RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE

MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITÉ IBN-KHALDOUN DE TIARET
FACULTÉ DES SCIENCES APPLIQUEES
DÉPARTEMENT DE GENIE ELECTRIQUE

Support de cours

Techniques de L'intelligence Artificielle

Domaine : Sciences et Technologie

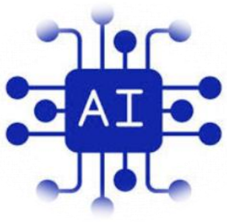
Filière : Électrotechnique

Spécialité: Commandes électriques

Manuscript prepared according to the provided program approved and confirmed by the CPNDST

Prepared by**: M^r MIHOUB Youcef MCA**

Experts:	**M^r Maasacri Mustapaa**	**MCA**
	M^r Bensattalah Aissa	**MCA**



2025 2026

Table of contents

OUTLINE

This course material has been prepared for Master's students in Electrical Control and Electrical Engineering, based on lecture notes delivered in French. It has been updated and revised in English, incorporating additional resources. The document is organized into four chapters:

Chapter 1 introduces the fundamentals of Artificial Intelligence, including its definitions, history, and applications.

Chapter 2 focuses on fuzzy reasoning, providing the first steps toward understanding fuzzy logic theory. The examples are designed to introduce fuzzy logic in a simple and progressive manner, starting from basic formulas and gradually moving toward more complex calculations. A practical choice is to use MATLAB, which offers the advantage of working with either command lines or specialized toolboxes such as the Fuzzy Logic Toolbox.

Chapter 3 presents the theory of fuzzy logic and fuzzy sets, using examples and comparisons with classical set theory. Two examples are provided, along with their solutions and MATLAB code.

Chapter 4 demonstrates the application of fuzzy logic in control systems, using real-world process control examples.

Chapter 5 is concerned with the use of artificial neural networks from the beginning to the application in control and identification, using Somme examples with MATLAB.

.

Chapter 6 is relative to the different neuro-fuzzy architectures and in particular ANFIS with MATLAB graphical user interface and line commands

Chapter 7 demonstrates the basic notions and the used vocabulary in Genetic algorithms. Somme examples with MATLAB were used to explain more details relative to this part.

Chapter 8 gives an overview about Particul swarm Optimization with an example resoloved wit MATLAB instructions

An interesting work home has been added in order to evaluate every step of this document .

.

1.1. Introduction

The term "**artificial intelligence**" was first proposed at the Dartmouth conference in 1956 to designate the field of research on a fundamental problem whose main idea is the possibility of designing an intelligent machine. Intelligence commonly refers to the potential of an individual's, animal or human, mental and cognitive abilities, enabling them to solve a problem or adapt to their environment. It is often synonymous with the brain. By extension, applied to machines: this is known as artificial intelligence. So a machine will be considered intelligent if it reproduces the behavior of a human being in a specific or general domain. A machine will be considered intelligent if it models the functioning of a human being.

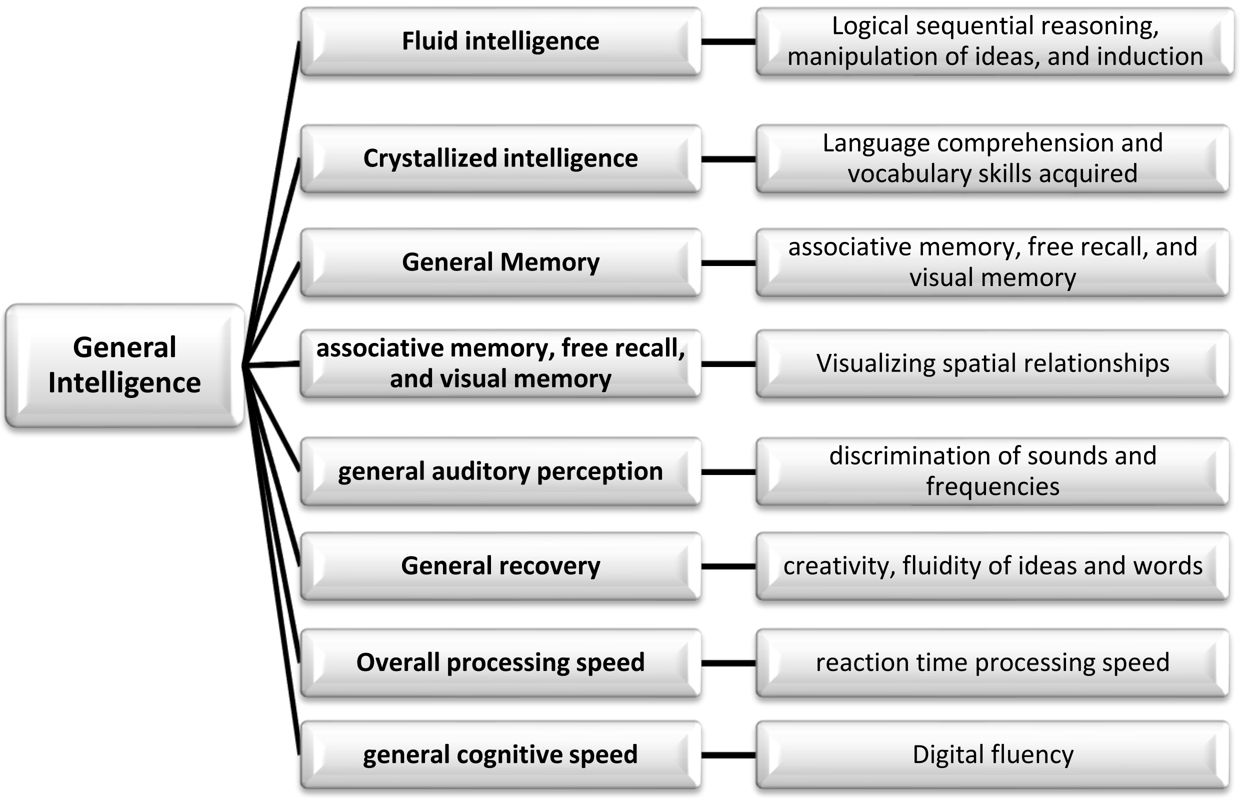

General Intelligence
Fluid intelligence
Logical sequential reasoning, manipulation of ideas, and induction
Crystallized intelligence
Language comprehension and vocabulary skills acquired
General Memory
associative memory, free recall, and visual memory
associative memory, free recall, and visual memory
Visualizing spatial relationships
general auditory perception
discrimination of sounds and frequencies
General recovery
creativity, fluidity of ideas and words
Overall processing speed
reaction time processing speed
general cognitive speed
Digital fluency

Figure I. 1 Classification of general intelligence.

1.2. Definition of Artificial Intelligence

Artificial intelligence is defined as the project of building intelligent machines—artifacts. It is favored in fields where there are no algorithms readily available to machines, or where machines can perform behaviors typically associated with human intelligence, such as:

• Driving a vehicle while avoiding traffic jams

• Translating a foreign language

• Holding a conversation

• Deleting spam emails from your inbox…

The four definitions can be grouped together in a table.

	Définitions	
think	A system that thinks like humans. **Complex**	A system that thinks rationally **Limited**
Act	A system that acts like humans Turing test (1950) **Theoretical**	A system that acts rationally **Pragmatic**
	Empirical	**Theoretical**

1.3. Approaches to Artificial Intelligence

Artificial intelligence approaches can be classified into two distinct groups:

• Reasoning-based approaches, which generally employ rules of the form IF condition then conclusion; fuzzy logic is one such group.

• Numerical approaches, which utilize numerical processing based on a pair (inputs, outputs) serving as a reference model. A program is then used to minimize the discrepancy between the model and the reference.

1.4. Historical Foundations of Artificial Intelligence

The key milestones in the history of AI are as follows:

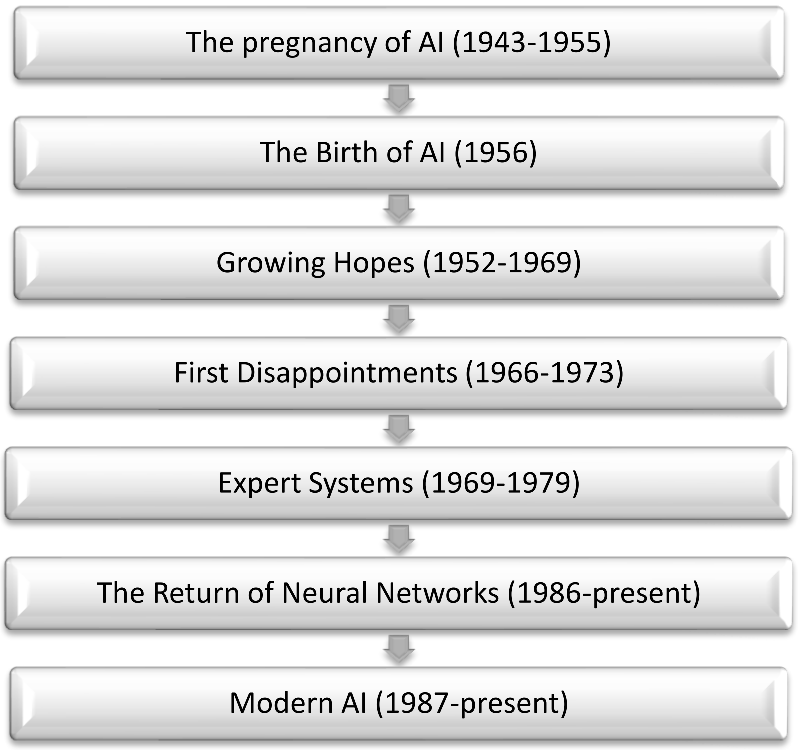

The pregnancy of AI (1943-1955)
The Birth of AI (1956)
Growing Hopes (1952-1969)
First Disappointments (1966-1973)
Expert Systems (1969-1979)
The Return of Neural Networks (1986-present)
Modern AI (1987-present)

Figure I. 2 Key milestones in artificial intelligence.

[1] .The pregnancy of AI (1943-1955)

- The work of Warren McCulloch and Walter Pitts, published in 1943, introduced one of the earliest mathematical models of artificial neurons. Their model described neural activity using logical and binary operations, laying the theoretical foundations for neural computation. Although the term artificial intelligence had not yet been coined, their contribution is widely regarded as one of the earliest milestones in the emergence of artificial intelligence.
- Subsequently, Donald Hebb proposed a learning rule describing how synaptic connections between neurons could be modified through experience. Known as Hebbian learning, this principle states that the connection strength between two neurons increases when they are activated simultaneously. This concept became fundamental in neural network learning algorithms and biological learning theories.
- Later, Marvin Minsky and Dean Edmonds built one of the first artificial neural network machines, known as the SNARC (Stochastic Neural Analog Reinforcement Calculator). This system demonstrated how networks of artificial neurons could be physically

implemented to perform learning tasks, further advancing the practical development of neural computation.

- In parallel, Alan Turing published his seminal paper "Computing Machinery and Intelligence" in 1950, in which he introduced the Turing Test as a criterion for machine intelligence. This test evaluates a machine's ability to exhibit intelligent behavior indistinguishable from that of a human, and it remains a foundational concept in artificial intelligence research.

[2] .The Birth of AI (1956)

- John McCarthy and Marvin Minsky, with the support of their senior colleagues Claude Shannon and Nathaniel Rochester, secured a grant of USD 7,500 from the Rockefeller Foundation to organize a summer research workshop at Dartmouth College in 1956. This workshop, held over a two-month period, focused on the ambitious goal of studying thinking machines and exploring how aspects of human intelligence could be simulated by computational systems.
- The conference brought together researchers from mathematics, engineering, and cognitive science, fostering interdisciplinary discussions that would shape the future of the field. Most importantly, it was during this event that the term "Artificial Intelligence" was formally adopted, marking the official birth of artificial intelligence as a distinct scientific discipline.
- The Dartmouth Conference is now widely recognized as a foundational moment in the history of artificial intelligence, as it established the conceptual framework and research agenda that would guide AI development for decades to come.

- **Growing Hopes (1952-1969)**

- A large number of pioneering artificial intelligence programs were developed during this period. Among the most notable was the Logic Theorist, created by Allen Newell and Herbert A. Simon, which is considered one of the first successful AI programs. It was designed to prove theorems from symbolic logic and demonstrated that machines could perform tasks previously thought to require human intelligence. Another significant system was the Geometry Theorem Prover, developed by Herbert Gelernter, which applied heuristic search techniques to solve geometric problems.

- The General Problem Solver (GPS), also developed by Newell and Simon, aimed to model human problem-solving behaviour. It was capable of solving simple puzzles by applying means–ends analysis, a reasoning strategy inspired by human cognitive processes. This work represented an important step towards understanding and formalizing human reasoning within computational systems.
- At the same time, several of Marvin Minsky's students investigated so-called "microworlds", which are simplified and highly constrained problem environments. These included analogy problems similar to those found in intelligence quotient (IQ) tests. Research on microworlds helped researchers explore fundamental cognitive mechanisms in a controlled setting before addressing more complex real-world problems.
- In parallel, John McCarthy published a highly influential paper addressing the challenge of building programs endowed with common-sense reasoning. His work highlighted the importance of representing everyday knowledge and reasoning about ordinary situations, a problem that remains central to artificial intelligence research.
- Finally, research on artificial neural networks continued alongside symbolic AI approaches, contributing to the development of learning models inspired by biological neural systems and laying the groundwork for later advances in connectionist AI.

[3] .First Disappointments (1966-1973)

- This period is often referred to as the first "AI winter", marked by significant setbacks and widespread disappointment. One of the most notable failures concerned machine translation. Despite five years of intensive research, automatic translation systems failed to meet expectations. As a result, in 1966, the United States government withdrew all funding for machine translation projects, following the conclusion that the technology was not delivering practical results.
- These difficulties were largely due to severe limitations in memory capacity and computational power, which constrained the complexity and scalability of early AI systems. The gap between ambitious research goals and available hardware resources led to growing skepticism about the feasibility of artificial intelligence.
- These shortcomings were formally criticized in the Lighthill Report published in 1973 in the United Kingdom. The report was highly critical of AI research, particularly in areas

such as robotics and general problem solving, and concluded that progress had been overstated. Its publication resulted in the termination of funding for the majority of artificial intelligence projects in Great Britain, significantly slowing the development of the field.

[4] .Expert Systems (1969-1979)

- The first expert system, known as DENDRAL, was developed in 1969. Designed to assist chemists in the identification of molecular structures, DENDRAL demonstrated that computers could effectively emulate the decision-making processes of human experts within a well-defined domain. This system marked a significant milestone in artificial intelligence by showing the practical value of knowledge-based reasoning.
- Following this success, several other expert systems were developed, among which MYCIN became one of the most influential. MYCIN was designed to perform the diagnosis of blood infections and to recommend appropriate antibiotic treatments. It relied on a rule-based inference mechanism and was notable for achieving performance comparable to that of medical specialists, despite operating under uncertainty.
- The development of systems such as DENDRAL and MYCIN contributed to a renewed interest in artificial intelligence during the 1970s and 1980s, highlighting the potential of expert systems in real-world applications and reinforcing the importance of knowledge representation and reasoning under uncertainty.

[5] .The Return of Neural Networks (1986-present)

- In the mid-1980s, four groups of researchers independently discovered the "backpropagation" learning rule (the idea had been proposed in 1969, but had received no attention in the scientific community).
- - Since then, machine learning has become one of the most active areas of AI (as seen, for example, in data mining).

[6] .Modern AI (1987-present)

- With the rapid development of artificial intelligence and the emergence of advanced technologies such as machine learning and deep learning, researchers generally agree on the distinction between three main types of artificial intelligence: **general, narrow (weak) and popular**

- **General Artificial Intelligence (GAI),** also referred to as strong or deep artificial intelligence, denotes an artificial system capable of performing any cognitive task that a human or an animal can accomplish. Such a system would exhibit general reasoning, learning, adaptation, and problem-solving abilities across a wide range of domains, rather than being limited to a specific task.
- At present, GAI remains largely theoretical and hypothetical. However, some scientists have begun to question whether recent large-scale language models, such as GPT-4, could represent an early or partial form of general artificial intelligence. While these models demonstrate impressive capabilities in language understanding and reasoning, there is ongoing debate regarding whether they truly possess general intelligence or merely advanced pattern recognition.
- Nevertheless, a large proportion of AI researchers believe that humanity now possesses the technological foundations necessary to develop GAI, particularly due to significant advances in artificial neural networks, computational power, and data availability. These developments have renewed interest in the long-term prospect of creating artificial systems with human-level cognitive abilities.

- **Narrow (Weak) Artificial Intelligence**
- The final category in the classification of artificial intelligence is Narrow Artificial Intelligence, also referred to as Weak AI. This type of artificial intelligence is designed to perform a single, well-defined task with a high level of efficiency and accuracy, often approaching or surpassing human performance, without requiring continuous human supervision.
- Narrow AI systems operate within a limited domain and do not possess general reasoning or understanding beyond their specific application. Despite this limitation, they represent the most widely used and developed form of artificial intelligence today. Such systems are deployed to automate and optimize processes across a broad range of sectors, including healthcare, finance, transportation, manufacturing, and information technology.
- Examples of narrow AI include speech recognition systems, image classification algorithms, recommendation engines, and autonomous control systems. Although these systems do not exhibit general intelligence, their practical impact has been significant, driving productivity gains and enabling new technological capabilities

- Applications of Artificial Intelligence in Medicine, Science, and Creative Domains
- In medicine, artificial intelligence is widely used for the diagnosis and prediction of diseases, enabling early detection and rapid intervention. AI-based systems assist clinicians by analyzing medical images, patient records, and biological data, thereby improving diagnostic accuracy and supporting personalized treatment strategies. Artificial intelligence is also extensively applied in pharmaceutical research, where it accelerates drug discovery processes, optimizes clinical trials, and contributes to the development of more effective therapies.
- In scientific research, AI plays a crucial role in analyzing large-scale datasets and facilitating discoveries in fields such as astrophysics, genomics, biology, and chemistry. By identifying complex patterns and correlations that are difficult for humans to detect, AI accelerates scientific progress and opens new avenues for research and innovation.
- **Popular I-A**: With the release of GPT-3.5 in November 2022, a powerful large language model (LLM), the potential of artificial intelligence expanded significantly. AI technologies are now increasingly used in creative domains, including text generation, image synthesis, and audiovisual content creation. Applications such as VALL·E, Midjourney, and GEN-2 illustrate how AI is transforming creative industries by enabling new forms of expression and production.
- Adobe Firefly is an artificial intelligence engine developed by Adobe that exclusively uses royalty-free and licensed images to generate new visual content. Adobe places strong emphasis on ethical considerations, positioning Firefly as one of the first ethically designed AI image-generation systems, aimed at respecting intellectual property rights and creative ownership.
- Bard AI is Google's intelligent conversational chatbot. Although it has not yet been made widely available in Europe, Google has expressed its commitment to deploying Bard within a framework of ethical artificial intelligence, with a focus on minimising misinformation and ensuring responsible use of AI technologies.
- Jasper is a software platform designed for written content creators. It enables users to produce articles and marketing content significantly faster by offering multiple writing tones, styles, and perspectives. Jasper illustrates how AI can enhance productivity in professional writing and digital communication.

- Spotify DJ represents the integration of artificial intelligence into the music streaming industry. Through this feature, Spotify is able to generate personalized playlists based entirely on user preferences, listening history, and interaction patterns. The recommendations dynamically evolve as the system continuously learns from user behavior.
- Gamma is a highly innovative artificial intelligence tool that enables the rapid creation of visual presentations and slide-based content. With only a few user inputs, Gamma can automatically generate structured and visually appealing presentations, demonstrating the growing role of AI in communication and knowledge dissemination.

1.5. References

[1] . Russell, S. & Norvig, P. Artificial Intelligence: A Modern Approach (4th ed.).Prentice Hall, 2020.

[2] . Mitchell, M.Artificial Intelligence: A Guide for Thinking Humans Farrar, Straus and Giroux, 2019.

[3] . Thomas Dunne Books, 2013. Discusses AI risks and the potential future impact of general intelligence. Wikipedia Kurzweil, R. The Age of Intelligent Machines.

[4] . MIT Press, 1990. A historical and philosophical exploration of AI's development and future.

[5] . Boden, M. Artificial Intelligence: A Very Short Introduction. Oxford University Press, 2018..

[6] . Springer chapter: A General Introduction to Artificial Intelligence, in Artificial Intelligence Technology, Springer, 2023

2.1. Introduction

One of the fundamental principles of pedagogy is to progress from simple concepts to more complex ones. In this context, this section presents an illustrative example commonly found in the MATLAB Fuzzy Logic Toolbox documentation. The example addresses the definition of a rule-based method for calculating the amount of a tip after a restaurant meal in a real-life scenario.

Depending on the individual, cultural background, and country, the method used to determine the tip may vary. However, the most critical issue lies in how the tip value should be defined and how one can ensure that the selected formulation is both appropriate and accurate.

In general, the perceived quality of service and the quality of the food are two key factors influencing the decision. Human reasoning in such situations is inherently subjective and varies according to personal experience and knowledge. Therefore, an essential task is to formalise this reasoning process in a structured and systematic manner. Adopting a progressive approach from the simplest to more complex representations highlight the intrinsic complexity of human reasoning and demonstrates the relevance of fuzzy logic as an effective modelling tool

2.2. Tip Calculation Formulas

Formula 1

In the USA, tipping is generally around 15% of the meal price; therefore, we will express it as follows:
Tip = 0.15 * PRICE of the meal

So for a $100 meal, the tip would be $15

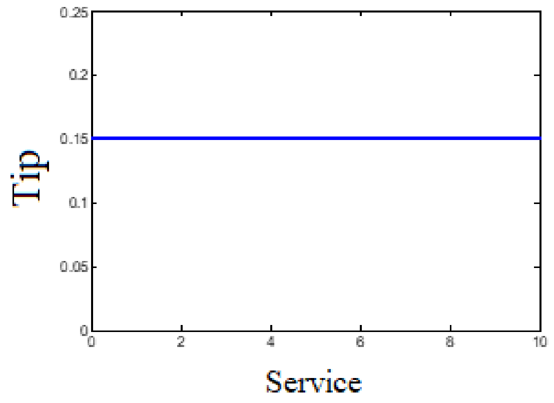


Figure II. 1 Fixed tip = 15% of price

Formula 2

• Tip and the quality of service, which is closer to reality, we must vary this value within a range of 5% to 25%.. On the other hand, the service is rated out of 10 and must therefore vary between 0 and 10.
• Consequently, the tip amount will vary linearly between these two values, increasing in an upward direction.

• The tangent to this line is equal to (25%-5%)/(10-0).

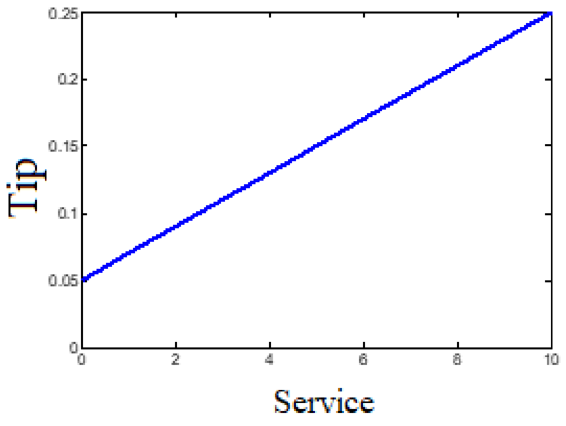


Figure II. 2 Tip and the quality of service

Formula 3

• Tip depending on the quality of service and the quality of the food, with the same weighting coefficient of **50%** for each parameter: • The quality of service and the quality of the meal are rated out of 10.

• Therefore, the tip amount will vary linearly for both parameters, and the total will be expressed similarly.If we want the quality of service to have more impact than the quality of food on a scale of **80%** versus **20%**.

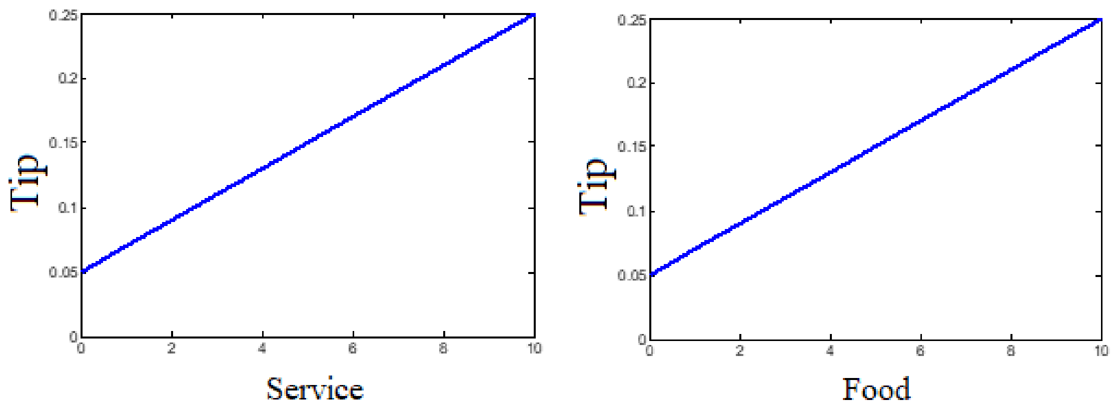


Figure II. 3 Tip and the quality of service and food

Formula 4

The reality we are working with is not always linear, resulting in areas where values may remain constant throughout a given interval.

• In addition to the scores for the two parameters, taken from 0 to 10 and weighted equally, we can divide the data into three zones:

1. From 0 to 3, where the tip will vary from 5% to 15%
2. From 3 to 7, where the tip will be constant at 15%
3. From 7 to 10, where the tip will vary from 15% to 25%

In the last case, (Formula 4) there are 9 possible ways to express this formula.

We choose one of the two parameters, for example, the service, and then for each interval, we check the corresponding second case

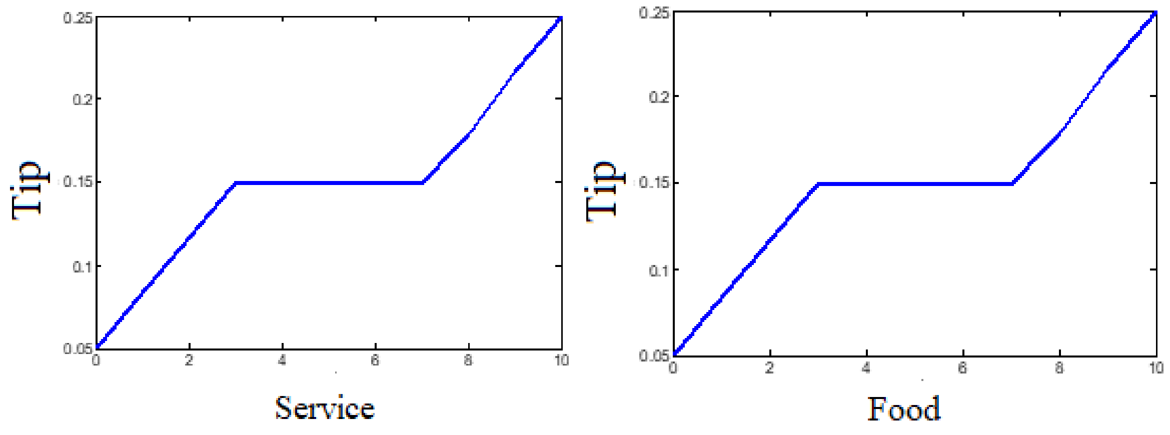


Figure II. 4 Tip and the quality of service and food

In this example, we started with the simplest formula, one without increasingly complex mathematical relationships, while trying to get as close as possible to reality.

The last formula, in particular, highlights the complexity of the calculation algorithm, hence the advantage of using a fuzzy logic approach

2.3. Fuzzy Approach

We consider our system to have a two-input block: the quality of service and the quality of food. The output is the amount of the tip to be given. Using linguistic variables to express the evaluation of the two inputs, we obtain the following statements :

Input 1 **Quality of service**	Input 2 **Quality of Food**	Output **Tip**
• Bad • Good • Excellent	• Rancid • Delicious	• Cheap • Average • Generous

We can therefore propose the following rules:

Input 1	**If**	service is	poor	**Then**	Tip	is	Cheap
	If	service is	good	**Then**	Tip	is	Average
	If	service is	excellent	**Then**	Tip	is	Generous
Input	**If**	Food is	rancid	**Then**	Tip	is	Cheap
	If	Food is	Delicious	**Then**	Tip	is	Generous

Our system can be represented by the following diagrams:

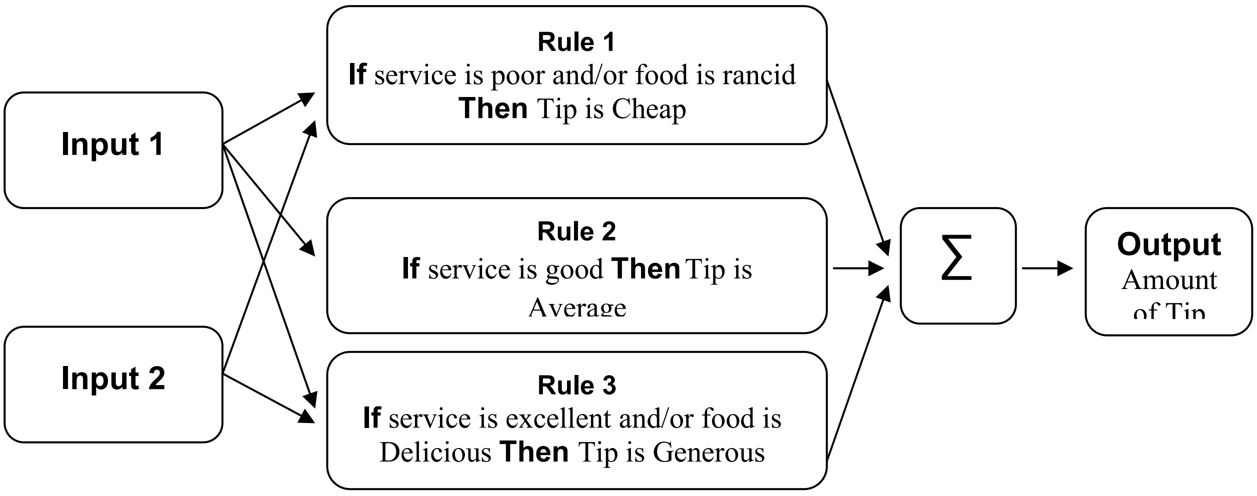

Input 1
Input 2
Rule 1
If service is poor and/or food is rancid
Then Tip is Cheap
Rule 2
If service is good Then Tip is
Average
Rule 3
If service is excellent and/or food is
Delicious Then Tip is Generous
∑
Output
Amount
of Tip

Figure II. 5 Propose rules for Tip and the quality of service and food

Many questions arise in this case:

- *How can the amount of this tip be numerically evaluated?*
  How can these rules be combined?
- *Can the proposed linguistic variables be easily quantified?*

To answer all questions in a way that closely resembles human reasoning, we will discuss the ***theory of fuzzy logic****.*

2.4. References

[1] . MathWorks – Fuzzy Logic Toolbox Documentation.

https://www.mathworks.com/help/fuzzy/ (consulté 2025)

[2] . Ross, Timothy J.Fuzzy Logic with Engineering Applications, 4th Edition, Wiley, 2016 (et éditions ultérieures).

[3] . Zadeh, L. A., et al. (Éditeurs)Fuzzy Logic: Theory and Applications, Academic Press, 1975

[4] . Mendel, J. M.Uncertain Rule-Based Fuzzy Logic Systems: Introduction and New Directions, Prentice Hall, 2001.

3.1. Introduction

The term "fuzzy set" first appeared in 1965 when Professor Lotfi A. Zadeh of the University of California, Berkeley, published an article entitled "Fuzzy Sets." Since then, he has made numerous major theoretical advances in the field and has been quickly joined by many researchers developing theoretical work. Fuzzy logic, based on fuzzy set theory, is establishing itself as an operational technique in the field of control and command. It is one of the artificial intelligence techniques based on the reasoning approach that has become established in several fields and has attracted the interest of many researchers.

3.2. History

Fuzzy logic bridges the gap between numerical and linguistic modeling, enabling remarkable industrial advancements through simple algorithms that translate symbolic knowledge into numerical data and vice versa. Fuzzy set theory has introduced an innovative approach to handling uncertainty. It has also influenced automatic classification techniques and contributed to the renewal of existing decision-support methods. Before the 1940s, the early approaches of American researchers to the concept of uncertainty laid the foundations of fuzzy logic. Zadeh contributed to the modeling of phenomena in fuzzy form, finally resolving the limitations due to uncertainties in classical models with differential equations. In 1974, M. Mamdani experimented with Zadeh's nonce theory on a steam boiler, a piece of equipment known for its complexity, thus introducing control into the regulation of an industrial process [Fu 03]. Several applications then emerged in Europe, for sometimes very complex systems, such as the regulation of kilns, cement plants, etc. In 1985, thanks to M. Sugeno, a Japanese researcher, fuzzy logic was introduced into industrial applications in Japan.

Table 3.1 presents the major applications based on fuzzy logic:

Table 3. 1 Applications of fuzzy logic

Year	Author	Application
1972	Zadeh.	• Linguistic approach.
1974	Mamdani & assilian.	• Control of a steam engine.
1980	Fukami et al.	• Fuzzy conditional inference.
1983	Sugeno & Takagi	• Derivation of fuzzy control rules.
1985	Togar & Watanabe.	• Fuzzy processor.
1988	Dubois & Prade	• Approximate reasoning.
1991	Barrat et al.	• Fuzzy temperature control of an oven.

3.3. Fuzzy Set Theory

Fuzzy set theory [Zad 65] is a mathematical framework primarily aimed at modeling the vague and uncertain concepts inherent in natural language. The notion of strict membership proves inadequate when dealing with uncertain or imprecise data, particularly when such data are difficult to express verbally. Classical set theory considers collections of elements grouped into well-defined sets, where the membership of an element in a set is unambiguous. In contrast, real-world situations often involve ambiguity, especially in everyday language. A fuzzy set is defined over a universe of discourse X through a membership function $\mu(x)(A)$.

μ(x)(A), which typically takes values between 0 and 1. This function quantifies the degree to which each element x in X. X belongs to the set, allowing a gradual and nuanced representation of membership rather than a strict binary classification.

In set theory, "either an element belongs to a set or it does not; the two cases are complementary." However, fuzzy logic theory takes into account the case of gradual membership. This degree of membership, denoted μ(x), is normalized between "0" and "1."

In the classical set theory, the set of elements that satisfy a certain condition, and to be represented as in set theory. Taking the case of a net set A:

So we can note: A = {x/x satisfies certain conditions}.

Relative to the scale we have taken, the only value that each element of this set can take by introducing the membership function is either "0" or "1".

$$\mu_A(x) = \begin{cases} 1 & if \;\; x \in A \\ 0 & if \;\; x \notin A \end{cases}. \qquad (3.1)$$

3.3.1. Fuzzy set

A fuzzy set B is characterized by a membership function and by the fact that µB(x) takes its values across the entire interval [0, 1], where x is called the universe of discourse.

There are two cases for this type of set:

1. Continuous: B is not in the following form:There are two cases for this type of set:

1. Continuous: B is not in the following form::

$$B = \int U \, \mu_A(x)/x \qquad (3.2)$$

2. Discrete: B is denoted in the following form:

$$B = \sum U \mu_A(x)/x \qquad (3.3)$$

The crossover point: is the point for which the degree of belonging is situated at the average value of the scale taken between "0" and "1" equal to 0.5.

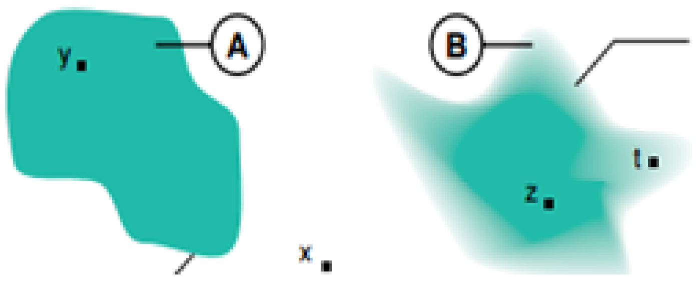


Figure III. 1 Classic set and fuzzy set

3.3.2. Degree of membership and membership functions

The diagram in Figure 3.10 shows the degree to which an element x belongs to a fuzzy set A. Depending on the type of membership function, different types of fuzzy sets are obtained. These include linear functions such as triangular and trapezoidal functions, and nonlinear functions such as Gaussian functions and the singleton function, where only one point in the sample space takes the value 1.

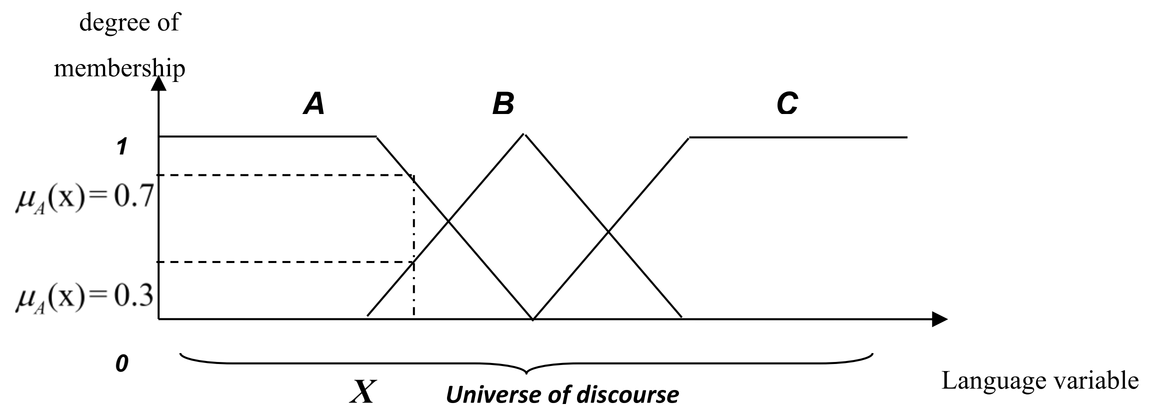


Figure III. 2 Elements of fuzzy logic.

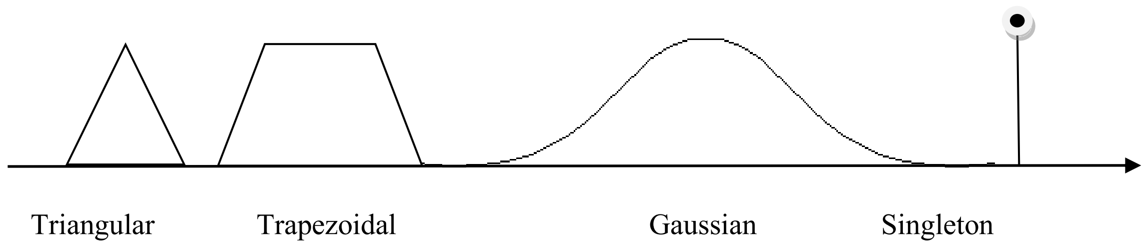


Figure III. 3 Forms of membership functions.

3.3.3. Operations

The logical operators "and," "or," and "not" can be defined using fuzzy sets, analogously to classical set theory. Recall that these are defined respectively by intersection, union, and complement.

If A and B are two fuzzy subsets and their membership function, we define::

- • The complement of A, by the membership function:

$$\mu(\bar{A}) = 1-\mu(A) \tag{3.4}$$

- The subset A and B, A∩B, by the membership function:

$$\mu(A\cap B) = \min(\mu(A), \mu(B)) \tag{3.5}$$

- • The subset A or B, A ∪ B, by the membership function:

$$\mu(A\cup B) = \max(\mu(A), \mu(B)) \tag{3.6}$$

***t-norm also called fuzzy intersection*:**

It's a binary operation: T:[$0,1$]x[$0,1$]→ [$0,1$]

It respects the following requirements:

- Commutativity: x ∈ y = y ∈ x
- Associativity: x ∈ (y ∈ z) = (x ∈ y) ∈ z
- Monotonicity: if x ≤ y and w ≤ z then x ∈ w ≤ y ∈ z.
- Boundary conditions: ∑_{0=0}^{t0} = 0 and x ∈ 1 = x

***t-norm or s-norm, also called fuzzy union*:**

It's a binary operation: S:[$0,1$]x[$0,1$]→ [$0,1$]

It respects the following requirements:

- Commutativity: x S y = y S x
- Associativity: x S (y S z) = (x S y) S z
- Monotonicity: if x ≤ y w ≤ z then x S w ≤ y S z.
- Boundary conditions: $0 S 0 = 0$ and $x S 1 = x$

3.3.4. Linguistic variable

- A linguistic variable is a variable whose values are words or phrases expressed in an artificial or natural language. It is defined by the name of the linguistic variable, X is the physical domain associated with the variable, V is also called the universe of discourse, and the set of fuzzy characteristics of the variable.
- • For example, we define the notions of low, medium, and high temperature. We can define the degree to which the variable "temperature" belongs to the set "low" as the "degree of truth" of the proposition "the temperature is low."
- • In Boolean logic, the degree of membership (μ) can only take two values (0 or 1). The temperature can be:

- low: $\mu_{\text{low}}(T)=1, \mu_{\text{medium}}(T)=0, \mu_{\text{high}}(T)=0$
- Medium: $\mu_{\text{low}}(T)=0, \mu_{\text{medium}}(T)=1, \mu_{\text{high}}(T)=0$
- High: $\mu_{\text{low}}(T)=0, \mu_{\text{medium}}(T)=0, \mu_{\text{high}}(T)=1$

It cannot take two attributes at once.

In fuzzy logic, the degree of membership becomes a function that can take a real value between 0 and 1 inclusive.$\mu_{\text{medium}}(T)$ For example, it allows us to quantify whether the temperature can be considered average.

In this case, the temperature can be considered both as low with a degree of membership of 0.2 and as medium with a degree of membership of 0.8 (Figure III.4).

μ: degré d'appartenance $\mu_{\text{Low}}(T)$=0.2, $\mu_{\text{Medium}}(T)$=0.8, $\mu_{\text{High}}(T)$=0

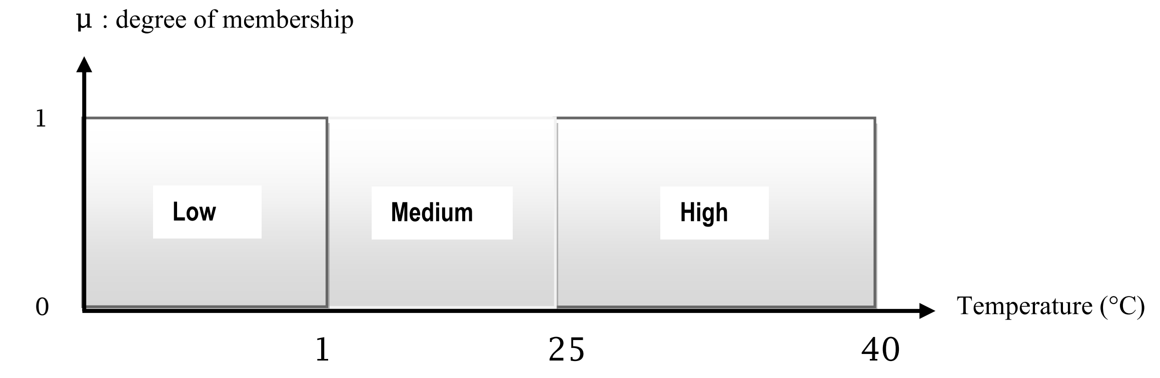


Figure III. 4 Example of sets considered in Boolean logic.

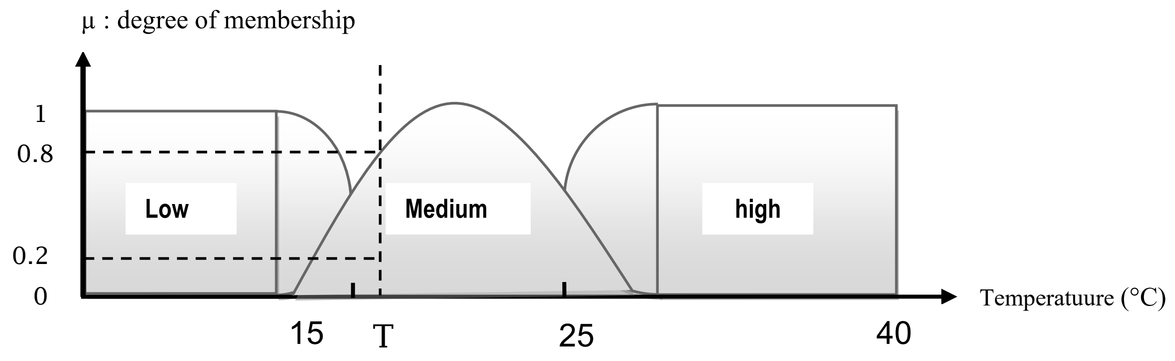


Figure III. 5 Example of sets considered in fuzzy logic.

In this example, the fuzzy variable is temperature, and the universe of discourse is the set of real numbers in the interval [0, 40]. These variables are assigned three fuzzy subsets: low, medium, and high. Each is characterized by its degree of membership function

n:$\mu_{\text{low}}(T)$, $\mu_{\text{meium}}(T)$ et $\mu_{\text{high}}(T)$.

We can define the degree of membership function μ_"average" over the entire universe of discourse::

$$\mu_{\text{Medium}}(x) = \begin{cases} \frac{1}{1+e^{(15-x)}} & ;x[0,20] \\ 1-\frac{1}{1+e^{(25-x)}} & ;x[20,40] \end{cases} \qquad (3.7)$$

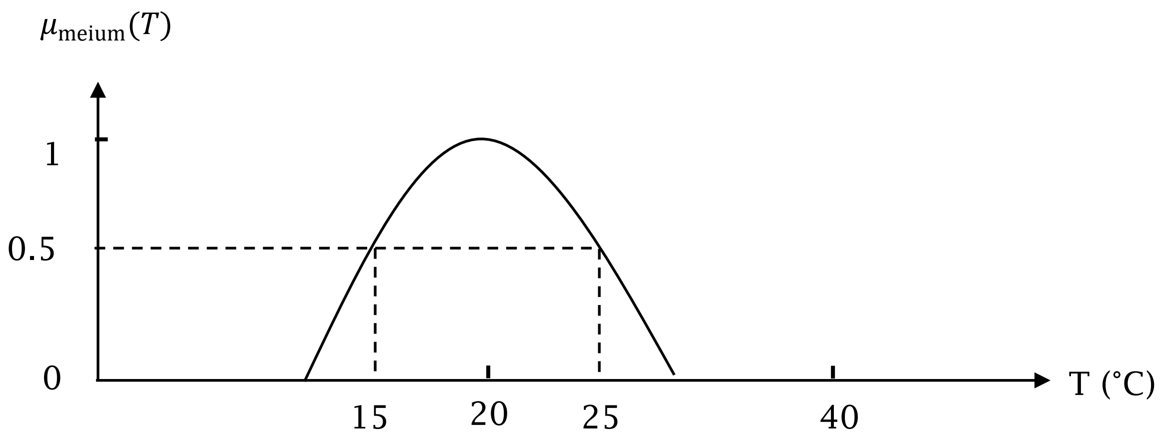


Figure III. 6 Case of the fuzzy set "average" of the temperature variable.

3.4. Definition of a Fuzzy System

A fuzzy system (FS) is a nonlinear relationship that allows taking numerical data (inputs) and passing it through a fuzzy domain, then obtaining a scalar output (a clear output). The general structure of this process is shown in the following figure:

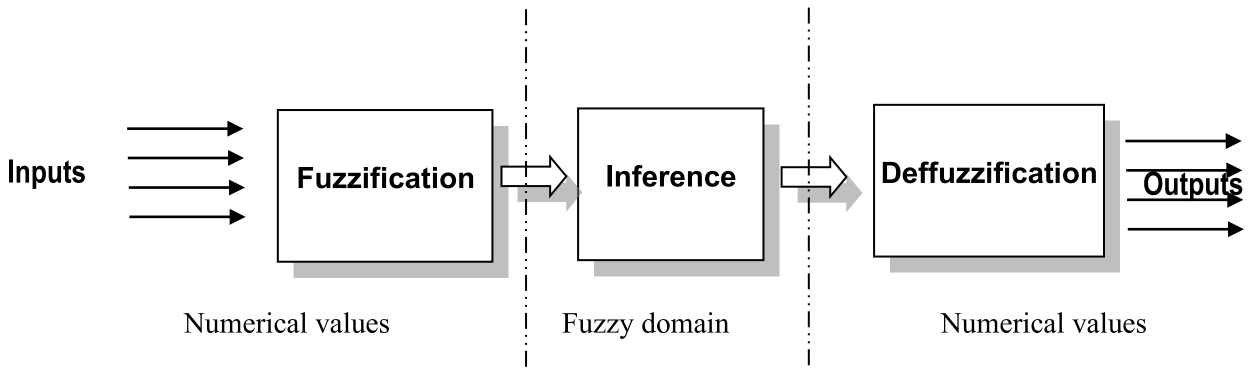


Figure III. 7 Fuzzy processing (overall scheme).

3.3.1 Constitution

A fuzzy system consists of four essential parts:

• *The knowledge base, comprising a database and a rule base,*

• *The inference system,*

• *The fuzzification interface.*

• *The defuzzification interface.*

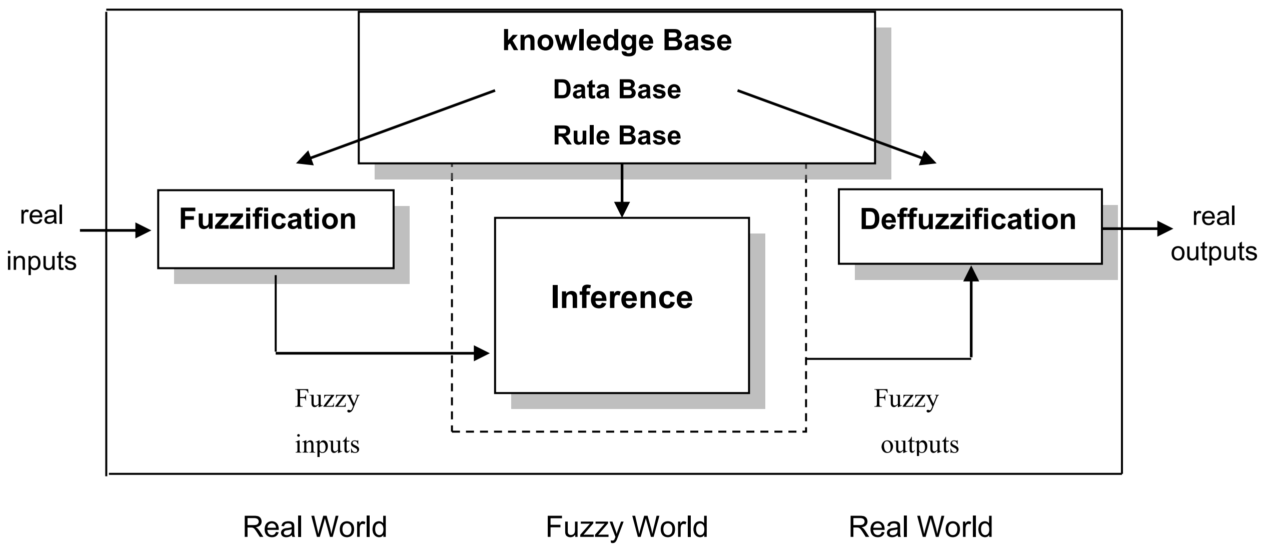


Figure III. 8 General diagram of a fuzzy system.

- **Inference**: This allows us to calculate the fuzzy set associated with the command and is done through fuzzy inference operations and rule aggregation. Fuzzy inference relies on the use of a fuzzy implication operator for each rule to be analyzed. This operator quantifies the strength of the link between the premise and the conclusion of the rule.

- **Fuzzification**: consists of transforming real quantities into linguistic variables that are associated with a database of sets characterizing them. These terms will be used to describe the rules. Comparative studies have shown that, with the different forms, the results are similar in closed loops. The number is odd and is distributed around zero. The numbers frequently used are 3, 5, or 7. The number depends on the desired precision

- **The rule base***:* This is the collection of rules that allows us to link fuzzy input and output variables. These rules take the form "if then" and can be written textually, referring to inputs and outputs. They are provided by experts in a direct numerical manner or through linguistic terms or variables via membership functions, as we will see. Depending on the desired behavior, a set of rules can be applied. There are several ways to express inferences, namely through linguistic description, inference matrix, or inference table. Two inference approaches are commonly used: Mamdani implication and Larsen implication. In the MIN MAX algorithm, the MIN operator is used for combining premises and inferring rules, and the MAX operator for aggregating rules. In the PROD MAX algorithm, the PROD operator is used for combining premises and implying rules, and the MAX operator is used for aggregating rules; that is, the product of the membership degrees obtained with each rule is used to define the membership degree for the output. This is for a fuzzy controller of the SUGENO type. The rules premises are also linguistic, but the conclusions are developed in polynomial form.

- ***The defuzzifier***: The defuzzification step consists of transforming the fuzzy set resulting from the aggregation of rules, as this processing of inference rules yields a fuzzy value. In the literature, several strategies exist for performing this operation, such as the mean of maxima, the center of areas, and the center of maxima. The centroid defuzzification method is the most widely used in fuzzy control because it intuitively provides the most representative value. It consists of calculating the centroid of the surface formed by the resulting membership function.

3.5. Application: Fuzzy Logic toolbox

MATLAB's fuzzy logic toolbox includes an editor for creating fuzzy inference systems (FIS). This toolbox generates "∗.fis" files, which represent fuzzy inference systems. The displayed format is a structure that can be saved and viewed in memory within the workspace.

These files can be created using command-line tools or, more easily, scripts. Additionally, Simulink provides Fuzzy Controller objects, allowing these files to be accessed from memory. The diagram in Figure 3.10 illustrates how fuzzy systems are integrated into the MATLAB environment.

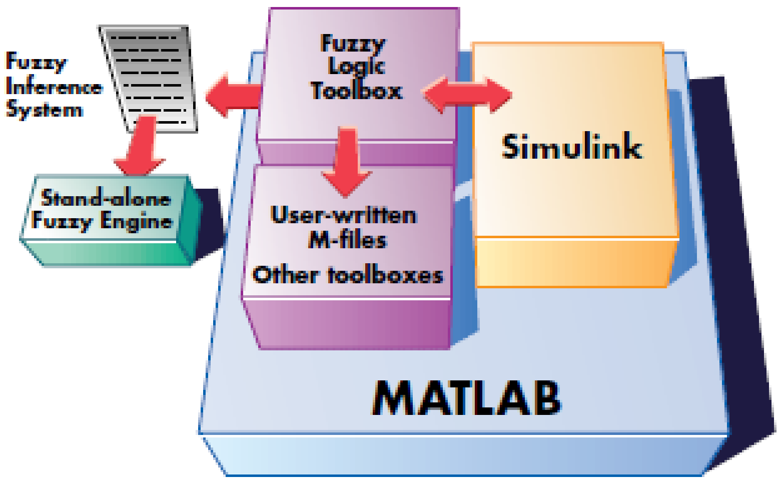


Figure III. 9 Integration of Fuzzy Inference Systems in the MATLAB Environment

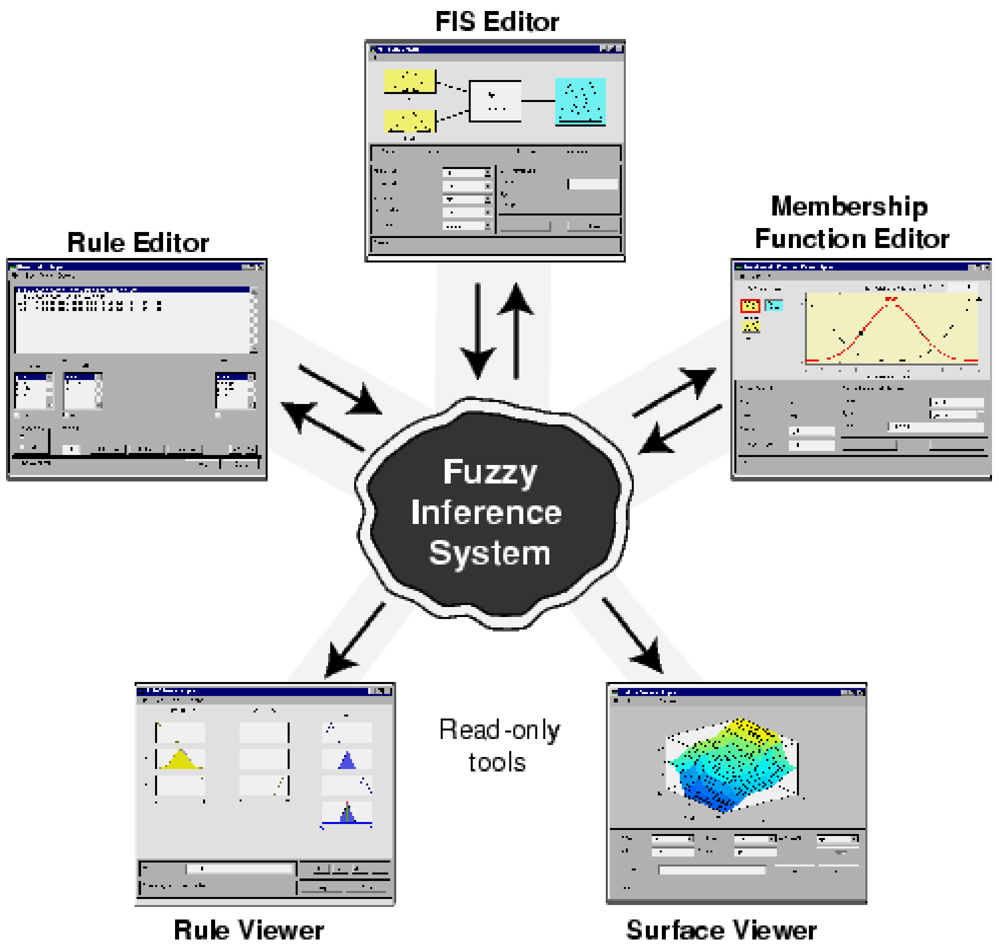


Figure III. 10 Fuzzy logic toolbox in the MATLAB environment

This toolkit has three editors:

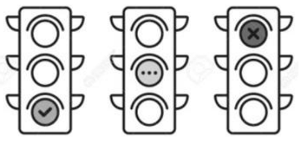
- **FIS Editor**: The fuzzy inference system editor, which is the main editor for defining the number of inputs and outputs, their names, and their type: Mamdani or Sugeno.
- **Membership Function Editor**: The membership function editor, which allows you to insert, delete, and configure membership functions. This is also where you can define the discourse universe.
- **Rule Editor**: The rule and membership function editor, which allows you to enter all the rules linking the inputs and outputs of the FIS. You can add, delete, and modify a rule, change the connector, and/or modify the weight.
- **Rule viewer** and surface viewer: Graphical interfaces that allow you to visualize inferences directly on the rule base, as well as control surfaces. In the Rule viewer window, you can verify the system's operation by applying net inputs (numerical values to observe the system's operation and obtain the net output).

3.6. Example of fuzzy Application Mamdani type

Homework

The aim of this example is to verify the application of fuzzy logic theory and compare it to human reasoning based on the rules of form: If condition then conclusion

If the light is red	If my speed is high	If the fire is nearby	So I …………………..
If the light is red	If my speed is low	If the fire is far away	So I …………………..
If the light is orange	If my speed is Medium	If the fire is far away	So I …………………
If the light is green	If my speed is low	If the fire is nearby	So I …………………….



When stopped at a red light, the following actions must be taken:

Brake hard,

brake gently,

accelerate,

maintain speed

The table shows the action that corresponds to each situation

1. Define the inputs and outputs of this system.
2. Define the discourse universes and membership functions of each variable.
3. The standards used for these traffic lights:
   a. Duration of the 3 lights: 60 seconds;
   b. Maximum speed: 50 km/h;
   c. Maximum distance: 20 m;
   d. Acceleration speeds: ±50 km/h.
4. Write a MATLAB program (script) that creates this system,

If the light is red	If my speed is high	If the fire is nearby	So I Brake hard
If the light is red	If my speed is low	If the fire is far away	So I maintain speed.
If the light is orange	If my speed is Medium	If the fire is far away	So I brake gently.
If the light is green	If my speed is low	If the fire is nearby	So I accelerate.

Inputs: Traffic Light, Speed, Position

Output: Acceleration

Variable names with the same meaning are acceptable.

For the output, some propose two options: accelerate or brake; this is correct.

Speed is the derivative of distance. Acceleration is the instantaneous rate of change of speed. Positive acceleration indicates increasing speed, zero acceleration indicates constant speed, and negative acceleration indicates decreasing speed, which corresponds to braking.

The discourse universes are defined according to the standards used in urban traffic lights, namely

Traffic lights 60 s [0 60]

Speed less than 50 km/h [0 50]

Distance less than 20 m [0 20]

Acceleration -50 km/h to +50 km/h [-50 50]

For the membership functions, we choose linear trapezoidal and triangular shapes distributed symmetrically, which best correspond to the system's behavior in reality.

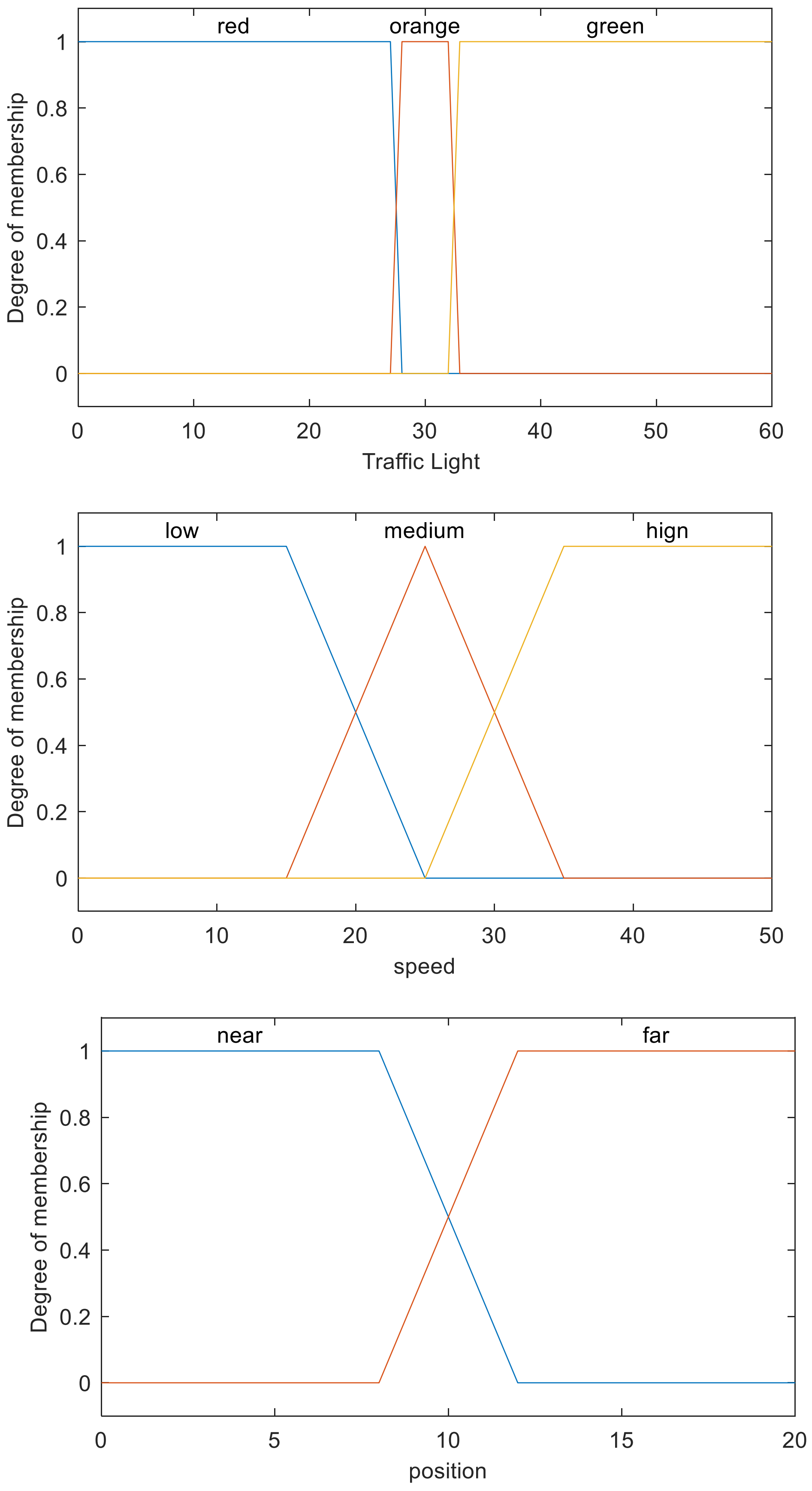


Figure III. 11 Fuzzy traffic lights inputs

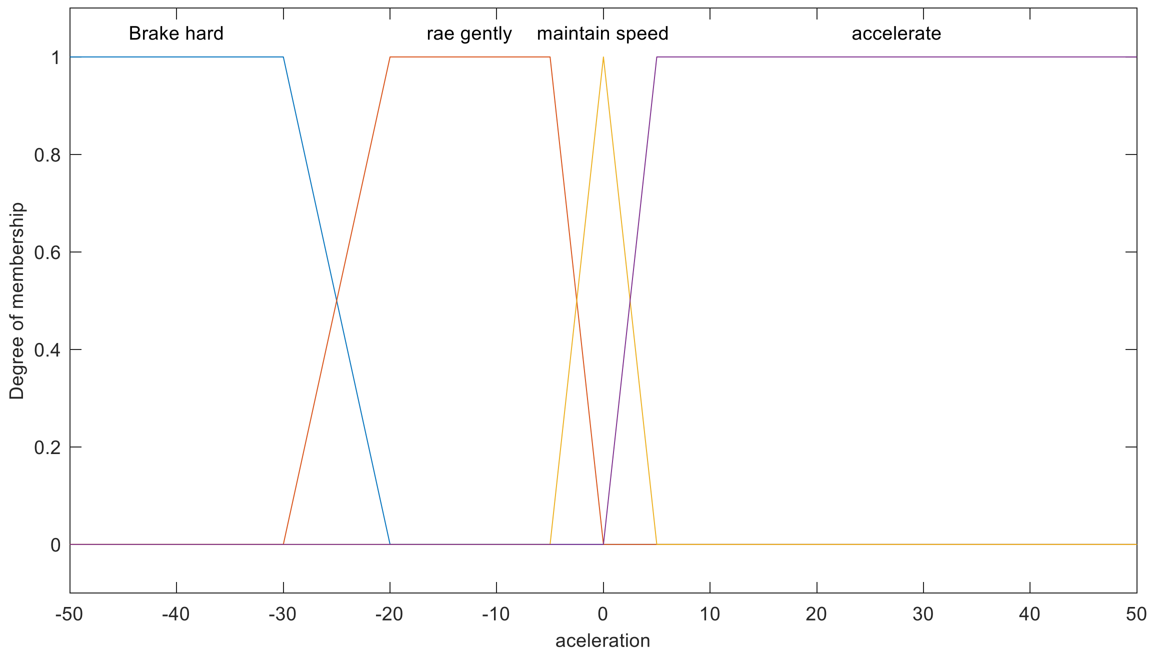


Figure III. 12 Fuzzy traffic lights output

```
1.  a=newfis('Traffic ')
2.  a=addvar(a,'input','Traffic Light ',[0 60]);
3.  a=addmf(a,'input',1,'red','trapmf',[0 0 27 28]);
4.  a=addmf(a,'input',1,'orange','trapmf',[27 28 32 33]);
5.  a=addmf(a,'input',1,'green','trapmf',[32 33 60 60]);
6.  a=addvar(a,'input','speed',[0 50]);
7.  a=addmf(a,'input',2,'low','trapmf',[0 0 15 25]);
8.  a=addmf(a,'input',2,'medium','trimf',[15 25 35]);
9.  a=addmf(a,'input',2,'hign','trapmf',[25 35 60 60]);
10. a=addvar(a,'input','position',[0 20]);
11. a=addmf(a,'input',3,'near','trapmf',[0 0 8 12]);
12. a=addmf(a,'input',3,'far','trapmf',[8 12 20 20]);
13. a=addvar(a,'output','aceleration',[-50 50]);
14. a=addmf(a,'output',1,' Brake hard ','trapmf',[-50 -50 -30 -20]);
15. a=addmf(a,'output',1,'rae gently','trapmf',[-30 -20 -5 0]);
16. a=addmf(a,'output',1,' maintain speed ','trimf',[-5 0 5 ]);
17. a=addmf(a,'output',1,'accelerate','trapmf',[0 5 50 50]);
```

3.7. Example of fuzzy Application Takagi-Sugeno type

/We want to control the production quality of Samsung mobile phones according to their weight P and width L using a fuzzy Takagi-Sugeno type system.

Homework

	Decision	Weight \ Width			
	Reparation = 0		Sale	Sale	rejection
	Sale =+1		Sale	Sale	rejection
	rejection = -1		Reparation	Reparation	rejection

1. Complete the table.

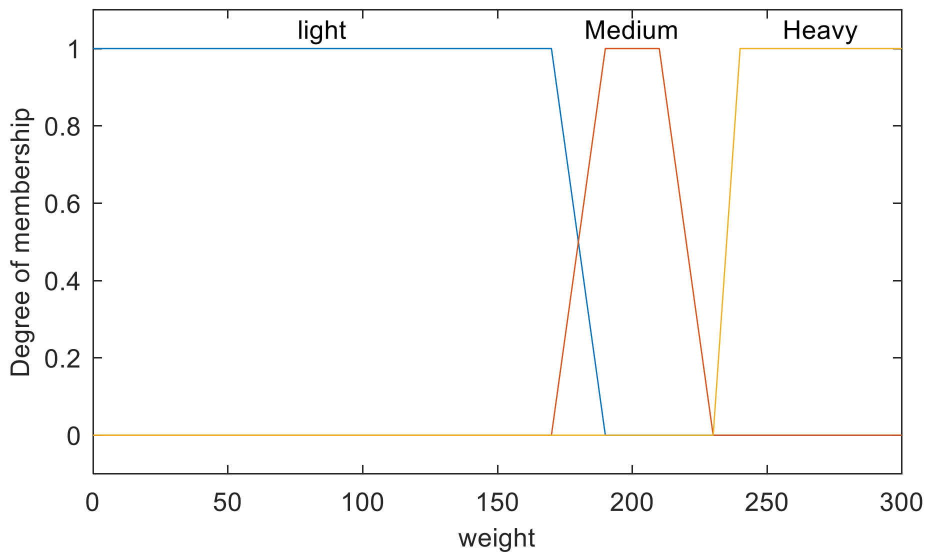
2. Propose a universe of discourse, linguistic variables, and membership functions for the system inputs.

3. Write a script (MATLAB file) that generates this fuzzy controller.

Weight:	100g, 150g, 250g
Width:	2cm, 4cm, 6cm
For weight: [0, 300]	Light, Medium, Heavy
For width: [0, 10]	Small, Medium, Large

Discourse universes are defined according to the values indicated in the table.

For membership functions, the linear, trapezoidal, and triangular shapes, distributed symmetrically and corresponding best to the system in reality, are chosen.



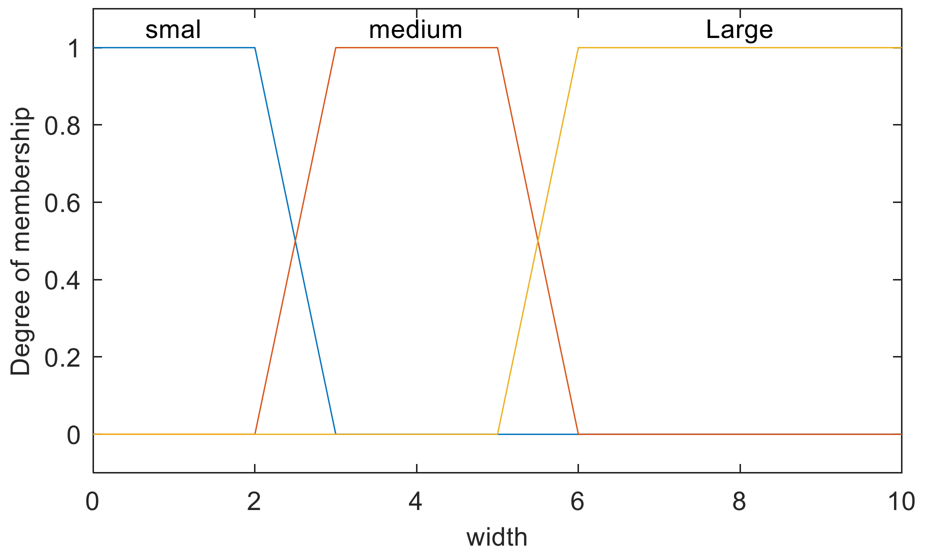


Figure III. 13 Fuzzy mobile decision inputs

```
a=newfis('mobile','sugeno')
a=addvar(a,'input',weight,[0 300]);
a=addmf(a,'input',1,'light','trapmf',[0 0 170 190]);
a=addmf(a,'input',1,'Medium','trapmf',[170 190 210 230]);
a=addmf(a,'input',1,'Heavy','trapmf',[230 240 300 300]);
a=addvar(a,'input','widthr',[0 10]);
a=addmf(a,'input',2,'smal','trapmf',[0 0 2 3]);
a=addmf(a,'input',2,'medium','trapmf',[2 3 5 6]);
a=addmf(a,'input',2,'Large','trapmf',[5 6 10 10]);
a=addvar(a,'output','decision',[-50 50]);
a=addmf(a,'output',1,'reparation','constant',[0]);
a=addmf(a,'output',1,'sale','constant',[1]);
a=addmf(a,'output',1,'rejectiont','constant',[-1]);
ruleList=[ ...
1 1 2 1 1
1 2 2 1 1
1 3 1 1 1
2 1 2 1 1
2 2 2 1 1
2 3 1 1 1
3 2 1 1 1
3 3 3 1 1]
a=addrule(a,ruleList);;
```

3.8. References

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

[2] . Zadeh, L. A.Fuzzy Sets as a Basis for a Theory of Possibility, Fuzzy Sets and

[3] . Zadeh, L. A., Kacprzyk, J. (Eds.) Computing with Words in Information/Intelligent Systems: Foundations, Methodologies, and Applications, Springer, 1999.

[4] . Klir, G. J. & Yuan, B.Fuzzy Sets and Fuzzy Logic: Theory and Applications, Prentice Hall, 1995.

[5] . Dubois, D. & Prade, H.Fuzzy Sets and Systems: Theory and Applications, Academic Press, 1980.

[6] . MathWorks – Fuzzy Logic Toolbox Documentation.

https://www.mathworks.com/help/fuzzy/ (consulté 2025)

[7] . Ross, Timothy J.Fuzzy Logic with Engineering Applications, 4th Edition, Wiley, 2016 (et éditions ultérieures

4.1. Introduction

In the field of research, new techniques continually emerge, either to complement existing methods or to completely replace them. Fuzzy logic initially appeared in a relatively simple form. Today, many researchers are turning to this powerful approach because it enables the integration of various types of qualitative knowledge. This provides significant advantages, particularly due to the fact that fuzzy logic is:

Close to human language, allowing intuitive knowledge representation,

Easy to understand thanks to its mathematical simplicity,

Flexible and adaptable to different problem domains,

Capable of handling imprecise, vague, or uncertain data,

It also enables the formalization and simulation of human expertise,

For control engineers, fuzzy logic is widely known under the name fuzzy control, where it is used to model human reasoning and decision-making processes in control systems.

4.2. Fuzzy modeling

The mathematical model that describes the interaction between inputs, outputs, and disturbances acting on a process enables the synthesis of appropriate control structures. Handling information that is both imprecise and expressed in natural language becomes more straightforward. Such manipulation of linguistic data is efficiently achieved using fuzzy set theory, which facilitates the formulation of rules to describe the behavior of the process.

The simplest approach consists of extracting knowledge from a human operator, where experts directly express control rules based on their experience. This knowledge is commonly represented in the form of rules such as:

"If <Premise (antecedent)> then <Conclusion (consequence)>."

One of the most significant contributions of fuzzy logic lies in its ability to represent imprecision when both premises and conclusions are expressed using natural language terms. In fuzzy logic, a simple rule can be formally expressed in this manner.

Two main classes of fuzzy models can be distinguished:

Fuzzy models with functional conclusions, known as Takagi–Sugeno models, Fuzzy models with symbolic (linguistic) conclusions, known as linguistic models or Mamdani models.

Both model types are based on a collection of if–then rules, in which the premises are expressed symbolically, allowing an intuitive and flexible representation of complex system behavior.

4.3. Fuzzy Controller

- A fuzzy controller is essentially a fuzzy system designed to control a process. Its general structure is typically represented by the block diagram shown in Figure (4.1).
- A fuzzy controller generally operates through the following steps:
- Selection of the fuzzification strategy,
- Construction of the rule base,
- Selection of the inference method,
- Selection of the defuzzification strategy.

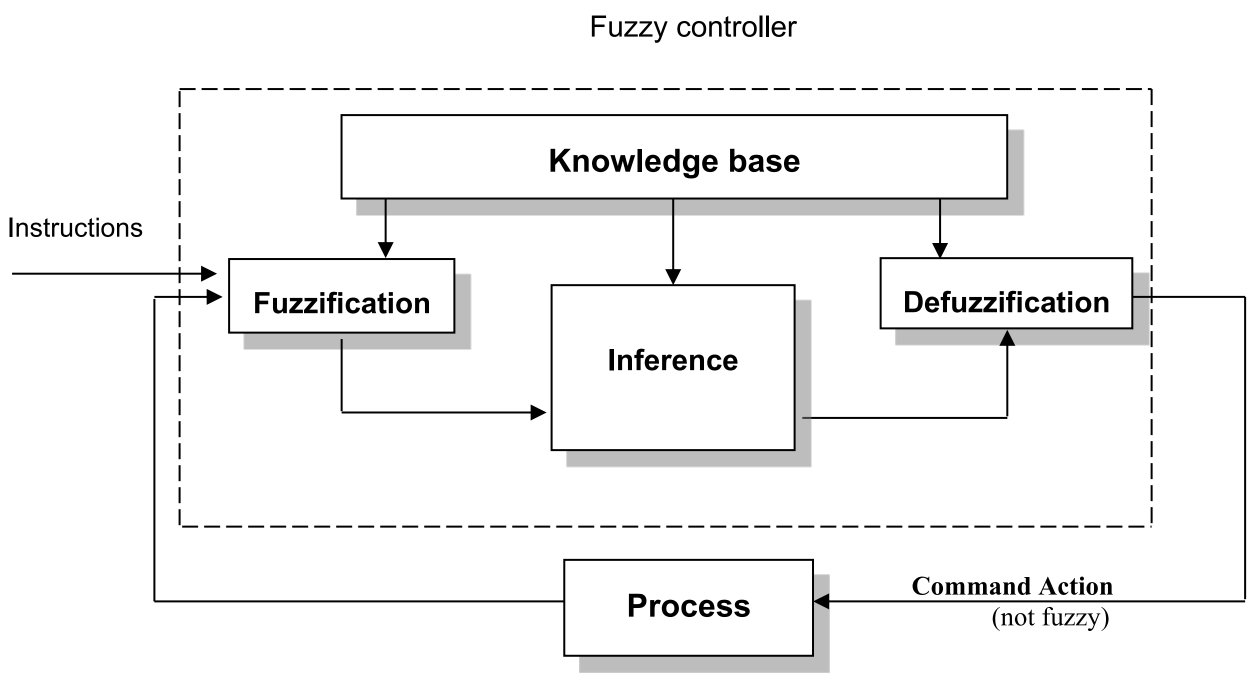


Figure IV. 1 General diagram of a fuzzy controller.

4.4. Conventional fuzzy controller example

We will present a speed controller within a vector control of the asynchronous machine. The basic diagram of the controller is represented by figure (4.2), it is based on the structure of a classic PI controller

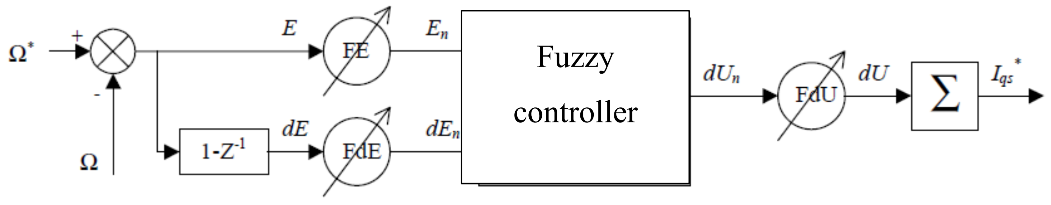


Figure IV. 2 Block diagram of a fuzzy PI speed controller

As in classical control tuning, the error between the reference speed and the measured speed is used as the first input. The variation of the error is then introduced as a second input in order to determine the direction of change.

Normalization gains are applied to scale the input and output variables, allowing the fuzzy controller to operate correctly within its effective range.

The fuzzy sets of the input variables (En, dEn) and the output variable (dUn) are defined by triangular, piecewise-linear membership functions. These functions typically consist of 7, 5, or 3 fuzzy sets (Figure 4.3). The different fuzzy sets are characterized by standard linguistic labels.:

- Negative Large (NL)
- Negative Medium (NM)
- Negative Small (NS)
- Approximately Zero (AZ)
- Positive Small (PS)
- Positive Medium (PM)Positive Large (PL)

In the same manner as in classical control tuning, the error between the reference speed and the measured speed is used as the first input. The variation of this error is then added in order to determine the direction of change.

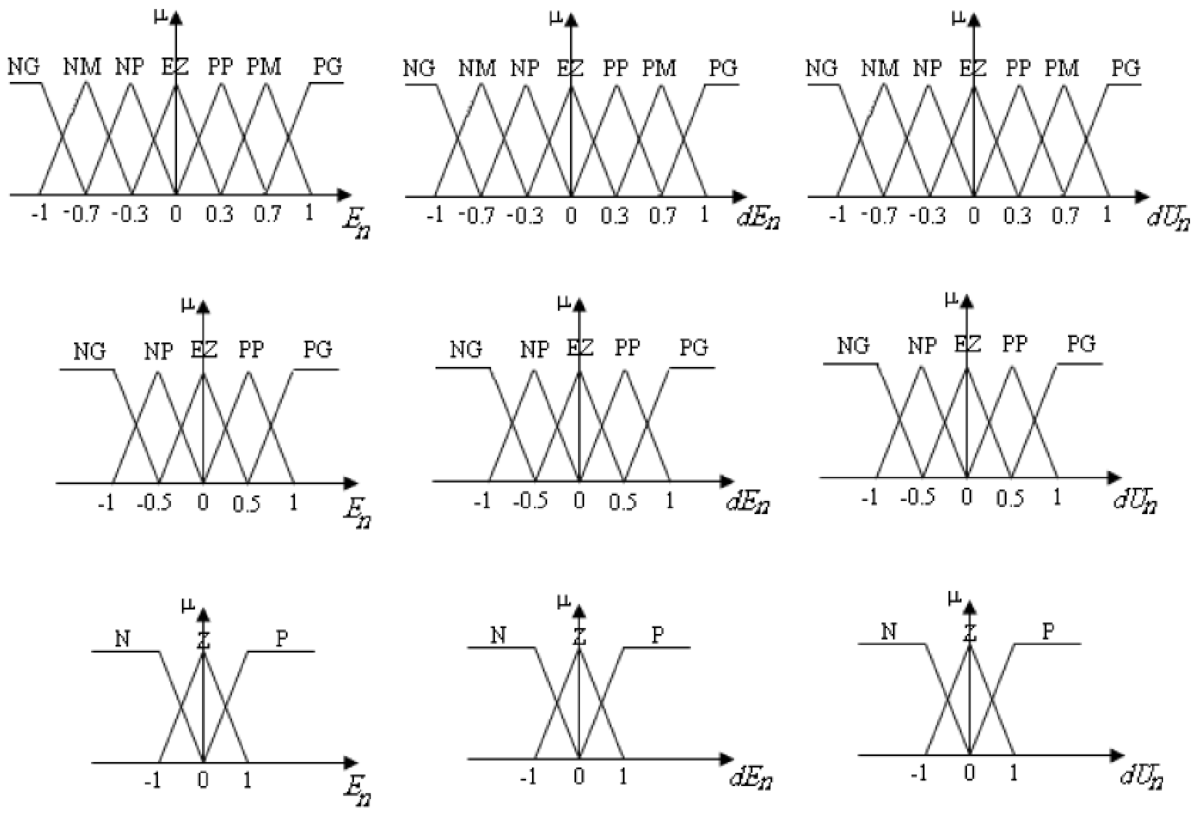


Figure IV. 3 Linear membership function or 3 5 and 7 sets

The rules of a fuzzy controller rely on the expertise and experience of human operators.

In the context of regulation, error and error variation are used, translated into fuzzy variables. Subsequently, it becomes possible to determine the rules in the temporal domain based on temporal analysis to control it according to the objectives set in a closed loop.

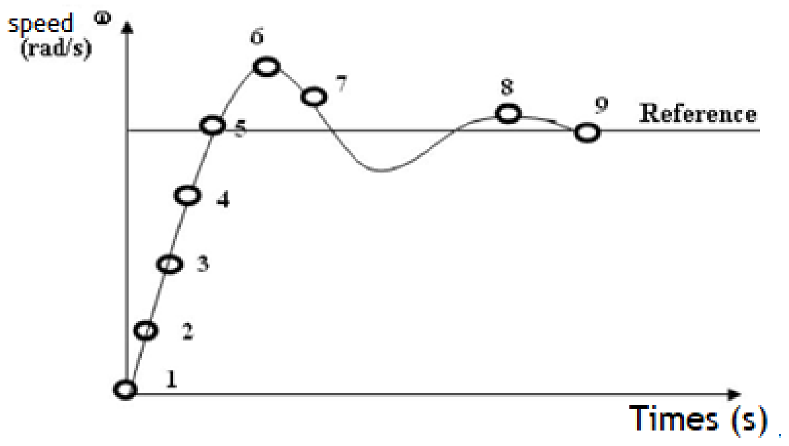


Figure IV. 4 Rules following temporal analysis.

From this, we can define the following rules:

IF E is PG AND dE is EZ THEN dU is PG (start-up)

IF En is PG AND dEn is NP THEN dUn is PM (increase in control to reach equilibrium)

IF E is PM AND dE is NP THEN dUn is PP (very small increase in control to avoid overshoot)

IF En is PP AND dEn is NP THEN dUn is EZ (convergence towards correct equilibrium)

IF En is EZ AND dE is NP THEN dU is NP (braking of the process)

IF E is NM AND dE is EZ THEN dU is NM (recall of the process towards equilibrium)

IF E is NP AND dE is PP THEN dU is EZ (braking and reversal of the control variation)

IF E is NP AND dE is EZ THEN dU,, is NP (convergence towards the correct equilibrium)

IF E,, is EZ AND dE,, EZ THEN dU,, is EZ (equilibrium)..

E_n / dE_n	NG	NM	NP	EZ	PP	PM	PG
NG	NG	NG	NG	NG	NM	NP	EZ
NM	NG	NG	NG	NM	NP	EZ	PP
NP	NG	NG	NM	NP	EZ	PP	PM
EZ	NG	NM	NP	EZ	PP	PM	PG
PP	NM	NP	EZ	PP	PM	PG	PG
PM	NP	EZ	PP	PM	PG	PG	PG
PG	EZ	PP	PM	PG	PG	PG	PG

dU_n		dE_n				
		NG	NP	EZ	PP	PG
E_n	NG	NG	NG	NP	NP	EZ
	NP	NG	NP	NP	EZ	PP
	EZ	NP	NP	EZ	PP	PP
	PP	NP	EZ	PP	PP	PG
	PG	EZ	PP	PP	PG	PG

dU_n		dE_n		
		N	Z	P
E_n	N	N	N	Z
	Z	N	Z	P
	P	Z	P	P

Figure IV. 5 Rule tables according to the inputs and output use membership functions

4.5. Adaptive fuzzy controller example

The proposed adaptive fuzzy controller (AFC) is based on Sugeno method thanks to its computational efficiency and it is well suited for linear technique, such as *PI* conventional controllers. The main difference between Mamdani and Sugeno is that the Sugeno output membership functions are either linear or constant. Inputs are the error between the actual and the reference speed, its first derivative and the V_{sat} variable due to the saturation of i_{sq} :

$$\begin{cases} E_r(t) = \Omega^*(t) - \Omega(t) \\ dE_r(t) = E_r(t) - E_r(t-1)\{ \\ V_{sat} = I_{sqmax} \end{cases} \quad (4.1)$$

Output is the weight to be used in order to adapt the ***PI*** controller by adjusting in real time proportional and integral action using the center of gravity method as follows:

$$W = \frac{\sum_{i=1}^{n} c_i . \mu_i}{\sum_{i=1}^{n} \mu_i} \quad (4\ 2)$$

Scale factors gain defined as GEr, GdEr, Gsat, GW_Integ and GW_Prop are used to make the AFC sensitive and near to the normalized defined input and output range values. Input variables fuzzy set are Negative: N. Positive: P. Zero: Z. The values range of output variables are: Zero: Z. Positive normal: PN. Positive Big: PB.

The following heuristic considerations have been noted from the observation of the process behavior: Integral action: Overshoot is mainly caused by integral term. Significant reduction causes system response to exceed the set point, Proportional action: Increasing proportional term reduces the leading time but increases the oscillations, Saturation: A variable depending on i_{sq} current is introduced for limitations due to the saturation. The use of triangular membership functions for the inputs and singleton in output is advantageous for time calculation

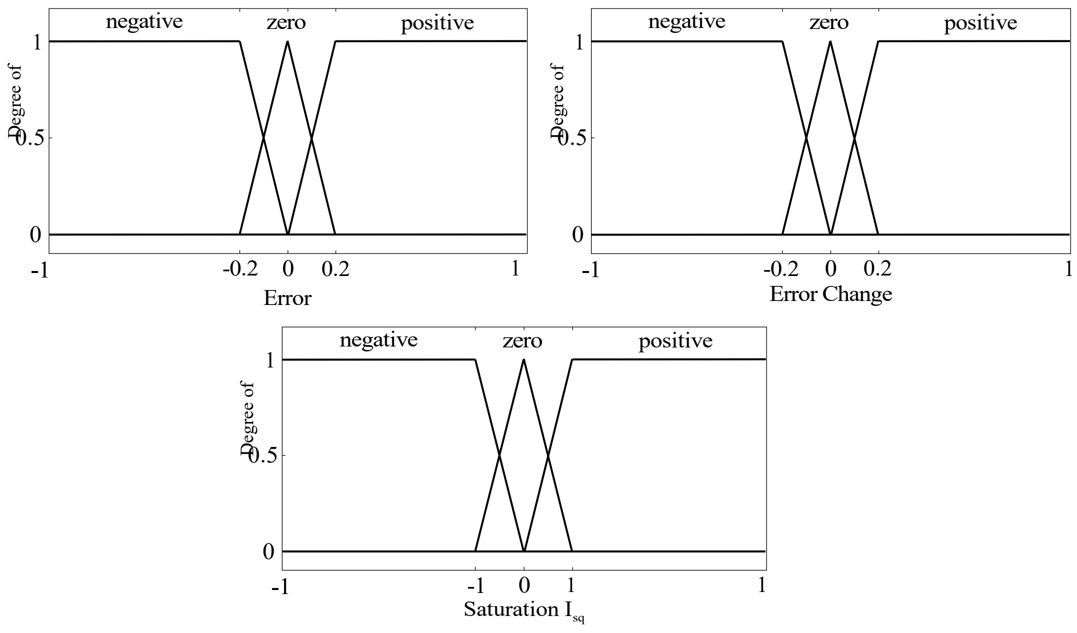


Figure IV. 6 . Input membership functions of the *AFC*

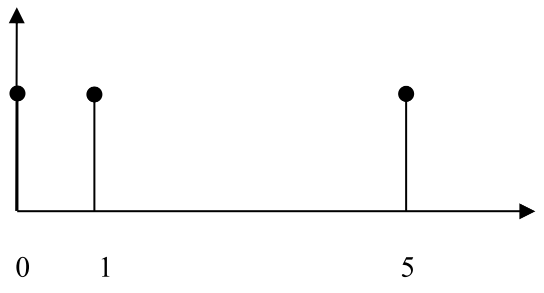


Figure IV. 7 Output membership functions of the *AFC*

Table 4. 1 Rule table:

Er	**dEr**	**Vsat**	**PI**
N	/	N	Z
N	/	P	PG
P	/	N	PG
P	/	P	Z
Z	/	/	PN
/	/	Z	PN
/	Z	/	PN
N	N	Z	PG
P	P	Z	PG

Homework

- Implement the Simulink model with a fuzzy controller based on a given system.
- Respect the system parameters and verify the resulting waveforms.
- Verify the operation after varying the normalization gains.
- Compare the results obtained with those of a conventional PI controller.

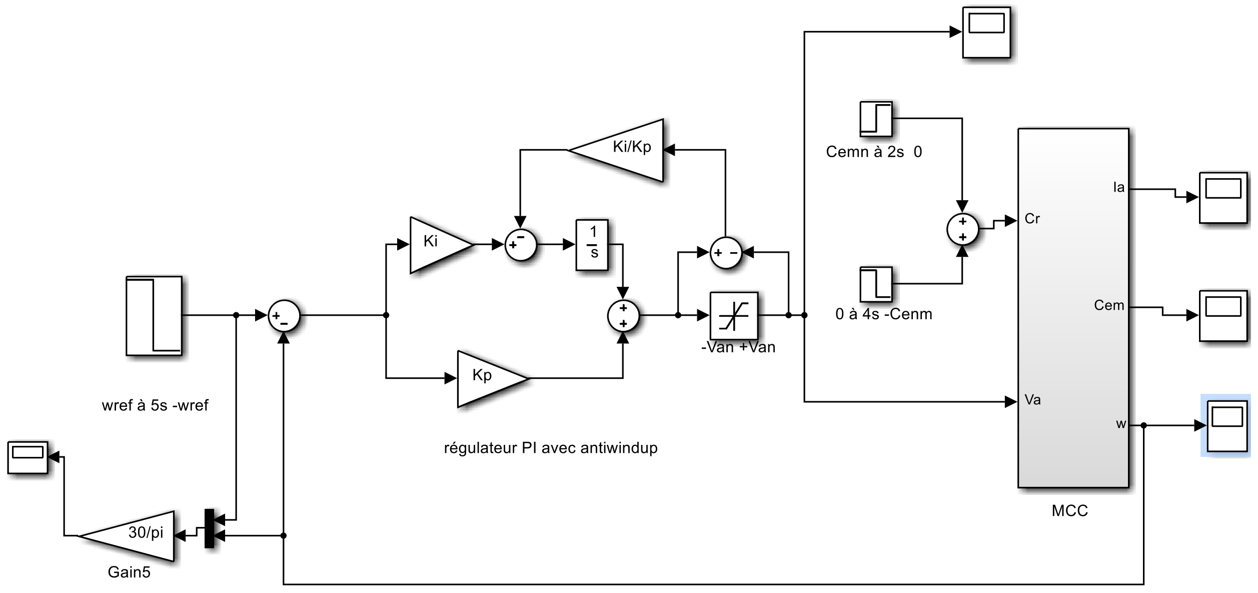


Figure IV. 8 Global system Simulink Bloc diagram

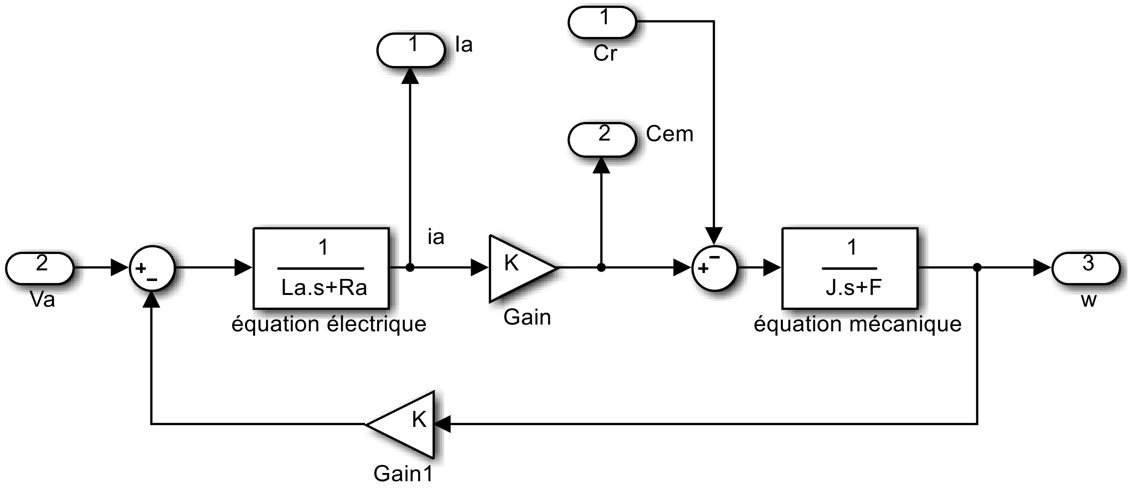


Figure IV. 9 Mcc Simulink Bloc Diagram

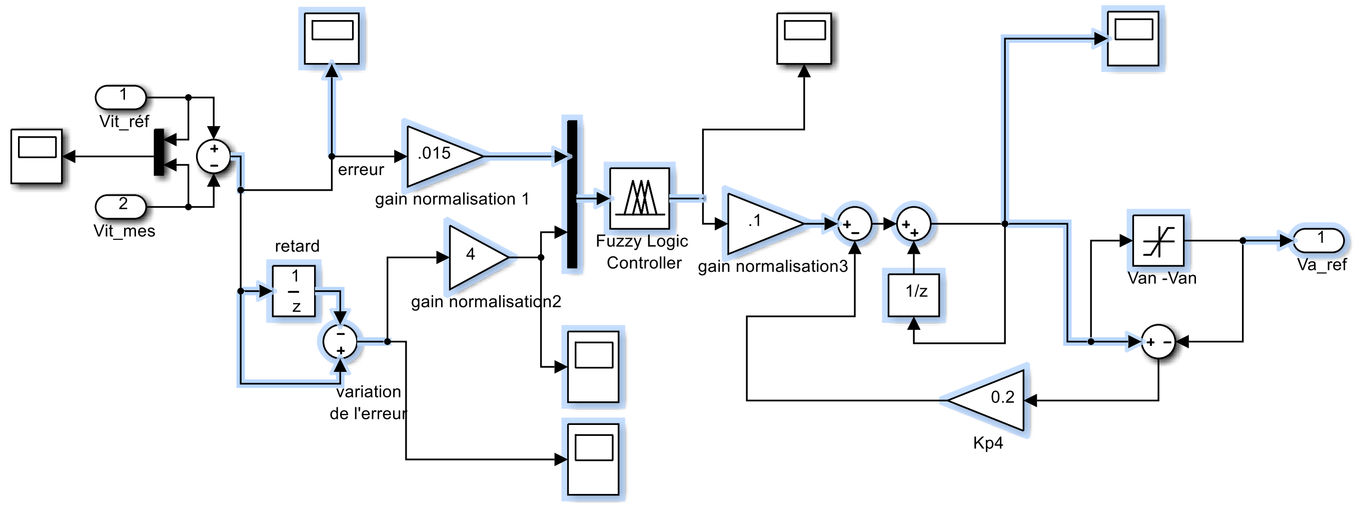


Figure IV. 10 Internal fuzzy controller Bloc Diagram

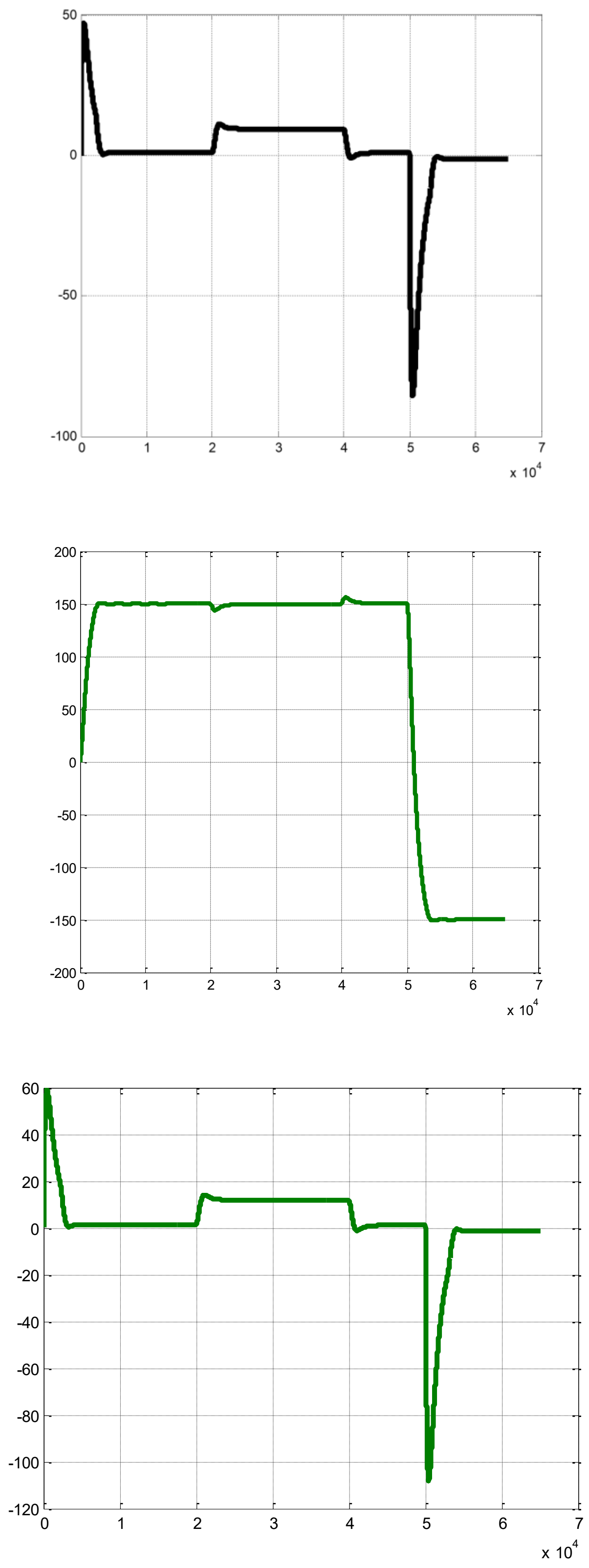


Figure IV. 11 System output characteristics (to check your model)

%System parameters

```
vn=220; ian=0.55;ien=0.1;

ven=220;  ra=43.07; Nn=2000;

pa=0.1*1e+3; la=0.7689; ta=0.0178;

ka=0.9788;  Csec=0.017; f=0.00042;

J=9.32*1e-4; tmec=2.219; kconv=47.67;
```

%PID Controller

```
kmw=0.00955; kp =2.5576 ki =49.6522 kd =0.0605
```

4.6. Advantages and disadvantages of fuzzy logic control

Advantages of Fuzzy Control

Can handle imprecise, vague, or uncertain information.

Does not require an exact mathematical model of the system.

Closer to human reasoning and decision-making, making it intuitive to design.

Flexible and adaptable to different systems and operating conditions.

Can integrate qualitative knowledge from human experts.

Disadvantages of Fuzzy Control

Design of the rule base can become complex for systems with many variables.

Performance depends on the quality and completeness of the rules.

Lack of standard methodology for tuning membership functions and rules.

Computationally more intensive than simple classical controllers for large-scale systems.

May be difficult to guarantee stability and robustness formally

4.7. References

[1] . Mamdani, E. H. & Assilian, S.An Experiment in Linguistic Synthesis with a Fuzzy Logic Controller, International Journal of Man-Machine Studies, vol. 7, pp. 1–13, 1975.

[2] . Zadeh, L. A. Outline of a New Approach to the Analysis of Complex Systems and Decision Processes, IEEE Transactions on Systems, Man, and Cybernetics, vol. 3, no. 1, pp. 28–44, 1973.

[3] . Ross, T. J. Fuzzy Logic with Engineering Applications, 4th Edition, Wiley, 2016.

[4] . Driankov, D., Hellendoorn, H., & Reinfrank, M. An Introduction to Fuzzy Control, 2nd Edition, Springer, 1996. ➤ Manuel classique dédié exclusivement à la commande floue.

[5] . Passino, K. M. & Yurkovich, S. Fuzzy Control, Addison-Wesley, 1998.

[6] . Tanaka, K. & Wang, H. O. Fuzzy Control Systems Design and Analysis: A Linear Matrix Inequality Approach, Wiley, 2001.

[7] . Lee, C. C. Fuzzy Logic in Control Systems: Fuzzy Logic Controller – Parts I & II,

[8] . IEEE Transactions on Systems, Man, and Cybernetics, 1990.

[9] . Commande Floue Adaptative et Neuro-FloueZadeh, L. A.Fuzzy Sets, Information and Control, vol. 8, no. 3, pp. 338–353, 1965.

5 1. Introduction

Artificial neural networks belong to the second major paradigm of artificial intelligence, commonly referred to as the numerical or connectionist approach. The primary objective of this approach is to minimize an error criterion between the output of a model and a reference signal, thereby producing a model that closely reproduces the behavior of the system under study. This optimization-based framework contrasts with symbolic AI, which relies on explicit rules and logical reasoning.

Neural networks are fundamentally based on parameterized nonlinear functions, whose parameters are adjusted through learning algorithms. Due to their strong approximation capabilities, neural networks have been successfully applied across a wide range of domains, including chemistry, biology, finance, medicine, signal processing, and control engineering.

In the fields of system identification and control, neural networks play a central role. They enable the construction, through learning, of a broad class of models and controllers capable of representing highly nonlinear and uncertain systems. Neural networks can be used both to identify unknown system dynamics from input–output data and to design control laws that adapt to changes in system behavior.

A neural network can be trained to perform a specific task—such as pattern recognition, function approximation, classification, or control—by adjusting the connection weights using learning algorithms such as gradient descent and backpropagation. During training, the network iteratively updates its parameters to minimize a predefined cost function, typically representing the discrepancy between the network output and the desired output. This learning capability allows neural networks to generalize from data and to adapt to new operating conditions.

In control engineering, neural networks are employed to design intelligent and adaptive controllers. They may be used as:

Direct controllers,

Inverse models,

Adaptive or self-tuning controllers,

Components of hybrid control structures (e.g. neural–PID or neural–fuzzy controllers).

Through learning, neural networks can construct a very broad class of models and control laws, enabling real-time adaptation to changes in system parameters or operating conditions

5 2. . Historical of Artificial Neural Networks

- ***1890 – William James***

  In 1890, the renowned American psychologist William James introduced the concept of associative memory in his seminal work on psychology. He proposed that mental associations are formed through repeated co-activation of ideas, a principle that would later serve as the conceptual foundation for Hebbian learning. James's insights highlighted the importance of experience-based learning in biological cognition.

- ***1943 – McCulloch and Pitts***

  In 1943, Warren McCulloch and Walter Pitts proposed one of the first mathematical models of a biological neuron. Their work introduced a network of formal neurons capable of performing basic logical operations such as AND, OR, and NOT. This model demonstrated that networks of simple processing units could implement logical reasoning, thereby establishing a theoretical link between neuroscience and computation.

- ***1949 – Donald Hebb***

  In 1949, the American physiologist Donald O. Hebb explained learning and conditioning in animals through the adaptive properties of neurons. In his book The Organization of Behavior, Hebb formulated a learning principle stating that the strength of a synaptic connection increases when the connected neurons are activated simultaneously. This principle, now known as the Hebbian learning rule, became a cornerstone of neural learning theory.

- ***1957 – Frank Rosenblatt***

  In 1957, Frank Rosenblatt developed the Perceptron model, one of the earliest learning algorithms for artificial neural networks. He also constructed the first neurocomputer based on this model. Inspired by the human visual system, the perceptron was designed for pattern recognition tasks. However, despite its innovative nature, the perceptron remained largely theoretical and suffered from significant limitations, particularly its inability to solve non-linearly separable problems.

- ***1960 – Bernard Widrow***

  In 1960, Bernard Widrow, a specialist in control engineering, introduced the ADALINE (Adaptive Linear Element) model. Structurally similar to the perceptron, ADALINE differed in its learning mechanism, which was based on minimising a mean square error criterion. This model played a significant role in adaptive signal processing and laid the groundwork for modern adaptive learning algorithms.

- ***1982 – John J. Hopfield***

  In 1982, the physicist John J. Hopfield revitalised interest in artificial neural networks by presenting a theoretical framework describing their operation and computational capabilities. Hopfield networks demonstrated how neural systems could function as associative memories and optimisation mechanisms, providing strong theoretical justification for neural computation.

- ***1983 – Boltzmann Machine***

  In 1983, the Boltzmann Machine was introduced as the first neural network model capable of overcoming many of the limitations associated with the perceptron. By incorporating stochastic elements and energy-based learning, it enabled more powerful representations. However, its practical application was limited by extremely slow convergence and prohibitive computational costs.

1985 – Backpropagation Algorithm

In 1985, the backpropagation of gradient algorithm emerged as a major breakthrough in neural network training. This learning algorithm enabled efficient training of multilayer neural networks by propagating error gradients backward through the network. Backpropagation remains one of the most extensively studied and widely applied learning algorithms, forming the foundation of modern deep learning systems.

5 3. . Biological Neuron

The neuron is the fundamental cell of nervous tissue and constitutes the basic functional unit of the nervous system. It is a specialized cell designed to receive, process, and transmit information in the form of electrical and chemical signals.

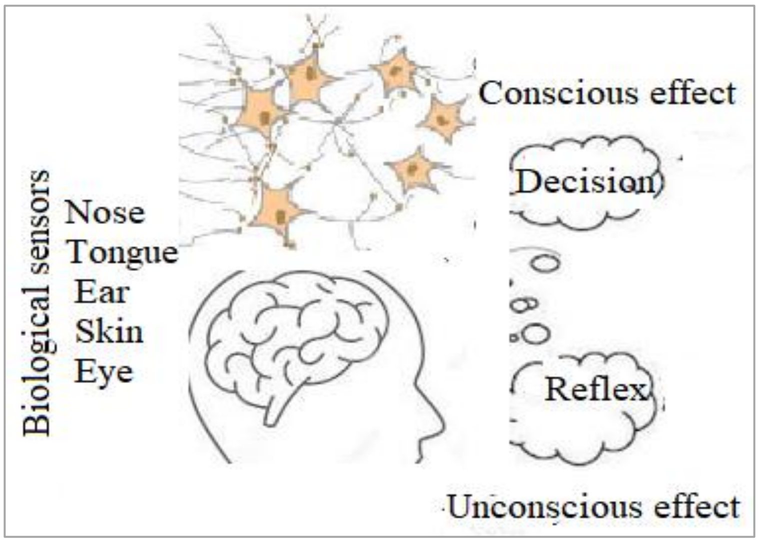


Figure V. 1 Connections of biological neural networks

A biological neuron is composed of a cell body, also known as the soma, which contains the nucleus and most of the cellular organelles. Extending from the cell body are multiple branched structures called dendrites, whose primary function is to receive information from other neurons or external stimuli. These incoming signals are conveyed from the dendrites towards the cell body, where they are integrated and processed.

The transmission of information from the neuron to other neurons is ensured by a long, slender projection known as the axon. The axon conducts electrical impulses, called action potentials, away from the cell body towards other neurons, muscles, or glands. In many neurons, the axon is covered by a myelin sheath, which increases the speed and efficiency of signal transmission.

Between neurons, there exists a very small intercellular gap, typically measuring a few tens of angstroms (on the order of 10^{-9} meters), known as the synapse. The synapse plays a critical role in neural communication, as it is the site where electrical signals are converted into chemical signals and transmitted to the next neuron via neurotransmitters.

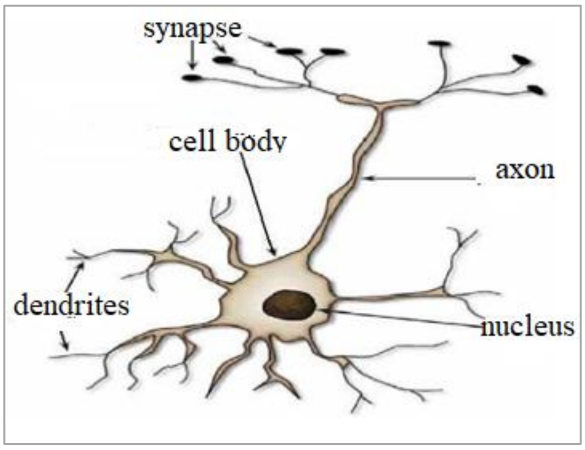


Figure V. 2 Simplified diagram of a biological neuron

Each component of the neuron performs a specific function, and the interaction between these components enables complex information processing in biological neural systems. The roles of the different parts of a biological neuron are summarized in the following table.

Table 5. 1 Functioning of the biological neuron.

Part	**Function**
Cell body	Nerve impulse reception Processing Activation
Axon	Transmission
Synapse	Electrical-chemical transformation Transmission of the signal from the axon Reception of the electrical-chemical signal
Dendrites	Signal transmission from other cell bodies.

5 4. . Formal (Artificial) Neuron

The formal neuron, also referred to as an artificial neuron, is a mathematical abstraction of the biological neuron. Its purpose is to model, in a simplified yet effective manner, the essential mechanisms of information processing observed in biological neural systems.

The earliest and most influential formalization of the neuron emerged from the seminal work of Warren McCulloch and Walter Pitts in 1943. Their model provided the first theoretical framework for representing neural activity using logical and mathematical principles. By idealizing neuronal behavior, they demonstrated that networks of simple artificial neurons could perform basic logical operations and, in principle, support complex computations.

Figure V.3 illustrates a basic model of a formal neuron. In this model, the neuron receives multiple input signals, each associated with a synaptic weight that represents the strength of the corresponding connection. These weighted inputs are combined through a summation process and compared to a threshold or bias. The result is then processed by an activation function, which determines the neuron’s output.

Despite its simplicity, the formal neuron constitutes the fundamental building block of artificial neural networks. By interconnecting large numbers of such neurons, it becomes possible to construct powerful computational models capable of learning, generalization, and nonlinear function approximation

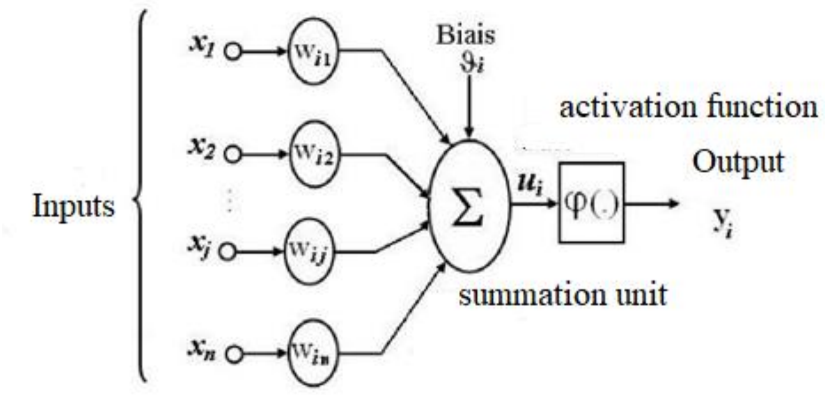


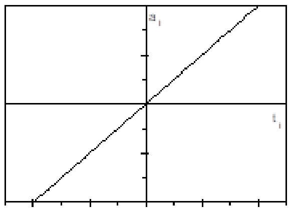
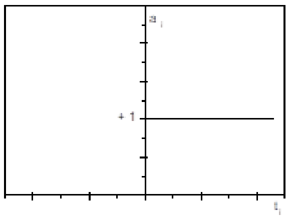
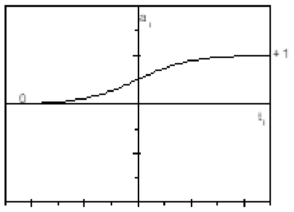
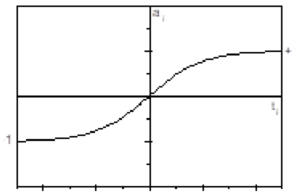
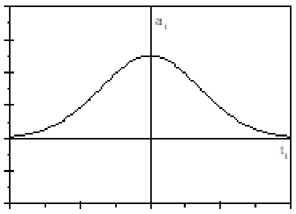
Figure V. 3 Basic model of a formal neuron

- The inputs of the formal neuron xi, i=1,2,...,n ;
- The weighting parameters, also called weights wij,
- The activation or thresholding function (non-linear, sigmoid shape, etc.),
- - The output S of the formal neuron.
- The output uk of the formal neuron is given by the following relation:
- The activation function will give the output value j of the neuron. This is the value that will be transmitted to the downstream neurons.

5 5. . Activation Function

There are many possible forms for the activation function. This is why an infinite number of possible values can be obtained.

Table 5. 2 Activation Functions

Linear function$a_i = \lambda . t_i$	
Threshold function $a_i = 1$ if $t_i \geq \vartheta_i$ $= 0$ else	
Sigmoid function $a_i = \frac{1}{1+\exp(-t_i)}$	
Hyperbolic tangent function$a_i = \frac{1-\exp(-t_i)}{1+\exp(-t_i)}$	
Gaussian function	

5 6. Neural Network Architecture

Artificial neural networks are composed of simple processing elements working in parallel. Their structure and operation are strongly inspired by the biological nervous system, particularly by the way biological neurons transmit and process information through synaptic connections. The overall behavior of a neural network is primarily determined by the pattern of interconnections between neurons and the numerical values of the associated synaptic weights

Artificial neural networks are strongly inspired by the biological nervous system, particularly by the structure and functioning of biological neurons and synapses. In biological systems, neurons communicate through electrical impulses transmitted across synapses, whose strengths change as a result of learning and experience.

Similarly, in artificial neural networks, the synaptic weights represent the strength of the connections between neurons. The global behavior of the network is not determined by individual neurons but by the collective interaction of many simple processing units operating in parallel

Formally, a neural network can be described as a system of interconnected nonlinear operators that receives external signals through its inputs and generates output signals accordingly. These operators, known as artificial neurons, are organized into layers and operate collectively to process information.

Open-Loop Networks

The output signal is obtained directly after the input signal is applied. They are

unidirectional without feedback (feedforward) and have the structure of a combinational system.

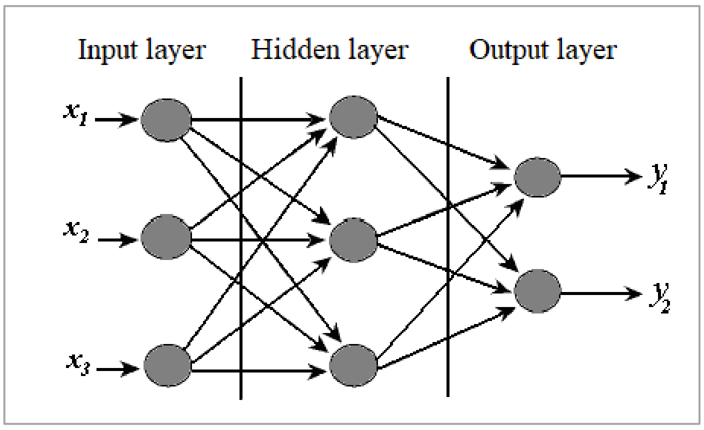


Figure V. 4 Feedforward network

Closed-loop networks

With feedback (feedback network or recurrent network) They have a structure similar to that of sequential systems

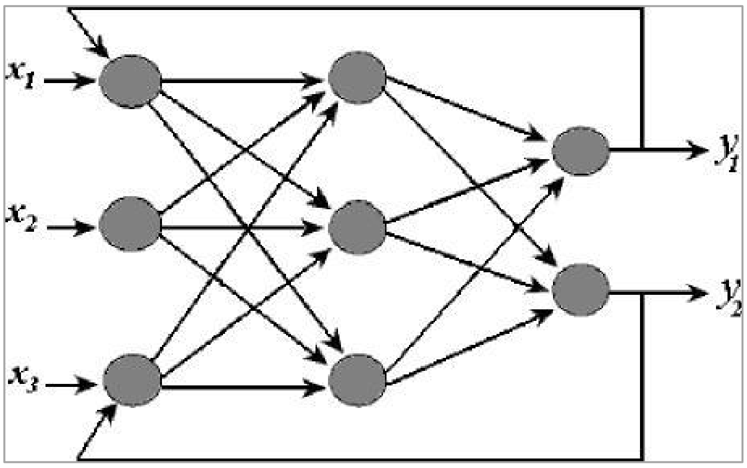


Figure V. 5 Feedback network

5 7. Learning

Learning and adaptation are the two essential characteristics of neural networks. The role of learning is to define the weight of each connection.

After this step, the network must make the correct associations for the input vectors it has not learned. This gives it the ability to recognize similar and even degraded forms of prototypes; this is the recognition phase.

Learning techniques are classified into three categories:

Supervised learning: A supervisor, or teacher, provides the network with input-output pairs using a learning method. Learning is complete when all the inputs and outputs are recognized by the network.

• **Unsupervised learning**: This learning method involves automatically detecting without the help of a supervisor—regularities in the presented examples and modifying the connection weights so that cases with the same regularity characteristics produce the same output.

• **Self-supervised** (**reinforcement) learning**: The neural network evaluates its own performance. An object is presented to the neural network's input, and the network has been informed of the class to which this object belongs.

5 8. Neural Networks in System Identification and Control

In system identification, neural networks are used to approximate unknown system dynamics based on observed input–output data. Given sufficient data and an appropriate network architecture, neural networks can accurately model nonlinear systems that are difficult or impossible to describe using classical analytical methods. Generally, neural network control involves an identification step and a control step. Identification uses learning to develop a neural model. The simplest models are based on learning from an existing conventional controller, others perform offline learning of the inverse model of the process or a reference model, and finally, others operate entirely online.

A neural network can be trained to perform a specific task—such as pattern recognition, classification, regression, prediction, or control—by adjusting its weights through a learning process. The most common training approach is supervised learning, where the network is provided with input–output pairs.

Direct Identification

The identifier neural network (INN) is used in parallel with a black-box process. The process output, y, is compared with the output of the neural network, y, and then the error between the actual measured value and the value estimated by the INN is used for correction

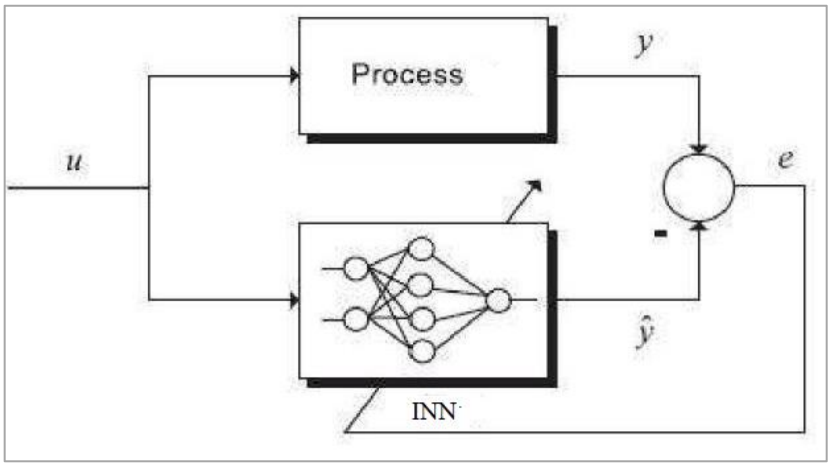


Figure V. 6 Diagram of direct process identification using a neural network

Inverse Identification

In this method, the input of the process is compared with the output of the neural identifier (INN). The output of the process is then injected as input to the neural network. Following this offline l/earning of the inverse model, the INN can be configured to provide direct control**.**

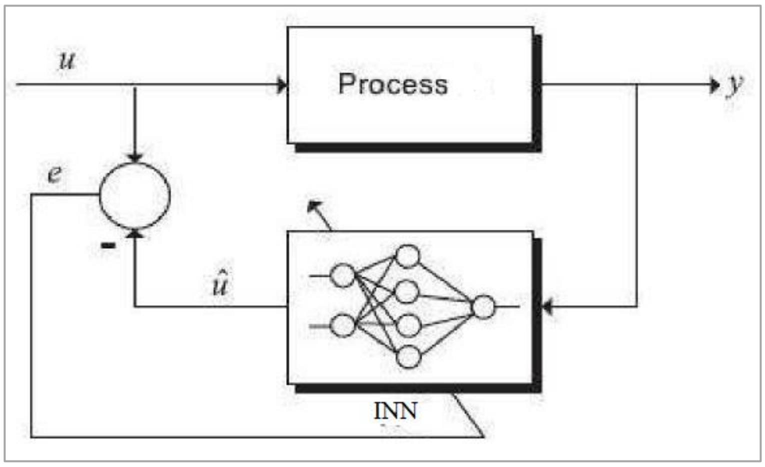


Figure V. 7 Inverse process identification scheme using a neural network.

Learning a Conventional Controller

A neural network can reproduce the behavior of an existing conventional controller (PI, PID, etc.). The RST-type control method poses serious problems in numerical integration and can therefore be used similarly.

The principle of direct identification of a conventional controller is presented by the following diagram.

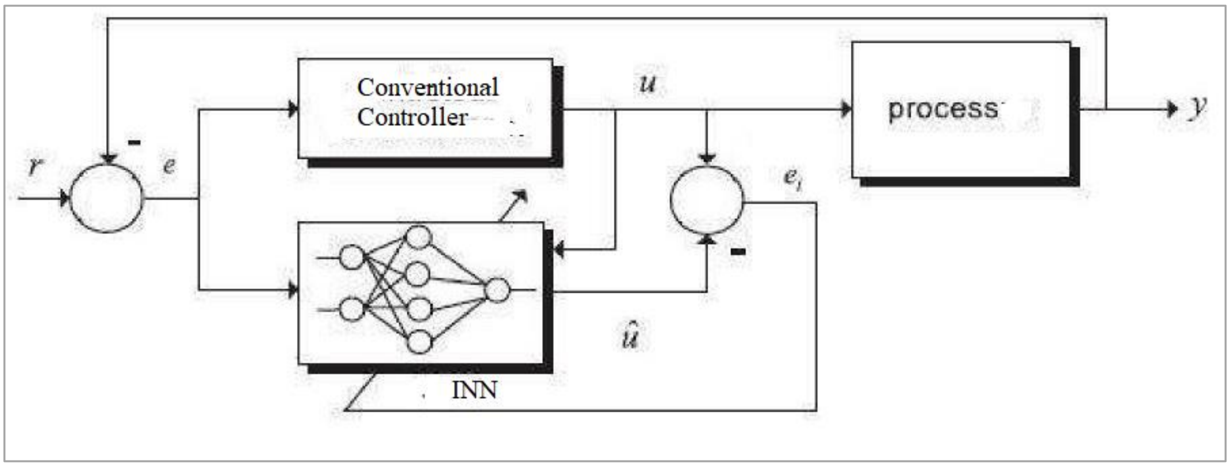


Figure V. 8 Basic learning scheme of a conventional controller

Inverse control with online learning

The inverse control scheme with a neural network control (NNC) is represented by the following figure

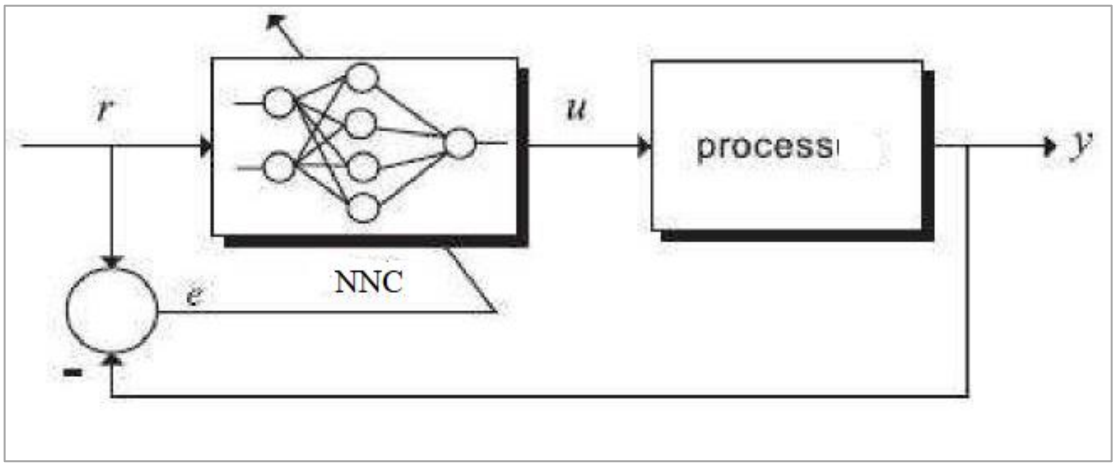


Figure V. 9 Reverse control scheme with a neural network control.

5 9. Advantages and Limitations

Advantages

- Strong capability for nonlinear approximation,
- Adaptation and learning from data,
- Robustness to noise and modelling uncertainties,
- Applicability to a wide range of problems.

Limitations

- Need for large data sets,
- Computational complexity,
- Lack of transparency (black-box behavior),
- Difficulty in guaranteeing stability in control applications.

5 10. Conclusion

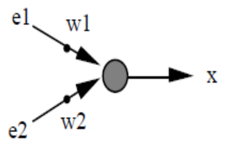
Artificial neural networks constitute a cornerstone of modern numerical artificial intelligence. Their ability to learn from data, approximate complex nonlinear relationships, and adapt to changing environments makes them particularly well suited for **system identification and intelligent control**. Inspired by biological neural systems yet grounded in mathematical optimization, neural networks continue to play a central role in the evolution of artificial intelligence and intelligent engineering systems..

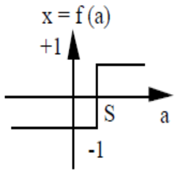
Homework

- Given that the weights of the two-input perceptron are:

  w1 = 0.5, w2 = 0.2, and that the threshold value is S = 0.0,

- Determine its behavior. Knowing that the behaviors of the logical AND, logical OR, and exclusive OR gates are recalled, we can manipulate the value of S.





e1	e2	AND	OR	XOR
1	1	1	1	1
1	-1	-1	1	-1
-1	1	-1	1	-1
-1	-1	-1	-1	1

Correction

For each combination, we determine the interval that must yield the same values shown in the column corresponding to the logical AND operation, and then the intersection so that all cases are verified simultaneously.

e1	e2	e1w1+e2w2	X	AND	Interval of S
1	1	1*0.5+1*0.2	0.7	1	]-∞ 0.7[
1	-1	1*0.5-1*0.2	0.3	-1	[0.3 +∞[
-1	1	-1*0.5+1*0.2	-0.3	-1	[-0.3 +∞[
-1	-1	-1*0.5-1*0.2	-0.7	-1	[-0.7 +∞[
Intersection of intervals					[0.3 0.7[

Within this interval of S, it behaves like a logical AND

Correction

e1	e2	e1w1+e2w2	X	OR	Interval of S
1	1	1*0.5+1*0.2	0.7	1	]-∞ 0.7[
1	-1	1*0.5-1*0.2	0.3	1	]-∞ 0.3[
-1	1	-1*0.5+1*0.2	-0.3	1	]-∞ -0.3[
-1	-1	-1*0.5-1*0.2	-0.7	-1	[-0.7 +∞[
Intersection of intervals					[-0.7 -0.3[

Within this interval of S, it behaves like a logical OR

e1	e2	e1w1+e2w2	X	XOR	Interval of S
1	1	1*0.5+1*0.2	0.7	1	]-∞ 0.7[
1	-1	1*0.5-1*0.2	0.3	-1	[0.3 +∞[
-1	1	-1*0.5+1*0.2	-0.3	-1	[-0.3 +∞[
-1	-1	-1*0.5-1*0.2	-0.7	1	]-∞ -0.7[
Intersection of intervals					Empty set ф

For these weight values, no value of S satisfies this behavior

Homework

To write a MATLAB program that approximates the function sin(x) with -2 ≤ x ≤ 2, we use the following conditions:

n = 10; % increasing difficulty function

k = 10; % limits the number of input data points

P = -2:(.4/k):2; T = sin(n*P);

• Explain the operation of the following instructions:

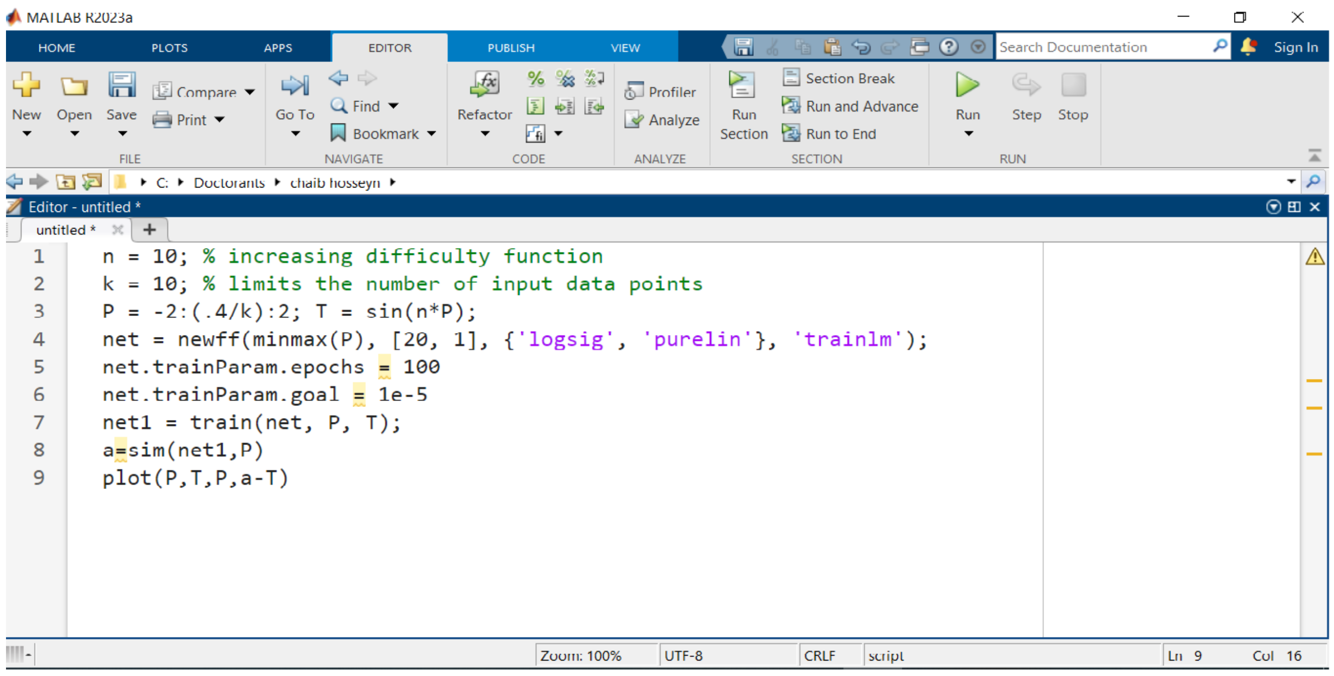
- net = newff(minmax(P), [20, 1], {'logsig', 'purelin'}, 'trainlm');
- net.trainParam.epochs = 100
- net.trainParam.goal = 1e-5
- [net, tr] = train(net, P, T);

• Write the command lines that will create a neural network object.

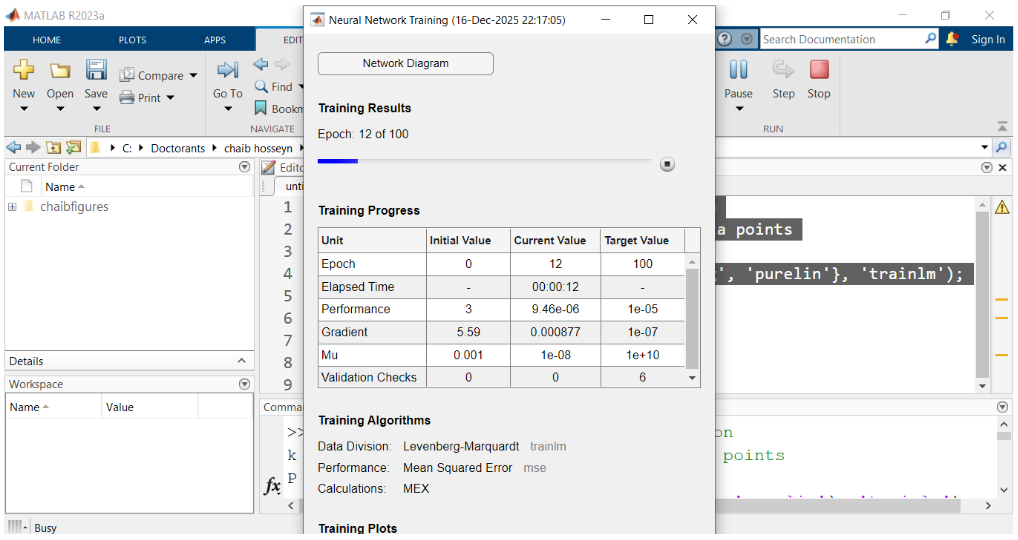
To verify the generalization capability, we will use a step size of 0.01.

• Write the commands that will simulate and display the results of the training.

Correction



```
n = 10; % increasing difficulty function
k = 10; % limits the number of input data points
P = -2:(.4/k):2; T = sin(n*P);
net = newff(minmax(P), [20, 1], {'logsig', 'purelin'}, 'trainlm');
net.trainParam.epochs = 100
net.trainParam.goal = 1e-5
net1 = train(net, P, T);
a=sim(net1,P)
plot(P,T,P,a-T)
```



Neural Network Training (16-Dec-2025 22:17:05)

Network Diagram

Training Results

Epoch: 12 of 100

Training Progress

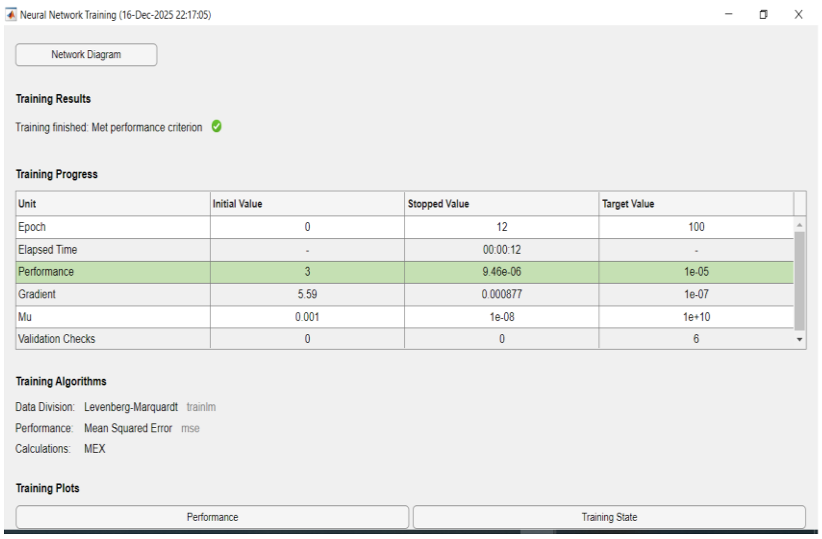
Unit	Initial Value	Current Value	Target Value
Epoch	0	12	100
Elapsed Time	-	00:00:12	-
Performance	3	9.46e-06	1e-05
Gradient	5.59	0.000877	1e-07
Mu	0.001	1e-08	1e+10
Validation Checks	0	0	6

Training Algorithms

Data Division: Levenberg-Marquardt trainlm
Performance: Mean Squared Error mse
Calculations: MEX

Training Plots

```
a points
', 'purelin'}, 'trainlm');
on
points
```



Neural Network Training (16-Dec-2025 22:17:05)

Network Diagram

Training Results

Training finished: Met performance criterion ✓

Training Progress

Unit	Initial Value	Stopped Value	Target Value
Epoch	0	12	100
Elapsed Time	-	00:00:12	-
Performance	3	9.46e-06	1e-05
Gradient	5.59	0.000877	1e-07
Mu	0.001	1e-08	1e+10
Validation Checks	0	0	6

Training Algorithms

Data Division: Levenberg-Marquardt trainlm
Performance: Mean Squared Error mse
Calculations: MEX

Training Plots

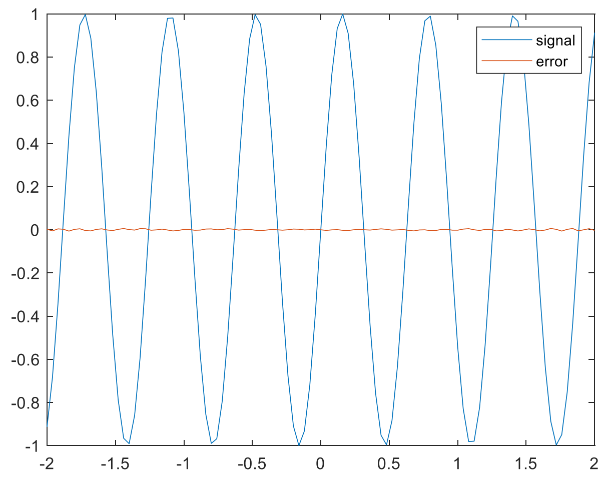


Figure V. 10 Artificial neural networks homework correction

5 11. References

[1] . Haykin, *Neural Networks and Learning Machines*, 3rd Edition, Pearson, 2009**.**

[2] . Bishop, Pattern Recognition and Machine Learning, Springer, 2006.

[3] . Rumelhart, D. E., McClelland, J. L.Parallel Distributed Processing, MIT Press, 1986.

[4] . Hertz, J., Krogh, A., & Palmer, R. G. Introduction to the Theory of Neural Computation, Addison-Wesley, 1991.

[5] . Goodfellow, I., Bengio, Y., & Courville, A. Deep Learning, MIT Press, 2016.

[6] . LeCun, Y., Bengio, Y., & Hinton, G.Deep Learning, Nature, vol. 521, pp. 436–444, 2015.

[7] . Schmid Huber, Deep Learning in Neural Networks: An Overview, Neural Networks, 2015.

[8] . McCulloch, W. S. & Pitts, W.A Logical Calculus of the Ideas Immanent in Nervous Activity, Bulletin of Mathematical Biophysics, 1943.

[9] . Hebb, D. O.The Organization of Behavior, Wiley, 1949. Introduction de la règle d'apprentissage de Hebb.

[10] . Rosenblatt, F. The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain, Psychological Review, 1958. Modèle fondateur du perceptron.

[11] . Widrow, B. & Hoff, M.Adaptive Switching Circuits, IRE WESCON Convention Record, 1960 Introduction de l'ADALINE et de la règle LMS.

[12] . Narendra, K. S. & Parthasarathy, K.Identification and Control of Dynamical Systems Using Neural Networks, IEEE Transactions on Neural Networks, 1990.

.

6 1. Introduction

Neuro-fuzzy systems provide a powerful framework for combining the learning capability of neural networks with the reasoning and interpretability of fuzzy logic. Depending on the degree of integration between the two paradigms, neuro-fuzzy architectures can be classified as parallel, cooperative, or hybrid, each offering distinct advantages for modelling, control, and decision-making under uncertainty.

6 2. Neuro-Fuzzy Systems

Neuro-fuzzy networks represent a class of intelligent systems in which fuzzy logic techniques are employed to enhance the learning process of artificial neural networks. In this approach, fuzzy concepts such as linguistic variables, fuzzy rules, and membership functions are integrated into neural learning mechanisms in order to improve interpretability, robustness, and adaptability when dealing with uncertainty and imprecise data.

6 3. Parallel Neural and Fuzzy Systems

In simultaneous neural–fuzzy systems, the neural network and the fuzzy system operate in parallel on the same task, without directly influencing each other's internal structures or parameters. Typically, the neural network is used either to pre-process the input data before it is fed into the fuzzy system or to post-process the outputs generated by the fuzzy inference mechanism. This architecture allows each subsystem to exploit its respective strengths while maintaining functional independence.

6 4. Cooperative Neuro-Fuzzy Models

In cooperative neuro-fuzzy models, the neural network is employed primarily as a learning and optimization tool for the fuzzy system. During the training phase, the neural network adjusts the parameters of the fuzzy system, including fuzzy rules, membership functions, and inference parameters. Once the learning process is completed, the fuzzy system operates independently, without further involvement of the neural network. This approach combines the learning capability of neural networks with the interpretability of fuzzy rule-based systems.

6 5. Hybrid Neuro-Fuzzy Models

Hybrid neuro-fuzzy models represent the most advanced and widely used neuro-fuzzy architectures in modern applications. In these systems, the neural network and the fuzzy system are tightly integrated within a unified and homogeneous architecture. Learning and fuzzy inference

are performed simultaneously, enabling automatic tuning of membership functions and rule parameters through neural learning algorithms. Well-known examples of this approach include Adaptive Neuro-Fuzzy Inference Systems (ANFIS). Hybrid models offer high modelling accuracy, adaptive behavior, and improved transparency compared to purely neural approaches.

6 6. ANFIS (Adaptive Network Fuzzy Inference System)

ANFIS represents a powerful hybrid framework that integrates fuzzy inference and neural learning into a unified architecture. By combining interpretability with adaptive learning, ANFIS provides an effective solution for modelling and controlling complex nonlinear systems under uncertainty.

Principle of Operation: The Adaptive Neuro-Fuzzy Inference System (ANFIS) is a hybrid intelligent system that combines the human-like reasoning capability of fuzzy logic with the learning and adaptation abilities of artificial neural networks. ANFIS was introduced by Jang (1993) as a systematic framework for constructing fuzzy inference systems whose parameters are automatically tuned using data-driven learning techniques.

ANFIS is typically based on a Sugeno-type fuzzy inference system, which is well suited for optimization and adaptive learning due to its mathematical structure.

ANFIS is organized as a multilayer feedforward network, where each layer performs a specific function in the fuzzy inference process. The architecture usually consists of five layers, each corresponding to a stage of fuzzy reasoning.

- Layer 1: Fuzzification Layer: In the first layer, the crisp input variables are transformed into fuzzy values using membership functions. Each neuron in this layer corresponds to a linguistic term (e.g. Low, Medium, High). Typical membership functions include Gaussian, triangular, and bell-shaped functions. The parameters of these membership functions are called premise parameters and determine the shape and position of the fuzzy sets.
- Layer 2: Rule Layer (Firing Strength Computation). Each node in the second layer represents a fuzzy rule. The output of each node is the firing strength of a rule, obtained by combining the membership degrees of the inputs using a fuzzy AND operator (usually multiplication). This layer determines the degree to which each fuzzy rule is activated.

- Layer 3: Normalization Layer: In this layer, the firing strengths of all rules are normalized to ensure that their relative influence is properly weighted. Normalization ensures numerical stability and facilitates parameter optimization.
- Layer 4: Consequent Layer: Each node in this layer computes the contribution of each rule to the output. In a Sugeno-type system, the consequent of each rule is a linear function of the inputs.
- Layer 5: Output Layer (Defuzzification). The final layer computes the overall system output by summing the outputs of all rules:

In Sugeno-type ANFIS, this summation directly provides a crisp output, eliminating the need for a separate defuzzification process. ANFIS employs a hybrid learning algorithm that combines: Least Squares Estimation (LSE) to optimize the consequent parameters, Gradient Descent to update the premise parameters. This two-step learning approach improves convergence speed and reduces computational complexity compared to purely gradient-based methods.

6 7. Advantages of ANFIS

- Automatic tuning of membership functions and fuzzy rules,
- High modelling accuracy for nonlinear systems,
- Interpretable rule-based structure,
- Strong capability for system identification and adaptive control.
- Typical Applications of ANFIS
- ANFIS has been successfully applied in:
- Nonlinear system identification,
- Adaptive and intelligent control,
- Signal processing,
- Fault diagnosis,
- Forecasting and decision-making systems.

6 8. ANFIS (Adaptive Network Fuzzy Inference System) in Matlab

The Anfisedit command displays the following window:

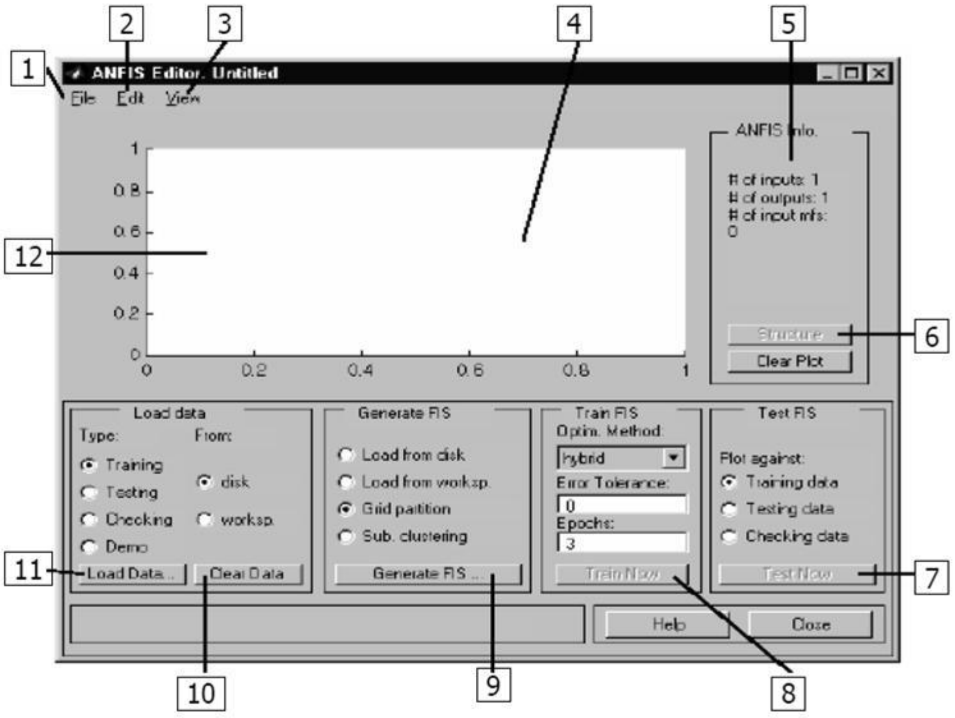


Figure VI. 1 ANFIS Window in MATLAB environment

1.File: Open or Save

2. Editing

3. Display via GUI

4. Graphics Area

5. Input/Output Status (Numbers)

6. Allows visualization of the graphical structure of the inputs/outputs.

7. Data Testing

8. Training the FIS after selecting the optimization method and error tolerance. It generates the error in the graphics area.

9. Import or create an FIS Model

10. Clear Data

11. Import Data

12. All data appears in this area.

Homework

Application 1

• The data vectors x = (0, 0, 1, 10) and y = sin(2*x)/exp(x/5) are used for training the system. Save this file as nfexemple1.fis

• Open this file with the fuzzy inference system editor.

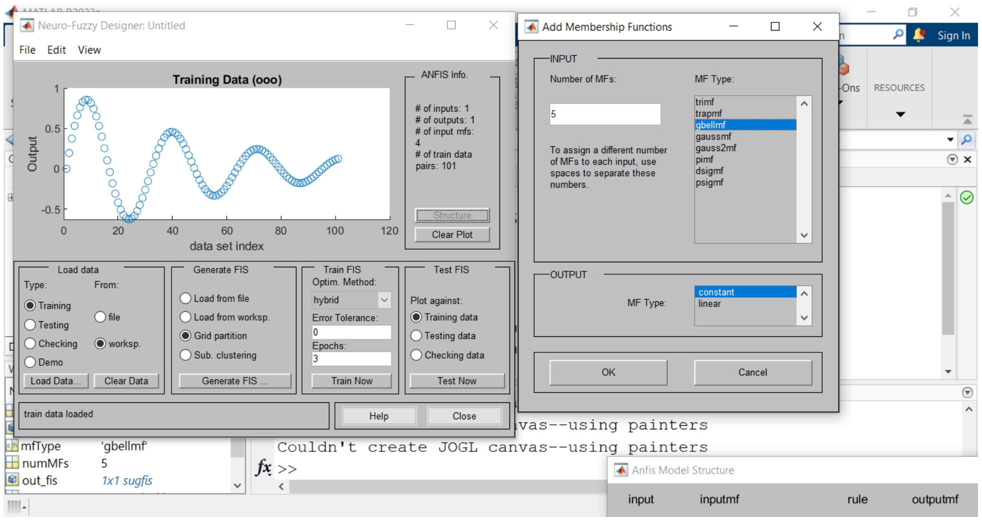
• Depending on the number and shape of the membership functions and the approximation method, choose the one that converges best

Application 2

Complete this table to obtain the following graphical structure

Program	Explanation

x = (0:0.1:10)';
 	
y = sin(2*x)./exp(x/5);
 	
trnData = [x y];
 	
numMFs = 5;
 	
mfType = 'gbellmf';
 	
epoch_n = 20;
 	
in_fis =genfis1(trnData,numMFs,mfType);
 	
out_fis = anfis(trnData,in_fis,20);
 	
plot(x,y,x,evalfis(x,out_fis));
 	
legend('Training Data','ANFIS Output')	



ANFIS Info.
# of inputs: 1
# of outputs: 1
# of input mfs: 4
# of train data pairs: 101

Type: [✓ Training] Testing Checking Demo
From: file [✓ worksp.]
Load from file / Load from worksp. / [✓ Grid partition] / Sub. clustering
Optim. Method: hybrid
Error Tolerance: 0
Epochs: 3
Plot against: [✓ Training data] Testing data Checking data

train data loaded

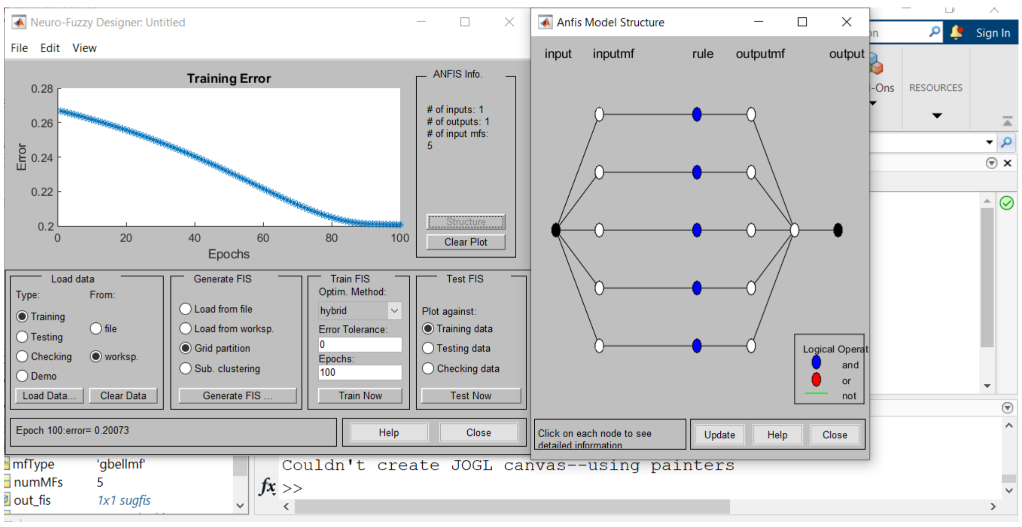
INPUT
Number of MFs: 5
To assign a different number of MFs to each input, use spaces to separate these numbers.
MF Type: trimf, trapmf, [✓ gbellmf], gaussmf, gauss2mf, pimf, dsigmf, psigmf

OUTPUT
MF Type: [✓ constant], linear

```
mfType    'gbellmf'
numMFs    5
out_fis   1x1 sugfis
```

```
vas--using painters
Couldn't create JOGL canvas--using painters
fx >>
```

Figure VI. 2 **Training Data and generating fis with grid partition**



ANFIS Info.
# of inputs: 1
# of outputs: 1
# of input mfs: 5

Type: [✓ Training] Testing Checking Demo
From: file [✓ worksp.]
Load from file / Load from worksp. / [✓ Grid partition] / Sub. clustering
Optim. Method: hybrid
Error Tolerance: 0
Epochs: 100
Plot against: [✓ Training data] Testing data Checking data

Epoch 100:error= 0.20073

Click on each node to see detailed information

```
mfType    'gbellmf'
numMFs    5
out_fis   1x1 sugfis
```

```
Couldn't create JOGL canvas--using painters
fx >>
```

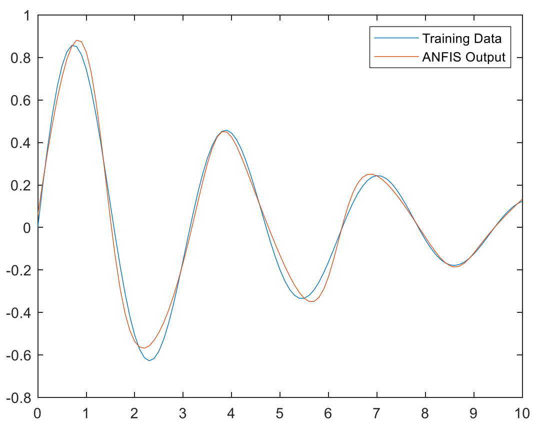
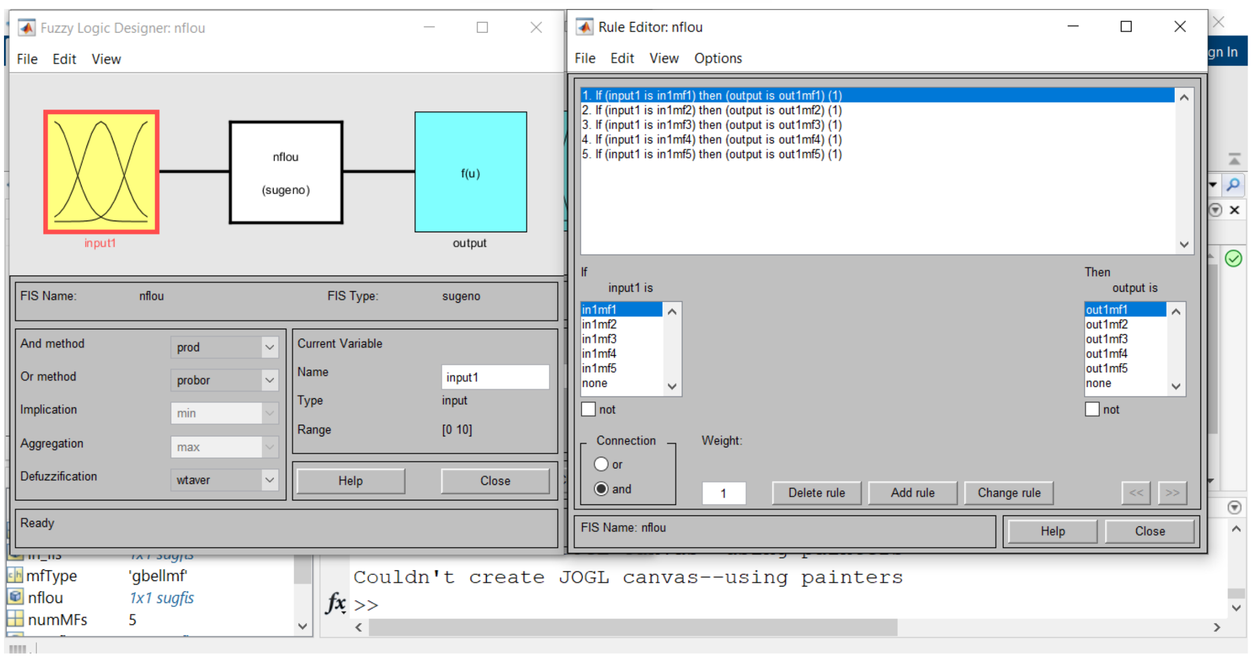


Figure VI. 3 Training data and Neural network structure.



FIS Name: nflou FIS Type: sugeno

And method prod
Or method probor
Implication min
Aggregation max
Defuzzification wtaver

Current Variable
Name input1
Type input
Range [0 10]

Ready

1. If (input1 is in1mf1) then (output is out1mf1) (1)
2. If (input1 is in1mf2) then (output is out1mf2) (1)
3. If (input1 is in1mf3) then (output is out1mf3) (1)
4. If (input1 is in1mf4) then (output is out1mf4) (1)
5. If (input1 is in1mf5) then (output is out1mf5) (1)

If input1 is: in1mf1, in1mf2, in1mf3, in1mf4, in1mf5, none — not
Then output is: out1mf1, out1mf2, out1mf3, out1mf4, out1mf5, none — not

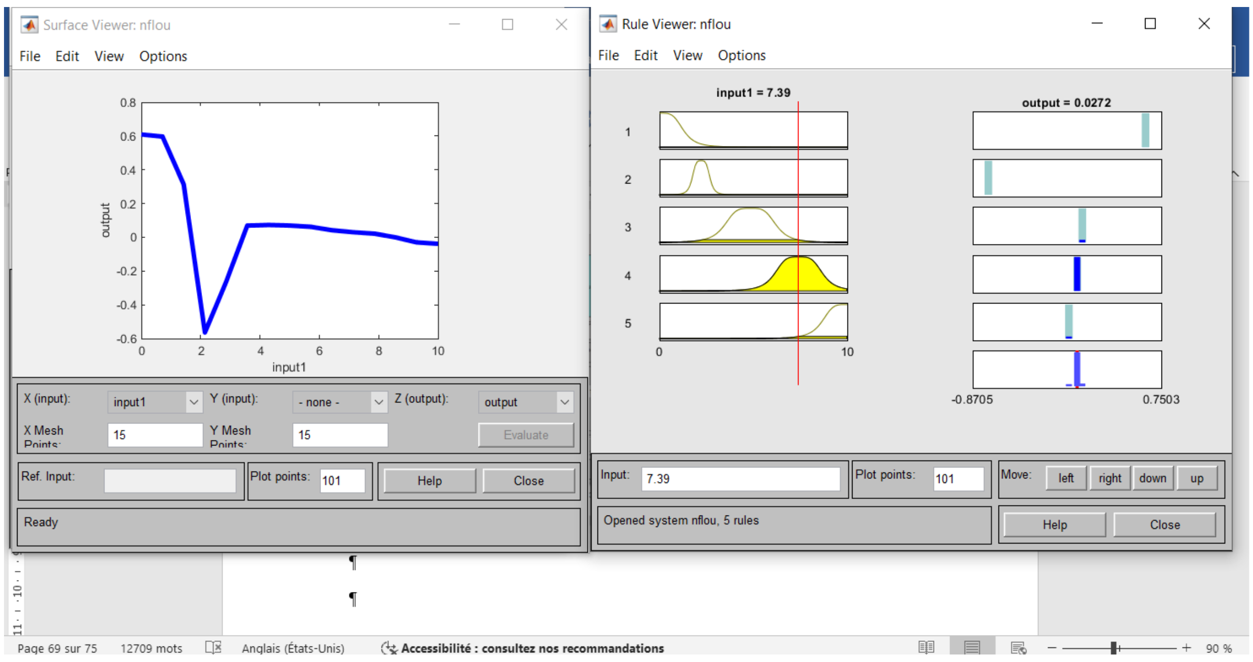
Connection: or [✓ and]
Weight: 1

FIS Name: nflou

mfType 'gbellmf'
nflou 1x1 sugfis
numMFs 5

```
Couldn't create JOGL canvas--using painters
>>
```

Figure VI. 4 Fuzzy inference system and Rule editor



X (input): input1 Y (input): - none - Z (output): output
X Mesh Points: 15 Y Mesh Points: 15
Ref. Input: Plot points: 101
Ready

Input: 7.39 Plot points: 101
Opened system nflou, 5 rules

Page 69 sur 75 12709 mots Anglais (États-Unis) Accessibilité : consultez nos recommandations 90 %

Figure VI. 5 Rule viewer and surface viewer

Program	Explanation
x = (0:0.1:10)';	**Data column vector x: inputs**
y = sin(2*x)./exp(x/5);	**Data column vector x: outputs**
trnData = [x y];	**Coupled reference model (inputs/outputs)**
numMFs = 5;	**Number of membership functions**
mfType = 'gbellmf';	**Shape of membership functions**
epoch_n = 20;	**Number of iterations**
in_fis =genfis1(trnData,numMFs,mfType);	**Generation of the system composed of data and parameters**
out_fis = anfis(trnData,in_fis,20);	**Calculation of outputs after data training**
plot(x,y,x,evalfis(x,out_fis));	**Display of the two reference shapes and estimated values**
legend('Training Data','ANFIS Output')	**Graph legend.**

6 9. References

[1] . Jang, J.-S. R.ANFIS: Adaptive-Network-Based Fuzzy Inference System, IEEE Transactions on Systems, Man, and Cybernetics, vol. 23, no. 3, pp. 665–685, 1993.

[2] . Jang, J.-S. R., Sun, C.-T., & Mizutani, E.Neuro-Fuzzy and Soft Computing: A Computational Approach to Learning and Machine Intelligence, Prentice Hall, 1997

[3] . Wang, L.-XA Course in Fuzzy Systems and Control,Prentice Hall, 1997.

[4] . Nauck, D., Klawonn, F., & Kruse, R.Foundations of Neuro-Fuzzy Systems,Wiley, 1997.

[5] . Passino, K. M. & Yurkovich, S.Fuzzy Control,Addison-Wesley, 1998.

7 1. Genetic Algorithms definition

A Genetic Algorithm (GA) is an iterative stochastic algorithm that uses a population of individuals representing the potential solutions to the optimization problem to be solved.

This population will evolve from generation to generation: the "best adapted" individuals will have a greater chance of reproducing and thus passing on their hereditary characteristics. An individual's genetic makeup is contained in a chromosome, which is made up of a set of genes whose values are in a binary alphabet or not. The evolutionary process is translated through selection and reproduction operators. Individuals are selected based on their adaptation. To reproduce, two mechanisms allow for the "production" of new individuals:

A Genetic Algorithm (GA) is a population-based search and optimisation algorithm inspired by the principles of natural selection and genetics. It belongs to the class of evolutionary algorithms, which aim to solve complex problems by mimicking the evolutionary processes observed in nature.

The fundamental idea behind a genetic algorithm is to combine a “survival of the fittest” strategy with a structured yet stochastic exchange of information. This approach allows the algorithm to efficiently explore large and complex search spaces, particularly in problems where the optimal solution is unknown or difficult to obtain using classical analytical methods.

7 2. Genetic Algorithms Vocabulary

- **Chromosome** In biology carrier of the genetic information necessary for the construction and functioning of the organism is seen as a possible solution to a problem of optimizing any function.

- **Genotype** In biological systems, the entire set of genetic material is called the genotype. In genomics, the set of chains is called the structure.

- **Phenotype** In biological systems, the organism formed by the interaction of all the genetic material with its environment is called the phenotype. In GAs, the decoded structures form a set of given parameters, or solutions, or points in the solution space.

- **Allele**. In natural systems, an allele is a component of a gene. Alleles are the different possible values that genes can take. In genetic engineering, an allele is the code used to represent a gene (binary, real, etc.).

- **Locus**: The locus is the position of a gene on the chromosome

- **Individual** In biological systems, an individual is a form that is the product of gene activity. For a genetic agent, it is reduced to a chromosome and is called a chromosome or an individual to designate the same object

- Population in a Genetic Algorithm (GA), a population refers to a collection of candidate solutions to a given optimization or search problem. Each candidate solution, called an individual, represents a possible solution encoded in the form of a chromosome. The **population** constitutes the fundamental working unit of the genetic algorithm. A generation corresponds to the state of the population at a particular iteration of the algorithm. As the algorithm evolves, the population is updated from one generation to the next through genetic operations such as selection, crossover, and mutation.

- **Parents** In a natural system, individuals who can reproduce by creating new individuals forming a new generation in order to ensure the continuity of life. In the context of a GA, parents correspond to individuals capable of producing new offspring to form a new generation.

- Offspring (**Children**) The new individuals forming a new generation, resulting from the genetic interactions between parents, produce children who can take on the genetic characteristics of their parents.

7 3. Genetic Algorithms Main steps

A Genetic Algorithm (GA) operates through a sequence of well-defined steps that are iteratively applied until a satisfactory solution is obtained or a convergence criterion is met.

- **1. Creation of the Initial Population** : The algorithm begins with the generation of an initial population composed of a set of individuals, each representing a candidate solution to the problem. This population is usually created randomly in order to ensure sufficient diversity and broad exploration of the search space. In some cases, heuristic or previously known solutions may also be included to accelerate convergence.

- **2. Evaluation of the Population** Each individual in the population is then evaluated using a fitness function, which quantitatively measures how well the candidate solution satisfies the objectives of the problem. The fitness value determines the quality of each individual relative to the others in the population.

- **3. Selection of the Best Individuals** Based on their fitness values, individuals are selected for reproduction. Selection mechanisms favor individuals with higher fitness, reflecting the principle of survival of the fittest. Common selection strategies include roulette wheel selection, tournament selection, and rank-based selection.
- 4**. Crossover and Mutation in the Mating Pool**: The selected individuals form a mating pool, from which new offspring are generated. Crossover combines genetic material from pairs of parent individuals to produce new solutions that inherit traits from both parents. Mutation introduces random modifications to certain genes, helping maintain genetic diversity and preventing premature convergence to suboptimal solutions.
- **5. Formation of a New Generation** : The offspring produced through crossover and mutation constitute a new generation. Depending on the replacement strategy, this new population may completely or partially replace the previous generation. Elitism is often applied to ensure that the best individuals are preserved.
- **6. Convergence Check and Iteration :** The algorithm then checks whether a convergence criterion has been satisfied. This criterion may be based on a maximum number of generations, a target fitness value, or stagnation of improvement. If convergence has not been achieved, the algorithm returns to Step 2 and the evolutionary cycle continues.

Through repeated evaluation, selection, and genetic variation, a genetic algorithm progressively improves the quality of solutions across generations, ultimately converging towards an optimal or near-optimal solution.

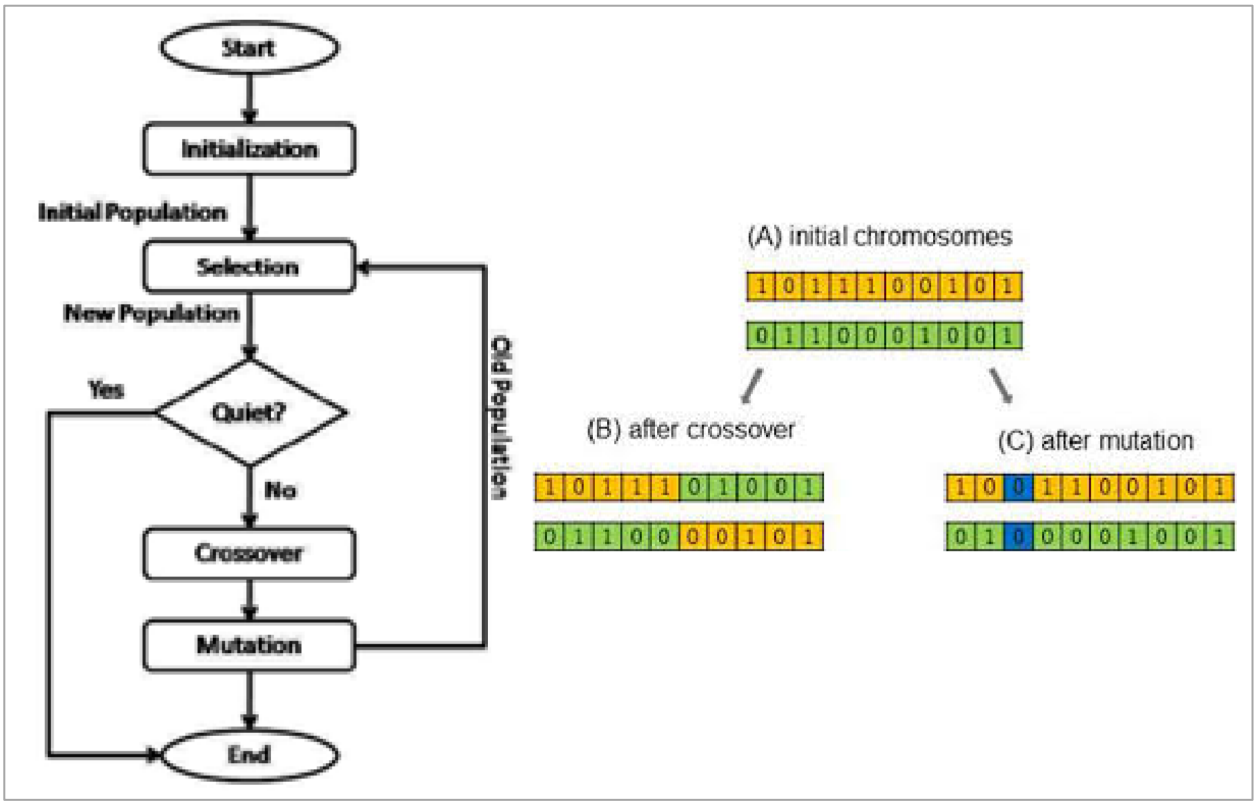


Figure VII. 1 Genetic Algorithms organizational chart

7 4. Evolutionary Process and Convergence

The population evolves over successive generations through repeated cycles of evaluation, selection, and reproduction. Over time, advantageous traits become more prevalent in the population, while less effective traits gradually disappear.

It can be demonstrated that, under appropriate conditions, genetic algorithms converge progressively towards high-quality solutions within a reasonable computational time. The stochastic nature of genetic algorithms allows them to escape local optima, making them particularly suitable for solving nonlinear, multimodal, and high-dimensional optimisation problems.

7 5. Analogy with Natural Evolution

As in biological evolution, the best traits—often referred to as dominant traits—are preserved and propagated across generations. Offspring inherit combinations of genes from their parents, resulting in continuous improvement of the population and increasing adaptation to the problem environment.

Genetic algorithms provide a robust and flexible optimization framework by combining randomness with structured evolutionary principles. Their ability to efficiently search complex solution spaces makes them widely applicable in areas such as control system design, machine learning, scheduling, and engineering optimization.

7 6. Using Genetic Algorithms in MATLAB

MATLAB provides a powerful and user-friendly environment for the implementation and application of Genetic Algorithms (GAs) through its Global Optimization Toolbox. This toolbox offers a wide range of built-in functions that facilitate the design, configuration, and execution of genetic algorithms for solving complex optimization problems. In MATLAB, a genetic algorithm is typically implemented by defining:

- An objective (fitness) function,
- A set of decision variables and constraints,
- A collection of algorithm parameters, such as population size, crossover rate, and mutation rate.
- The core function used to execute a genetic algorithm is the ga function, which applies evolutionary operators automatically to search for an optimal solution.

a. Defining the Fitness Function

The first step in using a genetic algorithm in MATLAB is to define the fitness function, which evaluates the quality of each candidate solution. This function must be written as a MATLAB function file or an anonymous function. The genetic algorithm seeks to minimise (or maximise) this fitness function.

Example: `fitnessFcn = @(x) x(1)^2 + x(2)^2;`

b. Setting Constraints and Variable Bounds

MATLAB allows the inclusion of linear and nonlinear constraints, as well as upper and lower bounds on the decision variables. This flexibility enables genetic algorithms to handle constrained optimisation problems commonly encountered in engineering applications.

c. Configuring Algorithm Parameters

The behavior and performance of the genetic algorithm can be customized using the Opti options function. Key parameters include:

- Population size,
- Selection function,
- Crossover fraction,

- Mutation function,
- Stopping criteria.

Example: options = optimoptions('ga','PopulationSize',100,'MaxGenerations',200);

d. Running the Genetic Algorithm

Once the fitness function, constraints, and options are defined, the genetic algorithm is executed using the ga function:

```
[x_opt, fval] = ga(fitnessFcn, nvars, [ ], [ ], [ ], [ ], lb, ub, nonlcon, options);
```

where:

- x_opt is the optimal solution found,
- fval is the corresponding fitness value,
- nvars is the number of decision variables.

e. Visualization and Analysis

MATLAB provides built-in tools for monitoring convergence and analysing algorithm performance. Users can plot fitness evolution, population diversity, and constraint satisfaction across generations. These visualization features are particularly useful for educational purposes and algorithm tuning.

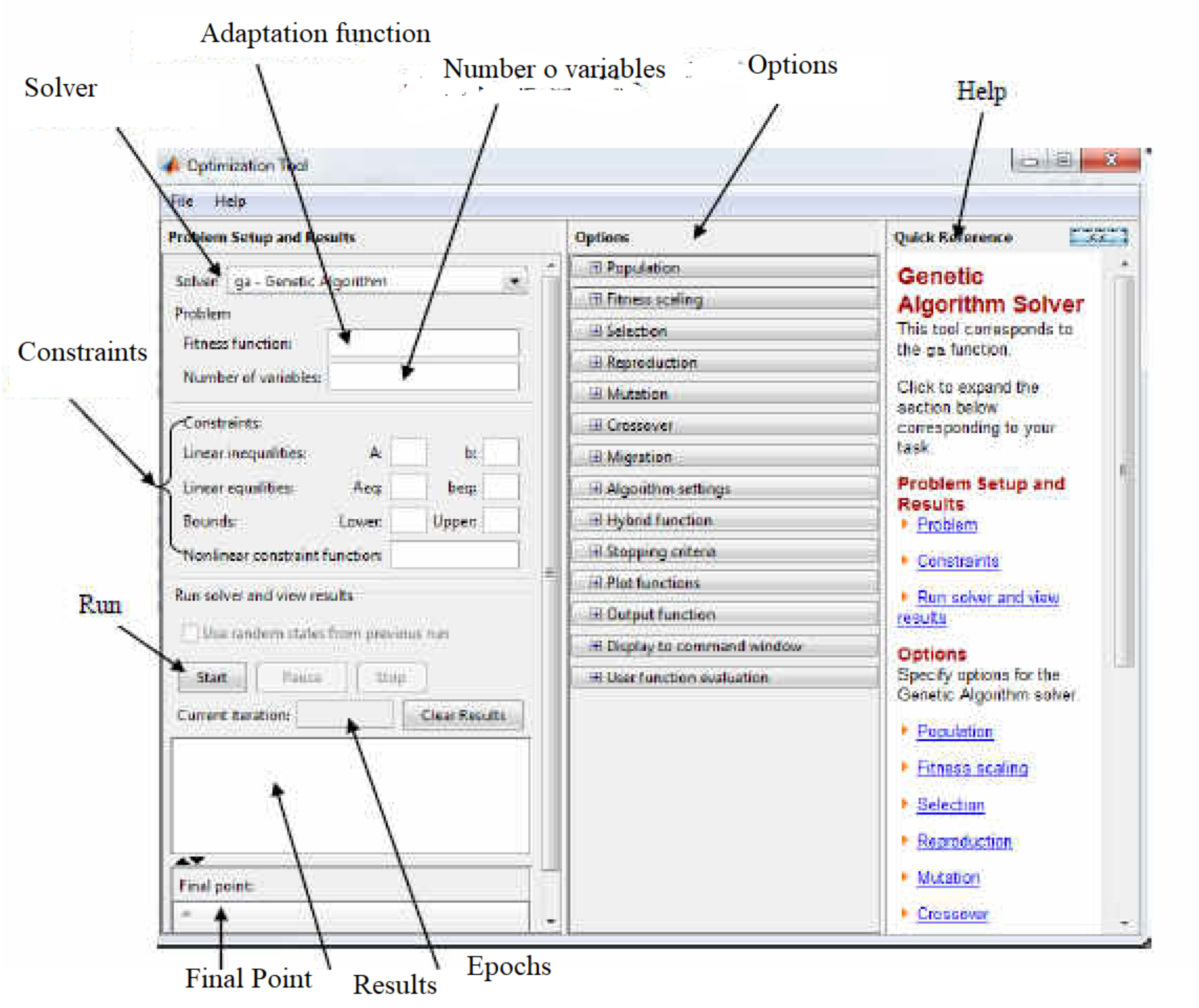
f. Applications in Engineering and Control

In MATLAB, genetic algorithms are widely used for:

- Parameter tuning of controllers (PID, fuzzy, and neural controllers),
- System identification,
- Feature selection and model optimization,
- Multi-objective optimization.

The combination of MATLAB's numerical capabilities and genetic algorithms makes it a highly effective platform for solving nonlinear, non-convex, and multimodal optimization problems.

Using genetic algorithms in MATLAB provides a flexible and efficient approach to solving complex optimization problems. The availability of high-level functions, graphical tools, and extensive documentation makes MATLAB an ideal environment for both research and teaching in evolutionary computation and intelligent systems.



Genetic Algorithm Solver

This tool corresponds to the ga function.

Click to expand the section below corresponding to your task.

Problem Setup and Results

- Problem
- Constraints
- Run solver and view results

Options

Specify options for the Genetic Algorithm solver.

- Population
- Fitness scaling
- Selection
- Reproduction
- Mutation
- Crossover

Figure VII. 2 Genetic Algorithm MATLAB Toolbox

Homework

We consider the minimization of a **nonlinear function**:

f = (x(1)-1)^2 + (x(2)-2)^2 + sin(3*x(1))*cos(3*x(2))

This function is **nonlinear and multimodal**, which makes it suitable for genetic algorithms.

- Create a file called fitness_function.m
- Create a script called GA_example.m

Generated Plots When you run the script, MATLAB automatically generates:

Plot 1 – Best Fitness Value vs Generations

- Shows how the best individual improves over generations
- Used to analyse convergence speed

Plot 2 – Average Distance Between Individuals

- Indicates population diversity
- Helps detect premature convergence

Correction

```
function f = fitness_function(x)
% x(1) = x, x(2) = y
f = (x(1)-1)^2 + (x(2)-2)^2 + sin(3*x(1))*cos(3*x(2));
end
```

```
clc;
clear;
close all;
% Number of variables
nvars = 2;
% Variable bounds
lb = [-5 -5];
ub = [5 5];
% GA options
options = optimoptions('ga', ...
'PopulationSize', 80, ...
'MaxGenerations', 100, ...
'CrossoverFraction', 0.8, ...
'MutationFcn', @mutationgaussian,..
'SelectionFcn', @selectiontournament, ...
'Display', 'iter', ...
'PlotFcn', {@gaplotbestf, @gaplotdistance});
% Run Genetic Algorithm
[x_opt, fval] = ga(@fitness_function, nvars, [], [], [], [], lb, ub, [], options);
% Display results
disp('Optimal solution found:');
disp(x_opt);
disp('Minimum function value:');
disp(fval);
```

```
[X,Y] = meshgrid(-5:0.1:5, -5:0.1:5);

Z = (X-1).^2 + (Y-2).^2 + sin(3*X).*cos(3*Y);

figure;

surf(X,Y,Z)

shading interp

xlabel('x')

ylabel('y')

zlabel('f(x,y)')

title('Objective Function Surface')
```

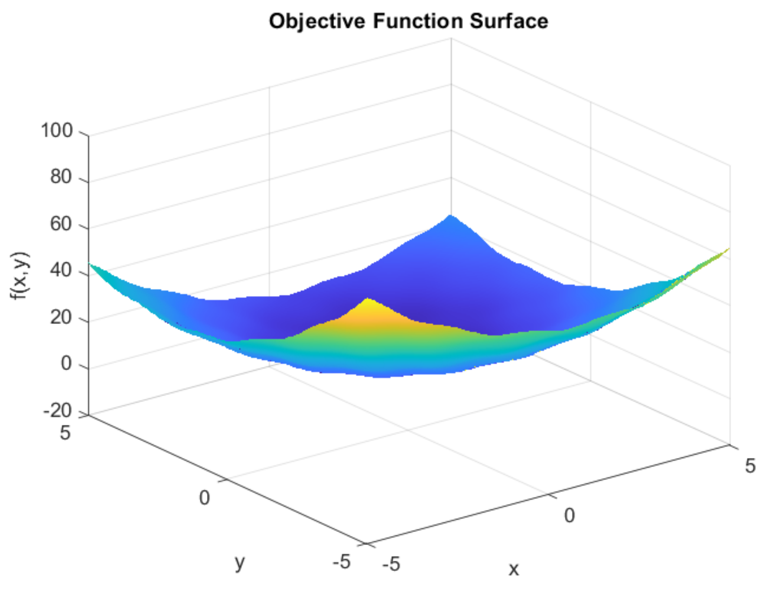


Figure VII. 3 Obtained function surface

Homework

We consider the minimization of a **nonlinear function**:

x.^2 + 10*sin(x)

- Create a file called fitness_function.m
- Create a script called GA_example.m

```
clc; clear; close all;
% Objective function
fitnessFcn = @(x) x.^2 + 10*sin(x);
% Number of variables
nvars = 1;
% Bounds
lb = -10;
ub = 10;
% GA options
options = optimoptions('ga', ...
    'PopulationSize', 50, ...
    'MaxGenerations', 100, ...
    'Display', 'iter', ...
    'PlotFcn', {@gaplotbestf});
% Run GA
[x_opt, fval] = ga(fitnessFcn, nvars, [], [], [], [], lb, ub, [], options);
fprintf('Optimal solution x = %.4f\n', x_opt);
fprintf('Minimum value f(x) = %.4f\n', fval);
```

Homework We consider the minimization of a **nonlinear function**:

- (x(1)-1)^2 + (x(2)-2)^2
- Create a file called fitness_function.m
- Create a script called GA_example.m

```
clc; clear; close all;
fitnessFcn = @(x) (x(1)-1)^2 + (x(2)-2)^2;
nvars = 2;
% Linear inequality: A*x ≤ b
A = [1 1];
b = 2;
lb = [-5 -5];
ub = [5 5];
options = optimoptions('ga', ...
    'PopulationSize', 80, ...
    'MaxGenerations', 100, ...
    'PlotFcn', {@gaplotbestf});
[x_opt, fval] = ga(fitnessFcn, nvars, A, b, [], [], lb, ub, [], options);
disp('Optimal solution:');
disp(x_opt);
```

7 7. References

[1] . The MathWorks*Global Optimization Toolbox™ User's Guide*,MathWorks Inc

[2] . Holland, J. H.*Adaptation in Natural and Artificial Systems*, University of Michigan Press, 1975

[3] . Goldberg, D. E.*Genetic Algorithms in Search, Optimization, and Machine Learning*,Addison-Wesley, 1989.

[4] . Mitchell, M.*An Introduction to Genetic Algorithms*, MIT Press, 1998.

[5] . Goldberg, D. E. & Holland, J. H.*Genetic Algorithms and Machine Learning*, *Machine Learning*, 1988.

8 1. Introduction to Particle Swarm Optimization

Particle Swarm Optimization (PSO) is a population-based stochastic optimization technique inspired by the collective behavior of social organisms, such as bird flocks, fish schools, or insect swarms. It was first introduced in 1995 by James Kennedy and Russell Eberhart as an alternative to evolutionary algorithms, particularly genetic algorithms (GAs).

PSO belongs to the broader class of swarm intelligence methods, which emphasize distributed problem solving, self-organization, and cooperation among simple agents. Unlike genetic algorithms, PSO does not rely on genetic operators such as crossover or mutation. Instead, candidate solutions—called *particles*—move through the search space by adjusting their trajectories based on their own experience and that of neighboring particles.

PSO has gained widespread popularity due to its:

- Conceptual simplicity,
- Ease of implementation,
- Low number of tuning parameters,
- Strong performance on nonlinear, multimodal, and high-dimensional optimization problems.

Particle Swarm Optimization is a powerful, flexible, and widely adopted optimization technique inspired by collective intelligence. Its mathematical simplicity, combined with strong empirical performance, makes PSO a preferred choice for solving complex optimization problems in control systems, artificial intelligence, and engineering design.

Modern research continues to enhance PSO through hybridization, adaptive parameter control, and integration with fuzzy logic, neural networks, and evolutionary strategies.

8 2. Optimizations Problem Formulation

PSO is designed to solve general optimization problems of the min or the max of a

Constraints may be:

- bound constraints,
- linear or nonlinear constraints,
- equality or inequality constraints.

PSO is particularly effective when:

- the objective function is nonlinear or non-convex,
- gradients are unavailable or expensive to compute,
- multiple local optima exist.

8 3. Biological Inspiration and Swarm Intelligence

The fundamental idea of PSO is derived from social sharing of information. In natural swarms: each individual has limited intelligence, global behavior emerges from local interactions, individuals adjust their motion based on neighbors and past experiences.

In PSO:

- each particle represents a potential solution,
- particles communicate indirectly through the global or local best solutions,
- the swarm collectively explores the search space.

The optimization process balances:

- exploration (searching new regions),
- exploitation (refining known good solutions).

8 4. Particle Representation and Swarm Structure

Each particle i is characterized by

- a position vector:
- a velocity vector:

At each iteration

particles move through the search space according to velocity update equations.

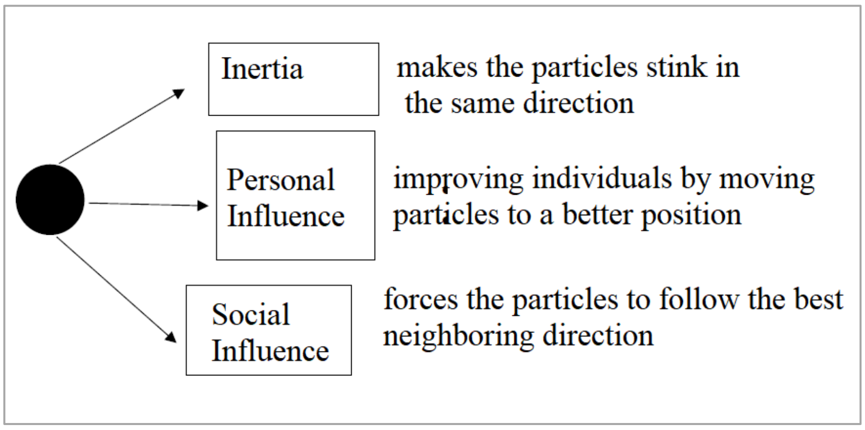

Inertia
makes the particles stink in the same direction
Personal Influence
improving individuals by moving particles to a better position
Social Influence
forces the particles to follow the best neighboring direction

Figure VIII. 1 Diagram explaining the calculation of particle velocity

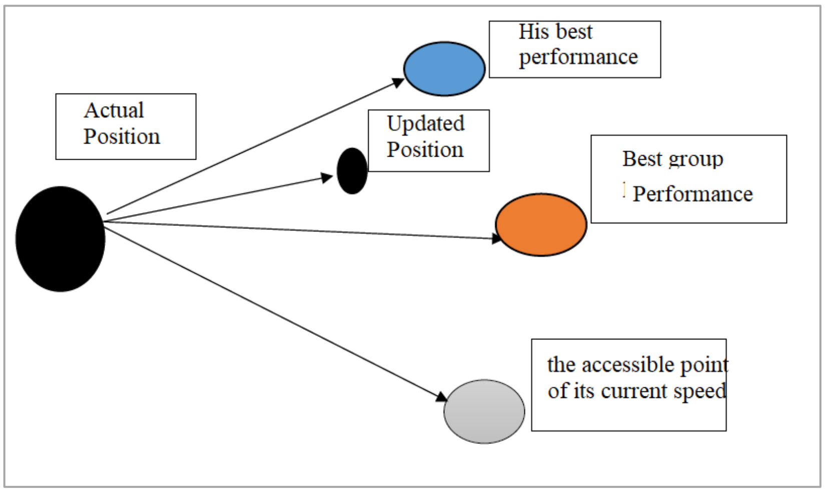


Figure VIII. 2 Particle movement.

8 5. . PSO Algorithm (Step-by-Step)

1. Initialize particle positions and velocities randomly
2. Evaluate fitness of each particle
3. Update personal bests
4. Update global (or local) best
5. Update velocities
6. Update positions
7. Check stopping criteria
8. Repeat until convergence

8 6. Applications of PSO in Engineering

Control Systems

- PID tuning
- Fuzzy controller optimization

Power Systems

- Economic dispatch
- Load frequency control
- Renewable energy optimization

Machine Learning

- Neural network training
- Feature selection
- Hyperparameter optimization

Robotics

- Path planning
- Trajectory optimization

8 7. Advantages and Limitations

Advantages

- Gradient-free
- Easy to implement
- Fast convergence
- Scalable

Limitations

- Premature convergence
- Sensitive to parameter choice
- Performance degrades in very high dimensions

[6] . Homework

Problem Definition

We consider the minimization of a **nonlinear, multimodal function**:

f(x)=x2+10sin (x)f(x) = x^2 + 10\sin(x)f(x)=x2+10sin(x)

This function contains **multiple local minima**, which makes it well suited to illustrate the effectiveness of PSO.

1. T PSO algorithm successfully avoids local minima.
2. The convergence plot (pswplotbestf) shows the best fitness value decreasing iteratively.
3. The final solution corresponds to the global minimum or a very close approximation.

---

2. Mathematical Formulation

min f(x)x ∈ [−10, 10] f(x)\min{x \in [−10,\,10]} \; f(x)x ∈ [−10,10]minf(x)

where:

- x is the decision variable,
- the objective function is **non-convex** and **nonlinear**.

```
clc;
clear;
close all;
% Objective function
fitnessFcn = @(x) x.^2 + 10*sin(x);
% Number of variables
nvars = 1;
% Lower and upper bounds
lb = -10;
ub = 10;
% PSO options
options = optimoptions('particleswarm', ...
    'SwarmSize', 50, ...
    'MaxIterations', 100, ...
    'Display', 'iter', ...
    'PlotFcn', {@pswplotbestf});
% Run PSO
[x_opt, fval] = particleswarm(fitnessFcn, nvars, lb, ub, options);
fprintf('Optimal solution x = %.4f\n', x_opt);
fprintf('Minimum value f(x) = %.4f\n', fval);
```

```
% Plot objective function
x = linspace(-10, 10, 1000);
y = x.^2 + 10*sin(x);
figure;
plot(x, y, 'LineWidth', 2);
hold on;
plot(x_opt, fval, 'ro', 'MarkerSize', 10, 'LineWidth', 2);
grid on;
xlabel('x');
ylabel('f(x)');
title('PSO Optimisation Result');
legend('Objective Function', 'PSO Solution');
```

8 1. References

[1] . Kennedy, J., & Eberhart, R., *Particle Swarm Optimization*, IEEE ICNN, 1995

[2] . Clerc, M., *Particle Swarm Optimization*, ISTE Press

[3] . MathWorks, *Global Optimization Toolbox – particle swarm*

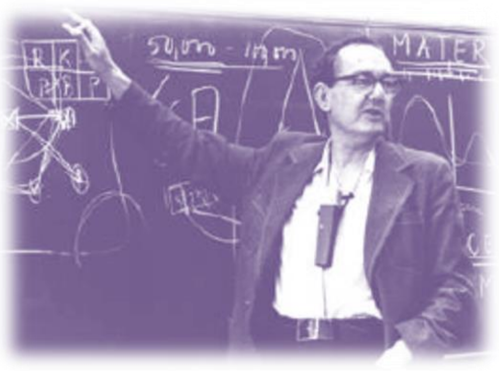
ANNEX

Norbert Wiener (1894-1964), A mathematician, Wiener launched cybernetics in the 1940s as the science of the functioning of the human mind. He wanted to model the mind as a "black box" with behavior dependent on feedback mechanisms.

Warren McCulloch (1898-1969), et **Walter Pitts (1923-1969),** In 1943, neurologists invented the first mathematical model of the biological neuron, the formal neuron.

Donald Hebb (1904-1985), neuropsychologist, invented in 1949 the rule which bears his name and makes it possible to endow formal neurons with learning capacities:
When an axon of cell A is near enough to excite B and repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A’s efficiency, as one of the cells firing B, is increased (The Organization of Behavior, 1949)

Herbert Simon (1916-2001) He studied economics in Chicago and earned his doctorate in political science there in 1943. Simon sought to understand how individuals make decisions, particularly within organizations. Simon introduced the concept of bounded rationality: an individual's rationality is limited by their environment, their history, and the lack of information necessary for decision-making (Administrative Behavior, 1947).



Annex

Allen Newell (1927-1992), after a Bachelor of Science in physics at Stanford, joined Princeton in 1949 to pursue a PhD in mathematics. During his studies, he was strongly influenced by the Hungarian mathematician Georges Polya (1887-1985), who had introduced the notion of heuristics for problem solving (How to solve it, 1945).

A heuristic (from the Greek eurisko, I believe) is an empirical method of problem-solving whose validity or effectiveness is not proven. Examples: protecting the queen in chess, choosing the crate with the shortest line, etc. Ultimately finding mathematics too abstract, Newell accepted a position in 1950 at the RAND Corporation in Santa Monica to conduct more practical work, particularly in defense aeronautics.

Simon is also a consultant at RAND (Research and Development), a non-profit research organization linked to the American military-industrial complex. RAND, created to study the development of an artificial satellite, gradually expanded its work to include computer science, economics, geopolitics, and other fields.

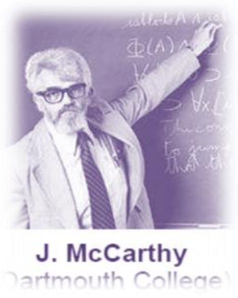
Simon's and Newell's ideas converge: Simon's bounded rationality implies that decision-making relies on procedures that compensate for information gaps by taking the context into account. For Newell, these procedures are heuristics.

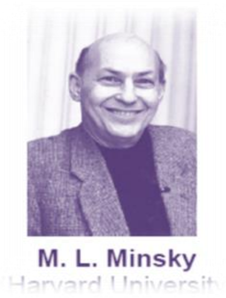
Simon and Newell considered that the necessary and sufficient condition for a machine to demonstrate intelligence was that it be a physical system of symbols. However, they placed the notion of heuristics at the heart of their work: they developed Logic Theorist in 1956, a program for the automatic proof of theorems.

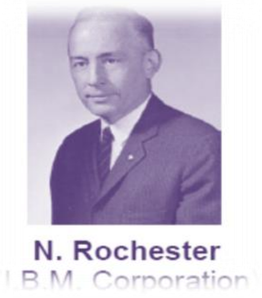
McCarthy and Minsky enlisted the help of their elders Shannon and Rochester to secure $7,500 from the Rockefeller Foundation to organize a workshop on thinking machines at Dartmouth College during the two summer months of 1956.

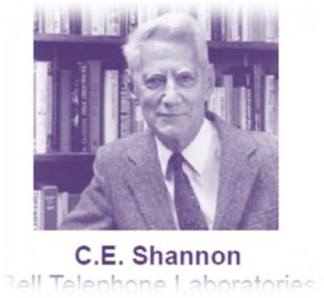
Annex

- **Marvin Minsky** (born in 1927) was at Harvard University in 1956, where he was a lecturer. After studying mathematics at Harvard (Bachelor of Arts in 1950), where he met the early cyberneticists, he completed a dissertation at Princeton entitled "Neural Nets and the Brain Model Problem" (1954). His dissertation focused on the creation of an artificial neural network, in collaboration with a doctoral student in electronics, Dean Edmonds.
- **John McCarthy (1927-2011)** earned a Bachelor of Science in mathematics from the California Institute of Technology in 1948, followed by a doctorate from Princeton in 1951. His dissertation focused on a type of partial derivative equation, but his time at Princeton led him to meet Minsky, with whom he discovered a shared passion for the idea of the thinking machine.
- **Claude Shannon (1916-2001)** earned a Bachelor of Science in electrical engineering and another in mathematics from the University of Michigan in 1936, and then continued his studies at the Massachusetts Institute of Technology. In his master's thesis, he established the link between Boolean algebra and electrical circuits (A Symbolic Analysis of Relay and Switching Circuits, 1937), thus laying the foundations of digital electronics. During his doctoral studies, he worked on the mathematical formalization of genetics (An Algebra for Theoretical Genetics, 1940).
- **Nathaniel Rochester (1914-2001)** He obtained a Bachelor of Science in electrical engineering from MIT in 1941 and joined International Business Machines (IBM) in 1948. There he developed the first symbolic assembly language. Rochester would become chief engineer for all the IBM 7xx series, in particular the IBM 701, the first general-purpose computer produced in series starting in 1953 (19 units were manufactured).









Citations

- Intelligence begins neither with self-knowledge, nor with knowledge of things as such, but with knowledge of their interaction. It organizes the world by organizing itself. Jean Piaget (1896-1980), in *The Construction of Reality*, 1936

- Intelligence is the ability to understand the relationships that exist between the elements of a situation and to adapt to them in order to achieve one's own ends. It is always understanding and invention. Gaston Viaud (1899-1961), French psychologist, in *Intelligence, Its Evolution and Forms*, 1946.

- Intelligence is everything that allows us to intuit a new underlying order. Horace Barlow (1921-1987), Professor of Physiology and neurobiologist, University of Cambridge.

- The set of mental functions whose object is conceptual and rational knowledge (as opposed to sensation and intuition). The ability to understand and adapt easily to new situations. From Terminology of Neuropsychology and Behavioral Neurology. Research and editing by Louise Bérubé, c1991, 176 p.

- Intelligence is the capacity to discover a new context, understand it, and react to this new situation appropriately. Richard Atkinson (1920–1994), British prehistorian and archaeologist.