PROBLEM 1

We are given a dynamic process guided by the fuzzy logic control system with the following two fuzzy control rules :-

Rule 1 If x is A_1 and y is B_1 Then z is C_1

Rule 2 If x is A_2 and y is B_2 Then z is C_2

Where x_0 and y_0 are the sensor readings for the linguistic input variables x and y and z is the consequent linguistic variables. The fuzzy predicates for the linguistic variables are given by A_1, A_2, B_1, B_2, C_1 and C_2, which membership functions are as:

$$\mu_{A_1}(x) = \begin{cases} \dfrac{x-2}{3} & 2 \le x \le 5 \\ \dfrac{8-x}{3} & 5 < x \le 8 \end{cases}$$

$\rightarrow$ putting $x = 2, 3, 4, 5$ in the equation $\frac{x-2}{3}$

for $x = 2 \rightarrow \mu_{A_1}(x) = 0$

for $x = 3 \rightarrow \mu_{A_1}(x) = 1/3$

for $x = 4 \rightarrow \mu_{A_1}(x) = 2/3$

for $x = 5 \rightarrow \mu_{A_1}(x) = 1$

putting $x = 6, 7, 8$ in the equation $\frac{8-x}{5}$

for $x = 6 \rightarrow \mu_{A_1}(x) = 2/3$

for $x = 7 \rightarrow \mu_{A_1}(x) = 1/3$

for $x = 8 \rightarrow \mu_{A_1}(x) = 0$

(for graph, refer figure A)

$$\mu_{B_1}(y) = \begin{cases} \frac{y-5}{3} & 5 \le y \le 8 \\ \frac{11-y}{3} & 8 < y \le 11 \end{cases}$$

putting $y = 5, 6, 7, 8$ in the equation $\frac{y-5}{3}$

for $y = 5 \rightarrow \mu_{B_1}(y) = 0$
for $y = 6 \rightarrow \mu_{B_1}(y) = 1/3$
for $y = 7 \rightarrow \mu_{B_1}(y) = 2/3$
for $y = 8 \rightarrow \mu_{B_1}(y) = 1$

putting $y = 9, 10, 11$ in the equation $\frac{11-y}{3}$

for $y = 9 \rightarrow \mu_{B_1}(y) = 2/3$
for $y = 10 \rightarrow \mu_{B_1}(y) = 1/3$
for $y = 11 \rightarrow \mu_{B_1}(y) = 0$

(for graph, refer figure B)

$$\mu_{C_1}(z) = \begin{cases} \frac{z-1}{3} & 1 \le z \le 4 \\ \frac{7-z}{3} & 4 < z \le 7 \end{cases}$$

putting $z = 1, 2, 3, 4$ ~~putting~~ in the equation $\frac{z-1}{3}$

for $z = 1 \rightarrow \mu_{C_1}(z) = 0$
for $z = 2 \rightarrow \mu_{C_1}(z) = 1/3$
for $z = 3 \rightarrow \mu_{C_1}(z) = 2/3$
for $z = 4 \rightarrow \mu_{C_1}(z) = 1$

Date:

putting $z = 5, 6, 7$ in the equation $\frac{7-z}{3}$

for $z = 5 \rightarrow \mu_{c_1}(z) = 2/3$
for $z = 6 \rightarrow \mu_{c_1}(z) = 1/3$
for $z = 7 \rightarrow \mu_{c_1}(z) = 0$

(for graph. refer figure c)

$$\mu_{A_2}(x) = \begin{cases} \frac{x-3}{3} & 3 \le x \le 6 \\ \frac{9-x}{3} & 6 < x \le 9 \end{cases}$$

putting $x = 3, 4, 5, 6$ in the equation $\frac{x-3}{3}$

for $x = 3 \rightarrow \mu_{A_2}(x) = 0$
for $x = 4 \rightarrow \mu_{A_2}(x) = 1/2$
for $x = 5 \rightarrow \mu_{A_2}(x) = 2/3$
for $x = 6 \rightarrow \mu_{A_2}(x) = 1$

putting $x = 7, 8, 9$ in the equation $\frac{9-x}{3}$

for $x = 7 \rightarrow \mu_{A_2}(x) = 2/3$
for $x = 8 \rightarrow \mu_{A_2}(x) = 1/3$
for $x = 9 \rightarrow \mu_{A_2}(x) = 0$

(for graph, refer figure D)

$$\mu_{B_2}(y) = \begin{cases} \dfrac{y-4}{3} & 4 \le y \le 7 \\ \dfrac{10-y}{3} & 7 < y \le 10 \end{cases}$$

putting $y = 4, 5, 6, 7$ in the equation $\frac{y-4}{3}$

- for $y = 4 \longrightarrow \mu_{B_2}(y) = 0$
- for $y = 5 \longrightarrow \mu_{B_2}(y) = 1/3$
- for $y = 6 \longrightarrow \mu_{B_2}(y) = 2/3$
- for $y = 7 \longrightarrow \mu_{B_2}(y) = 1$

putting $y = 8, 9, 10$ in the equation $\frac{10-y}{3}$

- for $y = 8 \longrightarrow \mu_{B_2}(y) = 2/3$
- for $y = 9 \longrightarrow \mu_{B_2}(y) = 1/3$
- for $y = 10 \longrightarrow \mu_{B_2}(y) = 0$

(for graph, refer figure E)

$$\mu_{C_2}(z) = \begin{cases} \dfrac{z-3}{3} & 3 \le z \le 6 \\ \dfrac{9-z}{3} & 6 \le z \le 9 \end{cases}$$

putting $z = 3, 4, 5, 6$ in the equation $\frac{z-3}{3}$

- for $z = 3 \longrightarrow \mu_{C_2}(z) = 0$
- for $z = 4 \longrightarrow \mu_{C_2}(z) = 1/3$
- for $z = 5 \longrightarrow \mu_{C_2}(z) = 2/3$
- for $z = 6 \longrightarrow \mu_{C_2}(z) = 1$

putting $z = 7, 8, 9$ in the equation $\frac{9-z}{3}$

for $z = 7 \rightarrow \mu_{C2}(z) = 2/3$

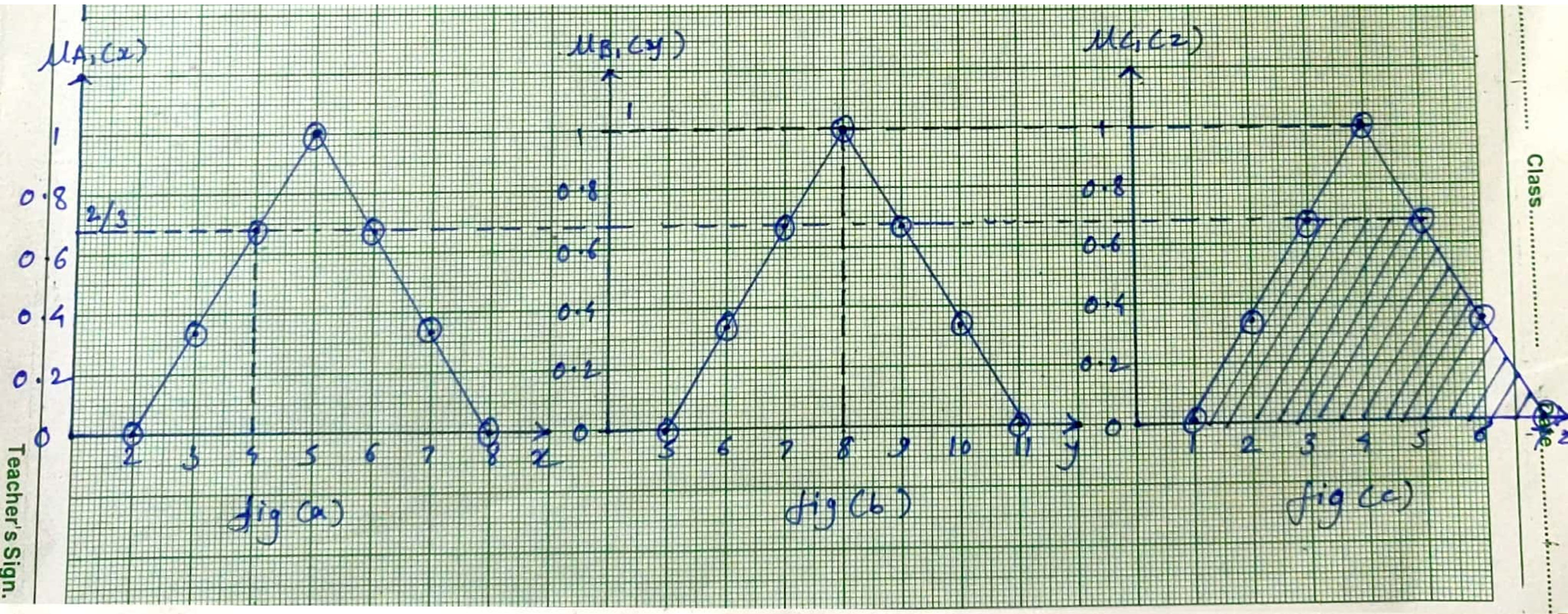
for $z = 8 \rightarrow \mu_{C2}(z) = 1/3$

for $z = 9 \rightarrow \mu_{C2}(z) = 0$

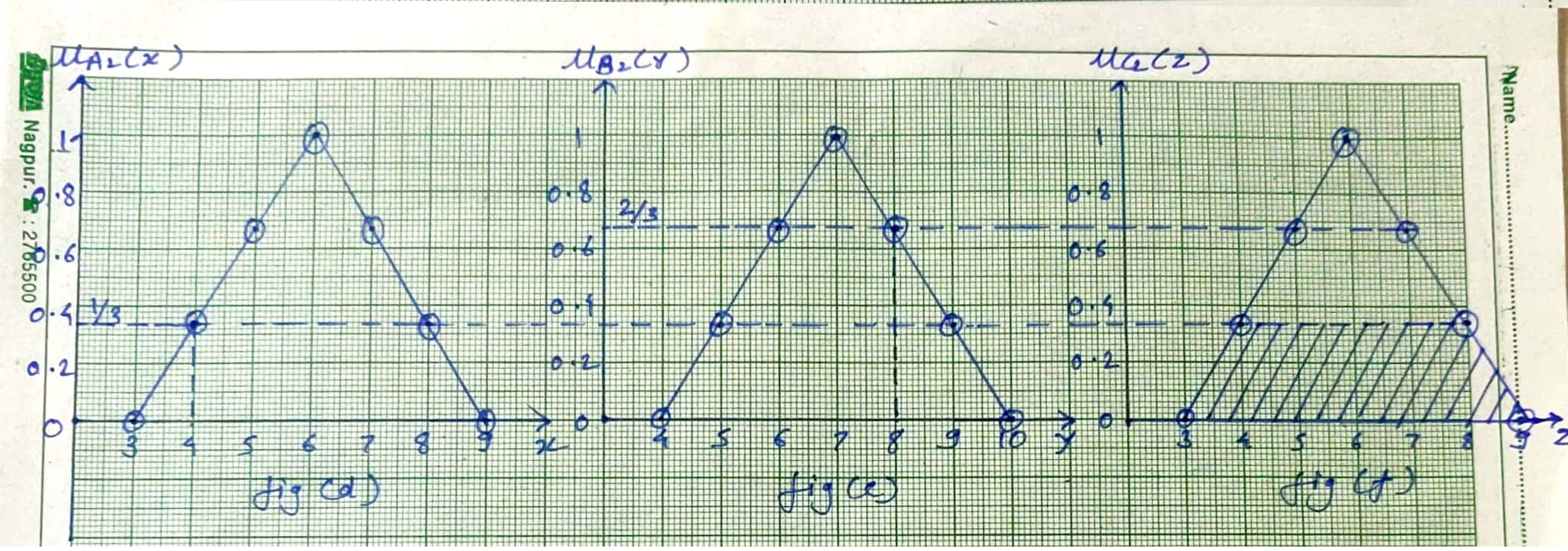
(for graph, refer figure F)

Therefore, if we use Mean of Maximum (MOM) defuzzification strategy, the output is 4

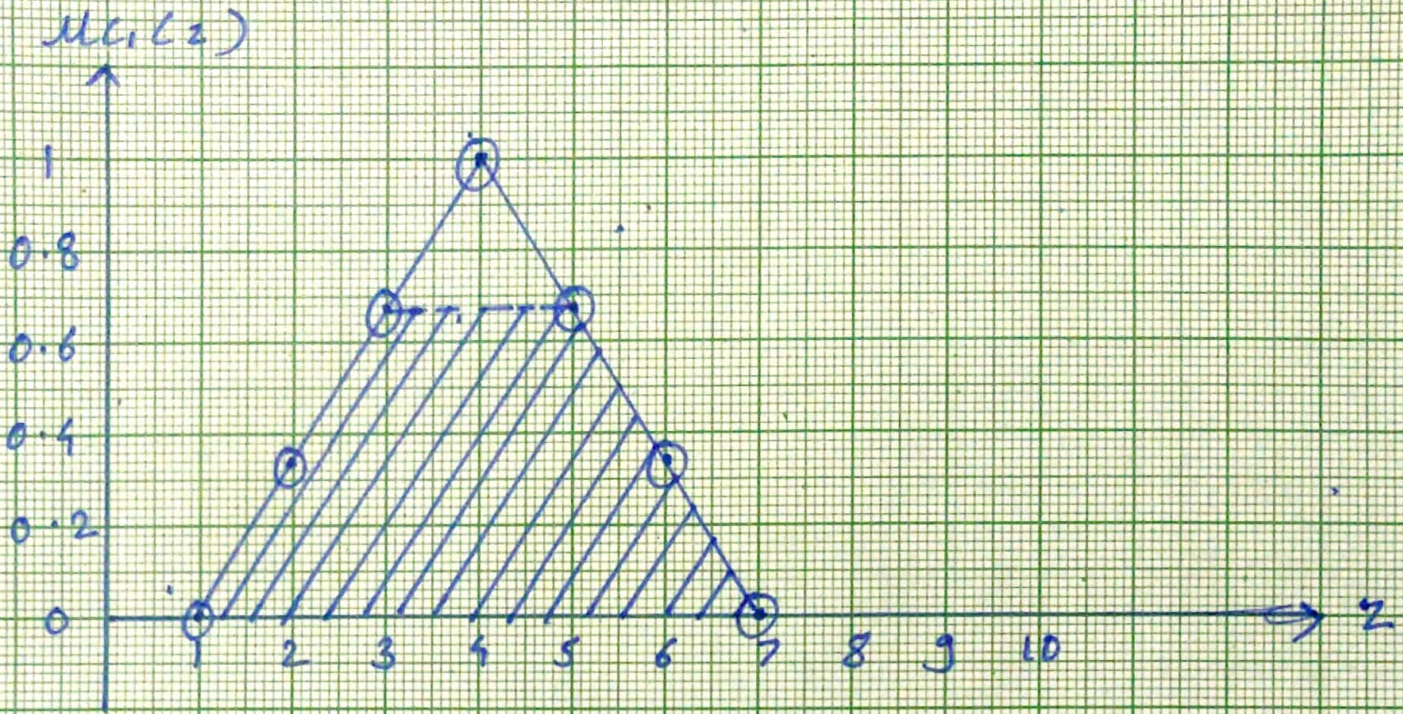
If we use the largest of maximum (lom) deffuzikation strategy, the output is 5

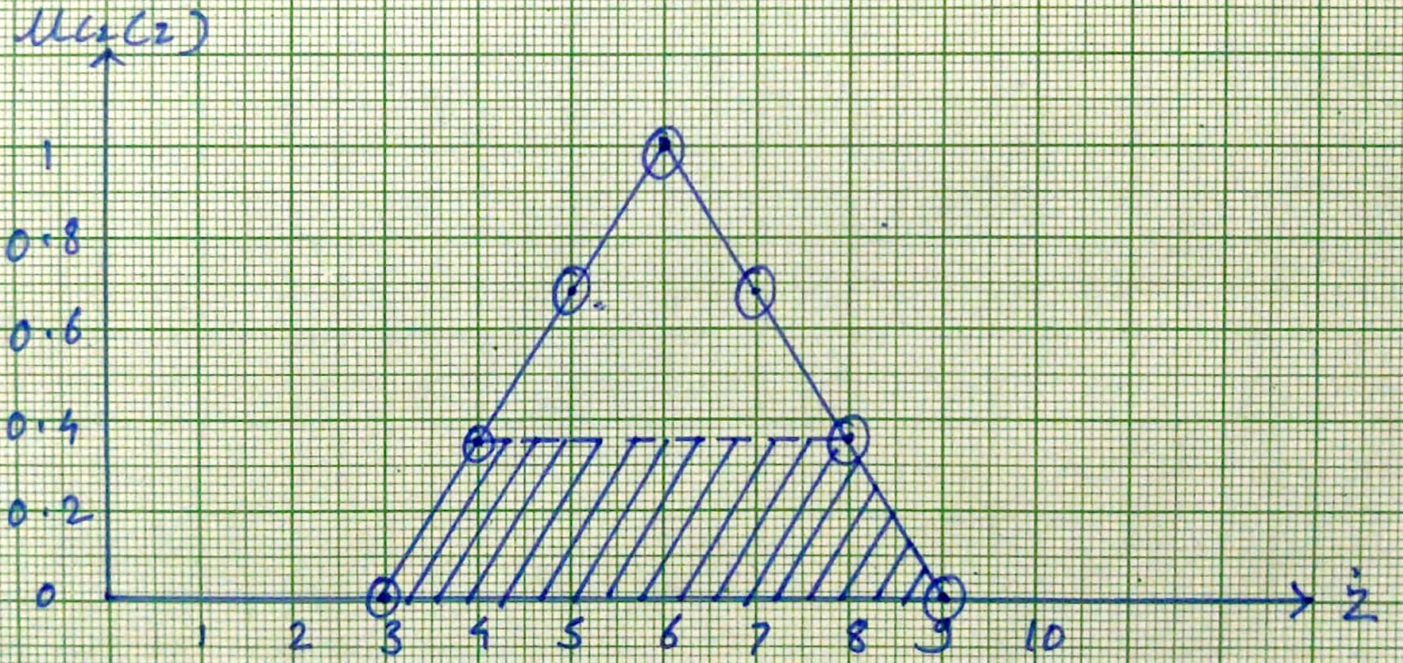


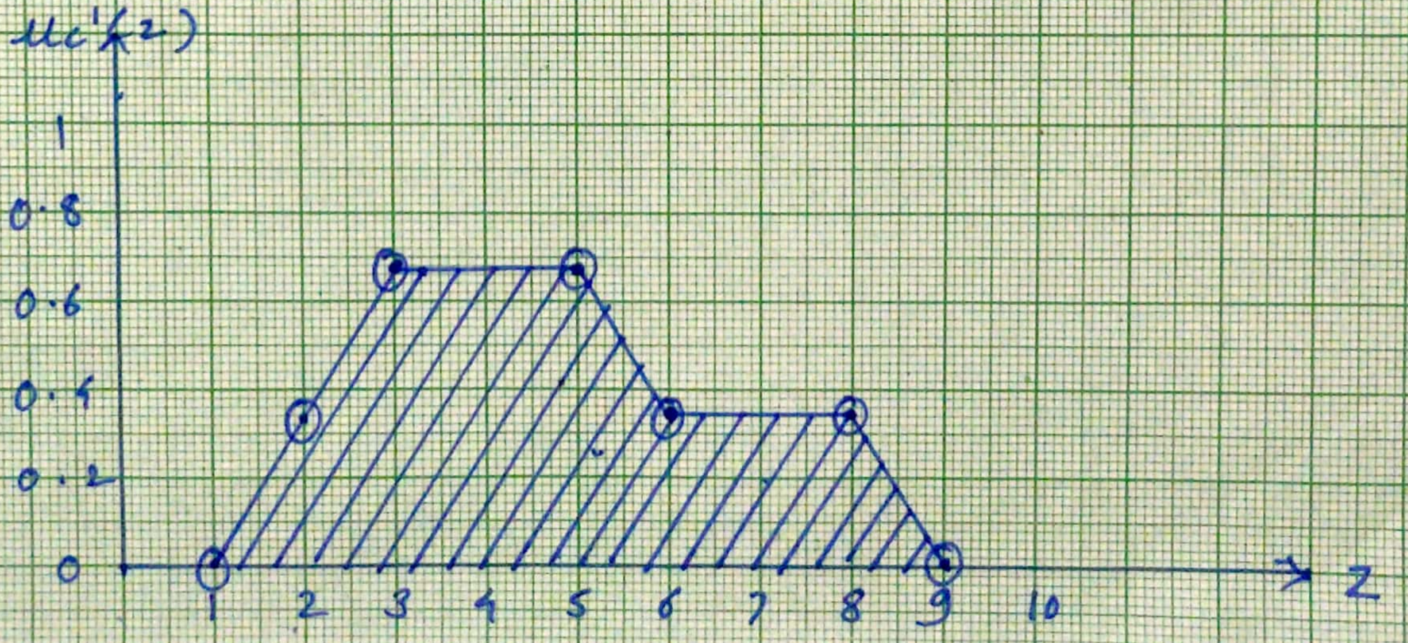
μA1(x)
μB1(y)
μC1(z)
2/3
fig (a)
fig (b)
fig (c)
Teacher's Sign.
Class



μA2(x)
μB2(y)
μC2(z)
1/3
2/3
fig (d)
fig (e)
fig (f)
Nagpur
Name


μc1(z)
1
0.8
0.6
0.4
0.2
0
1 2 3 4 5 6 7 8 9 10
z


μc2(z)
1
0.8
0.6
0.4
0.2
0
1 2 3 4 5 6 7 8 9 10
z


μc'(z)
1
0.8
0.6
0.4
0.2
0
1 2 3 4 5 6 7 8 9 10
z

Problem 2

$$x - \text{gyro bias} = A = \left\{ \frac{0.2}{1.7} + \frac{0.4}{1.8} + \frac{0.6}{1.9} + \frac{0.8}{2.0} + \frac{0.6}{2.1} + \frac{0.4}{2.2} + \frac{0.2}{2.3} \right\}$$

the membership functions look like

$$B = \left\{ \frac{0.1}{0.25} + \frac{0.4}{0.27} + \frac{0.9}{0.3} + \frac{0.4}{0.33} + \frac{0.1}{0.35} \right\}$$

To find

(a) Using classical implications operator $\mu_q = \max[\min(\mu_A, \mu_B), (1-\mu_A)]$, find a relation R for IF A THEN B

$$
\text{(a) } \min(\mu_A, \mu_B) \quad
\begin{bmatrix}
0.1 & 0.2 & 0.2 & 0.2 & 0.1 \\
0.1 & 0.4 & 0.4 & 0.4 & 0.1 \\
0.1 & 0.4 & 0.6 & 0.4 & 0.1 \\
0.1 & 0.4 & 0.8 & 0.4 & 0.1 \\
0.1 & 0.4 & 0.6 & 0.4 & 0.1 \\
0.1 & 0.4 & 0.4 & 0.4 & 0.1 \\
0.1 & 0.2 & 0.2 & 0.2 & 0.1
\end{bmatrix}
$$

$(1-\mu_A)$ $[0.8, 0.6, 0.4, 0.2, 0.4, 0.6, 0.8]$

Therefore,

$$
\mu_R(A_1, B_1) \quad
\begin{bmatrix}
0.8 & 0.8 & 0.8 & 0.8 & 0.8 \\
0.6 & 0.6 & 0.6 & 0.6 & 0.6 \\
0.4 & 0.4 & 0.6 & 0.4 & 0.4 \\
0.2 & 0.4 & 0.8 & 0.4 & 0.2 \\
0.4 & 0.4 & 0.6 & 0.4 & 0.4 \\
0.6 & 0.6 & 0.6 & 0.6 & 0.6 \\
0.8 & 0.8 & 0.8 & 0.8 & 0.8
\end{bmatrix}
$$

b) Say we have to change x-gyros and the new gyro has the following fuzzy bias

$$A' = \left\{ \frac{0}{1.7} + \frac{0.5}{1.8} + \frac{0.7}{1.9} + \frac{0.95}{2.0} + \frac{0.7}{2.1} + \frac{0.5}{2.2} + \frac{0}{2.3} \right\}$$

Calculate the associated accelerometer bias using

i) Max-min composition. $T = A' \circ R$

ii) Max-product composition. $T = A' \circ R$

⟹ [b] i)

$$\begin{array}{cc} \text{max rows} & \left[\begin{array}{cc} \text{min columns} & \left[\begin{bmatrix} 0 \\ 0.5 \\ 0.7 \\ 0.95 \\ 0.7 \\ 0.5 \\ 0 \end{bmatrix} \begin{bmatrix} 0.8 & 0.8 & 0.8 & 0.8 & 0.8 \\ 0.6 & 0.6 & 0.6 & 0.6 & 0.6 \\ 0.4 & 0.4 & 0.6 & 0.4 & 0.4 \\ 0.2 & 0.4 & 0.8 & 0.4 & 0.2 \\ 0.4 & 0.4 & 0.6 & 0.4 & 0.4 \\ 0.6 & 0.6 & 0.6 & 0.6 & 0.6 \\ 0.8 & 0.8 & 0.8 & 0.8 & 0.8 \end{bmatrix} \right] \end{array} \right] \end{array}$$

$$\begin{array}{cc} \text{max rows} & \left[\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0.5 & 0.5 & 0.5 & 0.5 & 0.5 \\ 0.4 & 0.4 & 0.6 & 0.4 & 0.4 \\ 0.2 & 0.4 & 0.8 & 0.4 & 0.2 \\ 0.4 & 0.4 & 0.6 & 0.4 & 0.4 \\ 0.5 & 0.5 & 0.5 & 0.5 & 0.5 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \right] \end{array}$$

$$[0.5, 0.5, 0.8, 0.5, 0.5]$$

Therefore $T \left\{ \frac{0.5}{0.25} + \frac{0.5}{0.27} + \frac{0.8}{0.3} + \frac{0.5}{0.33} + \frac{0.5}{0.35} \right\}$

ii)

$$\max_{rows}\left[[0\ \ 0.5\ \ 0.7\ \ 0.95\ \ 0.7\ \ 0.5\ \ 0] \cdot \begin{bmatrix} 0.8 & 0.8 & 0.8 & 0.8 & 0.8 \\ 0.6 & 0.6 & 0.6 & 0.6 & 0.6 \\ 0.4 & 0.4 & 0.6 & 0.4 & 0.4 \\ 0.2 & 0.4 & 0.8 & 0.4 & 0.2 \\ 0.4 & 0.4 & 0.6 & 0.4 & 0.4 \\ 0.6 & 0.6 & 0.6 & 0.6 & 0.6 \\ 0.8 & 0.8 & 0.8 & 0.8 & 0.8 \end{bmatrix} \right]$$

$$\max_{rows}\left(\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0.3 & 0.3 & 0.3 & 0.3 & 0.3 \\ 0.28 & 0.28 & 0.42 & 0.28 & 0.28 \\ 0.19 & 0.38 & 0.76 & 0.38 & 0.19 \\ 0.28 & 0.28 & 0.42 & 0.28 & 0.28 \\ 0.3 & 0.3 & 0.3 & 0.3 & 0.3 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}\right)$$

$[0.3, 0.38, 0.76, 0.38, 0.3]$

Therefore, $T = \left\{ \frac{0.3}{0.25} + \frac{0.38}{0.27} + \frac{0.76}{0.3} + \frac{0.38}{0.33} + \frac{0.3}{0.35} \right]$

PROBLEM 3
Discuss how genetic algorithm is different from genetic programming.

Genetic programming is a branch of genetic algorithms.

1. The main difference between genetic programming and genetic algorithms is the representation of the solution. Genetic programming creates computer programs in the lisp or scheme computer languages as the solution. Genetic algorithms create a string of numbers that represent the solution.
2. Genetic algorithms relate to the creation of shorter sections of code (expressed as strings called chromosomes), while genetic programming refers to the creation of full software programs (typically in the form of Lisp source code).
3. The fundamental distinction is in the structure. Genetic programming is expressed as a variable length parse tree structure of actions and values, but a genetic algorithm is represented as a chromosome, which is a binary string of 0 and 1 with fixed length. The tree-based representation increases GP's flexibility, but it also adds to its complexity and inefficiency.
4. We can observe from the implementation phases of these two algorithms that an offspring could be formed via crossover and then mutation for GA, whereas an offspring could only be generated by utilizing the operator chosen from crossover, reproduction, or mutation for GP.
5. Faulty states are frequently produced by genetic algorithms, whereas invalid states are rarely produced by genetic programs.
6. In terms of applications, GA is more commonly used to learn and discover both the contents and structures of solutions, whereas GP is more routinely used to optimize parameters for solutions when their structure is known.
7. Genetic algorithms search a solution space whereas genetic programming explores a program space.
8. Genetic programming, as opposed to genetic algorithms, is far more powerful. The genetic algorithm's output is a number, whereas the genetic programming's result is another computer programme. In essence, this is the beginning of self-programming computer programs.
9. For a variety of difficulties, genetic programming is the ideal solution. The first is when there isn't a perfect solution (for example, a programme that drives a car). There is no one-size-fits-all approach to driving an automobile. Some solutions drive safely at the cost of time, while others move quickly at the expense of safety. As a result, driving an automobile entails making trade-offs between speed and safety, among other things. In this situation, genetic programming will seek out a solution that tries to strike a balance and is the most efficient among a vast number of variables.
10. Genetic programming is also beneficial for discovering solutions in situations where the variables are constantly changing. The programme will identify one answer for a smooth concrete highway in the previous car example, but a completely different option for a bumpy gravel road.
11. The primary distinction between genetic programming and GAs is that genetic programming uses different coding of probable solutions. We can identify patterns

and portray them in a form that people can understand by combining knowledge from large volumes of data acquired from many sources. We know that certain equations suit particular numerical facts when we use mathematical modelling. It is applied to a wide range of scientific problems where theoretical grounds are insufficient to provide solutions to experiments.

12. Traditional methods aren't always sufficient because they assume a specific type of model. Genetic programming allows for a more “intelligent” search for a viable model, and it can be utilized to address highly complicated, non-linear, chaotic situations.

Problem 4

Design a fuzzy system:

Given:

The universe of each of the quantities are given below

Fuzzy Quantity	Minimum Value	Maximum Value	Step
D	0	10	0.5
A	0	90	1
S	0	5	0.2
ST	0	90	1

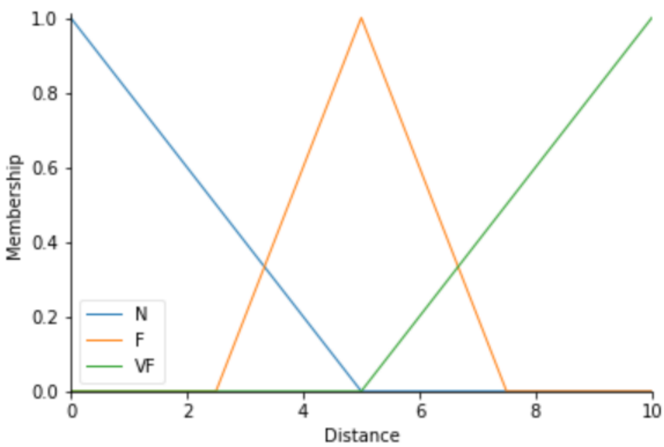
Universe of each Quantity

```
# GIVEN AS PER QUESTION
min_D = 0; max_D = 10; step_D = 0.5;
min_A = 0; max_A = 90; step_A = 1;
min_S = 0; max_S = 5; step_S = 0.2;
min_ST = 0; max_ST = 90; step_ST = 1;
```

Membership Function - **Triangular** membership function is chosen because, for this function as low, medium, high ME are the ways by which the values will be split and it is less complex, when compared to other membership functions.

i. **Distance [D] -** Triangle (N|0,0,3.5), Triangle (F|1.5,5,8.5), Triangle (VF|6.5,10,10)

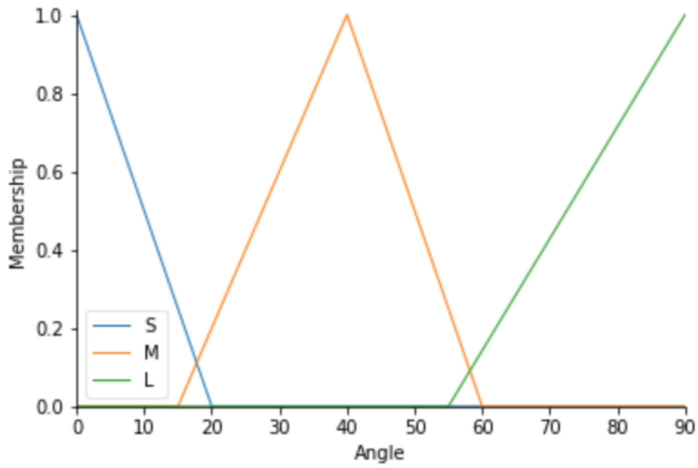
```
D = ctrl.Antecedent(np.arange(min_D, max_D + step_D, step_D),'Distance')
D["N"] = fuzz.trimf(D.universe,[0,0,5])
D["F"] = fuzz.trimf(D.universe, [2.5,5,7.5])
D["VF"] = fuzz.trimf(D.universe, [5,10,10])
```



Membership Function of Distance

ii. **Angle [A] -** Triangle (S|0, 0, 30), Triangle (M|10, 45, 80), Triangle (L|60, 90, 90)

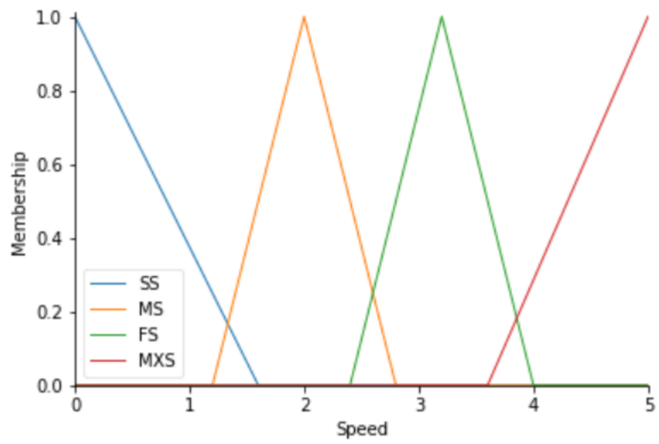
```
A = ctrl.Antecedent(np.arange(min_A, max_A + step_A, step_A),'Angle')
A["S"] = fuzz.trimf(A.universe,[0, 0, 20])
A["M"] = fuzz.trimf(A.universe, [15, 40, 60])
A["L"] = fuzz.trimf(A.universe, [55, 90, 90])
```



Membership Function of Angle

iii. **Speed [S] -** Triangle (SS|0,0,1.4), MS (Med|0.4,1.8,3), Triangle (FS|2,3.4,4.6), Triangle (MXS|3.4,5,5)

```
S = ctrl.Consequent(np.arange(min_S, max_S + step_S, step_S),'Speed',defuzzify_method="centroid")
S["SS"] = fuzz.trimf(S.universe, [0, 0, 1.6])
S["MS"] = fuzz.trimf(S.universe, [1.2,2,2.8])
S["FS"] = fuzz.trimf(S.universe, [2.4, 3.2, 4])
S["MX"] = fuzz.trimf(S.universe, [3.6, 5, 5])
```

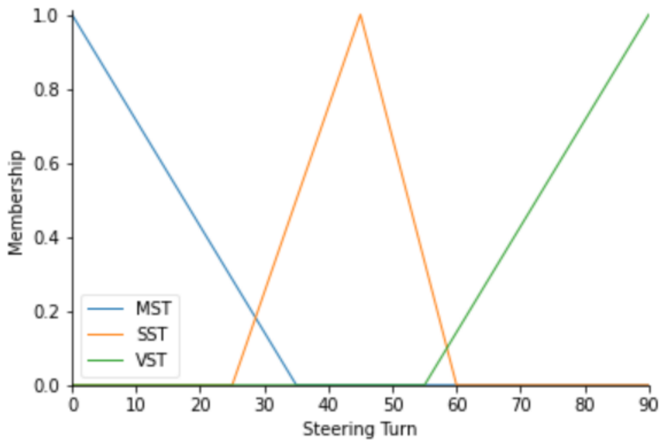


Membership Function of Speed

iv. Steering Turn [ST]

Triangle (MST| 0,20,45), SST (Med| 25,45,80), Triangle (VST| 60,90,90)

```
ST = ctrl.Consequent(np.arange(min_ST, max_ST + step_ST, step_ST),'Steering Turn',defuzzify_method="centroid")
ST["MST"] = fuzz.trimf(A.universe,[0, 0, 35])
ST["SST"] = fuzz.trimf(A.universe, [25, 45, 60])
ST["VST"] = fuzz.trimf(A.universe, [55, 90, 90])
```



Membership Function of Steering Turn

Rules:

Since there are three different states of input “distance” and three different states of input “angle”, we can define 9 rules according to the inputs and our requirements of outputs for our fuzzy control system.

1. If D is N and A is S, S is SS and ST is VST (obstacle in front)
2. Else if D is N and A is M, S is SS and ST is SST (obstacle near but not in front)
3. Else if D is N and A is L, S is MS and ST is MST (obstacle out of the way)
4. Else if D is F and A is S, S is MS, ST is VST (obstacle in the way but not close)
5. Else if D is F and A is M, S is MS, ST is SST (obstacle far but not in front)
6. Else if D if F and A is L, S is FS, ST is MST (obstacle far and out of the way)
7. Else if D is VF and A is S, S is FS, ST is VST (obstacle very far but on collision course)
8. Else if D is VF and A is M, S is MX, ST is SST (obstacle very far and medium angle)
9. Else if D is VF and A is M, S is MX, ST is MST (obstacle very far and at big angle -> no obstacle)

```
rule1 = ctrl.Rule(D['N'] & A['S'],(S['SS'],ST['VST']))
rule2 = ctrl.Rule(D['N'] & A['M'],(S['SS'],ST['SST']))
rule3 = ctrl.Rule(D['N'] & A['L'],(S['SS'],ST['SST']))
rule4 = ctrl.Rule(D['F'] & A['S'],(S['MS'],ST['SST']))
rule5 = ctrl.Rule(D['F'] & A['M'],(S['MS'],ST['SST']))
rule6 = ctrl.Rule(D['F'] & A['L'],(S['FS'],ST['MST']))
rule7 = ctrl.Rule(D['VF'] & A['S'],(S['FS'],ST['MST']))
rule8 = ctrl.Rule(D['VF'] & A['M'],(S['MX'],ST['MST']))
rule9 = ctrl.Rule(D['VF'] & A['L'],(S['MX'],ST['MST']))
```

Inferencing system:

Sugeno membership function is commonly used when output membership function is not present. It is more suited to "PID style controllers". Mamdani is more suitable for a system with human input, which is closer to this system. As an example, humans don't mathematically model the angle and distance of an obstacle to determine the degree of speed/steering adjustment; we just go by intuition, and Mamdani's ability to generate a fuzzy output is more suitable for this. We have used Mamdani as the inference system, which helps to obtain a crisp value by defuzzification whereas Sugeno calculates the output as weighted average. In terms of error criteria comparisons, " Mamdani" performs better than "Sugeno". overall, " Mamdani" is best, due to its advantages of being intuitive, having wide acceptance rate and it's well-suitedness to human input.

Defuzzification:

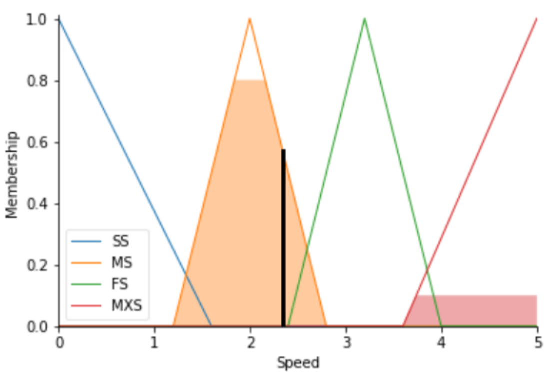
Defuzzification is the process of converting fuzzy output into a single crisp value. Centroid defuzzification is used because it returns the fuzzy set's center of gravity along the x-axis. This defuzzification is widely employed because it improves the repeatability of the crisp value and makes the fuzzy set more robust as the center of gravity of the entire fuzzy set.

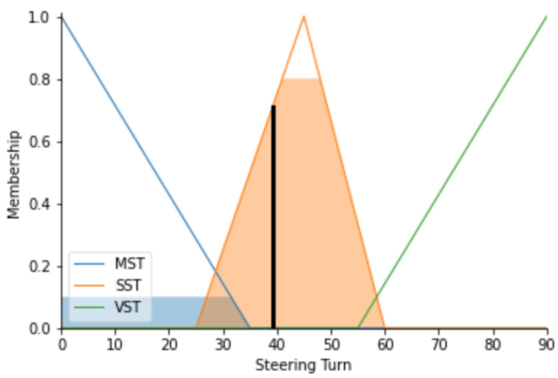
Simulation result:

Case 1

When input is D = 5.5, A = 35, the output S is 2.344587495375509, ST is 39.220790378006846.

```
simulation.input['Distance'] = 5.5
simulation.input['Angle'] = 35
```





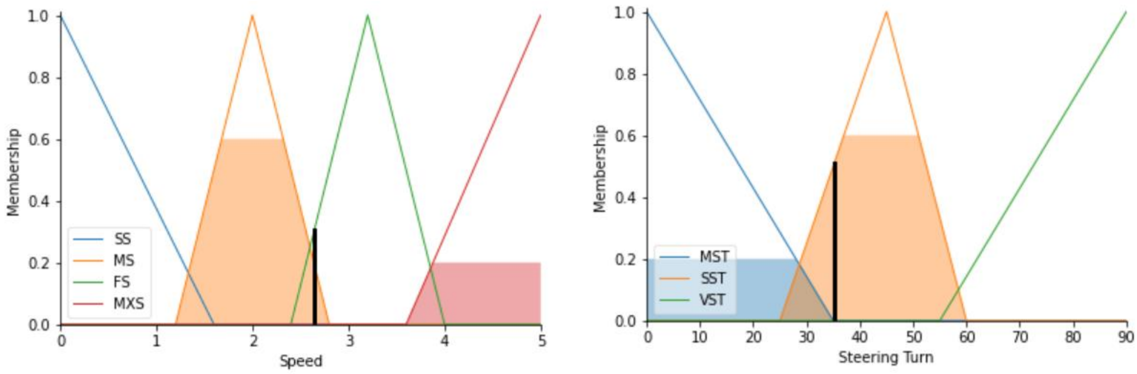
```
OrderedDict([('Speed', 2.344587495375509), ('Steering Turn', 39.220790378006846)])
```

Crisp value obtained is 0.55 for speed and 0.7 for steering turn.

Case 2

When input is D = 6, A = 45, the output S is 2.645656565656565, 35.18573797678273.

```
simulation.input['Distance'] = 6
simulation.input['Angle'] = 45
```




```
OrderedDict([('Speed', 2.645656565656565), ('Steering Turn', 35.18573797678273)])
```

Crisp value obtained is 0.3 for speed and 0.5 for steering turn.