COM2536 Project 1

Elvin Huseynli - 20290122 **Bekir Emirhan Akay - 19290286**

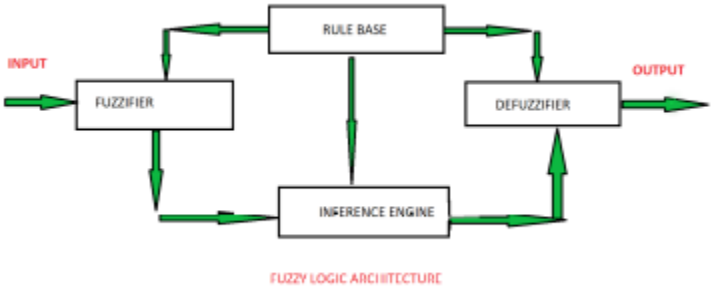
1. Abstract

In the past, the majority of scientists were reducing uncertainty as much as possible. However, it is not possible to completely remove uncertainty in our systems. Therefore, scientists have changed their ideas for uncertainty. This idea aims to eliminate the uncertainty for our advantage. Thus, they developed fuzzy logic systems. There are lots of fuzzy logic system types. Fuzzy logic system consists of 4 foundation parts. Firstly, the fuzzification interface. In this phase, we use some fuzzification functions such as triangular, sigmoidal, trapezoidal etc. Second part of the fuzzy logic system is the knowledge base. Knowledge base contains inputs for our Fuzzy logic system. Third part is fuzzy inference. Fuzzy inference is the process of formulating the mapping from a given input to output. There are a variety of methods for fuzzy inference such as Mamdani method, Sugeno method and so on. And the final one is defuzzification, which is the last part of the foundation of a fuzzy logic system. Defuzzification is the process of obtaining one crisp output from an aggregated fuzzy set. There are several methods to defuzzify the fuzzified input. For instance, the mean of maximum and center of area (centroid) defuzzification etc are some of the well-known examples of this. However, in this project, we are going to use Mamdani inference and center of area defuzzification methods.

2. Introduction

As we said in the abstract section, a fuzzy logic system consists of 4 foundations. Fuzzification, knowledge base, fuzzy inference and defuzzification. The schema of the fuzzy logic system is shown in Figure 1.

Figure1:



First we fuzzify input using the fuzzification function in the fuzzification phase.

2.1. Fuzzification

In this phase we convert crisp sets to fuzzy sets using membership functions.

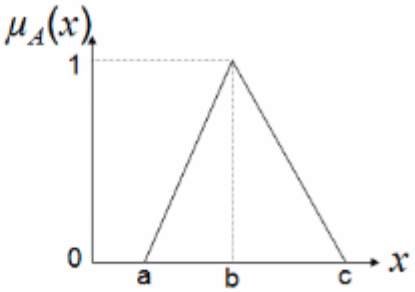
2.1.1. Membership functions and some examples

Membership function is defined on crisp domain and fuzzy codomain. Domain can be [-∞,+∞] but the codomain must lie in the interval [0,1]. In some cases domains can be separated into intervals or a single interval.

a)Triangular membership function:

Figure 2:

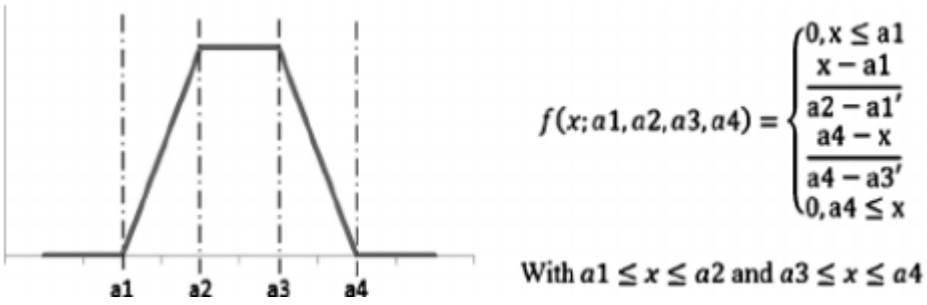
$$\mu_A(x) = \begin{cases} 0 & if\ x \le a \\ \frac{x-a}{b-a} & if\ a \le x \le b \\ \frac{c-x}{c-b} & if\ b \le x \le c \\ 0 & if\ x \ge c \end{cases}$$



Triangular membership function's domain is defined on the interval [a,c]. The image of 'b' is the height of the function (membership value takes 1). Membership value of the function increases from 0 to 1 between 'a' and 'b' (1 is not included) and decreases to 0 between 'b' and 'c'. Membership value takes its maximum value when *x* is equal to 'b'. Triangular membership function is one of the most basic membership functions to fuzzify crisp inputs. Representation of the triangular number is (a1,a2,a3). Description of triangular membership function is shown in Figure 2.

b)Trapezoidal membership function:

Figure 3:



Trapezoidal membership function is defined on [a1,a4] for all *IR*. Unlike the triangular membership function, there is no single maxima for membership function; instead, there is an interval. This maxima (membership value gets value 1) interval is defined on [a2,a3]. Between [a1,a2] membership value increases and between [a3,a4] membership value decreases. Representation of trapezoidal membership function is (a1,a2,a3,a4). Description of trapezoidal membership function is shown in Figure 3.
As we fuzzify the crisp input using membership functions, we combine it with the knowledge base.

2.2. Knowledge base

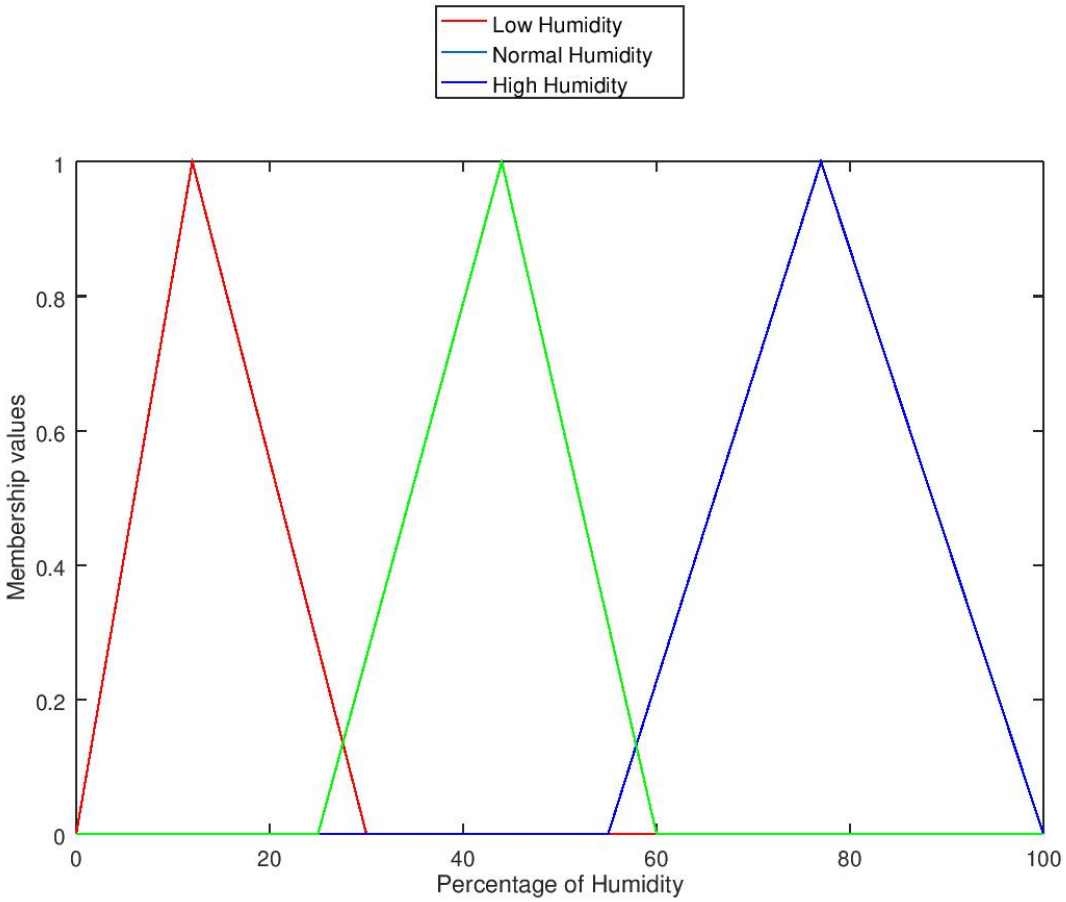
It is the second fundamental phase of the Fuzzy Logic System. It consists of 2 parts; rule base and database. For both of them we are going to

explain using a good example. Sample content is a Fuzzy Logic System where humidity and speed of wind is input, and the temperature is output.

2.2.1. Database

Database is the knowledge that is the main part of our crisp input. For instance, if the humidity takes values between [0,30], then it is dry; if the humidity is between [25,60], then it is normal; and if it takes values between [55,100], then the humidity is high. We can easily define a fuzzy set using the data in Figure 4. Fuzzy Logic Systems use this data to fuzzify it and to implement it in the Fuzzy Inference Systems.

Figure 4:



Following phases use these fuzzified membership values.

2.1.2. Rule base

Rule base is the auxiliary part of the database. With the help of a rule base we can determine rules and statements to infer our fuzzy set. For example, if the humidity is ‘high’ and wind speed is ‘fast’, then temperature is ‘low’. This is one of the rules for our Fuzzy logic system. We also use operators like ‘and’ & ‘or’ to describe our rules. If-then statements are also one of the important parts of the Mamdani inference system. A sample rule matrix and if-then statements are shown in Figure 5.

Figure 5:

Rule matrix	Low Humidity	Normal Humidity	High Humidity
Weak wind	Very hot	Normal	Normal
Normal wind	Hot	Normal	Cold
Strong wind	Normal	Cold	Cold

-if wind is weak and humidity is low then temperature is very hot.
-if wind is weak and humidity is normal then temperature is hot.
-if wind is weak and humidity is high then temperature is normal.
-if wind is normal and humidity is low then temperature is normal.
-if wind is normal and humidity is normal then temperature is normal.
-if wind is normal and humidity is high then temperature is cold.
-if wind is high and humidity is low then temperature is normal.
-if wind is high and humidity is normal then temperature is cold.
-if wind is high and humidity is high then temperature is cold.

Antecedent **Consequent**

2.3. Inference

Fuzzy Inference is a method that interprets the input values and uses some predefined rules and operators (‘and’ & ‘or’) to map inputs to output vectors. Then, this output vector goes to the last part of the Fuzzy logic system, which is the defuzzification phase to get one crisp output (If we use Tsukamoto Method, we don’t need to implement the last phase, as we get the crisp output directly). A fuzzy inference system can consist of **n** inputs & **m** outputs. As we mentioned above, during the inference phase we use the knowledge base to define our rules and operations. We have several methods to infer our fuzzy input to fuzzy output or crisp output.

2.3.1. Fuzzy Inference Types

Despite the fact that there are several fuzzy inference methods, we only mentioned the Mamdani method and Larsen method in the introduction part. Basic explanation and illustration of these methods are shown below.

2.3.1.1. Mamdani method

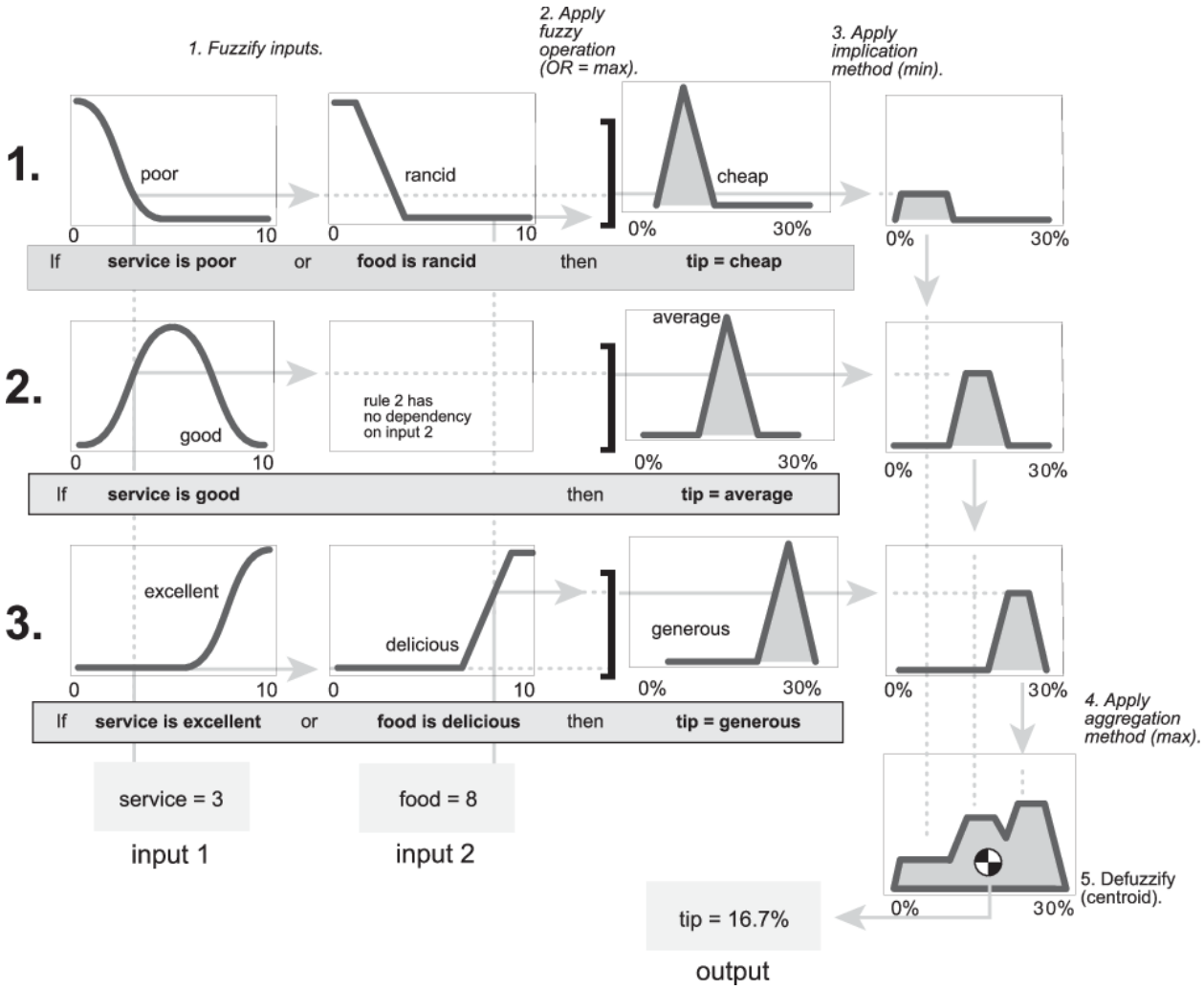
Mamdani's method is one of the widely used methods for fuzzy inference. It was developed by Ebhasim Mamdani in 1975. The reason this method is widely used arises from the logic the method uses is similar to human logic. It uses 2 logical operators : ∧ (logical ‘and’) and v (logical ‘or’). For example, in the system that has 2 inputs and 1 output (The example that we have mentioned; wind-humidity-temperature system): we get max of mins from fuzzified inputs.

-if wind is weak and humidity is normal then temperature is hot.

$(\mu_{weak} \wedge \mu_{normal}) \vee \mu_{hot}$ will be our result according to Mamdani inference method.

Example illustration of the mamdani inference system is shown in Figure 6.

Figure 6:



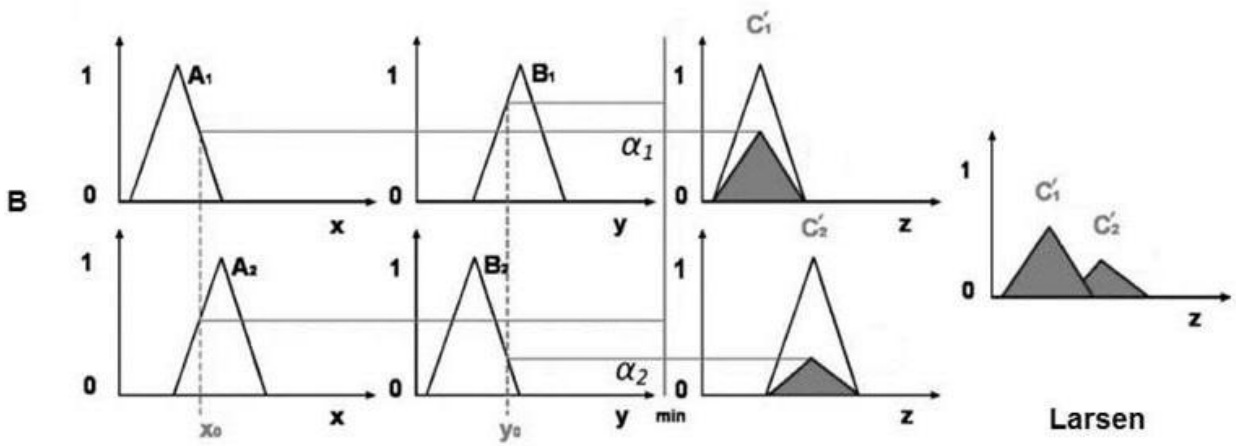
2.3.2.2. Larsen Method

Although this method is not widely used, it can be utilized in some conditions. This method uses product operator (.) for fuzzy implication and max product for the composition.

-if the wind is weak and humidity is normal then temperature is hot.

$(\mu_{weak} \cdot \mu_{normal}) \vee \mu_{hot}$ will be our result according to the Mamdani inference method.

Example illustration of the mamdani inference system is shown in Figure 7.



2.4. Defuzzification

The last part of the fuzzy logic system is defuzzification. Defuzzification is the process of converting a fuzzy set to a crisp set. In a fuzzy logic system we generally get one crisp output or crisp output vector. In a fuzzy logic system fuzzy input enters the inference phase, then results of inference enters the defuzzification phase to get final results. There are several methods to defuzzify a fuzzy set. The most commonly used method is the center of area method (centroid method). Other methods are also used in some areas and some cases. However, if we use the Tsukamoto Method in the inference phase, we do not need to defuzzify, because it directly transforms input into crisp output.

2.4.1. Defuzzification Methods

2.4.1.1 Center of area method:

The basic idea of the centroid method is to find the center of area, the membership function which is found in the inference phase. To find that area, we integrate f(x)*x over an interval [a,b] where ‘a’ is the lower bound of the output membership function alfa cut set 0 and ‘b’ is the upper bound of the output membership function alfa cut set 0. Then we integrate f(x) over an interval [a,b]. Lastly, we divide their results to get final crisp output. The formula of center of area for an infinite number of points is shown below.

$$CoA = \frac{\int_{x_{min}}^{x_{max}} f(x) * x \, dx}{\int_{x_{min}}^{x_{max}} f(x) dx}$$

2.4.1.2. Bisector of area method

This method is not commonly used as the centroid method, but this is used by some applications. The basic idea is that the method finds the vertical line that divides the fuzzy set into two sub-regions of equal area. It is sometimes, but not always, coincident with the centroid line. The formula of the bisector of the area is shown below.

$\int_{\alpha}^{x^*} \mu_A(x) \, dx = \int_{x^*}^{\beta} \mu_A(x) dx$, where α = min $\{x | \, x \in X\}$ and β = max $\{x | \, x \in X\}$

3. Construction

In this section we are going to give detailed information on how we implemented Fuzzy Logic System stages in our code. Using the MATLAB Programming language and its Fuzzy Logic toolbox, it became easier to utilize the functions to create a Fuzzy Inference System. Before creating and implementing the stages, the first process to be done was to create a Fuzzy Inference System in our code. By using built-in function `mamfis(Name, Value)`, we created a Mamdani Fuzzy Inference System. After this we're getting started to create other stages of the Fuzzy Inference System.

3.1. Fuzzification

In this section we are creating membership functions based on giving charts and data. But before adding membership functions, we had to define which membership functions are going to be part of Input and Output data. With the help of `addInput(fisIn, range, Name, Value)` and `addOutput(fisIn, range, Name, Value)` we have added all of the inputs and outputs to our Mamdani Inference System. After completing this process, it is time to create membership functions for our Mamdani Inference System. We are going to use triangular membership functions for our Mamdani FIS. To add those membership functions to our FIS, we use `addMF(fisIn, varName, type, parameters, Name, Value)` as a utilizer function. The way this function creates membership functions and adds them to our FIS is mainly dependent on `type` and `parameters`. `type` takes triangular membership function `trimf` as Input and `parameters` contains triangular number, which determines the domain of membership function. While creating those membership functions, we consider `Value` as one of the important Input values for `addMF(..)` function, as it will help us to determine and create rules. That's why it is important to create and add membership functions in the given order based on charts and data. Also when there are more than one input and output for each, the order of the inputs and outputs added to Mamdani FIS matters, as the function creates indexes for each of them, which is a useful and helpful condition when

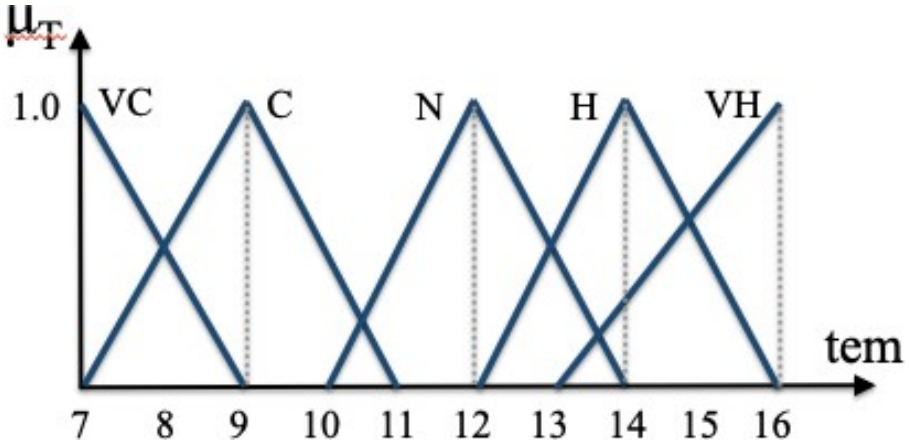
adding inputs and outputs to the Rule Base. In the following stage, we are going to explain our Rule Base, based on input and output data.

3.2. Knowledge Base

3.2.1. Database

In this phase we define crisp values for input and output data. In addition to this, we use this database to create membership functions, as they contain triangular numbers [a b c], which contain [a, c] as an interval. For example, if we take a look at Figure 8, we can see that the range values represent different temperature amplitudes. For “Cold”, abbreviated as “C”, we can observe that it has a triangular number of [7 9 11], which shows that if the temperature is in the range [7 11], it means the weather is cold. The same applies for “Normal”, abbreviated as “N”; it has a triangular number [10 12 14], meaning that if the temperature is in the interval [10 14] the weather is normal. When we used `addMF(fisIn, varName, type, parameters, Name, Value)` function to create membership functions, `parameters` attribute was taking the values of the triangular numbers, to create a fuzzy idea for each range.

Figure 8:



3.2.2. Rule Base

In this section we are going to discuss the rule base of our Mamdani Fuzzy Inference System. As MATLAB provides a function to create a rule base for our Mamdani FIS, it becomes easier to define rules, without using any verbal rules. In our code `addRule(fisIn, ruleDescription)` helps us to add rules to our system. `fisIn` takes the input of our Mamdani FIS, which we defined before, and `ruleDescription` defines our rule list. Our rule list is defined using vectors, but can be created separately, then being added to our rule list. To create our rule list, we need to get the main idea on creating the rule base. `rule = [a b c d e]` is the simplest way to define rules. Below the representations of each component is explained.

- 'a' - index of membership function for first input
- 'b' - index of membership function for second input
- 'c' - index of membership function for output
- 'd' - rule weight from 0 to 1
- 'e' - fuzzy operator (1 for 'AND' and 2 for 'OR')

As defined above, index of membership functions are defined as the order they are added to the input and output data. For rule weight, as the rules are all of the same weight, we are defining them to be 1. For fuzzy operators, we are choosing 1 ('AND' operator), as Mamdani FIS uses 'max of min' operation. After adding those rules as vectors to the rule list, we can add the rule list to our Mamdani FIS, using the `addRule(...)` function.

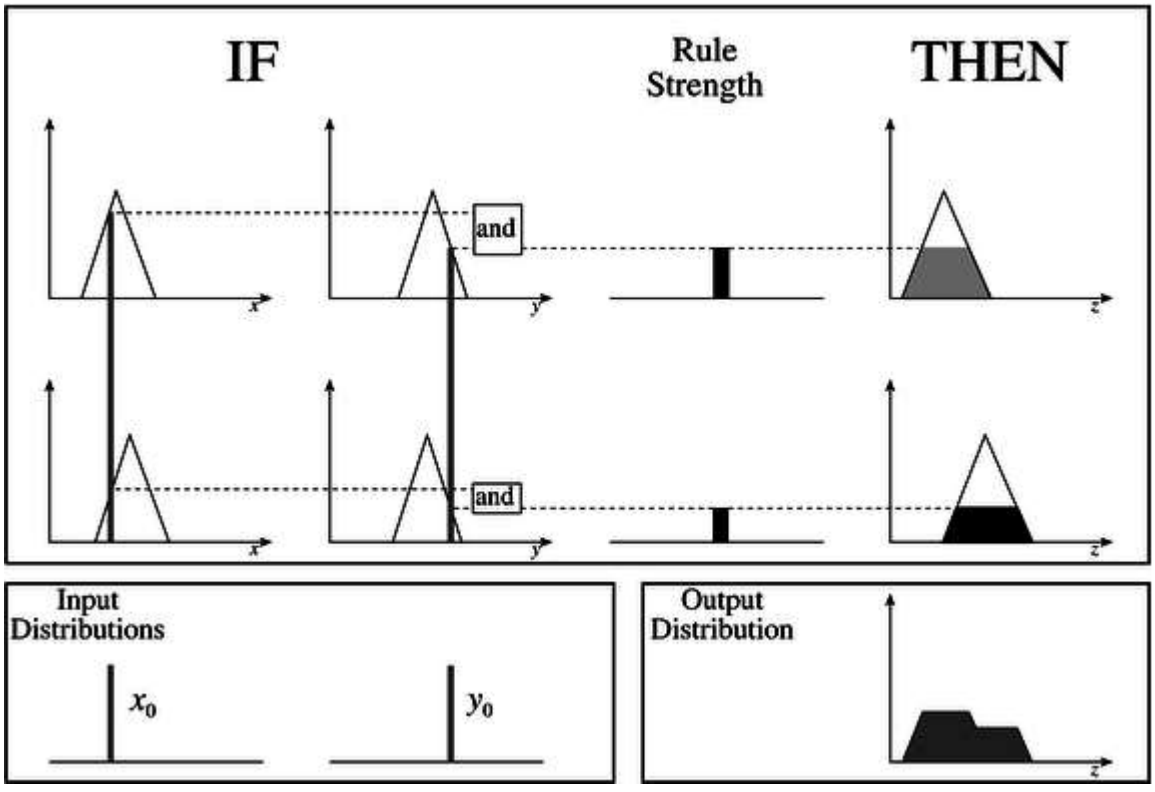
3.3. Inference

In this section we are supposed to discuss the Mamdani Inference System and its working principles. Mamdani Inference System is known to be one of the convenient methods to create a FIS. It was first introduced as a method to create a control system by synthesizing a set of linguistic control rules obtained from experienced human operators. In a Mamdani Inference System, the output of each rule is a fuzzy set. Since these systems have intuitive and easily understandable rule bases, they are convenient to expert system applications, where rules are created from human expert

knowledge. There are 6 general steps to compute the correct output for the Mamdani Inference System. Also the block diagram is shown in Figure 9.

- ☐ **Step 1:** Set of fuzzy rules need to be determined in this step.
- ☐ **Step 2:** In this step, by using the input membership function, the input would be made fuzzy.
- ☐ **Step 3:** Now establish the rule strength by combining the fuzzified inputs according to fuzzy rules.
- ☐ **Step 4:** In this step, determine the consequence of rule by combining the rule strength and the output membership function.
- ☐ **Step 5:** For getting output distribution combine all the consequences.
- ☐ **Step 6:** Finally, a defuzzified output distribution is obtained.

Figure 9:



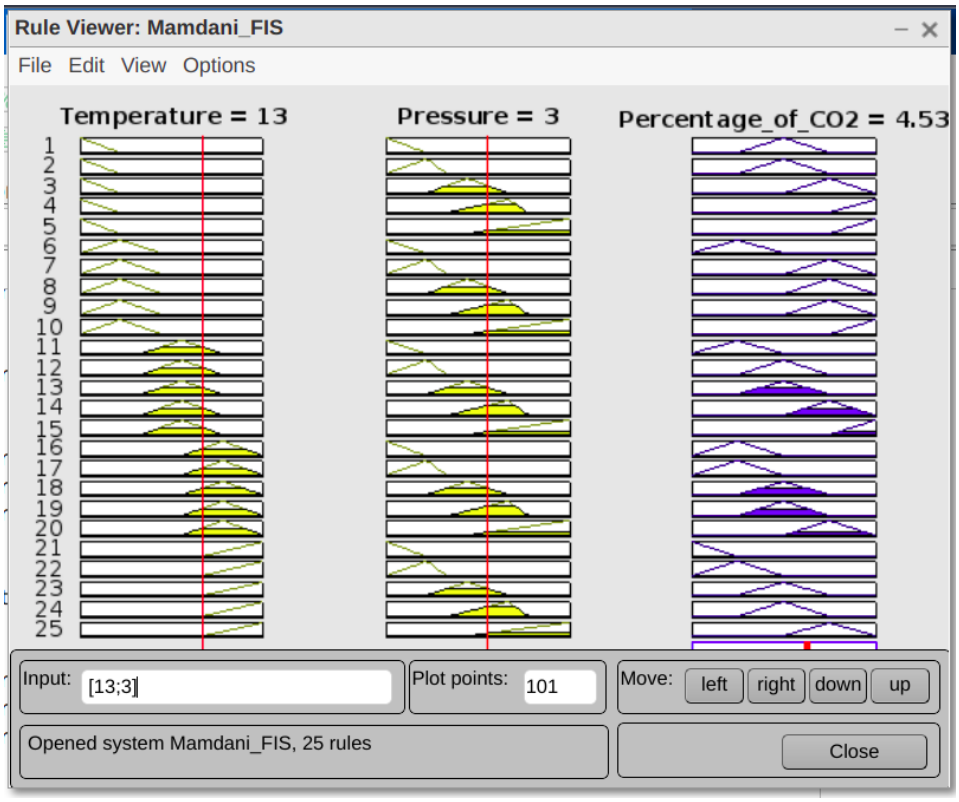
3.4. Defuzzification

In this phase we are implementing the “center of area method”, also known as “centroid method”. This method determines the center of the area of the fuzzy set and returns the corresponding crisp value. This method is applied by using a predefined function in MATLAB - `evalfis(fisIn, input)`, we can easily apply the centroid method. `fisIn` parameter takes Mamdani FIS as input and `input` attribute takes input values as vector to evaluate the result.

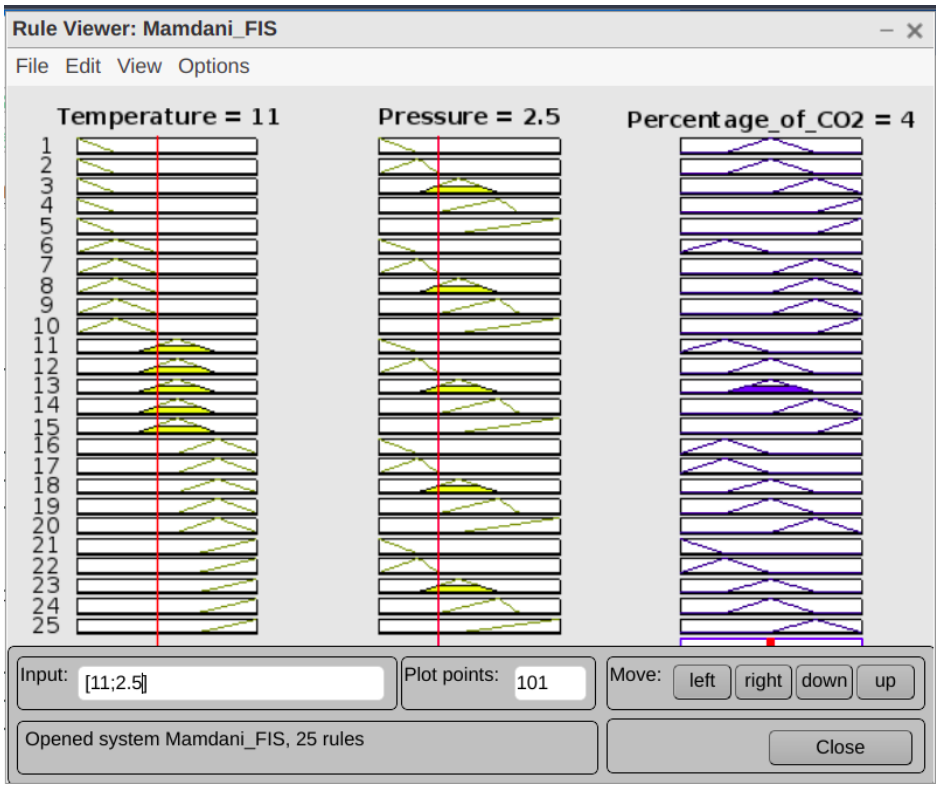
4. Implementation

In this section we are gonna demonstrate the source code for our Fuzzy Inference system and implement this code using some input and output data.

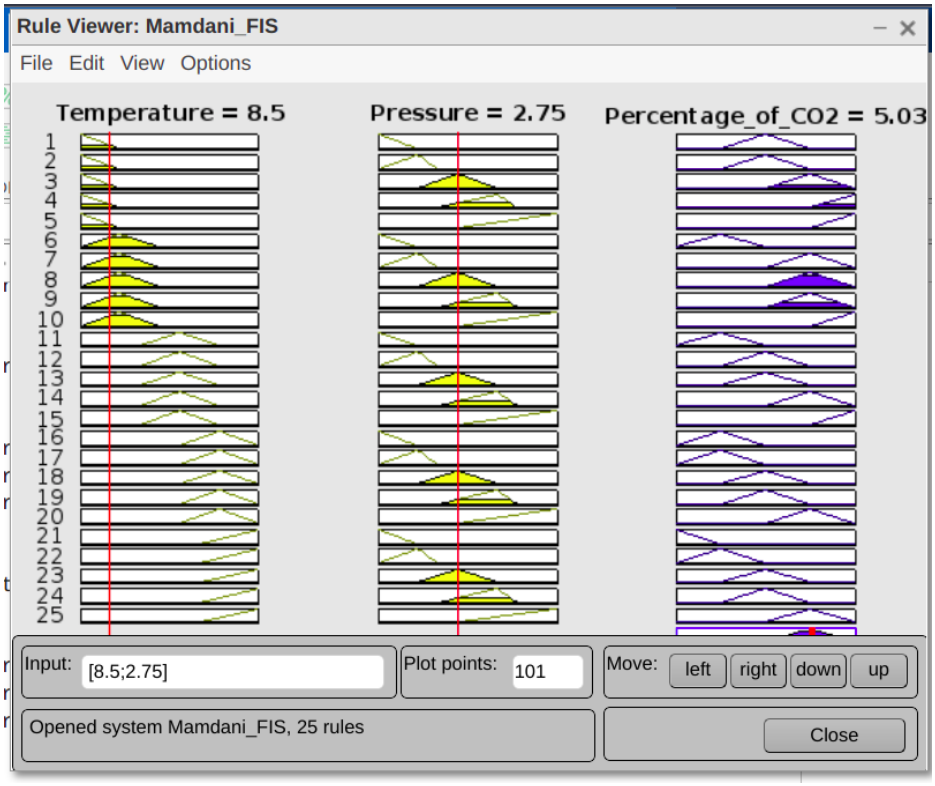
- Input 1:



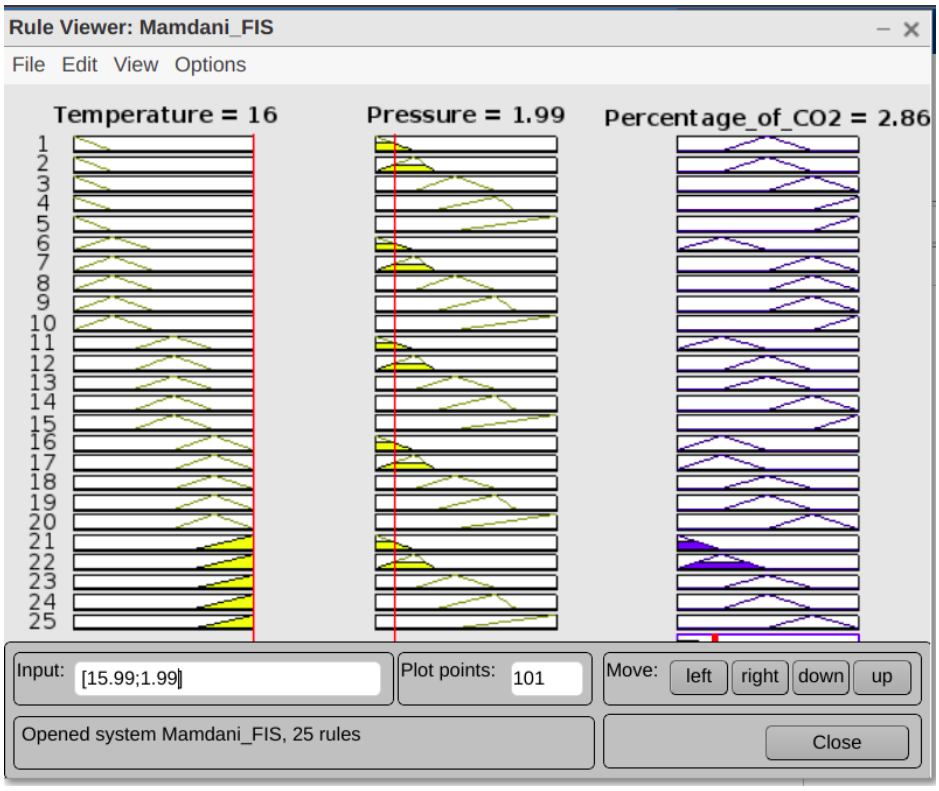
- Input 2:



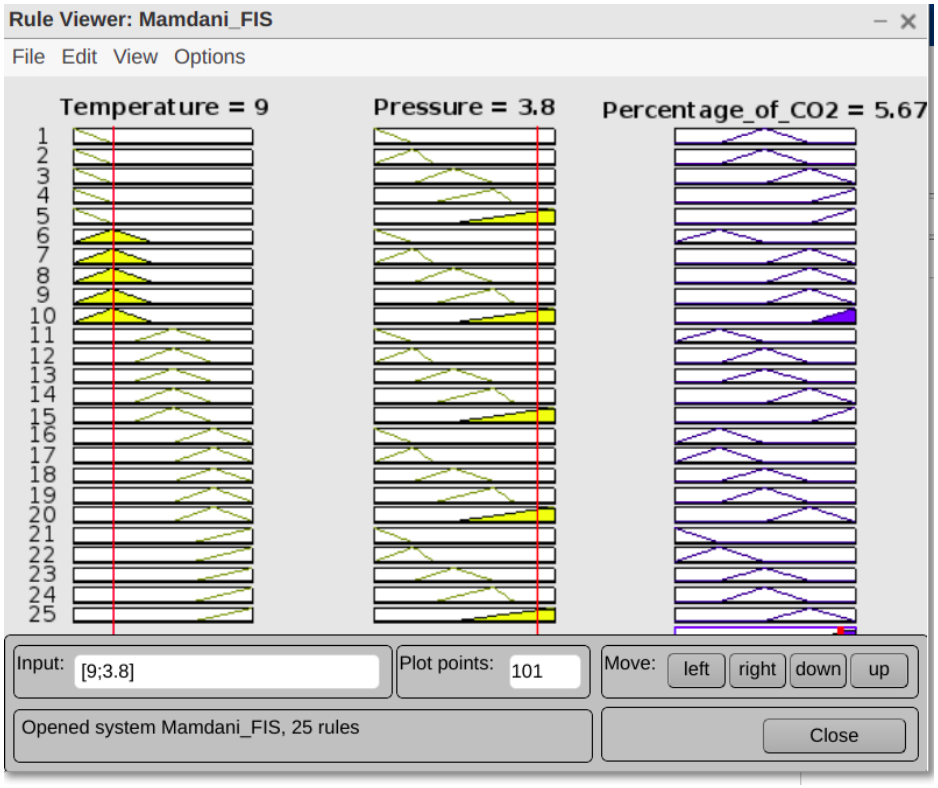
- Input 3



- Input 4



- Input 5



5. Code

- **Code**

```
%Project
fis = mamfis('Name',"Project");

fis = addInput(fis,[1.75 4],'Name',"Pressure");
fis = addInput(fis,[7 16],'Name',"Temperature");
fis = addOutput(fis,[2 6],'Name',"Percentage of CO2");

fis = addMF(fis,"Temperature","trimf",[7 7 9],'Name',"Very Cold");
fis = addMF(fis,"Temperature","trimf",[7 9 11],'Name',"Cold");
fis = addMF(fis,"Temperature","trimf",[10 12 14],'Name',"Normal");
fis = addMF(fis,"Temperature","trimf",[12 14 16],'Name',"Hot");
fis = addMF(fis,"Temperature","trimf",[13 16 16],'Name',"Very Hot");

fis = addMF(fis,"Pressure","trimf",[1.75 1.75 2.25],'Name',"Very
Bad");
fis = addMF(fis,"Pressure","trimf",[1.75 2.25 2.50],'Name',"Bad");
fis = addMF(fis,"Pressure","trimf",[2.25 2.75 3.25],'Name',"Normal");
fis = addMF(fis,"Pressure","trimf",[2.50 3.25 3.50],'Name',"Good");
fis = addMF(fis,"Pressure","trimf",[2.75 4.00 4.00],'Name',"Very
Good");

fis = addMF(fis,"Percentage of CO2","trimf",[2 2 3],'Name',"Very
Bad");
fis = addMF(fis,"Percentage of CO2","trimf",[2 3 4],'Name',"Bad");
fis = addMF(fis,"Percentage of CO2","trimf",[3 4 5],'Name',"Normal");
fis = addMF(fis,"Percentage of CO2","trimf",[4 5 6],'Name',"Good");
fis = addMF(fis,"Percentage of CO2","trimf",[5 6 6],'Name',"Very
Good");
```

```
ruleList = [1 1 3 1 1;
                1 2 2 1 1;
                1 3 2 1 1;
                1 4 2 1 1;
                1 5 1 1 1;
                2 1 3 1 1;
                2 2 4 1 1;
                2 3 3 1 1;
                2 4 2 1 1;
                2 5 2 1 1;
                3 1 4 1 1;
                3 2 4 1 1;
                3 3 3 1 1;
                3 4 3 1 1;
                3 5 3 1 1;
                4 1 5 1 1;
                4 2 4 1 1;
                4 3 4 1 1;
                4 4 3 1 1;
                4 5 3 1 1;
                5 1 5 1 1;
                5 2 5 1 1;
                5 3 5 1 1;
                5 4 4 1 1;
                5 5 4 1 1;
];
fis = addRule(fis,ruleList);
```