ANKARA UNIVERSITY

COM2536

FUZZY LOGIC

FUZZY INFERENCE SYSTEM

ENES OSMANOĞLU – 19290319

İREM PEKKIYAK – 19291039

MOHIELDIN AHMED – 20290028

Abstract

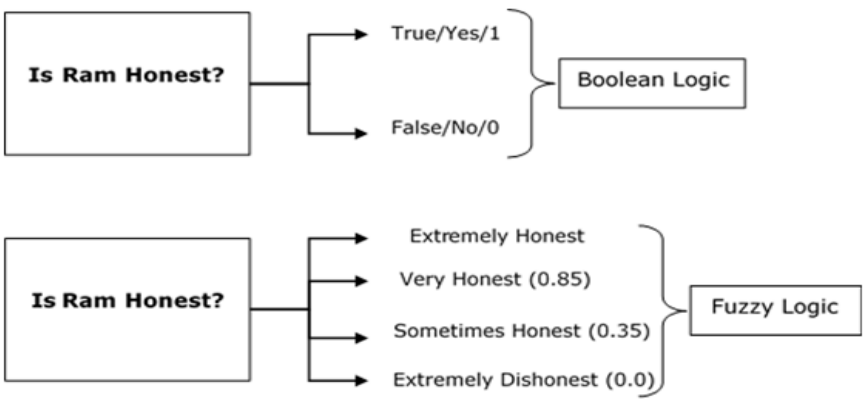
The word fuzzy refers to things which are not clear or are vague. Any event, process, or function that is changing continuously cannot always be defined as either true or false, which means that we need to define such activities in a Fuzzy manner. In this project we are going to develop a fuzzy logic system that determines the percentage of carbon dioxide in a specific Fizzy drink according to the given two parameters, temperature and pressure. Using temperature and pressure values, the case of carbon dioxide must be determined, whether it is very bad, bad, normal, good, and very good. According to case of those two parameters temperature and pressure, where pressure values are same as carbon dioxide values and temperature values are specify, whether it is very cold, cold, normal, hot, and very hot. The system is implemented using MATLAB programming language.

Introduction

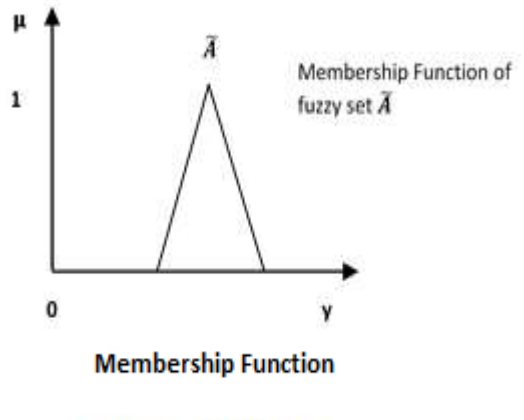
WHAT IS FUZZY LOGIC?

Fuzzy Logic is a computing approach based on "Degree of Truth" and is not limited to Boolean "true or false." The term 'Fuzzy' means something vague or not very clear. The fuzzy Logic system is applied to scenarios where it is difficult to categorize states as a binary "True or False." In computer science uses the idea in which human brain thinks and solves problems, The human brain can reason with uncertainties and judgments. Computers can only manipulate precise valuations. Fuzzy logic is an attempt to combine the two techniques.

The following diagram shows that in fuzzy systems, the values are indicated by a number in the range from 0 to 1. Here 1.0 represents **absolute truth** and 0.0 represents **absolute falseness**. The number which indicates the value in fuzzy systems is called the **truth value**.



Fuzzy logic was introduced in the 1930s by Jan Lukasiewicz, a Polish philosopher. While classical logic operates with only two values 1 (true) and 0 (false), Lukasiewicz introduced logic that extended the range of truth values to all real numbers in the interval between 0 and 1. He used a number in this interval to represent the possibility that a given statement was true or false.



Membership Function

Membership functions were first introduced in 1965 by Lofti A. Zadeh in his first research paper "fuzzy sets". Membership functions characterize fuzziness (i.e., all the information in fuzzy set), whether the elements in fuzzy sets are discrete or continuous. Membership functions can be defined as a technique to solve practical problems by experience rather than knowledge. Membership functions are represented by graphical forms. Rules for defining fuzziness are fuzzy too.

Fuzzy Inference

Fuzzy Inference System is the key unit of a fuzzy logic system having decision making as its primary work. It uses the "IF…THEN" rules along with connectors "OR" or "AND" for drawing essential decision rules.

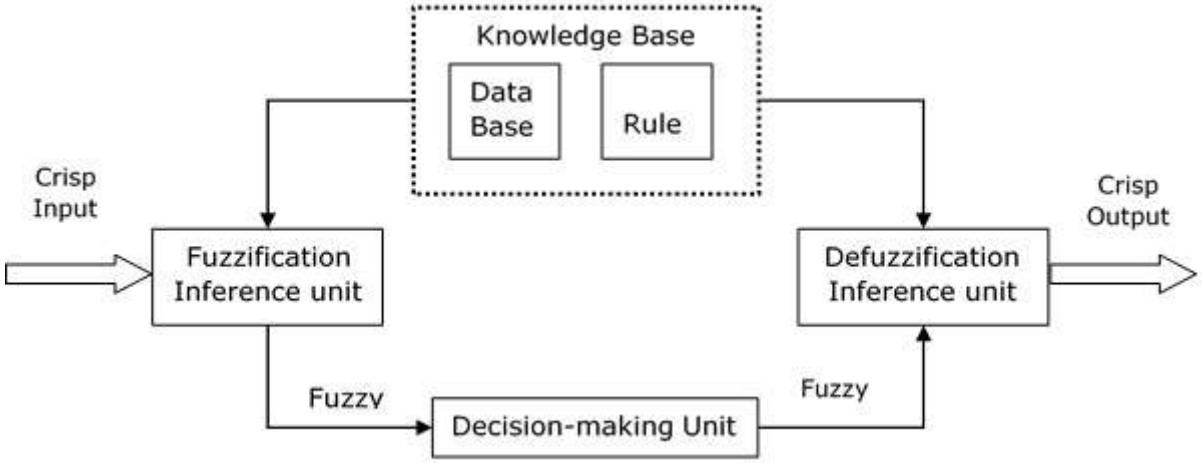
CHARACTERISTICS OF FUZZY INFERENCE SYSTEM

Following are some characteristics of FIS –

- The output from FIS is always a fuzzy set irrespective of its input which can be fuzzy or crisp.
- It is necessary to have fuzzy output when it is used as a controller.
- A defuzzification unit would be there with FIS to convert fuzzy variables into crisp variables.

FUNCTIONAL BLOCKS OF FIS

- **Rule Base** – It contains fuzzy IF-THEN rules.
- **Database** – It defines the membership functions of fuzzy sets used in fuzzy rules.
- **Decision-making Unit** – It performs operation on rules.
- **Fuzzification Interface Unit** – It converts the crisp quantities into fuzzy quantities.
- **Defuzzification Interface Unit** – It converts the fuzzy quantities into crisp quantities. Following is a block diagram of fuzzy interference system.



WORKING OF FIS

- A fuzzification unit supports the application of numerous fuzzification methods and converts the crisp input into fuzzy input.
- A knowledge base - collection of rule base and database is formed upon the conversion of crisp input into fuzzy input.
- The defuzzification unit fuzzy input is finally converted into crisp output.

MAMDANI FUZZY INFERENCE SYSTEM

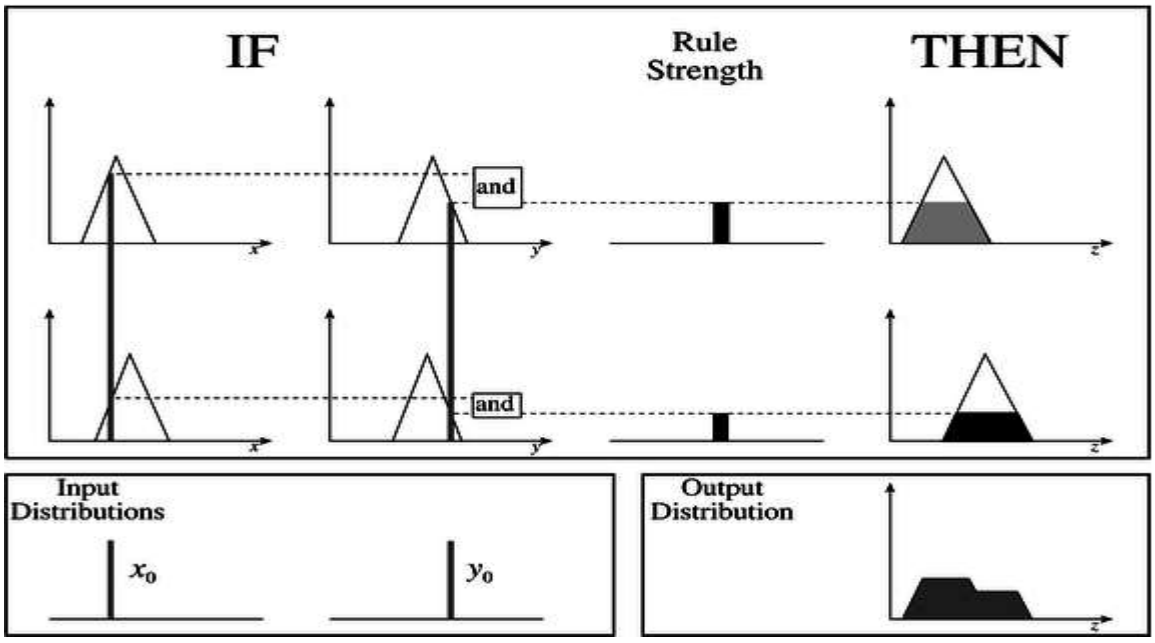
This system was proposed in 1975 by Ebhasim Mamdani. Basically, it was anticipated to control a steam engine and boiler combination by synthesizing a set of fuzzy rules obtained from people working on the system.

STEPS FOR COMPUTING THE OUTPUT

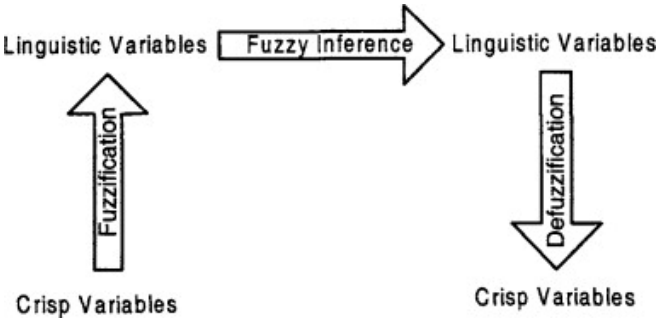
Following steps need to be followed to compute the output from this FIS

- **Step 1** – Set of fuzzy rules need to be determined in this step.
- **Step 2** – In this step, by using 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 consequent of rule by combining the rule strength and the output membership function.
- **Step 5** – For getting output distribution combine all the consequents.
- **Step 6** – Finally, a defuzzified output distribution is obtained.

Mamdani Fuzzy Interface System



Center of Area (CoA)



Defuzzification method is the center of area method (COA), also commonly referred to as the *centroid* method. This method determines the center of area of fuzzy set and returns the corresponding crisp value. The center of sums (COS) method and the mean of maximum method are two alternative methods in defuzzification. Where CoA is the center of area, x is the value of the linguistic variable, and xmin and xmax represent the range of the linguistic variable. The Center of Area defuzzification method effectively calculates the best compromise between multiple output linguistic terms.

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

Application:

AUTOMOTIVE

- Intelligent highway systems
- Traffic control

BUSINESS

- Decision-making support systems
- Personnel evaluation in a large company

ELECTRONICS

- Air conditioning systems
- Washing machine timing

FINANCE

- Banknote transfer control

INDUSTRIAL SECTOR

- Control of water purification plants

MANUFACTURING

- Optimization of milk production

MEDICAL

- Multivariable control of anesthesia
- Radiology diagnoses

SECURITIES

- Various security appliances

TRANSPORTATION

- Railway acceleration
- Braking and stopping

PATTERN RECOGNITION AND CLASSIFICATION

- Handwriting recognition
- Fuzzy image search

PSYCHOLOGY

- Fuzzy logic based analysis of human behavior

Construction

- To Create Fuzzy System -

mamfis

- Creates a Mamdani Fuzzy Inference System Object with default property values. The output of each rule is a fuzzy set.

  Syntax:
  - fis = mamfis
  - fis = mamfis(Name,Value)

  We used it as:

```
fis = mamfis(Name="Fizzy Drink");
```

- To Specify Input and Output Variables -

addInput

- Adds an input variable to the fuzzy inference system.

  Syntax:
  - *fisOut = addInput(fisIn)*
  - *fisOut = addInput(fisIn,range)*
  - *fisOut = addInput(___,Name,Value)*

  We used it as:

```
fis = addInput(fis,[7 16],Name="Temperature");
fis = addInput(fis,[1.75,4.0],Name="Pressure");
```

addOutput

- Adds an output variable to the fuzzy inference system.

  Syntax:
  - *fisOut = addOutput(fisIn)*
  - *fisOut = addOutput(fisIn,range)*
  - *fisOut = addOutput(___,Name,Value)*

  We used it as:

```
fis = addOutput(fis,[2.0,6.0],Name="Carbondioxide");
```

- To Specify Membership Functions -

addMF

- Adds a membership function to a fuzzy variable.

  Syntax:
  - *fisOut = addMF(fisIn,varName)*
  - *fisOut = addMF(fisIn,varName,type,parameters)*
  - *fisOut = addMF(___,Name,Value)*

We used it as:

```
fis = addMF(fis,"Temperature","trimf",[7,7,9],Name="Verycold");
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="Veryhot");
```

```
fis = addMF(fis,"Pressure","trimf",[1.75,1.75,2.25],Name="Verybad");
fis = addMF(fis,"Pressure","trimf",[1.75,2.25,2.5],Name="Bad");
fis = addMF(fis,"Pressure","trimf",[2.25,2.75,3.25],Name="Normal");
fis = addMF(fis,"Pressure","trimf",[2.5,3.25,3.5],Name="Good");
fis = addMF(fis,"Pressure","trimf",[2.75,4.0,4.0],Name="Verygood");
```

```
fis = addMF(fis,"Carbondioxide","trimf",[2.0,2.0,3.0],Name="Verybad");
fis = addMF(fis,"Carbondioxide","trimf",[2.0,3.0,4.0],Name="Bad");
fis = addMF(fis,"Carbondioxide","trimf",[3.0,4.0,5.0],Name="Normal");
fis = addMF(fis,"Carbondioxide","trimf",[4.0,5.0,6.0],Name="Good");
fis = addMF(fis,"Carbondioxide","trimf",[5.0,6.0,6.0],Name="Verygood");
```

- To Specify Fuzzy Rules -

The rules are specified for the fuzzy system and stored in variables such as *rule1*, *rule2*, ..., *rule25*.

```
rule1 = "If Temperature is Verycold and Pressure is Verybad then Carbondioxide is Normal";
rule2 = "If Temperature is Verycold and Pressure is Bad then Carbondioxide is Normal";
rule3 = "If Temperature is Verycold and Pressure is Normal then Carbondioxide is Good";
rule4 = "If Temperature is Verycold and Pressure is Good then Carbondioxide is Verygood";
rule5 = "If Temperature is Verycold and Pressure is Verygood then Carbondioxide is Verygood";
rule6 = "If Temperature is Cold and Pressure is Verybad then Carbondioxide is Bad";
rule7 = "If Temperature is Cold and Pressure is Bad then Carbondioxide is Good";
rule8 = "If Temperature is Cold and Pressure is Normal then Carbondioxide is Good";
rule9 = "If Temperature is Cold and Pressure is Good then Carbondioxide is Good";
rule10 = "If Temperature is Cold and Pressure is Verygood then Carbondioxide is Verygood";
rule11 = "If Temperature is Normal and Pressure is Verybad then Carbondioxide is Bad";
rule12 = "If Temperature is Normal and Pressure is Bad then Carbondioxide is Normal";
rule13 = "If Temperature is Normal and Pressure is Normal then Carbondioxide is Normal";
rule14 = "If Temperature is Normal and Pressure is Good then Carbondioxide is Good";
rule15 = "If Temperature is Normal and Pressure is Verygood then Carbondioxide is Verygood";
rule16 = "If Temperature is Hot and Pressure is Verybad then Carbondioxide is Bad";
rule17 = "If Temperature is Hot and Pressure is Bad then Carbondioxide is Bad";
rule18 = "If Temperature is Hot and Pressure is Normal then Carbondioxide is Normal";
```

```
rule19 = "If Temperature is Hot and Pressure is Good then Carbondioxide is Normal";
rule20 = "If Temperature is Hot and Pressure is Verygood then Carbondioxide is Good";
rule21 = "If Temperature is Veryhot and Pressure is Verybad then Carbondioxide is Verybad";
rule22 = "If Temperature is Veryhot and Pressure is Bad then Carbondioxide is Bad";
rule23 = "If Temperature is Veryhot and Pressure is Normal then Carbondioxide is Normal";
rule24 = "If Temperature is Veryhot and Pressure is Good then Carbondioxide is Normal";
rule25 = "If Temperature is Veryhot and Pressure is Verygood then Carbondioxide is Good";
```

Then, all the variables are stored in a "rules" named array.

```
rules = [
    rule1 rule2 rule3 rule4 rule5
    rule6 rule7 rule8 rule9 rule10
    rule11 rule12 rule13 rule14 rule15
    rule16 rule17 rule18 rule19 rule20
    rule21 rule22 rule23 rule24 rule25
];
```

addRule

- Adds a rule or rules (as an array) to the fuzzy inference system.

Syntax:

- *fisOut = addRule(fisIn)*
- *fisOut = addRule(fisIn,ruleDescription)*

We used it as:

```
fis = addRule(fis,rules);
```

- To Evaluate Fuzzy System -

evalfis

- Evaluates the fuzzy inference system and returns intermediate results from the fuzzy inference process.

Syntax:

- *output = evalfis(fis,input)*
- *output = evalfis(fis,input,options)*
- *[output,fuzzifiedIn,ruleOut,aggregatedOut,ruleFiring] = evalfis(___)*

To use it we set an input firstly.

```
input = [8.25,3.25];
```

Then we use the evalfis as:

```
[~,~,~,aggregatedOut,~] = evalfis(fis,input);
```

We will use only *aggregatedOut* argument so the symbol ~ is used to determine unused arguments in the code.

But what is the *aggregatedOut* ?

- Aggregated output for each output variable, returned as an array.
- For each output variable, *evalfis* combines the corresponding outputs from all the rules using the aggregation method specified in fis.
- For a Mamdani system, the aggregate result for each output variable is a fuzzy set. In this case, *aggregatedOut* is as an N_S-by-N_Y array, where N_Y is the number of outputs and N_S is the number of sample points used for evaluating output variable ranges. Each column of *aggregatedOut* contains the aggregate fuzzy set for one output variable.
- If input specifies multiple input combinations, then *aggregatedOut* corresponds to the combination in the last row of input.

defuzz

➢ Returns the defuzzified output value for a membership function *(mf)* at the variable values *(x)* using a specified defuzzification method.

Syntax:

- *output = defuzz(x,mf,method)*
  - *x : Variable values.*
  - *mf : Membership function values, specified as a vector with the same length as x. Each element of mf contains a fuzzy membership value for the corresponding variable value in x.*
  - *method : Defuzzification method, specified as one of the following:*
    - ✓ *'centroid' — Centroid of the area under the output fuzzy set*
    - ✓ *'bisector' — Bisector of the area under the output fuzzy set*
    - ✓ *'mom' — Mean of the values for which the output fuzzy set is maximum*
    - ✓ *'lom' — Largest value for which the output fuzzy set is maximum*
    - ✓ *'som' — Smallest value for which the output fuzzy set is maximum*

To use *defuzz*, we created *x* and *mf* variables firstly. For *x*, we used *linspace*. For *mf*, we used the *aggregatedOut* directly. And, we selected *method* as *'centroid'*.

linspace

➢ Generates linearly spaced vector.

Syntax:

- *y = linspace(x1,x2,n)*
  - ✓ *It generates n points. The spacing between the points is (x2-x1)/(n-1).*

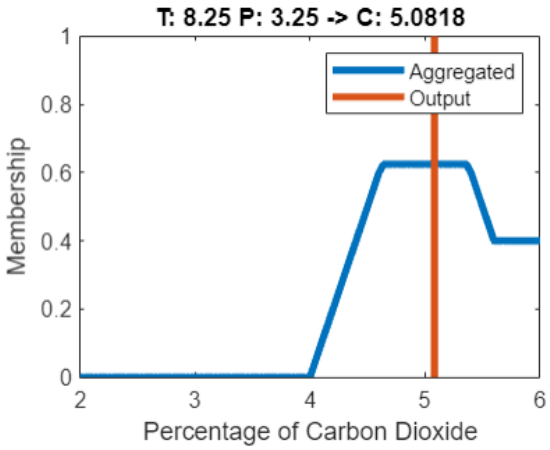
```
mfRange = fis.output.Range;
x1 = mfRange(1);
x2 = mfRange(2);
n = length(aggregatedOut);

x = linspace(x1,x2,n);
mf = aggregatedOut;
output = defuzz(x,mf,'centroid');
```

Finally, we plot the graphic using *plot* function for the *aggregated output* and the *defuzzified output*.

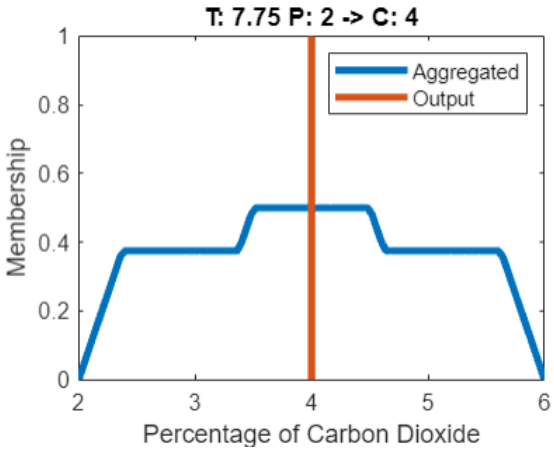
```
plot(x,mf,[output output],[0 1],LineWidth=3);
xlabel('Percentage of Carbon Dioxide');
ylabel('Membership');
title(['T: ' num2str(input(1)) ' P: ' num2str(input(2)) ' -> C: ' num2str(output)]);
legend('Aggregated','Output');
```

Implementation

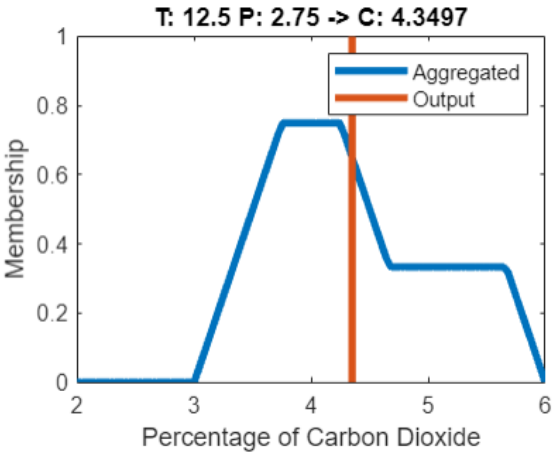
- Inputs:
  - ✓ Temperature: **8.25**
  - ✓ Pressure: **3.25**
- Output:
  - ✓ Percentage of Carbon Dioxide: **5.0818**



- Inputs:
  - ✓ Temperature: **7.75**
  - ✓ Pressure: **2**
- Output:
  - ✓ Percentage of Carbon Dioxide: **4**



- Inputs:
  - ✓ Temperature: **12.5**
  - ✓ Pressure: **2.75**
- Output:
  - ✓ Percentage of Carbon Dioxide: **4.3497**



FULL SOURCE CODE

```
fis = mamfis(Name="Fizzy Drink");

fis = addInput(fis,[7 16],Name="Temperature");
fis = addMF(fis,"Temperature","trimf",[7,7,9],Name="Verycold");
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="Veryhot");

fis = addInput(fis,[1.75,4.0],Name="Pressure");
fis = addMF(fis,"Pressure","trimf",[1.75,1.75,2.25],Name="Verybad");
fis = addMF(fis,"Pressure","trimf",[1.75,2.25,2.5],Name="Bad");
fis = addMF(fis,"Pressure","trimf",[2.25,2.75,3.25],Name="Normal");
fis = addMF(fis,"Pressure","trimf",[2.5,3.25,3.5],Name="Good");
fis = addMF(fis,"Pressure","trimf",[2.75,4.0,4.0],Name="Verygood");

fis = addOutput(fis,[2.0,6.0],Name="Carbondioxide");
fis = addMF(fis,"Carbondioxide","trimf",[2.0,2.0,3.0],Name="Verybad");
fis = addMF(fis,"Carbondioxide","trimf",[2.0,3.0,4.0],Name="Bad");
fis = addMF(fis,"Carbondioxide","trimf",[3.0,4.0,5.0],Name="Normal");
fis = addMF(fis,"Carbondioxide","trimf",[4.0,5.0,6.0],Name="Good");
fis = addMF(fis,"Carbondioxide","trimf",[5.0,6.0,6.0],Name="Verygood");

rule1 = "If Temperature is Verycold and Pressure is Verybad then Carbondioxide is Normal";
rule2 = "If Temperature is Verycold and Pressure is Bad then Carbondioxide is Normal";
rule3 = "If Temperature is Verycold and Pressure is Normal then Carbondioxide is Good";
rule4 = "If Temperature is Verycold and Pressure is Good then Carbondioxide is Verygood";
rule5 = "If Temperature is Verycold and Pressure is Verygood then Carbondioxide is Verygood";
rule6 = "If Temperature is Cold and Pressure is Verybad then Carbondioxide is Bad";
rule7 = "If Temperature is Cold and Pressure is Bad then Carbondioxide is Good";
rule8 = "If Temperature is Cold and Pressure is Normal then Carbondioxide is Good";
rule9 = "If Temperature is Cold and Pressure is Good then Carbondioxide is Good";
rule10 = "If Temperature is Cold and Pressure is Verygood then Carbondioxide is Verygood";
rule11 = "If Temperature is Normal and Pressure is Verybad then Carbondioxide is Bad";
rule12 = "If Temperature is Normal and Pressure is Bad then Carbondioxide is Normal";
rule13 = "If Temperature is Normal and Pressure is Normal then Carbondioxide is Normal";
rule14 = "If Temperature is Normal and Pressure is Good then Carbondioxide is Good";
rule15 = "If Temperature is Normal and Pressure is Verygood then Carbondioxide is Verygood";
rule16 = "If Temperature is Hot and Pressure is Verybad then Carbondioxide is Bad";
```

```
rule17 = "If Temperature is Hot and Pressure is Bad then Carbondioxide is Bad";
rule18 = "If Temperature is Hot and Pressure is Normal then Carbondioxide is Normal";
rule19 = "If Temperature is Hot and Pressure is Good then Carbondioxide is Normal";
rule20 = "If Temperature is Hot and Pressure is Verygood then Carbondioxide is Good";
rule21 = "If Temperature is Veryhot and Pressure is Verybad then Carbondioxide is Verybad";
rule22 = "If Temperature is Veryhot and Pressure is Bad then Carbondioxide is Bad";
rule23 = "If Temperature is Veryhot and Pressure is Normal then Carbondioxide is Normal";
rule24 = "If Temperature is Veryhot and Pressure is Good then Carbondioxide is Normal";
rule25 = "If Temperature is Veryhot and Pressure is Verygood then Carbondioxide is Good";

rules = [
    rule1 rule2 rule3 rule4 rule5
    rule6 rule7 rule8 rule9 rule10
    rule11 rule12 rule13 rule14 rule15
    rule16 rule17 rule18 rule19 rule20
    rule21 rule22 rule23 rule24 rule25
];

fis = addRule(fis,rules);

input = [9,3.25];
[~,~,~,aggregatedOut,~] = evalfis(fis,input);

mfRange = fis.output.Range;
x1 = mfRange(1);
x2 = mfRange(2);
n = length(aggregatedOut);

x = linspace(x1,x2,n);
mf = aggregatedOut;
output = defuzz(x,mf,'centroid');

plot(x,mf,[output output],[0 1],LineWidth=3);
xlabel('Percentage of Carbon Dioxide');
ylabel('Membership');
title(['T: ' num2str(input(1)) ' P: ' num2str(input(2)) ' -> C: ' num2str(output)]);
legend('Aggregated','Output');
```

REFERENCES

- ✓ https://www.mathworks.com/help/fuzzy/foundations-of-fuzzy-logic.html?searchHighlight=fuzzy%20logic&s_tid=srchtitle_fuzzy%20logic_2

- ✓ https://www.cs.princeton.edu/courses/archive/fall07/cos436/HIDDEN/Knapp/fuzzy004.htm

- ✓ https://www.guru99.com/what-is-fuzzy-logic.html#10

- ✓ https://www.logic.at/tbilisi05/Metcalfe-notes.pdf

- ✓ http://i-rep.emu.edu.tr:8080/jspui/bitstream/11129/1752/1/MusaIsmail.pdf