

$$\mathbf{Q}(k) = \sum_{n=1}^{k} \lambda^{k-n} \mathbf{q}^T(n)\mathbf{q}(n)$$

and $\theta(k)$ is the cross-correlation matrix given by:

$$\theta(k) = \sum_{n=1}^{k} \lambda^{k-n} d(n)\mathbf{q}(n)$$

Following the same procedure as outlined for the FFENN learning algorithm in section 4.2.4 of chapter 4, the following set of recursive update equations for the FFLE can be easily derived: Letting $\mathbf{P}(k) = \mathbf{Q^{-1}}(k)$

$$\mathbf{r}(k) = \lambda^{-1}\mathbf{P}(k)\mathbf{q}^T(k) \tag{6.17}$$

$$\mathbf{o}(k) = \mathbf{r}(k)[1 + \mathbf{q}(k)\mathbf{r}(k)]^{-1} \tag{6.18}$$

$$\mathbf{w}(k+1) = \mathbf{w}(k) + \mathbf{o}(k)e(k) \tag{6.19}$$

$$\mathbf{P}(k+1) = \lambda^{-1}\mathbf{P}(k) - \mathbf{o}(k)\mathbf{r}^T(k) \tag{6.20}$$

In the next section, two novel approaches for the DFE are presented.

6.3 The New Decision Feedback Functional-Link Equalizers (DFFLEs)

As discussed in chapter 3, in the case of the conventional DFE, the feedforward filter is a Linear Transversal Equalizer (LTE) whose decision regions are always delimited by hyperplanes [172]. The linearity of these decision regions in the input signal space actually limits the performance of the conventional DFE [186]. The main advantage of a DFE over an LTE is its ability to better compensate severe amplitude distortion without increasing the noise level at its output [149].

Siu et al [185] applied the non-linear Multi-Layered Perceptron (MLP) structure to the DFE and illustrated its superior performance over the conventional DFE and the feedforward MLP both in terms of convergence times and tolerance to noise. They also show how the use of a correct decision feedback signal can totally eliminate the ISI from previously detected symbols with a reduced risk of error propagation. Chen *et al* [193] and more recently Theodoridis *et al* [160] also applied the Radial Basis Function (RBF) neural network to the DFE to enhance equalizer performance and reduce computational complexity.

In this section, two new approaches for the DFE are proposed, termed the Decision Feedback Functional Link Equalizer (DFFLE) with Unexpanded Feedback Terms (DFFLE-UFT) and the DFFLE with Expanded Feedback Terms (DFFLE-EFT); which are discussed below.

6.3.1 The DFFLE with Unexpanded Feedback Terms (DFFLE-UFT)): Structure and Learning Algorithm

Figure 6.2 shows a DFFLE equalizer with Unexpanded Feedback Terms (UFT) [177] [178]. As can be seen, this structure employs the non-linear FFLE as its feedforward filter, whereas the feedback filter is linear. The operation of the $(m,M;p)$DFFLE-UFT is based on the m most recent channel observations $(y_k,\ldots,y_{k-m+1})$ and its p past decisions $(\hat{s}_0(k-\tau-1),\ldots,\hat{s}_0(k-\tau-p))$, with

m and p being referred to as the equalizer feedforward and feedback orders respectively. The output of the DFFLE-UFT is thus given by:

$$z(k) = \tanh(\sum_{i=0}^{M} f_i(k)w_i(k) + \sum_{j=1}^{p} q_j(k)\hat{s}(k-\tau-j))$$

where $(w_i\ i = 0,\ldots,M)$ and $(q_j\ j = 1,\ldots,p)$ are the feedforward and feedback weights respectively. They are updated in a similar way to the weights of the FFLE, using the stochastic gradient Delta Rule as follows: Defining a new $[1 \times (M+p+1)]$ weight vector $\mathbf{u}(k)$ as:

$$\mathbf{u}(k) = [w_0(k)w_1(k)\ldots w_M(k)q_1(k)\ldots q_p(k)]^T$$

and a $[1 \times (M+p+1)]$ vector $\mathbf{f}(k)$ comprising of:

$$\mathbf{f}(k) = [f_0(k)f_1(k)\ldots f_M(k)\hat{s}(k-\tau-1)\ldots\hat{s}(k-\tau-p)]^T$$

results in DFFLE-UFT output:

$$z(k) = \mathbf{u}^T(k)\mathbf{f}(k)$$

The DFFLE-UFT weights are updated as:

$$u_i(k+1) = u_i(k) + \mu e(k)(1-z(k)^2)f_i(k) \quad \text{for } i = 0,1,\ldots,(M+p)$$

where as before, the error $e(k) = s(k-\tau) - z(k)$ with $s(k-\tau)$ being the desired response available during the training period, and μ is the convergence factor.

6.3.2 Design Strategy for the (DFFLE-UFT)

As the $(m,M;p)$DFFLE-UFT employs the FFLE as its feedforward filter, its design strategy is the same as that for the FFLE described in section 6.2.2, except that an additional p decision feedback terms $(\hat{s}(k-\tau-1),\ldots,\hat{s}(k-\tau-p))$ are augmented with the input functional-link expansion model $F(k)$. Table 6.2 relates the total number of inputs m and decision feedback samples p to the total number of coefficients in the weight vector (including the feedforward and feedback weights).

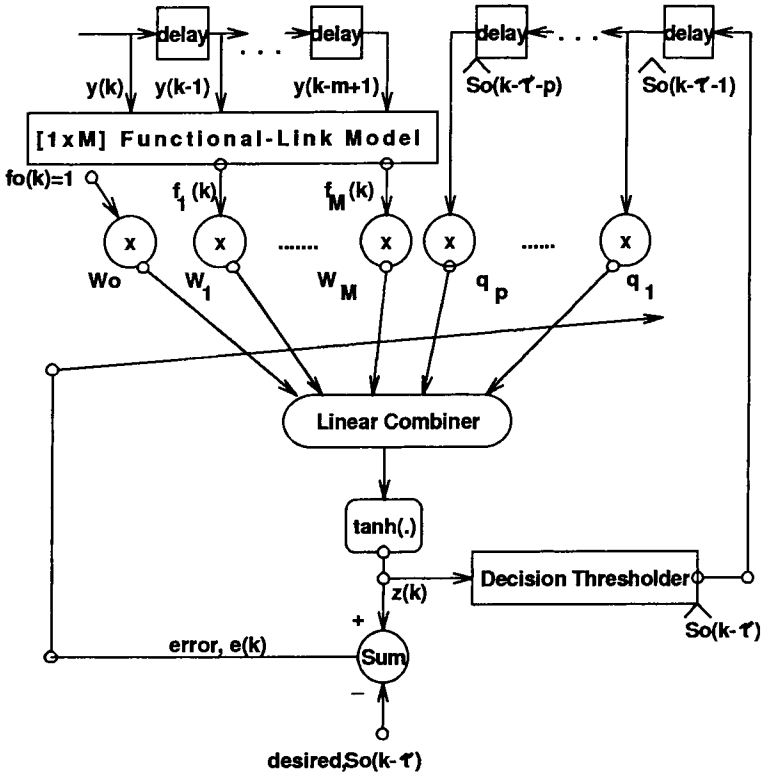

Figure 6.2: The Decision Feedback Functional-Link Equalizer with Unexpanded Feedback Terms (DFFLE-UFT)

(m,p)	$(M+p)$
$(m+p)$	$(2m^2+5m+\sum_{i=1}^{m} P_i^m)+p$

Table 6.2: General Design Strategy for $(m,M;p)$DFFLE-UFT

By comparing Table 6.2 with Table 6.1, it can be concluded that for the same total number of equalizer taps, the DFFLE-UFT has a dramatically lower computational requirement. For example, for a total number of equalizer (feedforward plus feedback) taps = 4, a fourth order $(4,M)$FFLE structure requires $(M = 67)$ terms, whereas the equivalent fourth order $(2,M;2)$DFFLE-UFT requires only $(M = 21 + 2 = 23)$ terms (which are exactly $p = 2$ terms more than those required by a feedforward second order $(2, 21)$FFLE).

In the following section, a new DFFLE structure with Expanded Feedback Terms (EFT) is presented which is also shown to possess a simpler computational requirement relative to the FFLE by virtue of a new functional-link expansion model.

6.3.3 The DFFLE with Expanded Feedback Terms (DFFLE-EFT)

The $(m,N;p)$DFFLE-EFT [179] is illustrated in Figure 6.3. This novel structure employs a new N-term functional-link expansion model which non-linearly combines both the equalizer's m-input and p-decision feedback symbols. The functionally-linked hidden layer therefore transforms the equalizer's input space R^{m+p} onto a new non-linear hidden space of increased dimension R^N. Thus the operation of the DFFLE-EFT is based on m most recent channel observations $(y_k,\ldots,y_{k-m+1})$ and its m past decisions $(\hat{s}(k-\tau-1),\ldots,\hat{s}(k-\tau-m))$, with m and p referred to as the equalizer feedforward and feedback orders respectively.

As can be seen from Figure 6.3, the N jointly expanded input and feedback terms are weighted and linearly combined before being fed into the sigmoid. The new weights (v_i for $i = 0,1,\ldots,N$) are updated using the stochastic gradient

Delta Rule as follows:

$$v_i(k+1) = v_i(k) + \mu e(k)(1 - z(k)^2) f_i(k)$$

where, $e(k)$ and μ are defined as before and the new DFFLE-EFT output from the sigmoidal function is defined as:

$$z(k) = \tanh(\sum_{i=0}^{N} f_i(k) v_i(k))$$

6.3.4 Design Strategy for the (DFFLE-EFT)

A novel design strategy for the DFFLE-EFT is proposed which is shown to result in a lower computational requirement relative to the FFLE. The reduction in complexity for the DFFLE-EFT is obtained by noting that the decision feedback symbols are of a binary nature. This fact can be utilized effectively to result in a reduced complexity expansion model $F(k) = f_0, \ldots, f_N$ for the DFFLE-EFT, since the trigonometric and signum functional-link expansions of the decision feedback symbols provide no additional or useful information for equalization.

Thus, it is proposed that the general library of user defined functional-link expansion model $F(k) = f_0, \ldots, f_N$ for the DFFLE-EFT(m,N;p) structure (of any feedforward order m and feedback order p) comprise the following terms for $2-ary$ PAM signalling (assuming the bias input term $f_0 = 1$ is already included):

1. actual equalizer input and decision feedback terms (resulting in $m+p$ terms)

2. trigonometric functional expansion model comprising sum of the following components

   (a) trigonometric sine and cosine functions of the m feedforward inputs only, $\sin(a\pi y_{k-i})$ and $\cos(a\pi y_{k-i})$ for $a = 1, 2, 3$ (resulting in a total of $6m$ terms).

   (b) product of each equalizer feedforward input and decision feedback sample with the trigonometric sine and cosine functions of *other* equalizer

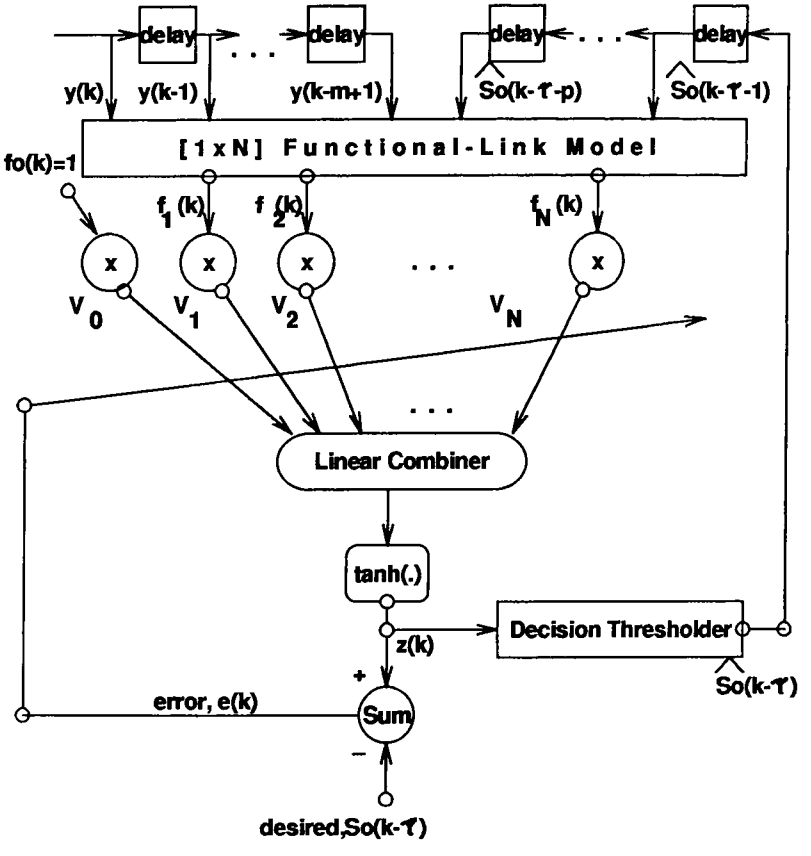

Figure 6.3: The Decision Feedback Functional-Link Equalizer with Expanded Feedback Terms (DFFLE-EFT)

feedforward inputs only; that is, not with the sine and cosine functions of other decision feedback samples (this will result in a total of $2m(m-1)+2mp$ terms).

3. signum function of each equalizer feedforward input only, $sgn(y_{k-i})$ (resulting in m terms)

4. Outer-product expansion of the m inputs **and** the p decision feedback samples $(= ((P_2^{m+p} + P_3^{m+p} +, \ldots, + P_{(m+p)-1}^{m+p}) + 1)$ terms for $m+p$ greater than two, where $P_i^{m+p} = \frac{(m+p)!}{((m+p)-i)!i!}$ where ! denotes factorial. Note that for $m+p=2$ the outer-product expansion will result in 1 term, and for $m+p=1$ there will be no outer product terms).

The above design strategy can best be illustrated by examples: For a two-tapped (1,N;1)DFFLE with a feedforward and feedback order of $m=p=1$, a functional-link expansion model $F(k)$ comprising $N=12$ terms is illustrated below: (letting $\hat{s}_0(k-\tau-1)=\hat{y}_{k-1}$):

$$\begin{aligned} F(k) \;=\; & y_k, \hat{y}_{k-1}, y_k\hat{y}_{k-1}, \\ & \sin(n\pi y_k), \cos(n\pi y_k), \\ & \hat{y}_{k-1}\sin(\pi y_k), \hat{y}_{k-1}\cos(\pi y_k) \\ & sgn(y_k) \quad for \;\; n=1,2,3 \end{aligned} \tag{6.21}$$

Note that an equivalent two-tapped (2,M)FFLE would require a total of $M=21$ terms for its expansion model as discussed in section 6.2.2. Thus, a reduction in complexity of $(N-M=21-12=9)$ terms is obtained for the (1,N;1)DFFLE-EFT.

For the (2,N;2)DFFLE with $m=p=2$, a functional-link expansion model $F(k)$ comprising $M=41$ terms is illustrated below: (letting $\hat{s}_0(k-\tau-1)=\hat{y}_{k-1}$ and $\hat{s}_0(k-\tau-2)=\hat{y}_{k-2}$):

$$F(k) \;=\; y_k, y_{k-1}, \hat{y}_{k-1}, \hat{y}_{k-2}, y_k y_{k-1}, \hat{y}_{k-1}\hat{y}_{k-2}, y_k\hat{y}_{k-1}, y_{k-1}\hat{y}_{k-1}, y_{k-1}\hat{y}_{k-2}$$

$$
\begin{aligned}
&y_k y_{k-1}\hat{y}_{k-2}, y_k\hat{y}_{k-2}, y_k y_{k-1}\hat{y}_{k-1}, y_{k-1}\hat{y}_{k-1}\hat{y}_{k-2}, y_k\hat{y}_{k-1}\hat{y}_{k-2},\\
&y_k y_{k-1}\hat{y}_{k-1}\hat{y}_{k-2}, \sin(n\pi y_k), \sin(n\pi y_{k-1}), \cos(n\pi y_k), \cos(n\pi y_{k-1}),\\
&y_k \sin(\pi y_{k-1}), y_k \cos(\pi y_{k-1}), y_{k-1} \sin(\pi y_k), y_{k-1} \cos(\pi y_k),\\
&\hat{y}_{k-1} \sin(\pi y_k), \hat{y}_{k-1} \sin(\pi y_{k-1}), \hat{y}_{k-1} \cos(\pi y_k), \hat{y}_{k-1} \cos(\pi y_{k-1}),\\
&\hat{y}_{k-2} \sin(\pi y_k), \hat{y}_{k-2} \sin(\pi y_{k-1}), \hat{y}_{k-2} \cos(\pi y_k), \hat{y}_{k-2} \cos(\pi y_{k-1}),\\
&sgn(y_k), sgn(y_{k-1}) \quad for\ n = 1, 2, 3
\end{aligned}
\tag{6.22}
$$

Note that the equivalent fourth order (4,M)FFLE would require a expansion model comprising $M = 67$. The reduction in complexity of the DFFLE (of $N - M = 67 - 41 = 26$ terms) is due to the binary nature of the decision samples fed back, which makes their corresponding trigonometric expansion terms redundant.

Hence, Table 6.3 can be constructed. By comparing Table 6.3 with the previous Tables 6.1 and 6.2, it can be seen that for the same total number of (feedforward and feedback) taps, the DFFLE-UFT has the minimal computational complexity requirements followed by the DFFLE-EFT and the FFLE. For example, a fourth order (4,M)FFLE requires 67 terms, whilst the equivalent order (2,N;2)DFFLE-EFT requires 41 terms compared to 23 terms required by the (2,M;2)DFFLE-UFT. Also an important note on the selection of DFFLE feedforward and feedback orders m and p. It has been shown in [193] and [195] for the case of the optimal symbol Bayesian DFE (which sets the optimal equalization performance limit attainable by any symbol-decision DFE structure), that for a channel of impulse response length $(l + 1)$ (as defined in equation 3.1 of chapter 3) with taps $(a_0, \ldots, a_l)$ and an equalization delay τ, the performance of the Bayesian DFE of feedforward order $m = (\tau + 1)$ is the same as that with m greater than $(\tau + 1)$. For feedforward TE structures without decision feedback, m is generally greater than $(\tau + 1)$; and it is for this reason that the use of a non-linear ANN based DFE structure can be expected to give a significant reduction in the relative computational complexity requirements compared to the corresponding ANN based TE structure. It has been suggested [193] [195] [129]

that if a_j is the channel tap of the largest magnitude (with most of the channel energy therefore presumed to lie between the taps a_0 to a_j), then the equalizer delay should be chosen as $\tau = j$. The feedforward and feedback orders of the DFFLEs (like the Bayesian DFE) can then be set to $m = (\tau + 1)$ and $p = l$ respectively.

Total No. of Taps	No. of expansion terms
$(m+p)$	(N)
$1+1$	12
$2+1$	29
$2+2$	41
$3+1$	54
$3+2$	76
$3+3$	114
$4+1$	91
$4+2$	131
$4+3$	203
$4+4$	339
$5+1$	148
$5+2$	222
(m,p)	$(2m^2+5m+2mp+\sum_{i=1}^{m+p} P_i^{m+p})$

Table 6.3: General Design Strategy for $(m,N;p)$DFFLE-EFT

The equalization capability of the DFFLE-EFT and the DFFLE-UFT resulting from use of the above proposed design strategies is investigated in section 6.4 when the two structures are employed in the equalization of both linear and nonlinear communication channels in the presence of ISI and additive (uncorrelated and correlated) noise sequences.

In the next section, the EKF algorithm is proposed to enhance the speed of

convergence of the DFFLEs' gradient descent learning algorithms.

6.3.5 Application of the Extended Kalman Filter (EKF) to the DFFLEs

Following the same procedure as illustrated for the derivation of the EKF learning algorithm for the FFLE in section 6.2.1 of this chapter, the following set of equations for each of the DFFLE structures can be written:

DFFLE-UFT(EKF):

System Equations:

Letting $\mathbf{q}(k) = \frac{\mathbf{f}(1-\hat{z}(k)^2)}{2}$, and $d(k) = s(k-\tau)$ where the DFFLE-UFT output is now defined as:

$$\hat{z}(k) = \mathbf{f}(k)\mathbf{u}(k)$$

where $\mathbf{u}(k) = [w_0(k)w_1(k)\dots w_M(k)q_1(k)\dots q_p(k)]^T$

and $\mathbf{f}(k) = [f_0(k)f_1(k)\dots f_M(k)\hat{s}(k-\tau-1)\dots\hat{s}(k-\tau-p)]$, gives the dynamical system:

$$\mathbf{u}(k+1) = \mathbf{u}(k) \tag{6.23}$$

$$d(k) = \mathbf{q}(k)\mathbf{u}(k) + e(k) \tag{6.24}$$

EKF Learning Algorithm:

Letting $\mathbf{P}(k) = \mathbf{Q}^{-1}(k)$

$$\mathbf{r}(k) = \lambda^{-1}\mathbf{P}(k)\mathbf{q}^T(k) \tag{6.25}$$

$$\mathbf{o}(k) = \mathbf{r}(k)[1 + \mathbf{q}(k)\mathbf{r}(k)]^{-1} \tag{6.26}$$

$$\mathbf{u}(k+1) = \mathbf{u}(k) + \mathbf{o}(k)e(k) \tag{6.27}$$

$$\mathbf{P}(k+1) = \lambda^{-1}\mathbf{P}(k) - \mathbf{o}(k)\mathbf{r}^T(k) \tag{6.28}$$

DFFLE-EFT(EKF):

System Equations:

Letting $\mathbf{q}(k) = \frac{\mathbf{f}(1-\hat{z}(k)^2)}{2}$, and $d(k) = s(k-\tau)$ where the DFFLE-EFT output is now defined as:

$$\hat{z}(k) = \mathbf{f}(k)\mathbf{v}(k)$$

where $\mathbf{v}(k) = [v_0(k)v_1(k)\dots v_N]^T$

and $\mathbf{f}(k) = [f_0(k)f_1(k)\dots f_N(k)]$ gives the dynamical system:

$$\mathbf{v}(k+1) = \mathbf{v}(k) \tag{6.29}$$

$$d(k) = \mathbf{q}(k)\mathbf{v}(k) + e(k) \tag{6.30}$$

EKF Learning Algorithm:

Letting $\mathbf{P}(k) = \mathbf{Q}^{-1}(k)$

$$\mathbf{r}(k) = \lambda^{-1}\mathbf{P}(k)\mathbf{q}^T(k) \tag{6.31}$$

$$\mathbf{o}(k) = \mathbf{r}(k)[1 + \mathbf{q}(k)\mathbf{r}(k)]^{-1} \tag{6.32}$$

$$\mathbf{v}(k+1) = \mathbf{v}(k) + \mathbf{o}(k)e(k) \tag{6.33}$$

$$\mathbf{P}(k+1) = \lambda^{-1}\mathbf{P}(k) - \mathbf{o}(k)\mathbf{r}^T(k) \tag{6.34}$$

In the next section, pruning strategies for optimising the sizes of the Feedforward and Decision Feedback Functional-Link Equalizers are proposed.

6.4 Pruning Strategies for the FFLE and DFFLEs

The use of the design strategies that have been proposed above for the FFLE and DFFLEs will nevertheless result in non-optimal sized equalizer structures, as not all the terms within the functional-link expansion models will be equally useful. That is, some terms will be more useful or significant in the sense that they will assist the equalizer in classifying the transmitted input data with a lower output mean squared error than other lesser useful or insignificant terms. Hence, pruning of the insignificant terms will yield an optimal sized equalizer structure

of reduced computational complexity. In some applications, particularly where larger order equalizer structures are required for effective channel equalization, pruning of the insignificant functions may also enhance the equalizer's generalization performance.

The design strategies proposed above for the FFLE and DFFLE structures can be considered to provide the first stage of any pruning algorithm. The use of the design strategy can be considered to produce a library of useful functions for any order FFLE or DFFLE structure. This initial pruning stage is essential as the ad hoc inclusion of functional terms within the functional-link expansion models of the FFLE and DFFLEs, can increase their complexity to an intolerable level.

The next stage of a pruning algorithm for optimizing the size of both the FFLE and DFFLE structures, may comprise a number of strategies, such as:

- The well established Orthogonal Least Squares (OLS) algorithm [187] can be used to provide an effective mechanism for the the selection of the most significant or useful functions from the library of useful functions set up by the use of the relevant design strategies. To employ the OLS, all the FL-equalizers are first modified to become *linear in the parameters* structures (by removing the non-linear sigmoidal activation function from their output layer). After the OLS has identified the parsimonious set of input functional terms, the sigmoidal non-linearity can then be re-inserted into the output layer of the FL-equalizers to further enhance their classification capability (for reasons outlined in section 6.2).

- Carrying out a performance evaluation for each of the terms in the library (other than the actual input and decision feedback samples), by:
  augmenting each of the $(M-m)$ or $(N-(m+p))$ terms in turn, with the m input terms of the FFLE or, $(m+p)$ input terms for the DFFLEs, and then training the resulting equalizer, to determine those expansion terms which reduce the equalizer's output MSE to just below the noise floor (for

any SNR) in the least number of training iterations; or, in the case of a failure to converge, result in the lowest MSE after a run of a maximum of an arbitrarily set, for example 1000 training iterations.

Thus an *ordered* list L of the most useful functions can be obtained. The next problem would then be the selection of a subset of the best terms from the above ordered list that would give similar performance to that achieved by use of the fully expanded equalizer comprising all the functional-link expansion terms (resulting from use of its respective design strategy). This could be done for example, by successively augmenting each of the terms from the ordered list to the equalizer input (or input plus decision feedback terms for the DFFLEs), and then working out the training MSE for each equalizer of successive sizes from $(m+1),\ldots,(m+M)$ for the FFLE, and $(m+p+1),\ldots,(m+p+M)$ for the DFFLE-UFT and $(m+p+1),\ldots,(m+p+N)$ for the DFFLE-EFT. Hence a graph or table of training MSE versus equalizer size (number of expansion terms) could be constructed, from which an appropriate equalizer size corresponding to the minimum or near minimum training MSE could be selected. The selected equalizer size would result in a near-optimal sized equalizer, in that inclusion of additional terms would not yield a significant performance improvement (in terms of bit-error rates during the data transmission mode). Some preliminary results based on this pruning strategy have been reported by Hussain, Soraghan and Durrani [178] for the FFLE and the DFFLE-UFT.

- Alternatively, the pruning strategy we employed for the new FFENN structure in chapter 4 for modeling of non-linear dynamical systems could also be employed in principle here. Namely, one starts by training a fully expanded equalizer structure resulting from use of the proposed design strategies. At the end of training, the insignificant terms (represented by the functions with the smallest weighting co-efficeints relative to the most significant term with the largest weight) are successively pruned one by one starting with the least significant term. After the pruning of each term, the resulting pruned

equalizer structure (comprising the remaining significant terms) is then re-trained and the MSE value recorded. The pruning process is stopped at the stage, when the training MSE of a pruned equalizer structure increases significantly (or is above a specified threshold) compared to that achieved without the pruning of that particular term.

Note that in all the above strategies, the criteria for construction of an optimal or near optimal sized equalizer has been the training MSE, with the equalizer structure giving the lowest training MSE assumed to be indicative of providing the best performance (bit-error rate) during data transmission. In theory, however, there is no known link or relationship between the bit-error rate criterion and the training Mean Squared Error criterion achieved by conventional linear and non-linear equalizers [194] (which are generally designed to minimise the Mean Squared Error between the equalizer output and the desired actual transmitted symbol). That is, the optimal performance criterion for any equalizer, which is the bit-error rate, cannot be guaranteed to be achieved by an equalizer yielding the lowest training MSE. This is because the theoretical optimal symbol Bayesian TE and DFE structures (which use Bayes decision theory for making their decisions) are known to provide the best attainable bit-error rate for all symbol decision based TE and DFE structures respectively; yet, they will not necessarily produce the best Mean Squared Error (MSE) performance on the training symbol set [194]. Hence theoretically, an optimal sized FFLE or DFFLE structure could only be determined by measuring the bit error rate achieved by each successively pruned equalizer. The pruned equalizer giving the best bit error rate performance (closest to that of the corresponding optimal symbol Bayesian TE or DFE) would then obviously be the optimal sized equalizer for a particular application. However, computing the bit-error rates for each successively pruned equalizer would be a highly computationally expensive task.

In the next section 6.5, we primarily investigate through the use of simulation results, the effectiveness of the functional-link expansion models resulting from the use of the proposed design strategies for the FFLE and DFFLE structures,

relative to other recently reported non-linear equalizer models. In section 6.5.1 their application to equalization of linear and non-linear communication channels in the presence of ISI and both uncorrelated and correlated noise is investigated.

Finally, in section 6.5.2 the FFLE and DFFLE structures are proposed as an alternative solution to the problem of overcoming co-channel interference in digital communications systems. Two simulation case studies are employed to compare their performance with other recently reported linear and ANN based equalizer solutions.

6.5 Application Examples of the FFLE and DFFLEs and Comparative Performance Analysis

6.5.1 Adaptive Equalization of Linear and Non-linear Communication Channels

We investigate the use of the FFLE, DFFLE-UFT and DFFLE-EFT structures in the equalization of the following channel models, with the input assumed to be $2-ary$ PAM for all simulations in order to enable comparison with other recently reported simulation results:

Linear Non-Minimum Phase (NMP) Channel Model

$$H(z) = 0.3482 + 0.8704z^{-1} + 0.3482z^{-2} \tag{6.35}$$

Such channels are quite likely to be encountered in a practical communication system [183] [55].

Non-linear Channel Model

Non-linear digital communications channels were discussed in chapter 3. The non-linear channel used in the simulations has the structure of the model shown

in Figure 6.4. The transmitted symbols $s(k)$ are passed through linear channel of transfer functions $H(z)$, and the output of the channel is added to non-linear harmonics. The value of the gain co-efficients G_2, G_3, and G_4 determine how severe the non-linear distortion effects will be. Such non-linear channel models are frequently encountered in data transmission over digital satellite links, especially when the signal amplifiers operate in their high-gain limits [55].

In order to demonstrate the ability of the FFLE and DFFLE structures to cope with non-linear distortion, we used the following linear channel model with the additive non-linearity gains set to $G_2 = 0.2$, $G_3 = G_4 = 0$.

$$H(z) = 0.3482 + 0.8704z^{-1} + 0.3482z^{-2} \tag{6.36}$$

The additive noise for this case was chosen to be *correlated* and was given by:

$$n(k) = 0.8\epsilon(k) + 0.6\epsilon(k-1)$$

where $\epsilon(k)$ is a Gaussian white sequence [184].

The following sets of experiments were carried out:

A. Performance Comparison of FFLE and DFFLE structures in Equalizing the Linear Channel Model

A fourth order ($m = 4$) FFLE(4,67) employing the $M = 67$ expansion model of equation 6.4 illustrated in the design strategy of section 6.2.2, was employed for the equalization of the linear NMP channel model (equation 6.35 above) with the equalization delay ($\tau = 1$) and a Signal to Noise Ratio of ($SNR = 24$dB) obtained by corrupting the channel outputs with an Additive White Gaussian Noise (AWGN) source of zero mean and variance 0.004. For comparison, we simulated similar ordered ($m = p = 2$) DFFLE-UFT(2,21;2) structure with unexpanded feedback terms, employing the $M = 21$ terms (illustrated in equation 6.1) augmented with its previous two decision feedback samples; and a DFFLE-EFT(2,41;2) with expanded feedback terms, employing the $N = 41$ term expansion model of equation 6.22. The detected bits were used as feedback samples

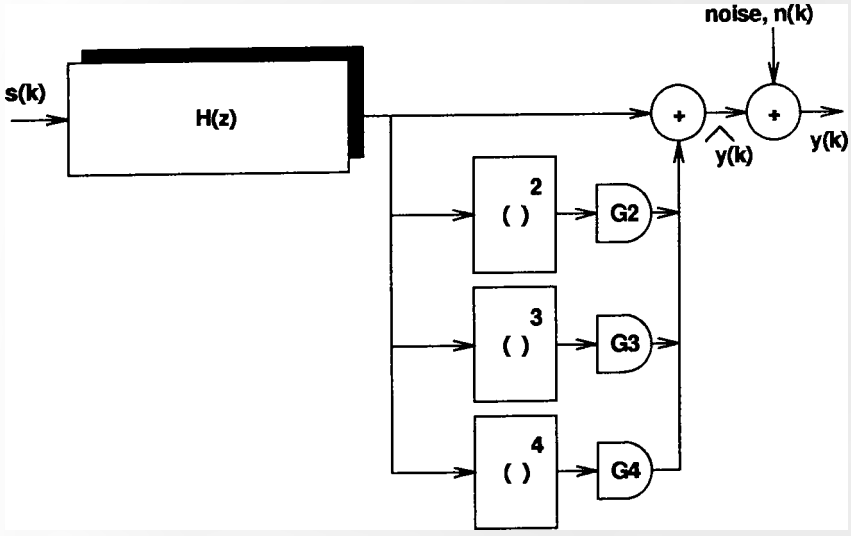

Figure 6.4: The model of the non-linear communication channel.

in both the DFFLEs. Also simulated for comparison was a fourth order LTE(4) structure trained using the LMS algorithm.

Speed of Convergence Characteristics

The Mean Squared Error (MSE) convergence curves of all the equalizer structures are shown in Figure 6.5. The plots were obtained from averaging of 25 independent trials involving the use of 1500 training symbols. The plots were then smoothed by a window of 64 samples. The Delta Rule was employed for updating the weights of all the Functional-Link (FL) based equalizer structures with a constant step size of $\mu = 0.1$. The LTE was trained by the LMS algorithm whose step size was also set to 0.1. As can be seen, the new DFFLE-EFT(2,41;2) employing a novel 41 term functional-link expansion model non-linearly combining both the equalizer input and decision feedback samples, gives the best convergence speed characteristics converging to the noise floor in about 350 iterations and a steady state MSE of -26dB, followed by the DFFLE-UFT(2,21;2) which converges to the noise floor in about 600 iterations and a steady error of -25dB, and the FFLE(4,67) structure which converges to the noise floor in about 800 iterations and a steady MSE around the noise floor of -24dB as well. As expected, the LTE(4) gives the worst performance compared to the non-linear equalizers converging to a steady training MSE of -10dB, on account of its inability to form non-linear decision regions required for the effective equalization of the NMP channel.

To investigate the enhancement of MSE convergence speed provided by use of the EKF training algorithm over the DR update, the above DFFLE-EFT(2,41;2) structure was trained using the EKF algorithm illustrated in equations 6.31 to 6.34. As can be seen from Figure 6.5, the new DFFLE structure trained by the EKF algorithm offers a faster speed of convergence over the same equalizer trained by DR, almost halving the time required to reach the noise floor. However, this improvement is achieved at the expense of an increased learning computational

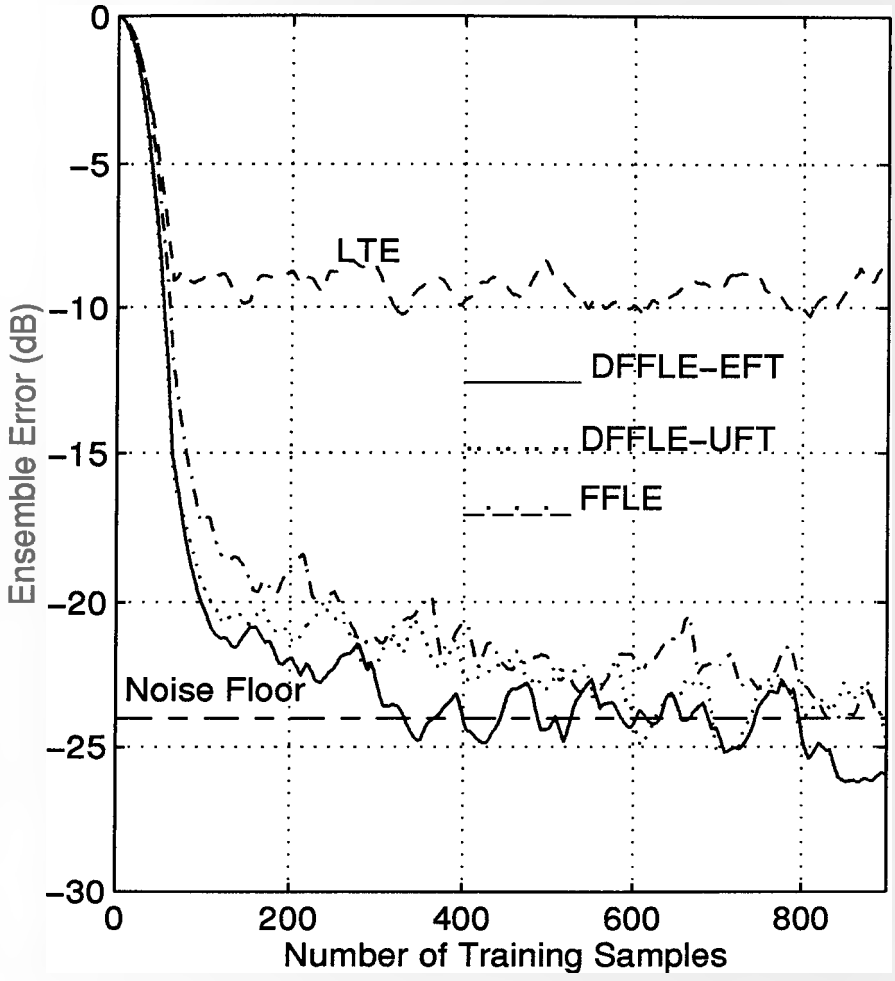

Figure 6.5: Comparison of Equalizers' MSE Convergence Characteristics.

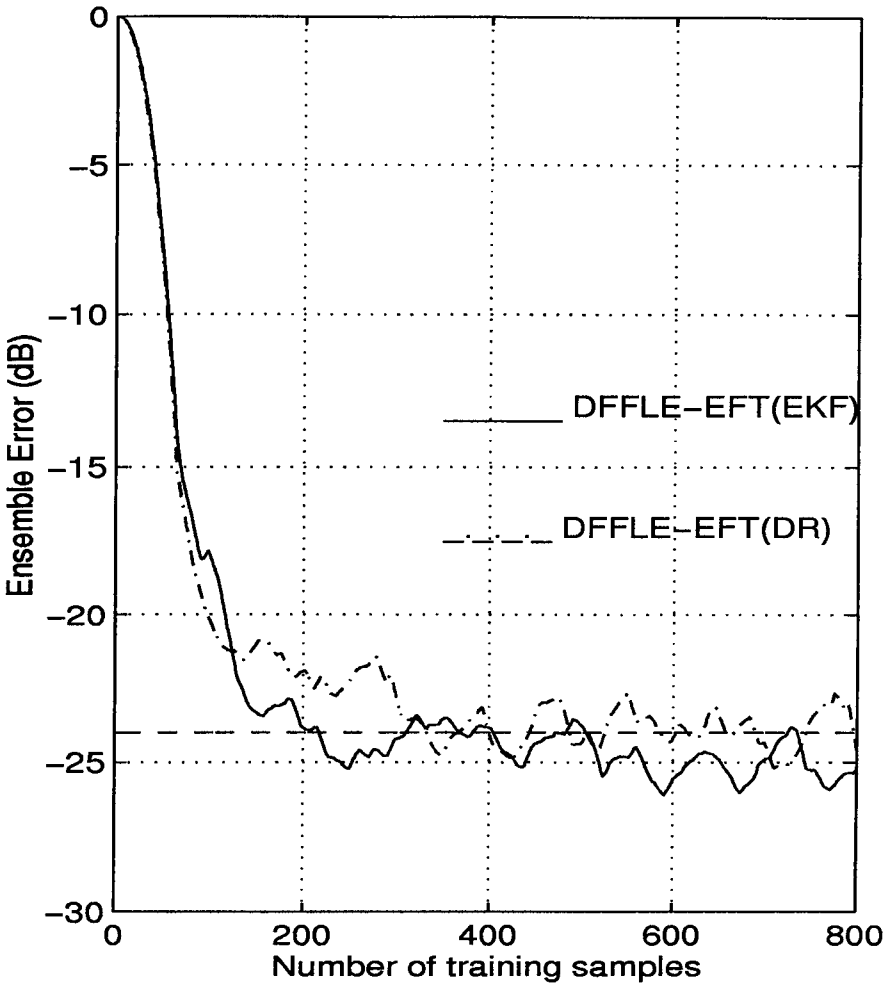

Figure 6.6: Comparison of MSE Convergence Characteristics: DR trained DFFLE-EFT and EKF trained DFFLE-EFT.

complexity. Nevertheless, such a structure would be highly desirable in applications requiring the equalizer training times to be very small, particularly in non-stationary environments (for example mobile radio channels).

Bit-Error Ratio (BER) Characteristics

The NMP channel model of equation 6.35, was also used to study the relative BER performance of the above equalizers under a variety of SNRs. Since the BER, which is defined as the ratio of misclassified to correct symbols at the equalizer output, is the ultimate performance criterion for an equalizer, the BERs

achieved by the above FFLE and DFFLE structures were assessed against various other recently reported non-linear feedforward and decision feedback equalizers. The BER plots for the FFLE and DFFLEs were obtained from averaging of over ten independent runs of half-a-million random bipolar test samples after the equalizers had converged to a steady state training error. The other non-linear equalizers that have been employed for the equalization of the above NMP channel under similar conditions (namely, for $2-ary$ PAM signalling and a decision delay of $\tau = 1$) are summarized below:

1. A fifth order (5-9-3-1) MLP feedforward equalizer reported by Gibson *et al* [173], comprising of two hidden layers with 9 and 3 sigmoidal nodes respectively (the numbers of which were determined by trial and error) and trained using the computationally expensive Back Propagation (BP) algorithm.

2. A fourth order (4,35) Adaptive Polynomial-Perceptron (APP) feedforward equalizer structure reported by Chen *etal.* [183], employing a polynomial expansion of degree 3. The equalizer was trained using the Delta Rule, as per the FFLE and DFFLEs

3. A fourth order (4,64) RBF feedforward equalizer reported by Chen *et al* [194], comprising 64 Gaussian centres and employing a combined learning of supervised clustering and LMS to train the centers and weights simultaneously.

4. A new first order (1,2;1) Recurrent Neural Network (RNN) based equalizer reported by Kechriotis *et al* [55] comprising 2 fully interconnected sigmoidal units, and trained by the computationally expensive Real-Time Recurrent Learning (RTRL) algorithm.

5. A fifth order (4-9-3-1;1) DFE-MLP equalizer (with $m = 4$, $p = 1$ and two hidden layers comprising 9 and 3 sigmoidal nodes respectively) reported by Siu *et al* [185] also trained using the BP algorithm.

6. The optimal symbol Bayesian TE reported in [183] and discussed in chapter 3.

The optimal sequence estimator namely, the adaptive MLVA, implemented as a Viterbi Algorithm with the exact channel model and a fixed decision delay of $\tau = 1$ [193] has been used as the benchmark for assessing the BER performance of all the non-linear equalizers.

In order to determine the best feedforward equalizer, the BERs achieved by all the feedforward (or TE) non-linear equalizers were first evaluated and plotted in Figure 6.7. As can be seen from the Figure 6.7, which shows the relative BERs achieved by the (4,67)FFLE, the (4,64)RBF, the (5-9-3-1)MLP and the (4,35)Adaptive Polynomial-Perceptron (APP) equalizers; the RBF equalizer can be seen to provide the best performance closely followed by the (4,67)FFLE and the (5-9-3-1)MLP equalizers.

The best performance attained by the RBF TE is due to the fact that, as shown in [194], it exactly implements the optimal symbol Bayesian TE (or MAP), for a completely specified communication channel (including the channel order and the additive noise variance). These parameters in practice, therefore require a priori estimation.

In Figure 6.8, the the BERs achieved by the new (2,41;2)DFFLE-EFT and the (2,21;2)DFFLE-UFT equalizers are plotted against those achieved by the (1,2;1)RNN and the (4-9-3-1;1)DFE-MLP structures. The optimal BERs achieved by the adaptive MLVA are also illustrated for comparison. As can be seen, the new DFFLE-EFT provides the nearest to optimal performance followed by the DFE-MLP and the DFFLE-UFT structures. Note that the DFE-MLP is of overall order $(m + p = 4 + 1 = 5)$ employing four feedforward taps and a single feedback sample, whereas the DFFLE structures employed in these simulations are of a lower order $(m + p = 2 + 2 = 4)$.

In Figure 6.9, the BERs achieved by the DFFLE-EFT, the optimal symbol Bayesian TE, and the optimal sequence adaptive MLVA are plotted. As can be seen, the simple 41 term DFFLE outperforms the more complex 64 term optimal

symbol Bayesian TE structure, and hence for this channel, the new DFFLE-EFT symbol equalizer can be used as a viable alternative to the optimal Bayesian TE in providing a closer approximation to the ultimate optimal sequence MLVA equalizer.

The BERs of both the DFFLEs illustrated in all the above figures were obtained with detected symbols $\hat{s}(k-\tau)$ fed back. Clearly this is the realistic scenario. In practice, the equalizer decisions cannot be guaranteed to be 100% correct, and thus when an error is made, error propagation will result in the DFFLE structures (or any DFE based structure employing decision feedback).

The effects of error-propagation can be investigated in simulation by comparing the BERs obtained using correct symbols and detected symbols as feedback, respectively. This is demonstrated in Figures 6.10 and 6.11 for the DFFLE-EFT and the DFFLE-UFT respectively. As can be seen from the dotted curves, the error propagation effects only very slightly degrade the BER performances of both the structures for this application, with the DFFLE-EFT seen to be more tolerant to the error propagation effects.

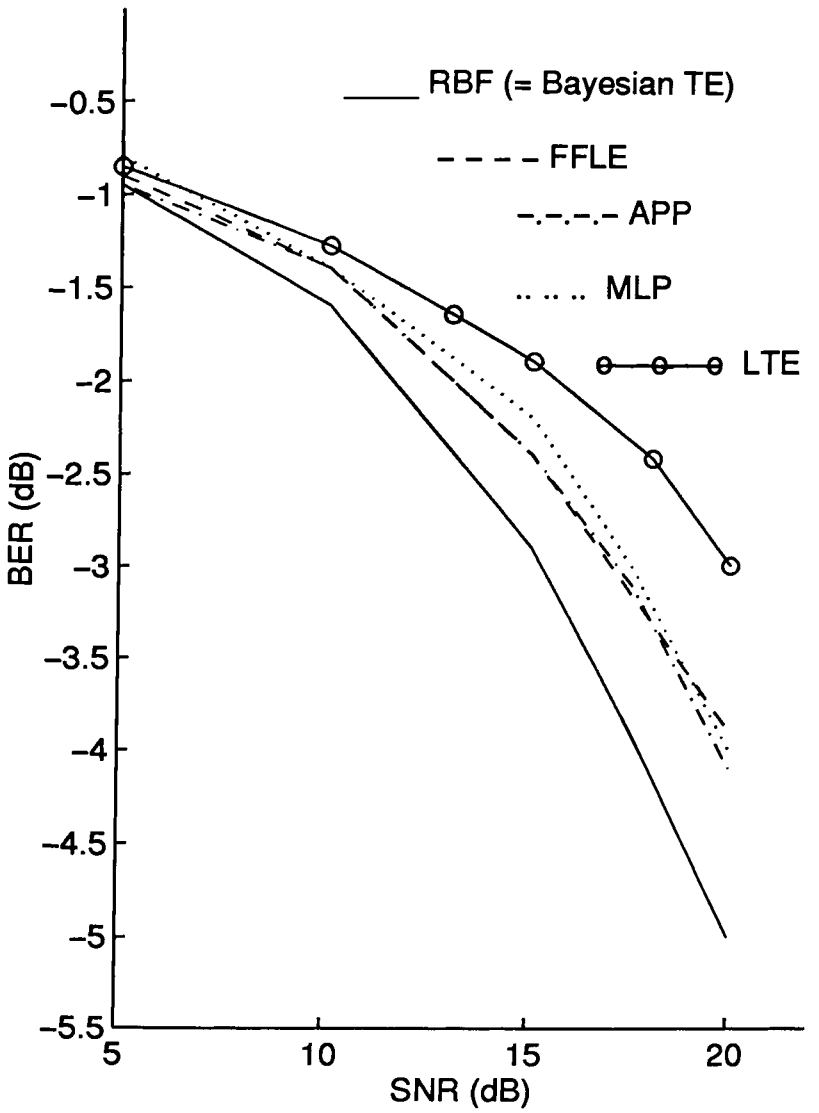

Figure 6.7: BER Performance Comparison of various TE based Structures.

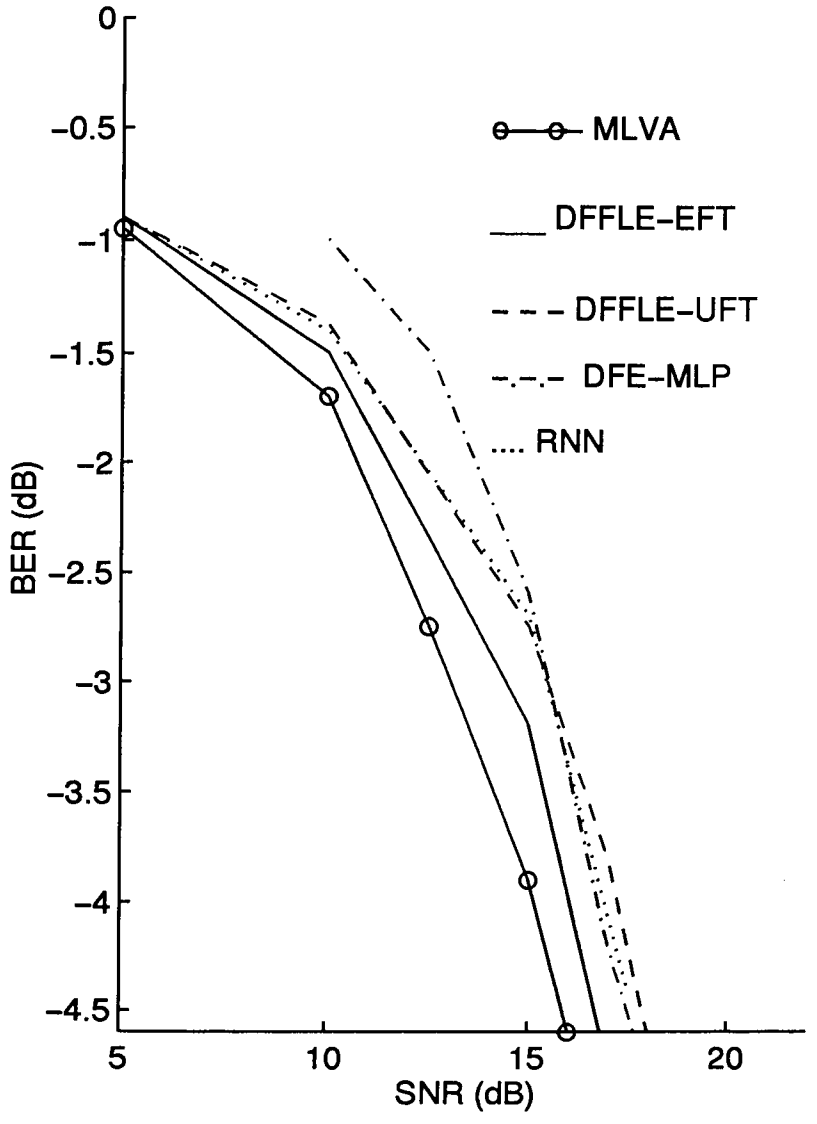

Figure 6.8: BER Performance Comparison of various Non-linear DFE based Structures.

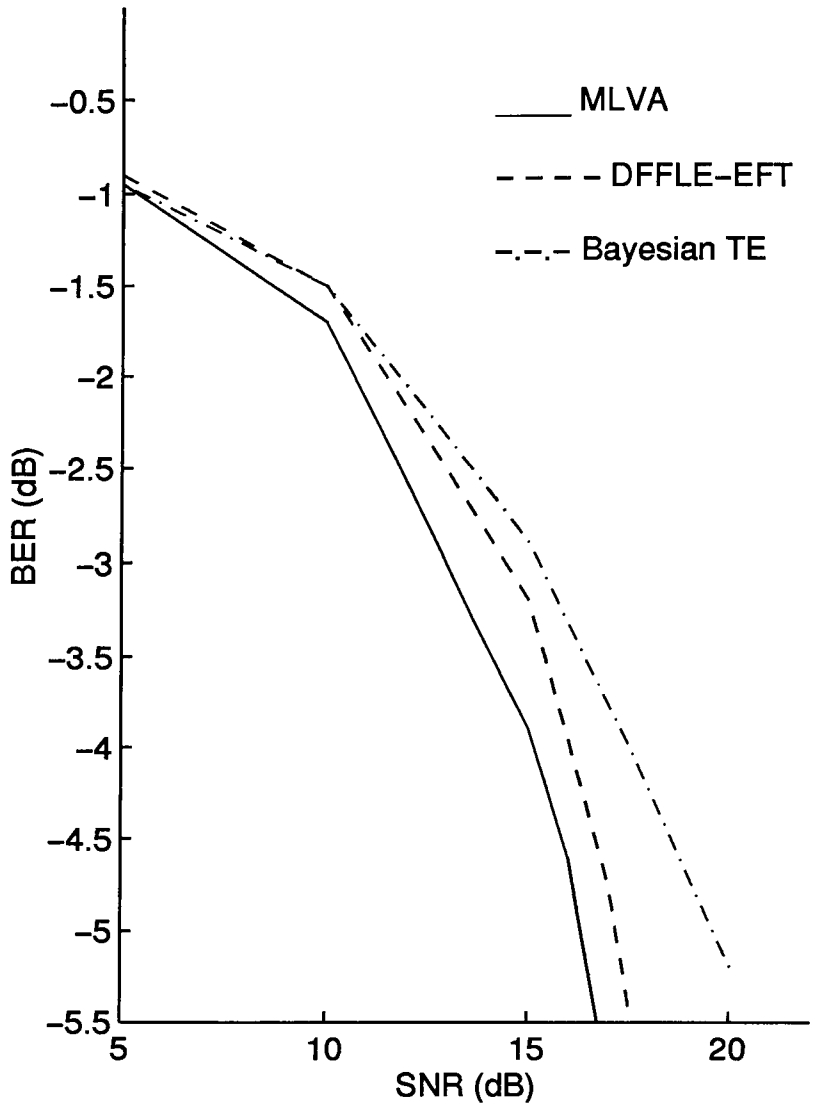

Figure 6.9: BER Performance Comparison of DFFLE-EFT with optimal MLVA and Bayesian TE.

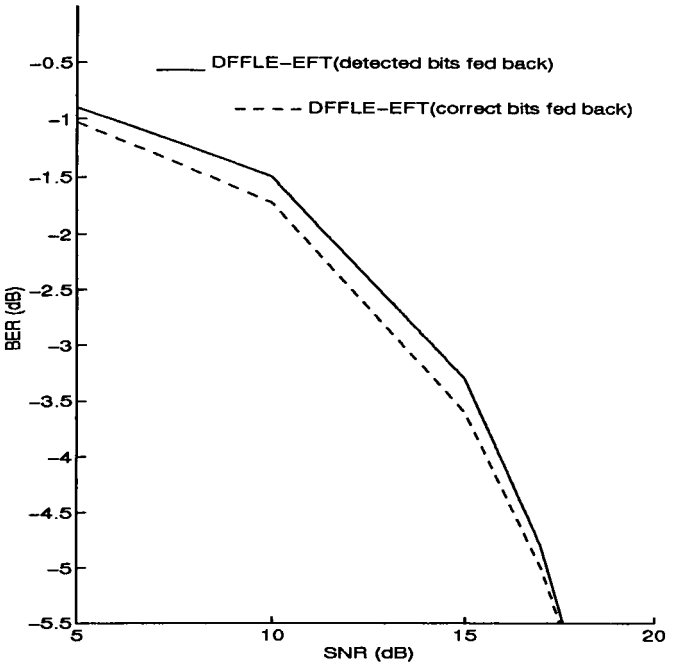

Figure 6.10: Error Propagation Effects in DFFLE-EFT for linear NMP Channel.

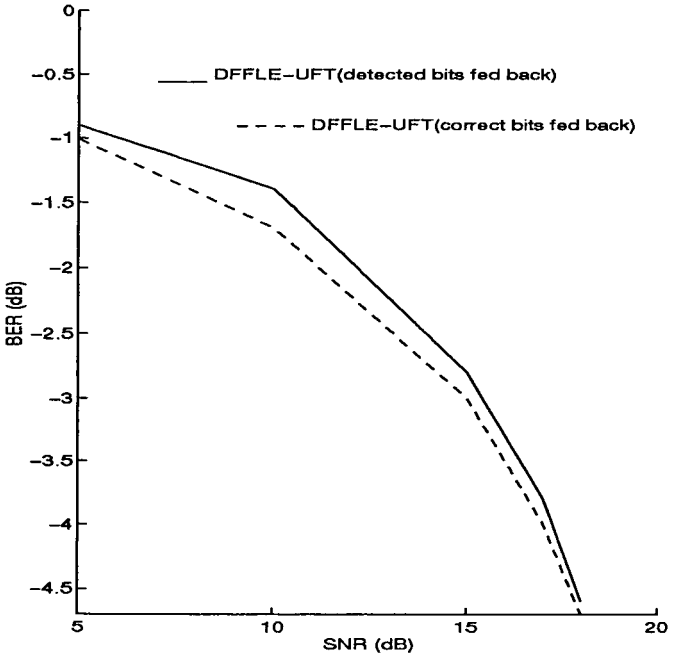

Figure 6.11: Error Propagation Effects in DFFLE-UFT for linear NMP Channel.

B. Performance Comparison of FFLE and DFFLE structures in Equalizing the Non-Linear Channel Model

The BER performance curves of the (4,67;1)FFLE, (2,41;2)DFFLE-EFT, the (2,21;2)DFFLE-UFT, are compared with those of the (3-9-5-1)MLP based equalizer and the optimal symbol Bayesian TE reported in [184]. The BER curves are plotted in Figure 6.12, for the non-linear channel model of equation 6.36 above (with an equalization delay of $\tau = 1$). As can be seen from Figure 6.12, the 41 term DFFLE-EFT and the 23 term DFFLE-UFT offer almost similar performance, outperforming the more complex Bayesian TE at high SNRs. The FFLE can be seen to give a closer approximation to the Bayesian TE compared to the MLP based TE, but at the expense of a greater relative computational requirement – 67 hidden layer nodes for the FFLE compared to 14 for the MLP. However, the learning complexity requirements of the DR trained FFLE and the DFFLEs are significantly less compared to those of the BP trained MLP. Furthermore, the FFLE and DFFLE structures are easier to analyse than the MLP which also lacks a general design strategy.

The ultimate BER performance however, for the non-linear channel case would also be provided by the optimal sequence MLVA as per the linear channel case. However, in practice, the use of the adaptive MLVA for equalization of non-linear channels would be even lesser attractive, as effective estimation of the non-linear channel model would require the use of non-linear modeling techniques, which will further add to the computational requirements of the MLVA. This is the reason why no significant attempts have been made to date in the development of adaptive MLVA structures for the equalization of non-linear channels in the presence of ISI and additive noise. To investigate the effects of error propagation in the DFFLEs for the above non-linear channel, the BERs were again computed with correct bits fed back. As per the linear channel equalization case, only a very small performance degradation was found to occur.

The above application has served to show that the new DFFLE-EFT structure is a viable alternative to the the optimal symbol Bayesian TE (and all other

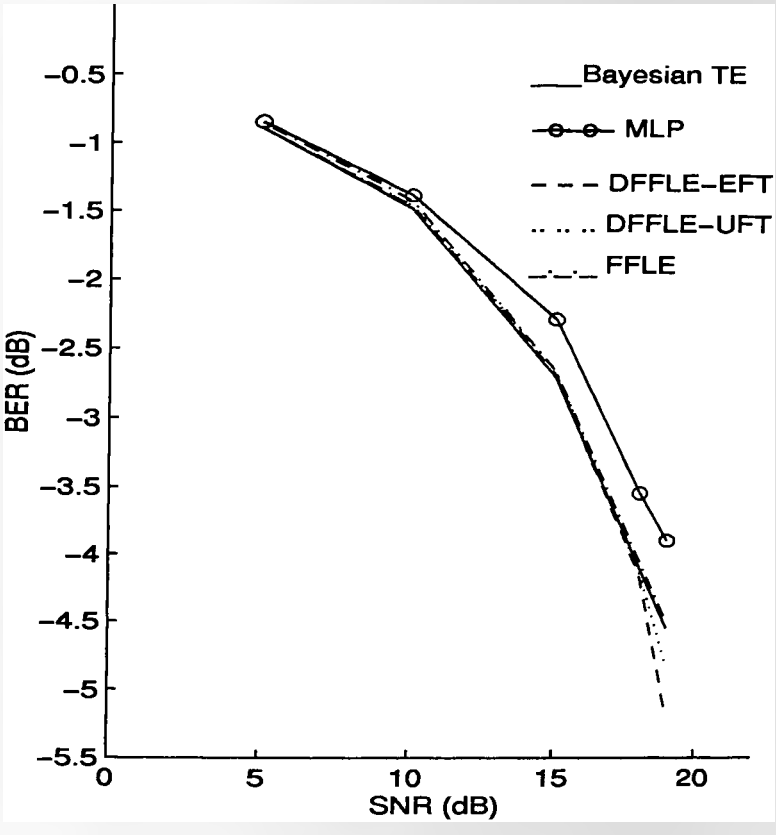

Figure 6.12: BER Performance Comparison of various non-linear equalizers in equalization of non-linear channel model

non-linear equalizers that have been reported to-date to approximate the optimal Bayesian TE) for the equalization of linear and non-linear channels in the presence of ISI and additive noise. The attraction of the DFFLE-EFT is not only its superior performance, but also its simple integral structure and computationally efficient learning algorithm. These characteristics will give the new structure a significant advantage over the Bayesian TE in terms of ease of practical implementation. Furthermore, unlike the Bayesian TE, the DFFLE-EFT is a truly indirect-modeling equalizer in that it requires no a priori knowledge or means of identification of the communication channel.

Note that for the equalization of non-linear channels however, the DFFLE-UFT structure may be able to provide a better complexity-performance trade-off relative to the DFFLE-EFT. This needs be investigated using additional non-linear channel examples.

6.5.2 Overcoming Co-channel Interference in Digital Communications systems

Two sets of experiments are performed using two different co-channel models. In the first set of experiments, we propose use of the FFLE for overcoming co-channel interference, and compare its performance with the optimal Bayesian equalizer. We show using an example that the FFLE is capable of providing a near-optimal solution to the problem of overcoming co-channel interference effects in digital communications systems.

In the second set of experiments, we use a realistic co-channel system to compare the performance of the DFFLE structure with the conventional LTE and the non-linear FFLE and recently reported RBF equalizers. Error propagation properties of the DFFLE are also investigated. Note that for this application, only the DFFLE-EFT structure has been employed.

In the first experiment, a second order (2,21)FFLE structure with ($m = 2$) and ($M = 21$) term expansion model (illustrated in equation 6.1) was employed

to equalize the following co-channel system:

$$H_0(z) = 1.0 + 0.5z^{-1}$$

$$H_1(z) = \beta(1.0 + 0.2z^{-1})$$

where the value of β was dictated by the Signal to Interference (SIR) requirement. In order to highlight the basic concepts involved in the equalization of the above co-channel system we first present an example. Let SIR = 10dB which gives rise to $\beta = 0.346$. For the above co-channel system, use of $2 - ary$ PAM input signal and an equalizer of order $m = 2$ with delay $\tau = 0$, results in 8 desired signal states (that is, outputs from the primary distorting channel H_0) and 8 interfering signal states (that is, outputs from the interfering co-channel H_1). The total number of channel observation states (that is, the overall outputs from the co-channel system $y(k)$) are thus 64 [190] which are plotted in the state diagram of Figure 6.13, where the circles o's represent the desired channel states when a $s_0(k - \tau) = +1$ input symbol is transmitted, and the crosses $\times$'s represent the desired channel states for a -1 transmitted input. The dots in Figure 6.13 represent noise-free channel observation states (that is outputs from the co-channel system without the addition of Gaussian noise, $n(k)$). Note that in the presence of additive Gaussian noise, the observations will form clusters with the means of these data clusters obviously being the noise-free observations, and their variance equal to that of the noise σ_n^2. As can be seen from Figure 6.13, the composite effects of the distorting channel and the interfering co-channels has led to the noise-free observations forming clusters around the desired signal states. It can be easily shown that for a high SIR value, the noise free observations states will concentrate around the desired signal states, whereas for a low SIR the noise-free observations states become more widely spread. In Figure 6.13, the optimal decision boundary for this co-channel system [190], for the case of additive noise power of $\sigma_n^2 = 0.0125$ is also shown in a dotted line. The optimal decision boundary partitions the observation space (which is also the equalizer input space) into two decision regions. When an observation vector

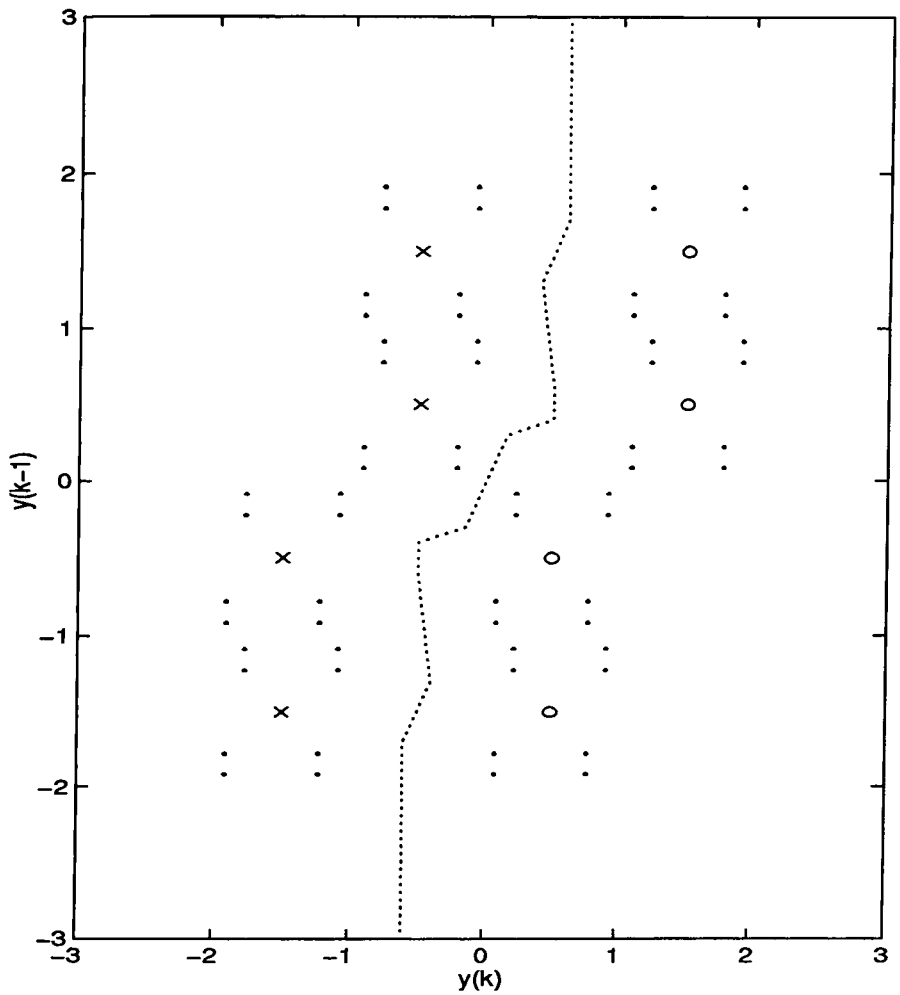

Figure 6.13: Outputs of Co-channel System for $2-ary$ PAM input and transmission delay = zero. The o and x denote desired signal states (+1 and −1 respectively), and the dots . indicate the noise-free observation states. The dotted line is the approximate optimal Bayesian decision boundary.

$(y(k),\ldots,y(k-m+1)) = (y(k)y(k-1))$ appears in the right hand region of the boundary, the underlying optimal Bayesian decision function (which was derived for co-channel systems in [190] by Chen and Mulgrew) is positive and a decision $\hat{s_0}(k-\tau) = +1$ is made corresponding to a transmitted $s_0(k-\tau) = +1$. If an observation vector appears in the left hand region of the optimal boundary then a decision $\hat{s_0}(k-\tau) = -1$ is made (as the optimal Bayesian decision function is negative) corresponding to a transmitted $s_0(k-\tau) = -1$. Similar to the case of equalization of dispersive channels in the presence of additive noise (without co-channel interference) which was discussed in chapter 3, this way of making symbol decisions (based on the values of the optimal Bayesian decision function) is optimal because it produces the minimum average error probability or BER [129]. A (2,21)FFLE was trained on 1000 observation samples using the DR with $\mu = 0.03$. The trained FFLE produced the decision boundary shown in Figure 6.14, where the current FFLE output is plotted against the previous FFLE output. As can be seen, the ability of the FFLE to form highly non-linear decision regions has enabled it to effectively separate the two classes of transmitted input symbols.

To compare the performance of the FFLE against the optimal Bayesian equalizer, a second scenario was simulated. The SIR was fixed at 16dB which gave rise to a value for $\beta = 0.174$. The noise power σ_n^2 was varied to give different SINR (Signal to Interference plus Noise Ratio) conditions. Figure 6.15 depicts the Bit Error Ratio (BER) curves of the (2,21)FFLE structure and a second order (2,64)RBF equalizer reported in [190] (which was realised using 64 non-linear Gaussian functions centered at the 64 noise free observation channel states and trained using a complex 2-stage strategy). As can be seen, the 21 term FFFLE gives a close performance to that of the 64 term RBF equalizer (which was shown in [190] to realise the optimal Bayesian TE for this co-channel system) using a significantly fewer number of combined joint-activation and trigonometric terms of the equalizer inputs.The BER plot for the FFLE was obtained from averaging

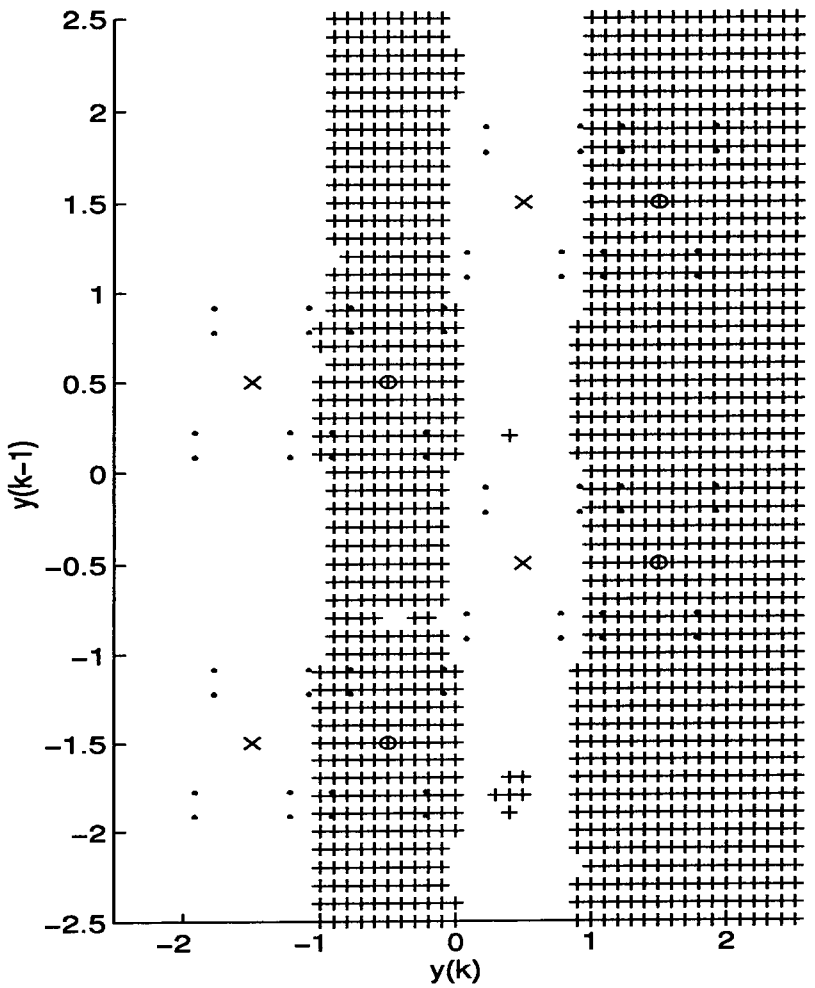

Figure 6.14: Decision Boundary Produced by (2,21)FFLE.

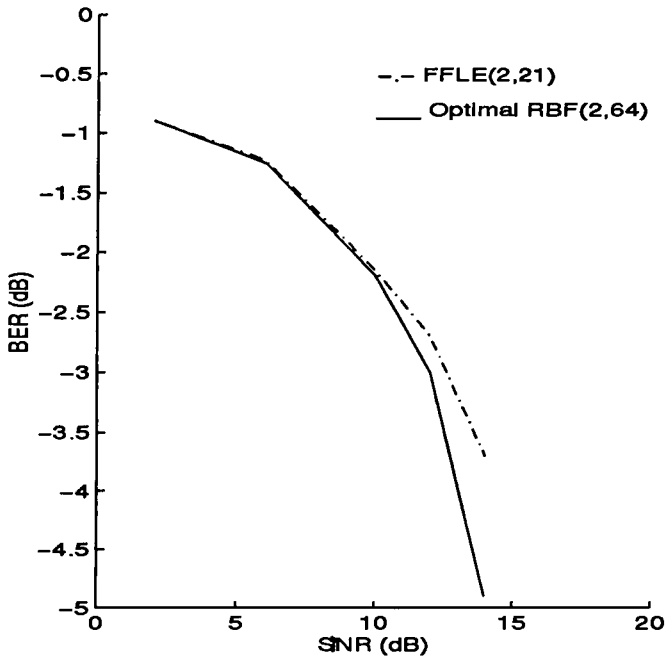

Figure 6.15: BER vs. SINR Performance Comparison.

of several independent runs of half-a-million random sequence bipolar test samples after the equalizer had converged to a steady state training error.

In the second set of experiments, a severe amplitude distorted co-channel system involving one interfering co-channel represented by

$$H_0(z) = 0.3482 + 0.8704z^{-1} + 0.3482z^{-2}$$

and

$$H_1(z) = \beta(0.6 + 0.8z^{-1})$$

was used to compare the performance of the DFFLE-EFT structure with the FFLE, and the LTE and RBF equalizers reported in [190] for the same system. For a fair comparison, all the Functional-Link (FL) based equalizers employed were chosen to be of order 4 and the equalization delay was set to ($\tau = 1$). It

was shown in [190] that a LTE of order 4 is optimal for this co-channel system, in that increasing the number of taps beyond will not better its performance (process referred to as noise-enhancement).

CASE I:

A value of $\beta = 0.0631$ was chosen to provide a constant SIR= 24dB, and the noise power σ_n^2 was varied to produce different Signal to Interference plus Noise (SINR). The BER curves for the (4,67)FFLE (employing the 67 term functional-link expansion model illustrated in equation 6.4), the (2,41;2)DFFLE-EFT (employing the 41−term functional-link expansion model shown in equation 6.22), the LTE(4) and the (4,64)RBF are depicted in Figure 6.16.

CASE II:

Next, the noise power was fixed at $\sigma_n^2 = 0.00398$, giving rise to a constant Signal to Noise Ratio $SNR = 24$dB. The interfering signal power was changed by choosing different values of β. The performance curves of the corresponding equalizers are plotted in Figure 6.17. The BER curves for all the FL based equalizers were obtained from averaging of over ten independent runs of half-a-million random sequence bipolar test samples after the equalizers had converged to a steady state training error (which took about one thousand training iterations for the DFFLEs with a step size $\mu = 0.05$, which is comparable to the training period required by the RBF equalizer [190]). Comparing Figures 6.16 and 6.17, it can be seen that the two BER curves of the Weiner (LTE) filter are similar, confirming that the LTE does not distinguish the interfering signal from the noise [168]. The best the LTE can do is to treat the interfering signals as additional coloured noise. The performance curves of all the non-linear equalizers are seen to exhibit significant differences with the performance obtained by changing the SIR for a fixed SNR being markedly better than that obtained by changing the SNR for a fixed SIR (which is consistent with the results of [190]). This confirms that the non-linear equalizers treat the noise and interfering signals differently and

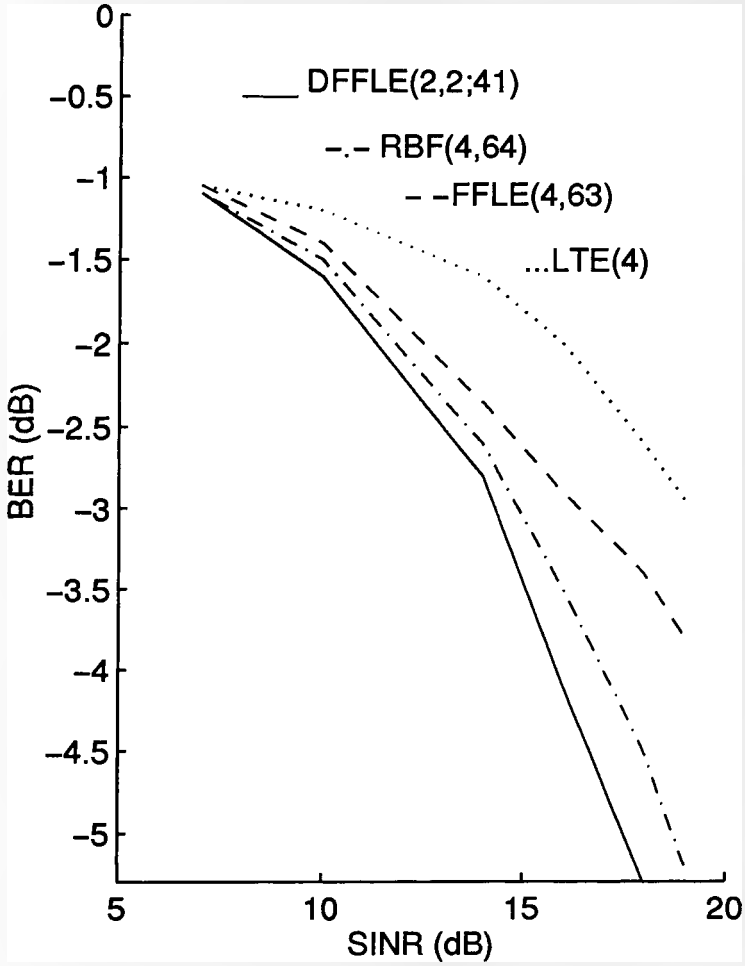

Figure 6.16: CASE I: BER Performance Comparison for SIR fixed at 24dB, and Noise Power σ_n^2 varied to produce different SINRs.

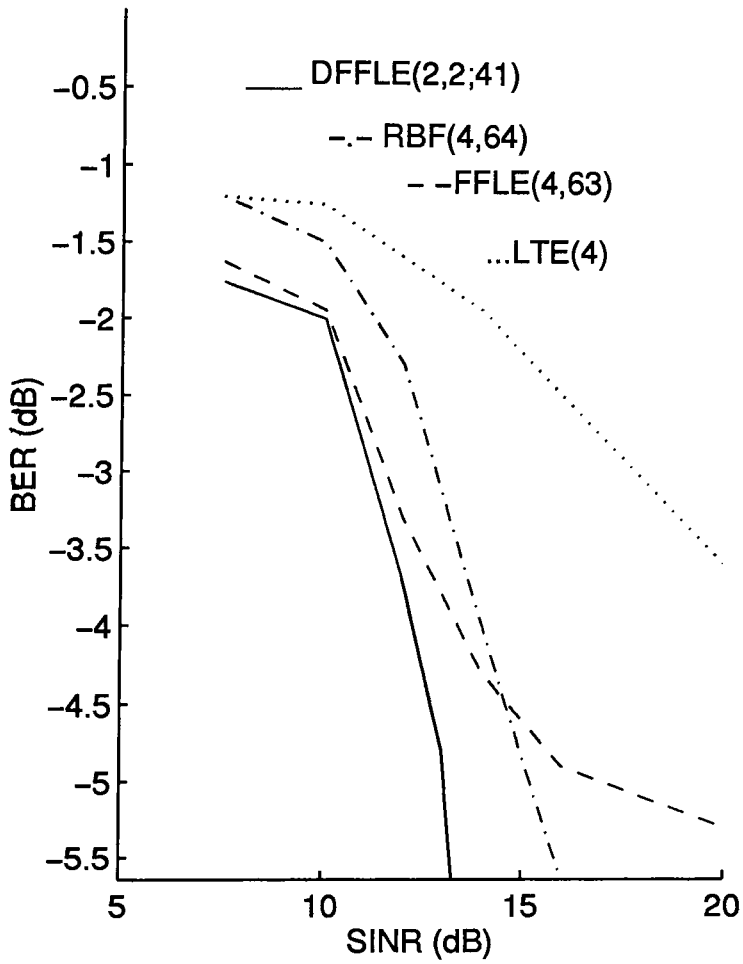

Figure 6.17: CASE II: BER Performance Comparison for SNR fixed at 24dB and β varied to produce different SINRs.

are more effective in approximating the optimal equalizer solution on account of their inherent non-linear decision making capability [190]. As can be seen, from the family of non-linear equalizers the 41−term DFFLE-EFT employing a novel functional-link expansion model combining both the equalizer input and Decision Feedback (detected) samples gives the best overall BER performance, followed by the feedforward structures of (4,64)RBF and the (4,67)FFLE. The LTE(4) provides the worst performance as expected. Note also the cross over in the BER curves of the FFLE with the RBF in Figure 6.17 (for Case 2). This is conjectured to be a result of the FFLE converging to a higher noise floor relative to the other non-linear equalizers; as a result of which the BER performance of the FFLE fails to improve relatively significantly, even with increasing SINRs (which are obtained for this case by reducing the interfering signal power for a fixed additive noise power).

To investigate the error propagation properties of the DFFLE-EFT, the BER results were again computed for the DFFLE-EFT with correct bits fed back. As can be seen from the dotted curves in Figures 6.18 and 6.19, the error propagation effects only very slightly degrade the DFFLE performance for both the cases considered in this application.

It was shown in [190] that computing the optimal Bayesian equalizer for the above co-channel system is extremely costly. The full RBF network able to realize the optimal solution would need a total of 2048 centers (corresponding to the number of observation states = number of desired states (64)× number of interfering states (32)) which makes the calculation of BERs impractical. Hence the new non-linear DFFLE-EFT based structure requiring just $M = 41$ terms, is a viable alternative to the (4,64)RBF, the (4,67)FFLE and the (4)LTE for providing a closer approximation to the underlying optimal co-channel equalization solution.

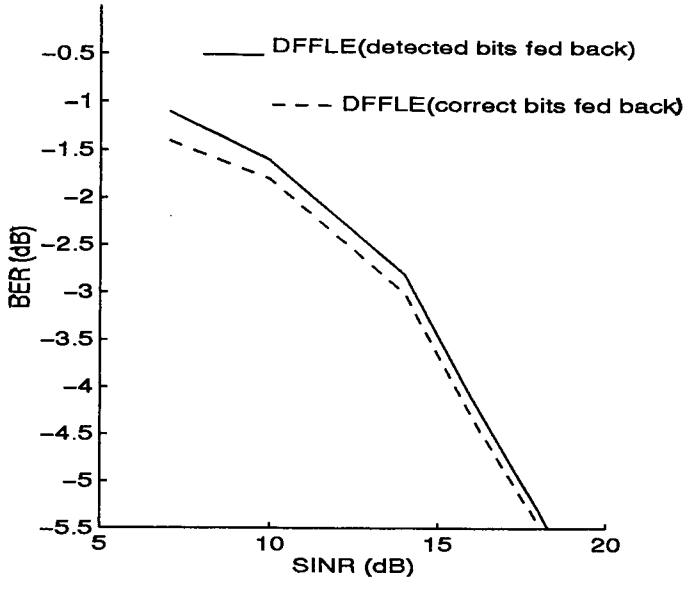

Figure 6.18: Error Propagation Effects in DFFLE-EFT for CASE I.

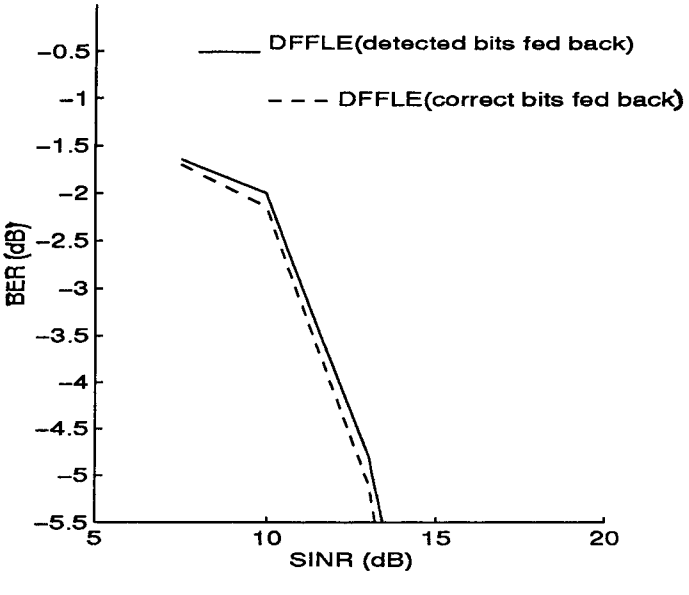

Figure 6.19: Error Propagation Effects in DFFLE-EFT for CASE II.

6.6 Conclusions

A new design strategy has been proposed for the FFLE which is shown to give useful insights into its computational complexity requirements with increasing input orders; in particular, it was shown that the increase in the size of its functional-link expansion model is not exponentional with increasing input dimensions.

New DFE structures have been presented namely the DFFLE-UFT and the DFFLE-EFT structures [177] [178] [179] together with their learning algorithms and design strategies. Key structural and computational complexity comparisons have been made between the new DFFLE structures and the FFLE, which show that the DFFLE-UFT has the minimal computational complexity requirements. The EKF algorithm has been applied to the FFLE and both the DFFLE structures in order to enhance their speed of MSE convergence characteristics. Pruning techniques for optimizing the sizes of the FFLE and DFFLE structures have also been proposed.

Various simulations have been performed for $2-ary$ PAM baseband systems to assess the performance of the new equalizers with other recently reported non-linear equalizer structures. The first application has served to show that the new DFFLE-EFT structure is a viable alternative to the the optimal symbol Bayesian TE (and all other non-linear equalizers that have been reported to-date to approximate the optimal Bayesian TE) for the equalization of linear and non-linear channels in the presence of ISI and additive noise, providing a better approximation to the underlying optimal sequence MLVA equalizer. The attraction of the DFFLE-EFT is not only its superior performance (in terms of error convergence speed and BERs), but also its simple integral structure (implying easier analysis) and a computationally efficient learning algorithm (important for tracking rapidly time varying channels). These characteristics will give the new structure a significant advantage over the Bayesian TE in terms of ease of practical implementation. Furthermore, unlike the Bayesian TE, the DFFLE-EFT requires no a priori knowledge or means of identification of the communication channel.

In the second digital communications application considered in this chapter, the FFLE and DFFLE structures have been proposed as an effective solution to the problem of overcoming co-channel interference. A severe amplitude distorted co-channel system has been used as an example to show that, relative to the feed-forward non-linear FFLE and RBF based equalizers, the DFFLE-EFT structure is not only computationally less expensive, but also significantly more effective in dealing with the co-channel interference problem, yielding a much closer approximation to the underlying optimal Bayesian DFE solution (which has been derived for co-channel systems in [196] by Chen and Grant *et al*). Also, unlike the optimal RBF based Bayesian TE reported in [190], no a priori knowledge of the orders of the distorting channel or the interfering co-channels is required in order to choose the size of the DFFLE functional-link expansion model $F(k)$ for superior performance. Note also that the relative effectiveness of the DFFLE-UFT (with the minimal computational complexity) in overcoming co-channel interference effects has yet to be investigated. The results in this study have considered single co-channel systems, but they can be readily extended to the multi-co-channel interference case.

Note that for both the digital communications applications considered in this chapter, the error-propagation effects in the DFFLEs were shown (by simulation) to result in very small performance degradation. Finally note that all results in this chapter were obtained by use of fully expanded (unpruned) FFLE and DFFLEs (resulting from use of the proposed design strategies), and their computational complexities can be further optimized by employing the pruning strategies proposed in section 6.4 [178].

Chapter 7

Conclusions and Future Work

This thesis has reported new ANN based structures and algorithms for two important signal processing applications namely, non-linear dynamical system modeling and digital communications applications.

Specifically, in the context of non-linear dynamical system modeling, a new Feedforward Functionally Expanded Neural Network (FFENN) was presented in chapter 4, and shown to consistently outperform other recently reported ANN based structures in the modeling of a large class of simulated and real-world non-linear dynamical time series processes. Its superior performance, both in terms of relative computational complexity requirements and modeling capability, is attributed to:

1. Firstly and most importantly, the nature of the proposed basis functions within the hidden layer, which were shown to emulate other universal approximators employed by the RBF, MLP and VNN structures. The numerous case studies carried out illustrated the respective contributions of the various non-linear basis functions responsible for the superior FFENN performance.

2. Secondly the linear in the parameters structure of the FFENN, which enabled fast least squares based learning in the output layer.

3. And finally the proposed pruning strategy, which was shown to consistently

result in parsimonious FFENN predictor models of complex non-linear dynamical processes.

Note that a study of all the optimally pruned FFENN predictor models evolved in the various case studies carried out in chapter 4, reveals an almost consistent absence of basis functions comprising the polynomial-subset outer-product expansion of the FFENN inputs. This suggests that compared to the other linear and non-linear basis functions proposed for inclusion in the FFENN's hidden layer, the outer-product terms are less significant in capturing the underlying dynamic system representation. The reason for this is attributed to the following fact: as discussed in section 4.2.3 the higher order, even and odd cross-terms namely, the squashing sigmoidal type $x_i \cos(x_j)$ basis function given by

$$x_i \cos(x_j) = x_i - x_i(x_j^2/2!) + x_i(x_j^4/4!) - , \ldots \text{ for } i \neq j, \ i, j = 1, \ldots, n$$

and the multi-quadratic type $x_i \sin(x_j)$ basis function given by

$$x_i \sin(x_j) = x_i x_j - x_i(x_j^3/3!) + x_i(x_j^5/5!) - , \ldots \text{ for } i \neq j, \ i, j = 1, \ldots, n$$

comprise higher order cross-products between each FFENN input with an infinite series of even and odd power expansion of the other FFENN inputs respectively. These basis functions can in fact account for the polynomial-subset outer-product activation functions to some extent (in fact, they account for all second order joint activation terms or outer-product expansion of the inputs). Therefore, the design strategy proposed for the FFENN in section 4.2.1 can be modified to delete the inclusion of the outer-product expansion model within the FFENN hidden layer $F(k)$ (that is, include steps 1 to 5 of section 4.2.1 and exclude step 6). This new design strategy will in turn, result in a dramatic reduction in the relative computational requirements of the FFENN with increasing input dimensions n, as reflected in Table 7.1. Table 7.1 compares the number of basis functions N employed in the previous FFENN expansion models $F(k$ (described in section 4.2.1) and the new expansion models $F(k)$ resulting from the above discussion. As can be seen from Table 7.1, a consequence of excluding the outer-

n	previous N	new N
1	8	8
2	20	19
3	38	34
4	64	53
5	102	76
6	160	103
7	254	134
8	416	169
9	710	208
10	1264	251
11	2334	298
12	4432	349
n	$1 + 2n^2 + 4n + \sum_{i=1}^{n} P_i^n$	$1 + 2n^2 + 5n$

Table 7.1: Comparison of new Design Strategy for (n, N)FFENN with previous Design Strategy (from Table 4.1)

product terms from FFENN's functionally expanded input model $F(k)$ is that it can now be more readily employed for modeling non-linear dynamical systems of very large input dimensions n (for example a $n = 21$ input system would require a FFENN model comprising fewer than a 1000 basis functions). The subsequent pruning times of these FFENN models would also be significantly reduced. Therefore for future work, the performance of the computationally more efficient FFENN predictor models resulting from the new design strategy described above, can be compared with the FFENN predictor models employed in chapters 4 and 5, and their application to modeling larger dimensional dynamical processes investigated.

Also for future work, extensions to the FFENN learning algorithm can include employment of the computationally more efficient Fast RLS type algorithms as discussed in chapter 4. The computationally efficient pruning strategy employed for the FFENN was based on an iterative pruning-re-training scheme, and employed a simple heuristic that assessed the relevance of the hidden nodes (basis functions) according to the magnitude of their corresponding weights. The hidden

nodes with small output weights tend to contribute less to the overall computation of the FFENN output, and thus are promising pruning candidates. The pruned network was then re-trained in order to achieve the desired performance on the training set prior to the next pruning step. This pruning scheme coupled with correlation and chi-squared statistic based model validity tests (used to validate the pruned FFENN model at each stage), is in fact completely general, and can thus be used to prune other ANNs such as the RBF, VNN as well as the MLP structures. It would be interesting to compare the results achieved by using our method with the other recently reported pruning strategies namely, the OLS algorithm [187], the optimal brain-damage (and surgeon) techniques [89] [107], and weight decay strategies [24]. Compared to the weight decay technique for instance, our method has the advantage of not requiring any a priori determination of a decay rate prior to the network training.

Also for future work, the FFENN's hidden layer comprising the functionally expanded input model can be be employed in hybrid multi-layered feedforward ANN structures; and also as a pre-processing input layer in the conventional Recurrent ANNs. The actual basis functions employed in the FFENN's expansion model could also be further optimised by, for example as in the RBF network, employing a learning algorithm in the input hidden layer in order to update the position and spread of the FFENN basis functions, by the inclusion of bias terms within each non-linear basis function. Computationally expensive evolutionary programming or multi-agent stochastic search techniques employed in [42] for evolving Recurrent (IIR type) and Transversal Perceptron networks, can also be employed for the pruning of FFENN (and RFENN) structures. Also for future work, the modeling of NARX (equation-error) type non-linear dynamical systems in the presence of correlated Gaussian and non-Gaussian noise sources needs to be investigated using the FFENN and other ANN models.

To enable further efficient modeling of a more general class of non-linear dynamical systems namely the output-error and NARMAX type processes, the FFENN was adapted to incorporate local output feedback thus resulting in a

new Recurrent Functionally Expanded Neural Network (RFENN). Its learning algorithm was derived and shown to possess a simpler computational requirement relative to the RTRL algorithm used for the training of conventional Recurrent Neural Networks (RNNs). This reduction in the relative complexity of the RFENN's learning algorithm was in fact achieved by employing non-linear basis functions only at the input single hidden layer of the RFENN, whereas in conventional multi-layered RNN structures non-linear basis functions are employed at both the hidden and output layers. The effective modeling capability of the RFENN was demonstrated through the use of both simulated and real-world time series processes. A new pruning strategy was also presented for the RFENN, which effectively made use of the optimally pruned expansion models evolved for its feedforward FFENN counterpart.

For future work, the RFENN structure can be readily employed in a Locally Recurrent Globally Feedforward (LRGF) architecture, as discussed in chapter 5. Structural and learning variations in the RFENN reported in chapter 5 can also be readily investigated. In particular, a detailed stability and convergence analysis for the RTRU needs to be carried out similar to that for the Recursive Prediction Error (RPE) type algorithms [181] using for example, stochastic Ordinary Differential Equation (ODE) methods. The ODE approach which has been conventionally employed to study the convergence analysis of adaptive IIR filters, is a powerful technique that requires relatively weak assumptions [79] [58]. However, it does not provide any information about the *rate* of convergence, only that an algorithm will asymptotically converge. As such, the convergence analysis of even conventional adaptive IIR filters is also somewhat limited and the problem is still largely unresolved [58]. Also for future work, simplifications to the RTRU learning algorithm can be investigated, such as the use of further approximations in determination of the RFENN output gradient estimates (similar to, for example, those used in the development of Pseudo-Linear Regression (PLR) type algorithms [58]). Omission of the Hessian matrix within the RTRU algorithm will also lead to the development of computationally efficient stochastic-gradient

(IIR-LMS) type algorithms [58]. Relative performance analysis can be carried out between these various types of RFENN learning algorithms. Also for future work, application of the RFENN to efficient modeling of output-error processes needs to be further investigated using additional simulated (and real) data, and compared with the conventional RNN structures.

Another point to note is that the RFENN structure presented in chapter 5 employs linear feedback terms, and a possible enhancement in its prediction (modeling) ability could be investigated by incorporating a non-linear expansion of the output feedback terms similar to that employed for the feedforward FENN input terms. The corresponding learning algorithms for this new structure can also be investigated. Note that, as mentioned in chapter 5, the architecture of the reported RFENN is completely general, in that it encompasses all other linear-in-the-parameters feedforward neural networks reported to-date, such as the RBF and VNN structures adapted to incorporate local output feedback, by simply substituting for their corresponding Gaussian and polynomial expansion models within the RFENN hidden layer's expansion model $F(k)$. Therefore, a relative performance evaluation can be carried between these different types of Recurrent Networks for various signal processing applications.

Finally in chapter 5, both the FFENN and RFENN structures were shown to be capable of performing efficient on-line (adaptive) non-linear prediction of highly non-stationary signals including real speech and laser time series data, significantly outperforming the conventionally used linear filterng approaches. A new hybrid RFENN-FIR adaptive structure was also developed and shown to be most effective in the modeling of a real speech signal, compared to both the stand-alone FFENN and RFENN structures (at the expense of increased computational complexity). Hence, further work could be carried out to investigate the role of these new structures (particularly the RFENN-FIR) as non-linear adaptive predictors in the Adaptive Differential Pulse Code Modulation (ADPCM) of speech for example, and their performance compared (both in terms of non-linear prediction ability and relative computational requirements) with the newly reported

cascaded Pipelined-Recurrent Neural Network [90]. In particular, application of the modified design strategy described earlier in this chapter for specifying the basis functions within the functionally expanded model $F(k)$ (employed in both the FFENN and RFENN and RFENN-FIR models), will result in a dramatic reduction in the relative computational complexities of the resulting adaptive FFENN, RFENN and RFENN-FIR models (as reflected in Table 7.1). This could in turn, further enhance their real-time prediction capability and enable them to be more effectively employed for adaptive modeling of larger dimensional MIMO time series processes. Therefore for future work, their relative performances (resulting from use of the modified design strategy described above) can be investigated for the time series processes modeled in chapters 4 and 5, and also for other simulated and real-world non-linear dynamical processes.

Also for future work, the FFENN and RFENN structures can be applied to control applications as a viable alternative to the conventionally employed multilayered MLP and RNN structures [34]. Use in other signal processing applications such as active noise cancellation, and image processing can also be readily investigated. Unsupervised learning in the FFENN can also be investigated, which will further widen the range of application areas to include, for example pattern classification.

In the context of digital communications applications, the key structural and computational aspects of the conventional FFLE were first highlighted in chapter 6. A new general design strategy was also presented, which was shown to give highly useful insights into the computational requirements of the FFLE with its increasing orders. Specifically, it was illustrated that the increase in computational complexity of the FFLE for $2 - ary$ PAM signalling is not exponential as previously conjectured [163] [164]. Two new DFE structures were presented namely the DFFLE-EFT and DFFLE-UFT. The DFFLE-UFT employs the non-linear FFLE as its feedforward filter, whereas the feedback filter is linear. In

contrast, the novel DFFLE-EFT structure non-linearly combines both the equalizer input and decision feedback samples. Learning algorithms were presented for both the structures along with their design strategies. They were shown to possess a significantly simpler computational requirement relative to the FFLE. The Extended Kalman Filter (EKF) learning algorithm was also proposed and applied to the FFLE and DFFLE structures in an attempt to enhance their speed of output error convergence characteristics. In the first application, the new structures were applied to the task of equalization of linear and non-linear digital communications channels in the presence of ISI and both correlated and uncorrelated white noise sequences. The new structures were shown (by simulation results) to outperform the FFLE both in terms of speed of MSE convergence and BER characteristics. In particular, the DFFLE-EFT structure was shown to offer the best performance (closest to the optimal sequence MLVA), outperforming the optimal symbol Bayesian (or MAP) equalizer and all other recently reported ANN based non-linear equalizers reported to date.

Finally in chapter 6, the FFLE and DFFLE structures were proposed as a novel solution to the problem of overcoming co-channel interference in digital communications systems in the presence of ISI and AWGN. For a severe amplitude distorted co-channel system, the DFFLE-EFT was shown to provide a viable alternative to the LTE, FFLE and the recently reported RBF based equalizers in approximating the underlying optimal symbol Bayesian solution. Note that the performance of the DFFLE-UFT in combating co-channel interference has yet to be investigated.

For future work, the performance of the DFFLE-EFT (and DFFLE-UFT) needs to be compared with that of the recently reported optimal Bayesian DFE in the equalization of linear and non-linear channels. For linear channels, such as the one employed in chapter 6, the optimal symbol Bayesian DFE has been shown in [195] to provide a performance close to the optimal sequence adaptive MLVA for small values of equalization delay τ. However, in our knowledge the Bayesian DFE has not been applied to the equalization of non-linear dispersive

channel models to-date. The performance of the Bayesian DFE (which like the Bayesian TE uses Bayes decision theory for making decisions) would obviously be expected to be superior to the non-linear DFFLE-EFT or the DFFLE-UFT, both of which do not fully exploit the links with Bayes decision theory. However, just as computationally efficient non-linear TEs have been used to approximate the computationally expensive Bayesian TE, similarly the DFFLEs or other DFE based neural network equalizer structures could also be investigated to provide a good approximation to the optimal symbol Bayesian DFE, which like the Bayesian TE, requires apriori knowledge (and hence estimation) of the channel characteristics. The optimal Bayesian DFE based solution has also been recently derived for the problem of co-channel interference suppression in digital communications systems [196], and its performance can be compared with both the DFFLE-EFT and DFFLE-UFT structures. A performance comparison of all these structures with various recently reported reduced state MLVA based equalizers [170] [29] has also yet to be carried out. Also in chapter 6, single co-channel systems have been considered and therefore, the multi-co-channel interference scenario can be readily simulated next.

Also for future work, the fully expanded FFLE, DFFLE-UFT and DFFLE-EFT structures can be pruned using the pruning strategies proposed in chapter 6. In particular the use of OLS selection algorithm can be investigated in forming optimal sized equalizers, and compared to the other proposed pruning strategies. It is also important to note that all results presented in chapter 6 were for $2-ary$ baseband PAM systems, and therefore extensions of the reported equalizer structures to multi-level $M-ary$ PAM and QAM based systems need to be investigated and compared with the existing techniques [194]. One way to equalize $M-ary$ PAM signals would be to employ multi-output FFLE and DFFLE structures (similar to the multi-output RBF equalizer reported by Chen and Grant *et al* [189]) with one output node assigned to each class. However, with this approach the equalizer complexity will increase in proportion with the number of transmitted symbol categories. Alternatively, a single multi-level sigmoidal function (recently

employed in an efficient MLP based equalizer for $M-ary$ PAM signalling [166]) can be employed at the FFLE and DFFLE output layer (replacing the two-level sigmoid tanh(.)). This multi-level sigmoidal activation function is given by:

$$\hat{s}_0(k-\tau) = \sum_{j=-((M/2-1)}^{(M/2-1)} \frac{1-e^{(-\alpha o_k+hj)}}{1+e^{(-\alpha o_k+hj)}} \tag{7.1}$$

where o_k is the output from the FFLE/DFFLE linear combiner, and the constants α and h are referred to as the slope and drift parameters respectively (which have been shown in [166] to give efficient equalization performance with values set to $(\beta = 1.0)$ and $(h = 15)$). The functions used within the FFLE and DFFLE hidden layer input expansion models (described in their respective design strategies in chapter 6) also need to be modified to accommodate $M-ary$ PAM signalling. This would simply involve scaling all the functional terms employing the trigonometric $\sin(\pi.)$ and $\cos(\pi.)$ activation functions by a scaling factor A, which for $M-ary$ PAM will be $A = (M-1)$. This scaling factor will enable these functional terms to cover the new range $(-(M-1),(M-1))$, from their previous $(-1,+1)$ range employed for $2-ary$ PAM signalling. The two-level signum function $sgn(.)$ employed in the FFLE and DFFLE hidden layers would also need to be replaced by a multi M−level signum threshold function for $M-ary$ signalling. The other terms within the FFLE and DFFLE input expansion models namely the equalizer input (and decision feedback terms), and the joint-activation terms would not need any modification. With these proposed design changes (which can be readily incorporated into the respective design strategies of the FFLE and DFFLEs), simulation case studies can be carried out similar to those in [164] in order to assess the relative importance of these various proposed functions in the equalization of $M-ary$ PAM and complex QAM based systems. The growth in relative complexities of the FFLE and DFFLE structures with increasing size of the transmitted signal alphabet can thus be investigated. The results can be compared with those recently reported for the optimal symbol Bayesian DFE, whose complexity has been shown to increase rapidly with both the size of the transmit alphabet M and length $l+1$ (see equation 3.1) of the

communication channel, of the order M^{l+1} [198] [129]. In fact, these excessive computational requirements of the Bayesian DFE have restricted its application to small signal alphabets such as 4-QAM and channels where the ISI extends over four or five symbol periods (as in mobile radio environments) [129]. The limitations of the FFLE and DFFLEs could also be similarly established.

Also, the use of the computationally expensive EKF algorithm proposed for updating the FFLE and DFFLE weights in chapter 6 needs to be further justified by investigating the trade-off between speed performance improvements and corresponding computational complexity requirements of the FFLE and DFFLE structures, for other linear, non-linear, and non-stationary channels.

Also for future work, application of the FFLE and DFFLE structures to Code Divison Multiple Access (CDMA) Spread Spectrum (SS) communications systems [129] can be investigated, and their performance compared with the recently reported MLP, RBF and adaptive Bayesian methods which have been successfully applied to CDMA systems to-date [129]. The use of the FFLE and DFFLE structures in blind adaptive equalization schemes can also be investigated. This can also include these equalizers operating in hybrid systems, where for example, a computationally expensive Higher Order Statistics (HOS) based blind equalizer [5] initially estimates the channel coefficients, and the system then switches to the computationally efficient FFLE, DFFLE or other ANN based adaptive non-linear equalizers operating in Decision Directed Mode (DDM). Finally, application of the FFLE and DFFLE structures to equalization of non-stationary communication channels (such as those encountered in fading mobile radio environments) can also be readily investigated. This would simply require an additional adaptive (on-line) channel estimator; which to-date has conventionally employed the FIR-LMS approach [194]. This approach works fine for the linear channel case, but would be unable to model the non-linear channel characteristics. We propose use of the computationally efficient FFENN and RFENN structures presented in chapters 4 and 5, for adaptive identification of both linear and non-linear dispersive channels. Note also that, similar to the recently reported RNN based

equalizer, the use of the RFENN reported in chapter 6 as an efficient adaptive non-linear equalizer structure can be also be investigated, and compared with the existing structures.

Appendix A

Review of Feedforward Artificial Neural Networks

We begin this section with a brief note on the Perceptron which is the basic building block of the MLP network, and also perhaps the most well known neural network due to its historical significance [19].

A.1 The Perceptron

The Perceptron was devised by Rosenblatt in 1958 [65]. As shown in Figure A.1, it first forms a weighted sum of the n components of the input vector (similar to the McCulloch and Pitts neuronal model [61] which is said to have pioneered the modern era of neural networks [56]). A bias value w_0 is added to the weighted sum and the result is then passed through a non-linearity (activation function). In the original Perceptron, Rosenblatt used a hard-limiting non-linearity (quantizer), which classifies the output y to be $on = 1$ or $off = 0$ depending on whether the weighted sum output u is greater than or less a certain threshold, 0 for example. The Perceptron learns or trains by adjusting the weights to allocate correctly, training-set inputs to the on or off classes. There are various learning algorithms for the Perceptron which include the original Perceptron learning algorithm [65],

the Least Mean Squares (LMS) algorithm devised by Widrow [70] and others [73].

Perceptions of the Perceptron were dealt a fatal blow by Minsky and Papert [44] who showed that Perceptrons (and their variants the Adaline networks which were developed later by Widrow in 1959 [70]) could only linearly separate their input classes, and so could not solve problems requiring non-linear decision boundaries, such as the exclusive-or function and its generalization, the parity function. Rosenblatt had considered Perceptrons with more than one layer and thought they could overcome the limitations, if only a learning algorithm could be found. However, it was not until 1986, when Rumelhart formally introduced the Back Propagation algorithm (which was actually originally invented by Werbos in his PhD thesis in 1974 [74]) for training Multi-Layered Perceptron (MLP) networks [75] (which are discussed in the next section), that it became possible for neural networks to solve complex large class pattern recognition problems. Primarily, Rumelhart's work and and an earlier paper presented by Hopfield in 1982, identifying the associative memory properties of neural networks namely recalling of a stored pattern from an imperfect input reference [76]; have been the most influential publications responsible for the resurgence of interest in ANNs in the 1980s.

A.2 Multi-Layered Perceptron (MLP)

The Multi-layered Perceptron (MLP) has one or more hidden layers between the input and output layers. In general, the MLP can have both feedforward and feedback connectivity between its layers. An MLP with feedforward connectivity belongs to the category of Feedforward ANNs. An MLP network with two hidden layers is illustrated in Figure A.2.

The individual Perceptrons in the MLP are called neurons (or nodes), and differ from Rosenblatt's Perceptron in that a sigmoidal non-linearity (activation function) is used in place of the hard-limiter. This is primarily because the sigmoid is differentiable (an essential property required for derivation of the

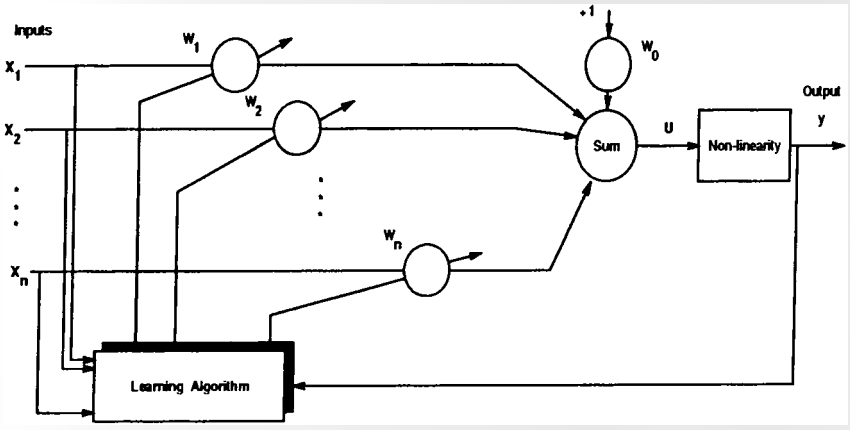

Figure A.1: The Perceptron

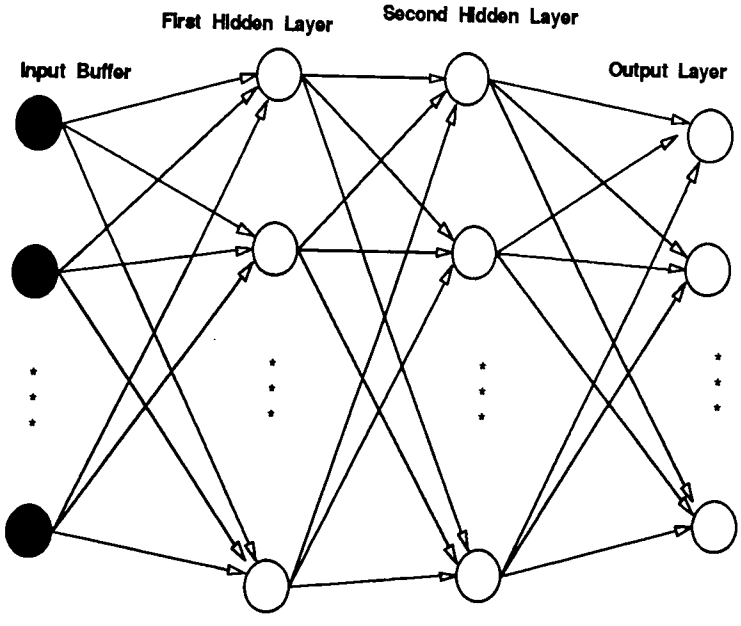

Figure A.2: Architecture of a typical Multi-Layered Perceptron

BP algorithm discussed in the next section), whereas the hard-limiter is non-differentiable. Furthermore, the logistic sigmoidal function offers other benefits namely, continuous valued outputs (between 0 and 1) rather than binary alone, and the possibility of output value's interpretation as a probability estimate [80]. However, for many applications such as equalization of digital communication channels (discussed in chapter 3), other continuous valued differentiable sigmoidal functions such as the $tanh(.)$ function (which is a biased and scaled logistic function) can provide additional benefits of giving outputs between -1 and $+1$ rather than positive only (between 0 and 1).

A.3 The Back Propagation Training Algorithm

The Back Propagation (BP) algorithm (also known as the Generalized Delta Rule (GDR)), is outlined below:

Defining the error signal at the output of neuron j at iteration n (that is, presentation of the nth training pattern) as:

$$e_j(n) = d_j(n) - y_j(n), \quad \text{neuron j is an output layer node} \tag{A.1}$$

where $d_j(n)$ is the desired response for neuron j, and $y_j(n)$ is the actual output of neuron j. The instantaneous sum of squared errors of the network is thus written as:

$$E(n) = \frac{1}{2}\sum_{j=1}^{L} e_j^2(n) \tag{A.2}$$

where the set L includes all neurons in the output layer of the network. The output $y_j(n)$ of neuron j is given by:

$$y_j(n) = f(\sum_{i=0}^{p} w_{ji}(n)y_i(n)) \tag{A.3}$$

$$y_j(n) = f(v_j(n)) \tag{A.4}$$

where $w_{ji}(n)$ denotes the synaptic weight connecting the output of neuron i to the input of neuron j. Neuron i lies in a hidden layer immediately preceding the

output layer, and produces an output $y_i(n)$, with p representing the total number of neurons in that layer. f is the non-linear activation function of the jth neuron and $v_j(n) = \sum_{i=0}^{p} w_{ji}(n)y_i(n)$ is termed the net internal activity of neuron j. The weight $w_{j0}(n)$ corresponding to a fixed input ($y_0 = -1$) is the threshold applied to neuron j.

The Back Propagation (BP) algorithm applies a correction $\Delta w_{ji}(n)$ to the synaptic weight $w_{ji}(n)$, which is proportional to the instantaneous gradient $\partial E(n)/\partial w_{ji}(n)$ as follows:

$$\Delta w_{ji}(n) = -\eta \frac{\partial E(n)}{\partial w_{ji}(n)} \tag{A.5}$$

where η is a constant that determines the rate of learning. The use of the negative sign in equation A.5 above accounts for *gradient descent* in weight space.

The correction or change in weight of any neuron in the output layer can be readily written as:

$$w_{ji}(n) = w_{ji}(n-1) + \eta\sigma_j(n)y_i(n) \tag{A.6}$$

where the local gradient $\sigma_j(n)$ for each of the output layer nodes is given by:

$$\sigma_j(n) = e_j(n)f'(v_j(n)) \tag{A.7}$$

where $f'(.)$ denotes the derivative of the non-linear neuronal activation function with respect to its net input $v_j(n)$. Equations A.6 and A.7 above are collectively known as the Delta Rule (DR) updating algorithm.

The change in weight of any neuron i in a hidden layer immediately preceding the output layer is given by:

$$w_{ih}(n) = w_{ih}(n-1) + \eta\sigma_i(n)y_h(n) \tag{A.8}$$

where $y_h(n)$ is the output of neuron h residing in a layer immediately preceding the hidden layer containing neuron i. The local error gradient $\sigma_i(n)$ for any neuron i in the hidden layer can be written as [56]:

$$\sigma_i(n) = f'(v_i(n)) \sum_{j=1}^{L} \sigma_j(n)w_{ji}(n) \tag{A.9}$$

The above equation A.9 in fact applies to all the MLP network's hidden layer nodes, and in words, states that the local error gradient for a hidden layer node a equals the product of its associated derivative (with respect to its net input) and the weighted sum of the error gradients computed for all the neurons in the *next* hidden (or output) layer that are connected to that neuron a. It can also be seen from the above equation, that the local error gradient $\sigma_j(n)$ for the output layer has been propagated back to the lower (immediately preceding) hidden layer. This procedure is repeated until the error has been propagated back to the network's first hidden layer and all the corresponding weights updated.

Therefore to summarize, the application of the BP algorithm involves two phases. During the first phase, the input pattern is presented to the network and all neurons are updated to produce an output. During the second phase, the error for the output layer is computed. This error is then propagated backwards (hence, the name of the algorithm) to the lower layers and the weights adjusted accordingly. The algorithm is balanced as the backward pass has the same computational complexity as the forward pass.

The Functional Capability of the MLP

In general, the functional capability of the MLP can be viewed from three different perspectives:

- Firstly, the ability of the MLP to partition the pattern space for classification problems. Lippmann [21] has demonstrated that a 2-hidden layer MLP can implement arbitrary convex decision boundaries for classification problems. Later it was shown that it can form an arbitrarily close approximation to any non-linear decision boundary [83], by combining hyperplanes formed by the hidden layer units

- Secondly, the ability of the MLP to implement Boolean functions also illustrate the MLP's functional capability. Morgan [84] has shown that a 2-hidden layer MLP can implement any arbitrary logic function (since only

two layers of fundamental operations -ANDs and ORs are needed to implement any logic function).However an extremely large number of hidden layer nodes are needed to implement certain logic functions [92]. Sun et al [94] has argued that with the addition of feedback connections and a mechanism to implement a unit time delay, the MLP can simulate a complete digital computer.

- Thirdly, the MLP capability can be viewed from its ability to implement non-linear transformations for problems involving functional approximation. It has been shown that a 2-hidden layer MLP can form an arbitrarily close approximation to any continuous non-linear mapping by combining rounded step functions formed by the hidden layer nodes [105].

Note that the reported research results which show that a 2-hidden layer MLP can implement an arbitrary function do not imply that any more benefit would be gained by having more than two layers [24]. This is because for many problems, an approximation with 2-hidden layers could require an impractically large number of hidden units, whereas an adequate solution could be obtained with a tractable network size by using more than two hidden layers. Typical problems of such nature have been reported in [106].

It has been recently shown in [24] that the MLPs are good at both classification and in estimating *a posteriori* probabilities (or confidence estimates). In some applications, this gives them a distinct advantage over other techniques. More recently, Haykin [57] has shown in a new case study on image compression and segmentation, that properly trained MLP based structures can outperform other conventional approaches including the optimal Mean Square Error (MSE)criterion based Karhunen-Loeve Transform (KLT).

A.4 Radial Basis Function (RBF) Neural Network

A RBF network [151] [12] depicted in Figure A.3 is a single hidden layered (or two layered) FANN. The RBF is a linear in the parameters (with respect to the output layer weights) neural network, whose response is linear function of its weights. Specifically, each RBF output layer node forms a weighted linear combination of the basis (or kernel) functions computed by the hidden layer nodes. Each hidden layer node contains a parameter vector called a center, and calculates the Euclidean distance between the center and the input vector. The result is then passed through the node's non-linear basis function. The basis functions produce a localized response to input stimuli, that is they produce a significant non-zero response when the input falls within a small localized region of the input space [124]. Mathematically, an RBF network with n inputs and m outputs, implements a mapping $f : R^n \rightarrow R^m$ according to:

$$y_j = f(\mathbf{x}) = w_0 + \sum_{i=1}^{N} w_i \phi(||\mathbf{x} - \mathbf{c}_i||) \quad \text{for } j = 1, \ldots, m \tag{A.10}$$

where $\mathbf{x} = [x_1 \ldots x_n]$ is the input vector, $\phi(.)$ are the non-linear basis functions which transform the input into an intermediate non-linear hidden space of an increased dimension N; $||.||$ denotes the Euclidean norm, w_i $i = 0, \ldots, N$ are the tap weights, $\mathbf{c}_i$, $i = 1, \ldots, N$ are the centres of the basis functions, and N is the number of RBF centers (equal to the number of hidden nodes). The centers are some fixed points in the $m-$dimensional input space, which they must suitably sample [12].

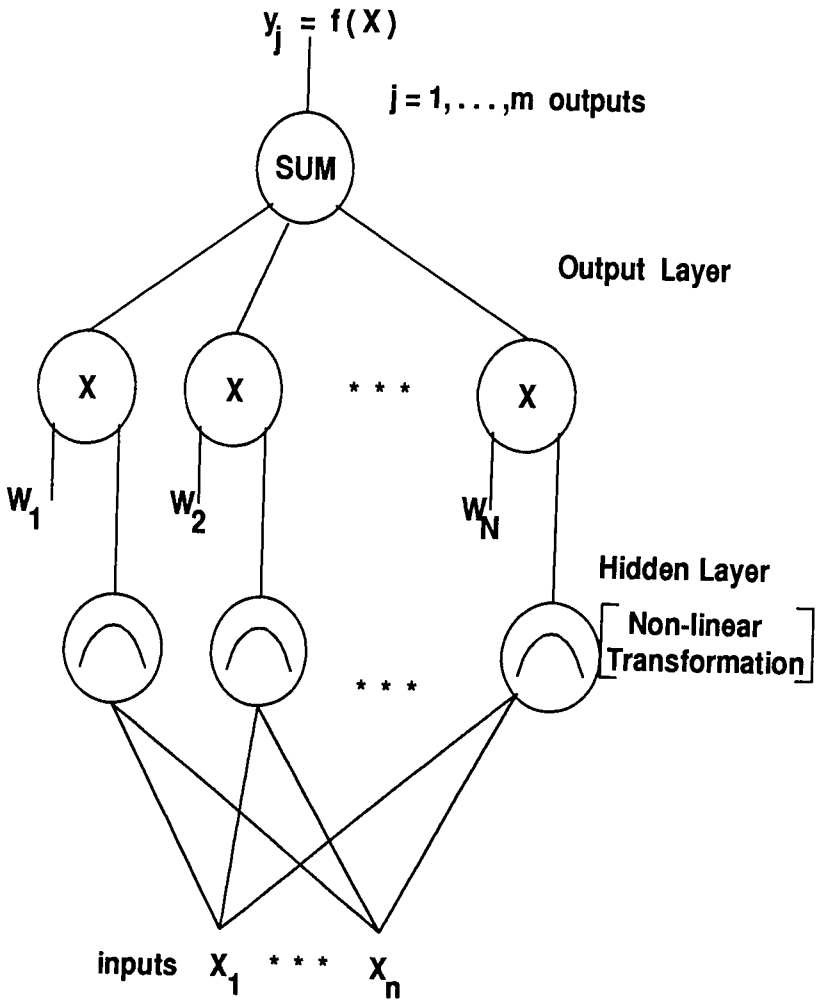

Figure A.3: Schematic of a Radial Basis Function (RBF) Network

Common choices for the non-linear basis functions $\phi(u)$ employed include [187]:

- The thin-plate spline function:

$$\phi(u) = u^2 log(u) \tag{A.11}$$

- The multi-quadratic function

$$\phi(u) = \sqrt{(u^2 + \sigma^2)} \tag{A.12}$$

- The inverse multi-quadratic function:

$$\phi(u) = \frac{1}{\sqrt{(u^2 + \sigma^2)}} \tag{A.13}$$

- And the most widely used Gaussian kernel:

$$\phi(u) = exp(-u^2/\sigma^2) \tag{A.14}$$

where σ is a real constant usually termed the width of the basis function. The name Radial Basis Function network infact comes from use of the Gaussian basis functions, which are radially symmetric, in that each node produces an identical output for inputs that lie a fixed radial distance from the center of the kernel. An extension of the RBF network includes variation on the basis functions by use of the Mahalanobis distance in the Gaussian kernels in order to increase their functionality [130] [16].

A.4.1 Learning Algorithms

Numerous approaches have been proposed for the training of RBF networks [24] [56]. The linear in the parameters structure of the RBF gives it a distinct advantage over the MLP in learning. Basically, once the hidden layer parameters are

fixed using for example, an unsupervised algorithm such as the $k-$means clustering technique [141], then because the network output is linear in the weights, fast least squares based algorithms such as the RLS can be used in the output layer to provide a means for real-time adaptation of the weights [192]. In another approach [116], once the initial learning in the hidden layer using the $k-$means clustering algorithms is completed, a supervised learning method can be simultaneously applied to both hidden and output layer to fine tune the parameters of the network.

Chen *et al* [187] have also shown that the choice of RBF centers is crucial to its performance. They suggested use of the Orthogonal Least Squares (OLS) algorithm for selecting the centers of the Gaussian basis functions as a subset of the training data, as opposed to the conventional method of random center selection. In the OLS approach, the samples are chosen one at a time in such a way that each new sample maximizes the amount of incremental gain in explaining the variance of the desired output. This technique is attractive as it provides a means for automatic determination of the number of hidden layer nodes, and hence the size of the RBF required for the application in hand. They have successfully applied the OLS method for efficient RBF model selection for non-linear system modelling and digital equalization applications. The OLS algorithm can then be followed by fast least squares based learning in the output layer.

Other learning algorithms, which incorporate methods for determining the number of hidden layer nodes can be found in [130]. However, all these methods have been devised specifically for application of the RBF network to solving pattern recognition problems, and involve use of supervised hierarchical clustering algorithms in the hidden layer, which either start with one node and create additional nodes as needed [108]; or alternatively, start with a large number of nodes and merge those of the same class together [130]. Both these techniques, also adapt the widths of the basis functions during the learning process in order to minimize the overlap between neighbouring nodes of opposite classes.

A.4.2 RBF Functional Capability

Like the MLP, the RBF can be used for both classification and functional approximation [24]. In theory, the RBF network, like the MLP is capable of forming an arbitrarily close approximation to any continuous non-linear mapping [88]. The topology of the RBF network is very similar to that of the two-layer Perceptron [192]. The primary difference between the two is in the nature of the basis functions, with the hidden layer nodes in the MLP employing sigmoidal basis functions which are non-zero over an infinitely large region of the input space, whereas the basis functions in the RBF network cover only small localized regions. Hush *et al* [24] have recently shown that some problems such as functional approximation, can be solved more efficiently with sigmoidal basis functions; while others such as classification problems are more amenable to localized (*e.g.* Gaussian) basis functions. One unified approach would be to employ **both** these types of basis functions within a single neural network layer so that the distinct approximating capabilities of both the MLP and RBF networks can be employed. This approach is developed in this thesis to yield a new linear-in-the-parameters Feedforward Functionally Expanded Neural Network (FFENN) in chapter 4, and the resulting structure is shown to outperform both the MLP and RBF based networks in a variety of simulated and real-world non-linear dynamical system modeling applications.

A.5 Other Feedforward Neural Networks: The Volterra and Functional-Link Neural Networks

Thus far, although this section has focused primarily on the MLP and RBF networks, various other FANNs have also been reported to date (see for example

[24]). However, the MLP and RBF networks are arguably the most popular FANNs [24] [56] [19]. Next, we shall briefly describe two of the recently reported FANNs, namely the linear-in-the-parameters Volterra Neural Network (VNN) and the non-linear-in-the-parameters Functional Link Neural Network (FLNN).

A.5.1 Volterra Neural Network

The single-hidden layered (or two-layered) VNN, based on the non-linear Volterra Filter [60], was proposed by Rayner *et al* [63]. The structure of the linear in the parameters VNN is similar to the RBF network illustrated in Figure A.3, with the difference being in the nature of the non-linear basis functions employed in the hidden layer. The VNN employs polynomial basis functions as opposed to Gaussian basis functions. The output of the VNN is related to its inputs via the discrete Kolmogorov-Gabor polynomial expansion [77] [60] (which is closely related to the Volterra expansion for a non-linear dynamic system [60]):

$$\begin{aligned} y &= w_0 + \sum_{i_1=1}^{n} w_{i_1} x_{i_1} + \sum_{i_1=1}^{n} \sum_{i_2=1}^{n} w_{i_1 i_2} x_{i_1} x_{i_2} + \dots \\ &\quad + \sum_{i_1=1}^{n} \sum_{i_2=1}^{n} \dots \sum_{i_k=1}^{n} w_{i_1 i_2 \dots i_k} x_{i_1} x_{i_2} \dots x_{i_k} \end{aligned} \tag{A.15}$$

The above is specified as an (n, k) VNN, that is, n network inputs and upto $k-$th order product terms. The form of the non-linear polynomial expansion model for a (3,2)VNN comprises the following 13 hidden nodes [63]:

$$(1, x_1, x_2, x_3, x_1^2, x_1x_2, x_1x_3, x_2x_1, x_2^2, x_2x_3, x_3x_1, x_3x_2, x_3^2)$$

Hence, like the RBF, each output node of the VNN forms a weighted linear combination of the non-linear (polynomial) basis functions computed by the hidden layer nodes as illustrated in equation 2.15 above.

Hence, learning in the VNN output layer can be achieved by the use of stochastic gradient based algorithms such as the LMS algorithm used in [63], or alternatively the least squares based RLS algorithm. The problem with the

VNN is that the number of polynomial terms (or hidden nodes) grow exponentially with increasing input dimensions, and hence a very large number of terms from the Volterra series expansion may be required in order to achieve the desired approximation ability [64]. Rayner *et al* [64] proposed a self-structuring LMS algorithm for adaptively pruning the redundant terms in the polynomial expansion model. However, further work is required in order to determine the scaling potential of their algorithm to larger input complex real-world systems.

A.5.2 Functional Link Neural Network (FLNN)

The conventional static FLNN was developed by Pao [31] for functional approximation. It is a two-layered (or single hidden layered) FANN, whose response unlike the RBF and VNN, is a non-linear function of its weights, given by:

$$y_j = f(\sum_{i=1}^{N} w_i g_i(\mathbf{x})) \quad \text{for } j = 1, \ldots, m \text{ outputs}$$

where $(g_i(\mathbf{x})\ ,\ i = 1, \ldots, N)$ constitute the functional-link model which expand the n dimensional input $\mathbf{x}$ into a non-linear hidden space of increased dimension N, $f(.)$ is a differentiable output activation function commonly chosen to be the $tanh(.)$ sigmoidal activation function [31] [136], and w_i are the output layer weights. The structure of the FLNN is in fact equivalent to a Single Layered Perceptron (SLP) with a functionally transformed input [164].

Pao showed that the functional approximation capability of the FLNN is enhanced if the input functional-link expansion model is chosen to comprise an outer product model of the input data and a subset of the functional (trigonometric) components of the input data. The outer product model truly introduces higher order terms in the enhanced input representation in the sense that some of these terms represent joint activations. In contrast, the trigonometric functional expansion model expands the dimension of the representation space without introducing joint activations. In this model each input node is acted upon individually and the functions may comprise a subset of a complete set of orthonormal basis functions spanning a n-dimensional representation space such as $\sin(\pi\mathbf{x})$,

$\sin(2\pi\mathbf{x})$,$\cos(\pi\mathbf{x})$, $\cos(2\pi\mathbf{x})$ ... *etc.* However, in subsequent use of the FLNN for noise cancellation and system identification applications [136] [32] [191], a purely polynomial expansion model of the inputs has been used to-date, which makes the FLNN functional-link expansion model strictly identical to the corresponding model used in the Volterra Neural Network (VNN). In general however, the functional-link expansion model of the FLNN is not restricted to a polynomial only expansion [31]. Recently Gan *et al* [163] [164] devised an adaptive non-linear equalizer based on the above conventional FLNN for efficient equalization of both linear and non-linear digital communication channels in the presence of Inter-Symbol Interference (ISI) and additive noise. The use of the conventional FLNN in digital communications applications is further discussed in chapters 3 and 6.

Note that the learning algorithm employed by the conventional FLNN is the non-linear Delta Rule (DR) [136] [164] which forms the basis of the computationally complex Back Propagation (or Generalized Delta Rule) algorithm described earlier. Pao [31] has also showed how unsupervised learning can be accomplished in the FLNN. Other unsupervised methods based on the Adaptive Resonance Theorem (ART) [14] and the Kohonen's Self Organizing Map (SOM) methods [72] [54] can also be used. In this thesis, we shall focus our attention on supervised ANNs.

In chapter 4, a new linear-in-the-parameters Feedforward Functionally Expanded Neural Network (FFENN) structure is devised which differs from the above network, firstly in that its response is a linear function of its weights; thus enabling fast least squares based learning similar to that used in the RBF; thereby guaranteeing convergence to the single global minimum of its uni-modal output error surface. Secondly, new non-linear basis functions have been proposed for the FFENN's single hidden layer (functional expander) that are shown to emulate other universal approximators namely the MLP's squashing type sigmoidal

activation functions and the RBF's Gaussian bell shaped and multi-quadratic type activation functions. These characteristics of the new FFENN coupled with proposed general design and pruning strategies, are shown to give it distinct advantages over the conventional neural network structures in the modeling of a large class of both simulated and real-world non-linear dynamical systems.

A.6 Feedforward ANNs: Issues and Limitations

This section reviews issues and limitations that relate to most Feedforward ANNs (FANNs).

Slow Training

The BP algorithm which is mainly used for the training of feedforward multi-layered structures such as the MLP, implements gradient descent and is intrinsically slow to converge in a complex landscape, due to the complexity of the energy surface. The network requires presentation of the whole set of training input-out pairs (termed an epoch), and usually many times over for the network to be able to create internal representations for small problems [56]. However, for large problems it is possible to train the network over a statistically valid subset of the input space and then make use of the generalization properties to achieve correct classification.

The form of input training data presentation plays a significant role in determining the number of iterations required to train the MLP. A common cause of problems stems from presenting data to the Back Propagation network as raw data. In most cases some preprocessing of the input data is required and this often becomes the computational bottle-neck.

The best way to train the network on a large problem is to start the learning process with a large value of the learning coefficient η (near one) and then gradually reduce it. Also, assigning different ηs to each layer, and using a higher value of η for the first hidden layer than all the succeeding hidden layers and the output

layer, has been shown to speed convergence [120]. Vogal et al [97] have also shown that the convergence can be accelerated by incorporating a *momentum* term into the weight adaptation rule:

$$\Delta w_{ji}(n) = \eta \sigma_j(n) y_i(n) + \alpha \Delta w_{ji}(n-1) \tag{A.16}$$

where α controls the strength of influence of the previous change. This momentum term smooths the learning process by helping to avoid oscillations. It also acts like a short term memory and discourages rapid weight changes [56].

Convergence rates can also be improved by altering the learning coefficient adaptively as training proceeds. Other attempts to speed convergence include variations on simple gradient descent [96], line search methods [23], and second order methods [99]. Although most of these have been successful, they usually introduce additional parameters which are difficult to determine, must be varied from one problem to the other, and if not chosen properly can actually slow convergence rates. Recently, Kariayiannis [100] has developed fast learning algorithms for feed-forward neural networks based on the minimisation of an alternative criterion after initial adaptation cycles. The potential application of his proposed criterion is an interesting problem for future research.

For linear-in-the-parameters feedforward ANNs with a single hidden layer, such as the RBF, the VNN and the new Feedforward Functionally Expanded Neural Network (FFENN) developed in chapter 4, since the output is a linear function of the weights, fast least squares based algorithms can be used to provide a means for real-time weight adaptation.

Choice of correct Network size

As stated earlier, Cybenko [105] has shown that the MLP can form close approximations to arbitrary non-linear mappings provided that the network size grows arbitrarily large. Choosing the correct network size is therefore vital in that if the network is too small, it will not be capable of forming a good representational model of the problem; whereas on the other hand, if the network is too big then

the solution learned during any given learning trial will most likely be a poor approximation to the actual problem [133]. In general, the optimal (finite) network size will vary from problem to problem, since each problem will require different capabilities from the network.

Whilst there has been a lot of research into the analytical determination of both the number of hidden layers and the number of units in each layer required to solve a problem, no satisfactory general rule exits. With little or no prior information about a problem, the network size has to be determined by trial and error. A useful approach is to *grow a network* i.e. start with a single node and then create additional nodes as they are needed during the training process. Approaches that include such a technique include the Cascade Correlation [47], Projection Pursuit [50], the Algorithm for Synthesis of Polynomial Networks (ASPN) [8] and the Group Method of Data Handling (GMDH) [8]. An alternative approach is to start with a large network and then apply a pruning technique to destroy weights that contribute little or nothing to the solution [107]. However, with this approach an idea of how *large* a network should be for a particular problem is required [24].

For a 2-hidden layer MLP which is the most widely multi-layered FANN structure used, numerous bounds on the number of hidden nodes exist. For example it has been shown [92], that the number of hidden nodes should always be much less than the number of training samples, otherwise the network would simply learn the training data resulting in poor generalization. Bounds on the number of hidden nodes expressed as a function of n (the input pattern dimensionality) have also been considered. Whilst there are problems requiring number of hidden nodes to be an exponential in n, it is generally recommended that MLPs be used for problems that require no more than a polynomial number of hidden nodes [185].

Recently, Weymaere *et al.* [134] have proposed a method based on standard pattern classification techniques such as clustering and nearest neighbour classification for properly initializing the parameters (weights) of a two-layer MLP

(which is a crucial task), and for identifying without the need for any BP training, an appropriate network size and topology for solving the classification problem under investigation. Once, an appropriate network is selected, it can then be trained using the standard BP algorithm.

For the single-hidden layered linear in the parameters RBF network, as discussed earlier, an Orthogonal Least Squares (OLS) algorithm has been proposed for choosing an appropriate number of centres for the application in hand [187]. This algorithm can be applied to optimise the size of any single hidden layered FANN whose response is a linear function of the network output layer weights, such as the VNN and the new FFENN structure developed in chapter 4 (but not the conventional FLNN discussed in Appendix A.5.2 as its output is a non-linear function of its weights).

Local Minima

All multi-layered FANNs requiring use of non-linear learning algorithms such as the BP or DR, have the problem of multi-minima error surfaces. These algorithms implement steepest descent in the error surface, and if the surface is very complicated, the algorithm may find a local (false) minimum instead of the optimum global minimum solution. There are a number of strategies which can help minimize the risk of finding a local minimum solution.

In practice, if a large number of hidden nodes are used (i.e. greater than the lower limit required for that problem), then the network has been shown to escape the local minima. Local minima can be considered to occur when two or more classes are categorised as being the same. This amounts to poor internal representation within the hidden nodes, and so if more nodes are added, hyperplanes could separate the classes thus reducing the complexity of the optimisation process, at the expense of increased network complexity. Thus, in general a large network is capable of finding the global minimum at the expense of increased computational burden.

Another method is to add random noise to perturb the gradient descent

algorithm from the line of steepest descent, and often this noise is enough to knock a system out of the local minimum [122]. This approach also has the advantage that it takes very little computation time, and so is not noticeably slower than the direct gradient descent algorithm.

Statistical training methods can also help avoid the local minima traps, but they tend to be very slow. These include use of Simulated Annealing, Boltzmann machine, Iterated Descent and Stochastic Diffusion techniques [102]. A new Stochastic-Iterated Descent technique has also been proposed by Jha and Durrani [101] [102].

The problem of local minima can be avoided by using single-hidden layered, linear in the parameters FANNs such as the RBF and VNN structures, as their error surfaces are quadratic. The new FFENN developed in chapter 4, is also shown to possess an error-surface that is uni-modal and hence free of local minima.

Training Algorithm Termination, Overtraining and Generalization

In stochastic gradient algorithms, a minimum solution is computed by iteratively computing the gradient and varying the weights. The process of computing the gradient and adjusting the weights is repeated until a minimum is reached. In practice, automatic termination of the algorithm may become a problem. However, there are a number of stopping criteria which can be considered [24]:

- Firstly, the algorithm can be terminated when the gradient magnitude J is sufficiently small (implying $\approx$ zero gradient).

- Second, one could stop the algorithm when a fixed number of learning iterations have been performed. However, this does not guarantee that the algorithm will terminate at a minimum.

- Thirdly, the algorithm could be terminated when J falls below a certain threshold. However, this requires a priori knowledge of the required minimal value of J which is not generally available. Even if it is, however, this stopping criterion may not yield a solution generalizing well to new data.

The above three criteria are sensitive to the choice of the parameters, which if not chosen properly can lead to poor results due to either premature termination or *overtraining* of the network. During learning, the network performance continues to improve on the *training data* , but its performance on the *test data* will only improve upto a point, beyond which it starts to degenerate. It is at this point that the network begins to overfit the training data (i.e. the network begins to learn spurious trends in the training data deviating from the previously learnt general statistical trends -termed overtraining) and the algorithm should be terminated. A stopping method based on this criteria, termed *cross-validation* can be used to monitor the generalization performance of the network during learning, and stop the algorithm when there is no longer an improvement [24]. This method therefore also avoids premature termination and actually improves the generalization performance of the network. However, if the cross validation method is to yield statistically accurate results, it is generally necessary to perform several independent splits on the available data set into a training and test set, and then average the results to obtain an overall estimate of the generalization performance [24]. Hence, although the cross-validation method is widely accepted, it can be extremely time consuming if lengthy learning times are required for each of the splits. Furthermore, if the number of data samples is limited, cross validation will reduce the size of the training set even further. Alternative techniques for predicting the generalization performance of neural networks have also been proposed such as the Predicted Squared Error (PSE) and the Generalised Prediction Error (GPE) [125] [126] which predict the generalisation performance as function of the network's performance on the training set, the actual or effective number of free parameters in the network, and the training set size. However, these measures have not been extensively used in practical applications.

Various computationally expensive techniques have also been proposed for improving the generalisation performance through pruning of the redundant network weights, such as the Optimal Brain Damage and Optimal Brain Surgeon methods [107] [89], in which the neural network weights with the smallest saliency

are pruned after the network has been initially trained. After pruning, the network is then re-trained to obtain the final solution. Another method of reducing the number of network weights and thereby improving the generalization ability, is through the use of local connections and weight sharing [98] where the individual network nodes process only a local region of the input space.

Alternative methods for improving the generalization performance are based on the complexity regularization scheme, in which a term representing a measure of the network's complexity (such as the number of weights) is added to the criterion function that discourages the learning algorithm from seeking solutions that are too complex. The resulting criterion function is of the form [24]:

$$Cost = Mapping + Model\ \ Complexity \tag{A.17}$$

The neural network model that minimises the above cost function will then provide a minimal description of the data. Cost functions of the above type have been used in the methods of weight decay [87], weight elimination [115] [117], and soft weight sharing [121]. The latter technique combines the advantages of weight sharing and weight decay into a single unified approach.

Recently, Murray *et al.* [132] have shown that addition of analogue synaptic weight noise during the neural network training not only reduces its training time, but also enhances its generalisation and fault tolerance capability. Jean *et al.* [135] have also proposed a simple weight smoothing algorithm in which a smoothing constraint is incorporated into the objective function of BP to seek solutions with smoother connection weights. The technique has been shown to improve the generalization capability of feedforward neural networks for pattern recognition applications.

Temporal Instability

If the network is learning to recognize characters, it does no good to learn the letter C if in doing so it forgets B -referred to as the stability-plasticity dilemma.

A process is required to teach the network to learn an entire training set without disrupting what it has already learned. Rumelhart's convergence proof [75] does accomplish this but requires that the network be shown all training set vectors (termed an epoch) before adjusting any weights. The required weight changes must be accumulated over the entire set (thereby requiring additional storage). After a **number** of such training iterations, the weights will converge to the minimal error. This method may however, not be useful if the network is subjected to a continuously changing learning environment where the same input vector may may never appear twice. In this case, the learning process may never converge and may instead wander aimlessly or oscillate wildly. In this respect back-propagation fails to mimic biological systems.

Recently, Haykin *et al.* [90] proposed a new Recurrent Neural Network type architecture for successful on-line (adaptive) prediction of non-stationary time series processes. In chapter 5, the application of the new FFENN and its recurrent version termed the Recurrent FENN (RFENN), to on-line modelling of non-stationary non-linear dynamical processes is investigated.

Appendix B

The Real Time Recurrent Learning Algorithm

Recalling from chapter 2 (section 2.3.2), the entire dynamics of a m input n output $(m;n)$RTRN can be represented by the following set of equations:

$$s_i(k+1) = \sum_{j=1}^{n} w_{i,j}(k) y_j(k) + \sum_{j=1}^{m} w_{i,j+n}(k) x_j(k) \tag{B.1}$$

$$y_i(k+1) = f(s_i(k+1)) \quad \text{for} \quad i = 1, \ldots, n \tag{B.2}$$

Letting $d_i(k)$ denote the desired (target) response of neuron i at time k, then the error at the i-th unit is given by:

$$e_i(k+1) = d_i(k+1) - y_i(k+1) \text{ for } \quad i = 1, \ldots, n \tag{B.3}$$

It has been assumed without loss of generality that there exist desired values for all nodes in the RNN. In general, the number of *visible* nodes which provide externally reachable outputs can be less than n, that is the error index $i \leq n$; with the remaining neurons being referred to as *hidden* nodes [56]. The total *instantaneous* error at time $k+1$ is then given by:

$$J(k+1) = \frac{1}{2} \sum_{i=1}^{n} e_i^2(k+1) \tag{B.4}$$

The objective is to change the weights in the direction that minimizes $J(k+1)$. This can be achieved by using an approximation to the method of steepest

descent, in which an instantaneous estimate of the error gradient with respect to the weights is assumed as follows:

$$\delta w_{g,h} = -\mu \frac{\partial J(k+1)}{\partial w_{g,h}} \quad \text{for } g = 1, \ldots, n \;\; h = 1, \ldots, m+n$$

where $\delta w_{g,h}$ is the incremental change in a network weight $w_{g,h}$ along the direction of the steepest descent, and μ is the learning rate parameter.

From equation B.4, we note that

$$\frac{\partial J(k+1)}{\partial w_{g,h}} = \sum_{i=1}^{n} e_i(k+1) \frac{\partial e_i(k+1)}{\partial w_{g,h}}$$

Using the RNN output error equation B.3,

$$\frac{\partial J(k+1)}{\partial w_{g,h}} = -\sum_{i=1}^{n} e_i(k+1) \frac{\partial y_i(k+1)}{\partial w_{g,h}}$$

Using the chain rule for differentiation:

$$\frac{\partial y_i(k+1)}{\partial w_{g,h}} = \frac{\partial y_i(k+1)}{\partial s_i(k+1)} . \frac{\partial s_i(k+1)}{\partial w_{g,h}}$$

which readily gives

$$\frac{\partial y_i(k+1)}{\partial w_{g,h}} = f'(s_i(k+1)).[\sum_{j=1}^{n} w_{i,j} \frac{\partial y_i(k)}{\partial w_{g,h}} + \sum_{j=1}^{m} \frac{w_{i,j+n}}{\partial w_{g,h}} . x_j(k)]$$

where $f'(.)$ denotes the derivative of $f(.)$. Noting that the derivative $\frac{w_{i,j+n}}{\partial w_{g,h}}$ equals one only when $i = g$ and $j + n = h$ and zero otherwise, the above equation can be re-written as:

$$\frac{\partial y_i(k+1)}{\partial w_{g,h}} = f'(s_i(k+1)).[\sum_{j=1}^{n} w_{i,j} \frac{\partial y_i(k)}{\partial w_{g,h}} + \sigma_{gi} . x_h(k)]$$

where σ_{gi} is the kroneker delta equal to 1 when $i = g$ and zero otherwise.

Letting $p^i_{g,h}(k+1) = \frac{\partial y_i(k+1)}{\partial w_{g,h}}$, defined as the sensitivity [118], then a recursive estimator for it can be written as below:

$$p^i_{g,h}(k+1) = f'(s_i(k+1)).[\sum_{j=1}^{n} w_{i,j} p^i_{g,h}(k) + \sigma_{gi} . x_h(k)] \text{ for } g, i = 1, \ldots, n \; h = 1, \ldots, m+n \tag{B.5}$$

with initial conditions $p^i_{g,h}(0) = 0$.

Finally, the weight update equation at time k, which is defined as:

$$w_{g,h}(k+1) = w_{g,h}(k) + \delta w_{g,h}$$

can be written as:

$$w_{g,h}(k+1) = w_{g,h}(k) + \mu \sum_{i=1}^{n} e_i(k+1) p^i_{g,h}(k+1) \tag{B.6}$$

The weights can be initialized by choosing them from a set of uniformly distributed random numbers [56].

Equations B.1, B.2, B.3, B.5 and B.6 are collectively known as the Real Time Recurrent Learning (RTRL) algorithm.

Appendix C

Author's Publications

List of Published Papers :

1. Amir Hussain, John J.Soraghan, Tariq S.Durrani, *Optimal Functional Link Net Based Feed Forward and Decision Feedback Equalizers*, Signal Processing VII:Theories and Applications, Eds. M.Holt, C.Cowan, P.Grant, W.Sandham, pp.1524-1527, September 1994.

2. Amir Hussain, John J.Soraghan, Tariq S.Durrani, *New non-linear Decision Feedback Equalizers based on the Functional Link Neural Network*, Proceedings of IEE Colloqium on Non-linear Filters, London, 18 May 1994.

3. Amir Hussain, John J.Soraghan, Tariq S.Durrani, *Adaptive Functional-Link Neural Network Based Non-Linear Equalizers for Overcoming Co-Channel Interference*, Proceedings of IEEE Workshop of Signal Processing Methods in Multipath Environments, Glasgow, 20-21 April 1995.

4. Amir Hussain, John J.Soraghan, Tariq S.Durrani *Real-Time Adaptive Non-linear Prediction of Non-stationary Signals*, Proceedings of IMA/IEE/IEEE International Conference on Mathematics in Signal Processing, Warwick, 17-19 December 1996.

5. Amir Hussain, John J.Soraghan, Tariq S.Durrani *A new class of computationally efficient Recurrent Neural Networks*, Proceedings of International

Conference on Signal Processing and Applications Technology (ICSPAT), U.S.A, 7-10 Oct'1996.

Bibliography

[1] S. Amari, *Mathematical Foundations of neurocomputing* , Proc. of IEEE, Vol.78, pp.1443-1463, 1990.

[2] J.A.Anderson, J.W.Silverstein, S.A.Ritz, S.R.Jones, *Distinctive Features, Categorical Perception, and Probability Learning: Some applications of a Neural Model* , Psychological Review, Vol.84, No.5, Sept. 1977.

[3] Anderson, IEEE Trans. Neural Networks, 1995

[4] D.H.Ackney, G.E.Hinton, T.J.Sejnowski, *A learning algorithm for Boltzmann Machines* , Cognitive Science, Vol.9, pp.147-169, 1985.

[5] A.Alkulaibi, J.J.Soraghan, T.S.Durrani, *Blind Equalization using Cepstral Methods*, Proc. IEEE/IEE Workshop on Signal Processing Methods in Multipath Environments, pp.95-104, Glasgow, April 1995.

[6] D.Hatzinakos, C.Nikias, *Blind Equalization using a Tricepstrum Based Algorithm*, IEEE Trans. Communications, Vol.39, no.5, pp.669-682, May 1991.

[7] A.D.Back, A.C.Tsoi, *A time series modelling methodology using FIR and IIR synapses* , Proc. Workshop on Neural Networks for Statistical and Economic Data, Dublin, DOSES, Statistical Office of European Communitiesm F.Murtagh Ed., pp.187-194, 1990.

[8] A.R.Barron, R.Barron, *Statistical learning networks: A unifying view* , in E.G.Wegman, D.I.Gantz, and J.J.Miller, Eds., Computing Science and Statistics: Proc. of the 20th Symposium on the Interface, pp.192-202, 1989.

[9] C.A.Belfiore, J.H.Park, *Decision Feedback Equalization* . Proc. IEEE, Vol.67, no.8, pp.1143-1156, 1979.

[10] T.Bohlin, *Maximum power validation of models without higher order fitting* , Automatica, Vol.4, pp.137-146, 1978.

[11] D.S.Broohmead, G.P.King, *Extracting qualitative dynamics from experimental data* , Physica D, Vol.20, pp.217-236, 1986.

[12] D.S.Broomhead, D.Lowe, *Multi-variable functional interpolation and adaptive networks* , Complex Systems, Vol.2, no.3, pp.269-303, 1988.

[13] P.Cardaliaguet, G.Euvrard, *Approximation of a function and its derivative with a neural network* , Neural Networks, Vol.5, pp.207-220, 1992.

[14] G.Carpenter, S.Grossberg, *The ART of adaptive pattern recognition by a self organizing neural network* , IEEE Computer Magazine, pp.77-88, 1988.

[15] M.Casdagli, *Non-linear prediction of chaotic time series* , Physica D, Vol.35, pp.335-356, 1989.

[16] C.P.Callender, C.F.N.Cowan, *A comparison of six different non-linear equalization techniques for digital communications systems* , Signal Processing VII:Theories and Applications, Eds. M.Hot, C.Cowan, P.Grant, W.Sandham, pp.1524-1527, 1994.

[17] R.O.Duda, P.E.Hart, *Pattern Classification and Scene Analysis* , Wiley, NewYork, NY, 1973.

[18] S.Elanayar, Y.C.Shin, *Radial Basis Function Neural Network for Approximation and Estimation of Non-linear Stochastic Dynamic Systems* , IEEE Trans. Neural Networks, Vol.5, no.4, pp.594-603, 1994.

[19] O.E.B.Nielsen, J.L.Jensen, W.S.Kendall, Eds., *Networks and Chaos-Statistical and Probabilistic Aspects* , Chapman and Hall, 1994.

[20] M.Niranjan, V.Kadirkamanathan, *A non-linear model for time series prediction and signal interpolation* , ICASSP Proc., pp.1713-1716, Toronto, 1991.

[21] R.P.Lippmann, *An introduction to computing with neural nets* , IEEE Acoustics, Speech and Signal Processing Magazine, Vol.4, pp.4-22, April 1987.

[22] D.Lowe, A.R.Webb, *Time series prediction by adaptive networks: a dynamical systems perspective* , IEE Proc.-F, Vol.138, no.1, pp.17-25, 1991.

[23] D.R.Hush, J.M.Salas, *Improving the learning rate of Back Propagation with the gradient reuse algorithm*Proc. of IEEE Intern Conf. Neural Networks, Vol.1, pp.441-448, 1988.

[24] D.R.Hush, B.G.Horne, *Progress in Supervised Neural Networks: What's new since Lippmann?* , IEEE Signal Processing Magazine, pp.9-39, January 1993.

[25] R.P.Lippmann, *A critical overview of neural network classifiers* , Proc. of IEEE Workshop on Neural Networks for Signal Processing, pp.266-278, 1991.

[26] K.Hornik, M.Stinchcombe, H.White, *Multi-layer feedforward networks are universal approximators* , Neural Networks, Vol.2, pp.359-366, 1989.

[27] R.J.Hunt, D.Sbarbaro, R.Zbikowski, P.J.Gawthrop, *Neural networks for control systems - A survey* , Automatica, Vol.28, no.6, pp.1083-1112, 1992.

[28] B.W.Wah, *Special Issue on Artificial Neural Networks: Guest Editor's Introduction* , IEEE Trans. on Computers, Vol.40, No.12, pp.1317-1319, December 1991.

[29] S.W.Wales, *Technique for cochannel interference suppression in TDMA mobile radio systems*, IEE Proc. Communications, Vol.142, no.2, April 1995.

[30] U.A.Muller, *Achieving supercomputing performance for neural net simulation with an array of Digital Signal Processors* , IEEE Micro, pp.55-64, October 1992.

[31] Y.H.Pao, *Adaptive Pattern Recognition and Neural Networks* , Addison Wesley, 1989.

[32] Y.H.Pao, S.M.Phillips, D.J.Sobajic, *Neural net computing and the intelligent control of systems* , Int. J. Control, Vol.56, no.2, pp.263-289, 1992.

[33] A.Lapedes, R.Farber, *Non-Linear Signal Processing using Neural Networks: Prediction and System Modelling* , Los Alamos National Laboratory report LA-UR-87-2662, 1987.

[34] K.S.Narendra, K.Parthasarathy, *Identification and control of dynamical systems using Neural Networks* ,IEEE Transactions on Neural Networks, Vol.1, pp4-27, 1990.

[35] K.S.Narendra, K.Parthasarathy, *Gradient methods for the optimization of dynamical systems containing neural networks*,IEEE Transactions on Neural Networks, Vol.2, pp252-262, March 1991.

[36] A.U.Levin, K.S.Narendra, *Recursive identification using feedforward neural networks*, Int. J. of Control, Vol.61, no.3, pp.533-547, 1995

[37] P.S.Sastry, G.Santharam, K.P.Unnikrishnan, *Memory Neural Networks for Identification and Control of Dynamical Systems* , IEEE Trans. on Neural Networks, Vol.5, no.2, pp.320-330, March 1994.

[38] B.Srinivasan, U.R.Prasad, N.J.Rao, *Back Propagation through Adjoints for the Identification of Nonlinear Dynamic Systems Using Recurrent Neural models* , IEEE Trans. on Neural Networks, Vol.5, no.2, pp.213-228, March 1994.

[39] M.C.Mackey, L.Glass, *Oscillation and chaos in physiological control systems* , Science, Vol.197, 1977.

[40] V.J.Mathews, *Adaptive polynomial filters* , IEEE Signal Processing Magazine, pp.10-26, July, 1991.

[41] MATLAB, *Matlab User's Guide*, Version 4.2, 1994.

[42] J.R.McDonnel, and D.Wagen, *Evolving Recurrent Perceptrons for Time-Series Modelling* , IEEE Transactions on Neural Networks,Vol.5, no.1, pp.24-38, 1994.

[43] W.C.Mead, R.D.Jones, Y.C.Lee, C.W.Barnes, G.W.Glake, L.A.Lee, M.K.O'Rourke, *Prediction of chaotic time series using CNLS-NET - example: The Mackey-Glass equation* , Technical Report LA-UR-91-720, Los Alamos National Laboratory, Los Alamos, NM, 1991.

[44] M.Minsky, S.Papert, *Perceptrons: An introduction to computational geometry*, MIT Press, 1969.

[45] N.Morgan, H.Bourlard, *Continuous Speech Recognition*, IEEE Signal Processing Magazine, Vol.12, no.3, May 1995.

[46] M.Gori, Y.Bengio, R.D.Mori, *BPS: A learning algorithm for capturing the dynamic nature of speech* , IJCNN,Vol.2, pp.417-423, 1989.

[47] S.F.Fahlman, C.Lebiere, *The cascade-correlation learning architecture* , Advances in Neural Information Processing Systems, Vol.2, pp.524-532, 1990.

[48] P.V.Foukal, *The variable sun* , Scientific American, Vol.262, p.34, February, 1990.

[49] P.Frasconi, M.Gori, G.Soda, *Local feedback Multi-layered networks* , Neural Computation, Vol.4, pp.120-130,1992.

[50] J.H.Friedman, W.Stuuetzle, *Projection pursuit regression* , J. Amer. Stat. Assoc., Vol.76, pp.817-823, 1981.

[51] C.L.Giles, C.W.Omlin, *Pruning Recurrent Neural Networks for improved generalization performance* , IEEE Trans. Neural Networks, Vol.5, no.5, pp.848-851, 1994.

[52] J.T.Connor, R.D.Martin, L.E.Atlas, *Recurrent Neural Networks and Robust Time Series Prediction* , IEEE Trans. on Neural Networks, Vol.5,pp.240-254,March1994.

[53] T.Kailath, *Linear Systems* , Prentice-Hall, Englewood Cliffs, N.J.,1980.

[54] T.Kohonen, *An adaptive discrete signal detector based on self-organizing maps*, Proc. IJCNN, pp.249-252, Washington, U.S.A, January 1990.

[55] G.Kechriotis, E.Zervas, E.S.Manolakos, *Using Recurrent Neural Networks for Adaptive Communication Channel Equalization* , IEEE Trans. on Neural Networks, Vol..5, no.2, pp.267-278, 1994.

[56] S.Haykin, *Neural Networks - A Comprehensive Foundation* , MacMillan, 1994

[57] S.Haykin, *Neural Networks expand SP's horizons*, IEEE Signal Processing Magazine, Vol.13, no.2, pp.24-49, March 1996.

[58] J.J.Shynk, *Adaptive IIR filtering* , IEEE ASSP Magazine, April1989.

[59] C.Svaver, L.K.Hansen, J.Larsen, *On design and evaluation of tapped-delay neural network architectures* , IEEE Int. Conf. Neural Networks, San Francisco, 1992.

[60] Schetzen, *The Volterra and Weiner Theories of Non-linear Systems* , New York, NY, John Wiley, 1980.

[61] W.S.McCulloch, W.Pitts, *A logical calculus of the ideas immanent in nervous activity* , Bulletin of Mathematical Biophysics, Vol.5, pp.115-133, 1943.

[62] N.S.Jayant, P.Noll, *Digital Coding of waveforms*, Prentice-Hall, 1984.

[63] P.J.W.Rayner, M.R.Lynch, *A new connectionist model based on a non-linear adaptive filter* , IEEE ICASSP, pp.1191-1194, Glasgow, 1989.

[64] M.R.Lynch, S.B.Holden, P.J.Rayner, *Complexity reduction in Volterra connectionist networks using a self structuring LMS algorithm* , 2nd IEE Intern. Conf. Artificial Neural Networks, pp.44-48, Bournemouth, U.K. 1991.

[65] F.Rosenblatt, *The perceptron: A probabilistic model for information storage and organization in the brain* , Psychological Review, Vol.65, pp.386-408, 1958.

[66] K.C.Sharman, *High Resolution Directional Spectral Analysis for sensor arrays* , PhD Dissertation, Dept. of EEE, University of Strathclyde, U.K, 1988.

[67] J.L.Elman, *Finding structure in time*, Cognitive Science, Vol.14, pp.179-211, 1990.

[68] Feldman, Reviewer of *Handbook of Intelligent Control: Neural, Fuzzy and Adaptive Approaches* , IEEE Trans. Neural Networks, Vol.5, no.5, pp.852-853, September 1994.

[69] P.J.Werbos, *Backpropagation through time: What it does and how to do it* , Proc. IEEE, Vol.78, pp.1550-1560, 1990.

[70] B.Widrow, *Adaptive Sampled-data systems, a statistical theory of adaption* , IRE WESCON Convention Record, part 4, New York: Institute of Radio Engineers, 1959.

[71] B.Widrow, R.Winter, *Neural Nets for Adaptive Filtering and Adaptive Pattern Recognition*, Computer (IEEE), pp.25-39, March 1988.

[72] T.Kohonen, *Self Organization and Associative Memory* , Second Edition, Springer-Verlag, New York, 1988.

[73] J.T.Tou, R.C.Gonzalez, *Pattern Recognition Principles* , Addison-Wesley, Reading, MA, 1974.

[74] P.J Werbos, *Beyond Regression: New tools for Prediction and analysis in the behavioural Sciences* , Doctoral Dissertation, Applied Mathematics , Harvard University, Boston, MA, November 1974.

[75] D.E.Rumelhart and J.L.McClelland. *Parallel Disributed Processing: Explorations in the Microstructure of Cognition: Foundations* , Vol.1, MIT press, Cambridge, MA, 1986.

[76] J.Hopfield, *Neural Networks and physical systems with emergent collective computational abilities* , Proc. of National Acad. of Sciences of USA, Vol.79, pp.2544-2558, 1982.

[77] A.G.Ivakhnenko, *Heuristic self-organization in problems of engineering cybernetics* , Avtomatika, Vol.6, pp.207-219, 1970.

[78] S.M.Kay, *Fundamentals of Statistical Signal Processing*, Prentice-Hall, 1993.

[79] L.Ljung, *System Identification: Theory for the user*, Prentice-Hall, Englewood Cliffs, N.J., 1987.

[80] M.D.Richard, R.P.Lippmann, *Neural Network classifiers estimate Bayesian aposteriori probabilities* , Neural Computation, Vol.3, no.4, pp.461-483, 1991.

[81] Ralph, J.J.Soraghan, *Fuzzy adaptive filters for channel equalization* , Project Report, Signal Processing Division, University of Strathclyde, Glasgow, U.K, 1994.

[82] A.C.Tsoi, A.Back, *Locally Recurrent Globally Feedforward Networks: A critical Review of Architectures* , IEEE Trans. on Neural Networks, Vol.5, no.2, pp229-239, March 1994.

[83] A.E.Makhoul, and R.Schwartz, *Formation of disconnected decision regions with a single hidden layer* , IJCNN, Vol.1, pp.455-460, 1989.

[84] D.P Morgan and C.L. Scofield, *Neural Networks and Speech Processing* , Kluwer Academic Publishers, 1991.

[85] E.Mumolo, A.Carini, D.Francescato, *ADPCM with non-linear Predictors*, Signal Processing VII: Theories and Applications, M.Holt, C.Cowan, P.Grant, W.Sandham, (Eds.), pp.387-390, 1994.

[86] O.Nerrand, P.Roussel-Ragot, D.Urbani, L.Personnaz, G.Dreyfus, *Training Recurrent Neural Networks: Why and how; An illustration in Dynamical Process Modeling* , IEEE Trans. Neural Networks, Vol.5, no.2, pp.178-184, 1994.

[87] S.J.Hanson, L.Y.Pratt, *Comparing biases for minimal network construction with back propagation* , Advances in Neural Information Processing Systems, Vol.1, pp.177-185, 1989.

[88] E.J.Hartman, J.D.Keeler, J.M.Kowalski, *Layered neural networks with Gaussian hidden units as universal approximators* , Neural Computation, Vol.2 no.2, pp.210-215, 1990.

[89] B.Hassibi, D.G.Stork, G.J.Wolff, *Optimal Brain Surgeon and general network pruning* , Proc. IEEE Intern. Conf. Neural Networks, Vol.1, pp.293-299, San Francisco, CA, 1993.

[90] S.Haykin, L.Li, *Non-linear Adaptive Prediction of Nonstationary Signals* , IEEE Trans. Signal Processing, Vol4, no.2, pp.526-535, 1995.

[91] M.L.Honig,*Echo cancellation of voiceband data signals using Recursive Least Squares and stochastic gradient algorithms* , IEEE Trans. Communications, Vol.33, no.1, pp.65-73, 1984.

[92] S.C.Huang and Y.F.Huang, *Bounds on the number of hidden neurons in multilayer perceptrons* , IEEE Transactions on Neural Networks, Vol.2, No.1,:pp.47-55, 1991.

[93] H. Schutze, Z.Ren, *Numerical characteristics of FRLS transversal adaptive algorithms - a comparative study* , Signal Processing, pp.317-332, 1992.

[94] G.Z.Sun, H.H.Chen, Y.C.Lee, and C.L.Giles, *Turing equivalence of neural networks with second order connection weights* , IJCNN, Vol.2, pp.357-362, 1991.

[95] H.Tong, K.S.Lim, *Threshold autoregression, limit cycles and cyclical data* , Journal Royal Statistical Society B, Vol.42, pp.245, 1980.

[96] R.A.Jacobs,*Increases rates of convergence through learning rate adaptation* , Neural Networks, Vol.1, no.4, pp.295-308, 1988.

[97] T.P.Vogal, J.K.Mangis, A.K.Rigler, W.T.Zink, and D.L.Alton, *Accelerating the convergence of the back-propagation method"*, Bio. Cybernetics, Vol.59, pp.257-263, 1988.

[98] A.Waibel, G.Hanazawa, G.Hinton, K.Shikano, K.J.Lang, *Phoneme recognition using time-delay neural networks* , IEEE Trans. Acoustics, Speech and Signal Processing (ASSP), Vol.37, pp.328-339, 1989.

[99] R.Battiti, *First and second order methods for learning: Between steepest descent and Newton's method* , Neural Computation, Vol.4, pp.141-166, 1992.

[100] N.B.Karayiannis, and A.N.Venetsanopoulos, *Fast Learning Algorithms for Neural Networks"*, IEEE Trans. on Cirsuits and Systems-II: Analog. and Dig. Signal Proc., vol.39, No.7, July 1992.

[101] S.Jha, T.S.Durrani, *Bearing estimation using Neural Optimisation Methods*, Proc. ICASSP, 1990.

[102] S.Jha, *Artificial Neural Networks for DSP Applications* , PhD Dissertation, Strathclyde University, Dept. of EEE, pp.97-132, 1990.

[103] D.Ruelle, *Chaotic Evolution and Strange Attractors*, Cambridge University Press, 1989.

[104] E.Ott, *Chaos in Dynamical Systems,* Cambridge University Press 1993.

[105] G.Cybenko, *Approximation by superpositions of a sigmoidal function* , Mathematics of Control, Signals and Systems ,Vol.2, no.4, pp.303-314, 1989.

[106] D.L.Chester, *Why two hidden layers are better than one* , IJCNN, vol.1 pp.265-268, Erlbaum, 1990.

[107] Y.Le Cun, J.S.Denker, S.A.Solla, *Optimal Brain Damage* , Advances in Neural Information Processing Systems, Vol.2, pp.598-605, 1990.

[108] S.Lee, R.M.Kil, *A gaussian potential function network with hierarchically self-organizing learning* , Neural Networks, Vol.4, pp.207-224, 1991.

[109] S.A.Billings, W.S.F.Voon, *Correlation based model validity tests for non-linear models* , Int. J. Control, Vol.44, pp.193-199. 1986.

[110] S.A.Billings, H.B.Jamaluddin, S.Chen, *Properties of Neural Networks with applications to modelling non-linear dynamical systems* , Int. J. Control, Vol.55, pp.193-224, 1992.

[111] I.J.Leontaritis, S.A.Billings, *Model Selection and Validation methods for non-linear systems* , Int. J. Control. Vol.41, pp.311-341, 1987

[112] B. de Vries, J. Principe, *The Gamma Model - A new neural model for temporal processing* , Neural Networks, Vol.5, no.4, pp.565-576, 1992.

[113] E.A.Wan, *Time Series Predictions by using a connectionist network with internal delay lines* in Time Series Prediction: Forecasting the Future and Understanding the Past, Eds. A.S. Weigend and N.A.Gershenfeld, Santa Fe Institue Studies in the Sciences of Complexity. pp.195-217, Reading, MA: Addison-Wesley, 1994.

[114] L.X.Wang, J.M.Mendel, *Fuzzy adaptive filters, with application to non-linear channel equalization* , IEEE Trans. Fuzzy Systems, Vol.1, no.3, August 1993.

[115] A.S.Weigend, D.E.Rumelhart, and B.A.Huberman, *Predicting the future: A connectionist approach* , Technical Report Stanford-PDP-90-01, 1990; also in the International Journal of Neural Systems, I, pp.193-209, 1990.

[116] D.Wettschereck, T.Dietterich, *Improving the performance of radial basis function networks by learning center locations* , Advances in Neural Information ProcessingSystems, Vol.4, pp.1133-1140, 1992.

[117] A.S.Weigend, B.A.Huberman, and D.E.Rumelhart, *Predicting sunspots and exchange rates with connectionist networks* , in M.Casdagli and S.Eubank, Eds., Non-linear Modeling and Forecasting, Santa Fe Institute Studies in the Sciences of Complexity, Vol.12, Addison Wesley, 1991.

[118] R.J.Williams, D.Zipser, *A learning algorithm for continually running fully recurrent neural network* . Neural Computation, Vol.1, no.2, pp.270-280, 1989.

[119] R.J.Williams, D.Zipser, *Gradient based learning algorithms for recurrent connectionist networks* , Technical Report NU–CCS-90-9, College of Computer Science, Northeastern University, Vol.2, 1990.

[120] *NeuralWorks Professional II/Plus and NeuralWorks Explorer: Neural Computing Volume "*, NeuralWare, Inc., Pittsburgh, PA 15276, 1990.

[121] S.J.Nowlan, G.E.Hinton, *Simplifying neural networks by soft weight sharing* , Neural Computation, Vol.4, no.4, pp.473-493, 1992.

[122] T.Masters, *Artificial Neural Networks and C++ Recipes*, 1993.

[123] F.R.Magee, J.G.Proakis, *Adaptive maximum likelihood sequence estimation for digital signalling in the presence of intersymbol interference* , IEEE Trans. Information Theory, Vol.19, no.1, pp.120-124, 1973.

[124] J.E.Moody, C.J.Darken, *Fast learning in networks of locally tuned processing units* , Neural Computation, Vol.1, pp.281-293, 1989.

[125] J.E.Moody, *Note on generalization, regularization, and architecture selection in non-linear learning systems* , Proc. of IEEE Workshop on Neural Networks for Signal Processing , pp.1-10, 1991.

[126] J.E.Moody, *The effective number of parameters: An analysis of generalization and regularization in non-linear learning systems* , Advances in Neural Information Processing Systems, Vol.4, pp.847-854, 1992.

[127] B.Mulgrew, *Kalman Filter techniques in adaptive filtering*, IEE Proc. Part-F, Vol.134, no.3, pp.239-243, June 1987.

[128] B.Mulgrew, C.F.N.Cowan, *Adaptive Filters and Equalizers* , Kluwer Academic Publishers, 1988.

[129] B.Mulgrew, *Applying Radial Basis Functions*, IEEE Signal Processing Magazine, Vol.13, no.2, pp.50-65, March 1996.

[130] M.T.Musavi, W.Ahmed, K.H.Khan, K.B.Faris, D.M.Hummels, *On the training of radial basis function classifiers* , Neural Networks, Vol.5, pp.595-603, 1992.

[131] O.Munoz, J.Fernandez, *Adaptive Arrays for frequency non-selective and selective channels*, Signal Processing VII: Theories and Applications, M.Holt, C.Cowan, P.Grant, W.Sandham (Eds.), pp.1536-1539, 1994.

[132] A.F.Murray, P.Edwards, *Enhanced MLP Performance and Fault Tolerance Resulting from synaptic weight noise during training* , IEEE Trans. Neural Networks, Vol.5, no.5, September 1994.

[133] E.B.Baum, D.Haussler, *What size net gives valid generalization*? , Neural Computation, Vol.1, pp.151-160, 1989.

[134] N.Weymaere, J.P.Martens, *On the initialization and optimization of Multilayer Perceptrons* , IEEE Trans. Neural Networks, Vol.5, no.5, pp.738-751, September 1994.

[135] J.S.N.Jean, J.Wang, *Weight smoothing to improve network generalization* , IEEE Trans. Neural Networks, Vol.5, no.5, pp.752-763, September 1994.

[136] G.H.Park, *System identification and noise cancellation with neural networks* , M.S.thesis, Electrical Engineering, Case Western Reserve University, Cleveland, Ohio, 1990.

[137] M.B.Priestley, *Non-linear and non-stationary time series analysis* , Academic Press, 1988.

[138] *Digital Signal Processing in Telecommunications* , BT Telecomms. Series 3, Edited by Westall and Ip, 1994.

[139] D.F.Specht, *Generation of polynomial discriminant functions for pattern recognition* , IEEE Trans. Electron. Comput., Vol. EC-16, no.3, pp.308-319, 1967.

[140] K.Abend, B.D.Fritchman, *Statistical detection for communication channels with intersymbol interference* , Proc. IEEE, Vol.58, no.5, pp.779-785, 1970.

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

[142] R.W.Lucky, *Survey of communication theory literature: 1968-73* , IEEE Trans. Inform. Theory, pp. 725-739, November 1973.

[143] G.D.Forney, *Maximum Likelihood sequence esstimation of digital sequences in the presence of intersymbol interference* , IEEE Trans. Inform. Theory, Vol 18, pp.378-383, 1972.

[144] R.D.Gitlin, S.B.Weinstein, *Fractionally Spaced Equalization: An improved Digital Transversal Equalizer* . BSTJ, pp.275-296, 1981.

[145] C.F.N.Cowan, P.M.Grant, *Adaptive Filters* , Prentice Hall, 1985.

[146] P.M.Grant, M.J.Rutter, *Application of gradient adaptive lattice filters to channel equalization* , IEE Proc., Part-F, pp.473-479, 1984.

[147] M.J.Rutter, P.M.Grant, *Timed gradient adaptive lattice equalizer* , IEE Proc, Part-F, pp.181-186, 1985.

[148] S.U.H.Qureshi, *Adaptive Equalization* , IEEE Communications Society Magazine, Vol.21, no.2, pp.9-16, 1982.

[149] S.U.H.Qureshi, *Adaptive Equalization* , Proc. IEEE, Vol.73, pp.1349-1387, 1985.

[150] S.W.Piche, *Steepest Descent Algorithms for Neural Network Controllers and Filters* , IEEE Trans. Neural Networks, Vol.5, no.2, 1994.

[151] M.J.D.Powell, *Radial Basis Function approximations to polynomials* , Numerical Analysis Proceedings, pp.223-241, Dundee, U.K., 1988.

[152] J.G.Proakis, *Digital Communications* , New York, McGraw Hill, 1983.

[153] J.G.Proakis, *Digital Communications* , Englewood Cliffs, NJ: Prentice Hall Inc., 1988.

[154] J.G.Proakis, *Digital Communications* , 2nd Edition, McGraw Hill, 1989.

[155] J.G.Proakis, *Adaptive equalization for TDMA Digital Mobile radio* , IEEE Trans. on VT, pp.333-341, 1991.

[156] M.E.Salgado, G.C.Goodwin, R.H.Middleton, *Modified Least squares algorithm incorporating exponentional resetting and forgetting* , Intern. Journal Control, Vol.47, pp.477-491, 1988.

[157] N.R.Sripada, D.G.Fisher, *Improved least squares identification*, Intern. Journal Control, Vol.46, pp.1889-1913, 1987.

[158] D.Janecki, *New recursive parameter estimation algorithms with varying but bounded gain matrix*, Intern. Journal Control, Vol.47, pp.75-84, 1988.

[159] M.Montazeri, P.Duhamel, *Set of algortihms linking NLMS and RLS algorithms*, IEEE Trans. Signal Processing, Vol.43, No.2, February 1995.

[160] S.Theoridis, C.F.N.Cowan, C.P.Callender, C.M.S.See, *Schemes for equalization of communications channels with non-linear impairments*, IEE Proc. Communications, Vol.142, no.3, pp.165-171, June 1995.

[161] E.Biglieri, A.Gersho, R.D.Gitlin, T.L.Lim, *Adaptive cancellation of nonlinear ISI for voiceband data transmission* , IEEE Journal on Selected Areas Communications, Vol. SAC-2, No.5, pp.765-777, 1984.

[162] Z.Ding, R.A.Kennedy, *On the whereabouts of local minima for blind adaptive equalizers* , IEEE Trans. Circuits and Systems, Vol.39, pp.119-123, 1992.

[163] W.S.Gan, J.J.Soraghan, T.S.Durrani, *Application of the Functional-Link Technique for channel equalization* , Electronic Letters Vol.28, No.17, pp1643-164, August 1992.

[164] W.S.Gan, *Algorithms and Parallel Architectures for linear and Neural Network based Adaptive filtering* , PhD Thesis, Dept. of EEE, University of Strathclyde, Glasgow, 1993.

[165] N.Beamish, A.D.Fagan, *On the equalization of baseband LNL channel* , EUSIPCO Proc., pp.1528-1531, Edinburgh, U.K., September 1994.

[166] K.Hacioglu, M.Abdelhafez, *Reconstruction of PAM signals using a multilayered perceptron with a multi-level sigmoidal function* , EUSIPCO Proc., pp.1811-1514, Edinburgh, U.K., September 1994.

[167] J.C.Campbell, A.J.Gibbs, B.M.Smith, *The cyclostationary nature of crosstalk interference from digital signals in multipair cable - Part I:fundamentals; Part II:applications and further results* , IEEE Trans. Comm., Vol.COM-31, no.5, pp.629-649, 1983.

[168] B.R.Peterson, D.D.Falconer, *Exploiting cyclostationary subscriber-loop interference by equalization* , Proc. GLOBECOM 90, San Diego, California, USA, December 1990, pp-1156-1160.

[169] G.J.M.Janssen, *BER and outage performance of a dual signal receiver for narrowband BPSK modulated cochannels in a Rican fading channel* , Personal and Indoor mobile radio Conference, The Hague, pp.601-606, 1994.

[170] K.Giridhar, S.Chari, J.J.Shynk, R.P.Gooch *Joint demodulation of cochannel signals using MLSE and MAPSD Algorithms* , IEEE ICASSP, Vol.4, pp.160-163, 1993.

[171] S.Singhal, L.Wu, *Training feed-forward networks with the extended Kalman Filter* , IEEE ICASSP, pp.1187-1190, Glasgow, 1990.

[172] G.J.Gibson, S.Siu, C.F.N.Cowan, *Multilayer Perceptron Structures Applied to Adaptive Equalizers for Data communications* , IEEE ICASSP, pp1183-1184, Glasgow 1989.

[173] G.J.Gibson, C.F.N.Cowan, *The Application of Non-Linear Structures to Reconstruction of Binary signals* , IEEE Trans. on Signal Processing, pp.1877-1884, August 1991.

[174] A.Hussain, J.J.Soraghan, T.S.Durrani, *Artificial Neural Networks for Array Processing* , Proc. IEEE-IEE Intern. Workshop on Natural Algorithms in Signal Processing, Vol.1, pp.151-161, Chemsford, Essex, 14-16 November 1993.

[175] A.Hussain, J.J.Soraghan *Real-Time Adaptive Non-linear Prediction of Non-stationary Signals*, Proc. IMA-IEE-IEEE International Conference on Mathematics in Signal Processing, Warwick, 17-19 December 1996.

[176] A.Hussain, J.J.Soraghan, T.S.Durrani, *A new class of computationally efficient Recurrent Neural Networks*, Proc. of International Conference on Signal Processing and Applications Technology (ICSPAT), U.S.A, 7-10 Oct'1996.

[177] A.Hussain, J.J.Soraghan, T.S.Durrani, *New non-linear Decision Feedback Equalizers based on the Functional Link Neural Network*, Proc. of IEE Colloqium on Non-linear Filters, London, 18 May 1994.

[178] A.Hussain, J.J.Soraghan, T.S.Durrani, *Optimal Functional Link Net Based Feed Forward and Decision Feedback Equalizers*, Signal Processing VII:Theories and Applications, Eds. M.Holt, C.Cowan, P.Grant, W.Sandham, pp.1524-1527, September 1994.

[179] A.Hussain, J.J.Soraghan, T.S.Durrani, *Adaptive Functional-Link Neural Network Based Non-Linear Equalizers for Overcoming Co-Channel Interference*, Proc. of IEEE Workshop of Signal Processing Methods in Multipath Environments, Glasgow, 20-21 April 1995.

[180] S.Chen, S.A.Billings, *Representation of non-linear systems: the NARMAX model* , Intern. J. Control, Vol.49, pp.1013-1032, 1989.

[181] S.Chen, S.A.Billings, C.F.N.Cowan, P.M.Grant, *Parallel recursive prediction error algorithm for training layered neural networks* , International Journal of Control, Vol.51, No.6, pp.1215-1228, 1990.

[182] S.Chen, S.A.Billings, P.M.Grant, *Non-linear system identification using neural networks* , International Journal of Control, Vol.51, No.6, pp.1191-1214, 1990.

[183] S.Chen, G.J.Gibson, C.F.Cowan, *Adaptive Channel Equalization using a polynomial-perceptron structure* , IEE Proc., vol.137, pt.I, no.5, pp257-264, 1990

[184] S.Chen, G.J.Gibson, C.F.Cowan, P.M.Grant, *Adaptive equalization of finite non-linear channels using Multi-Layered Perceptrons* , Signal Processing, pp.107-119, 1990.

[185] S.Siu, G.J.Gibson, C.F.N.Cowan, *Decision Feedback equalisation using neural network structures and performance comparison with standard architecture* , IEE Proc., 137, Pt 1, No.4, pp221-225, 1990.

[186] S.F.A.Ip, E.Grontier, M.Pope, *Application of MLP to the detection of distorted digital signals* , BT Technol J, Vol.10 No.3, July 1992.

[187] S.Chen, C.F.N.Cowan, P.M.Grant, *Orthogonal Least Squares learning algorithm for Radial Basis Function networks* , IEEE Trans. Neural Networks, Vol.2, no.2, pp.302-309, 1991.

[188] S.Chen, G.J.Gibson, C.F.N.Cowan, P.M.Grant, *Reconstruction of binary signals using an adaptive radial-basis-function equalizer* , Signal Processing, Vol.22, pp.77-93, 1991.

[189] S.Chen, P.M.Grant, C.F.N.Cowan, *Orthogonal Least Squares Algorithm for Training Multi-Output Radial Basis Function Networks*, Proc. IEE 2nd Intern. Confer. on Artificial Neural Networks, pp.336-339, Bournemouth Intern. Center, U.K., November, 1991.

[190] S.Chen, B.Mulgrew, *Overcoming co-channel interference using an adaptive radial basis function equalizer* , Signal Processing 28, Elsevier, pp 91-107, 1992.

[191] S.Chen, S.A.Billings,*Neural Networks for nonlinear dynamic system modelling and identification* , Int.J.Control, Vol56,No.2, pp.319-346, 1992.

[192] S.Chen, S.A.Billings, P.M.Grant, *Recursive hybrid algorithm for non-linear system identification using radial basis function networks* , Int. J. Control, Vol.55, no.5, pp.1051-1070, 1992.

[193] S.Chen, B.Mulgrew, S.McLaughlin, *Adaptive Bayesian Decision Feedback Equalizer based on a RBF network* , IEEE International Conference on Communications, Vol.3, pp.1267-1271, San Francisco, CA, 1992.

[194] S.Chen, B.Mulgrew, P.M.Grant,*Clustering technique for Digital Communications Channel Equalization Using RBF Networks* , IEEE Trans. on Neural Networks, Vol.4, July 1993.

[195] S.Chen, B.Mulgrew, S.McLaughlin, *Adaptive Bayesian Equalizer with Decision Feedback* , IEEE Trans. Signal Processing, Vol.41, no.9, September, 1993.

[196] S.Chen, S.McLaughlin, B.Mulgrew, P.M.Grant, *Adaptive Bayesian decision feedback equalizer incorporating co-channel interference suppression*, Proc. IEEE Int. Conf. Comms., Vol.1, pp.530-533, May 1994.

[197] S.Chen, *Radial Basis Functions for Signal Prediction and System Modeling*, Journal of Applied Science and Engineering, Vol.1, No.1, June 1994.

[198] S.Chen, S.McLaughlin, B.Mulgrew, P.M.Grant, *Adaptive Bayesian decision feedback equalizer for dispersive mobile radio channels*, IEEE Trans. Communications, Vol.43, no.5, pp.1937-1946, May 1995.

[199] G.U.Yule, *On a method of investigating periodicities in disturbed series with special reference to Wolfer's sunspot numbers* , Philos. Trans. Roy. Soc. Lon. Ser. A, Vol.226, pp.267, 1927.