PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA

MINISTRY OF HIGHER EDUCATION AND SCIENTIFIC

University Mohamed El-Bachir El-Ibrahimi - Bordj Bou

Institute of Electronics and Telecommunications

Dissertation

Submitted in Partial Fulfillment of the Requirement for an

ACADEMIC MASTER DEGREE

FIELD: Electronic

Specialty: Industrial Electronic

Realised by: Frik Meriem

Mokris Ichrak

Titled

Modeling of a Nonlinear System Using Fuzzy Neural Networks, Application to Concentration Identification in a Chemical Reactor

Evaluated on: /06/2026

By the evaluation committee composed of :

Last & First name	*Grade*	*Quality*	*Establishment*
Dr .Meguellati Sabrina	*MCB*	*President*	*Univ-BBA*
Dr. Djellal Djamel	*MCB*	*Examiner*	*Univ-BBA*
Dr. Samai Aib	*MCB*	*Supervisor*	*Univ-BBA*

2025/2026

Acknowledgements:

First and foremost, we thank our Almighty God for giving us the strength, patience, and courage to complete this work.

We would like to express our deepest gratitude to Dr. Samia Aib, our supervisor, for her availability, valuable advice, constant support, and the trust she placed in us throughout the completion of this work. Her wise guidance and scientific rigor have been a source of inspiration and have greatly contributed to the success of this thesis.

We would also like to extend our sincere thanks to Dr. Meguellati Sabrina, who honored us by presiding over the defense jury. We are grateful for the time she devoted to evaluating this work and for the interest she has shown in it.

Our thanks also go to Dr. Djellal Djamel, who agreed to examine this work in the capacity of examiner. We are grateful for his constructive remarks and the attention he kindly gave to our thesis.

Finally, we would like to thank all those who, directly or indirectly, contributed to the success of this work.

إهداء

إلى والديّ العزيزين، أحمد وخديجة، اللذين كانا سندي وقوتي طوال مشواري الدراسي، واللذين لم يبخلا عليّ بالدعم والدعاء والتشجيع في كل خطوة.

وإلى إخوتي وأصدقائي، وإلى صديقاتي العزيزات مريم وأمينة وردينة، اللواتي شاركنني لحظات التعب والفرح، وكنّ دائماً مصدر دعم ومحبة وابتسامة خففت عني صعوبات هذه المرحلة.

وإلى كل من أحب، وإلى كل من وقف إلى جانبي وآمن بي وبقدراتي، ولو بكلمة طيبة أو دعاء صادق، أهدي هذا العمل المتواضع عربون شكر وامتنان.

كما أتقدم بخالص الشكر والتقدير إلى أستاذتي سامية على توجيهاتها القيمة ونصائحها ومرافقتها لي خلال إنجاز هذا العمل.

إلى جميع هؤلاء أهدي ثمرة جهدي، راجيةً أن يكون هذا الإنجاز وفاءً لجزء بسيط مما قدموه لي من دعم ومحبة.

إشراق

الحمد لله الذي بنعمته تتم الصالحات، وبه نستعين وعليه التوكل

First and foremost, all praise and gratitude belong to Almighty **Allah**, by Whose grace all good things are accomplished.I lovingly dedicate this work to my beloved parents, whose endless support, prayers, and sacrifices have been my guiding light.To my dear siblings, and to all my fellow classmates and companions who walked this path with me and shared this journey.
I humbly offer this effort, praying that Almighty **Allah** accepts it and makes it useful and beneficial.

Mary

Abstract

In this work, we study the application of fuzzy neural networks to the identification of uncertain, nonlinear systems. This approach was investigated using two different architectures:

- ANFIS (Adaptive Network Fuzzy Inference System), based on the Takagi-Sugeno model.
- FLP (Fuzzy Logic Processor), based on the Mamdani model.

Each architecture is implemented through two methods: a series-parallel (one step ehead) model and a parallel (long range) model.

An example of simulation is used, identification of concentration in a chemical reactor to demonstrate the effectiveness of these new approaches and to validate the results obtained.

Keywords: Neural networks, fuzzy logic, neuro-fuzzy, identification.

Résumé

Dans ce travail, nous avons étudié l'application des réseaux de neurones flous à l'identification de systèmes non linéaires incertains. Cette approche a été examinée selon deux architectures différentes :

ANFIS (Adaptive Network Fuzzy Inference System), basé sur le modèle de Takagi-Sugeno. FLP (Fuzzy Logic Processor), basé sur le modèle de Mamdani.

Chaque architecture est mise en œuvre selon deux méthodes : un modèle série-parallèle (récurrent) et un modèle parallèle.

Un exemple de simulation est utilisé, l'identification de la concentration dans un réacteur chimique, pour démontrer l'efficacité de ces nouvelles approches et pour valider les résultats obtenus.

Mots-clés : Réseaux de neurones, logique floue, neuro-flou, identification.

ملخص

في هذا العمل، درسنا تطبيق الشبكات العصبية الضبابية (القمئية) في التعرف على الأنظمة غير الخطية غير المؤكدة. تم فحص هذا النهج وفقًا لبنيتين مختلفتين:

نظام الاستنتاج الضبابي القائم على الشبكة التكيفية (ANFIS): القائم على نموذج "تاكاجي-سوجينو" (Takagi-Sugeno).

معالج المنطق الضبابي (FLP): القائم على نموذج "ممداني" (Mamdani).

تمت قيد التنفيذ لكل بنية وفقًا لطريقتين: نموذج تسلسلي-متوازي (تكراري) ونموذج متوازي.

وقد استُخدم مثال محاكاة، وهو التعرف على التركيز في مفاعل كيميائي، لإثبات كفاءة هذه المقاربات الجديدة وللتحقق من صحة النتائج التي تم الحصول عليها.

الكلمات المفتاحية: الشبكات العصبية، المنطق الضبابي، العصبوني-الضبابي، التعرف (النمذجة).

Chapter 1: 3
Artificial Neural Network Theory and Fuzzy Logic 3
1.1. Introduction 4
1.2. Artificial neural networks 4
1.2.1. Introduction 4
1.2.2. State of the art of ANN 4
1.2.3. The formal neuron: 5
1.2.4. Definition of Artificial Neural Networks (ANN): 8
1.2.5. Architecture of Neural Networks 8
1.2.6. Training of Neural Networks 10
1.2.7. Back-propagation method 11
1.2.8. Identification by Neural Networks 13
1.2.9. Neural network control methods 14
a) Supervised control 14
b) Direct controller by inverse model 14
c) Adaptive neural control 15
d) back-propagation of utility 15
e) Reinforcement learning control 15
1.2.10. Characteristics and advantages of neural networks 16
1.2.11. Limitations of neural networks 17
1.2.12. Extension 18
1.3. Fuzzy Logic Theory 18
1.3.1. Introduction 18
1.3.2. The Basics of Fuzzy Logic 18
1.3.3. Fuzzy Logic Modeling 19
1.3.4. Characteristics and advantages of fuzzy logic systems 21
1.3.5. Limitations and challenges of fuzzy systems 22
1.3.6. Comparative Overview of Fuzzy Logic Aspects 22
1.3.7. Conclusion 23
Chapter 2: 24
Fuzzy Neural Networks 24
2.1. Introduction 25
2.2. Different Approaches to Fuzzy Neural Networks 25
2.2.1. Functional Approach 25
2.2.2. Structural Approach 25
2.3. ANFIS architecture 26
2.3.1. Layer 1: Fuzzification Layer (Adaptive) 28

2.3.2. Layer 2 : Rule Layer 28
2.3.3. Layer 3 : Normalization Layer 29
2.3.4. Layer 4 : Defuzzification Layer (Adaptive) 29
2.3.5. Layer 5 : Summation Layer 30
2.4. The Learning Procedure 30
2.4.1. Hybrid Learning Algorithm in ANFIS 30
a) Least Squares Estimator (LSE) 31
b) Back-propagation: 31
. 31
2.5. Advantages of ANFIS 32
2.6. Disadvantages 33
2.7. FLP architecture 33
2.7.1. Fuzzification Layer: 33
2.7.2. AND and OU neurons: 34
a) OU function: 34
b) AND function: 34
2.7.3. Defuzzification layer 35
2.8. Advantages of FLP 37
2.9. Disadvantages of FLP 37
2.10. Conclusion 38
Chapter 3: 39
3.1 Introduction 40
3.2 Identification of Nonlinear Systems 40
3.2.1 Qualitative stage 40
3.2.2 Quantitative stage 41
3.2.3 Neuro-Fuzzy Identification Algorithm 41
3.3 Description of the Model 42
3.3.1. Operating Conditions 43
3.4 Identification of Concentration 43
3.4.1. Network Structure 43
3.4.2. Identification by using ANFIS Network 44
a) Training Results 45
b) Validation 47
c) Training Results 48
3.4.3. Identification by FLP Network 52
a) Validation 54
3.5 Conclusion 55
General Conclusion 57

References 60

List of Figures:

Figure2.1. Biological Neuron 18

Figure2.2. Formal Neuron 18

Figure2.3. Structure of dynamic network 21

Figure2.4. Structure of feed-forward network 22

Figure2.5. Back-propagation method 24

Figure2.6. Direct modeling structure 26

Figure2.7. Mamdani fuzzy logic system architecture diagram 32

Figure2.8. Sugeno fuzzy model diagram 32

Figure3.1. Functional approach 38

Figure3.2. Structural approach 38

Figure3.3. ANFIS architecture description 40

Figure3.4. ANFIS hybrid learning algorithm: two-pass procedure 45

Figure3.5. Logic processor 49

Figure3.6. FLP Architecture 50

Figure4.1. Convergence of the Series-Parallel Neuro-Fuzzy Model (series-parallel structure) 63

Figure4.2. ANFIS series-parallel Learning 64

Figure4.3. Convergence of the Series-Parallel Neuro-Fuzzy Model (parallel structure) 65

Figure4.4. ANFIS parallel Learning 65

Figure4.5. Full comparison: RK4 vs Series-Parallel vs Parallel FLP identification concentration and RMSE error 68

List of Tables:

Tab2.1. Comparative Overview of Fuzzy Logic Aspects 34

Tab 3.1. Layer-by-layer summary of the ANFIS architecture 42

Tab 3.2. ANFIS hybrid learning algorithm: two-pass procedure 44

Tab 4.1. Nominal reactor parameters 59

Tab 4.2. Fuzzification of Ca(k-1) 61

Tab 4.3. Fuzzification of Ca(k) 61

Tab 4.4. Fuzzification of qc(k) 61

Tab 4.5. Fuzzification of ca(k-1) 62

Tab 4.6. Fuzzification of ca(k) 62

Tab 4.7. Fuzzification of qc(k) 62

Tab 4.8. Fuzzification of the input qc 67

Tab 4.9. Fuzzification of the input ca 67

Tab 4.10. Fuzzification of the output ca 68

List of Acronyms:

ANN : Artificial Neural Network

ANFIS : Adaptive Network Fuzzy Inference System

FLP : Fuzzy Logic Processor

FNN : Fuzzy Neural Network

CSTR : Continuous Stirred Tank Reactor

NARMA : Nonlinear AutoRegressive Moving Average

SISO : Single Input Single Output

MISO : Multiple Input Single Output

LSE : Least Squares Estimator

RK4 : Fourth-order Runge-Kutta method

MF : Membership Function

FIS : Fuzzy Inference System

MRCA : Model-Reference-based Adaptive Neural Control

STR : Self-Tuning Regulator

RMSE : Root Mean Square Error

CSTR Variables

$\boldsymbol{Ca(t)}$: Reactant concentration in the reactor at time t

$\boldsymbol{Qc(t)}$: Coolant flow rate (control input)

$\boldsymbol{T(t)}$: Temperature inside the reactor

$\boldsymbol{Ca0, T0}$: Inlet concentration and inlet temperature

$\boldsymbol{Tc0}$: Inlet cooling water temperature

$\boldsymbol{k0, k1, k2, k3}$: Thermodynamic and kinetic constants

$\boldsymbol{V}$: Reactor volume

Neural Network Variables

$\boldsymbol{xi}$: Input signals to a neuron

$\boldsymbol{wi}$: Synaptic weights

$\boldsymbol{b}$: Bias term (negative threshold)

$\boldsymbol{s}$: Weighted sum of inputs

$\boldsymbol{f}(.)$: Activation function (e.g., sigmoid, linear threshold)

$\boldsymbol{L}$: Loss function (error criterion)

$\boldsymbol{\eta}$: Learning rate

Fuzzy System Variables

$\mu\boldsymbol{Ai}(\boldsymbol{x})$: Membership grade of input x in fuzzy set Ai

$\boldsymbol{ai}, \boldsymbol{bi}, \boldsymbol{ci}$: Premise parameters of the membership function

$\boldsymbol{wi}$: Firing strength (product of membership grades)

$\hat{\mathrm{w}}\boldsymbol{i}$: Normalized firing strength

$\boldsymbol{pi}, \boldsymbol{qi}, \boldsymbol{ri}$: Consequent parameters (Sugeno model)

$\boldsymbol{OL}, \boldsymbol{i}$: Output of node i in layer L

$\boldsymbol{U}$: Output of AND/OR neuron

$\boldsymbol{t-nom}, \boldsymbol{s-nom}$: Operators for AND/OR logic

$\boldsymbol{m}, \boldsymbol{b}$: Parameters of modified sigmoid function

$\boldsymbol{y}$: Final crisp output after defuzzification

Time-series / Model Terms

$\boldsymbol{k}$: Discrete time index

$\bar{\mathrm{Y}}(\boldsymbol{k}+\mathbf{1})$: Model prediction at step k+1

$\boldsymbol{U}(\boldsymbol{k})$: Manipulated input variable at time k

$\boldsymbol{RMSE}$: Root Mean Square Error (model performance measure)

General Introduction

The need for controlling industrial processes is essentially due to the phenomenon known as the stability-precision dilemma. The controller, regarded as the "intelligent" part of a feedback system, notably includes the corrector, whose role is to ensure the performance specified in the requirements, these performance specifications dictate the choice of the regulator to be implemented, whether as a classical analog or digital corrector or as a more or less complex regulation algorithm.

For the control engineer, identifying a system means determining the relationship between the manipulated variable (input) and the variable to be controlled (output).

It is well known that over the last three decades, linear models have been widely used in the control and identification of systems. However, in practice, most control systems encountered are nonlinear. In many cases, it is not desirable to represent them with linear models, and as a result, nonlinear models have been considered.

Several methods have been developed for the identification of linear systems. In contrast, for nonlinear systems, there are no general methods available for identification, due to the complexity of nonlinear systems, especially those with unknown structures. Hence the need to explore other approaches. Fuzzy logic theory and artificial neural networks are among the tools that can be used for the identification and control of nonlinear systems.

The objective of this work is to highlight the capabilities of the neuro-fuzzy approach in the identification of nonlinear systems. This approach combines artificial neural network theory and fuzzy logic theory.

Neuro-fuzzy identification ultimately consists of representing a system in the form of a neuro-fuzzy model. In fact, it involves building a neuro-fuzzy model representing the nonlinear system while adjusting its parameters so that its output gets as close as possible to that of the unknown system.

Several representations can be used for neuro-fuzzy identification. In this project, we considered the identification of concentration in a chemical reactor using two methods: the first

method uses an ANFIS-type network (Adaptive Network Fuzzy Inference System), whose inference is based on the Takagi-Sugeno fuzzy model, and the second uses a network with fuzzy logic operators (Fuzzy Logic Processors) with inference based on the Mamdani model.

To this end, we have organized our work as follows:

The second chapter gives a general overview of fuzzy logic theory and neural networks, their definitions and basic concepts.

The third chapter is devoted to an overview of neuro-fuzzy networks, presenting a detailed study of two neuro-fuzzy architectures, ANFIS and FLP.

The fourth chapter presents an application to the identification of concentration in a chemical reactor, a classic problem in nonlinear identification. It includes the simulation results of the system to be identified and of the two identification models.

The fifth chapter gives a general conclusion to our work.

Chapter 1:

Artificial Neural Network Theory and Fuzzy Logic

1.1. Introduction

In this chapter, we provide a brief introduction to the basic concepts of artificial neural network theory and fuzzy logic. These concepts will help to clarify the following chapter3, which is devoted to the theory of fuzzy neural networks.

1.2. Artificial neural networks

1.2.1. Introduction

The key question that the pioneers of computer science asked themselves was: "Can we find a substitute for the human brain?" It was this dream that drove research towards a new scientific discipline namely, artificial intelligence. From this line of thinking that neural networks emerged. Artificial Neural Networks (ANNs) are computational models inspired by the structure and function of the human brain. They form the backbone of modern machine learning and deep learning systems[1].

In this section, we will discuss the state of the art of ANN, examine how artificial neural networks work, the models and architectures commonly used, their key properties, the tasks they can perform, and their most common applications; we will also highlight their advantages and limitations.

1.2.2. State of the art of ANN

The history of Artificial Neural Networks (ANN) has passed through various critical phases whereby the simplest mathematical concepts have been transformed to the current advanced deep learning models.

The first mathematical model of a biological neuron was proposed by Warren McCulloch and Walter Pitts which showed that simple networks could think logically and symbolically[2]. Donald Hebb presented the principle of Hebbian Learning in 1949, in his book "The Organization of Behavior", by describing the strengthening of synaptic connections when neurons are activated together[3].

In 1958, Frank Rosenblatt introduced the Perceptron, the first neural network to be able to perform simple supervised learning based on the visual system[1]. This was at the same time that Widrow and Hoff created the ALE (Adaptive Linear Element) model which was later used as the basis building block to the multi-layer networks[2].

A major blow to the field was taken in 1969 when Marvin Minsky and Seymour Papert published their book “Perceptrons”. They mathematically demonstrated the limitations of single-layer networks, namely the inability to solve non-linear problems (such as the XOR problem) and the research and funding dropped[4].

Revitalization of interest started in the 1980s with the publication of the Backpropagation algorithm by Rumelhart, Hinton and Williams (1986).The algorithm was also applied by Yann LeCun [5]. This enabled to train multi-layers networks effectively. Hopfield networks (1982) and Kohonen maps (1984) were developed in this period too[6] .

This trend has been leveled off in the 1990s but, in the early 2010s, the onslaught of big data and more computing resources resulted in a spectacular resurgence. Deep Learning produced incredibly profound structures that can conduct image recognition, speech processing and autonomous control at a level never before achieved[7].

1.2.3. The formal neuron:

Inspired by their research on biological neurons as shown in (Fig 2.1), the two researchers Mac Culloch and Pits introduced the first model of the neuron in the 1943. This model was called the formal neuron, it is a model of the biological neuron as shown in figure (Fig.2.1). It is a binary element whose state is +1 (active) or -1 (inactive); it periodically updates its state in the following way: It calculates the sum of its inputs (which are the outputs of the formal neurons to which it is connected). The value of each input is modulated by the corresponding synaptic efficiency, and a decision is made by comparing this sum to a threshold specific to it, If the sum is greater than the threshold, the neuron enters its active state (+1), Otherwise, it enters its inactive state (-1). Here the threshold is implicit in the bias term b, as shown in Figure (fig. 2.2) [3]. So the bias is defined as b it's the negative threshold, shifted to the other side of the equation. Instead of asking "does the input exceed the threshold?", the neuron asks "does the weighted sum plus bias exceed zero?". The formal neuron is a Boolean automaton whose inputs and output are Boolean,

All neurons therefore make their decisions simultaneously, taking into account the evolution of the overall state of the network.

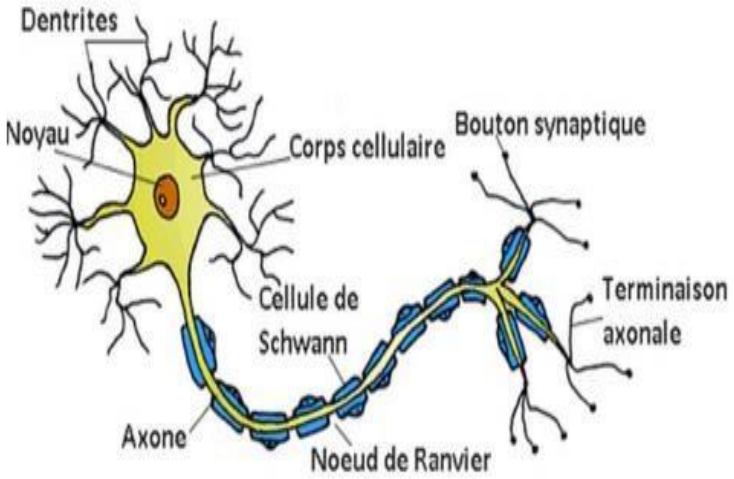


Fig. 2.1 Biological Neuron

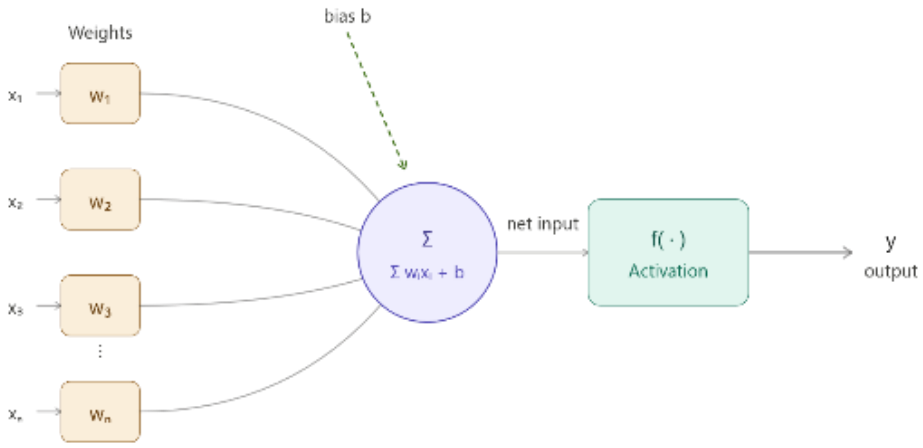


Fig. 2.2 Formal Neuron

The processing carried out by a neuron is given by:

$$y = f(s)$$
$$S = \sum_i w_i x_i + b$$

Where: w_i are the synaptic coefficients (weights of input connections), b is the bias value and S is the weighted sum called the potential, f is the activation function.

- ✓ S is called **the input function**; it can be:

Linear : $S(x_1, x_2, x_3, ...x_n) = \sum_{i=1}^{n} w_i x_i$

Refined : $S(x_1, x_2, x_3, ...x_n) = \sum_{i=1}^{n} w_i x_i + b$

- ✓ **The activation function** maybe of different kinds:

a) **Threshold binary function** such as the heaviside function or the sign function.

b) **Linear threshold function** or saturation function, let SATURE be a linear threshold function:

$$SATURE = \begin{cases} x & si \quad x \in [u \quad v] \\ u & si \quad x \le u \\ v & si \quad x \ge v \end{cases}$$

c) **Sigmoid function** such that:

$$f(x) = \frac{a(e^{kx} - 1)}{e^{kx} + 1} \quad \text{.....(2.1)}$$

d) **Stochastic function:**

$$f(x) = \begin{cases} 1 & if \quad p(x) = 1/(1 + e^{-x/t}) \\ 0 & if \ not \end{cases}$$

The parameter t is called temperature, if it tends towards zero, this function tends towards the binary threshold function.

Remark:

The activation function is generally confused with the output of the neuron. Currently, the sigmoid function is usually used because it is continuous and differentiable, making it well suited to recently discovered learning algorithms.

1.2.4. Definition of Artificial Neural Networks (ANN):

An ANN is a system of interconnected nodes (neurons) organized into layers. Each neuron performs a weighted summation of input signals modulated by an activation function (a non-linear function) and generates an output that will be applied to other neurons via weighted connections. By adjusting the strength (weight) of each connection during training, the network learns to map inputs to outputs.[7] With this simple structure and by choosing an appropriate number of neurons, these networks are capable of approximating any complex function with a certain degree of accuracy in various application domain such as:

- Domain of recognition.
- Identification
- Classification
- Speech domain
- Vision domain
- Control system
- Signal prediction
- Image compression optimization
- Signal processing

1.2.5. Architecture of Neural Networks

Depending on the structure of the neurons, neural networks can be classified into two main categories: recurrent (dynamic) networks and non-recurrent or feed - forward (static) networks.

a) Recurrent network (dynamic network):

In recurrent networks, several neurons are interconnected to organize the network. The Hopfield network shown in (Fig.2.3) belongs to this type of structure.

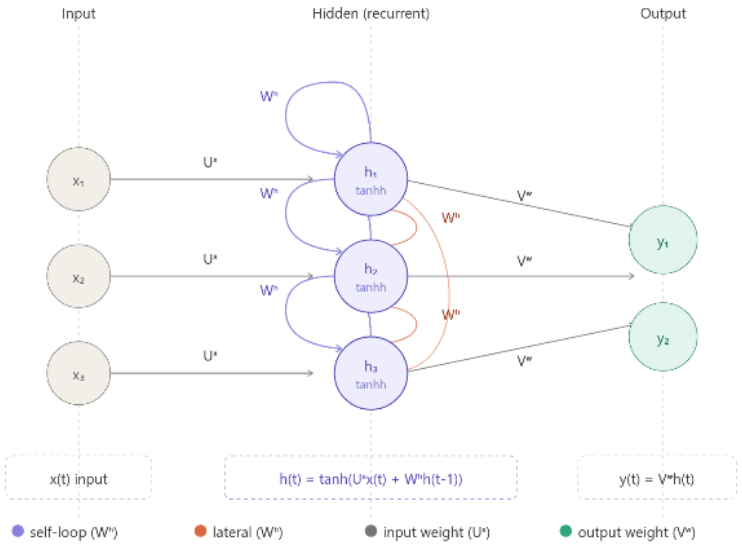


Fig.2.3 Structure of dynamic network

b) Non-recurrent network (or feed-forward network):

This is the simplest type of neural network. In a feed-forward network, information moves in only one direction from the input layer to the output layer, as shown in (Fig.2.4) there is no going back. In this type of network, there is no interconnection between neurons in the same layer. This type of network is good for static problems, where the order of the data does not matter, such as classifying images or predicting a number [6]

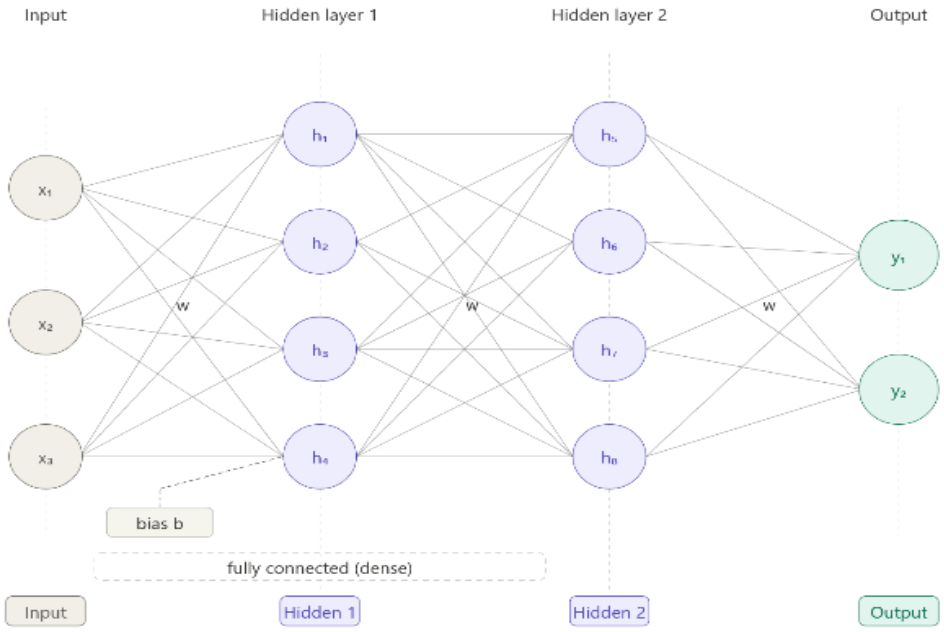


Fig.2.4 Structure of feed-forward network

1.2.6. Training of Neural Networks

Training of neural networks comprises two phases: a learning phase and a second operation phase. Learning is a process through which the network parameters (connection weights) are adjusted to reflect the information contained in the network structure. Once the network is trained, it represents a static knowledge base that can be called during the operation phase. Training generally requires two sets of data:

- A training dataset for calculating synaptic weights.
- A test dataset, different from the previous ones, to evaluate the quality of the learning and the ability of the networks to process unknown data.

There are three types of learning: supervised learning, unsupervised learning, and semi-supervised learning.

a) Supervised learning:

The network is fed with the desired input and output values; the network then adjusts its weights based on the calculated output error. This type of learning is used for tasks like image classification and predicting prices[5].

b) Unsupervised learning:

In unsupervised learning, we give the network data without any correct answers. The network has to find patterns and groups in the data by itself. This is useful when we do not know what patterns to look for in advance. It is often used for tasks like grouping customers by behavior or detecting unusual events in a system [6].

c) Semi-supervised learning

The network receives implicit scalar evaluations of previous inputs. The output error value is no longer available; the only available information is a failure signal or simply the sign of the error. This is very limited information. Reinforcement learning belongs to this family of algorithms.[5]

1.2.7. Back-propagation method

Back-propagation is the most widely used method for training neural networks; it is a supervised learning algorithm that involves adjusting the weights of the network connections simply by using input/output data. This algorithm allows the error standard thus formulated to propagate from the network output to the input via its layers, hence the name back-propagation. The procedure is based on calculating the derivatives of an entity (error norm) with respect to the set of weights; therefore, this calculation requires the continuity and differentiability of the activation functions of the neurons. [5]

The objective of back-propagation is to adjust the network parameters in order to minimize the mean squared error across all training samples, as shown in Fig.2.5

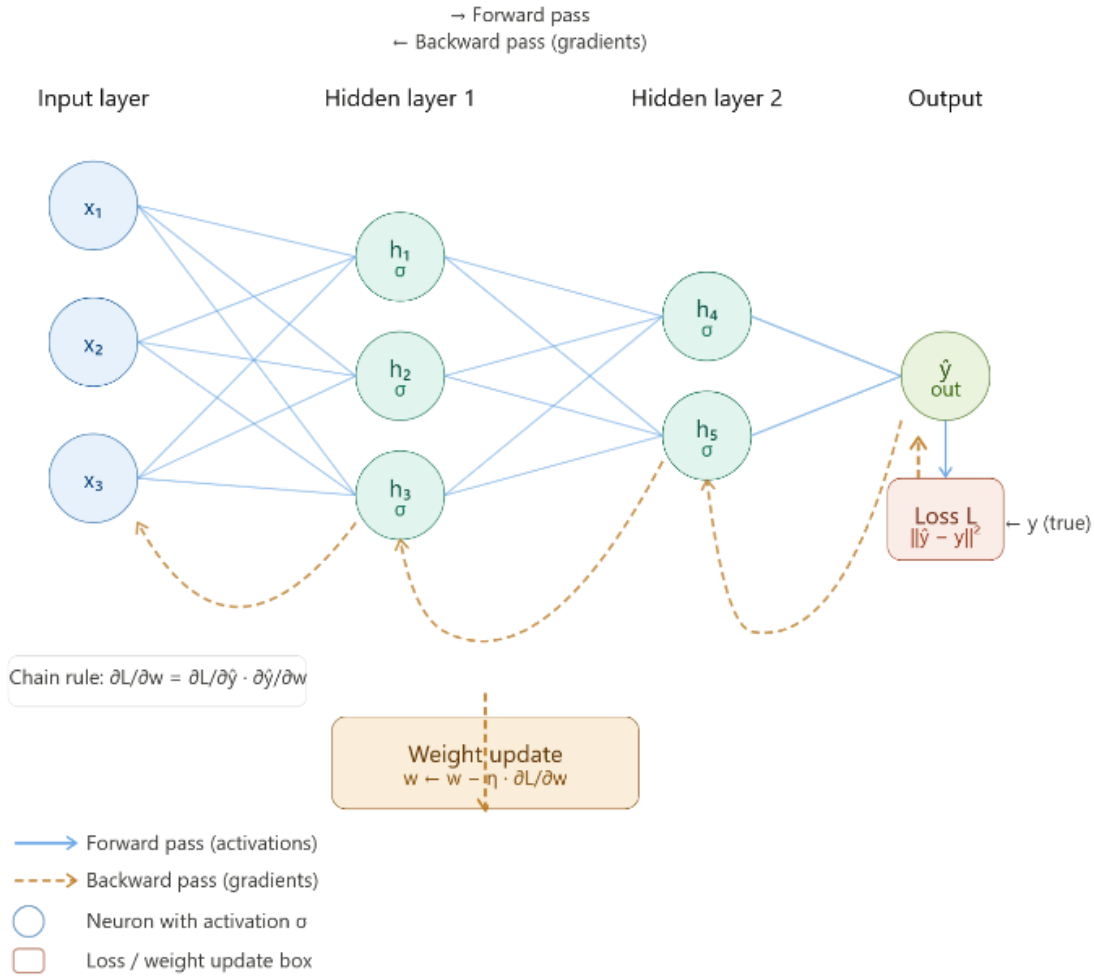


Fig.2.5 Back-propagation method

Let L be the error criterion, given the network parameter vector w. The back-propagation algorithm is given by:

$$w_{k+1} = w_k - \eta \partial L_k / \partial w_k$$

Where η is the learning rate, which influences the stability and convergence speed of the algorithm.

The back-propagation algorithm presents a convergence problem, but remains the most widely used algorithm in neuro computing.

1.2.8. Identification by Neural Networks

Neural networks are considered universal approximators, a neural network with a suitable structure can approximate any correspondence between the input space and the output space.

Given a certain number $\varepsilon \succ 0$, a neural network with a sufficiently large number of nodes can be determined so that $\|f(x) - f_n(x)\| = \|e(k)\| \prec \varepsilon \ for\ all \ \ x \in D.$

Training the neural network f_n to approximate the function f on the domain D is equivalent to minimizing the following error criterion:

$$j_k = \sum \|e(k)\|^2$$

There are two models, direct and indirect.

a) Direct modeling

Direct modeling involves training a neural network to represent the direct dynamics of the system. The difference between the model output and the system output is used to adjust the network parameters. As shown in Fig.2.6

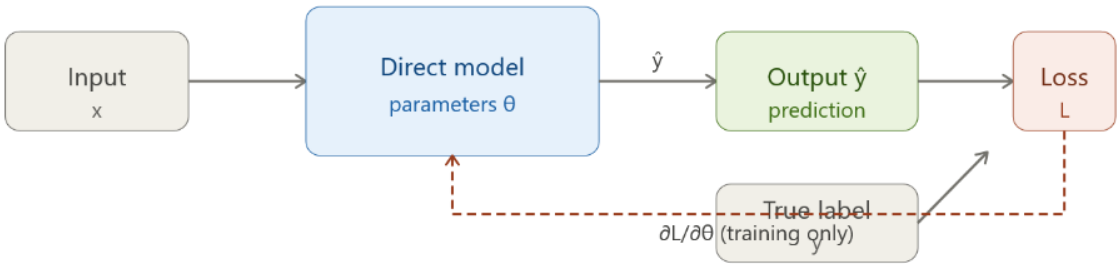


Fig.2.6 Direct modeling structure

The simplified figure shows the three core elements: the input x feeds into the direct model (with parameters θ), which produces a prediction $\hat{y}$. During training, a loss function compares $\hat{y}$ against the true label y, and the gradient $\partial L/\partial\theta$, flows back to update θ. At inference time, only the solid arrows matter, the loss and gradient path disappear.

b) Inverse modeling

It involves training a neural network to learn the inverse dynamics of the system. The goal in control theory is to enable the system's output to behave in the desired manner. Consequently, this procedure does not directly target the objective. A so-called specialized learning architecture, based on the difference between the system output and the desired output, has been proposed to overcome this problem. [ ].

1.2.9. Neural network control methods

Despite the apparent complexity required by specific applications, there are only five neural control structures currently in use.

a) Supervised control

The neural network learns the correspondence between inputs and desired outputs during training, by presenting the neural network with a set of examples (inputs, desired outputs) and we adjust the synaptic coefficients to reduce the difference between the actual outputs and the desired outputs. After learning, the network becomes capable of associating the received information with the corresponding desired actions.

The advantage of this technique is the possibility of replacing a human operator when human presence is impossible. In this case, the controller training takes place during a simulation phase.

b) Direct controller by inverse model

A neural network learns the inverse dynamics of the system, the resulting inverse model (controller) is integrated into the system to be controlled in order to establish an identity function between the actual output and the desired output. Reconstructing the control signal using the inverse model allows for direct control of the error on the control vector. [1]

c) Adaptive neural control

Adaptive neural control is directly derived from standard adaptive techniques such as model-reference-based adaptive neural control (MRCA) and self-regulating controllers (STRs), In the reference model adaptive neural control proposed by Narenda and Parthasarathy, the desired performance is expressed in terms of a reference model, the neural controller is trained to minimize the difference between the real output of the system and that generated by the reference model. Two structures were proposed, the direct control structure and the indirect control structure. [4]

d) back-propagation of utility

Utility backp-ropagation is a method that allows solving planning problems over time (long-term optimization). The objective is to train an optimal controller so that it can generate the command to drive the system from the initial state to the desired state over a finite time T, this requires the back-propagation of an information stream through the time of T, T-1, T-2, and so on; in this case, the desired behavior of the system is specified by a performance criterion (utility function) J defined on the horizon T. The learning processor allows finding the controller parameters that optimize the utility function J; a system model is required to transform the error on the system state into an error on the controller output.

This approach requires storing time series of observations, which is completely impossible from a biological perspective. It also requires a form of batch learning that is unsuitable for real-time applications. [5]

e) Reinforcement learning control

The reinforcement learning controller is based on the fact that the tendency to repeat an action that leads to a satisfactory state or improved performance must be reinforced. Therefore, when the desired actions are not known, the controller's performance is evaluated indirectly by considering the effect of its outputs on the system state.

A reinforcement learning control structure generally includes an evaluation network and an action network; the output of the performance evaluation network is used as a "reward" or "punishment" signal for the output of the action network; the parameters of the latter are adapted to optimize a certain performance criterion.

Unfortunately, when the size of the control vector is large enough, a credit allocation problem arises; it becomes difficult to determine which components deserve rewards or punishments. [4]

1.2.10. Characteristics and advantages of neural networks

The current interest in neural networks is justified by the few fascinating properties they possess. Among these properties, we can mention:

- ✓ **Parallelism**

This notion of parallelism is at the basis of the architecture of neural networks, considered as sets of elementary entities operating simultaneously; parallelism allows for faster, superior computational speed, but requires thinking about and posing the problems to be solved differently.

- ✓ **Adaptability**

This manifests itself through the learning capacity that allows the network to take into account new constraints or new data from the outside world. This capability is of crucial importance for all evolutionary problems.

- ✓ **Distributed memory**

In neural networks, the memory of a fact corresponds to an activation map of neuronal; this map is, in a way, a code of the memorized fact. The advantage of this distribution of memory over several inputs is what is called resistance to noise; on the one hand, the loss of an element does not correspond to the loss of a memorized fact. In computing, data corresponds to a memory location, and the loss of this location results in the loss of the stored data.

- ✓ **The ability to generalize**

This capability is essential; its purpose is to provide a system that, from sets of examples, can learn to identify underlying rules or mimic the behaviors that solve

the problem. In other words, an input vector close to one of the examples will result in the output of a vector close to the output associated with that example.

- ✓ **Ease of construction**

Computer simulation of a neural network for a small application is simple and requires relatively little development time. For more complex applications, the use of simulation or acceleration maps proves beneficial. However, it is worth noting how easy these simulators are to use.

There are also other advantages, which are as follows:

- ✓ **Resilience to neuronal failures**, thanks to the distribution of knowledge.

- ✓ **No need for a template:**

A neural network does not know the dynamic equation of the system to be controlled. It only associates inputs with outputs based on the training it has received.

- ✓ **Robustness to noise inherent in the measurements**
- ✓ **Universals approximation**

1.2.11. Limitations of neural networks

- They need a lot of data and computing power to train, which can be expensive and slow.
- They are hard to understand. It is difficult to know why the network made a specific decision. This is called the black box problem[4].
- If the network is too complex or the data is too small, it may memorize the training data instead of learning general rules. This is called overfitting[5].
- They are sensitive to the settings chosen before training, such as the learning rate and the number of layers. Finding the right settings often requires many trials.

1.2.12. Extension

It is possible to link layers not only to the next layer, but also to other, more distant layers.

1.3. Fuzzy Logic Theory

1.3.1. Introduction

Fuzzy logic is considered one of the most important approaches used to deal with uncertainty in real-world systems. Unlike classical logic, which is limited to binary values (true or false), fuzzy logic allows intermediate values between 0 and 1. This makes it more suitable for representing real situations where information is often imprecise or vague.This concept was introduced by Lotfi Zadeh in 1965 and has since been widely applied in many fields such as control systems, artificial intelligence, and industrial automation [8]. Thanks to its ability to imitate human reasoning, fuzzy logic provides flexible and efficient solutions, especially when mathematical models are difficult to establish.[9]

1.3.2. The Basics of Fuzzy Logic

a) Understanding the universe of discourse, fuzzy sets, and membership degree

The universe of discourse refers to the range of all possible values that a variable can take. Within this range, fuzzy sets are defined to represent linguistic concepts such as “low”, “medium”, or “high”[9].

Unlike classical sets, where an element either belongs or does not belong to a set, fuzzy sets allow partial membership. Each element is assigned a membership degree between 0 and 1, indicating how strongly it belongs to the set. For example, a temperature of 25°C may be considered “warm” with a degree of 0.8.[8]

b) Support, switching point, and fuzzy singleton explained

The support of a fuzzy set includes all values whose membership degree is greater than zero.[9] It defines the effective range of the fuzzy set.

A switching point generally represents a value where the behavior of the system changes or where two membership functions intersect.[10]

A fuzzy singleton is a special case of a fuzzy set where only one element has a membership value equal to 1, while all other elements have a value of 0. It is often used to represent precise input values in fuzzy systems.[10]

c) How fuzzy logic operators work

Fuzzy logic operators are used to combine different conditions in fuzzy rules. The most common operators are AND, OR, and NOT.[10]

The AND operator is typically implemented using the minimum function, while the OR operator uses the maximum function. The NOT operator represents the complement of a membership value.[9] These operators allow fuzzy systems to simulate human reasoning when making decisions.

1.3.3. Fuzzy Logic Modeling

a) Mamdani model and its main components

The Mamdani model is one of the most widely used fuzzy inference systems because of its intuitive structure and its ability to represent human knowledge using linguistic rules.[11]

A Mamdani fuzzy system is composed of several main stages:

Fuzzification: This step transforms crisp numerical inputs (such as sensor data) into fuzzy values using membership functions.[10]

Knowledge base: It contains the set of fuzzy rules and the membership functions defined by experts. These rules are usually expressed in the form of "IF–THEN" statements.[11]

Inference mechanism: This stage processes the fuzzified inputs using fuzzy operators (such as MIN and MAX) to generate fuzzy outputs based on the defined rules.[4]

Defuzzification: In this final step, the fuzzy output is converted into a crisp value that can be used for control or decision-making. The centroid method is commonly used for this purpose.[10]

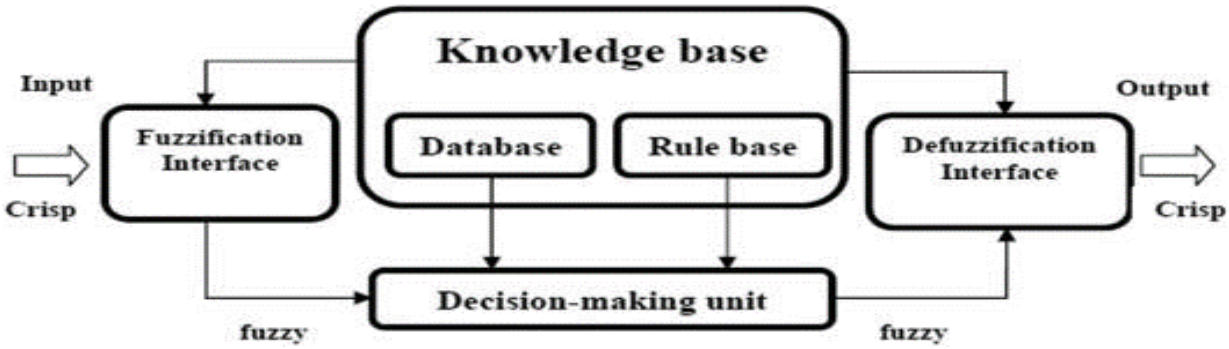


Fig.2.7. Mamdani fuzzy logic system architecture diagram

b) Sugeno model and its specificity

The Sugeno model, also known as the Takagi-Sugeno model, differs from the Mamdani model mainly in the form of its output. Instead of producing fuzzy outputs, it generates mathematical expressions, often linear or constant functions.[12]

This characteristic makes the Sugeno model more efficient in terms of computation and more suitable for optimization and real-time control applications. It is commonly used in advanced systems such as robotics and adaptive control.[12]

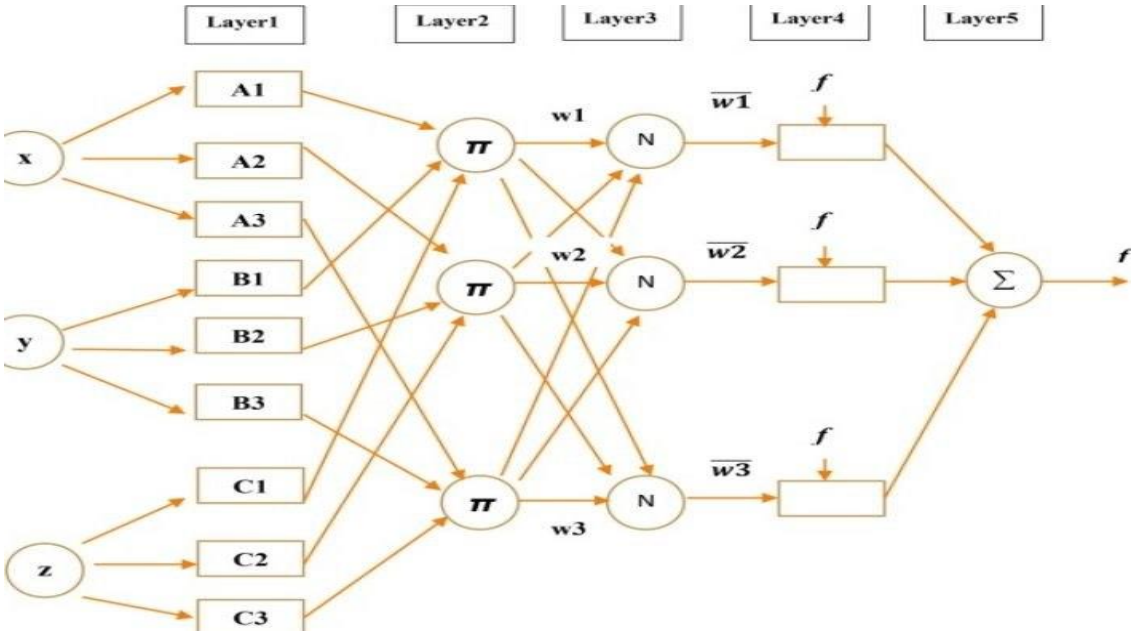


Fig. 2.8. Sugeno fuzzy model diagram

1.3.4. Characteristics and advantages of fuzzy logic systems

a) Main characteristics of fuzzy logic systems

Fuzzy logic systems are designed to handle uncertainty and imprecision. They rely on linguistic variables and membership functions [9] to describe system behavior in a flexible and intuitive way.

b) Why fuzzy logic is advantageous in engineering

Fuzzy logic offers several important advantages. It can effectively handle noisy and uncertain data, making it suitable for real-world applications.[13] It is also capable of modeling nonlinear systems that are difficult to represent using classical methods.

Another key advantage is the possibility of integrating human expertise directly into the system through simple IF–THEN rules.[11] Additionally, fuzzy systems do not require an exact mathematical model, which simplifies their design in many cases.

1.3.5. Limitations and challenges of fuzzy systems

Despite its advantages, fuzzy logic also has some limitations. Designing and tuning membership functions can be complex and often depends on expert knowledge.[13]

Moreover, there is no universal design methodology that guarantees optimal performance in all cases. In systems with many input variables, the number of rules can increase significantly, leading to higher computational cost and reduced efficiency.[13]

1.3.6. Comparative Overview of Fuzzy Logic Aspects

To summarize the key characteristics and distinctions within fuzzy logic and control, the following table (Tab.2.1) provides a comparative overview of different concepts and their defining features.

Aspect	Description	Key Characteristics	Example / Application
Fuzzy Logic	A form of AI that allows reasoning with uncertainty and partial truths (degrees between 0 and 1).	Handles imprecise data, mimics human reasoning, linguistic variables.	Decision-making in complex market analysis.
Fuzzy Set	An extension of classical sets where elements have degrees of membership [0, 1]	Partial belonging, defined by membership functions.	Defining "warm" or "cold" temperatures.
Membership Function (MF)	Maps input values to their degree of membership in a fuzzy set.	Shapes include triangular, Gaussian, trapezoidal; parameterizes fuzzy controllers.	A 25°C temperature having 0.8 membership in "warm".
Fuzzification	Process of converting crisp (precise) inputs into fuzzy sets.	First step in fuzzy system, uses MFs to assign degrees of membership.	Converting a sensor reading (e.g., 60 km/h) into "medium speed".

Inference Mechanism	Processes fuzzified inputs against the rule base to generate fuzzy outputs.	Simulates human reasoning, uses fuzzy operators (MIN, MAX)	Applying "IF temperature is HIGH, THEN fan speed is FAST" rule.
Defuzzification	Process of converting fuzzy outputs back into crisp, actionable values.	Common methods: Centroid, Mean of Maximum.	Translating "FAST fan speed" into 1500 RPM.
Mamdani Controller	Fuzzy controller with fuzzy sets as consequents in IF-THEN rules.	Intuitive, human-knowledge-based, often computationally intensive.	Controlling washing machine cycles or air conditioning.
Takagi-Sugeno Controller	Fuzzy controller with mathematical functions (e.g., linear) as consequents.	Computationally efficient, analytically tractable, suitable for optimization.	Adaptive control in robotics or power electronic systems.

Tab.2.1. Comparative Overview of Fuzzy Logic Aspects

1.3.7. Conclusion

Fuzzy logic represents a powerful tool for dealing with complex and uncertain systems. By allowing gradual truth values instead of strict binary decisions, it provides a more realistic representation of real-world problems.

Both Mamdani and Sugeno models demonstrate the effectiveness of fuzzy logic in control applications. Although some challenges remain, fuzzy systems continue to be widely used due to their flexibility, simplicity, and ability to incorporate human reasoning.[12][13]

Chapter 2:

Fuzzy Neural Networks

2.1. Introduction

Fuzzy Neural Networks (FNN) combines two powerful computational paradigms: fuzzy logic systems and neural networks. On one hand, fuzzy systems provide transparent, interpretable rules that are easy to understand. On the other hand, neural networks have the ability to learn from data and adapt to changes through training. Because of this powerful combination, FNN are widely used in many applications including control systems, prediction, pattern recognition, and decision support.

In this chapter, we first present the two main approaches to integrating fuzzy logic with neural networks. Then, we focus on the Adaptive Neuron fuzzy Inference System (ANFIS) and Fuzzy Logic Processor (FLP) architecture the most well-known structural FNN model and explain its layer-by-layer structure and hybrid learning process in detail.

2.2. Different Approaches to Fuzzy Neural Networks

The integration of fuzzy logic and neural networks can be divided into two main approaches, depending on how the two components are combined within the system [1], [2].

2.2.1. Functional Approach

In this approach, the neural network is used to support an existing fuzzy system. The overall structure of the fuzzy system whether Mamdani or Sugeno type remains unchanged. The neural network is employed only for specific subtasks, such as automatically defining membership function parameters or extracting fuzzy rules from training data, as shown in fig.3.1.

As a result, the system retains its transparency and interpretability, which is a significant advantage in applications where human understanding of the decision process is critical [3].

2.2.2. Structural Approach

Unlike the functional approach, the structural approach takes a more integrated path by embedding fuzzy logic directly into the architecture of the neural network itself. In this design, each stage of the fuzzy inference process, fuzzification, rule evaluation, and defuzzification, is mapped to a dedicated layer within the network. The result is a single, unified system that can be trained end-to-end using standard learning algorithms, as shown in fig.3.2. The most well-known model built on this approach is ANFIS (Adaptive Neuro-Fuzzy Inference System), and FLP (Fuzzy Logic Processor) which have been widely studied and adopted in the field [1], [4].

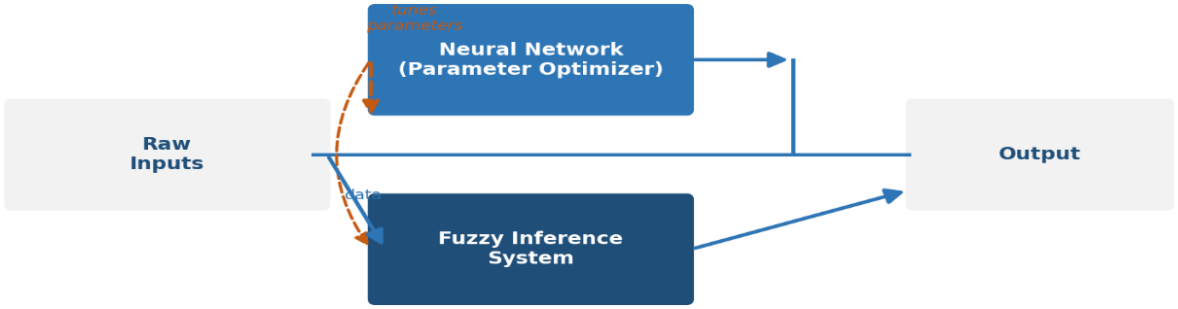


Fig.3.1. Functional approach

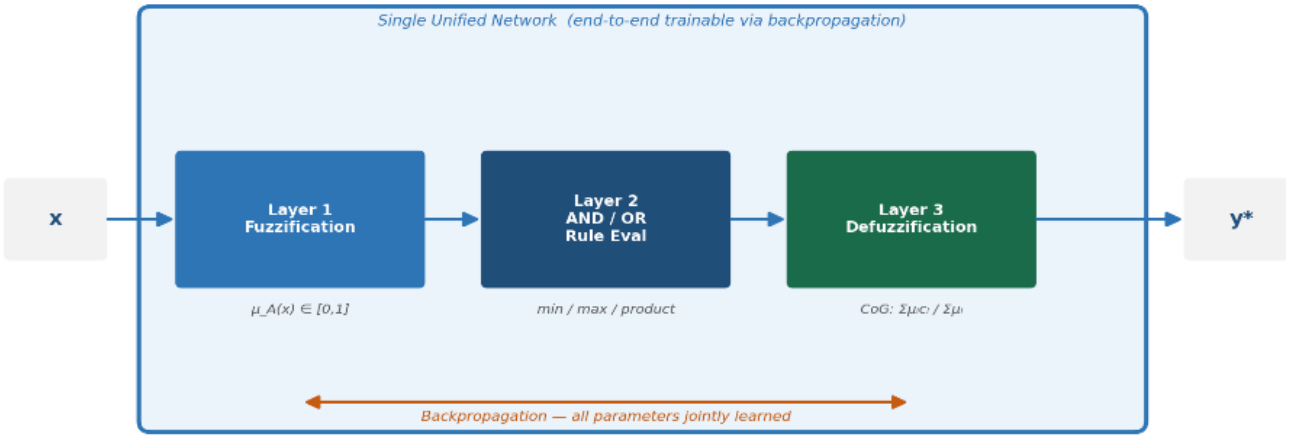


Fig.3.2. Structural approach

2.3. ANFIS architecture

Several FNN architectures have been proposed in the literature based on the structural approach. The Wang-Mendel network and Mamdani-based FNNs are notable examples. However, ANFIS remains the most widely adopted architecture due to its systematic training procedure, compatibility with standard optimization tools such as MATLAB's Fuzzy Logic Toolbox, and its solid mathematical foundation [4].

The ANFIS (Adaptive Neuro-Fuzzy Inference System) was first introduced by Jang in 1993 [1]. This is an adaptive system based on Sugeno's first order fuzzy model. The basic idea is simple: Using the powerful learning ability of neural networks and the powerful reasoning ability of fuzzy logic. This makes it possible to learn the parameters that control a fuzzy inference system (FIS) [1] automatically.

To understand the working of the ANFIS architecture, let's consider an example to get an idea. We will build two fuzzy IF-THEN rules where x_1 and x_2 are the inputs, and the rules will be based on a first order Sugeno model.

$$Rule\ 1: IF\ x1\ is\ A_1\ AND\ x2\ is\ B_1, THEN\ f_1 = p_1x1 + q_1x2 + r_1$$

$$Rule\ 2: IF\ x1\ is\ A_2\ AND\ x2\ is\ B_2, THEN\ f_2 = p_2x1 + q_2x2 + r_2$$

X_1 and x_2 are two inputs to which are associated A_i and B_i fuzzy sets. Now, let's use f_i to denote the output that you would expect in each of these fuzzy regions, and then p_i, q_i, r_i are the consequent parameters that will be determined during the training process.

ANFIS has five-layer architecture. The system uses adaptive nodes (square) and fixed nodes (circle). Each layer performs a specific function as described below and summarised in Table.3.1

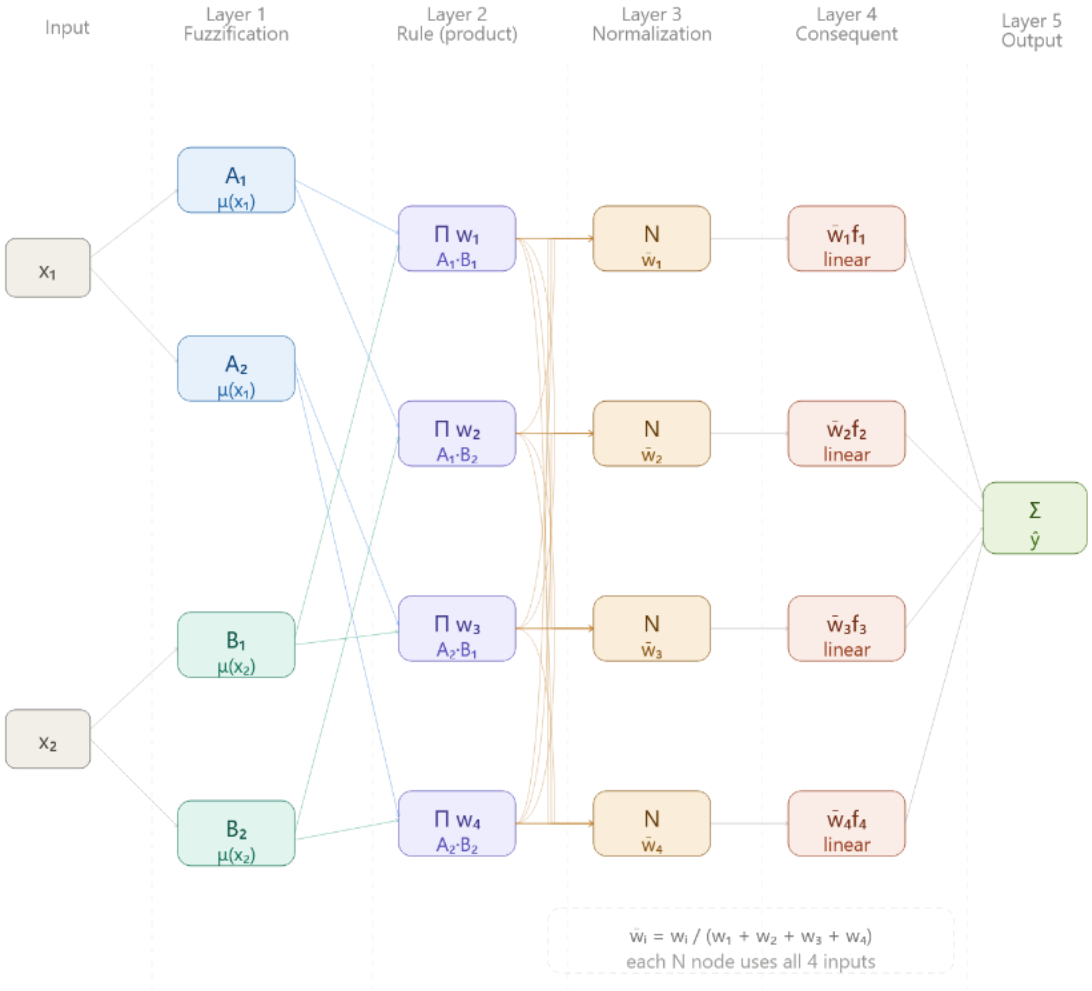


Fig.3.3. ANFIS architecture description

2.3.1. Layer 1: Fuzzification Layer (Adaptive)

All nodes in Layer 1 are adaptive. The output of node i represent the fuzzy membership grade of the input, defined as:

$$O1, i = \mu Ai(x1), \quad i = 1, 2 \text{..(3.1)}$$

$$O1, i = \mu Bi - 2(x2), \quad i = 3, 4 \text{..(3.2)}$$

For example, if a Bell-shaped membership function is used:

$$\mu Ai(x) = 1 / (1 + (((x - ci) / ai)2) b_i) \text{ (Bell-shaped) ....(3.3)}$$

Or if a Gaussian membership function is used:

$$\mu Ai(x) = exp(-((x - ci) / ai)2) \ (Gaussian) \text{ ....(3.4)}$$

Where a_i, b_i, c_i are the premise parameters of the membership function that are updated during training.

In our study we choose Gaussian membership function, so the derivatives of the output with respect to c_i and a_i become:

$$\frac{\partial O_{1,i}}{\partial c_i} = 2\left((x - c_i)/a_i^2\right) \exp\left(-\left((x - c_i)/a_i\right)^2\right) \text{..(3.5)}$$

$$\frac{\partial O_{1,i}}{\partial a_i} = 2((x - c_i)^2/a_i^3) \exp\left(-\left((x - c_i)/a_i\right)^2\right) \text{..(3.6)}$$

2.3.2. Layer 2 : Rule Layer

Inference layer, each node of this layer is a circle, it multiplies the input signals (computes the firing strength w_i) and sends the product, this output are called the normalized firing strengths and represents the degree of activation of the corresponding rule.

$$O2, i = wi = \mu Aq(x) \cdot \mu Bq + m(y), \text{ ....(3.7)}$$

$i = 1, 4; q=1,2; m=1,2$

2.3.3. Layer 3 : Normalization Layer

Each node normalizes the firing strengths from the previous layer:

$$O_{3,i} = \hat{w}_i = \frac{w_i}{\sum_{i=1}^{I} w_i}, \quad i = 1,4 \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(3.8)$$

I: is the inference ruler number

The derivative of the output with respect to w_i is given as follows

$$\frac{\partial \hat{w}_i}{\partial w_j} = \frac{\sum w_i - w_i}{(\sum w_i)^2}, \quad \text{if} \quad i = j \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(3.9)$$

$$\frac{\partial \hat{w}_i}{\partial w_j} = \frac{-w_i}{(\sum w_i)^2}, \quad \text{if} \quad i \neq j \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(3.10)$$

2.3.4. Layer 4 : Defuzzification Layer (Adaptive)

Each node computes the product of the normalized firing strength and a first-order polynomial:

$$O_{4,i} = \hat{w}_i . f_i = \hat{w}_i (p_i x_1 + q_i x_2 + r_i) \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(3.11)$$

Where p_i, q_i, r_i are the consequent parameters updated by the Least Squares method.

The derivatives of the output with respect to these parameters are calculated as:

$$\frac{\partial O_{4,i}}{\partial p_i} = \hat{w}_i x_1 \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(3.12)$$

$$\frac{\partial O_{4,i}}{\partial q_i} = \hat{w}_i x_2 \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(3.13)$$

$$\frac{\partial O_{4,i}}{\partial r_i} = \hat{w}_i \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(3.14)$$

2.3.5. Layer 5 : Summation Layer

The single output node computes the overall output as the weighted sum of all inputs:

$$O_5 = \sum_{i=1}^{I} O_{4,i} = \sum_{i=1}^{I} \hat{w}_i f_i \quad \text{.....(3.15)(3.16)}$$

Table.3.1 below summarises the role of each layer in the ANFIS architecture:

Layer	Name	Type	Operation	Parameters
1	Fuzzification	Adaptive	$O_{1,i} = \mu_{A_i}(x_i)$, Gaussian MF	ai, bi, ci $(premise)$
2	Rule Layer	/	$O_{2,i} = w_i = \mu_{A_i}(x_1).\mu_{A_i}(x_2)$	/
3	Normalization	/	$O_{3,i} = \hat{w}_i = \frac{w_i}{\sum_{i=1}^{I} w_i}$	/
4	Defuzzification	Adaptive	$O_{4,i} = \hat{w}_i(p_i x_1 + q_i x_2 + r_i)$	pi, qi, ri $(consequent)$
5	Summation	/	$O_5 = \sum_{i=1}^{I} \hat{w}_i f_i$	/

Tab.3.1 Layer-by-layer summary of the ANFIS architecture.

2.4. The Learning Procedure

After defining the ANFIS structure, the system must be trained. The training objective is to minimize the error between the actual output and the desired output by adjusting the network parameters.

2.4.1. Hybrid Learning Algorithm in ANFIS

ANFIS is a hybrid learning algorithm, which is the mixture of two complementary learning algorithms, Least Squares Estimator (LSE) and Back-propagation (Gradient Descent) algorithm[6]. The reason for this combination is that the overall output of ANFIS can be represented as:

$$\hat{y} = \sum_{i=0}^{I} \hat{w}_i(p_i x_1 + q_i x_2 + r_i) \quad \text{.....(3.17)}$$

Expanding this expression:

$$\hat{y} = \hat{w}_1 p_1 x_1 + \hat{w}_1 q_1 x_2 + \hat{w}_1 r_1 + \hat{w}_2 p_2 x_1 + \hat{w}_2 q_2 x_2 + \hat{w}_2 r_2 + \cdots \text{ ... ...}\text{.....(3.18)}$$

The following equation is linear in the parameters pi, qi and ri. This linearity is the crucial point that warrants the hybrid strategy.

a) Least Squares Estimator (LSE)

Least Squares Estimator (LSE) is the method ANFIS uses (in Layer 4) to find the best consequent parameters. Here's the core idea first, followed by an illustrative example.

The goal: given a set of data points, find a line (or plane) that fits them as closely as possible by minimizing the total squared error between predictions and actual values.

Example: if we have a system defined by the following formula:

$$A\theta = y$$

(A: known inputs, θ: unknown parameters, y: target outputs), LSE solves it in one shot:

$$\theta^* = (A^T A)^{-1} A^T y.$$

This gives the parameter vector that produces the minimum possible sum of squared errors.

Why squared? Squaring the errors makes them always positive, punishes large errors more than small ones, and leads to a clean closed-form solution.

In ANFIS specifically: Layer 4 computes:

$$O_{4,i} = \hat{w}_i . f_i \text{ , where } f_i = p_i x_1 + q_i x_2 + r_i.$$

The unknowns are the parameters: p_i, q_i and r_i of each rule. LSE finds the best values for all of them at once from the training data, no iteration needed, just one matrix calculation.

b) Back-propagation: is already demonstrated in Chapter 2 in the section (2.2.6)

Pass	Method	Parameters Updated
Forward Pass	Least Squares Estimator (LSE)	Consequent parameters: pi, qi, ri $(Layer\ 4)$
Backward Pass	Back-propagation	Premise parameters : ai, bi, ci $(Layer\ 1)$

.

Tab.3.2 ANFIS hybrid learning algorithm: two-pass procedure

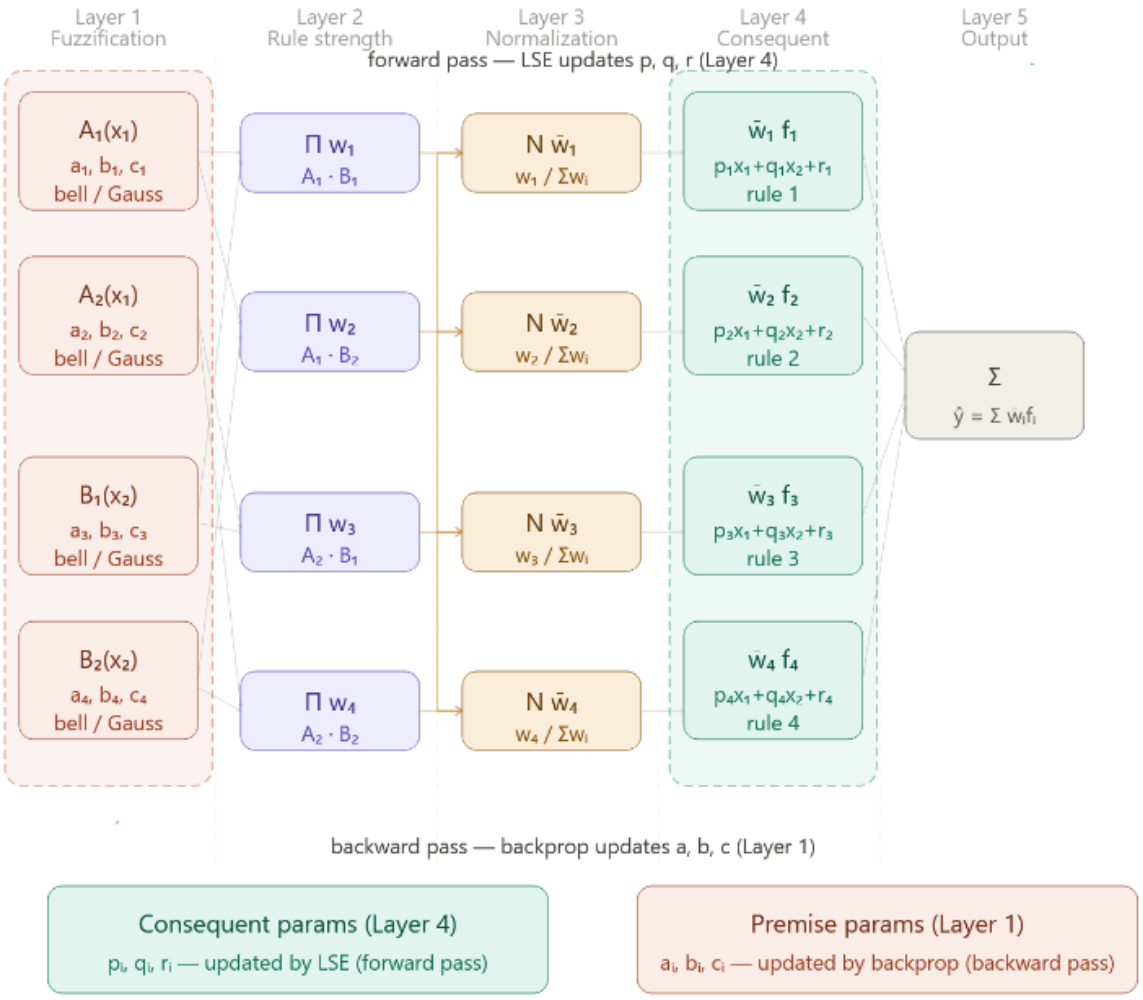


Fig.3.4. ANFIS hybrid learning algorithm: two-pass procedure

By solving for the consequent parameters analytically with LSE, the hybrid algorithm convergences much faster than the pure gradient descent algorithm alone, as the number of the parameters to be estimated is reduced by the reduction of the dimension of the search space. It has been shown to be very efficient in training ANFIS systems[1, 5, 7].

2.5. Advantages of ANFIS

- ✓ Interpretability: ANFIS learns from data and explains its decisions through clear fuzzy IF-THEN rules, which makes it easy to understand and verify how the system actually work

- ✓ Fast convergence: The hybrid learning algorithm combining LSE and Back-propagation, allows ANFIS to train significantly faster than using gradient descent alone.
- ✓ No expert knowledge required: The membership functions are learned automatically from the training data, so you don't need prior domain expertise to define them manually.

- ✓ Universal approximator: Mathematically, ANFIS can approximate any smooth nonlinear function, which makes it a powerful tool for modeling complex real-world patterns (Universal Approximation Theorem)

- ✓ Flexible MF types: ANFIS supports various membership function types: Bell-shaped, Gaussian, Triangular, Trapezoidal, and more that giving you the freedom to choose whichever fits your data best.

- ✓ Transparent rule base: The fuzzy rule base is explicit and straightforward, making it easy for anyone to inspect and for domain experts to validate.

2.6. Disadvantages

- Curse of dimensionality: The number of fuzzy rules grows exponentially with the number of inputs. For example, with 6 inputs each having 3 membership functions, the system generates $3^6 = 729$ rules [1], [2].

- Single-output constraint: Standard ANFIS only supports single-output (MISO) systems. Multiple outputs require multiple independent ANFIS models.

- Sensitivity to initialization: Results are highly sensitive to the initial values of the membership function parameters [5].

- ANFIS has trouble handling really complex or real-time data unless you simplify that data first. It can also "overfit," meaning it learns the training data too well but then doesn't work effectively on new data. This happens if the training data isn't complete enough or if it's trained for too long[3] .

2.7. FLP architecture

In the FLP (Fuzzy Logic Processor) architecture, the network is made up of processors performing operations on fuzzy variables (Figure 3.5). These processors are interconnected as shown in Figure 3.6, where the structure is organized into five layers: Input layer, fuzzification layer, hidden AND neuron layer, followed by a hidden OR neuron layer and defuzzification layer. We studied each layer in detail.

2.7.1. Fuzzification Layer:

The fuzzification layer is the entry point of the FLP, its role is to convert each numerical input into linguistic variables (fazzy variables). Each neuron in this layer implements a membership function $\mu A(x)$, that computes the degree to which an input value x belongs to a linguistic variable.

The parameters of these functions, such as the center c and width σ of a Gaussian membership function are not fixed by an expert. They are learned automatically during training via back-propagation. [6] In this thesis, Gaussian membership functions are used for the fuzzification

layer of the FLP model applied to the CSTR. Gaussian functions are smooth and differentiable everywhere, which makes learning easier and tends to produce more stable convergence than Discontinuous discrete linear function.

2.7.2. AND and OU neurons:

These two operators are performed using the t-norm chosen as the $product$ ($atb = ab$) and the s-norm as the algebraic sum ($asb = a + b - ab$).

a) OU function:

$U = OU(X, W)$, is defined using the s-t norms:

$$U = S_{i=1}^{I}[x_i t w_i] \quad \text{.....(3.19)}$$

By selecting the s and t norms, the OR function can be calculated as follows:

$$U = 1 - \prod_{i=1}^{I}(1 - x_i w_i) \quad \text{.....(3.20)}$$

The derivatives of the output with respect to the neuron weights and the inputs are given by:

$$\frac{\partial U}{\partial w_i} = x_i \prod_{\substack{j=1 \\ i \neq j}}^{I}(1 - x_j w_j) \quad \text{.....(3.21)}$$
$$\frac{\partial U}{\partial x_i} = w_i \prod_{\substack{j=1 \\ i \neq j}}^{I}(1 - x_j w_j) \quad \text{.....(3.22)}$$

b) AND function:

$U = AND(X, W)$, is defined using the s-t norms:

$$U = T_{i=1}^{I}[x_i s w_i] \quad \text{.....(3.23)}$$

By selecting the s and t norms, the AND function can be calculated as follows:

$$U = \prod_{i=1}^{I}(x_i + w_i - x_i w_i) \quad \text{.....(3.24)}$$

The derivatives of the output with respect to the neuron weights and the inputs are given by:

$$\frac{\partial U}{\partial w_i} = (1 - x_i) \prod_{\substack{j=1 \\ i \neq j}}^{I}(x_j + w_j - x_j w_j) \quad \text{.....(3.25)}$$
$$\frac{\partial U}{\partial x_i} = (1 - w_i) \prod_{\substack{j=1 \\ i \neq j}}^{I}(x_j + w_j - x_j w_j) \quad \text{.....(3.26)}$$

The output U of the AND or OR neuron can be followed by a nonlinear element f where $Z = f(U)$; this element can be taken as a sigmoid function modified by its parameters m and b.

$$Z = f(U) = \frac{1}{1 + e^{-(U-m)b}} \quad \text{.....(3.27)}$$

The gradient with respect to the inputs, and the weights of the connections becomes:

$$\frac{\partial Z}{\partial W} = \frac{\partial Z}{\partial U}\frac{\partial U}{\partial W} \quad \text{(3.28)}$$

$$\frac{\partial Z}{\partial U} = bZ(Z-1) \quad \text{(3.29)}$$

and with respect to the parameters of the sigmoid function:

$$\frac{\partial Z}{\partial b} = (U-m)Z(1-Z) \quad \text{(3.30)}$$

$$\frac{\partial Z}{\partial m} = bZ(1-Z) \quad \text{(3.31)}$$

Where $X = [x_i]$ and $W = [w_i]$ are input functions and weight vectors respectively.

2.7.3. Defuzzification layer

The final layer of the FLP converts the fuzzy output of the previous layer, into a single crisp numerical value that can drive an actuator or update a control signal. The most commonly used method for obtaining a numerical value at the output is the center of gravity, where:

$$y = \frac{\sum_{i=1}^{I} x_i w_i}{\sum_{i=1}^{I} x_i} \quad \text{(3.32)}$$

w_i: represent the weights (the centers of the output's membership functions).
x_i: are the input signals of the fifth layer (values of the membership functions of the output interface).

The output derivatives can be calculated with respect to the weights as:

$$\frac{\partial y}{\partial w_i} = \frac{x_i}{\sum_{k=1}^{I} x_k} \quad \text{(3.33)}$$

The output derivatives with respect to input signals, can be calculate such as:

$$\frac{\partial y}{\partial x_i} = \frac{w_i - y}{\sum_{k=1}^{I} x_k} \quad \text{(3.34)}$$

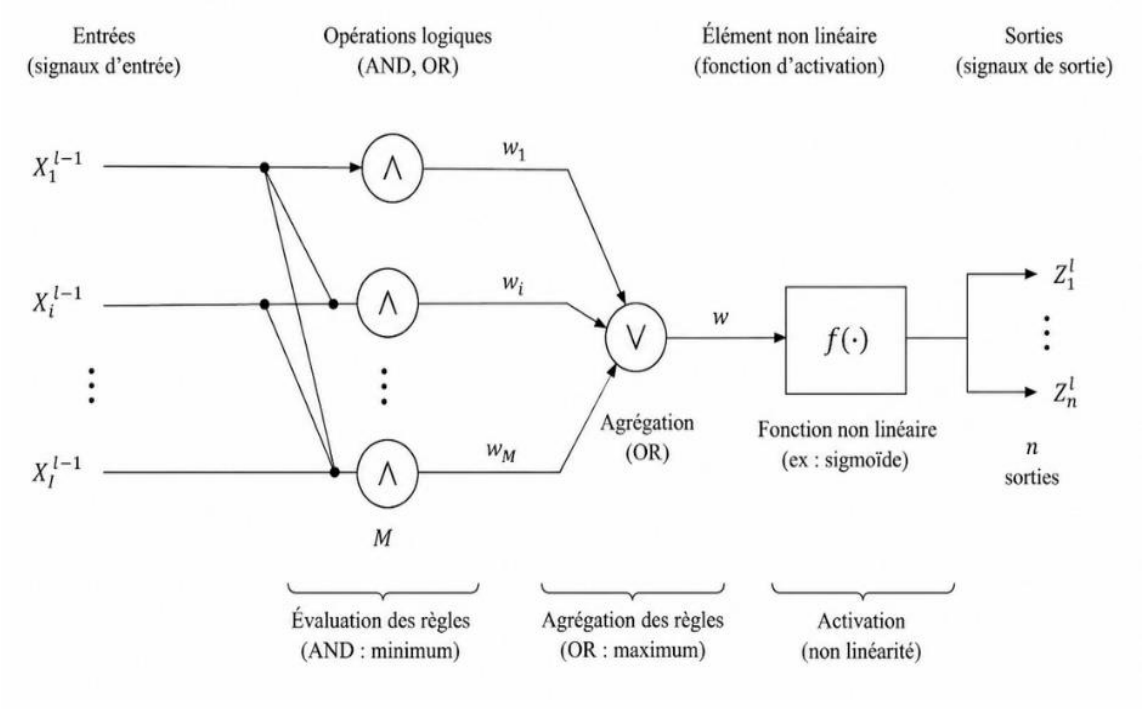


Fig.3.5. Logic processor

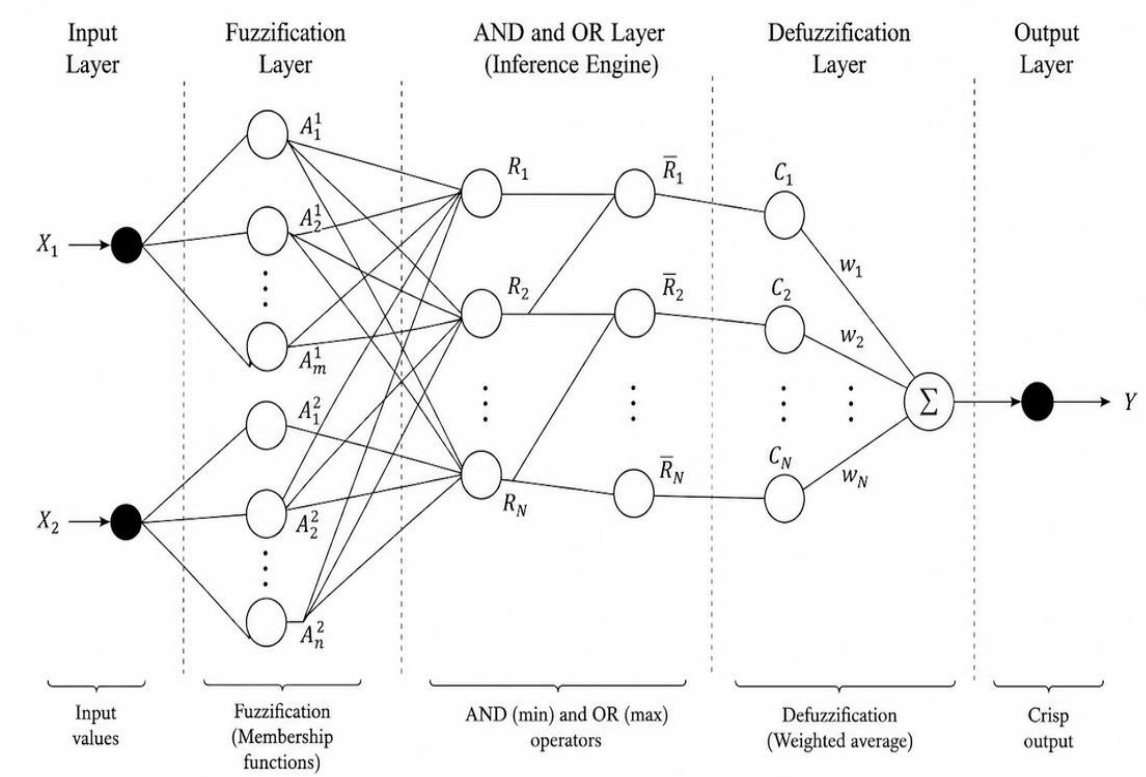


Fig.3.6. FLP Architecture

2.8. Advantages of FLP

✓ One of the main strengths of the FLP architecture is its ability to learn member-ship function parameters directly from data through back-propagation, eliminating the need for manual expert tuning[15]. The use of Gaussian membership functions further sup-ports this by ensuring smoothness and differentiability everywhere, which leads to more stable convergence during training[16].

✓ The architecture also preserves linguistic interpretability. Each layer has a clearly defined role fuzzification, rule aggregation, and defuzzification making it possible to un-derstand how inputs map to outputs, which is a significant advantage over black-box models in process control contexts[23].

✓ Additionally, all parameters are updated within a single unified training framework ensuring that the different components of the network adapt consistently with one an-other. [15]

2.9. Disadvantages of FLP

✓ The main limitation of the FLP is its sensitivity to initialization, as gradient-based training can converge to local minima depending on the starting parameter values[17]. The computational cost is also relatively high, since the AND and OR neurons rely on product-based algebraic operations that become increasingly demanding as the number of inputs or rules grows[18].

✓ The model's performance is also strongly tied to the quality and coverage of the training data any gap in the operating space will directly affect generalization[22]. Finally, the network topology must be defined before training, meaning the number of mem-bership functions and neurons per layer cannot be adjusted automatically, which re-quires some prior knowledge or experimentation on the part of the designer[17].

2.10. Conclusion

In this chapter, we laid out the two main ways to think about Fuzzy Neural Networks: first, the functional approach, where neural networks mainly support an existing fuzzy system; and second, the structural approach, which integrates fuzzy logic directly into the neural network's architecture. In this chapter, we focus on studying the theory of two main architecture of fuzzy neural networks:

ANFIS architecture, which is probably the best-known example of a structural FNN model. We went into detail about its five-layer setup, the specific math behind each layer, and its hybrid learning algorithm that cleverly combines Least Squares Estimation with Back-propagation. This hybrid method is really what makes ANFIS so efficient: it takes advantage of how linear the output function is with respect to the consequent parameters, allowing for fast and accurate training. Sure, ANFIS has its limitations, particularly with scalability and that single-output restriction. But even so, it remains one of the most effective and widely used neuro-fuzzy models, with a solid track record in areas like control systems, prediction, and adaptive learning environments. Coming up in the next chapter, we'll actually apply this ANFIS framework to tackle our specific problem.

The FLP architecture was then examined through its three layers. The fuzzification layer, maps crisp inputs to membership degrees using Gaussian functions whose parameters are learned during training. The AND/OR reasoning layer, evaluates fuzzy rule antecedents, using the algebraic product instead of the minimum operator to ensure differentiability. The defuzzification layer aggregates rule firing strengths via the Center of Gravity method, producing a smooth and continuous output that is itself parameterized and trainable.

The practical importance of these properties will become clear in the following chapter, where the chapter 4 of simulation can rules between FLP and ANFIS architecture in non-linear system modeling.

Chapter 3:

Identifying the Concentration in a Chemical Reactor

3.1 Introduction

In this chapter we present the problem of identifying the concentration in a chemical reactor using the ANFIS and FLP neuro-fuzzy networks, considering two identification structures: a series-parallel structure (one step ahead) and a parallel structure (long range). Before presenting the simulation results, we briefly introduce a few notions on the identification of nonlinear systems.

3.2 Identification of Nonlinear Systems

Unlike the identification of linear systems, which is a well-established theory, the identification of nonlinear systems is more difficult. It mainly consists of selecting the model structure and estimating its parameters. In other words, building an acceptable model, shown in Figure (4.1), that would produce an output $\hat{y}(k)$ approximating y(k) when subjected to the same input U(k) as the system being identified, with the goal of optimizing a certain performance criterion based on the error between the output of the system being identified and the identification model. It should be noted that several techniques for identifying nonlinear systems are available in the literature.

Identification is carried out in two stages: the qualitative stage (characterization) and the quantitative stage (parameter estimation).

3.2.1 Qualitative stage

This involves the mathematical representation of the system. When the physical laws are known, the model structure is known, and we then obtain the knowledge-based model. On the other hand, if we are unaware of the phenomena involved, we settle for a mathematical description with no apparent connection to physical reality, in this case we speak of a representation model built from a series of measurements of the inputs and outputs of the process being identified.

Narendra proposed several classes of input/output identification models for nonlinear systems.[ ] In our case, we used the model described by the following equation:

$$\hat{y}(k+1) = f[y(k), y(k-1), ..., y(k-n+1) ; U(k), U(k-1), ..., U(k-m+1)] \quad (4.1)$$

Where [Y(k), U(k)] represents the input-output pair of a SISO (single-input single-output)

system at instant k, and $m \leq n$ (this model is the most general representation of NARMA systems). []

There are two identification structures: the series-parallel structure and the parallel structure. Using the previous model for identification, we obtain:

a) Series-parallel structure

This structure is based on a one-step-ahead prediction: the estimate at step k+1 is obtained from the previous measurements.

$$\hat{y}(k+1) = f[Y(k), Y(k-1), ..., Y(k-n+1) ; U(k), U(k-1), ... , U(k-m+1)] \quad (4.2)$$

Where Y is the output of the system being identified.

b) Parallel structure

In this structure, called long-range prediction, the estimate at k+1 is obtained from the previous estimates.

$$\hat{y}(k+1) = f[\hat{y}(k), \hat{y}(k-1), ..., \hat{y}(k-n+1) ; U(k), U(k-1),....., U(k-m+1)] \quad (4.3)$$

Where $\hat{Y}$ is the output of the identification model.

3.2.2 Quantitative stage

In this stage, the parameters of the identification model are adjusted based on the minimization of an error criterion. This criterion represents the difference between the output of the system being identified and the output of the identification model.

3.2.3 Neuro-Fuzzy Identification Algorithm

Neuro-fuzzy system identification algorithm combines the learning ability of artificial neural networks with the interpretability of fuzzy logic to build a model capable of representing complex nonlinear systems from experimental data [4], [9]. In this framework, the input variables are first transformed into fuzzy membership degrees through membership functions. Then, a set of if-then fuzzy rules is constructed, followed by the computation of each rule's firing strength and the aggregation of partial outputs to produce the final model output [4].

Neuro-fuzzy identification consists of building a neuro-fuzzy model that approximates the dynamics of the system being identified. In general, the different steps of a neuro-fuzzy identification are:

1. Choice of the neuro-fuzzy model structure
2. Choice of the input variables and output variables of the neuro-fuzzy model.
3. Initialization of the weights of each layer of the neuro-fuzzy model.
4. Carry out neuro-fuzzy reasoning, passing through the fuzzification layer, the hidden layers representing the inference engine, and the defuzzification layer, to obtain the output of the neuro-fuzzy model.
5. Estimation of the parameters of the neuro-fuzzy model, using a carefully chosen adjustment algorithm, based on the error between the output of the system and that of the neuro-fuzzy system.
6. Validation of the neuro-fuzzy model.

3.3 Description of the Model

The process is a continuous, perfectly mixed, irreversible exothermic reactor, A → B, operating at constant volume. The reactor is cooled by circulating water vapor (steam) through the jacket. The concentration produced is represented by ca(t), which is controlled by manipulating the coolant flow rate $q_c(t)$.

The reactor dynamics are described by two coupled non-linear deferential equations [2], [5]:

$$\frac{d\,ca(t)}{dt} = \left(\frac{q}{v}\right)(ca_0 - ca(t)) - k_0\, ca(t)\, e^{\left(-\frac{E}{R\,T(t)}\right)} \qquad (4.2)$$

$$\frac{dT(t)}{dt} = \left(\frac{q}{v}\right)(T_0 - T(t)) - k_1\, ca(t) e^{\left(-\frac{E}{R\,T(t)}\right)} + k_2 q_c(t)\left(1 - e^{\left(-\frac{k_3}{q_c(t)}\right)}\right)(T_{c0} - T(t)) \qquad (4.3)$$

Where: q_c: is the cooling flow rate.

$$k_1 = \frac{-\Delta H k_0}{p c_p} \quad \text{.....(4.4)}$$

$$k_2 = \frac{p_c c_{pc}}{p c_p v} \quad \text{.....(4.5)}$$

$$k_2 = \frac{h_a}{p_c c_{pc}} \quad \text{.....(4.6)}$$

All other parameters ($k_0, k_1, k_2, k_3, E/R, V$) are thermodynamic and chemical constants. [2] The nominal reactor parameters are show in table (4.1)

3.3.1. Operating Conditions

The process simulation is performed by solving the two equations (4.4) and (4.5) using the fourth-order Runge-Kutta method, with an integration step $dt = 6\,sec$. The control variable qc (l/sec) is introduced randomly within the interval [1.5, 1.8333].

Nominal operating point: ca = 0.1 mol/l, T = 438.54 K, qc = 1.7235 l/sec.

3.4 Identification of Concentration

3.4.1. Network Structure

In this section we address neuro-fuzzy identification using the two structures, series-parallel and parallel, employing the two methods ANFIS and FLP.

For the series-parallel structure, the network is trained using the following vector X:

$$X = [ca(k-1), ca(k-2), q_c(k-1)] \quad (4.7)$$

The concentration ca is obtained from the simulation model of the system to be identified.

For the parallel structure, the network is trained using the following vector X:

$$X = [F(k-1), F(k-2), q_c(k-1)] \quad (4.8)$$

F is the output of the identification model

Parameter	Description	Nominal Value
$\boldsymbol{q}$	Product inflow rate	1.666 l/sec
$\boldsymbol{v}$	Reactor volume	100 l
$\boldsymbol{k_0}$	Reactor speed (pre-exponential factor)	$\boldsymbol{1.2 \times 10^9}$ sec^{-1}
$\boldsymbol{E/R}$	Activation energy / Gas constant	$\boldsymbol{10^4}$ K
$\boldsymbol{T_0}$	Product temperature at inlet	350 K
$\boldsymbol{T_{c0}}$	Cooling temperature	350 K
$\boldsymbol{\Delta H}$	Heat of reaction	$\boldsymbol{-2 \times 10^5}$ cal/mol
$\boldsymbol{c_p, c_{pc}}$	Specific heat capacities	1 cal/(g·K)
$\boldsymbol{\rho, \rho_c}$	Liquid densities	$\boldsymbol{10^3}$ g/l
$\boldsymbol{h_a}$	Heat transfer coefficient	$\boldsymbol{1.166 \times 10^4}$ cal/(sec·K)
$\boldsymbol{ca_0}$	Inlet concentration	0.1 mol/l

Tab.4.1. Nominal reactor parameters

3.4.2. Identification by using ANFIS Network

The network contains the following neurons:

9 input neurons (fuzzification) corresponding to three linguistic values for each linguistic variable $Ca(k), Ca(k-1), Qc(k)$.

27 neurons for the second, third, and fourth layers.

1 neuron in the output layer.

The adjustment algorithm used is the gradient descent algorithm.

The fuzzy sets are chosen in a Gaussian form given by equation (3-20).

The parameters of the membership functions (c, the center of the membership function, and d, the width of the membership function) are initialized as indicated in tables (4.5) and (4.6).

The linguistic values are «LOW», «MEDIUM», «HIGH».

Table (4.5) Fuzzification of Ca(k-1) and Ca(k)

	Values of (c, d)	**Fuzzy Variables**
1	(0.076, 0.02)	L
2	(0.11, 0.02)	M
3	(0.143, 0.02)	H

Table (4.6) Fuzzification of qc(k)

	Values of (c, d)	**Fuzzy Variables**
1	(1.55, 0.11)	L
2	(1.66, 0.11)	M
3	(1.77, 0.11)	H

The values of c and d are adjusted with different learning steps:

μc = [0.001 0.0001 0.0001; 0.0001 0.0001 0.0001; 0.01 0.001 0.001]

μd = [0.0001 0.0001 0.0001; 0.0001 0.0001 0.0001; 0.001 0.001 0.001]

The parameters of the consequence functions (of the fourth layer) are initialized as follows:
for the series-parallel structure at 0.01
for the parallel structure, they are taken randomly within the interval [0, 1]. These parameters are adjusted with a learning step equal to 0.001.

a) Training Results

The network is trained for 200*3999 iterations (3999 data vectors repeated 200 times). After training, the connection weights for the ANFIS and FLP networks are obtained.

We give below some of the parameters of the optimal ANFIS network.

For the ANFIS network, series-parallel structure, the membership function parameters have the values given in tables (4.2), (4.3), (4.4).

Table (4.2) Fuzzification of Ca(k-1)

	Values of (c, d)	**Fuzzy Variables**
1	(0.0284, 0.0092)	L
2	(0.1024, 0.0220)	M
3	(0.1780, 0.0106)	H

Table (4.3) Fuzzification of Ca(k)

	Values of (c, d)	**Fuzzy Variables**
1	(0.1271, 0.0274)	L
2	(0.1723, 0.0102)	M
3	(0.1777, 0.0179)	H

Table (4.4) — Fuzzification of qc(k)

	Values of (c, d)	**Fuzzy Variables**
1	(0.9500, 0.0993)	L
2	(1.84, 0.1364)	M
3	(2.6417, 0.0871)	H

For the parallel structure, the new values of the membership function parameters are illustrated in tables (4.5), (4.6), (4.7).

Table (4.5) Fuzzification of ca(k-1)

	Values of (c, d)	**Fuzzy Variables**
1	(0.1, 0.0201)	L
2	(0.022, 0.0207)	M
3	(0.106, 0.0101)	H

Table (4.6) Fuzzification of ca(k)

	Values of (c, d)	**Fuzzy Variables**
1	(0.0274, 0.0276)	L
2	(0.0102, 0.0173)	M
3	(0.0179, 0.0097)	H

Table (4.7) Fuzzification of qc(k)

	Values of (c, d)	**Fuzzy Variables**
1	(0.0993, 0.0924)	L
2	(0.1364, 0.1107)	M
3	(0.0871, 0.1170)	H

b) Validation

As a result of our simulation, we calculate and trace the RMSE curve (**Root Mean Square Error)**. It's one of the most common metrics used to evaluate how well a model (ANFIS or FLP), fits or predicts a nonlinear system's behavior.

$$RMSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \qquad (4.9)$$

Where: yi: real value.
$\hat{y}_i$: estimated value.

c) Training Results

- **ANFIS network (series parallel structure)**

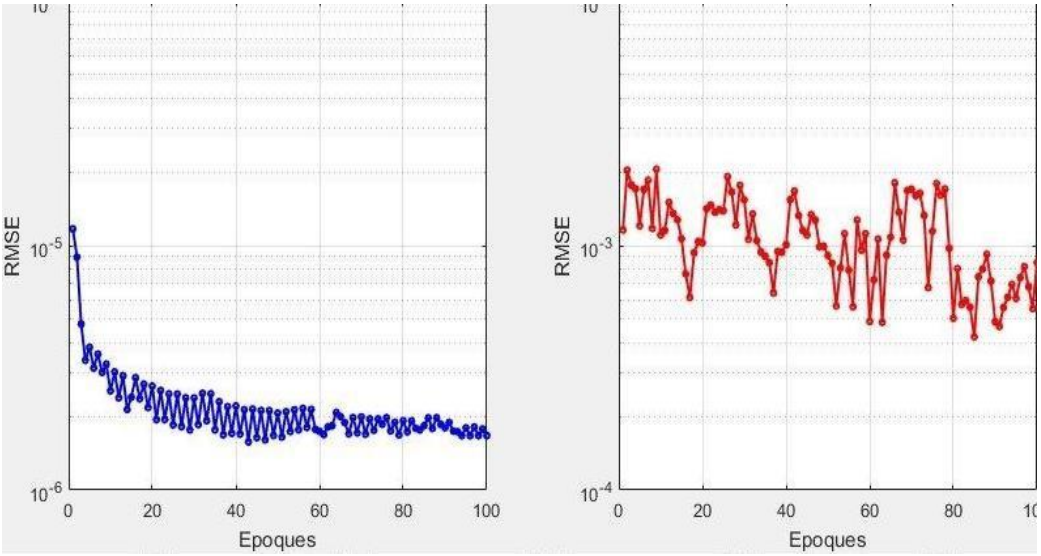


Fig 4.1. Convergence of the Series-Parallel Neuro-Fuzzy Model**(**series parallel structure)

For concentration ca **(blue)**: convergence is excellent: the RMSE drops rapidly within the first 10 epochs, falling from approximately 10^{-5} to a stable plateau around 2×10^{-6} mol/L. This behavior is characteristic of well-conditioned learning, free of overfitting or numerical instability.

For temperature T **(red)**: the behavior is markedly different: the RMSE remains around 10^{-3} K and exhibits noticeable oscillations throughout training, indicating a slower and less smooth convergence compared with the concentration model. This suggests that the temperature output presents a more challenging identification problem, likely related to its strongly nonlinear coupling with ca through the reaction kinetics[2].

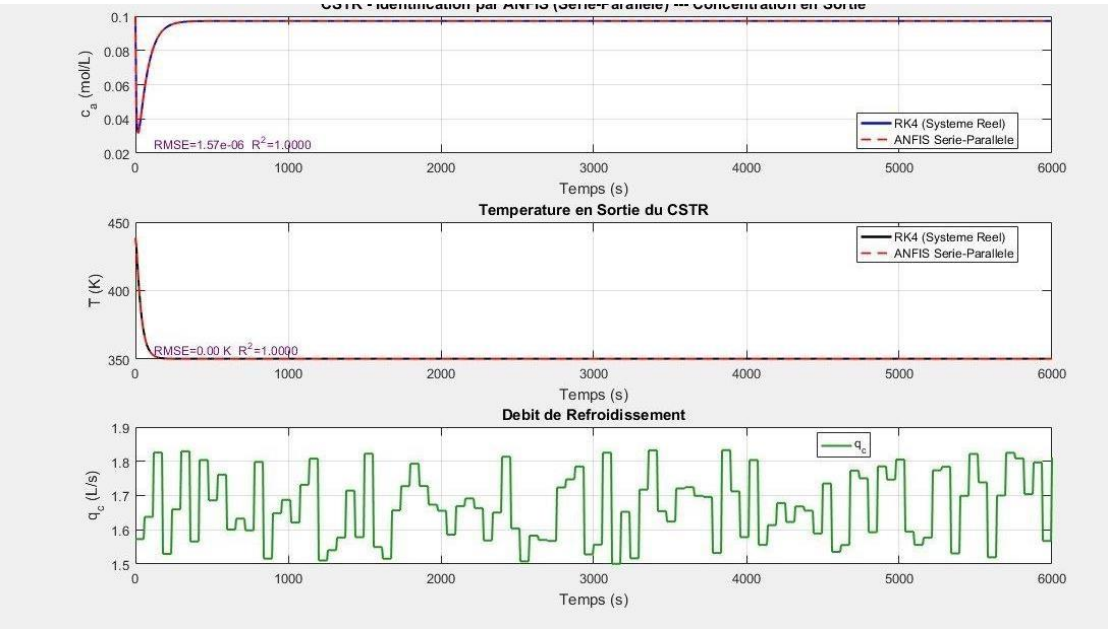


Fig 4.2. ANFIS series parallel Learning

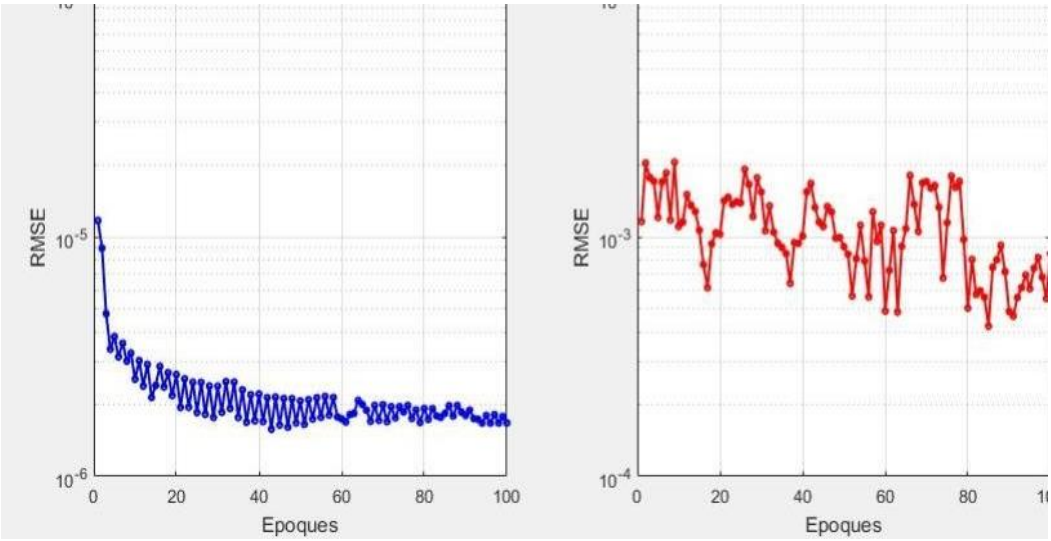
The second figure confirms the overall model performance in simulation over 6000 seconds. For *ca*, the RK4 (blue) and series-parallel ANFIS (red dashed) curves overlap very closely across the entire time horizon , including during the initial transient phase (0–500 s), which is often the most challenging. The RMSE = 1.57×10^{-6} mol/L demonstrate a very good restitution of the true dynamic behavior. For *T*, the visual agreement is equally strong, with RMSE = 0.00 K displayed (rounded value). It is worth noting that the cooling flow rate q*c* (green, bottom subplot) exhibits a well-constructed pseudo-random excitation signal, covering a wide range of values between 1.5 and 1.85 L/s

Results Interpretation

These results strongly validate the effectiveness of the series-parallel ANFIS architecture in capturing the nonlinear dynamics of the CSTR. The identified model faithfully reproduces the dynamic trajectories of the true system integrated by RK4. The oscillatory behavior observed in the temperature training error does not compromise the final model accuracy, as confirmed by the numerical performance metrics. Nevertheless, it highlights that the temperature dynamics present a more demanding identification challenge, warranting further investigation in future work.

- **ANFIS network (parallel structure)**

Fig 4.3. Convergence of the Series-Parallel Neuro-Fuzzy Model(parallel structure)



For concentration ca **(blue)**: convergence is excellent: the RMSE decreases rapidly within the first 10 epochs and stabilizes around 2×10^{-6} mol/L, indicating well-conditioned training with no signs of overfitting.

For temperature T **(red)**: the error remains around 10^{-3} K with persistent oscillations throughout training, indicating a slower and less smooth convergence compared to the concentration model. This confirms that the temperature output presents a more demanding identification problem regardless of the model structure used.

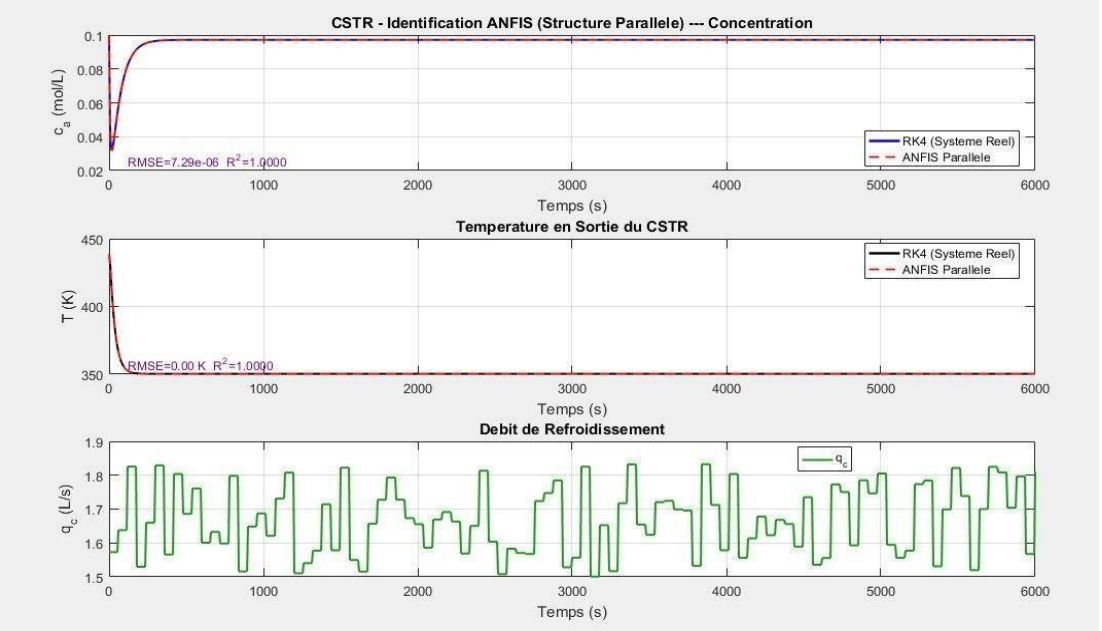


Fig4.4 ANFIS parallel learning

This figure (Fig.4.4) is the most informative for comparing the two architectures. The parallel ANFIS model operates in closed-loop prediction mode, relying entirely on its own estimated states[8].

For *ca*, the RK4 and parallel ANFIS curves remain visually very close over the full 6,000-second horizon. The reported metrics are: RMSE = 7.29×10^{-6} mol/L

For T, the visual agreement is equally strong: RMSE = 0.00 K (rounded display value)

Result Interpretation

The parallel ANFIS structure demonstrates very good identification performance, with errors remaining small in absolute terms. The systematic degradation relative to the series-parallel results is physically meaningful and academically important to discuss.

In the series-parallel structure, the model is supplied with true system states during training and validation, which eliminates error propagation. In the parallel structure, the model operates autonomously meaning any small estimation error is fed back and can accumulate over time. The fact that accuracy remains high despite this feedback mechanism strongly validates

the generalization capability of the identified ANFIS model.

This comparison provides a rigorous basis for two conclusions:

The series-parallel structure offers superior accuracy, benefiting from access to true system states.

The parallel structure better reflects real deployment conditions, where true states are unavailable. Both results together constitute a complete and credible validation of the neuro-fuzzy identification approach applied to this CSTR[7], [8].

3.4.3. Identification by FLP Network

For both structures, the FLP network contains the following neurons:

- 9 neurons in the input layer (fuzzification), corresponding to three linguistic values for each linguistic variable (k), $Ca(k-1)$, $Qc(k)$.
- 27 AND neurons, corresponding to the 27 chosen inference rules
- 5 OR neurons
- 1 neuron in the output layer corresponding to the defuzzification neuron

The fuzzy sets of the inputs/output are chosen to be triangular and symmetric.
The fuzzification results of the flow rate qc and the concentration ca at the input, with three linguistic values, are given in the two tables (4.2) and (4.3) respectively.

The linguistic values are «LOW», «MEDIUM», «HIGH».

The weights of the connections preceding the AND neurons are initialized to 0, while the weights connecting the AND layer and the OR layer are initialized to 0.1. These latter weights represent the only parameters adjusted in the network. The adjustment is performed using the gradient descent algorithm with a learning step equal to 0.01. The weights of the last layer represent the centers of the symmetric membership functions of the output.

The fuzzification of the output Ca(k+1) with five linguistic values is given in table (4.4). The linguistic values are «VERY LOW», «LOW», «MEDIUM», «HIGH», «VERY HIGH».

Table (4.8) Fuzzification of the input qc

	Interval of qc	**Fuzzy Variables**
1	qc ≤ 1.6665	L
2	1.53 < q_c < 1.7498	M
3	qc ≥ 1.6665	H

Table (4.9) Fuzzification of the input ca

	Interval of ca	**Fuzzy Variables**
1	ca ≤ 0.09	L
2	0.06 < ca < 0.12	M
3	ca ≥ 0.09	H

Table (4.10) Fuzzification of the output ca

	Interval of ca	**Fuzzy Variables**
1	ca ≤ 0.08	VL
2	0.06 ≤ ca < 0.1	L
3	0.08 < ca < 0.115	M
4	0.1 ≤ ca < 0.13	H
5	ca ≥ 0.115	VH

a) Validation

To facilitate a direct comparison, figure (Fig.4.5) brings together the RK4 reference, the Series-Parallel FLP, and the Parallel FLP on the same concentration plot, followed by the respective identification errors for each structure.

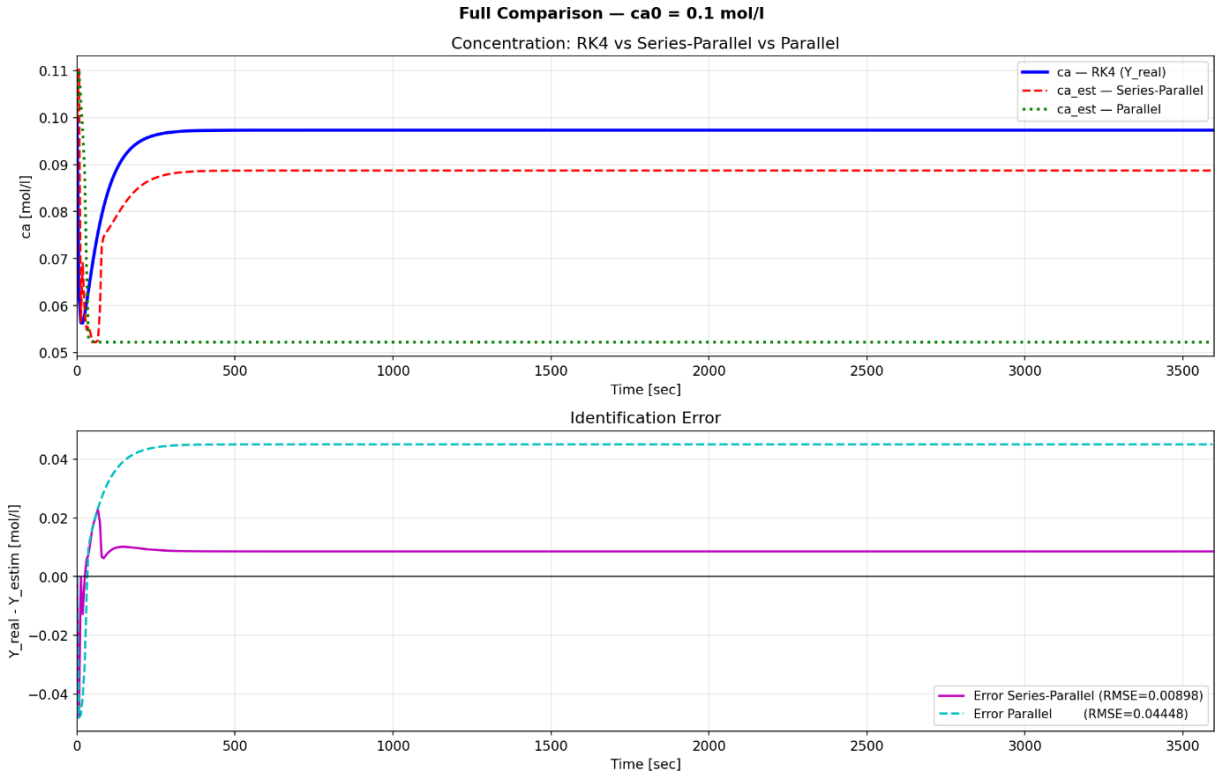


Fig4.5 Full comparison: RK4 vs Series-Parallel vs Parallel FLP identification concentration and RMSE error

Discussion

When the inlet concentration is set to $ca0$ = 0.1 mol/l, the CSTR operates close to its nominal point and the concentration dynamics remain confined within the narrow range [0.056, 0.100] mol/l.

Under these conditions, the Series-Parallel structure performed noticeably better, with an RMSE of 0.008981, a result that is expected, since the model is continuously corrected by real RK4 measurements at each step, leaving no room for error to build up over time.

Discussion

When the inlet concentration is set to $ca0$ = 0.1 mol/l, the CSTR operates close to its nominal point and the concentration dynamics remain confined within the narrow range [0.056, 0.100] mol/l.

Under these conditions, the Series-Parallel structure performed noticeably better, with an RMSE of 0.008981, a result that is expected, since the model is continuously corrected by real RK4 measurements at each step, leaving no room for error to build up over time.

The Parallel structure, on the other hand, showed a higher error (RMSE = 0.044477), which reflects the natural consequence of removing that corrective mechanism: small estimation mistakes compound through the feedback loop and accumulate over the course of the simulation. That said, both structures successfully followed the concentration profile without departing from the physically admissible range, confirming that the 25-rule Mamdani FLP captures the essential dynamics of the reactor even in the more constrained autonomous mode.

3.5 Conclusion

Throughout this chapter, a complete Fuzzy Neuron Network architecture was designed and tested for identifying the reactant concentration ca(t) in a nonlinear CSTR. Built around the Takagi suggino and Mamdani inference model with Gaussian membership functions and operating at the correct inlet concentration of $ca0$ = 0.1 mol/l,

First, this study compared two ANFIS-based architectures [4]. Both models demonstrated sufficient learning capacity, achieving low training error. However, their behavior during simulation differs significantly due to the way input feedback is managed.

The Series-Parallel structure uses measured outputs during training, which stabilizes the learning process and produces accurate simulation results with negligible deviation from the reference trajectory [4]. The Parallel structure uses predicted outputs as feedback, which introduces error accumulation during the transient phase, leading to observable deviations at the start of simulation before gradually converging near steady state [5].

The key distinction between the two architectures lies in how the input regressor is constructed: using real measured outputs versus model-generated predictions. Although this

appears to be a minor implementation difference, it has a substantial impact on prediction accuracy, training speed, and simulation stability.

In practice, the Series-Parallel structure is recommended for system identification and validation tasks, while the Parallel structure is better suited for autonomous free-running simulation and control-oriented applications where recursive prediction is required [6].

Second, the FLP architecture consistently kept the estimated concentration within the physically meaningful range of [0.056, 0.100] mol/l.

The five-layer architecture with its 25 IF-THEN rules proved both readable and effective. Between the two configurations tested, the Series-Parallel structure stood out for its accuracy (RMSE = 0.008981), while the Parallel structure (RMSE = 0.044477), despite its larger error, offered a more autonomous mode of operation that does not depend on real-time plant measurements, a quality that could prove advantageous for actual deployment in conditions where sensor feedback is limited or delayed.

Chapter 4:

General Conclusion

General Conclusion

In this work we have demonstrated the effectiveness of the fuzzy neural network approach in modeling unknown nonlinear systems. The main objective was to identify the output concentration of the reactor from its measurable inputs, taking advantage of the ability of neuro-fuzzy systems to capture complex nonlinear dynamics while remaining interpretable.

The thesis was organized into five chapters. The first chapter introduced the general context and motivation of the work. The second chapter reviewed the theoretical background needed for the rest of the thesis, covering artificial neural networks and fuzzy logic, with particular attention to the Mamdani and Sugeno inference models that serve as a basis for the hybrid architectures studied afterward.

The third chapter was dedicated to fuzzy neural network architectures. Two models were studied in detail: the Fuzzy Logic Processor (FLP) and the Adaptive Neuro-Fuzzy Inference System (ANFIS). Both share the same general idea of integrating fuzzy inference into a neural network structure, but they differ in how they represent rules and how they learn. ANFIS is built on a first-order Sugeno model and uses a hybrid learning algorithm that combines least squares estimation for the consequent parameters with gradient descent for the premise parameters. FLP, on the other hand, relies on product-based t-norms and s-norms within its AND and OR neuron layers, and uses a center-of-gravity method at the output.

The fourth chapter presented the application of these two architectures to the CSTR identification problem. Each model was tested under two structural configurations: the series-parallel structure, where the real plant output is used as feedback during training, and the parallel structure, where the model feeds back its own predicted output. The training and validation data were generated from a simulation of the CSTR under realistic conditions, including random disturbances applied to the coolant flow rate.

The results of all four configurations were compared in terms of training accuracy, validation performance, and convergence behaviour. The ANFIS model in its series-parallel configuration gave the best results overall. It showed the lowest training and validation errors, converged in a stable manner, and was able to track the reactor concentration well across the entire operating range. The parallel ANFIS configuration also gave satisfactory results, though slightly less accurate. The FLP models, in both configurations, produced acceptable results but were less precise than ANFIS under the same training conditions.

These results confirm that ANFIS in the series-parallel configuration is the most suitable architecture for identifying the concentration in the CSTR among the four configurations tested. The main reason behind this is its hybrid learning algorithm, which avoids having to estimate all parameters by gradient descent alone, by solving analytically for the consequent parameters, it reduces the search space and converges faster and more reliably than a fully gradient based approach like FLP.

More generally, this work shows that neuro-fuzzy systems are a relevant and practical choice for modeling nonlinear chemical processes. They combine the learning capability of neural networks with the transparency of fuzzy rules, which makes them easier to validate and interpret compared to purely black-box models.

We have detailed each part of our thesis as required, and we have included descriptive diagrams intended to give a clear understanding to those who wish to consult this modest work.

As for future work, several directions can be explored. The identified ANFIS model could be used inside a closed-loop control system, for example a model predictive controller; to regulate the reactor concentration under changing setpoints and coolant disturbances. It would also be interesting to test the model under noisier or more degraded conditions to evaluate its robustness.

Finally, extending the approach to identify multiple outputs simultaneously, such as both concentration and temperature, would be a natural next step toward a more complete representation of the CSTR behavior.

References

[1] Narendra, K. S., & Parthasarathy, K. (1990). Identification and control of dynamical systems using neural networks. *IEEE Transactions on Neural Networks*, 1(1), 4–27.

[2] Seborg, D. E., Edgar, T. F., Mellichamp, D. A., & Doyle, F. J. (2016). *Process Dynamics and Control*. John Wiley & Sons.

[3] Hagan, M. T., Demuth, H. B., & Beale, M. H. (1996). *Neural Network Design*. PWS Publishing Co.

[4] Jang, J. S. (1993). ANFIS: Adaptive-network-based fuzzy inference system. *IEEE Transactions on Systems, Man, and Cybernetics,* 23(3), 665–685.

[5] Bequette, B. W. (2003). *Process Control: Modeling, Design, and Simulation*. Prentice Hall.

[6] Wang, L. X. (1997). *A Course in Fuzzy Systems and Control*. Prentice Hall.

[7] Nelles, O. (2020). *Nonlinear System Identification: From Classical Approaches to Neural Networks and Fuzzy Models*. Springer Nature.

[8] Ljung, L. (1999). *System Identification: Theory for the User*. Prentice Hall.

[9] Babuška, R. (2012). *Fuzzy Modeling for Control*. Springer Science & Business Media.

[10] Passino, K. M., & Yurkovich, S. (1998). *Fuzzy Control*. Addison-Wesley.

[11] Soderstrom, T., & Stoica, P. (1989). *System Identification*. Prentice Hall.

[12] H.-P. Phan and X.-T. Tran, Design and Modeling of a Voltage-Frequency Controller for Network-on-Chip Routers based on Fuzzy-Logic, VNU Journal of Science, vol. 31, no. 2, pp. 56-65, 2015.

[13] T. J. Ross, Fuzzy Logic with Engineering Applications, 3rd ed. Wiley, 2010.

[14] C. T. Lin and C. S. G. Lee, Neural Fuzzy Systems: A Neuro-Fuzzy Synergism to Intelligent Systems. Upper Saddle River, NJ: Prentice-Hall, 1996.

[15] D. Nauck, F. Klawonn, and R. Kruse, Foundations of Neuro-Fuzzy Systems. Chichester, UK: John Wiley & Sons, 1997.

[16] A. Al-Hmouz, J. Shen, R. Al-Hmouz, and J. Yan, "Modeling and Simulation of an Adaptive Neuro-Fuzzy Inference System (ANFIS) for Mobile Learning," IEEE Transactions on Learning Technologies, vol. 5, no. 3, pp. 226–237, 2012.

[17] L. A. Zadeh, "Fuzzy sets," Information and Control, vol. 8, no. 3, pp. 338–353, 1965

[18] H R. Berenji and P. Khedkar, "Learning and tuning fuzzy logic controllers through reinforcements," IEEE Transactions on Neural Networks, vol. 3, no. 5, pp. 724–740, 1992.

[19] J. J. Buckley and Y. Hayashi, "Fuzzy neural networks: A survey," Fuzzy Sets and Systems, vol. 66, no. 1, pp. 1–13, 1994

References