University of Alberta

Neuro-fuzzy architecture based on complex fuzzy logic

by

Sara Aghakhani

A thesis submitted to the Faculty of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of

Master of Science
in
Software Engineering and Intelligent Systems

Department of Electrical and Computer Engineering

Spring 2010
Edmonton, Alberta

Examining Committee

Dr. Scott Dick, Department of Electrical and Computer Engineering

Dr. Petr Musilek, Department of Electrical and Computer Engineering

Dr. Paul Lu, Department of Computing Science

To my lovely husband Amir

For your enduring love, support and encouragement

To my parents Parichehr and Jalal

For your guidance and unconditional love

To my brothers Amir Hossein and Shahriar

For your support all the way since the beginning of my studies

Abstract

Complex fuzzy logic is a new type of multi-valued logic, in which truth values are drawn from the unit disc of the complex plane; it is thus a generalization of the familiar infinite-valued fuzzy logic. At the present time, all published research on complex fuzzy logic is theoretical in nature, with no practical applications demonstrated. The utility of complex fuzzy logic is thus still very debatable. In this thesis, the performance of ANCFIS is evaluated. ANCFIS is the first machine learning architecture to fully implement the ideas of complex fuzzy logic, and was designed to solve the important machine-learning problem of time-series forecasting. We then explore extensions to the ANCFIS architecture. The basic ANCFIS system uses batch (offline) learning, and was restricted to univariate time series prediction. We have developed both an online version of the univariate ANCFIS system, and a multivariate extension to the batch ANCFIS system.

Acknowledgement

I would like to express my gratitude to my Supervisor Dr. Scott Dick, for his support and guidance through this project. Without his knowledge and assistance this study would not have been possible.

My family has been a continued source of support. I would like to thank my parents for their encouragement and continued support that they have always given me. Also I am grateful to my brothers for their help and encouragement since the beginning of my studies.

Most importantly, I am heartily thankful to my husband, my friend Amir who has inspired me through this graduate study and for all he has done for me.

Table of contents

List of Tables

List of Figures

Chapter 1: Introduction

Complex fuzzy sets are a recently-proposed extension to the standard type-1 fuzzy set theory. Whereas a type-1 fuzzy set has membership function with a codomain of [0, 1], a complex fuzzy membership function has the unit disc of the complex plane as its codomain. Equivalently, a complex fuzzy set is a set of ordered pairs $(x, \mu(x))$ where $x \in \mathbf{X}$ is an object from some universal set $\mathbf{X}$, and $\mu(x) \in \mathbf{D}$ is the set of complex numbers whose modulus is less than or equal to one [1][2][3]. Significant progress has been made in clarifying the properties of complex fuzzy sets [1][4], but there have not been any applications of complex fuzzy logic to real-world problems. Until such applications demonstrate the utility of complex fuzzy logic, it will remain a theoretical curiosity.

The previous work by Dick [1] was proposed that complex fuzzy sets could be a practical model for approximately periodic phenomena; it means that it repeats itself but never become exactly the same. Complex fuzzy sets might be periodic as a result of the phase term [1]. We have been trying to build a complex fuzzy inferential system [3] in order to model regular phenomena. One of the important expressions of regularity can be found in the form of time series data and problems in time series forecasting. Thus, we have been trying to build a time series forecasting algorithm using the complex fuzzy logic. Standard fuzzy systems can be developed in two ways: either by elicitation from a domain expert or by learning from input-output data. Since complex fuzzy sets have 2-dimensional membership function, they could not simply be represented by 2 distinct fuzzy sets. Also experts need to present definition of all fuzzy sets and rules. As no one knows how to do this in complex fuzzy sets, the experts cannot be used here. So we used inductive learning and artificial neural network in order to build the complex fuzzy inferential system. The artificial neural network which is used here is called ANCFIS (Adaptive Neuro Complex Fuzzy Inference System) which is based on ANFIS architecture [5][6][7]. This network is

originally developed by Chen et al. [8]. However, in their work, Chen et al. [8] did not complete a full performance analysis of ANCFIS and the architecture was only limited to batch (offline) learning and univariate problems.

In this thesis, an extensive performance analysis of the ANCFIS architecture is presented, and the architecture is extended to incorporate online learning and multivariate time series forecasting. All of these algorithms are evaluated using the typical forecasting methodology: a one-step-ahead predication, in a single-split design with all training data chronologically earlier than the test data. The performance of univariate ANCFIS technique is tested against five time series datasets: Mackey-Glass [5][9][10], Santa Fe A (laser) [11] – [14], Sunspot [15]-[21], Star [21]-[23] and Waves [22] . Also, online ANCFIS is compared to two different time series: Sunspot and Waves. Multivariate ANCFIS applies four multivariate time series datasets: Transport and tourism-motel [22], Hydrology-river flow [22][24], Macro-economic [22] and Car-road-accident [25]-[27]. Main research work for ANCFIS is summarized as follows:

1. The experimental evaluation of Offline ANCFIS. Five different time series datasets are used and their results are compared to the related literature consisting of time series forecasting techniques used up to the year 2009. It is subsequently observed that ANCFIS could achieve very good performance and is comparable to the published forecasting results.
2. The development of an online ANCFIS architecture which is based on online-learning; using downhill-simplex algorithm (a derivative-free algorithm instead of VNCSA) and Recursive Least Square (RLS). The experimental evaluation of online ANCFIS includes two problems in time series predictions. We compared the results of online learning to those of offline learning and to the relevant results found in the literatures.
3. The development of multivariate ANCFIS architecture. Multivariate ANCFIS is used when there are multiple input vectors, where there is usually a correlation between the input vectors. We apply algebraic product operation which is one of the complex fuzzy conjunctions for

layer two. Also, layer five is changed to be compatible with multiple outputs. Experimental work on Multivariate ANCFIS includes four different multivariate time series datasets. We contrast our results with univariate ANCFIS for each variate separately and all variates together. This means that each variate is predicted separately with univariate ANCFIS. Then, the forecasting results of univariate ANCFIS for each variate are combined to find the forecasting results for all variates. Multivariate ANCFIS predicts the result of the whole dataset. Then the predicted and actual values for each variate are collected and each variate can be calculated separately.

The remainder of this thesis is organized into six chapters: Chapter 2 presents a literature review covering the topics of type-1 fuzzy logic, fuzzy relations and fuzzy reasoning, Type-1 fuzzy inferential systems, ANFIS, online and offline learning, complex fuzzy logic and complex-valued neural networks. Chapter 3 covers the original ANCFIS architecture which includes the structure of ANCFIS, the node functions of the forward pass, ANCFIS error back propagation and also VNCSA algorithm. In Chapter 4, we present the experimental results for univariate ANCFIS architecture on five time series forecasting datasets: MackyGlass, Santa Fe A (laser), Sunspot, Stellar and Waves. Chapter 5 develops online-learning architecture for ANCFIS and presents the procedure of downhill-simplex algorithm. The experimental work is done on two time series datasets: Sunspot and Waves. In Chapter 6, we present the Multivariate ANCFIS architecture with the experimental work for four different datasets: Transport and Tourism-motel, Hydrology-river, Macro-economic and Car-road-accident. Finally Chapter 7 includes summary and discussion of future work.

Chapter 2: Literature Review

Since Zadeh published his first paper on fuzzy sets [28], the research and applications on type-1 fuzzy logic have made an incredible improvement. In the past few years, type-1 fuzzy logic is applied in different ways and many disciplines. Its applications can be found in many areas from home appliances such as air conditioners, cameras, refrigerators, washing machines to medical instruments such as blood pressure monitors and even more. A type- 1 allows the gradual evaluation of the membership of elements in a set, and is based on two dimensional membership function (MF). It provides an organized calculus in order to solve vague and incomplete information linguistically. Here the linguistic information is converted to numerical values using linguistic labels specify by MF [5][29][30]. Also a type-1 fuzzy inference system uses fuzzy if-then rules to be able to model human expertise for a specific application. A type-2 fuzzy logic is a generalization of type-1 fuzzy logic in a way that uncertainty is not only limited to linguistic variables but also to the definition of membership functions [31]-[33]. It uses a function which is itself a type-1 fuzzy number and for this reason sometimes type-2 is referred as fuzzy-fuzzy. A type- 2 fuzzy set is illustrated by a three- dimensional membership function. The main idea in using type-2 fuzzy set is that most/all applications in general area of decision making modeling need to handle the imprecise data, knowledge and etc. A type-2 fuzzy set use this imprecision and make better computer systems [32]. As complex fuzzy set is an extension of type-1 fuzzy sets, we discuss only type-1 fuzzy logic. In this chapter a description on Type-1 fuzzy set will be presented first, and then set- theory operation and fuzzy relation with linguistic variables will be discussed.

2.1. Type-1 Fuzzy sets

A fuzzy set describes the degree to which an element belongs to a set. If we consider X as a collection of objects ($x \in X$), then a type-1 fuzzy set A in X is a set of ordered pairs:

$$A = \{(x, \mu_A(x)) \mid x \in X\} \quad \mu_A : X \to [0,1] \tag{2.1}$$

In equation (2.1) μ_A is called membership function and maps elements of X into their membership in the fuzzy set A [28]. X usually is considered as universe of discourse and it may contain continuous or discrete space.

The basic operations on fuzzy sets are union, intersection, containment and complement. The maximum and minimum is classical fuzzy operator for union and intersection on fuzzy sets. Standard fuzzy logic defines these operations as follows:

1. *Union (Disjunction)*: Union of two fuzzy sets A and B generate another fuzzy set C whose membership function is derived from membership function of A and B:

   $$\mu_c(x) = \mu_{A \cup B}(x) = \max(\mu_A(x), \mu_B(x)), x \in X \tag{2.2}$$

   The equivalent definition of "Union" is the smallest fuzzy set which contains both A and B .

2. *Intersection (Conjunction)*: Intersection of two fuzzy set A and fuzzy set B generate another fuzzy set C whose membership function is derived from membership function of A and B:

   $$\mu_c(x) = \mu_{A \cap B}(x) = \min(\mu_A(x), \mu_B(x)), x \in X \tag{2.3}$$

   C is the largest fuzzy set which is enclosed in both A and B

3. *Containment(Subset)*: Fuzzy set A is contained in fuzzy set B (A is a subset of B) or fuzzy set B contains fuzzy set A when:

   $$A \subseteq B \Rightarrow \mu_A(x) \leq \mu_B(x), x \in X \tag{2.4}$$

4. *Complement (negation):* Complement of fuzzy set A represents with $\overline{A}$. Membership of $\overline{A}(\mu_A(x))$ means the degree to which *x* does not belong to $\overline{A}$. The membership function is represented as:

$$\mu_{\overline{A}}(x) = 1 - \mu_A(x), x \in X \quad (2.5)$$

2.1.1 Membership functions

A fuzzy set is usually described by its membership function (MF). The common choices of parameterized membership functions are triangular MF which is specified with three parameters, trapezoidal MF with four parameters, Gaussian MF with two parameters and generalized bell MF with three parameters. Figure 2.1 represents examples of mentioned MFs.

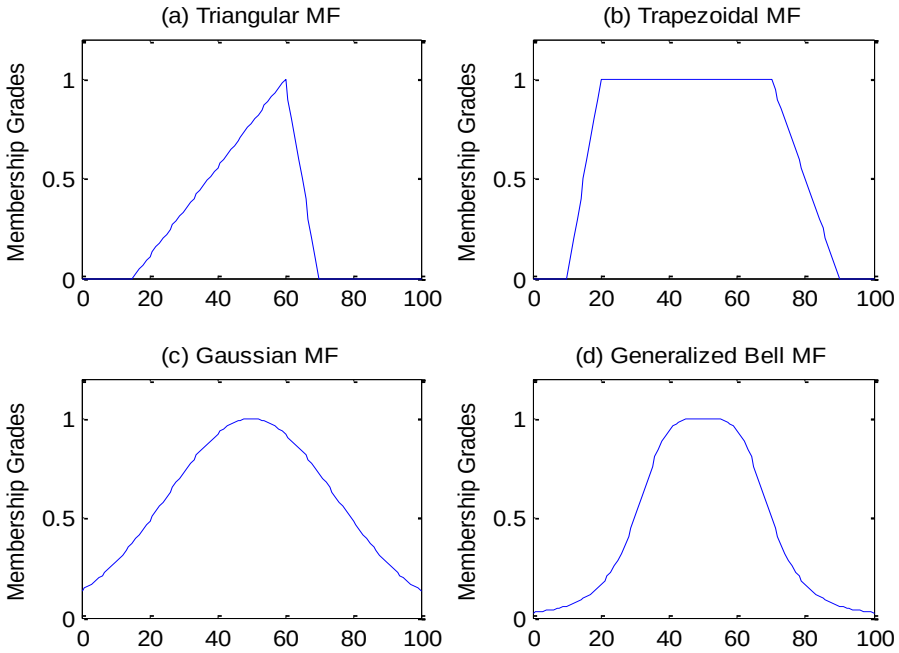


Figure 2.1: Example of four classes of parameterized membership functions: Triangular (x; 15,60,70)**; (b) trapezoid** (x; 10, 20, 70, 90)**;**
(a) (c) Gaussian (x; 50,25)**; (d) bell** (x; 20, 2, 50) **[5]**

2.1.2 Fuzzy relation

A fuzzy relation represents the degree of present or absence of association, interaction or interconnectedness between elements of two or more crisp sets. In explanation fuzzy relation, let's consider *U* and *V* as two crisp sets, then fuzzy relation *R (U, V)* is a fuzzy subset of a Cartesian product of $U \times V$ [34] which can be expressed as: $R : U \times V \to [0,1]$. The membership function of *R* is $\mu_R(x, y)$ which $x \in U$ and $y \in V$. Here for each pair of (x,y) there is a membership value between zero and one. If $R(x, y)$ =1, then it means that x and y are fully related and if $R(x, y)$ =0, it means that these two elements of x and y are not related at all. Obviously the values between zero and one for $R(x, y)$ reflect the partial association.

Relation can be described by an example from daily life using discrete fuzzy sets. First let us consider the relationship between color (x) and grade of maturity (y) of a fruit and then characterize the linguistic variable color by a crisp set X and grade of maturity by Y.

$X = \{green, yellow, red\}$

$Y = \{verdant, half\_mature, mature\}$

As it is presented above, X and Y both have three linguistic terms. Table 2.1 shows the crisp formulation of a relation $X \to Y$ between these two crisp sets. Here zeros and ones represent the grade of membership for this relation.

	verdant	**half-mature**	**mature**
green	1	0	0
yellow	0	1	0
red	0	0	1

Table 2.1: Relation $X \to Y$ between two crisp sets [35]

Degree of association can be presented by membership grades in a fuzzy relation and it is similar to the way as degrees of the set membership are expressed in a fuzzy set. After applying fuzzy relation, table 2.1 changes to table 2.2.

	verdant	half-mature	mature
green	1	0.5	0
yellow	0.3	1	0.4
red	0	0.2	1

Table 2.2: Fuzzy relation $X \rightarrow Y$ between two crisp sets [35]

2.1.3 Linguistic variable

As it was mentioned by Zadeh [36], conventional techniques for system analysis are essentially unsuitable for dealing with humanistic systems, whose behavior is strongly influenced by human judgment. This is what is called principle of incompatibility: ”As the complexity of a system increases, our ability to make a precise and significant statement about its behavior diminished until a threshold is reached which precision and significance become almost mutually exclusive characteristic”[36]. Because of this belief Zadeh proposed the concept of linguistic variables as alternative approach to model human thinking [34][37]. An example for linguistic variable can be “age”. For example if a man at age 60 is old, then we don’t know whether a 58 years old man is old or not. Hedges are not used too often; the most common hedges used are very, quite, more or less and etc. So we can say that a 58 years old man is quit old. The formal definition of linguistic variables is as below.

A linguistic variable is characterized by a 5-tuple (x, T(x), X, G, M) [38] where x is the name of the variable, X is the universe of discourse, T is the term set, which is the set of terminal symbols that can actually appear in a linguistic term. G is a syntactic rule which generates linguistic terms using the terminal symbols from T. Most commonly, G is null, so that the set of linguistic terms is exactly T. G is normally used to add linguistic hedges to atomic terms in T and M is a semantic rule which associate with each linguistic value B its meaning M(B), where M(B) indicates a fuzzy set in X.

Term set consists of primary term and/or hedges such as very, quite, etc [5]. As an example, *age* is interpreted as a linguistic variable with its term set to T(age) and it can be:

T(age) = { young, old, middle aged, very young,…}

Here each term in T (age) is specified by a fuzzy set of a universe of discourse X = [0,100]. We can say "age is old" to assign the linguistic value "old" to linguistic variable age. Here primary terms are (young, middle aged, old) and hedges are (very, more or less, quit).

The idea of linguistic variables is used in fuzzy reasoning for modeling and control problems. While variables in mathematics use numerical values, in fuzzy logic applications, the non-numeric linguistic variables are often used to facilitate the expression of rules and facts [39].

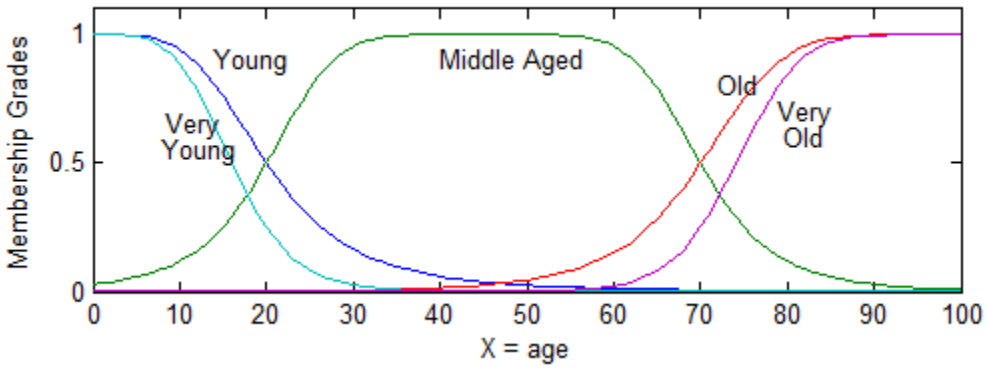


Figure 2.2: Typical membership functions of term set T(age) [5]

2.1.4 Fuzzy reasoning

Fuzzy reasoning is an inference process which uses fuzzy if-then rules and known facts to derive new facts. Fuzzy if-then rule or Fuzzy rule is expressed in the form of:

if x is A then y is B, (2.6)

where *A* and *B* both are linguistic values. Here the first part which is "*if x is A"* is antecedent or premise and "*y is B*" is consequence or conclusion. Modus ponens

is basic rule of inference in traditional two-valued logic. Here the truth of proposition B can be determined from the truth of A and implication $A \to B$. For example, if A is identified by "the tomato is red" and B with "tomato is ripe", then if it is true that "the tomato is red", it is true that "the tomato is ripe" [5]. For single rule with single antecedent, the inference procedure is described as:

premise1 (fact): x is A,

premise2 (rule): if x is A then y is B

Consequence (conclusion): y is B

However, in most of human reasoning, modus ponens is used in an approximate manner. As an example if we have the same application rule "if the tomato is red, then it is ripe", knowing that "tomato is more or less red", then it may infer that "the tomato is more or less ripe".

premise1 (fact): x is A',

premise2 (rule): if x is A' then y is B

Consequence (conclusion): y is B'

where A' is close to A and B' is close to B. Also A, B, A' and B' all are fuzzy sets of approximate universe. This inference procedure is called approximate reasoning or generalized modus ponens [40][41]. The MF of B' can be determined from equation 2.7 where $\vee$ and $\wedge$ are max and min:

$$\mu_{B'}(y) = \max_y \min[\mu_{A'}(x), \mu_R(x, y)] = \vee_x [\mu_{A'}(x) \wedge \mu_A(x)] \wedge \mu_B(y) \qquad (2.7)$$

If there is a single rule with multiple antecedents, the individual conditions are combined together by *and* connective. The inference procedure is defined as below:

premise1 (fact): x is A' and y is B',

premise2 (rule): if x is A and y is B, then z is C

Consequence (conclusion): y is C'

Then the membership function for C' is calculated as:

$$\mu_{C'}(z) = \vee_{x,y}[\mu_{A'}(x) \wedge \mu_{B'}(y)] \wedge [\mu_A(x) \wedge \mu_B(y) \wedge \mu_C(z)]$$
$$= \{\vee_x[\mu_{A'}(x) \wedge \mu_A(x)]\} \wedge \{\vee_y[\mu_{B'}(y) \wedge \mu_B(y)]\} \wedge \mu_C(z) \quad (2.8)$$
$$= (\omega_1 \wedge \omega_2) \wedge \mu_C(z)$$

Here $\omega_1 \wedge \omega_2$ is firing strength which reflects the degree to which antecedent part of the rule met, also $\vee$ and $\wedge$ are max and min.

2.1.5 Type-1 Fuzzy Inferential Systems

Fuzzy inference system is based on fuzzy reasoning, fuzzy if-then rules and fuzzy set theory. Its application can be found in many fields such as decision analysis, time series prediction, expert system, etc. As Fuzzy inference system is multi-disciplinary, it is known by many names such as fuzzy model [42] [43], fuzzy associative memory [44], etc.

Fuzzy inference systems are the most important modeling tool based on fuzzy set theory. The basic components for structure of fuzzy inference system are: first a rule base, which is made up of fuzzy rules; second a database, which stores membership functions used in fuzzy rules; third a reasoning mechanism, which implements generalized modus ponens; fourth a fuzzification interface which converts crisp data inputs into membership degrees for the fuzzy set antecedents; fifth a defuzzification inference that transforms a fuzzy consequent into a crisp output [5][7]. A fuzzy inference system is shown in figure 2.3.

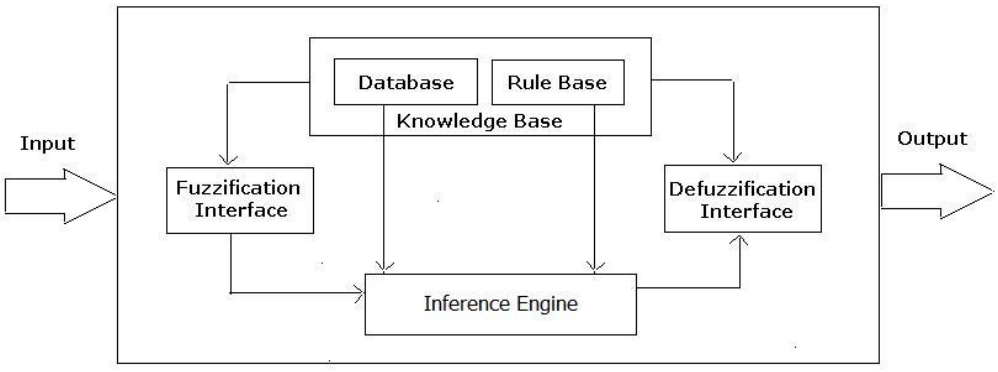


Figure 2.3: Fuzzy inference system [5][7]

There are three principal models for fuzzy inference systems: the Mamdani, TSK and Tsukamoto models [45] [46]. These three models are shown in Fig. 2.4. The main difference among them is in the consequences of their fuzzy rules, aggregation and defuzzification procedures.

The Mamdani fuzzy model [47] was the first fuzzy controller. The problem was to control the combination of steam engine and boiler by the aid of a set of linguistic control rules expressed by experienced human operators. As the plant takes only crisp values as inputs, a defuzzifier must be used to convert a fuzzy set to crisp value. There is more than one method for defuzzification available. The most common one is to adopt the centroid of the area under the output membership function. The computation of the centroid of an area z_{COA} is represented by:

$$z_{COA} = \frac{\int_z \mu_A(z) z dz}{\int_z \mu_A(z) dz} \tag{2.9}$$

where $\mu_A(z)$ is the aggregated output membership function.

Sugeno fuzzy model which is also known as TSK fuzzy model was proposed by Takagi, Sugeno and Kang [5][42][43]. The aim of this model is to develop a systematic approach to generate fuzzy rules from a given input-output data set. Here a typical fuzzy rule presents like this:

If x is A and y is B then z = f(x, y) (2.10)

In 2.10 *A* and *B* are fuzzy sets in antecedent and z is a crisp function in the consequence. Mainly *f(x,y)* is a polynomial function but it can be any function as far as it can completely represent the output of model with in the fuzzy region defined by the antecedents of the rule. If the function is first-order, then there is a first-order TSK fuzzy model and so on. The output level z_i of each rule is weighted by the firing strength w_i of the rule. The final output is weighted average of all outputs represented as:

$$z = \frac{\sum_{i=1}^{N} w_i z_i}{\sum_{i=1}^{N} z_i} \tag{2.11}$$

where z_i is a first order polynomial.

In the Tsukamoto fuzzy model [48], consequent of each fuzzy if-then rule is described by a fuzzy set with monotonic MF. Each rule's inferred output is defined as a crisp value made by the rule's firing strength. The overall output is the weighted average of each rule's output, presented as:

$$z = \frac{\sum_{i=1}^{N} w_i z_i}{\sum_{i=1}^{N} w_i} \tag{2.11}$$

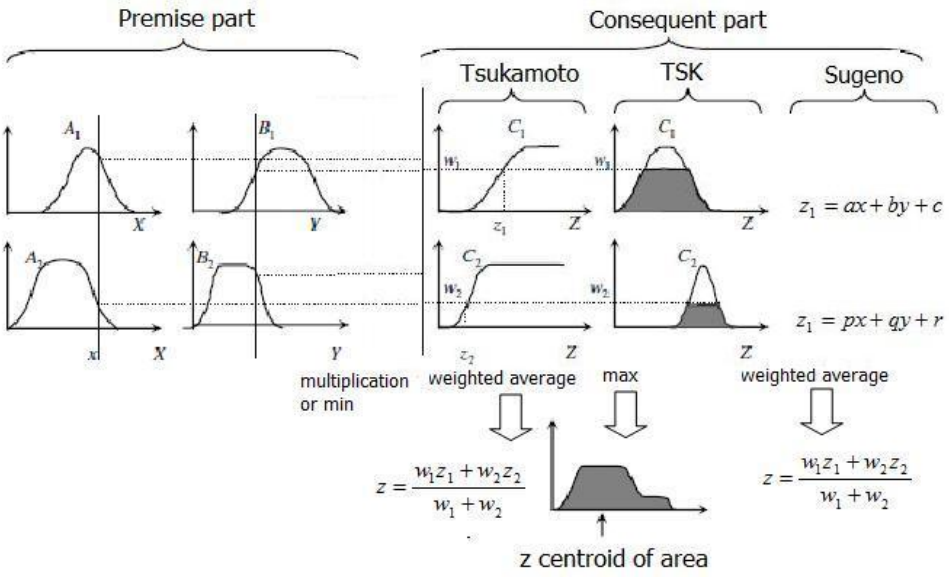


Figure 2.4: fuzzy if- then rules and Fuzzy inference mechanism [5][7]

2.2 ANFIS Review

ANFIS (Adaptive Network-based Fuzzy Inference System) is a class of adaptive networks that are functionally equal to fuzzy inference systems. Below adaptive

networks are explained as well as the architecture of ANFIS and how this system works.

2.2.1 Adaptive Network Review

An adaptive network is a network structure, consisting of a number of nodes connected through directed links. Each node corresponds to a process unit and the links between nodes represent communication links. All or some of the nodes are adaptive which means the outputs of these nodes depend on modifiable parameters. The learning rule defines the way that these parameters should be updated in order to minimize the error measure which can be the difference between actual output and desired output.

Adaptive networks are usually classified into two categories based on the type of the connection they have: *feedforward* and *recurrent* [5]. We call an adaptive network feedforward if the output of each node spreads from the input side to output side. If there is a feedback link which causes a circular path or loop in a network then we have a recurrent network. Figure 2.5 and 2.6 represent two types of adaptive network. Here square nodes are adaptive and circle nodes are fixed nodes.

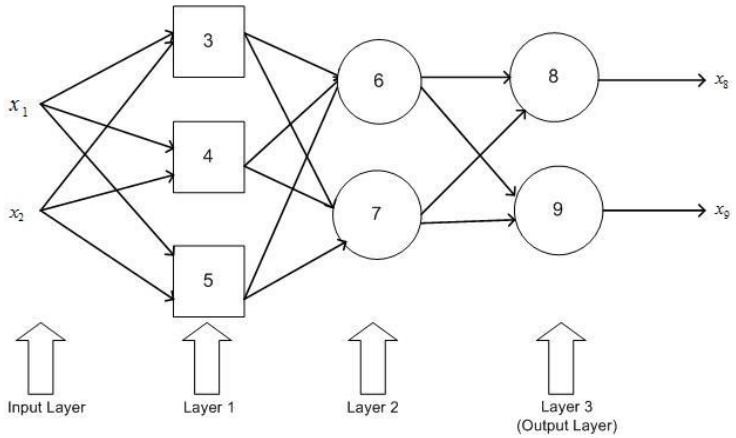


Figure 2.5: A feedforward (left-to-right) adaptive network in layered representation [5]

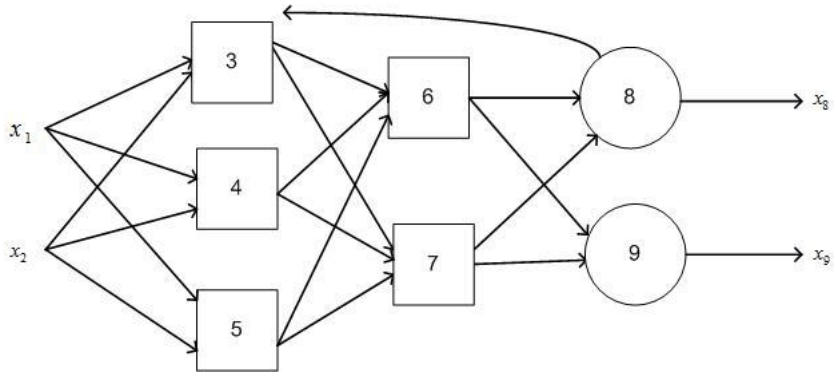


Figure 2.6: A recurrent adaptive network [5]

As it is shown in figure 2.6 node 8 has a feedback link to node 3 which changes the network from feedforward to recurrent.

2.2.2 On-line vs. Off-line learning

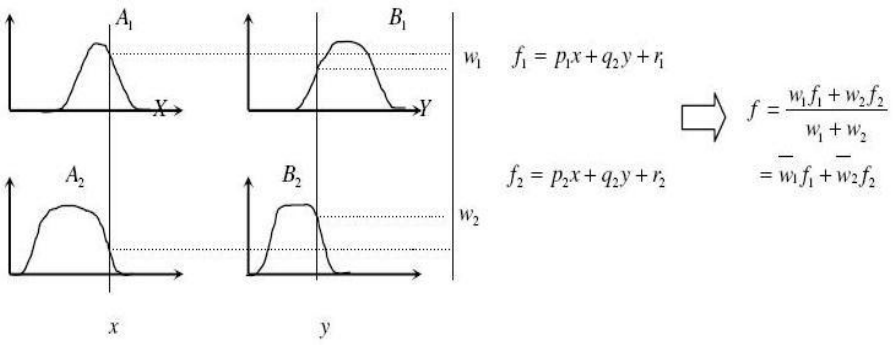
The main significance of a neural network is its ability to learn from its environment and to improve from its performance through learning. A neural network learns about its environment by an interactive process of adjustments applied to its synaptic weights and bias levels. Usually the network earns more knowledge about its environment after each iteration of the learning process. The learning process of a neural network starts by stimulation of the network by an environment. Then, it changes its free parameters as a result of this stimulation. Finally, it reacts in a new way to the environment because of the changes that occurred in its internal structure. There are two main learning strategy for adaptive networks; online learning and batch/offline learning.

In online learning the learning strategy is based on the online parameter identification for systems with changing characteristics. In online learning the training parameters/weights are updated after each presentation of training pattern (training vector) [49]. By changing the weights after each pattern, they could go backward and forward with each iteration although this may cause waste of considerable amount of time. This may happen because all the training data is not available at the same time. This method is used for online learning, in which

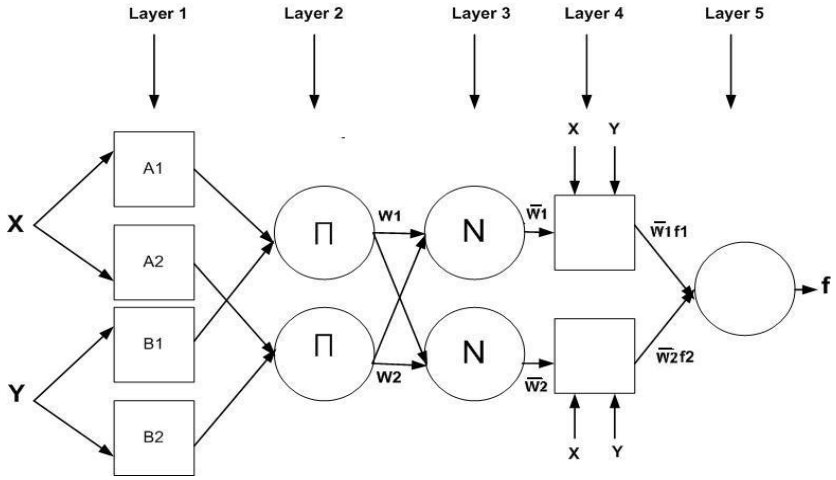
learning or updating is needed when data arrives in real time. We can minimize the random fluctuation of weights by batch learning in which weights adjustment is based on total error derivative over whole training set [5][49][50].
The batch/offline learning is different from the on-line training concerning the convergence speed and the quality of approximation. In batch learning, the weight update for each training vector is noted but the weights are not changed until all the input patterns have been presented [5][49]. Although using several/all training data pairs gives a better estimation for the predicted error than just using one, but batch updating requires extra memory storage for weight corrections before the weights are updated. This is critical especially when the network has large number of weights. Also, averaging the weight corrections may cause extra computational complexity for the algorithm and finally the smoothing effect of batch updating may cause the learning algorithm converge to local minima. Generally the performance of batch updating is case dependent [49][50].

2.2.3 ANFIS Definition

ANFIS (Adaptive Neuro-Fuzzy Inference System) is a feed-forward adaptive network with five layers and is equivalent to a TSK fuzzy inference system [6][7]. Each node indicates a processing unit and the directed links represent the flow direction of signals between connected nodes. To represent ANFIS architecture, consider a two input first-order Sugeno fuzzy model with two fuzzy if-then rules as it is shown in Fig. 2.7(a) then the equivalent ANFIS architecture can be shown in Fig. 2.7(b).
For a fist-order Sugeno fuzzy model, a common rule set with two fuzzy if-then rules is as below:
Rule 1: If x is A_1 and y is B_1, then $f_1 = p_1 x + q_1 y + r_1$,
Rule 2: If x is A_2 and y is B_2, then $f_2 = p_2 x + q_2 y + r_2$



(a)



(b)

Figure 2.7: (a) A two-input first-order Sugeno Fuzzy model with two rules (b) Equivalent ANFIS architecture [5]

In 2.7 (b) square nodes are adaptive and circle nodes are fixed nodes. Layer one and layer four have adaptive or parameterize nodes. The parameters in layer one are premise parameters and the parameters in layer four are consequent parameters. Nodes in the same layer have the same node functions which are different from other layers.

ANFIS uses offline or batch learning; each epoch consists of a forward pass and a backward pass. In the forward pass the parameters of layer one are fixed and the output nodes are calculated from first layer to fourth layer from left to right, and the parameters in fourth layer are identified by least-squares optimization. After

the parameters of polynomial in layer four are identified, and then the error is measured for training data pairs and summed together. Then in backward pass consequence parameters are fixed and error signals propagate from layer five to layer one from right to left and premise parameters in layer one are updated with gradient descent method. Below five layers of ANFIS are explained:

Layer 1:

Nodes in this layer are adaptive nodes. The node function of each node *i* in layer 1 is as following:

$$O_{1,i} = \mu_{A_i}(x), i = 1,2$$
$$O_{1,i} = \mu_{B_{i-2}}(y), i = 3,4 \qquad (2.13)$$

Here x or y is input to node i and A_i or B_{i-2} is linguistic label related to that node. Equivalently, $O_{1,i}$ is membership grade of fuzzy set A (where $A = A_1, A_2, B_1 or B_2$) which implies in what degree the inputs (x or y) satisfy of quantifier A. The membership function for A is any appropriate parameterized membership function such as bell function.

$$\mu_A(x) = \frac{1}{1 + \left| \frac{x - c_i}{a_i} \right|^{2b}} \qquad (2.14)$$

Here $\{a_i, b_i, c_i\}$ is the parameter set. Parameters of this layer are called premise parameters.

Layer 2:

Nodes in this layer which are labeled by Π are fixed nodes. The output of each node is product of all incoming signals and represents the firing strength of a rule. Generally speaking, any T-norm operation that presents fuzzy AND is suitable to be used as node function in this layer.

$$O_{2,i} = \omega_i = \mu_{A_i}(x)\mu_{B_i}(y), i = 1,2 \qquad (2.15)$$

Layer 3:

Every node in this layer is labeled by N and is a fixed node that normalizes each weight by sum of all rules firing strength.

$$O_{3,i} = \overline{\omega_i} = \frac{\omega_i}{\omega_1 + \omega_2}.i = 1,2 \tag{2.16}$$

Layer 4:

Each node in layer four is an adaptive node with this node function:

$$O_{2,i} = \overline{\omega_i} f_i = \overline{\omega_i}(p_i x + q_i y + r_i), \tag{2.17}$$

In this function, $\overline{\omega_i}$ is the output of layer three and p_i, q_i, r_i are consequent parameters for this node.

Layer 5:

This layer has only one fixed node which calculates the overall output and is the summation of all incoming signals.

$$\text{Overall output} = O_{5,1} = \sum_i \overline{\omega_i} f_i = \frac{\sum_i \omega_i f_i}{\sum_i \omega_i} \tag{2.18}$$

If the values of the premise parameters in ANFIS are fixed during forward pass, then the overall output can be expressed as a linear combination of consequent parameters. The output of figure 2.2 (b) can be expressed like below:

$$\begin{aligned} f &= \frac{w_1}{w_1 + w_2} f_1 + \frac{w_2}{w_1 + w_2} f_2 \\ &= \overline{w_1}(p_1 x + q_1 y + r_1) + \overline{w_2}(p_2 x + q_2 y + r_2) \\ &= (\overline{w_1} x) p_1 + (\overline{w_1} y) q_1 + (\overline{w_1}) r_1 + (\overline{w_2} x) p_2 + (\overline{w_2} y) q_2 + (\overline{w_2}) r_2 \end{aligned} \tag{2.19}$$

In order to avoid converging to local minima and speed up the identification of the parameters of adaptive network, *hybrid learning* is presented which integrates both gradient method and least square estimation. The main part of gradient descent in ANFIS is the calculation of gradient vector [5]. The gradient vector is derivative of an error measure regarding to a parameter. This is referred to back

propagation learning rule as the way it is calculated is from the output towards the inputs. The p^{th} error measure E_P can be gained from each training pair in our training data. The main goal is to minimize the total error measure.

$$E = \sum_P E_P \tag{2.20}$$

The error measured can be minimized by adjusting individual parameters of the network. In another way, a small change in parameter α (this parameter belongs to a node) may cause a high impact on the output of the node containing α. The target of calculating the gradient vector is to pass a form of derivative information from output layer, layer by layer towards the input layer.

Suppose an adaptive network with L layers and k^{th} layer has N (k) nodes, then output of node i in layer l can be expressed as $x_{l,i}$ and its function as $f_{l,i}$. Since an output node depends on its incoming signals and its parameter set, it can be written this way:

$$x_{l,i} = f_{l,i}(x_{l-1,l},...x_{l-1,N(l-1)}, a, b, c....) \tag{2.21}$$

Here *a, b, c,* etc. are parameters related to this node.

Suppose that the given training dataset has P entries. Here the error measure for the pth ($1 \le p \le P$) entry of training data entry can be defined as the sum of squared error as below equation.

$$E_P = \sum_{k=1}^{N(L)} (d_k - x_{L,k})^2 \tag{2.22}$$

In equation 2.22, d_k is the kth component of pth target output vector and $x_{L,k}$ is the kth component of actual output vector generated by the current pth input vector. Consequently, the overall error measure can be expressed as:

$E = \sum_{P=1}^{P} E_P$.

To perform a learning procedure that applies gradient descent of E in the whole parameter set, the computation of error rate $\frac{\partial E_p}{\partial x}$ is the primary need. Here error rate is for pth training data and for each node output x. The error rate for output node at *(L, i)* can be derived from equation 2.23.

$$\varepsilon_{L,i} = \frac{\partial^+ E_P}{\partial x_{L,i}} = c = -2(d_i - x_{L,i}) \tag{2.23}$$

Also the error rate for internal node *(l, i)*; $1 \le l \le L-1$ can be executed by the below chain rule:

$$\varepsilon_{l,i} = \frac{\partial^+ E_P}{\partial x_{l,i}} = \sum_{m=1}^{N(l+1)} \frac{\partial^+ E_P}{\partial x_{l+1,m}} \frac{\partial f_{l+1,m}}{\partial x_{l,i}} = \sum_{m=1}^{N(l+1)} \varepsilon_{l+1,m} \frac{\partial f_{l+1,m}}{\partial x_{l,i}} \tag{2.24}$$

This means that the error rate in internal node can be stated as a linear sum of the error rates of the nodes on the next layer.

The gradient vector is described as the derivative of the error measure with respect to each parameter. The chain rule should be applied again to find gradient vector. Consider α as a parameter of *i*th node in layer *l*, and then we have:

$$\frac{\partial^+ E_P}{\partial \alpha} = \frac{\partial^+ E_P}{\partial x_{l,i}} \frac{\partial f_{l,i}}{\partial \alpha} = \varepsilon_{l,i} \frac{\partial f_{l,i}}{\partial \alpha}, \tag{2.25}$$

The derivative of overall error measure E regarding to α is:

$$\frac{\partial^+ E}{\partial \alpha} = \sum_{p=1}^{P} \frac{\partial^+ E_P}{\partial \alpha}, \tag{2.26}$$

Consequently the generic parameter α can be updated as:

$$\Delta\alpha = -\eta \frac{\partial E}{\partial \alpha} \tag{2.27}$$

In above equation η is *learning rate* and can be written as:

$$\eta = \frac{k}{\sqrt{\sum_{\alpha} \left(\frac{\partial E}{\partial \alpha} \right)^2}} \tag{2.28}$$

In 2.28 k is *step size* which is the length of each gradient transition in parameter space. *Step size* can affect the convergence rate. ANFIS use batch learning mode to update α based on formula 2.26.

Figure 2.8 illustrate an ANFIS example of two inputs, first order TSK fuzzy model with two fuzzy if-then rules and bell function membership function; the same as Figure 2.2 (b).

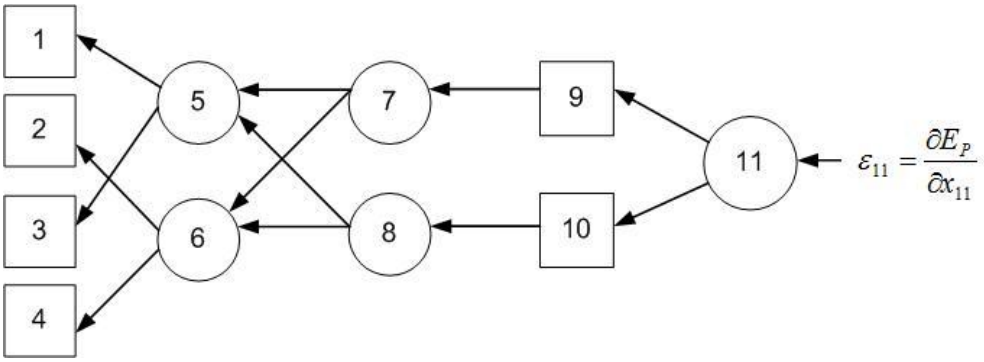


Figure 2.8: Error propagation for Fig. 2.2 (b)

$$\varepsilon_{11} = \frac{\partial E_P}{\partial x_{11}} = -2(d_{11} - x_{11}) = -2\left(d_{11} - \sum_i \overline{w_i} f_i \right), i = 1,2 \ ,$$

$$\varepsilon_{10} = \frac{\partial E_P}{\partial x_{10}} = \frac{\partial E_p}{\partial x_{11}} \frac{\partial f_{11}}{\partial x_{10}} = \varepsilon_{11} \frac{\partial f_{11}}{\partial x_{10}} = \varepsilon_{11} \frac{\partial\left(\sum_i \overline{w_i} f_i \right)}{\partial\left(\overline{w_2}\right) f_2} = \varepsilon_{11}, i = 1,2 \ ,$$

$$\varepsilon_9 = \frac{\partial E_P}{\partial x_9} = \frac{\partial E_p}{\partial x_{11}} \frac{\partial f_{11}}{\partial x_9} = \varepsilon_{11} \frac{\partial f_{11}}{\partial x_9} = \varepsilon_{11} \frac{\partial\left(\sum_i \overline{w_i} f_i \right)}{\partial\left(\overline{w_1}\right) f_1} = \varepsilon_{11}, i = 1,2 \ ,$$

$$\varepsilon_8 = \frac{\partial E_P}{\partial x_8} = \frac{\partial E_p}{\partial x_{10}} \frac{\partial f_{10}}{\partial x_8} = \varepsilon_{10} \frac{\partial f_{10}}{\partial x_8} = \varepsilon_{10} \frac{\partial\left(\overline{w_2} f_2\right)}{\partial\left(\overline{w_2}\right)} = \varepsilon_{10} f_2 = \varepsilon_{11} f_2 \ ,$$

$$\varepsilon_7 = \frac{\partial E_P}{\partial x_7} = \frac{\partial E_p}{\partial x_9} \frac{\partial f_9}{\partial x_7} = \varepsilon_9 \frac{\partial f_9}{\partial x_7} = \varepsilon_9 \frac{\partial\left(\overline{w_1} f_1\right)}{\partial\left(\overline{w_1}\right)} = \varepsilon_9 f_1 = \varepsilon_{11} f_1 \ ,$$

$$\varepsilon_6 = \frac{\partial E_P}{\partial x_6} = \frac{\partial E_p}{\partial x_8} \frac{\partial f_8}{\partial x_6} + \frac{\partial E_p}{\partial x_7} \frac{\partial f_7}{\partial x_6} = \varepsilon_8 \frac{\partial f_8}{\partial x_6} + \varepsilon_7 \frac{\partial f_7}{\partial x_6} = \varepsilon_8 \frac{\partial\left(\frac{w_2}{w_1 + w_2} \right)}{\partial(w_2)} + \varepsilon_7 \frac{\partial\left(\frac{w_1}{w_1 + w_2} \right)}{\partial(w_2)}$$

$$= \varepsilon_8 \frac{w_1}{(w_1 + w_2)^2} + \varepsilon_7 \frac{-w_1}{(w_1 + w_2)^2} = \varepsilon_{11} \frac{w_1 f_2 - w_1 f_1}{(w_1 + w_2)^2},$$

$$\varepsilon_5 = \frac{\partial E_P}{\partial x_5} = \frac{\partial E_p}{\partial x_8}\frac{\partial f_8}{\partial x_5} + \frac{\partial E_p}{\partial x_7}\frac{\partial f_7}{\partial x_5} = \varepsilon_8 \frac{\partial f_8}{\partial x_5} + \varepsilon_7 \frac{\partial f_7}{\partial x_5} = \varepsilon_8 \frac{\partial\left(\frac{w_2}{w_1+w_2}\right)}{\partial(w_1)} + \varepsilon_7 \frac{\partial\left(\frac{w_1}{w_1+w_2}\right)}{\partial(w_1)}$$

$$= \varepsilon_8 \frac{-w_2}{(w_1+w_2)^2} + \varepsilon_7 \frac{w_2}{(w_1+w_2)^2} = \varepsilon_{11} \frac{w_2 f_1 - w_2 f_2}{(w_1+w_2)^2},$$

$$\varepsilon_4 = \frac{\partial E_P}{\partial x_4} = \frac{\partial E_p}{\partial x_6}\frac{\partial f_6}{\partial x_4} = \varepsilon_6 \frac{\partial f_6}{\partial x_4} = \varepsilon_6 \frac{\partial(\mu_{A_1}\mu_{B_2})}{\partial\left(\mu_{B_2}\right)} = \varepsilon_6 \mu_{A_2},$$

$$\varepsilon_3 = \frac{\partial E_P}{\partial x_3} = \frac{\partial E_p}{\partial x_5}\frac{\partial f_5}{\partial x_3} = \varepsilon_5 \frac{\partial f_5}{\partial x_3} = \varepsilon_5 \frac{\partial(\mu_{A_1}\mu_{B_1})}{\partial\left(\mu_{B_1}\right)} = \varepsilon_5 \mu_{A_1},$$

$$\varepsilon_2 = \frac{\partial E_P}{\partial x_2} = \frac{\partial E_p}{\partial x_6}\frac{\partial f_6}{\partial x_2} = \varepsilon_6 \frac{\partial f_5}{\partial x_2} = \varepsilon_6 \frac{\partial(\mu_{A_2}\mu_{B_2})}{\partial\left(\mu_{A_2}\right)} = \varepsilon_5 \mu_{B_2},$$

$$\varepsilon_1 = \frac{\partial E_P}{\partial x_1} = \frac{\partial E_p}{\partial x_5}\frac{\partial f_5}{\partial x_1} = \varepsilon_5 \frac{\partial f_5}{\partial x_1} = \varepsilon_5 \frac{\partial(\mu_{A_1}\mu_{B_1})}{\partial\left(\mu_{A_1}\right)} = \varepsilon_5 \mu_{B_1},$$

The derivative of bell membership; $\mu_A(x) = \frac{1}{1+\left[\left(\frac{x-c_i}{a_i}\right)^2\right]^{b_i}}$; function regarding to parameters are as below:

$$\frac{\partial \mu_A}{\partial a_i} = \frac{2b_i}{a_i}\mu_A(1-\mu_A),$$

$$\frac{\partial \mu_A}{\partial b_i} = \begin{cases} 2\ln\left|\frac{x-c_i}{a_i}\right|\mu_A(1-\mu_A), if x \neq c_i \\ 0, if x = c_i \end{cases}$$

$$\frac{\partial \mu_A}{\partial c_i} = \begin{cases} \frac{2b_i}{x-c_i}\mu_A(1-\mu_A), if x \neq c_i \\ 0, if x = c_i \end{cases} \qquad (2.29)$$

Then the gradients for parameters a_i, b_i, c_i in related nodes $i = 1,2,3,4$ of layer one can be derived according to equation 2.29. Therefore for node one, we have:

$$\frac{\partial E_P}{\partial a_1} = \frac{\partial E_P}{\partial x_1}\frac{\partial f_1}{\partial a_1} = \varepsilon_1 \frac{\partial f_1}{\partial a_1} = \varepsilon_1 \frac{\partial(\mu_{A_1})}{\partial a_1} = \varepsilon_1 \frac{2b_1}{a_1}\mu_{A_1}(1-\mu_{A_1}),$$

$$\frac{\partial E_P}{\partial b_1} = \frac{\partial E_P}{\partial x_1}\frac{\partial f_1}{\partial b_1} = \varepsilon_1 \frac{\partial f_1}{\partial b_1} = \varepsilon_1 \frac{\partial(\mu_{A_1})}{\partial b_1} = \begin{cases} 2\varepsilon_1 \ln\left|\frac{x-c_1}{a_1}\right| \mu_{A_1}(1-\mu_{A_1}), if x \neq c_1 \\ 0, if x = c_1 \end{cases}$$

$$\frac{\partial E_P}{\partial c_1} = \frac{\partial E_P}{\partial x_1}\frac{\partial f_1}{\partial c_1} = \varepsilon_1 \frac{\partial f_1}{\partial c_1} = \varepsilon_1 \frac{\partial(\mu_{A_1})}{\partial c_1} = \begin{cases} \frac{2b_1}{x-c_1}\varepsilon_1 \mu_{A_1}(1-\mu_{A_1}), if x \neq c_1 \\ 0, if x = c_1 \end{cases}$$

And also for node two we have:

$$\frac{\partial E_P}{\partial a_2} = \frac{\partial E_P}{\partial x_2}\frac{\partial f_2}{\partial a_2} = \varepsilon_2 \frac{\partial f_2}{\partial a_2} = \varepsilon_2 \frac{\partial(\mu_{A_2})}{\partial a_2} = \varepsilon_2 \frac{2b_2}{a_2}\mu_{A_2}(1-\mu_{A_2})$$

$$\frac{\partial E_P}{\partial b_2} = \frac{\partial E_P}{\partial x_2}\frac{\partial f_2}{\partial b_2} = \varepsilon_2 \frac{\partial f_2}{\partial b_2} = \varepsilon_2 \frac{\partial(\mu_{A_2})}{\partial b_2} = \begin{cases} 2\varepsilon_2 \ln\left|\frac{x-c_2}{a_2}\right| \mu_{A_2}(1-\mu_{A_2}), if x \neq c_2 \\ 0, if x = c_2 \end{cases}$$

$$\frac{\partial E_P}{\partial c_2} = \frac{\partial E_P}{\partial x_2}\frac{\partial f_2}{\partial c_2} = \varepsilon_2 \frac{\partial f_2}{\partial c_2} = \varepsilon_2 \frac{\partial(\mu_{A_2})}{\partial c_2} = \begin{cases} \frac{2b_2}{x-c_2}\varepsilon_2 \mu_{A_2}(1-\mu_{A_2}), if x \neq c_2 \\ 0, if x = c_2 \end{cases}$$

And the same equations for node 3:

$$\frac{\partial E_P}{\partial a_3} = \frac{\partial E_P}{\partial x_3}\frac{\partial f_3}{\partial a_3} = \varepsilon_3 \frac{\partial f_3}{\partial a_3} = \varepsilon_3 \frac{\partial(\mu_{A_3})}{\partial a_3} = \varepsilon_3 \frac{2b_3}{a_3}\mu_{A_3}(1-\mu_{A_3})$$

$$\frac{\partial E_P}{\partial b_3} = \frac{\partial E_P}{\partial x_3}\frac{\partial f_3}{\partial b_3} = \varepsilon_3 \frac{\partial f_3}{\partial b_3} = \varepsilon_3 \frac{\partial(\mu_{A_3})}{\partial b_3} = \begin{cases} 2\varepsilon_3 \ln\left|\frac{x-c_3}{a_3}\right| \mu_{A_3}(1-\mu_{A_3}), if x \neq c_3 \\ 0, if x = c_3 \end{cases}$$

$$\frac{\partial E_P}{\partial c_3} = \frac{\partial E_P}{\partial x_3}\frac{\partial f_3}{\partial c_3} = \varepsilon_3 \frac{\partial f_3}{\partial c_3} = \varepsilon_3 \frac{\partial(\mu_{A_3})}{\partial c_3} = \begin{cases} \frac{2b_3}{x-c_3}\varepsilon_3 \mu_{A_3}(1-\mu_{A_3}), if x \neq c_3 \\ 0, if x = c_3 \end{cases}$$

And for node 4:

$$\frac{\partial E_P}{\partial a_4} = \frac{\partial E_P}{\partial x_4}\frac{\partial f_4}{\partial a_4} = \varepsilon_4 \frac{\partial f_4}{\partial a_4} = \varepsilon_4 \frac{\partial(\mu_{A_4})}{\partial a_4} = \varepsilon_4 \frac{2b_4}{a_4}\mu_{A_4}(1-\mu_{A_4})$$

$$\frac{\partial E_P}{\partial b_4} = \frac{\partial E_P}{\partial x_4}\frac{\partial f_4}{\partial b_4} = \varepsilon_4 \frac{\partial f_4}{\partial b_4} = \varepsilon_4 \frac{\partial(\mu_{A_4})}{\partial b_4} = \begin{cases} 2\varepsilon_4 \ln\left|\frac{x-c_4}{a_4}\right| \mu_{A_4}(1-\mu_{A_4}), if x \neq c_4 \\ 0, if x = c_4 \end{cases}$$

$$\frac{\partial E_P}{\partial c_4} = \frac{\partial E_P}{\partial x_4}\frac{\partial f_4}{\partial c_4} = \varepsilon_4 \frac{\partial f_4}{\partial c_4} = \varepsilon_4 \frac{\partial(\mu_{A_4})}{\partial c_4} = \begin{cases} \frac{2b_4}{x - c_4}\varepsilon_4 \mu_{A_4}(1-\mu_{A_4}), if x \neq c_4 \\ 0, if x = c_4 \end{cases}$$

Therefore adaptive parameters will be updated with respect to equations 2.26, 2.27 and 2.28.

2.3 Complex fuzzy theory

Complex fuzzy sets are an extension to type-1 fuzzy sets. It is a combination of traditional fuzzy set and complex numbers and is characterized by complex-value membership function [2] [51]. The result of this idea is the development of complex-valued membership fuzzy sets [2] and complex fuzzy logic [1][4]. Complex fuzzy sets appear to be good model for "approximately periodic" temporal phenomena [1]. We call it "approximately periodic" because usually it never repeats itself exactly. An example of approximately periodic phenomena is traffic congestion. The traffic is heavy in the morning from one way to work and in the afternoon the opposite way back from work and during night the roads are quit empty. This behavior repeats every day but it never repeats itself exactly the same each day [4].

Complex fuzzy sets are different from the complex fuzzy numbers which Buckley developed in [52] [53]. The fuzzy set representing the fuzzy complex number is an ordinary fuzzy set with membership grades in range of [0,1]. In another way, a fuzzy complex number is a type-1 fuzzy set whose members are elements of complex plane. Fuzzy complex numbers were used in the solution of fuzzy relational equation in [54] [55]. In [56] a complex fuzzy set was defined as a membership function which maps the complex plane into $[0,1]\times[0,1]$; this is very similar to complex fuzzy set defined in [2] and close to the formulation in [57].

2.3.1 Complex Fuzzy Sets

Complex fuzzy set S which is defined on the universe of discourse U is characterized by membership function $\mu_S(x)$ where $x \in U$. The values of membership function are in the form of:

$$\mu_S = r_S(x).e^{jw_s(x)} \ , \ j = \sqrt{-1} \tag{2.30}$$

Here terms of $r_S(x)$ and $w_S(x)$ as amplitude and phase; are real values and $r_S(x) \in [0,1]$. The complex fuzzy set can be presented as a set of ordered pairs as below [2].

$$S = \{(x, \mu_S) \mid x \in U\} \tag{2.31}$$

In complex fuzzy sets, membership values are drawn from unit disc of complex plane. The membership function of a complex fuzzy set is a vector in complex plane and its magnitude is less or equal to one. A complex fuzzy set is shown in Fig. 2.9 where the membership function is visualized by placing the complex plane $R \times I$ at right angles to the universe of discourse U. The complex membership function then forms a trajectory within the cylinder formed by projecting the unit disc D along U.

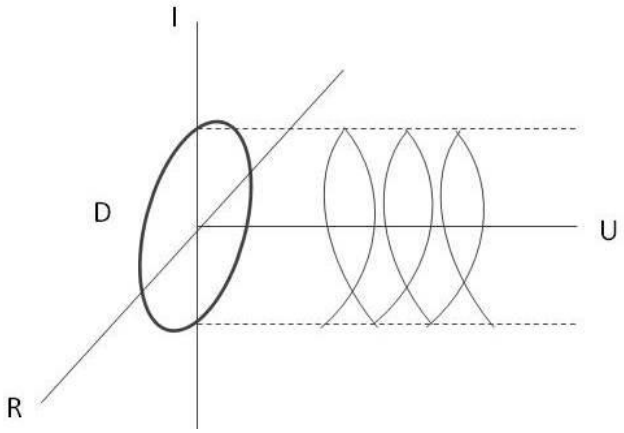


Figure 2.9: Complex fuzzy set [1]

As the complex fuzzy sets are extension of the ordinary fuzzy sets, it is possible to convert the complex fuzzy set in to traditional fuzzy by setting $r_S(x) = \mu_s(x)$ and also $w_S(x) = 0$ which means that the phase term is not considered at this point. The amplitude term is the same as real-valued grade of membership which lies in range of [0, 1] and can be considered as the degree to which *x* is a member of complex fuzzy set *S*. Though the phase term is an innovative parameter of membership which with this, the complex fuzzy sets are recognized from the ordinary fuzzy sets. Also it is the phase which makes complex fuzzy logic different from the conventional fuzzy logic. The phase term allows a new type of interaction happen between membership functions. Complex fuzzy membership function can be in form of wave where phase term let them interfere with each other [2].

2.3.2 Complex fuzzy logic

Complex fuzzy logic [1][4] employ rules built with complex fuzzy sets to develop a complex fuzzy logic system. These rules are represented in the form of If-Then statements similar to traditional fuzzy logic.

The complex fuzzy implication relation is characterized by a complex-valued membership function and is represented as $\mu_{A\to B}(x, y)$. The amplitude term ($r_{A\to B}(x, y)$) is real-valued grade of membership and shows the degree of truth of the implication relation. The phase term ($\omega_{A\to B}(x, y)$) represents the phase associated with implication. Phase term is of little consequence by itself though it becomes a more important parameter when several implication relations are combined at the same time, similar to what occurs in complex fuzzy systems. The implication function employed in complex fuzzy logic is the product implication:

$$\mu_{A\to B}(x, y) = \mu_A(x).\mu_B(y) \tag{2.16}$$

Also the amplitude and phase are calculated this way:

$$r_{A\to B}(x, y) = r_A(x).r_B(y) \tag{2.17}$$

$$\omega_{A\to B}(x,y) = \omega_A(x) + \omega_B(y) \tag{2.18}$$

Complex fuzzy implication can be used in order to build complex fuzzy inference rules in the *generalized modus ponens* form.

Premise 1: "X is A^*"

Premise 2: "IF X is A, THEN Y is B"

Consequence: "Y is B^*"

Where all the sets A, B, A^* and B^* are complex fuzzy sets.

The amplitude and phase term of membership function B^* can be given by:

$$r_{B^*} = \sup[r_{A^*}(x) \otimes r_{A\to B}(x,y)] = \sup[r_{A^*}(x) \otimes (r_A(x).r_B(y))] \tag{2.18}$$

$$\omega_{B^*} = f[g(\omega_{A^*}(x), \omega_{A\to B}(x,y))] = f[g(\omega_A(x), ((\omega_A(x) + \omega_B(y)))] \tag{2.19}$$

Where g refers to any function used to compute the intersection of two membership phases and f is the membership phase equal to *sup* operation [4]. Both of them are application dependent.

2.3.3 Complex Valued Neural Networks

ANCFIS architecture is complex-value neural networks (CVNN) architecture. CVNN accept complex-valued inputs and outputs and it is possible that their neuron weights and biases be complex-valued as well [58][59]. Early models of CVNN have generalized the Hopfield model, back propagation and perception learning rule in order to handle the complex inputs. Noest [60] introduced an associative memory network with local variables assuming one of q equidistant positions on the unit circle (q-state phasors) in the complex plane. Leung and Haykin [61] extended real-valued back propagation networks to complex-valued back propagation networks in order to solve problems related to radar signal processing and communication in which complex-valued representation of signal is required. Kim and Guest [62] also extended back propagation to the complex domain, to process frequency-domain data. More recent work was done by Hirose

et al. They have used complex- valued neurons in the coherent neural network architecture. All input signal, output signal and weight are complex numbers. The neural connection weight ω_{nm} was explained by $|\omega_{nm}| \exp[i2\pi f \tau_{nm}], i = \sqrt{-1}$. Here $|\omega_{nm}|$ is connection amplitude τ_{nm} is delay time and *f* is the carrier frequency which modifies the phase of selected neuron weights. Two applications for development-learning architecture are proposed [63] [64]. Dynamics of complex-value NNs with real-imaginary-type activation was evaluated when it was used in complex-plane transform [65][66]. Also the characteristics of activation functions discussed in [67][68]. Associative memories (mainly Hopfield networks) are an important area for complex-value NNs research. One of recent work on this area is related to an exploration of the properties of different neuron activation functions [69] and an application to traffic signal coordination [70]. In [71], quantum associative memory uses complex-valued neuron weights with a distribution function that is a solution of Schrodinger's diffusion equation. Nitta proposed a quaternion-valued neural network [72]. Generally speaking a good overview of complex-value NNs can be found in [63].

2.3.4 Implementation of Complex Fuzzy Logic in ANFIS

Previously there was just one attempt to develop an inductive learning architecture using complex fuzzy sets which was named CANFIS [73]. It was considered for complex-value input-output pairs with modeling a simple lead-lag compensator transfer function in form of $k\frac{(1+TS)}{(1+T'S)}$ where $S = j\omega$. The architecture of CANFIS is a hybrid of complex fuzzy sets and a complex-valued single layer neural network. The real-valued adaptive parameters of the complex MFs for each rule were updated by a steepest descent algorithm and weights and bias parameters were updated by complex least square estimator. Here the idea is derived from [2] where phase and magnitude are considered separately. As the phase and amplitude treat separately, the nature of complex fuzzy sets is ignored.

Also, the key property of rule interference [6] was not implemented. An earlier architecture developed by Li and Jang [74] also named CANFIS was an extension of ANFIS architecture with complex-valued inputs and outputs. Here each input has its real and imaginary component and 2 Gaussian type-1 fuzzy sets are assigned to each component of each input. The number of rules for example for a three complex input system is 64. Similar to ANFIS, the premise parameters were updated with gradient descent and the consequent parameters were updated via the least square algorithm. In ANCFIS architecture phase and magnitude are coupled with each other [1]. This approach leads to a parsimonious network structure, different from other proposed architectures [73] [74].

Chapter 3: An Introduction to ANCFIS

The original ANCFIS architecture was developed by Chen et al. [8]. ANCFIS is a six-layer adaptive neural network with complex-valued signals through much of the network [8]. Its adaptive nodes are in layer one and five. Figure 3.1 represents one example of ANCFIS architecture with one input and three rules.

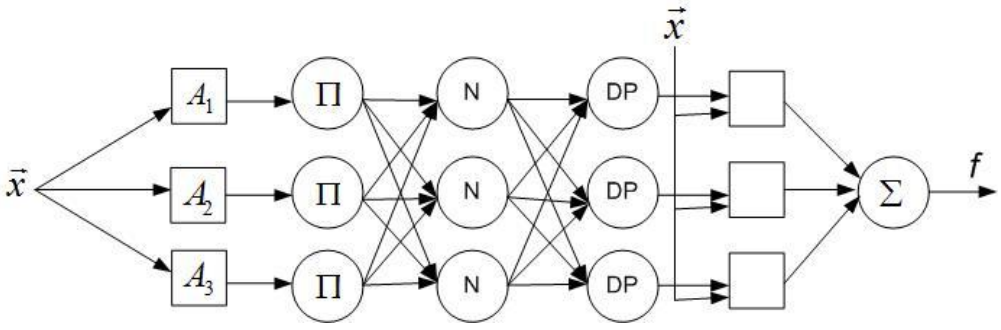


Figure 3.1: An ANCFIS architecture [8]

ANCFIS is based on ANFIS [7] architecture with some modifications. First ANFIS needs several inputs to capture a segment of univariate time series which leads to a combinatorial exploration in number of inferential rules, but ANCFIS get only a single input which is a sliding window of observed values in the time series. Consequently by requiring a lower input dimensionality, ANCFIS step aside from the "curse of dimensionality". Also input presentation is a natural consequence of using complex fuzzy sets; thus, a segment of the time series must be available to match its phase and frequency with our proposed complex fuzzy sets. Thus, usual practice of selecting prior values of the time series as orthogonal inputs to a learning algorithm ("lagged" representation) destroys this phase information, and cannot be used. The main modification in ANCFIS is addition of layer four which is a dot-product layer. This layer is closely related to the rule inference.

In ANCFIS a parameterized form of complex fuzzy set is required. The best choice is sinusoid. Under general conditions, an arbitrary periodic function can be

represented by a Fourier series, the sum of a series of sine and cosine terms. The complex fuzzy set which is used for ANCFIS is represented by:

$$r(\theta) = d\sin(a\theta + b) + c \tag{3.1}$$

where, $r(\theta)$ is magnitude, and θ is the phase of complex membership value. Also, the set of $\{a, b, c, d\}$ are adaptive parameters that manipulate the sine function, where *a* means frequency of a sine wave, *b* represents a phase shift, *c* shifts the sine wave vertically and *d* varies the amplitude of the wave. There are two constraints for the amplitude of the complex fuzzy membership, as it is within the unit disc: $0 \le d + c \le 1, 0 \le d \le c \le 1$.

Same as ANFIS, in ANCFIS, least-square optimization is applied for forward pass, and parameters of layer five (consequents) are determined. In backward pass ANFIS employs gradient descent method to determine parameters of layer one (premise). However, in ANCFIS, gradient descent is used until it reaches a point where it does not work properly, that is, the partial derivatives of the parameters of the complex fuzzy sets in ANCFIS do not have a closed form solution. Then, derivative-free optimization technique is used to update the complex fuzzy set parameters of layer one. In the basic ANCFIS architecture, the derivative-free optimization technique method which is applied is VNCSA.

3.1 The VNCSA Algorithm

In simulated annealing (SA), the value of an objective to be minimized is similar to energy in a thermodynamic annealing process. At high "temperatures," SA allows function evaluations at faraway points and it is likely to accept a new point with higher energy. As the "temperature" decreases, the algorithm is increasingly restricted to a local neighborhood. The disadvantage of SA lies in its dependence on the random number generator; a random search of a large solution space can be very slow. Chaotic maps provide an alternative means of exploring a solution space, which is confined to a (possibly fractal) subspace and thus can be much faster. A transiently chaotic neural network (TCNN) [75] was proposed to solve

combinatorial optimization problems by introducing simulated annealing to Hopfield neural network (HNN). The nonlinear dynamics approach [76] used an additive chaotic forcing function in determining the global minimum of a continuous, unconstrained or bound-constrained cost function. The Chaotic Simulated Annealing (CSA) algorithm [77] introduced the concepts of chaotic initialization and chaotic sequences to SA.

In mathematics and physics, many one–dimensional maps exhibit sensitivity to initial conditions, with small changes in initial conditions leading to large changes in the long-term outcome (popularly known as the "butterfly effect"). Due to this sensitivity, the behavior of nonlinear chaotic maps seems to be random, as any measurement error in initial conditions is magnified exponentially through time. However, these maps are deterministic, analytical functions (hence the term "deterministic chaos") [64]. An example is the logistic map given by:

$$x_{n+1} = \mu x_n (1 - x_n), \; x_n \in [0,1] \tag{3.2}$$

This map is known to be chaotic for parameter values of μ= 4.0 [8]. The long-term behavior of this map covers the entire codomain of the map (i.e. [0,1]), as long as the initial point of the map (x_0) is not one of the fixed points of the logistic map; these consist of the set {0, 0.25, 0.5, 0.75, 1.0, $\frac{5+\sqrt{5}}{8}$, $\frac{5-\sqrt{5}}{8}$, $\frac{2+\sqrt{3}}{4}$, $\frac{2-\sqrt{3}}{4}$}[8]. The Ulam-von Neumann Map is another chaotic map defined by

$$y_{n+1} = 1 - r y_n^{\,2}, \; y_n \in [-1,1] \tag{3.3}$$

This function is chaotic and covers the codomain of [-1,1] for parameter values $r \geq 2.0$. These two maps are used to replace the random number generator in the simulated annealing algorithm.

The VNCSA algorithm was developed to solve nonlinear constrained optimization problems:

$$\text{minimize} \quad f(S) \text{ subject to } c_i(S) = 0, \; c_j(S) \geq 0, \; i \in E, \; j \in I \tag{3.4}$$

where $f()$ is the objective function, S is the control variable(s), c_i is the set of equality constraints in the problem, and c_j is the set of inequality constraints; both

these sets are assumed finite. VNCSA simulates the cooling of a physical system whose possible energies correspond to the values of the objective function to be minimized, and allows solution candidates of worse quality than the current solution (uphill moves) in order to escape from local minima. The probability of allowing an uphill move is represented by the temperature parameter, which is reduced over time. The algorithm starts by producing a group of solutions satisfying all given constraints, by iterating the Logistic map from a random initial point with the parameter μ=4.0. Then, three steps are iterated until the stopping condition is reached: 1) A new solution S_{new} is generated from the neighborhood $N(S)$ by iterating the Ulam-von Neumann map; 2) S_{new} is checked against the constraints, and discarded if it does not meet them all (goto (1)); 3) $f(S_{new})$, $f(S_{current})$ and temperature T are compared to decide if S_{new} is accepted as the new current solution. The structure of the VNCSA is given in Algorithm 1. In this algorithm, L_{max} is the number of iterations at a given temperature, and M is the number of solutions generated per iteration (equally, iterations of the Ulam-von Neumann map). The distinguishing characteristic of VNCSA (differentiating it from e.g. [77]) is the variable neighborhood step, which modifies $N(S)$. After L_{max} iterations at a given temperature, both the temperature and the size of the search neighborhood are updated; the new search neighborhood will be based on the magnitude of the update at the previous temperature. This will (usually) cause the neighborhood to contract, leading to a further speedup in the algorithm.

```
Begin
   S_current := GenerateInitialSolutionPopulationLogisticMap()
   T_0 := SetInitializationTemperature()
   While (T_k < T_min) do
      While(l ≤ L_max) do
         While(m < M)
            S_new := PickNeighborAtUlam_von_NeumannMap()
            if f(S_new) < f(S_current) then
               S_current := S_new
            else
               accept S_new as new solution with probability
                     exp(−(f(S_new) − f(S_current))/T_k)
            end if
         end while
      end while
      UpdateNeighborhood( D )
      UpdateTemperature( T_k )
   end while
   S_best := S_current
   output: S_best viewed as optimization solution for x
End
```

Algorithm 1: VNCSA

3.2 ANCFIS Architecture

As mentioned before ANCFIS has six layers. In the following, it is described how each layer works:

Layer 1: Premise parameters

Here premise parameters are $\{a_i, b_i, c_i, d_i\}$ with $i = 1, 2\ldots, n\text{-}CMF$, where *n-CMF* represents the number of complex membership functions. In this layer the convolution of each membership function (MF) and input vector are computed. The MF is sampled by:

$$r_k(\theta_k) = d\sin(a\theta_k + b) + c \tag{3.5}$$

$$\theta_k = \frac{2\pi}{n} k, \tag{3.6}$$

Here n is length of input vector and k is the element index of complex samples, k=1, 2, …, n. The sampled points are converted from polar to rectangular coordinates using well-known transforms [4]:

$$x_k = r_k \cos(\theta_k) \tag{3.7}$$

$$y_k = ir_k \sin(\theta_k) \tag{3.8}$$

These samples with complex value are convolved with the original real-valued input vector. Considering f is input vector and g is sampled point vector and n is length of f and m is length of g, a vector of length m+n-1 is called h whose kth element is:

$$h(k) = \sum_j f(j) g(k+1-j) \tag{3.69}$$

$$g(k+1-j) = x_{k+1-j} + iy_{k+1-j} = r_{k+1-j} \cos(\theta_{k+1-j}) + ir_{k+1-j} \sin(\theta_{k+1-j}) \tag{3.10}$$

This sum covers all values of j which lead to subscript for $f(j)$ and $g(k+1-j)$;

$j = \max(1, k+1-n) : \min(k, m)$.

Suppose that both two vectors are within the same length; m=n; the result is:

$h(1) = f(1) * g(1)$

$h(2) = f(1) * g(2) + f(2) * g(1)$

....

$h(n) = f(1) * g(n) + f(2) * g(n-1) + ... + f(n) * g(1)$

....

$h(2n-1) = f(n) * g(n))$

In conclusion the convolution sum is equal to:

$$\sum_{k=1}^{2n-1} h(k) = \sum_{k=1}^{2n-1} \sum_{j=\max(1,k+1-n)}^{\min(k,n)} f(j) g(k+1-j) \tag{3.11}$$

Convolution sum can be expressed as a form of neural network in Figure 3.2. SMF_{ikj} is the weight which connecting the j^{th} element of input vector ANCFIS to k^{th} sample point of i^{th} membership function in first layer. This insight provides essential guidance in driving the gradient descent formula for ANCFIS learning algorithm (however, note that this is only a conceptual tool, the neural network in

Figure 3.1 does not in fact exist). So the convolution sum equation can be represented this way:

$$Convolution\ sum = \sum_{k=1}^{n}\left(O_{o,j} \times \sum_{k=1}^{n} SMF_{ikj} \right) \text{ [8]} \tag{3.12}$$

Where n is the length of the input vector, and $O_{0,j}$, $j = 1,2,\ldots n$, denotes the jth element of the input vector presented to ANCFIS. This sum might not be restricted to the unit disc of the complex plane; to enforce this restriction without changing the phase of the sum, the Elliot function is employed [78]:

$$f(z) = \frac{z}{1+|z|} \tag{3.13}$$

which z is a complex number.

Substituting Eq. (2.13) into Eq. (2.14), the output of the nodes in layer one is given by:

$$O_{1,i} = \frac{\sum_{k=1}^{n}\left(O_{o,j} \times \sum_{k=1}^{n} SMF_{ikj} \right)}{1+\left| \sum_{k=1}^{n}\left(O_{o,j} \times \sum_{k=1}^{n} SMF_{ikj} \right) \right|}, i = 1,2,...,n\_CMF \tag{3.14}$$

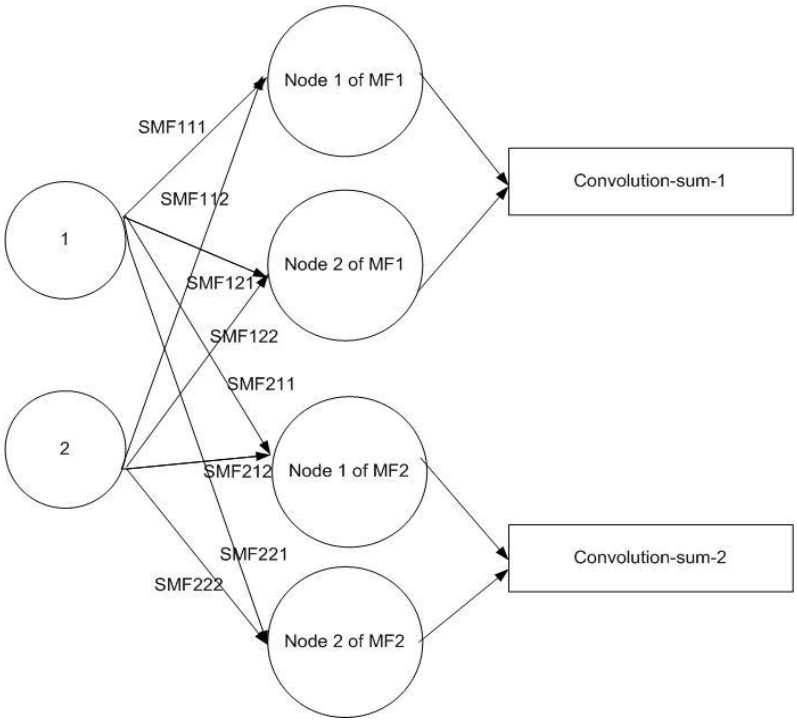


Figure 3.2: Implicit structure in the convolution of input vector and sampled points generated from complex membership function ***(*** *SMF111 = SMF112* **,** *SMF121 = SMF122* **,** *SMF211 = SMF 212* **,** *SMF221 = SMF 222* ***)*** **[8]**

Layer 2: Firing strength

Each node in this layer is a fixed node and output is product of its complex inputs.

$$O_{2,i} = \prod_i O_{1,i}, i = 1,2,..., |O_1| \tag{3.15}$$

In Figure 3.1, the network only has one input vector $\vec{x}$ (univariate time series) so the output of layer two is the same as layer one. Each output of layer two represents the firing strength of a fuzzy rule. In multivariate time series, it can also involve interaction between inputs. The algebraic product is identified as a complex fuzzy conjunction in [1], and was proposed as a complex fuzzy intersection in [79].

Layer 3: Normalized firing strength

This layer is responsible for normalizing the firing strength:

$$O_{3,i} = \overline{w_i} = \frac{w_i}{\sum_{j=1}^{|O_2|} |w_j|}, i = 1,2,...,|O_2| \quad (3.16)$$

Here $\sum_{j=1}^{|O_2|} |w_j|$ represents the sum of magnitude of each weight w_j and $|O_2|$ is the number of rules. In this layer, only magnitude is normalized and phase does not change.

Layer 4: Dot product

Output of each node in this layer is dot product of normalized firing strength and sum of outputs of all nodes in the previous layer. The equation is:

$$O_{4,i} = w_i^{DP} = \overline{w_i} \bullet \sum_{i=1}^{|O_3|} \overline{w_i}, i = 1,2,...,|O_3| \quad (3.17)$$

$|O_3|$ is the number of nodes in layer 3 and $\sum_{i=1}^{|O_3|} \overline{w_i}$ is a complex sum. Output of this layer is a real value.

Layer 5: Consequent parameters

Each node in this layer is an adaptive node in the hybrid learning rule. Calculation of each output node i is represented by:

$$\begin{aligned} O_{5,i} &= w_i^{DP} f_i = w_i^{DP} (\overrightarrow{p_i} \vec{x} + r_i) \\ &= w_i^{DP} (p_{i,1} x_1 + p_{i,2} x_2 + .. + p_{i,j} x_j + ... + p_{i,n} x_n + r_i), j = 1,2,..,n \end{aligned} \quad (3.18)$$

where, w_i^{DP} is the i^{th} output of the layer four and x_j is the j^{th} value in the input vector, n is the input vector length and $\{\overrightarrow{p_i}, r_i\}$ is the parameter set for the linear output function. The length of $\overrightarrow{p_i}$ is the same as input vector $\vec{x}$ and r_i is a constant. Parameters in this layer are called 'consequent parameters', which are identified in forward pass using a linear least square estimator ([7], [80], [81]).

Layer 6: Final output

This layer has just one node which is the overall summation of incoming signals of layer five.

$$O_{6,1} = w_i^{DP} = \sum w_i^{DP} f_i \tag{3.19}$$

3.3 ANCFIS Back propagation

In this section, back propagation for updating the sampled points will be discussed. Here premise parameters are not updated as there is no closed-form solution for derivative of network error [8]. Instead, the SMF (sample membership function) points are updated.

The sum of squared error is used for error measure for p^{th} entry of training dataset. Here, $1 \le p \le P$, and sum of squared error is defined as below:

$$E_p = \sum_{q=1}^{N(L)} (d_q - x_{L,q})^2 \tag{3.20}$$

where, d_q is q^{th} of p^{th} desired output vector, L refers to layer number and $x_{L,q}$ is q^{th} component of p^{th} actual output vector of the whole network. The main purpose of ANCFIS is to minimize the total errors of the system which is sum of all E_p. In terms of defining the error signal, E_p is defined as the derivative of E_p regarding to the output of node *i* in layer *l*.

$$\varepsilon_{l,i} = \frac{\partial^+ E_p}{\partial x_{l,i}} \tag{3.21}$$

The error signal of *i*th output node in layer L is calculated by equation 3.19.

$$\varepsilon_{L,i} = \frac{\partial^+ E_p}{\partial x_{L,i}} = \frac{\partial E_p}{\partial x_{L,i}} = -2(d_i - x_{L,i}) \tag{3.22}$$

Then, for each internal node of ANCFIS network the error signal is calculated as below:

$$\varepsilon_{l,i} = \frac{\partial^{+} E_p}{\partial x_{l,i}} = \sum_{l=1}^{N(m+1)} \frac{\partial E_p}{\partial x_{l,i}} \frac{\partial f_{l+1,m}}{\partial x_{L,i}} = \sum_{l=1}^{N(m+1)} \varepsilon_{l+1,m} \frac{\partial f_{l+1,m}}{\partial x_{l,i}} \tag{3.23}$$

In equation 3.23, the index i refer to node position in layer l and error signals and m refers to nodes in layer $l+1$. For each internal node, error signal is as a linear combination of error signals from next layer that is l+1.

For calculating the partial derivatives related to complex variables, the approach in ANCFIS is similar to [82] and alternative approach is described in [63]. If one writes:

$$f(z) = u(z) + iv(z) = u(x, y) + iv(x, y) \tag{3.24}$$

And let $z = x + iy, \bar{z} = x - iy$ and $dz = dx + idy$; then:

$$x = \frac{z + \bar{z}}{2} \tag{3.25}$$

$$y = \frac{z - \bar{z}}{2i}$$

Then, the partial derivatives of the Elliot function, $\frac{\partial u}{\partial x}, \frac{\partial v}{\partial x}, \frac{\partial u}{\partial y}, \frac{\partial v}{\partial y}$ can be stated as:

$$u_x = \begin{cases} \frac{\left(y^2 + |z|\right)}{|z|\left(1 + |z|\right)^2} & \text{if } |z| \neq 0 \\ 1 & |z| = 0 \end{cases} \tag{3.26}$$

$$u_y = \begin{cases} \frac{-xy}{|z|\left(1 + |z|\right)^2} & \text{if } |z| \neq 0 \\ 0 & |z| = 0 \end{cases}$$

$$v_x = \begin{cases} \frac{-xy}{|z|\left(1 + |z|\right)^2} & \text{if } |z| \neq 0 \\ 0 & |z| = 0 \end{cases}$$

$$v_y = \begin{cases} \frac{\left(x^2 + |z|\right)}{|z|\left(1 + |z|\right)^2} & \text{if } |z| \neq 0 \\ 1 & |z| = 0 \end{cases}$$

After finding all values related to all $\varepsilon_{l,i}$ s, the gradient for SMF in the first layer can be computed:

$$x_{1,i} = f(z_{1,i}) = u_{1,i} + iv_{1,i}\ ,\ z_{1,i} = x^{1,i} + iy^{1,i} = \sum_{j=1}^{n}\left(x_{0,j} \times \sum_{k=1}^{n} SMF_{ikj} \right) \tag{3.27}$$

Here $x_{1,i}$ is the neuron i in the first layer of the network. The partial derivatives for x and y with respect to that real and imaginary parts are R and I, are as follows:

$$\frac{\partial x^{1,i}}{\partial SMF_{ikjR}} = x_{0,j}\,, \frac{\partial y^{1,i}}{\partial SMF_{ikjR}} = 0\,, \frac{\partial x^{1,i}}{\partial SMF_{ikjI}} = 0\,, \frac{\partial y^{1,i}}{\partial SMF_{ikjI}} = x_{0,j}\,. \tag{3.28}$$

The function E_p includes both $u_{1,i}(x^{1,i}, y^{1,i})$ and $v_{1,i}(x^{1,i}, y^{1,i})$, and $x^{1,i}$ and $y^{1,i}$ are both functions of SMF_{ikjR} and SMF_{ikjI}. The gradient of error function with respect to the real part of SMF_{ikjR} is expressed by:

$$\frac{\partial^+ E_p}{\partial SMF_{ikjR}} = \frac{\partial^+ E_p}{\partial u_{1,i}}\left(\frac{\partial u_{1,i}}{\partial x^{1,i}} \frac{\partial x^{1,i}}{\partial SMF_{ikjR}} + \frac{\partial u_{1,i}}{\partial y^{1,i}} \frac{\partial y^{1,i}}{\partial SMF_{ikjR}} \right) +$$

$$\frac{\partial^+ E_p}{\partial v_{1,i}}\left(\frac{\partial v_{1,i}}{\partial x^{1,i}} \frac{\partial x^{1,i}}{\partial SMF_{ikjR}} + \frac{\partial v_{1,i}}{\partial y^{1,i}} \frac{\partial y^{1,i}}{\partial SMF_{ikjR}} \right)$$

$$= \left(\frac{\partial^+ E_p}{\partial u_{1,i}} \frac{\partial u_{1,i}}{\partial x^{1,i}} + \frac{\partial^+ E_p}{\partial v_{1,i}} \frac{\partial v_{1,i}}{\partial x^{1,i}} \right) x_{0,j}\,, \tag{3.29}$$

In 3.29, $\frac{\partial^+ E_p}{\partial u_{1,i}}$ and $\frac{\partial^+ E_p}{\partial v_{1,i}}$ are the real and the imaginary parts of the error signals of node i in the first layer, the terms $\frac{\partial u_{1,i}}{\partial x^{1,i}}$ and $\frac{\partial v_{1,i}}{\partial x^{1,i}}$ are the partial derivatives of the real and imaginary parts of the Elliott function with respect to the real part of the convolution sum. Now for the gradient of the error function with respect to the imaginary part SMF_{ikjI}, the equation is as written by:

$$\frac{\partial^+ E_p}{\partial SMF_{ikjI}} = \frac{\partial^+ E_p}{\partial u_{1,i}}\left(\frac{\partial u_{1,i}}{\partial x^{1,i}} \frac{\partial x^{1,i}}{\partial SMF_{ikjI}} + \frac{\partial u_{1,i}}{\partial y^{1,i}} \frac{\partial y^{1,i}}{\partial SMF_{ikjI}} \right) +$$

$$\frac{\partial^+ E_p}{\partial v_{1,i}}\left(\frac{\partial v_{1,i}}{\partial x^{1,i}}\frac{\partial x^{1,i}}{\partial SMF_{ikjI}}+\frac{\partial v_{1,i}}{\partial y^{1,i}}\frac{\partial y^{1,i}}{\partial SMF_{ikjI}}\right)$$

$$=\left(\frac{\partial^+ E_p}{\partial u_{1,i}}\frac{\partial u_{1,i}}{\partial y^{1,i}}+\frac{\partial^+ E_p}{\partial v_{1,i}}\frac{\partial v_{1,i}}{\partial y^{1,i}}\right)x_{0,j}\ , \tag{3.30}$$

Now, for gradient of error function E_p with respect to the complex sample SMF_{ikj}, both equations 3.26 and 3.27 are used:

$$\frac{\partial^+ E_p}{\partial SMF_{ikj}}=\frac{\partial^+ E_p}{\partial SMF_{ikjR}}+i\frac{\partial^+ E_p}{\partial SMF_{ikjI}}$$

$$=\left(\frac{\partial^+ E_p}{\partial u_{1,i}}\frac{\partial u_{1,i}}{\partial x^{1,i}}+\frac{\partial^+ E_p}{\partial v_{1,i}}\frac{\partial v_{1,i}}{\partial x^{1,i}}\right)x_{0,j}+i\left(\frac{\partial^+ E_p}{\partial u_{1,i}}\frac{\partial u_{1,i}}{\partial y^{1,i}}+\frac{\partial^+ E_p}{\partial v_{1,i}}\frac{\partial v_{1,i}}{\partial y^{1,i}}\right)x_{0,j} \tag{3.31}$$

After calculation of all $\varepsilon_{l,i}$, the update for a generic weight SMF_{ikj} can be executed by:

$$\frac{\partial^+ E_p}{\partial SMF_{ikj}}=\varepsilon_{1,i}\frac{\partial f_{1,i}}{\partial SMF_{ikj}} \tag{3.32}$$

and,

$$\frac{\partial^+ E}{\partial SMF_{ikj}}=\sum_{p=1}^{P}\frac{\partial^+ E_p}{\partial SMF_{ikj}} \tag{3.33}$$

where, $f_{1,i}$ is related to the function of node i of the first layer.

By using steepest descent, the formula for generic complex sample is updated as 3.34.

$$\Delta SMF_{ikj}=-\eta\frac{\partial^+ E}{\partial SMF_{ikj}}=-\eta\sum_{p=1}^{P}\frac{\partial^+ E_p}{\partial SMF_{ikj}} \tag{3.34}$$

where, η is the learning rate parameter which is defined by user. As with ANFIS, the learning rate parameter is adapted using the following heuristic rules [7]:

1. If the parameter undergoes m consecutive reductions, increase η by a user-defined factor: $\eta=\eta*increase\ rate$

2. If the parameter undergoes *n* consecutive combinations of one reduction and one increase, decrease η by a user-defined factor: $\eta = \eta * decrease\ rate$

Once all ε_i's are obtained, one can straightforwardly calculate the gradient for a generic weight SMF_{ikj} in node i according to Equations 3.32 and 3.33.

Chapter 4:

Off-Line experiments on Univariate Datasets

In this chapter, the results of a series of univariate forecasting experiments using ANCFIS are described. The experimental design used in this chapter is typically applied in time series forecasting: we use a single-split design, in which all training data are chronologically earlier than the holdout test sample. The learning objective is a one-step-ahead prediction task. In the context of ANCFIS, this means predicting the time series value immediately following the most recent entry in the input window. The time series are normalized to the interval [0,1]. The size of the input window is selected to approximately cover one "period" in the time series. Our experiments include five real-world datasets: Sunspot [15]-[21], Santa Fe A (laser) [11]–[14], Waves [23], Mackey-Glass [5][9][10] and Stellar (Star) [21]-[23]. Mackey-Glass and Santa Fe A (Laser) are known to be chaotic time series.

The performance of ANCFIS is compared to current results for each dataset in the literature. The time-series forecasting literature employs several different measures of forecast error; these include the Mean Squared Error (MSE), Normalized Mean Squared Error (NMSE), the Non-Dimensional Error Index (NDEI), Absolute Error, (AE), and Average Relative Variance (ARV); this latter is equivalent to NMSE. These measures are defined as:

$$AbsError = \sum_{i=1}^{n} \frac{|y_i - \hat{y}_i|}{n} \tag{4.1}$$

$$RMSE = \sqrt{\sum_{i=1}^{n} \frac{(y_i - \hat{y}_i)^2}{n}} \tag{4.2}$$

$$MSE = \sum_{i=1}^{n} \frac{(y_i - \hat{y}_i)^2}{n} \tag{4.3}$$

$$NMSE = \frac{1}{\sigma_x^{\,2} n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \tag{4.4}$$

$$NDEI = \frac{\sqrt{MSE}}{\sigma_x} = \frac{\sqrt{\sum_{i=1}^{n} \frac{(y_i - \hat{y}_i)^2}{n}}}{\sigma_x} \tag{4.5}$$

$$\sigma_x^{\,2} = \frac{1}{n-1} \sum_{i=1}^{n} (y_i - \overline{y_i})^2 \tag{4.6}$$

where, σ_x is standard deviation and $\sigma_x^{\,2}$ is variance of the test time series. y_i is the desired output, $\hat{y}_i$ is the estimated output and n is the total number of examples in the testing dataset.

4.1 Mackey-Glass dataset

The time series used is the Mackey-Glass function, given by:

$$\dot{x}(t) = \frac{0.2x(t-\tau)}{1 + x^{10}(t-\tau)} - 0.1x(t) \tag{4.7}$$

This is the same time series as Jang [5][9][10] where time step is 0.1, initial condition $x(0)$ =1.2 and t =17 , with 1000 points; these run from t=124 to t=1123, to avoid initialization transients. As in Jang's work, the first 500 data points are used as the training set, while the remaining 500 pairs are the test data set. A window of 44 data points is used as our input vector.

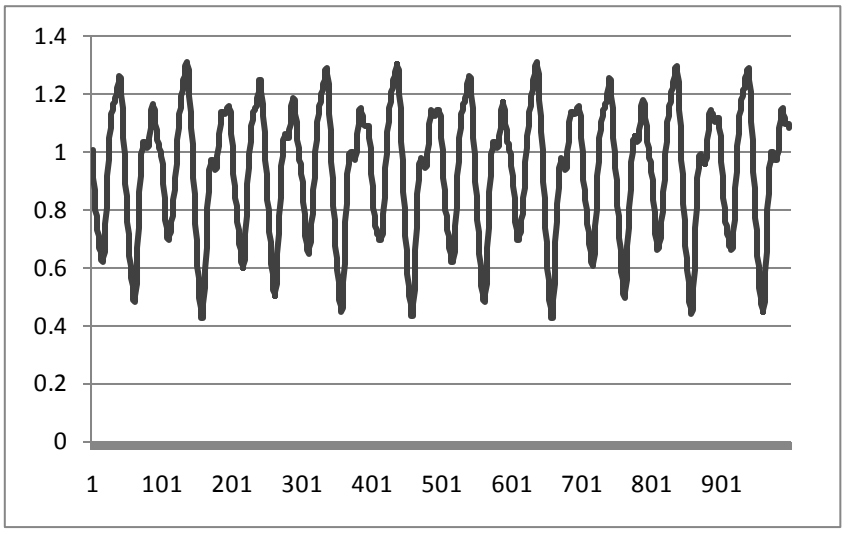


Figure 4.1: Mackey-Glass dataset

Table 4.1 presents all the parameters applied to ANCFIS to find best results for Laser dataset and also Table 4.2 presents the best results found for different error measurements.

Variates	Input Length	Outputs	CFS per Input	Step Size	Increase Rate	Decrease Rate	Tmax	Lmax	Alpha	Weight	Tmin	Beta	M
1	44	1	3	0.001	1.1	0.8	100	2	0.99	0.95	0.01	0.98	400

Table 4.1: Training Parameters for Mackey-Glass Dataset

RMSE	MSE	NMSE	NDEI
0.000141	3.099e (-7)	0.0027	1.3721e(-6)

Table 4.2: Different error measurements for Mackey-Glass

The results for ANCFIS are compared to the published literature in Table 4.3. ANCFIS consistently had lower errors on the various measures reported in the literature as compared to the existing techniques.

Method	NDEI	MSE	NMSE
ANCFIS	0.0027	3.099×10^{-7}	1.37×10^{-6}
ANFIS [83]	0.007	-	-
AR model [83]	0.19	-	-
Cascade-correlation NN [84]	0.06	-	-
Backpropagation NN [84]	0.02	-	-
6^{th} – order Polynomial [84]	0.04	-	-
Linear predictor [84]	0.55	-	-
TSK-NFIS [21]	0.0406	2.18×10^{-5}	-
AR model [21]	0.0492	3.2×10^{-5}	-
NAR [21]	0.0466	2.89×10^{-5}	-
Neural Network [21]	0.0488	3.21×10^{-5}	-
SVR [17]	-	-	4.5×10^{-3}
Bagging SVR [17]	-	-	2.0×10^{-3}
Boosting SVR (median) [17]	-	-	8.2×10^{-4}
Boosting SVR (mean) [17]	-	-	8.0×10^{-4}

Table 4.3: Comparison of Test Error for Mackey-Glass Dataset

Further analysis of the ANCFIS results are provided in Figure4.2 and 4.3, where we plot the predicted and actual outputs in part 4.2 and the prediction errors in 4.3. Plainly, the ANCFIS forecasts track this chaotic time series quite closely, using only three rules.

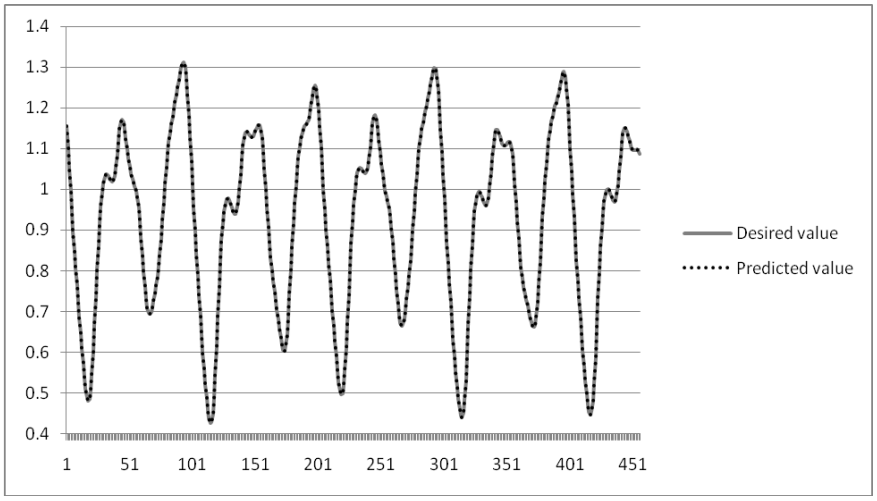


Figure 4.2: Mackey-Glass test results for one-step prediction

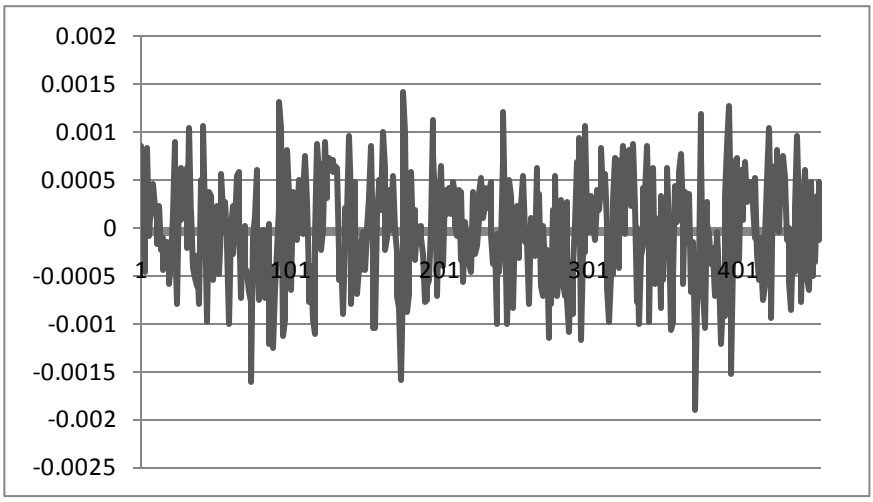


Figure 4.3: Mackey-Glass test errors

A detailed explanation of the analysis of the learning taking place in Layer 1 of ANCFIS is presented in Table 4.4, where we present the three complex membership functions after training. As a usual practice, these parameters are initialized to small random values prior to the start of the training process. It should be noted that the phase of the third membership function is constant (a=0), meaning that the entire membership function becomes a constant value.

	a	b	c	d
CFS 1	0.969617	48.03118	0.861239	0.085948
CFS 2	6.31E-05	39.60439	0.129598	0.031515
CFS 3	0	83.98123	0.345205	0.119537

Table 4.4: Membership Functions after Training for Mackey-Glass dataset

4.2 Santa Fe A (Laser) dataset

Santa Fe A is the first dataset in a series of six datasets of Santa Fe time series competition [14] in 1991. This is a univariate time series dataset which was contributed by Udo Huebner [11]–[13] and were collected primarily by N. B. Abraham and C. O. Weiss. These data were recorded from a Far-Infrared-Laser in a chaotic state. Specially, the measurements were made on an 81.5-micron 14NH3

cw (FIR) laser, pumped optically by the P(13) line of an N2O laser via the vibration of aQ(8,7) NH3 transition. The normalized laser data are shown in Figure 4.4. The first 900 data points are used as training data, while the last 100 are used as testing data (in common with [54]). The length of input vector is set to 8 leading to 892 input-output data pairs as training data set and 92 data pairs as testing data set for ANCFIS.

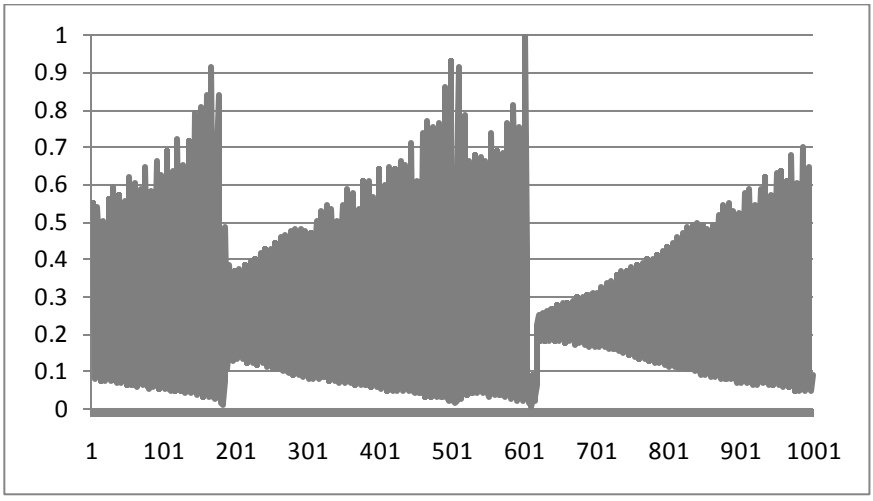


Figure 4.4: Santa Fe A after normalization

Table 4.5 represents all the parameters applied to ANCFIS to find best results for Laser dataset and also Table 4.6 represents the best results found for different error measurements.

Variates	Input Length	Outputs	CFS per Input	Step Size	Rate Increase	Rate Decrease	Tmax	Lmax	Alpha	Weight	Tmin	Beta	M
1	8	1	2	0.001	1.1	0.7	100	2	0.99	0.95	0.01	0.98	400

Table 4.5: Parameters for the Santa Fe A Dataset

RMSE	MSE	NMSE	NDEI
0.033	0.001089	0.0274	0.1655

Table 4.6: Different error measurements for Santa Fe A

We compare the results for ANCFIS and the published literature in Table 4.7. Overall, ANCFIS is in the general range of recent forecasting results for this dataset (while using only 2 rules), but does not yield the lowest error.

Method	**NMSE**
ANCFIS	0.0274
[85]	0.028
[80]	0.026
[11]	0.0701
[86]	0.099
MLP [87]	0.0996
MLP IT [87]	0.1582
LSTM [87]	0.3959
LSTM IT [87]	0.3642
Linear [87]	1.2505
FIRN [88]	0.023
sFIRN [88]	0.0273
MLP [89]	0.0177
RSOM [89]	0.0833
EP-MLP [90]	0.2159
[91]	0.077
[92]	0.016
[93]	0.029
Method	**MSE**
ANCFIS	0.001089
[86]	0.0014

Table 4.7: Comparison of testing errors for Santa Fe dataset A (Laser)

Further analyses of the ANCFIS results are presented. Actual versus predicted outputs and prediction errors are given in Figures 4.5 and 4.6. Again, it can be observed that ANCFIS tracks this dataset fairly well, except for a small number of points near the "peaks" of the output. The trained CFS parameters are presented in Table 4.8.

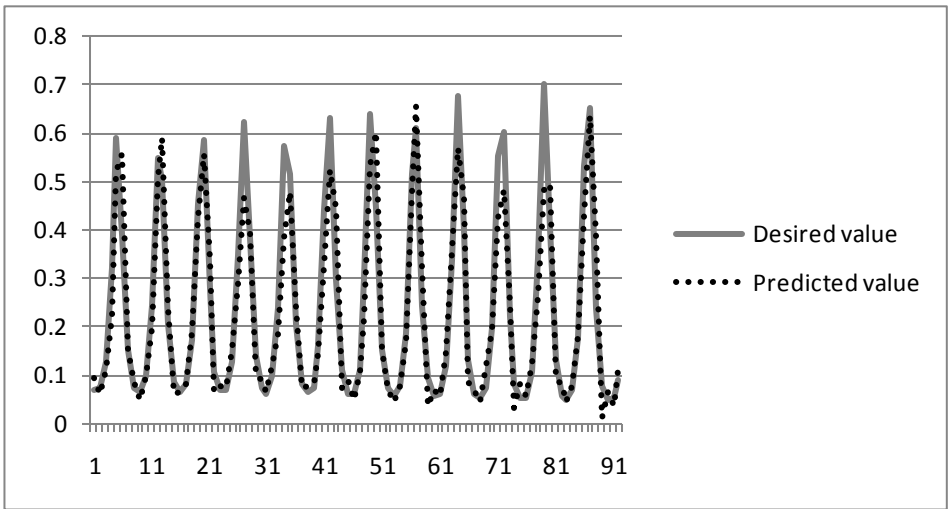


Figure4.5: Santa Fe A test results for one-step prediction

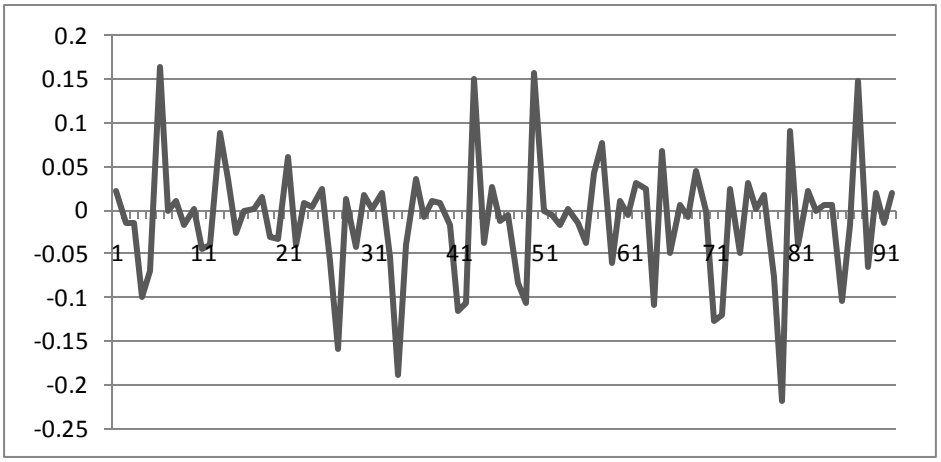


Figure4.6: Santa Fe A prediction error

	a	B	c	d
CFS 1	6.988808	9.29139	0.690029	0.174359
CFS 2	8.956521	6.035124	0.354891	0.117247

Table 4.8: Membership Functions after Training for Santa Fe A dataset

4.3 Sunspot dataset

Zurich or Wolf sunspot number (now commonly referred to as sunspot number) is the average number of sunspots per year as measured from 1700 to 1979. This

time series was chosen because it is a commonly cited time series dataset [15][16][94][95] and [17]-[21]. The normalized data is shown in Figure 4.7.

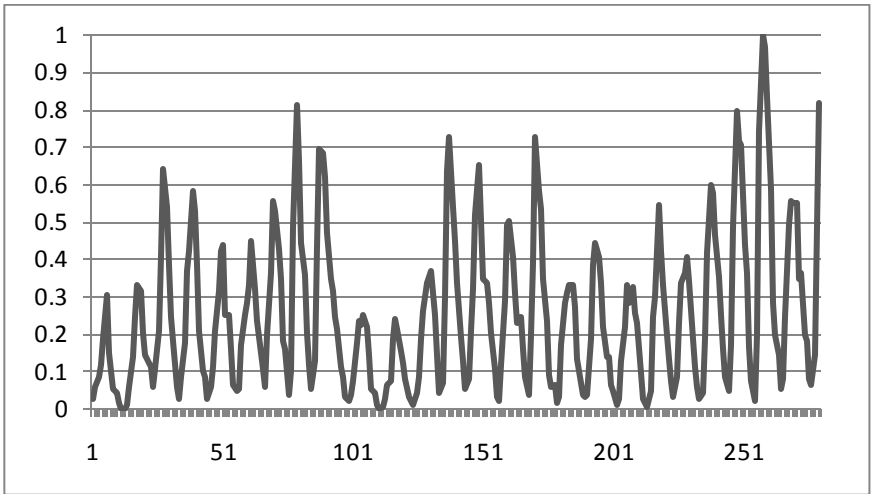


Figure4.7: Sunspot after normalization

The years 1700-1920 are used as training data, while the remaining years up to 1979 are used as testing data. This is consistent with the experiments from [18]. In other papers [15]-[17], [94][95] and [19]-[21], authors used different number of data points as training data and testing data.

Variates	Input Length	Outputs	Input CFS per	Step Size	Rate Increase	Rate Decrease	Tmax	Lmax	Alpha	Weight	Tmin	Beta	M
1	12	1	3	0.001	1.1	0.8	100	2	0.99	0.95	0.01	0.98	400

Table 4.9: Training Parameters for Sunspot Dataset

RMSE	MSE	NMSE	NDEI
0.091	0.00829	0.3608	0.1302

Table 4.10: Different error measurements for Sunspot

The forecasting results are compared to the literature in Table 4.11. ANCFIS is often superior, except against one method in a 1997 article, and one publication in 2009 (last four rows). However, while the four methods investigated by [96]

yielded a lower MSE, the ANCFIS architecture is superior when the same results are measured by NDEI. This likely means the four methods in [96] and ANCFIS are not significantly different on this dataset.

Method	ARV	NMSE	MSE	NDEI
ANCFIS	0.1302	0.1302	8.29×10^{-3}	0.3608
ARMA [97]	0.252	-	-	-
Elman [97]	0.348	-	-	-
Extended Elman [97]	0.162	-	-	-
FIR [97]	0.115	-	-	-
SVR [17]	-	0.64	-	-
Bagging SVR [17]	-	0.58	-	-
Boosting SVR (median) [17]	-	0.27	-	-
Boosting SVR (mean) [17]	-	0.33	-	-
SVM [20]	-	0.178	-	-
SVM Ensemble [20]	-	0.1541	-	-
[98]	-	0.28	-	-
[99]	-	0.35	-	-
TSK-NFIS [21]	-	-	1.32×10^{-3}	0.38
AR [21]	-	-	1.36×10^{-3}	0.385
NAR [21]	-	-	2.68×10^{-3}	0.541
Neural Network [21]	-	-	2.17×10^{-3}	0.486
	ANCFIS	**[58]**	**[18]**	
Absolute Error	0.068	13.83	13.73	

Table 4.11: Comparison of Testing Error for Sunspot Dataset

Further analysis of ANCFIS is provided in Figures 4.8 and 4.9. Again, Figure 4.8 presents actual versus predicted values, while 4.9 presents prediction errors. The trained membership function parameters are presented in Table 4.12.

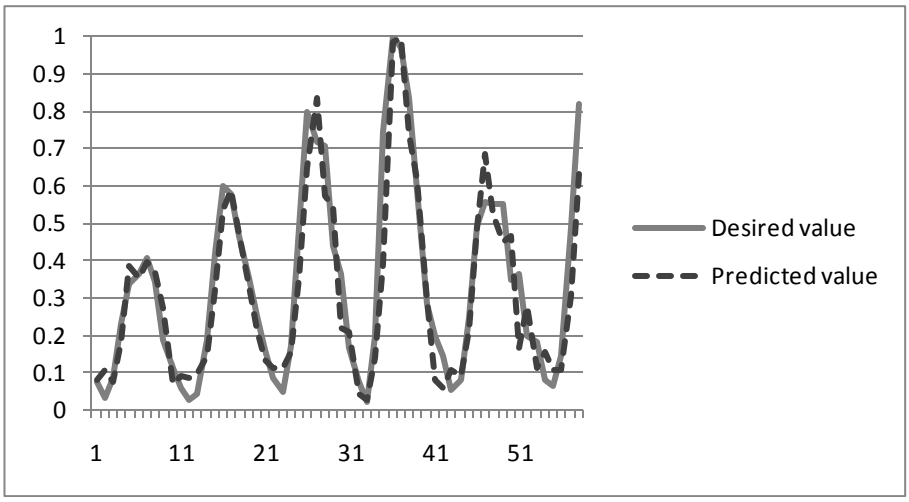


Figure4.8: Sunspot test results for one-step prediction

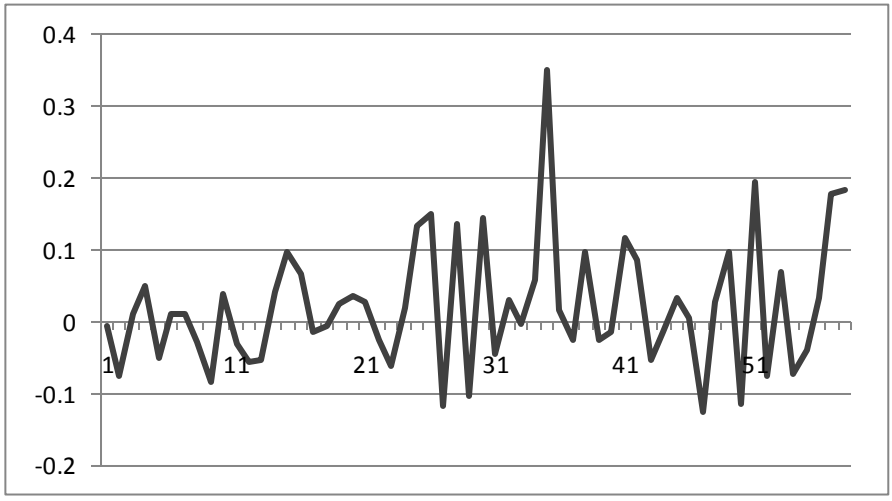


Figure4.9: Sunspot prediction error

	a	b	c	d
CFS 1	1.000299	81.34112	0.771363	0.109397
CFS 2	1.76E-05	16.28847	0.463418	0.258022
CFS 3	2.421611	65.95746	0.196038	0.011797

Table 4.12: Membership Functions after Training for Sunspot dataset

4.4 Stellar (Star) dataset

Star dataset refers to a record of daily brightness of a variable star during 600 nights; see the normalized data in Figure 4.10. There are 480 data points that are chosen as training dataset and the rest which is 120 is used as testing dataset. The

length of input vector is set at 27 which give a total number of 93 for testing data pairs and 453 for training data pairs. Table 4.13 represents training parameters applied to Star testing dataset.

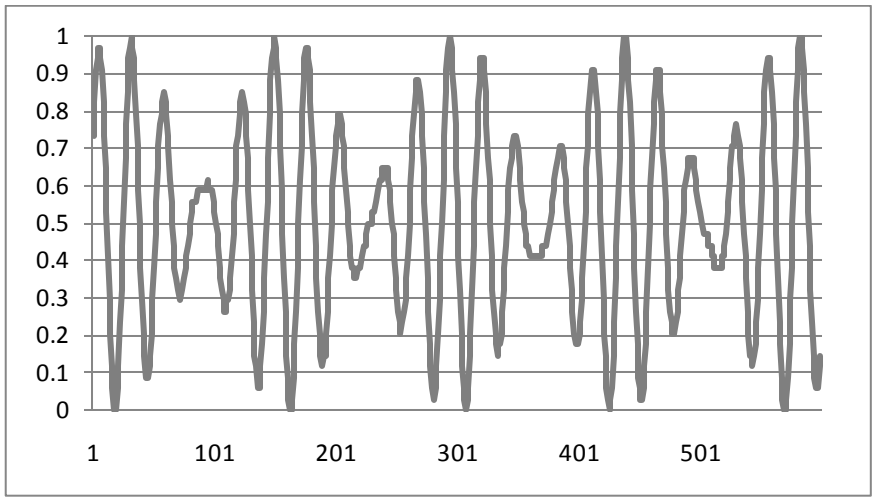


Figure4.10: Star after normalization

Variates	Length Input	Outputs	Input Cross per	Step Size	Rate Increase	Rate Decrease	Tmax	Lmax	Alpha	Weight	Tmin	Beta	M
1	27	1	3	0.001	1.1	0.8	100	2	0.99	0.95	0.01	0.98	400

Table 4.13: Training Parameters for the Star Dataset

RMSE	MSE	NMSE	NDEI
0.00749	5.6106e(-5)	0.0029	0.00084

Table 4.14: Different error measurements for Star

Our forecasting results are compared to the literature in Table 4.15. Again, further analysis of the ANCFIS results are provided in Figures 4.11 and 4.12. Actual versus predicted outputs are plotted in Figure 4.11, while prediction errors are plotted in 4.12.

Method	MSE	NDEI
ANCFIS	5.61×10^{-5}	0.0029
TSK-NFIS [21]	3.31×10^{-4}	0.0609
AR [21]	3.22×10^{-4}	0.0601
NAR [21]	3.12×10^{-4}	0.0592
Neural Network [21]	3.11×10^{-4}	0.0591

Table 4.15: Comparison of Testing Error for Star dataset

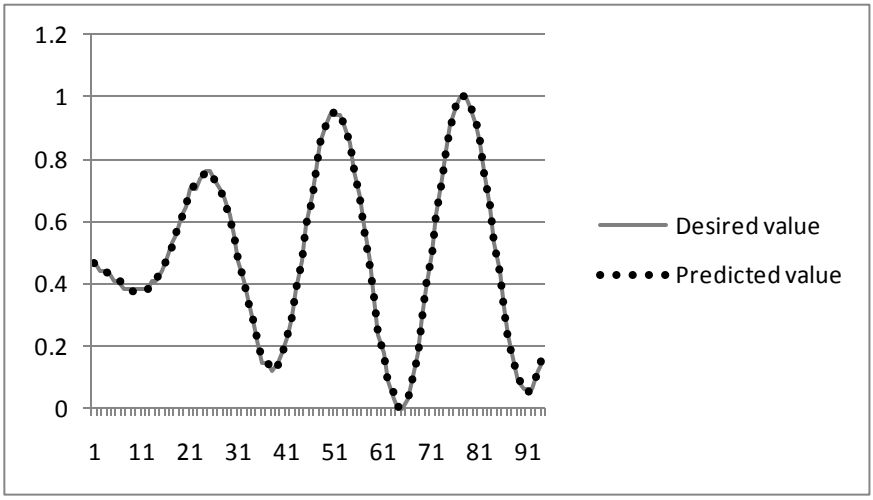


Figure4.11: Star test results for one-step prediction

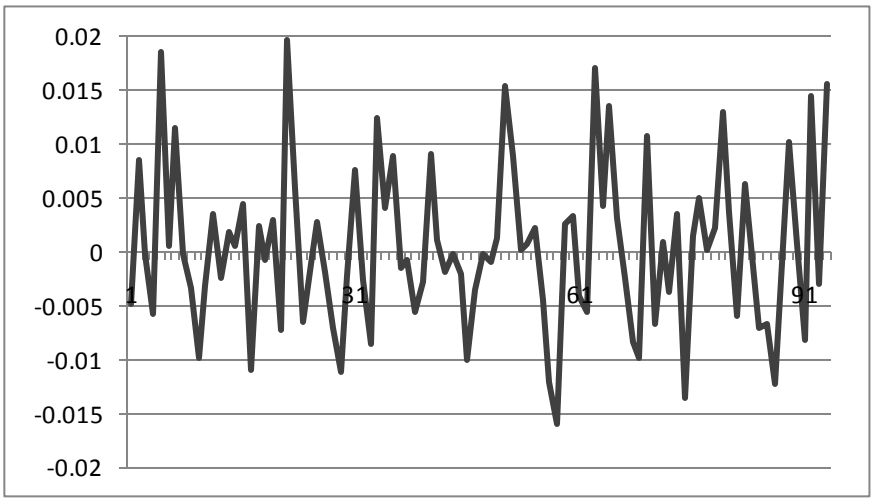


Figure4.12: Star prediction error

The trained membership function parameters appear in Table 4.16. It is interesting to note that the values for the *d* parameter are extremely small for the two of the three CFSs. This will mean that the magnitude of the CFS will be nearly constant (equal to the value of *c*).

	a	b	c	d
CFS 1	1.305374	57.13477	0.184654	0.0001
CFS 2	0.859259	12.02859	0.281818	0.094417
CFS 3	6.663302	58.42737	0.431561	0.0001

Table 4.16: Membership Functions after Training for Star dataset

4.5 Waves dataset

Wave's dataset [23] is a time series that records forces on a cylinder suspended in a tank of water with sampling interval 0.15 second and contains 320 data points as shown in Figure 4.13. We choose the first 256 points as the training data set, and the remaining 64 points as the testing data. Training parameters are listed in Table 4.17.

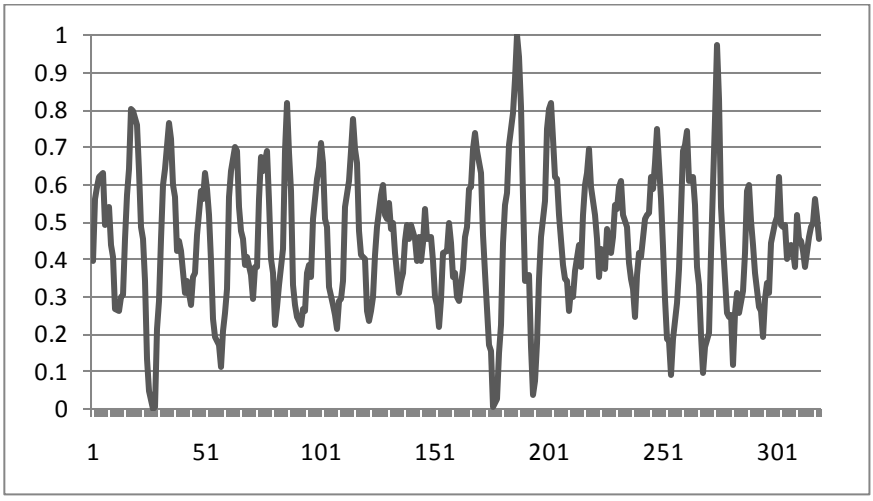


Figure4.13: Waves after normalization

Variates	Input Length	Outputs	CFS per Input	Step Size	Increase Rate	Decrease Rate	Tmax	Lmax	Alpha	Weight	Tmin	Beta	M
1	12	1	3	0.001	1.1	0.8	100	2	0.99	0.95	0.01	0.98	400

Table 4.17: Training Parameters for the Waves Dataset

Table 4.18 presents the best results found for different error measurements.

RMSE	MSE	NMSE	NDEI
0.0567	0.003215	0.3136	0.0983

Table 4.18: Different error measurements for Waves

Actual versus predicted outputs are plotted in Figure 4.14 and prediction errors are plotted in Figure 4.15.

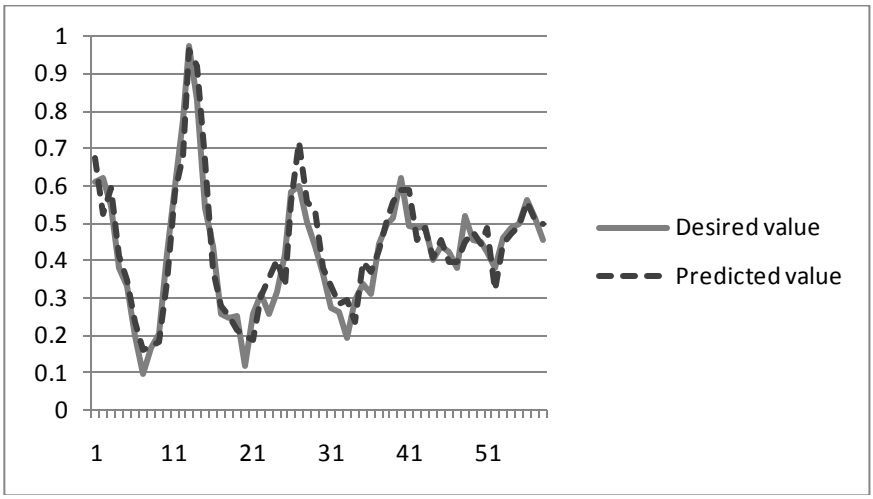


Figure4.14: Waves test results for one-step prediction

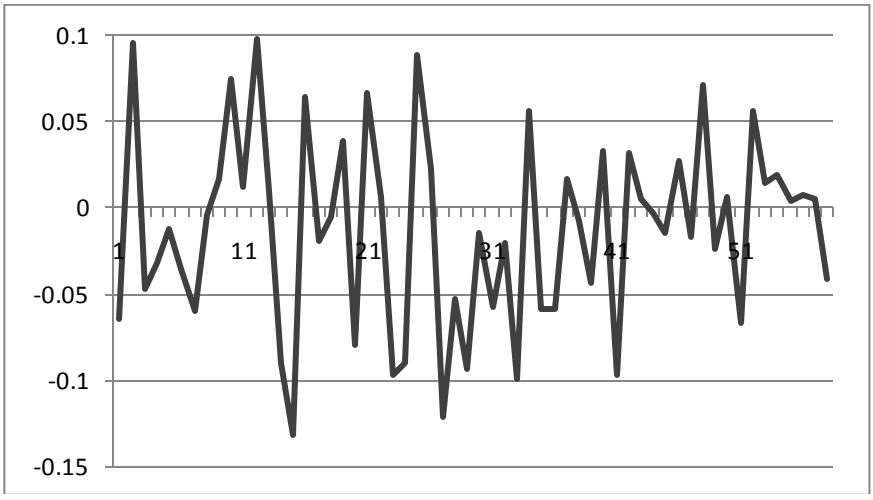


Figure4.15: Waves prediction error

This dataset has seen subject to limited use. There is only one article that reports on it, where the MSE error was not explicitly reported (the value in Table 4.19 was determined from observation of a graph, not a numeric value). However, ANCFIS does outperform this technique. Membership functions after training are reported in Table 4.20.

Method	MSE
ANCFIS	0.003215
[100]	~0.007

Table 4.19: Comparison of Testing Error for Waves Dataset

	a	b	c	d
CFS 1	9.680839	78.30599	0.025103	0.0001
CFS 2	0.000834	78.62769	0.568644	0.000369
CFS 3	5.99933	31.41722	0.338266	0.012879

Table 4.20: Membership Functions after Training for Waves dataset

In the five experimental contrasts, Off-line ANCFIS is comparable to published results on all five real-world datasets. It has not been pursued a test of statistical significance, as the training and testing sets vary widely between different investigations for any dataset. However, ANCFIS is a viable forecasting algorithm. It is also worth noting that the ANCFIS network achieves this performance with an extremely parsimonious network; no more than three rules are used for any dataset, including the two chaotic ones. In general, ANCFIS provides a very compact representation of a time series.

Chapter 5:

Online Learning for Univariate Case

There are many situations such as in real-time applications, where data becomes available gradually and is not available in a single batch. For example, in applications related to sensor networks, transaction log analysis and internet traffic measurements, data become available incrementally. This is particularly true in time series forecasting, where we often seek to create prediction models for ongoing phenomena. In these situations, it is important to use online learning algorithms [49][50][101], as they are a better fit to the learning problem. It is also cheaper to update the existing model instead of building a new model [102][103]. There are lots of applications which prefers online learning to batch learning as online learning does not need to do retraining when new data is available [104]-[107].

The original ANCFIS architecture [8], however, exclusively uses offline or batch learning. Thus, there is a need to create a variant of ANCFIS that employs online learning in time series forecasting.

We have developed online ANCFIS to meet this need. The online ANCFIS architecture is based on the original ANCFIS architecture, with substantial changes. To update the premise parameters in layer 1, we replace VNCSA with another derivative-free optimization algorithm. Also, recursive-least-square (RLS) [108]-[114] replaces the regular least square algorithm for updating consequence parameters in layer 5. We evaluate online ANCFIS on two of the datasets from Chapter 4. We find that, while batch learning does outperform online learning, the differences are small and the online ANCFIS still outperforms almost all published results on these datasets.

The remainder of this chapter is organized as follows. We first explain the Down-Hill simplex algorithm and RLS estimation in Section 5.1. The online ANCFIS design will be discussed in Section 5.2, and we evaluate the design on two

datasets in Section 5.3. We offer a discussion and summary of this chapter in Section 5.4.

5.1 Down-Hill Simplex Algorithm

Downhill simplex search [115][116][5] is a derivative-free method to optimize a multidimensional function. The concept behind this method is simpler than other methods such as simulated-annealing and it is faster but the disadvantage of this method is that it may find a local minimum instead of a global minimum depending on initial starting values for parameters with which it is initialized. So it may be important to run the procedure with different starting points in order to ascertain whether it will coverage the same, assumedly global, minimum.

This method uses the concept of simplex, which is a collection of *(n + 1)* vertices in *n* dimensions. For example, in a two-dimensional space, the simplex is a triangle. A function of *n* variables is minimized by repeatedly comparing its values at the *(n + 1)* vertices and replacing the vertex with the highest value by another point. The simplex under consideration changes directions and adapts itself to local landscape to find the neighborhood of the global minimum. Changes in the shape and direction of the simplex are due to a number of rules and operations, which are described next.

To generate the simplex, we begin with an initial start point P_0 and compute the remaining *n* points can be calculated using equation 5.1:

$$P_i = P_0 + \lambda_i e_i, i = 1,...,n \quad (5.1)$$

Here e_i's are unit vectors which span the *n*-dimensional space and λ_i is a constant which reflects the guess of length scale of the optimization problem in question. The function value at P_i is y_i .

$$l = \arg\min_i (y_i) \quad (l \text{ for "low"})$$

$$h = \arg\max_i (y_i) \quad (h \text{ for "high"}) \quad (5.2)$$

Here *l* and *h* are indices for minimum and maximum of y_i .

$$y_l = \min_i (y_i)$$
$$y_h = \max_i (y_i) \qquad (5.3)$$

The average of the (n+1) points is characterized by $\overline{P}$. Each cycle in Downhill Simplex method begins with P^*. The four operations which are used in this method, depend on the value at P^*; first one is reflection from P_h; then second one is reflection and expansion away from P_h; third is contraction on one dimension which connects P_h and $\overline{P}$; and the last one is shrinkage of P_l on all dimensions. Figure 5.1 shows these four operations for a two input function.

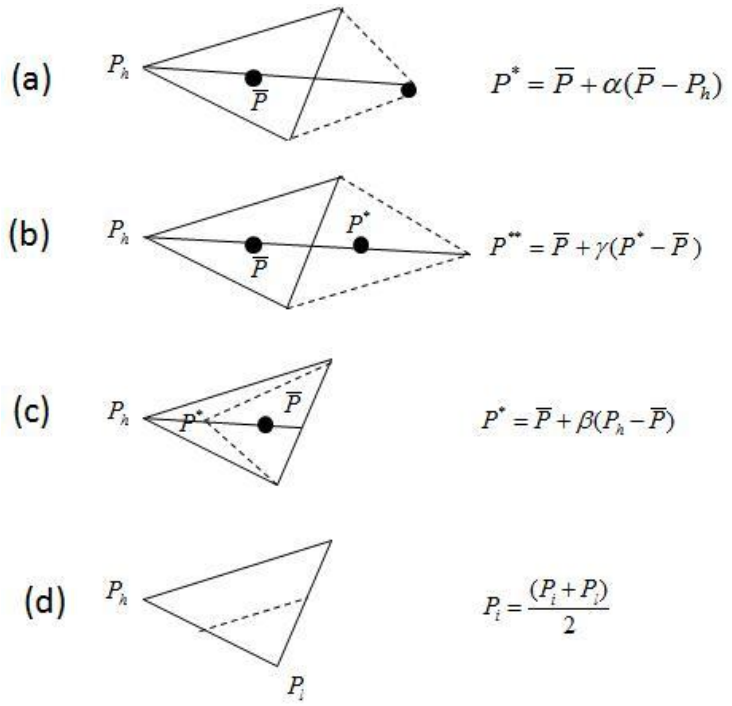


Figure 5.1: Outcomes for a cycle in the downhill simplex search after (a) reflection away from P_h; (b) reflection and expansion away from P_h; (c) contraction along one dimension connecting P_h and $\overline{P}$; (d) shrinkage toward P_l alone all dimensions[5]

1. Reflection away from P_h

   The equation of reflection is 5.3.

$$P^* = \overline{P} + \alpha(\overline{P} - P_h)$$
$$y^* = f(P^*) \qquad (5.3)$$

The reflection point is defined by P^* and its value is y^*. The reflection coefficient α is a positive constant. As shown in figure 5.1, P^* is on the line which connects P_h to $\overline{P}$. Based on the value of y^*, one of the four below steps will happen:

- If $y^* \le y_l$, then do expansion.
- If $y_l < y^* \le \max_{i,i\neq h\{y_i\}}$, then replace P_h with P^* and finish cycle.
- If $\max_{i,i\neq h\{y_i\}} < y^* \le y_h$, then replace P_h with P^* and go to contraction.
- If $y_h < y$, then go to contraction.

2. Expansion from P_h

   The equation of expansion is 5.4.

$$\begin{aligned} P^{**} &= \overline{P} + \gamma(P^* - \overline{P}) \\ y^{**} &= f(P^{**}) \end{aligned} \tag{5.4}$$

   The expansion point is defined by P^{**} and its value is y^{**}. The expansion coefficient γ is greater than unity. Depend on the value of y^*, we have:

- If $y^* \le y_l$, then replace P_h with P^{**} and finish cycle.
- If $y^* > y_l$, then replace P_h with original reflection point P^* and finish cycle.

3. Contraction along one dimension connecting P_h and $\overline{P}$

   The equation of contraction is 5.5.

$$\begin{aligned} P^{**} &= \overline{P} + \beta(P_h - \overline{P}) \\ y^{**} &= f(P^{**}) \end{aligned} \tag{5.5}$$

   The contraction point is defined by P^{**} and its value is y^{**}. The contraction coefficient β is between 0 and 1. Here also we have:

- If $y^* \le y_l$ or $y_l < y^* \le \max_{i,i\neq h\{y_i\}}$ or $\max_{i,i\neq h\{y_i\}} < y^* \le y_h$, then replace P_h with P^{**} and finish cycle. Otherwise go to shrinkage.

4. Shrinkage toward P_l along all dimensions.

   Here each P_i is replaced with $\frac{(P_i + P_l)}{2}$

Here it is important to find best values for constants α, β and γ. Finding these values depends on their applications and can be found by trial-and-error but a good starting point is (α, β, γ) = (1, 0.5, 2) which is suggested by Nedler and Mead's original paper [115][116].

5.1.1 Recursive Least-Squares Estimation

In the online ANCFIS, RLS (Recursive Least-Squares) [110]-[114] estimation is applied to update parameters in layer five. Here the effect of old data pairs should gradually decay as the new data pairs are presented. For this reason a weight is assigned to RLS method which put more importance on recent data. This weight parameter usually varies between 0.9 and 1. If this parameter becomes smaller, it removes the effect of old data faster. The recursive least squares method is based on Equation (5.6) and (5.7).

$$\theta_i = \theta_i + P_{i+1} a_{i+1} (y_{i+1}^T - a_{i+1}^T \theta_i) \tag{5.6}$$

$$P_{i+1} = \frac{1}{\lambda}[P_i - \frac{P_i a_{i+1} a_{i+1}^T P_i}{\lambda + a_{i+1}^T P_i a_{i+1}}] \tag{5.7}$$

5.2 Online ANCFIS Design

While the design of online ANCFIS is based on offline ANCFIS design [5], still there are major differences. Similar to offline ANCFIS in forward pass we have to determine a complex-valued membership given a segment of time series and a CFS membership function; thus in backward pass, we used another derivative-free optimization technique which is Downhill-simplex instead of VNCSA to determine the CFS parameters $\{a, b, c, d\}$. These changes are done due to the fact

that VNCSA does a global optimization, which would overfit a single training pattern. In this work only a single iteration of the downhill-simplex method is done (instead of running it until convergence on a single pattern), in order to avoid over fitting. Also consequence parameters are found using RLS. Although the suggested range of values for lambda in RLS is between 0.9 and 1 [5], we tried to apply smaller values for Lambda to investigate the performance of the system as the effect of old data decays faster.

To start the downhill simplex search, in each epoch for the first training vector we must initialize the simplex of (4+1=5) points in five dimensional space. First, we need the initial starting point P_0. Parameters for P_0 are the same parameters chosen in forward pass for that related node. Then the rest of four points can be calculated using equation 5.1. Here, Lambda is chosen by a random number generator. After each training vector presentation, it is needed to find the premise values of $\{a,b,c,d\}$. First, we should calculate the magnitude and phase of each weight $SMF_{ijk_{new}}$ by Equation 3.7 and 3.8, and then there is a set of *m* magnitude-phase for each complex membership function. In Equation 5.8, we try to optimize a model by minimizing a squared error measure between magnitudes of the updated weights and the fitting value of complex membership function at given phase related to magnitudes of the updated weights.

$$E(x_1,x_2,x_3,x_4)=\sum_{m=1}^{n^2}(magniture_m-[x_4*\sin(x_1*phase_m+x_2)+x_3])^2 \qquad (5.8)$$

Where x_1,x_2,x_3,x_4 are the premise parameters $\{a,b,c,d\}$, respectively, *n* is the length of input vector, $magniture_m$ and $phase_m$ are magnitude-phase data pair of the updated weight $SMF_{ijk_{new}}$. After presentation of each training vector, the values found for each simplex point after the single iteration of the downhill-simplex algorithm are kept to be used as the initial values of the simplex for the next training vector. This continues until all training vectors are covered. Then, the best founded minimum value is used with its parameters of a,b,c,d as the final value for that training epoch. Then, the cycle is repeated.

5.3 Experimental Results

In this section, the results of forecasting experiments using online learning are reported. Two time series dataset is used; Sunspot and Waves. The experimental design for each dataset is the same as in Chapter 4; a single-split, one-step-ahead prediction design where all training data are chronologically earlier than the holdout test sample. The size of the input window is selected to approximately cover one "period" in the time series. The performance of online ANCFIS is compared to current results for each dataset in the literature and also to the results for offline ANCFIS. We again compute several different measures of forecast errors for comparison with the literature; the definitions of these measures may be found in Chapter 4. Different Lambda values are explored for each dataset. Lambda values decrease in intervals of 0.1 from 1 to the point that training errors become really huge and do not exceed any more. Also, other stopping criteria are defined for our work. One of these stopping points is defined as difference value between training errors in two sequential epochs. If this value exceeds the minimum value we defined, the program stops. The other stopping point is minimum error value defined by the user. If the training error becomes smaller than this value, again the program stops. Even if none of these criteria are met, the program stops when it reaches the maximum epoch number.

5.3.1 Sunspot Results

As Sunspot is a very popular time series dataset [17]-[21] and [15][16][94][95], we used it to compare the results found in literature with our online experimental results for ANCFIS. More detail about this dataset is indicated before in chapter 4. All training parameters applied to online ANCFIS, are presented in table 5.1. As it is shown, the values related to alpha, beta and gamma are the same values as what Nelder and Mead suggested in their paper [101].

Variates	Input Length	Outputs	CFS per Input	Step Size	Increase Rate	Decrease Rate	Alpha	Beta	Gamma
1	12	1	3	0.001	1.1	0.8	1	0.5	2

Table 5.1: Training Parameters for sunspot dataset used in online-ANCFIS

In the comparison between online ANCFIS results and results which are found in literature, ANCFIS is often superior, except against one method published in 2009 (last four rows). Though, the four methods in [21] yielded a lower MSE, the ANCFIS architecture is superior when the same results are measured by NDEI. This may indicate that the four methods in [21] and ANCFIS don't have much difference on this dataset. Moreover, the results for online ANCFIS is very close to offline ANCFIS which represents the good performance of this system.

Method	**ARV**	**NMSE**	**MSE**	**NDEI**
ANCFIS (offline)	0.068	0.1302	8.29×10^{-3}	0.3608
ANCFIS (online)	0.069	0.1473	9.2×10^{-3}	0.38
ARMA [97]	0.252	-	-	-
Elman [97]	0.348	-	-	-
Extended Elman [97]	0.162	-	-	-
FIR [97]	0.115	-	-	-
SVR [17]	-	0.64	-	-
Bagging SVR [17]	-	0.58	-	-
Boosting SVR (median) [17]	-	0.27	-	-
Boosting SVR (mean) [17]	-	0.33	-	-
SVM [20]	-	0.178	-	-
SVM Ensemble [20]	-	0.1541	-	-
[98]	-	0.28	-	-
[99]	-	0.35	-	-
TSK-NFIS [21]	-	-	1.32×10^{-3}	0.38
AR [21]	-	-	1.36×10^{-3}	0.385
NAR [21]	-	-	2.68×10^{-3}	0.541
Neural Network [21]	-	-	2.17×10^{-3}	0.486

	ANCFIS (offline)	**ANCFIS (online)**	**[97]**	**[60]**	
Absolute Error	0.068	0.069	13.83	13.73	

Table 5.2: Comparison of Testing Error for Sunspot Dataset

Further analyses of the online ANCFIS results are presented in Figure 5.2 and 5.3. Figure 5.2 shows the actual versus predicted outputs and Figure 5.3 presents the prediction errors.

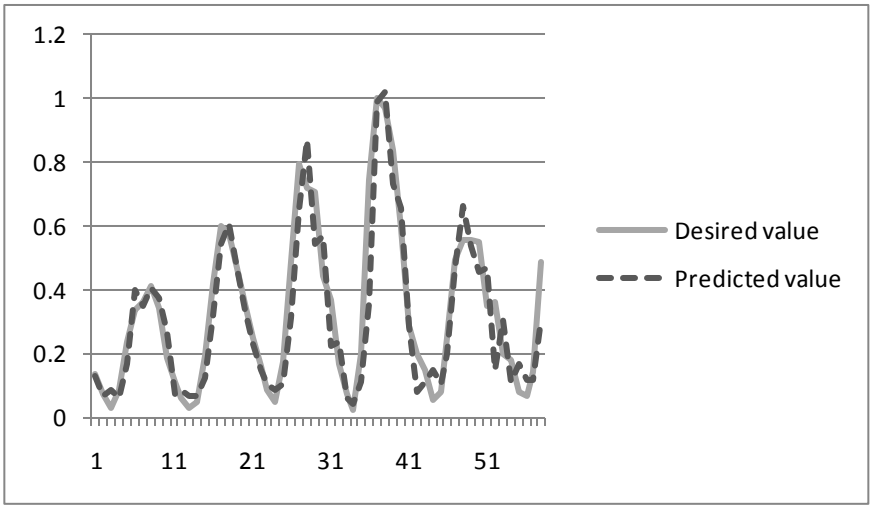


Figure 5.2: Sunspot test results for online prediction

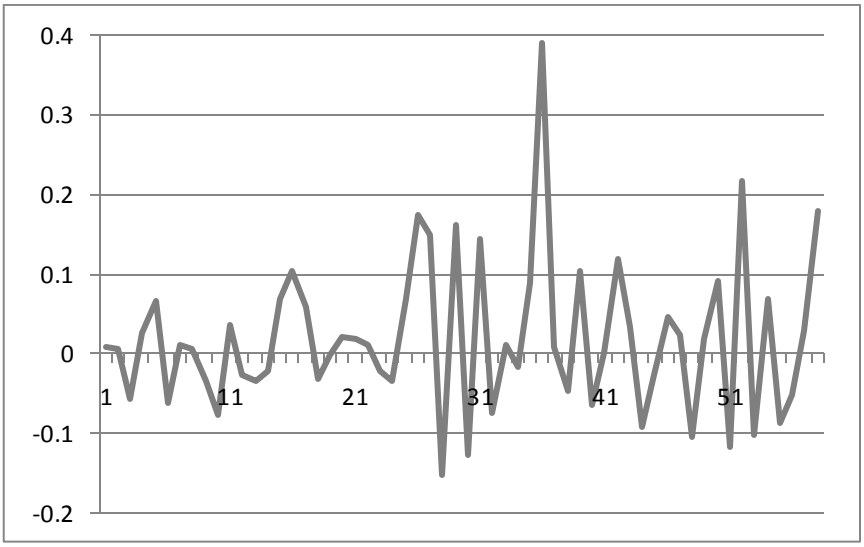


Figure 5.3: Sunspot test errors for online prediction

The best testing error is found with considering Lambda=1; the NMSE error measured is 0.1473. As the values of Lambda decrease, test errors increase. Figure 5.4 presents the different error values for different Lambda from 1 to 0.5 and also Figure 5.5 shows these test error values from 1 to 0.3 where the error

becomes really huge. Regarding these two figures, it is clear that after decreasing Lambda below 0.9 the test error increases faster.

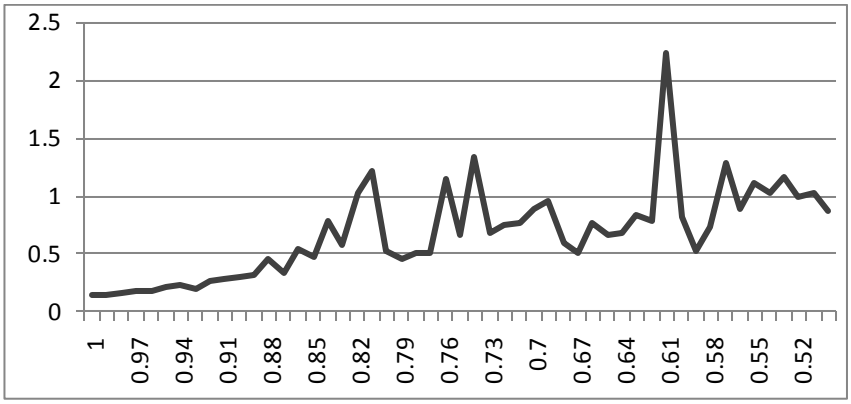


Figure 5.4: Testing NMSE errors for different lambda from 1 to 0.5

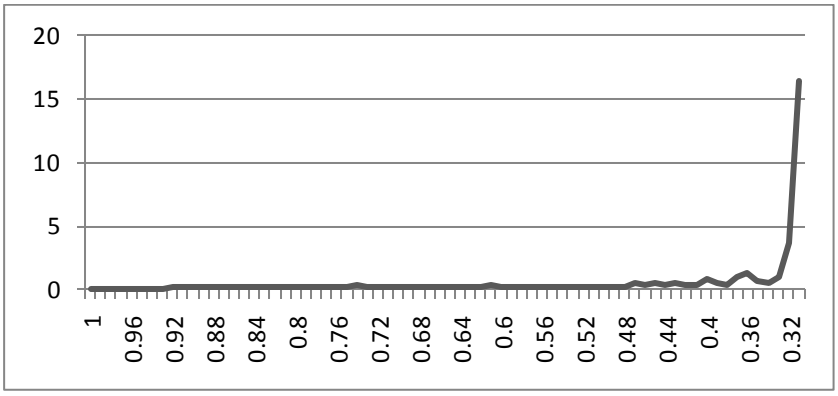


Figure 5.5: Testing NMSE errors for different lambda from 1 to 0.3

5.3.2 Waves Results

The Waves dataset [23] is another time series dataset which is used for online ANCFIS experiments. The experimental results for online ANCFIS is compared to offline ANCFIS and the results found in literature This dataset is explained in details in chapter four. Table 5.3 represents all the parameters applied to online ANCFIS to find best results for Waves dataset. Again, the parameter values of the

three coefficients α, β, γ are the same suggested values by Nelder and Mead [101].

Variates	Input Length	Outputs	CFS per Input	Step Size	Increase Rate	Decrease Rate	Alpha	Beta	Gamma
1	12	1	3	0.001	1.1	0.8	1	0.5	2

Table 5.3: Training Parameters for Waves dataset used in online-ANCFIS

Table 5.4 presents best results found for different error measurements for offline and online learning methods. As it is shown, the results are quite similar.

	MSE	**NDEI**	**NMSE**
Offline method	0.0032	0.3136	0.0983
Online method	0.0034	0.3330	0.1109

Table 5.4: Different error measurements for Waves

This dataset is not as popular as Sunspot dataset. We found only one article which applies this dataset, though the MSE error was not mentioned numerically and it was determined from the observation of a graph. Table 5.5 presents the forecasting results; both online and offline ANCFIS results are superior to the results in literature.

Method	**MSE**
ANCFIS (offline learning)	0.003215
ANCFIS (online learning)	0.003492
[66]	~0.007

Table 5.5: Comparison of Testing Error for Waves Dataset

More analysis is provided in Figure 5.6 and 5.7; which 5.8 again presents actual versus predicted values and 5.9 presents prediction errors.

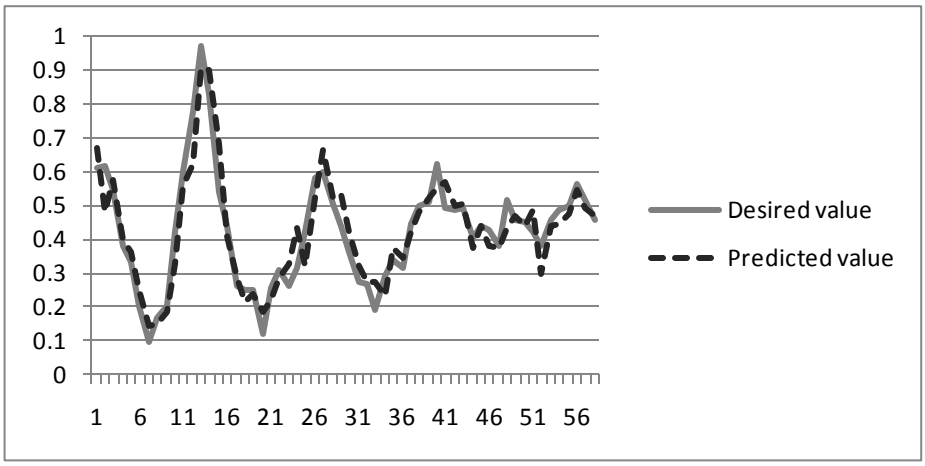


Figure 5.6: Waves test results for online prediction

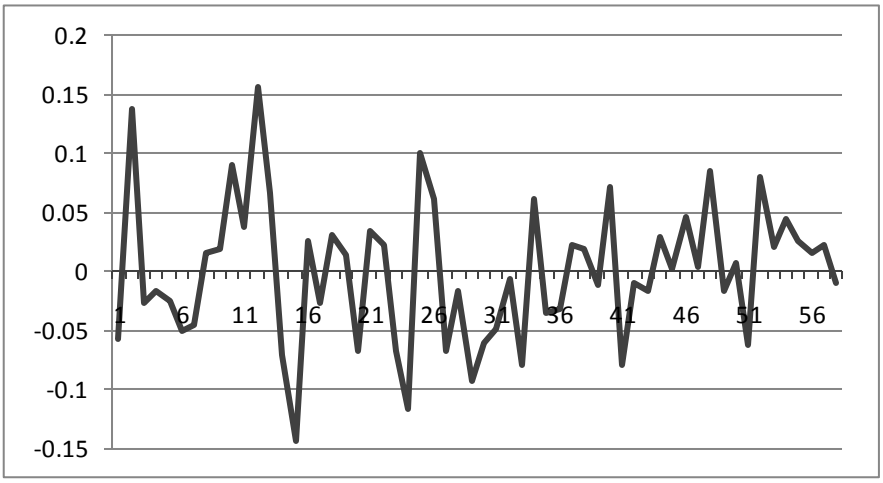


Figure 5.7: Waves test errors for online prediction

The best forecasting error for this time series dataset in online ANCFIS is found for Lambda=0.99. Figure 5.6 and 5.7 represent the best forecasting results of the online system with different Lambda; as the Lambda reduces our results become worse till we reach to the point (Lambda=0.35) that the error becomes really huge.

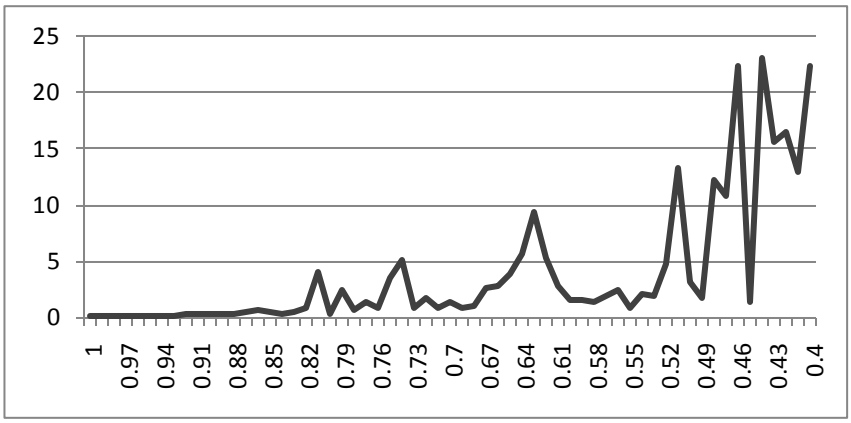


Figure 5.6: Testing NMSE errors for different lambda from 1 to 0.4

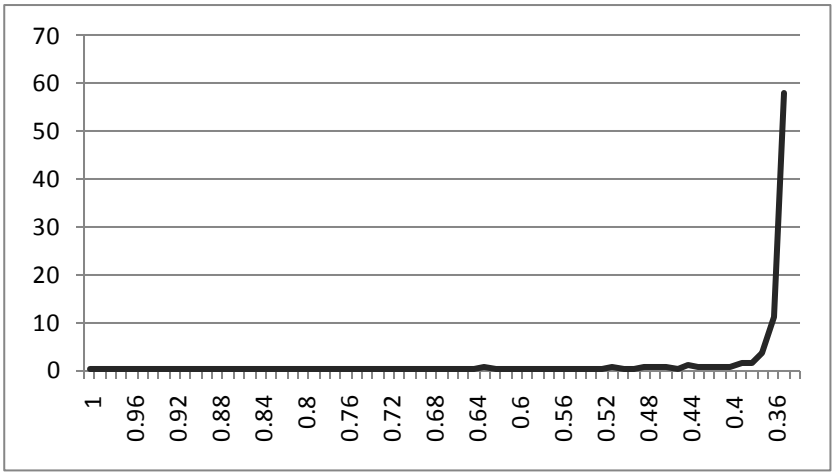


Figure 5.7: Testing errors for different lambda from 1 to 0.35

5.4 Discussion and Conclusions

In this chapter, we have presented and evaluated the design of an online learning algorithm for ANCFIS. The key design decisions were to replace VNCSA with the downhill simplex algorithm (iterated only once per pattern), and to replace least-squares estimation with recursive least-squares in learning the layer 5 consequent parameters. Although ANCFIS in batch mode does give us better forecasting performance, online ANCFIS still outperforms almost all other methods on these two datasets. Again, this is accomplished with three or fewer rules in each dataset. We also found that this performance is reasonably robust

against changes in the λ parameter, which controls the contribution of older patterns to the recursive least squares procedure. Our results show that complex fuzzy logic systems can also be designed using online learning; this is the first time that this has been demonstrated.

Chapter 6: Multivariate ANCFIS

Multivariate forecasting consists of two or more observations recorded sequentially over equal time increments. Usually, due to theory or physical property, these observations should relate to each other in some fashion. This relationship should in turn allow us to improve upon the forecasting of these time series; the possibility of dynamic interactions among them is important [117][49]. Previously, the ANCFIS architecture was developed only for one input vector; that is called univariate forecasting. However, a multivariate fuzzy inference system is needed for the case where there are multiple input vectors.

In order to implement multivariate fuzzy inference, a multivariate ANCFIS architecture is developed. The main difference between univariate and multivariate ANCFIS system is related to the layer two where the algebraic product (a complex fuzzy conjunction) is applied. This provision was made in the original ANCFIS architecture, but was never fully implemented. Also, an extension is needed in layer five to accommodate multivariate outputs. Multivariate ANCFIS is applied to the domain of time-series forecasting, an important machine learning problem. Four different time-series datasets are used: Transport and tourism-motel [22], Hydrology-river flow [22][24], Macro-economic [22] and Car-road-accident [25]-[27]. We compare these results against the forecasts obtained by the univariate ANCFIS on each individual time series. However, our performance evaluation is disappointing; in our experiments the univariate version of ANCFIS is able to model the individual variates better than the multivariate version can, even though the variates in each dataset are highly correlated with each other. We provide extensive detail on our multivariate design and experimental results, as a starting point for future work on this topic.

The remainder of this chapter is organized as follows. We first discuss the multivariate ANCFIS design in section 6.1. We then present our experimental

results in section 6.2. We conclude the chapter with a summary and discussion in section 6.3.

6.1 Multivariate ANCFIS Design

Multivariate ANCFIS is an adaptive network with six layers. The architecture of multivariate ANCFIS is similar to the original univariate ANCFIS with some modifications. The difference in their architecture is related to layer two and layer five. Layer two contains fixed nodes and is responsible for multiplication of incoming signals and transferring them to the next layer. This layer output is the firing strength of a fuzzy rule. As with ANFIS, the operations for this layer should be conjunctions; here the algebraic product is applied, as it was proposed as a complex fuzzy conjunction in [1][5][10]. The function related to this layer is as below:

$$f_{2,m} = \prod_{i=1}^{n} (x_{1,i} + iy_{1,i}) \tag{6.1}$$

Where m refers to nodes in layer two and n refers to total number of nodes in layer one related to node m in layer two. When there are two input vectors, there are two different nodes in layer one connected to each node in layer two. In this case, the Equation 6.1 can be expressed as below:

$$f_{l,m} = \prod_{i=1}^{2} (x_{1,i} + iy_{1,i}) = (x_{1,1} + iy_{1,1})(x_{1,2} + iy_{1,2}) =$$
$$(x_{1,1}x_{1,2} - y_{1,1}y_{1,2}) + i(x_{1,1}y_{1,2} + x_{1,2}y_{1,1}) \tag{6.2}$$

Layer five includes adaptive nodes. The number of nodes in this layer is related to the number of multivariate outputs. Each node in this layer is calculated by:

$$O_{5,i} = w_m^{DP} f_i = w_m^{DP} [\sum_{k=1}^{In\_n} (\vec{p}_i \vec{x}_k) + r_i]$$
$$= w_m^{DP} [\sum_{k=1}^{In\_n} (p_{i,1}x_{k,1} + p_{i,2}x_{k,2} + ... + p_{i,j}x_{k,j} + ... + p_{i,n}x_{k,n}) + r_i], \tag{6.3}$$
$$, j = 1,2,...,n, m = (i - (i\%In\_n))$$

where w_m^{DP} is the *mth* output of fourth layer, $x_{k,j}$ is the *jth* value of *kth* input vector, *n* is the length of input vector, *In_n* is the total numbers of input vector, $\{\vec{p}_i, r_i\}$ is the parameter set for linear output function, $\vec{p}_i$ is a vector of the same length as input vector $\vec{x}$ and r_i is a constant. Figure 6.1 presents a multivariate ANCFIS for two input vectors (two variates).

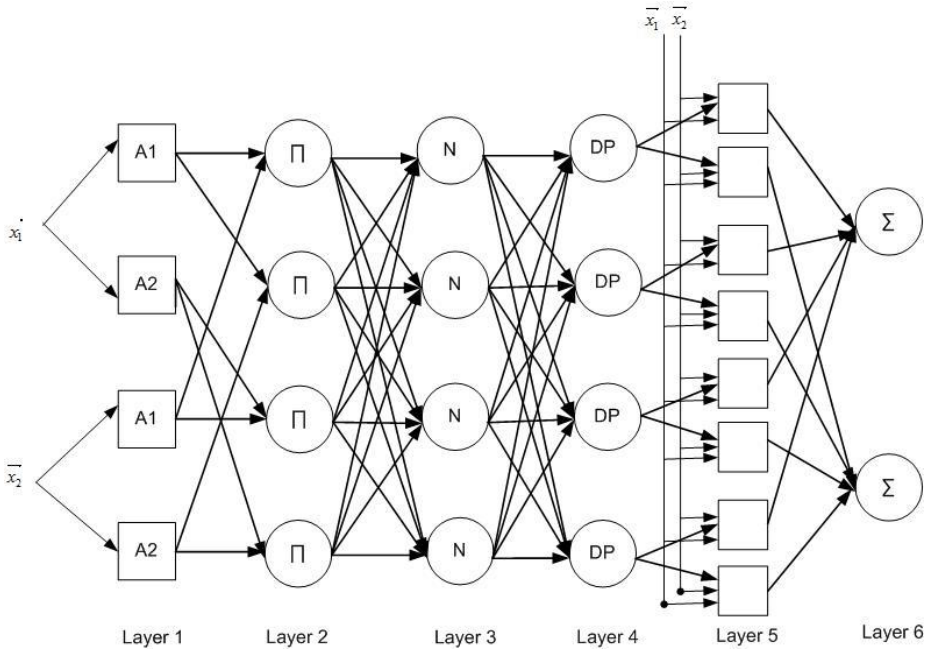


Figure 6.1: Multivariate ANCFIS network

As discussed before, for each internal node of ANCFIS network the error signal was calculated by Equation 3.23. Regarding that we use algebraic product for layer two, the error signals for layer two is calculated by:

$$\varepsilon_{l,i} = \frac{\partial^+ E_P}{\partial x_{l,i}} = \sum_{i=1}^{N(m+1)} \frac{\partial^+ E_P}{\partial x_{l+1,m}} \frac{\partial f_{l+1,m}}{\partial x_{l,i}} = \sum_{i=1}^{N(m+1)} \varepsilon_{l+1,m} \frac{\partial f_{l+1,m}}{\partial x_{l,i}} =$$

$$\varepsilon_{l+1,1} \frac{\partial f_{l+1,1}}{\partial x_{l,i}} + \varepsilon_{l+1,2} \frac{\partial f_{l+1,2}}{\partial x_{l,i}} + .. + \varepsilon_{l+1,m} \frac{\partial f_{l+1,m}}{\partial x_{l,i}} =$$

$$\varepsilon_{l+1,1} \frac{\partial((x_{l,i} + iy_{l,i})(x_{l,j} + iy_{l,j})...((x_{l,k} + iy_{l,k}))}{\partial x_{l,i}} \tag{6.4}$$

$$+ ... + \varepsilon_{l+1,m} \frac{\partial((x_{l,i} + iy_{l,i})(x_{l,j'} + iy_{l,j'})...((x_{l,k'} + iy_{l,k'}))}{\partial x_{l,i}} =$$

$$\varepsilon_{l+1,1}((x_{l,j} + iy_{l,j})...((x_{l,k} + iy_{l,k})) + ... + \varepsilon_{l+1,m}\partial((x_{l,j'} + iy_{l,j'})...((x_{l,k'} + iy_{l,k'}))$$

Where the index i refers to node position in layer l and error signals and m refers to nodes in layer $l+1$. Equation 6.4 shows that algebraic product used for complex fuzzy numbers, which acts the same way as it is used for real numbers. Next, the back propagation of two inputs multivariate ANCFIS is described.

6.1.1 Multivariate ANCFIS back propagation example

The back propagation equations can be computed through equations 3.20 to 3.31, the same as original ANCFIS. Here errors in different layers are calculated from last layer to first layer.

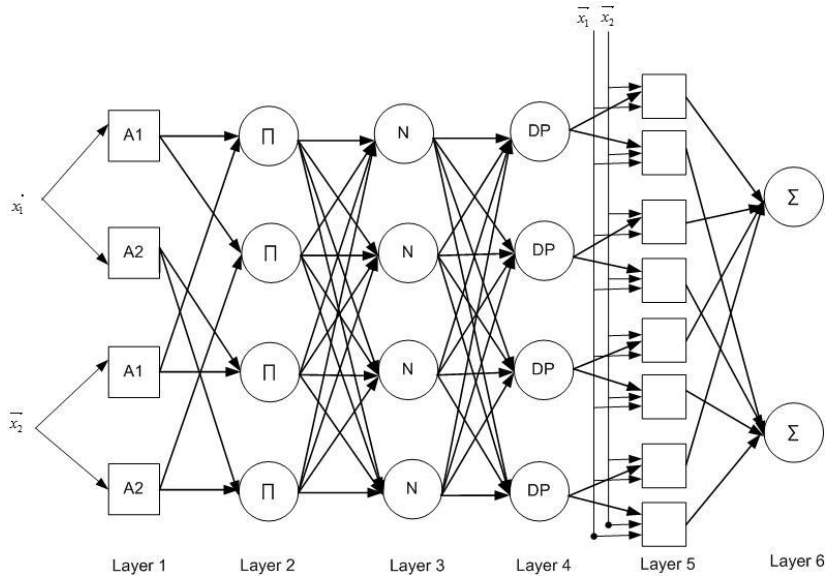


Figure 6.2: Forward pass of a Multivariate ANCFIS structure with two inputs and two complex membership functions for each input

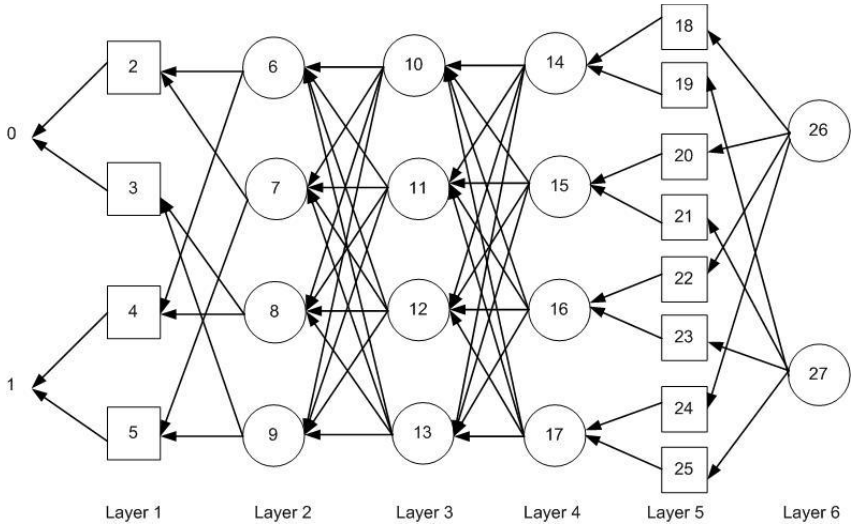


Figure 6.3: Backward pass of a Multivariate ANCFIS structure with two inputs and two complex membership functions for each input

Figure 6.2 and 6.3 present the structure of multivariate ANCFIS for two input vectors and in total four membership functions (two MF for each input). In below this ANCFIS example is described.

From the output to *layer 6:*

$$\varepsilon_{27} = \frac{\partial E_P}{\partial x_{27}} = -2(d_{27} - x_{27}) \tag{6.6}$$

$$\varepsilon_{26} = \frac{\partial E_P}{\partial x_{26}} = -2(d_{26} - x_{26}) \tag{6.7}$$

From *layer 6* to *layer 5*:

$$\varepsilon_{25} = \frac{\partial E_P}{\partial x_{25}} = \frac{\partial E_P}{\partial x_{27}} \frac{\partial f_{27}}{\partial x_{25}} = \varepsilon_{27} \frac{\partial f_{27}}{\partial x_{25}} = \varepsilon_{27} \frac{\partial (\sum w_i^{DP} f_i)}{\partial (\overline{w_4} f_4)} = \varepsilon_{27} \tag{6.8}$$

$$\varepsilon_{24} = \frac{\partial E_P}{\partial x_{24}} = \frac{\partial E_P}{\partial x_{26}} \frac{\partial f_{26}}{\partial x_{24}} = \varepsilon_{26} \frac{\partial f_{26}}{\partial x_{24}} = \varepsilon_{26} \frac{\partial (\sum w_i^{DP} f_i)}{\partial (\overline{w_4} f_4)} = \varepsilon_{26} \tag{6.9}$$

$$\varepsilon_{23} = \frac{\partial E_P}{\partial x_{23}} = \frac{\partial E_P}{\partial x_{27}} \frac{\partial f_{27}}{\partial x_{23}} = \varepsilon_{27} \frac{\partial f_{27}}{\partial x_{23}} = \varepsilon_{27} \frac{\partial (\sum w_i^{DP} f_i)}{\partial (\overline{w_3} f_3)} = \varepsilon_{27} \tag{6.10}$$

$$\varepsilon_{22} = \frac{\partial E_P}{\partial x_{22}} = \frac{\partial E_P}{\partial x_{26}} \frac{\partial f_{26}}{\partial x_{22}} = \varepsilon_{26} \frac{\partial f_{26}}{\partial x_{22}} = \varepsilon_{26} \frac{\partial (\sum w_i^{DP} f_i)}{\partial (\overline{w_3} f_3)} = \varepsilon_{26} \tag{6.11}$$

$$\varepsilon_{21} = \frac{\partial E_P}{\partial x_{21}} = \frac{\partial E_P}{\partial x_{27}} \frac{\partial f_{27}}{\partial x_{21}} = \varepsilon_{27} \frac{\partial f_{27}}{\partial x_{21}} = \varepsilon_{27} \frac{\partial (\sum w_i^{DP} f_i)}{\partial (\overline{w_2} f_2)} = \varepsilon_{27} \tag{6.12}$$

$$\varepsilon_{20} = \frac{\partial E_P}{\partial x_{20}} = \frac{\partial E_P}{\partial x_{26}} \frac{\partial f_{26}}{\partial x_{20}} = \varepsilon_{26} \frac{\partial f_{26}}{\partial x_{20}} = \varepsilon_{26} \frac{\partial (\sum w_i^{DP} f_i)}{\partial (\overline{w_2} f_2)} = \varepsilon_{26} \tag{6.13}$$

$$\varepsilon_{19} = \frac{\partial E_P}{\partial x_{19}} = \frac{\partial E_P}{\partial x_{27}} \frac{\partial f_{27}}{\partial x_{19}} = \varepsilon_{27} \frac{\partial f_{27}}{\partial x_{19}} = \varepsilon_{27} \frac{\partial (\sum w_i^{DP} f_i)}{\partial (\overline{w_1} f_1)} = \varepsilon_{27} \tag{6.14}$$

$$\varepsilon_{18} = \frac{\partial E_P}{\partial x_{18}} = \frac{\partial E_P}{\partial x_{26}} \frac{\partial f_{26}}{\partial x_{18}} = \varepsilon_{26} \frac{\partial f_{26}}{\partial x_{18}} = \varepsilon_{26} \frac{\partial (\sum w_i^{DP} f_i)}{\partial (\overline{w_1} f_1)} = \varepsilon_{26} \tag{6.15}$$

From *layer 5* to *layer 4*:

$$\varepsilon_{17} = \frac{\partial E_P}{\partial x_{17}} = \frac{\partial E_P}{\partial x_{25}}\frac{\partial f_{25}}{\partial x_{17}} + \frac{\partial E_P}{\partial x_{24}}\frac{\partial f_{24}}{\partial x_{17}} = \varepsilon_{25}\frac{\partial f_{25}}{\partial x_{17}} + \varepsilon_{24}\frac{\partial f_{24}}{\partial x_{17}} =$$
$$\varepsilon_{25}\frac{\partial(w_4^{DP} f_8)}{\partial(w_4^{DP})} + \varepsilon_{24}\frac{\partial(w_4^{DP} f_7)}{\partial(w_4^{DP})} = \varepsilon_{25} f_8 + \varepsilon_{24} f_7 \tag{6.16}$$

$$\varepsilon_{16} = \frac{\partial E_P}{\partial x_{16}} = \frac{\partial E_P}{\partial x_{23}}\frac{\partial f_{23}}{\partial x_{16}} + \frac{\partial E_P}{\partial x_{22}}\frac{\partial f_{22}}{\partial x_{16}} = \varepsilon_{23}\frac{\partial f_{23}}{\partial x_{16}} + \varepsilon_{22}\frac{\partial f_{22}}{\partial x_{16}} =$$
$$\varepsilon_{23}\frac{\partial(w_3^{DP} f_6)}{\partial(w_3^{DP})} + \varepsilon_{22}\frac{\partial(w_3^{DP} f_5)}{\partial(w_3^{DP})} = \varepsilon_{23} f_6 + \varepsilon_{22} f_5 \tag{6.17}$$

$$\varepsilon_{15} = \frac{\partial E_P}{\partial x_{15}} = \frac{\partial E_P}{\partial x_{21}}\frac{\partial f_{21}}{\partial x_{15}} + \frac{\partial E_P}{\partial x_{20}}\frac{\partial f_{20}}{\partial x_{15}} = \varepsilon_{21}\frac{\partial f_{21}}{\partial x_{15}} + \varepsilon_{20}\frac{\partial f_{20}}{\partial x_{15}} =$$
$$\varepsilon_{21}\frac{\partial(w_2^{DP} f_4)}{\partial(w_2^{DP})} + \varepsilon_{20}\frac{\partial(w_2^{DP} f_3)}{\partial(w_2^{DP})} = \varepsilon_{21} f_4 + \varepsilon_{20} f_3 \tag{6.18}$$

$$\varepsilon_{14} = \frac{\partial E_P}{\partial x_{14}} = \frac{\partial E_P}{\partial x_{19}}\frac{\partial f_{19}}{\partial x_{14}} + \frac{\partial E_P}{\partial x_{18}}\frac{\partial f_{18}}{\partial x_{14}} = \varepsilon_{19}\frac{\partial f_{19}}{\partial x_{14}} + \varepsilon_{18}\frac{\partial f_{18}}{\partial x_{14}} =$$
$$\varepsilon_{19}\frac{\partial(w_1^{DP} f_2)}{\partial(w_1^{DP})} + \varepsilon_{18}\frac{\partial(w_1^{DP} f_1)}{\partial(w_1^{DP})} = \varepsilon_{19} f_2 + \varepsilon_{18} f_1 \tag{6.19}$$

From *layer 4* to *layer 3*:

$$\varepsilon_{13} = \frac{\partial E_P}{\partial x_{13}} = \frac{\partial E_P}{\partial x_{17}}\frac{\partial f_{17}}{\partial x_{13}} + \frac{\partial E_P}{\partial x_{16}}\frac{\partial f_{16}}{\partial x_{13}} + \frac{\partial E_P}{\partial x_{15}}\frac{\partial f_{15}}{\partial x_{13}} + \frac{\partial E_P}{\partial x_{14}}\frac{\partial f_{14}}{\partial x_{13}} =$$
$$\varepsilon_{17}\frac{\partial f_{17}}{\partial x_{13}} + \varepsilon_{16}\frac{\partial f_{16}}{\partial x_{13}} + \varepsilon_{15}\frac{\partial f_{15}}{\partial x_{13}} + \varepsilon_{14}\frac{\partial f_{14}}{\partial x_{13}} =$$
$$\varepsilon_{17}\frac{\partial(w_4^{DP})}{\partial(\overline{w_4})} + \varepsilon_{16}\frac{\partial(w_3^{DP})}{\partial(\overline{w_4})} + \varepsilon_{15}\frac{\partial(w_2^{DP})}{\partial(\overline{w_4})} + \varepsilon_{14}\frac{\partial(w_1^{DP})}{\partial(\overline{w_4})} =$$
$$\varepsilon_{17}\frac{\partial[(\overline{w}_4 \bullet (\overline{w_1} + \overline{w_2} + \overline{w_3} + \overline{w_4})]}{\partial(\overline{w_4})} + \varepsilon_{16}\frac{\partial[(\overline{w}_3 \bullet (\overline{w_1} + \overline{w_2} + \overline{w_3} + \overline{w_4})]}{\partial(\overline{w_4})} +$$
$$\varepsilon_{15}\frac{\partial[(\overline{w}_2 \bullet (\overline{w_1} + \overline{w_2} + \overline{w_3} + \overline{w_4})]}{\partial(\overline{w_4})} + \varepsilon_{14}\frac{\partial[(\overline{w}_1 \bullet (\overline{w_1} + \overline{w_2} + \overline{w_3} + \overline{w_4})]}{\partial(\overline{w_4})} \tag{6.20}$$

$$\varepsilon_{12} = \frac{\partial E_P}{\partial x_{12}} = \frac{\partial E_P}{\partial x_{17}}\frac{\partial f_{17}}{\partial x_{12}} + \frac{\partial E_P}{\partial x_{16}}\frac{\partial f_{16}}{\partial x_{12}} + \frac{\partial E_P}{\partial x_{15}}\frac{\partial f_{15}}{\partial x_{12}} + \frac{\partial E_P}{\partial x_{14}}\frac{\partial f_{14}}{\partial x_{12}} =$$

$$\varepsilon_{17}\frac{\partial f_{17}}{\partial x_{12}} + \varepsilon_{16}\frac{\partial f_{16}}{\partial x_{12}} + \varepsilon_{15}\frac{\partial f_{15}}{\partial x_{12}} + \varepsilon_{14}\frac{\partial f_{14}}{\partial x_{12}} =$$

$$\varepsilon_{17}\frac{\partial(w_4^{DP})}{\partial(\overline{w_3})} + \varepsilon_{16}\frac{\partial(w_3^{DP})}{\partial(\overline{w_3})} + \varepsilon_{15}\frac{\partial(w_2^{DP})}{\partial(\overline{w_3})} + \varepsilon_{14}\frac{\partial(w_1^{DP})}{\partial(\overline{w_3})} =$$

$$\varepsilon_{17}\frac{\partial[(\overline{w}_4 \bullet (\overline{w_1} + \overline{w_2} + \overline{w_3} + \overline{w_4})]}{\partial(\overline{w_3})} + \varepsilon_{16}\frac{\partial[(\overline{w}_3 \bullet (\overline{w_1} + \overline{w_2} + \overline{w_3} + \overline{w_4})]}{\partial(\overline{w_3})} +$$

$$\varepsilon_{15}\frac{\partial[(\overline{w}_2 \bullet (\overline{w_1} + \overline{w_2} + \overline{w_3} + \overline{w_4})]}{\partial(\overline{w_3})} + \varepsilon_{14}\frac{\partial[(\overline{w}_1 \bullet (\overline{w_1} + \overline{w_2} + \overline{w_3} + \overline{w_4})]}{\partial(\overline{w_3})} \qquad (6.21)$$

$$\varepsilon_{11} = \frac{\partial E_P}{\partial x_{11}} = \frac{\partial E_P}{\partial x_{17}}\frac{\partial f_{17}}{\partial x_{11}} + \frac{\partial E_P}{\partial x_{16}}\frac{\partial f_{16}}{\partial x_{11}} + \frac{\partial E_P}{\partial x_{15}}\frac{\partial f_{15}}{\partial x_{11}} + \frac{\partial E_P}{\partial x_{14}}\frac{\partial f_{14}}{\partial x_{11}} =$$

$$\varepsilon_{17}\frac{\partial f_{17}}{\partial x_{11}} + \varepsilon_{16}\frac{\partial f_{16}}{\partial x_{11}} + \varepsilon_{15}\frac{\partial f_{15}}{\partial x_{11}} + \varepsilon_{14}\frac{\partial f_{14}}{\partial x_{11}} =$$

$$\varepsilon_{17}\frac{\partial(w_4^{DP})}{\partial(\overline{w_2})} + \varepsilon_{16}\frac{\partial(w_3^{DP})}{\partial(\overline{w_2})} + \varepsilon_{15}\frac{\partial(w_2^{DP})}{\partial(\overline{w_2})} + \varepsilon_{14}\frac{\partial(w_1^{DP})}{\partial(\overline{w_2})} =$$

$$\varepsilon_{17}\frac{\partial[(\overline{w}_4 \bullet (\overline{w_1} + \overline{w_2} + \overline{w_3} + \overline{w_4})]}{\partial(\overline{w_2})} + \varepsilon_{16}\frac{\partial[(\overline{w}_3 \bullet (\overline{w_1} + \overline{w_2} + \overline{w_3} + \overline{w_4})]}{\partial(\overline{w_2})} +$$

$$\varepsilon_{15}\frac{\partial[(\overline{w}_2 \bullet (\overline{w_1} + \overline{w_2} + \overline{w_3} + \overline{w_4})]}{\partial(\overline{w_2})} + \varepsilon_{14}\frac{\partial[(\overline{w}_1 \bullet (\overline{w_1} + \overline{w_2} + \overline{w_3} + \overline{w_4})]}{\partial(\overline{w_2})} \qquad (6.22)$$

$$\varepsilon_{10} = \frac{\partial E_P}{\partial x_{10}} = \frac{\partial E_P}{\partial x_{17}}\frac{\partial f_{17}}{\partial x_{10}} + \frac{\partial E_P}{\partial x_{16}}\frac{\partial f_{16}}{\partial x_{10}} + \frac{\partial E_P}{\partial x_{15}}\frac{\partial f_{15}}{\partial x_{10}} + \frac{\partial E_P}{\partial x_{14}}\frac{\partial f_{14}}{\partial x_{10}} =$$

$$\varepsilon_{17}\frac{\partial f_{17}}{\partial x_{10}} + \varepsilon_{16}\frac{\partial f_{16}}{\partial x_{10}} + \varepsilon_{15}\frac{\partial f_{15}}{\partial x_{10}} + \varepsilon_{14}\frac{\partial f_{14}}{\partial x_{10}} =$$

$$\varepsilon_{17}\frac{\partial(w_4^{DP})}{\partial(\overline{w_1})} + \varepsilon_{16}\frac{\partial(w_3^{DP})}{\partial(\overline{w_1})} + \varepsilon_{15}\frac{\partial(w_2^{DP})}{\partial(\overline{w_1})} + \varepsilon_{14}\frac{\partial(w_1^{DP})}{\partial(\overline{w_1})} =$$

$$\varepsilon_{17}\frac{\partial[(\overline{w}_4 \bullet (\overline{w_1} + \overline{w_2} + \overline{w_3} + \overline{w_4})]}{\partial(\overline{w_1})} + \varepsilon_{16}\frac{\partial[(\overline{w}_3 \bullet (\overline{w_1} + \overline{w_2} + \overline{w_3} + \overline{w_4})]}{\partial(\overline{w_1})} +$$

$$\varepsilon_{15}\frac{\partial[(\overline{w}_2 \bullet (\overline{w_1} + \overline{w_2} + \overline{w_3} + \overline{w_4})]}{\partial(\overline{w_1})} + \varepsilon_{14}\frac{\partial[(\overline{w}_1 \bullet (\overline{w_1} + \overline{w_2} + \overline{w_3} + \overline{w_4})]}{\partial(\overline{w_1})} \qquad (6.23)$$

If $\overline{w_1} = x_1 + iy_1$, $\overline{w_2} = x_2 + iy_2$, $\overline{w_3} = x_3 + iy_3$ and $\overline{w_4} = x_4 + iy_4$ then the derivative formula in (6.20) yields:

$$\varepsilon_{13} = \varepsilon_{17}\frac{\partial[(\overline{w_4}\bullet\overline{w_1} + \overline{w_4}\bullet\overline{w_2} + \overline{w_4}\bullet\overline{w_3} + \overline{w_4}\bullet\overline{w_4})]}{\partial(\overline{w_4})} + \varepsilon_{16}\frac{\partial[(\overline{w_3}\bullet\overline{w_1} + \overline{w_3}\bullet\overline{w_2} + \overline{w_3}\bullet\overline{w_3} + \overline{w_3}\bullet\overline{w_4})]}{\partial(\overline{w_4})} +$$
$$\varepsilon_{15}\frac{\partial[(\overline{w_2}\bullet\overline{w_1} + \overline{w_2}\bullet\overline{w_2} + \overline{w_2}\bullet\overline{w_3} + \overline{w_2}\bullet\overline{w_4})]}{\partial(\overline{w_4})} + \varepsilon_{14}\frac{\partial[(\overline{w_1}\bullet\overline{w_1} + \overline{w_1}\bullet\overline{w_2} + \overline{w_1}\bullet\overline{w_3} + \overline{w_1}\bullet\overline{w_4})]}{\partial(\overline{w_4})} =$$
$$\varepsilon_{17}\frac{\partial[(x_1x_4 + y_1y_4) + (x_2x_4 + y_2y_4) + (x_3x_4 + y_3y_4) + (x_4x_4 + y_4y_4)]}{\partial(x_4 + iy_4)} +$$
$$\varepsilon_{16}\frac{\partial[(x_1x_3 + y_1y_3) + (x_2x_3 + y_2y_3) + (x_3x_3 + y_3y_3) + (x_3x_4 + y_3y_4)]}{\partial(x_4 + iy_4)} +$$
$$\varepsilon_{15}\frac{\partial[(x_1x_2 + y_1y_2) + (x_2x_2 + y_2y_2) + (x_2x_3 + y_2y_3) + (x_2x_4 + y_2y_4)]}{\partial(x_4 + iy_4)} +$$
$$\varepsilon_{14}\frac{\partial[(x_1x_1 + y_1y_1) + (x_1x_2 + y_1y_2) + (x_1x_3 + y_1y_3) + (x_1x_4 + y_1y_4)]}{\partial(x_4 + iy_4)} = \qquad (6.24)$$
$$\frac{1}{2}\varepsilon_{17}((x_1 + x_2 + x_3 + 2x_4) + i(y_1 + y_2 + y_3 + 2y_4)) +$$
$$\frac{1}{2}\varepsilon_{16}(x_3 - iy_3) + \frac{1}{2}\varepsilon_{15}(x_2 - iy_2) + \frac{1}{2}\varepsilon_{14}(x_1 - iy_1) =$$
$$\frac{1}{2}\varepsilon_{17}(\overline{\overline{w_1}} + \overline{\overline{w_2}} + \overline{\overline{w_3}} + \overline{\overline{w_4}}) + \frac{1}{2}\varepsilon_{17}\overline{\overline{w_4}} + \frac{1}{2}\varepsilon_{16}\overline{\overline{w_3}} + \frac{1}{2}\varepsilon_{15}\overline{\overline{w_2}} + \frac{1}{2}\varepsilon_{14}\overline{\overline{w_1}}$$

Here $\overline{\overline{w_i}}$ reflects the complex conjugate of normalized weights. With the use of derivative formula of complex function in (6.21) we have:

$$\varepsilon_{12}=\varepsilon_{17}\frac{\partial[(\overline{w}_4\bullet\overline{w}_1+\overline{w}_4\bullet\overline{w}_2+\overline{w}_4\bullet\overline{w}_3+\overline{w}_4\bullet\overline{w}_4)]}{\partial(\overline{w}_3)}+\varepsilon_{16}\frac{\partial[(\overline{w}_3\bullet\overline{w}_1+\overline{w}_3\bullet\overline{w}_2+\overline{w}_3\bullet\overline{w}_3+\overline{w}_3\bullet\overline{w}_4)]}{\partial(\overline{w}_3)}+$$
$$\varepsilon_{15}\frac{\partial[(\overline{w}_2\bullet\overline{w}_1+\overline{w}_2\bullet\overline{w}_2+\overline{w}_2\bullet\overline{w}_3+\overline{w}_2\bullet\overline{w}_4)]}{\partial(\overline{w}_3)}+\varepsilon_{14}\frac{\partial[(\overline{w}_1\bullet\overline{w}_1+\overline{w}_1\bullet\overline{w}_2+\overline{w}_1\bullet\overline{w}_3+\overline{w}_1\bullet\overline{w}_4)]}{\partial(\overline{w}_3)}=$$
$$\varepsilon_{17}\frac{\partial[(x_1x_4+y_1y_4)+(x_2x_4+y_2y_4)+(x_3x_4+y_3y_4)+(x_4x_4+y_4y_4)]}{\partial(x_3+iy_3)}+$$
$$\varepsilon_{16}\frac{\partial[(x_1x_3+y_1y_3)+(x_2x_3+y_2y_3)+(x_3x_3+y_3y_3)+(x_3x_4+y_3y_4)]}{\partial(x_3+iy_3)}+$$
$$\varepsilon_{15}\frac{\partial[(x_1x_2+y_1y_2)+(x_2x_2+y_2y_2)+(x_2x_3+y_2y_3)+(x_2x_4+y_2y_4)]}{\partial(x_3+iy_3)}+$$
$$\varepsilon_{14}\frac{\partial[(x_1x_1+y_1y_1)+(x_1x_2+y_1y_2)+(x_1x_3+y_1y_3)+(x_1x_4+y_1y_4)]}{\partial(x_3+iy_3)}= \qquad (6.25)$$
$$\frac{1}{2}\varepsilon_{17}(x_4-iy_4)+\frac{1}{2}\varepsilon_{16}((x_1+x_2+2x_3+x_4)+$$
$$i(y_1+y_2+2y_3+y_4))+\frac{1}{2}\varepsilon_{15}(x_2-iy_2)+\frac{1}{2}\varepsilon_{14}(x_1-iy_1)=$$
$$\frac{1}{2}\varepsilon_{17}\overline{\overline{w}}_4+\frac{1}{2}\varepsilon_{16}(\overline{\overline{w}}_1+\overline{\overline{w}}_2+\overline{\overline{w}}_3+\overline{\overline{w}}_4)+\frac{1}{2}\varepsilon_{16}\overline{\overline{w}}_3+\frac{1}{2}\varepsilon_{15}\overline{\overline{w}}_2+\frac{1}{2}\varepsilon_{14}\overline{\overline{w}}_1$$

Again the use of derivative formula of complex function in (6.22) represents:

$$\varepsilon_{17}\frac{\partial[(x_1x_4+y_1y_4)+(x_2x_4+y_2y_4)+(x_3x_4+y_3y_4)+(x_4x_4+y_4y_4)]}{\partial(x_2+iy_2)}+$$
$$\varepsilon_{16}\frac{\partial[(x_1x_3+y_1y_3)+(x_2x_3+y_2y_3)+(x_3x_3+y_3y_3)+(x_3x_4+y_3y_4)]}{\partial(x_2+iy_2)}+$$
$$\varepsilon_{15}\frac{\partial[(x_1x_2+y_1y_2)+(x_2x_2+y_2y_2)+(x_2x_3+y_2y_3)+(x_2x_4+y_2y_4)]}{\partial(x_2+iy_2)}+$$
$$\varepsilon_{14}\frac{\partial[(x_1x_1+y_1y_1)+(x_1x_2+y_1y_2)+(x_1x_3+y_1y_3)+(x_1x_4+y_1y_4)]}{\partial(x_2+iy_2)}=$$
$$\frac{1}{2}\varepsilon_{17}(x_4-iy_4)+\frac{1}{2}\varepsilon_{16}(x_3-iy_3)+\frac{1}{2}\varepsilon_{15}((x_1+2x_2+x_3+x_4)+$$
$$i(y_1+2y_2+y_3+y_4))+\frac{1}{2}\varepsilon_{14}(x_1-iy_1)=$$
$$\frac{1}{2}\varepsilon_{17}\overline{\overline{w}}_4+\frac{1}{2}\varepsilon_{16}\overline{\overline{w}}_3+\frac{1}{2}\varepsilon_{15}(\overline{\overline{w}}_1+\overline{\overline{w}}_2+\overline{\overline{w}}_3+\overline{\overline{w}}_4)+\frac{1}{2}\varepsilon_{15}\overline{\overline{w}}_2+\frac{1}{2}\varepsilon_{14}\overline{\overline{w}}_1 \qquad (6.26)$$

The same case happens for (6.23):

$$\varepsilon_{10} = \varepsilon_{17}\frac{\partial[(\overline{w}_4 \bullet \overline{w}_1 + \overline{w}_4 \bullet \overline{w}_2 + \overline{w}_4 \bullet \overline{w}_3 + \overline{w}_4 \bullet \overline{w}_4)]}{\partial(\overline{w}_1)} + \varepsilon_{16}\frac{\partial[(\overline{w}_3 \bullet \overline{w}_1 + \overline{w}_3 \bullet \overline{w}_2 + \overline{w}_3 \bullet \overline{w}_3 + \overline{w}_3 \bullet \overline{w}_4)]}{\partial(\overline{w}_1)} +$$
$$\varepsilon_{15}\frac{\partial[(\overline{w}_2 \bullet \overline{w}_1 + \overline{w}_2 \bullet \overline{w}_2 + \overline{w}_2 \bullet \overline{w}_3 + \overline{w}_2 \bullet \overline{w}_4)]}{\partial(\overline{w}_1)} + \varepsilon_{14}\frac{\partial[(\overline{w}_1 \bullet \overline{w}_1 + \overline{w}_1 \bullet \overline{w}_2 + \overline{w}_1 \bullet \overline{w}_3 + \overline{w}_1 \bullet \overline{w}_4)]}{\partial(\overline{w}_1)} =$$
$$\varepsilon_{17}\frac{\partial[(x_1x_4 + y_1y_4) + (x_2x_4 + y_2y_4) + (x_3x_4 + y_3y_4) + (x_4x_4 + y_4y_4)]}{\partial(x_1 + iy_1)} +$$
$$\varepsilon_{16}\frac{\partial[(x_1x_3 + y_1y_3) + (x_2x_3 + y_2y_3) + (x_3x_3 + y_3y_3) + (x_3x_4 + y_3y_4)]}{\partial(x_1 + iy_1)} +$$
$$\varepsilon_{15}\frac{\partial[(x_1x_2 + y_1y_2) + (x_2x_2 + y_2y_2) + (x_2x_3 + y_2y_3) + (x_2x_4 + y_2y_4)]}{\partial(x_1 + iy_1)} +$$
$$\varepsilon_{14}\frac{\partial[(x_1x_1 + y_1y_1) + (x_1x_2 + y_1y_2) + (x_1x_3 + y_1y_3) + (x_1x_4 + y_1y_4)]}{\partial(x_1 + iy_1)} =$$
$$\frac{1}{2}\varepsilon_{17}(x_4 - iy_4) + \frac{1}{2}\varepsilon_{16}(x_3 - iy_3) + \frac{1}{2}\varepsilon_{15}(x_2 - iy_2) +$$
$$\frac{1}{2}\varepsilon_{14}((2x_1 + x_2 + x_3 + x_4) + i(2y_1 + y_2 + y_3 + y_4)) =$$
$$\frac{1}{2}\varepsilon_{17}\overline{\overline{w}}_4 + \frac{1}{2}\varepsilon_{16}\overline{\overline{w}}_3 + \frac{1}{2}\varepsilon_{15}\overline{\overline{w}}_2 + \frac{1}{2}\varepsilon_{14}(\overline{\overline{w}}_1 + \overline{\overline{w}}_2 + \overline{\overline{w}}_3 + \overline{\overline{w}}_4) + \frac{1}{2}\varepsilon_{14}\overline{\overline{w}}_1 \qquad (6.27)$$

From *layer 3* to *layer 2*:

$$\varepsilon_9 = \frac{\partial E_P}{\partial x_9} = \frac{\partial E_P}{\partial x_{13}}\frac{\partial f_{13}}{\partial x_9} + \frac{\partial E_P}{\partial x_{12}}\frac{\partial f_{12}}{\partial x_9} + \frac{\partial E_P}{\partial x_{11}}\frac{\partial f_{11}}{\partial x_9} + \frac{\partial E_P}{\partial x_{10}}\frac{\partial f_{10}}{\partial x_9} =$$
$$\varepsilon_{13}\frac{\partial f_{13}}{\partial x_9} + \varepsilon_{12}\frac{\partial f_{12}}{\partial x_9} + \varepsilon_{11}\frac{\partial f_{11}}{\partial x_9} + \varepsilon_{10}\frac{\partial f_{10}}{\partial x_9} =$$
$$\varepsilon_{13}\frac{\partial(\overline{w}_4)}{\partial(w_4)} + \varepsilon_{12}\frac{\partial(\overline{w}_3)}{\partial(w_4)} + \varepsilon_{11}\frac{\partial(\overline{w}_2)}{\partial(w_4)} + \varepsilon_{10}\frac{\partial(\overline{w}_1)}{\partial(w_4)} =$$
$$\varepsilon_{13}\frac{\partial[\frac{w_4}{|w_1| + |w_2| + |w_3| + |w_4|}]}{\partial(w_4)} + \varepsilon_{12}\frac{\partial[\frac{w_3}{|w_1| + |w_2| + |w_3| + |w_4|}]}{\partial(w_4)} +$$
$$\varepsilon_{11}\frac{\partial[\frac{w_2}{|w_1| + |w_2| + |w_3| + |w_4|}]}{\partial(w_4)} + \varepsilon_{10}\frac{\partial[\frac{w_1}{|w_1| + |w_2| + |w_3| + |w_4|}]}{\partial(w_4)} \qquad (6.28)$$

$$\varepsilon_8 = \frac{\partial E_P}{\partial x_8} = \frac{\partial E_P}{\partial x_{13}}\frac{\partial f_{13}}{\partial x_8} + \frac{\partial E_P}{\partial x_{12}}\frac{\partial f_{12}}{\partial x_8} + \frac{\partial E_P}{\partial x_{11}}\frac{\partial f_{11}}{\partial x_8} + \frac{\partial E_P}{\partial x_{10}}\frac{\partial f_{10}}{\partial x_8} =$$

$$\varepsilon_{13}\frac{\partial f_{13}}{\partial x_8} + \varepsilon_{12}\frac{\partial f_{12}}{\partial x_8} + \varepsilon_{11}\frac{\partial f_{11}}{\partial x_8} + \varepsilon_{10}\frac{\partial f_{10}}{\partial x_8} =$$

$$\varepsilon_{13}\frac{\partial(\overline{w_4})}{\partial(w_3)} + \varepsilon_{12}\frac{\partial(\overline{w_3})}{\partial(w_3)} + \varepsilon_{11}\frac{\partial(\overline{w_2})}{\partial(w_3)} + \varepsilon_{10}\frac{\partial(\overline{w_1})}{\partial(w_3)} =$$

$$\varepsilon_{13}\frac{\partial[\dfrac{w_4}{|w_1|+|w_2|+|w_3|+|w_4|}]}{\partial(w_3)} + \varepsilon_{12}\frac{\partial[\dfrac{w_3}{|w_1|+|w_2|+|w_3|+|w_4|}]}{\partial(w_3)} +$$

$$\varepsilon_{11}\frac{\partial[\dfrac{w_2}{|w_1|+|w_2|+|w_3|+|w_4|}]}{\partial(w_3)} + \varepsilon_{10}\frac{\partial[\dfrac{w_1}{|w_1|+|w_2|+|w_3|+|w_4|}]}{\partial(w_3)} \qquad (6.29)$$

$$\varepsilon_7 = \frac{\partial E_P}{\partial x_7} = \frac{\partial E_P}{\partial x_{13}}\frac{\partial f_{13}}{\partial x_7} + \frac{\partial E_P}{\partial x_{12}}\frac{\partial f_{12}}{\partial x_7} + \frac{\partial E_P}{\partial x_{11}}\frac{\partial f_{11}}{\partial x_7} + \frac{\partial E_P}{\partial x_{10}}\frac{\partial f_{10}}{\partial x_7} =$$

$$\varepsilon_{13}\frac{\partial f_{13}}{\partial x_7} + \varepsilon_{12}\frac{\partial f_{12}}{\partial x_7} + \varepsilon_{11}\frac{\partial f_{11}}{\partial x_7} + \varepsilon_{10}\frac{\partial f_{10}}{\partial x_7} =$$

$$\varepsilon_{13}\frac{\partial(\overline{w_4})}{\partial(w_2)} + \varepsilon_{12}\frac{\partial(\overline{w_3})}{\partial(w_2)} + \varepsilon_{11}\frac{\partial(\overline{w_2})}{\partial(w_2)} + \varepsilon_{10}\frac{\partial(\overline{w_1})}{\partial(w_2)} =$$

$$\varepsilon_{13}\frac{\partial[\dfrac{w_4}{|w_1|+|w_2|+|w_3|+|w_4|}]}{\partial(w_2)} + \varepsilon_{12}\frac{\partial[\dfrac{w_3}{|w_1|+|w_2|+|w_3|+|w_4|}]}{\partial(w_2)} +$$

$$\varepsilon_{11}\frac{\partial[\dfrac{w_2}{|w_1|+|w_2|+|w_3|+|w_4|}]}{\partial(w_2)} + \varepsilon_{10}\frac{\partial[\dfrac{w_1}{|w_1|+|w_2|+|w_3|+|w_4|}]}{\partial(w_2)} \qquad (6.30)$$

$$\varepsilon_6 = \frac{\partial E_P}{\partial x_6} = \frac{\partial E_P}{\partial x_{13}}\frac{\partial f_{13}}{\partial x_6} + \frac{\partial E_P}{\partial x_{12}}\frac{\partial f_{12}}{\partial x_6} + \frac{\partial E_P}{\partial x_{11}}\frac{\partial f_{11}}{\partial x_6} + \frac{\partial E_P}{\partial x_{10}}\frac{\partial f_{10}}{\partial x_6} =$$

$$\varepsilon_{13}\frac{\partial f_{13}}{\partial x_6} + \varepsilon_{12}\frac{\partial f_{12}}{\partial x_6} + \varepsilon_{11}\frac{\partial f_{11}}{\partial x_6} + \varepsilon_{10}\frac{\partial f_{10}}{\partial x_6} =$$

$$\varepsilon_{13}\frac{\partial(\overline{w_4})}{\partial(w_1)} + \varepsilon_{12}\frac{\partial(\overline{w_3})}{\partial(w_1)} + \varepsilon_{11}\frac{\partial(\overline{w_2})}{\partial(w_1)} + \varepsilon_{10}\frac{\partial(\overline{w_1})}{\partial(w_1)} =$$

$$\varepsilon_{13}\frac{\partial[\dfrac{w_4}{|w_1|+|w_2|+|w_3|+|w_4|}]}{\partial(w_1)} + \varepsilon_{12}\frac{\partial[\dfrac{w_3}{|w_1|+|w_2|+|w_3|+|w_4|}]}{\partial(w_1)} +$$

$$\varepsilon_{11}\frac{\partial[\dfrac{w_2}{|w_1|+|w_2|+|w_3|+|w_4|}]}{\partial(w_1)} + \varepsilon_{10}\frac{\partial[\dfrac{w_1}{|w_1|+|w_2|+|w_3|+|w_4|}]}{\partial(w_1)} \qquad (6.31)$$

If we assume that $w_1 = x_1 + iy_1$, $w_2 = x_2 + iy_2$, $w_3 = x_3 + iy_3$ and $w_4 = x_4 + iy_4$.

Then with complex function in (6.28) we have:

$$\varepsilon_9 = \varepsilon_{13}\frac{\partial[\dfrac{(x_4+iy_4)}{\sqrt{x_1^2+y_1^2}+\sqrt{x_2^2+y_2^2}+\sqrt{x_3^2+y_3^2}+\sqrt{x_4^2+y_4^2}}]}{\partial(x_4+iy_4)} +$$

$$\varepsilon_{12}\frac{\partial[\dfrac{(x_3+iy_3)}{\sqrt{x_1^2+y_1^2}+\sqrt{x_2^2+y_2^2}+\sqrt{x_3^2+y_3^2}+\sqrt{x_4^2+y_4^2}}]}{\partial(x_4+iy_4)} +$$

$$\varepsilon_{11}\frac{\partial[\dfrac{(x_2+iy_2)}{\sqrt{x_1^2+y_1^2}+\sqrt{x_2^2+y_2^2}+\sqrt{x_3^2+y_3^2}+\sqrt{x_4^2+y_4^2}}]}{\partial(x_4+iy_4)} +$$

$$\varepsilon_{10}\frac{\partial[\dfrac{(x_1+iy_1)}{\sqrt{x_1^2+y_1^2}+\sqrt{x_2^2+y_2^2}+\sqrt{x_3^2+y_3^2}+\sqrt{x_4^2+y_4^2}}]}{\partial(x_4+iy_4)} \qquad (6.32)$$

Consider $M = \sqrt{x_1^2 + y_1^2} + \sqrt{x_2^2 + y_2^2} + \sqrt{x_3^2 + y_3^2} + \sqrt{x_4^2 + y_4^2}$ and

$\frac{x_4 + iy_4}{M} = u_4 + iv_4$ and $\frac{x_3 + iy_3}{M} = u_3 + iv_3$ and $\frac{x_2 + iy_2}{M} = u_2 + iv_2$ and

$\frac{x_1 + iy_1}{M} = u_1 + iv_1$. Then,

$$\frac{\partial u_4}{\partial x_4} = \frac{M - \frac{x_4^2}{\sqrt{x_4^2 + y_4^2}}}{M^2}, \quad \frac{\partial u_4}{\partial y_4} = \frac{-x_4 y_4}{M^2\sqrt{x_4^2 + y_4^2}} = \frac{\partial v_4}{\partial x_4} \text{ and } \frac{\partial v_4}{\partial y_4} = \frac{M - \frac{y_4^2}{\sqrt{x_4^2 + y_4^2}}}{M^2}$$

$$\frac{\partial\left[\frac{(x_4 + iy_4)}{\sqrt{x_1^2 + y_1^2} + \sqrt{x_2^2 + y_2^2} + \sqrt{x_3^2 + y_3^2} + \sqrt{x_4^2 + y_4^2}}\right]}{\partial(x_4 + iy_4)} = \frac{1}{2}\left[\left(\frac{\partial u_4}{\partial x_4} + \frac{\partial v_4}{\partial y_4}\right) + \left(-i\frac{\partial u_4}{\partial y_4} + i\frac{\partial v_4}{\partial x_4}\right)\right]$$

And here is $\frac{x_3 + iy_3}{M} = u_3 + iv_3$ with $u_3 = \frac{x_3}{M}$, $v_3 = \frac{y_3}{M}$ and

$M = \sqrt{x_1^2 + y_1^2} + \sqrt{x_2^2 + y_2^2} + \sqrt{x_3^2 + y_3^2} + \sqrt{x_4^2 + y_4^2}$:

$$\frac{\partial u_3}{\partial x_4} = \frac{-x_3 x_4}{M^2\sqrt{x_4^2 + y_4^2}}, \frac{\partial u_3}{\partial y_4} = \frac{-x_3 y_4}{M^2\sqrt{x_4^2 + y_4^2}}, \frac{\partial v_3}{\partial x_4} = \frac{-y_3 x_4}{M^2\sqrt{x_4^2 + y_4^2}} \quad \text{and}$$

$$\frac{\partial v_3}{\partial y_4} = \frac{-y_3 y_4}{M^2\sqrt{x_4^2 + y_4^2}}$$

$$\frac{\partial\left[\frac{(x_3 + iy_3)}{\sqrt{x_1^2 + y_1^2} + \sqrt{x_2^2 + y_2^2} + \sqrt{x_3^2 + y_3^2} + \sqrt{x_4^2 + y_4^2}}\right]}{\partial(x_4 + iy_4)} = \frac{1}{2}\left[\left(\frac{\partial u_3}{\partial x_4} + \frac{\partial v_3}{\partial y_4}\right) + \left(-i\frac{\partial u_3}{\partial y_4} + i\frac{\partial v_3}{\partial x_4}\right)\right]$$

And let $\frac{x_2 + iy_2}{M} = u_2 + iv_2$ with $u_2 = \frac{x_2}{M}$, $v_2 = \frac{y_2}{M}$ and

$M = \sqrt{x_1^2 + y_1^2} + \sqrt{x_2^2 + y_2^2} + \sqrt{x_3^2 + y_3^2} + \sqrt{x_4^2 + y_4^2}$ then,

$$\frac{\partial u_2}{\partial x_4} = \frac{-x_2 x_4}{M^2\sqrt{x_4^2+y_4^2}},\ \frac{\partial u_2}{\partial y_4} = \frac{-x_2 y_4}{M^2\sqrt{x_4^2+y_4^2}},\frac{\partial v_2}{\partial x_4} = \frac{-y_2 x_4}{M^2\sqrt{x_4^2+y_4^2}} \text{ and}$$

$$\frac{\partial v_2}{\partial y_4} = \frac{-y_2 y_4}{M^2\sqrt{x_4^2+y_4^2}}$$

$$\frac{\partial[\frac{(x_2+iy_2)}{\sqrt{x_1^2+y_1^2}+\sqrt{x_2^2+y_2^2}+\sqrt{x_3^2+y_3^2}+\sqrt{x_4^2+y_4^2}}]}{\partial(x_4+iy_4)} = \frac{1}{2}\left[\left(\frac{\partial u_2}{\partial x_4}+\frac{\partial v_2}{\partial y_4}\right)+\left(-i\frac{\partial u_2}{\partial y_4}+i\frac{\partial v_2}{\partial x_4}\right)\right]$$

Also let $\frac{x_1+iy_1}{M} = u_1+iv_1$ with $u_1 = \frac{x_1}{M}$, $v_1 = \frac{y_1}{M}$ and

$M = \sqrt{x_1^2+y_1^2}+\sqrt{x_2^2+y_2^2}+\sqrt{x_3^2+y_3^2}+\sqrt{x_4^2+y_4^2}$, then:

$$\frac{\partial u_1}{\partial x_4} = \frac{-x_1 x_4}{M^2\sqrt{x_4^2+y_4^2}},\ \frac{\partial u_1}{\partial y_4} = \frac{-x_1 y_4}{M^2\sqrt{x_4^2+y_4^2}},\ \frac{\partial v_1}{\partial x_4} = \frac{-y_1 x_4}{M^2\sqrt{x_4^2+y_4^2}} \text{ and}$$

$$\frac{\partial v_1}{\partial y_4} = \frac{-y_1 y_4}{M^2\sqrt{x_4^2+y_4^2}}$$

$$\frac{\partial[\frac{(x_1+iy_1)}{\sqrt{x_1^2+y_1^2}+\sqrt{x_2^2+y_2^2}+\sqrt{x_3^2+y_3^2}+\sqrt{x_4^2+y_4^2}}]}{\partial(x_4+iy_4)} = \frac{1}{2}\left[\left(\frac{\partial u_1}{\partial x_4}+\frac{\partial v_1}{\partial y_4}\right)+\left(-i\frac{\partial u_1}{\partial y_4}+i\frac{\partial v_1}{\partial x_4}\right)\right]$$

Continuing with this, we can have:

$$\frac{\partial u_4}{\partial x_3} = \frac{-x_3 x_4}{M^2\sqrt{x_3^2+y_3^2}},\ \frac{\partial u_4}{\partial y_3} = \frac{-x_3 y_3}{M^2\sqrt{x_3^2+y_3^2}},\frac{\partial v_4}{\partial x_3} = \frac{-x_3 y_4}{M^2\sqrt{x_3^2+y_3^2}} \text{ and}$$

$$\frac{\partial v_4}{\partial y_3} = \frac{-y_3 y_4}{M^2\sqrt{x_3^2+y_3^2}}$$

And

$$\frac{\partial u_3}{\partial x_3} = \frac{M - \frac{x_3^2}{\sqrt{x_3^2 + y_3^2}}}{M^2} \;,\; \frac{\partial u_3}{\partial y_3} = \frac{-x_3 y_3}{M^2 \sqrt{x_3^2 + y_3^2}} = \frac{\partial v_3}{\partial x_3} \text{ and } \frac{\partial v_3}{\partial y_3} = \frac{M - \frac{y_3^2}{\sqrt{x_3^2 + y_3^2}}}{M^2}$$

And

$$\frac{\partial u_2}{\partial x_3} = \frac{-x_2 x_3}{M^2 \sqrt{x_3^2 + y_3^2}} \,,\; \frac{\partial u_2}{\partial y_3} = \frac{-x_2 y_3}{M^2 \sqrt{x_3^2 + y_3^2}} \,,\; \frac{\partial v_2}{\partial x_3} = \frac{-y_2 x_3}{M^2 \sqrt{x_3^2 + y_3^2}} \text{ and}$$

$$\frac{\partial v_2}{\partial y_3} = \frac{-y_2 y_3}{M^2 \sqrt{x_3^2 + y_3^2}}$$

And

$$\frac{\partial u_1}{\partial x_3} = \frac{-x_1 x_3}{M^2 \sqrt{x_3^2 + y_3^2}} \,,\; \frac{\partial u_1}{\partial y_3} = \frac{-x_1 y_3}{M^2 \sqrt{x_3^2 + y_3^2}} \,,\; \frac{\partial v_1}{\partial x_3} = \frac{-y_1 x_3}{M^2 \sqrt{x_3^2 + y_3^2}} \text{ and}$$

$$\frac{\partial v_1}{\partial y_3} = \frac{-y_1 y_3}{M^2 \sqrt{x_3^2 + y_3^2}}$$

And

$$\frac{\partial u_4}{\partial x_2} = \frac{-x_2 x_4}{M^2 \sqrt{x_2^2 + y_2^2}} \,,\; \frac{\partial u_4}{\partial y_2} = \frac{-x_4 y_2}{M^2 \sqrt{x_2^2 + y_2^2}} \,,\; \frac{\partial v_4}{\partial x_2} = \frac{-y_4 x_2}{M^2 \sqrt{x_2^2 + y_2^2}} \text{ and}$$

$$\frac{\partial v_4}{\partial y_2} = \frac{-y_2 y_4}{M^2 \sqrt{x_2^2 + y_2^2}}$$

And

$$\frac{\partial u_3}{\partial x_2} = \frac{-x_2 x_3}{M^2 \sqrt{x_2^2 + y_2^2}} \,,\; \frac{\partial u_3}{\partial y_2} = \frac{-x_3 y_2}{M^2 \sqrt{x_2^2 + y_2^2}} \,,\; \frac{\partial v_3}{\partial x_2} = \frac{-y_3 x_2}{M^2 \sqrt{x_2^2 + y_2^2}} \text{ and}$$

$$\frac{\partial v_4}{\partial y_2} = \frac{-y_2 y_3}{M^2 \sqrt{x_2^2 + y_2^2}}$$

And

$$\frac{\partial u_2}{\partial x_2} = \frac{M - \dfrac{x_2^2}{\sqrt{x_2^2 + y_2^2}}}{M^2}, \quad \frac{\partial u_2}{\partial y_2} = \frac{-x_2 y_2}{M^2\sqrt{x_2^2 + y_2^2}} = \frac{\partial v_2}{\partial x_2} \text{ and } \frac{\partial v_2}{\partial y_2} = \frac{M - \dfrac{y_2^2}{\sqrt{x_2^2 + y_2^2}}}{M^2}$$

And

$$\frac{\partial u_1}{\partial x_2} = \frac{-x_1 x_2}{M^2\sqrt{x_2^2 + y_2^2}}, \quad \frac{\partial u_1}{\partial y_2} = \frac{-x_1 y_2}{M^2\sqrt{x_2^2 + y_2^2}}, \quad \frac{\partial v_1}{\partial x_2} = \frac{-y_1 x_2}{M^2\sqrt{x_2^2 + y_2^2}} \text{ and}$$

$$\frac{\partial v_1}{\partial y_2} = \frac{-y_1 y_2}{M^2\sqrt{x_2^2 + y_2^2}}$$

And

$$\frac{\partial u_4}{\partial x_1} = \frac{-x_1 x_4}{M^2\sqrt{x_1^2 + y_1^2}}, \quad \frac{\partial u_4}{\partial y_1} = \frac{-x_4 y_1}{M^2\sqrt{x_1^2 + y_1^2}}, \frac{\partial v_4}{\partial x_1} = \frac{-y_4 x_1}{M^2\sqrt{x_1^2 + y_1^2}} \text{ and}$$

$$\frac{\partial v_4}{\partial y_1} = \frac{-y_1 y_4}{M^2\sqrt{x_1^2 + y_1^2}}$$

And

$$\frac{\partial u_3}{\partial x_1} = \frac{-x_1 x_3}{M^2\sqrt{x_1^2 + y_1^2}}, \quad \frac{\partial u_3}{\partial y_1} = \frac{-x_3 y_1}{M^2\sqrt{x_1^2 + y_1^2}}, \frac{\partial v_3}{\partial x_1} = \frac{-y_3 x_1}{M^2\sqrt{x_1^2 + y_1^2}} \text{ and}$$

$$\frac{\partial v_4}{\partial y_2} = \frac{-y_1 y_3}{M^2\sqrt{x_1^2 + y_1^2}}$$

And

$$\frac{\partial u_2}{\partial x_1} = \frac{-x_1 x_2}{M^2\sqrt{x_1^2 + y_1^2}}, \quad \frac{\partial u_2}{\partial y_1} = \frac{-x_2 y_1}{M^2\sqrt{x_1^2 + y_1^2}}, \quad \frac{\partial v_2}{\partial x_1} = \frac{-y_2 x_1}{M^2\sqrt{x_1^2 + y_1^2}} \text{ and}$$

$$\frac{\partial v_2}{\partial y_1} = \frac{-y_1 y_2}{M^2\sqrt{x_1^2 + y_1^2}}$$

And

$$\frac{\partial u_1}{\partial x_1} = \frac{M - \frac{x_1^2}{\sqrt{x_1^2 + y_1^2}}}{M^2} \;,\; \frac{\partial u_1}{\partial y_1} = \frac{-x_1 y_1}{M^2 \sqrt{x_1^2 + y_1^2}} = \frac{\partial v_1}{\partial x_1} \text{ and } \frac{\partial v_1}{\partial y_1} = \frac{M - \frac{y_1^2}{\sqrt{x_1^2 + y_1^2}}}{M^2}$$

From *layer 2* to *layer 1*:

$$\varepsilon_5 = \frac{\partial E_P}{\partial x_5} = \frac{\partial E_P}{\partial x_7}\frac{\partial f_7}{\partial x_5} + \frac{\partial E_P}{\partial x_9}\frac{\partial f_9}{\partial x_5} = \varepsilon_7 \frac{\partial f_7}{\partial x_5} + \varepsilon_9 \frac{\partial f_9}{\partial x_5} =$$
$$\varepsilon_7 \frac{\partial((x_2 + iy_2)(x_5 + iy_5))}{\partial x_5} + \varepsilon_9 \frac{\partial((x_3 + iy_3)(x_5 + iy_5))}{\partial x_5} =$$
$$\varepsilon_7 \frac{\partial((x_2 x_5 - y_2 y_5) + i(x_2 y_5 + x_5 y_2))}{\partial(x_5)} + \varepsilon_9 \frac{\partial((x_3 x_5 - y_3 y_5) + i(x_3 y_5 + x_5 y_3))}{\partial(x_5)} =$$
$$\varepsilon_7(x_2 + iy_2) + \varepsilon_9(x_3 + iy_3) \qquad (6.33)$$

$$\varepsilon_4 = \frac{\partial E_P}{\partial x_4} = \frac{\partial E_P}{\partial x_6}\frac{\partial f_6}{\partial x_4} + \frac{\partial E_P}{\partial x_8}\frac{\partial f_8}{\partial x_4} = \varepsilon_6 \frac{\partial f_6}{\partial x_4} + \varepsilon_8 \frac{\partial f_8}{\partial x_4} =$$
$$\varepsilon_6 \frac{\partial((x_2 + iy_2)(x_4 + iy_4))}{\partial x_5} + \varepsilon_8 \frac{\partial((x_3 + iy_3)(x_4 + iy_4))}{\partial x_5} =$$
$$\varepsilon_6 \frac{\partial((x_2 x_4 - y_2 y_4) + i(x_2 y_4 + x_4 y_2))}{\partial(x_4)} + \varepsilon_8 \frac{\partial((x_3 x_4 - y_3 y_4) + i(x_3 y_4 + x_4 y_3))}{\partial(x_4)} =$$
$$\varepsilon_6(x_2 + iy_2) + \varepsilon_8(x_3 + iy_3) \qquad (6.34)$$

$$\varepsilon_3 = \frac{\partial E_P}{\partial x_3} = \frac{\partial E_P}{\partial x_8}\frac{\partial f_8}{\partial x_3} + \frac{\partial E_P}{\partial x_9}\frac{\partial f_9}{\partial x_3} = \varepsilon_8 \frac{\partial f_8}{\partial x_3} + \varepsilon_9 \frac{\partial f_9}{\partial x_3} =$$
$$\varepsilon_6 \frac{\partial((x_3 + iy_3)(x_4 + iy_4))}{\partial x_5} + \varepsilon_8 \frac{\partial((x_3 + iy_3)(x_5 + iy_5))}{\partial x_5} =$$
$$\varepsilon_8 \frac{\partial((x_3 x_4 - y_3 y_4) + i(x_3 y_4 + x_4 y_3))}{\partial(x_3)} + \varepsilon_9 \frac{\partial((x_3 x_5 - y_3 y_5) + i(x_3 y_5 + x_5 y_3))}{\partial(x_3)} =$$
$$\varepsilon_8(x_4 + iy_4) + \varepsilon_9(x_5 + iy_5) \qquad (6.35)$$

$$\varepsilon_2 = \frac{\partial E_P}{\partial x_2} = \frac{\partial E_P}{\partial x_6}\frac{\partial f_6}{\partial x_2} + \frac{\partial E_P}{\partial x_7}\frac{\partial f_7}{\partial x_2} = \varepsilon_6 \frac{\partial f_6}{\partial x_2} + \varepsilon_7 \frac{\partial f_7}{\partial x_2} =$$
$$\varepsilon_6 \frac{\partial((x_2 + iy_2)(x_4 + iy_4))}{\partial x_5} + \varepsilon_8 \frac{\partial((x_2 + iy_2)(x_5 + iy_5))}{\partial x_5} =$$
$$\varepsilon_6 \frac{\partial((x_2x_4 - y_2y_4) + i(x_2y_4 + x_4y_2))}{\partial(x_2)} + \varepsilon_7 \frac{\partial((x_2x_5 - y_2y_5) + i(x_2y_5 + x_5y_2))}{\partial(x_2)} =$$
$$\varepsilon_6(x_4 + iy_4) + \varepsilon_7(x_5 + iy_5) \qquad (6.36)$$

After calculating all ε_i for all nodes, then the gradient for generic weight of SMF_{ijk} in node *i* is easy to be calculated according to Equations (3.32) and (3.33).

6.2 Experimental Comparison of Multivariate ANCFIS

In this section, the results of forecasting experiments using multivariate ANCFIS are reported. The experimental design for multivariate ANCFIS is similar to univariate ANCFIS; a single-split, one-step-ahead prediction design with all training data before testing data. In all of the experiments, the normalized mean squared error (NMSE) is reported. All variates in each dataset are already co-registered, which means that the data points relate to the same moments in time. It is also important to normalize multivariate data, so the results are not dominated by the variates that have a larger absolute variance. The window size is again based on finding one approximate period in the time series.

Four different time-series dataset is used; river-flow-hydrology, micro-economics, motel-tourism and car-road-accidents. River-flow is geographically indexed data, collected over time and at different locations [118]-[121]. Although economics data does not necessarily contain geographic component, it represents a source of high-volume multivariate time series data.

We have compared the results of Multivariate ANCFIS with those of univariate ANCFIS for individual variates and all variates together on four different datasets. Each variate is forecasted separately with univariate ANCFIS. Then, the forecasting results of univariate ANCFIS for each variate, are combined to find the forecasting results for all variates. We then forecast the whole dataset using multivariate ANCFIS. The predicted and actual values for each variate are collected, and the forecast error for each variate can be calculated separately as well as in aggregate.

6.2.1 Transport and Tourism-Motel

This dataset is collected by Australian Bureau of Statistics [22]. There are 186 observations with two different variates related to monthly data of hotels, motels and guesthouses in Victoria between Jan 1980 - June 1995. First variate is total number of room night occupied and second variate is about total takings from accommodation. The Pearson's correlation between these columns is 0.943 which means they are very strongly correlated. Figure 6.4 presents this correlation.

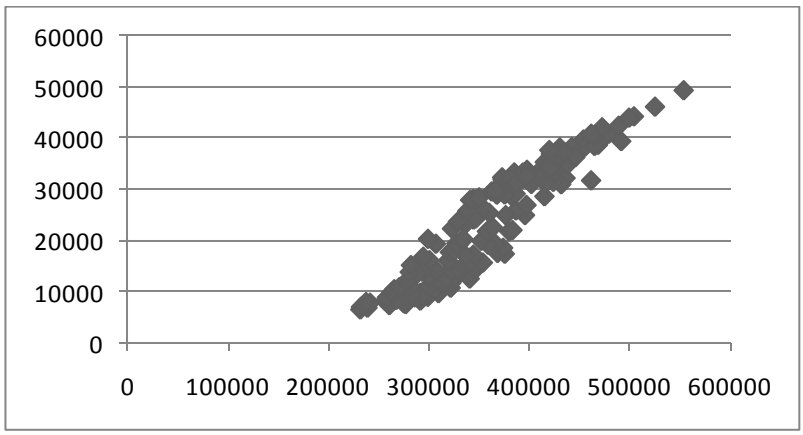


Figure 6.4: Pearson's correlation between two variates of Motel dataset

Here the length of window is set to six, and the first 166 observations are used as the training dataset and the rest are used for the testing. Figure 6.5 and 6.6 represents the dataset before and after normalization.

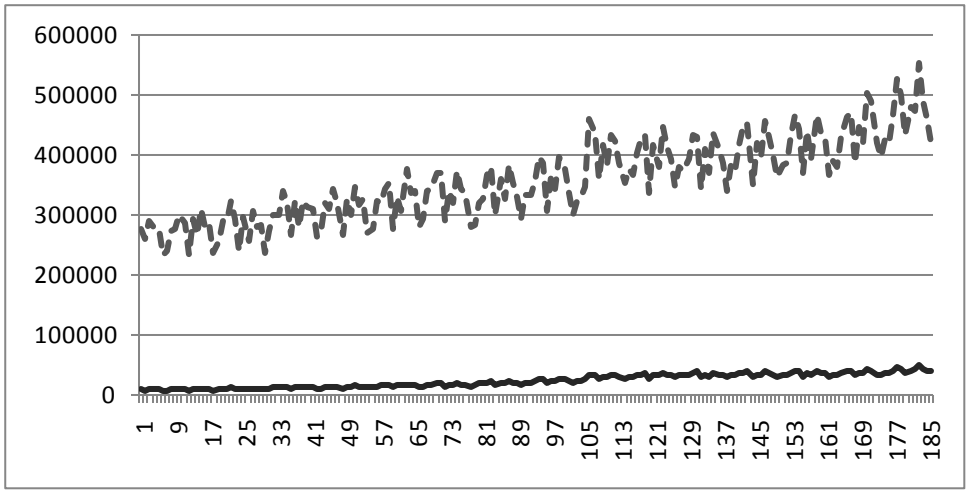


Figure 6.5: Original Motel dataset

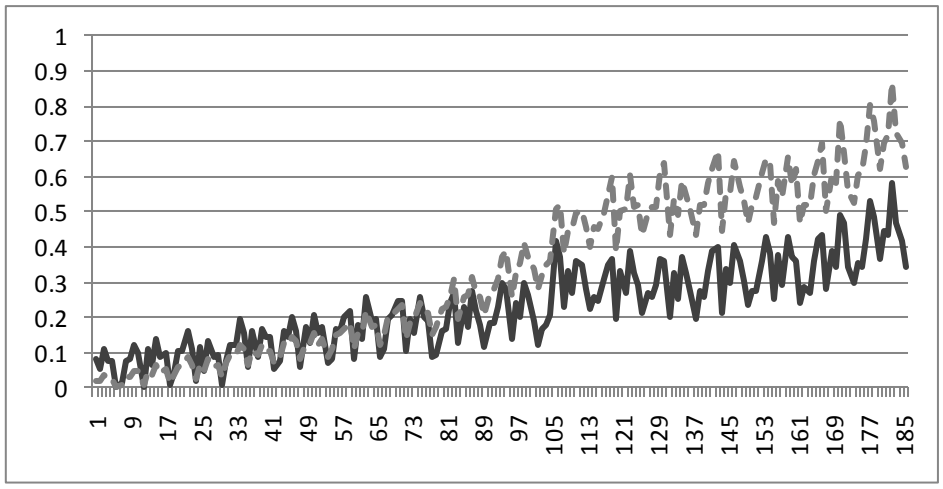


Figure 6.6: Normalized Motel dataset

We compared results of multivariate with univariate ANCFIS in Table 6.2 where individual variates and all together are compared. Also the training parameters for multivariate ANCFIS are represented in Table 6.1.

Variates	Input Length	Outputs	Input CFS per	Step Size	Increase Rate	Decrease Rate	Tmax	Lmax	Alpha	Weight	Tmin	Beta	M
2	6	2	2	0.001	1.2	0.8	100	2	0.99	0.95	0.01	0.98	400

Table 6.1: Training Parameters for the Motel Dataset

	Column 1	Column 2	Column 1 & 2
univariate system	0.314275	0.113525	0.146618
multivariate system	0.410481	0.109129	0.162816

Table 6.2: NMSE testing error comparison for Motel

Further analysis of Multivariate ANCFIS results is provided in Figure 6.7 where the actual output is plotted versus predicted outputs and prediction errors in Figure 6.8.

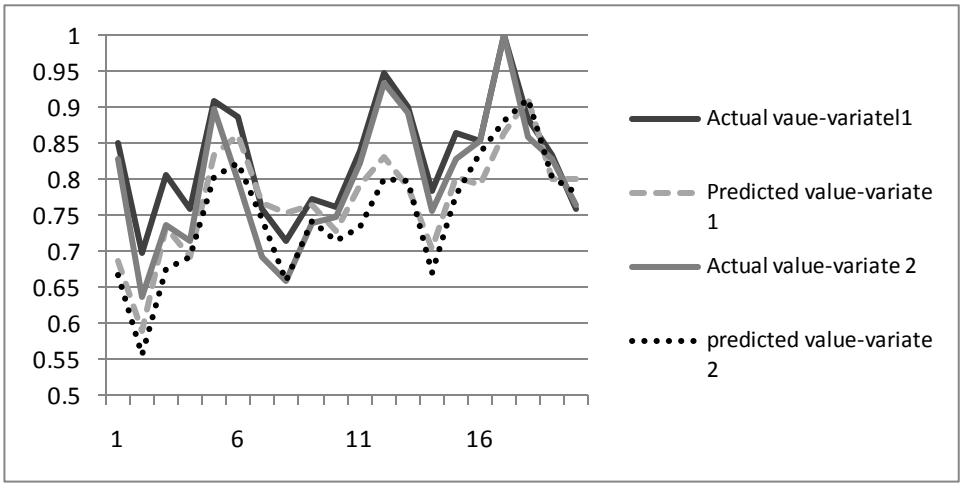


Figure 6.7: Multivariate test results for one-step prediction of Motel dataset

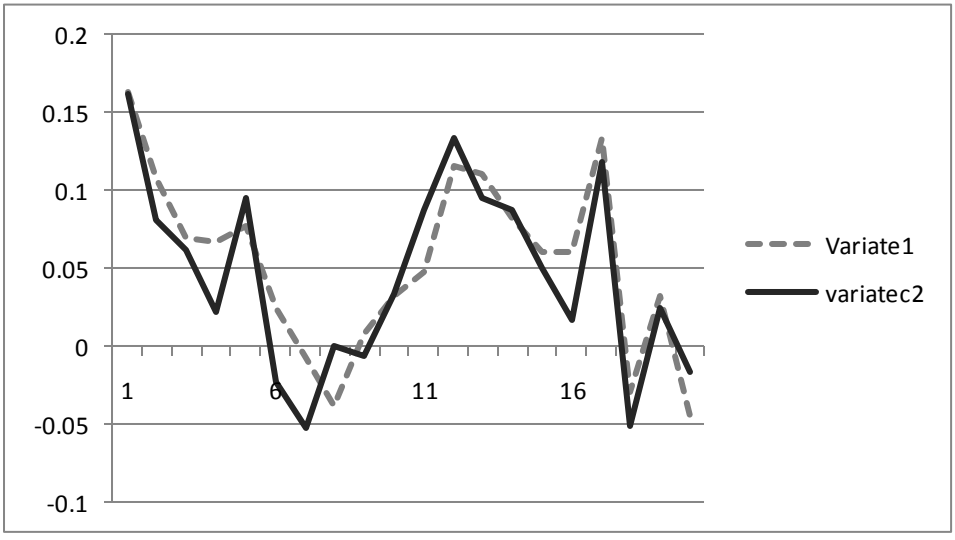
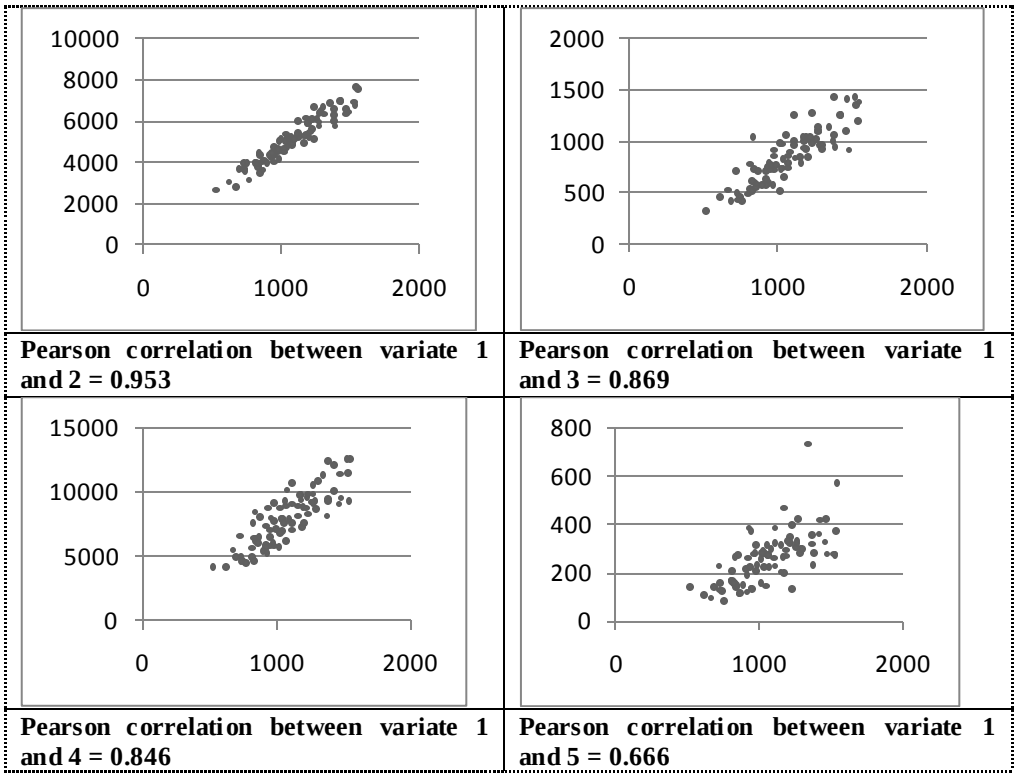


Figure 6.8: Multivariate system prediction errors for Motel dataset

6.2.2 Hydrology- River flow

This dataset is originally from Pablo Baldazo [22][24]. The tabulation of 6 unregulated (natural) annual river flow series is represented. Here the first River is Snake River near Moran, Wyoming. Second river is Snake River near Heise, Idaho. The third and fourth rivers are Boise River near Twin Springs and Salmon River near Whitebird, Idaho. The last river is Bruneau River near Hot Springs, Idaho. The data is collected from 1912-1994. The Pearson's correlations between different variates are presented in Figure 6.9. Variate one, two, three and four are strongly correlated to each other, though their correlation with variate five is not as strong as the other variates.



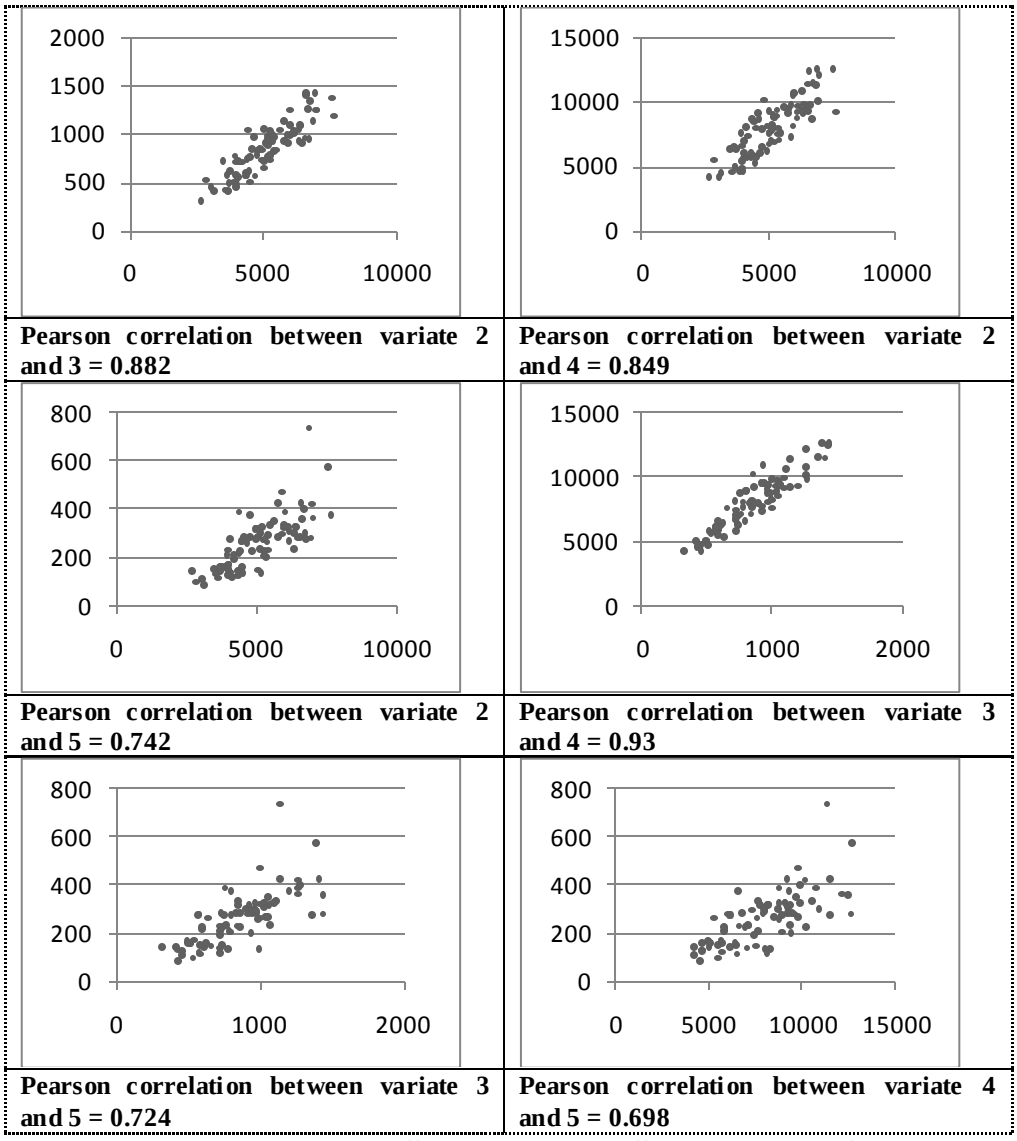


Figure 6.9: Pearson correlation efficient between different variates of River dataset

Here the length of window is considered eight and the first 73 is as training set and the rest as testing. The original and normalize datasets are plotted in Figure 6.10 and Figure 6.11.

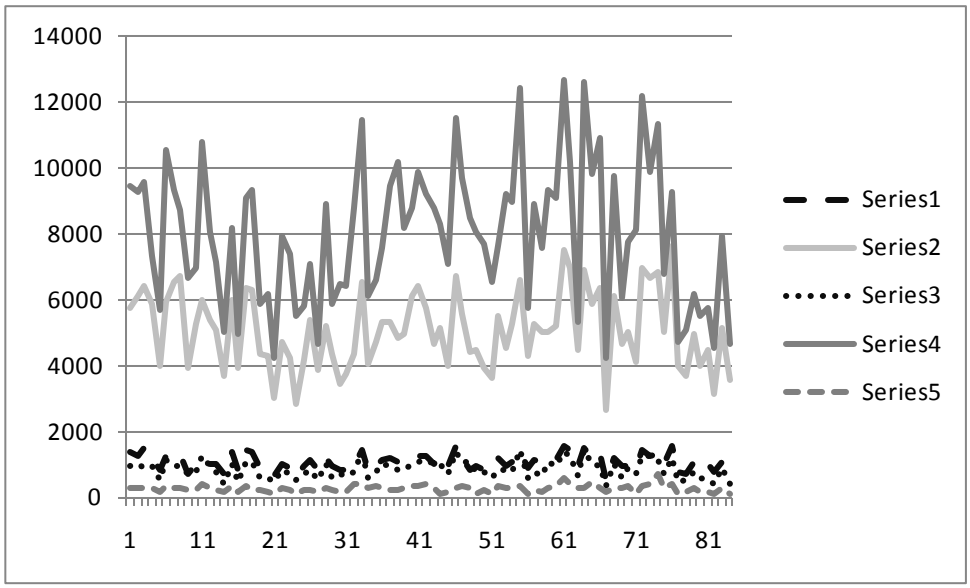


Figure 6.10: Original River-flow dataset

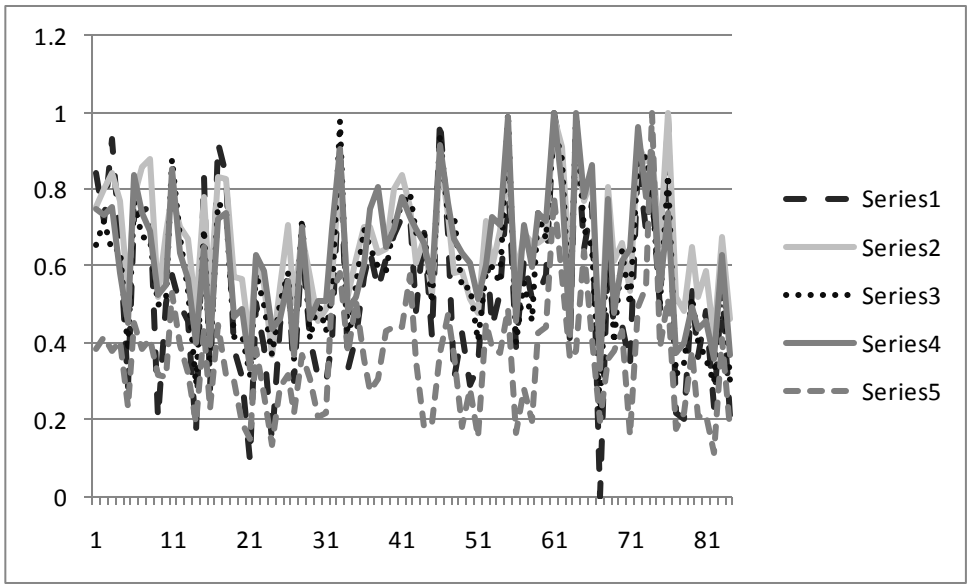


Figure 6.11: Normalized River-flow dataset

Tables 6.4, 6.6, 6.8 and 6.10 compare the NMSE measurements for two, three, four and all five variates separately and all together, calculated with univariate and multivariate ANCFIS as described before. Also, their training parameters are presented in Tables 6.3, 6.5, 6.7 and 6.9.

Variates	Input Length	Outputs	CFS per Input	Step Size	Rate Increase	e Rate Decreas	Tmax	Lmax	Alpha	Weight	Tmin	Beta	M
2	8	2	2	0.001	1.1	0.9	100	2	0.99	0.95	0.01	0.98	400

Table 6.3: Training Parameters for two variates of the River Dataset

	Column 1	Column 2	Column 1 & 2
univariate system	0.3048	0.3219	0.3059
multivariate system	0.5675	0.4636	0.5065

Table 6.4: NMSE testing error comparison for two variates of the River Dataset

Variates	Input Length	Outputs	CFS per Input	Step Size	Rate Increase	e Rate Decreas	Tmax	Lmax	Alpha	Weight	Tmin	Beta	M
3	8	3	2	0.001	1.1	0.8	100	2	0.99	0.95	0.01	0.98	400

Table 6.5: Training Parameters for three variates of the River Dataset

	Column 1	Column 2	Column 3	Column 1 & 2 &3
univariate system	0.3048	0.3219	1.4632	0.3960
multivariate system	0.46319	0.4577	0.4101	0.4344

Table 6.6: NMSE testing error comparison for three variates of the River Dataset

Variates	Input Length	Outputs	CFS per Input	Step Size	Rate Increase	e Rate Decreas	Tmax	Lmax	Alpha	Weight	Tmin	Beta	M
4	8	4	2	0.001	1.1	0.9	100	2	0.99	0.95	0.01	0.98	400

Table 6.7: Training Parameters for four variates of the River Dataset

	Column 1	Column 2	Column 3	Column 4	Column 1 & 2 &3&4
univariate system	0.3048	0.3219	1.4632	0.4526	0.4144
multivariate system	0.7550	0.5834	1.0601	0.4760	0.7197

Table 6.8: NMSE testing error comparison for four variates of the River Dataset

Variates	Input Length	Outputs	CFS per Input	Step Size	Rate Increase	e Rate Decreas	Tmax	Lmax	Alpha	Weight	Tmin	Beta	M
5	8	5	2	0.001	1.1	0.8	100	2	0.99	0.95	0.01	0.98	400

Table 6.9: Training Parameters for five variates of the River Dataset

	Column 1	Column 2	Column 3	Column 4	Column 5	All 5 columns
univariate system	0.3048	0.3219	1.4632	0.4526	0.4246	0.4311
multivariate system	0.6754	1.4512	1.9426	1.6309	2.5974	1.3111

Table 6.10: NMSE testing error comparison for five variates of the River Dataset

Again, further analysis of the ANCFIS results are provided in Figure 6.12 where we plot actual versus predicted outputs and then, prediction errors in Figure 6.13.

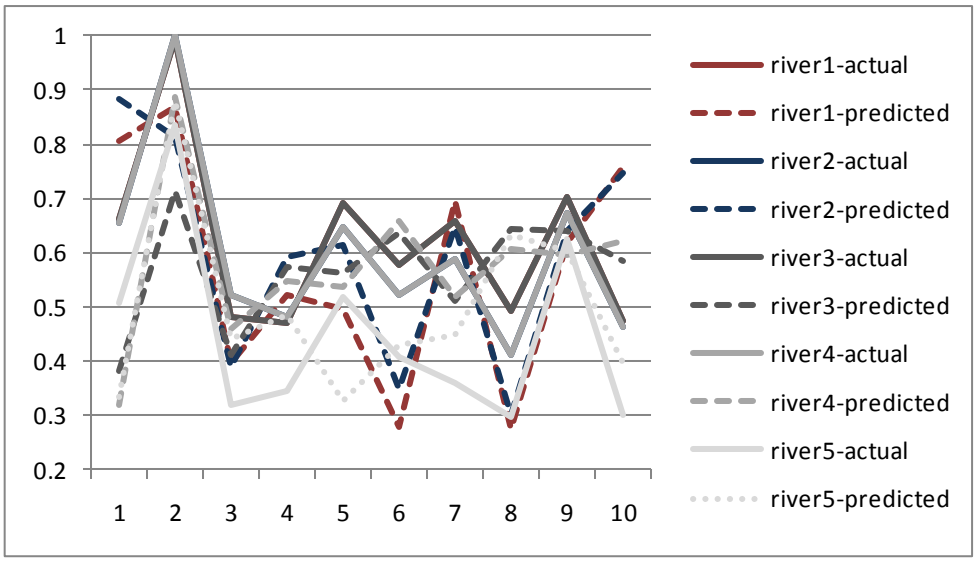


Figure 6.12: Multivariate test results for one-step prediction of River-flow dataset

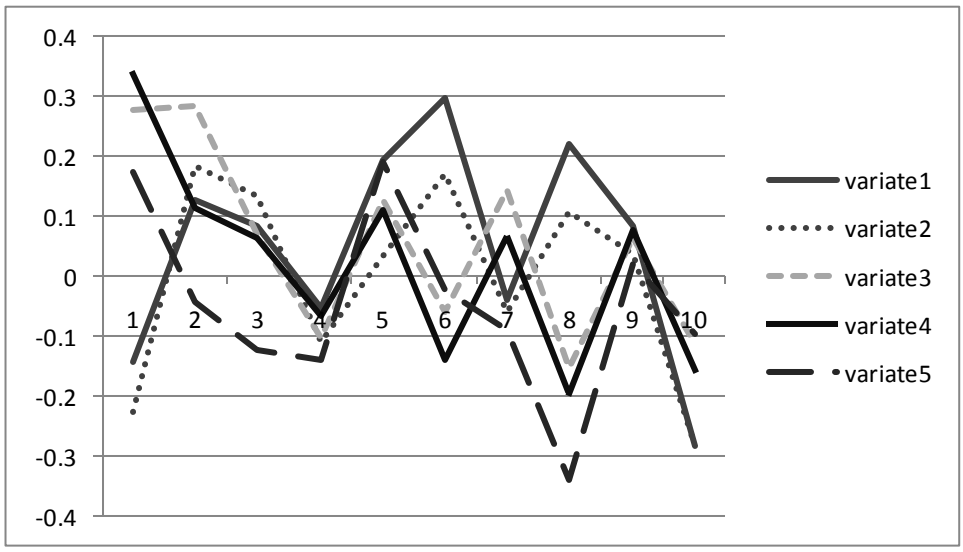


Figure 6.13: Multivariate system prediction errors for River-flow dataset

6.2.3 Macro Economics

This dataset is collected by Australian Bureau of Statistics [22]. It contains 141 observations with three variates during March 1956 to March 1991. The first variate is related to quarterly average weekly male earnings in Australia for all industries. The second variate is related to CPI for same quarters and the last variate is real average weekly male earnings. Here the length of window is considered four and the first 127 data points are considered as training data and the rest as testing data. The Pearson's correlation between the first two variates is 0.998 (which means they are highly correlated), between the first and third variate is 0.819 (still strong), and between the second and third variate is 0.794, (moderately strong). Figure 6.13 shows these correlation plots.

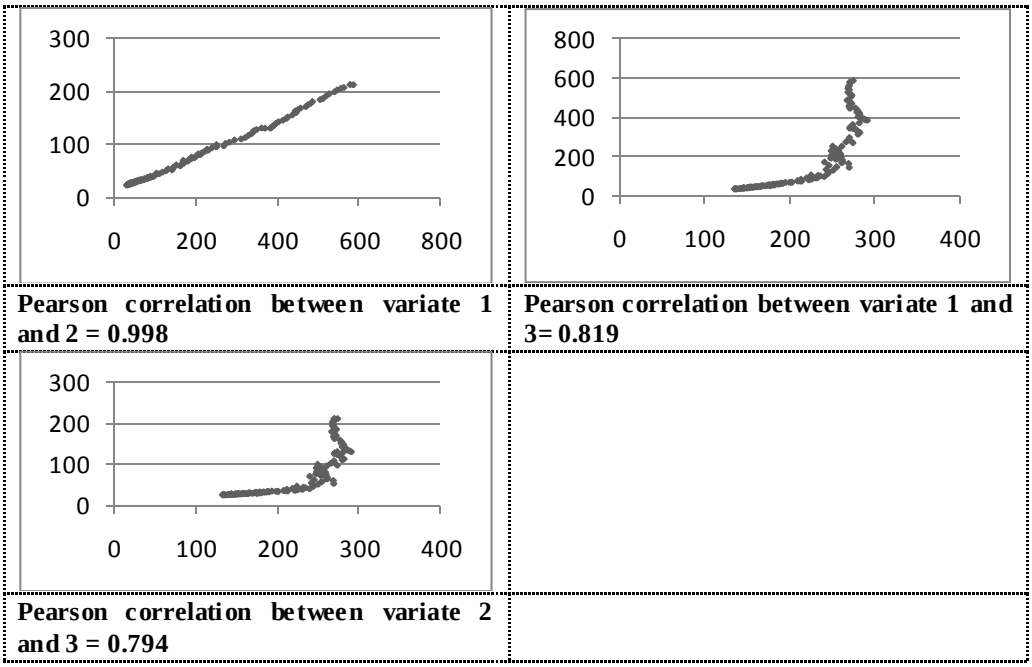


Figure 6.14: Pearson correlation efficient between different variates of Macro dataset

Figure 6.15 and Figure 6.16 show the Macro Economics original and normalized dataset.

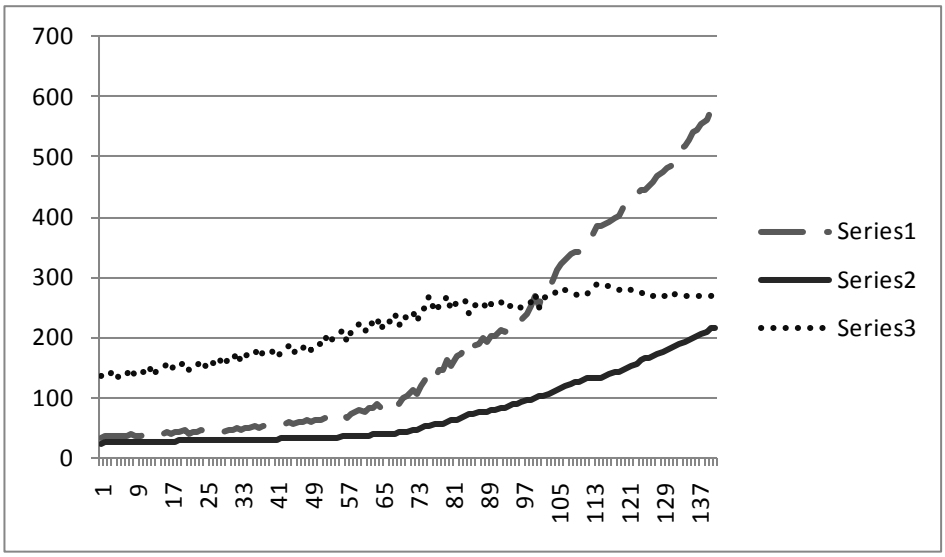


Figure 6.15: Original Macro dataset

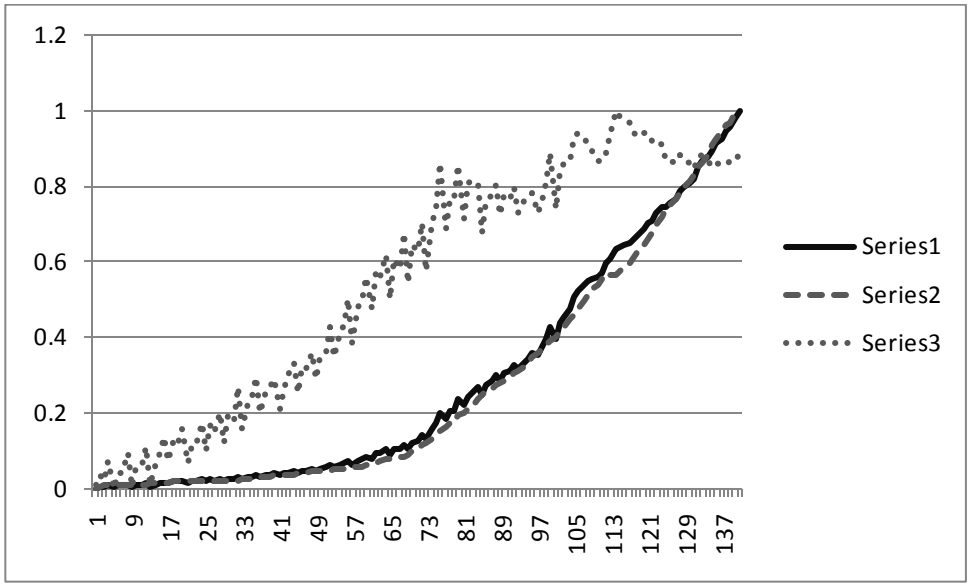


Figure 6.16: Normalized Macro dataset

All training parameters for the two and three variates are represented in Table 6.11 and 6.13. Table 6.12 and 6.14 contain the forecasting results for two variates and three variates, separately and all together.

Variates	Input Length	Outputs	CFS per Input	Step Size	Rate Increase	Decrease Rate	Tmax	Lmax	Alpha	Weight	Tmin	Beta	M
2	4	2	2	0.001	1.1	0.75	100	2	0.99	0.95	0.01	0.98	400

Table 6.11: Training Parameters for two variates of the Macro Dataset

	Column 1	Column 2	Column 1 & 2
Univariate system	0.00079	0.00069	0.0149
Multivariate system	0.4143	0.4473	0.432

Table 6.12: NMSE testing error comparison for two variates of the Macro dataset

Variates	Input Length	Outputs	CFS per Input	Step Size	Rate Increase	Decrease Rate	Tmax	Lmax	Alpha	Weight	Tmin	Beta	M
3	4	3	2	0.001	1.1	0.75	100	2	0.99	0.95	0.01	0.98	400

Table 6.13: Training Parameters for three variates of the Macro Dataset

	Column 1	Column 2	Column 3	Column 1 & 2 &3
Univariate system	0.00079	0.00069	0.0018	0.0029
Multivariate system	1.558	1.419	1.441	1.2543

Table 6.14: NMSE testing error comparison for three variates of Macro dataset

More analysis of the ANCFIS results are shown in Figure 6.17; actual versus predicted outputs and Figure 6.18; prediction errors.

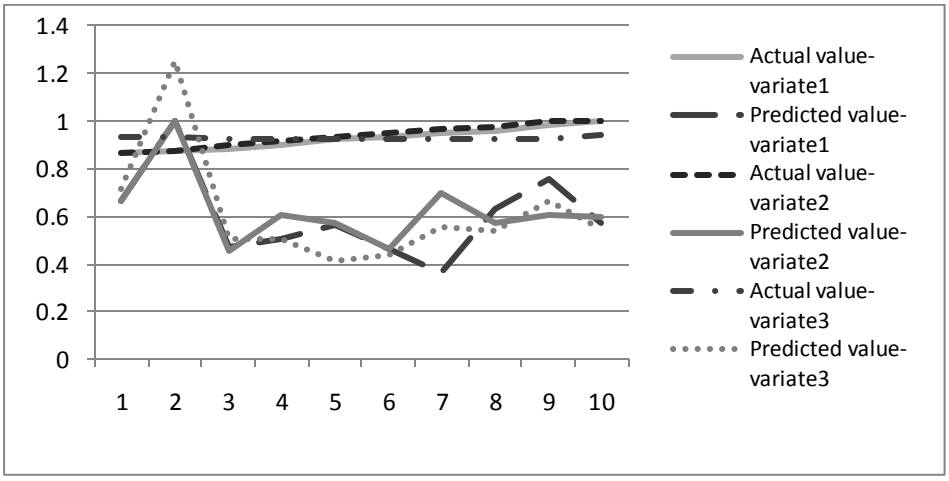


Figure 6.17: Multivariate test results for one-step prediction of MacroEconomics dataset

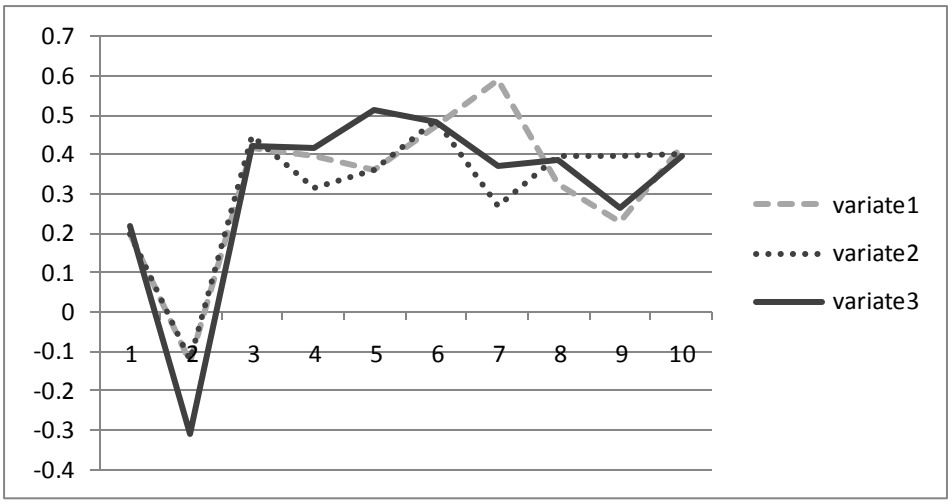


Figure 6.18: Multivariate system prediction errors for MacroEconomic dataset

6.2.4 Car road accident

This dataset is total number of annual car road accident casualties in Belgium from 1974 to 2004. Here we have five variates; the first one is number of killed persons, second one is mortally wounded, third variate represents the number of people died within thirty days, fourth one is severely wounded and finally the last variates indicates light casualties [25]-[27]. Figure 6.19 represents Pearson's correlation between different variates. It shows that among all variates, variates 2 & 3 have the highest correlation. On the other hand, variates 2 & 5 and 3 & 5 have the lowest correlation among the other variates.

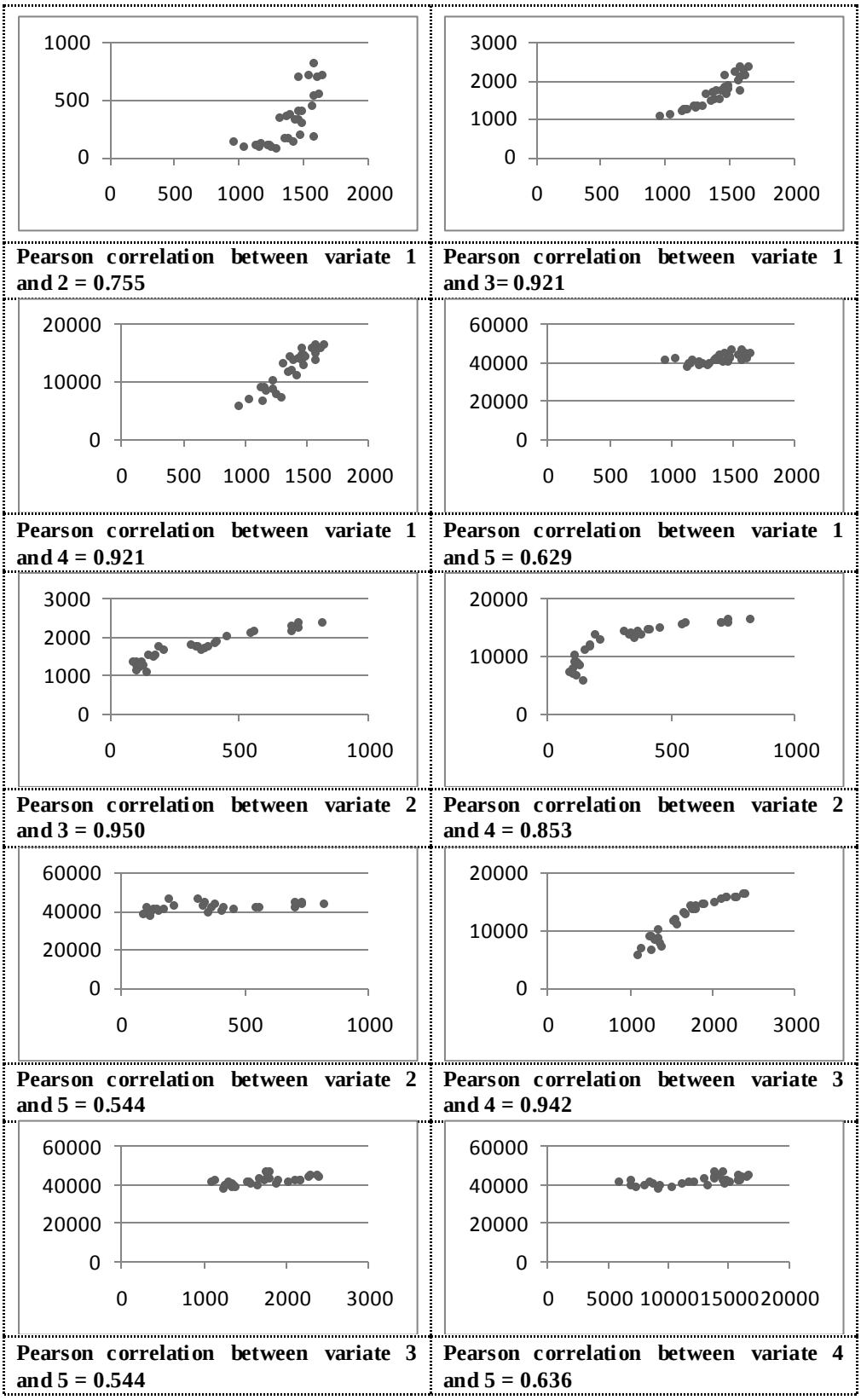


Figure 6.19: Pearson correlation efficient between different variates of Car Accident dataset

Here the length of input vector is 8 and the first 19 is considered as training dataset and the rest as testing dataset. Figure 6.20 and 6.21 present Car accident original and normalized datasets.

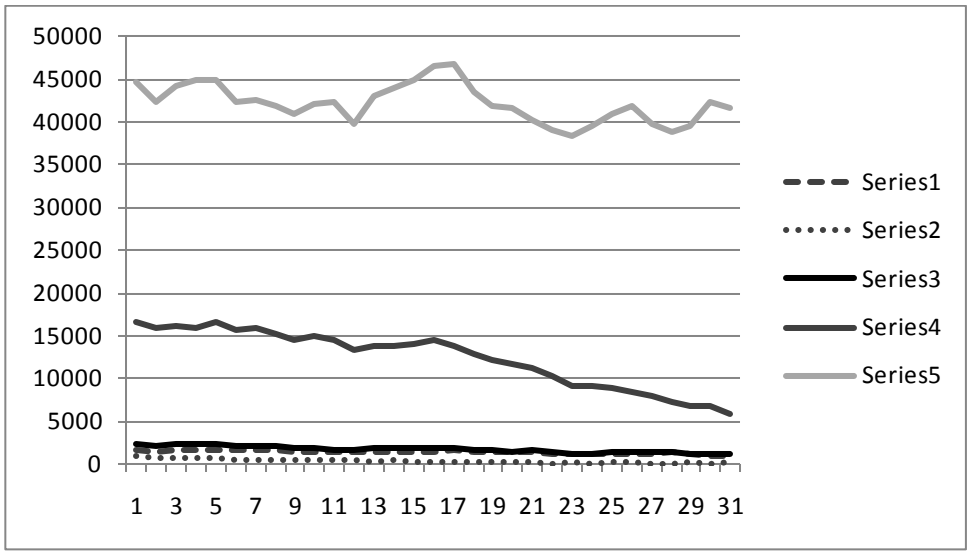


Figure 6.20: Original Car Accident dataset

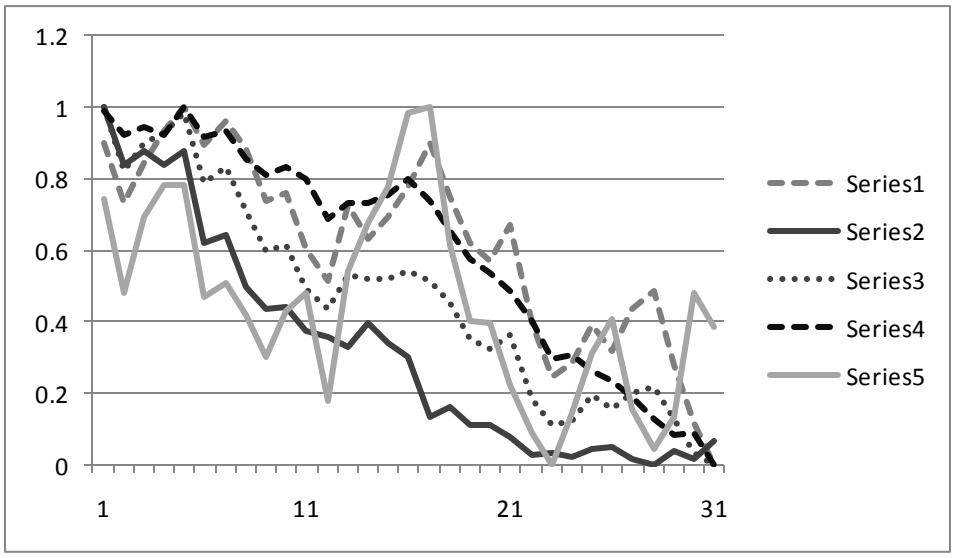


Figure 6.21: Normalized Car Accident dataset

Also the training parameters and Multivariate forecasting results are represented in Table 6.15 and 6.16.

Variates	Input Length	Outputs	CFS per Input	Step Size	Increase Rate	Decrease Rate	Tmax	Lmax	Alpha	Weight	Tmin	Beta	M
5	8	5	2	0.001	1.2	0.6	100	2	0.99	0.95	0.01	0.98	400

Table 6.15: Training Parameters for the Car Accident dataset

	Column 1	Column 2	Column 3	Column 4	Column 5	All columns
Univariate system	0.278	0.4753	0.024	0.056	0.775	1.09
Multivariate system	0.1007	0.500	0.047	0.1689	0.3177	2.03

Table 6.16: Testing error comparison for Car Accident dataset

The error measurements in [25] are AFER and MSE (for non-normalized dataset). AFER error measurement is defined as:

$$AFER = (\sum_{i=1}^{n} |(y_i - \hat{y}_i) / y_i| / n) \times 100 \qquad (6.37)$$

where y_i is the desired output, $\hat{y}_i$ is the estimated output and n is the total number of examples in the testing dataset. AFER cannot be used with normalized data; as one of the normalized values becomes zero due to normalization, leading to a divide-by-0 error. Knowing that ANCFIS system is designed in a way to work with normalized data, first the normalized dataset is used in order to find the prediction values. Then, both actual and predicted values are un-normalized. Table 6.17 reflects the results regarding to AFER and table 6.18 presents the AFER error comparison between multivariate ANCFIS and other methods in [96]. As it is shown, multivariate ANCFIS result is not comparable to any of them.

	Column 1	Column2	Column 3	Column 4	Column 5	All columns
AFER	2.70	13.07	1.55	8.5	1.73	5.52

Table 6.17: AFER testing error for the Car Accident dataset

	AFER
Multivariate ANCFIS	5.52
Jilani and Burney –method 1 (2008) [26]	2.6951
Jilani and Burney –method 2 (2008)[26]	5.244
Lee et al. (2006) [27]	5.248
Egrioglu et al. (2009) [25]	2.1715

Table 6.18: AFER error comparison for the Car Accident dataset

Further analysis of multivariate ANCFIS is provided in Figure 6.22 and Figure 6.23. Again Figure 6.22 shows the actual versus predicted results for all variates and Figure 6.23 is plotted prediction errors.

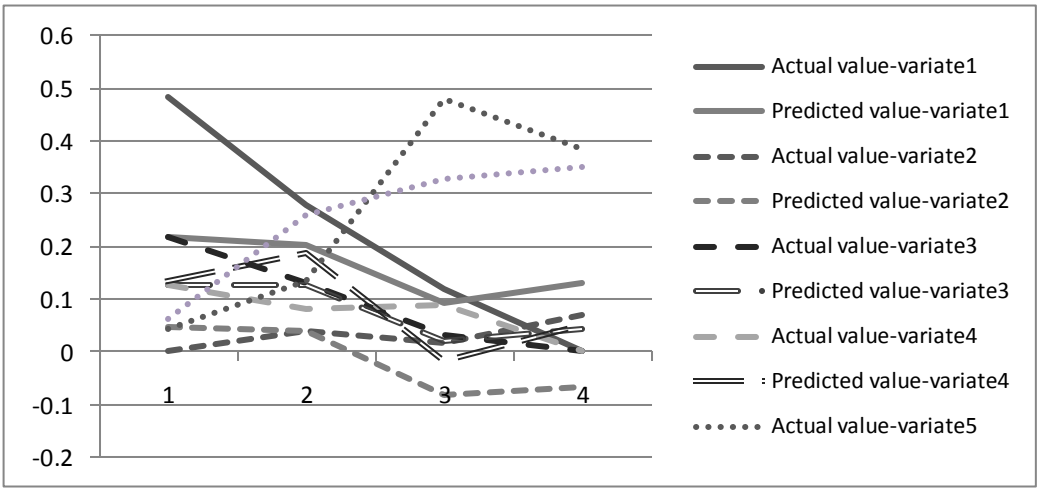


Figure 6.22: Multivariate test results for one-step prediction of Car Accident dataset

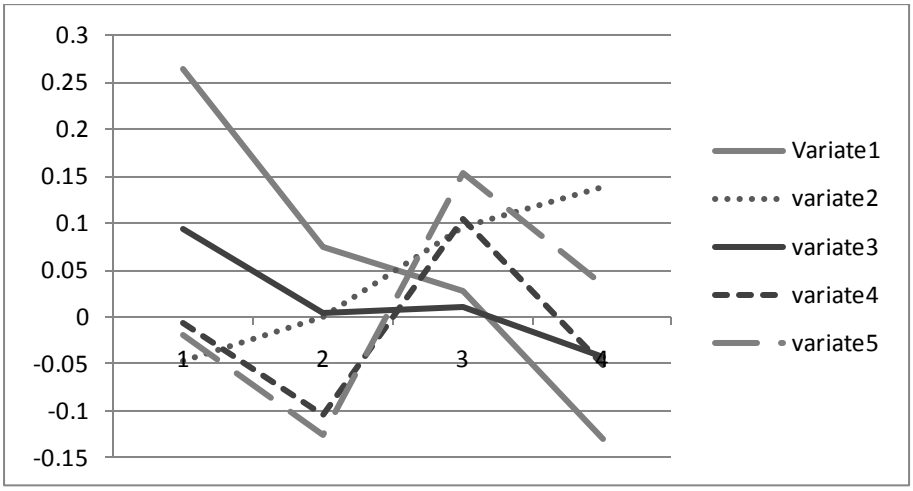


Figure 6.23: Multivariate system prediction errors for Car Accident dataset

6.3 Discussion and Conclusion

In our four experimental contrasts, we realized that multivariate ANCFIS does not work properly. Univariate ANCFIS was consistently superior on all datasets, despite having strong linear correlations between the variates. One of the possible problems that multivariate ANCFIS architecture can be related is in choosing the proper complex fuzzy conjunction for layer two. Here we used algebraic product; this may be an ineffective choice despite its mathematical appeal. Another possible problem can be related to layer four which is related to rule interference; dot product may not work for multiple inputs.

Future work on this area can be focused on exploring different complex fuzzy logic conjunctions for layer two. As possible candidates of complex fuzzy conjunctions, we can average the magnitudes and use weighted average for phase, or even we can compare the magnitudes. Definitely, this needs both theoretical and experimental work. Also, we have to ask how rule interference will be implemented; having the weight of a given rule depend on the weights of other rules is somewhat unique and not yet well-understood. An exploration on layer four to find other options for rule interference is needed.

Chapter 7: Summary and Future Work

In this thesis, we completed a performance evaluation for ANCFIS on five real-world time series datasets. Also, ANCFIS is compared to published forecasting results on all five datasets, and achieves results which in most of the cases are superior to the literature. In particular, ANCFIS performed the best on the chaotic Mackey-Glass time series. The ANCFIS network obtained this performance with no more than three rules per dataset. It leads us to a key finding that ANCFIS is a viable forecasting algorithm.

We have also developed an online version of ANCFIS for those cases when the data becomes available incrementally. Down-hill-simplex and recursive-least-square are employed to determine the updates for layer one and layer five parameters. The performance evaluation for online ANCFIS is done on two time series dataset. Also, the forecasting results are compared to the literature and offline ANCFIS. The results are comparable to offline ANCFIS and generally superior to published results found in literature.

A multivariate ANCFIS architecture is also developed and evaluated. Algebraic product is applied for layer two and also the node functions in layer 5 are extended for multiple variates. The performance evaluation for multivariate ANCFIS is done on four different multivariate time series datasets. Usually, having multiple variates lead us to better results as a result of interaction between them, but our results were significantly worse than the univariate ANCFIS results. This may be caused by layer two complex fuzzy conjunctions (algebraic product) or even by layer four rule interferences (dot product).

Future work in this area will concentrate on exploring multivariate time series forecasting; what other complex fuzzy conjunction operators work best in Layer 2? As little is currently known about complex fuzzy conjunctions, this work will inform theoretical investigations of complex fuzzy logic as well as practical time-series forecasting problems. Also, we have to ask how rule interference is

implemented; dot-product might not be the right choice. In addition, we will investigate linguistic interpretations of complex fuzzy rules. At this point in time, it is not at all clear how to interpret a complex-valued membership function – especially a sinusoidal one. Traditional linguistic interpretations of fuzzy rules assume convex type-1 or type-2 membership functions rather than sinusoids (or any other multi-modal function). Finally there are a huge number of neuro-fuzzy systems and fuzzy neural network architectures that can be extended to use complex fuzzy sets when modeling the phenomenon of regularity.

References

[1] S. Dick, "Towards Complex Fuzzy Logic," *IEEE Transactions on fuzzy systems*, vol.13, pp. 405 – 414, 2005.

[2] D. Ramot, R. Milo, M. Friedman, A. Kandel, " Complex Fuzzy Sets," *IEEE Transactions on Fuzzy Systems*, vol. 10, no. 2, pp. 171- 186 , 2002

[3] D. Ramot, M. Friedman, G. Langholz, and A. Kandel, "Complex Fuzzy Logic*," IEEE Transactions on Fuzzy Systems*, vol. 11, pp. 450 – 461, 2003.

[4] J.Y. man, Z. Chen, S. Dick, "Towards Inductive Learning of Complex Fuzzy Inference Systems," *IEEE Fuzzy Information Processing Society*, pp. 415-420, June 2007

[5] R. Jang, C.T. Sun, E. Mizutani, *Neuro-Fuzzy and soft computing A computational Approach to Learning and Machine Intelligence*, Prentice Hall, 1996.

[6] R. Jang, " Neuro-Fuzzy Modeling for Dynamic System Identification," *in Proceedings of the 1996 Asian Soft Computing in Intelligence Systems and Information Processing*, pp.320-325, 1996.

[7] R. Jang, "ANFIS: Adaptive-Network-Based Fuzzy Inference System," *IEEE Transactions on Systems, Man and Cybernetic*, vol. 23, pp.665-685, 1993.

[8] Z. Chen, S. Aghakhani, J. Man, S. Dick," ANCFIS: A Neuro-Fuzzy Architecture Employing Complex Fuzzy Sets," *submitted to IEEE Transactions on Fuzzy Systems*, August 2009.

[9] R. Jang, C.T. Sun, "Predicting chaotic time series with fuzzy if-then rules", *Proc. Of IEEE international conference of fuzzy systems*, San Francisco, 1993.

[10] R. Jang,"ANFIS: Adaptive-network-based fuzzy inference systems", *IEEE Trans. On Systems, Man, and Cybernetics*, pp. 665-685, 1993.

[11]U. Huebner, W. Klische, N. B. Abraham, and C. O. Weiss: "On problems encountered with dimension calculations." *Measures of Complexity and Chaos*; Ed. by N. B. Abraham et. al., Plenum Press, pp. 133, 1989.

[12]U. Huebner, W. Klische, N. B. Abraham, and C. O. Weiss: "Comparison of Lorenz-like laser behavior with the Lorenz model." *Coherence and Quantum Optics VI*; Ed. by J. Eberly et. al., Plenum Press, pp. 517, 1989.

[13]A. Weigend, "The Sante Fe Time Series Conpetition Data," http://wwwpsych.stanford.edu/~andreas/Time-Series/SantaFe.html

[14] U. Huebner, N. B. Abraham, and C. O. Weiss: ``Dimensions and entropies of chaotic intensity pulsations in a single-mode far-infrared NH3 laser.", *Phys. Rev*. ,pp. 6354, 1989.

[15]H. H. Sargent, "A prediction for the next sunspot cycle," *in Proc. 28th IEEE Vehicular Technology Conference*, vol. 28, pp. 490 – 496, 1978.

[16]T. Nobuhiko, H. van Dijk, "Combined forecasts from linear and nonlinear time series models," *International Journal of Forecasting*, vol. 18, pp. 421- 438, 2002.

[17]Y.F. Deng, X. Jin, Y. X. Zhong, "Ensemble SVR for prediction of time series ", *IEEE proceeding of the fourth international conference on machine learning and cybernetics*, vol. 6,pp. 3528-3534, 2005.

[18]J.T. Tsai, J.H. Chou, T.K. Liu, "Tuning the structure and parameters of a neural network by using hybrid Tanguchi-Genetic algorithm", *IEEE Transaction on neural networks*, vol. 14pp. 79-88, 2003.

[19]J.T. Tsai, J.H. Chou, T.K. Liu, "Tuning the structure and parameters of a neural network by using hybrid Tanguchi-Genetic algorithm ", *IEEE transaction on neural networks*, vol. 17, pp. 69-80, 2006.

[20]L. J. Cao, "Support vector machines experts for time series forecasting ", *Neurocomputing*, pp. 321-339, 2003.

[21]D. Graves, W. Pedrycz, "Fuzzy prediction architecture using recurrent NN", *Neuro-computing* , vol. 72 , pp. 1668-1678, 2009.

[22]R. J. Hyndman, "Time series data library," http://www-personal.buseco.monash.edu.au/~hyndman/TSDL/

[23]T. J. Cholewo, J. M. Zurada, "Sequential network construction for time seriesprediction*," in Proc. International Conference on Neural Networks 1997*, vol. 4, pp. 2034 – 2038, 1997.

[24]A. C. Robertson, R. J. Sutter, P. J. Baldazo, R. H. Lutz, "Stream flows in the Snake river basin 1989 conditions of use and management", *Idaho department of water resources Boise*, Idaho, June 1989.

[25]E. Egrioglu, C. H. Aladag, U. Yolcu, V. R. Uslu, M. A. Basaran, "A new approach based on artificial neural networks for high order multivariate fuzzy time series", *Expert Systems with Applications*, 2009.

[26]T. A. Jilani, S. M. A. Burney, "Multivariate stochastic fuzzy forecasting models", *Expert Systems with Applications*, pp. 691–700, 2008.

[27]L.W. Lee, L.H. Wang, S.M. Chen, Y.H. Leu , "Handling forecasting problems based on two factors high order fuzzy time series", *IEEE Transaction on Fuzzy systems*, pp. 468-477, 2006.

[28]L. A. Zadeh, "Fuzzy sets", *Inform. Control*, vol. 8, pp.808-821, 1965.

[29]G. J. Klir, and B. Juan, *Fuzzy sets Fuzzy logic and Fuzzy systems: selected papers by Lotfi A. Zadeh*, World scientific Pub Co Inc, 1996.

[30]D. Wang, L. Acar, **"**An analysis of type-1 and type-2 fuzzy logic systems," in *Proc.1999, IEEE International Symposium on Intelligent Control/Intelligent Systems and Semiotics*, pp.353-358, 1999.

[31]W. Pedrycz, F. Gomide, *Fuzzy systems engineering: Toward human-centric computing*, Wiley-IEEE Press, 2007.

[32]H. Hagnis, "Type-2 FLCs: A new generation of fuzzy controller," *IEEE Transaction on Computational Intelligence*, vol. 2, pp. 30-43, 2007.

[33]J. M. Mendel, *Uncertain Rule-Based Fuzzy Logic Systems*, Prentice-Hall, 2001.

[34] L. A. Zadeh, "Quantitative fuzzy semantics," *Information Sciences*, pp.159-176, 1971.

[35] C. Schmid, "Dynamics of multidisplicinary and controlled Systems", (http://www.esr.ruhr-uni-bochum.de/rt1/syscontrol/main.html).

[36]L. A. Zadeh, "Outline of a new approach to the analysis of the complex systems and decision processes," *IEEE Transactions on Systems, Man, and Cybernetics*, pp. 28-44, 1973.

[37]L. A. Zadeh, "The Concept of a Linguistic Variable and its Application to Approximate Reasoning—Parts I, II, III," *Information Sciences,* vol. 8, pp. 199- 249, 1975.

[38]S. Dick, A. Kandel, "Granular Computing in Neural Networks," in *Granular Computing: an emerging paradigm*, 2001.

[39]L. A. Zadeh , *Fuzzy Sets, Fuzzy Logic, Fuzzy Systems*, World Scientific Press, 1996.

[40]A. Kandel, *Fuzzy expert systems*, CRC press, 1993.

[41]N. N. Morsi and A.A. Fahmy, "On generalized modus ponens with multiple rules and a residuated implication", *Fuzzy Sets and Systems,* pp. 267- 274, 2002.

[42]M. Sugeno and G. T. Kang, "Structure identification of fuzzy model", *Fuzzy Sets and Systems*, pp. 15-33, 1988.

[43]T. Takagi and M. Sugeno, "Fuzzy identification of fuzzy systems and its applications to modeling and control", *IEEE transactions on Systems, Man, and Cybernetics*, pp. 116-132, 1985.

[44]B. Kosko, *Neural networks and fuzzy systems: a dynamical systems approach*, Prentice Hall, Upper Saddle River, NJ, 1991.

[45]E. H. Mamdani, "Application of fuzzy logic to approximate reasoning using linguistic synthetic synthesis," *IEEE Transaction on Computers*, vol. 26, pp.1182-1191, 1977.

[46]Y. Tsukamoto, "An approach to fuzzy reasoning method," *In Madan M. Gupta, Rammohan K, Ragade, and Ronald R. Yager, editors. Advances in Fuzzy Set Theory and Applications*, pp. 137- 149, 1979.

[47]E. H. Mamdani, S. Assilian, "An experiment in linguistic synthesis with a fuzzy logic controller", *International Journal of Man-machine studies*, pp. 419-435, 1990.

[48]Y. Tsukamoto, "An approach to fuzzy reasoning method", *Advances in fuzzy set theory and applications*, pp. 137- 149, 1979.

[49]S. Samarasinghe, *Neural Networks for Applied Sciences and Engineering: from fundamentals to complex pattern recognition,* Auerbach publication, 2007.

[50]F.M. Ham, and I. Kostanic, *Principle of Neurocomputing for Science & Engineering*, McGraw-Hill, 2001.

[51]J. J. Buckley, "Fuzzy complex numbers", *Fuzzy set and systems*, Vol.33, pp. 333-345, 1989.

[52]J.J. Buckley, "Fuzzy complex analysis I: Differentiation," *Fuzzy sets System* , vol.41, pp.269-284, 1991.

[53]J.J. Buckley, "Fuzzy complex analysis II: Integration," *Fuzzy sets Sysem.*, vol.49, pp.171-179, 1992.

[54]J. J. Buckley, Y. Qu, "Solving linear and quadratic fuzzy equations," *Fuzzy Sets System,* vol. 38, pp. 43–59, 1990.

[55]J. J. Buckley, Y. Qu, "Solving fuzzy equations: a new solution concept," *Fuzzy Sets System,* vol. 39, pp.291–301, 1991.

[56]D. Moses, O. Degani, H.-N. Teodorescu, M. Friedman, and A. Kandel, "Linguistic coordinate transformations for complex fuzzy sets," in *Proc. 1999 IEEE Int. Conf. Fuzzy Systems*, pp. 1340–1345, Korea, 1999.

[57]A. Kaufman and M. M. Gupta, *Introduction to Fuzzy Arithmetic*, New York: Van Nostrand Reinhold, 1985.

[58]. E. Michael, A. A. S. Awwal, and D. Rancour, " Artificial neural networks using complex numbers and phase encoded weights-electronic and optical implementations*," in Proc. Int. Joint C. Neural Networks*, pp.1213-1218, 2006.

[59]H. Akira, *Complex-Valued Neural Networks*, Springer, 2006.

[60]A.J. Noest, "Discrete-state phasor neural nets*," Physical Review A*, vol.38, pp. 2196-2199, 1988.

[61]H. Leung, and S. Haykin, "The Complex back propagation algorithm," *IEEE Trans. Signal Process*, vol. 39, pp.2101-2104, 1991.

[62]M.S. Kim, and C.C.Guest, "Modification of Backpropagation for complex-valued signal processing in frequency domain," *in Proc. Int. Conf. Neural Networks*, San Diego, CA, USA, vol. 3, pp. 27-31, 1990.

[63]A. Hirose, *Complex-Valued Neural networks*, Berlin, Germany: Springer-Verlag, 2006.

[64]A. Hirose, Y. Asano, and T. Hamano, "Developmental learning based on coherent neural networks with behavioural mode tuning by carrier-frequency modulation," *in Proc. Int. Joint C. Neural Networks*, pp. 8874-8881, 2006,.

[65]T. Nitta, "An analysis of the fundamental structure of complex-valued neurons," *Neural Processing Letters*, vol. 12, pp.239-246, 2000.

[66]T. Nitta, "On the inherent property of the decision of melodies by complex-valued network," *Neurocomputing*, vol. 50, pp.291-303, 2003.

[67]T. Kim and T. Adali, "Fully complex multi-layer perceptron network for nonlinear signal processing," *Journal of VLSI Signal Processing Systems for Signal Image and Video Technology*, vol. 32, pp. 29-43, 2002.

[68]T. Kim and T. Adali, "Approximation by fully complex multilayer perceptrons," *Neural Computation*, vol. 15, pp. 1641-1666, 2003.

[69]Y. Kuroe and Y. Taniguchi, "Models of self-correlation type complex-valued associative memories and their performance comparison," *in Proc. Int. Joint. C. Neural Networks*, pp. 605-609, 2006.

[70]I. Nishikawa, T. Iritani, and K. Sakakibara, "Improvements of the traffic signal control by complex-valued Hopfield network," *in Proc. Int. Joint. C. Neural Networks*, pp. 1186-1191, 2006.

[71]G. Rigatos, "Energy spectrum of quantum associative memories," *in Proc. Int. Joint. C. Neural Networks*, pp. 599-604, 2006.

[72]T. Nitta, "An extension of back propagation algorithm to quaternions," *in Proc. Int. C. Neural Information Processing*, pp. 247- 250, 1996.

[73]A. Malekzadeh-A, M. Akbarzadeh-T, " Complex-Valued Adaptive Neuro Fuzzy Inference System-CANFIS," in *Proc.2004, World Automation Congress*, Sevilee, Spain, vol. 17, pp. 447-482, 2004.

[74] Y. Li, Y. T. Jang, "Complex adaptive fuzzy inference systems," *Soft Computing in Intelligent Systems and Information Processing, Proceedings of the 1996 Asian*, pp. 551- 556, 1996.

[75] L. Chen and K. Aihara, "Chaotic simulated annealing by a neural network model with transient chaos," *Neural Networks*, vol. 8, pp. 915–930, 1995.

[76] R. Konnur, "Additive chaotic forcing scheme for determination of the global minimum of functions," *Communications in Nonlinear Science and Numerical Simulation,* vol. 9, pp. 499-513, 2004.

[77] J. Mingjun and T. Huanwen, "Application of chaos in simulated annealing," *Chaos, Solitons & Fractals,* vol. 21, pp. 933-941, 2004.

[78] G. M. Georgiou and C. Kutsougeras, "Complex Domain Backpropagation," *IEEE T. Circuits and Syst. II,* vol. 39, pp. 330-334, 1992.

[79] M. Gruber, *Regression Estimation a Comparative Stud*, Burlington, MA: academic Press, Inc., 1990.

[80] A. Lendasse, F. Corona, J. Hao, N. Reyhani, M. Verleysen, "Determination of the Mahalanobis matrix using nonparametric noise estimation", *European symposium on ANN Bruges*, April 2006.

[81] T.C. Hsia, *System identification: least-squares methods*, Lexington, MA, USA: D.C. Health and Company, 1977.

[82] H. Leung and H. S., "The complex back propagation algorithm," *IEEE T. Signal Proc.,* vol. 39, pp. 2101-2104, 1991.

[83] J.-S.R. Jang, C.-T. Sun, "Neuro-fuzzy modeling and control," *proceedings of IEEE*, Vol. 83, pp. 378-406. 1995

[84] R. S. Crowder, "Predicting the Mackey-Glass timeseries with cascade-correlation learning," in *Connectionist Models: Proceedings of the 1990 Summer School* D. S. Touretzky, J. L. Elman, and T. J. Sejnowski, Eds. SanFrancisco, CA, USA: Morgan Kaufmann, pp. 117-123, 1991.

[85] A. S. Weigend, N. A. Gershenfeld, "Results for time series prediction competition at the Santa Fe Institute", *IEEE international conference on Neural Networks*, Vol. 3, pp. 1786-1793 , 1993

[86]J. A. B. Tome and J. P. Carvalho, "One step ahead prediction using Fuzzy Boolean Neural Networks," in *Proc. Joint EUSFLAT - LFA Conf.*, Barcelona, Spain, 2005, pp. 500-505.

[87]F. A. Gers, D. Eck, and J. Schmidhuber, "Applying LSTM to time series predictable through time-window approaches," in *Proc. Int. C. Artificial Neural Networks*, Vienna, Austria, 2001, pp. 669-676.

[88]E. A. Wan, "Time series prediction by using a connectionist network with internal time delays," *Time Series Prediction: Forecasting the future and understanding the past*, pp. 195-217, 1994.

[89]T. Koskela, M. Varsta, J. Heikkonen, and K. Kaski, "Reccurent SOM with local linear models in time series prediction," *in 6th European Symposium on Artificial Neural Networks*. ESANN'98. Proceedings. D-Facto, Brussels, Belgium, pp. 167- 72, 1998.

[90]R. Bakker, J. C. Schouten, C. L. Giles, F. Takens, and C. M. van den Bleek, "Learning chaotic attractors by neural networks," *Neural Computation*, vol. 12, no. 10, 2000.

[91]T. Sauer, "Time series prediction using delay coordinate embedding," *in Time Series Prediction: Forecasting the Future and Understanding the past*, Addition-Wesley, 1994.

[92]A. S. Weigend and D. A. Nix, "Predictions with confidence intervals (local error bars)," *in proceedings of the International Conference on Neural Information Proceeding* (ICONIP'94), (Seoul, Korea), pp. 847-852, 1994.

[93]B. H. Bontempi G., Birattari M., "Local learning for iterated time-series prediction," *in Machine Learning: Proceedings of the sixteenth International Conference*, San Francisco, USA, pp. 32- 38,1999.

[94]J. Wu, M. Liu, "Improving generalization performance of artificial neural networks with genetic algorithms," *in Proc. IEEE International Conference on Granular Computing*, vol. 1, pp. 288-291, 2005.

[95]T. T. Nguyen, C. P. Willis, D. J. Paddon, H. S. Nguyen, "A hybrid system for learning sunspot recognition and classification," *in Proc. International*

Conference on Hybrid Information Technology, vol. 2, pp. 257 – 264, 2006.

[96] G. Dangelmayr, S. Gadaleta, D. Hundley, M. Kirby, "Time series prediction by estimating Markov probabilities through topology preserving maps", *proc. SPIE applications and science of NN, fuzzy sys. And evolutionary computing*, 1999.

[97] T. J. Cholewo, J. M. Zurada, "Sequential network construction for time series prediction", *IEEE International conference on NN*, pp. 2034-2038, 1997.

[98] H.Tong, K.S. Lim, "Threshold autoregressive, limit cyclical data", *J. Royal Statistical Society B*, vol. 42, pp. 245-292, 1980.

[99] A. S. Weigend, B. A. Huberman, D.E. Rumelhart, "Predicting the future: a connectionist approach", *Int. J. Neural Systems*, vol.1, pp. 193-209, 1990.

[100] X. Hong , J. Harris, "Experimental design and model construction algorithms for radial basis function networks", *International journal of system science*, vol. 34, pp. 733-745, 2003.

[101] J. Sum, K. Ho, *"On-line estimation of the final prediction error via recursive-least-square method*", Neurocomputing, pp. 2420-2424, 2006.

[102] L. Cohen, G. Avrahami, M. Last, A. Kandel, "Info-fuzzy algorithms for mining dynamic data stream", *applied soft computing*, pp. 1283-1294 ,2008

[103] S. Blazic, I. Skrjanc, S. Gerksic, G. Dolanc, S. Strmcnik, M. B. Hadjiski, A. Stathaki, "Online fuzzy identification for an intelligent controller based on a simple platform", *Engineering applications of artificial intelligence*, pp. 628-638, 2009

[104] H. Duan, X. Shao, W. Hou, G. He, Q. Zeng, "An incremental learning algorithm for Lagrangian support vector machines", *Pattern recognition letters*, pp. 1384-1391, 2009

[105] N. Y. Liang, G. B. huang, "A fast accurate online sequential learning algorithm for feedforward networks", *IEEE transaction on neural networks*, vol. 17, pp. 1411-1423, November 2006

[106] X. Deng, X. Wang, "Incremental learning of dynamic fuzzy neural networks for accurate system modeling", *fuzzy sets and systems*, pp. 972-987, 2009

[107] S. Okada, O. Hasegawa, "Incremental learning, recognition, and generation of time-series patterns based on self-organizing segmentation", *Journal of advanced computational intelligence and intelligent informatics*, vol. 10, 2006

[108] S. Haykin, *Adaptive Filter Theory*, Third ed. Englewood Cliffs, NJ: Prentice-Hall, 1996.

[109] T. Kailath, A. Sayed, and B. Hassibi, *Linear Estimation*. Englewood Cliffs, NJ: Prentice-Hall, 2000.

[110] C.S. Leung, K.W. Wong, P.F. Sum, L.W. Chan, "A pruning method for recursive least squared algorithm", *Neural Networks*, pp. 147-174, 2001.

[111] Bjorck A. (1996). *Numerical Methods for Least Squares Problems*, SIAM PA.

[112] Y. Engel, S. Mannor, R. Meir, "The Kernel Recursive Least-Square Algorithm*", IEEE transaction on signal processing,* vol. 52, August 2008

[113] S. Lewis and J. N. Hwang, "Recursive Least Squares Learning Algorithms for Neural Networks," *SPIE's 1990 Int'l Symposium on Optical and Optoelectronic Applied Science and Engr.*, pp. 28-39, 1990.

[114] G.C. Goodwin and K.S. Sin, *Adaptive Filtering, Prediction and Control*, Printice Hall, Englewood Cliffs, NJ (1984).

[115] J. A. Nedler, and R. Mead, "A simplex method for function minimization", *The Computer Journal*, pp. 308- 313, 1965.

[116] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, *Numerical Recipes in C++: The Art of Scientific Computing*, Pearson Education, 1992.

[117] H. Lütkepohl**,** *Introduction to Multiple Time Series Analysis*, Springer Verlag, New York, 1991.

[118] P. J. Brockwell, R.A. Davis, S.E. Feinberg, *Time series: Theory and methods,* Springer Series in Statistics, Springer-Verlag 1987.

[119] K. Chakraborty, K. Mehrotra, C.K. Mohan and Sanjay Ranka, "Forecasting the behaviour of multivariate time series using neural networks ", *Neural Networks*, vol. 5, pp. 961-970, 1992.

[120] H. Kantz and T. Schreiber, *Nonlinear Time Series Analysis*, Cambridge Nonlinear Science Series, Cambridge University press, 2003.

[121] H. Kantz and T. Schreiber, *Nonlinear Time Series Analysis*, 2nd ed., Cambridge Nonlinear Science Series, Cambridge University press, 2003.