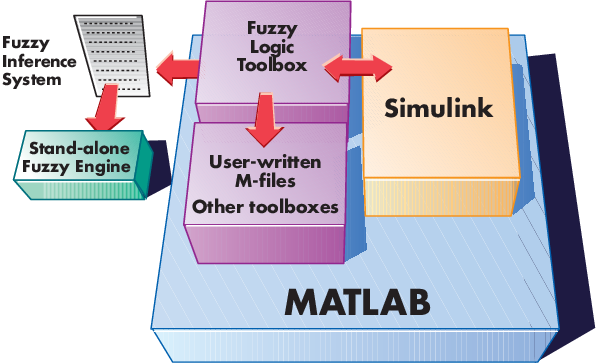
**What Can Fuzzy Logic Toolbox Software Do?**

You can create and edit fuzzy inference systems with Fuzzy Logic Toolbox software. You can create these systems using graphical tools or command-line functions, or you can generate them automatically using either clustering or adaptive neuro-fuzzy techniques.

If you have access to Simulink® software, you can easily test your fuzzy system in a block diagram simulation environment.

The toolbox also lets you run your own stand-alone C programs directly. This is made possible by a stand-alone Fuzzy Inference Engine that reads the fuzzy systems saved from a MATLAB session. You can customize the stand-alone engine to build fuzzy inference into your own code. All provided code is ANSI® compliant.



Because of the integrated nature of the MATLAB environment, you can create your own tools to customize the toolbox or harness it with another toolbox, such as the Control System Toolbox™, Deep Learning Toolbox™, or Optimization Toolbox™ software.

### Fuzzy Sets

Fuzzy logic starts with the concept of a fuzzy set. A fuzzy set is a set without a crisp, clearly defined boundary. It can contain elements with only a partial degree of membership.

### Membership Functions

A membership function (MF) is a curve that defines how each point in the input space is mapped to a membership value (or degree of membership) between 0 and 1. The input space is sometimes referred to as the universe of discourse, a fancy name for a simple concept.

### Membership Functions in Fuzzy Logic Toolbox Software

The only condition a membership function must really satisfy is that it must vary between 0 and 1. The function itself can be an arbitrary curve whose shape we can define as a function that suits us from the point of view of simplicity, convenience, speed, and efficiency.

A classical set might be expressed as

*A*={*x*↓*x*>6}

A fuzzy set is an extension of a classical set. If X is the universe of discourse and its elements are denoted by x, then a fuzzy set A in X is defined as a set of ordered pairs.

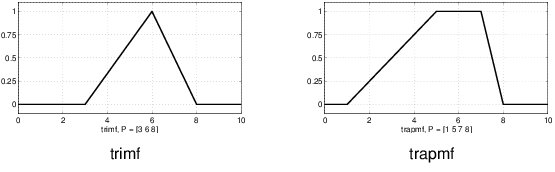
*A*{*x*,*μA*(*x*)↓*x*∈*X*}

*µA*(*x*) is called the membership function (or MF) of *x* in *A*. The membership function maps each element of *X* to a membership value between 0 and 1.

The toolbox includes 11 built-in membership function types. These 11 functions are, in turn, built from several basic functions:

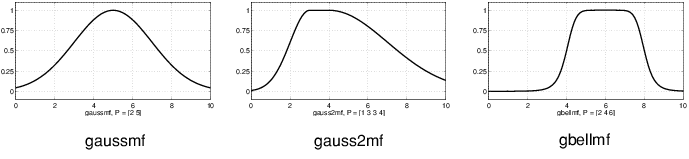
* Piece-wise linear functions
* Gaussian distribution function
* Sigmoid curve
* Quadratic and cubic polynomial curves

The simplest membership functions are formed using straight lines. Of these, the simplest is the triangular membership function, and it has the function name trimf. This function is nothing more than a collection of three points forming a triangle. The trapezoidal membership function, trapmf, has a flat top and really is just a truncated triangle curve. These straight line membership functions have the advantage of simplicity.

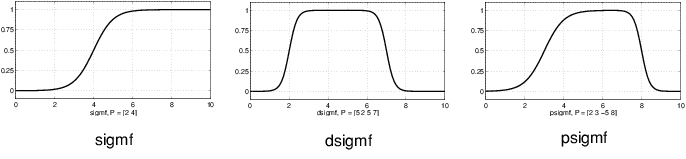


Two membership functions are built on the Gaussian distribution curve: a simple Gaussian curve and a two-sided composite of two different Gaussian curves. The two functions are gaussmf and gauss2mf.

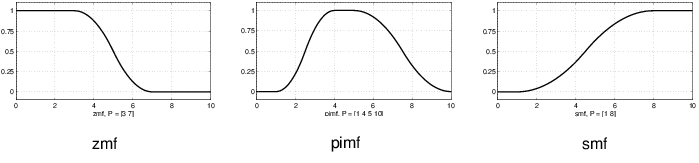
The generalized bell membership function is specified by three parameters and has the function name gbellmf. The bell membership function has one more parameter than the Gaussian membership function, so it can approach a non-fuzzy set if the free parameter is tuned. Because of their smoothness and concise notation, Gaussian and bell membership functions are popular methods for specifying fuzzy sets. Both of these curves have the advantage of being smooth and nonzero at all points.



Although the Gaussian membership functions and bell membership functions achieve smoothness, they are unable to specify asymmetric membership functions, which are important in certain applications. Next, you define the sigmoidal membership function, which is either open left or right. Asymmetric and closed (i.e. not open to the left or right) membership functions can be synthesized using two sigmoidal functions, so in addition to the basic sigmf, you also have the difference between two sigmoidal functions, dsigmf, and the product of two sigmoidal functions psigmf.



Polynomial based curves account for several of the membership functions in the toolbox. Three related membership functions are the Z, S, andPi curves, all named because of their shape. The function zmf is the asymmetrical polynomial curve open to the left, smf is the mirror-image function that opens to the right, and pimf is zero on both extremes with a rise in the middle.



There is a very wide selection to choose from when you're selecting a membership function. You can also create your own membership functions with the toolbox. However, if a list based on expanded membership functions seems too complicated, just remember that you could probably get along very well with just one or two types of membership functions, for example the triangle and trapezoid functions. The selection is wide for those who want to explore the possibilities, but expansive membership functions are not necessary for good fuzzy inference systems. Finally, remember that more details are available on all these functions in the reference section.

#### Summary of Membership Functions

* Fuzzy sets describe vague concepts (e.g., fast runner, hot weather, weekend days).
* A fuzzy set admits the possibility of partial membership in it. (e.g., Friday is sort of a weekend day, the weather is rather hot).
* The degree an object belongs to a fuzzy set is denoted by a membership value between 0 and 1. (e.g., Friday is a weekend day to the degree 0.8).
* A membership function associated with a given fuzzy set maps an input value to its appropriate membership value.

# Summary of If-Then Rules

Interpreting if-then rules is a three-part process. This process is explained in detail in the next section:

1. **Fuzzify inputs**: Resolve all fuzzy statements in the antecedent to a degree of membership between 0 and 1. If there is only one part to the antecedent, then this is the degree of support for the rule.
2. **Apply fuzzy operator to multiple part antecedents**: If there are multiple parts to the antecedent, apply fuzzy logic operators and resolve the antecedent to a single number between 0 and 1. This is the degree of support for the rule.
3. **Apply implication method**: Use the degree of support for the entire rule to shape the output fuzzy set. The consequent of a fuzzy rule assigns an entire fuzzy set to the output. This fuzzy set is represented by a membership function that is chosen to indicate the qualities of the consequent. If the antecedent is only partially true, (i.e., is assigned a value less than 1), then the output fuzzy set is truncated according to the implication method.

Fuzzy inference process comprises of five parts:

## [Fuzzification of the input variables](https://in.mathworks.com/help/fuzzy/fuzzy-inference-process.html#FP346)

## [Application of the fuzzy operator (AND or OR) in the antecedent](https://in.mathworks.com/help/fuzzy/fuzzy-inference-process.html#FP347)

## [Implication from the antecedent to the consequent](https://in.mathworks.com/help/fuzzy/fuzzy-inference-process.html#FP348)

## [Aggregation of the consequents across the rules](https://in.mathworks.com/help/fuzzy/fuzzy-inference-process.html#a1054218661b1)

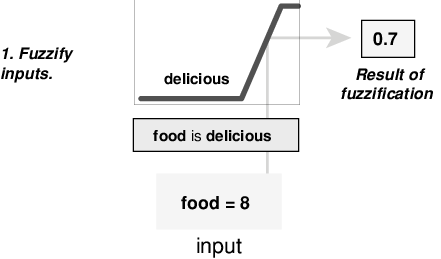
## [Defuzzification](https://in.mathworks.com/help/fuzzy/fuzzy-inference-process.html#a1054218744b1)

## A [fuzzy inference diagram](https://in.mathworks.com/help/fuzzy/fuzzy-inference-process.html#FP350) displays all parts of the fuzzy inference process — from fuzzification through defuzzification.

### Fuzzify Inputs

The first step is to take the inputs and determine the degree to which they belong to each of the appropriate fuzzy sets via membership functions (fuzzification). In Fuzzy Logic Toolbox™ software, the input is always a crisp numerical value limited to the universe of discourse of the input variable (in this case, the interval from 0 through 10) . The output is a fuzzy degree of membership in the qualifying linguistic set (always the interval from 0 through 1). Fuzzification of the input amounts to either a table lookup or a function evaluation.

This example is built on three rules, and each of the rules depends on resolving the inputs into several different fuzzy linguistic sets: service is poor, service is good, food is rancid, food is delicious, and so on. Before the rules can be evaluated, the inputs must be fuzzified according to each of these linguistic sets. For example, to what extent is the food delicious? The following figure shows how well the food at the hypothetical restaurant (rated on a scale from 0 through 10) qualifies as the linguistic variable delicious using a membership function. In this case, we rate the food as an 8, which, given the graphical definition of delicious, corresponds to µ = 0.7 for the delicious membership function.



In this manner, each input is fuzzified over all the qualifying membership functions required by the rules.

### Apply Fuzzy Operator

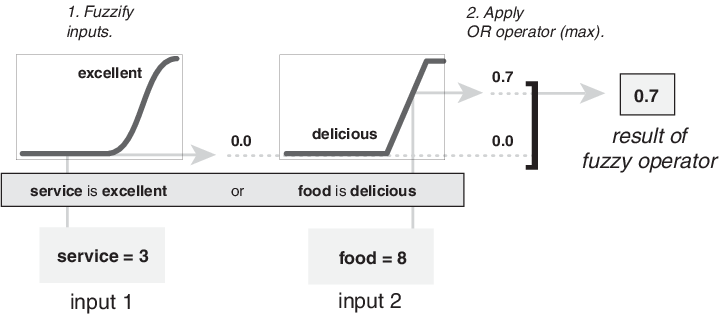
After the inputs are fuzzified, you know the degree to which each part of the antecedent is satisfied for each rule. If the antecedent of a rule has more than one part, the fuzzy operator is applied to obtain one number that represents the result of the rule antecedent. This number is then applied to the output function. The input to the fuzzy operator is two or more membership values from fuzzified input variables. The output is a single truth value.

As is described in [Logical Operations](https://in.mathworks.com/help/fuzzy/foundations-of-fuzzy-logic.html#bp78l70-5) section, any number of well-defined methods can fill in for the AND operation or the OR operation. In the toolbox, two built-in AND methods are supported: min (minimum) and prod (product). Two built-in OR methods are also supported: max (maximum), and the probabilistic OR method probor. The probabilistic OR method (also known as the algebraic sum) is calculated according to the equation:

|  |
| --- |
| probor(a,b) = a + b - ab |

In addition to these built-in methods, you can create your own methods for AND and OR by writing any function and setting that to be your method of choice.

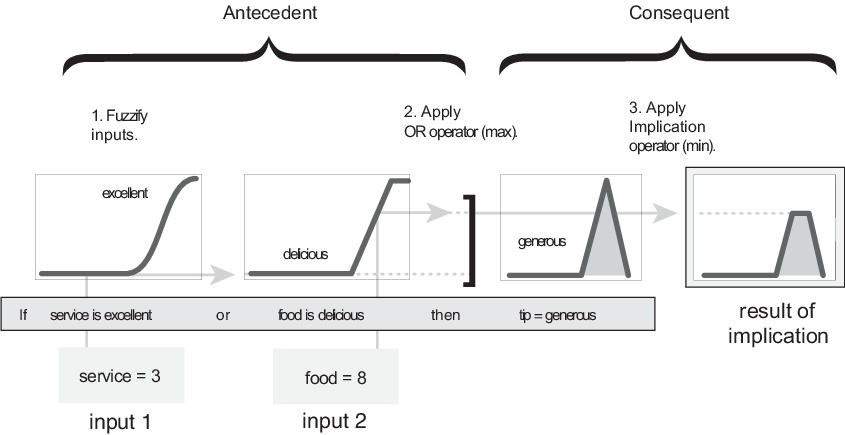
The following figure shows the OR operator max at work, evaluating the antecedent of the rule 3 for the tipping calculation. The two different pieces of the antecedent (service is excellent and food is delicious) yielded the fuzzy membership values 0.0 and 0.7 respectively. The fuzzy OR operator simply selects the maximum of the two values, 0.7, and the fuzzy operation for rule 3 is complete. The probabilistic OR method would still result in 0.7.



### Apply Implication Method

Before applying the implication method, you must determine the rule weight. Every rule has a weight (a number from 0 through 1), which is applied to the number given by the antecedent. Generally, this weight is 1 (as it is for this example) and thus has no effect on the implication process. However, you can decrease the effect of one rule relative to the others by changing its weight value to something other than 1.

After proper weighting has been assigned to each rule, the implication method is implemented. A consequent is a fuzzy set represented by a membership function, which weights appropriately the linguistic characteristics that are attributed to it. The consequent is reshaped using a function associated with the antecedent (a single number). The input for the implication process is a single number given by the antecedent, and the output is a fuzzy set. Implication is implemented for each rule. Two built-in methods are supported, and they are the same functions that are used by the AND method: min (minimum), which truncates the output fuzzy set, and prod (product), which scales the output fuzzy set.



### Note

Sugeno systems always use the product implication method.

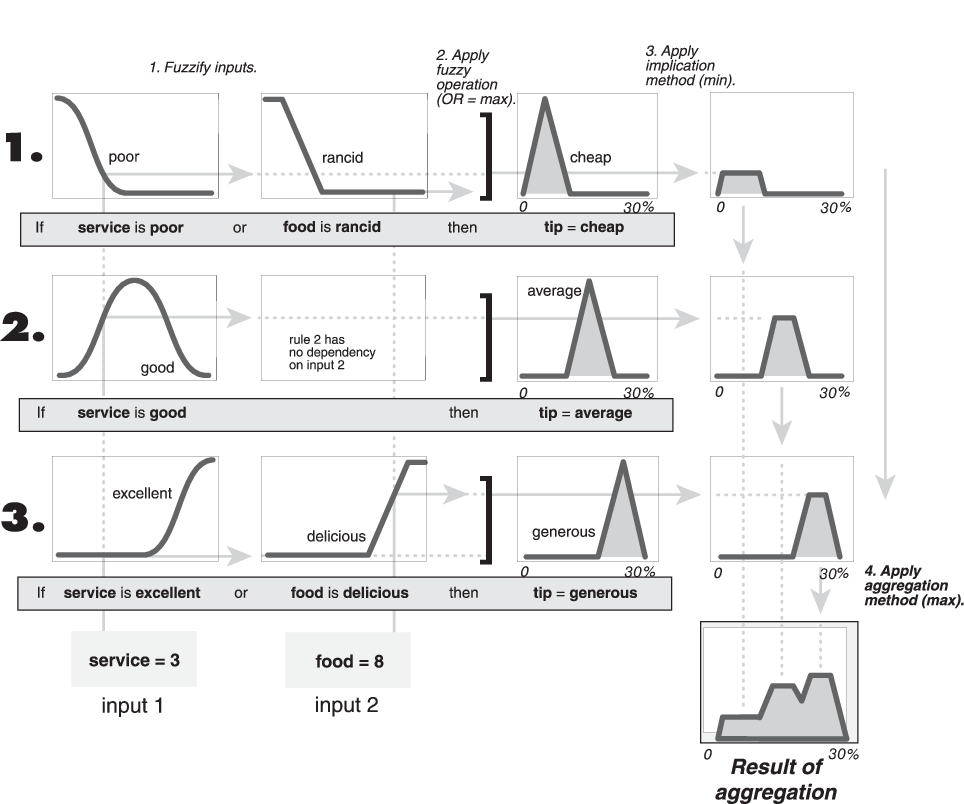
### Aggregate All Outputs

Since decisions are based on testing all the rules in a FIS, the rule outputs must be combined in some manner. Aggregation is the process by which the fuzzy sets that represent the outputs of each rule are combined into a single fuzzy set. Aggregation only occurs once for each output variable, which is before the final defuzzification step. The input of the aggregation process is the list of truncated output functions returned by the implication process for each rule. The output of the aggregation process is one fuzzy set for each output variable.

As long as the aggregation method is commutative, then the order in which the rules are executed is unimportant. Three built-in methods are supported:

* max (maximum)
* probor (probabilistic OR)
* sum (sum of the rule output sets)

In the following diagram, all three rules are displayed to show how the rule outputs are aggregated into a single fuzzy set whose membership function assigns a weighting for every output (tip) value.



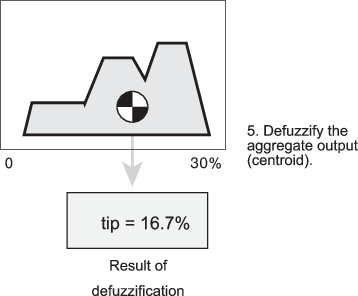
### Note

Sugeno systems always use the sum aggregation method.

### Defuzzify

The input for the defuzzification process is a fuzzy set (the aggregate output fuzzy set) and the output is a single number. As much as fuzziness helps the rule evaluation during the intermediate steps, the final desired output for each variable is generally a single number. However, the aggregate of a fuzzy set encompasses a range of output values, and so must be defuzzified to obtain a single output value from the set.

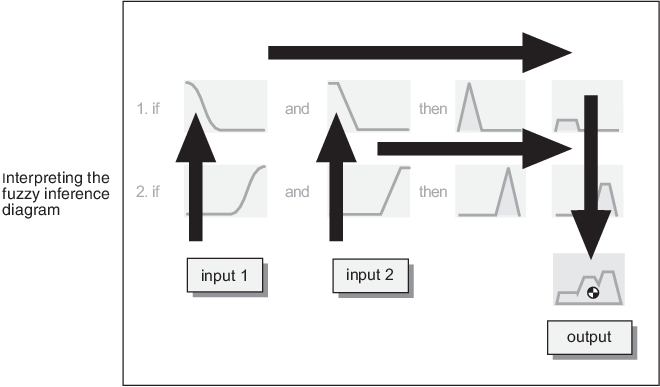
There are five built-in defuzzification methods supported: centroid, bisector, middle of maximum (the average of the maximum value of the output set), largest of maximum, and smallest of maximum. Perhaps the most popular defuzzification method is the centroid calculation, which returns the center of area under the curve, as shown in the following:



While the aggregate output fuzzy set covers a range from 0% though 30%, the defuzzified value is between 5% and 25%. These limits correspond to the centroids of the cheap and generous membership functions, respectively.

### Fuzzy Inference Diagram

The fuzzy inference diagram is the composite of all the smaller diagrams presented so far in this section. It simultaneously displays all parts of the fuzzy inference process you have examined. Information flows through the fuzzy inference diagram as shown in the following figure.



In this figure, the flow proceeds up from the inputs in the lower left, across each row, and then down the rule outputs in the lower right. This compact flow shows everything at once, from linguistic variable fuzzification all the way through defuzzification of the aggregate output.

The following figure shows the actual full-size fuzzy inference diagram. Using a fuzzy inference diagram, you can learn a lot about how the system operates. For instance, for the particular inputs in this diagram, you can see that the implication method is truncation with the min function. The max function is used for the fuzzy OR operation. Rule 3 (the bottom-most row in the diagram shown previously) has the strongest influence on the output. The Rule Viewer described in [The Rule Viewer](https://in.mathworks.com/help/fuzzy/building-systems-with-fuzzy-logic-toolbox-software.html#FP484) is an implementation of the fuzzy inference diagram.

