ASSIGNMENT 4 : Answers

Answer - Problem 1

Given 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

Here x & y are the inputs (antecedent) while z is output (consequent) for the given linguistic variables

To apply this rules we need to create fuzzy sets for each of the variables A_1, A_2, B_1, B_2

Part a] Creating the fuzzy set from given membership functions

i)

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

The above equation resemble to a triangular membership function. which is given as

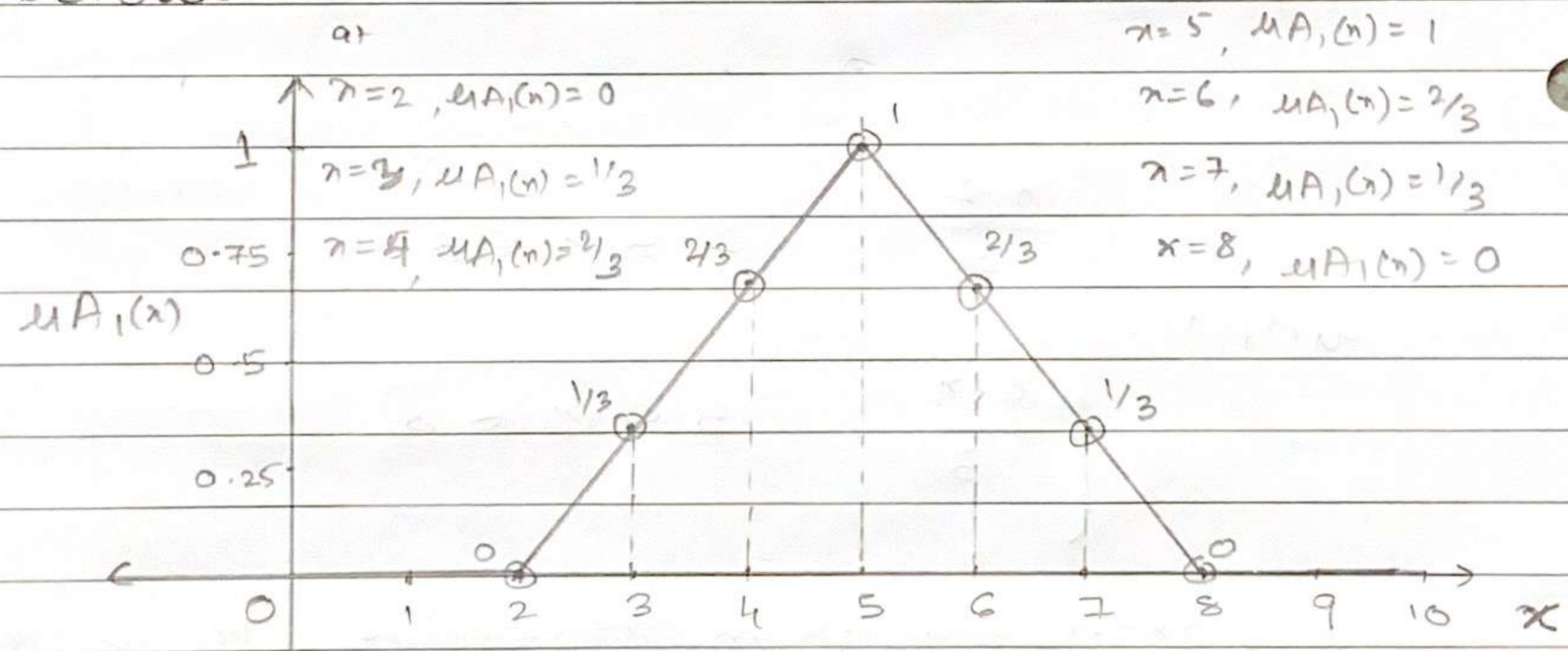
$$\text{triangle}(x \mid a,b,c) = \begin{cases} 0 & x < a \\ \dfrac{x-a}{b-a} & a \le x \le b \\ \dfrac{c-x}{c-b} & b \le x \le c \\ 0 & x \ge c \end{cases}$$

$$= \max\left[\min\left(\frac{x-a}{b-a}, \frac{c-x}{c-b}\right), 0\right]$$

here a is start point b is the peak point & c is the end point of the triangle

Therefore (i) μA_1 can be compared to a triangular membership function with $a = 2$, $b = 5$, and $c = 8$

Cross checking the function by putting the values of x in μA_1 we get following membership function as shown below.



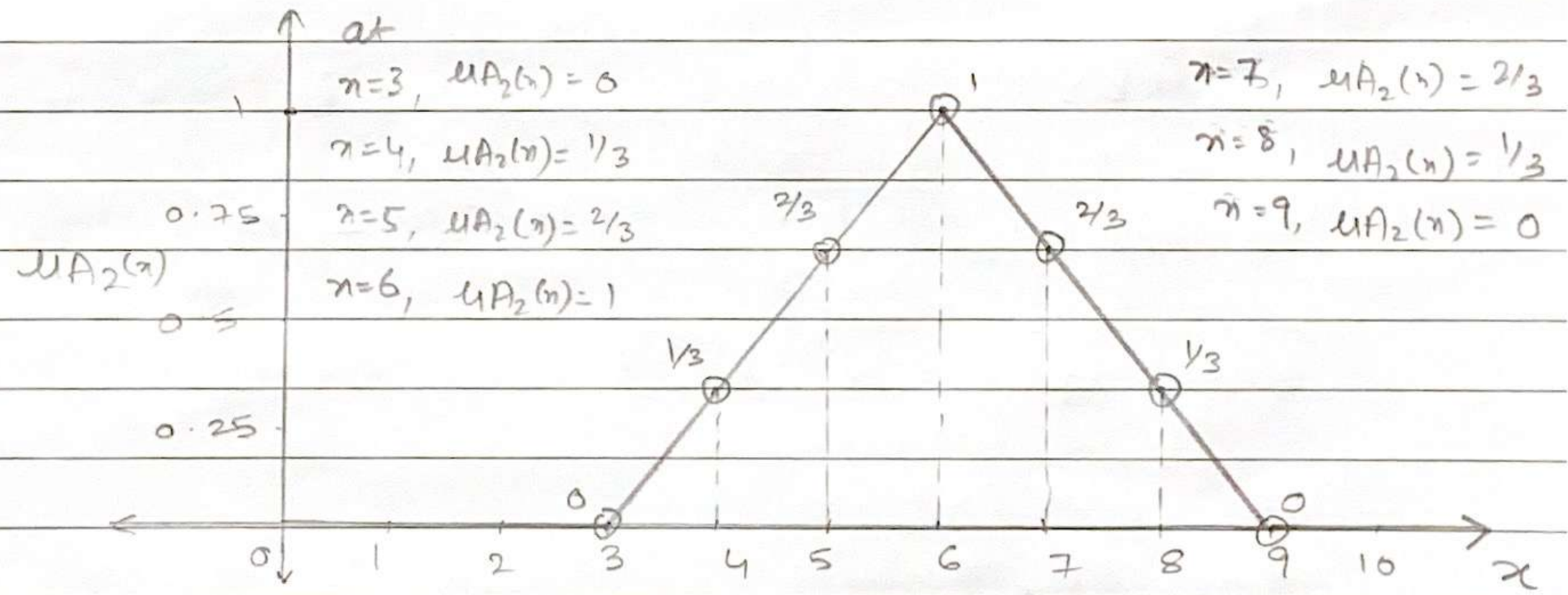
Fuzzy set for A_1

Similarly comparing all the other membership functions μA_2, μB_1, μB_2, μC_1 & μC_2 with triangular functions we get the membership function plots as below.

ii) $$\mu A_2(x) = \begin{cases} \dfrac{x-3}{3} & 3 \le x \le 6 \\ \dfrac{9-x}{3} & 6 < x \le 9 \end{cases}$$

here $a = 3$, $b = 6$, and $c = 9$

Generating the triangular function from a b & c as below.

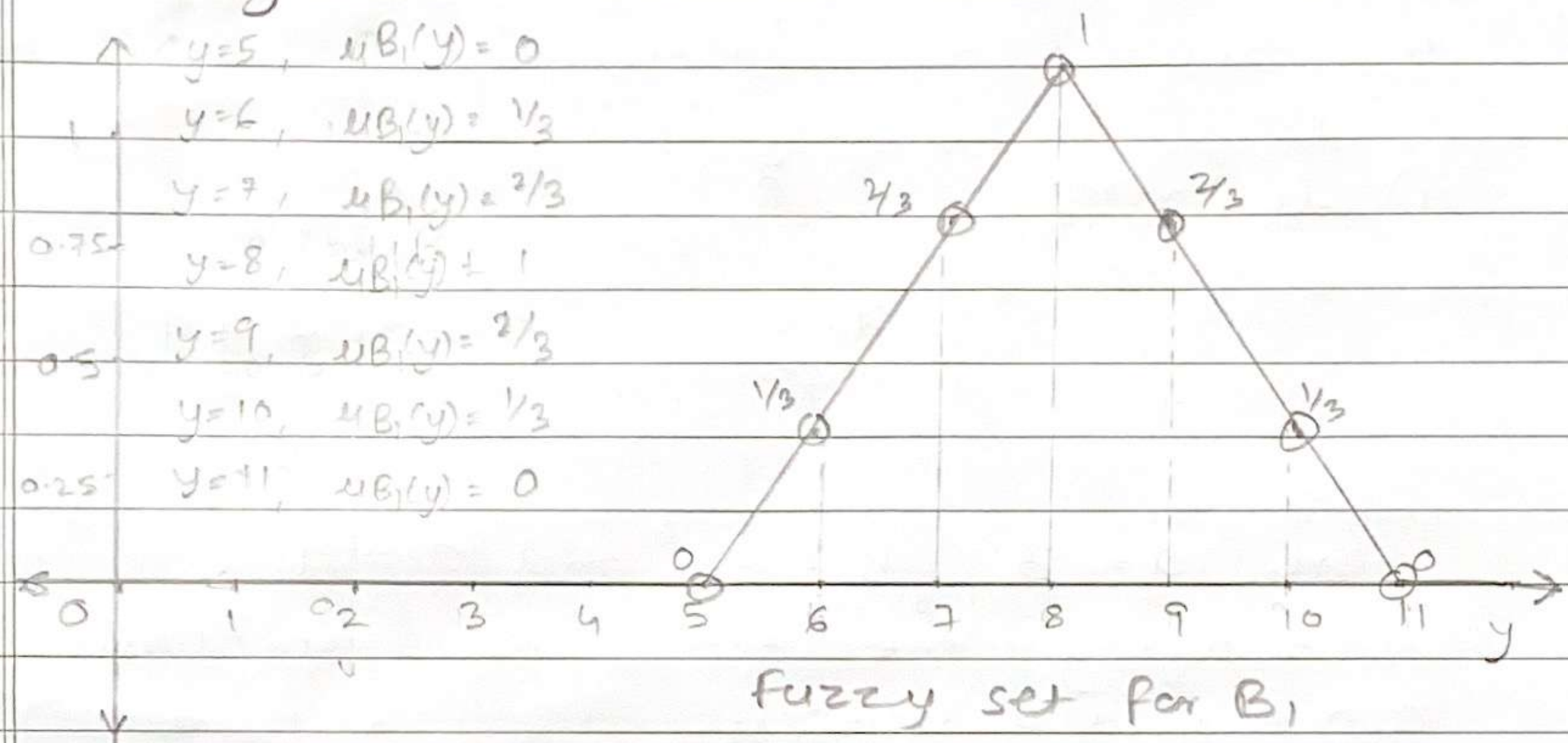


fuzzy set for A_2

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

here $a = 5$, $b = 8$ and $c = 11$

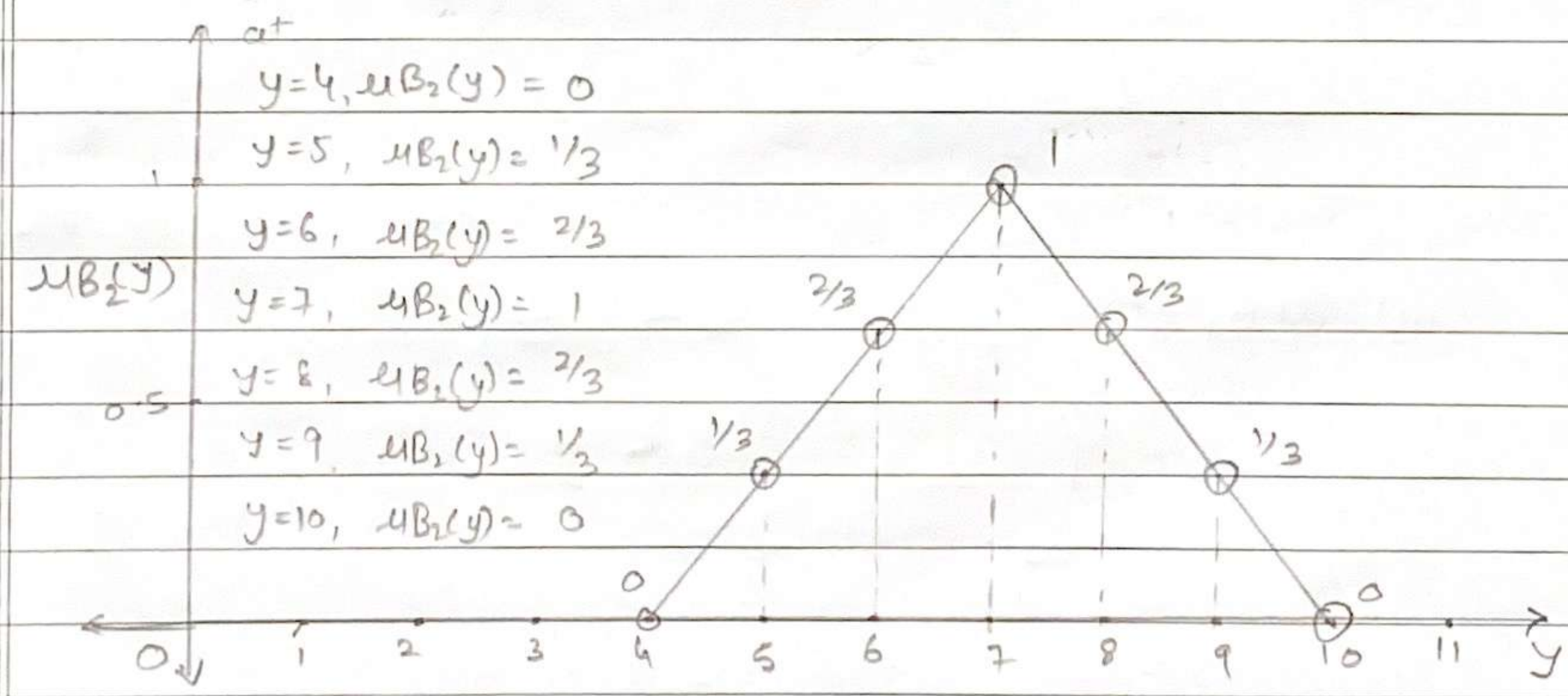
generating triangular membership function using this values of a, b + c we get.



fuzzy set for B_1

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

here a = 4, b = 7 and c = 10



fuzzy set for B_2

Sundaram
FOR EDUCATIONAL USE

Drawing the membership ~~for~~ function for Consequent C_1 & C_2

v) $$\mu_{C_1}(z) = \begin{cases} \dfrac{z-1}{3} & 1 \le z \le 4 \\[2ex] \dfrac{7-z}{3} & 4 < z \le 7 \end{cases}$$

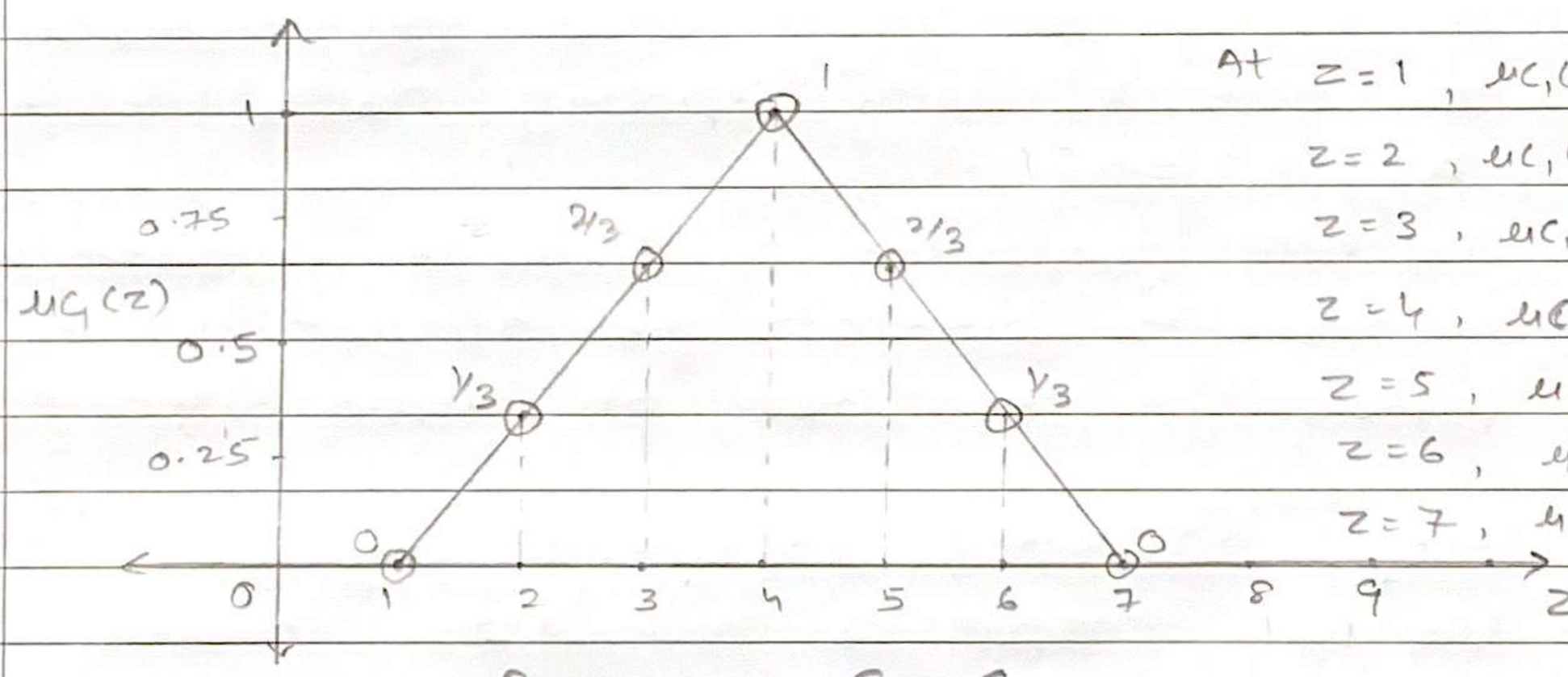
from the above equation we can get.

$a = 1$, $b = 4$, $c = 7$.

Comparing it with

$\dfrac{x-a}{b-a}$ & $\dfrac{c-x}{c-b}$

we get following function as shown below.



At $z = 1$, $\mu_{C_1}(z) = 0$

$z = 2$, $\mu_{C_1}(z) = 1/3$

$z = 3$, $\mu_{C_1}(z) = 2/3$

$z = 4$, $\mu_{C_1}(z) = 1$

$z = 5$, $\mu_{C_1}(z) = 2/3$

$z = 6$, $\mu_{C_1}(z) = 1/3$

$z = 7$, $\mu_{C_1}(z) = 0$

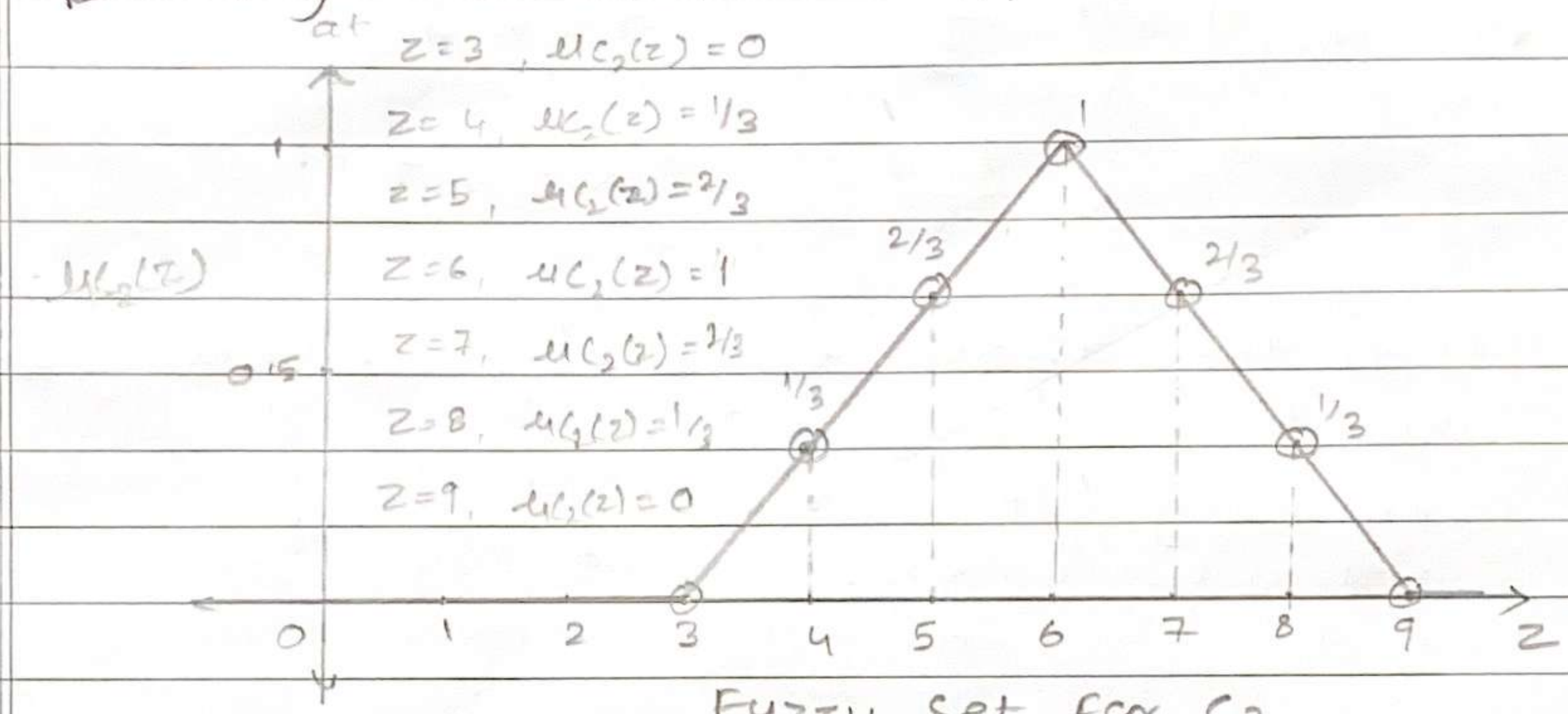
fuzzy set for C_1

vi) $$\mu_{C_2}(z) = \begin{cases} \dfrac{z-3}{3} & 3 \le z \le 6 \\[2ex] \dfrac{9-z}{3} & 6 \le z \le 9 \end{cases}$$

Sundaram

FOR EDUCATIONAL USE

Comparing it with triangular function equation we get $a = 3$, $b = 6$ & $c = 9$
Drawing the membership function as



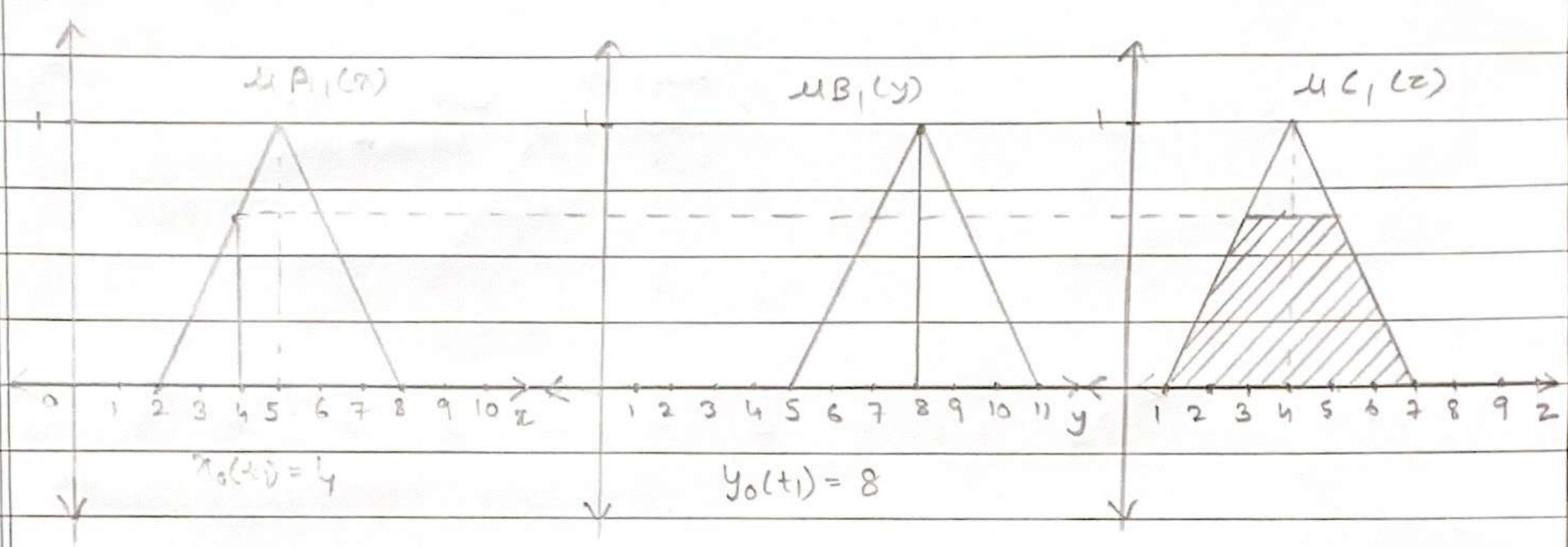
Fuzzy set for C_2

Assuming an input from the sensor at time t_1 with reading $x_0(t_1) = 4$ & $y_0(t_1) = 8$
We will generate a Mamdani inference model to predict the output value based on the set of rules (rule1 & rule 2)

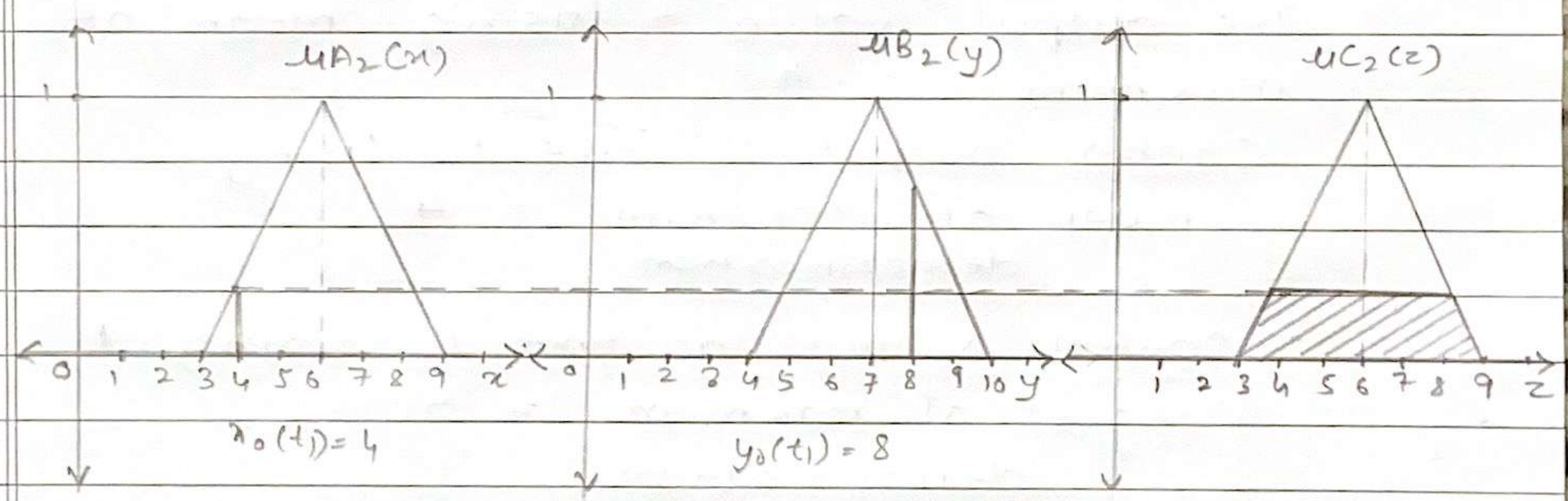
Part b] Creating inference model.
We will create a mamdani model using max-min operator for aggregation

We can visualize the rules using membership functions as below.

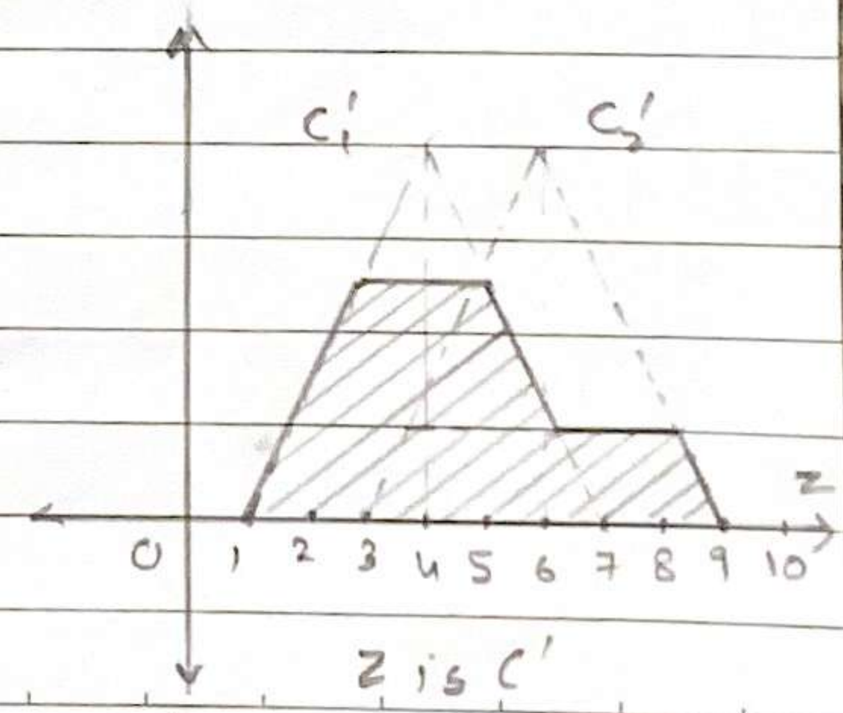
Rule 1: If x is A_1 & y is B_1 then z is C_1



Rule 2: If x is A_2 & y is B_2 Then z is C_2



The control inference using mamdani is as shown beside



fuzzy set for control model

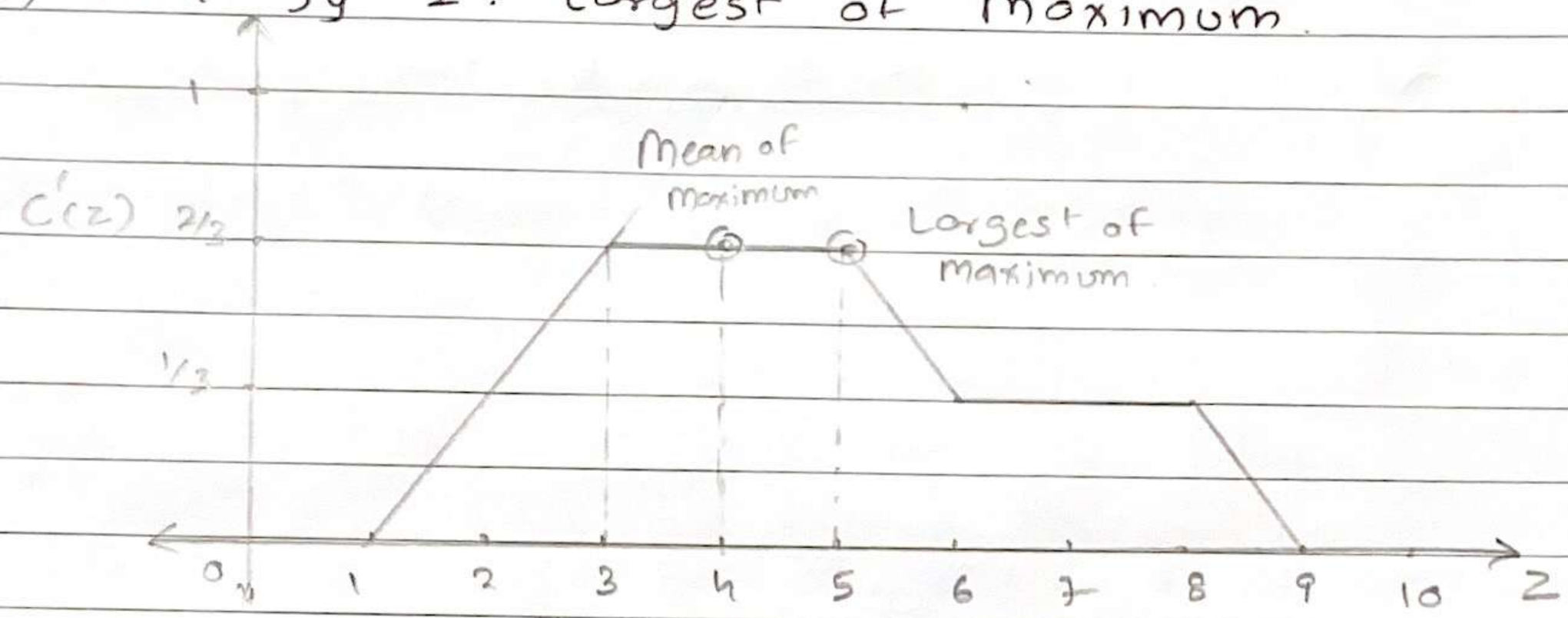
Now for defuzzification we can generate crisp ~~vo~~ output values for given input

Sundaram
FOR EDUCATIONAL USE

Part c] Defuzzification

i) Strategy 1: mean of maximum.

ii) Strategy 2: Largest of maximum.



Fuzzy set for output generation

from the above fig which is inference model for $x_0(t_1) = 4$ and $y_0(t_1) = 8$

The output value = using mean of maximum is 4.

Control output at time t_1 using

$\boxed{\text{mean of maximum defuzzification} = 4}$

& Control output at time t_1 using

$\boxed{\text{Largest of maximum defuzzification} = 5}$

Sundaram

FOR EDUCATIONAL USE

Assignment - 4

Problem - 2

The fuzzy set of gyro bias x_{gb} is given by

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

The fuzzy set of accelerator bias is given by

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

(a) Find Relation R for $A \rightarrow B$

Implication operator $\mu_R = \text{Max}[\min(\mu_A, \mu_B), (1-\mu_A)]$

$$A' = (1-\mu_A) = \left\{ \frac{0.8}{1.7} + \frac{0.6}{1.8} + \frac{0.4}{1.9} + \frac{0.2}{2.0} + \frac{0.4}{2.1} + \frac{0.6}{2.2} + \frac{0.8}{2.3} \right\}$$

$$\mu_A(x) = [0.2, 0.4, 0.6, 0.8, 0.6, 0.4, 0.2]$$

$$\mu_B(x) = [0.1, 0.4, 0.9, 0.4, 0.1]$$

$$\mu_A'(x) = [0.8, 0.6, 0.4, 0.2, 0.4, 0.6, 0.8]$$

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

$$\mu_R(A,B) = \max\left[\begin{bmatrix} 0.1 & 0.2 & 0.2 & 0.2 & 0.1 \\ 0.1 & 0.4 & 0.4 & 0.4 & 0.1 \\ 0.1 & 0.4 & 0.6 & 0.4 & 0.1 \\ 0.1 & 0.4 & 0.8 & 0.4 & 0.1 \\ 0.1 & 0.4 & 0.6 & 0.4 & 0.1 \\ 0.1 & 0.4 & 0.4 & 0.4 & 0.1 \\ 0.1 & 0.2 & 0.2 & 0.2 & 0.1 \end{bmatrix}, \begin{bmatrix} 0.8 \\ 0.6 \\ 0.4 \\ 0.2 \\ 0.4 \\ 0.6 \\ 0.8 \end{bmatrix}\right]$$

$$\mu_R = \begin{bmatrix} 0.8 & 0.8 & 0.8 & 0.8 & 0.8 \\ 0.6 & 0.6 & 0.6 & 0.6 & 0.6 \\ 0.4 & 0.4 & 0.6 & 0.4 & 0.4 \\ 0.2 & 0.4 & 0.8 & 0.4 & 0.2 \\ 0.4 & 0.4 & 0.6 & 0.4 & 0.4 \\ 0.6 & 0.6 & 0.6 & 0.6 & 0.6 \\ 0.8 & 0.8 & 0.8 & 0.8 & 0.8 \end{bmatrix}$$

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

(i) Max-Min Composition $T = A' \circ R$

$$\text{Max}\left(\text{Min}\left(\begin{bmatrix} 0 \\ 0.5 \\ 0.7 \\ 0.95 \\ 0.7 \\ 0.5 \\ 0 \end{bmatrix}, \begin{bmatrix} 0.8 & 0.8 & 0.8 & 0.8 & 0.8 \\ 0.6 & 0.6 & 0.6 & 0.6 & 0.6 \\ 0.4 & 0.4 & 0.6 & 0.4 & 0.4 \\ 0.2 & 0.4 & 0.8 & 0.4 & 0.2 \\ 0.4 & 0.4 & 0.6 & 0.4 & 0.4 \\ 0.6 & 0.6 & 0.6 & 0.6 & 0.6 \\ 0.8 & 0.8 & 0.8 & 0.8 & 0.8 \end{bmatrix}\right)\right)$$

$$\underset{\text{rows}}{\text{Max}} \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0.5 & 0.5 & 0.5 & 0.5 & 0.5 \\ 0.4 & 0.4 & 0.6 & 0.4 & 0.4 \\ 0.2 & 0.4 & 0.8 & 0.4 & 0.2 \\ 0.4 & 0.4 & 0.6 & 0.4 & 0.4 \\ 0.5 & 0.5 & 0.5 & 0.5 & 0.5 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$T = \begin{bmatrix} 0.5 & 0.5 & 0.8 & 0.5 & 0.5 \end{bmatrix}$$

(ii) Max - Product Composition

$$\mu_{A \circ R}(A,B) = \text{Sup}(\mu_A \cdot \mu_R(A,B))$$

$$= \underset{\text{row}}{\text{Max}} \left(\begin{bmatrix} 0 & 0.5 & 0.7 & 0.95 & 0.7 & 0.5 \end{bmatrix} \cdot \begin{bmatrix} 0.8 & 0.8 & 0.8 & 0.8 & 0.8 \\ 0.6 & 0.6 & 0.6 & 0.6 & 0.6 \\ 0.4 & 0.4 & 0.6 & 0.4 & 0.4 \\ 0.2 & 0.4 & 0.8 & 0.4 & 0.2 \\ 0.4 & 0.4 & 0.6 & 0.4 & 0.4 \\ 0.6 & 0.6 & 0.6 & 0.6 & 0.6 \\ 0.8 & 0.8 & 0.8 & 0.8 & 0.8 \end{bmatrix} \right)$$

$$\underset{\text{rows}}{\text{Max}} \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0.3 & 0.3 & 0.3 & 0.3 & 0.3 \\ 0.28 & 0.28 & 0.42 & 0.28 & 0.28 \\ 0.19 & 0.38 & 0.76 & 0.38 & 0.19 \\ 0.28 & 0.28 & 0.42 & 0.28 & 0.28 \\ 0.3 & 0.3 & 0.3 & 0.3 & 0.3 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$T = \begin{bmatrix} 0.3 & 0.38 & 0.76 & 0.38 & 0.3 \end{bmatrix}$$

Difference between Genetic Algorithm and Genetic Programming

Genetic Algorithm and Genetic Programming are very similar to each other. Both are specific type of a broad class of Evolutionary Algorithms. Both of them are used to evolve the answer to a problem by comparing the fitness of each candidate in a population of potential candidates over generations. In each generation, new candidates are found by randomly changing (mutation) or swapping parts (crossover) of other candidates. The least 'fit' candidates are removed from the population. Mostly, the differences are in how solutions are encoded and how they are evaluated.

Structural differences:

The main difference between Genetic Algorithm and Genetic Programming is the representation of its structure. Let us take the following example: 2*3 + x*5 - 1 to understand their structural difference.

The **Genetic Algorithm** is typically represented as a **string** of **actions and values**. For the above example a parser has to be written to understand how to turn this into a function. The resulting function might look like this: function(x) {return 2*3 + x * 5 - 1;}. The parser also needs to know how to deal with invalid states, because mutation and crossover operations don't care about the semantics of the algorithm. For example, the following string could be produced: 1+/3-2* as well. Thus, an approach needs to be decided to deal with these invalid states.

The **Genetic Programming** is represented as a **tree structure** of **actions and values**, which is usually a nested data structure. The tree structure for the above example is given by:

```
        -
      /   \
     +     1
   /   \
  *     *
 / \   / \
2   3 x   5
```

A parser also must be written for this encoding, but genetic programming does not (usually) produce invalid states because mutation and crossover operations work within the structure of the tree.

Core differences:

The core difference is that **Genetic Algorithm** are trying to optimize a fixed number of variables whereas **Genetic Programming** is trying to optimize an entire structure with dynamic size, as well as any parameters of that structure. For example, you can use Genetic Algorithm to find the best slope (a) and constant parameter (b) for a simple regression of the type y = a*x + b. With Genetic Programming you could try different functions on x such as a*x, x^a, a^x, sin(a*x), $a*x^2 + b*x + c$, etc. that would better predict y. Thus, Genetic Programming can be seen as a more flexible version of Genetic Algorithm, or we can say that Genetic Algorithm is a special case of Genetic Programming with a constant number of variables.

Programming differences:

The **Genetic Algorithm** uses a binary encoding of all parameters. For example, the code considering parameters would generate a number, efficiency, or anything else that we are trying to optimize. We can encode those properties as real numbers, binary encoded numbers, gray-coded etc. but we always decode them into the form required by the fitness function.

In **Genetic programming**, rather than evolving parameters, we evolve the executable code itself. The representation generally isn't something like binary-coded decimal numbers but rather abstract syntax trees, often expressed in syntactically regular languages like Lisp.

Practical differences:

Genetic algorithm:

- Inherently have a fixed length, which means the resulting function has bounded complexity.
- Often produces invalid states. Hence, these need to be handled non-destructively.
- Often rely on operator precedence (in our example multiplication happens before subtraction) which could be seen as a limitation.

Genetic programming:

- Inherently have a variable length, which means they are more flexible but often grow in complexity.
- Rarely produces invalid states, these can usually be discarded.
- Use an explicit structure to avoid operator precedence entirely.

Minor differences:

Practitioners found that the normal Evolutionary Algorithm parameters that work well for **Genetic Algorithm** often don't work well for **Genetic programming**. Parameters such as population size can differ dramatically, with 100 or so being very common in Genetic Algorithm, while Genetic Programming might use 500 or 1000 individuals in a population. Genetic Programming approaches typically require some additional methods to control things like bloat, where the Genetic Algorithm system may evolve expressions like "2 + 2 - 1 + x + 4 - 2 * x" instead of the equivalent "3x+7" or evolve the same subexpressions repeatedly.

Answer : Problem 4

Given : A robot navigation environment with following states.

D Distance from obstacle N= Near, F = For, VF = very for
A Angle with obstacle S = Small, M = Medium, L = Large
S Speed SS = Slow Speed, MS = Medium Speed,
FS = Fast Speed, MX = Maximum Speed
ST Steering Turn MST = mild turn,
SST = sharp Turn, VST = Very sharp turn

1] Fuzzy System Design

a) membership functions for each state mentioned above.

i) Distance from obstacle : Universe (0 - 10, step = 0.5)

Assumptions :

- For ~~Near~~ Distance = Near we are considering the robot to be in a range from 0 - 2
- for Distance = For we are considering the robot to be in a range from 4 - 7
- For Distance = very for we consider the robot to be in range greater than 9
- Note : All the above ranges are core values for each of the sub states in Distance.
- We are using trapezoidal membership function as we have a range of core value rather than a single value.

Following the assumptions we define membership functions as below

i) $\mu N(d) = \begin{cases} 1 & , d \leq 2 \\ \frac{4-d}{2} & , 2 \leq d \leq 4 \\ 0 & , d \geq 4 \end{cases}$

here d = distance

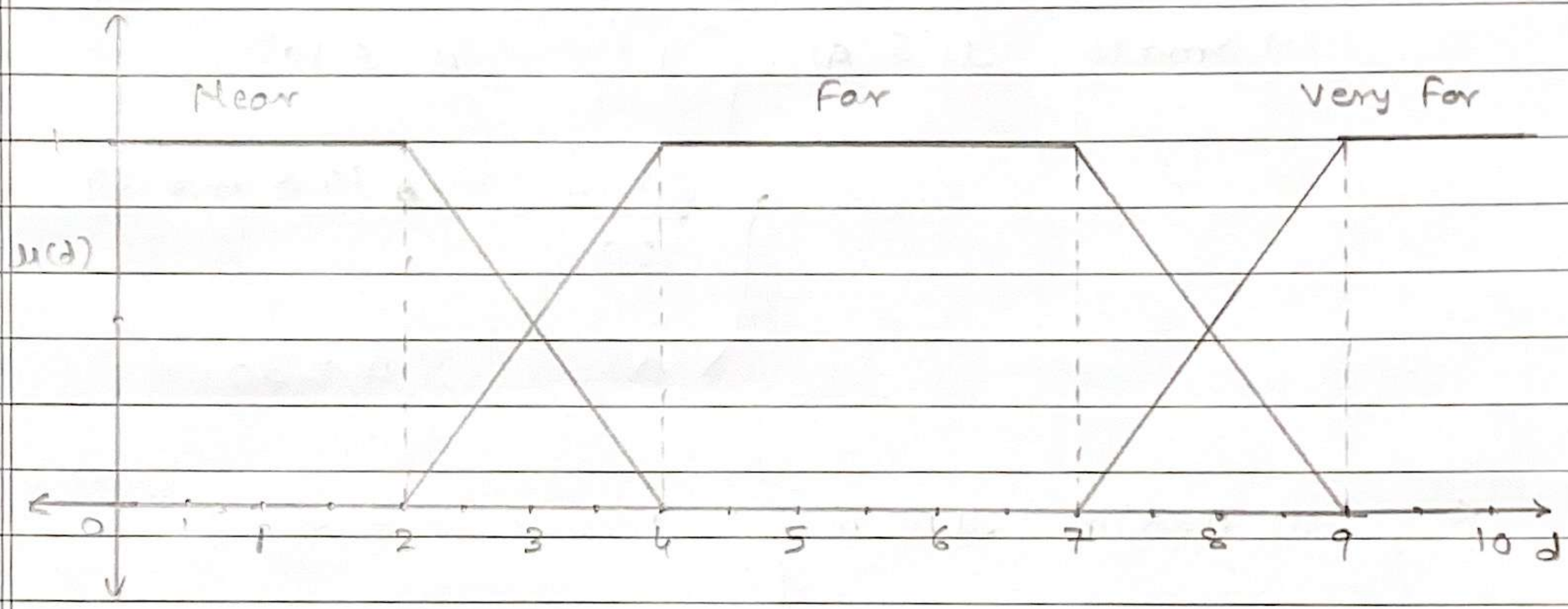
ii) $\mu F(d) = \begin{cases} 0 & , d \leq 2 \\ \frac{d-2}{2} & , 2 \leq d \leq 4 \\ 1 & , 4 \leq d \leq 7 \\ \frac{9-d}{2} & , 7 \leq d \leq 9 \\ 0 & , d \geq 9 \end{cases}$

iii) $\mu VF(d) = \begin{cases} 0 & , d \leq 7 \\ \frac{d-7}{2} & , 7 \leq d \leq 9 \\ 1 & , d \geq 9 \end{cases}$

Sundaram®

FOR EDUCATIONAL USE

Drawing the membership function from above equation we get. universe of 'Distance' as



ii) Angle with obstacle: universe (0 - 90 , step = 1)

Assumptions:

- For Angle = Small we consider robot to be at an angle between 0°-10°
- For Angle = Medium we consider robot to be at an angle between 30° - 60°
- For Angle = Large we consider robot to be at an angle between 80° - 90°
- Note: All the above ranges are core values for each subset of 'Angle'
- We use trapezoidal membership function as we have a range of core value rather than a single core value.

Sundaram
FOR EDUCATIONAL USE

Based on the assumption the membership function are given as below:

i) Small $\mu S(A) = \begin{cases} 1 & , A \leq 10^\circ \\ \frac{30-A}{20} & , 10 \leq A \leq 30 \\ 0 & , A \geq 30 \end{cases}$

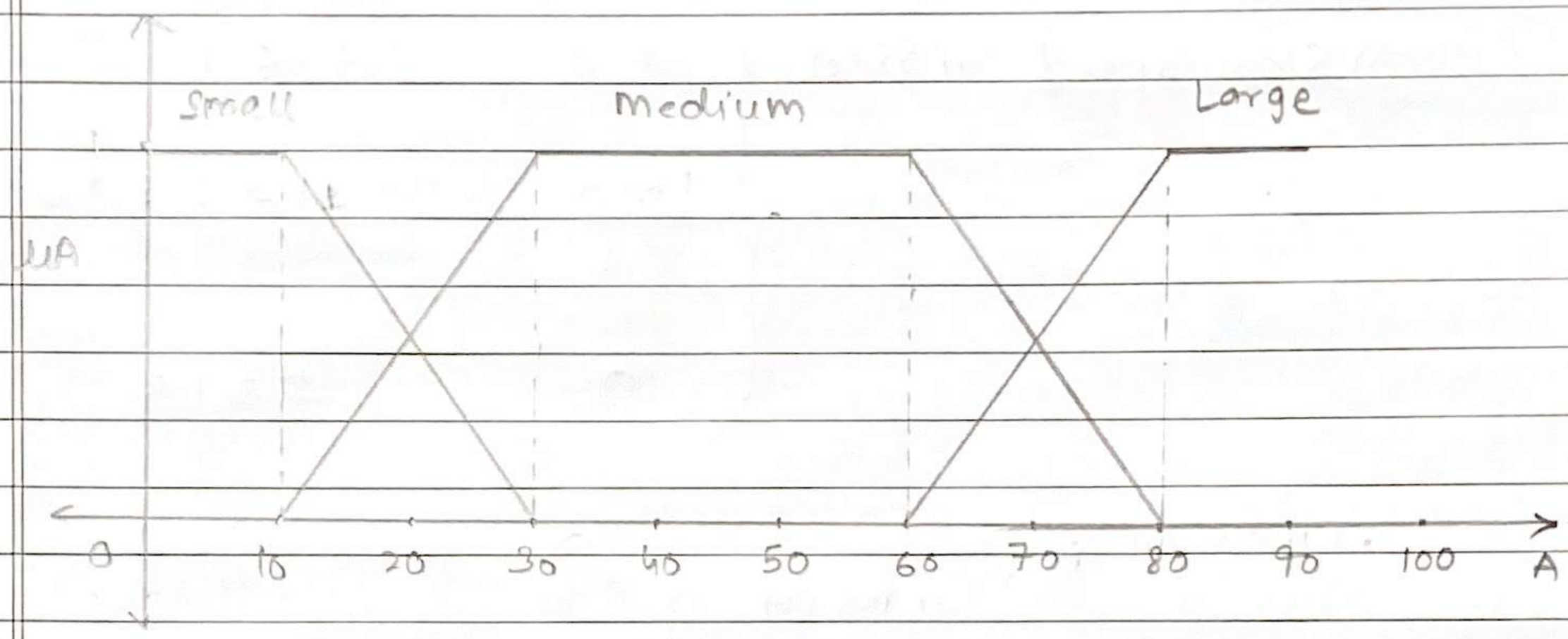
ii) Medium $\mu M(A) = \begin{cases} 0 & , A \leq 10 \\ \frac{A-10}{20} & , 10 \leq A \leq 30 \\ 1 & , 30 \leq A \leq 60 \\ \frac{80-A}{20} & , 60 \leq A \leq 80 \\ 0 & , A \geq 80 \end{cases}$

iii) Large $\mu L(A) = \begin{cases} 0 & , A \leq 80 \\ \frac{A-60}{20} & , 60 \leq A \leq 80 \\ 1 & , A \geq 80 \end{cases}$

Sundaram

FOR EDUCATIONAL USE

Drawing the membership function from above equations for 'Angle with obstacle' we get.



iii) Speed : universe (0 - 5 , step = 0.2)

Assumptions

- For Speed = Slow Speed (SS) we consider robot to be running at speed in range 0 - 1.6
- For speed = Medium Speed (MS) we consider robot to be running at speed in range 1.6 - 2.8
- For speed = Fast Speed (FS) we consider robot to be running at speed ranging from 3.4 - 4.4
- For speed = Maximum Speed (MX) we consider robot running at speed > 5 or approx > 4.8
- Note: All the above ranges of speed are core values for the respective subset of speed.
- As we have multiple or a range of core values we use trapezoidal membership function

Sundaram®
FOR EDUCATIONAL USE

Based on the assumption the membership functions for speed are given below:

i) Slow Speed

$$\mu SS(s) = \begin{cases} 1 & , s \leq 1 \\ \dfrac{1.6 - s}{0.6} & , 1 \leq s \leq 1.6 \\ 0 & , s \geq 1.6 \end{cases}$$

ii) Medium Speed

$$\mu MS(s) = \begin{cases} 0 & , s \leq 1 \\ \dfrac{s-1}{0.6} & , 1 \leq s \leq 1.6 \\ 1 & , 1.6 \leq s \leq 2.8 \\ \dfrac{3.4 - s}{0.6} & , 2.8 \leq s \leq 3.4 \\ 0 & , s \geq 3.4 \end{cases}$$

iii) Fast Speed

$$\mu FS(s) = \begin{cases} 0 & , s \leq 2.8 \\ \dfrac{s - 2.8}{0.6} & , 2.8 \leq s \leq 3.4 \\ 1 & , 3.4 \leq s \leq 4.4 \\ \dfrac{5 - s}{0.6} & , 4.4 \leq s \leq 5 \\ 0 & , s \geq 5 \end{cases}$$

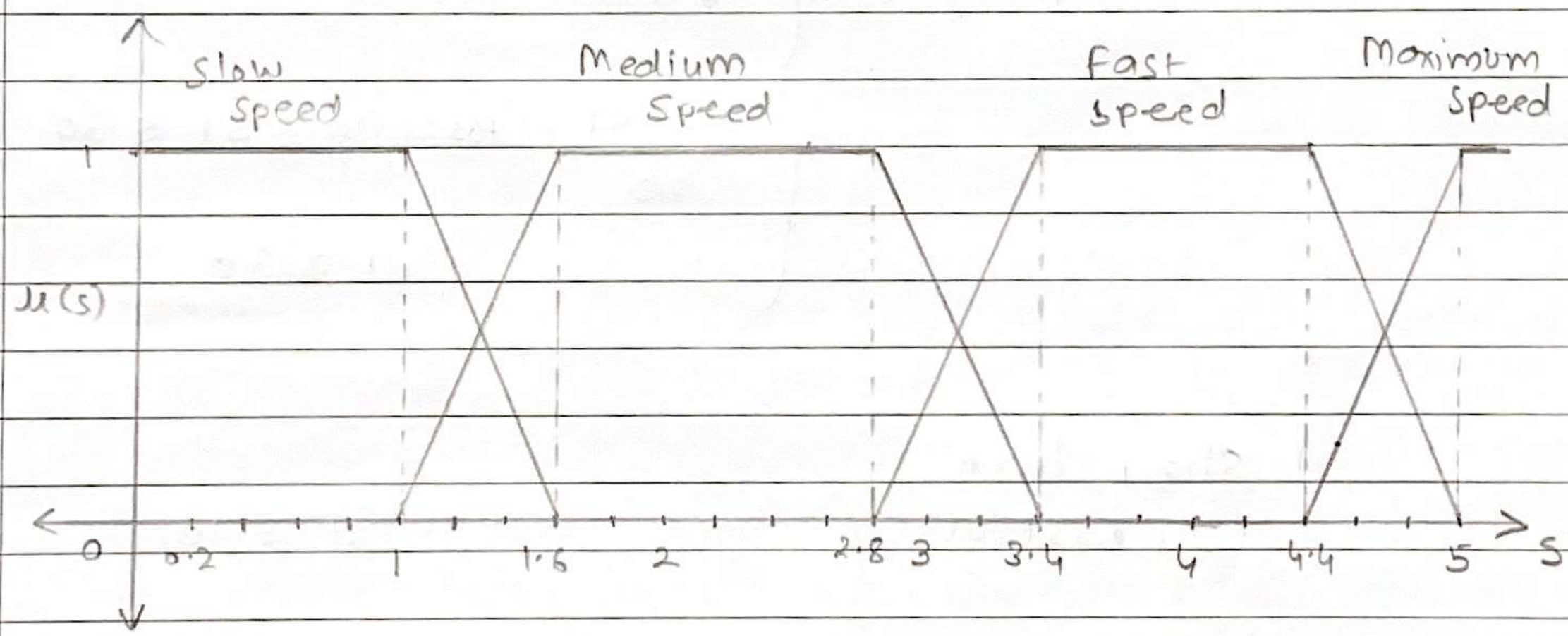
Sundaram

FOR EDUCATIONAL USE

iv) maximum Speed

$$\mu_{Mx}(s) = \begin{cases} 0 & , s \leq 4.4 \\ \dfrac{s - 4.4}{0.6} & , 4.4 \leq s \leq 5 \\ 1 & , s \geq 5 \end{cases}$$

Drawing the membership function for speed we get the following figure.



iv) Steering turn : universe (0 - 90 , step = 1)

Assumptions

- For steering turn = Mild turn (MST) we consider robot to turn between 0 - 10°
- For steering turn = Sharp turn (SST) we consider robot to turn between 30° - 60°
- For steering turn = Very sharp turn (VST) we consider robot to turn between 80° - 90°

Sundaram
FOR EDUCATIONAL USE

- Note: All the above ranges are core values for the respective sub set of steering turn.
- As we have multiple values ~~for~~ as core value for a subset we use trapezoidal membership function.

Based on the assumptions the membership functions for ~~speed~~ steering turn are given below.

i) Mild turn

$$\mu_{MST}(st) = \begin{cases} 1 & , st \leq 10 \\ \dfrac{30 - st}{20} & , 10 \leq st \leq 30 \\ 0 & , st \geq 30 \end{cases}$$

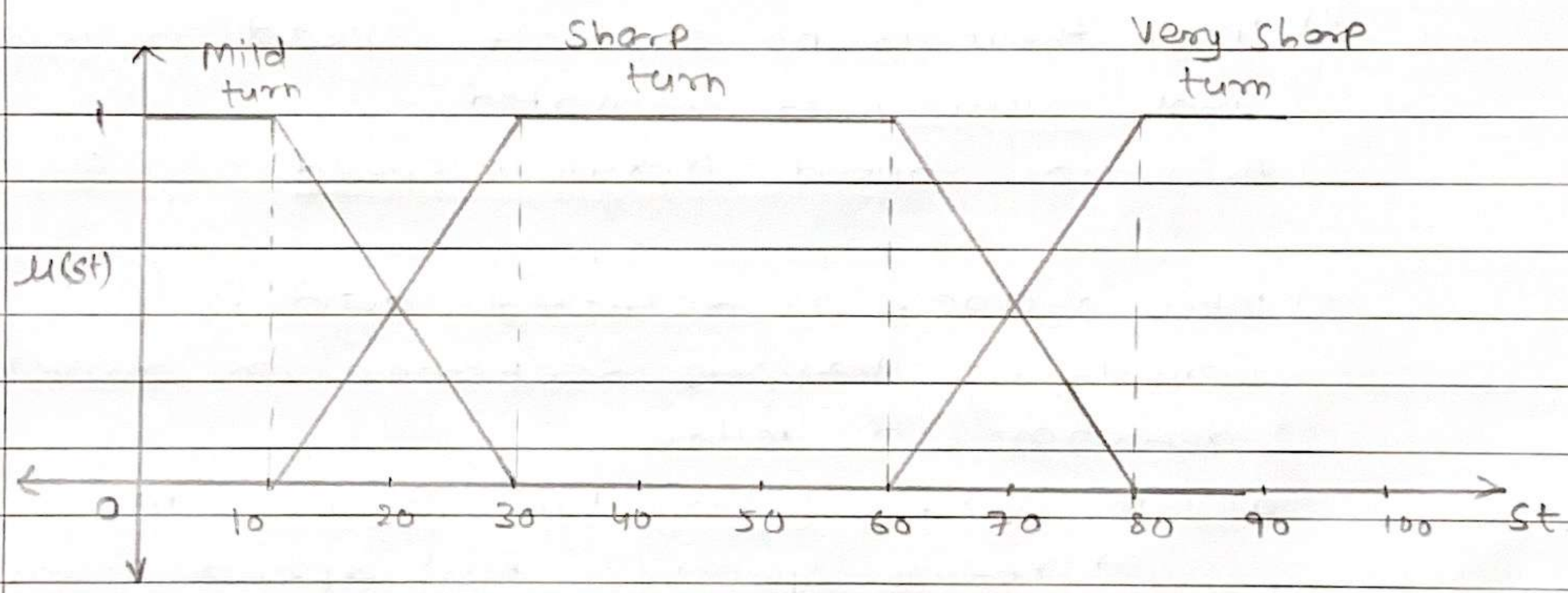
ii) Sharp turn

$$\mu_{SST}(st) = \begin{cases} 0 & , st \leq 10 \\ \dfrac{st - 10}{20} & , 10 \leq st \leq 30 \\ 1 & , 30 \leq st \leq 60 \\ \dfrac{80 - st}{20} & , 60 \leq st \leq 80 \\ 0 & , st \geq 80 \end{cases}$$

iii) very sharp turn

$$\mu_{VST}(st) = \begin{cases} 0 & , st \leq 60 \\ \frac{st-60}{20} & , 60 \leq st \leq 80 \\ 1 & , 80 \leq st \end{cases}$$

Drawing the membership function fo steering turn we get



b) The rule base:

For robot to safely navigate without hitting any obstacles we define a set of rules which are infered by the robot navigation control system. This rules are based on two antecedents "Distance from obstacle" & "Angle with obstacle" controlling two consequent "speed" and "Steering ~~ang~~ turn"

To define rules we design a set a behaviour for robot navigation This set of behaviour is as follows.

i) When there is no obstacle move forward
Assumption: No obstacle is considered when Distance = very far and Angle = large or medium
Speed will be maximum speed and steering turn will not be triggred as we need to move forward without turning.

ii) When there is no obstacle speed is increased and cruising is activated.
Assumption: Speed = maximum speed

iii) When obstacle is detected, avoid it.
Considering various scenarios for obstacle detection as follow.
a) With Distance = 'very far', Angle = small
↳ reduce speed to fast speed from maximum speed
↳ make a mild turn to start avoiding the obstacle
b) with Distance = 'far' & Angle = Large
↳ be at fast speed
↳ make a mild turn to avoid the obstacle.
c) with Distance = 'far' & Angle = medium
↳ further reduce speed to medium speed
↳ start turning more make a sharp turn

Sundaram
FOR EDUCATIONAL USE

d) With Distance = 'Far' & Angle = 'small'
↳ be at medium speed
↳ keep making a sharp turn to avoid obstacle

e) With Distance = 'Near' & Angle = 'Large' or 'Medium'
↳ Now the robot is near so reduce speed to slow speed
↳ make sharp turns

f) with Distance = 'Near' & Angle = ~~Medium~~ 'small'
↳ The obstacle is very close so be at slow speed
↳ make a very sharp turn to avoid it.

iv) Speed is always reduced when turning away or avoiding the obstacle

Based on all of the above behaviours we define set of rules for our fuzzy control system and it is as follows.

Rule 1: If D is VF and A is L, Then S is MX

Rule 2: If D is VF and A is M, Then S is MX

Rule 3: If D is VF and A is S, then S is FS, also ST is MST

Rule 4: If D is F and A is L, then S is FS also ST is MST

Sundaram
FOR EDUCATIONAL USE

Rule 5: If D is F and A is M, Then
S is MS also ST is SST

Rule 6: If D is F and A is S, Then
S is MS also ST is SST

Rule 7: If D is N and A is L, Then
S is SS also ST is SST

Rule 8: If D is N and A is M, Then
S is SS also ST is SST

Rule 9: If D is N and A is S, Then
S is SS also ST is VST

This are the 9 rules used for inference and generating control system for the robot navigation.

c) We will use Mamdani inferencing model with max-min operator. As we have known set of data mamdani is useful over Sugeno model

d) For defuzzification we will use centroid to give us better results as we have range of core values in our membership function so a centroid will return us center of gravity of a particular fuzzy set.

2] Explaination for choices 1-c and 1-d

Inferencing Model:

Mamdani inference model is used as we have known set of inputs and outputs.
Sugeno model is used when no output membership function is present.
In mamdani the consequent is not dependent on function of inputs. Also Mamdani is useful with systems having multiple input single output or multiple input multiple output. Here in robot navigation we had to control multiple outputs i.e. speed and steering turn based on inputs of distance & angle. Sugeno is not useful as it can only be used in system with MISO
As we had designed set of output for speed and steering turns we used mamdani which generates crisp results after defuzzification
Overall Mamdani fits well in this application due to its advantages of being intutive and wide acceptance rate

Defuzzification Method:

Defuzzification is the process of combining the successful fuzzy output set produced by the inference mechanism.
As we are dealing with robot navigation operation LOM or SOM ie. Largest of maximum or smallest of maximum defuzzification method will ~~bot~~ not

be useful because it doesnt consider the weightage of complete membership function which is useful in the navigation application.
Centroid ie. Center of gravity (COG) / Centroid of Area (COA) method considers total area of the membership function distribution used to represent the combined control action.
The combined control actions are divided into a number of sub areas, then ~~COG~~ COG of each sub area is calculated & summation of all the subareas is taken to find the defuzzified value.
The repeatibility of the crisp value is prominent in centroid as centre of gravity of entire fuzzy set is considered
Hence we used centroid as our defuzzifier for generating crisp outputs of speed and steering turn angles.

Sundaram®
FOR EDUCATIONAL USE

Q4 Fuzzy implementation

July 29, 2021

20092002

Problem 4 part 3

[1]:
```
import numpy as np
import skfuzzy as fuzz
from skfuzzy import membership as fuzz_mem
from skfuzzy import control as ctrl
import warnings
warnings.filterwarnings("ignore")

import matplotlib.pyplot as plt
%matplotlib inline
```

Defining universe for each Antecedent and Consequent

[2]:
```
# Universe of Antecedent and Consequent
min_dist = 0; max_dist = 10; step_dist = 0.5;
min_angle = 0; max_angle = 90; step_angle = 1;
min_speed = 0; max_speed = 5; step_speed = 0.2;
min_turn = 0; max_turn = 90; step_turn = 1;
```

Membership function of each Antecedent and Consequent

[3]:
```
# Distance membership function
distance = ctrl.Antecedent(np.arange(min_dist, max_dist+step_dist, step_dist),␣
 ↪'Distance from Obsctacle')
distance['Near'] = fuzz_mem.trapmf(distance.universe, [0, 0, 2, 4])
distance['Far'] = fuzz_mem.trapmf(distance.universe, [2, 4, 7, 9])
distance['Very_Far'] = fuzz_mem.trapmf(distance.universe, [7, 9, 10, 20])
```

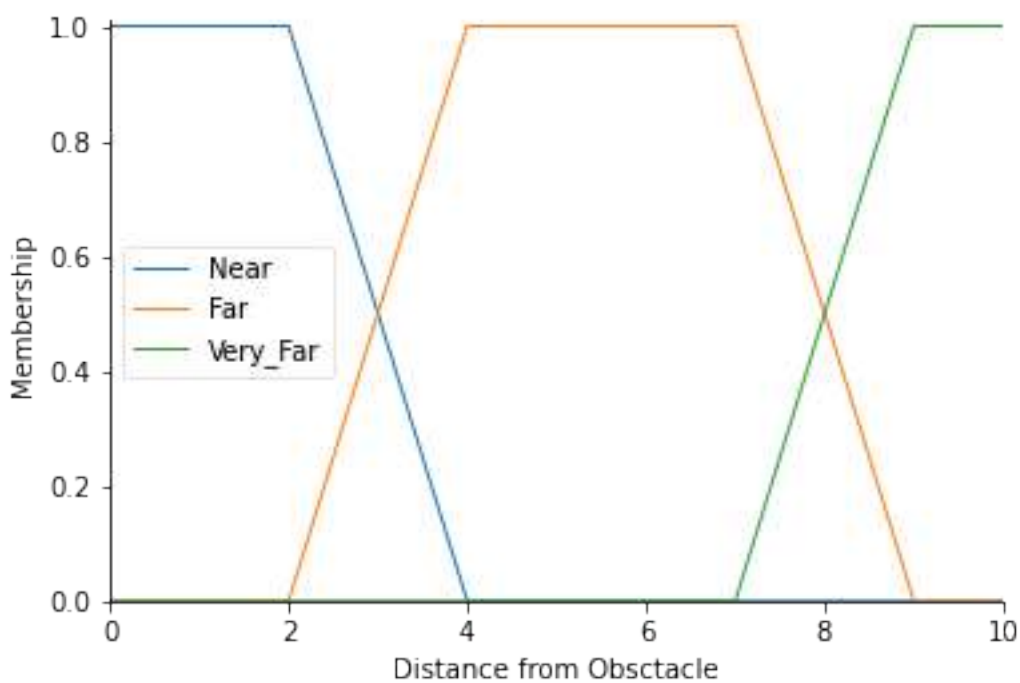
[4]:
```
# Angle from obstacle membership function
angle = ctrl.Antecedent(np.arange(min_angle, max_angle+step_angle, step_angle),␣
 ↪'Angle with obstacle')
angle['Small'] = fuzz_mem.trapmf(angle.universe, [0, 0, 10, 30])
angle['Medium'] = fuzz_mem.trapmf(angle.universe, [10, 30, 60, 80])
angle['Large'] = fuzz_mem.trapmf(angle.universe, [60, 80, 90, 90])
```

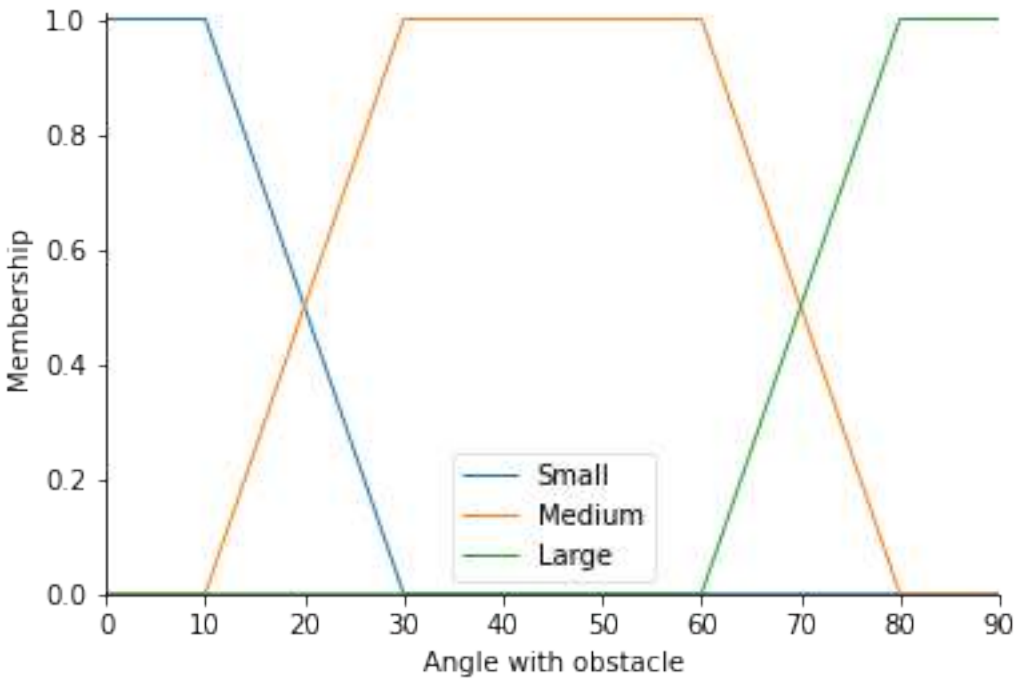
[5]:
```
# Speed membership function
speed = ctrl.Consequent(np.arange(min_speed, max_speed+step_speed, step_speed),␣
↪'Speed')
speed['Slow_Speed'] = fuzz_mem.trapmf(speed.universe, [0, 0, 1, 1.6])
speed['Medium_Speed'] = fuzz_mem.trapmf(speed.universe, [1, 1.6, 2.8, 3.4])
speed['Fast_Speed'] = fuzz_mem.trapmf(speed.universe, [2.8, 3.4, 4.4, 5])
speed['Maximum_Speed'] = fuzz_mem.trapmf(speed.universe, [4.4, 5, 5, 5])
```

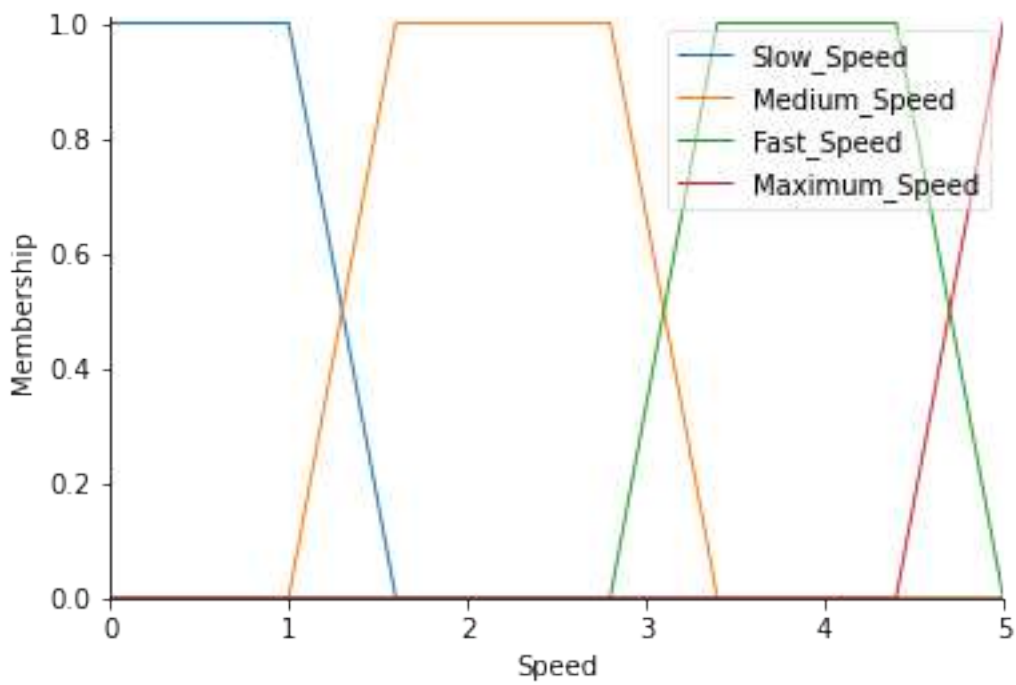
[6]:
```
# Steering Turn membership function
st = ctrl.Consequent(np.arange(min_turn, max_turn+step_turn, step_turn),␣
↪'Steering Turn')
st['MST'] = fuzz_mem.trapmf(st.universe, [0, 0, 10, 30])
st['SST'] =  fuzz_mem.trapmf(st.universe, [10, 30, 60, 80])
st['VST'] = fuzz_mem.trapmf(st.universe, [60, 80, 90, 90])
```

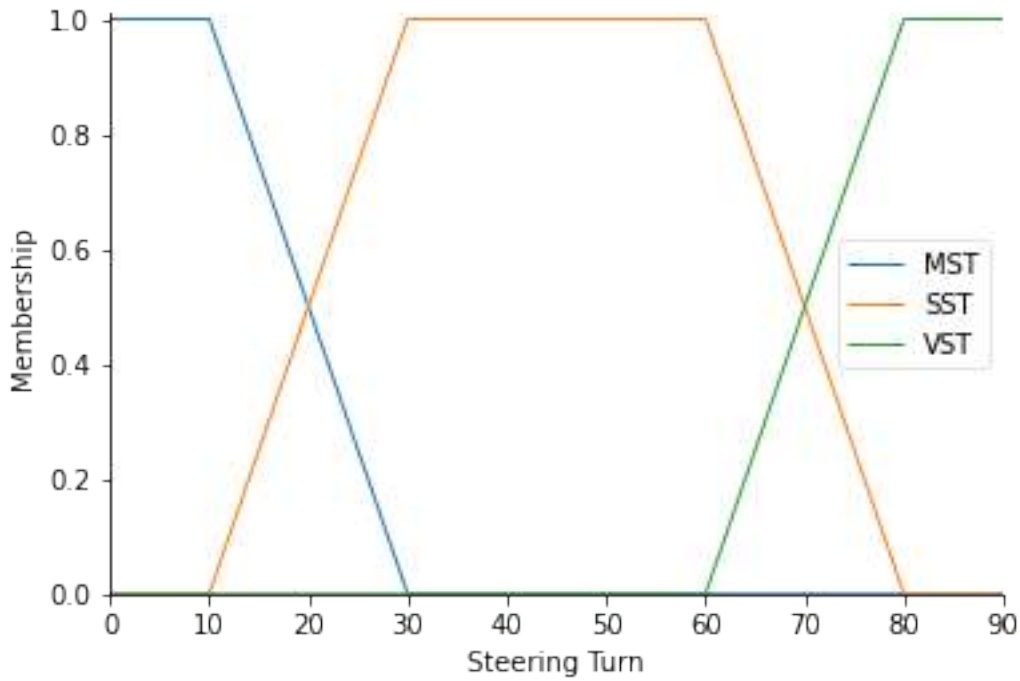
Plot each Membership Function

[7]:
```
distance.view()
angle.view()
speed.view()
st.view()
```




1.0
0.8
0.6
0.4
0.2
0.0
Membership
0
10
20
30
40
50
60
70
80
90
Angle with obstacle
Small
Medium
Large


1.0
0.8
0.6
0.4
0.2
0.0
Membership
0
1
2
3
4
5
Speed
Slow_Speed
Medium_Speed
Fast_Speed
Maximum_Speed



Rules for the Robot navigation control system

[8]:
```
# Rules for Speed
rule1 = ctrl.Rule(distance['Very_Far'] & angle['Large'], speed['Maximum_Speed'])
rule2 = ctrl.Rule(distance['Very_Far'] & angle['Medium'],␣
 ↪(speed['Maximum_Speed'], st['MST']))
rule3 = ctrl.Rule(distance['Very_Far'] & angle['Small'], (speed['Fast_Speed'],␣
 ↪st['MST']))
rule4 = ctrl.Rule(distance['Far'] & angle['Large'], (speed['Fast_Speed'],␣
 ↪st['MST']))
rule5 = ctrl.Rule(distance['Far'] & angle['Medium'], (speed['Medium_Speed'],␣
 ↪st['SST']))
rule6 = ctrl.Rule(distance['Far'] & angle['Small'], (speed['Medium_Speed'],␣
 ↪st['SST']))
rule7 = ctrl.Rule(distance['Near'] & angle['Large'], (speed['Slow_Speed'],␣
 ↪st['SST']))
rule8 = ctrl.Rule(distance['Near'] & angle['Medium'], (speed['Slow_Speed'],␣
 ↪st['SST']))
rule9 = ctrl.Rule(distance['Near'] & angle['Small'], (speed['Slow_Speed'],␣
 ↪st['VST']))
```

Simulating the robot navigation system

[9]:
```
# Applying the rules to the fuzzy Control system
```

```
robot_navigation = ctrl.ControlSystem([rule1, rule2, rule3, rule4,rule5 ,rule6,␣
↪rule7, rule8, rule9])
```

[10]:
```
# Simulating fuzzy control system
robot_nav = ctrl.ControlSystemSimulation(robot_navigation)
```

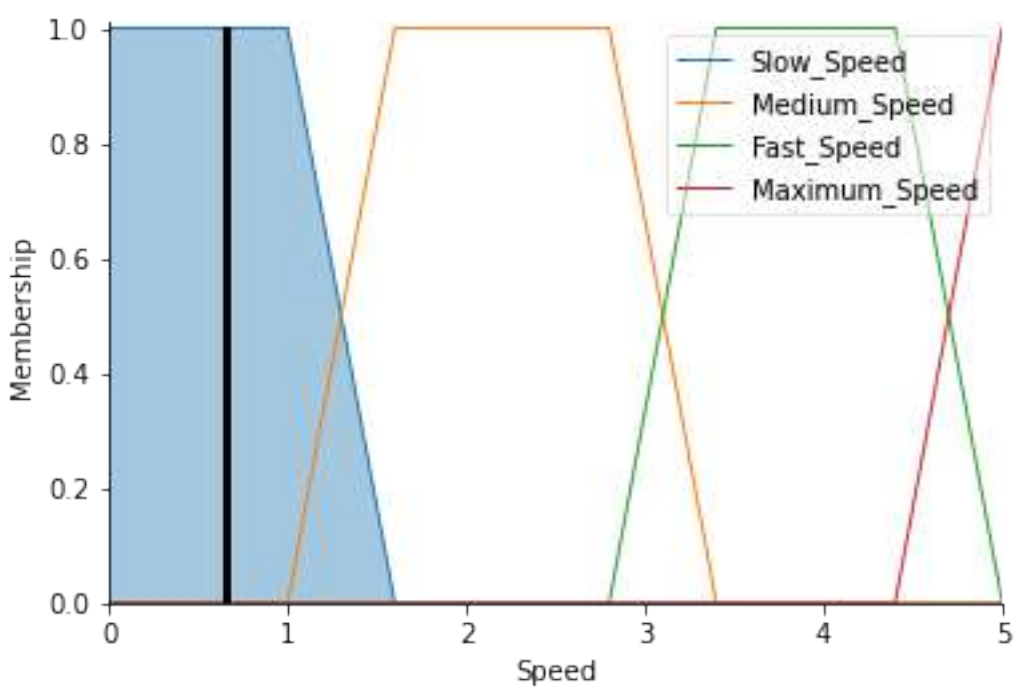
Check for different input values of Distance and Angle from obstacle

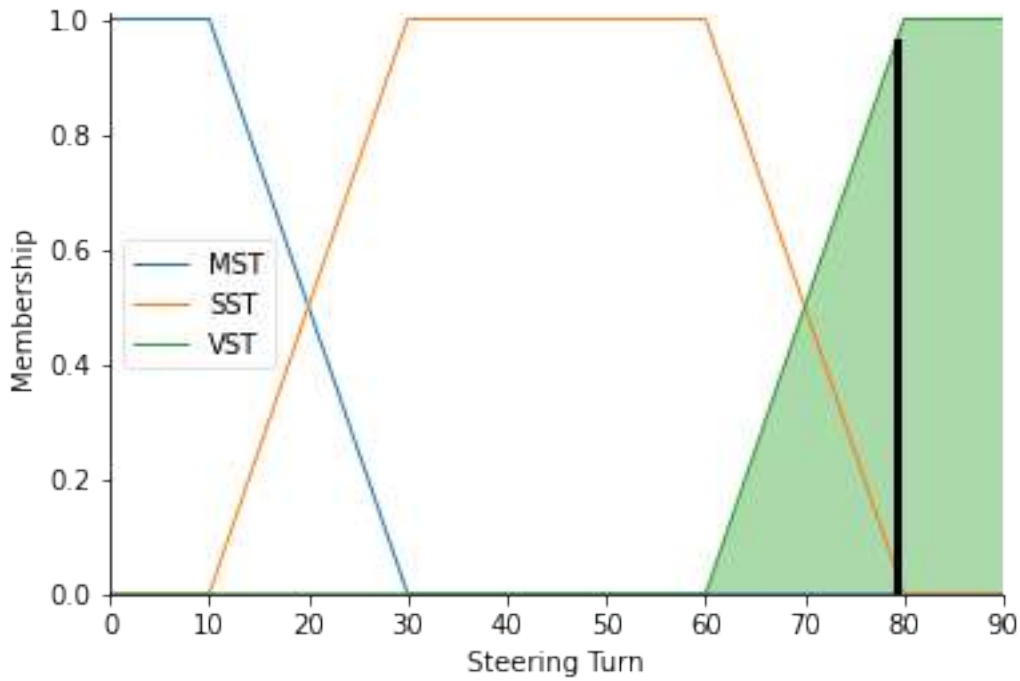
[11]:
```
robot_nav.input['Distance from Obsctacle'] = 2
robot_nav.input['Angle with obstacle'] = 10

robot_nav.compute()

speed.view(sim=robot_nav)
st.view(sim=robot_nav)
print("The Speed value = {}, when Distance from obstace is 2 and Angle with␣
↪obstacle is 10 degree ".format(round(robot_nav.output['Speed'], 3)))
print("The Steering Turn angle = {} degree, when Distance from obstace is 2 and␣
↪Angle with obstacle is 10 degree".format(round(robot_nav.output['Steering␣
↪Turn'], 1)))
```

```
The Speed value = 0.662, when Distance from obstace is 2 and Angle with obstacle
is 10 degree
The Steering Turn angle = 79.2 degree, when Distance from obstace is 2 and Angle
with obstacle is 10 degree
```





As per the given input values the obstacle is very close to the robot hence we get following values Defuzzification used is Centroid and the crisp value obtained is **speed = 0.66** and **steering turn = 79.2 degree** very sharp turn

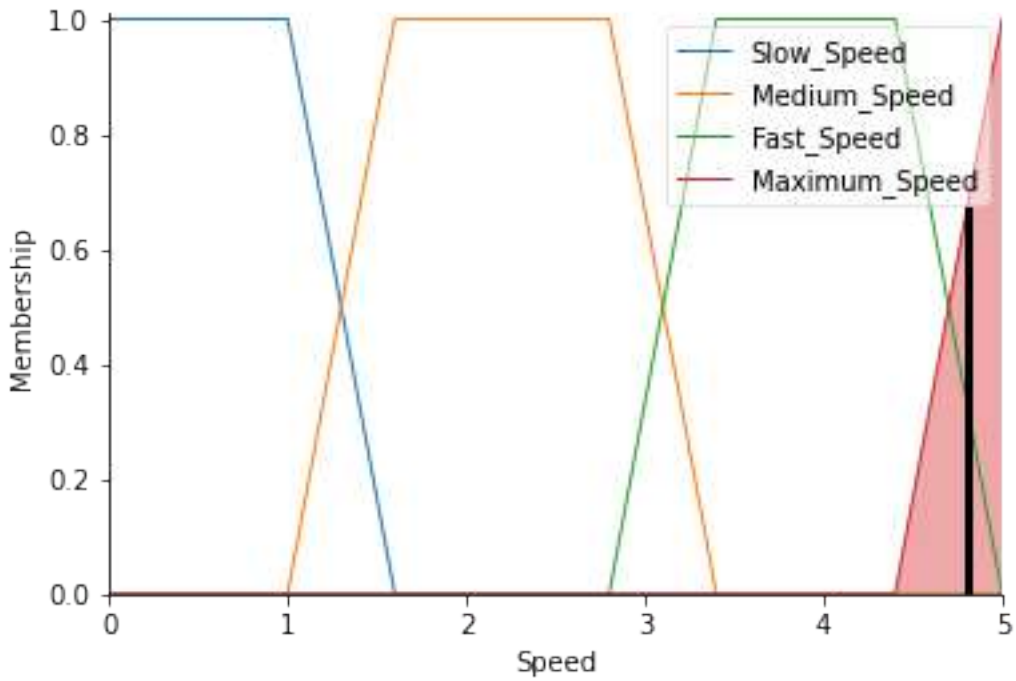
We have choosen the input as **Distance = 2** which is considered as **'Near'** and **Angle = 10** which is **'Small'** we received the values for **speed as 0.66 -> 'Slow speed'** and **steering turn as 79.2 -> 'Very sharp turn'** this is as per set of rules.

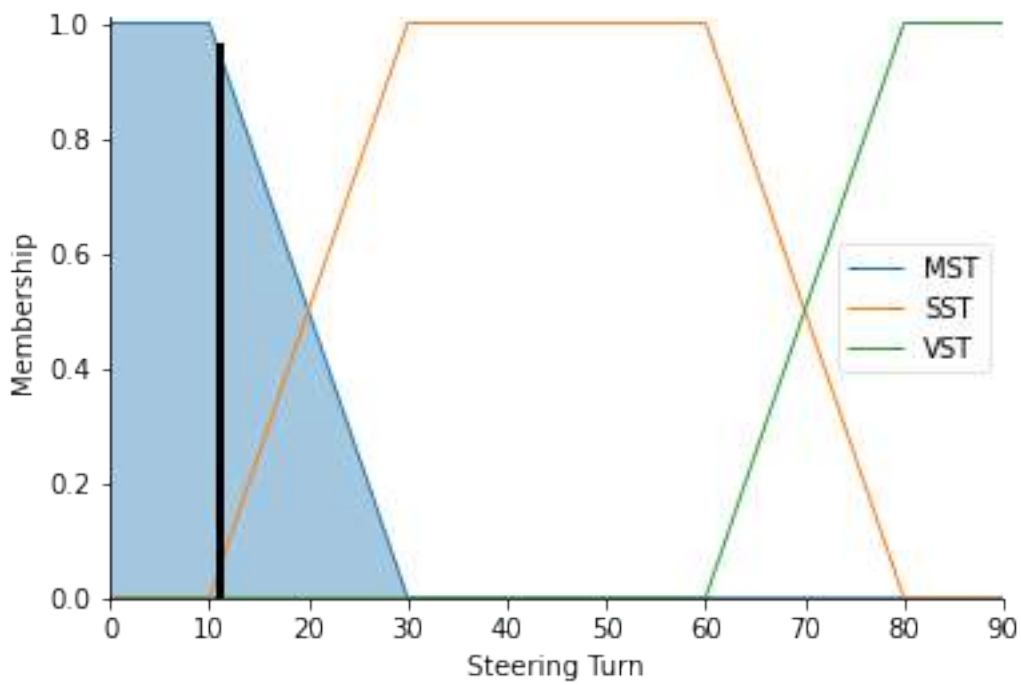
[12]:
```
robot_nav.input['Distance from Obsctacle'] = 10
robot_nav.input['Angle with obstacle'] = 50

robot_nav.compute()

speed.view(sim=robot_nav)
st.view(sim=robot_nav)
print("The Speed value = {}, when Distance from obstace is - 10 and Angle with␣
↪obstacle is 50 degree".format(round(robot_nav.output['Speed'], 3)))
print("The Steering Turn angle = {}, when Distance from obstace is - 10 and␣
↪Angle with obstacle is 50 degree".format(round(robot_nav.output['Steering␣
↪Turn'], 1)))
```

```
The Speed value = 4.8, when Distance from obstace is - 10 and Angle with
obstacle is 50 degree
The Steering Turn angle = 10.8, when Distance from obstace is - 10 and Angle
with obstacle is 50 degree
```





As per the given inputs the obstacle is far away from robot hence we get following output values Defuzzification used is Centroid and the crisp value obtained is ___Speed = 4.8___and **Steering**

Turn = 10.8 degree

We have choosen the input as **Distance = 10** which is considered as **'Very_Far'** and **Angle = 50** which is **'Medium'** we received the values for **speed as 4.8 'almost Maximum speed'** and **steering turn as 10.8 -> 'Mild steering turn'** this is as per set of rules.

0.0.1 Conclusion and Comments

Inference Model - Mamdani

Mamdani inference model is used as we have know set of inputs and outputs. Sugeno membership function is used when output membership function is not present. In this problem, We have mentioned the membership function for the two outputs, hence have used mamdani as the inference system, Mamdani is also used 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 intutive, having wide acceptance rate and it's well-suitedness to human input.

Defuzzifier - Centroid

Converting fuzzy values back to crisp values for further feed it into the system to generate an output is called as Defuzzincation. Here Centroid defuzzification is chosen as it returns the center of gravity of the fuzzy set. This defuzzification is commonly used as it provides better repeatibility of the crisp value and more robust as the center of gravity of the entire fuzzy set is considered