Dynamic non-linear system modelling using wavelet-based soft computing techniques

Mahdi Amina

School of Electronics and Computer Science

Dynamic Non-Linear System Modelling Using Wavelet-Based Soft Computing Techniques

MAHDI AMINA

PhD

September 2011

Thesis submitted in partial fulfilment of the requirements for
The degree of Doctor of Philosophy

Abstract

The enormous number of complex systems results in the necessity of high-level and cost-efficient modelling structures for the operators and system designers. Model-based approaches offer a very challenging way to integrate a priori knowledge into the procedure. Soft computing based models in particular, can successfully be applied in cases of highly nonlinear problems. A further reason for dealing with so called soft computational model based techniques is that in real-world cases, many times only partial, uncertain and/or inaccurate data is available.

Wavelet-Based soft computing techniques are considered, as one of the latest trends in system identification/modelling. This thesis provides a comprehensive synopsis of the main wavelet-based approaches to model the non-linear dynamical systems in real world problems in conjunction with possible twists and novelties aiming for more accurate and less complex modelling structure.

Initially, an on-line structure and parameter design has been considered in an adaptive Neuro-Fuzzy (NF) scheme. The problem of redundant membership functions and consequently fuzzy rules is circumvented by applying an adaptive structure. The growth of a special type of *Fungus (Monascus ruber van Tieghem) is* examined against several other approaches for further justification of the proposed methodology.

By extending the line of research, two *Morlet* Wavelet Neural Network (WNN) structures have been introduced. Increasing the accuracy and decreasing the computational cost are both the primary targets of proposed novelties. Modifying the synoptic weights by replacing them with Linear Combination Weights (LCW) and also imposing a Hybrid Learning Algorithm (HLA) comprising of Gradient Descent (GD) and Recursive Least Square (RLS), are the tools utilised for the above challenges. These two models differ from the point of view of structure while they share the same HLA scheme. The second approach contains an additional Multiplication layer, plus its hidden layer contains several sub-WNNs for each input dimension. The practical superiority of these extensions is demonstrated by simulation and experimental results on real non-linear dynamic system; *Listeria Monocytogenes* survival curves in Ultra-High Temperature **(**UHT) whole milk, and consolidated with comprehensive comparison with other suggested schemes.

At the next stage, the extended clustering-based fuzzy version of the proposed WNN schemes, is presented as the ultimate structure in this thesis. The proposed Fuzzy Wavelet Neural network (FWNN) benefitted from Gaussian Mixture Models (GMMs) clustering feature, updated by a modified Expectation-Maximization (EM) algorithm. One of the main aims of this thesis is to

illustrate how the GMM-EM scheme could be used not only for detecting useful knowledge from the data by building accurate regression, but also for the identification of complex systems.

The structure of FWNN is based on the basis of fuzzy rules including wavelet functions in the consequent parts of rules. In order to improve the function approximation accuracy and general capability of the FWNN system, an efficient hybrid learning approach is used to adjust the parameters of dilation, translation, weights, and membership. Extended Kalman Filter (EKF) is employed for wavelet parameters adjustment together with Weighted Least Square (WLS) which is dedicated for the Linear Combination Weights fine-tuning. The results of a real-world application of Short Time Load Forecasting (STLF) further re-enforced the plausibility of the above technique.

Table of Contents

Contents.....**X**

1- Introduction.....**1**

1.1 Understanding of Soft Computing and Intelligent System.....**1**

1.2 Rationale of present research.....**3**

1.3 Outline of the Thesis.....**6**

1.4 Contributions of thesis.....**7**

2 - Identification Systems: State of the Art.....**9**

2.1 Mathematical Modelling.....**9**

2.2 Soft Computing Techniques.....**16**

2.2.1 Neural Networks.....**17**

2.2.1.1 Structure of Neural Networks.....**19**

2.2.1.2 Learning Using Neural Networks.....**21**

2.2.2 Fuzzy Systems.....**22**

2.2.2.1 Identification with Fuzzy modelling.....**23**

2.2.3 Hybrid Schemes.....**24**

2.3 Problem Description and Proposed Methodology.....**28**

3- Computational Intelligence Methodologies.....**30**

3.1 Artificial Neural Network (ANN).....**30**

3.1.1 Multi Layer Perceptron (MLP).....**31**

3.1.2 Backpropagation Algorithm.....**35**

3.1.3 Momentum Effect.....**37**

3.2 Elman Neural Network.....**38**

3.3 Radial Basis Functions (RBF).....**40**

3.3.1 Orthogonal Least Squares.....**41**

3.4 Fuzzy Logic.....**44**

3.4.1 TSK Fuzzy modelling.....**48**

3.5 Neuro-Fuzzy Systems.....**50**

3.5.1 Adaptive Neuro-Fuzzy Inference System(ANFIS).....**50**

3.5.2 FALCON.....**52**

3.5.3 NEFCON.....**53**

3.6 Adaptive Neuro-Fuzzy Network.....**54**

3.7 Case Study: Fungus Growth Modelling By Adaptive Neuro-Fuzzy Network.....**59**

3.7.1 Fungus Growth Modelling By MLP.....**65**

3.7.2 Fungus Growth Modeling By OLS-RBF……...……**66**

4 - Wavelet Neural Networks.....**69**

4.1 Time - Frequency Analysis.....**69**

4.2 Principles of Wavelet Transform.....**70**

4.3 Wavelet Neural Network.....**74**

4.4 Proposed Structure Scheme (WNN-LCW).....**78**

4.4.1 The Hybrid parameter learning scheme.....**81**

4.4.2 Case Analysis - Prediction of pressure inactivation of *Listeria monocytogenes* in whole milk.....**85**

4.4.3 Initialisation of the network parameters.....**86**

4.4.4 Dynamic System Identification.....**87**

4.4.5 Model Development.....**90**

4.4.5.1 Primary Modelling.....**90**

4.4.5.2 Non-Parametric Modelling.....**90**

4.4.6 Model Validation.....**91**

4.5 Proposed Structure Scheme II (MWNN-LCW).....**105**

4.5.1 The parameter learning scheme.....**107**

4.5.2 Case Analysis and Simulation Results.....**109**

5- Data Clustering Techniques.....**118**
5.1 Fuzzy Partitions.....**119**
5.2 Distance Norms.....**120**
5.3 Fuzz C-Mean Clustering.....**121**
5.4 Gustafson – Kessel Clustering.....**123**
5.5 Gath-Gava Clustering.....**125**
5.6 Subtractive Clustering.....**126**
5.7 Gaussian Mixture Models and Expectation Maximization.....**128**
5.7.1 Gaussian Mixture Model.....**128**
5.7.2 Maximum Likelihood Estimation (MLE).....**130**
5.7.3 Jensen's Inequality.....**131**
5.7.4 Expectation – Maximization for GMMs.....**132**
5.8 Identification with Fuzzy Clustering.....**135**

6 - Fuzzy Wavelet Neural Networks.....**136**
6.1 Clustering Based -FWNN structure and Construction.....**137**
6.2 CB-FWNN Antecedent Parameters update.....**141**
6.3 Estimation of local linear models.....**143**
6.3.1 Extended Kalman Filter.....**144**
6.3.2 Estimating Wavelet Neural Network (WNN) parameters using EKF....**146**
6.4 TSK Clustering Based –Fuzzy Neural Networks.....**147**
6.4.1 Consequence Parameter Updating.....**150**
6.5 Case Study – Short Term Load Forecasting in Power System.....**150**
6.5.1 Cluster Analysis for Case Study.....**154**
6.5.2 CB-FWNN Short Term Load Forecasting Results.....**155**

7 - Conclusion and Future Enhancements.....**163**
Bibliography.....**167**
Appendix I : Statistical Error Criteria.....**177**
Appendix II: List of Publications.....**179**

Table of Abbreviations

ANN	Artificial Neural Networks
AFNN	Adaptive Fuzzy Neural Network
AR	Auto Regressive
ARMA	Auto Regressive Moving Average
ARMAX	Auto Regressive Moving Average
BP	Back Propagation
CB	Clustering Based
CI	Computational Intelligence
CWT	Continuous Wavelet Transform
DWT	Discrete Wavelet Transform
EC	Evolutionary Computing
EKF	Extended Kalman Filter
EM	Expectation Maximization
FL	Fuzzy Logic
FWNN	Fuzzy Wavelet Neural Network
GA	Genetic Algorithm
GD	Gradient Descent
GMM	Gaussian Mixture Model
HLA	Hybrid Learning Algorithm
LCW	Linear Combination Weight
LS	Least Square
MA	Moving Average
MF	Membership Function
MLP	Multi Layer Perceptron
MRA	Multi Resolution Analysis
NARMAX	Non-Linear Auto-Regressive Moving Average with Exogeneous Input
NF	Neuro Fuzzy
NN	Neural Network
OLS	Orthogonal Least Square

PLS	Partial Least Square
RBF	Radial Basis Functions
RLS	Recursive Least Square
SC	Soft Computing
STLF	Short Term Load Forecasting
WLS	Weighted Least Square
WNN	Wavelet Neural Network
MWNN	Wavelet Neural Network with Multiplication Layer
ω_{ip}	p'th component of of i'th linear combination weight
$\hat{y}$	Estimated Output
E	Error
v_{jp}	Synoptic weight between j'th neuron and p'th input dimension
C	Total number of rules/clusters
μ	Centre of a univariate/multivariate Gaussian
Σ	Spread/width of a multivariate Gaussian in Matrix form
σ	Spread/width of a univariate Gaussian
γ_i	Firing Strength of i'th rule/cluster
$\tilde{\mu}$	Membership degree of p'th input on j'th membership function
η	Learning rate
ζ	Momentum
I_j^{ℓ}	Input to neuron j from layer ℓ
m_i^p	Wavelet dilation of i'th neuron of p'th sub-wavelet network
n_i^p	Wavelet shift of i'th neuron of p'th sub-wavelet network
O_j^{ℓ}	Output of neuron j from layer ℓ
P	Dimension of input patterns
W_i	Linear Combination Weight between the i'th rule/cluster to output

List of Figures

Fig 2.1 AR Model.....12

Fig 2.2 MA Model.....13

Fig 2.3 ARMA Model.....13

Fig 2.4 ARX Model.....14

Fig 2.5 ARMAX Model.....14

Fig 2.6 Box-Jenkins Model.....15

Fig 2.7 Some of possible hybrid Soft Computing techniques.....17

Fig 2.8 Neural Network identification structure.....18

Fig 2.9 **a)** NN with time-delayed direct inputs and time-delayed recurrent outputs from the modelled system.....20
b) NN with time-delayed direct inputs and time-delayed recurrent outputs from ..... the Actual Plant.....20

Fig 3.1 A single perceptron.....31

Fig 3.2 MLP Structure.....32

Fig 3.3 Tangent Hyperbolic.....34

Fig 3.4 Sigmoid Function.....34

Fig 3.5 Effect of various learning rates on convergence of the weights.....37

Fig 3.6 Elman Network structure.....39

Fig 3.7 RBF network with Gaussian activation.....41

Fig 3.8 Fuzzy logic system 45

Fig 3.9 Three types of membership functions.....46

Fig 3.10 The structure of ANFIS (type III) with two inputs and one output.....51

Fig 3.11 Falcon Neuro-Fuzzy architecture.....53

Fig 3.12 NEFCON architecture.....54

Fig 3.13 Three input and one output Adaptive Neuro-Fuzzy scheme.....55

Fig 3.14 Training samples simulation results for Adaptive Fuzzy Neural Network.....61

Fig 3.15 (a) Initial membership functions for each normalized input.(b)-Memberships after structure learning process for each normalized input (c)- Final membership function together with parameter learning 63

Fig 3.16 a) Temperature and Water Activity vs. estimated output curve for normalized training data and the actual output curve b) Predicted Temperature and PH output surface and actual output surface c) PH and Water Activity predicted and desired output surface.....64

Fig 3.17 Number of epochs and the related sum square error.....65

Fig 3.18 MLP training results for each input pattern of *Fungus* Growth.....65

Fig 3.19 a) OLS-RBF regressors index and their contribution to error reduction 67
b) OLS-RBF training results for each input pattern of Fungus growth data.....67

Fig 4.1 Haar Wavelet functions.....71

Fig 4.2 DWT using Haar Wavelet.....74

Fig 4.3 Structure of wavelet neural network.....76

Fig 4.4 Morlet Wavelet basis function.....77

Fig 4.5 Linear Combination Weight Wavelet Neural Network Structure.....80

Fig 4.6 a) Series Parallel dynamic system b) One step ahead prediction c)Parallel mode.....89

Fig 4.7 Survival Curves of *Lysteria* in various pressures.....92

Fig 4.8 Survival curves of *Listeria monocytogenes* in UHT whole milk during high pressure processing at 350 MPa (•), 450 MPa (▲), 550 MPa (◆), and 600 MPa (■), generated by the re-parameterised Gompertz model (a), the modified Weibull model (b), the Geeraerd model (c), and the wavelet neural network(d). Data points are mean values of two independent experiments with two replications each.....94

Fig 4.9 Observed values and predicted survival curves of *Listeria monocytogenes* in UHT whole milk during high pressure treatment at 400 MPa, generated by the reparameterized Gompertz model (a), the modified Weibull model (b), and the wavelet neural network (c). Data points are mean values of two independent experiments with two replications each.....96

Fig 4.10 Observed values and predicted survival curves of *Listeria monocytogenes* in UHT whole milk during high pressure treatment at 500 MPa, generated by the

reparameterized Gompertz model (a), the modified Weibull model (b), and the wavelet neural network (c). Data points are mean values of two independent experiments with two replications each.....96

Fig 4.11 Continuous Survival curves of *Lysteria monocytogenes*.....98

Fig 4.12 PLS regression model on a two-input case.....99

Fig 4.13 Survival curves of *Listeria monocytogenes* during high pressure treatment at 400MPa fitted with different modelling schemes.....101

Fig 4.14 Survival curves of *Listeria monocytogenes* during high pressure treatment at 500 fitted with different modelling schemes.....101

Fig 4.15 Convergence speed comparison by number of epochs using pure GD.....105

Fig 4.16 Convergence speed using Hybrid Learning method 105

Fig 4.17 Architecture of WNN with multiplication layer and Linear Weights.....107

Fig 4.18 Survival curves of *Listeria monocytogenes* in UHT whole milk during high pressure processing at 350 MPa (•), 450 MPa (▲), 550 MPa (◆), and 600 MPa (■), generated by the re-parameterised Gompertz model (a), the modified Weibull model (b), the Geeraerd model (c), and the wavelet neural network (d). Data points are mean values of two independent experiments with two replications each.....110

Fig 4.19 Observed values and predicted survival curves of *Listeria monocytogenes* in UHT whole milk during high pressure treatment at 400 MPa, generated by the re-parameterised Gompertz model (a), the modified Weibull model (b), the Geeraerd model (c), and the wavelet neural.....111

Fig 4.20 Observed values and predicted survival curves of *Listeria monocytogenes* in UHT whole milk during high pressure treatment at 500 MPa, generated by the re-parameterised Gompertz model (a), the modified Weibull model (b), the Geeraerd model (c), and the wavelet neural network (d). Data points are mean values of two independent experiments with two replications each.....112

Fig 4.21 Survival curves of *Listeria monocytogenes* during high pressure treatment at 400MPa fitted with different modelling schemes.....114

Fig 4.22 Survival curves of *Listeria monocytogenes* during high pressure treatment at 500MPa fitted with different modelling schemes.....114

Fig 4.23 Epochs using Hybrid Method. For MWNN structure.....116

Fig 4.24 Epochs using only Gradient Descent for WNN-M structure.....116

Fig 5.1 Gaussian Mixture Model.....129

Fig 6.1 Proposed CB-FWNN.....139

Fig 6.2 TSK Clustering-Based Fuzzy Neural Network.....148

Fig 6.3 Proposed modular architecture for the STLF problem.....153

Fig 6.4 Eigenvectors of a 3-dimension hyper-ellipsoidal cluster.....154

Fig 6.5 Projection of normalized Multivariate (4-dimesnion) clusters on their eigenvectors.155

Fig 6.6 Training performance for max load.....157

Fig 6.7 Training performance for min load.....157

Fig 6.8 Testing performance for max load.....159

Fig 6.9 Testing performance for min load.....159

List of Tables

TABLE 3.1 Normalised means and deviations of all Membership Functions with and without parameter learning 62

TABLE 3.2 Adaptive Neuro-Fuzzy Error Coefficients for Training and Testing dataset.....62

TABLE 3.3 MLP Error coefficients for training and testing ……....66

TABLE 3.4 OLS-RBF Error coefficients for training and testing.……68

TABLE 4.1 Parameters and statistics of secondary models for the effect of high pressure on the kinetic parameters of *Listeria monocytogenes* in UHT whole milk.....93

TABLE 4.2 Parameter estimation and statistical indices of the different models used for fitting the survival of *L. monocytogenes* in whole UHT milk during high pressure treatment. 95

TABLE 4.3 Performance indices of various methods for *Lysteria Monocytogen* data.....102

TABLE 4.4 Convergence comparison of existing models on prediction problem.....104

TABLE 4.5 MWNN-LCW and other methods Statistical index comparison.....115

TABLE 4.6 Performance comparison of existing models on prediction problem.....115

TABLE 4.7 Wavelet Parameters of MWNN-LCW after Optimisation.....117

TABLE 6.1 Performance indices of proposed CB-FWNN for Short Term Load Forecasting....156

TABLE 6.2 Comparison of performance indices of 2hr STLF for various methods.....160

TABLE 6.3 Comparison of performance indices of 14hr STLF for various methods.....161

Acknowledgment

Many people have aided the production of this project and I greatly indebted to them. This thesis could not have been documented without the support of them.

First and foremost, I would like to express my profound gratitude to my Director of studies, *Dr. Vassilis Kodogiannis.* His inspiring ideas and insightful guidance have been always a massive motivation. I am very grateful to him for letting me join his creative research group of Computational Intelligence. He has been a driving force behind my research by providing me with critical advice and always having an open ear for problems. Having the opportunity to work with him has been a great pleasure indeed.

I also deeply appreciate the support of *Dr. Andrzej Tarczynski* during the first steps of my research. His critical and detailed questions and stimulating discussions directed my scientific methodologies.

I also owe thanks to academic staff of Agricultural University of Athens, Department of Food Science and Technology- Greece, for their cooperation and providing the *Ascomycetous Fungus* and *Listeria monocytogenes* data sets.

In particular, I am most grateful to my lovely parents *Hamid* and *Nasrin* and beloved sisters *Setareh* and *Taraneh* who provided never-ending spiritual support, encouragement and love throughout my PhD study. I sincerely thank my sister, *Taraneh,* for making suggestions for last-minute improvements.

"I declare that all the material contained in this thesis is my own work"

Mahdi Amina

Chapter 1
Introduction

1.1 Understanding of Soft Computing and Intelligent Systems

When we interact with a system, we need some concept of how its variables relate to each other, with a broad definition such an assumed relationship called *model* of the system. Model is normally known as simplified representation of a system, in time or space intended to promote understanding of the real system. On the list of data analysis tasks frequently occurring in applications, modelling occupies very high, if not the highest rank. As a consequence a large variety of methods to tackle these tasks have been developed ranging from different sorts of mathematical modelling to more advanced soft computing techniques. Among all these methods, soft computing approaches draw intense interest in data analysis. This interest is mainly due to extremely rapid growth of complex plants/systems which has rendered mathematical modelling virtually impossible. When attempting to solve real-world problems, we realize that there are typically ill-defined systems to analyze and difficult to model. In these cases, precise models are impractical, too expensive, or non-existent. Furthermore, the relevant available information are mainly in form of empirical prior knowledge and input-output data representing instances of the systems' behaviour. Therefore, we need an approximate reasoning system capable of handling such imperfect information. Soft Computing techniques originated from emulating intelligent phenomenon in nature, their main scope is on the study of adaptive mechanism to enable or facilitate intelligent behaviour in complex and changing environment. It includes paradigms like neural networks, evolutionary computation, swarm intelligence, fuzzy system and so forth.

Traditional quantitative models in modelling have two main disadvantages. First, conventional mathematical and statistical models usually require complicated formulae, and hence they may be considered as “grey-box” type models for their users even if they are familiar with advanced mathematics. Since these models require advanced mathematical skills and notations, they are often laborious with respect to calculations and computations. Second, models based on bivalent logic already seem outdated because they often yield excessively coarse or otherwise problematic outcomes (even paradoxes). Hence, the research community requires the development of more user-friendly and powerful theories and models. These difficulties lead to a number of challenging problems, i.e., “incorporate” the human intelligence into a machine, because there is a huge gap between the human intelligence and the machine intelligence. In order to cope with the difficulties mentioned above, an emerging framework - soft computing - has been developed recently, which has the following properties:

- *Soft computing* is pointed towards the analysis and design of intelligent systems. It consists of fuzzy logic, artificial neural networks and probabilistic reasoning including evolutionary algorithms, and parts of machine learning and has the attributes of approximation.
- *Soft computing* is aiming at a formalisation of the human ability to make rational decision in uncertain and imprecise environment;
- The constituents of *soft computing* are complementary rather than competitive. The experiments gained over the past decade have indicated that it can be more effective to use them in a hybrid manner, rather than solely;
- *Soft computing* is an open framework. This means its framework can always be incremented by newly created techniques come from the imitating of the human/natural.

Applications of hybrid soft computing systems are currently used in such diverse industrial and commercial fields. In these areas, some combinations of hybrid soft computing systems, such as fuzzy logic controller tuned by neural networks and evolutionary computing, neural network tuned by evolutionary computing or fuzzy logic system, and evolutionary computing tuned by fuzzy logic systems have been considered. Applications in diagnostic systems, control, and prediction were received greater attention in past years.

1.2 Rationale of present research

Modelling and identification of nonlinear dynamic systems is a chanllenging task because nonlinear processes are unique in the sense that they do not share many properties. A major goal for any nonlinear system modelling and identification scheme is universalness: that is the capability of decribing a wide class of structurally different systems. In this context, a great effort is being made within the area of system identification, towards the development of nonlinear models of real processes. In addition to more classical identification methods such as NARMAX modelling, a new set of methods has been developed recently which apply artificial neural networks and fuzzy systems to the tasks of identification of dynamic systems. These works are supported by two of the most important capabilities of neural networks, i.e. their ability to learn (based on the optimization of an appropriate error function) and their good performance for the approximation of nonlinear functions, as well as the main characteristic of fuzz systems, i.e. fuzzy rules / defuzzification schemes. Fuzzy systems accept numeric inputs and convert theses into linguistic values (represented by fuzzy numbers) that can be manipulated with linguistic IF-THEN rules and with fuzzy logic operations, such as fuzzy implication and composition rules of inference. However, at present there is no systematic procedure for the design of a fuzzy system. Usually the fuzzy rules are generated by converting human operators' experience into fuzzy linguistic form directly and by summarizing the system behaviour (sampled input-output pairs) of the operators. But designers find it difficult to obtain adequate fuzzy rules and membership functions because these are most likely to be influenced by the intuitiveness of the operators and the designers. Neural network models basically use the sigmoid activation function in neurons. However, the sigmoid function normally appeared in neural networks is not orthogonal, and the energy of the sigmoid function is limitless, and this leads to slow convergence. Wavelet function is a waveform that has limited duration and an average value of zero. The integration of the localisation properties of wavelets and the learning abilities of neural networks shows advantages of wavelet neural networks over neural networks in complex nonlinear system modelling in terms of learning efficiency and structure transparency.

Neurofuzzy hybrid modelling approaches have been introduced as an ideal technique for utilising such knowledge and empirical data. Based on the similarities between fuzzy systems and some neural networks, neurofuzzy approaches combine the desired attributes of both the fuzzy and the neural paradigms hence producing flexible models which can learn from empirical data and can be

represented linguistically by fuzzy rules. For modelling of dynamic processes, neurofuzzy systems incorporating a Takagi-Sugeno-Kang (TSK) scheme possess a very good interpretation, which is superior to most, if not all, alternative defuzzification approaches. However, in the case of modelling of complex nonlinear processes, TSK-type fuzzy systems may require a high number of rules in order to achieve the desired accuracy. Increasing the number of the rules leads to an increase in the number of parameters needed to be calculated.

This thesis investigates the ability of wavelet-based soft computing approaches to learn how to identify adequately complex nonlinear systems. A hybrid soft computing framework has been constructed and applied to the identification of nonlinear dynamic modelling. The major motivation for this research is that current hybrid implementations of soft computing techniques suffer from the lack of efficient constructive methods, both in determining the parameters and in choosing network structure. To remedy the weakness of traditional computational intelligent systems, in this thesis some novel wavelet-based methods have been proposed in order to improve the performance of existing modelling schemes.

Encouraged by the potential strengths of the idea of combining both wavelet decompositions and the feed-forward neural networks, a Wavelet Neural network scheme has been proposed [1]. Inspired by theory of multi-resolution analysis(MRA) of wavelet transforms and fuzzy concepts, the Fuzzy Wavelet Networks(FWNNs) concept was introduced in [2]. The combination of fuzzy logic and WNNs in FWNNs not only reserves the multi-resolution capability of WNNs, but also enjoy the advantages of high approximation accuracy and good generalization performance. However, existing WNN/FWNN methods for dynamic system identification suffer from i) lack of an efficient constructive model, ii) slow convergence rate when high dimensional data exist, iii) low identification accuracy when imprecision in the measured data exists and iv) the need to find the model structure by trial and error, a problem that is has been addressed with the proposed in this thesis novel FWNN concept.

In this research, through innovative applications and adroit integration of emerging information technologies, a signal processing method (wavelets), and two soft computing methods (fuzzy logic and neural network), novel WNNs and Fuzzy Wavelet Neural network models have been developed for modelling and identification purposes.

A step-by-step constructive approach has been adopted in the presentation of the developed methodologies. Initially, a study on one popular neurofuzzy system scheme has been performed in order to investigate its strength over alternative non-hybrid schemes as well as its major

weaknesses. The specific neurofuzzy scheme was adopted due to its TSK defuzzification scheme which has influenced the design of the proposed in this thesis FWNN. This neurofuzzy scheme was evaluated using real food data, acquired from Agricultural University of Athens. The specific experiment was performed in order to verify scheme's performance to the static identification of a nonlinear process. Nevertheless, as the main focus of this thesis is the modelling of dynamic rather static nonlinear processes, in the next stage, two novel wavelet neural networks have been developed. Their design has been influenced by the classic TSK neurofuzzy systems. The static weights scheme appeared in classic wavelet neural networks, have been replaced here by a linear TSK-combination weight scheme.

The efficiency of the new WNN structures has been evaluated through the dynamic identification of a complex nonlinear case study related to food analysis and acquired from Agricultural University of Athens.

Emphasis in this particular case study has been given to the performance (accuracy / training speed) of the developed WNN schemes, through the comparison against existing regression and intelligent methodologies. The challenge with the specific dataset was the rather limited number of samples/patterns and thus methods how to handle small number of samples with dynamic behaviour had to be developed.

The ultimate goal of this thesis is the development of a prototype FWNN. However in order to develop such efficient and novel scheme, a number of sub-components related to FWNN had to be developed. The developed WNNs have replaced the classic linear TSK defuzzification part. However, in a hybrid fuzzy-based system, accuracy is not the only issue of consideration. The computational cost, associated with the number of fuzzy rules, is of equally importance. It is well known that efficient performance in hybrid systems is closely related to the number of samples/patterns. But in traditional hybrid schemes, this results in to an increased number of required fuzzy rules and subsequently to a large number if parameters to be calculated. In this thesis a new type of clustering technique has been introduced as an essential part of the proposed FWNN. The proposed FWNN concept has been evaluated with a large dataset related to load forecasting of the power system of the island of Crete, Greece. The embedded in the FWNN clustering sub-systems managed to provide accurate predictions, and such result was also associated with an efficient relatively small structure (i.e. fuzzy rules).

In general wavelet-based hybrid methods and their applications are comparatively new and research is being carried out continuously in many universities and research institutions worldwide.

1.3 Outline of the Thesis

The thesis is organised into seven chapters. Following this introductory chapter the next chapter, Chapter 2, gives an overview of modelling focusing on some traditional linear systems and fundamental concept of some of classic techniques mentioned above. It also introduces the various types of hybrid architectures highlighting some of their weaknesses and advantages, followed by explaining the necessity of such merging. This chapter finalised by explaining the problems this thesis trying to tackle and the criteria going to be considered when proposing new schemes in following chapters.

Chapter 3 delves further beyond and presents detailed discussions and mathematical formulation of some selected computational intelligence architectures. It starts with dynamics of neural networks and its training algorithms and analysing some other variants of it according to connections layout and activation functions. Fuzzy systems and its concept, together with three hybrid neuro-fuzzy structures are the other main topics outlined in this chapter. Finally, a Neuro-Fuzzy scheme equipped with adaptive structure learning was developed, and tested on a Food Microbiology dataset.

Chapter 4 attempts to give a detailed introduction of wavelet transform. This chapter then addressed some existing WNNs through some literature review. Two proposed new WNN structures with a hybrid learning scheme are then proposed. Each of proposed schemes have been examined with a real dynamic biological system. We then provide comprehensive result analysis, and performances evaluated against many other techniques.

Chapter 5 is a background introductory chapter to the concept of clustering and mainly focuses on various fuzzy clustering techniques with objective function and its applications. This chapter reviews the potential of clustering algorithms to reveal the underlying structures, not only for classification and pattern recognition, but also for the reduction of complexity in modelling and optimization. More specifically, the Expectation-Maximization (EM) and Gaussian Mixture Models (GMM) as a probabilistic framework discussed in this chapter. The latter will be utilised in the following chapter. This chapter's aims focus on providing sufficient background theory to be able to study and develop a novel scheme in the following chapter.

Chapter 6 elaborates on the fusion of the two former chapters in order to determine the most extensive of this research. A novel clustering-based FWNN suggested and explained. The flows of the signal from input to output and the dynamics of the structure are elaborated. The EM clustering

method and two training algorithms, i.e. Extended Kalman Filter and Weighted Least Square which are used in conjunction with each other to adjust non-linear and linear parameters of the networks, are deployed. The mentioned section which is the core part of the research, applied to a dynamic application known as Short Term Load Forecasting (STLF).
Chapter 7 draws conclusions and possible directions for future work. Four recommended enhancements, which were out of the scope of this project, are presented here.

1.4 Contributions of thesis

Over the period of this research project, certain contributions to the field of hybrid soft computing techniques have been offered, by exploring a number of modifications and innovations. They are mainly around the wavelet-based neural networks and Fuzzy -neural networks. This area is relatively new and has growing importance. Below is a list of these innovations. A full description of each point can be found latter in this document.

1. Initially, the general practicability of conventional Neuro-Fuzzy modelling has been enhanced by applying an Adaptive Neuro Fuzzy structure into a Biological application.

2. Wavelet Neural Network conventional structure was reviewed and revised. Two distinguished new fully tuneable schemes of WNNs were introduced. They are different in number of layers, activation function and mainly the connection configuration between the layers. They have levels of novelty both in the structure and in the learning algorithm.

    - Local Linear Combination weights applied in conjunction with a hybrid learning algorithm.
    - One-Step-Ahead prediction of *Lysteria Monocytogene Bactria Survival* curves

    The new structures contributed significantly to both accuracy and computational cost when facing a real world dataset.

3. A novel FWNN scheme proposed and functionality approved through comprehensive comparisons on a real world dynamic dataset, and several error criteria. The novel

FWNN scheme is considered as an evolutionary version of previously proposed FWNN and addressed several drawbacks existing in hybrid methods.

- Clustering for the first time embedded into a wavelet-based structure. Significant reduction in fuzzy rule and overcoming with problems occur as the result of increasing number of features and dimensions (*curse of dimensionality*) are the main outcomes. The clustering conducted in input-output space.
- New hybrid of learning method i.e. Extended Kalman Filter together with Weighted Least Square, alleviate the convergence speed.
- Also a modified version of Expectation-Maximization responsible for partitioning the data as well as finding the cluster parameters. The modified version enabled with a feedback link from output error into the clustering process.
- Probabilistic interpretation of fuzzy clusters (Gaussian Mixture Model) which assists in extracting the fuzzy membership without projecting them onto the input axis.

4. Automated number of clusters and initialisation of cluster parameters. This performed via Subtractive Clustering.

Chapter 2

State-of-the-Art in System Identification

How to better understand and replicate the real world around us is a long-established issue. Models of the real world have provided a vital means of creating a link between theory and proof. In information processing, the objective is generally to gain an understanding of the phenomena involved, and to evaluate relevant parameters quantitatively. This is usually accomplished trough *'modelling'* or *'identification'* of the system, either experimentally or analytically.

System modelling is a technique to express, visualise, analyse and transform the architecture of a system. In loose terms, a system is an object in which variables of different kinds interact and produce observable signals. The observable signals that are of interest to us are usually called outputs. The system can be affected by external stimuli as well. External signals that can be manipulated by the observer are called the inputs. The activities and tasks that turn the inputs into products and services are called Processes.

A system may consist of software components, hardware components, or both and also the connections between these components. In this sense a system model is then considered as a skeletal model of the system. System modelling uses three elements: inputs, processes, and outcomes.

2.1 Mathematical Modelling

In mathematical modelling, we translate those behaviours into the language of mathematics. This has many advantages;

1. Mathematics is a very precise language. This helps us to formulate ideas and identify underlying assumptions.
2. Mathematics is a concise language, with well-defined rules for manipulations.
3. All the results that mathematicians have proved over hundreds of years are at our disposal.
4. Computers can be used to perform numerical calculations.

The primary concern of a system modeller is to obtain a mathematical representation of system's behaviour under study in terms of physically significant variables. Any system modelling consists of two steps, model design and performance evaluation. There is a large element of compromise in mathematical modelling. The majority of interacting systems in the real world are far too complicated to model in their entirety. Hence the first level of compromise is to identify the most important parts of the system. These will be included in the model, the rest will be excluded. The second level of compromise concerns the amount of mathematical manipulation which is worthwhile. Although mathematics has the potential to prove general results, these results depend critically on the form of equations used. Small changes in the structure of equations may require enormous changes in the mathematical methods. Using computers to handle the model equations may never lead to elegant results, but it is much more robust against alterations. Mathematical models in terms of their nature can be in various ways:

Dynamic vs. Static Models: Dynamic systems may be complex industrial plants where the dynamic relationship between the inputs and the plant behaviour must be modelled. The inputs to the dynamic system often represent the system state at a previous time step and the mapping is between the current system state and the one at the next time step. The output from such a system is often a continuous value or series of values which may vary independently unlike the classification systems where they are normally linked. Applications of this type include attempts to identify the underlying processes in financial assets, engineering and control applications, food microbiology and load forecasting. In contrast to the dynamic systems that are described by differential or difference equations, the static systems are described by algebraic equations. Typical examples of identification of static system include problems where input variables are not time-dependent or pattern recognition problems.

Linear vs. Nonlinear Models : Dynamic system models are either linear or nonlinear. A linear model obeys the principle of superposition and homogeneity[3]. The following equations are true for linear models.

$$
\begin{aligned}
y_1 &= f(x_1) \\
y_2 &= f(x_2) \\
f(x_1 + x_2) &= f(x_1) + f(x_2) = y_1 + y_2 \\
f(ax_1) &= af(x_1) = ay_1
\end{aligned}
\tag{2.1}
$$

Where x_1 and x_2 are the system inputs, y_1 and y_2 are the system outputs, and 'a' is a constant. Conversely, nonlinear models do not obey the principles of superposition or homogeneity. Many real-world systems are nonlinear, though we can many times linearize them to simplify a design or analysis procedure. Linear modelling techniques are capable of modelling nonlinear processes if the nonlinear characteristics are weak. Their strengths come from the fact that they contain a small number of parameters and so long, there are few noisy measurements they perform adequately. This means that often it is possible to calculate a linear model for a data set that is too sparse for more complex nonlinear models. A linear model is simply a weighted sum of a set of inputs that describe a hyper-plane across the input space. The parameters can be estimated simply using a least squares technique, with online optimisation realized using a recursive least squares technique. Generally linear models can be divided into parametric and non-parametric models.

- Parametric models assume that the process can be modelled with a finite number of parameters. These parameters often have a direct relationship to the physical qualities of the process. Examples of these types of models can be found in differential equation models. Linear regression techniques can be used to identify the parameter. These models in turn may be used for the approximation of non-parametric techniques where the number of parameters has been reduced to a finite number.
- Non-parametric models often require an infinite number of parameters to describe the process exactly. They are used when less structure is to be imposed on the model. Although in theory these methods have no fixed parameters, in the end they require a finite number of parameters to be imposed during implementation.

This section is concerned with parametric models based around the time domain, as these are the methods most commonly used in process and control engineering.

The general linear model structure process is presumed to consist of a series of inputs u(t) and an output process y(t). If the system is purely deterministic, i.e. the noise process is negligible, then the system output y(t) can be computed by passing a set of input parameters or a state vector

through a linear filter called the Input Transfer Function. Variable q here denotes the forward shift operator. A stochastic white noise model can be added to this by filtering the white noise process, v(k), through a second linear filter called the Noise Transfer Function. Each of these can be assumed to possess a numerator and a denominator, often with an assumed shared denominator factor. As a result the general linear model can be given by [4].

$$y(k) = \frac{B(q)}{F(q)A(q)}u(k) + \frac{C(q)}{D(q)A(q)}v(k) \tag{2.2}$$

Not all of the numerators and denominators in the general linear model are used in each modelling scheme. For some applications the input variables are unknown or too numerous to identify properly. In these applications such as these the stochastic series represented by the previous system outputs are generally used. In terms of the general linear model this leads to the u(k) term being discarded. The presence or absence of the terms numerators and denominators of the v(k) and u(k) parts of eq (2.2) further classifies these systems. The simplest form of these is the stochastic model with just the denominator D(q) present. These are called autoregressive (AR) models as shown below.

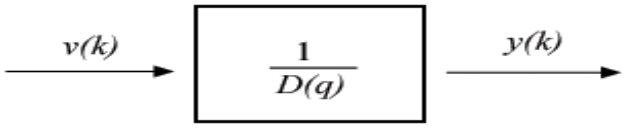


Fig 2.1- AR Model

So the transfer equation is as follows.

$$y(k) = \frac{1}{D(q)}v(k) \tag{2.3}$$

The parameters can be calculated using a simple Least Squares (LS) technique making them easy to identify. However they are capable of modelling only series with AR characteristics. They also suffer from model order selection problems when the data sets are too small. This leads to the model containing spurious peaks [5].

When just the numerator is present then it is called a moving average (MA) model. These are generally far less applied in engineering applications as the parameter identification process is

nonlinear. Again there are real problems with finding the minimum order for the MA model and although solutions have been proposed they are often computationally intensive requiring frequency analysis [6].

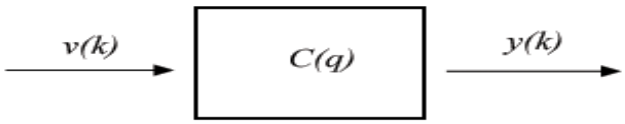


Fig 2.2- MA Model

So the transfer equation corresponding with figure 2.2 is

$$y(k) = C(q)v(k) \tag{2.4}$$

As can be seen the noise parameters at each stage must be estimated. This leads to the need for another model to estimate the parameters. Joining both these schemes together give the autoregressive moving average (ARMA) model.

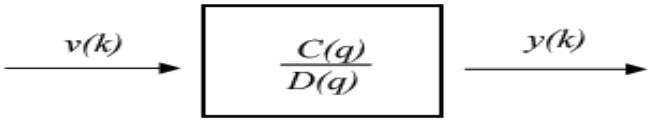


Fig 2.3- ARMA Model

Here, the transfer equation is as follows:

$$y(k) = \frac{C(q)}{D(q)}v(k) \tag{2.5}$$

This resolves the problem of needing to estimate the v(k) parameters in the MA model. A two-stage optimisation can be used. First the AR parameters are estimated as normal using technique such as LS. The resulting AR model is then used to provide v values for each reading and a simple linear optimisation technique can then be used for the MA model. There are a number of other optimisation techniques can be used to identify the parameters to the ARMA such as correlation based techniques and maximum likelihood (ML) method. These integrated techniques are capable of modelling both AR and MA series as well as ones which integrate both types of patterns [7].

Adding in extra inputs or exogenous inputs may extend all of these methods. These turn the purely stochastic models described here into stochastic-deterministic hybrids. The inclusion of exogenous

inputs to an AR model is called an autoregressive model with eXogenous inputs or ARX model. It retains the output feedback of the AR model but adds to this a number of parameters that are known to affect the system state.

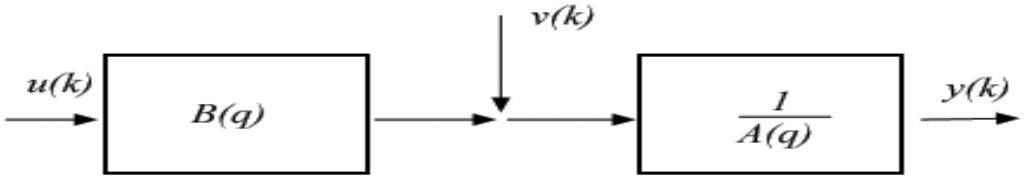


Fig 2.4- ARX Model

So the transfer equation is as follows.

$$y(k)=\frac{B(q)}{A(q)}u(k)+\frac{1}{A(q)}v(k) \tag{2.6}$$

The ARX model is widely used because the parameters can be computed simply with linear techniques such as LS. The technique runs into difficulty when it is modelling data that deviate systematically from the mean. Also the assumption that the system is capable of being modelled in a purely deterministic fashion is also often inaccurate. The inclusion of a more complete stochastic noise model leads to the ARMAX model. This model assumes that there is a shared denominator for the noise transfer function and the input transfer function.

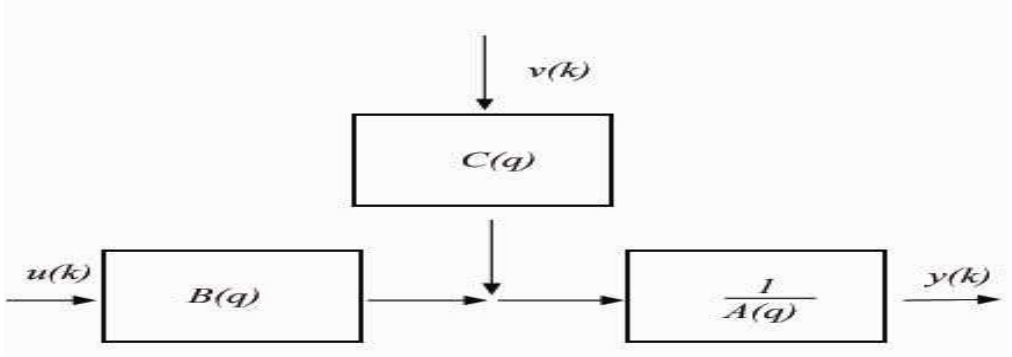


Fig 2.5- ARMAX Model

So the transfer equation is as follows.

$$y(k)=\frac{B(q)}{A(q)}u(k)+\frac{C(q)}{A(q)}v(k) \tag{2.7}$$

Each of the previous models incorporates different parts of the general linear model. For all of these linear techniques with exogenous inputs model order selection can be a problem often requiring a heuristic approach. For a full implementation of it the Box-Jenkins model may be used.

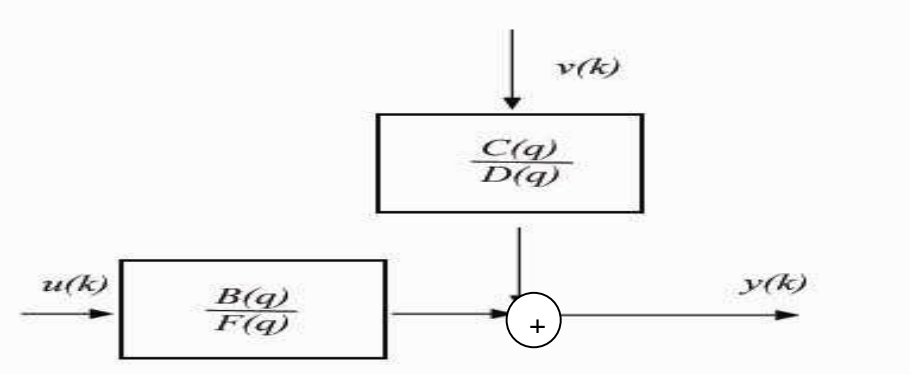


Fig 2.6- Box-Jenkins Model

So the transfer equation is as follows.

$$y(k) = \frac{B(q)}{F(q)} u(k) + \frac{C(q)}{D(q)} v(k) \tag{2.8}$$

This model does suffer from the fact that it has a large number of parameters that must be estimated. If the data is sparse or noisy then this becomes difficult and is highly unlikely to yield a valid model. The parameter estimation process is also inherently nonlinear and is usually tackled by estimating an ARX model and then using this to estimate the parameters for the MA part [8]. There are nonlinear extensions of the ARMAX and ARX models called, unsurprisingly Nonlinear ARMAX (NARMAX) and Nonlinear ARX (NARX). Here the simple linear function used in ARMA and ARMAX is replaced with a nonlinear mapping function. Often it is the NARX form that is most generally applicable to the widest range of nonlinear dynamic systems. In practice the form of the nonlinearity is unknown and as a result all forms of the polynomial must be considered. This generally means that a prohibitively large number of coefficients must be evaluated [9].

In general,there are several limitations on modelling based on mathematical analysis. First, it always relies on the accuracy of the mathematical model, which is never a prefect representation of the plant. And second, there is a need for the development of analysis techniques for even more sophisticated non-linear systems.

2.2 Soft Computing Techniques

Following our overview of conventional mathematical modelling, problems exist in traditional techniques. We can easily conclude that the currently ongoing complicated plants cannot be accurately described by traditional rigorous mathematical models. Especially non-linear dynamic systems can exhibit extremely complex dynamic behaviour. As discussed earlier, the traditional approaches for predicting the behaviour of such systems based on analytical conventional techniques in many cases can prove to be insufficient. In addition, there is need for the development of highly precise models and autonomous behaviour in system identification, control, and artificial life communities. However in real applications, precision has a cost (computational/financial), therefore in order to solve the problem with an acceptable cost, we need to aim at a decision with only the necessary degree of precision and not exceeding the requirements. These deficiencies lead to a fundamental remedy, which is the core part of soft computing concept i.e. embedding the human intelligence into a machine. So, it is of great importance to change the direction toward intelligent computational tools that will enable the identification of the best model by a series of input-output pairs.

Soft Computing(SC) techniques refers to a collection of computational tools which have their origins in biological or behavioural phenomena related to humans. Unlike traditional Hard computing techniques, SC can tolerate imprecision, uncertainty and partial truth without loss of performance and effectiveness. The term SC in its broadest sense, encompasses a number of technologies that include, but not limited to, evolutionary computation (EC) realizes intelligence through the simulated evolution artificial neural networks (ANNs) realize intelligence through the simulated behaviour of neurons in brain, fuzzy logic (FL) realizes intelligence through the simulated behaviour of human reasoning process[10].

It was at the beginning of 1990s when researchers realised that the *Hybrid* use of the methodologies mentioned, would lead to tools that were certainly more powerful than if the techniques were employed individually. Combination of soft computing techniques is considered to be the new frontier of *Artificial Intelligence*.

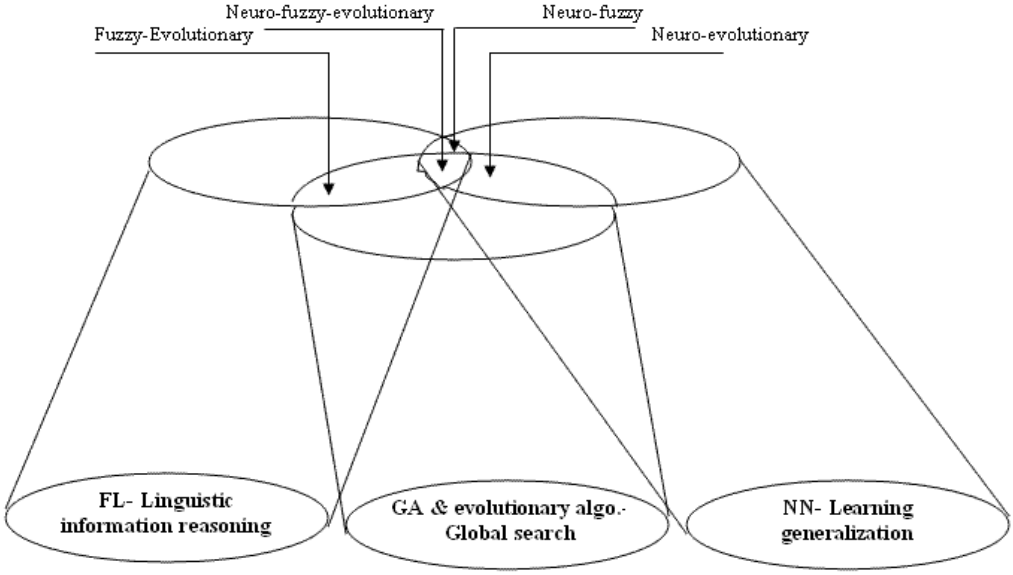


Fig 2.7 – Some of possible hybrid Soft Computing techniques

The modern techniques of artificial soft computing have found applications in almost all the fields, however the great emphasis is given to engineering area[11].

There are some common problems to be solved in soft computing identification, independently of the data type and description method. As a prelude, in this chapter, we provide a brief overview of two of the most common artificial intelligence modelling approaches Artificial Neural Networks (ANN) and Fuzzy Logic (FL) systems together with their hybrid Neuro-Fuzzy (NF) systems. These two classic approaches, ANN and FL, are examined in some depth.

2.2.1 Neural Networks

Neural networks are composed of simple elements operating in parallel. These elements are inspired by biological nervous systems. As in nature, the network function is determined largely by the connections between elements. Neural Networks approach the modelling by using precise inputs and outputs which are used to 'train' a generic model which has sufficient degrees of freedom for a good approximation between inputs and outputs. A neural network can be trained to perform a particular function by adjusting the values of the connections (weights) between the elements. One of the most common processes for which NNs are used in system modelling is the one involves : placing the NN in parallel with physical system, applying the system input to the

input of the network, using system output as the desired output for the neural network, and train the neural network until the error between the system output and the network reaches and acceptable level[12]. Here, the network is adjusted, based on the comparison between the output and the target. The Schematic diagram of such an identification of a time-invariant, causal system is shown in figure 2.8

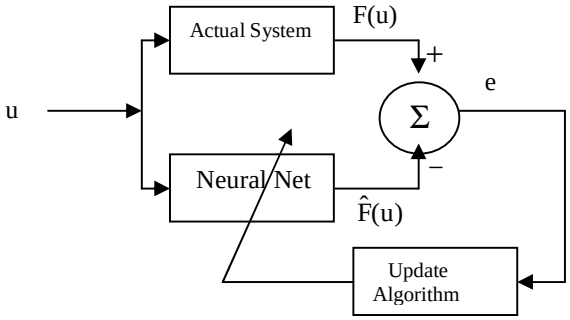


Fig 2.8 – Neural Network identification structure

In general, by a function F, compact input sets $U_j \subset \Re^p$ are mapped into elements y_j for j=1,..,N in the output space. Whereas, in the case of a dynamic system, we have input-output pairs of time u(t),y(t). The main objective in both type is to determine $\hat{F}$ such that, the input and output of the plant is given by u and F(u) respectively. The error e is the difference between the observed system output and the output generated by $\hat{F}$.

$$\left\| y - \hat{y} \right\| = \left\| F(u) - \hat{F}(u) \right\| < e \tag{2.9}$$

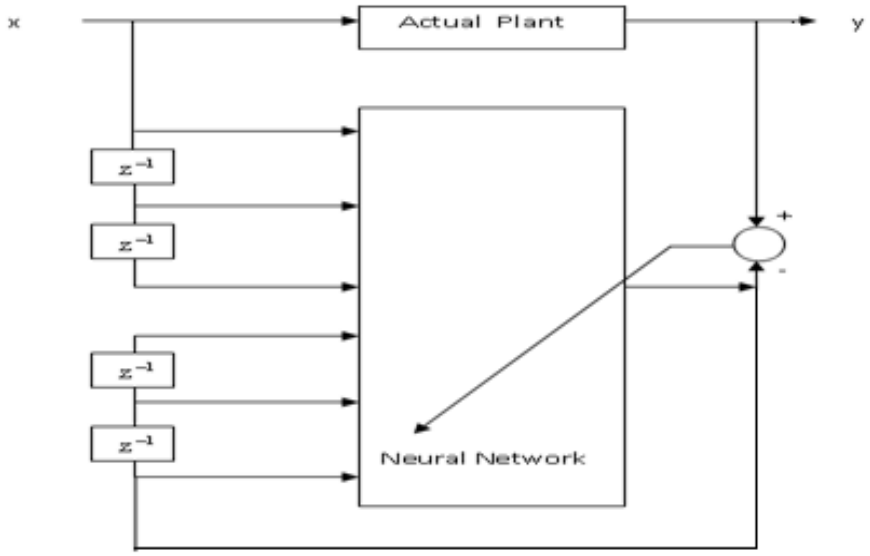
The main characteristic of the neural networks is the fact that these structures can learn with examples (training vectors, input and output samples of the system). The neural network modifies its internal structure and the weights of the connections between its artificial neurons to make the mapping of the relation input/output that represent the behaviour of the modelled system.

2.2.1.1 Structure of Neural Networks

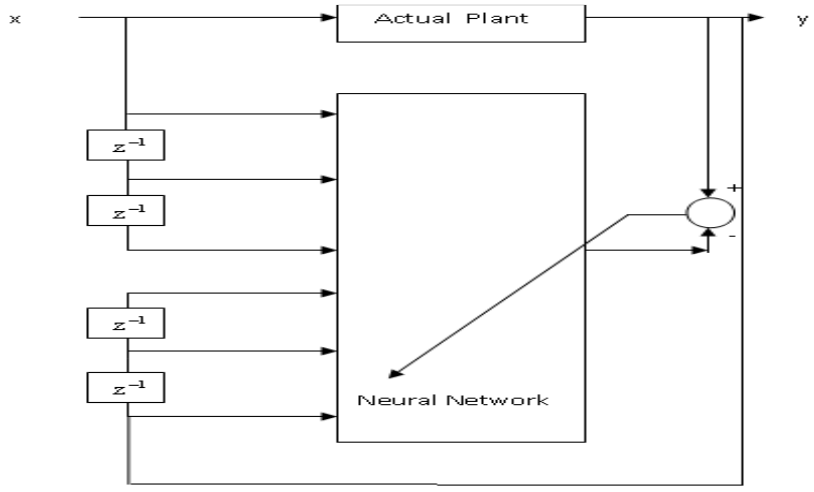
Two classes of neural networks which have received considerable attention in the area of AI in recent years are

- **Feed forward Multilayer Perceptron;** A feed-forward MLP is one whose topology has no closed paths. Its input nodes are the ones with no arcs to them, and its output nodes have no arcs away from them[13]. When the states of all the input nodes are set, all the other nodes in the network can also set their states as values propagate through the network. The operation of a feedforward network consists of calculating outputs given a set of inputs in this manner. It represents static nonlinear maps. It is proved extremely successful in pattern recognition problems. From a systematic point of view, multilayer perceptron can be a versatile non-linear structure for identification problems.

- **Recurrent Network ;** Sometimes it is necessary to introduce a time delay Δ into the structure in order to model the finite time that is required for an input series to move through a physical process[14]. Indeed the length of this delay can be a parameter that is adjusted to minimize the residual error in the neural network model. Also, since most dynamic systems have temporal behaviour, time delayed versions of the output signal are needed to properly model the system. The feedback loops can be both local and external.
  The local loops redirect the output of each neuron to itself or to a lower layer neuron within the network. The external feedback normally connects the output of the structure to input of the network**.**
  This external feedback can be implemented in two different ways. In the first approach, it comes from neural network output (figure 2.9a). Unfortunately, this recurrent network can easily become unstable due to the feedback loop between its output and input, and there is no guarantee that the output that the output will converge to a stable configuration[14]. This can be solved in the second approach, in which, the feedback is sourced from the actual plant, not the NN output, as illustrated in figure 2.9b.



a)



b)

Fig 2.9– a) NN with time-delayed direct inputs and time-delayed recurrent outputs from the modelled system.
b) NN with time-delayed direct inputs and time-delayed recurrent outputs from the Actual Plant.

The advantages of the neural networks are:

- learning capacity;
- generalization capacity;
- robustness in relation to disturbances.

There are, however critics who point out the disadvantages of using neural networks.

- First, the design of the neural network is a very complex procedure that still relies mostly on trial and error. In addition, because the neural network can only produce accurate results if provided with a large volumes of examples in the training phase.
- Impossible interpretation of the functionality; the most often disadvantage of the neural network is the inherent "black-box" nature of its operations. Neural Network although able to generate solution to many problems, but are unable to explain how they arrive at their results.

2.2.1.2 Learning Using Neural Networks

Artificial neural nets have been successfully used for recognizing objects from their feature patterns. The neural networks should be trained prior to the phase of recognition process. The process of training a neural net can be broadly classified into two typical categories, namely;Supervised learning and Unsupervised learning.

- **Supervised Learning:** The supervised learning process requires a trainer that submits both the input and the target patterns for the objects to get recognized. Given such input and output patterns for a number of objects, the task of supervised learning calls for adjustment of network parameters (such as weights and non-linearities), which consistently can satisfy the input-output requirement for the entire object class. Among the supervised learning algorithms, most common are the back-propagation training

- **Unsupervised Learning:** The process of unsupervised learning is required in many recognition problems, where the target pattern is unknown. The unsupervised learning process attempts to generate a unique set of weights for one particular pattern. The objective of unsupervised learning process is to adjust the weights autonomously, until an equilibrium condition is reached when the weights do not change further. The process of unsupervised learning, thus, maps a class of objects to a class of weights. Generally, the weight adaptation process is described by a recursive functional relationship. Depending on the topology of neural nets and their applications, these recursive relations are constructed intuitively. Among the typical class of unsupervised learning Hopfield nets are the most popular ones.

2.2.2 Fuzzy Systems

The fuzzy sets theory was conceived by Lofti Zadeh [16] in 1965 to represent and manipulate data and information that possess non-statistical uncertainty. Fuzzy systems propose a mathematic calculus to translate the subjective human knowledge of the real processes. This is a way to manipulate practical knowledge with some level of uncertainty. The behaviour of such systems is described through a set of fuzzy rules, like:

IF <premise> THEN <consequent> (2.10)

that uses linguistics variables with symbolic terms. Each term represents a fuzzy set. The terms of the input space (typically 5-7 for each linguistic variable) compose the fuzzy partition[11]. Fuzzy modelling is the most important issue in fuzzy theory. The fuzzy modelling is a system description with fuzzy quantities. Fuzzy quantities are expressed in terms of fuzzy numbers or fuzzy sets associated with linguistic labels. Therefore, the relation between input and output variables can be viewed as a set of fuzzy logical rules or fuzzy-set associations. Since functional variables are stored in a distributed rule-based fashion, the value of the function at any point in the input space is derived by aggregating the consequences of fuzzy logical rules. It has been shown that fuzzy systems are capable of approximating any real continuous function to any desired degree of accuracy [25-26].

The fuzzy inference mechanism consists of three stages: in the first stage, the values of the numerical inputs are mapped by a function according to a degree of compatibility of the respective fuzzy sets; this operation can be called fuzzification. In the second stage, the fuzzy system processes the rules in accordance with the firing strengths of the inputs. In the third stage, the resultant fuzzy values are transformed again into numerical values; this operation can be called defuzzification.

2.2.2.1 Identification with Fuzzy modelling

The two usual aspects of identification are: Structure identification and Parameter identification. For a given pre-assigned input candidates, the structure identification of a fuzzy system divide into two parts. Initially, it starts with finding the number of fuzzy rules in a fuzzy model. By structure identification in a ordinary systems theory, what we mean is to find the relations between the inputs and outputs[15]. On the contrary, in a fuzzy model, the structure identification is stated in different way. The number of fuzzy rules in a fuzzy model corresponds to the order in a conventional mode.

Second, identification implies determining how the input space should be partitioned. There are two parts of IF-then rules. The premise part and consequent part. This part of identification deals with premise structure. The premise space of the input variables of fuzzy model is partitioned into several fuzzy subspaces (Fuzzy sets); where the number of rules corresponds to the number of subspaces. These two parts of structure identification are linked together. Therefore, we need a heuristic method to optimized partitioning with some criterion, i.e. output error.

The parameter identification in a fuzzy model includes those in fuzzy sets. The parameter identification and the structure identification cannot be performed separately. However in some approaches, the parameter identification can be separately done subsequent of the structure identification.

The advantages of the fuzzy systems are:

- Capacity to represent inherent uncertainties of the human knowledge with linguistic variables; everything is imprecise if you look closely enough, but more than that, most

things are imprecise even on careful inspection. Fuzzy reasoning builds this understanding into the process rather than tackling it on to the end.

- Simple interaction of the expert of the domain with the engineer designer of the system; In direct contrast to neural networks, which use training data and generate opaque, impenetrable models, fuzzy-logic lets you rely on the experience of people who already understand the system.
- Easy interpretation of the results, because of the natural rules representation; The basis of fuzzy logic is human communication. This observation underpins many of the other statements about fuzzy logic. Because fuzzy logic is built on the structures of qualitative description used in everyday language, fuzzy logic is easy to interpret.
- Easy extension of the base of knowledge through the addition of new rules; with any given system, it is easy to add on more functionality without starting again from scratch.
- Robustness in relation of the possible disturbances in the system.

And its disadvantages are :

- One of the foremost problems of these systems is that they are unable to learn. Suppose that the problem we have has a bulk of instances. In such a context, it would be good to have a system that adapts itself to this dataset. The basic approach is to build a system using the available information and test it against the available datasets. This calls for a lot of work over and over again by the designer to adapt the system to give a decent performance in the scenario given.

- The other problem of these systems is a fixed architecture. The number and type of MFs, their parameters, rules, etc have to be specified beforehand. This needs to be judiciously designed by the designer of the system. This affects the performance as the designer may make a sub-optimal design of the complete system.

2.2.3 Hybrid Schemes

Hybridization of intelligent systems using soft computing techniques has been identified as a promising research field of computational intelligence. The main premise behind combining two

or more soft computing algorithms is to develop a hybrid technique that exploits the synergy between them, leveraging their benefits and overcoming their respective limitations[16]. This has indeed proven quite powerful for a variety of applications, such as: pattern recognition, intelligent control, data mining [6], and classification. Examples of promising hybridization techniques include:

• **Neuro-Fuzzy**: While neural networks and fuzzy logic have added a new dimension to many engineering fields of study, their weaknesses have not been overlooked. Prompted by the weaknesses inherent in the two technologies and their complementary strengths, researchers have looked at ways of combining neural networks and fuzzy logic. The NF model is a hybrid framework that is obtained by combining the concepts of fuzzy logic and neural networking into a unified platform. A hybrid neuro-fuzzy system is a fuzzy system that uses a learning algorithm based on gradients or inspired by the neural networks theory (heuristical learning strategies) to determine its parameters (fuzzy sets and fuzzy rules) through the patterns processing (input and output)[17]. Hybrid techniques in this category combine ANN and FL in novel ways for modelling, control or for classification applications. This system can be totally created from input output data or initialised with the à prior knowledge in the same way of fuzzy rules. The resultant system by fusing fuzzy systems and neural networks has as advantages of learning through patterns and the easy interpretation of its functionality. However, there remain some problems to be solved, for instance, how to automatically partition the input space for each variables, how many fuzzy rules are really needed for properly approximating the unknown nonlinear systems. Also, as is well known, the curse-of-dimensionality is an unsolved problem in the field.

• **Neural Genetic algorithm**: Genetic algorithms are a family of computational models inspired by the way living organisms adapt to the harsh realities of life in a hostile world, i.e., by evolution and inheritance. The algorithm imitates the process of evolution of populations by selecting only fit individuals for reproduction. Therefore, a genetic algorithm is an optimum search-technique based on the concepts of natural selection and survival of the “fittest” [18]. It works with a fixed-size population of possible solutions of a problem, called individuals, which are evolving in time. An evolutionary algorithm (EA) maintains a population of candidate solutions for the problem at hand, and makes it evolve by iteratively applying a (usually quite small) set of stochastic operators, known as mutation, recombination, and selection. Evolutionary artificial neural networks

(EANN's) refer to a special class of artificial neural networks (ANN's) in which evolution is another fundamental form of adaptation in addition to learning[19]. GAs have been used in synthesizing and tuning ANNs in many ways. One way is to use the GAs to evolve the network topology before Back Propagation is used to tune the network. GAs have also replaced Back Propagation as a technique for finding the optimal weight. Another application of GAs in ANNs has been making the reward function adaptive by using GAs to evolve the reward function. Many combinations of ANNs with GAs can be considered a continuation of the earlier discussions of the hybrid methods to exploit the advantages and overcome the disadvantages of GAs and ANNs. For example, ANNs using Back Propagation are able to exploit their local knowledge. Hence, they are faster to converge than GAs, but this is at the expense of risking the ANN getting stuck in the local search, which happens frequently and causes the whole ANN to get stuck in local minima. On the other hand, even though GAs are not exposed to this problem, but they are slower due to their global search characteristic. In the neuro-genetic algorithm only a specific subset of NN architectures, named MLP, is considered for neural encoding[20]. While the great advantage of GAs is the fact that they find a solution without utilizing derivatives, but the following drawbacks are undeniable :

- Need much more function evaluation comparing to linearized models.
- No guarantee to convergence even to local minimum

• **Fuzzy Genetic algorithms**: Genetic Algorithms (GAs) and FL have also been combined to generate the hybrid field of Fuzzy-Genetic Algorithms (FGAs). Similar to the case of Fuzzy Neural Networks, the fusion has gone also two ways. GAs controlled by FL as well as FL controllers tuned by GAs. FL has been used to manage the tools of GAs such as population size and selection pressure during the transition between these two phases [21, 22]. GAs resource managed by FL resulted in adaptive algorithms, which significantly improved its efficiency and speed of convergence[23]. Also, research has been active on the use of GAs to tune FL controllers. An exhaustive survey of the research in this area was indicated in[24] . In the latter case, a GA-Fuzzy system is basically a fuzzy system augmented by a learning process based on a genetic algorithm(GA) [25]. Recent results of the hybridization of FL and GA have been reported in variety of applications such as fuzzy logic based controllers. From the optimisation point of view, the task of finding an appropriate fuzzy knowledge base (KB) for particular problem, is equivalent

to parameterise the fuzzy KB (rules and membership functions), and to find those parameter values that are optimal with respect to the design criteria. The KB parameters constitute the optimisation space, which is transformed into a suitable genetic representation on which the search process operates. Based on mentioned fact, the general algorithm consists of three steps: First, they defined the initial rule base using intuitive heuristics. Second, they used GAs to generate a better rule base. Finally they use GAs to tune membership functions[26].

• **Wavelet Neural Networks :** Mixing the wavelet transform theory with the basic concept of neural networks, a new mapping network called wavelet neural network or wavenets (WNN) is proposed as an alternative to feedforward neural networks for approximating arbitrary nonlinear functions[27]. Kreinovich proves in [28] that if we use a special type of neurons (wavelet neurons), then the resulting neural networks are *optimal* approximators. The network structures applied for representation are determined by using wavelet analysis. The parameter of the initialized network is updated using the well-known steepest gradient-descent method of optimization. Each hidden unit has a square window in the time-frequency plane. The optimization rule is only applied to the hidden units where the selected point falls into their windows. Therefore, the learning cost can be reduced. Literature reveals that there are two major approaches to design wavelet neural networks i.e.

- In the first approach, the wavelet and neural network processing parts are preformed individually. In this format, the wavelet decomposition is a pre-processing step before feeding the input into Neural Network. The input signal first decomposed using some wavelet basis.

- The second approach combines the two theories; which means that the wavelet is implemented inside the neurons. In this case the, two possible structures can be assumed

  I) The one with fixed wavelet bases, where the dilation and translation parameters of wavelet basis are fixed, and only the output layer weights are adjustable. For WNNs with fixed wavelets, the main problem is the selection of wavelet bases/frames. The wavelet bases have to be selected appropriately since the choice of the wavelet basis can be critical to approximation performance. It is well known that by using regularly truncated wavelet frames, the number of wavelet

candidates would drastically increase with the dimension. Therefore, constructing and storing wavelet bases/frames for large dimension problems are of prohibitive cost.

II) The other, is the type which Translations and Dilations of the *wavelons* along with weights are optimized during the training.

The scope of this research is focussed on the latter type and, hereafter, by WNN will refer to the second one.

• **FWNN :** By utilizing two important properties, viz., multi-resolution and compression of wavelets along with Fuzzy Logic and neural networks FWNNs are proposed[29]. The local details of non-stationary signals can be analyzed by wavelet transforms whereas Fuzzy logic allows us to reducing the complexity of the data and to deal with uncertainty. The approximation accuracy of the plant can be improved by the self-learning capabilities of neural networks. Their combination allows us to develop a system with fast learning capability that can describe nonlinear systems that a characterized with uncertainties. In FWNN, each fuzzy rule corresponds to a sub-WNN consisting of wavelets with a specified dilation value and the rule which determines the effect of each sub-WNN on the output. Due to the relative youth of this field of study, a consensus on the best way to utilize their individual strengths and compensate for their individual shortcomings has not yet been established. Consequently, research into Fuzzy-Wavelet systems is targeting many directions.

2.3 Problem Description and Proposed Methodology

Traditional mathematical system modelling relies heavily on accuracy of the mathematical model and this accuracy needs as many as parameters to be involved. Treating in this way either is impossible for complicated systems or even if is viable, it brings us with a very sophisticated mathematical expression.

The soft computing techniques so far introduced, alleviate the problem to a higher degree. However, there were some intrinsic problems in each of them. Hybrid schemes by pure SC techniques as described were the tools to overcome some of the short comings, but as mentioned

still there were plenty rooms for improvement. Despite the fact that embedding a signal processing technique such as wavelet was a way to generalize soft computing techniques to a wider spectrum of problems, but the drawbacks such as slow learning algorithms for NNs and curse of dimensionality for fuzzy systems are still remained untouched.

The aim of this research is proposing new versions of WNNs and ultimately FWNN, enabled with some clustering techniques and also hybrid learning algorithms combination of Expectation Maximization, Recursive Least Square and Extended Kalman Filter to target the aforementioned problems.

Chapter 3
Computational Intelligence Methodologies

3.1 Artificial Neural Network (ANN)

Neural Networks originated in an attempt to replicate the processing patterns of the human brain. Humans are capable of dealing with vast quantities of information very quickly yet the structure of the brains individual components is very simple. A single biological neuron is not in itself intelligent. Yet the hundred billion or so of interconnected neurons coupled with their supporting cells in each of our heads are capable of representing not just the knowledge each of us posses, but the personalities and unique problem solving capabilities that make humans individual. An NN is an information processing paradigm that is inspired by the way biological nervous systems, such as the brain, process information. The key element of this paradigm is the novel structure of the information processing system.

Initial work by McCulloch and Pitts in 1943 presented simplified artificial neurons that were shown to have basic logical properties. In 1957 Frank Rossenblatt put forward the concept of the Perceptron [30, 31] and B. Widrow (Adaline) developed the first training algorithm .

Neural Networks(NN) have been widely used in a broad range of applications. These applications include pattern recognition, function approximation optimization, simulation and estimation among many other application areas. Nowadays, NNs have been trained to solve complex problems that are difficult by conventional approaches [32]. NNs overcome the limitations of the conventional approaches by extracting the desired information by using the input data. A NN does not need such a specific equation form. Instead, it needs sufficient input–output data. Also, it can continuously be re-trained, so that it can conveniently adapt to new data.

In its simple form, each single perceptron (neuron) is connected to other neurons of a previous layer through adaptable synaptic weights. This model is based on the concept of the perceptron originated by Frank Rosenblatt in 1957 [33, 34]. Figure 3.1 presents how information is processed through a single node. The node receives weighted activation from other nodes through its incoming connections. First, these are added up (summation). The result is then passed through an activation function; the outcome is the activation of the node. For each of the outgoing connections, this activation value is multiplied by the specific weight and transferred to the next node.

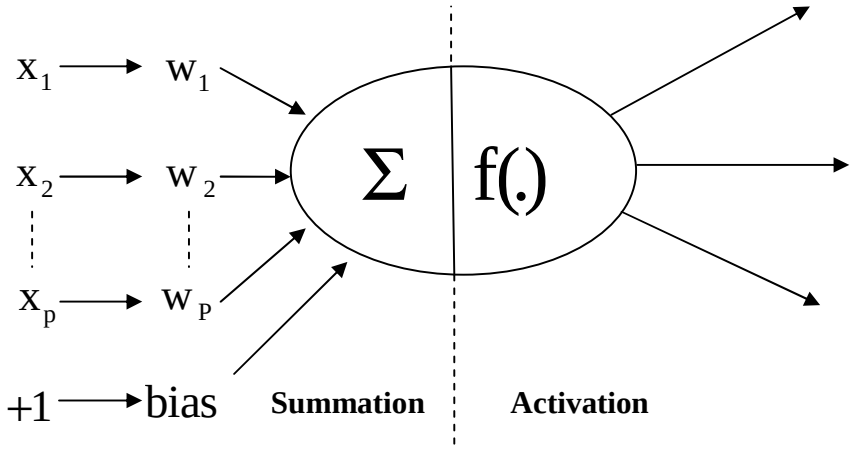


Fig 3.1 –A single perceptron

Knowledge is usually stored as a set of connection weights (presumably corresponding to synapse efficacy in biological neural systems).

3.1.1 Multi Layer Perceptron (MLP)

This is probably the most widely known and used Artificial Neural Network (ANN) structure. MLP networks consist of layers of perceptrons with each layer connected to each of the layers in the previous layer. The weights connecting each of the perceptrons are considered the parameters of the network. The network usually consists of an input layer, some hidden layers and an output layer. Its structure is illustrated with one Hidden Layer in figure 3.2.

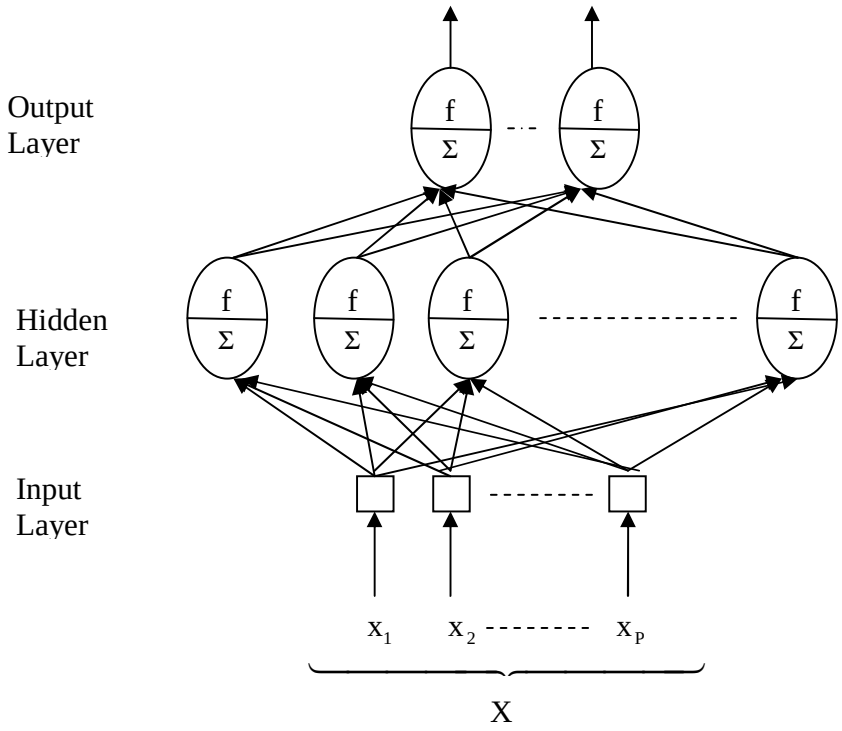


Fig 3.2- MLP Structure

The complexity and the representational capabilities of the MLPs are defined by the number of neurons in each layer. The model information is contained in the weights connecting the neurons in each layer. The optimisation of these weights represents the learning or training process and it's a non-linear optimisation process working from initial parameter values to a set which can model the function in question.

An MLP is characterised by:

- Its pattern of connections between the neurons called structure (architecture). The architecture of a network refers to the number of neurons, their arrangement and connectivity. It also covers the arrangement of the neurons into layers. Many neural nets have an input layer, in which, the function within a unit is equal to an external input signal. The net depicted in figure 3.2 consists of input units, output units and one hidden(middle) layer. Typically there is layer of weighs between two adjacent levels of units.

- The method of determining the weights on the connections and other transfer function parameters (if any) called the training (learning) algorithm. In addition to architecture, the method of setting the values of the weights (training) is an important issue. The procedure used to carry out the learning process . The training algorithm is applied to the network to in order to obtain a desired performance. The type of training is determined by the way in which the adjustment of the free parameters in the neural network takes place. Supervised and Unsupervised training are the most common methods of training. In supervised one, the training is accomplished by pre-setting a sequence of training inputs with a corresponding target output vector, whereas in Unsupervised, no target(output) vector specified and the MLP modifies the weights so that the most similar vectors are assigned to the same cluster unit.
- The activation function. The basic operation of an artificial neuron involves summing its weighted input signal and applies them on an activation function. The perceptron neuron model receives information in the form of a set of numerical input signals. This information is then integrated with a set of free parameters to produce a message in the form of a single numerical output signal.

In following lines, we adopt a compact matrix–vector notation of the network[35] description in order to express the dynamics of Neural Network. Let *P* and *N* stand for the number of input nodes and the number of hidden layer neurons, respectively.

Denote by X, W and V for the inputs, the gains of output and the weights from input layer to hidden layer neurons, the following

$$
\begin{aligned}
&X = [x_1 \quad x_2 \dots x_p]^T \in \mathbb{R}^P \\
&W = [w_1 \quad w_2 \cdots w_N]^T \in \mathbb{R}^N \\
&V = \begin{bmatrix} v_{11} & v_{12} & \cdots & v_{1P+1} \\ v_{21} & v_{22} & \cdots & v_{2P+1} \\ \vdots & \vdots & \vdots & \vdots \\ v_{N1} & v_{N2} & \cdots & v_{NP+1} \end{bmatrix} \in \mathbb{R}^{N\times(P+1)}
\end{aligned}
\tag{3.1}
$$

The activation function for hidden neurons is normally a symmetric '*S*' shape function with well-defined first derivative such as the hyperbolic tangent or the binary sigmoid defined as :

$$f(x) = \frac{1}{1+e^{-x}} \tag{3.2}$$

As can it be seen from figure 3.3, the shapes of the function provide a graded output between 1 and 0 or -1 and 1. This allows smooth interpolation between data points.

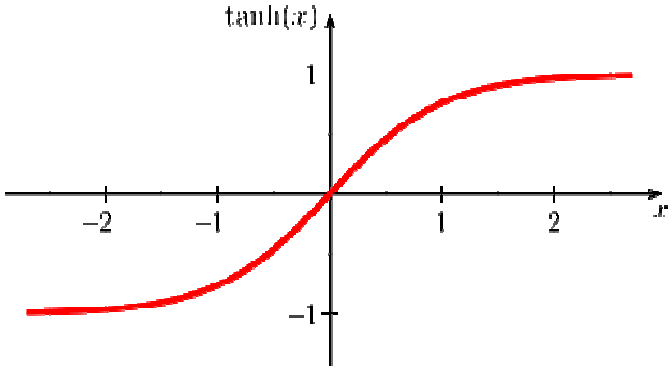


Fig 3.3 –Tangent Hyperbolic

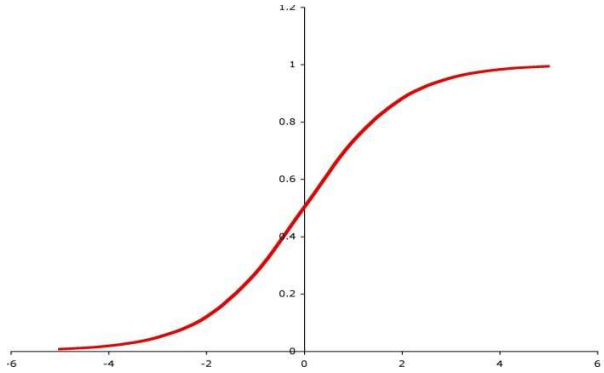


Fig 3.4 - Sigmoid Function

One single neuron makes the simple operation of a weighted sum of the incoming signals and a bias term (b), fed through an activation function (f) and resulting the output value of the neuron. A network with one hidden layer is described in element-wise notation as

$$\tilde{y}^t = \sum_{j=1}^{N} f\left(\sum_{p=1}^{P} v_{jp} x_p^t + b_j\right) w_j = \sum_{j=1}^{N} f\left(net_j^t\right) w_j \tag{3.3}$$

Here x is the input with dimension P and y the output of the network and N is the number of hidden layer neurons. The interconnection matrices are W and V for output layer and hidden layer respectively. This network is applied in a variety of problem domains. It does however suffer from a number of well-established problems. The massive interdependency of the structure means that the model is global in nature. The result of this is that it is often hard to establish exactly what information the network has learnt as the model is not readily interpreted due to the massive interdependency between each of the artificial neurons. The sequential nature of the training methods such as Back-Propagation (BP) means that information can be unlearned as patterns early in the training data can be overwritten by patterns latter in the series. It can also be difficult to establish the number of artificial neurons needed to accurately represent the training data and can often only be done through trial and error which is very time-consuming. There are a number of algorithms available for the learning of the parameters with perhaps the most well known being the Back-Propagation (BP) algorithm. This method uses the partial derivative of the mean squared error between the system output and the desired output of a given training sample to improve the fit of the parameters to the data. This is a gradient descent method and as such is susceptible to problems of identifying a local optimum parameter set rather than a global optimum. In addition to this the training is often slow requiring the training set to be presented to the network a large amount of times in order to find a minimum value. Techniques such as using momentum terms help to overcome the problems of local minima and there are a number of advanced BP algorithms including factors such as the second derivative of the error function in order to speed training. Other learning algorithms include those related to the calculation of the second derivative such as conjugate gradient and quasi Newton schemes. These concepts are computationally expensive [36].

3.1.2 Backpropagation Algorithm

Given a training set of input/output data, the original rule for training MLP is the backpropagation (BP) algorithm [37]. It is an iterative process based on an error signal obtained from measuring the output signal from each neuron in the output layer. The weightings to a particular neuron are modified using new data from training. It employs the quadratic or sum of squared errors metric given by

$$E_t = \frac{1}{2}\sum_{pp=1}^{Q}(y^d{}_{t,pp} - \tilde{y}_{t,pp})^T(y^d{}_{t,pp} - \tilde{y}_{t,pp}) \tag{3.4}$$

In which $y^{d}_{t,pp}$ is the desired value of pp^{th} output and $\tilde{y}_{t,pp}$ is the observed output for the t^{th} training sample, the error indicates how far the desired output is far from its observed value. Let θ be a vector formed by all the network weights (V and w) and $\partial\theta$ be the gradient of E at $\theta = \theta(t)$, with t = 1; 2; 3; … ; M ,. Where t is the pattern counter, The BP algorithm is illustrated through the following steps:

For each input-output pattern do begin

1. Apply the input vector X
2. Compute the output at the last layer through forward calculation.
   Each output unit receives a target pattern corresponding to input training pattern
   We define the instantaneous value of the error energy for t'th pattern is given by eq (3.4).
3. Compute δs at the last layer and propagate it to the previous layer by using eq (3.7).
4. Adjust weights of each neuron by using expression (3.6).
5. Repeat from step 1 until the error at the last layer is within a desired margin.

End For

The adaptation of the weights for all training instances, following the above steps, is called a learning epoch. A number of learning epochs are required for the training of the network. Generally a performance criterion is used to terminate the algorithm. For instance, suppose we compute the square norm of the output error vector for each pattern and want to minimize the sum. So, the algorithm will be continued until the sum is below a given margin.

The error of a given output node, which is used for propagation to the previous layer, is designated by δ, which is given by the following expression

$$\delta = (y^{d}_{t} - \tilde{y}_{t}) f'(.) \tag{3.5}$$

The weight adaptation is described by the following expressions

$$\begin{aligned} \Delta w_{q,q',\ell} &= \eta.\delta_{q'\ell}.o_{q\ell'} \\ w_{q,q',\ell}(t+1) &= \Delta w_{q,q',\ell} + w_{q,q',\ell}(t) \end{aligned} \tag{3.6}$$

$w_{q,q',\ell}(t+1)$ is the weight from neuron q to neuron q', at t'th step, where q' lies in the layer ℓ and neuron q in $(\ell - 1)$'th layer counted from the input layer.

$\delta_{q'\ell}$ is the error generated at neuron q', laying in layer ℓ.

$o_{q\ell'}$ is the output of neuron q, positioned at layer ℓ'

For generating error at neuron q, lying in layer ℓ', we use the following expression

$$\delta_{q,\ell'} = O_{q,\ell'}(1-O_{q,\ell'})(\sum_{q'} \delta_{q',\ell} w_{q,q',\ell}) \tag{3.7}$$

η in eq (3.6) is the Learning Rate and δ is the error signal for unit j.

This is a very simple means of updating the parameters but suffers from a number of problems. If the learning rate is large then the network will train initially very quickly. However it will be prone to overshooting the optimum parameter measures and zigzag about the desired values. If the learning rate is too small then the network will take a long time to train as it is taking very small steps at each stage.

3.1.3 Momentum Effect

The problems with the fixed learning rate forms of BP lead to further modifications of the original BP algorithm. Figure 3.5 shows the effects that different learning rates have on the convergence of parameters

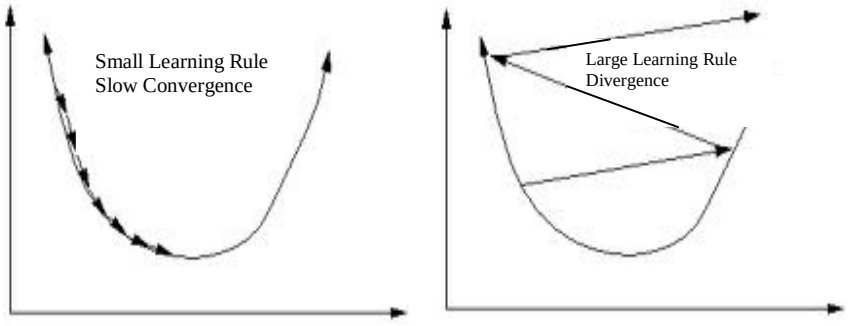


Fig 3.5 – Effect of various learning rates on convergence of the weights

The problem lies around the fact that, although the calculated gradient identifies the direction in which the parameter optimisation must be carried out, it does not identify amount by which each parameter needs to be changed. Moreover, if the error function contains many local minima, the

network might get trapped in some local minimum or get stuck in a very flat plateau[38]. As a result a number of techniques were proposed, attempting to modify the learning rate at each parameter update, so that to reduce the chances of over shoot and to increase the speed of convergence. The simplest method of doing this is to include a momentum term. Applied to backpropagation, the concept of momentum ζ is that previous changes in the weights should influence the current direction of movement in weight space. This concept is implemented by the revised weight-update rule:

$$\Delta w_{ij}(t+1) = \eta.\delta_{ij}.o_{ij} + \zeta \Delta w_{ij}(t) \qquad (3.7)$$

Once the weights start moving in a particular direction in weight space, they tend to continue moving in that direction. Imagine a ball rolling down a hill that gets stuck in a depression half way down the hill. If the ball has enough momentum, it will be able to roll through the depression and continue down the hill. Similarly, If the gradient has changed direction, then the momentum has the effect of dampening the change to the parameters. In this way the zigzagging effect is reduced.

3.2 Elman Neural Network

The recurrent networks have state variables for the delays and incorporate temporal aspects better than Feed-forward neural networks[39]. The Elman Network proposed in 1990 by J.L. Elman is one of the simplest among the available recurrent networks.

In contrast to the feed-forward loop, the Elman Networks are a form of recurrent Neural Networks which the back-forward loop employs copy layer which is sensitive to the history of input data. At each time step, the values of the hidden layer units are copied to the state layer and this information can be stored for future use. This means that the function learnt by the network can be based on the current inputs plus a record of the previous state(s) and outputs of the network. The feedback idea is a convenient way to accumulate previous knowledge as “experiences” and perform future predictions based on these “experiences”.

However, although the Elman neural network has found various applications in speech recognition and time series prediction, its training and converge speed are usually very slow and not suitable for some critical applications. Correlative study shows that The dynamic memory property developed by Elman has been proved to be effective for modelling linear systems not higher than

the first order[40] with standard back-propagation learning algorithm and more suitable for time series.

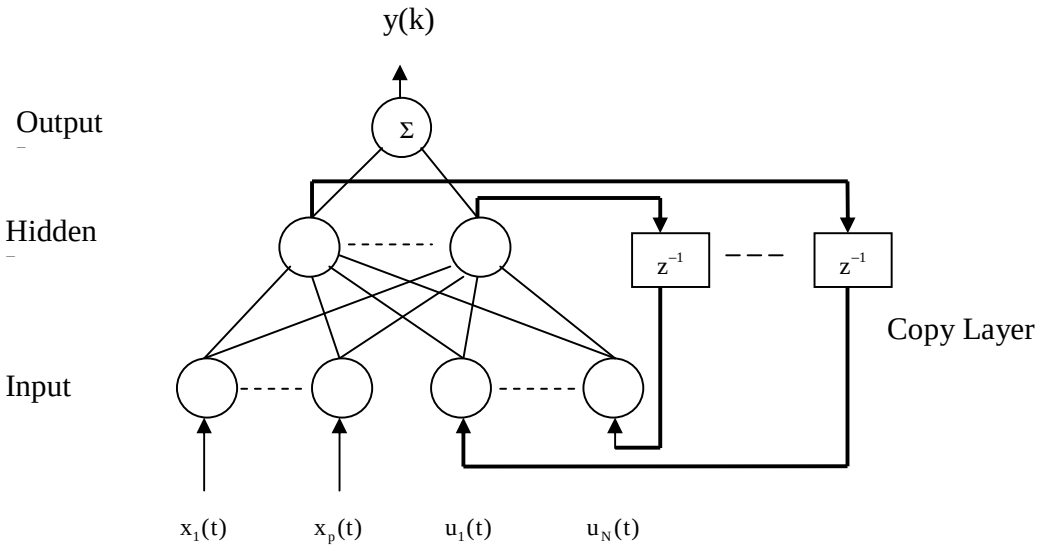


Fig 3.6 – Elman Network structure

The basic structure of Elam network is illustrated in figure 3.6. It comprises four layers namely input layer, hidden layer, output layer and copy layer. Tuneable weights exist between two neighbouring layers. For hidden and copy layer the number of nodes is an adjustable parameter, and the optimal number is acquired through simulations[41]. The inputs of the network are $x(t)\in R^{p}, u(t)\in R^{p'}, \varphi(t)\in R^{N}$, and then the outputs in each layer can be given by

$$\tilde{y}(t) = \sum_{j=1}^{N} w_j * \varphi_j(t)$$

$$\varphi_j(t) = f\left(\sum_{p=1}^{P} v_{pj}(t) * x_p(t) + \sum_{p'=1}^{N} v'_{p'j}(t) * u_{p'}(t)\right) \qquad (3.8)$$

$$u_{p'}(t) = \varphi_j(t-1)$$

Where,

N : The total number of hidden layer nodes
v_{pj} : The weight connection input node to hidden layer
$v'_{p'j}$: The weight connect copy node to hidden node
w_j : The weight connects hidden node to output node

f(.) : The non-linear function of hidden layer

3.3 Radial Basis Functions (RBF)

Radial basis functions, emerged as a variant of Neural Networks, were first introduced by Powell[42] in 1980's to solve the real multivariate interpolation problem. The MLP neuron bisects the information space along a single linear line. As a result each neuron covers the entire information space within its layer. As opposed to an '*S*' shaped function the RBF neuron uses a bell shaped activation function as shown in figure 3.7. One major difference from MLP is that RBFs utilise a local learning strategy vs. MLP's global learning, thus resulting a higher rate of accuracy and faster training times. Such a system consists of three layers (input, hidden, output)
In the RBF neuron the parameters define the centre point of the neuron and the size and shape of the area covered by the activation function. In RBF there is a built-in distance criterion with respect to a centre. This means a graded output is given from 0 at the edges of the area covered by the function to 1 at the functions centre.
Learning is equivalent to finding a multidimensional function that provides a best fit to the training data, with the criterion for "best fit" being evaluated by means of a cost function usually assumed to be mean squared error as depicted . RBFs are embedded in a two layer neural network where each hidden unit implements a radial activated function. The non-linearity within an RBF network can be chosen from a few typical non-linear functions. Gaussian function is the most typical one

$$\varphi(x) = \exp(-x^2 / \sigma) \tag{3.9}$$

The parameter σ is called unit width and is determined using the heuristic rule "*global first nearest-neighbour*"[43] . All the widths in the network are fixed to the same value σ and this result in a simpler training strategy. The activation of a neuron in the output layer is determined by a linear combination of the fixed nonlinear basis functions, *i.e*

$$Y(x) = \sum_{i=1}^{C} w_i \varphi_i(x) \tag{3.10}$$

where $\varphi_i(x) = \varphi(\|x - c_i\|)$ and w_i are the adjustable weights that link the output nodes with the appropriate hidden neurons. These weights in the output layer can then be learnt using the least-

squares method. This research study adopts a systematic approach to the problem of centre selection. Because a fixed centre corresponds to a given regressor in a linear regression model, the selection of RBF centres can be regarded as a problem of subset selection. The orthogonal least squares (OLS) method can be employed as a forward selection procedure that constructs RBF networks in a rational way.

The output units implement a weighted sum of hidden unit outputs.

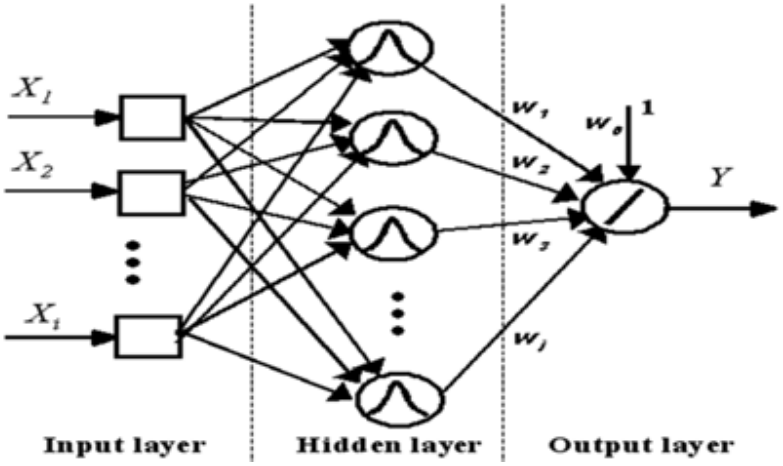


Fig 3.7– RBF network with Gaussian activation

The use of radial activation functions provides a nonlinear method of interpolating between numbers of different regions in the information space. RBF networks train rapidly, usually orders of magnitude faster than MLP, while exhibiting none of its training pathologies such as local minima problems[44]. In practice the centres are normally chosen from the data points. The key question is that how to select centres appropriately from dataset.

3.3.1 Orthogonal Least Squares

The most popular RBF training algorithm is the Orthogonal Least Squares (OLS). This method treats the RBF network as a special case of the linear regression model. It creates a series of regression vectors from the input data and then uses the Gram-Schmidt algorithm to build an orthogonal set of basis vectors from this which spans the same space[45]. Utilising each of the input vectors as the mean parameter of a RBF activation function provides a perfect mapping between *x* and *y*. OLS is an iterative technique that selects the new centres so that the increase in

variance of the output is maximised. The algorithm allows the selection of the centres one by one in a rational procedure, each selected centre maximises the increment to the explained variance of the desired output. Thus the algorithm manages to reduce the size of the network without significantly degrading its performance.

It views the RBF network as a form of the linear regression model with each column vector φ being a regression vector or regressor.

$$\begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix} = \begin{bmatrix} \varphi_1(x_1) & \cdots & \varphi_1(x_N) \\ \vdots & \ddots & \vdots \\ \varphi_C(x_1) & \cdots & \varphi_C(x_N) \end{bmatrix} \begin{bmatrix} w_1 \\ \vdots \\ w_C \end{bmatrix} + \begin{bmatrix} \varepsilon_1 \\ \vdots \\ \varepsilon_C \end{bmatrix}$$

$$y_t = \sum_{i=1}^{C} \varphi_i(x_t) w_i + \varepsilon_t \tag{3.11}$$

where y_t is the desired output and is also called the dependent variable, the w_i are the parameters, and $\varphi_i(x)$ known as the regressors which are some fixed functions of x_t:

$$\varphi_i(x_t) = \varphi(\|x_t - c_i\|) \tag{3.12}$$

the error signal ε_t is assumed to be uncorrelated with the regressors $\varphi_i(x_t)$. The problem of how to select a suitable set of RBF centers from the data set can be regarded as an example of how to select a subset of significant regressors from a given candidate set. An efficient learning procedure for selecting a subset model can readily be derived based on the OLS method. Rewrite eq (3.11) into the matrix form as

$$Y = \Phi W + E \tag{3.13}$$

Where

$$Y = [y_1 .... y_N]^T \tag{3.14}$$

$$\Phi = [\varphi_1 .... \varphi_C], \ \varphi_i = [\varphi_i(x_1) .... \varphi_i(x_N)]^T, \ 1 \le i \le C$$

$$W = [w_1 .... w_C]^T \tag{3.15}$$

$$E = [\varepsilon_1 ... \varepsilon_N]^T \tag{3.16}$$

Note that number of centers equals to C, since all C data samples are employed as centers to initialize the model. Vectors φ_i form a set of basis vectors, and the linear squares solution $\hat{W}$ satisfies the condition that the square of the projection $\Phi\hat{W}$ is part of the desired output energy that can be counted by the regressors. Because different regressors are generally correlated, it is not clear how an individual regressor contributes to this output energy. The OLS method involves the transformation of the set of φ_i into a set of orthogonal basis vectors, and thus makes it possible to calculate the individual contribution to the desired output energy from each basis vector. The regression matrix Φ can be decomposed into

$$\Phi = RA \tag{3.17}$$

Where A is a $C \times C$ triangular matrix with 1's on the diagonal and 0's below the diagonal, that is,

$$A = \begin{bmatrix} 1 & \alpha_{12} & ... & \alpha_{1C} \\ 0 & 1 & ... & ... \\ ... & ... & ... & \alpha_{C-1C} \\ 0 & ... & 0 & 1 \end{bmatrix} \tag{3.18}$$

and R is an $N \times C$ matrix with orthogonal columns r_i such that

$$R^T R = H \tag{3.19}$$

Where ***H*** is diagonal matrix with elements h_i :

$$h_i = r_i^T r_i = \sum_{t=1}^{N} r_i^t \times r_i^t \quad 1 \le i \le C \tag{3.20}$$

And eq(3.11) can be re-written as

$$y = Rg + E \tag{3.21}$$

the OLS solution $\hat{g}$ is given by

$$\hat{g} = G^{-1} R^T Y$$

or

$$\hat{g}_i = r_i^T y / (r_i^T r_i) \quad 1 \le i \le C \tag{3.22}$$

the quantities $\hat{g}$ and $\hat{W}$ satisfy

$$A\hat{W} = \hat{g} \tag{3.23}$$

The OLS method is to use for subset selection of the candidate RBF centres. In practice, the number of data is often very large and centres are to be chosen as a subset of data set. Due to its linear computational procedure at the output layer, the RBF is shorter in training time algorithm.

This method ensures that each new neuron added reduces the overall error of the system by the maximum amount. Training thereby continues until a predefined accuracy is reached. The main drawback with this method is that it uses a single predefined value for the width of each of the neurons. This is defined before the training process starts and although there are a number of heuristics for this such as nearest neighbour it is often necessary to manually modify it through trial and error which is not guaranteed to find an optimal result [46]. This use of a single width parameter for the entire network introduces severe problems. The assumption that the regions with different properties in the input domain can be accurately identified using identically sized local area functions is often erroneous.

3.4 Fuzzy Logic

Fuzzy Logic, is a generalization of Boolean logic[47]. It is seen as a technique based on the key elements that the activity of human brain are not numbers but rather indicators of *fuzzy sets* of which are a generalization of Crisp sets in classical set theory, in which the transition membership and non-membership is gradual between 0 and 1. Having this main characteristic of fuzzy logic, it is easier to deal with imprecise concepts in a well-defined way.

In general, NNs provide a means of learning data through very low level numerical analysis. However the heavily interconnected structure of NN often makes analysis of the information contained within it difficult. Fuzzy Logic (FL) provides a framework by which nonlinear models can be learnt and readily understood by humans. FL incorporates a simple, rule-based IF X AND Y THEN Z approach to a solving modelling problem rather than attempting to model a system mathematically. FL is capable of mimicking this type of behaviour but at very high rate.

Each of the parameters of the model is divided into a number of regions which can be given a linguistic label. Each label is associated with a membership function which produces a membership value for each region between 1 and 0.

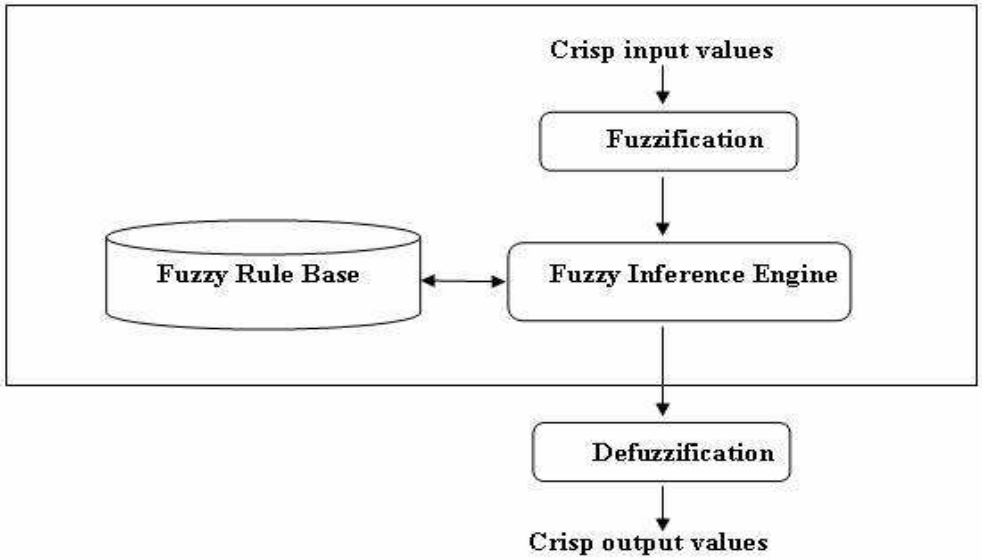


Fig 3.8 - Fuzzy logic system

The typical architecture of a fuzzy system, illustrated in figure 3.8, is comprised of four principal components:-

Fuzzification : Transforms crisp measured data into suitable fuzzy sets. Crisp inputs are exact inputs measured by sensors and passed into the control system for processing.

A fuzzy set is defined in terms of a membership function which is a mapping from the universal set U to the interval [0,1]. Larger values denote higher degrees of set membership. The shape of the membership function should be representative of the variable. However this shape is also restricted by the computing resources available. Complicated shapes require more complex descriptive equations or large lookup tables. These shapes can be diverse but we will usually work with triangles, trapezoidal and Gaussian (see figure 3.9). For this reason we need at least three (for triangles), four (for trapezoids) and two parameters (for Gaussian) to define one MF of one variable.

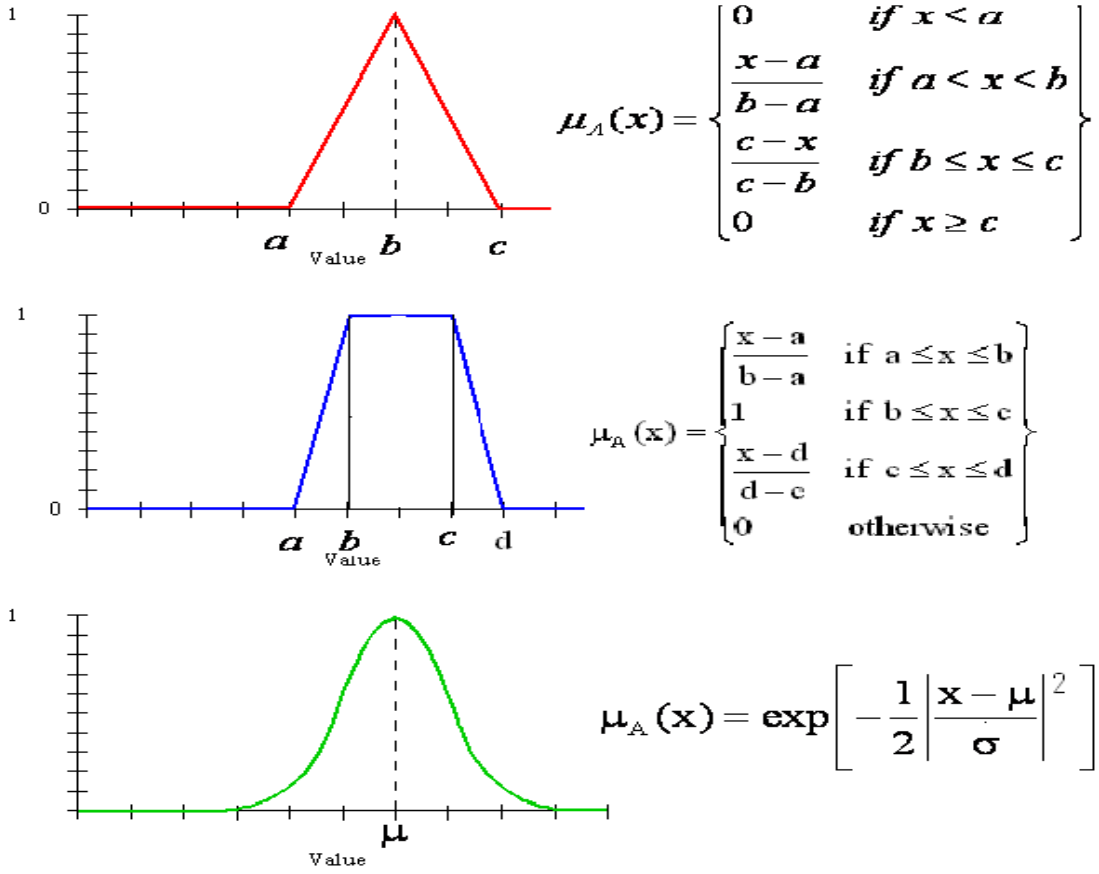


Fig 3.9 – Three types of membership functions with corresponding mathematical expressions a) Triangular b) Trapezoidal c) Gaussian

Fuzzy Rule Base : Stores the observed knowledge of the operation of the process [48]. *Fuzzy rules* are linguistic IF-THEN- constructions that have the general form "IF A THEN B" where A and B are (collections of) propositions containing linguistic variables. A is called the *premise* and B is the *consequence* of the rule. In effect, the use of linguistic variables and fuzzy IF-THEN-rules exploits the tolerance for imprecision and uncertainty. There are several kinds of fuzzy rules used to construct fuzzy models. These fuzzy rules can be classified into the following three types according to their consequent form.

Type I : Fuzzy rules with constant consequent

IF x_1 is $A_{i,1}$ AND x_2 is $A_{i,2}$ AND.... x_P is $A_{i,P}$ **THEN** $f_i = \omega_i$

Type II : Fuzzy rules with linear combination consequent (Takagi Sugeno Kang Model)

IF x_1 is $A_{i,1}$ AND x_2 is $A_{i,2}$ AND.... x_P is $A_{i,P}$ **THEN**

$f_i = \omega_{i0} + \omega_{i1}x_1 + ... + \omega_{i0}x_p$

Type III: Fuzzy rules with fuzzy set consequent (Mamdani Model)

IF x_1 is $A_{i,1}$ AND x_2 is $A_{i,2}$ AND.... x_P is $A_{i,P}$ **THEN** $f_i = B_i$

In the rules X and f denote input and output variables, respectively. The antecedent linguistic terms A_{ip} the consequent linguistic term B_i are parameterized fuzzy sets whose shape can be any of the described above. In type I and II fuzzy rules ω_i denotes a constant value and $\omega_{i0} + \omega_{i1}x_1 + ... + \omega_{i0}x_p$ denotes a linear combination of input variables where are constant coefficients.

Fuzzy rule equations are “AND” rule, which means all the conditions of the IF part must be met simultaneously in order for the result of the THEN part to occur.

- **Inference Engine:** The Inference Engine is the heart of a FL and it has the capability of simulating human decision making by performing approximate reasoning[49]. During the process, it derives a reasonable action with respect to a specific situation based on the given rule base. The membership values measured in Fuzzification step, are aggregated to obtain a single degree of membership or firing value. The most common of which are t-norms such as MIN connective, denoted with $\wedge$. Thus, the degree of fire of rule i for the new input vector, X , is calculated as follows

$$\gamma_i(X) = \underset{p=1}{\overset{P}{\text{AND}}} \tilde{\mu}_i(x_p) \qquad i = 1,..,C$$

$$y_i^* = \gamma_i(X) \times f_i \qquad i = 1,..,C$$

- **Defuzzification :** Lastly, the Defuzzifier converts the fuzzy action to the non-fuzzy action that can be accepted by the real world. This step depends on consequent structure of the rule base, In TSK fuzzy rule base structures, the model output of each rule is aggregated by taking weighted average of the scalar output of each rule

$$\tilde{y} = \frac{\sum_{i=1}^{C} y_i^*}{\sum_{i=1}^{C} \gamma_i(x)} \tag{3.24}$$

The training of a fuzzy system is the process by which the positions and sizes of the fuzzy regions are set and a valid form of consequent is identified. In the simplest form where the consequent is simply a singleton fuzzy variable this can be achieved by means of a table lookup system. This involves creating a set of partitions that cover the whole range of the model data. A rule is then generated for each input-output pair in the model data. Following this the most likely set of rules from the complete set are selected so that there are no conflicts in the rule base. An interesting feature in fuzzy logic is the concept of Adaptive fuzzy logic systems. Because of the arbitrary positioning of the partitions by the developer, it is often necessary to fine-tune the parameters of the rule base. The Back-Propagation (BP) method can also be used for this. Since it works by taking the partial derivative of a mean squared error function with respect to the parameters of a model it can be used to derive update equations gradient descent learning as is done in the Adaptive Fuzzy Logic System (AFLS) [50]. Despite this it is generally accepted that FL is better applied to domains in which it is possible to incorporate expert knowledge and NN are better applied to domains where little is known of the interdependencies of the model requiring low-level numerical analysis to discover them. FL systems also suffer from the so called curse of dimensionality. This term refers to the problem that every possible combination of possible rules must be considered. As a result the number of possible rules increases exponentially with the number of model parameters and the number of partitions in each.

3.4.1 TSK Fuzzy modelling

TSK fuzzy model is one of the most outstanding fuzzy models in the literature which are suitable to model a large class of non-linear systems. It consists of number of local linear models; possessing excellent ability to describe uncertain system and to approximate a nonlinear model

with any given accuracy. The basic idea of this method is to decompose the input space into "fuzzy partitions" and to approximate the system in every region by a simple pricewise linear model. The overall fuzzy model is thus considered as a combination of interconnected subsystems with simpler models. Typically, in a TSK model, the employed IF–THEN rules can be viewed as the expansion of pricewise linear partition and they are presented as

$$\begin{aligned} R_i \ &: \text{IF } x_1 \text{ is } A_{i,1} \text{ AND } x_2 \text{ is } A_{i,2} \text{ AND} \ldots . x_P \text{ is } A_{i,P} \\ &\textbf{THEN } f_i = \omega_{i0} + \omega_{i1} x_1 + \ldots + \omega_{ip} x_p \end{aligned} \tag{3.25}$$

The R_i represents the i'th fuzzy inference rule, x_p and A_{ip} are the premise fuzzy variables and fuzzy sets with Gaussian membership functions. The rule consequent indicates linear equations which are linear in the parameters $\omega_{i,p}$ belonging to i'th rule and p'th input variable. The working region of any fuzzy rule is defined by the membership functions of antecedent part. The output of the TSK fuzzy system with C rules is aggregated as weighted sum of fuzzy rule outputs known also as defuzzification.

$$\tilde{y} = \sum_{i=1}^{C} (\omega_{i0} + \omega_{i1} x_1 + \ldots + \omega_{ip} x_p) \gamma_i(X) \tag{3.26}$$

Where $\gamma_i(X)$ is the normalized firing strengths of the rule i and obtained as :

$$\gamma_i(X) = \frac{\prod_{p=1}^{P} \tilde{\mu}_i(x_p)}{\sum_{i=1}^{C} \prod_{p=1}^{P} \tilde{\mu}_i(x_p)} = \frac{\alpha_i(X)}{\sum_{i=1}^{C} \alpha_i(X)} \tag{3.27}$$

With the $\tilde{\mu}_i(x_p)$ is the membership of $A_{i,p}$ with , Gaussian membership.

Where the μ_{jp} denote the centres and σ_{jp} depicts the standard deviation for membership functions associated with rule i. The parameters are obtained by fitting the eq(3.26) to the set of data points by numerical optimization.

3.5 Neuro-Fuzzy Systems

The two soft computing or “intelligent” computing techniques described above are both inherently mathematical but possess strengths and weaknesses. A striking example of particularly effective combination is what has come to known as “Neuro-Fuzzy”. NF systems attempt to incorporate the low-level numerical analysis of the NN with the model transparency of FL [51]. For example, while neural networks are good at recognizing patterns, they are not good at explaining how they reach their decisions. Fuzzy logic systems, which can reason with imprecise information, are good at explaining their decisions but they cannot automatically acquire the rules they use to make those decisions. These limitations have been a central driving force behind the creation of intelligent hybrid systems where two or more techniques are combined in a manner that overcomes the limitations of each other techniques. In theory, neural networks, and fuzzy systems are equivalent in that they are convertible, yet in practice each has its own advantages and disadvantages. As it was noted above, in case of dynamic work environment, the automatic knowledgebase correction is necessary. On the other hand artificial neural networks are successfully used in problems connected to knowledge acquisition using learning by examples with required degree of precision.
There are many different algorithms falling under the banner NF systems. All of them range in complexity and fall to somewhere along the line joining FL systems and NN**.**

3.5.1 Adaptive Neuro Fuzzy Inference System(ANFIS)

ANFIS is a Neuro-Fuzzy model proposed by Jang[52]. ANFIS is an example of a NF system that directly implements the TSK rule system. ANFIS model has a fuzzy inference system in the form of an adaptive network for system identification and a predictive tool that maps a given input space to its corresponding output space based on a representative training data set.
The structure of ANFIS with five layers is shown in figure 3.9. Xs are the inputs for ANFIS. The ANFIS is composed of two parts. The first part is the antecedent part and the second part is the conclusion(consequent) part. These are connected to each other by the fuzzy rules in form of a network. It can be described as a multi-layered neural network .The first layer executes a fuzzification process, the second layer executes the fuzzy AND of the antecedent part of the fuzzy

rules, the third layer normalizes the MFs, the fourth layer executes the conclusion part of the fuzzy rules, and the last layer computes the output of the fuzzy system by summing up the outputs of the fourth layer which is the defuzzification process.

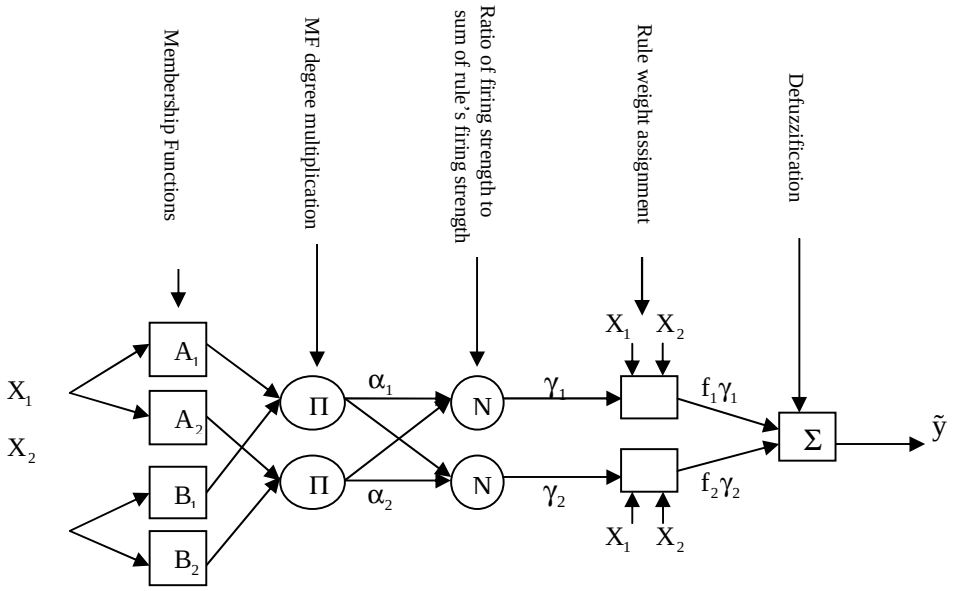


Fig 3.10 -The structure of ANFIS (type III) with two inputs and one output

The feed-forward equations of the ANFIS structure and all the parameters depicted in figure.3.10 are the same as eq(3.25-3.27) for p=1,2 and i=1,2 with two inputs and two membership functions. They are shown as

$$\alpha_i = \tilde{\mu}_{A_i}(x_1) \times \tilde{\mu}_{B_i}(x_2) \qquad i = 1,2$$

$$\gamma_i = \frac{\alpha_i}{\alpha_1 + \alpha_2} \qquad i = 1,2$$

$$\tilde{y} = \frac{\alpha_1 f_1 + \alpha_2 f_2}{\alpha_1 + \alpha_2} = \gamma_1 f_1 + \gamma_2 f_2$$

The ANFIS uses fuzzy MFs in antecedent part for splitting each input dimension; the input space is covered by the overlapped MFs, that is, several local regions can be activated simultaneously by a single input.

Subsequent to the development of ANFIS approach, a number of methods have been proposed for learning rules and for obtaining an optimal set of rules[53]. For instance, Mascioli *et al.*[53, 54] have proposed to use a combination of Min–Max and ANFIS model to determine neuro-fuzzy network and create optimal set of fuzzy rules. Jang and Mizutani[55] have introduced application of Levenberg–Marquardt algorithm, which is essentially a nonlinear least-squares technique, for learning the ANFIS network structure. In another paper, Jang has proposed a scheme for input selection and Kumar and Garg[56] have used Kohonen's map for training. Jang introduced four methods to update the parameters of the ANFIS structure, as listed below according to their computation complexities:

- Gradient Decent only: all parameters are updated by the GD.
- Gradient Decent only and one pass of least square estimation: the least square estimation is applied only once at the very beginning to get the initial values of the conclusion parameters and then the gradient decent takes over to update all parameters.
- Gradient Decent only and least square estimation: this is the Jang's proposed hybrid learning method.
- Sequential least square estimation: using EKF to update all parameters

The performance of the network is indeed very good. Nevertheless, the network suffers general faults identified with all fuzzy systems in terms of the curse of dimensionality; the number of input fuzzy partitions is large and hence the required number of rules and consequence parameters will be very large. The least-squares estimation algorithm cannot be implemented easily because the calculation of very large matrices is required. Thus, the application of the network is limited to some low-dimensional systems.

3.5.2 FALCON

Fuzzy Adaptive Learning Control Network (FALCON) is another general modelling structure that integrates the basic elements of the fuzzy structure into a connectionist model. The input and output nodes represent the system input and outputs in the same manner as general NN structures. The hidden nodes represent the fuzzy basis functions and the rules as can be seen in figure 3.11.

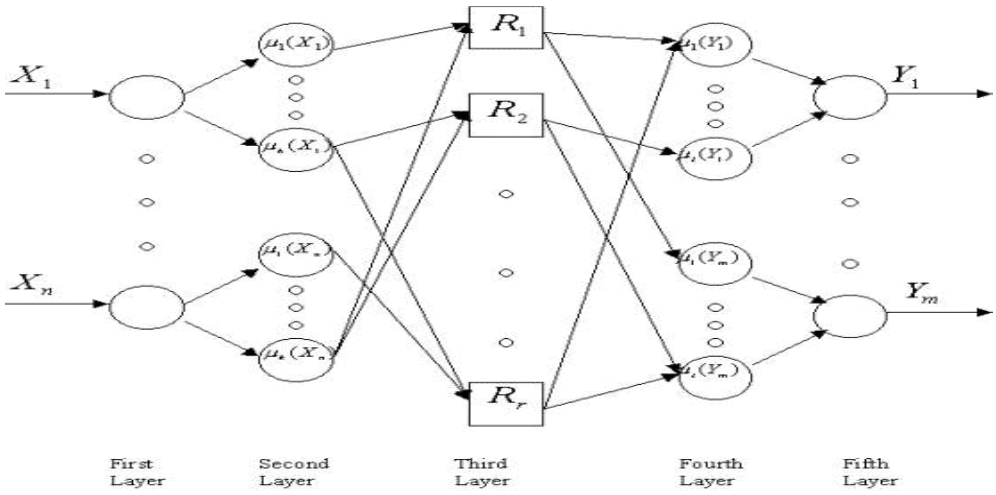


Fig 3.11- Falcon Neuro-Fuzzy architecture

The difference between traditional FL and FALCON is that the connectionist structure permits the use of NN learning techniques. This means that the proper basis functions and parameters can be determined within the connectionist structure and then the rule base extracted from this structure. In theory, this means that the normally black-box trait of the traditional NN architectures can be bypassed. Expert knowledge can be readily incorporated into the network structure as each of the hidden nodes has a transparent action. The training of these networks is often done in a two-phase approach. The first phase is to use statistical clustering techniques to identify initial Fuzzy basis functions. Competitive learning is then used to identify which of the combinations of fuzzy basis neurons represent valid rules. Rule nodes are then merged if they satisfy certain conditions relating similarities between consequents and preconditions. Often a second phase of learning is required to fine-tune the network using a gradient descent technique [31].

3.5.3 NEFCON

NEFCON is a model for neural fuzzy controllers developed by Nauck [57], and it is based on the architecture of the fuzzy perceptron. The learning algorithm for NEFCON is based on a mixture of reinforcement learning with back propagation algorithm. Figure 3.12 shows a NEFCON system with two input variables, one output variable and five rules. The connections in this architecture are weighted with fuzzy sets and rules using the same antecedents (called shared weights), which are represented by the drawn ellipses.

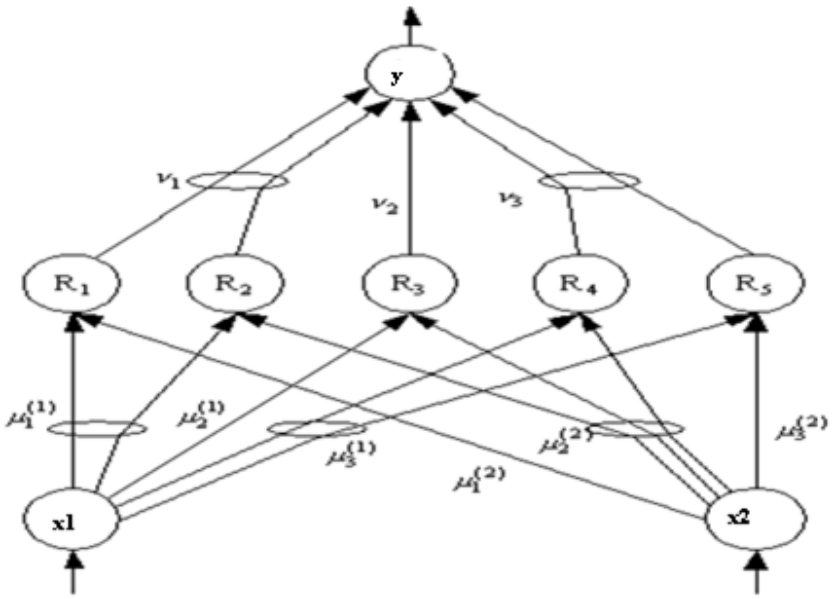


Fig 3.12 – NEFCON architecture

The feed-forward connections between the layers are weighted with fuzzy sets. Each of the layers contains a number of units, where the hidden "rule units'' use a t-norm **as** activation function, and the output unit combines fuzzy sets and applies a defuzzification procedure. The input units just contain the input values and are doing no further computation. The input variables x_1 and x_2 are state variables of a technical system which has to be controlled. NEFCON's output *y* is the control action applied to *S*. The units of the hidden layer represent fuzzy rules[58].

3.6 Adaptive Neuro-Fuzzy Network

An alternative adaptive fuzzy neural network (AFNN) proposed by J.Theocharis [59] has been implemented and validated in the framework of this research study. Its main characteristics are the self-construction ability, parameter learning ability and rule extraction ability. An outline is shown in figure 3.13. In contrary to ordinary ANFIS, the adaptive FNN has a structure-learning mechanism which creates/adjusts the structure of its premise part as training proceeds[60].
In conventional ANFIS structure the number of membership functions and therefore the number of rules is fixed, and increases significantly by increasing the number of membership functions (MFs) and input dimensionality

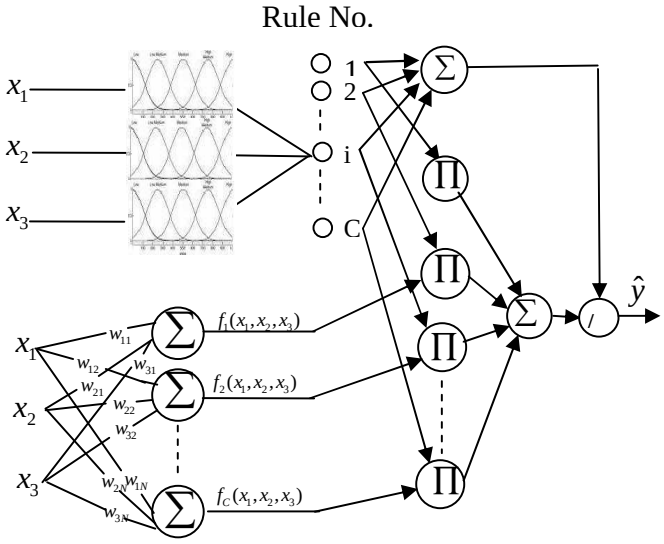


Fig 3.13- Three input and one output Adaptive Neuro-Fuzzy scheme

The fuzzy inference system considered in this network follows Takagi Sugeno's IF-THEN rules in the form of eq(3.25) where as mentioned, *p* is the dimension of input components, *i* depicts the counter of rules and ω_{ip} are the polynomial coefficients, linearly connecting the input variables to the rules' outputs f_i. Finally, A_p^j denote the labels of fuzzy sets outputs. Each linguistic label A_p^j, is associated with a membership function. The membership functions considered here are of Gaussian type as appears in figure 3.8. The degree of fulfilment represents the degree to which each rule participates in the output defined by a

t-norm(*) operator is defined as follows:

$$\gamma_i = \tilde{\mu}_{A_1^i}(x_1) * \ldots * \tilde{\mu}_{A_p^i}(x_p) \tag{3.28}$$

The algebraic product of the membership functions of each premise axis is chosen as the t-norm operator. For each input membership term the below equation is considered

$$\begin{aligned} &\tilde{\mu}_p^{max} = \max\{\tilde{\mu}_{A_p}(x_p) \quad \} \\ &p = 1,\ldots,P. \end{aligned} \tag{3.29}$$

The term $\tilde{\mu}_p^{max}$ represents the max value of the membership belonging to the term set p for x_p.

The structure of proposed FNN comprises of three major modules:

- Premise Part: It calculates not only the rule coordinates through the MFs, but also firing strengths for each rule. The structure of the fuzzy inference system is determined by adjusting this part. Both structure learning and parameter influence premise part.
- Consequent Part: It deals with consequent functions f_i as in eq(3.25) which is linear polynomials of input vector components.
- Defuzzification Part: It involves the defuzzification process. This is performed by combining the outputs of the premise and consequent part and provides the final output of the fuzzy system. The Weighted-Average scheme has been used to produce a fuzzy output $\hat{y}$, for each input vector

The training of AFNN is performed by the following three phases:

- Initial membership functions and corresponding rules creation by using a subset of the training data set. This step is conducted off-line, which is referred to as respective premise/consequent parameter setting .
- Selection of input patterns by computing node outputs in all network layers and the max-membership terms, $\tilde{\mu}_p^{max}$, for each premise axis. By observing the maximum stimulating level

of the term nodes $\tilde{\mu}_p^{max}$ it can be verified whether they are greater or smaller than the prescribed lower membership threshold $\hat{\delta}$. For those where the degree of fulfilment by current MFs is less than a predefined threshold, or in other words the input vector is not adequately spanned by current MFs ($\tilde{\mu}_p^{max} < \hat{\delta}$), network inserts a new membership function in the respective term set and calculates its parameters (mean and deviation) according to eq (3.30)(3.31). Let $A_{p,new}(\mu_{p,new}, \sigma_{p,new})$ denote the new MF

$$\mu_{p,new} = x_p(t+1)$$
$$\sigma_{p,new} = \sigma_{p,nearest} \cdot \frac{\left|\mu_{p,new} - x^+\right|}{\left|x^+ - \mu_{p,nearest}\right|} \qquad (3.30)$$

Where

$$x^{+} = \mu_{i,nearest} + h^{+}.\sigma_{i,nearest} \quad \text{if} \quad \mu_{i,new} > \mu_{i,nearest}$$
$$x^{+} = \mu_{i,nearest} - h^{+}.\sigma_{i,nearest} \quad \text{if} \quad \mu_{i,new} < \mu_{i,nearest} \qquad (3.31)$$
$$h^{+} = \sqrt{[2Ln(\frac{1}{\Omega})]}$$

And Ω exhibits the degree of overlapping between membership functions. In this case when at least one new membership is created, then a new fuzzy rule is created by combining the new memberships and an appropriate set of already existing memberships. The consequent parameters (polynomial weights) of the new rule are initially defined by:

$$\omega_{0i} = y_i^d(t+1) \qquad i = 1,...,C \qquad (3.32)$$

Where $y_i^d(t)$ is the desired output for the t'th input training instance. The remaining weight parameter are set to zero ($\omega_{1i} = \omega_{2i} = \cdots = \omega_{pi} = 0$).

In the case when $\tilde{\mu}_i^{max} > \hat{\delta}$ meaning that the input vector is sufficiently covered by the existing fuzzy membership functions, the term set remains unchanged.

In an alternative scenario where the degree of fulfilment by the membership functions is large enough but the respective fuzzy rule is missing, a fuzzy rule is created by proper permutation of term node coordinates.

Parameter fine tuning performed using the classic back-propagation algorithm. The back-propagation (BP) method is a gradient based algorithm which is usually used to perform parameter learning of both neural networks and fuzzy neural systems. BP is a simple, well established and easily applicable optimization method for this scheme; the learning task is accomplished by minimizing a single objective function, as shown in eq(3.33).

$$e_t = \frac{1}{2}(\tilde{y}(t) - y_d(t))^T(\tilde{y}(t) - y_d(t)) \qquad (3.33)$$

As training proceeds, parameter learning is simultaneously conducted to adjust the network parameters. The final updated equations are:

$$\begin{cases} \mu_{jp}(t+1) = \mu_{jp}(t) - \eta_{\mu} \dfrac{\partial e_t}{\partial \mu_{jp}} + \zeta_{\mu} \dfrac{\partial e_{t-1}}{\partial \mu_{jp}} \\ \dfrac{\partial e_t}{\partial \mu_{jp}} = \sum_{p} [\hat{y}(t) - y^{d}(t)] \times (f_p - \hat{y}(t)) \times \dfrac{\mu_p}{\sum_{i=1}^{C} \mu_i} \times \dfrac{(x_p(t) - \mu_{jp})}{\sigma_{jp}^2} \end{cases} \tag{3.34}$$

$$\begin{cases} \sigma_{jp}(t+1) = \sigma_{jp}(t) - \eta_{\sigma} \dfrac{\partial e_k}{\partial \sigma_{jp}}(t) \\ \dfrac{\partial e_k}{\partial \sigma_{pj}} = \sum_{p} [\tilde{y}(k) - y^{d}(k)] \times (f_p - \tilde{y}(k)) \times \dfrac{\mu_p}{\sum_{i=1}^{C} \mu_i} \times \dfrac{(x_p(k) - \mu_{pj})}{\sigma_{pj}^3} \end{cases} \tag{3.35}$$

$$\begin{cases} \omega_{jp}(t+1) = w_{jp}(t) - \eta_{w} \dfrac{\partial e_k}{\partial \omega_{pj}} \\ \dfrac{\partial e_k}{\partial \omega_{jp}} = [\tilde{y}(k) - y^{d}(k)] \times \dfrac{\mu_j}{\sum_{i=1}^{C} \mu_i} . x_p(k) \\ \dfrac{\partial e_k}{\partial \omega_{0j}} = [\tilde{y}(k) - y^{d}(k)] \times \dfrac{\mu_j}{\sum_{i=1}^{C} \mu_i} \end{cases} \tag{3.36}$$

η_{μ} , η_{σ} and η_{w} representing the mean, deviation and polynomial-weights learning rates respectively and ζ_{μ} is momentum which is used for updating means. Training is carried out on-line on the basis of real-time data.

The predictive capability of above adaptive neuro-fuzzy system was tested on fungus growth in comparison to conventional neural networks approaches. More specifically, the purpose of the present work is (i) to develop an intelligent methodology based on neuro-fuzzy networks to predict the combined effect of temperature, water activity and pH on the maximum specific growth rate of *Monascus ruber van Tieghem*, and (ii) to compare the prediction accuracy of the proposed intelligent scheme and classic neural networks

3.7 Case Study: Fungus Growth Modelling

Growth-predictive models are currently accepted as informative tools that assist rapid and cost-effective assessment of microbial growth for product development, risk assessment, and education purposes [61]. More recently, predictive microbiology has been used to forecast the growth of spoilage micro-organisms in order to study the shelf life of a food product. Fungal spoilage of food commodities causes significant economic losses. Although industrial standards have been greatly improved in the last years, food spoilage by fungi is still a major concern for both food producers and regulatory agencies. Today, there is a need for understanding fungal growth in foods, particularly those factors associated with new manufacturing processing and packaging techniques [62]. Fungal presence in food may adversely affect not only the organoleptic value of the commodity but most importantly its nutritional value by producing toxic metabolites, thus a public health risk is inevitable [63]. Improvement of food quality and safety, demands the development of appropriate tools allowing prediction of fungal growth.

Polynomial models have been widely used in predictive microbiology for the quantitative assessment of the effects of various environmental factors on fungal growth[64]. However, a major disadvantage of these models is that they are developed from linear and quadratic combinations of variables; use of such simplified models may not be justified. Neural networks (NNs) have been deployed in recent years as an alternative to conventional statistical models, due to their ability to describe highly complex and non-linear problems in many fields of science. The NN-based methodologies have been applied in predictive food microbiology[65]. The main characteristics of NNs, such as (i) non-linearity, allowing better fit to the data, (ii) noise-insensitivity, providing more accurate predictions in the presence of uncertain data and measurement errors, enabling application of the model to unknown data make them an interesting tool in an area which is dominated by statistical analysis tools[66]. Several published works indicate that neural network-based models produce better estimation of kinetic parameters of micro-organisms than response surface models. In a recent study, NNs have been compared with response surface models in modelling the growth rate of *L. plantarum* and *E. coli.* It was reported that the NN approach outperformed the statistical models based on its lower standard error of prediction (SEP) term, despite the fact that NN models had higher degree of complexity[67].

Monascus is an ascomycetous fungus traditionally used for the production of food colouring, fermented foods and beverages in southern China, Taiwan, Japan, Thailand, Indonesia and the

Philippines[44]. Members of the genus can commonly survive heat treatments and grow under reduced oxygen levels, resulting in food spoilage. *Monascus ruber* is a widespread ascomycetous fungus in Europe, as it is common in silage and deteriorating grain. One characteristic of the genus is the production of ascospores capable of surviving heat treatment. Subsequently they can grow under reduced- oxygen environment and cause food spoilage. Spoilage may result from the development of a mycelia mat on the surface of the olives, and from a softening of the fruits and changes in the pH of the final product. Temperature, pH and water activity (a_w) are generally regarded as the principal controlling factors during fermentation and subsequent storage of table olives. A combination of these factors could effectively control the growth of the fungus during storage. Predictive modeling has been extensively used mainly to predict bacterial growth as a function of environmental factors such as temperature, pH and Water Activity. However, model development of filamentous fungal growth has not received the same level of attention as that of bacterial growth. A few studies concerning fungal growth have dealt with the predictive modelling approach[60].

This section illustrates the ability of the AFNN to perform combined structure and parameter learning of a non-linear three input and one output system. The predictive capability of an adaptive neuro-fuzzy system was tested to predict the combined effect of temperature, water activity and pH on the maximum specific growth rate of *Monascus ruber van Tieghem* in comparison to conventional neural networks approaches.

The three dimension input data is normalised in such a way that the maximum of each input column is equal to 0.9 and minimum equal to 0.1. However the error evaluation is based on de-normalised data. The fixed parameters of the system are defined in the form of a vector $\text{Param} = [\eta_\mu, \eta_\sigma, \eta_w, \zeta_\mu, \Omega, \hat{\delta}] = [0.001, 0.0001, 0.01, 0.05, 0.5, 0.6]$. The rule base is automatically generated along a model formed by the FNN input-output components. As can be seen from fig. 3.14, the trained FN approximates the desired function quite accurately such that the observed output almost completely overlaps the desired output.

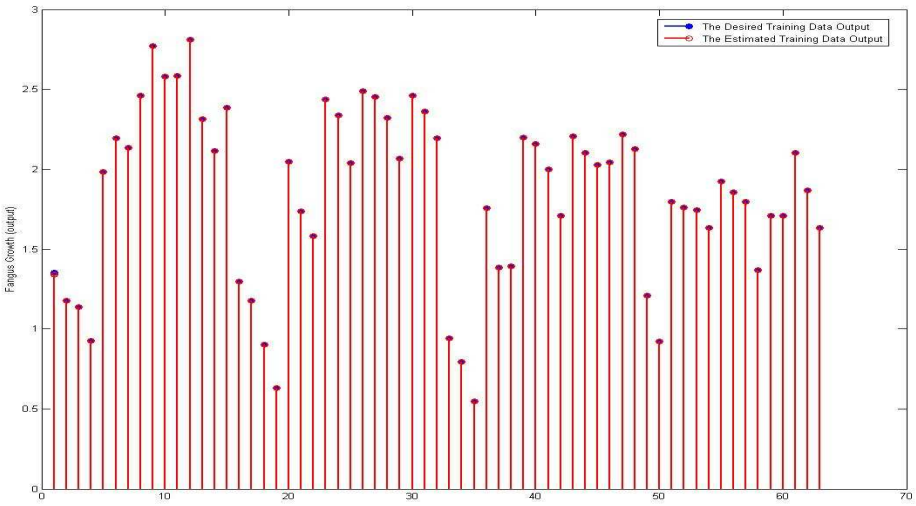


Fig 3.14– Training samples simulation results for Adaptive Fuzzy Neural Network

The training procedure starts with just one single rule and only one membership for each premise axis as depicted in figure 3.15 (a). The total MFs inserted as a result of structure learning is shown in figure 3.15 (b) and the final position of the MFs after imposing parameter learning and fine tunings illustrated in figure 3.15(c), five MFs for temperature axis, four MFs for water –activity and four

dedicated to pH axis. The exact centre positions and also deviation with and without parameter learning are mentioned in table 3.1. Figure 3.16a, 3.16b and 3.16c depict each input pair and output of AFNN, for this particular training case we attempted 63 pairs, the error measure is decreasing until a MSE of 3.1623e-005 is finally achieved on training dataset.

At the end, the structure finalised with 16 epochs and 35 fuzzy rules. Observing the curves (figure 3.16) reveals that the maximum growth occurs in case of lower Water-Activity, mid-high temperatures and in almost any pH depends on temperature and water activity, which of course shows that pH does not play a significant role in growth comparing to other two. The error criteria are as in table 3.2, the detailed definition of each error criterion can be found in appendix.

TABLE 3.1 - Normalised means and deviations of all Membership Functions with and without parameter learning **.**

		Without Parameter Learning					**With Parameter Learning**				
Temp. Axis	Mean	0.1	0.3	0.5	0.7	0.9	0.0531	0.2530	0.4530	0.6529	0.8529
	Dev.	0.08	0.08	0.08	0.08	0.08	0.0397	0.0396	0.0395	0.0395	0.0394
W.A. Axis	Mean	0.1	0.45	0.77	0.9	-----	0.0585	0.4089	0.7307	0.8558	---------
	Dev.	0.07	0.07	0.07	0.07	-----	0.0323	0.0296	0.0269	0.0243	---------
pH Axis	Mean	0.1	0.36	0.63	0.9	-----	0.1152	0.3818	0.6484	0.9150	---------
	Dev.	0.06	0.06	0.06	0.06	-----	0.0808	0.0807	0.0806	0.0805	---------

TABLE 3.2 - Adaptive Neuro-Fuzzy Error Coefficients for Training and Testing dataset

Error Coefficients	**Training**	**Testing**
Mean Square Error	3.1623e-005	0.0818
Root Mean Square Error	0.0056	0.2860
Mean Absolute Error	0.0041	0.2215
Mean Absolute Relative Error	0.3095	3.5379
Coeff. of Determination	0.9999	0.8984

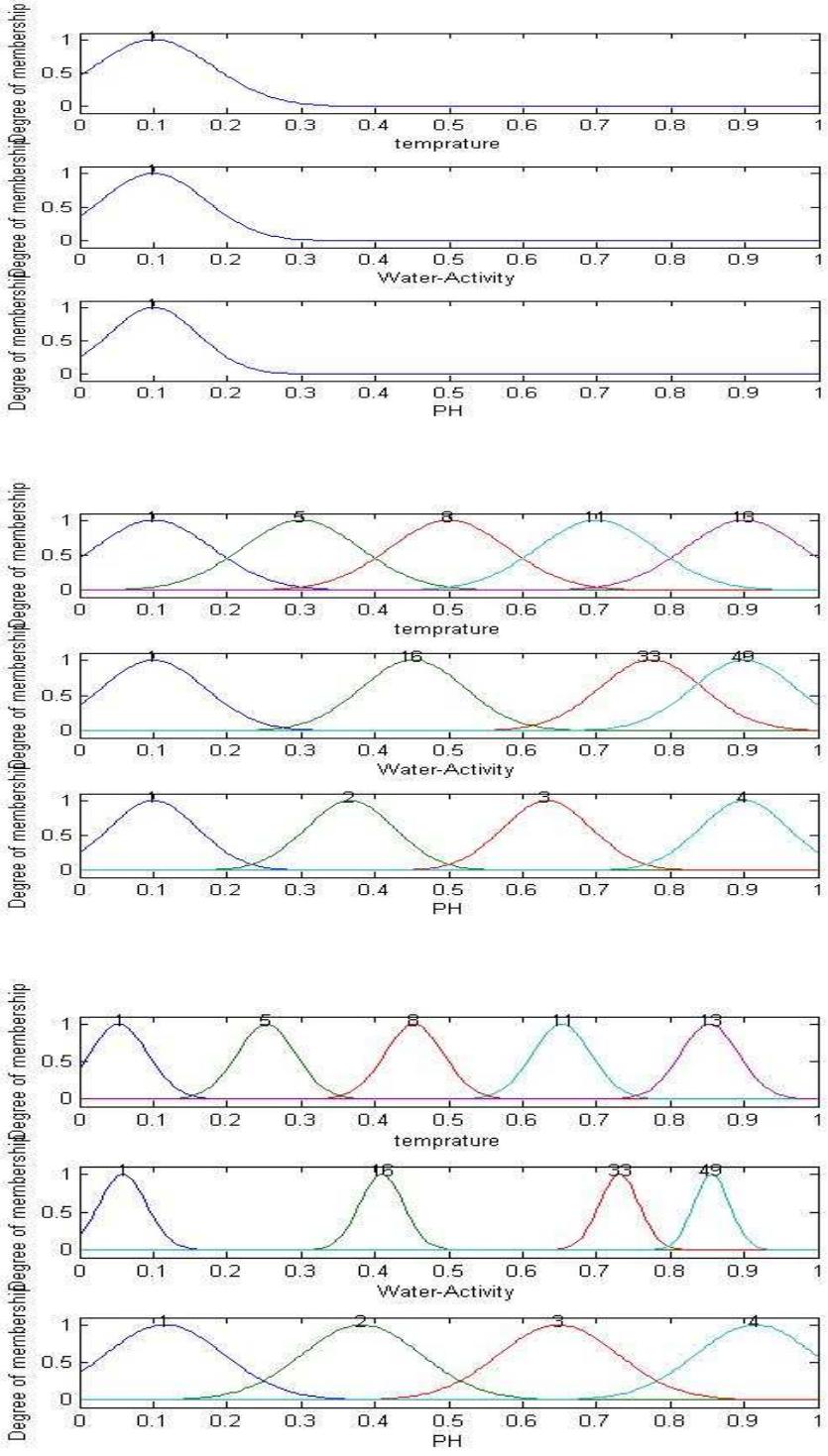


Fig 3.15 - (a) Initial membership functions for each normalized input.(b)-Memberships after structure learning process for each normalized input (c)- Final membership function together with parameter learning

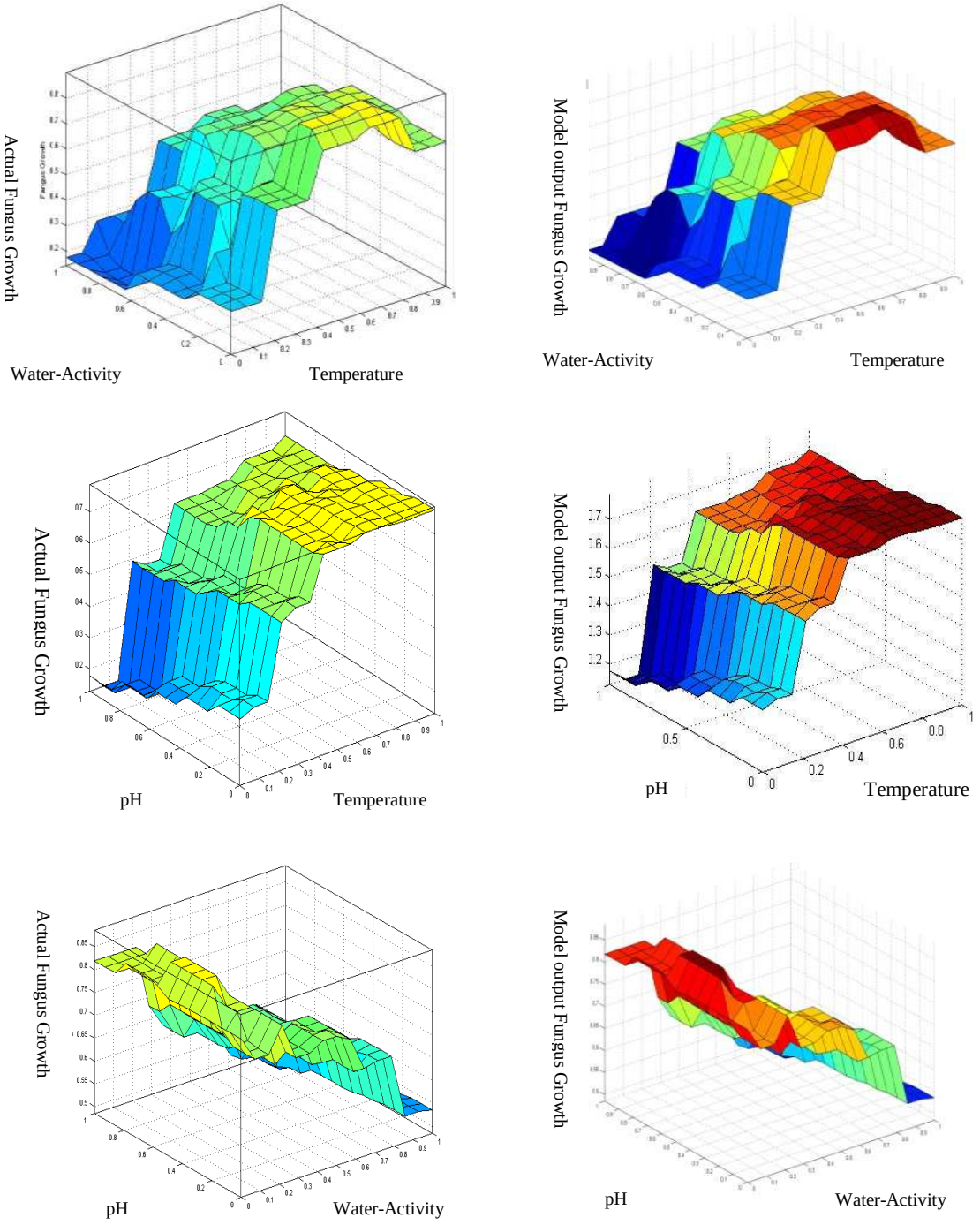


Fig 3.16 – a) Temperature and Water Activity vs. estimated output curve for normalized training data and the actual output curve b) Predicted Temperature and PH output surface and actual output surface c) PH and Water Activity predicted and desired output surface.

3.7.1 Fungus Growth Modelling By MLP

A three-input one-output Multi Layer Perceptron (MLP) has been designed to estimate the output. MLP has a simple structure but as it shown in figure 3.17 it requires more than 17000 epochs to achieve the desired output. The suggested network contains a single hidden layer (3 layer MLP) comprises of 30 neurons with sigmoid as an activation function and Back propagation (BP) training algorithm. The learning rate used here is $\eta = 0.15$ and momentum is $\zeta = 0.45$.

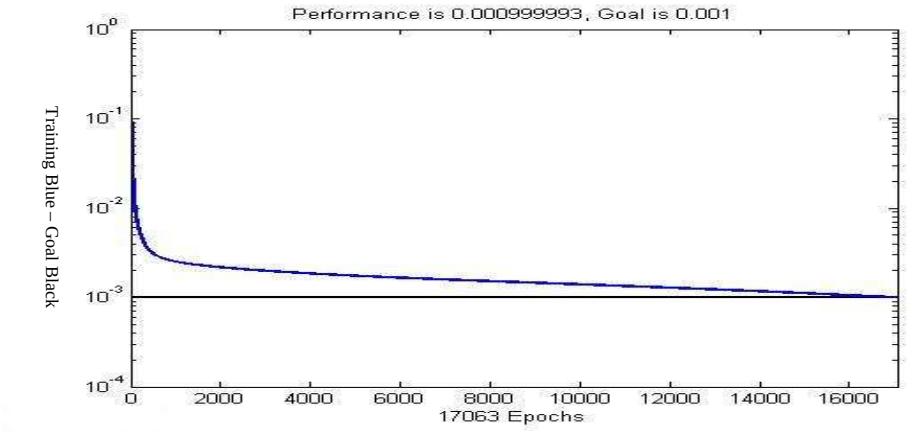


Fig 3.17– Number of epochs and the related sum square error

The training results demonstrated in figure 3.18 show the difference between the desired output and observed output.

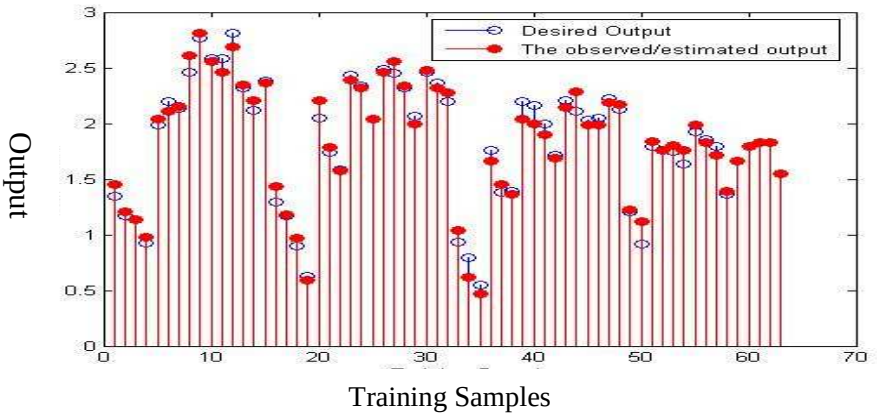


Fig 3.18 -MLP training results for each input pattern

The final statistical error coefficients are logged in table 3.3.

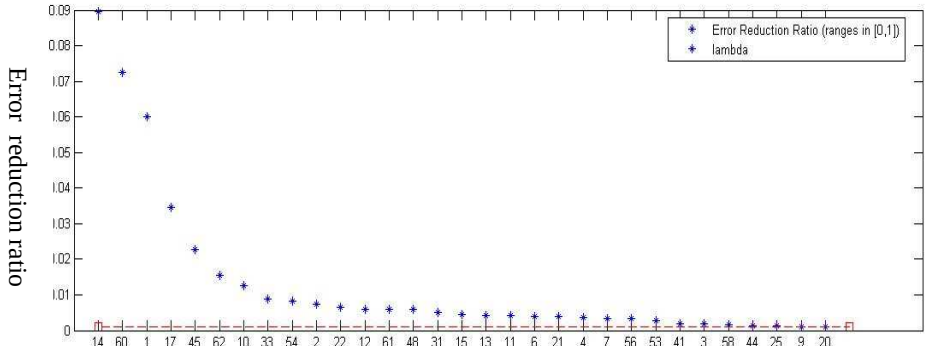
TABLE 3.3 – MLP Error coefficients for training and testing

Error Coefficients	Training	Testing
Mean Square Error	0.0080	0.1071
Root Mean Square Error	0.0895	0.3273
Mean Absolute Error	0.0753	0.3133
Mean Absolute Relative Error	4.4037	17.9191
Coefficient of Determination R^2	0.982	0.4283

3.7.2 Fungus Growth Modelling By OLS-RBF

In this section, we use OLS-RBF to model the fungus growth in accordance to three inputs (temperature, water activity and pH). OLS-RBF networks approximate an unknown function by locally constructing receptive fields around a set of centres, while these centres chosen by Orthogonal Least Square (OLS) algorithm. The RBF network and the OLS algorithm have the following fixed constants: The RBF is a Gaussian with width $\sigma = 0.8$ and desired Error Reduction Ratio set as $\rho = 0.001$.

a)



b)

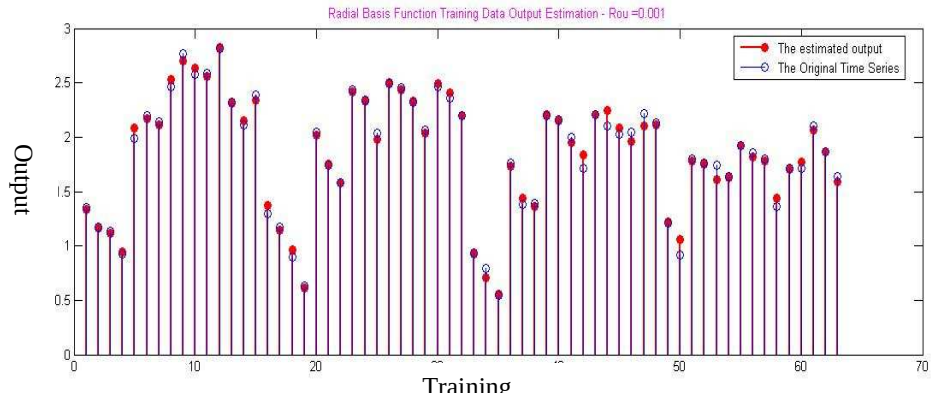


Fig 3.19- a) OLS-RBF regressors index and their contribution to error reduction
b) OLS-RBF training results for each input pattern

The results are plotted in figure 3.19(a), the asterisks represent the regressor indices versus the error ratio reduction, in this network 31 regressors has been chosen in the sequence of ; {14,60,1,17,45,62,10,33,54,2,22,12,61,48,31,15,13,11,6,21,4,7,56,53,41,3,58,44,25,9,20}.This sequence shows the data indices which are chosen to act as regressors, obviously selecting more than 31 regressors cannot further improve the model quality significantly. The accuracy of the function approximation shown in figure 3.19 (b). It is visible that the greater deviation σ is chosen for the RBFs, the smaller number of regressors is used, however this reduction comes of the expense of increasing MSE. Table 3.4 summarizes the estimated error values.

TABLE 3.4 – OLS-RBF Error coefficients for training and testing

Error Coefficients	Training	Testing
Mean Square Error	0.0027	0.0844
Root Mean Square Error	0.0516	0.2906
Mean Absolute Error	0.0442	0.2547
Mean Absolute Relative Error	2.7199	18.8620
Coefficient of Determination R^2	0.9911	0.8999

According to Table 3.2, 3.3 and 3.4, the simulation results have shown that AFNN method generates superior results and outperforms the other two.

Because the initial structure and weights of the neural network are set properly, FNN exhibits faster convergence, it can be seen from the number of epochs the AFNN is a much faster algorithm than Back Propagated MLP and RBF. RMSE values of the AFNN performed well for the training and reasonably good for the testing data set based on both graphical plots and statistical indices. In summary, the applied FNN training algorithm, as expected, well fitted to any microbiological system. It serves as a better alternative to microbiological processes predictive modelling scheme based on some of its interesting properties such as: containing only necessary number of rules, fast convergence, simple structure, less training time and of course adjustable performance.

Chapter 4

Wavelet Neural Networks

4.1 Time - Frequency Analysis

Many signals are non-stationary or in other words the spectrum of the signal can be time-varying. Thus, the standard Fourier Transform is not useful for analysing the signal. It can be spotted in many applications such as speech processing, in which we are interested in the frequency content of a signal locally in time. In this scenario characterisation of non-stationary signals in the frequency domain must therefore include the time dimension, which resulting in the time-frequency analysis. In order to do that, we usually calculate a spectrum of a signal at sufficiently short regular intervals of time. Taking an interval of time function is known as windowing which is equivalent to multiplying the signal by a window function and taking Fourier Transform (FT) of each segment, also called Short-Time Fourier Transform (STFT).

$$STFT(f,\tau) = \int_{-\infty}^{\infty} x(t)g(t-\tau)e^{-j2\pi ft}dt \qquad (4.1)$$

It might seem that the time-frequency analysis is perfect, but having a closer look reveals the problem behind the above equation. The problem is the width of the window. If we use a window of infinite length, we return back to FT, which gives perfect frequency resolution, but no time information. Likewise, in order to obtain the stationarity, a short enough window should be used, in which the signal is stationary. The narrower we make the window, the better the time resolution, and better the assumption of stationarity, but poorer the frequency resolution. The problem is as a result of choosing a window function once, and freezes it, and recycling that window throughout

the entire analysis. A Multi-Resolution Analysis (MRA) which enables us to process data at different scales or resolutions can be an ultimate solution to overcome this problem. Characteristics localization of time series in spatial (or time) and frequency (or scale) domains can be accomplished efficiently through wavelet decomposition. The power of wavelets for time series analysis stems from three features : First, wavelet analysis can determine the sharp transitions simultaneously in both frequency and time domains. Thus, wavelets can help identify nonlinear, chaotic or fractal behaviour displayed in any signal. Second, wavelet analysis allows for an effective representation of discontinuities in the chaotic time series. The wavelet representation of information in the time series allows for its hierarchical decomposition. In this way, the information can be analyzed in components of desired characteristics and at various levels of details. Third, when the information in time series is transformed into the wavelet domain less storage is required for its effective representation, resulting in computational efficiency for large time series[68].

4.2 Principles of Wavelet Transform

Wavelets with oscillation of effectively finite duration look like a small wave [69] which means it grows and decays in a finite time period as opposed to sinus and cosines used in FT who are big waves[70] and they grow and decays repeatedly in over an infinite time period. The fundamental idea behind wavelets is to analyse according to scale or Multi Resolution Analysis. The Wavelet Transform, similar to the STFT, also maps a time function into a two-dimensional function of α and τ (see eq 4.6). The parameter α is called the scale; it scales a function by compressing and stretching it, temporal analysis is performed with a contracted, high-frequency version of the wavelet, while frequency analysis is performed with a dilated, low-frequency version of the same wavelet. τ is the translation of the wavelet function along the time axis. Wavelet analysis is accomplished by first choosing a representative prototype function called the mother wavelet φ, or analyzing wavelet. A function $\varphi(t)$, defined over the real axis $(-\infty,+\infty)$ is considered as wavelet, if fulfils the following criteria

(i) The integral of φ is zero

$$\int \varphi(t)dt = 0 \qquad (4.2)$$

It ensures it has zero dc component, or in other words any excursions the wavelet function makes above zeros, must be cancelled out by excursions below zero.

(ii) Finite energy of the function. Function is leading to rapid decay toward zero with time.

$$\int_{-\infty}^{+\infty} \varphi^2(t) < \infty \qquad (4.3)$$

(iii) Admissibility Condition, is a requirement that should be fulfilled in order to have invertible transform.

$$a_\varphi \equiv \int_0^\infty \frac{|\varphi_f(f)|^2}{f} df \quad \text{satisfies} \quad 0 < a_\varphi < \infty \qquad (4.4)$$

Where $\varphi_f(f)$ is the Fourier transform of $\varphi(t)$.

One of the oldest and possibly simplest wavelet functions is the Haar wavelet(see figure 4.1), named after A.Haar who developed it in 1910. It is a step function by the definition

$$\varphi_{Haar}(x) \equiv \begin{cases} +1 & 0 \le x \le 0.5 \\ -1 & 0.5 \le x < 1 \\ 0 & \text{else} \end{cases} \qquad (4.5)$$

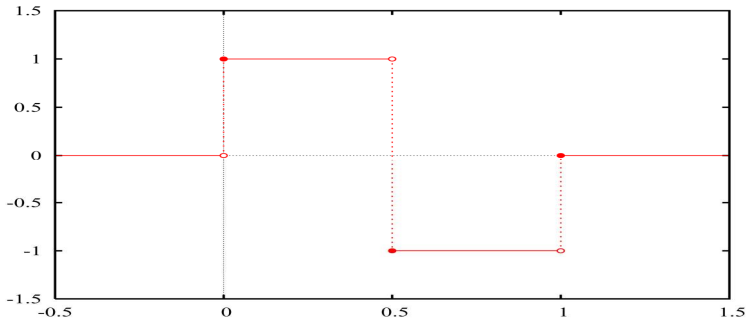


Fig 4.1 – Haar Wavelet function

There are two main types of wavelet transforms: Continuous (CWT) and discrete (DWT). The first is designed to work with functions defined over the entire horizontal axis ,whereas, DWT deals with functions that are defined over a range of integers usually $\tau = 0,1,...,T$ where T denotes the number of values in the time series. The Continuous Wavelet Transform (CWT) of a signal x(t) is as follows:

$$< x, \varphi_{\alpha,\tau} >= \frac{1}{\sqrt{\alpha}} \int x(t) \varphi(\frac{t-\tau}{\alpha}) dt \tag{4.6}$$

where the $\alpha^{-1/2}$ is for energy normalisation across the different scales, and $\varphi(t)$ are the so-called Mother Wavelet functions that satisfy certain mathematical requirements explained earlier. Since it is continuous, the parameters τ and α used for creating the wavelet family both vary continuously. The idea of transform is, for a given dilation α and a translation τ of the mother wavelet $\varphi(t)$ to calculate the amplitude coefficient which makes $\varphi_{\alpha,\tau}$ best fit the signal x(t) by eq(4.6). By integrating with eq(4.6), we can demonstrate a picture of how wavelet function fits the signal from one dilation to next one can be shown. By shifting τ, we can see how the nature of the signal changes over time. The set of coefficients $\{< x, \varphi_{\alpha,\tau} > | \alpha > 0, -\infty < \tau < \infty\}$ is called the CWT of x(t). CWT keeps all the information from main signal. If the wavelet function $\varphi(t)$ fulfils the admissibility condition and the original signal is energy limited, which means

$$\int_{-\infty}^{\infty} x^2(t) dt < \infty \tag{4.7}$$

The signal can be recovered from CWT coefficients by using the inverse transform

$$x(t) = \int_{-\infty}^{\infty} \int_{\alpha \geq 0}^{\infty} \frac{1}{\sqrt{\alpha}} < x, \varphi_{\alpha,\tau} > \varphi_{\tau,\alpha}(t) d\alpha d\tau \tag{4.8}$$

CWT and its computation is a very redundant presentation and impracticable and also may consume significant amount of time and resources, depending on the resolution required as parameters τ, α are continuous variables. The wavelet transform is calculated by continuously

shifting a continuously scalable function over a signal and calculating the correlation between the two. For most practical applications we would like to remove this redundancy like as it is always easier to deal with lower volume amount of data. The Discrete Wavelet Transform (DWT) overcomes this problem by a discrete grid of time-scale plane and is found to yield a fast computation of Wavelet Transform[71]. We can, in fact, retain the key features of the transform by only considering subsamples of the CWT. It is easy to be implemented and reduces the computation time and resources required and leading to a discrete set of continuous basis functions. Wavelet function in DWT introduced as below

$$\varphi_{mn}(t) = a_0^{-m/2}\varphi(a_0^{-m}t - n\tau_0) \tag{4.9}$$

And therefore the discrete wavelet transform of signal x (t) will be

$$DWT(m,n) = \int x(t)\varphi_{mn}(t)dt$$

(12) a_0 and τ_0 are constants that determine the sampling intervals , e.g $a_0 = 2$ and $\tau_0 = 1$, for having standard *dyadic* lattice. The perfect reconstruction achieved by

$$x(t) = \sum_m \sum_n DWT(m,n)\varphi_{mn}(t) \tag{4.10}$$

Figure 4.2 shows the DWT of a signal using *Haar* wavelet using the MATLAB Wavelet Toolbox. The diagram shows the transform for dilation m of up to 3. The signal used has the mathematical definition of

$$x(t) = \begin{cases} -2.186x - 12.864 & -10 \le x < -2 \\ 4.246x & -2 \le x < 0 \\ 10e^{-0.05x-0.5}\sin[(0.03x+0.7)x] & 0 \le x \le 10 \end{cases} \tag{4.11}$$

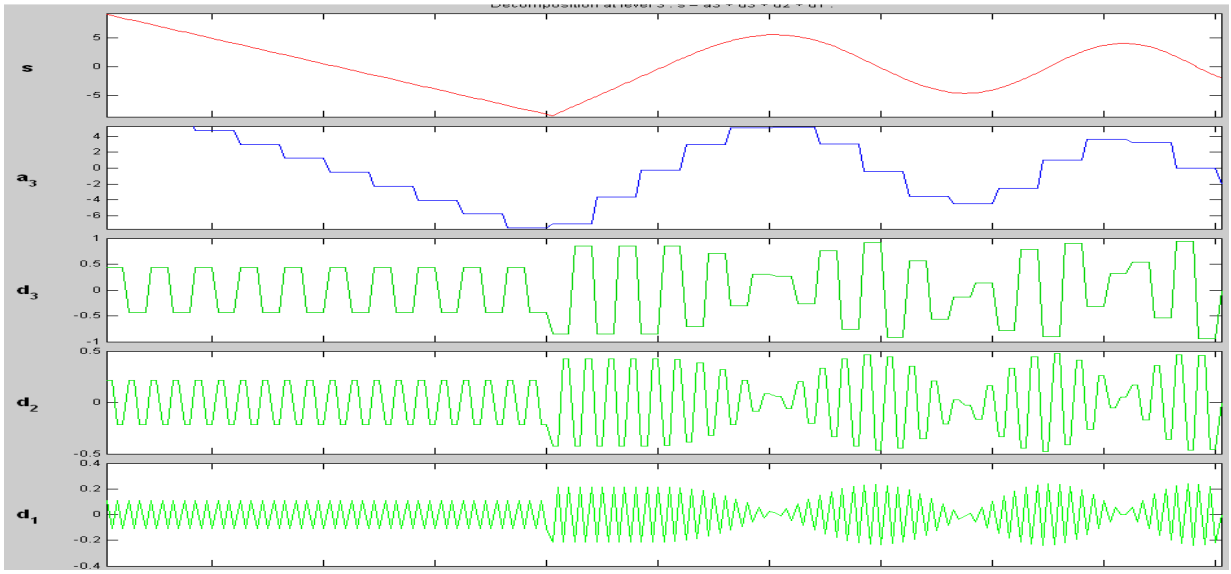


Fig 4.2 – DWT using Haar Wavelet

The last level of the transform, eliminates the high frequency components of the signal. Prior transforms remove lower and lower frequency features from the signal and we finally left with an approximation of the signal which is a lot smoother. This approximation indicates any underlying trends and the overall shape of the signal.

As with the CWT, the original signal can be reconstructed fully from its DWT. The sub-sampling performed at just dyadic scales, as a result of that, not only it seems to be a significant reduction in analysis but also it does not incur any loss in data.

4.3 Wavelet Neural Network

In the past decades, neural networks have been established as a general approximation tool for fitting nonlinear models from input-output data. A neural network derives its computing power through its massively parallel distributed structure and its ability to learn. However, the implementation of neural networks suffers from the lack of efficient constructive methods, both for determining the parameters of neurons and for choosing network structures. ANNs have limited ability to characterize local features of a time series, which are generally critical to accurately classifying or modelling the series. Since these features are often localized in time

and/or frequency, employing wavelets enables the Neural Network to take advantage of the multi-resolution analysis offered by wavelets to focus the network on these local features.

Wavelet techniques can offer added insight and performance in data analysis situations where Fourier techniques have previously been used. The idea of using wavelets in neural networks has been proposed by Zhang and Benveniste [1]. Zhang *et al.*[72] described a wavelet-based neural network for function learning and estimation, and the structure of this network is similar to that of the RBF network except that the radial functions are replaced by orthonormal scaling functions. From the point of view of function representation, the traditional RBF networks can represent any function that is in the space spanned by the family of basis functions. However, the basis functions in the family are generally not orthogonal and are redundant. It means that the RBF network representation for a given function is not unique and is probably not the most efficient. Bakshi and Stephanopoulos creatively presented an orthogonal WNN for approximation and classification based on multi-resolution analysis [73].

Wavelets have become a very active subject in many scientific and engineering research areas. Especially, wavelet neural networks (WNN), inspired by both the feed-forward neural networks and wavelet decompositions, have received considerable attention and have become a popular tool for function approximation[74]. The main characteristic of WNNs is that, as opposed to classical ANNs which use sigmoidal-based activation functions, they typically employ the DWT - which are drawn from a family of orthonormal wavelets - as the activation function for the hidden layer neurons instead of the usual sigmoid function. Each neuron in the hidden layer represents a wavelet coefficient. Since the wavelet transform results in a sparse representation, not all of the wavelet coefficients are necessary for an accurate reconstruction of the original signal. In fact, the inclusion of all of the coefficients would likely cause over training of the neural network, and result in poor convergence. For this reason, wavelet coefficients that do not contribute to the local features of the signal are identified during the iterative training of the WNN, and their corresponding neurons are pruned from the network . The simplest structure of WNN is very similar to Neural Network as shown in figure 4.3 where each neuron is commonly applied to all input variables. Here, the hidden layer consists of neurons, are usually referred to as *wavelons.*

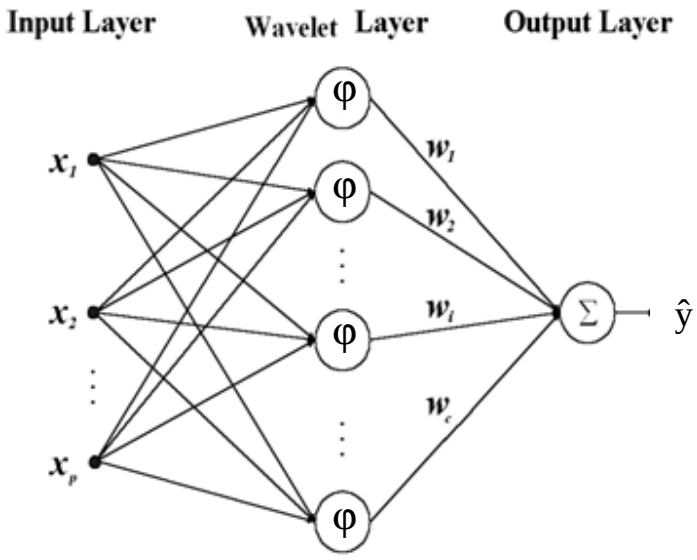


Fig 4.3 – Structure of wavelet neural network

The WNN consists of three layers: input layer, hidden layer and output layer. The connections between input units and hidden units, and between hidden units and output units are called weights v_{jp} and W_j respectively. In this WNN, the training procedure is described as follows:

- Initialising the dilation parameter m_j, translation parameter n_j and node connection weights v_{jp}, W_j to some random values. All those random values are limited in the interval (0, 1).

- Input data $x_p(t)$ and the corresponding output values y_t^d, where p varies from 1 to P, representing the number of the input nodes, t represents the t'th data sample of training set, and d represents the desired output state.

- The output value of the sample t, $\hat{y}_t$ calculated with the following formula:

$$\hat{y}_t = \sum_{j=1}^{N} W_j \varphi\left(\frac{\sum_{p=1}^{P} v_{jp} x_t^p - m_j}{n_j}\right) \qquad (4.12)$$

where φ is considered as mother wavelet, such as the Morlet wavelet filter which is shown in figure 4.4, and is represented by

$$\varphi(x) = \cos(2\pi\beta x)\exp(-0.5x^2) \tag{4.13}$$

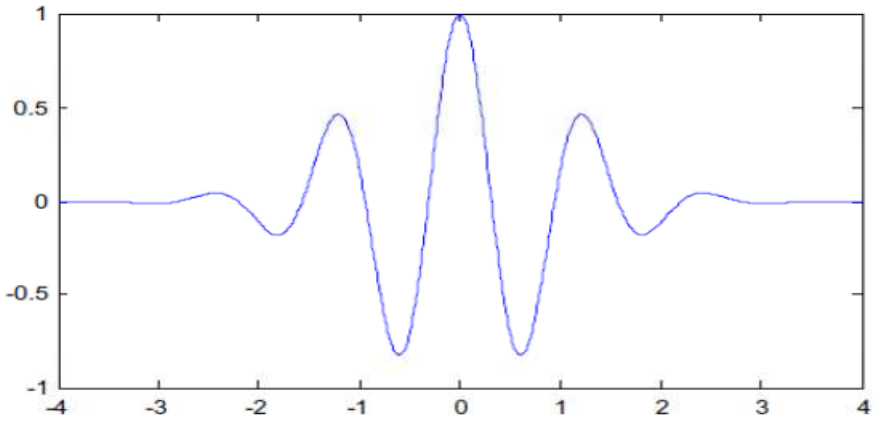


Fig 4.4 - Morlet Wavelet basis function

To reduce the error, v_{jp}, W_j, m_j, n_j are adjusted using $\Delta v, \Delta W, \Delta m, \Delta n$ .In the WNN, the gradient descend algorithm is employed, through the following equations,

$$\begin{aligned}
\Delta W_j(t+1) &= -\eta\frac{\partial E_t}{\partial W_j(t)} + \xi\Delta W_j(t) \\
\Delta v_{jp}(t+1) &= -\eta\frac{\partial E_t}{\partial v_{jp}(t)} + \xi\Delta v_{jp}(t) \\
\Delta m_j(t+1) &= -\eta\frac{\partial E_t}{\partial m_j(t)} + \xi\Delta m_j(t) \\
\Delta n_j(t+1) &= -\eta\frac{\partial E_t}{\partial n_j(t)} + \xi\Delta n_j(t)
\end{aligned} \tag{4.14}$$

Where the error function E taken as:

$$E_t = \frac{1}{2}\sum_{t=1}^{M}(y_t^d - \hat{y}_t)^2 \tag{4.15}$$

M is standing for the data number of training set. η and ζ being the learning rate and the momentum term respectively.

- The process continues until E satisfies the given error criteria, and the whole training of the WNN is completed .

Incorporating the time-frequency localisation properties of wavelets and the learning abilities of general neural network, WNN has shown its advantages over the regular methods such as NN for complex nonlinear system modelling.

4.4 Proposed Structure Scheme (WNN-LCW)

As it has been already mentioned, two key problems in designing of WNN are how to determine the WNN architecture and what learning algorithm can be effectively used for training the WNN. These problems are related to determining an optimal WNN architecture, to arrange the windows of wavelets, and to find the proper orthogonal or non-orthogonal wavelet basis.

The WNN is a kind of basis function neural network in the sense of that the wavelets consist of the basis functions. Note that an intrinsic feature of the basis function networks is the localised activation of the hidden layer units, so that the connection weights associated with the units can be viewed as locally accurate piecewise constant models whose validity for a given input is indicated by the activation functions. Compared to the MLP, this local capacity provides some advantages such as the learning efficiency and the structure transparency. However, the problem of basis function networks is also led by it. The aim of this part of research study is to investigate the feasibility of utilising WNN methodology as an alternative to classical neural networks in the area of food microbiology. The proposed, in this thesis, WNN scheme incorporates some modifications compared to classic WNNs, in order to enhance its performance. A classic WNN employs nonlinear wavelet basis functions (named wavelets) instead of using common sigmoid activation functions. The output of the network is a weighted sum of a number of wavelet functions. In the proposed linear-weights wavelet neural network (WNN-LCW), the connection weights between the hidden layer neurons and output neurons are replaced by a local linear model, similar to the output layer appeared in ANFIS neuro-fuzzy system. The output of the network is a weighted sum of a number of wavelet functions. The linear-weights wavelet neural network (WNN-LCW) is an improvement of wavelet neural network, in which the connection weights between the hidden

layer neurons and output neurons are replaced by a local linear model (similar to the TSK – as in NF systems).

A WNN approximates any desired signal y(t) by generalizing a linear combination of a set of daughter wavelets $\varphi_{m,n}(t)$ which are generated by step sizes dilation and translation *m* and *n* from a mother wavelet with either of the forms below :

$$\varphi_{m,n} = \varphi(\frac{t-n}{m})$$
$$\varphi_{m,n} = \varphi(2^{-m}t - n) \tag{4.16}$$

Where m>0. Note that eq (4.16) is similar to eq(4.6) and eq(4.9) but without the energy normalisation. For the n-dimensional input space, the multivariate wavelet basis function can be calculated by the tensor product of P single wavelet basis functions as follows:

$$\varphi(x) = \prod_{p=1}^{P} \varphi(x_p) \tag{4.17}$$

Due to the crudeness of the local approximation, a large number of basis function units have to be employed for system identification a given system. Two shortcomings of the wavelet neural network are:

- For higher dimensional problems many hidden layer units are needed
- Due to the parameters inside the activation functions in the network more epochs should be elapsed to achieve a particular accuracy [75-78].

In order to take advantage of the local capacity of the wavelet basis functions while not having too many hidden units and reasonable number of epochs, an alternative type of wavelet neural network has been adopted. Its output in the output layer is given by

$$y = \sum_{j=1}^{N} (\omega_{j0} + \omega_{j1}x_1 + ... + \omega_{jp}x_p)\varphi_j(\bar{x}) \tag{4.18}$$

Where $\bar{x}_j$ is the summation of product of weights and inputs from input layer to neuron j of hidden layer. It is shown in figure 4.5

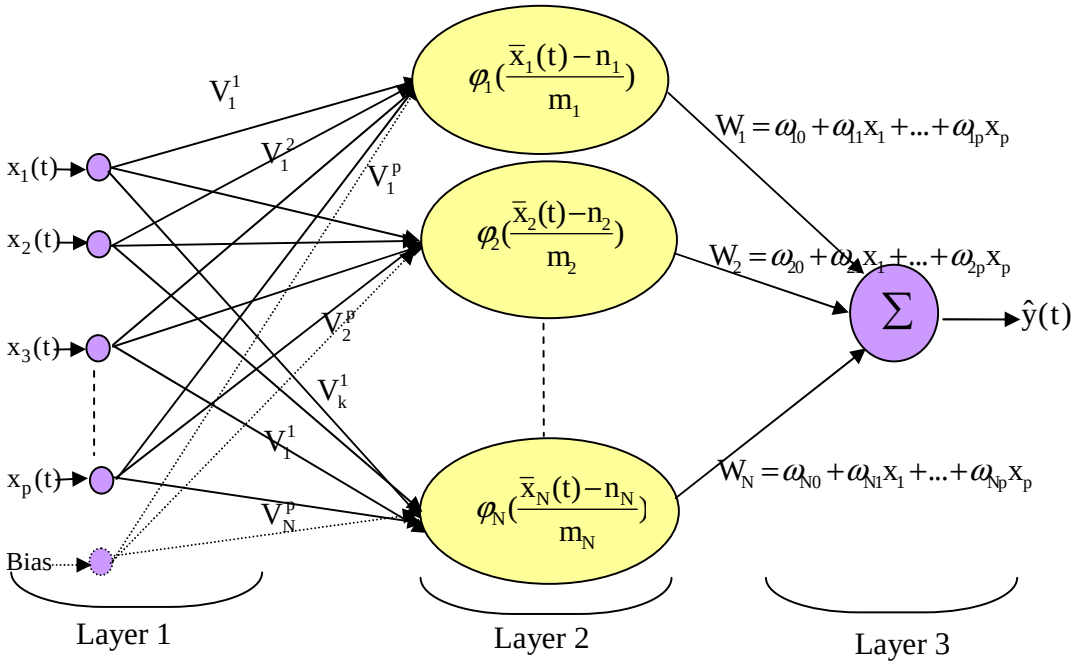


Fig 4.5 – Linear Combination Weight Wavelet Neural Network Structure

Instead of static weights between hidden layer and output layer a linear combination of weights is provided W_j. The major motivations for introducing the linear weights are mainly i) They showed good performances in TSK neuro-fuzzy systems ii) local linear models should provide a more parsimonious interpolation in high-dimension spaces. The scale and translation parameters and local linear model parameters and first-to-second layer weights are randomly initialised at the beginning and are optimized by gradient descent backpropagation algorithm utilizing partial derivatives and chain rule. The wavelet function adopted in hidden layer nodes is a modified differentiable version of Morlet wavelet as appeared in eq(4.13). This wavelet is derived from a function that is proportional to the cosine function and Gaussian probability density function. It is non-orthogonal and has infinite support[79]. Substituting (4.16) in (4.13) the activation function of j^{th} wavelet node connected with the input data will be as follows:

$$\varphi_{m_j,n_j}(\bar{x}_j) = \cos(2\pi\beta(\frac{\bar{x}_j - m_j}{n_j}))e^{-\frac{(\frac{\bar{x}-m_j}{n_j})^2}{\upsilon}} \quad (4.19)$$

.

4.4.1 The Hybrid parameter learning scheme

In the tuning phase, emphasis has been given to the efficient optimisation of the network's parameters. A hybrid learning approach has been adopted. As the proposed architecture consists of linear and non-linear parts, a two-stage learning scheme, consisting of a recursive least-squares (RLS) and the gradient descent (GD) methods has been applied.

The classical formula of least squares is in batch form, meaning that all measurements are collected first and then processed simultaneously. Such a formula poses major computational problems since the computational complexity is in the order of $O(\Omega^3)$ which grows continuously with the number of data collected[80], where Ω is the number of parameters to be estimated. To increase the efficiency of LS algorithms, a recursive variant, known as Recursive Least Squares (RLS), has been derived and is used to incrementally train a linear regression model.

The parameter learning is based on the training data after one-step-ahead prediction process is accomplished. Motivated by the fact that many output layers weights of WNN are linear, thus, it seems reasonable to employ the RLS technique to tune the parameters of the output layer during training, along with Gradient Descent (GD) for other parameters. This class of hybrid learning can speed up the learning process substantially and, simultaneously, enhance its stability[35]. We have used a hybrid method of learning comprising the Recursive Least Square (RLS) and GD. The parameters are divided into two categories; linear and non-linear parameters. For updating linear parameters RLS is utilised and for non-linear ones GD algorithm seems the simplest option. Both algorithms used such that that E in (4.15) should be minimised. Modifying (4.15) for single output and a three-layer structure we have

$$E_t = \frac{1}{2}(y_t^d - O_t^3)^2 \qquad t=1,\ldots,M \quad (4.20)$$

Where O_t^ℓ is the output of the *l*'th layer for t'th training sample. In a three layer structure O_t^3 is the final estimated output of the system and y_t^d is the desired output for the same sample and M

represents total number of inputs. The hybrid-learning algorithm of Linear Weight WNN combines the recursive least-squares (RLS) method and the back propagation gradient descent (BP/GD) to identify the parameters.

1) RLS – In forward pass node outputs go until last layer and the linear weights identified by RLS, given fixed values of Wavelet parameters, which the output can be expressed as a linear combination of the linear parameters. The estimated linear parameters are known to be globally optimal[81].

2) BP/GD – In backward pass we calculate the error signals recursively from the output layer backward to the hidden and input nodes. Thus the wavelet parameters are fine tuned by GD here.

For updating Linear Connection Weights elements $(\omega_{0j},\ldots,\omega_{Pj})$ between third and second layer first we need to make eq(4.21) linearized in terms of parameters and in order to do that the following steps are taken

$$y=\sum_{j=1}^{N} W_j(x)\varphi_j(\bar{x}_j) \tag{4.21}$$

$$\begin{cases} W_1(x)=\omega_{01}+\omega_{11}x_1+\ldots+\omega_{P1}x_P \\ W_2(x)=\omega_{02}+\omega_{12}x_1+\ldots+\omega_{P2}x_P \\ \vdots \\ W_N(x)=\omega_{0N}+\omega_{1N}x_1+\ldots+\omega_{PN}x_P \end{cases} \tag{4.22}$$

Hence, we can re-write and expand equation (4.21) in the form of

$$y=\sum_{j=1}^{N}\omega_{0j}\varphi_j(\bar{x}_j)+\sum_{j=1}^{N}\omega_{1j}x_1\varphi_j(\bar{x}_j)+\cdots+\sum_{j=1}^{N}\omega_{pj}x_p\varphi_j(\bar{x}_j) \tag{4.21}$$

If we define

$$\begin{aligned}\Phi(x)=[&\varphi_1(\bar{x}_1),\varphi_2(\bar{x}_2),\cdots,\varphi_N(\bar{x}_N),x_1\varphi_1(\bar{x}_1),x_1\varphi_2(\bar{x}_2),\cdots x_1\varphi_N(\bar{x}_N),\cdots\\ &x_p\varphi_1(\bar{x}_1),x_p\varphi_2(\bar{x}_2),\cdots x_p\varphi_N(\bar{x}_N)]^T\end{aligned} \tag{4.22}$$

And

$$\theta=[\omega_{01},\omega_{20},\cdots,\omega_{N0},\omega_{11},\omega_{21},\cdots,\omega_{N1},\cdots,\omega_{p1},\omega_{p2},\cdots,\omega_{pN}] \tag{4.23}$$

So that,

$$y(x \mid \theta) = \theta^{T}\Phi(x) \tag{4.24}$$

To compute an estimate of the θ at each time step we use following equation

If we define an $M \times P$ matrix which consists of x^t data vectors stacked on top of each other[82] we have

$$\Phi(M) = \begin{bmatrix} (x^1)^T \\ (x^2)^T \\ \vdots \\ (x^M)^T \end{bmatrix}$$

Where M is the total number of data vectors and P is the dimension of each data vector. In RLS a type of Gaussian Newton algorithm is used to update the estimated parameters θ (consists of the weights and thresholds)

$$\theta(T) = \theta(T-1) + K(T)(y_t - \Phi(T)^{T}\theta(T-1)) \tag{4.25}$$

Where y_t is the desirable output and K(t) is the data dependent Kalman Gain or the updating step size[83], given by :

$$K(t) = \frac{S(T-1)\Phi(T)}{\lambda + \Phi(T)^{T}P(T-1)\Phi(T)} \tag{4.26}$$

The Covariance matrix S(t) is updated recursively according to

$$S(T) = \lambda^{-1}(I - K(T)\Phi(T)^{T})S(T-1) \tag{4.27}$$

The initial value of the K(T) matrix, K(0), is set to zero and the initial value of S(T) matrix, S(0) is set to ϑI where I is the identity matrix and ϑ is a large positive number typically between 100-10000. Small values of ϑ may cause slow learning and large ϑ may cause the estimated parameters not to converge properly[83]. λ is called the forgetting factor is normally fixed to constant value $0 < \lambda \leq 1$ which thorough this study we have considered it as 1.

Back propagation algorithm mentioned in chapter 3, has been used according to relevant updating expressions appeared in 4.14, for wavelet parameters (m_j, n_j). They are updated by using chain rule and taking into account subscripts P and N denote the dimension of input and number of hidden layer neurons respectively. The derivatives of the error function obtained by differentiating the cost function with respect to each free tuneable parameter

$$\begin{cases} \dfrac{\partial E_t}{\partial n_j} = \dfrac{\partial E_t}{\partial O_t^3} \times \dfrac{\partial O_t^3}{\partial I_t^3} \times \dfrac{\partial I_t^3}{\partial O_j^2} \times \dfrac{\partial O_j^2}{\partial n_j} \\ \dfrac{\partial E_t}{\partial m_j} = \dfrac{\partial E_t}{\partial O_t^3} \times \dfrac{\partial O_t^3}{\partial I_t^3} \times \dfrac{\partial I_t^3}{\partial O_j^2} \times \dfrac{\partial O_j^2}{\partial m_j} \end{cases} \qquad p=1,..,P \quad , \quad j=1,..,N \tag{4.28}$$

$$\begin{cases} \dfrac{\partial E_t}{\partial n_j} = -(y_t^d - O_t^3) \times f'_{out} \times W_j \times \dfrac{\partial f_{hidden}}{\partial n_j} \\ \dfrac{\partial E_t}{\partial m_j} = -(y_t^d - O_t^3) \times f'_{out} \times W_j \times \dfrac{\partial f_{hidden}}{\partial m_j} \end{cases} \tag{4.29}$$

I_t^{ℓ} and O_t^{ℓ} are the input and output of layer *l* for the sample at time instance t respectively.

The output layer function - throughout this research - is linear, therefore it's derivative against its corresponding input from previous layer, f'_{out} =1.

Where, by considering eq(4.19)

$$\begin{cases} \dfrac{\partial f_{hidden}}{\partial n_j} = \dfrac{2\pi\beta}{m_j} \sin(2\pi\beta(\dfrac{x-n_j}{m_j}))e^{\frac{-(\frac{x-n_j}{m_j})^2}{\upsilon}} + (\dfrac{\frac{2(x-n_j)}{m_j^{\,2}}}{\upsilon})\cos(2\pi\beta\dfrac{x-n_j}{m_j})e^{\frac{-(\frac{x-n_j}{m_j})^2}{\upsilon}} \\ \dfrac{\partial f_{hidden}}{\partial m_j} = \dfrac{2\pi\beta(x-n)}{m^2} \sin(2\pi\beta(\dfrac{x-n_j}{m_j}))e^{\frac{-(\frac{x-n_j}{m_j})^2}{\upsilon}} + (\dfrac{(\frac{x-n_j}{m_j^{\,2}})(\frac{x-n_j}{m_j})}{\upsilon})\cos(2\pi\beta\dfrac{x-n_j}{m_j})e^{\frac{-(\frac{x-n_j}{m_j})^2}{\upsilon}} \end{cases} \tag{4.30}$$

$$\begin{cases} n_j(t+1) = n_j(t) + \Delta n_j \\ m_j(t+1) = m_j(t) + \Delta m_j \end{cases} \tag{4.31}$$

Where

$$\begin{cases} \Delta n_j = \eta_n \dfrac{\partial E}{\partial n_j} + \zeta n_j(t-1) \\ \Delta m_j = \eta_m \dfrac{\partial E}{\partial m_j} + \zeta m_j(t-1) \end{cases} \tag{4.32}$$

Similarly, the updated law of connection weights between input and wavelet layer weights (v_{ij}) is given as follow

$$\frac{\partial E_t}{\partial v_{jp}} = \frac{\partial E_t}{\partial O_t^3} \times \frac{\partial O_t^3}{\partial I_t^3} \times \frac{\partial I_t^3}{\partial O_j^2} \times \frac{\partial O_j^2}{\partial I_j^2} \times \frac{\partial I_j^2}{\partial v_{jp}} \qquad p=1,..,P \quad ,j=1,..,N \tag{4.33}$$

$$\frac{\partial E_t}{\partial \omega_{jp}} = -(y_t^d - O_t^3) \times f'_{out} \times W_j \times \frac{\partial \varphi_j}{\partial \overline{x}_j^{\,t}} \times x_p$$

By getting the first deviation from (18)

$$\frac{\partial \varphi_j}{\partial \overline{x}_j} = -\frac{2\pi\beta}{m_j} \sin(2\pi\beta \frac{x - n_j}{m_j}) e^{\frac{-(\frac{x-n_j}{m_j})^2}{\upsilon}} - \frac{2x}{m_j \upsilon} \cos(2\pi\beta \frac{x - n_j}{m_j}) e^{\frac{-(\frac{x-n_j}{m_j})^2}{\upsilon}} \tag{4.34}$$

$$v_{jp}(t+1) = v_{jp}(t) + \Delta v_{jp} \tag{4.35}$$

Where

$$\Delta v_{jp} = \eta_v \frac{\partial E}{\partial v_{jp}} + \zeta v_{jp}(t-1) \tag{4.36}$$

The η_m , η_n and η_v represent learning rates for m, n and v respectively and ζ is the momentum. As we can see from above discussed equations, the use of Linear Combination Weights proposed in this thesis, does not complicate the implementation of the tuning procedures significantly, providing at the same time a higher rate of convergence and better accuracy.

4.4.2 Case Analysis - Prediction of pressure inactivation of *Listeria monocytogenes* in whole milk

Listeria monocytogenes is a ubiquitous food-borne pathogen associated with outbreaks of listeriosis from consumption of various food commodities such as vegetables, dairy products, seafood and meat[84]. The pathogen is of great health concern for the food industry because it is characterised by high mortality rates, especially in pregnant women, neonates, elderly and immune-compromised[85]. The pathogen can grow at refrigeration temperatures and survive in

foods for prolonged periods of time under adverse conditions[86]. It is a very hardy micro-organism that can grow over a wide range of pH values (4.3 to 9.1) and temperature range from 0 to 45 °C. In addition, it is relatively resistant to desiccation and can grow at aw values as low as 0.90 [87]. There has been continued interest in the food industry in using high hydrostatic pressure processing as a non-thermal preservation technique. Its primary advantage is that it can inactivate microorganisms and certain detrimental enzymes at ambient temperatures, and thus avoid the effects of cooking temperatures on various food quality attributes, such as nutritional qualities, flavour and taste. Although the inactivation kinetics of microorganisms using heat has been extensively studied, information on the inactivation kinetics of microorganisms under high pressure, especially under simultaneous application of pressure and other processing techniques, is still limited. Accurate prediction of the effectiveness of high pressure processing against foodborne pathogens based on inactivation kinetics is essential to permit production of safe products. The overall objective of this study is to design one-step ahead predictive schemes to model the survival of L. monocytogenes in ultra high temperature (UHT) whole milk during high pressure treatment using the proposed WNN-LCW structure. Its performance will be judged against a MLP and a linear PLS regression model. Two nonlinear conventional statistical models (Weibull, Gompertz) used in predictive food microbiology will be also considered and an evaluation will be made to compare the goodness-of-fit of these models. L. monocytogenes NCTC 10527 from the collection of the Laboratory of Microbiology and Biotechnology of Foods were used throughout this study. The data for different pressures, (300, 350, 400, 450, 500, 550 and 600 MPa) were provided by Agricultural University of Athens, Greece.

4.4.3 Initialisation of the network parameters

Initialising the wavelet network parameters is an important issue. Similar to Radial Basis Function networks (and in contrast to neural networks using sigmoidal functions), a random initialisation of all the parameters to small values (as usually done with neural networks) is not desirable since this may make some wavelets too local (small dilations) and make the components of the gradient of the cost function very small in areas of interest. In general, one wants to take advantage of the input space domains where the wavelets are not zero.

We denote by $\left[a_p, b_p\right]$ the domain containing the values of the p^{th} component of the input vectors of the examples. We initialise the vector m of wavelet j at the centre of the parallelepiped defined by intervals $\left\{\left[a_p, b_p\right]\right\}: m_{jp} = \frac{1}{2}\left(a_p + b_p\right)$. The dilation parameters are initialised to the value $0.2\left(b_p - a_p\right)$ in order to guarantee that the wavelets extend initially over the whole input domain. Throughout this research we have used $\beta = 0.5$ and $\upsilon = 1$ as the optimal choice for our dataset. The remaining parameters are initialised to small random values.

4.4.4 Dynamic System Identification

In nonlinear systems prediction, the purpose of modelling is different for different applications. In many cases the data are ill-conditioned and the support of delayed versions of outputs and inputs are needed to achieve the desired accuracy, which make us to switch from static system modelling to dynamic system modelling.

In general, dynamic systems are complex and nonlinear. An important step in nonlinear systems identification is the development of a nonlinear model. In recent years, computational-intelligence techniques, such as neural networks, fuzzy logic and combined hybrid systems algorithms have become very effective tools of identification of nonlinear plants. The problem of identification consists of choosing an identification model and adjusting the parameters, such that the response of the model approximates the response of the real system to the same input. In the framework of this research study, the proposed WNN-LCW structure will be utilised as a nonlinear model.

Different methods have been developed in the literature for nonlinear system identification. These methods use a parameterised model. The parameters are updated to minimise an output identification error. A wide class of nonlinear dynamic systems with an input *u* and an output can be described by the models mentioned in Chapter 2, generally defined as

$$y_m(k) = f(\boldsymbol{\varphi}(x_k), \Theta)$$

Where, $y_m(k)$ is the output of the model, $\boldsymbol{\varphi}(x_k)$ is the regressor vector and Θ includes all the weights and other wavelet parameters in the network. Depending on the choice of the regressors in $\boldsymbol{\varphi}(x_k)$, different models can be derived [8]

- NARX(non-linear Autoregressive with eXogenous inputs) which is series parallel model.

As figure 4.6a illustrates, it means the outputs of the actual plant are used as input to the model. Only one step ahead prediction is possible(fig 4.6b). The model said to have external dynamics.

$$\varphi(k) = \left(u(k), u(k-1), u(k-2), ..., u(k-n_u), y(k-1), y(k-2), ..., y(k-n_y)\right) \quad (4.37)$$

- NOE(Non-linear Output Error) which is parallel model. It means the model output itself create time-lagged inputs, as depicted in fig 4.6c. This model can be considered as fully recurrent model. The parallel model is able to give predictions over a short period of time. The model is said to have internal dynamics.

$$\varphi(k) = \left(u(k), u(k-1), u(k-2), ..., u(k-n_u), y_m(k-1), y_m(k-2), ..., y_m(k-n_y)\right) \quad (4.38)$$

In both cases the prediction error of the model, compared with the true plant outputs are used as a measure to optimise the model parameters. For dynamic systems, the model must have some way to implement time lags. In other words, some memory function must be present in the model. In modelling using computational intelligence schemes, such as neural networks, neuro-fuzzy systems, WNNs, this can be done in two ways: either, delayed inputs and outputs are used as extra external inputs, or some memory is included in the individual neurons.

Models with external dynamics can be seen as one-step-ahead predictors. Models with internal dynamics are best used for simulation purposes, as the model doesn't need the true plant outputs. The latter case has a higher potential for output errors in the long term. the prediction error can accumulate during iteration and larger error can occur[88]. This is certainly the case for nonlinear systems, where the internal nonlinearities can drive the system into an unstable state. Since for nonlinear problems the complexity usually increases strongly with the input space dimensionality (curse of dimensionality) the application of lower dimensional NARX or NOE models is more widespread. One drawback of these models is that the choice of the dynamic order,, is crucial for the performance and really efficient methods for its determination are not available. Often the user is left with a trial-and-error approach.

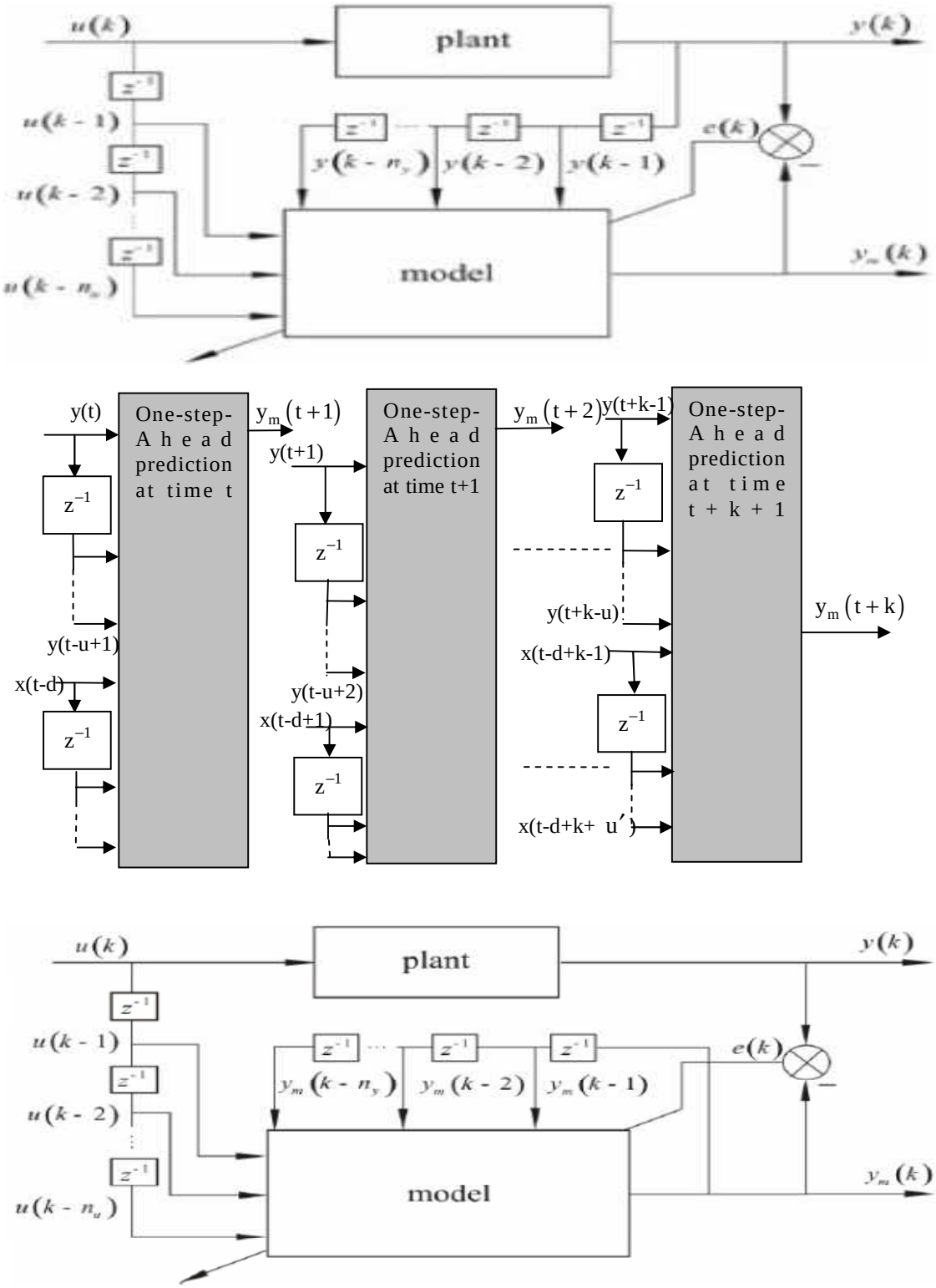


Fig 4.6 – a) Series Parallel dynamic system b) One step a head prediction c)Parallel mode

4.4.5 Model Development

4.4.5.1 Primary Modelling

The survival curves of *L. monocytogenes* during high pressure inactivation were fitted with two primary models to determine the kinetic parameters of *L. monocytogenes* in UHT whole milk. The first model applied was the re-parameterized Gompertz equation[89], determined by the following equation

$$\log_{10} N_{Patho}(t) = \log_{10} N_{Patho}(0) + A \cdot \exp\left\{-\exp\left[\frac{k \cdot e}{A} \cdot (t_s - t) + 1\right]\right\} \qquad (4.39)$$

where t_s [min] is the duration of the shoulder, k [min^{-1}] is the maximum specific inactivation rate, $N_{Patho}(0)$ [log CFU ml^{-1}] is the initial population density of the pathogen, and A [log CFU ml^{-1}] is the difference between the initial and residual population.
The second model was based on the modified Weibull equation[90] which can be defined as:

$$\log_{10} N_{Patho}(t) = \log_{10}\left[\left(N_{Patho}(0) - N_{res}\right) \cdot 10^{\left(-\left(\frac{t}{\delta}\right)^h\right)} + N_{res}\right] \qquad (4.40)$$

where δ [min] is a scale parameter denoting the time for the first decimal reduction, and h [dimensionless unit] is the shape factor of the curve. For $h > 1$, convex curves are obtained whereas for $h < 1$ concave curves are described. Finally, $N_{Patho}(0)$ and N_{res} [log CFU ml^{-1}] are the initial and residual population of the pathogen, respectively.

4.4.5.2 Non-Parametric Modelling

Partial least squares (PLS) regression, a multivariate calibration technique, projects the initial input-output data down into a latent space, extracting a number of principal factors (also known as

latent variables) with an orthogonal structure, while capturing most of the variance in the original data. In brief, it can be expressed as a bilinear decomposition of both $\boldsymbol{X}$ and $\boldsymbol{Y}$ as:

$$X = \boldsymbol{T}\boldsymbol{W}^{T} + E_{X} \tag{4.41}$$

and

$$Y = \boldsymbol{U}\boldsymbol{Q}^{T} + E_{Y} \tag{4.42}$$

such that the scores in the *X*-matrix and the scores of the yet unexplained part of *Y* have maximum covariance. Here, $\boldsymbol{T}$ and $\boldsymbol{W}$, $\boldsymbol{U}$ and $\boldsymbol{Q}$ are the vectors of *X* and *Y* PLS scores and loadings (weights), respectively, while E_X, E_Y are the *X* and *Y* residuals[91]. The decomposition models of X and Y and the expression relating these models through regression constitute the linear PLS regression model. In case of one Y-variable, y, the model can be expressed as a regression equation

$$y = bX + E \tag{4.43}$$

where *b* is the regression coefficient. The PLS model is developed in two stages; the initial dataset is divided into training and testing subsets. The former dataset is used to build the models and compute a set of regression coefficients ($\boldsymbol{b}_{PLS}$), which are subsequently used to make a prediction of the dependent variable in the test subset.

Multilayer Perceptron structure is probably the most widely used neural network paradigm and has long proven nonlinear modelling capabilities/performance. The knowledge of the network is stored in the weights connecting the artificial neurons. The massively interconnected structure of the MLP provides a great number of these weights and as such a great capacity for storing complex information. The generalised delta rule is applied for adjusting the weights of the feedforward networks in order to minimise a predetermined cost error function.

4.5.6 Model Validation

The Wavelet network, and PLS and MLP schemes as well as the statistical models were comparatively evaluated to determine whether they could successfully predict the responses of the pathogen at pressure levels other than those initially selected for model development. For this reason, two different high pressure levels, within the range employed to develop the models, were selected namely 400 and 500 MPa. At predetermined time intervals the surviving population of *L. monocytogenes* was enumerated and compared with the survival curves predicted by the developed in this study models. The accuracy of the prediction was estimated by the calculation of the bias

(B_f) and accuracy (A_f) factors [92], the regression coefficient (R^2), the standard error of prediction (SEP), the mean absolute percentage error (MAPE) and the root mean square error (RMSE) .

The shapes of the survival curves that follow those experimental data change considerably depending on the treatment pressure levels. However, in all pressure levels assayed, a clear inactivation pattern was observed including a lag phase (or shoulder), a log-linear and a tailing phase. As expected, the duration of shoulder was pressure dependent, so higher pressures resulted in lower shoulder time. At different pressure levels, survival curves showed a pronounced curvature and tailing indicating that a small population of the pathogen could resist pressurization and eventually survive in milk.

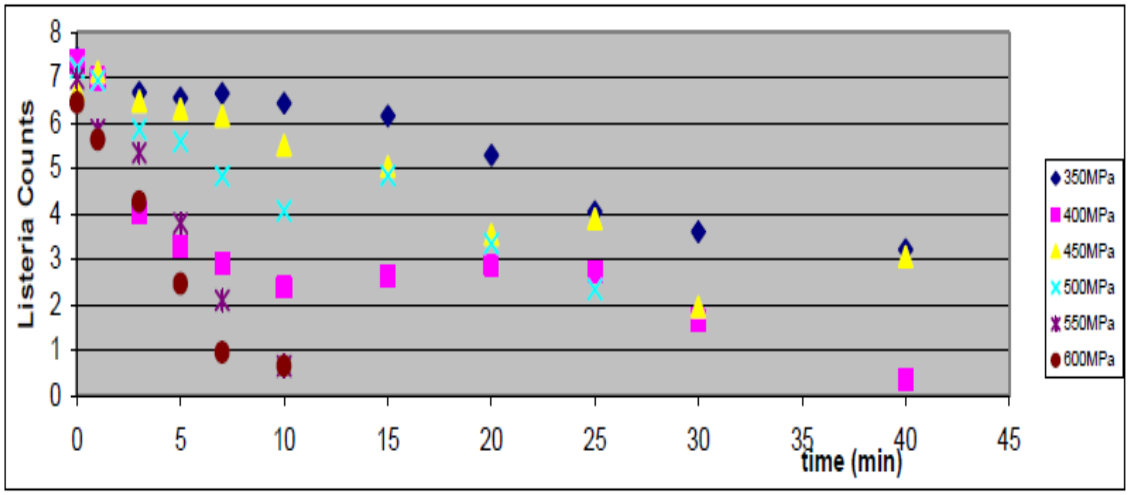


Fig 4.7 – Survival Curves of *Lysteria* in various pressures

The estimated kinetic parameters of inactivation based on the models of re-parameterized Gompertz and modified Weibull are presented in Table 4.1. All models fitted the experimental data well as can be inferred by the high values of regression coefficient ($R^2 > 0.97$) and low values of root mean square error (RMSE < 0.45). Figure 4.18 (a & b) illustrates the models' performance on the training data.

The prediction capability of those models was considered by adopting a two-step standard procedure commonly applied in predictive microbiology [56]. Initially, the primary models (i.e., Gompertz, Weibull) were fitted to high pressure inactivation data and the respective kinetic

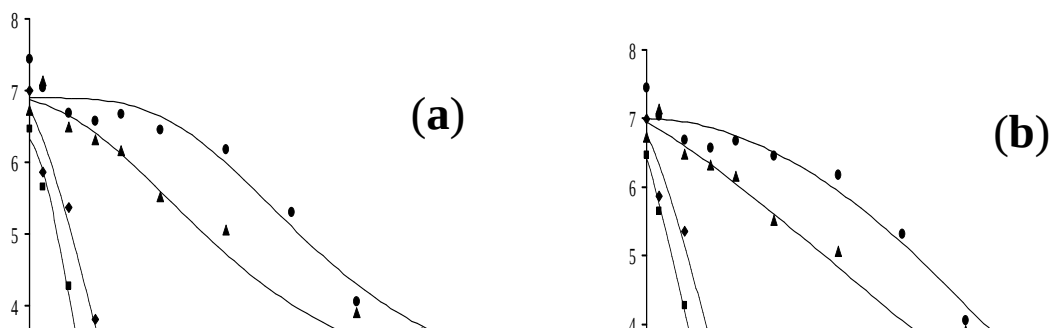
parameters were calculated (Table 4.1). Subsequently, the derived kinetic parameters were related to high pressure levels through the development of first or second order secondary polynomial models (Table 4.2) and their new estimates were determined at 400 and 500 MPa, which have been pre-selected for model validation. For the kinetic parameters which did not present a clear trend with pressure, their respective values at 400 and 500 MPa were determined by interpolation.

TABLE 4.1 - Parameters[a] and statistics of secondary models for the effect of high pressure on the kinetic parameters of *Listeria monocytogenes* in UHT whole milk.

Model type	Parameter	Equation	Estimated value	h	R^2
Gompertz [b]	k_{max}	$k_{max} = a_1 \cdot P^2 + a_2 \cdot P + a_3$	$a_1 = 4.43 \cdot 10^{-5} \pm 0.71 \cdot 10^{-5}$ $a_2 = -0.034 \pm 0.007$ $a_3 = 6.908 \pm 1.674$	0.009 0.005 0.002	0.981
Weibull [c]	δ	$\ln(\delta) = a_1 \cdot T + a_2$	$a_1 = -0.009 \pm 0.001$ $a_2 = 6.175 \pm 0.517$	0.006 0.011	0.977

[a] Data are values ± standard deviation.
[b] The parameters N_0, A and t_s of the Gompertz model at 400 and 500 MPa were determined by interpolation.
[c] The parameters N_0, N_{res} and the shape factor (h) of the Weibull model at 400 and 500 MPa were determined by interpolation.

Finally, based on the new values of the kinetic parameters at the selected pressures for validation, equations 4.39 and 4.40 were refitted and compared with survival data of the pathogen at the same pressures, in order to determine the potential of the models for generalisation, i.e., their ability to foresee survival curves at pressures for which there was no previous training. The performance against the unknown 400MPa and 500MPa curves is illustrated in figure 4.9 and 4.10, respectively.



log N (CFU ml^{-1})

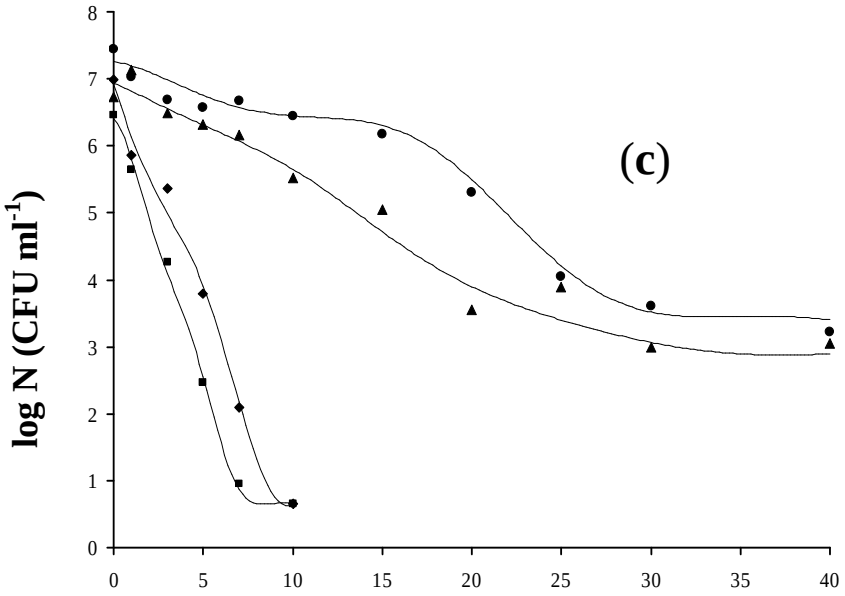


Fig 4.8 - Survival curves of *Listeria monocytogenes* in UHT whole milk during high pressure processing at 350 MPa (●), 450 MPa (▲), 550 MPa (◆), and 600 MPa (■), generated by the re-parameterised Gompertz model (a), the modified Weibull model (b), the Geeraerd model (c), and the wavelet neural network . Data points are mean values of two independent experiments with two replications each

TABLE 4.2 -Parameter estimation[a] and statistical indices of the different models used for fitting the survival of *L. monocytogenes* in whole UHT milk during high pressure treatment.

Model type	$\log_{10} N_0$ [CFU ml^{-1}]	$\log_{10} A$[b] [CFU ml^{-1}]	$\log_{10} N_{res}$ [CFU ml^{-1}]	k_{max} [min^{-1}]	t_s [min]	δ [min]	h [-]	RMSE	R^2
Gompertz									
350 MPa	6.90 ± 0.16	4.08 ± 0.59		0.41 ± 0.03	10.07 ± 2.51			0.307	0.969
450 MPa	6.99 ± 0.42	4.15 ± 0.69		0.39 ± 0.04	2.25 ± 0.16			0.313	0.971
550 MPa	7.36 ± 0.32	6.21 ± 0.18		1.68 ± 0.17	- [c]			0.432	0.987
600 MPa	6.51 ± 0.58	6.30 ± 0.96		2.26 ± 0.16	-			0.311	0.993
Weibull									
350 MPa	7.00 ± 0.15		3.27 ± 0.25			14.53 ± 1.88	1.88 ±0.40	0.266	0.977
450 MPa	6.94 ± 0.24		3.09 ± 0.25			7.71 ± 2.14	1.13 ± 0.28	0.318	0.970
550 MPa	6.74 ± 0.37		0.61 ± 0.48			2.05 ± 0.74	1.24 ± 0.34	0.4136	0.988
600 MPa	6.41 ± 0.10		0.65 ± 0.10			1.46 ± 0.14	1.11 ± 0.07	0.102	0.999
Wavelet									
350 MPa								0.168	0.994
450 MPa								0.249	0.986
550 MPa								0.200	0.995
600 MPa								0.110	0.998

[a] Data are values ± standard deviation.
[b] A is the difference between the initial population (N_0) and the residual population (N_{res}).
[c] No shoulder was observed.

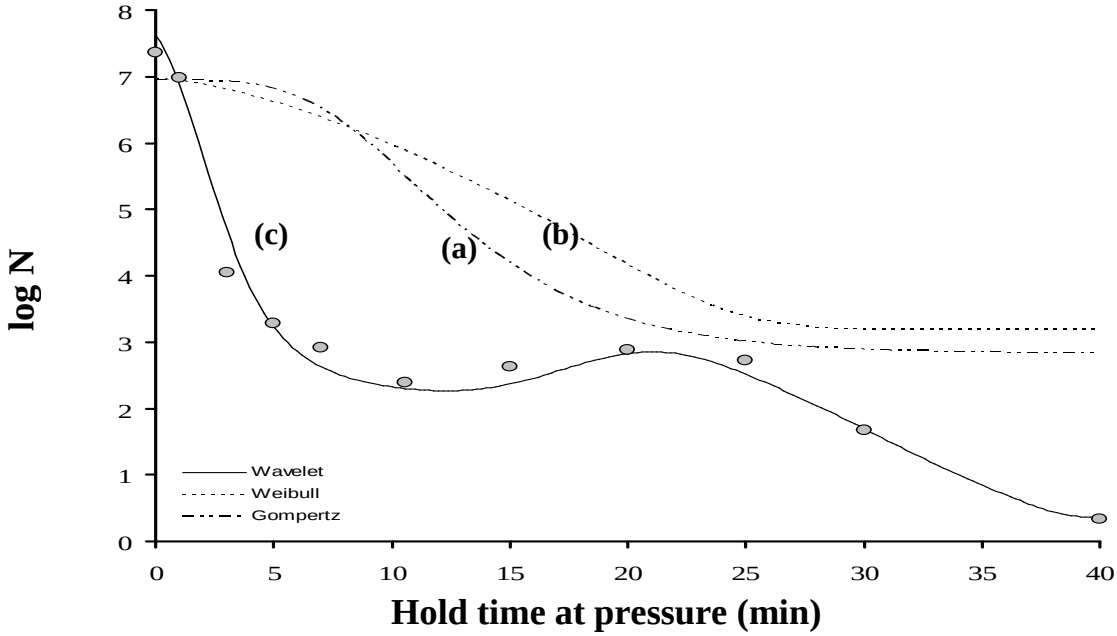


Fig 4.9- Observed values and predicted survival curves of *Listeria monocytogenes* in UHT whole milk during high pressure treatment at 400 MPa, generated by the reparameterized Gompertz model (a), the modified Weibull model (b), and the wavelet neural network (c). Data points are mean values of two independent experiments with two replications each.

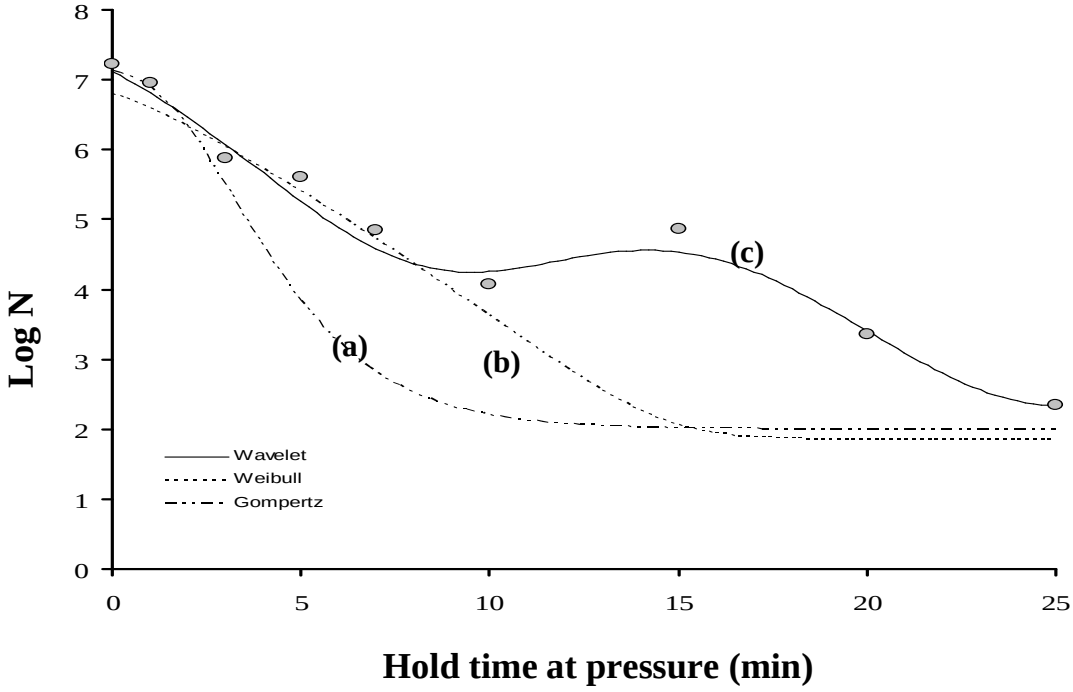


Fig 4.10- Observed values and predicted survival curves of *Listeria monocytogenes* in UHT whole milk during high pressure treatment at 500 MPa, generated by the reparameterized Gompertz model (a), the modified Weibull model (b), and the wavelet neural network (c). Data points are mean values of two independent experiments with two replications each.

It must be pointed out that despite the plethora of proposed inactivation models in the literature none is flexible enough to account for all changes of shapes with the intensity of stress[93]. The selected models were able to describe the survival of the pathogen at 350, 450, 550, and 600 MPa quite accurately. However, the prediction at 400 and 500 MPa was not very accurate as the experimental values of the pathogen showed a pattern which possibly indicated the presence of two subpopulations, one sensitive to high pressure that was inactivated within the first 10 min of the process (figure. 4.9 and 4.10) and a second more resistant to the applied stress. The discrepancy observed in the prediction at these pressure levels could be attributed to the fact that both models did not account for the presence of a mixed population of the pathogen with a variable resistant to high pressure.

Small data set conditions exist in many fields, such as food analysis, disease diagnosis, fault diagnosis or deficiency detection in mechanics, aviation and navigation, etc. The main reason that small data sets cannot provide enough information as that of large ones is that there exist gaps between samples; even the domain of samples cannot be ensured[94]. It is hard to catch the pattern of high order non-linear functions by a standard feed-forward neural network-like scheme, with a small sample set, since they have shown weakness in providing sufficient information for forming population patterns. Lacking the whole picture of a function means the network cannot precisely identify which sections of the function are ascending and which sections are descending. Hence, for learning systems that lack sufficient data, the knowledge learned is often unacceptably rough or unreliable.

How to fill up the gaps is the primary problem to be solved. Inspired by the way the RBF network approximates a nonlinear function through Gaussian local-basis functions, we employed such a network to each “survival curve” defined from the experimental data. The aim was to associate each local-basis-function to each sample, and therefore easy then to generate new data that satisfy each “survival curve”. An RBF network using the regularised orthogonal least squares learning algorithm has been employed for this task [44].

The inputs included the type of pressure level and the sampling time-step, while the output was related to the bacteria counts. Each “continuous survival curve” has been verified against the real experimental samples.

For each pressure level case, an RBF network has been associated. As the real number of samples for each pressure level is very limited, we associated each RBF centre with the real samples. Then

with a constant time-step 0.5 min, through a 2-inputs network, continuous survival curve" has been obtained for each pressure level, as shown in figure 4.11.

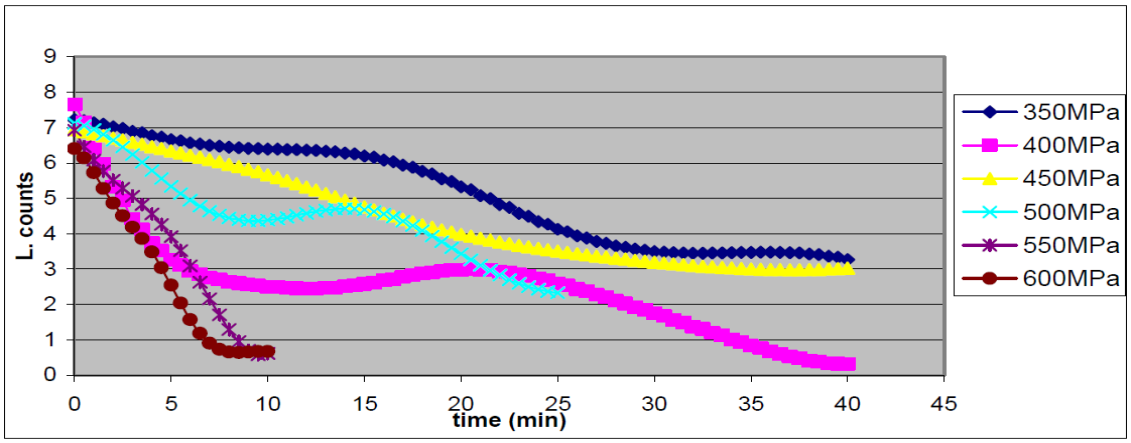


Fig 4.11 – Continuous Survival curves of *Lysteria monocytogenes*

Based on these continuous datasets, the capabilities of proposed WNN-LCW architecture has been verified as a one-step-ahead prediction system. Comparative studies have been conducted with the utilisation of a PLS regression model and an MLP neural network. Pressure levels of 400 MPa and 500 MPa have been used as testing datasets, while the remaining levels as training ones.

The PLS model was initially constructed using the "continuous survival curve" dataset which is comprised of two inputs and one output. Two latent variables were selected and the resulting equation has the following form

$$Y_1 = 12.8684247 - 0.0150321224 X_1 - 0.1096417853 X_2 \qquad (4.44)$$

Figure 4.12 illustrates the performance of the produced linear model on the testing data curves. Obviously, the dynamic behaviour of the *Listeria* survival curve cannot be adequately modelled by a static linear system.

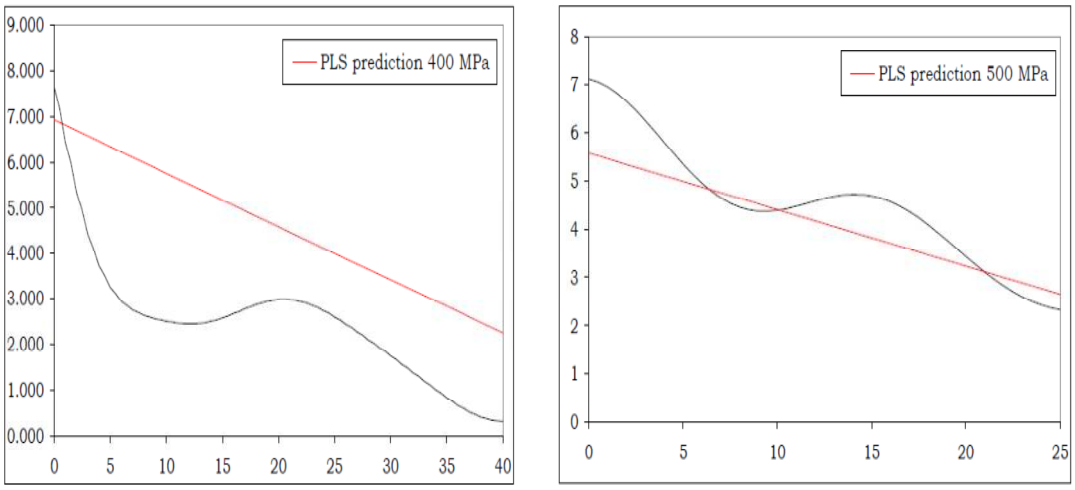


Fig 4.12 – PLS regression model on a two-input case

Following the principles of nonlinear identification, NARX models using the WNN-LCW, the MLP and the PLS schemes have been developed. The training dataset, consisting of 204 data from 350, 450, 550, and 600 MPa "continuous survival curves", was employed, while 81 data from 400MPa and 51 data from 500MPa curves were kept for validation.

The following structure has been adopted as a NARX model:

$$\mathrm{Count}(t) = f\left(\mathrm{Count}(t-1), \mathrm{Count}(t-2), \mathrm{MPa}, \mathrm{Sample\_Time}(t), \mathrm{Sample\_Time}(t-1), \mathrm{Sample\_Time}(t-2)\right) \quad (4.45)$$

During trials, it has been found that the model is sensitive to the previous number of bacteria counts, proving thus its dynamic behaviour.

In the proposed WNN-LCW, 25 wavelet Morlet functions have been used, while the network's learning parameter vector was $\lambda = [\eta_m, \eta_n, \eta_v, \zeta] = [0.001, 0.17, 0.17, 0.2]$. The hybrid parameter learning algorithm has been utilised, which resulted a high speed training process, i.e. less than 10 epochs. Figure 4.18d shows the performance of the WNN model, especially against the real experimental points from the training survival curves. The fitting performance of the developed WNN was comparable with the statistical models based on the comparison of the same indices (Table 4.1), as the root mean square error index ranged from 0.110 to 0.249, while the values of R^2 were also high (0.986-0.988). The high fitting performance of the WNN approach was expected as the network has been trained on these particular datasets. WNN and MLP schemes have been

implemented using MATLAB (ver. R2009, Mathworks). Results showed that the WNN was more effective in predicting the response of the pathogen compared with statistical models as illustrated by graphical plots (figure 4.13 and 4.14) implying that although the WNN has been trained on different survival curves, it has managed to learn the underlying process with high accuracy.

In a similar way, an MLP neural network using the classic backpropagation learning algorithm was constructed with the same input structure as WNN. Through trial and error, eventually two hidden layers with 12 and 8 nodes respectively have been employed. The learning algorithm was responsible for the network's slow convergence, which took approximately 5000 epochs. Figure 4.11& 4.12 illustrate the MLP performance for both testing survival curves.

The PLS-NARX scheme was certainly much more accurate from the previous simple PLS case. Like WNN and MLP, the PLS regression model was constructed to anticipate the dynamic nature of the specific problem, by including past values of the *Listeria* counts as inputs. The calculated by XLSTAT software, equation has the following form

$$Y_1 = 0.1646857 - 0.000172X_1 - 0.0055X_2 - 0.0055X_3 - 0.0055X_4 - 0.955X_5 + 1.9412X_6 \quad (4.46)$$

Figure 4.13 & 4.14 illustrate the PLS performance for both testing survival curves. With regard to the assessment of the quality of the overall model predictions various statistical criteria were calculated at all the tested validation experiments.

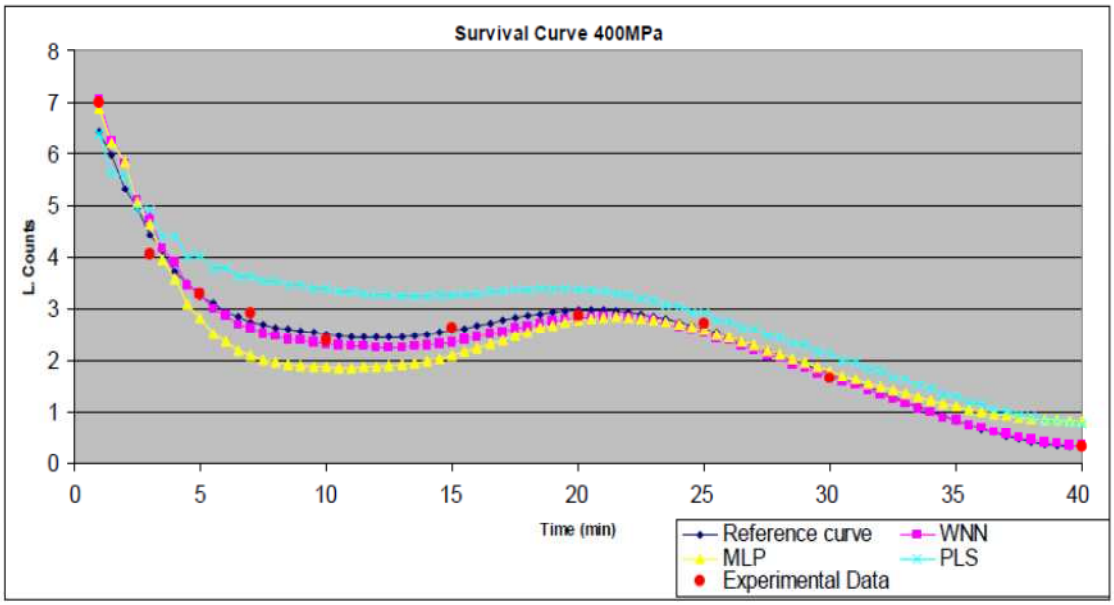


Fig 4.13- Survival curves of Listeria monocytogenes during high pressure treatment at 400MPa fitted with different modelling schemes

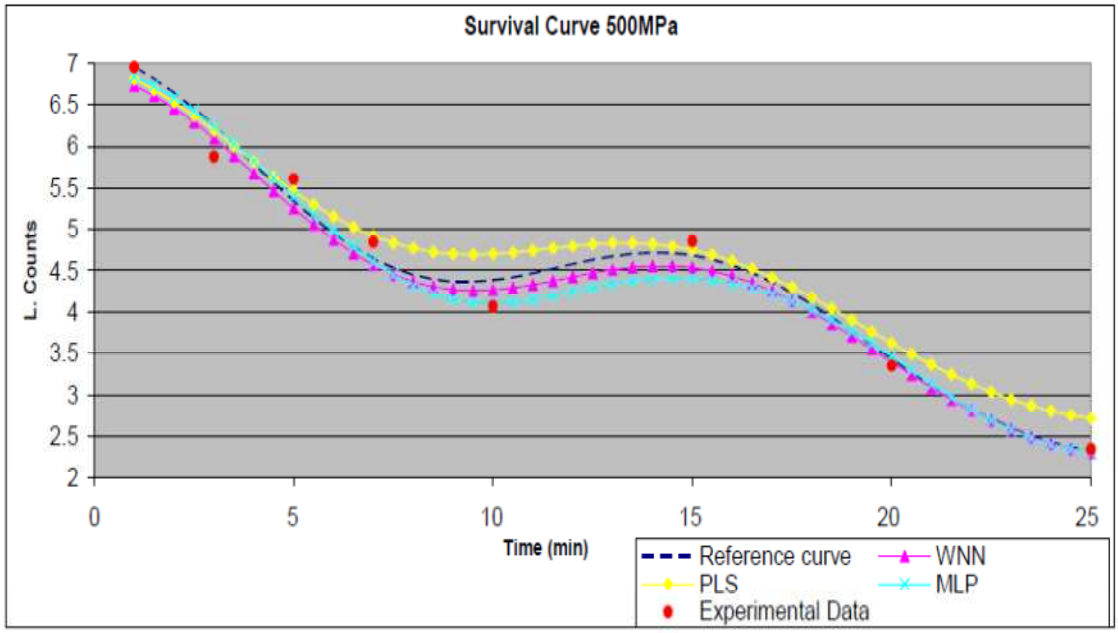


Fig 4.14- Survival curves of Listeria monocytogenes during high pressure treatment at 500 fitted with different modelling schemes

The regression coefficient (R^2) is often used as an overall measure of the prediction attained. It measures the fraction of the variation about the mean that is explained by a model. The higher the value ($0 \le R^2 \le 1$), the better is the prediction by the model[95]. The wavelet neural network developed herein was found to yield better agreement with experimental observations for the test data set compared to data predicted by the MLP and the PLS model. The values of the coefficient of determination (R^2), as shown in Table 4.3, indicate a very good fit of the experimental data from the WNN-based approach.

TABLE 4.3- Performance indices of various methods for *Lysteria Monocytogen* data

Statistical index	**Model**	**Testing Data sets**	
		400MPa	**500MPa**
Coefficient of determination (R^2)	MLP	0.9526	0.9933
	WNN	0.9935	0.9996
	PLS	0.9796	0.9966
Root mean square error (RMSE)	MLP	0.3830	0.1733
	WNN	0.1627	0.1128
	PLS	0.5532	0.2246
Mean absolute percentage error (MAPE) (%)	MLP	23.2939	2.7674
	WNN	5.3072	2.0750
	PLS	32.8082	5.3354
Mean Square Error	MLP	0.1467	0.0300
	WNN	0.0265	0.0127
	PLS	0.3061	0.0504
Standard error of prediction (SEP) (%)	MLP	15.9921	3.9301
	WNN	6.7934	2.5569
	PLS	23.1004	5.0930
Bias factor (B_f)	MLP	1.0177	0.9751
	WNN	0.9750	0.9792
	PLS	1.2960	1.0491
Accuracy factor (A_f)	MLP	1.2182	1.0288
	WNN	1.0551	1.0212
	PLS	1.2983	1.0525

However, R^2 is a suitable criterion for model comparison on the assumption that the error is normally distributed and not dependent on the mean value; In fact, the distribution of the error is not clearly known in the case of microbial/bacteria growth, so this term must be used with caution, particularly in non-linear regression models[96] and hence additional indices must be employed for model comparison.

The RMSE values of the WNN were also significantly better for the two "test" survival curves, i.e. 400MPa and 500MPa. This index is calculated between the desired and output values and then averaged across all data and it can be used as an estimation of the goodness of fit of the models. It can also provide information about how consistent the model would be in the long run . The RMSE values for both networks (WNN and MLP) were lower those from the linear PLS model, indicating the ability of non-linear networks to make better predictions on data for which there was no previous training.

The MAPE term provides information about the average deviation from the observed value. The relevant figures from Table 4.3 indicate again better performance for WNN. Especially, the high-nonlinear features of 400MPA curve proved to be difficult to be modelled from the PLS and MLP models. The SEP index is determined as the relative deviation of the mean prediction values and it has the advantage of being independent on the magnitude of the measurements [96]. Based on this index, the WNN scheme was superior from both MLP and PLS models for the two test curves.

The benefits of mathematical models to predict pathogen growth, survival and inactivation in foods include the ability to account for changes in microbial load in food as a result of environment and handling; the use of predictive microbiology in management of foodborne hazards. The usual measures of goodness-of-fit for model comparison in food microbiology is performed by calculating in addition to squared correlation coefficient (R^2) the bias (B_f) and accuracy (A_f) indices as proposed by Ross[92]. Bias factor is a multiplicative factor that compares model predictions and is used to determine whether the model over- or under-predicts the response time of bacterial growth. A B_f greater than 1.0 indicates that a growth model is fail-dangerous. Conversely, a B_f less than 1.0 generally indicates that a growth model is fail-safe. Perfect agreement between predictions and observations would lead to B_f of 1. The accuracy

factor (A_f), is a simple multiplicative factor indicates the spread of results about the prediction. A value of one indicates that there is perfect agreement between all the predicted and measured values. Table 4.3 also shows the bias and accuracy factor values obtained for the two testing survival curves. The B_f parameters for both WNN and MLP were superior to those of the PLS, however the WNN was just under the optimal 1.0, providing thus a fail-safe condition. The relevant figures for A_f indicate again better performances for the WNN scheme, which is more evident at the 400MPa survival curve. In order to further justify the plausibility of embedding Local Linear weights along with Hybrid learning algorithm, a comparison has been performed to verify the proposed scheme's performance over traditional WNNs. Table 4.4 and figure 4.15 & 4.16 illustrate the related results.

TABLE 4.4 – Convergence comparison of existing models on prediction problem

	Epochs (Ave.)	Independent Parameters.	Target Training MSE
WNN-LCW by GD only	721	375	0.00005
WNN-LCW Using Hybrid	7	375	0.00005
WNN-Static weights	1620	225	0.00005

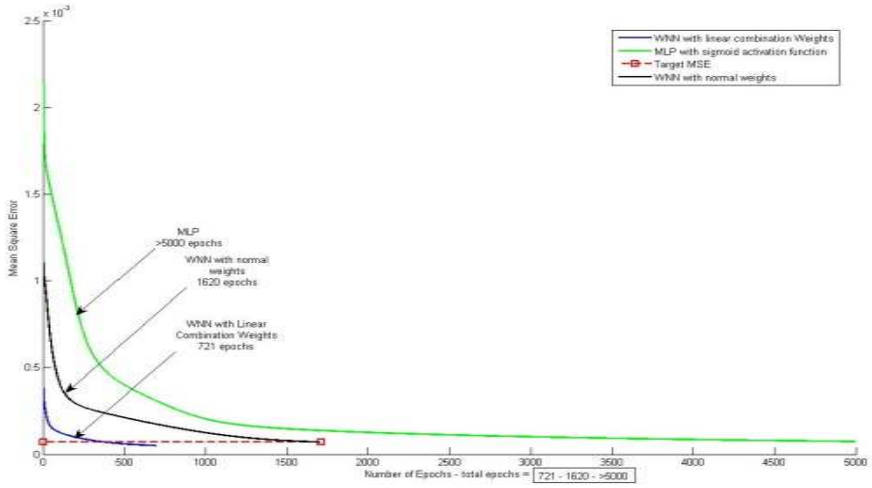


Fig 4.15- Convergence speed comparison by number of epochs using pure GD

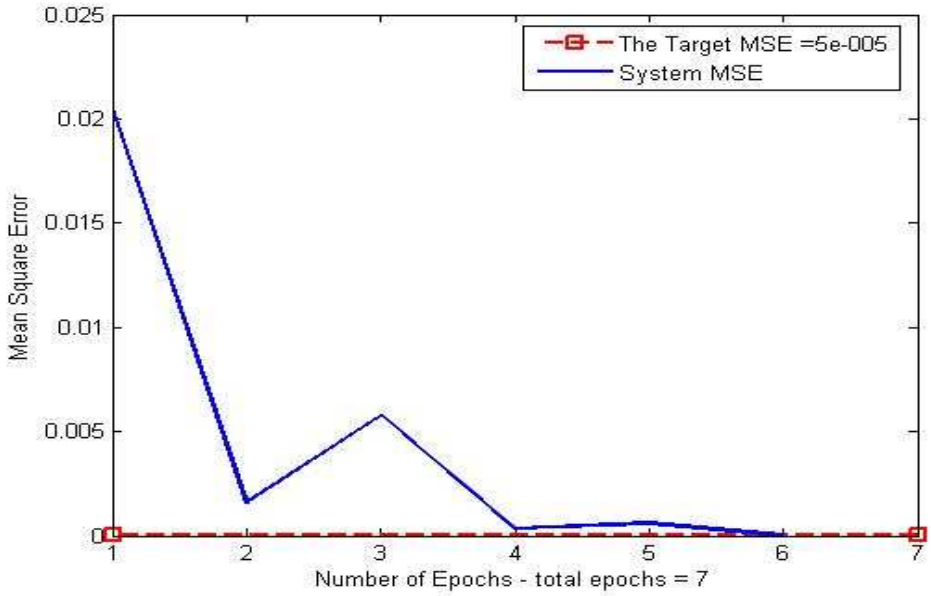


Fig 4.16 - Convergence speed using Hybrid Learning method

4.5 Proposed Structure Scheme II (MWNN-LCW)

For modelling the non-linear systems an alternative four-layered wavelet network structure is proposed herby, which is comprised of an input layer, hidden (wavelet) layer, product layer and

finally output layer. Referring to figure 4.17, Layer 1 accepts the input variables which are in form of $x = [x_1, x_2, ..., x_p]^T$, while Layer 2 is used to calculate the wavelet "membership" values. In this layer, each node performs a membership function and acts as an element for membership degree calculation, where a wavelet function is adopted as the membership function. Generally, a WNN approximates any desired signal y(t) by generalising a linear combination of a set of daughter wavelets $\varphi_{m,n}$ which are generated by step sizes dilation and translation and from a mother wavelet. We adopt the wavelet function as its node function in MWNN-LCW in form of $\varphi_{m,n} = \varphi(2^{-m}t - n)$. A modified differentiable version of Morlet wavelet deployed for j'th wavelet node connection to the p'th input data, expressed as

$$\varphi_{m^j_p, n^j_p}(x) = \cos(2\pi\beta(2^{-m^j_p}x - n^j_p))e^{\frac{-(2^{-m^j_p}x - n^j_p)^2}{\upsilon}} \tag{4.52}$$

This wavelet is derived from a function that is proportional to the cosine function and Gaussian probability density function. Its non-orthogonal, infinite support and maximum energy lies around origin with the narrow band [97]. The nodes of Layer 3 are regarded as the "wavelet" rules in association to the fuzzy rules in a neuro-fuzzy architecture. The number of the "wavelet" membership functions for each input variable is equal to the number of "wavelet" inference rules. These units are fixed, meaning that no modifiable parameter is associated with them. The multiplicative inference (Larsen product operator) has been used [14], thus the output of this inference layer is given by

$$\phi_j(x_p) = \prod_{p=1}^{P} \varphi_{m^j_p, n^j_p}(x_p) \tag{4.53}$$

The proposed approach differs from the conventional fuzzy rule table approaches. In those models, an input space is divided into $K_1 \times K_2 \times \cdots \times K_n$ fuzzy subspaces, where, $K_i, i = 1, 2, \cdots, n$ is the number of fuzzy subsets for the input variable. There is a fuzzy rule for each of these subspaces. The main drawback of that approach is that the number of fuzzy rules increases exponentially with respect to the number of inputs n. The fourth layer is connected to third layer via Linear Combination Weights. The difference of this proposed scheme compared to the previous one, includes the adoption of one extra layer, the multiplication layer. The proposed scheme has

similarities with the AFLS [16], in terms that this "multiplication" layer represents the fuzzy rules. The scheme has some interest, as it is desirable to minimise the number of "wavelet" function nodes, and this can be achieved by "clustering" them similarly as in the AFLS case.

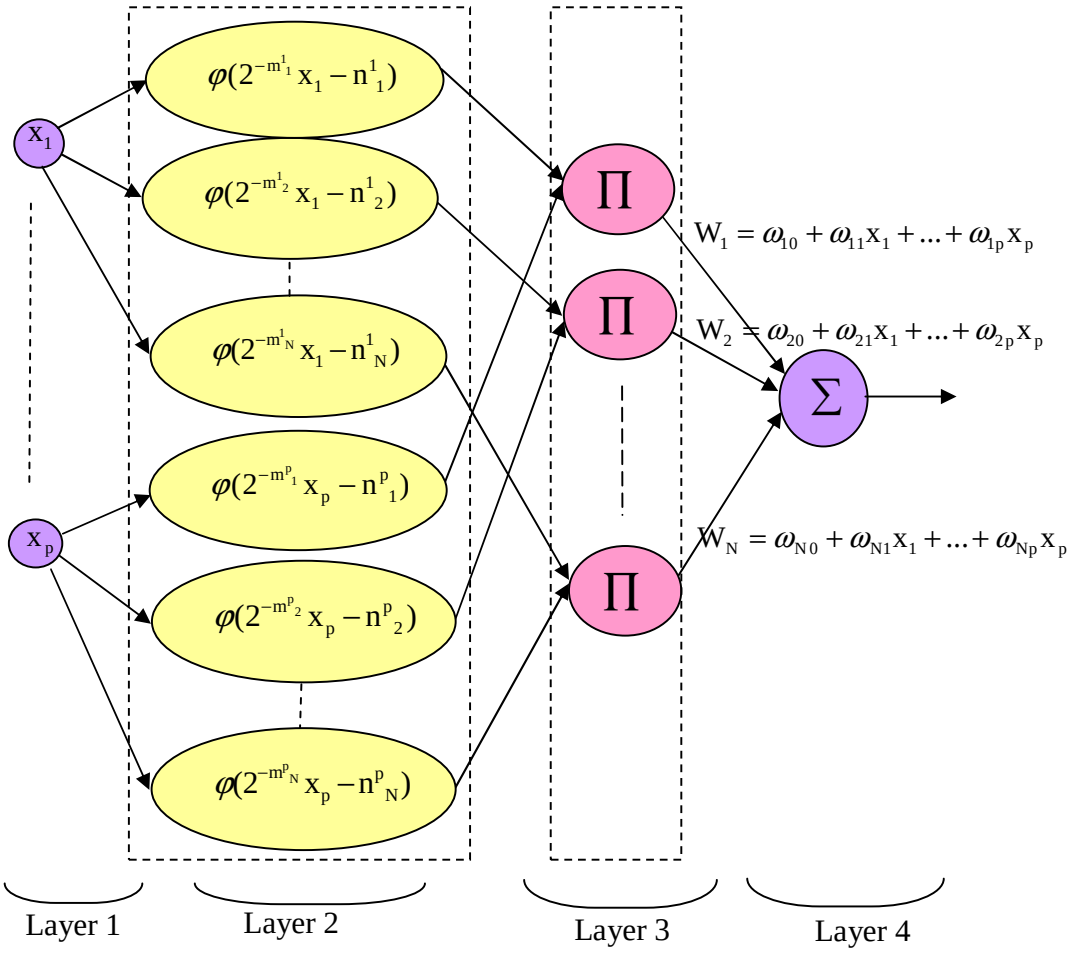


Fig 4.17 – Architecture of WNN with multiplication layer and Linear Weights

4.5.1 The parameter learning scheme

After the initial WNN is constructed, the parameters of the network are obtained via minimisation of the cost function E after a number of training epochs according to desired MSE. The linear parameters which are Linear Combination Weights have been updated exactly the same as the way described in *structure 1*. However the only minor difference is that the Φ matrix is generated

according to output of Multiplication layer. For non-linear parameters, in this case it is also used the steepest descending gradient method, hence, it is necessary to calculate the gradient vector $\frac{\partial E}{\partial p}$ for all the trainable non-linear parameters. m_j^p and n_j^p are the parameters that determine the location of the centre (translation) and the width (dilation) of the wavelets. Using the Gradient-based procedure we obtain the incremental updating algorithm of each parameter. Without any major change, just in this case we have one layer more, so the observed output for sample t is O_t^4. Therefore the updating equations derived from *proposed structure 1* can be modified accordingly. The updating expressions are in accordance to following rules

$$\begin{cases} \dfrac{\partial E}{\partial m_{pj}} = \dfrac{\partial E}{\partial O^4} \times \dfrac{\partial O^4}{\partial I^4} \times \dfrac{\partial I^4}{\partial O_j^3} \times \dfrac{\partial O_j^3}{\partial I_j^3} \times \dfrac{\partial I_j^3}{\partial O_{pj}^2} \times \dfrac{\partial O_{pj}^2}{\partial m_{pj}} \\ \dfrac{\partial E}{\partial n_{pj}} = \dfrac{\partial E}{\partial O^4} \times \dfrac{\partial O^4}{\partial I^4} \times \dfrac{\partial I^4}{\partial O_j^3} \times \dfrac{\partial O_j^3}{\partial I_j^3} \times \dfrac{\partial I_j^3}{\partial O_{pj}^2} \times \dfrac{\partial O_{pj}^2}{\partial n_{pj}} \end{cases} \tag{4.54}$$

$$\begin{cases} \dfrac{\partial E}{\partial m_{pj}} = -(y_t^d - O_t^4) \times f'_{out} \times W_j \times \dfrac{\prod_{i=1}^{p} \varphi_{pj}(x_p)}{\varphi_{pj}(x_p)} \times \dfrac{\partial \varphi_{pj}}{\partial m_{pj}} \\ \dfrac{\partial E}{\partial n_{pj}} = -(y_t^d - O_t^4) \times f'_{out} \times W_j \times \dfrac{\prod_{i=1}^{p} \varphi_{pj}(x_i)}{\varphi_{pj}(x_p)} \times \dfrac{\partial \varphi_{pj}}{\partial n_{pj}} \end{cases} \tag{4.55}$$

$p = 1,..,P \quad j = 1,...,N$

Where

$$\begin{cases} \dfrac{\partial \varphi_{pj}}{\partial m_{pj}} = x 2^{m_{pj}} \log(2) \sin(2\pi\beta(2^{m_{pj}} x - n_{pj})) e^{\frac{-(2^{m_{pj}} x - n_{pj})^2}{\upsilon}} - 2\cos(2\pi\beta(2^{m_{pj}} x - n_{pj})) \dfrac{x 2^{m_{pj}} \log(2)(2^{m_{pj}} x - n_{pj})}{\upsilon} e^{\frac{-(2^{m_{pj}} x - n_{pj})^2}{\upsilon}} \\ \dfrac{\partial \varphi_{pj}}{\partial n_{pj}} = 2\pi\beta \sin(2\pi\beta(2^{m_{pj}} x - n_{pj})) e^{\frac{-(2^{m_{pj}} x - n_{pj})^2}{\upsilon}} + \dfrac{2(2^{m_{pj}} x - n_{pj})}{\upsilon} \cos(2\pi\beta(2^{m_{pj}} x - n_{pj})) e^{\frac{-(2^{m_{pj}} x - n_{pj})^2}{\upsilon}} \end{cases} \tag{4.56}$$

Following the same procedure mentioned for updating m_{pj} and n_{pj}, here also we can use expressions (4.31) and (4.32).

4.5.2 Case Analysis and Simulation Results

In this section, the proposed model was applied to the same Milk-Listeria dataset, of course after imposing the one-step-ahead-prediction defined earlier. The inputs included the type of pressure level and the sampling time-step, while the output was related to the bacteria counts. Each "continuous survival curve" has been verified against the real experimental samples. Based on these continuous datasets, the capabilities of the proposed MWNN-LCW architecture has been verified as a one-step-ahead prediction system. Pressure levels of 400 MPa and 500 MPa have been used as testing datasets, while the remaining levels as training ones.

Following the principles of nonlinear identification, NARX models using the MWNN-LCW, the MLP, the RBF and the Elman recurrent networks have been developed and comparative studies have been conducted. The training dataset, consisting of 204 data from 350, 450, 550, and 600 MPa "continuous survival curves", was employed, while 81 data from 400MPa and 51 data from 500MPa curves were kept for validation.

During trials, it has been found that the model is sensitive to the previous number of bacteria counts, thus proving its dynamic behaviour. In the proposed MWNN-LCW, 6 Morlet wavelet functions have been used, while the network's learning parameter vector was $\lambda = [\eta_m, \eta_n, \zeta] = [0.1, 0.1, 0.2]$. The hybrid parameter learning algorithm has been utilised, which resulted a quick training of 5 epochs. Figure 4.18d shows the performance of the WNN model, especially against the real experimental points from the training survival curves. Following the principles of nonlinear identification, NARX models using the MWNN-LCW, the MLP, the RBF and the Elman recurrent networks have been developed. The training dataset, consisting of 204 data from 350, 450, 550, and 600 MPa "continuous survival curves", was employed, while 81 data from 400MPa and 51 data from 500MPa curves were kept for validation. The following structure has been adopted as a NARX model:

$$\mathrm{Count}(t) = f\left(\mathrm{Count}(t-1), \mathrm{Count}(t-2), \mathrm{MPa}\right)$$

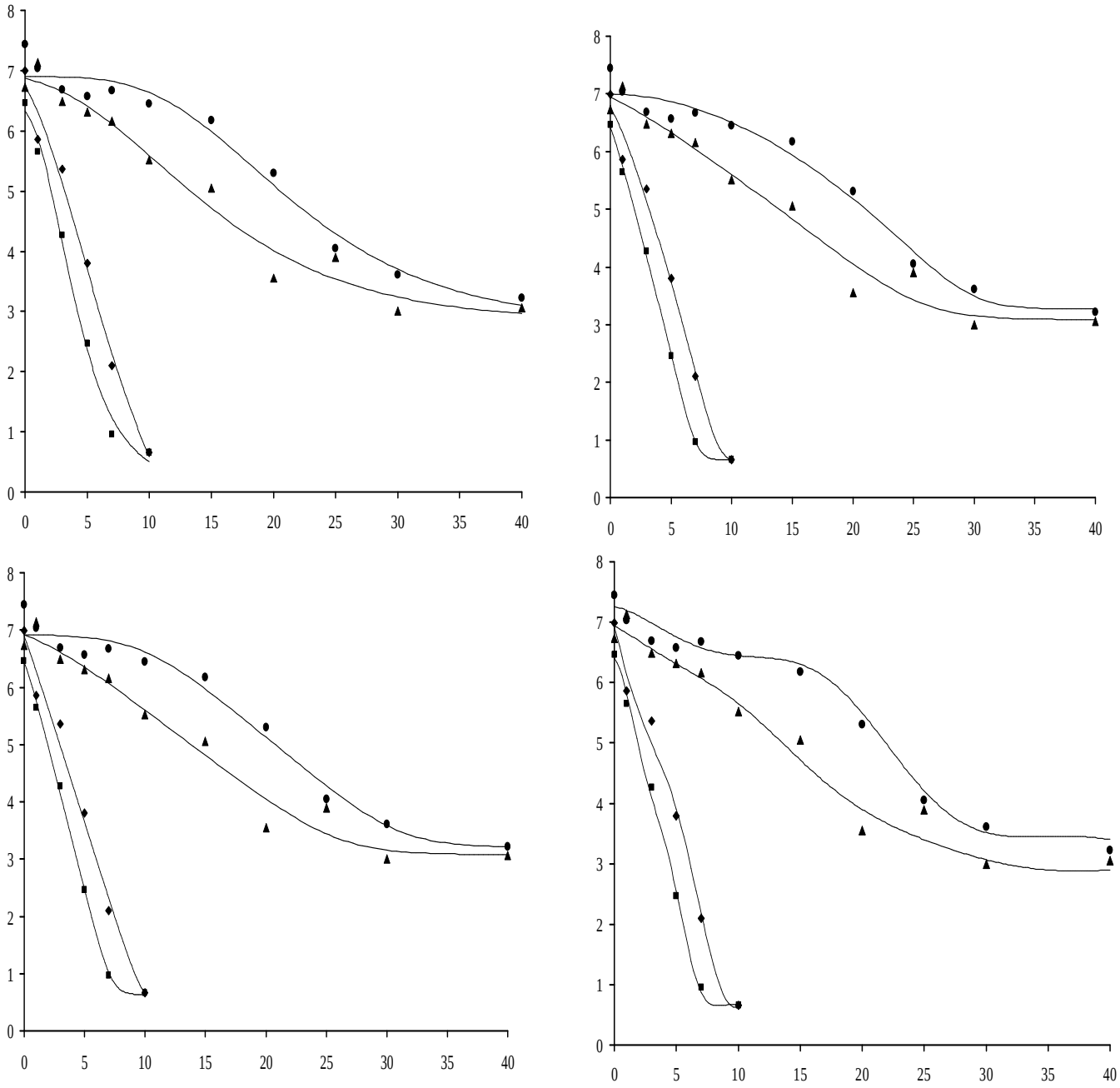


Fig 4.18 - Survival curves of *Listeria monocytogenes* in UHT whole milk during high pressure processing at 350 MPa (•), 450 MPa (▲), 550 MPa (◆), and 600 MPa (■), generated by the re-parameterised Gompertz model (a), the modified Weibull model (b), the Geeraerd model (c), and the wavelet neural network (d). Data points are mean values of two independent experiments with two replications each

The fitting performance of the developed WNN was comparable with the statistical models based on the comparison of the same indices (Table 1), as the root mean square error index ranged from 0.054 to 0.124, while the values of R^2 were also high (> 0.989). The high fitting performance of the WNN approach was expected as the network has been trained on these particular datasets. Results showed that the WNN was more effective in predicting the response of the pathogen compared with statistical models as illustrated by graphical plots (figure 4.19, 4.20), implying that although the WNN has been trained on different survival curves, it has managed to learn the underlying process with high accuracy.

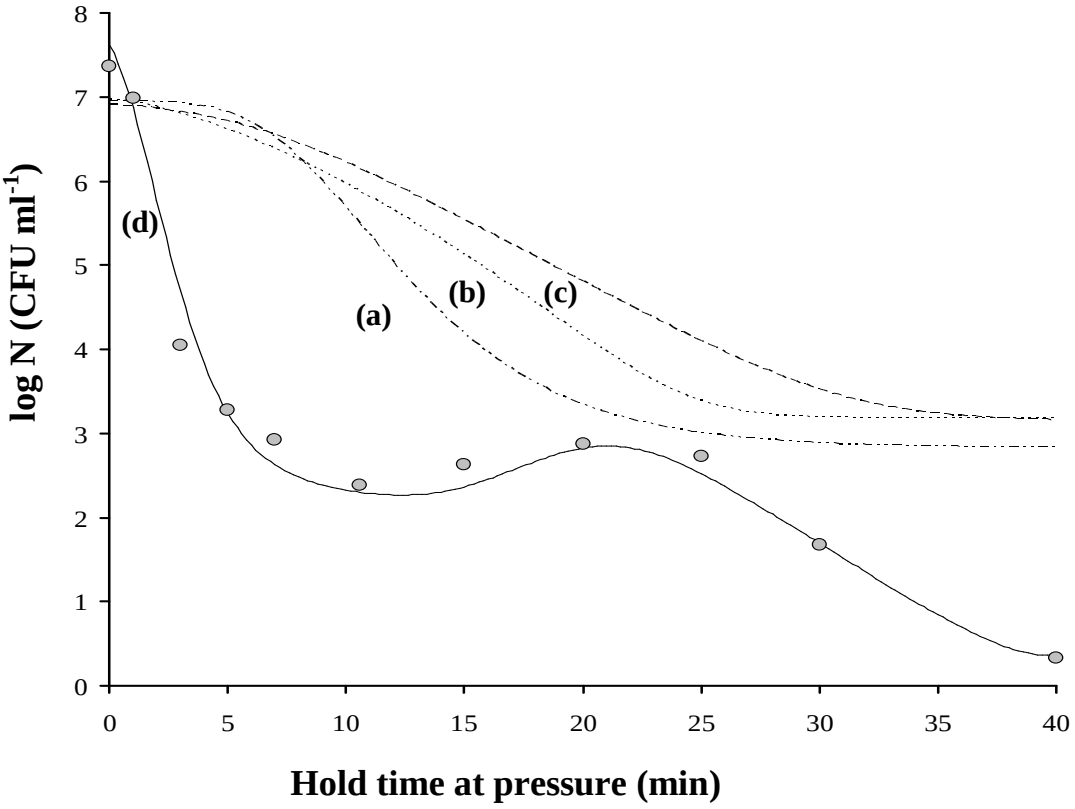


Fig 4.19: Observed values and predicted survival curves of *Listeria monocytogenes* in UHT whole milk during high pressure treatment at 400 MPa, generated by the re-parameterised Gompertz model (a), the modified Weibull model (b), the Geeraerd model (c), and the wavelet neural network (d). Data points are mean values of two independent experiments with two replications each.

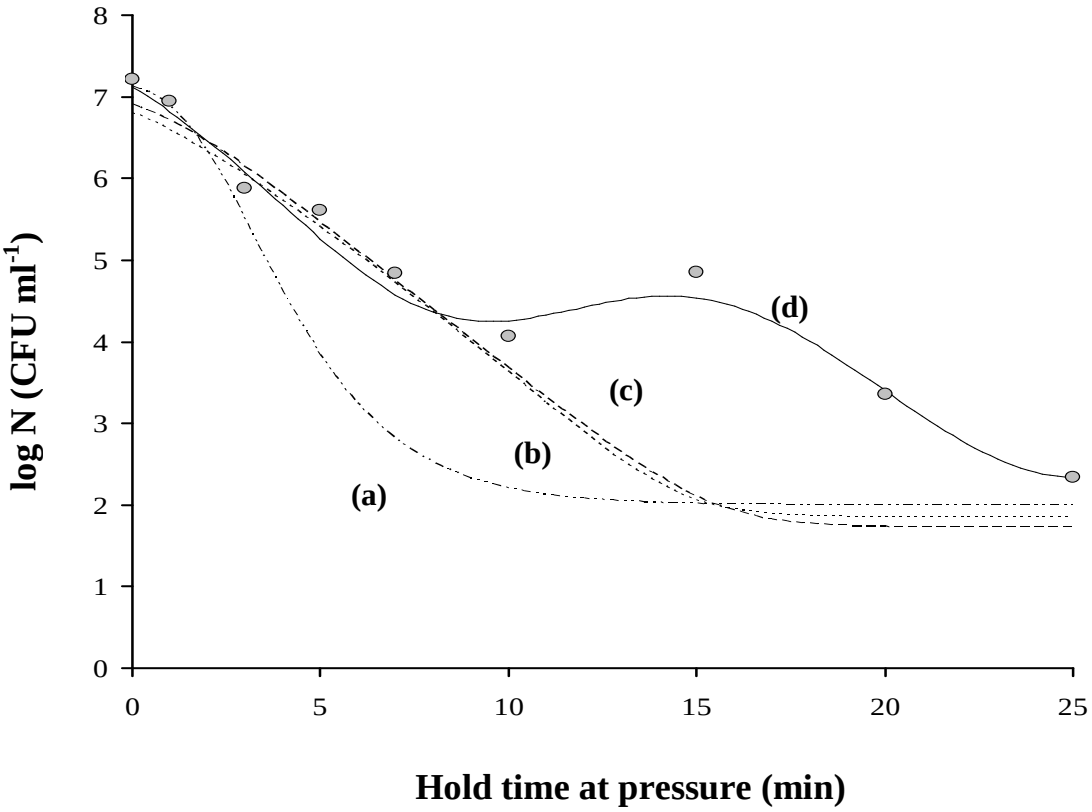


Fig 4.20 - Observed values and predicted survival curves of *Listeria monocytogenes* in UHT whole milk during high pressure treatment at 500 MPa, generated by the re-parameterised Gompertz model (a), the modified Weibull model (b), the Geeraerd model (c), and the wavelet neural network (d). Data points are mean values of two independent experiments with two replications each.

Two other approaches, based on neural network technology, the RBF and MLP schemes were constructed with the same input structure as the WNN. These two well-established networks are known for their generalisation capabilities despite the fact they have different learning strategies (global vs. local). An RBF network tends to converge rapidly compared with the MLP one. The RBF network based on the OLS algorithm contained 30 Gaussian nodes in the hidden layer and one spread parameter σ for all input variables $(\sigma = 0.25)$. In contrast to RBF, the MLP network structure consisted of two hidden layers (12 and 6 nodes for each hidden layer) and a single sigmoidal output node. The learning algorithm was responsible for the MLP's slow convergence, which took approximately more than 5000 epochs. Figure 4.21 & 4.22 illustrate the MLP and RBF performances for both testing survival curves.

The use of dynamic neural networks presents an alternative solution to the specific problem. Here, the focus was to use one dynamic network (Elman) that was given some kind of memory to encode past history, with the additional requirements of short training time. The improved, compared to the standard MLP structures, results reveal the advantages of using memory neuron structures. The inclusion of memories and the related recurrence in the first hidden layer, enable the network to carry out accurate predictions. Although this method is dependent on the number of "memories" in the "recurrent" nodes and therefore it can be considered as a partially recurrent network, it proved to be one faster in training time than the MLP scheme. In this specific Elman network, 8 and 4 nodes have been used for the two hidden layers. Figure 4.21 & 4.22 illustrate the Elman network's performance for both testing survival curves.

The relevant figures from Table 4.5 indicate again an improved performance for WNN. Especially, the high-nonlinear features of 400MPA curve proved to be difficult to be modelled from the other models.

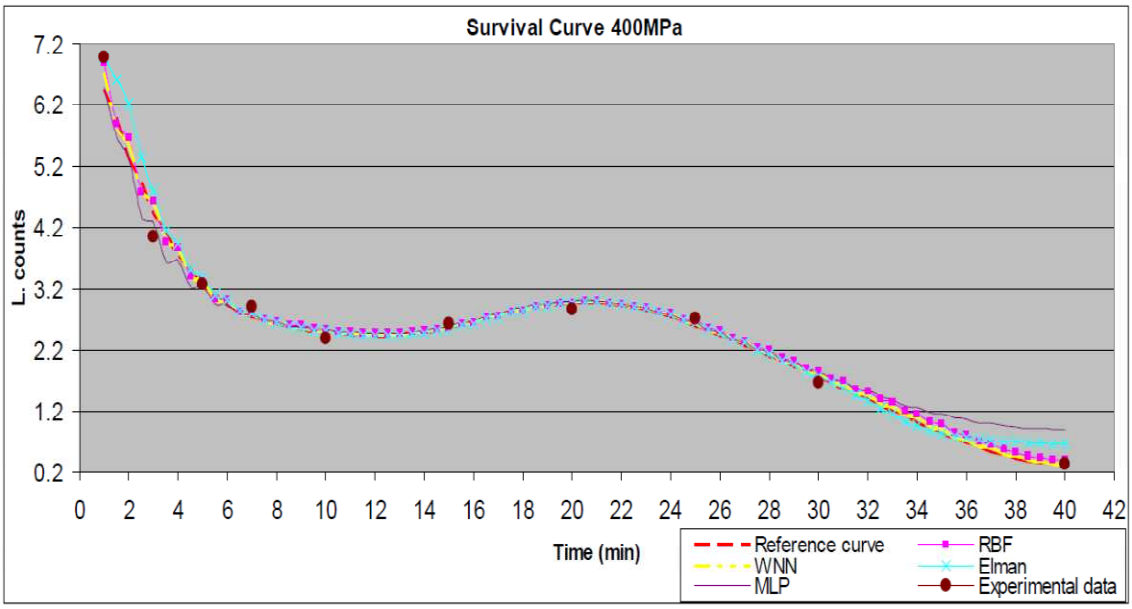


Fig 4.21 - Survival curves of Listeria monocytogenes during high pressure treatment at 400MPa fitted with different modelling schemes

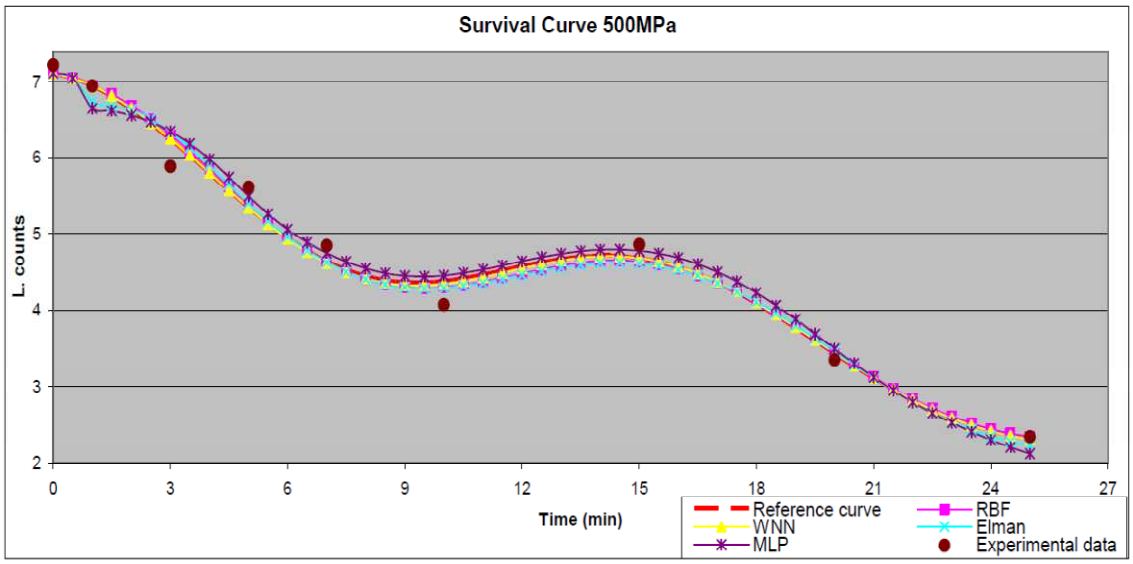


Fig 4.22 - Survival curves of Listeria monocytogenes during high pressure treatment at 500MPa fitted with different modelling schemes

TABLE 4.5 – MWNN-LCW and other methods Statistical index comparison.

	Model	**Testing Data sets**	
		400MPa	**500MPa**
Coefficient of determination (R_2)	MLP	0.9937	0.9963
	MWNN	0.9985	0.9999
	RBF	0.9975	0.9992
	Elmann NN	0.9926	0.9983
Root mean square error (RMSE)	MLP	0.2126	0.1151
	MWNN	0.0670	0.0198
	RBF	0.1014	0.0494
	Elmann NN	0.1800	0.0761
Mean absolute percentage error (MAPE) (%)	MLP	17.8272	2.4127
	MWNN	2.0035	0.3659
	RBF	5.3391	0.8871
	Elmann NN	8.9175	1.4507
Mean Square Error	MLP	0.0452	0.0132
	MWNN	0.0045	0.0003912
	RBF	0.0103	0.0024
	Elmann NN	0.0324	0.0058
Standard error of prediction (SEP) (%)	MLP	8.8778	2.6101
	MWNN	2.7967	0.4485
	RBF	4.2339	1.1209
	Elmann NN	7.5142	1.7250
Bias factor (B_f)	MLP	1.1186	1.0084
	MWNN	1.0137	0.9982
	RBF	1.0469	0.9973
	Elmann NN	1.0614	0.9911
Accuracy factor (A_f)	MLP	1.1336	1.0244
	MWNN	1.0198	1.0037
	RBF	1.0512	1.0089
	Elmann NN	1.0733	1.0147

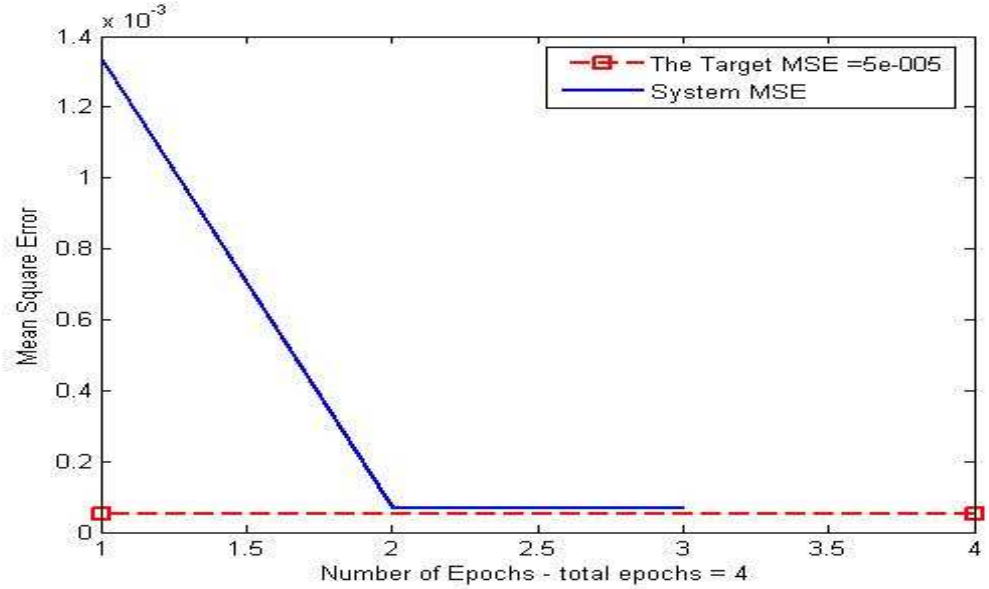


Fig 4.23 - Epochs using Hybrid Method for MWNN structure

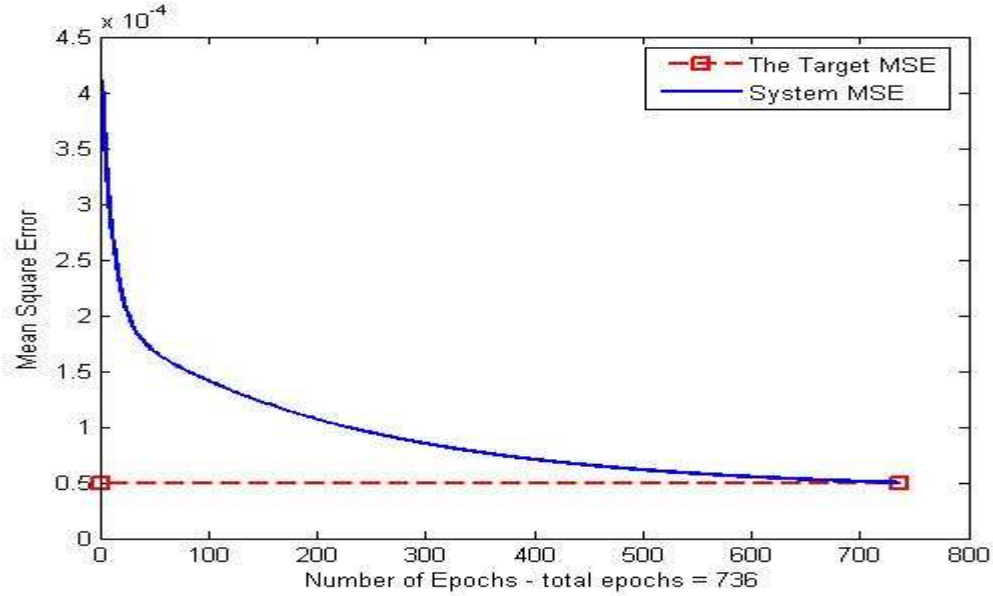


Fig 4.24- Epochs using only Gradient Descent for MWNN structure

TABLE 4.6 –Convergence comparison of existing models on prediction problem

	Epochs (Ave.)	Independent Parameters	Target Training MSE
MWNN-LCW using GD only	721	375	0.00005
WNN-LCW Using Hybrid method	7	375	0.00005
MWNN-LCW With Multiplication Layer using Hybrid Method	4	114	0.00005
MWNN-LCW With using GD only	736	475	0.00005
MWNN-Static Weights	1620	225	0.00005

The wavelet specifications are listed in Table 4.7. As we know three inputs, out of six inputs, are simply delayed version of the other inputs so it can be easily justified that why the corresponding wavelets in input 1,2 and 3- after parameter learning- end up with almost similar scales and translations. The scales input 5 and 6 (of course input 5 and input 6 are both delayed versions of output) remain the same throughout the nodes which may imply the fact that there is only one dominant frequency in them.

TABLE 4.7– Wavelet Parameters of MWNN-LCW after Optimisation

	Dilation						**Translation**					
Node	Input1	Input2	Input3	Input4	Input5	Input6	Input1	Input2	Input3	Input4	Input5	Inpu6
1	-0.8307	0.28677	0.	0.28677	-3.01	-3.0379	0.43937	0.80117	0.80117	0.80117	1.546	1.5645
2	-1.5168	-0.0041	-0.0041	-0.0041	-3.01	-3.0378	0.43815	0.49301	0.49301	0.49301	1.546	1.5648
3	-1.5166	-0.009369	-0.0093	-0.0093	-3.01	-3.0379	0.44046	0.78455	0.78455	0.78455	1.546	1.5644
4	-0.8302	0.26699	0.26699	0.26699	-3.01	-3.0379	0.43888	0.48505	0.48505	0.48505	1.546	1.5644
5	-1.5186	0.27192	0.27192	0.27192	-3.01	-3.0379	0.44015	0.794	0.794	0.794	1.546	1.5644
6	-0.8290	0.05435	0.05435	0.05435	-3.01	-3.0378	0.72172	0.48495	0.48495	0.48495	1.546	1.5647

Chapter 5

Data Clustering Techniques

The world we are living is saturated with overwhelming amount of data. On daily basis, people encounter a large amount of information to store or for further analysis and management. One of the vital means in dealing with these data is to classify or group them into a set of categories or clusters[98]. Clustering is an effective approach to identification of complex non-linear systems by partitioning the available data into subsets and approximate each subset by a simple model. Clustering techniques are among the unsupervised (learning) methods, since they do not use prior class identifiers. Most clustering algorithms also do not rely on assumptions common to conventional statistical methods, such as the underlying statistical distribution of data, and therefore they are useful in situations where little prior knowledge exists.

The aim of clustering is furnished by gathering the objects are more similar to each other in one cluster. The term "similarity" in many cases considered as a *distance norm* from a data vector to a prototype object called as centre[99, 100]. The concept of dissimilarity (or distance) is the essential component of any form of clustering that helps us navigate through the data space and form clusters. By computing dissimilarity, we can sense and articulate how close together two patterns are and, based on this closeness, allocate them to the same cluster[101].

While most classical clustering algorithms assign each datum to exactly one cluster, thus forming a crisp partition of the given data, fuzzy clustering allows for degrees of membership, to which the transitions of the subsets are gradual rather than abrupt (soft membership). In fuzzy clustering, instead of determining whether or not an event occurs, as is the case with probability, fuzziness measures the degree to which an event occurs. Thus the membership degree shared among various

clusters. This creates the concept of fuzzy boundaries which differs from the traditional concept of well-defined boundaries[102].

5.1 Fuzzy Partitions

The objective of clustering is to partition the data set X into C clusters. For the time being, let us assume that C is known, based on prior knowledge. In every fuzzy clustering method there is a fuzzy partition matrix $\gamma=[\gamma_{ni}]_{N\times C}$ which demonstrates the degree of membership of each sample to cluster c. γ_{ni} are the values of the membership function of the i-th fuzzy subset of X_n we assume that γ_{ni} are constrained labels satisfying

$$\begin{aligned} 0 \le \gamma_{ji} \le 1 \\ \sum_{i=1}^{C} \gamma_{ji} = 1 \\ 0 < \sum_{j=1}^{N} \gamma_{ji} < N \end{aligned} \tag{5.1}$$

Where the latter, means no existence of empty clusters [100].
Most fuzzy clustering algorithms are objective function based: they determine an optimal (fuzzy) partition of a given data set X = { x_j | j = 1, . . . , N} into c clusters by minimizing an objective function.

$$J(X,\Upsilon,C)=\sum_{i=1}^{C}\sum_{j=1}^{N}\gamma_{ji}^{s}d_{ji} \tag{5.2}$$

where d_{ji} is the distance between datum x_j and cluster i. The parameter s, s > 1, is called the fuzzifier or weighting exponent. It determines the “fuzziness” of the classification: with higher values for s the boundaries between the clusters become softer, with lower values they get harder[103]. The value of the cost function (eq 5.2) is a measure of the total weighted within-group squared error incurred by the presentation of the C clusters normally defined by their prototype μ_i. Statistically, eq (5.2) can be considered as a measure of the total variance of each data vector x from μ_i.

This approach is usually called probabilistic fuzzy clustering, because the membership degrees for a datum formally resemble the probabilities of its being a member of the corresponding clusters. The minimization of the eq(5.2) represents a non-linear optimization problem that can be solved using a variety of available methods. Some of the most popular methods presented in this chapter.

5.2 Distance Norms

The distance measure D in 5.2 has the general format of

$$d = (x - \mu_i)^T \Sigma^{-1} (x - \mu_i) \tag{5.3}$$

The shape of the clusters is determined by the certain Σ in distance measure (eq 5.3). A common choice is $\Sigma = I$, which induces the standard Euclidean norm:

$$d = (x - \mu_i)^T (x - \mu_i) \tag{5.4}$$

The Euclidean norm induces hyperspherical clusters, i.e., clusters whose surface of constant membership are hyperspheres. Σ can be selected as a $P \times P$ diagonal matrix that accounts for different variances in the orientations of the coordinate axes of X. In this case, Matrix induces a *Diagonal* norm on $\mathbb{R}^P$.

$$\Sigma_D = \begin{bmatrix} \sigma_1^2 & 0 & \cdots & 0 \\ 0 & \sigma_2^2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \vdots & \sigma_P^2 \end{bmatrix} \tag{5.5}$$

Finally, Σ can be realized as the inverse of a $P \times P$ Covariance Matrix of X

$$\Sigma_M = \frac{1}{N} \sum_{j=1}^{N} (X_j - \bar{X})(X_j - \bar{X})^T \tag{5.6}$$

The $\bar{X}$ shows the sample mean of data. In this case Σ is the *Mahalanobis* norm on $\mathbb{R}^P$.

Both the diagonal and the *Mahalanobis* norm generate hyperellipsoidal clusters, the difference is that with the diagonal norm, the axes of the hyperellipsoids are parallel to the coordinate axes while with the *Mahalanobis* norm the orientation of the hyperellipsoids is arbitrary.

5.3 Fuzzy C-Mean Clustering

Fuzzy c-means is one of the most commonly used fuzzy clustering techniques for different degree estimation problems. FCM determines each cluster location using maximum membership defuzzification and neighbourhood smoothing techniques. FCM employs two simple and straightforward statistical features, namely mean and standard deviation. This method developed by Dunn in 1973[104] and improved by Bezdek in 1981 [105], proposes a generalisation by means of a family of objective function and is frequently used in pattern recognition. It is based on the minimisation of the following objective function:

$$J(X, \Upsilon, C) = \sum_{i=1}^{C} \sum_{j=1}^{N} \gamma_{ji}^{s} \left\| \mu_i - x_j \right\| \qquad (5.7)$$

As denoted, Fuzzy C-Mean Clustering uses Euclidian distance in its cost function. All parameters are all described earlier and $\|.\|$ is any norm expressing the similarity between any measured data and the centre. The centroid of a cluster is the mean of all points, weighted by their degree of belonging to the cluster:

$$\mu_i = \frac{\sum_{j=1}^{N} \gamma_{ij}^{s} . x_j}{\sum_{j=1}^{N} \gamma_{ij}^{s}} \qquad (5.8)$$

This iteration will stop when $\{| \gamma_{ij}^{k+1} - \gamma_{ij}^{k} |\} < \varepsilon$, where ε is a termination criterion between 0 and 1, whereas k are the iteration steps. The degree of belonging is related to the inverse of the distance to the cluster, then the coefficients are normalised and fuzzified with a real parameter $s > 1$ so that their sum is 1.

$$\gamma_{ij} = \frac{1}{\sum_{k=1}^{C} \left(\frac{\left\| x_j - \mu_i \right\|}{\left\| x_j - \mu_k \right\|} \right)^{\frac{2}{s-1}}} \tag{5.9}$$

Several investigations have been made on the best value to choose for the fuzzification exponent, s, which is chosen a priori. A recent study[106] concludes empirically that m = 2.0 is a "good" value. For s equals to 2, this is equivalent to normalising the coefficient linearly to make their sum one. The algorithm determines the following steps:

Step 1. Randomly initialising the membership matrix γ.

Step 2. Calculating the centroid c_i *by using eq (5.8).*

Step 3. Compute dissimilarity between centroids and data points using eq(5.4). Stop if its improvement over previous iteration is below a threshold.

Step 4. Compute a new γ *using eq(5.9). Go to step 2.*

The FCM algorithm has proven to be a very popular method of clustering for many reasons. In terms of programming implementation, it is relatively straightforward. It employs an objective function that is intuitive and easy-to-grasp. Because of its fuzzy basis, it performs robustly: it always converges to a solution, and it provides consistent membership values.

FCM strength over the famous K-Means algorithm[107] is that, given an input point, it yields the points membership value in each of the classes. On the other hand the weaknesses are:

- It requires the number of clusters to look for to be known as a priori
- Initialisation
- If the iterative algorithm commonly employed for finding solutions of the FCM objective function is used, it may find more than one solution depending on the initialisation. This relates to the general problem of local and global optimisation.
- Fuzzy C-Means (FCM) clustering method discovers spherical clusters with equal volumes and density. However, in a number of real data problems as performance analysis, time-series data as well as some forecasting and modelling tasks the identified clusters are not

spherical as they are presumed to be characterized with a different shape and orientation in the space.

- Its accuracy is sensitive to noise and outliers.

5.4 Gustafson – Kessel Clustering

The Gustafson-Kessel (GK) algorithm[108] is another powerful clustering technique with a large number of applications in various domains including image processing, classification and system identification. FCM algorithm as already mentioned uses point prototypes and an Euclidian norm-induced distance measure. As a consequence, its performance is acceptable only when the data set contains clusters that are well-apart or when clusters of approximately the same size and shape, whereas,

GF extended the standard fuzzy C-mean algorithm by employing and adaptive *Mahalonobis* distance norm, in order to detect clusters of different geometrical shape by estimating the cluster covariance matrix. In addition it is relatively insensitive to the data scaling and initialization of the partition matrix[109]. The Gustafson–Kessel algorithm is based on iterative optimization of an objective functional very similar to c-means type:

$$J(X, \Upsilon, C, \{A_i\}) = \sum_{i=1}^{C} \sum_{j=1}^{N} \gamma_{ji}^{s} (\mu_i - x_j)^T \Sigma_i (\mu_i - x_j) \tag{5.10}$$

In this objective function, the number of clusters has to be fixed in advance. The distance norm $d_{ij\{\Sigma_i\}}$ can account for clusters of different topology[110]. This algorithm is capable of detecting ellipsoidal cloud clusters of dissimilar sizes and orientations. The minimization of the GK objective functional is obtained by using the optimization method according to the following popular algorithm

Step 1. Computing the cluster centres (prototypes)

$$\mu_i = \frac{\sum_{j=1}^{N} (\gamma_{ij}^{k-1})^s x_j}{\sum_{j=1}^{N} (\gamma_{ij}^{k-1})^s}, \qquad 1 \le i \le C \tag{5.11}$$

Step 2. Compute the cluster covariance matrices

$$\Sigma_i = \frac{\sum_{j=1}^{N} (\gamma_{ij}^{k-1})^s (\mu_i^k - x_j)^T (\mu_i^k - x_j)}{\sum_{j=1}^{N} (\gamma_{ij}^{k-1})^s} \tag{5.12}$$

The matrix Σ_i determines the shape and orientation of the selected cluster. Thus, the GK algorithm employs an adaptive distance norm unique for every cluster as the norm inducing matrix Σ_i is calculated by estimates of the data covariance:

Step 3. Compute the distances

$$d_{ij\{\Sigma_i\}} = (x_j - \mu_i^k)^T (x_j - \mu_i^k) [\frac{\rho_i |\Sigma_i|^{1/P}}{\Sigma_i}] \tag{5.13}$$

Without any prior knowledge, the cluster volumes ρ_i are simply fixed at one for each cluster.

Step 4. Update the partition matrix

The GK algorithm like other FCM-based clustering algorithms utilises the Lagrange multiplier method to minimize the cost function. It iteratively determines the membership degree

$$\gamma_{ij}^k = \frac{1}{\sum_{l=1}^{C} \left(\frac{d_{ij\{\Sigma_i\}}}{d_{lj\{\Sigma_i\}}} \right)^{\frac{2}{s-1}}} \tag{5.14}$$

The algorithm runs until $\left\| \gamma^k - \gamma^{k-1} \right\| < \varepsilon$.

The GK suffers from a numerical problem mostly occurs in Step 3 of the algorithm, where the cluster covariance matrix Σ_i is inverted. In case of small number of data samples or when the data inside a cluster are linearly correlated, the covariance matrix may approach to singularity. Under this scenario, the computed covariance matrix is not a reliable estimate of the underlying data distribution.

5.5 Gath-Gava Clustering

The algorithm by Gath and Geva (GG)[111] is an extension of the Gustafson-Kessel(GK) algorithm that also takes into account the size and density of the clusters [107]. GG clustering algorithm are also based on minimization of the aforementioned objective function which its parameters have been explained in the previous section.

The most important part of objective function J, which is the characteristic of different fuzzy clustering methods, is the distance function D_{ij}. GG assumes that the i'th Gaussian distribution with expected value c_i and covariance matrix Σ_i is chosen for generating a datum, with a priori probability P_i ,hence, in the GG method d_{ij} - distance of the jth data point from the ith cluster - is defined as follows :

$$d_{ij} = \frac{P_i}{\sqrt{\det(\Sigma_i)}} \exp\left(\frac{1}{2}(x_j - \mu_i)^T \Sigma_i^{-1}(x_j - \mu_i)\right) \tag{5.15}$$

where the parameters of each cluster, μ_i and Σ_i are centre and covariance respectively of i'th cluster. P_i is the priori probability, known also as the coefficient designed for eliminating the sensitivity of the algorithm to number of data points in different clusters which is computed by the following formula

$$P_i = \frac{\sum_{j=1}^{N} \gamma_{ij}^s}{\sum_{i}^{C}\sum_{j}^{N} \gamma_{ij}^s} \tag{5.16}$$

Minimization of the objective function with respect to membership degree by considering the fact that sum of membership values of a data point to all clusters becomes one, leads to the following equation for computing γ_{ij}:

$$\gamma_{ij} = \frac{(\frac{1}{d_{ij}})^{\frac{1}{s-1}}}{\sum_{i=1}^{C}(\frac{1}{d_{ij}})^{\frac{1}{s-1}}} \tag{5.17}$$

In the GG algorithm, centre and covariance matrix of clusters and membership degree of data points are estimated in the following iterative process[112] :

Step 1.Choose number of clusters, initial values of centre and covariance matrix for each cluster.
Step 2. Calculate distances of data points to all clusters using eq(5.12).
Step 3. Compute degree of membership for all data points using eq(5.14).

Step 4. Estimate centre and covariance matrix for each cluster using the following equations.

$$\mu_i = \frac{\sum_{j=1}^{N} \gamma_{ij}^{s} x_j}{\sum_{j=1}^{N} \gamma_{ij}^{s}}$$

$$\Sigma_i = \frac{\sum_{j=1}^{N} \gamma_{ij}^{s} (x_j - \mu_i)(x_j - \mu_i)^T}{\sum_{j=1}^{N} \gamma_{ij}^{s}} \qquad (5.18)$$

Step 5. Go to the second step until a termination criterion satisfies.

5.6 Subtractive Clustering

Clustering algorithms typically require the user to pre-specify the number of cluster centres and their initial locations; the locations of the cluster centres are then adapted in a way such that these can better represent a set of data points covering the range of data behaviour. Fuzzy Subtractive approach is a fast, one pass algorithm for estimating the number of clusters and clusters centres in a set of data. The subtractive clustering method assumes each data point is a potential cluster centre and calculates a measure of the likelihood that each data point would define the cluster centre, based on the density of surrounding data points[113]. For better results it is recommended to normalize each point into a unit hyper-box to make each dimension identical[114, 115].The algorithm starts by finding the first large cluster, and then goes to find the second, and so on[116]. The algorithm is illustrated in the following lines:

• Selects the data point with the highest potential to be the first cluster centre. It is done by introducing a *potential measure* at data point x_j, defined as

$$Pot_j = \sum_{k=1}^{N} \exp\left(-\frac{\left\|x_j - x_k\right\|^2}{(r_a/2)^2}\right) \quad j = 1,...,N \tag{5.19}$$

Where r_a is a positive constant representing a neighbourhood radius and $\left\|x_j - x_k\right\|^2$ is the square of Euclidean distance between x_j and x_k. Hence, a data point will have a high density value if it has many neighbouring data points[117]. After calculating the potential for each vector, the one with the higher potential is selected as the first cluster centre[105].

• Remove all data points in the vicinity of the first cluster centre (as determined by radii of all data points to the newly selected cluster centre) in order to determine the next data cluster and its centre location. The first cluster centre x_{c1} is chosen as the point having the largest density value Pot_{c1}. Next, the density measure of each data point x_j is revised as follows:

$$Pot_j = Pot_j - Pot_{c1} \exp\left(\frac{\left\|x_j - x_{c1}\right\|^2}{(r_b/2)^2}\right) \tag{5.20}$$

Usually the r_b variable is taken to be as $1.5r_a$.

The process of acquiring new cluster centre is based on potential value in relation to an acceptance threshold $\bar{\varepsilon}$, rejection threshold $\underline{\varepsilon}$, and the relative distance criterion. A data point with the potential greater than the acceptance threshold is directly accepted as a cluster centre. The acceptance of a data point with a potential between the upper and the lower thresholds depends on the relative distance equation, defined as

$$\frac{d_{min}}{r_a} + \frac{Pot_k}{Pot_1} \geq 1 \tag{5.21}$$

where d_{min} is the shortest distance between the candidate cluster centre and all previously found cluster centres. If (5.21) is greater than 1, then the associated x will be considered as a new cluster centre.

• Iterates on this process until a sufficient number of clusters is attainted. Since the algorithm is fixed and does not rely on any randomness, the results are fixed. However, we can test the effect of the four parameters, namely, acceptance ratio $\bar{\varepsilon}$, reject ratio $\underline{\varepsilon}$, cluster radius r_a and squash factor r_b .These parameters have influence on the number of clusters and error performance measures. Large values of $\bar{\varepsilon}$ and $\underline{\varepsilon}$ will result in small number of rules. Conversely, small values of $\bar{\varepsilon}$ and $\underline{\varepsilon}$ will increase the number of clusters. A large value of r_a generally results in fewer clusters that lead to a coarse model[117, 118]. A small value of r_a can produce excessive number of clusters that may result in an over-defined system.

5.7 Gaussian Mixture Models (GMM) and Expectation Maximization(EM)

In the probabilistic point of view, data can be assumed to be generated according to several probability distributions. They can be derived from different types of probability density functions (e.g., multivariate Gaussian distribution), or the same families, but with different parameters[98]. In such a mixture model the probability density function of the process that generated the data is assumed to be a mixture of a certain number of probability density functions, each of which is described by a cluster[119].

A Gaussian Mixture Model (GMM) is a parametric conditional probability density function represented as a sum of Gaussian component densities in some proportions. These types of models rely on the assumption that the data comes from a known distribution (usually Gaussian distribution). In GMMs “similarity” should be understood as the probability that a data belongs to a specific density. Most databases contain a large amount of categorical data, where the notion of distance as a clustering metric is not natural and has to be defined according to the case. Gaussian mixture models not only can be used for conditional density estimation, but due to their probabilistic nature they also provide means for dealing with the problem of missing data and active data selection.

5.7.1 Gaussian Mixture Model

In a mixture model it is assumed that a given data set $X = \{ x_j \in \Re^P \mid j = 1, . , N\}$ has been drawn from a population of C clusters[120]. Each cluster is characterized by a probability distribution, specified as a prior probability, together with a conditional probability density function (cpdf)[121]. The data generation process may then be imagined as follows: first a cluster c, $c \in \{1, \ldots, C\}$ is chosen for a datum, indicating the cpdf to be used, and then the datum is sampled from this PDF. The weighted sum (mixture probability) of a given and finite C component Gaussian densities for an individual member of dataset x_j expressed as

$$p(x_j \mid \Theta) = \sum_{i=1}^{C} P(\theta_i) p_i(x_j \mid z_j, \theta_i) \qquad x_j \in \mathbb{R}^p \tag{5.22}$$

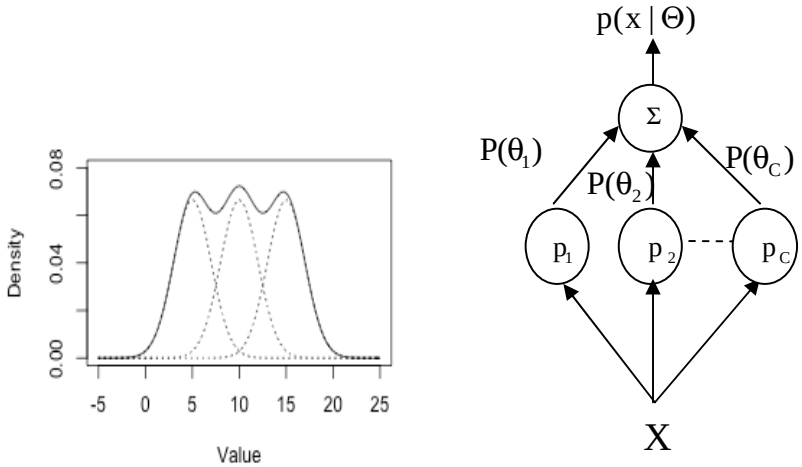


Fig 5.1 – Gaussian Mixture Model

Here z is a random variable that has the cluster indices as possible values and associate to each x_j we have z_j. $p_i(x_j \mid z_j, \theta_i)$ is the i'th component (cluster) conditional density given the cluster specified by z.

$$p_i(x_j \mid z_j, \theta_i) = \frac{1}{(2\pi)^{p/2}\sqrt{|\Sigma_i^x|}} \exp\left(-\frac{1}{2}(x_j - \mu_i)^T (\Sigma_i^x)^{-1}(x_j - \mu_i)\right) \tag{5.23}$$

Each Gaussian function as in eq (5.23) is integrated to one

$$\int_{\Re^P} p_i(x_j \mid z_j, \theta_i)dx = 1 \tag{5.24}$$

The variable $X_j = [x_1, x_2, ..., x_P]^T$ is multivariate input random vector in P dimensions, describing the attribute values of the data points. $\Theta = \{\theta_1, ..., \theta_C\}$ representing the unknown set of parameters for all clusters. In case of Gaussian components, the mixture density contains the following adjustable parameters for each θ_i:

mean $\mu_i = [\mu_1, \mu_2, ..., \mu_p]^T$ which is a $p \times 1$ dimensional matrix containing corresponding centres of the one-dimensional Gaussians as components (T denotes the vector transpose) and the other tuneable parameter is covariance matrix $\Sigma_i^{-1} = \text{diag}(1/\sigma_{i1}, 1/\sigma_{i2}, ..., 1/\sigma_{iP})$ which is the inverse of $p \times p$ covariance matrix created by product of P one-dimensional Gaussians.

$P(\theta_i)$ is the probability of i'th component which also reflects the relative importance of each cluster and since usually each point is assumed to belong to just one distribution and in eq(5.22) $p(x_j \mid \Theta)$ is a density function, it must be non-negative and integrate to one as well[122]. We have

$$1 = \int_{\Re^p} p(x_j \mid \Theta)dx = \int_{\Re^p} \sum_{i=1}^{C} P(\theta_i) p_i(x_j \mid z_j, \theta_i)dx = \sum_{i=1}^{C} P(\theta_i) \int_{\Re^p} p_i(x_j \mid z_j, \theta_i)dx = \sum_{i=1}^{C} P(\theta_i) \tag{5.25}$$

Hence, as eq (5.25) states,

$$\sum_{i=1}^{C} P(\theta_i) = 1 \cdot \tag{5.26}$$

The goal is to find the parameters Θ and $P(\theta_i)$ that maximizes the likelihood (or minimizes the minus likelihood).

5.7.2 Maximum Likelihood Estimation (MLE)

Given a set of parameter values, the associated PDF demonstrates that which data are more likely than others. In reality, however, we have already known/observed the data. Accordingly, we have to deal with inverse of the problem: Given the observed data and a model of interest, searching for

one PDF, among all the probability densities that the model presents, that is most probable to have generated the data[123]. In order to find solution for this inverse problem, we define the likelihood function by swapping the roles of the sample vector x and the parameter vector θ in $p(x|\theta)$ i.e.

$$L(\theta|x) = p(x|\theta) \tag{5.27}$$

Hence, $L(\theta|x)$ is the likelihood of parameter θ. To alleviate the computational load, the MLE estimate is obtained by maximizing the log-likelihood function $Log(L(\theta|x))$. Quality of a given set of parameters Θ is determined by how well the corresponding pdf model fits the data[124]. This is quantified by the log-likelihood of the data. If the random observations are independent of each other, the probability of generating N observations x_j (j=1,...,N) - according to probability theory - is the product. Given as

$$p(\{x_1,...,x_N\}|\Theta) = \prod_{j=1}^{N} p(x_j|\Theta) \tag{5.28}$$

or alternatively, in logarithm form

$$\begin{aligned} L(\Theta|X) &= \log(\prod_{j=1}^{N} p(x_j|\Theta)) \\ L(\Theta|X) &= \sum_{j=1}^{N} \log p(x_j|\Theta) \end{aligned} \tag{5.29}$$

The so-called *log-likelihood* ,can ease the technical task[99]. Substituting eq(5.22) in eq(5.29) ,

$$L(\Theta|X) = \sum_{j=1}^{N} \log\{\sum_{i=1}^{C} P(z_i;\theta_i) p_i(x_j|z_j,\theta_i)\} \tag{5.27}$$

In maximum likelihood estimation the unknown parameter $\Theta = \{\theta_1, \theta_2, \ldots, \theta_C\}$ is estimated so that the log-likelihood function is maximized by using a set of observed sample.

$$\partial Log(L(\Theta|x))/\partial\Theta = 0 \tag{5.30}$$

Unfortunately, since the solutions of eq(5.30) cannot be obtained analytically in most circumstances and therefore no closed-form solution for it, iterative routines are required to approximate MLE estimates. Among these methods, the Expectation-Maximization(EM) is one of the most popular schemes.

5.7.3 Jensens's Inequality

In order to connect the logarithm of sum to expectation operator later in the section, herby we explain briefly some useful results of Jensens's inequality.

Jensen's inequality is often employed to bound the logarithm of a sum of terms: Given C non-negative numbers $\pi_1, \cdots, \pi_C$ with the summation equal to one (it can be assumed as discrete probability distribution) and C arbitrary numbers $\alpha_1, ..., \alpha_C$, as the result of convexity of the logarithm we can conclude that[122]

$$\log \sum_{i=1}^{C} \pi_i \alpha_i \geq \sum_{i=1}^{C} \pi_i \log(\alpha_i) \tag{5.31}$$

Considering this inequality, some other useful expressions can be extracted out, such as

$$\log \sum_{i=1}^{C} \alpha_i = \log \sum_{i=1}^{C} \alpha_i \frac{\pi_i}{\pi_i} \geq \sum_{i=1}^{C} \pi_i \log \frac{\alpha_i}{\pi_i} \tag{5.32}$$

The inequality in eq(5.32) associates the logarithm of a sum with expected value of logarithm.

5.7.4 Expectation – Maximization for GMMs

In this section the iterative computation of maximum-likelihood is discussed when the observations can be considered as incomplete data. Since each iteration of the algorithm consists of an expectation step followed by a maximization step we call it the EM algorithm[125]. These two steps are repeated until convergence.

The general idea underlying the EM algorithm is to describe a value that is missing by a random variable. The domain of this random variable is the set of values that could be the actual, but unknown value. As a consequence, the likelihood of the data set becomes a random variable. This, of course, makes it impossible to maximize the likelihood directly, as it does not have a unique value anymore. However, since it is a random variable, we can compute its expected value and choose the parameters in such a way that this expected value is maximized.

In order to apply EM, a standard approach to handle these problems consists in assuming that for each X, there is a discrete unobserved (hidden) indicator vector $z_j \in \{1, ..., C\}$. The indicator vector specifies the mixture component from which the observation X is drawn[126]. Note that the combination of observations X and the 'hidden-states' Z constitute the complete-data[120, 124].

$$
\begin{aligned}
L(\Theta^t \mid X) &= \log p(X \mid \Theta^t) \\
&= \log p(X \mid \Theta^t) \sum_z P(Z \mid X, \Theta^t) \\
&= \log(\sum_z P(Z \mid X, \Theta^t) p(X \mid \Theta^t))
\end{aligned}
\tag{5.33}
$$

Equation above (5.33) maintains, simply because $\sum_z P(Z \mid X, \Theta^t) = 1$ [107]. Considering the definition of conditional probability, we have

$$
p(X \mid \Theta^t) = \frac{p(Z, X \mid \Theta^t)}{P(Z \mid X, \Theta^t)} \tag{5.34}
$$

Using eq5.33 in eq5.34 and also Jensen's Inequality (eq.5.32) the following results achieved

$$
\begin{aligned}
L(\Theta^t \mid X) &= \log(\sum_z P(Z \mid X, \Theta^t) \frac{p(Z, X \mid \Theta^t)}{P(Z \mid X, \Theta^t)}) \\
&\geq \sum_z \{P(Z \mid X, \Theta^t) \log(\frac{p(Z, X \mid \Theta^t)}{P(Z \mid X, \Theta^t)})\} \\
&= \sum_z P(Z \mid X, \Theta^t) \log p(Z, X \mid \Theta^t) - \sum_z P(Z \mid X, \Theta^t) \log P(Z \mid X, \Theta^t)
\end{aligned}
\tag{5.35}
$$

It might look that there are two random variables in eq (5.33-5.35) but the key issue is that X is constant and $\Theta = \{\theta_1, \theta_2, \ldots, \theta_C\}$ is a normal variable that wish to adjust and z is a random variable governed by it's the marginal distribution $p(z_i \mid X, \theta_i)$ and it is dependent on both observed data X and current estimate of parameters[127]. Also, recalling that $E[h(z) \mid X = x] = \sum_z h(z) p(z \mid x)$, therefore, eq (5.35) can be re-written as

$$
E_z[\log p(z, X \mid \theta^T) \mid X, \theta^{T-1}] - E_z[\log P(z \mid X, \theta^T) \mid X, \theta^{T-1}] \tag{5.36}
$$

$E_z[\ ]$ denotes expectation with respects to z. Thus, denote

$$
Q(\theta \mid \theta^{T-1}) = E_z[\log p(z, X \mid \theta^T) \mid X, \theta^{T-1}] \tag{5.37}
$$

The evaluation of expectation eq(5.37) called the E-Step. It is important to distinguish between the first and second argument of the Q functions. The second argument X, θ^{T-1} is regarded as fixed and known at every E-Step[128].

The second step of the algorithm is to maximize the expected value computed in the first step

$$\begin{aligned}\Theta^{T} &= \arg\max_{\Theta} Q(\Theta, \Theta^{T-1}) \\ &= \arg\max_{\Theta} E_{z}[\log p(z, X \mid \Theta^{T}) \mid X, \Theta^{T-1}] \\ &= \arg\max_{\Theta} \sum_{z \in \{1,..,C\}^{N}} P(z \mid X, \Theta^{t-1}) \sum_{j=1}^{N} \log p(x_{j}, z_{j} \mid \Theta) \\ &= \arg\max_{\Theta} \sum_{z \in \{1,..,C\}^{N}} \{\prod_{l=1}^{N} P(z_{l} \mid x_{l}, \Theta^{t-1})\} \sum_{j=1}^{N} \log p(x_{j}, z_{j} \mid \Theta)\end{aligned} \tag{5.38}$$

In the last step, we need a transformation, which replaces the complex sum over all possible vectors of cluster indices by a simple sum over the clusters. This transformation justified by Bilmes [129]. The final result is shown in the following equation

$$\Theta^{t} = \arg\max \sum_{i=1}^{C} \sum_{j=1}^{N} P(\theta_{i} \mid x_{j}) \log p(x_{j}, z_{j} \mid \theta) \tag{5.39}$$

In eq(5.39), $P(\theta_{i} \mid X_{j})$ computed by Bayesian rule as

$$P(\theta_{i} \mid x_{j}) = \frac{\frac{P(\theta_{i})}{(2\pi)^{P/2}\sqrt{|\Sigma_{i}|}} \exp(-\frac{1}{2}(x_{j} - \mu_{i})^{T}(\Sigma_{i})^{-1}(x_{j} - \mu_{i}))}{\sum_{i=1}^{C} \frac{P(\theta_{i})}{(2\pi)^{P/2}\sqrt{|\Sigma_{i}|}} \exp(-\frac{1}{2}(x_{j} - \mu_{i})^{T}(\Sigma_{i})^{-1}(x_{j} - \mu_{i}))} \tag{5.40}$$

Eq (5.40) illustrates the relative probability of the different clusters at the location of each $x_{j} \in X$ with a given set of cluster parameters[130]. The basic idea behind the EM iterative algorithm is that we would like to find Θ in order to maximize $\log p(x, z \mid \theta_{i})$, however we don't have/know the data z. So instead, first we can find the expectation of $\log p(x, z \mid \theta_{i})$ with the respect to unknown data z given the data X and our current estimate of Θ. The whole procedure carried out explicitly declaring a variable representing the expectation of complete data as a function of the incomplete data X [131].

Finding the derivative expressions with respect to every parameter in eq(5.39) and set them to zero, we obtain three groups of equations for the mean μ, standard deviations Σ and mixing probability $P(\theta)$. Start with some initial guess $(\mu_i^t, \Sigma_i^t, P(\theta_i^t))$, EM iterates the following computations until convergence to a local maximum of the likelihood function

$$\mu_i^{t+1} = \frac{\sum_{j=1}^{N} P(\theta_i^t \mid x_j) X_j}{\sum_{j=1}^{N} P(\theta_i^t \mid x_j)}$$

$$\Sigma_i^{t+1} = \frac{\sum_{j=1}^{N} P(\theta_i^t \mid x_j)(X_j - \mu_i^{t+1})(X_j - \mu_i^{t+1})^T}{\sum_{j=1}^{N} P(\theta_i^t \mid x_j)} \qquad i = 1,\ldots,C \qquad j = 1,\ldots,N \qquad (5.41)$$

$$P(\theta_i^{t+1}) = \frac{1}{N} \sum_{j=1}^{N} P(\theta_i^t \mid x_j)$$

Note that the updating of eq(5.38) for each cluster and recursively eq(5.39), perform both the expectation step and maximization step simultaneously.

5.7.5 Identification with Fuzzy Clustering

The main aim of this research is the development of an efficient modelling and identification scheme. In the system identification, the purpose of clustering is to find relationships between independent system variables, called the regressors, and future values of dependent variables, called the regressands [132]. One should however keep it in mind that, the relations defined by clustering are just associations among the data vectors, and as such do not yet constitute a prediction model of the given system. To achieve such a model, extra steps need to be taken. In the next chapter, in order to increase the efficiency of clustering algorithm, a clustering based algorithm is proposed. The main idea of these algorithms is that when the available input-output data set is clustered in the product space of the regressors and the model output, the obtained clusters would approximate the regression surface of the model.

Chapter 6

Fuzzy Wavelet Neural Networks

Neuro-fuzzy systems combine the learning ability of NNs and inference properties of fuzzy systems. In general, these systems derive fuzzy rules from a given input–output dataset. In Fuzzy Wavelet Neural Network, the aim is to combine Neuro-Fuzzy systems with wavelet functions in order to increase the performance of Neuro-Fuzzy and WNN systems significantly. Wavelets are known to have good modelling properties over a range of frequencies, and for this reason they have been used as activation functions in Neuro-Fuzzy systems, yielding fuzzy wavelet neural networks (FWNNs).

In the literature, several combination of fuzzy and WNN for solving time-series prediction, system identification and control problems have been reported [133, 134], [135], [136], [137]. The FWNN proposed in [131] uses summation of dilated and translated versions of wavelet functions in consequent part of fuzzy rules for system identification and control purposes. In [132] three types of FWNN models were developed for prediction and identification of nonlinear dynamic systems. Each fuzzy rule is associated by a sub-WNN. The resulting network has been used for function approximation. These models use wavelet functions in the consequent part of fuzzy rules. In all the models, translation and dilation parameters of wavelet functions, weights, and constant terms are adjusted by fast learning (second order) gradient-based algorithms.

However, models differ at the consequent parts of the fuzzy rules. In the first model, consequent parts consist of weighted summation of dilated and translated versions of single-dimensional wavelet functions. In the second model, consequent parts of the rules consist of radial function of wavelets and a constant term. In the last model, multiplication of single-dimensional wavelet functions and a constant term form the THEN part of fuzzy rules. In [135], a dynamic recurrent fuzzy wavelet network is proposed for identified nonlinear dynamic systems. In [136], the inputs enter into a discrete wavelet transform block, and then the output of this block is fuzzified and it

forms the input to a single NN. This model has been also used for system identification and control problems. In [137], the proposed model combines discrete wavelet transform with Takagi–Sugeno–Kang (TSK) fuzzy systems and it consists of a set of IF–THEN rules and THEN parts which are series expansion of wavelets functions. This model has been also applied to system modelling. In [79], both sigmoid and wavelet functions are used in the hidden layer of a WNN and the output of this new WNN is calculated by multiplication and summation of these results. Then, this WNN is used in consequent parts of the IF–THEN rules in FWNN.

This part of research presents fuzzy wavelet neural network that integrates wavelet functions with the TSK fuzzy model. The consequent parts of TSK type fuzzy IF–THEN rules are represented by either a constant or a function. As a function, most of the fuzzy and Neuro-Fuzzy models use linear functions. Here, the consequent part replaced by sequence of sub-WNNs explained in Chapter 4. FWNN systems can describe the considered problem by means of combination of sub-WNNs constructed at consequence part of each rule. In FWNN, fuzzy rules provide the influence of each WNN to the output of FWNN. The use of WNN with different dilation and translation values allows capturing different behaviours and essential features of the nonlinear model under these fuzzy rules. Sometimes these systems need more rules for modelling complex nonlinear processes in order to obtain the desired accuracy. Increasing the number of the rules leads to increasing number of neurons in the hidden layer of the wavelet network. To improve the computational power of the FWNN system, we use clustering technique to avoid the development of a large and complicated network.

6.1 Clustering Based -FWNN structure and Construction

To improve the computational power of the neuro-fuzzy system, we use wavelets in the consequent part of each rule. In this study we propose a new structure as in figure 6.1. In traditional ANFIS, consequent parts of the structure are linear functions and the gradient decent method is usually used to train the non-linear antecedent parameters. However in the proposed structure instead of a linear function, a novel Linear Combination Weight Wavelet Neural Network (LCW-WNN) recently presented by *Amina et. al.*, [138] has been applied. The MWNN-LCW model integrated two learning schemes; Weighted Least Square (WLS) and Extended Kalman Filter (EKF). Furthermore a Linear Combination Weight has been developed for further training speed and accuracy.

Since one cluster in the input-output space corresponds to one potential fuzzy logic rule, for constructing the CB-FWNN, the first step of the structure learning is to determine the number of fuzzy sets in the universal of discourse of each input variable[139]. Fuzzy clusters, similar to fuzzy rules, are well suited for presentation of the resulting model to the user. Although the traditional cluster algorithm works on unsupervised datasets, the extension in this chapter allows cluster models to be built based output trajectory and then used directly as fuzzy rules, which are then optimized. The fuzzy wavelet neural network depicted in figure 6.1 has got a modular structure. In the first step the whole dataset enters the first block for finding the optimum number of clusters. In this research the number of clusters determined by subtractive clustering based on the training data as described in previous chapter and remains constant throughout. Once the number of clusters defined, the layout of the desired CB-FWNN can be sketched. During the training, candidate models representing possible states of a structure, are clustered using the EM technique described in previous chapter but with some modifications. The obtained clusters are multivariate Gaussians each with different size and orientation from the other. The outputs of the clustering block are the firing strength multiplied by the consequence part of the structure.

At the consequence side, different scales of wavelet-neurons $\varphi_{x^p}(m_i^p, n_i^p)$ assigned for every dimension of the input. Although the number of different scales/translation allocated to each dimension is fixed, each of inputs can hire different scales/translations. By knowing the optimum number of clusters, the number of different scales which each input dimension going to be decomposed is then determined. In parallel, on the antecedent side, by knowing the number of clusters the number of fuzzy rules C and consequently the number of unknown parameters can be figured out. In the proposed scheme all the clustering processes are done in Cartesian product space of the inputs X and outputs y with P+1 dimensional data. However, the final obtained clusters are in P dimension with centres in the domain of input data X. With this scheme the eq(3.26) can be re-formulated as

$$\begin{aligned}\hat{y}_j &= \sum_{i=1}^{C} \gamma_i(X_j) W_i(X_j) \phi_i(X_j) \\ &= \sum_{i=1}^{C} \gamma_i(X_j) W_i(X_j) \prod_{p=1}^{P} \varphi_{m_i^p, n_i^p}(x_j) \\ &= \sum_{i=1}^{C} W_i(X_j) \gamma_i(X_j) \left(\prod_{p=1}^{P} \cos\left(2\pi\beta\left(\frac{x_j^p - n_i^p}{m_i^p}\right)\right) e^{\frac{-\left(\frac{x_j^p - n_i^p}{m_i^p}\right)^2}{\upsilon}}\right)\end{aligned} \tag{6.1}$$

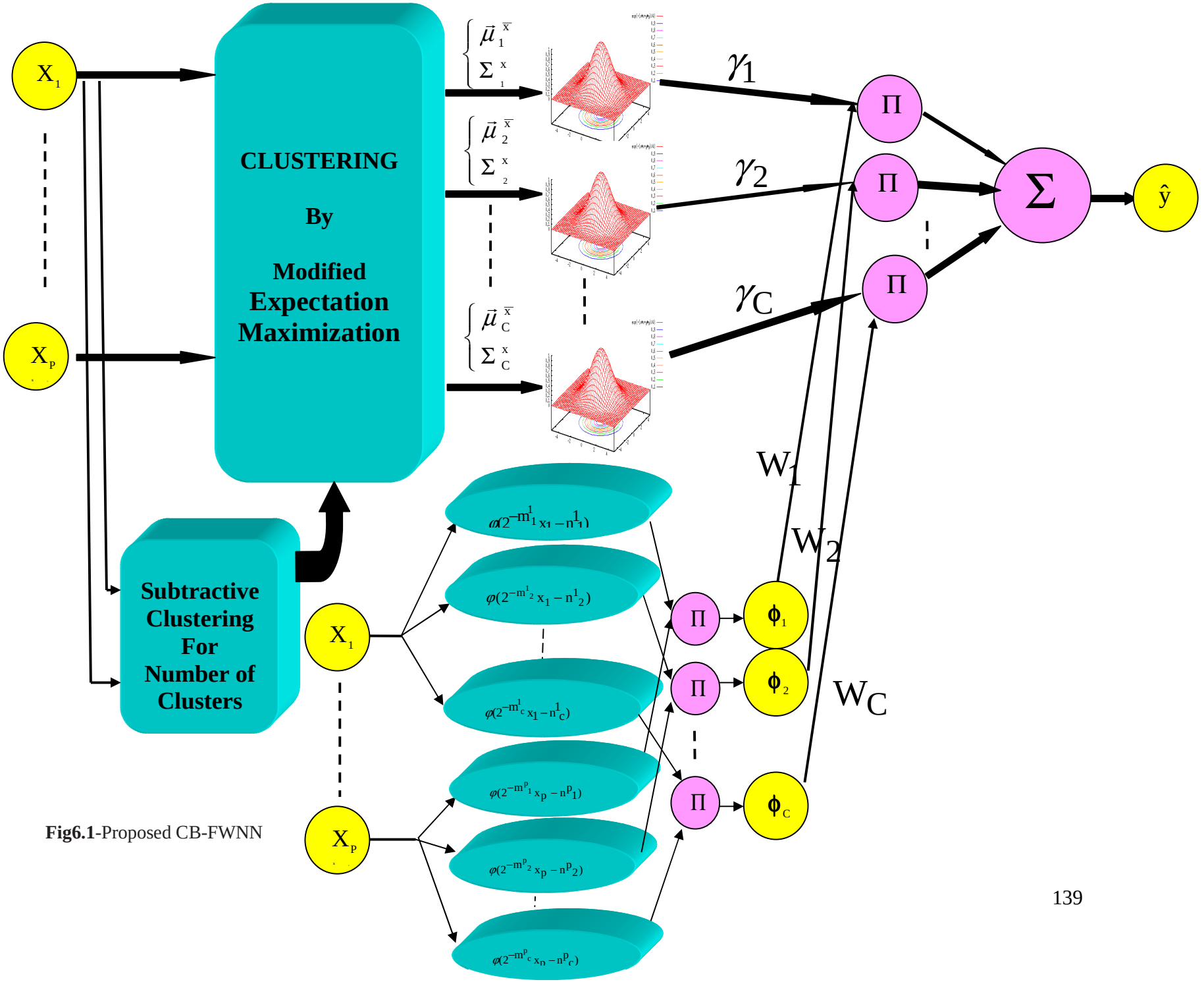


Fig6.1-Proposed CB-FWNN

Where the latter in eq(6.1) is obtained by substituting eq(4.52) and eq(4.53) in eq(3.26).
Discussing from nonparametric regression point of view, the main goal of parametric regression is to estimate a function from the knowledge of a limited number of points $\hat{y}_j = f(x_j)$. In many applications, the data-points are obtained experimentally and may even be corrupted with noise. Considering from standard non-parametric regression problem: Let (X,y) be a pair of random variables with values in $X \in \mathbb{R}^P$, $y \in \mathbb{R}$. Assume that $y_j = f(x_j) + \boldsymbol{\varepsilon}_j$ where $\boldsymbol{\varepsilon}_j$ is independent $N(0, \boldsymbol{\sigma})$ normally distributed variable. A function y=f(x) is the regression function of Y on X if

$$f(x) = E[y \mid X = x] \tag{6.2}$$

The regression problem can also be rephrased in the probabilistic framework, and as the conditional density $p(y \mid x)$ is also a mixture of Gaussians[140], therefore

$$\begin{aligned} \hat{y}_j &= f(x) = E[y_j \mid x_j] \\ &= \int_y y_j p(y_j \mid x_j) dy = \frac{\int y_j p(x_j, y_j) dy}{p(x_j)} \\ &= \sum_{i=1}^{C} \frac{[W_i \phi_i(x_j)] p(x_j \mid \theta_i) p(\theta_i)}{p(x_j)} = \sum_{i=1}^{C} W_i \times P(\theta_i \mid x_j)[\phi_i(x_j)] \end{aligned} \tag{6.3}$$

$\phi_i(x_j)$ is the i'th output, out of C outputs of LCW-WNN for the j'th input vector . $\phi_i(x_j)$ determines the contribution of each wavelet to the output of FWNN. $W_i(X_j)$ is the linear combination weight multiplying by $\phi_i(x_j)$, and $P(\theta_i \mid X_j)$ is the probability that the i'th Gaussian component is generated by the input vector X_j as depicted in eq(5.40). Merging the eq(6.1) and eq(6.3) gives us the final output based on dynamic of the proposed structure

$$\hat{y}_j = \sum_{i=1}^{C} W_i(X_j) \times P(\theta_i \mid X_j) \left(\prod_{p=1}^{P} \cos\left(2\pi\beta\left(\frac{x_j^p - n_i^p}{m_i^p}\right)\right) e^{\frac{-\left(\frac{x_j^p - n_i^p}{m_i^p}\right)^2}{\upsilon}} \right) \tag{6.4}$$

6.2 CB-FWNN Antecedent Parameters update

The design of FWNN (figure 6.1) includes determination of the unknown parameters that are the parameters of the antecedent and the consequent parts of the fuzzy IF–THEN rules. In the antecedent parts, the input-output space is divided into a set of fuzzy regions, and in the consequent parts the system behaviour in those regions is described. As mentioned earlier, recently, a number of different approaches have been used for designing fuzzy IF–THEN rules based on clustering.

EM could be a sophisticated candidate for training and estimate parameters for Fuzzy Multivariate membership functions. For each incoming pattern x_j rule firing strength can be regarded as the probability to which the incoming pattern maintained according to the corresponding PDF. The cluster parameters are estimated by Expectation Maximization knowing that EM approach avoids the numerical instabilities encountered in Gradient Decent and improved learning convergence, and the Wavelet Neural Network parameters are trained by Extended Kalman Filter and Weighted Least Square. If we think of a conditional density function $p(V|\theta)$ that is governed by the set of parameters (θ could be the means μ and covariance Σ of Gaussian densities) and we also have a data set V=(**X**,y) of size N and P+1 as dimensionality, supposedly drawn from this density.

$$X = \begin{bmatrix} \vec{x}_1 \\ \vec{x}_2 \\ \vdots \\ \vec{x}_P \end{bmatrix}, \quad y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_P \end{bmatrix}, \quad V = [X\ y]^T$$

For the V data we have a Mixture of Gaussians that model the $p(V|\theta)$ as described in eq(5.23)

$$p(v_j|\Theta) = \sum_{i=1}^{C} P(\theta_i) p_i(v_j|\theta_i) \qquad v_j \in \mathbb{R}^{p+1} \tag{6.5}$$

Where,

$$p_i(v_j|\theta_i) = \frac{1}{(2\pi)^{(P+1)/2}\sqrt{|\Sigma_i^v|}} \exp\left(-\frac{1}{2}(v_j - \mu_i^v)^T (\Sigma^v{}_i)^{-1}(v_j - \mu_i^v)\right) \tag{6.6}$$

Eq(6.5) can be further expanded according to probability theorem as follows:

$$p(v_j|\Theta) = \sum_{i=1}^{C} P(\theta_i) p_i(x_j|\theta_i) p(y_j|x_j, \theta_i) \tag{6.7}$$

The input distribution $p_i(x_j \mid \theta_i)$ defines the domain of influence of cluster

$$p_i(x_j \mid \theta_i) = \frac{1}{(2\pi)^{(P)/2}\sqrt{|\Sigma_i^x|}} \exp\left(-\frac{1}{2}(x_j - \mu_i^x)^T(\Sigma^x{}_i)^{-1}(x_j - \mu_i^x)\right) \tag{6.8}$$

And the output distribution considered as

$$p(y_j \mid x_j, \theta_i) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{1}{2}\frac{(y_j - \phi_i(X_j))^T(y_j - \phi_i(X_j))}{\sigma_y^2}\right) \tag{6.9}$$

So, eq(6.7) can be re-written

$$p(v_j \mid \Theta) = \sum_{i=1}^{C} \frac{P(\theta_i)}{(2\pi)^{(P)/2}\sqrt{|\Sigma_i^x|}} \exp\left(-\frac{1}{2}(x_j - \mu_i)^T(\Sigma^x{}_i)^{-1}(x_j - \mu_i)\right) \times \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{1}{2}\frac{(y_j - \phi_i(X_j))^T(y_j - \phi_i(X_j))}{\sigma_y^2}\right) \tag{6.10}$$

In order to find out the probability of a cluster, of which, a data pair is generated we use eq(5.40) in form of

$$p(\theta_i \mid v_j) = \frac{\frac{P(\theta_i)}{(2\pi)^{(P)/2}\sqrt{|\Sigma_i^x|}} \exp\left(-\frac{1}{2}(x_j - \mu_i^x)^T(\Sigma^x{}_i)^{-1}(x_j - \mu_i^x)\right) \times \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{1}{2}\frac{(y_j - \phi_i(x_j))^T(y_j - \phi_i(x_j))}{\sigma_y^2}\right)}{\sum_{i=1}^{C} \frac{P(\theta_i)}{(2\pi)^{(P)/2}\sqrt{|\Sigma_i^x|}} \exp\left(-\frac{1}{2}(x_j - \mu_i^x)^T(\Sigma^x{}_i)^{-1}(x_j - \mu_i^x)\right) \times \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{1}{2}\frac{(y_j - \phi_i(x_j))^T(y_j - \phi_i(x_j))}{\sigma_y^2}\right)} \tag{6.11}$$

By taking the partial derivatives of eq(6.10) with respect to parameters and set it to zero and also taking into account eq(6.4) the equations for updating the parameters at each step is calculated as

$$
\begin{aligned}
p(\theta_i) &= \frac{1}{N}\sum_{j=1}^{N} P(\theta_i \mid x_j, y_j) \\
\mu_i^x &= \frac{\sum_{j=1}^{N} x_j P(\theta_i \mid x_j, y_j)}{\sum_{j=1}^{N} P(\theta_i \mid x_j, y_j)} \\
\Sigma_i^x &= \frac{\sum_{j=1}^{N} (x_j - \mu_i^x)(x_j - \mu_i^x)^T P(\theta_i \mid x_j, y_j)}{\sum_{j=1}^{N} P(\theta_i \mid x_j, y_j)} \\
\sigma_i^y &= \frac{1}{N}\frac{\sum_{j=1}^{N} (y_j - \phi_i(x_j))^T ((y_j - \phi_i(x_j))P(\theta_i \mid x_j, y_j)}{P(\theta_i)}
\end{aligned}
\tag{6.12}
$$

6.3 Estimation of local linear models

A common choice for the cost function to estimate ω is the squared error. We assume that every local linear model is assigned with an equal weight to the error at all times while applying Least Square method[141]. In our case W_i in each model is fired by different rule weights, based on that, Weighted Least Square (WLS) is applied to estimate the linear model parameters. The number of parameters in each linear model is equal to (p+1) which P is the dimension of the input data. Associated to each cluster, there is one linear model, so the total number of parameter to estimate in local linear models is $C\times(p+1)$.The least square parameter estimating is accomplished by minimizing the following condition[140]

$$
J = \min_{\omega}\frac{1}{N}(y^d - X_{ext}\omega_i)^T \Upsilon_i (y^d - X_{ext}\omega_i) \tag{6.13}
$$

X_{ext} is the input data matrix extended by a unitary column and the Υ_i is an N-by-N matrix having membership degrees multiplied by the output of product layer from wavelet network on its main diagonal

$$
\Upsilon_i = \begin{bmatrix} \gamma_1^i\phi_1^i & 0 & \cdots & 0 \\ 0 & \gamma_2^i\phi_2^i & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \gamma_N^i\phi_N^i \end{bmatrix} = \begin{bmatrix} p(\theta_i \mid v_1)\phi_1^i & 0 & \cdots & 0 \\ 0 & p(\theta_i \mid v_2)\phi_1^i & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & p(\theta_i \mid v_N)\phi_1^i \end{bmatrix} \tag{6.14}
$$

To determine an estimate of the Linear Combination parameters by least squares (LS) minimization of J,

$$\omega_i = (X_{ext}^T \Upsilon_i X_{ext})^{-1} X_{ext}^T \Upsilon_i y^d \tag{6.15}$$

ω_i is a (P+1)– by – 1 vector of coefficients.

Note that the value of N is typically in the hundreds, whereas the value of C is typically 5–15. Thus the conditioning of the matrix $X_{ext}^T \gamma_i X_{ext}$, is generally good and does not pose problems for the inversion required by eq(6.15).

6.3.1 Extended Kalman Filter

Kalman filter (KF) is widely used in studies of dynamic systems, analysis, estimation, prediction, processing and control. KF is a set of mathematical equations which provide an efficient computational solution to sequential systems. The filter is derived by finding the estimator for a linear system, subject to additive white Gaussian noise. However, the real system is non-linear; Linearization using the approximation technique has been used to handle the non-linear system. This extension of the nonlinear system is called the Extended Kalman Filter (EKF)[142].

Consider the following discrete-time nonlinear stochastic system:

$$\begin{cases} \alpha_k = f(\alpha_{k-1}) + W_{k-1} \\ Y_k = h(\alpha_{k-1}) + V_k \end{cases} \tag{6.16}$$

Where X_k and Y_k denote the state vector and the measurement vector at the time k, respectively, f(.) is a non-linear representation and h(.) is a non-linear observation model. If the nonlinearities in eq (6.16) are sufficiently smooth, we can expand them around the state estimate using Taylor series

$$\begin{aligned} f(\alpha_k) &= f(\hat{\alpha}_k) + \Upsilon_k \times (\alpha_k - \hat{\alpha}_k) + \text{H.O.T.} \\ h(\hat{\alpha}_k) &= h(\hat{\alpha}_k) + H_k \times (\alpha_k - \hat{\alpha}_k) + \text{H.O.T.} \\ \Upsilon_k &= \frac{\partial f(\hat{\alpha}_{k-1})}{\partial \hat{\alpha}_{k-1}} |_{\alpha_{k-1} = \hat{\alpha}_{k-1}} \\ H_k &= \frac{\partial h(\hat{\alpha}_{k,k-1})}{\partial \hat{\alpha}_k} |_{\alpha_k = \hat{\alpha}_{k,k-1}} \end{aligned} \tag{6.17}$$

The random variables W_k and V_k represent the artificial additive process and measurement noises. They assumed to be independent (of each other), white, and with the following statistic characteristic[143]

$$\begin{cases} E[W_k] = 0 & E[W_k W_j^T] = Q_k \delta_{kj} \\ E[V_k] = 0 & E[V_k V_j^T] = R_k \delta_{kj} \\ E[W_k V_j^T] = 0 \end{cases} \tag{6.18}$$

Where E is the expectation operator and δ_{kj} is the Kronecker delta. Q_k denotes the covariance matrix of process noises, R_k is the covariance matrix of the measurement noises. The main idea of EKF is to expand the nonlinear functions f(.) and h(.) at the point of filtered values $\hat{\alpha}$ by means of the Taylor series neglecting higher-order terms in eq(6.17). The Extended Kalman Filter algorithm includes two groups of equations

The prediction equations:

$$\begin{aligned} \hat{\alpha}_{k,k-1} &= f(\hat{\alpha}_k) \\ P_{k,k-1} &= \Upsilon_{k,k-1} P_{k-1} \Upsilon^T_{k,k-1} + Q_{k-1} \end{aligned} \tag{6.19}$$

In the equation above, let us define by $\hat{\alpha}_{k,k-1}$ the predicted value of the state vector at time k based on all information available before time instant k, and $P_{k,k-1}$ its associated covariance error matrix.

The measurement equations:

$$\begin{aligned} L_k &= P_{k-1} H_k (H_k^T P_{k-1} H_k + R_k)^{-1} \\ P_k &= P_{k-1} - L_k H_k P_k + Q_k \\ \hat{\alpha}_k &= \hat{\alpha}_{k,k-1} + L_k e_k \\ e_k &= Y_k - h(\hat{\alpha}_{k,k-1}) \end{aligned} \tag{6.20}$$

L_k is the Kalman Gain matrix.

6.3.2 Estimating Wavelet Neural Network (WNN) parameters using EKF

In this section we briefly outline how EKF can be applied to WNN network optimization illustrated in figure 6.1. Let the Transitions and Dilations of the feed-forward WNN be the states of the extended Kalman filter and the final output of the network be the measurements of the filter. Let us suppose that there are C centres for each dimension and the dimensionality of data is P, and associated (C + 1) linear component output weights. The updating of Linear Weights is described in the previous section so they are excluded from EKF estimation. In order to cast the optimization problem in a form suitable for Kalman Filtering, we let the elements of the Translation and the elements of the Dilations constitute the state of a nonlinear system α, and we let the output of the WNN network constitute the output of the nonlinear system to which the Kalman Filter is applied. α is considered as an array which all WNN parameters are arranged in there.

$$\alpha = \left[m_1^1 \quad m_1^2 \quad \cdots \quad m_1^c \quad \cdots m_p^1 \quad m_p^2 \quad \cdots m_p^c \quad n_1^1 \cdots n_1^c \cdots n_p^1 \cdots n_p^c \right]$$

The actual output at k'th iteration of the optimization algorithm is given as

$$\begin{aligned} y &= [y_1 \cdots y_N]^T \\ h(\hat{\theta}_k) &= [\hat{y}_1 \cdots \hat{y}_N]_k^T \end{aligned} \tag{6.21}$$

$h(\hat{\theta}_k)$ is the actual output of the WNN network given the WNN parameters at the k'th iteration of the Kalman recursion. H_k is the partial derivative of the WNN output with respect to the WNN network parameters at the k'th iteration of the Kalman recursion. It is denoted as below

$$H_k = \begin{bmatrix} H_{Dilation} \\ H_{Trans} \end{bmatrix} \tag{6.22}$$

Where

$$H_{Dilation}=\frac{\partial\hat{y}}{\partial m}=\begin{bmatrix} W_1^1\times\frac{\partial\varphi_1^1(x_1^1)}{\partial m_1^1}\times\frac{\phi_1(\vec{X}_1)}{\varphi_1^1(x_1^1)} & \cdots & W_1^N\times\frac{\partial\varphi_1^1(x_N^1)}{\partial m_1^1}\times\frac{\phi_1(\vec{X}_N)}{\varphi_1^1(x_1^1)} \\ \vdots & \cdots & \vdots \\ W_C^1\times\frac{\partial\varphi_C^1(x_1^1)}{\partial m_C^1}\times\frac{\phi_C(\vec{X}_1)}{\varphi_C^1(x_1^1)} & \cdots & W_C^N\times\frac{\partial\varphi_C^1(x_N^1)}{\partial m_C^1}\times\frac{\phi_C(\vec{X}_N)}{\varphi_C^1(x_1^1)} \\ W_1^1\times\frac{\partial\varphi_1^2(x_1^2)}{\partial m_1^2}\times\frac{\phi_1(\vec{X}_1)}{\varphi_1^2(x_1^2)} & \cdots & W_1^N\times\frac{\partial\varphi_1^2(x_N^1)}{\partial m_1^2}\times\frac{\phi_1(\vec{X}_N)}{\varphi_1^2(x_N^1)} \\ \vdots & \cdots & \vdots \\ W_C^1\times\frac{\partial\varphi_C^2(x_1^2)}{\partial m_C^2}\times\frac{\phi_C(\vec{X}_1)}{\varphi_C^2(x_1^2)} & \cdots & W_C^N\times\frac{\partial\varphi_C^2(x_N^2)}{\partial m_C^2}\times\frac{\phi_C(\vec{X}_N)}{\varphi_C^2(x_N^2)} \\ \vdots & \cdots & \vdots \\ W_1^1\times\frac{\partial\varphi_1^P(x_1^P)}{\partial m_1^P}\times\frac{\phi_1(\vec{X}_1)}{\varphi_1^P(x_1^P)} & \cdots & W_1^N\times\frac{\partial\varphi_1^P(x_N^P)}{\partial m_1^P}\times\frac{\phi_1(\vec{X}_N)}{\varphi_1^P(x_N^P)} \\ \vdots & \cdots & \vdots \\ W_C^1\times\frac{\partial\varphi_C^P(x_1^P)}{\partial m_C^P}\times\frac{\phi_C(\vec{X}_1)}{\varphi_C^P(x_1^P)} & \cdots & W_C^N\times\frac{\partial\varphi_C^P(x_N^P)}{\partial m_C^P}\times\frac{\phi_C(\vec{X}_N)}{\varphi_C^P(x_N^P)} \end{bmatrix}$$

$$H_{Trans}=\frac{\partial\hat{y}}{\partial n}=\begin{bmatrix} W_1^1\times\frac{\partial\varphi_1^1(x_1^1)}{\partial n_1^1}\times\frac{\phi_1(\vec{X}_1)}{\varphi_1^1(x_1^1)} & \cdots & W_1^N\times\frac{\partial\varphi_1^1(x_N^1)}{\partial n_1^1}\times\frac{\phi_1(\vec{X}_N)}{\varphi_1^1(x_1^1)} \\ \vdots & \cdots & \vdots \\ W_C^1\times\frac{\partial\varphi_C^1(x_1^1)}{\partial n_C^1}\times\frac{\phi_C(\vec{X}_1)}{\varphi_C^1(x_1^1)} & \cdots & W_C^N\times\frac{\partial\varphi_C^1(x_N^1)}{\partial n_C^1}\times\frac{\phi_C(\vec{X}_N)}{\varphi_C^1(x_N^1)} \\ W_1^1\times\frac{\partial\varphi_1^2(x_1^2)}{\partial n_1^2}\times\frac{\phi_1(\vec{X}_1)}{\varphi_1^2(x_1^2)} & \cdots & W_1^N\times\frac{\partial\varphi_1^2(x_N^1)}{\partial n_1^2}\times\frac{\phi_1(\vec{X}_N)}{\varphi_1^2(x_N^1)} \\ \vdots & \cdots & \vdots \\ W_C^1\times\frac{\partial\varphi_C^2(x_1^2)}{\partial n_C^2}\times\frac{\phi_C(\vec{X}_1)}{\varphi_C^2(x_1^2)} & \cdots & W_C^N\times\frac{\partial\varphi_C^2(x_N^2)}{\partial n_C^2}\times\frac{\phi_C(\vec{X}_N)}{\varphi_C^2(x_N^2)} \\ \vdots & \cdots & \vdots \\ W_1^1\times\frac{\partial\varphi_1^P(x_1^P)}{\partial n_1^P}\times\frac{\phi_1(\vec{X}_1)}{\varphi_1^P(x_1^P)} & \cdots & W_1^N\times\frac{\partial\varphi_1^P(x_N^P)}{\partial n_1^P}\times\frac{\phi_1(\vec{X}_N)}{\varphi_1^P(x_N^P)} \\ \vdots & \cdots & \vdots \\ W_C^1\times\frac{\partial\varphi_C^P(x_1^P)}{\partial n_C^P}\times\frac{\phi_C(\vec{X}_1)}{\varphi_C^P(x_1^P)} & \cdots & W_C^N\times\frac{\partial\varphi_C^P(x_N^P)}{\partial n_C^P}\times\frac{\phi_C(\vec{X}_N)}{\varphi_C^P(x_N^P)} \end{bmatrix}$$

(6.23)

$H_{Dilation}$ and H_{Trans} are both $[C\times P]\times N$ matrix and H_K in eq(6.22) is an $[2\times C\times P]\times N$. Having the H_K matrix ready, we can now execute recursively the eq(6.20)

6.4 TSK CB-FNN

Some approaches for modelling TSK fuzzy rules have been proposed in the literature and they use one-dimensional (1-D) (univariate) fuzzy sets, such as triangular or Gaussian ones, and partitioned multidimensional input spaces by grid Cartesian products of these univariate membership functions. The advantages of this approach are the simple and transparent representation of the membership functions and the straightforward application of the model. But, when the model is obtained by grid-type partitioning of its input space, the number of rules grows exponentially with the number of input variables, which leads to an unnecessarily complex model (curse of dimensionality).

The scheme described earlier can be generalized to all sorts of TSK-FNNs to overcome this problem. In this way, the number of rules can be significantly reduced. Although in the next

evaluation section the implementation of the WNN structure is well-justified in the results' table, in this sub-section we briefly mention how it could be applied to even simple non-wavelet-based structures.

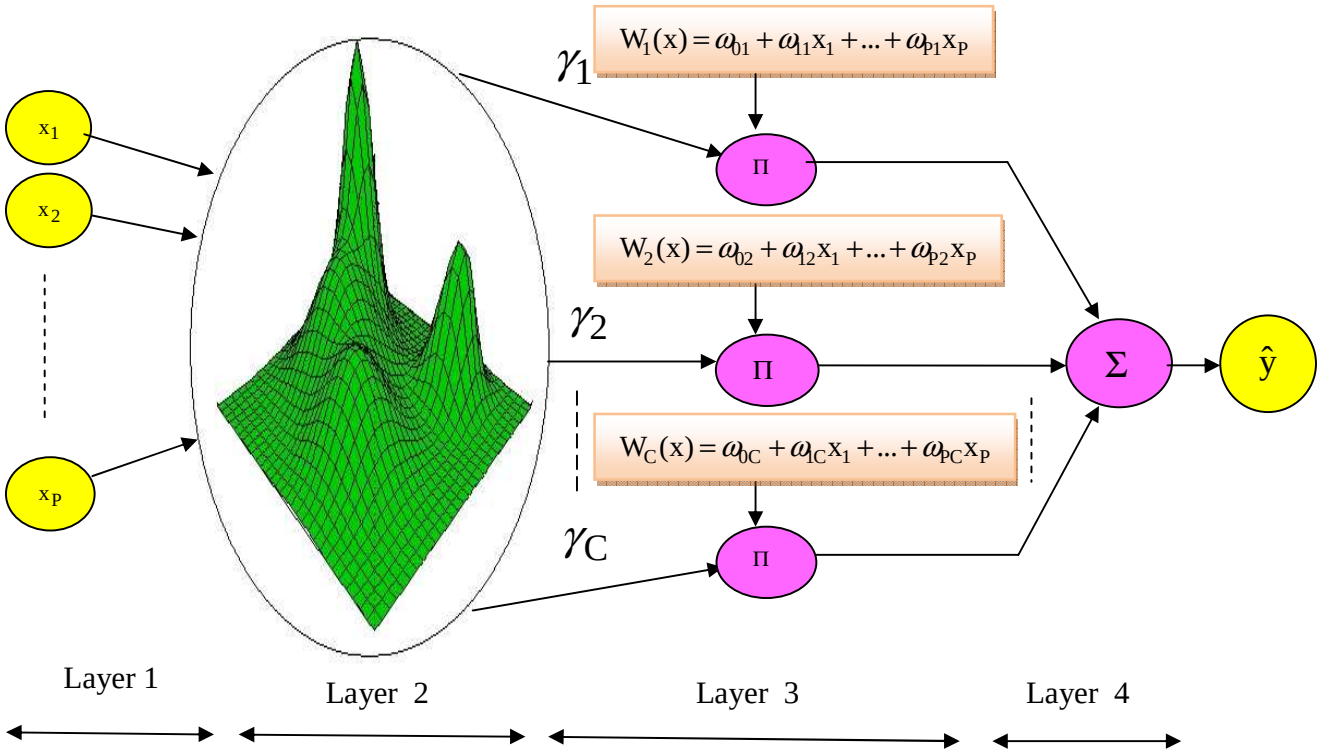


Fig 6.2 –TSK Clustering-Based Fuzzy Neural Network

Layer 1 : The input nodes are located at this layer. P nodes needed for a P-dimensional dataset.

Layer 2: Gaussian Mixture Models play the role of clusters as discussed earlier, and the tuning method is the modified Expectation-Maximization. However, due to the absence of WNN part, the output distribution is simplified as

$$p(y_j \mid x_j, \theta_i) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{1}{2}\frac{(y_j - W_i(X_j))^T (y_j - W_i(X_j))}{\sigma_y^2}\right) \qquad (6.24)$$

As the tuning of the clusters is imposed on product space of input-output, therefore the distribution of the P+1 dimensional data (including outputs) is given as

$$p(v_j|\Theta)=\sum_{i=1}^{C}\frac{P(\theta_i)}{(2\pi)^{(P)/2}\sqrt{|\Sigma_1^x|}}\exp(-\frac{1}{2}(x_j-\mu_1)^T(\Sigma_i^x)^{-1}(x_j-\mu_1))\times\frac{1}{\sqrt{2\pi\sigma_y^2}}\exp(-\frac{1}{2}\frac{(y_j-W_1(X_j))^T(y_j-W_1(X_j))}{\sigma_y^2}) \quad (6.25)$$

The probability of a cluster of which a datum is generated is modified as the output distribution also takes part in its value.

$$p(\theta_i|x_j,y_j)=\frac{\frac{P(z_1;\theta_1)}{(2\pi)^{(P)/2}\sqrt{|\Sigma_1^x|}}\exp(-\frac{1}{2}(x_j-\mu_1)^T(\Sigma_i^x)^{-1}(x_j-\mu_1))\times\frac{1}{\sqrt{2\pi\sigma_y^2}}\exp(-\frac{1}{2}\frac{(y_j-W_1(x_j))^T(y_j-W_1(x_j))}{\sigma_y^2})}{\sum_{i=1}^{C}\frac{P(z_1;\theta_1)}{(2\pi)^{(P)/2}\sqrt{|\Sigma_1^x|}}\exp(-\frac{1}{2}(x_j-\mu_1)^T(\Sigma_i^x)^{-1}(x_j-\mu_1))\times\frac{1}{\sqrt{2\pi\sigma_y^2}}\exp(-\frac{1}{2}\frac{(y_j-W_1(x_j))^T(y_j-W_1(x_j))}{\sigma_y^2})} \quad (6.26)$$

The updating formula for P-dimension cluster parameters are

$$\begin{aligned}
p(\theta_i) &= \frac{1}{N}\sum_{j=1}^{N} P(\theta_i \mid x_j, y_j) \\
\mu_i^x &= \frac{\sum_{j=1}^{N} x_j P(\theta_i \mid x_j, y_j)}{\sum_{j=1}^{N} P(\theta_i \mid x_j, y_j)} \\
\Sigma_i^x &= \frac{\sum_{j=1}^{N} (x_j-\mu_i^x)(x_j-\mu_i^x)^T P(\theta_i \mid x_j, y_j)}{\sum_{j=1}^{N} P(\theta_i \mid x_j, y_j)} \\
\sigma_i^y &= \frac{1}{N}\frac{\sum_{j=1}^{N} (y_j-W_i(x_j))^T((y_j-W_i(x_j))P(\theta_i \mid x_j, y_j)}{P(\theta_i)}
\end{aligned} \quad (6.27)$$

Layer 3 : Multiplication of each cluster's output take place in layer 3. The TSK local linear models are activated based on the degree of membership of the datum to their corresponding clusters. The output of this layer is

$$\phi_i = \gamma_i W_i \quad (6.28)$$

Layer 4 : The output of the whole structure comes as $\hat{y}_j$ for the input vector X_j. $\hat{y}_j$ is computed by aggregating ϕ_i collected from previous layer

$$\hat{y}_j = \sum_{i=1}^{C} \gamma_i W_i(X_j) = \sum_{i=1}^{C} P(\theta_i \mid X_j) W_i(X_j) \tag{6.29}$$

6.4.1 Consequence Parameter Updating

Weighted Least Square (WLS) can be applied to update the local linear models. The criterion which should be minimized is as eq 6.13. Despite the similarity in cost function the Υ_i matrix which has the membership degrees on its main diagonal is different as follow

$$\Upsilon_{i,FNN} = \begin{bmatrix} \gamma_1^i & 0 & \cdots & 0 \\ 0 & \gamma_2^i & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \gamma_N^i \end{bmatrix} = \begin{bmatrix} p(\theta_i \mid x_1) & 0 & \cdots & 0 \\ 0 & p(\theta_i \mid x_2) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & p(\theta_i \mid x_N) \end{bmatrix} \tag{6.30}$$

The estimate of the consequent parameters is given by

$$\omega_i = (X_{ext}^T \Upsilon_{i,FNN} X_{ext})^{-1} X_{ext}^T \Upsilon_{i,FNN} y^d \tag{6.31}$$

The definition of X_{ext}^T and y^d are the same as eq(6.13 - 6.15).

6.5 Case Study – Short Term Load Forecasting in Power System

Short term electric load (STLF) forecasting is the cornerstone of the operation of today's power systems. Precise load forecasting helps the electric utility to make unit commitment decisions, reduce spinning reserve capacity and schedule device maintenance plan properly. The system operators use the load forecasting result as a basis of off-line network analysis to determine if the system might be vulnerable. If so, corrective actions should be prepared, such as load shedding, power purchases and bringing peaking units on line. With the recent trend of deregulation of electricity markets, STLF has gained more importance and greater challenges[144]. In the market

environment, precise forecasting is the basis of electrical energy trade and spot price establishment for the system to gain the minimum electricity purchasing cost. In the real-time dispatch operation, forecasting error causes more purchasing electricity cost or breaking-contract penalty cost to keep the electricity supply and consumption balance.

In recent years two different paradigms of STLF have emerged, namely those that are based on statistical analysis and those that are based on CI techniques. The former include such traditional statistical approaches as linear and nonlinear regression analysis[145], time series models[146] and Kalman Filtering models[147]. Most statistical based methods for STLF are linear models that make certain assumptions about the characteristics of the load series. However, the relationship between the variables that affect load demand and actual load demand is complex and nonlinear making the accuracy of different statistical models system dependent. Fan *et al.* [148] described an implementation of ARIMAX (autoregressive integrated moving average with exogenous variables) models for load forecasting, while Yang and Huang[149] proposed a fuzzy autoregressive moving average with exogenous input variables (FARMAX) for one day ahead hourly load forecasting.

Most recently, the scientific community has turned to CI for solving the problem of STLF. CI-based models are able to learn nonlinear dependencies directly from the historical data. These models can be divided into three subgroups depending on the artificial intelligence paradigm that they represent, namely neural networks (NN), including the multilayer perceptron (MLP)[150], radial basis function (RBF)[151] and support vector machine (SVM)[152], fuzzy systems[153] and hybrid models[154]. Although the NN-based models (particularly the MLP) and the fuzzy systems have received the most attention in STLF literature, a growing interest exists for the case of hybrid schemes. Yang[155] presented an integrated method that combines an increment regression tree and SVM for STLF. Both increment and non-increment tree are built according to the historical data to provide the data space partition and input variable selection. SVM was employed to the samples of regression tree nodes for further fine regression. The integration of genetic algorithms (GA) with SVM has found its application also in STLF cases. A novel GA-based SVM forecasting model with deterministic annealing clustering has been presented by Sun[156]. The experimental results demonstrated its superiority over a classic MLP network. Amongst the above neural based forecasting techniques most of them generally can be classified into two categories in accordance with techniques they employ. One approach treats the load pattern as a time series signal and predicts the future load by using the already mentioned techniques. In the second approach the

load pattern is considered to be heavily dependent both on weather variables and previous load patterns.

Many attempts by researchers have been made to improve load-forecasting process in many worldwide regions. Khan *et al*.[157] used a hybrid of neural network and fuzzy logic to forecast the load in Czech Republic. They found that hybrid fuzzy neural network and radial basis function networks are the best candidates for the analysis of the load in Czech Republic. An Adaptive Neuro Fuzzy Inference System (ANFIS) has been utilised by Yuill *et al*.[158] for the development of a STLF model for South African power networks, by considering temperature and humidity as the main weather parameters affecting the load. Another study by Kodogiannis *et al.* [159] discussed the development of improved neural-network-based forecasting models for the power system of the Greek island of Crete. The performance was evaluated through a simulation study, using metered data provided by the Greek Public Power Corporation. Their results indicated that the load-forecasting models developed provided more accurate forecasts than the conventional methods.

NN models basically use the sigmoid activation function in neurons. However, the sigmoid function is not orthogonal, and the energy of the sigmoid function is limitless, and this leads to slow convergence. Wavelet function is a waveform that has limited duration and an average value of zero. The integration of the localisation properties of wavelets and the learning abilities of NN shows advantages of Wavelet neural networks (WNN) over NN in complex nonlinear system modelling in terms of learning efficiency and structure transparency. A STLF model of wavelet-based networks was proposed in[160] to model the highly nonlinear, dynamic behaviour of the system loads and to improve the performance of traditional NNs. To investigate the performance of the proposed evolving wavelet-based networks on load forecasting, the practical load and weather data for the Taiwan power systems were employed.

The comparison against an STLF NN version revealed the superiority of WNN forecasting in terms of more accurate forecasting result and faster training speed. Here, a modular-constructed forecasting system is proposed, where 24 neural blocks with a single output have to be developed and trained separately to represent the 24 hourly loads respectively. The outline of the proposed architecture is illustrated in figure 6.3.

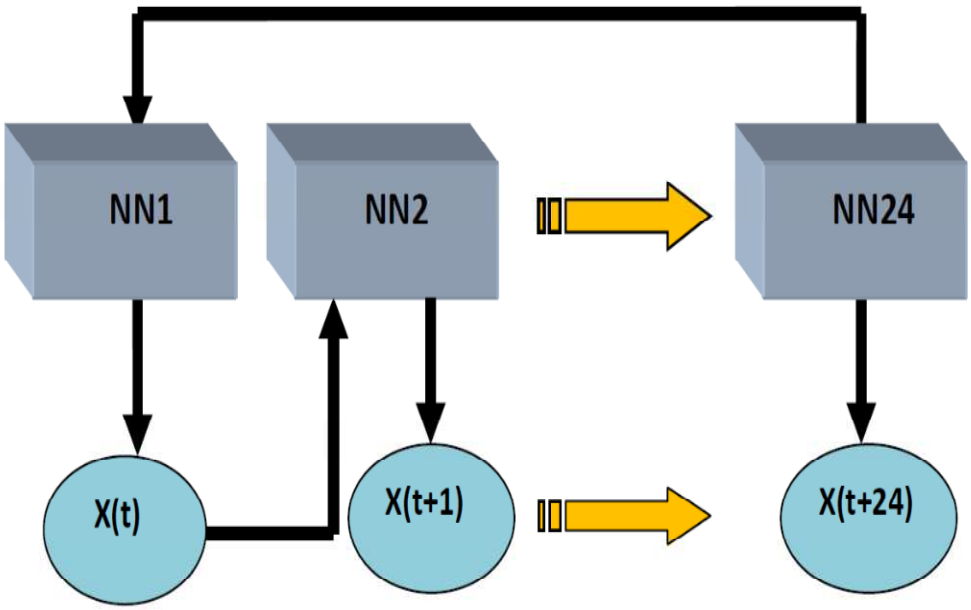


Fig 6.3 - Proposed modular architecture for the STLF problem

The main objective of the proposed system is the development of sufficiently accurate blocks representing the individual hourly loads. An assumption has been made in the case of black-outs, which occurred during the whole year. All the zero load values have been removed from both training and testing sets, and were replaced by the mean value of the preceding and subsequent load value. In this section the results and the statistics of forecasts obtained from the application of the developed STLF models on the power system of the island of Crete presented. Only results that correspond to hours with the maximum (14:00h) and minimum (02:00h) load consumption are illustrated.

Case studies for the proposed methods were carried out for a 24-hour load forecasting. The complete results for the STLF problem, for the hours with minimum and maximum load consumption, are illustrated in Table I. Training has been conducted using power load data for 1994 (365 data points), while testing has been evaluated using data from the 4 first months of 1995.

6.5.1 Cluster Analysis for Case Study

Each cluster could be oblique towards the input axis and the covariance matrix is not necessarily diagonal; therefore projecting the clusters on to the axis won't give a precise univariate membership functions. In P-dimensional space each cluster can also be recognized by its corresponding eigenvectors and eigenvalues.

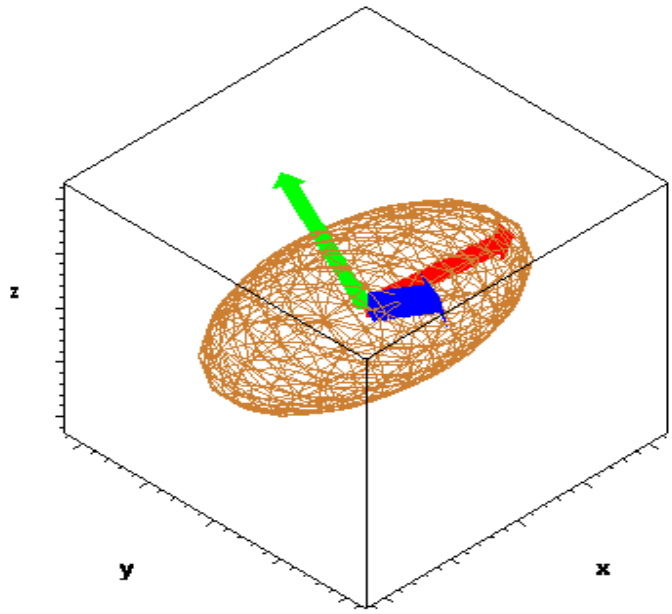


Fig 6.4- Eigenvectors of a 3-dimension hyper-ellipsoidal cluster

Projecting each cluster on to its associated eigenvectors makes better estimation of decomposed univariate elements of each cluster. Let us denote β_{ip} and $\vec{k}_{ip}$ as the eigenvalues and the unitary eigenvectors of Σ_i^x, respectively. However, in this case to find the degree of membership the *input domain transformation* is needed

$$\tilde{x}_{ip} = t_{ip}^T X \qquad (6.32)$$

Based on eq (6.32), the univariate Gaussian membership functions are given by

$$A_{ip}(\tilde{x}_{ip}) = \exp(-\frac{1}{2}\frac{(\tilde{x}_{ip} - \tilde{\mu}_{ip})^2}{\tilde{\sigma}_{ip}^2}) \tag{6.33}$$

Where

$$\begin{aligned} \tilde{\mu}_{i,p} &= \vec{k}_{i,p}^{T} \mu_i^x \\ \tilde{\sigma}_{i,p}^2 &= \beta_{i,p}^2 \end{aligned} \tag{6.34}$$

The eigenvector projection of each cluster after variable transformation is illustrated in figure 6.5.

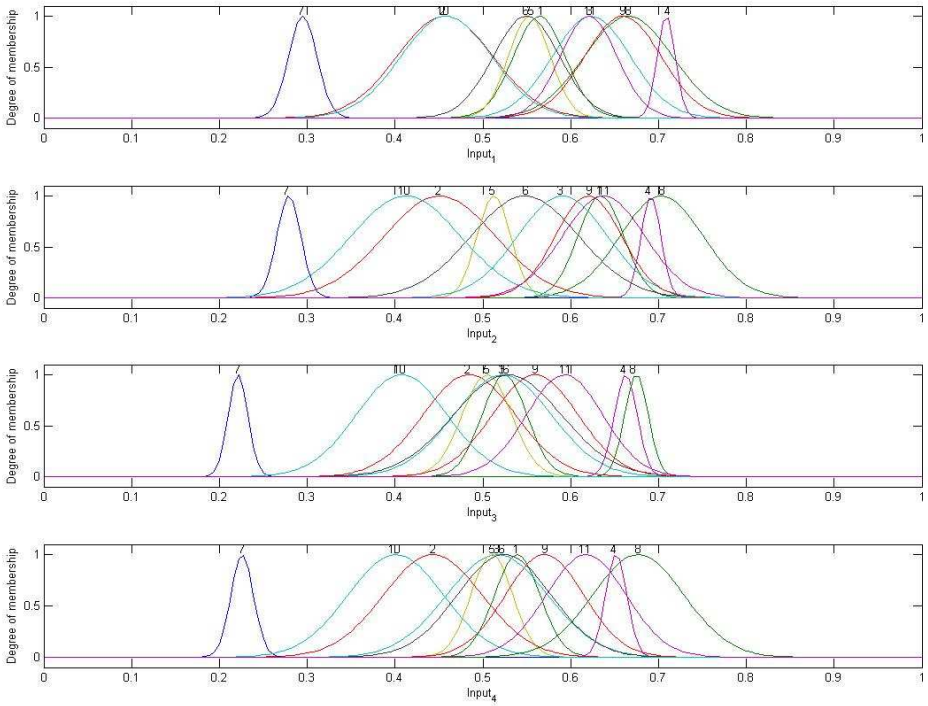


Fig 6.5– Projection of normalized Multivariate (4-dimesnion) clusters on their eigenvectors

6.5.2 CB-FWNN Short Term Load Forecasting Results

An obvious advantage of the proposed modular architecture is that, since the complete system consists of 24 neural blocks, each one with a single output, training is easier and faster compared to traditional neural approaches, which treat the output as a 24x1 vector. After many trials, it has

been found that only two previous time load parameters are necessary for the proposed CB-FWNN to achieve an acceptable performance, whereas the number of cluster/fuzzy rules was determined with the aid of subtractive clustering to be 10.

Figure 6.6 and 6.7 illustrate the training performances on both minimum and maximum power consumption cases, while Figure 6.8 and 6.9 illustrate the testing performances for both cases.

Table 6.1summarizes the various performance indices.

TABLE 6.1 – Performance indices of proposed CB-FWNN for Short Term Load Forecasting

Statistical index	**Testing Data sets**	
	14:00	**02:00**
Coefficient of determination (R^2)	0.9810	0.9673
Root mean square error (RMSE)	2.6573	1.8986
Mean relative percentage error (MRPE) (%)	1.9964	1.2826
Mean absolute percentage error (MAPE) (%)	0.8601	0.4935
Standard error of prediction (SEP) (%)	2.665	2.0018
Bias factor (B_f)	0.9911	0.9949
Accuracy factor (A_f)	1.0204	1.0129

This performance was associated also with a fast training speed, of 280 epochs. The regression coefficient (R^2) is often used as an overall measure of the prediction attained. It is common practice to use this index to compare different statistical models. It measures the fraction of the variation about the mean that is explained by a model. The higher the value ($0 \le R^2 \le 1$), the better is the prediction by the model. The CB-FWNN scheme developed herein was found to yield high level agreement with experimental observations for the test data set. The values of the coefficient

of determination (R^2), as shown in Table 6.1, indicate a very good fit of the experimental data from the CB-FWNN based approach.

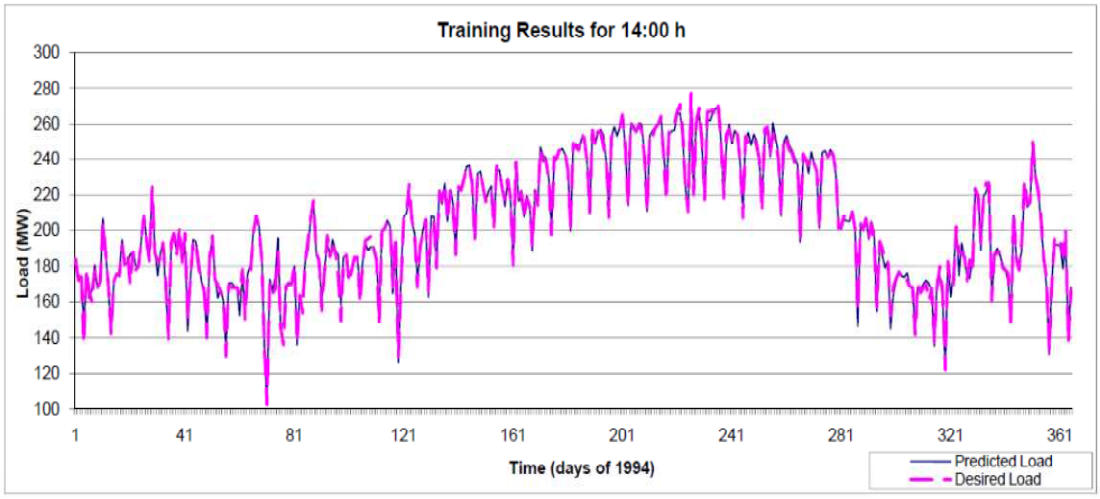


Fig 6.6 - Training performance for max load

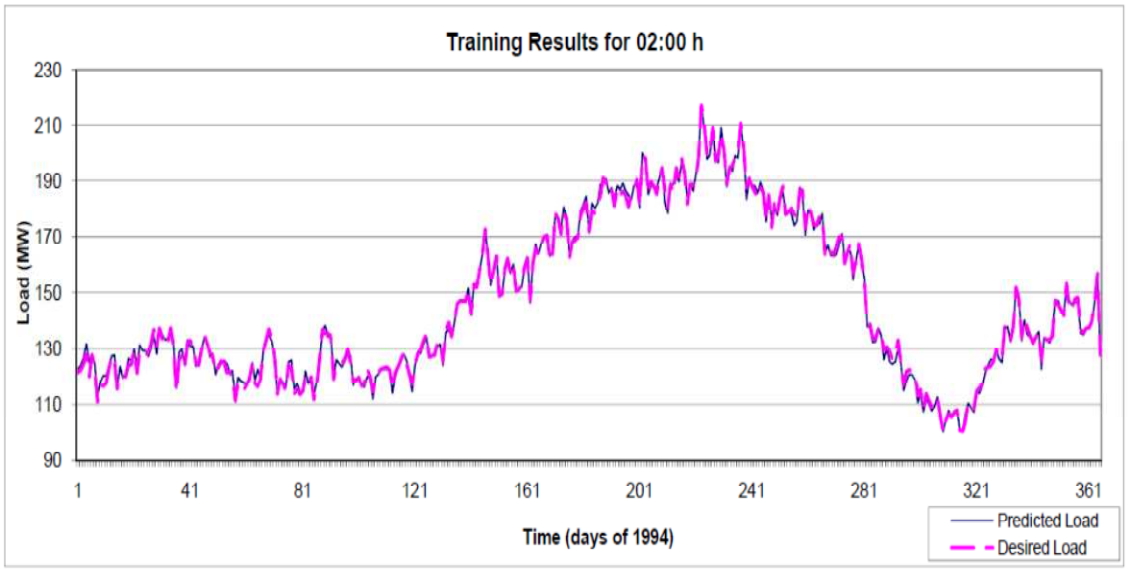


Fig 6.7 - Training performance for min load

However, R^2 is a suitable criterion for model comparison on the assumption that the error is normally distributed and not dependent on the mean value; In fact, the distribution of the error is not clearly known, so this term must be used with caution, particularly in non-linear regression models and hence additional indices must be employed for model comparison.

RMSE index is calculated between the desired and output values and then averaged across all data and it can be used as an estimation of the goodness of fit of the models. It can also provide information about how consistent the model would be in the long run. The related RMSE values for the proposed scheme are very low, as shown in Table 6.1, indicating the ability of CB-FWNN to make better prediction on data for which there was no previous training.

In order to evaluate the goodness of the current performance of the proposed CB-FWNN scheme, a comparison against the same models that have been employed for the specific datasets has been carried out. Tables II and III provide a summary of those statistical performances. More specifically, the CB-FWNN scheme has been compared against an autoregressive linear model (AR), a multilayer perceptron utilizing an adaptive learning rate (ABP), a spread encoding multilayer neural network (SE), a window random activation weight neural network (WRAWN), a radial basis function (RBF) network and the proposed in this section CB-FNN.

From these four schemes, only SE and RBF managed to provide a "similar" but inferior to CB-FWNN performance, however with high training time computational cost. Compared to the proposed CB-FWNN structure, the above mentioned methodologies were also criticized by their large input dimensionality (*i.e.* 6-8 input variables) for performances shown in Tables 6.2 and 6.3. The alternative also CB-FNN structure performed also very satisfactory. The MRPE term provides information on how close forecasts or predictions are to the eventual outcomes. The MRPE is an index that provides information about the bias of the model. A value of zero means that there is no bias in predictions. Positive values indicate under-prediction of the power load, i.e. the predicted values are lower than the observed, and thus the model is 'fail-safe'.

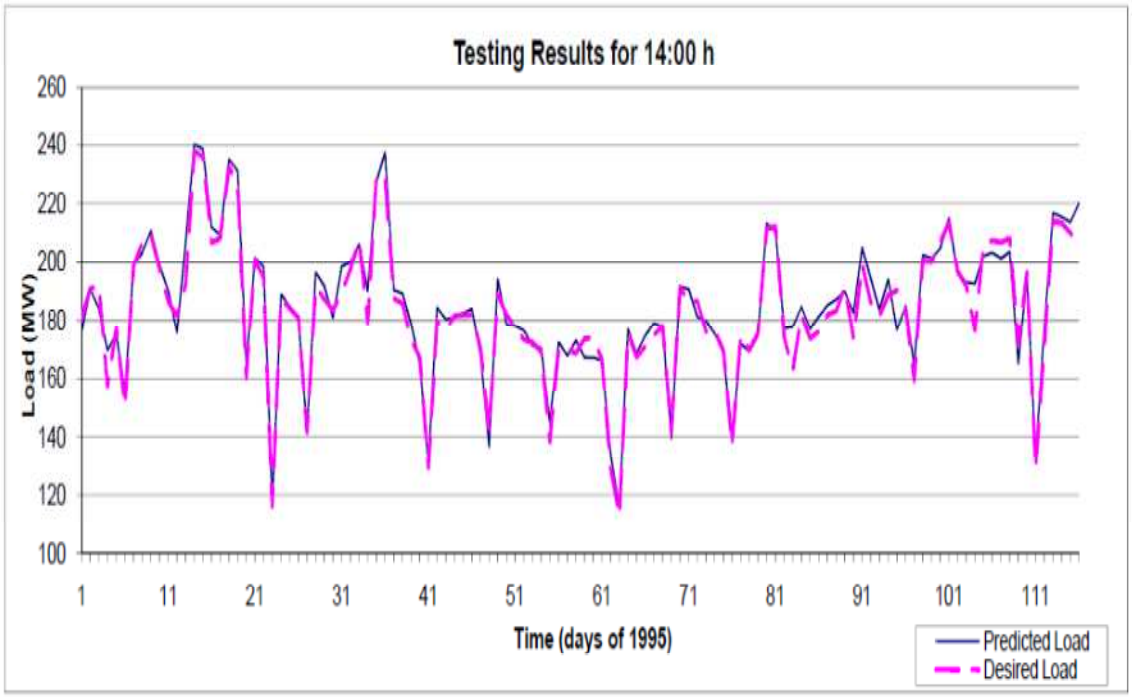


Fig 6.8 - Testing performance for max load

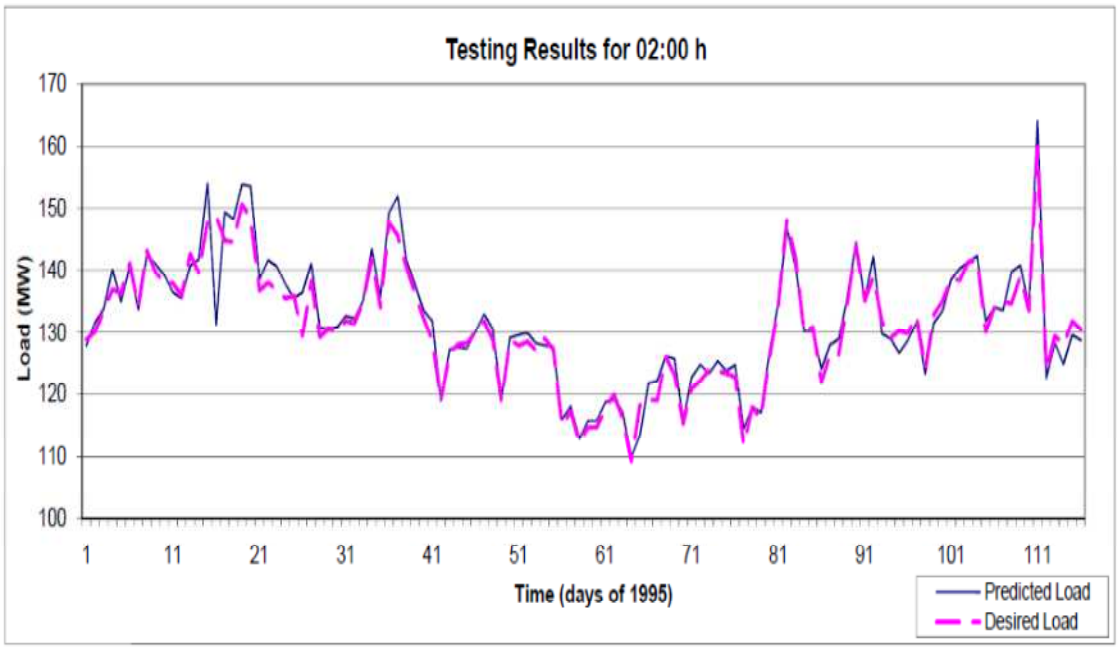


Fig 6.9 - Testing performance for min load

TABLE 6.2– Comparison of Performance Indices for 2hr STLF for various methods

Statistical index Testing Dataset 2hour	**CB-FNN 2h**	**AR 2h**	**ABP 2h**	**SE 2h**	**WRAWN 2h**	**RBF 2h**
RMSE	1.9223	5.8020	3.8063	2.0314	4.0633	1.8672
Mean relative percentage error (MRPE)(%)	1.3104	3.8630	2.7346	1.5174	2.7542	1.3512
MAPE (%)	0.4939	0.5715	1.0849	1.0632	0.6675	0.4706
Standard error of prediction (SEP) (%)	2.0299	5.7642	3.8215	2.0728	4.0916	1.9086
Bias factor (B_f)	1.0048	1.0041	1.0102	1.0105	1.0059	1.0045
Accuracy factor (A_f)	1.0132	1.0388	1.0275	1.0152	1.0277	1.0136

Negative values indicate over-prediction, i.e. the model over-estimates power load and thus is 'fail-dangerous'. CB-FWNN achieved a very good performance, by scoring 1.9964% and 1.2826% for 14h and 2h respectively. Comparing with CB-FWNN, the alternative methods cannot match the same performance especially in the case of 14h. Tables 6.2 and 6.3 provide a good indication of their performances.

TABLE 6.3 – Comparison of Performance Indices for 14hr STLF for various methods

Statistical index Testing Data set 14hour	**CB-FNN 14**	**AR 14**	**ABP 14**	**SE 14**	**WRAWN 14**	**RBF 14**
Mean relative percentage error (MRPE) (%)	2.3183	11.1837	11.9674	3.4001	10.5902	2.7924
MAPE (%)	0.8731	1.3304	6.4261	1.1397	1.3385	0.8927
Standard error of prediction (SEP) (%)	3.1998	15.0745	14.3387	4.5409	12.8006	3.4007
Bias factor (B_f)	1.0052	0.9993	1.0525	1.0104	1.0033	1.0073
RMSE	3.3069	17.2049	17.6778	4.6194	15.1935	3.6061
Accuracy factor (A_f)	1.0236	1.1173	1.1180	1.0343	1.1071	1.0281

It is clear that the SE network outperformed the ABP, while the RBF network proves its traditional superiority against MLP-style networks. This statistic is similar to the bias factor (B_f) introduced by Ross [92]. Models describing predictions (B_f) within the range of $\{0.9-1.05\}$ could be considered good, in the range of $\{0.7-0.9\} \| \{1.06-1.15\}$ are considered acceptable, while for $\{<0.7 \parallel >1.15\}$ are considered unacceptable. Bias factor is a multiplicative factor that compares model predictions and is used to determine whether the model over- or under-predicts the power consumption. In this case a B_f value greater than 1 indicates that the model over-estimates load and is thus 'fail-dangerous', whereas a value less that 1 indicates under-prediction of load and thus a 'fail-safe' model

The B_f parameters of all models were in an acceptable range; however the related parameter for CB-FWNN was just under the optimal 1.0, providing thus a fail-safe condition.

SEP index is determined as the relative deviation of the mean prediction values and it has the advantage of being independent on the magnitude of the measurements [19]. Based on this index, the CB-FWNN scheme achieved again a very good performance for both cases. Only for the 2h case, the RBF network was slightly better than the proposed scheme. For the 14h case, although RBF and SE were superior to ABP and WRAWN models, they both fall behind to the CB-FWNN.

The accuracy factor (A_f), is a simple multiplicative factor that indicates the spread of results about the prediction. A value of one indicates that there is perfect agreement between all the predicted and measured values. Table I shows the accuracy factor values obtained for the two testing datasets. The relevant figures for A_f indicate again better performances for the CB-FWNN scheme, which is more evident at the 14h load case. The MAPE term provides information about the average deviation from the observed value and it is similar to the accuracy factor (A_f). Based on this index, the average deviation of the predicted power load values for the CB-FWNN case was 0.86% and 0.49% for 14h and 2h load cases.

The results of MAPE were in good agreement with the values of the accuracy factor (A_f) estimated for both data sets. Some differences between the two indices can be attributed to different computational methods followed. The above comparison results reveal the superiority of the proposed CB-FWNN scheme in terms of modelling accuracy and training speed. The CB-FNN performed also satisfactory however its accuracy was lower to CB-FWNN.

Chapter 7

Conclusions and Future Enhancements

Soft computing approaches have been developed and applied to many scientific and engineering areas in recent years. There have been also many successful researches for the identification and modelling of nonlinear dynamic systems by using various soft computing techniques with different computational architecture. In the early stages of this research (chapter 3), inspired by the method presented by Theocharis *et all* in the frame of Neuro-Fuzzy schemes, an adaptive modelling structure created. The adaptive structure evolved and adjusted in an online manner to reduce as much as possible the network redundancies. Hence, since the models are created automatically and not pre-designed, the difficulties in determining of the architecture of soft computing models can be avoided to some extent. *Fungus growth* modelling, a real food data analysis problem, was examined in presence of three inputs; temperature, water activity and pH and relatively better results based on standard various error criteria achieved.

In the literature, an amount of work exploring the hybrid learning algorithms to identify the structure parameters and also novel structure of wavelet-based neural networks has been reported. The well established real world problems revealed the fact that a great deal of further research is still needed. Throughout main parts of this thesis, we have attempted to identify flaws within existing applications and structures of Wavelet-based variants of neural networks. The first structure and training algorithm proposed in Chapter 4 approaches a simple by efficient techniques applicable to all types of NNs with wavelet family. The incorporation of Linear Combination Weights on hidden-output connection links boosts the output accuracy and also training speed to a higher level. In the same scheme, a Hybrid Learning Algorithm was implemented to tackle the main problem associated with the use of GD algorithm, i.e. the problem of low convergence rate.

Based on the first structure, a more advanced structure of WNNs proposed MWNN-LCW which appeared to be more modular. The alteration on the structure makes not only the structure more interpretable but also leads to much higher accuracy.

Both algorithms were then utilised to process real data provided via ongoing research collaboration with Agricultural University of Athens-Greece relating to *Lysteria Monocytogenes Bacteria* Count prediction. In conjunction with various types of WNNs and also other soft computing techniques, the proposed technique produces out-performing results according to accuracy and computational cost.

Despite the excellent performance of the proposed WNNs, research has highlighted some limitations to the approach. The first was a significant increase in the number of neurons in higher dimensional data. This yields that, although it was an improvement in convergence speed and accuracy but the network's tuneable parameters dramatically "grow" with network size which can be prohibitive due to memory constraints. The second was that still the appropriate number of neurons for each input was a matter of randomness and on trial and error basis.

The methodology proposed in Chapter 6, gearing forward a long way toward covering aforementioned drawbacks. A clustering-based FWNN approach has been presented. Choosing a clustering algorithm, itself, can be a challenging task. Most structures generally assume some explicit structure in the dataset. However, usually we have little or no information regarding the structure, which is, paradoxically, what someone wants to uncover. Two clustering methods have been utilised in serial form, i.e. Subtractive Clustering and Mixture Densities with Expectation Maximization. Initially, the subtractive clustering was used as a pre-processing technique to find out initial set and adequate number of clusters and ultimately number of neurons in each sub-WNN inside wavelet network, an optimum number of neurons can strongly influence the time required to obtain a solution and then the GMM-EM was responsible for forming the multi-dimensional Gaussians, which later used as multidimensional membership functions. The rationale behind the choice of former one was its ability to detect number of clusters without any prior knowledge of the data and the latter one selected due to the platform it can creates for feeding back the desired-predicted output error into the clustering process and also deriving the membership degree directly from clusters without the need of projecting them on the input's axis. Two learning algorithms namely, Weighted Least Square and Extended Kalman Filter were incorporated to adjust the linear and non-linear parameters respectively.

We demonstrated the usefulness and benefits of the network by applying it to a real database for dynamic system modelling. The case study was related to the *Power Load forecasting for Greek Island of Crete*. Prediction of the maximum load at 14:00 and minimum load at 02:00 was set as a

target. Comprehensive comparison conducted against some popular existing methods for both cases in accordance to several error criteria.

There are several topics that have been left out of the scope in the present dissertation which can be considered as a future work. Future enhancements of this research's field can be aligned in several directions. Points mentioned below, are just some of the possible extensions can be done as future work.

Linguistic Data/Patterns : The term Computing With Words (CWW), pioneered by Zadeh in the mid-1990s[161]. In principle, the proposed schemes are only applicable to numerical data (or linguistic data converted to numerical data) but for the humans the only fully natural means of articulation and communication is natural language. In coming years the use of words in place of numbers is destined to be in centre of attention. This is certain to happen, therefore, maybe we could develop models, tools, techniques, algorithms, etc., that could operate on natural language (words) and can serve the same (or similar) purpose as their numerical counterparts, i.e., maybe instead of traditional computing with numbers (from measurements), it would be better to compute with words (from perceptions). Therefore, we may skip an "artificial'' interface and try to operate on what is human-specific: natural language[161, 162]. Perhaps the most direct extension of this work is by the means modifying the proposed structures to deal with this type of data.

Adaptive structure/parameter learning: Off-line clustering methods - as we did in chapter 4 & 6 - require that data be ready before the modelling. Obviously, it is difficult for human experts to examine all the input–output data from a real complex system to find a number of proper rules for the fuzzy system. Hence, an immune way to the above-mentioned deficiency is online identification methods appeared in the literature and several methods proposed [163]. Generally, these approaches consist of two learning phases, the structure learning phase and the parameter learning phase. These two phases are done simultaneously. In terms of structure identification, there are no rules initially in an online structure. They are created and adapted as on-line learning proceeds via simultaneous structure and parameter identification. This idea was touched upon at the end of Chapter 3. However, the goal in this case is to develop an online clustering method along with an online WNN structure. Extended Kalman Filter as a learning algorithm provides us with a potential platform for online learning however Weighted Least Square should be replaced

by a recursive variant of it. An adaptive type of Expectation Maximization also proposed in [126], which can be modified accordingly.

Recurrent connections: Most of these problems demand nonlinear adaptive systems which can learn from observed data. Recurrent wavelet neural networks with arbitrarily connected neurons have great potential to meet this demand. However, how to train recurrent WNN networks effectively remains an open problem, which hinders wide applications of recurrent networks in the aforementioned areas. As a future step of the work presented in this Thesis, we could extend the work to WNN recurrent networks. Our goal is twofold. First, we would like to develop a framework for fully recurrent networks based on MWNN-LCW proposed in chapter 4 with internal and external loops. Related suggestions can be found in [164-166]. Second, we would like to apply a learning algorithm for training that recurrent wavelet network so that avoid divergence occurring over tuning such an autonomous system.

Incremental Learning algorithm: The learning algorithm should be able to supply a way that enables clustering part and consequence WNN part to accommodate new data, including examples that correspond to previously unseen dataset. It means, the learning algorithm should be in such a way that does not require access to previously used data during subsequent incremental learning sessions, while at the same time, preserving the knowledge learnt by the network on previous learning[167]. There may even be further improvements to be imposed, i.e. agents should not only acquire new knowledge but also modify or delete old knowledge. However, these modification and deletion are not always efficient in learning; hence embedding this type algorithm could be a challenge. Incremental learning has been addressed in a few numbers of published papers and on very primitive structural platforms [168, 169]. Expanding these types of algorithms over hybrid methods is one that would certainly merit further investigations.

Bibliography

[1] Q. Zhang and A. Benveniste, "Wavelet networks," *IEEE Transactions on Neural Networks*, vol. 3, pp. 889-898, 1992.

[2] D. W. C. Ho, P. A. Zhang, and J. Xu, "Fuzzy wavelet networks for function learning," , *IEEE Transactions on Fuzzy Systems*, vol. 9, pp. 200-211, 2001.

[3] I. Nagrath, *Control systems engineering*: New Age International, 2005.

[4] L. Ljung, *System Identification: Theory for the user*. Englewood Cliffs: Prentice Hall, 1987.

[5] S. M. Biyiksiz, "ARMA modeling based on partial AR and MA parameter approximation," in *Spectrum Estimation and Modeling, 1988., Fourth Annual ASSP Workshop on*, Minneapolis, MN USA, 1988, pp. 397-401.

[6] A. O. Steinhardt, "The partial realization problem for moving average models," in *Acoustics, Speech, and Signal Processing, 1988. ICASSP-88., 1988 International Conference on*, New York, NY USA, 1988, pp. 2360 - 2363.

[7] G. P. Zhang, "Time series forecasting using a hybrid ARIMA and neural network " *Neurocomputing*, vol. 50, pp. 159-175, 2003.

[8] O. Nelles, *Nonlinear system identification*, 1 ed. vol. 1. Berlin: Springer - Verlag, 2000.

[9] F. M. Ham and I. Kostanic, *Principles of Neurocomputing for science and engineering*. New York: McGraw Hill, 2001.

[10] Y.Chen, "Hybrid softcomputing approach to identification and control of nonlinear systems,Ph.D.Thesis,Kumamoto University," 2001.

[11] J. Vieira, F. M. Dias, and A. Mota, "Neuro-Fuzzy Systems: A Survey," 2004.

[12] J. C. Patra, R. N. Pal, B. Chatterji, and G. Panda, "Identification of nonlinear dynamic systems using functional link artificial neural networks," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, ,* vol. 29, pp. 254-262, 2002.

[13] D. J. Montana and L. Davis, "Training feedforward neural networks using genetic algorithms," 1989, MA Cambridge,Citeseer, Vol.123

[14] L. H. Tsoukalas and R. E. Uhrig, *Fuzzy and neural approaches in engineering*: John Wiley & Sons, Inc. New York, NY, USA, 1996.

[15] K. Shahida, Ibraheem, Moinuddin, and M. Farooq, "A table lookup scheme for fuzzy logic based model identification applied to time series prediction," in *Information Fusion, 2003. Proceedings of the Sixth International Conference of*, 2003, pp. 1449-1456.

[16] A. Saad, "An Overview of Hybrid Soft Computing Techniques for Classifier Design and Feature Selection," in *Hybrid Intelligent Systems, 2008. HIS '08. Eighth International Conference on*, 2008, pp. 579-583.

[17] D. Nauck, F. Klawonn, and R. Kruse, *Foundations of neuro-fuzzy systems*: John Wiley & Sons, Inc. New York, NY, USA, 1997.

[18] C. da Costa Pereira and A. Tettamanzi, "Fuzzy-Evolutionary Modeling for Single-Position Day Trading Natural Computing in Computational Finance." vol. 100, A. Brabazon and M. O'Neill, Eds., ed: Springer Berlin / Heidelberg, 2008, pp. 131-159.

[19] X. Yao, "Evolving artificial neural networks," *Proceedings of the IEEE*, vol. 87, pp. 1423-1447, 1999.

[20] A. Azzini and A. G. B. Tettamanzi, "A neural evolutionary approach to financial modeling," 2006, Proceedings of the 8th annual conference on Genetic and evolutionary Computation 1-59593-186-4 Seattle, Washington, USA, pp 1605-1612, 2006

[21] A. Michael and H. Takagi, "Dynamic control of genetic algorithms using fuzzy logic

techniques," Computer Science Department University of California Davis, CA 95616, 1993, pp. 76-83.
[22] F. Herrera, M. Lozano, and J. L. Verdegay, "Tuning fuzzy logic controllers by genetic algorithms* 1," *International Journal of Approximate Reasoning,* vol. 12, pp. 299-315, 1995.
[23] P. P. Bonissone, "Soft computing: the convergence of emerging reasoning technologies," *Soft Computing-A Fusion of Foundations, Methodologies and Applications,* vol. 1, pp. 6-18, 1997.
[24] O. Cordón and F. Herrera, "A hybrid genetic algorithm-evolution strategy process for learning fuzzy logic controller knowledge bases," 1996. in *Genetic Algorithms and Soft Computing.* Berlin, Germany: Physica- Verlag, 1996, pp. 251–278.
[25] O. Cordón, F. Gomide, F. Herrera, F. Hoffmann, and L. Magdalena, "Ten years of genetic fuzzy systems: current framework and new trends," *Fuzzy Sets and Systems,* vol. 141, pp. 5-31, 2004.
[26] A. S. Moussa, "The Implementation of Intelligent OoS Networking by the Development and Utilization of Novel Cross-Disciplinary Soft Computing Theories and Techniques," Ph.D Dissertation at the Florida State University, 2003.
[27] W. Bellil, C. B. Amar, and A. M. Alimi, "Comparison between beta wavelets neural networks, RBF neural networks and polynomial approximation for 1D, 2D functions approximation," *Trans. on Eng., Comp. and Tech,* vol. 13, pp. 102-107, 2006.
[28] V. Kreinovich, O. Sirisaengtaksin, and S. Cabrera, "Wavelet neural networks are asymptotically optimal approximators for functions of one variable," in *Proc. IEEE Int. Conf. Neural Networks*, Orlando, FL, June 1994, vol. 1,pp. 299–304.
[29] M. Singh, S. Srivastava, M. Hanmandlu, and J. R. P. Gupta, "Type-2 fuzzy wavelet networks (T2FWN) for system identification using fuzzy differential and Lyapunov stability algorithm," *Applied Soft Computing,* vol. 9, pp. 977-989, 2009.
[30] L. Fausett, *Fundamentals of neural networks*, 1 ed. vol. 1. New Jersey: Prentice - Hall, 1994.
[31] C.-T. Lin and C. S. G. Lee, *Neural Fuzzy Systems*. New Jersey: Prentice - Hall, 1996.
[32] H. Esen, M. Inalli, A. Sengur, and M. Esen, "Performance prediction of a ground-coupled heat pump system using artificial neural networks," *Expert Systems with Applications,* vol. 35, pp. 1940-1948, 2008.
[33] J. A. K. Suykens, J. Vandewalle, and B. L. R. De Moor, *Artificial neural networks for modelling and control of non-linear systems*: Springer, 1996.
[34] C. T. Lin and C. S. G. Lee, *Neural fuzzy systems: a neuro-fuzzy synergism to intelligent systems*: Prentice-Hall, Inc. Upper Saddle River, NJ, USA, 1996.
[35] V. B. Daohang Sha, "An on-line hybrid learning algorithm for multilayer perceptron in identification problems," *Elsevier Computers and Electrical Engineering,* 2002.
[36] R. Battiti, "First-and second-order methods for learning: between steepest descent and Newton's method," *Neural computation,* vol. 4, pp. 141-166, 1992.
[37] L. Fausett, *Fundamentals of neural networks: architectures, algorithms, and applications*: Prentice-Hall, Inc. Upper Saddle River, NJ, USA, 1994.
[38] K.-B. Kim, D.-J. Yoon, J.-C. Jeong, and S.-S. Lee, "Determining the stress intensity factor of a material with an artificial neural network from acoustic emission measurements," *NDT & E International,* vol. 37, pp. 423-429, 2004.
[39] S. I. Ao, *Applied Time Series Analysis and Innovative Computing* vol. 59: Springer Verlag, 2010.

[40] V. Kodogiannis and A. Lolis, "Prediction of foreign exchange rates by neural network and fuzzy system based techniques," in *European Symposium on Artificial Neural Networks Bruges (Belgium),*, 2001.
[41] E. Cheng, H. Jin, Z. Han, and J. Sun, "Network-Based Anomaly Detection Using an Elman Network," in *Networking and Mobile Computing*. vol. 3619, X. Lu and W. Zhao, Eds., ed: Springer Berlin / Heidelberg, 2005, pp. 471-480.
[42] M. Powell, "The theory of radial basis function approximation in 1990," *DAMTP Report*, no. NA11, Dec. 1990.
[43] J. Moody and C. Darken, *Learning with localized receptive fields*: Yale Univ., Dept. of Computer Science, 1988.
[44] E. Z. Panagou, V. Kodogiannis, and G. J. Nychas, "Modelling fungal growth using radial basis function neural networks: the case of the ascomycetous fungus Monascus ruber van Tieghem," *International journal of food microbiology*, vol. 117, pp. 276-86, Jul 15 2007.
[45] S. Chen, C. Cowan, and P. Grant, "Orthogonal least squares learning algorithm for radial basis function networks," *IEEE Transactions on Neural Networks*, vol. 2, pp. 302-309, 2002.
[46] S. Chen, P. M. Grant, and C. F. N. Cowan, "Orthogonal least squares algorithm for training multi-output radial basis function networks," in *Artificial Neural Networks, Second International Conference on*, 1991, pp. 336-339.
[47] D. Radojević, A. Perović, Z. Ognjanović, and M. Rašković, "Interpolative Boolean Logic," in *Artificial Intelligence: Methodology, Systems, and Applications*. vol. 5253, D. Dochev, M. Pistore, and P. Traverso, Eds., ed: Springer Berlin / Heidelberg, 2008, pp. 209-219.
[48] K. Tahera, R. N. Ibrahim, and P. B. Lochert, "A fuzzy logic approach for dealing with qualitative quality characteristics of a process," *Expert Systems with Applications*, vol. 34, pp. 2630-2638, 2008.
[49] S. Saraswati, P. K. Agarwal, and S. Chand, "Neural networks and fuzzy logic-based spark advance control of SI engines," *Expert Systems with Applications*, vol. 38, pp. 6916-6925, 2011.
[50] V. S. Kodogiannis, M. Boulougoura, E. Wadge, and J. N. Lygouras, "The usage of soft-computing methodologies in interpreting capsule endoscopy," *Engineering Applications of Artificial Intelligence*, vol. 20, pp. 539-553, 2007.
[51] G. R. G. Pe, R. D. Fellow, F. Cardullo, and N. Y. Binghamton, "Application of Neuro-Fuzzy Systems to Behavioral Representation in Computer Generated Forces." Proceedings of the eighth conference on computer generated forces and behavioural representation (May 11–13, Orlando, Florida) , 575–585.
[52] J. S. R. Jang, "ANFIS: Adaptive-network-based fuzzy inference system," *IEEE Transactions on systems, man and cybernetics*, vol. 23, pp. 665-685, 1993.
[53] M. A. Shoorehdeli, M. Teshnehlab, A. K. Sedigh, and M. A. Khanesar, "Identification using ANFIS with intelligent hybrid stable learning algorithm approaches and stability analysis of training methods," *Applied Soft Computing*, vol. 9, pp. 833-850, 2009.
[54] F. M. Frattale Mascioli, G. M. Varazi, and G. Martinelli, "Constructive algorithm for neuro-fuzzy networks," in *Fuzzy Systems, 1997., Proceedings of the Sixth IEEE International Conference on*, 1997, vol.1,pp. 459-464.
[55] J. S. R. Jang and E. Mizutani, "Levenberg-Marquardt method for ANFIS learning," in: Proc. Biennial Conf. of the North American Fuzzy Information Processing Society NAFIPS'96, Berkeley, CA, IEEE, New York, 1996, pp. 87-91, pp. 87-91.
[56] M. Kumar and D. P. Garg, "Intelligent learning of fuzzy logic controllers via neural

network and genetic algorithm," In: Proceeding of 2004 JUSFA, Japan–USA symposium on flexible automation. pp. 19-21.
[57] D. Nauck and R. Kruse, "NEFCON-I: an X-Window based simulator for neural fuzzy controllers," in *Neural Networks, 1994. IEEE World Congress on Computational Intelligence., 1994 IEEE International Conference on*, 1994, vol.3,pp. 1638-1643.
[58] A. Nürnberger, D. Nauck, and R. Kruse, "Neuro-fuzzy control based on the NEFCON-model: recent developments," *Soft Computing - A Fusion of Foundations, Methodologies and Applications*, vol. 2, pp. 168-182, 1999.
[59] J. Theocharis and G. Vachtsevanos, "Adaptive Fuzzy Neural Networks as identifiers of discrete-time nonlinear dynamic systems," *Journal of Intelligent and Robotic Systems*, vol. 17, pp. 119-168, 1996.
[60] M. Amina, V. Kodogiannis, and A. Tarczynski, "Predictive modeling in food mycology using adaptive neuro-fuzzy systems," in *Computer Systems and Applications, 2009. AICCSA 2009. IEEE/ACS International Conference on*, 2009, pp. 821-828.
[61] T. Ross and T. McMeekin, "Predictive microbiology," *International journal of food microbiology*, vol. 23, pp. 241-264, 1994.
[62] A. M. Gibson and A. D. Hocking, "Advances in the predictive modelling of fungal growth in food," *Trends in Food Science & Technology*, vol. 8, pp. 353-358, 1997.
[63] P. A. Murphy, S. Hendrich, C. Landgren, and C. M. Bryant, "Food Mycotoxins: An Update," *Journal of Food Science*, vol. 71, pp. R51-R65, 2006.
[64] H. Cuppers, S. Oomes, and S. Brul, "A model for the combined effects of temperature and salt concentration on growth rate of food spoilage molds," *Applied and Environmental Microbiology*, vol. 63, p. 3764, 1997.
[65] M. N. Hajmeer, I. A. Basheer, and Y. M. Najjar, "Computational neural networks for predictive microbiology II. Application to microbial growth," *International journal of food microbiology*, vol. 34, pp. 51-66, 1997.
[66] I. Basheer and M. Hajmeer, "Artificial neural networks: fundamentals, computing, design, and application," *Journal of Microbiological Methods*, vol. 43, pp. 3-31, 2000.
[67] R. M. Garcíaa-Gimeno, C. Hervás-Martíanez, E. Barco-Alcalá, G. Zurera-Cosano, and E. Sanz-Tapia, "An Artificial Neural Network Approach to Escherichia Coli O157:H7 Growth Estimation," *Journal of Food Science*, vol. 68, pp. 639-645, 2003.
[68] X. Jiang, "Dynamic fuzzy wavelet neural network for system identification, damage detection and active control of highrise buildings," PhD final Thesis. The Ohio State University, 2005.
[69] L. Cheng-Jian, "Wavelet Neural Networks with a Hybrid Learning Approach," *Journal of Information Science and Engineering*, vol. 22, pp. 1367-1387, 2006.
[70] S. Suhartono, "Development of Model Building Procedures in Wavelet Neural Networks for Forecasting Non-Stationary Time Series," *European Journal of Scientific Research*, vol. 34, pp. 416-427, 2009 2009.
[71] G. Lekutai and H. F. vanLandingham, "Self-tuning control of nonlinear systems using neural network adaptive frame wavelets," in *Systems, Man, and Cybernetics, 1997. 'Computational Cybernetics and Simulation'., 1997 IEEE International Conference on*, 1997, vol.2,pp. 1017-1022.
[72] J. Zhang, G. G. Walter, Y. Miao, and W. N. W. Lee, "Wavelet neural networks for function learning," *IEEE Transactions on Signal Processing*, vol. 43, pp. 1485-1497, 2002.
[73] M. Amina, E. Z. Panagou, V. S. Kodogiannis, and G. J. E. Nychas, "Wavelet Neural Networks for modelling high pressure inactivation kinetics of Listeria monocytogenes in

UHT whole milk," *Chemometrics and Intelligent Laboratory Systems*, vol. In Press, Accepted Manuscript.
[74] V. Kodogiannis, I. Petrounias, and J. Lygouras, "Adaptive Wavelet Neural Networks for Non-linear Modelling and Control," *NEURAL PARALLEL AND SCIENTIFIC COMPUTATIIONS*, vol. 15, p. 221, 2007.
[75] Y. Chen, "Stock Index Modelling using EDA based local Linear neural Network," in *IEEE*, 2005.
[76] Y. Chen, "Time-series prediction using a local linear wavelet neural network," *Elsevier, Neurocomputing*, vol. 69, pp. 449-465, 2006.
[77] C. Yuehui, D. Xiaohui, and Z. Yaou, "Stock Index Modeling using EDA based Local Linear Wavelet Neural Network," in *Neural Networks and Brain, 2005. ICNN&B '05. International Conference on*, 2005, pp. 1646-1650.
[78] Y. Chen, B. Yang, and J. Dong, "Time-series prediction using a local linear wavelet neural network," *Neurocomputing*, vol. 69, pp. 449-465, 2006.
[79] M. F. A. A Banakar, "Artificial Wavelet Neural Network and its application in Neuro-fuzzy models," *Elsevier Applied Soft Computing*, 2008.
[80] Y. Z. Jin Jiang, "A revisit block and recursive least squares for parameter estimation," *Elsevier Computers and Electrical Engineering*, vol. 30, pp. 403-416, 2004.
[81] H. T. Nguyen, N. R. Prasad, and E. A. Walker, *A first course in fuzzy and neural control*: CRC Pr I Llc, 2002.
[82] K. M. Passino and S. Yurkovich, *Fuzzy control*: Citeseer, 1998.
[83] N. A. M. I. Mohammd Subhi Al-batah, "Modified Recursive Leat Squares Algorithm to train the Hybrid Multilayred Perceptron Network," *Elsevier Applied Soft Computing*, 2009.
[84] J. Farber and P. Peterkin, "Listeria monocytogenes, a food-borne pathogen," *Microbiology and Molecular Biology Reviews*, vol. 55, p. 476, 1991.
[85] J. McLauchlin, R. Mitchell, W. Smerdon, and K. Jewell, "Listeria monocytogenes and listeriosis: a review of hazard characterisation for use in microbiological risk assessment of foods," *International journal of food microbiology*, vol. 92, pp. 15-33, 2004.
[86] C. Little, F. Taylor, S. Sagoo, I. Gillespie, K. Grant, and J. McLauchlin, "Prevalence and level of Listeria monocytogenes and other Listeria species in retail pre-packaged mixed vegetable salads in the UK," *Food microbiology*, vol. 24, pp. 711-717, 2007.
[87] D. A. Nolan, D. C. Chamblin, and J. A. Troller, "Minimal water activity levels for growth and survival of Listeria monocytogenes and Listeria innocua," *International journal of food microbiology*, vol. 16, pp. 323-335, 1992.
[88] Y. Huang, *Automation for food engineering*: CRC, 1998.
[89] M. Zwietering, I. Jongenburger, F. Rombouts, and K. Van't Riet, "Modeling of the bacterial growth curve," *Applied and Environmental Microbiology*, vol. 56, p. 1875, 1990.
[90] I. Albert and P. Mafart, "A modified Weibull model for bacterial inactivation," *International journal of food microbiology*, vol. 100, pp. 197-211, 2005.
[91] K. P. Singh, P. Ojha, A. Malik, and G. Jain, "Partial least squares and artificial neural networks modeling for predicting chlorophenol removal from aqueous solution," *Chemometrics and Intelligent Laboratory Systems*, vol. 99, pp. 150-160, 2009.
[92] T. Ross, "Indices for performance evaluation of predictive models in food microbiology," *Journal of Applied Microbiology*, vol. 81, pp. 501-508, 1996.
[93] L. Coroller, I. Leguerinel, E. Mettler, N. Savy, and P. Mafart, "General model, based on two mixed Weibull distributions of bacterial resistance, for describing various shapes of inactivation curves," *Applied and Environmental Microbiology*, vol. 72, p. 6493, 2006.

[94] Y. Le Marc, C. Pin, and J. Baranyi, "Methods to determine the growth domain in a multidimensional environmental space," *International journal of food microbiology*, vol. 100, pp. 3-12, 2005.
[95] M. Te Giffel and M. Zwietering, "Validation of predictive models describing the growth of Listeria monocytogenes," *International journal of food microbiology*, vol. 46, pp. 135-149, 1999.
[96] E. Z. Panagou and V. S. Kodogiannis, "Application of neural networks as a non-linear modelling technique in food mycology," *Expert Systems with Applications*, vol. 36, pp. 121-131, 2009.
[97] A. Banakar and M. F. Azeem, "Comparative Study of Different Type of Wavelet in Artificial Wavelet Neuro-Fuzzy Model," in *Adaptive and Learning Systems, 2006 IEEE Mountain Workshop on*, 2006, pp. 165-170.
[98] R. Xu and D. Wunsch, "Survey of clustering algorithms," *Neural Networks, IEEE Transactions on*, vol. 16, pp. 645-678, 2005.
[99] S. Miyamoto, H. Ichihashi, and K. Honda, "Algorithms for fuzzy clustering," ed: Springer, Heidelberg, 2008.
[100] R. Babuska, *Fuzzy modeling for control*: Kluwer Academic Publishers Norwell, MA, USA, 1998.
[101] W. Pedrycz, *Knowledge-based clustering*: Wiley Online Library, 2005.
[102] G. Fung, "A comprehensive overview of basic clustering algorithms," Technical report. http://www.cs.wisc.edu/¡«gfung/ clustering.pdf (accessed October 1, 2009).
[103] C. Borgelt and R. Kruse, "Shape and size regularization in expectation maximization and fuzzy clustering," *Knowledge Discovery in Databases: PKDD 2004*, pp. 52-62, 2004.
[104] J. C. Dunn, "A Fuzzy Relative of the ISODATA Process and Its Use in Detecting Compact Well-Separated Clusters," *Journal of Cybernetics*, vol. 3, pp. 32 - 57, 1973.
[105] J. C. Bezdek, *Pattern recognition with fuzzy objective function algorithms*: Kluwer Academic Publishers Norwell, MA, USA, 1981.
[106] N. R. Pal and J. C. Bezdek, "On cluster validity for the fuzzy c-means model," *Fuzzy Systems, IEEE Transactions on*, vol. 3, pp. 370-379, 1995.
[107] F. Höppner, F. Klawonn, R. Kruse, and T. Runkler, *Fuzzy cluster analysis*: John Wiley & Sons Chichester, UK, 1999.
[108] E. G. Donald and C. K. William, "Fuzzy clustering with a fuzzy covariance matrix," in *Decision and Control including the 17th Symposium on Adaptive Processes, 1978 IEEE Conference on*, 1978, pp. 761-766.
[109] O. Georgieva and D. Filev, "Gustafson-Kessel algorithm for evolving data stream clustering," international conference computer systems and technologies for PhD students in computing 2009, pp. 1-6.
[110] R. Babuška, P. Van der Veen, and U. Kaymak, "Improved covariance estimation for Gustafson-Kessel clustering," 2002, pp. 1081–1085.
[111] I. Gath and A. B. Geva, "Unsupervised optimal fuzzy clustering," *Pattern Analysis and IEEE Transactions on Machine Intelligence* , vol. 11, pp. 773-780, 1989.
[112] H. Soleimani-B, C. Lucas, and B. Araabi, "Recursive Gath–Geva clustering as a basis for evolving neuro-fuzzy modeling," *Evolving Systems*, vol. 1, pp. 59-71, 2010.
[113] M. Alata, M. Molhim, and A. Ramini, "Optimizing of fuzzy c-means clustering algorithm using GA," Proceedings of World Academy of Science, Engineering and Technology, vol. 39, 2008 .

[114] A. Priyono, M. Ridwan, A. J. Alias, R. Atiq, K. Rahmat, A. Hassan, M. Ali, and M. Alauddin, "Generation Of Fuzzy Rules With Subtractive Clustering," *Jurnal Teknologi (D)*, vol. 43, pp. 143-153, 2005.

[115] X. Sheng-Wu, N. Xiao-Xiao, and L. Hong-Bing, "Support vector machines based on subtractive clustering," in *Machine Learning and Cybernetics, 2005. Proceedings of 2005 International Conference on*, 2005, Vol. 7,pp. 4345-4350.

[116] D. Jiamei, R. Stobart, and A. Plianos, "Combined hybrid clustering techniques and neural fuzzy networks to predict diesel engine emissions," in *Systems, Man and Cybernetics, 2007. ISIC. IEEE International Conference on*, 2007, pp. 3609-3614.

[117] K. Hammouda and F. Karray, "A comparative study of data clustering techniques," *Tools of intelligent systems design. In Course Project, SYDE*, vol. 625.

[118] S. L. Chiu, "Fuzzy model identification based on cluster estimation," *Journal of intelligent and Fuzzy systems*, vol. 2, pp. 267-278, 1994.

[119] C. Borgelt, "Prototype-based classification and clustering," PhD Thesis, Otto-von-Guericke-Universität Magdeburg, Universitätsbibliothek, 2006.

[120] S. Y. Kung, M. W. Mak, and S. H. Lin, "Biometric authentication: a machine learning approach," *Prentice Hall Information And System Sciences Series*, p. 496.

[121] C. Borgelt and R. Kruse, "Fuzzy and probabilistic clustering with shape and size constraints," In: Proceedings of the 11th International Fuzzy Systems Association World Congress, IFSA'05, Beijing, China, pp. 945–950. 2005.

[122] C. Tomasi, "Estimating gaussian mixture densities with EM: A tutorial," *Duke University*, 2004. http://www.cs.duke.edu/courses/spring04/cps196.1/handouts/EM/tomasiEM.pdf

[123] I. J. Myung, "Tutorial on maximum likelihood estimation," *Journal of Mathematical Psychology*, vol. 47, pp. 90-100, 2003.

[124] S. Borman, "The Expectation Maximization Algorithm A short tutorial," *Submitted for publication*, 2004.

[125] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 39, pp. 1-38, 1977.

[126] Z. Zivkovic, "Online (recursive) estimation, unsupervised learning, finite mixtures, model selection, EM-algorithm, *IEEE Transactions on Pattern Analysis and Machine Intelligence, ,* vol. 26, pp. 651-656, 2004.

[127] J. A. Bilmes, "A gentle tutorial of the EM algorithm and its application to parameter estimation for Gaussian mixture and hidden Markov models," *International Computer Science Institute*, vol. 4, 1998.

[128] T. K. Moon, "The expectation-maximization algorithm," *Signal Processing Magazine, IEEE*, vol. 13, pp. 47-60, 2002.

[129] J. A. Bilmes, "A gentle tutorial of the EM algorithm and its application to parameter estimation for Gaussian mixture and hidden Markov models," *International Computer Science Institute*, vol. 4, p. 126, 1998.

[130] X. Yang and J. Liu, "Mixture density estimation with group membership functions," *Pattern Recognition Letters*, vol. 23, pp. 501-512, 2002.

[131] V. S. Kodogiannis, J. N. Lygouras, A. Tarczynski, and H. S. Chowdrey, "Artificial Odor Discrimination System Using Electronic Nose and Neural Networks for the Identification of Urinary Tract Infection," *Information Technology in Biomedicine, IEEE Transactions on*, vol. 12, pp. 707-713, 2008.

[132] J. Abonyi and B. Feil, *Cluster analysis for data mining and system identification*: Birkhauser, 2007.

[133] R. H. Abiyev, "Fuzzy wavelet neural networks for identification and control of dynamic plants—a novel structure and a comparative study," *Industrial Electronics, IEEE Transactions on*, vol. 55, pp. 3133-3140, 2008.
[134] S. Yilmaz and Y. Oysal, "Fuzzy Wavelet Neural Network Models for Prediction and Identification of Dynamical Systems," *IEEE Transactions on Neural Networks*, vol. 21, pp. 1599-1609, 2010.
[135] W. Zhigang, P. Hong, and W. Jun, "Research for a Dynamic Recurrent Fuzzy Wavelet Network," in *Intelligent Systems Design and Applications, 2006. ISDA '06. Sixth International Conference on*, 2006, pp. 914-918.
[136] S. Srivastava, M. Singh, M. Hanmandlu, and A. N. Jha, "New fuzzy wavelet neural networks for system identification and control," *Applied Soft Computing*, vol. 6, pp. 1-17, 2005.
[137] E. Karatepe and M. AlcI, "A new approach to fuzzy wavelet system modeling," *International Journal of Approximate Reasoning*, vol. 40, pp. 302-322, 2005.
[138] M. Amina, E. Z. Panagou, V. S. Kodogiannis, and G. J. E. Nychas, "Wavelet neural networks for modelling high pressure inactivation kinetics of Listeria monocytogenes in UHT whole milk," *Chemometrics and Intelligent Laboratory Systems*, vol. 103, pp. 170-183.2010
[139] C.-J. Lin and C.-H. Chen, "Identification and prediction using recurrent compensatory neuro-fuzzy systems," *Fuzzy Sets and Systems*, vol. 150, pp. 307-330, 2005.
[140] J. Abonyi, R. Babuska, and F. Szeifert, "Modified Gath-Geva fuzzy clustering for identification of Takagi-Sugeno fuzzy models," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 32, pp. 612-621, 2002.
[141] V. Calhoun, T. Adal, M. Kraut, and G. Pearlson, "A weighted least-squares algorithm for estimation and visualization of relative latencies in event-related functional MRI," *Magnetic Resonance in Medicine*, vol. 44, pp. 947-954, 2000.
[142] R. Havangi, M. A. Nekoui, and M. Teshnehlab, "Adaptive Neuro-Fuzzy Extended Kalman Filtering for Robot Localization," *Arxiv preprint arXiv:1004.3267*.
[143] H. Yang, J. Li, and F. Ding, "A neural network learning algorithm of chemical process modeling based on the extended Kalman filter," *Neurocomputing*, vol. 70, pp. 625-632, 2007.
[144] G. Feng, G. Xiaohong, C. Xi-Ren, and A. Papalexopoulos, "Forecasting power market clearing price and quantity using a neural network method," in *Power Engineering Society Summer Meeting, 2000. IEEE*, 2000, vol. 4,pp. 2183-2188.
[145] S. J. Huang and K. R. Shih, "Short-term load forecasting via ARMA model identification including non-Gaussian process considerations," *IEEE Transactions on Power Systems, ,* vol. 18, pp. 673-679, 2003.
[146] N. Amjady, "Short-term hourly load forecasting using time-series modeling with peak load estimation capability," *Power Systems, IEEE Transactions on*, vol. 16, pp. 798-805, 2001.
[147] M. Gastaldi, R. Lamedica, A. Nardecchia, and A. Prudenzi, "Short-term forecasting of municipal load through a Kalman filtering based approach," 2004, vol. 3,pp. 1453-1458.
[148] J. Y. Fan and J. D. McDonald, "A real-time implementation of short-term load forecasting for distribution power systems," *IEEE Transactions on Power Systems*, vol. 9, pp. 988-994, 1994.
[149] H. T. Yang and C. M. Huang, "A new short-term load forecasting approach using self-organizing fuzzy ARMAX models," *IEEE Transactions on Power Systems ,* vol. 13, pp. 217-225, 1998.

[150] C. N. Lu, H. T. Wu, and S. Vemuri, "Neural network based short term load forecasting," *Power Systems, IEEE Transactions on,* vol. 8, pp. 336-342, 1993.
[151] C. Zhang and P. Ma, "Short-Term Electricity Price Forecasting Based on PSO Algorithm and RBF Neural Network Algorithm," 2010, pp. 334-337.
[152] A. Jain and B. Satish, "Clustering based Short Term Load Forecasting using Support Vector Machines," in *Proceedings of the IEEE PES Power Systems Conference Exposition (PSCE)*, 2009, pp. 1–7, pp. 1-8.
[153] P. Mastorocostas, J. Theocharis, and A. Bakirtzis, "Fuzzy modeling for short term load forecasting using the orthogonal least squares method," *IEEE Transactions on Power Systems, ,* vol. 14, pp. 29-36, 1999.
[154] M. Hanmandlu and B. K. Chauhan, "Load Forecasting Using Hybrid Models," *IEEE Transactions on Power Systems, IEEE Transactions on,* vol. 26, pp. 20-29, 2011.
[155] J. Yang and J. Stenzel, "Short-term load forecasting with increment regression tree," *Electric Power Systems Research,* vol. 76, pp. 880-888, 2006.
[156] W. Sun, "A Novel Hybrid GA Based SVM Short Term Load Forecasting Model," in Knowledge Acquisition and Modelling, 2009. KAM '09. Second International Symposium on, 2009, pp. 227-229.
[157] M. R. Khan and A. Abraham, "Short Term Load Forecasting Models in Czech Republic Using Soft Computing Paradigms," *Arxiv preprint cs/0405051,* 2004.
[158] W. Yuill, R. Kgokong, S. Chowdhury, and S. Chowdhury, "Management of short term load forecasting in South African power networks," in Power System Technology (POWERCON), 2010 International Conference on, 2010, pp. 1-8.
[159] V. S. Kodogiannis and E. M. Anagnostakis, "A study of advanced learning algorithms for short-term load forecasting," *Engineering Applications of Artificial Intelligence,* vol. 12, pp. 159-173, 1999.
[160] C. M. Huang and H. T. Yang, "Evolving wavelet-based networks for short-term load forecasting," *Generation, Transmission and Distribution, IEE Proceedings-,* vol. 148, pp. 222-228, 2001.
[161] L. A. Zadeh, "Fuzzy logic = computing with words," *IEEE Transactions on Fuzzy Systems,* vol. 4, pp. 103-111, 1996.
[162] J. Kacprzyk and S. Zadrozny, "Computing with words is an implementable paradigm: fuzzy queries, linguistic data summaries, and natural-language generation," *Fuzzy Systems, IEEE Transactions on,* vol. 18, pp. 461-472, 2010.
[163] W. Yu and X. Li, "Online fuzzy modeling with structure and parameter learning," *Expert Systems with Applications,* vol. 36, pp. 7484-7492, 2009.
[164] S. J. Yoo, Y. H. Choi, and J. B. Park, "Generalized predictive control based on self-recurrent wavelet neural network for stable path tracking of mobile robots: adaptive learning rates approach," *IEEE Transactions on Circuits and Systems I: Regular Papers,* vol. 53, pp. 1381-1394, 2006.
[165] X. Jiang and H. Adeli, "Dynamic wavelet neural network model for traffic flow forecasting," *Journal of transportation engineering,* vol. 131, p. 771, 2005.
[166] L. Mirea and R. J. Patton, "Recurrent wavelet neural networks applied to fault diagnosis," in *Control and Automation, 2008 16th Mediterranean Conference on,* 2008, pp. 1774-1779.
[167] R. Polikar, L. Upda, S. Upda, and V. Honavar, "Learn++: An incremental learning algorithm for supervised neural networks," *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews ,* vol. 31, pp. 497-508, 2001.

[168] L. Bruzzone and D. Fernàndez Prieto, "An incremental-learning neural network for the classification of remote-sensing images," *Pattern Recognition Letters*, vol. 20, pp. 1241-1248, 1999.

[169] T. Seipone and J. A. Bullinaria, "Evolving improved incremental learning schemes for neural network systems," in *Evolutionary Computation, 2005. The 2005 IEEE Congress on*, 2005, Vol. 3, pp. 2002-2009.

Appendix I

Statistical Error Criteria

Coefficient of Determination (*R*2): The coefficient of determination indicates how much of the total variation in the dependent variable can be accounted for by the model. It is computed as a value between 0 (0 percent) and 1 (100 percent). The higher the value, the better the fit.

$$\text{Coefficient of Determination}(R^2) = 1 - \frac{\sum(\text{Output}_{\text{estimated}} - \text{Output}_{\text{desired}})^2}{\sum(\text{Output}_{\text{desired}} - \bar{\text{O}}\text{utput}_{\text{desired}})^2}$$

Root Mean Square Error (RMSE): Expressing the formula in words, the difference between forecast and corresponding observed values are each squared and then averaged over the sample. Finally, the square root of the average is taken. Since the errors are squared before they are averaged, the RMSE gives a relatively high weight to large errors.

$$\text{Root Mean Square Error}(\text{RMSE}) = \sqrt{\frac{\sum_{i=1}^{N}(\text{Output}_{\text{estimated}} - \text{Output}_{\text{desired}})^2}{N}}$$

Mean Absolute Percentage Error (MAPE) (%): The MAPE measures the average magnitude of the errors in a set of forecasts, without considering their direction, in percentage.

$$\text{Mean Absolute Percentage Error}(\text{MAPE}) = \frac{100}{N} \times \sum_{i=1}^{N} \left| \text{Output}_{\text{estimated}} - \text{Output}_{\text{desired}} \right|$$

Standard Error of Prediction (SEP) (%): The standard error of prediction is the standard deviation of the prediction errors. It is computed like any other standard deviation - the square root of the error sum of squares divided by the number of samples

$$\text{Standard Error Of Prediction(SEP)} = 100 \times \sqrt{\left(\sqrt{\frac{\sum_{i=1}^{N} (\text{Output}_{\text{estimated}} - \text{Output}_{\text{desired}})^2}{N}} \right)}$$

Mean Relative Percentage Error (MRPE)(%) : Mean Relative Error is a number that compares how incorrect a quantity is from a number considered to be true. Unlike absolute error, where the error has the units of what is being measured, relative error is expressed as a percentage, defined as the absolute error divided by the true value.

$$\text{Mean Absolute Percentage Error(MRPE)} = \frac{100}{N} \times \sum_{i=1}^{N} \frac{|\text{Output}_{\text{estimated}} - \text{Output}_{\text{desired}}|}{\text{Output}_{\text{desired}}}$$

Bias factor (*B*f): The simplest relative measure is a ratio of the desired and estimated output. The ratio alone, however, may be misleading because, for example, a 'factor of 10' over-prediction (predicted/observed=10) will have more weight than a 'factor of 10' under-prediction of generation time (predicted/observed = 0.1). Thus, the logarithm of the ratio was chosen so that over- and under-prediction were given equal weight in determining the average deviation. The antilogarithm of this value (average relative deviation) may be interpreted as the average ratio of the predicted and observed values.

$$\text{Bias Factor} = 10^{(\sum \log(\frac{\text{Output}_{\text{desired}}}{\text{Output}_{\text{estimated}}})/N)}$$

Accuracy factor (*A*f): In order that under- and over-prediction not to 'cancel out' each other (because the logarithm of the ratios will have opposite signs) and consequently have some indication of the average accuracy of estimates, the average of the absolute values of the logarithm of the ratio was calculated in Accuracy Factor. The antilogarithm of this value will always be greater than or equal to one

$$\text{Accuracy Factor} = 10^{(\sum \left| \log \frac{\text{Output}_{\text{desired}}}{\text{Output}_{\text{estimated}}} \right| / N)}$$

Appendix II

List of Publications

Chapters in Book

1) V.S. Kodogiannis , **M. Amina** , J.N. Lygouras and G.-J.E. Nychas, “Application of wavelet neural networks as a non-linear modelling technique in food microbiology”, accepted in Animal Feed: Types, Nutrition, and Safety, Nova Science Publishers, Inc, 2011, (*in press*).

Journals

1) **M. Amina**, V.S. Kodogiannis, J.N. Lygouras, “Short-term Load Forecasting in Power System Using Clustering-Based Fuzzy Wavelet Neural Networks”, submitted to IEEE Trans on Power Systems, 2011.

2) **M. Amina**, V. S. Kodogiannis, J. N. Lygouras, and G. J. E. Nychas, "Identification of the Listeria monocytogenes survival curves in UHT whole milk utilising local linear wavelet neural networks," Elsevier, *Expert Systems with Applications*, vol. 39, pp. 1435-1450, 2012.

3) V.S.Kodogiannis, **M.Amina**, J.N.Lygouras “Power Load Forecasting using Extended Normalised Radial basis Function Networks”, accepted to *Journal of Computational Methods in Sciences and Engineering* (JCMSE), 2011, IOS Press.

4) **M.Amina,** E.Z. Panagou, V.S. Kodogiannis, G.-J.E.Nychas, “Wavelet Neural Networks for modelling high pressure inactivation kinetics of Lysteria Monocytogenes in UHT whole milk”, Elsevier *Chemometrics and intelligent laboratory systems*, Vol.103, No.2, pp.170-183, 2010.

Conference

1) **M. Amina** and V. S. Kodogiannis, "Load forecasting using fuzzy wavelet neural networks," in *Fuzzy Systems (FUZZIEEE), IEEE International Conference on*, Taipei, Taiwan, June 2011, pp. 1033-1040.

2) **M.Amina**,V.S.Kodogiannis, E.Z. Panagou and G.E.Nychas, "Modelling Listeria survival/death monocytogenes survival/death curves using Wavelet Neural Networks", The 2010 International Joint Conference on Neural Networks (IJCNN),Barcelona, Spain, 18-23 July 2010, pp. 1-8, ISBN: 978-1-4244-6916-1.

3) **M.Amina**, V.S.Kodogiannis, A.Tarczynski " Predictive Modelling in Food Mycology using Adaptive Neuro-Fuzzy Systems", 2009 IEEE/ACS International Conference on Computer Systems and Applications, AICCSA'2009, May 10-13, 2009. Rabat, Morocco, pp. 821-828, ISBN:978-1-4244-3806-8