Table of Contents

Abstract **3**
Introduction to Fuzzy Logic **3**
Evolution of Fuzzy Logic Control **4**
Fuzzy Logic Principles **5**
Fuzzy Logic Control Components **7**
Fuzzification ***7***
Fuzzy Processing ***9***
Defuzzification ***12***
Fuzzy Logic Application Example **14**
System Description and Operation ***14***
Membership Functions and Rule Creation ***15***
Guidelines for Controlling Complex Systems **16**
Control Objectives ***16***
Control System Configuration ***17***
Input/Output Determination ***17***
Fuzzy Inference Engine ***17***
Conclusion **19**

PLCs and Fuzzy Logic

Industrial Text and Video Company

Abstract

Fuzzy logic provides programmable logic controllers (PLCs) with the ability to make "reasoned" decisions about a process, thereby allowing them to make autonomous output calculations based on system inputs. Fuzzy logic systems use a three-step process—fuzzification, fuzzy processing, and defuzzification—to evaluate system parameters. In fuzzification, the system translates inputs into sets of data defined by membership functions and labels. During fuzzy processing, the system analyzes the input sets by comparing them against a set of predetermined rules. During defuzzification, the system translates the fuzzy processing data into a control system output. Thus, by using this three-step process, a PLC with fuzzy logic capabilities can judiciously determine the proper control response to system inputs based on predefined criteria.

INTRODUCTION TO FUZZY LOGIC

Fuzzy logic is the branch of artificial intelligence that deals with the reasoning algorithms used to emulate human thinking and decision making in machines. These algorithms are used in applications where process data cannot be represented in binary form. For example, the statements "the air feels cool" and "he is young" are not discrete statements. They do not provide concrete data about the air temperature or the person's age (i.e., the air is at 65°F or the boy is 12 years old). Fuzzy logic interprets vague statements like these so that they make logical sense. In the case of the cool air, a PLC with fuzzy logic capabilities would interpret both the level of coolness and its relationship to warmth to ascertain that "cool" means somewhere between hot and cold. In straight binary logic, hot would be one discrete value (e.g., logic 1) and cold would be the other (e.g., logic 0), leaving no value to represent a cool temperature.

In contrast to binary logic, fuzzy logic can be thought of as *gray logic*, which creates a way to express in-between data values. Fuzzy logic associates a *grade*, or *level*, with a data range, giving it a value of 1 at its maximum and 0 at its minimum. For example, Figure 1a illustrates a fuzzy logic representation of a cool air temperature range, where 70°F indicates perfectly cool air (i.e., a grade value of 1). Any temperature over 80°F is considered hot, and any temperature below 60°F is considered cold. Thus, temperatures above 80°F and below 60°F have values of 0 cool, meaning they are not cool at all. Figure 1b shows another representation of the cool temperature range, where the dotted line shows that hot and cold temperatures are not cool. At 65°F, the fuzzy logic algorithm considers the temperature to be 50% cool and 50% cold, indicating a level of coolness. Below 60°F, the fuzzy logic algorithm considers the temperature to be cold.

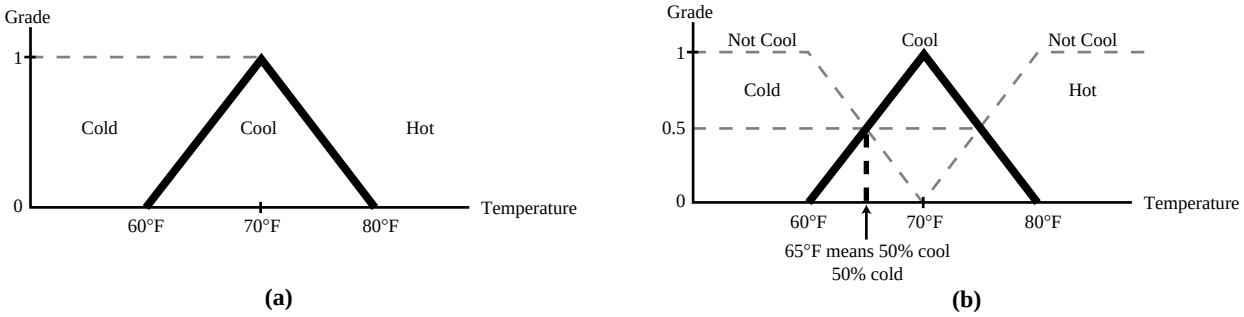


Figure 1. (a) Cool air temperature range with **(b)** dotted lines showing not cool range.

Fuzzy logic requires knowledge in order to reason. This knowledge, which is provided by an expert who knows the process or machine, is stored in the fuzzy system. For example, an expert may specify that a steam valve should be turned clockwise a "little bit" if the temperature rises in a batching operation. The fuzzy system might interpret this expression as a 10-degree clockwise rotation that closes the output valve opening by 5%. As the name implies, a description like a "little bit" is a fuzzy description, meaning that it does not have a definite value. A fuzzy logic system takes this vague description and translates it into a decisive output.

EVOLUTION OF FUZZY LOGIC CONTROL

Fuzzy logic has existed since ancient times, when Aristotle developed the law of the excluded middle. In this law, Aristotle pointed out that the middle ground is lost in the art of logical reasoning—statements are either true or false, never in-between. When PLCs were developed, their discrete logic was based on the ancient reasoning techniques. Thus, inputs and outputs could belong to only one set (i.e., ON or OFF); all other values were excluded. Fuzzy logic breaks the law of the excluded middle in PLCs by allowing elements to belong to more than just one set. In the previous cool air example, the 65°F temperature input belonged to two sets, the cool set and the cold set, with grade levels indicating how well it fit into each set.

The origins of fuzzy logic date back to the early part of the twentieth century when Bertrand Russell discovered an ancient Greek paradox that states:

> A Cretan asserts that all Cretans lie. So, is he lying? If he lies, then he is telling the truth and does not lie. If he does not lie, then he tells the truth and, therefore, he lies.

In either case—that all Cretans lie or that all Cretans do not lie—a contradiction exists, because both statements are true and false. Russell found that this same paradox applied to the set theory used in discrete logic. Statements must either be totally true or totally false, leading to areas of contradiction.

Fuzzy logic surmounted this problem in classical logic by allowing statements to be interpreted as both true and false. Therefore, applying fuzzy logic to the Greek paradox yields a statement that is both true and false: Cretans tell the truth 50% of the time and lie 50% of the time. This interpretation is very similar to the idea of a glass of water being half empty or half full. In fuzzy logic, the glass is both—50% full and 50% empty. Even as the amount of water decreases, the glass still retains percentages of both conditions.

Around the 1920s, independent of Bertrand Russell, a Polish logician named Jan Lukasiewicz started working on multivalued logic, which created fractional binary values between logic 1 and logic 0. In a 1937 article in *Philosophy of Science*, Max Black, a quantum philosopher, applied this multivalued logic to lists (or sets) and drew the first set of fuzzy curves, calling them *vague sets*. Twenty-eight years later, Dr. Lofti Zadeh, the Electrical Engineering Department Chair at the University of California at Berkeley, published a landmark paper entitled "Fuzzy Sets," which gave the name to the field of fuzzy logic. In this paper, Dr. Zadeh applied Lukasiewicz's logic to all objects in a set and worked out a complete algebra for fuzzy sets. Due to this groundbreaking work, Dr. Zadeh is considered to be the father of modern fuzzy logic.

Around 1975, Ebrahim Mamdani and S. Assilian of the Queen Mary College of the University of London published a paper entitled "An Experiment in Linguistic Synthesis with a Fuzzy Logic Controller." This paper proved the feasibility of fuzzy logic control applying fuzzy control to a steam engine. Since then, the term *fuzzy logic* has come to mean mathematical or computational reasoning that utilizes fuzzy sets.

FUZZY LOGIC PRINCIPLES

Figure 2 illustrates the operation of a fuzzy logic control system. The input to the fuzzy system is the output of the process, which is entered into the system via input interfaces. For example, in a temperature control application, the input data would be entered into the fuzzy controller using an analog input module. This input information would then go through the fuzzy logic process, where the processor would compare the input data to a database to obtain an output. Fuzzy logic processing involves the execution of IF...THEN rules, which are based on the input conditions. An input's *grade* specifies how well it fits into a particular graphic input data set. Input data may be represented in a variety of forms, including count value and percentage of error deviation. Thus, if a fuzzy logic system uses an analog input that has a count range from 0 to 4095, the graphs representing the input sets will cover a span from 0 to 4095 counts.

The output of a fuzzy controller is also defined by grades, with the grade determining the appropriate output value for the control element. For example, the output of a fuzzy system could control an air conditioner, which runs faster or slower according to the output's grade on the output chart.

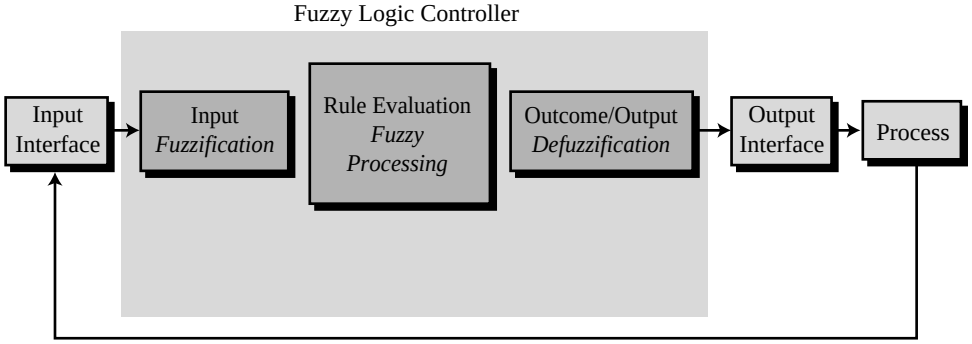


Figure 2. Fuzzy logic control system.

Figure 3 illustrates the fuzzy logic output chart for this cooling system. The chart has both input and output grades, with the horizontal axis being the input condition (temperature) and the vertical axis being the output (air conditioner motor speed). In this chart, a single input can trigger more than one output condition. For example, if the input temperature is 137.5°F, then the temperature is part of two input sets—it is 50% too cool and 50% normal. Consequently, the input will trigger two outputs—the too cool input condition will trigger a less speed output, while the normal input will trigger a normal speed output. Since the fuzzy logic controller can have only one output, it must complete a process called *defuzzification* to determine the actual final output value.

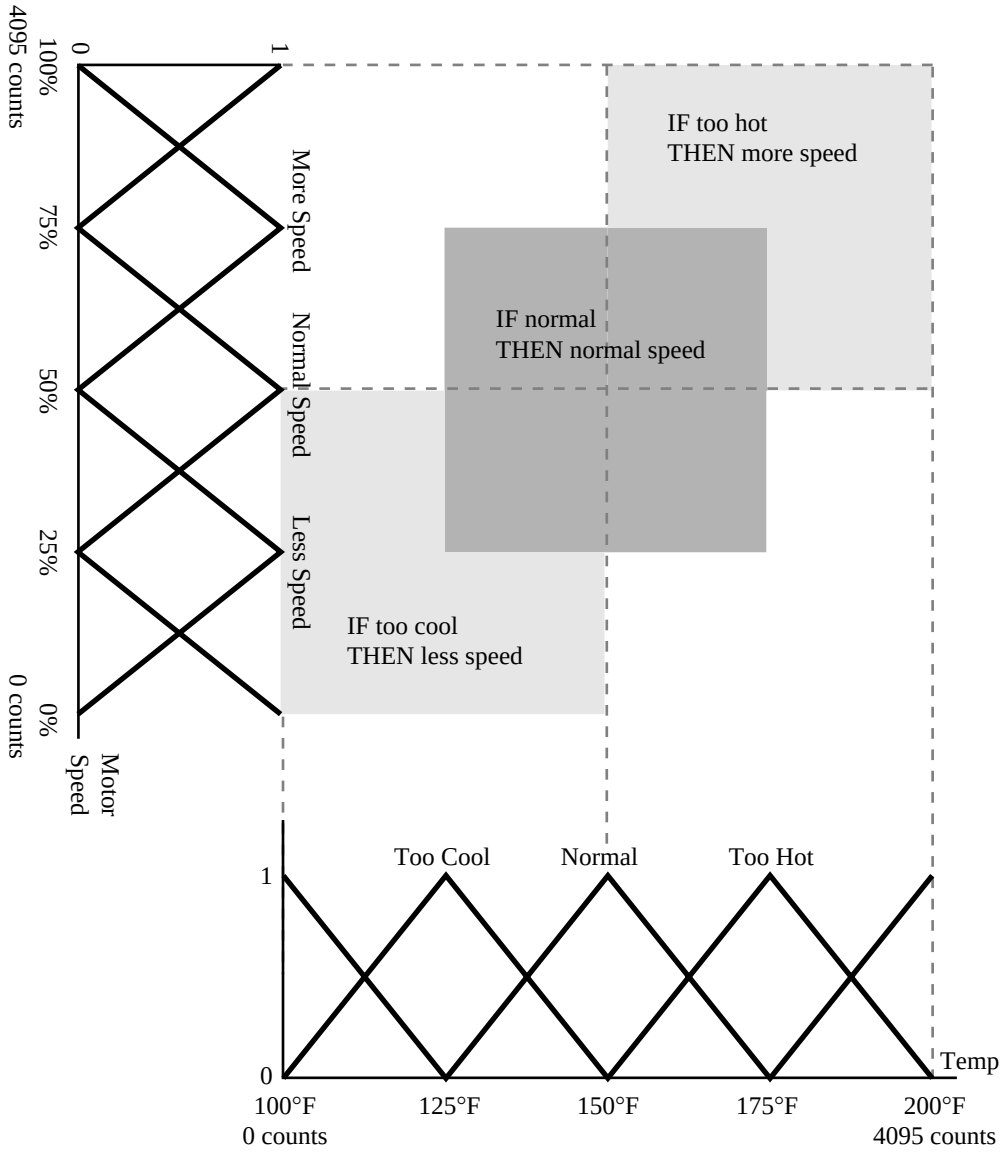


Figure 3. Fuzzy logic output chart.

The implementation of a fuzzy logic control system is similar to the implementation of PID control using intelligent interfaces, where the module reads the input, processes the information, and provides an output. However, fuzzy controllers are usually independent interfaces. They typically plug into the PLC rack and use the PLC's I/O system to communicate with the process that is being controlled by fuzzy logic.

FUZZY LOGIC CONTROL COMPONENTS

A fuzzy logic controller's control process consists of three main components, or actions, that must be completed sequentially to determine the appropriate output value. These components are:

- fuzzification
- fuzzy processing
- defuzzification

As shown previously in Figure 2, when a fuzzy controller receives input data, it translates it into a fuzzy form. This process is called *fuzzification*. The controller then performs *fuzzy processing*, which involves the evaluation of the input information according to IF…THEN rules created by the user during the fuzzy control system's design stage. Once the fuzzy controller finishes the rule-processing stage and arrives at an outcome conclusion, it begins the *defuzzification* process. In this final step, the fuzzy controller converts the output conclusions into "real" output data (e.g., analog counts) and sends this data to the process via an output module interface. If the fuzzy logic controller is located in the PLC rack and does not have a direct or built-in I/O interface with the process, then it will send the defuzzification output to the PLC memory location that maps the process's output interface module.

Fuzzification

The fuzzification process is the interpretation of input data by the fuzzy controller. Fuzzification consists of two main components:

- membership functions
- labels

Membership Functions. During fuzzification, a fuzzy logic controller receives input data, also known as the *fuzzy variable*, and analyzes it according to user-defined charts called *membership functions* (see Figure 4). Membership functions group input data into sets, such as temperatures that are too cold, motor speeds that are acceptable, etc. The controller assigns the input data a grade from 0 to 1 based on how well it fits into each membership function (e.g., 0.45 too cold, 0.7 acceptable speed). Membership functions can have many shapes, depending on the data set, but the most common are the S, Z, Λ, and Π shapes shown in Figure 5. Membership functions consist of connected line segments defined by the lines' end points. Each membership function can have up to three line segments with a maximum of four end points. The grade at each end point must have a value of 0 or 1. A membership function's shape does not have to be symmetrical; however, it must comply with the previously discussed specifications.

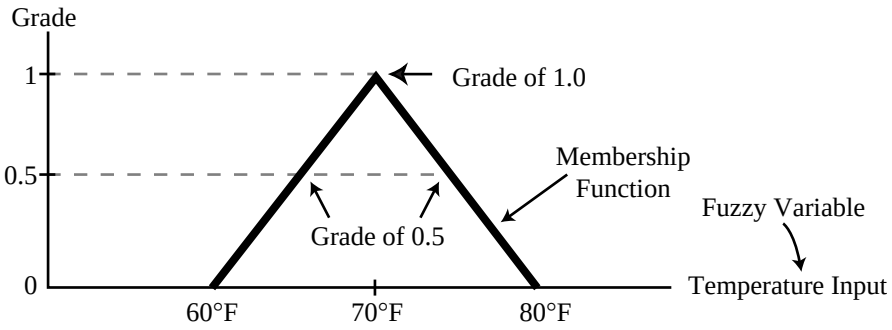


Figure 4. Membership function chart.

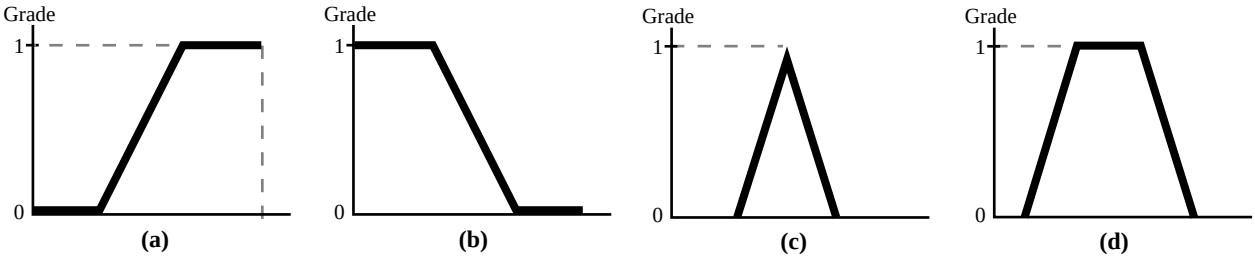


Figure 5. Membership function shapes: **(a)** S, **(b)** Z, **(c)** Λ, and **(d)** Π.

Labels. Each fuzzy controller input can have several membership functions, with seven being the norm, that define its conditions. Each membership function is defined by a name called a *label*. For example, an input variable such as temperature might have five membership functions labeled as cold, cool, normal, warm, and hot. Generically, the seven membership functions have the following labels, which span from the data range's minimum point (negative large) to its maximum point (positive large):

- NL (negative large)
- NM (negative medium)
- NS (negative small)
- ZR (zero)
- PS (positive small)
- PM (positive medium)
- PL (positive large)

Figure 6 shows an example of an input variable with seven Λ-shaped membership functions using all of the possible labels. A group of membership functions forms a *fuzzy set*. Although most fuzzy sets have an odd number of labels, a set can also have an even number of labels. For example, a fuzzy set may have four or six labels in any shape, depending on how the inputs are defined in relationship to the membership function.

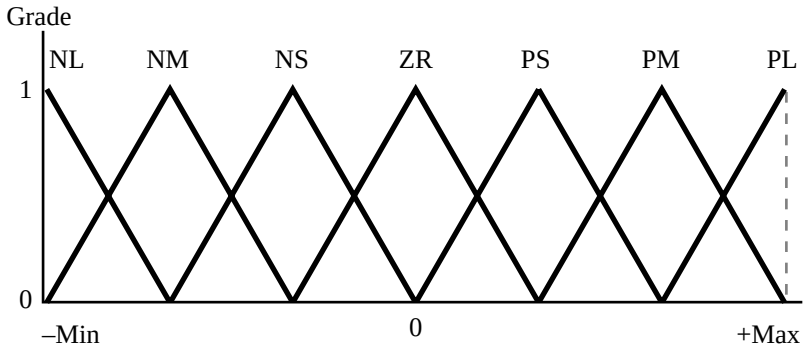


Figure 6. Fuzzy logic input with seven membership functions.

Fuzzy Processing

During fuzzy processing, the controller analyzes the input data, as defined by the membership functions, to arrive at a control output. During this stage, the processor performs two actions:

- rule evaluation
- fuzzy outcome calculation

Rule Evaluation. Fuzzy logic is based on the concept that most complicated problems are formed by a collection of simple problems and can, therefore, be easily solved. Fuzzy logic uses a reasoning, or *inferencing*, process composed of IF...THEN rules, each providing a response or outcome. Basically, a rule is activated, or *triggered*, if an input condition satisfies the IF part of the rule statement. The triggering of the IF part of the rule results in a control output based on the THEN part of the rule statement. In a fuzzy logic system, many rules may exist, corresponding to one or more IF conditions. A rule may also have several input conditions, which are logically linked in either an AND or an OR relationship to trigger the rule's outcome. Different fuzzy logic controllers have different rule evaluation capabilities. Some can process more rules, inputs, and outputs than others.

Sometimes, more than one rule is triggered at a time in a fuzzy control process. In this case, the controller evaluates all the rules to arrive at a single outcome value and then proceeds to the defuzzification process. For instance, if two inputs are logically ANDed or ORed in several rules, then they will produce several outcomes, of which only one will be logically added to determine the final outcome. Figure 7a illustrates an example of two fuzzy inputs, X_1 and X_2, and one fuzzy output, Y_1. The rules shown in Figure 7b represent four of nine possible rules that cover the two inputs. The four shown, however, cover the four possible triggering points for the two input value readings, X_1 and X_2. Given the input values in Figure 7a, the inputs will trigger rule 1 because X_1 = ZR AND X_2 = NL. This will generate two outputs for Y_1 = NL. One output will be at a grade of 0.6 (due to the ZR input value of X_1), and the other will be at a grade of 0.75 (due to the NL input value of X_2). In a fuzzy logic situation where a two-input rule with an AND relationship produces two outcome values, the controller will choose the outcome with the *smallest* grade, in this case 0.6NL. If the rule utilizes OR logic, the chosen outcome will be the one with the *largest* grade. If rule 1 had used an OR function instead of an AND function, then the controller would have selected the Y_1 = 0.75NL outcome, the largest of the two outcomes.

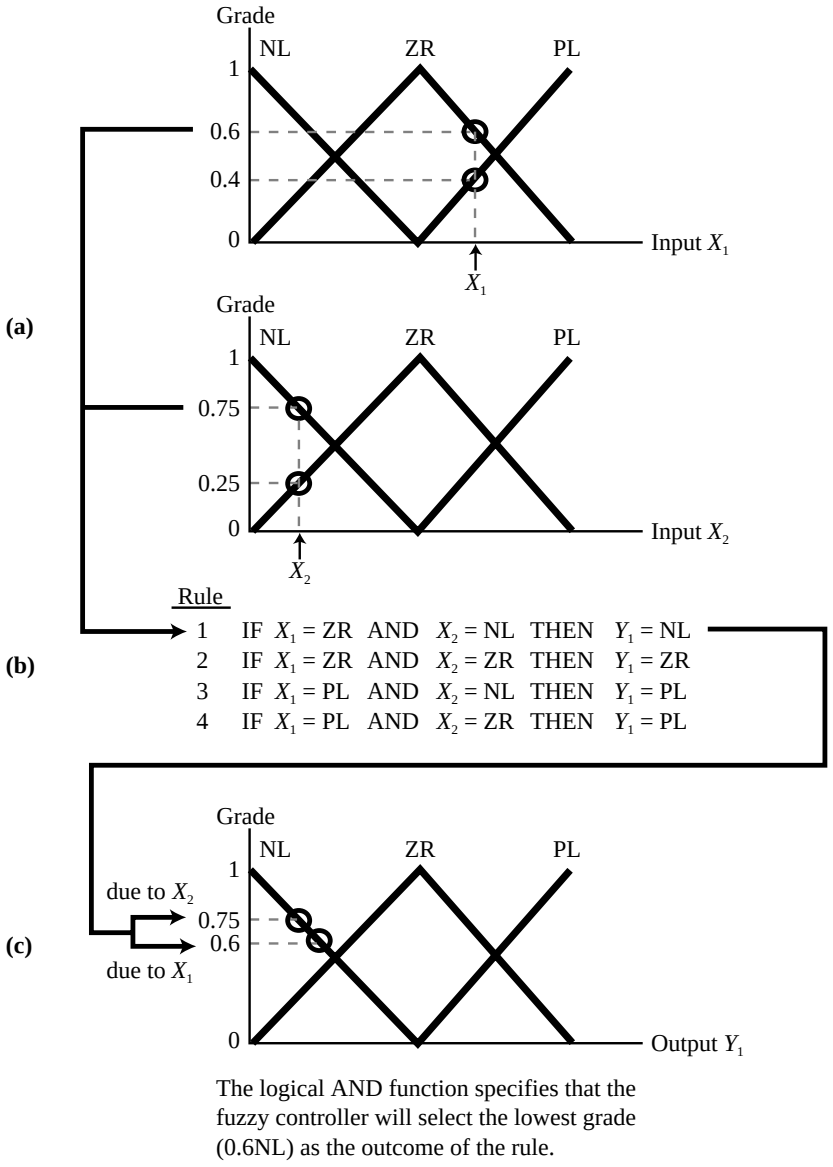
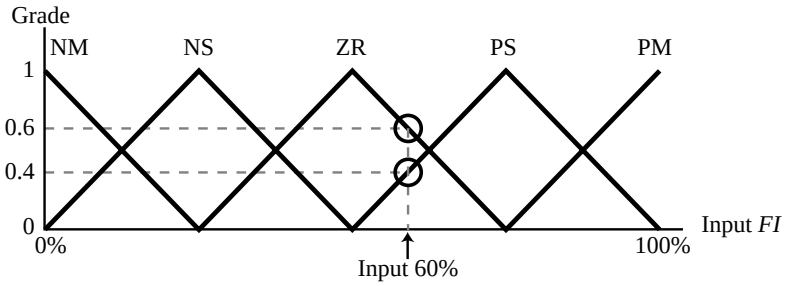


Figure 7. Fuzzy processing example showing **(a)** two fuzzy input values, **(b)** the four rules that they trigger, and **(c)** the resulting output.

Fuzzy Outcome Calculations. Once a rule is triggered, meaning that the input data belongs to a membership function that satisfies the rule's IF statement, the rule will generate an output outcome. This fuzzy output is composed of one or more output membership functions (with labels) that have grades associated with them. The outcome's membership function grade is affected by the grade level of the input data in its input membership function. However, which output membership function grade the fuzzy controller selects for the final output value depends on the user's programming of the IF...THEN rules.

For example, in Figure 8, the 60% input value (fuzzy input *FI*) triggers rules 3 and 4 because that value of *FI* belongs to both membership functions ZR and PS. Rules 3 and 4 indicate that both fuzzy output action ZR and action PL must be applied to the process. These output actions will be

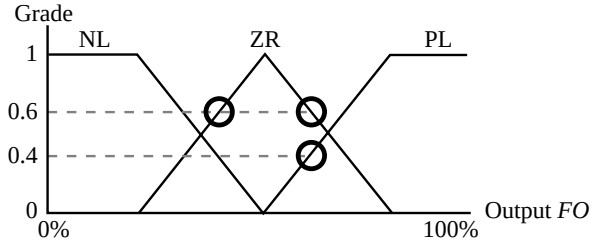
(a)



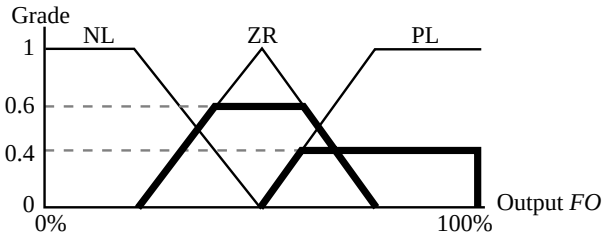
(b)

Rule			Rules Triggered
1	IF *FI* = NM	THEN *FO* = NL	
2	IF *FI* = NS	THEN *FO* = ZR	
3	IF *FI* = ZR	THEN *FO* = ZR	✓
4	IF *FI* = PS	THEN *FO* = PL	✓
5	IF *FI* = PM	THEN *FO* = PL	

(c)

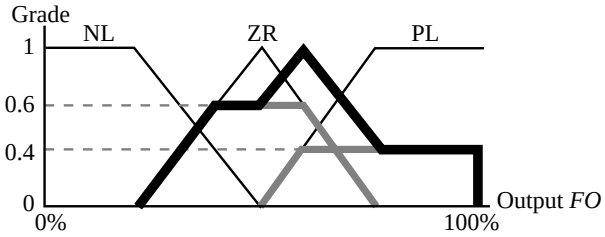


(d)



Output 1 is 60% (or 0.6) of the ZR output.
Output 2 is 40% (or 0.4) of the PL output.

(e)



The result of adding outputs 1 and 2 is a
new curve that defines the final output.

(f)

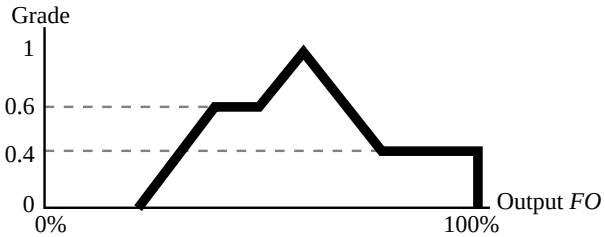


Figure 8. Fuzzy logic process: **(a)** inputs, **(b)** rules, **(c)** outputs, **(d)** output curves, **(e)** combined output curve, and **(f)** the output signal for the field device.

applied at a value that corresponds to the grades generated in the input membership functions (i.e., output 0.6ZR and 0.4PL). Note that the 0.6 grade is applied to output ZR and the 0.4 grade is applied to output PL because the user programmed the rules that way. Figure 8c shows these two output grades for fuzzy output *FO*. To arrive at a final outcome value, the fuzzy logic controller logically adds both fuzzy outcomes to produce an aggregate outcome curve, which is illustrated in Figure 8e. The controller then generates an output signal (during defuzzification) that controls the process's field device (e.g., valve, motor, etc.) according to the input data (see Figure 8f).

A fuzzy logic controller may represent its output membership functions as noncontinuous functions that resemble spikes rather than as continuous geometrical shapes. Each spike has a specific output value associated with it. Figure 9 shows the three output membership functions from Figure 8 represented as noncontinuous spikes. Each membership function corresponds to a particular output count value (i.e., 0 counts, 2048 counts, and 4095 counts). The shaded bars represent the outcome grade levels of 0.6ZR and 0.4PM, which were determined by the rule evaluation.

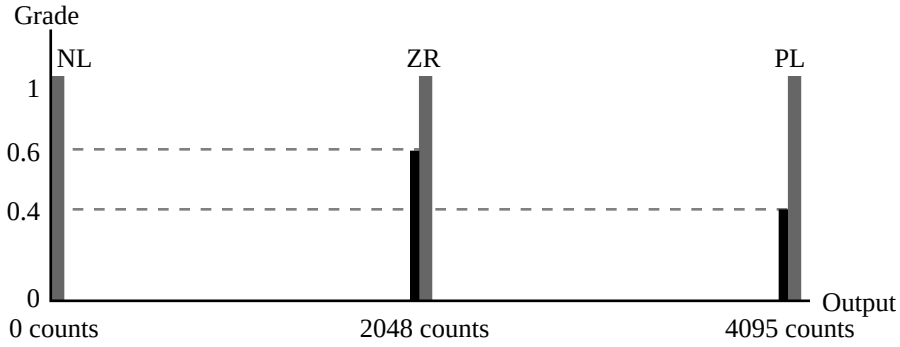


Figure 9. Three output membership functions represented as noncontinuous spikes.

Defuzzification

The final output value from the fuzzy controller depends on the defuzzification method used to compute the outcome values corresponding to each label. The defuzzification process examines all of the rule outcomes after they have been logically added and then computes a value that will be the final output of the fuzzy controller. The PLC then sends this value to the output module. Thus, during defuzzification, the controller converts the fuzzy output into a real-life data value (e.g., 1720 counts).

There are many defuzzification methods, but all are based on mathematical algorithms. The two most common defuzzification methods are:

- maximum value
- center of gravity

Maximum Value Method. The maximum value defuzzification method bases the final fuzzy output value on the rule output with the highest membership function grade. This method is mainly used with discrete output membership functions. Referring to the chart in Figure 9, the maximum value defuzzification method would specify that the output value of 2048 counts be chosen as the final output value because it has the largest grade value. If two or more outcomes from two or more rules

have the same grade level, then the controller will select the final outcome value based on criteria supplied by the user during the fuzzy system programming. Such criteria specifies choosing either the left-most or right-most grade value of the two equal labels and their corresponding number of counts. The left-most criteria selects the lowest output value (the one with the fewest counts), while the right-most criteria selects the highest output value (the one with the most counts). So if both ZR and PL in Figure 9 had output grade values of 0.6, a fuzzy controller using the left-most criteria would select the ZR output value (2048 counts). A controller using the right-most criteria would select the PL output (4095 counts).

Center of Gravity Method. The center of gravity defuzzification method, also referred to as "calculating the centroid," mathematically obtains the center of mass of the triggered output membership functions. In mathematical terms, a centroid is the point in a geometrical figure whose coordinates equal the average of all the other points comprising the figure. This point is the center of gravity of the figure. In simple terms, the center of gravity for a fuzzy output is the output data value (as shown on the X-axis), that divides the area under the fuzzy membership function curve into two equal parts. The center of gravity method is the most commonly used defuzzification method because it provides an accurate result based on the weighted values of several output membership functions. The output value that is sent to the output interface module is the output data value at the intersection of the horizontal axis and the centroid.

The center of gravity method applies to noncontinuous output membership functions as well as continuous ones. In noncontinuous functions, the final output value for a seven-label output membership function (labels A through G) is expressed by the formula:

$$\text{Output data} = \frac{\sum_{n=\text{A}}^{n=\text{G}} \left[\left(FO_n \right) \left(FGrade_n \right) \right]}{\sum_{n=\text{A}}^{n=\text{G}} FGrade_n}$$

where:

Output data = the number of counts to be used for the output
FO = the fuzzy output in counts for labels *n* = A through G
FGrade = the fuzzy grade level for levels *n* = A through G

This equation implies that the final value of the output will be equal to the sum of each rule outcome grade multiplied by its actual output data value (i.e., counts) divided by the sum of the rule outcome grades. Referring to the noncontinuous membership example in Figure 9, the fuzzy logic controller will decide to send an output of 2867 counts to the output interface after completing the center of gravity calculation, which is as follows:

$$\text{Output data} = \frac{\sum_{n=\text{NL}}^{n=\text{PL}} \left[\left(FO_n \right) \left(FGrade_n \right) \right]}{\sum_{n=\text{NL}}^{n=\text{PL}} FGrade_n}$$

$$= \frac{(2048)(0.6) + (4095)(0.4)}{0.6 + 0.4}$$

$$= 2867 \text{ counts}$$

The fuzzy controller's output is more than it would have been using the maximum value method (the maximum output label is ZR, which is 2048 counts). This indicates that the weighted value of the 0.4PL label pulls the value to the right (more counts).

Fuzzy controllers that use continuous membership functions and the center of gravity defuzzification method also use the previous summation equation to approximate the centroid value. However, in this case, the controller uses approximate digitized values for each membership function to compute each of the points in the summation.

FUZZY LOGIC APPLICATION EXAMPLE

Fuzzy logic can be applied to a wide array of applications and industries, given that the application requires reasoned output values derived from system inputs or feedback. Figure 10 shows an example of a typical fuzzy logic application, a two-conveyor packaging system. The objective of this application is to synchronize two conveyors so that parts and packaging boxes are positioned correctly, regardless of the part and packaging box positions and the speed of conveyor.

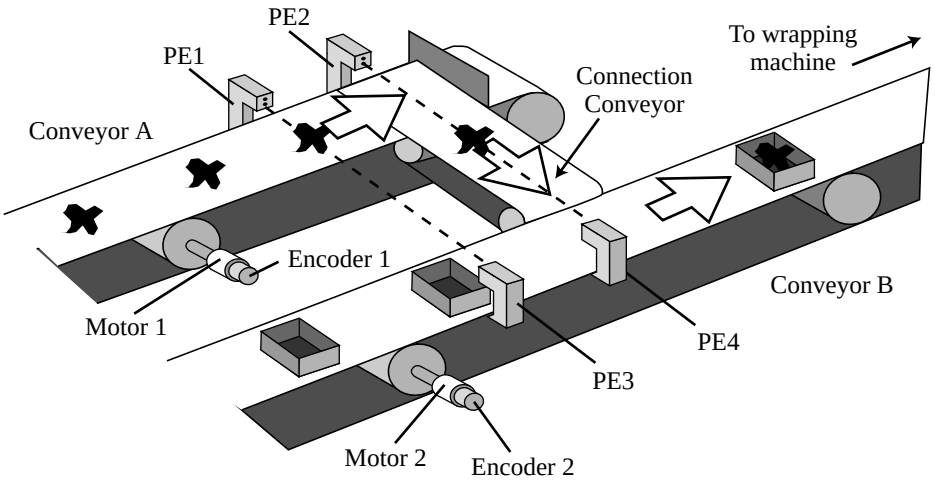


Figure 10. Two-conveyor packaging system.

System Description and Operation

In the system, the parts travel on conveyor A, pass onto the connecting conveyor, and then go to conveyor B, where they are boxed before going to the wrapping machine. The photoelectric sensors PE1 and PE2 detect the presence of a part and initiate a count to determine the part's position from encoder 1. PE3 and PE4 detect the presence of a box and determine its position based on the count inputs from encoder 2.

The control objective is to adjust the speed of conveyor B so that the packaging boxes arrive at the same time as the parts, meaning that they meet at the connecting conveyor. The process information required to implement this control is:

- the offset between the part and the packaging box
- the rate of change of the offset

The parts on conveyor A travel at random intervals, but they travel at a constant speed. The boxes on conveyor B occur at regular intervals, and the speed of conveyor B can be controlled. The photoelectric sensors will be used in the PLC program to detect when to start timing and computing the data from the encoders. The two fuzzy input variables are the part/box offset and the rate of change of the offset.

If a box is present at PE3 and a part is present at PE1, conveyor B should run at the same speed as conveyor A (the reference speed set initially by the operator). If the box is at PE3 but the part is behind PE1, the system will slow conveyor B until the part is at PE1. At this time, the fuzzy controller will indicate an increase in the speed of conveyor B so that it will catch up with conveyor A. The distance traveled by the box is calculated, using the input data from encoder 2, as the difference between the time the box passes PE3 and the time the part passes PE1. The difference in counts between encoder 1 and encoder 2 provides the part/box offset data. This value, denoted as *X*, is calculated as:

$$X = (\text{Encoder 1 counts}) - (\text{Encoder 2 counts})$$

The rate of change of the offset is calculated as the difference between the current offset reading (X_n) and the previous one ($X_{(n-1)}$):

$$\Delta X = X_n - X_{(n-1)}$$

Membership Functions and Rule Creation

To provide enhanced resolution and accuracy, this system uses five–membership function (five-label) fuzzy sets for the two inputs and a seven–membership function fuzzy set for the output. The offset input is named *X* (deviation between part and box) and the offset rate of change input is named ΔX (rate of change of deviation). The fuzzy set for the output is named *S* (speed), which corresponds to the motor speed of conveyor B. The range of each fuzzy input and output variable is from 0 to 4095 counts. This corresponds to a range of ±24 inches for the deviation between the part and box positions, a range of ±10 inches/second for the rate of change of the offset, and a range of ±10 inches/second for the speed of the box conveyor.

The fuzzy logic database for this system contains 25 rules. Figure 11 shows a matrix of the rules, describing the desired output according to the deviation between the part and the box and the rate of change of deviation. This matrix includes a description of the rule inputs and outputs, as well as their respective membership function labels. Once the fuzzy controller receives the inputs, it will determine the final output value based on a logical addition of the selected outcomes.

Deviation / Rate ΔX \ X	NL Box is ahead	NS Box is just ahead	ZR About even	PS Part is just ahead	PL Part is ahead
NL Box is slower	NM Slow the box	NS Slow the box a little	PS Speed up the box a little	PM Speed up the box	PL Speed up the box a lot
NS Box is a little slower	NM Slow the box	NS Slow the box a little	PS Speed up the box a little	PM Speed up the box	PL Speed up the box a lot
ZR About even	NM Slow the box	NS Slow the box a little	ZR No change	PS Speed up the box a little	PM Speed up the box
PS Box is a little faster	NL Slow the box a lot	NM Slow the box	NS Slow the box a little	PS Speed up the box a little	PM Speed up the box
PL Box is faster	NL Slow the box a lot	NL Slow the box a lot	NS Slow the box a little	PS Speed up the box a little	PM Speed up the box

Figure 11. Fuzzy logic rule matrix.

GUIDELINES FOR CONTROLLING COMPLEX SYSTEMS

All fuzzy logic applications are different. However, certain design guidelines apply to all fuzzy logic systems. These guidelines provide the basic elements for the successful implementation of a fuzzy logic control system. They include:

- control objectives
- control system configuration
- input/output determination
- fuzzy inference engine design

Control Objectives

Fuzzy logic can be applied to virtually any type of control system, but it is especially suited for applications that rely heavily on human intuition and experience. The primary objective of applying fuzzy logic to an existing process is to improve the overall process and to automate tasks that previously required human judgment. In a new system, the primary objective of using fuzzy logic is to implement control that cannot be implemented using standard control methods. A system designer should not use fuzzy logic control just because it is available. Rather, the designer should use it because it will enhance the system. Otherwise, the outcome may not be enhanced; it may just become confusing.

Typical applications of fuzzy logic involve batching systems and temperature control loops, where process control involves “tweaking” the output based on judgments about input conditions. For example, a temperature control loop application typically requires a knowledgeable operator who can regulate the control element based on decisions such as “if the temperature is a little high but all other inputs are OK, then turn the steam valve a little clockwise.” This rationale lends itself to fuzzy logic control.

Control System Configuration

The control objective leads to the selection of the fuzzy logic system configuration. There are several types of fuzzy configurations. Fuzzy logic does not have to be applied only in dedicated fuzzy control applications. It can also be used as a complementary system that supports another more conventional control method, such as a PID control loop. When used in this manner, the system is said to be a conventional fuzzy hybrid control system.

Input/Output Determination

Once the fuzzy system configuration has been established, the next step is to determine which inputs and outputs will be used in the fuzzy logic controller. The input conditions, or fuzzy input variables, must be able to be expressed by IF…THEN statements. That is, the input conditions to the fuzzy controller must be able to trigger conditional rules, meaning that they specify one or more output conditions. Inputs should be selected according to the process situations they describe. These variables should relate to similar process elements. If two inputs have little to do with each other, they will generate an outcome that is less effective than the outcome generated by two inputs that deal with the same element. For example, in a temperature-regulating batching operation, the batch temperature and tank jacket temperature both relate to the regulation of the steam valve output. By analyzing these two inputs together, a fuzzy controller can make a precise decision about how much to adjust the steam valve. An analysis of two unrelated inputs, such as batch temperature and liquid level, would not provide such an informed decision.

Fuzzy Inference Engine

The selection of the fuzzy inference engine encompasses the determination of how the fuzzification process will take place (e.g., the number and form of membership function, etc.), how the rules determine an outcome, and how the fuzzy controller implements defuzzification.

Fuzzification. The fuzzification process, which utilizes the membership functions defined by the user, assigns a grade to each fuzzy input received. This grade determines the level of outcome that will be triggered. Therefore, the shape of a fuzzy set's membership functions is important, since this shape determines the input signals' grades, which are mapped to the output membership function.

Some fuzzy controllers allow the user to choose the shape of the membership functions by trial and error, while others have predefined membership function shapes. When using trial and error to determine the function shapes in a closed-loop fuzzy control system, the input membership functions should begin with overlapped Λ-shaped labels (see Figure 12). This ensures smoother control for the first trial due to the coverage provided by the Λ shape and the overlapping at the minimum and maximum points, which creates a balance (i.e., when one label grade is 1, the other is 0). The number of labels, or membership functions, that will form the fuzzy set is also an important part of the system design. For example, if a fuzzy set has five labels covering the same input data range as a three-label fuzzy set, the one with five labels will provide more fine-tuned control, especially if the output membership function also has five labels.

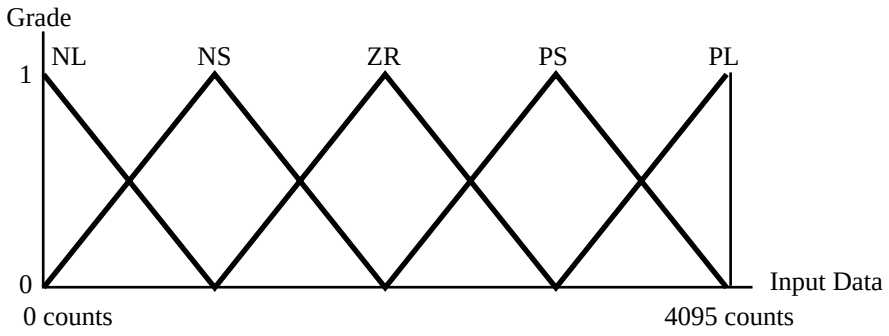


Figure 12. Fuzzy input set with five overlapping membership functions.

Although membership functions do not have to be symmetrical, asymmetrical fuzzy sets should be carefully designed to ensure that they describe the fuzzy variable input properly. In Figure 13a, the inner membership functions provide more sensitivity near the zero label (from NS to PS) than at the NL and PL labels (from NL to NS and from PS to PL). Asymmetrical membership functions are typically used in open-loop system applications.

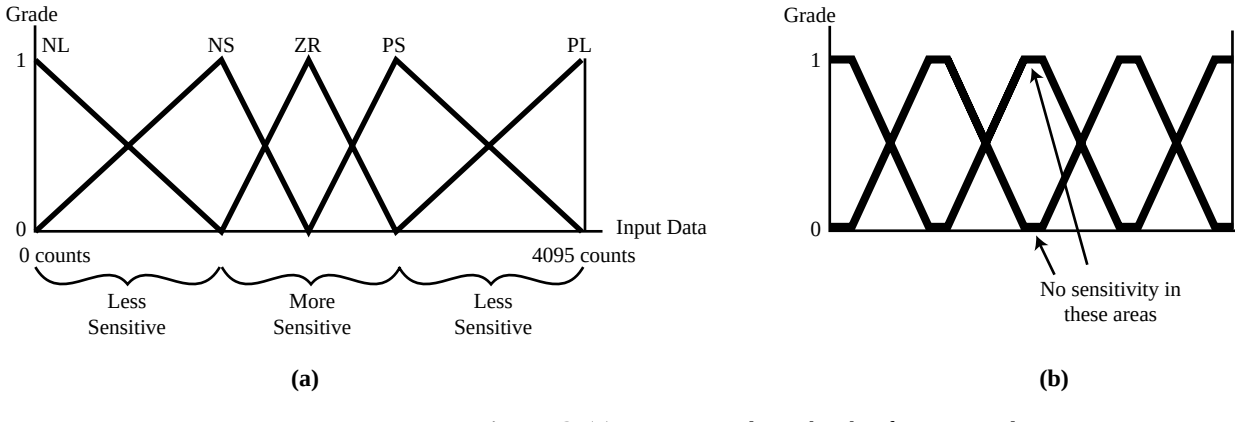


Figure 13. (a) Asymmetrical membership functions and **(b)** membership functions with flat sections.

Sometimes, a membership function in a fuzzy set may not provide any sensitivity between two labels. As illustrated in Figure 13b, the flat sections of the membership functions do not influence neighboring functions or the output. Therefore, the output will not change if the input variable falls in these regions.

Rule Decision Making and Outcome Determination. The easiest way to formulate the rules for a fuzzy logic controller is to first write them as IF…THEN statements that describe how the inputs affect the outcome. Some fuzzy controllers are capable of handling two outputs at the same time, thus allowing two rules to be combined. For example, the rules:

IF A = PS AND B = NS THEN C = ZR
IF A = PS AND B = NS THEN D = NS

can be combined into one rule:

IF A = PS AND B = NS THEN C = ZR and D = NS

This rule gives two outcomes, thus invoking two defuzzification processes, one for each controlling output. It is easiest, however, to create each rule individually (with only one outcome) and then combine them later.

A fuzzy logic controller may or may not provide a choice of output membership function shapes (Λ, Π, S, or Z). Moreover, it may or may not provide a choice about whether the functions are continuous or noncontinuous. However, before defuzzification occurs, all fuzzy controllers add the outcomes based on the appropriate rule logic. If the rule contains a logical AND function, the controller will select the lowest output value; if the rule contains an OR function, the controller will select the highest output value.

If an application requires a highly accurate or smooth output, the rules should be designed so that an input condition triggers two or more rules. To do this, either the input membership functions must overlap or two input conditions must influence the same output.

Defuzzification. During the implementation of a fuzzy logic system, the system designer may be required to choose a defuzzification method, especially if the output membership function is noncontinuous. Defuzzification methods include the center of gravity (centroid), the left-most maximum, and the right-most maximum. If the selected defuzzification method is the center of gravity approach, the triggering rules must be arranged so that at least one rule is triggered at all times. Thus, there must always be an output from a rule. The controller will generate an error if there is no output due to a gap in input condition coverage (see Figure 14).

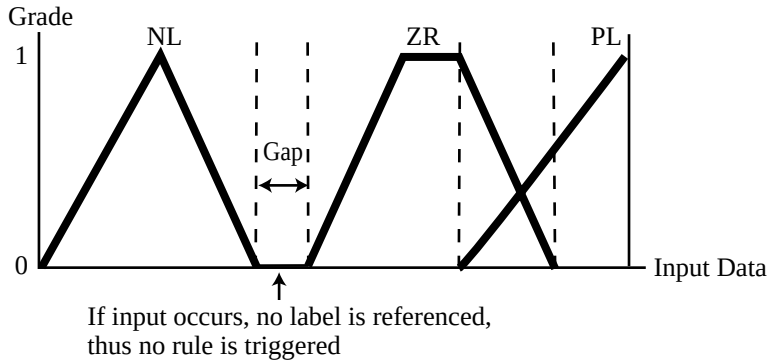


Figure 14. A gap in a fuzzy input set.

CONCLUSION

Fuzzy logic processing is a three-step procedure consisting of fuzzification, fuzzy processing, and defuzzification. Using this three-step process, a fuzzy controller can take vague nondiscrete input data and convert it into a specific output. This conversion process depends on the membership

functions and rules established by the system designer during system implementation. When used correctly, fuzzy logic controllers can improve the performance of PLC systems that control both closed-loop and open-loop systems. They can also lead to the automation of tasks that previously required human intervention. Together, PLCs and fuzzy logic technology form a powerful tool for enhancing complex system automation.