Question_4_A4

July 31, 2021

[6]:
```
# Importing neccessary liberaries
import numpy as np
import skfuzzy as fuzz
from skfuzzy import control as ctl
import warnings
warnings.filterwarnings('ignore')
```

0.0.1 Fuzzy Inference System:

Fuzzy inference systems are used to map inputs or antecedents to outputs/Consequents using fuzzy logic. The decisions of mapping in this is based upon rules defined. These systems are used to design controller using fuzzy control if-then rules and use human expetise for designing that controller. These are robust , inexpensive and efficient control systems which emulates human deductive thinking. The different types of fuzzy systems are Mamdani, Sugemo and Tsukamoto.

Following are the teps needed ti design Fuzzy Controller Using Mamdani inference system.

1) Identification od Antecedents and Consequents : Here , there are 2 antecedents namely, 'Distance' and 'Angle' and 2 Consequents , namely , 'Speed' and 'Steering turn'.
2) Assigning descriptors to Fuzzy subsets.
3) Obtaining membership functions.
4) Making fuzzy rule base.
5) Evaluation of those rules and obtaining fuzzy output.
6) Defuzzification : Ti convert obtained fuzzy values into crisp values.

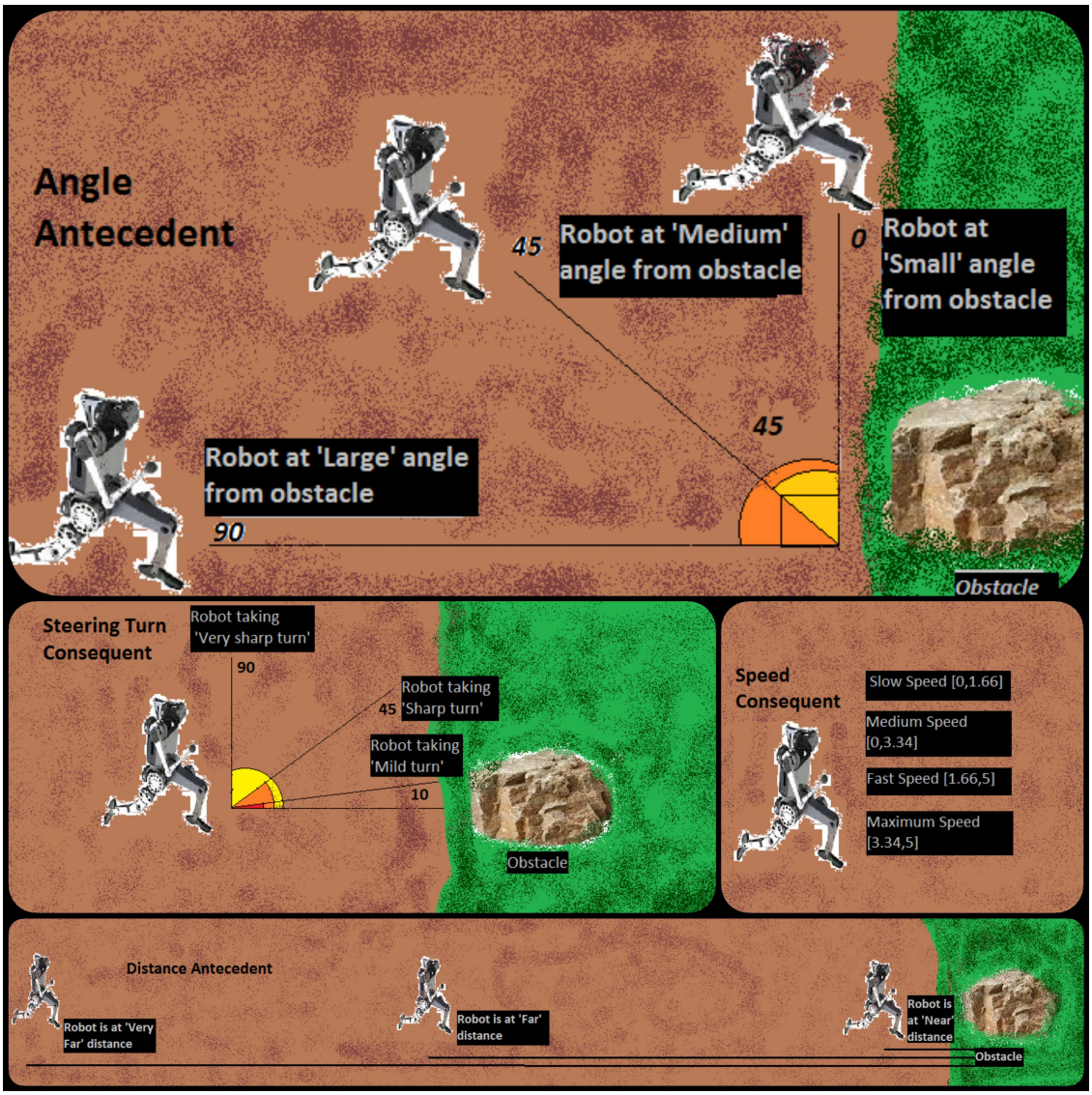
In the following,all of these steps will be explained in detail.

1 Part 1)

1.0.1 Defining Fuzzy quantities:

- Here, Distance and Angle is defined under Antecedent as 'D' and 'A', respectively , they are used as the bases , on which the decisions of Consequents or Speed and Steering_turn will be made.
- The Antecedents are the input or sensor variables in fuzzy control systems.

- The Antecedent Distance varies in range(0,10) with stepsize of 0.5 and has three states namely, 'Near' as 'N','Far' as 'F' and 'Very_far' as 'VF', where 0 is minimum distance and 10 is maximum distance, in given range.
- The Antecedent Angle varies in range(0,90) with stepsize of 1 and has three states namely, 'Small' as 'S','Medium' as 'M' and 'Large' and 'L', where 0 degree is the minimum angle and 90 degree is the maximum angle from the obstacle , in given range.
- The Consequents are the output or control variables in fuzzy control systems.
- The Consequent Speed varies in range(0,5) with stepsize of 0.2 and has four states namely, 'Slow' as 'SS','Medium' as 'MS','Fast' as 'FS' and 'Maximum' as 'MX', where 0 is the minimum speed and 5 is the maximum speed , in the given range.
- The Consequent Steering turn varies in range(0,90) with stepsize of 1 and has three states namely, 'Mild' as 'MST','Sharp' as 'SST' and 'Very_shape' as 'VST', where 0 degree is the minimum steering turn angle and 90 degree is the maximum steering turn angle, in the given range.



[7]:
```
D = ctl.Antecedent(np.arange(0,10.5,0.5),'D')
A = ctl.Antecedent(np.arange(0,91,1),'A')
S = ctl.Consequent(np.arange(0,5.2,0.2),'S')
ST = ctl.Consequent(np.arange(0,91,1),'ST')
```

1.0.2 Applying Membership functions:

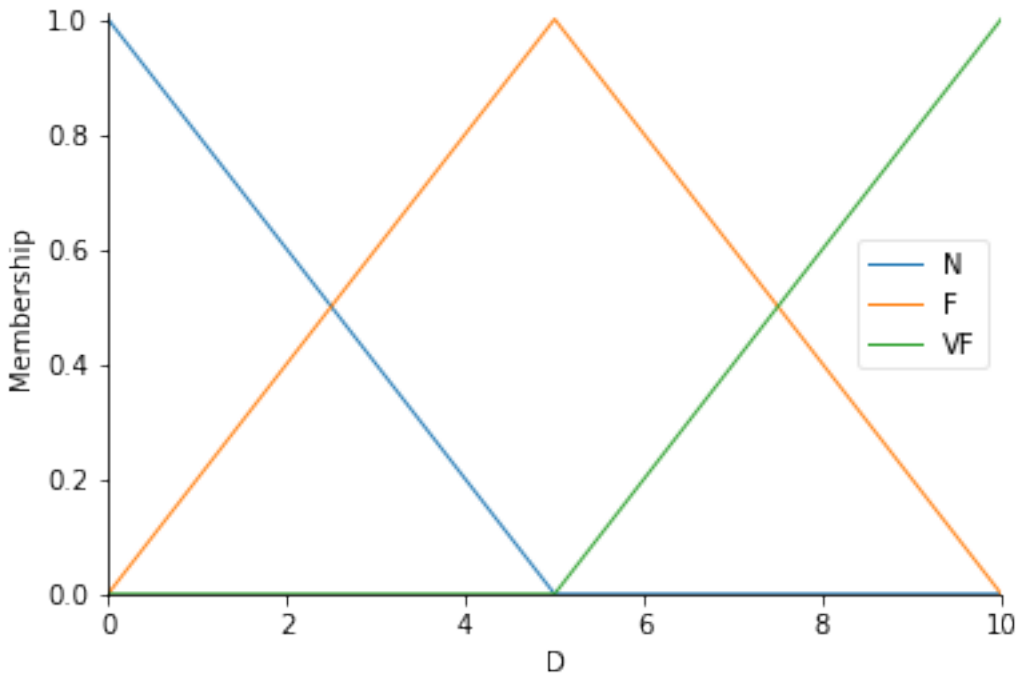
- Membership functions is used to map Set X or input to a values in range [0,1] .This generated value is called degree of member ship. They are used for graphical representation of fuzzy set, where x-axis defines the universe of particular quantity and y-axis is membership value. There are three types of membership funxtions namely, Triangular, Trapezoidal and Gaussian.
- Firstly, for inputs, the automf is applied to generate Term names for both quantities.
- Then, in Speed and Steering_turn Consequents, the trimf is applied, also known as 'Triangular Membership Function'. This MF has lower limit , upper limit and a value m.
- So, in Speed Control, the universe space is divided into four states where 'SS' ranges between [0,1.66] , 'MS' ranges between [0,1.66] and [1.66,3.34] , 'FS' ranges between [1.66,3.34] and [3.34,5] , and 'MX' ranges within [3.34, 5].
- So, in Steering_turn Control, the universe space is divided into three states where 'MST' ranges between [0,45] , 'SST' ranges between [0,45] and [45,90] , and 'VST' ranges within [45,90].
- Here, the convinient method to divide them according to their states turned out to be Triangular Membership function. As trapezoid membership functions are complex and also, according to space complexity, triangular membership functions are better.

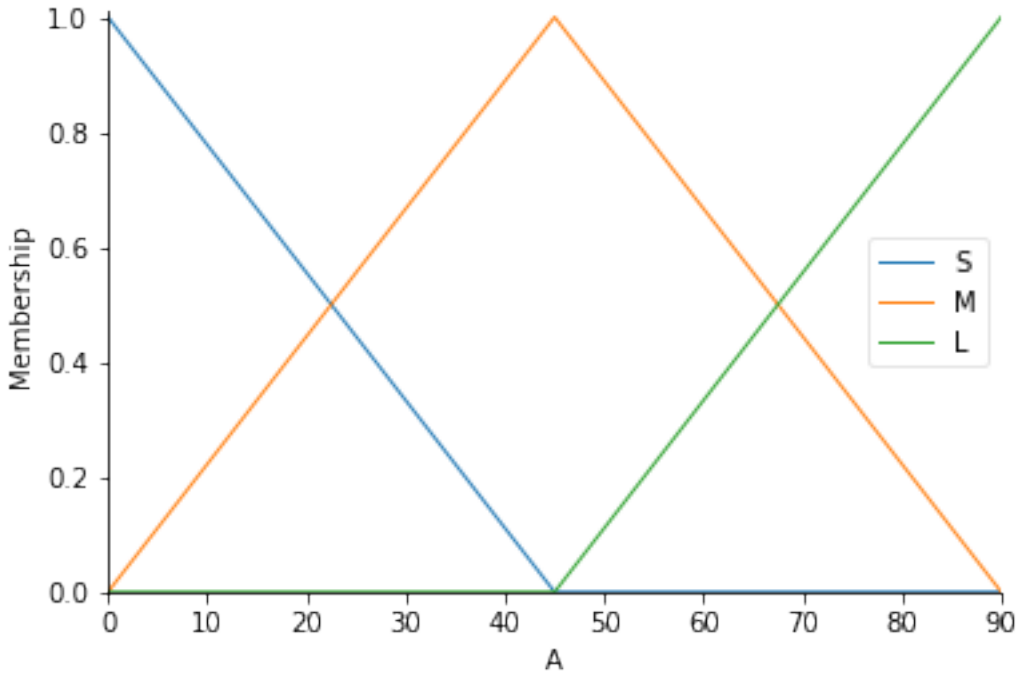
[8]:
```
# applying automf for inputs
D.automf(number=3, names=['N','F','VF'])
A.automf(number=3, names=['S','M','L'])

# applying trimf for outputs
S['SS'] = fuzz.trimf(S.universe,[0,0,(5/3)])
S['MS'] = fuzz.trimf(S.universe,[0,(5/3),(10/3)])
S['FS'] = fuzz.trimf(S.universe,[(5/3),(10/3),5])
S['MX'] = fuzz.trimf(S.universe,[(10/3),5,5])
ST['MST'] = fuzz.trimf(ST.universe,[0,0,45])
ST['SST'] = fuzz.trimf(ST.universe,[0,45,90])
ST['VST'] = fuzz.trimf(ST.universe,[45,90,90])
```

1.0.3 Visualization of fuzzy sets:

Antecedents:

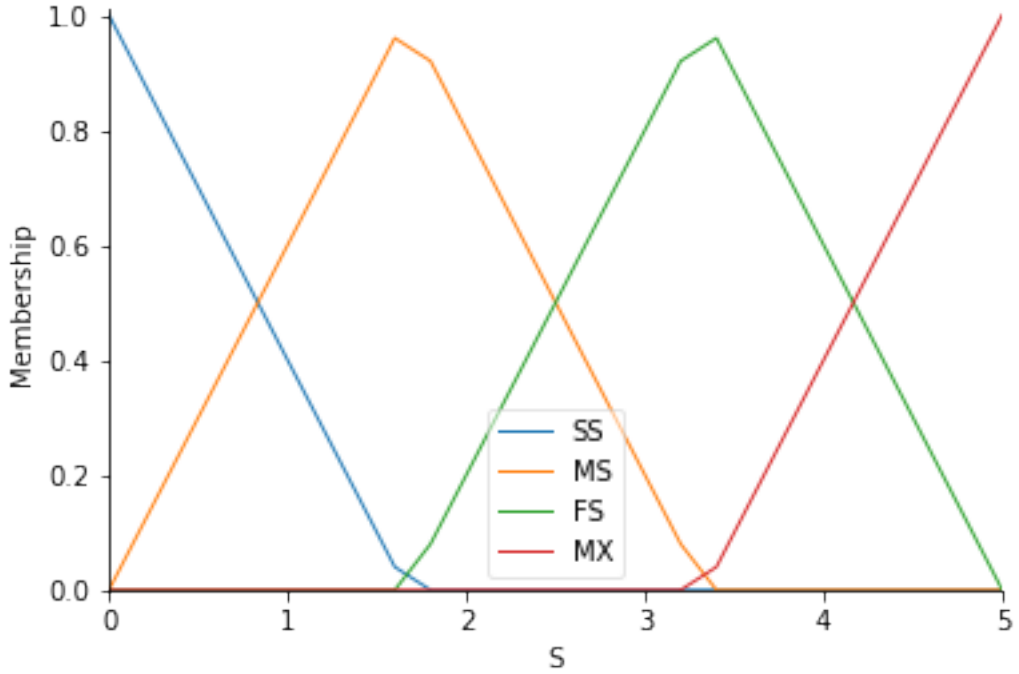
[9]:
```
# Input quantities visualization
D.view()
A.view()
```

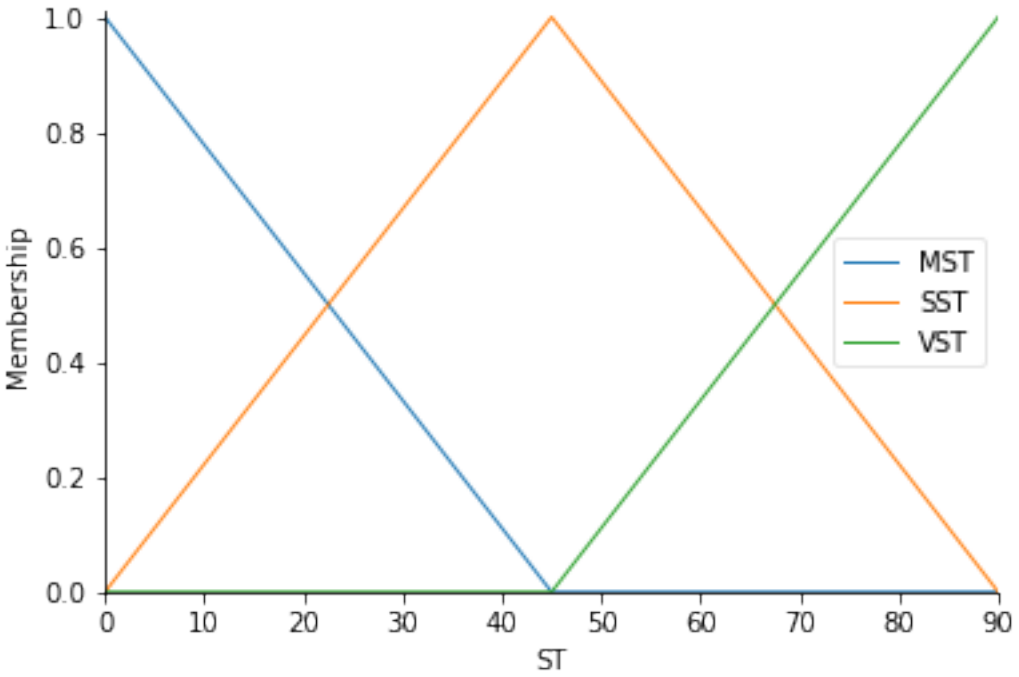




Consequents:

[10]:
```
# Output quantities visualization
S.view()
ST.view()
```





1.1 By hand implementation of Obtaining membership functions:

1. Designing a fuzzy System:

a. Obtaining membership functions for each fuzzy quantity

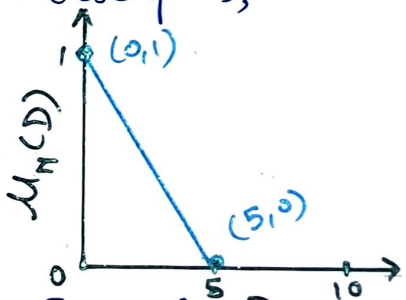
In the given problem, there are two antecedents and two consequents namely, 'D' = Distance, 'A' = Angle and 'S' = Speed, 'ST' = Steering turn, respectively. So, here we will implement Triangular Membership function.

Obtaining Membership function for 'D' input:→

① for $\mu_N(D)$ ('Near'-descriptor);

$$\frac{y_2 - y_1}{x_2 - x_1} = \frac{\mu_N(D) - y_1}{D - x_1}$$

$$\Rightarrow \frac{0-1}{5-0} = \frac{\mu_N(D) - 1}{D - 0}$$



$$\Rightarrow \mu_N(D) = \frac{5-D}{5} \; ; \; [0,5] \text{ — ①eq.}$$

or $0 \le D \le 5$.

② for $\mu_F(D)$ (i.e, 'far' descriptor);

→ for ① line;

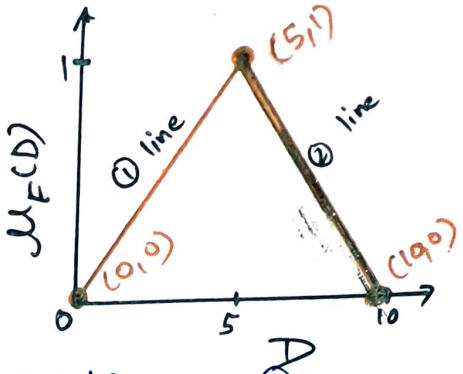
$$\frac{y_2 - y_1}{x_2 - x_1} = \frac{\mu_F(D) - y_1}{D - x_1} \Rightarrow \frac{1-0}{5-0} = \frac{\mu_F(D) - 0}{D - 0}$$

$\Rightarrow \mu_F(D) = \frac{D}{5}$; $0 \le D \le 5$ —— ② eq. (2).

→ For ② line;

$\Rightarrow \frac{y_2 - y_1}{x_2 - x_1} = \frac{\mu_F(D) - y_1}{D - x_1}$

$\Rightarrow \frac{0-1}{10-5} = \frac{\mu_F(D) - 1}{D - 5}$

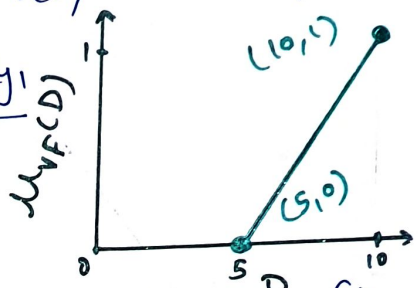


$\Rightarrow \mu_F(D) = \frac{10 - D}{5}$; $5 < D \le 10$ —— ③ eq.

③ For $\mu_{VF}(D)$ (ie, 'Very Far descriptor)

$\Rightarrow \frac{y_2 - y_1}{x_2 - x_1} = \frac{\mu_{VF}(D) - y_1}{D - x_1}$

$\Rightarrow \frac{1-0}{10-5} = \frac{\mu_{VF}(D) - 0}{D - 5}$



$\Rightarrow \mu_{VF}(D) = \frac{D-5}{5}$; $5 \le D \le 10$. —— ④ eq.

Here, using Triangular Member ship function, we obtained equation of lines. And, in order to decide its range, we divided the whole into 'no of descriptors − 1' scales and applied the above membership function. The membership function values lies in between [0,1]. As, it is 'trim', thus, range of 'Near' is [0,5], 'Far' is [0,5] and [5,10]; and 'Very far' is [5,10].

Similarly in order to decide ranges for other antecedents and consequents. in Triangular member ship function, we will divide the whole set into (no. of descriptor − 1) scales and assign ranges ('equal') for all subsets. ③

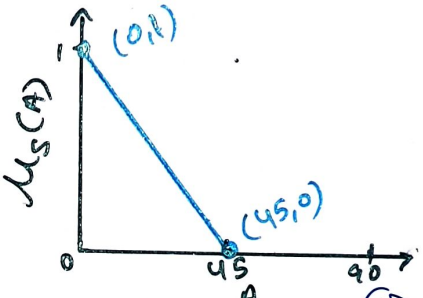
Obtaining Membership function values for 'A' Input :→

① For $\mu_S(A)$ (i.e, 'Small' descriptor).

$$\frac{y_2 - y_1}{x_2 - x_1} = \frac{\mu_S(A) - y_1}{A - x_1}$$

$$\Rightarrow \frac{0-1}{45-0} = \frac{\mu_S(A) - 1}{A - 0}$$

$$\Rightarrow \mu_S(A) = \frac{45 - A}{45}$$



$0 \le A \le 45$ ——— ⑤ eq.

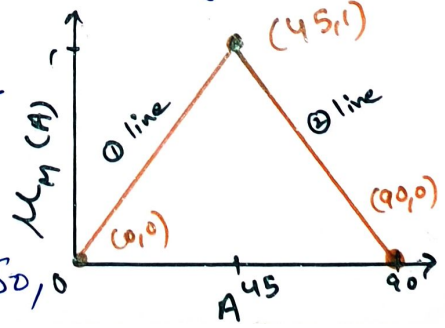
Here, for 'small', the range provided is [0,45].

② For $\mu_M(A)$ (i.e 'Medium' angle descriptor)

$$\frac{y_2 - y_1}{x_2 - x_1} = \frac{\mu_M(A) - y_1}{A - x_1}$$

→ For ① line :

Here range is [0,45]. So, 0



(4)

$$\Rightarrow \frac{1-0}{45-0} = \frac{\mu_M(A) - 0}{A - 0}$$

$$\Rightarrow \mu_M(A) = \frac{A}{45}; \quad 0 \le A \le 45 \text{ ——— (6) eq.}$$

→ For (2) line;
here range is [45, 90]. So,

$$\frac{0-1}{90-45} = \frac{\mu_M(A) - 1}{A - 45}$$

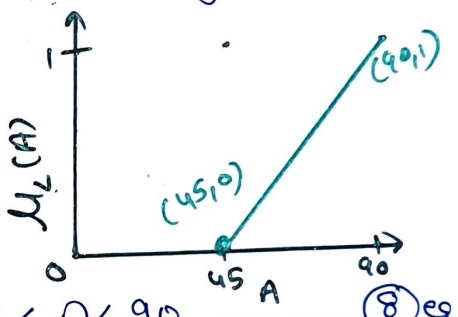
$$\Rightarrow \mu_M(A) = \frac{90 - A}{5}; \quad 45 < A \le 90 \text{ ——— (7) eq.}$$

(3) For $\mu_L(A)$ (i.e 'Large' angle descriptor).

$$\frac{y_2 - y_1}{x_2 - x_1} = \frac{\mu_L(A) - y_1}{A - x_1}$$

$$\Rightarrow \frac{1-0}{90-45} = \frac{\mu_L(A) - 0}{A - 45}$$

$$\Rightarrow \mu_L(A) = \frac{A-45}{45}; \quad 45 \le A \le 90 \text{ ——— (8) eq.}$$



Here, there are 3 descriptors so, it will be divided into (3 − 1 = 2) 2 scales of A which will (90/2 = 45) be. [0, 45] and [45, 90]. Now, Each range is applied to descriptor as per the conditions.

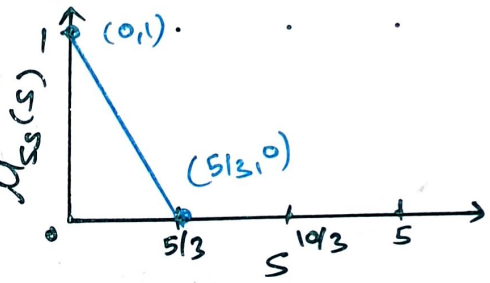
Obtaining Membership function for 'S' Output.

Here, there are 4 descriptors so, the scale will be divided into 4-1 = 3 parts. which will provide the range as. [0, 5/3] ; [5/3, 10/3] and [10/3, 5]. Now, we will apply descriptors in appropriate range and find member function.

① for $\mu_{SS}(S)$ (i.e, 'Slow-Speed' descriptor)

$$\frac{y_2 - y_1}{x_2 - x_1} = \frac{\mu_{SS}(S) - y_1}{S - x_1}$$

$$\Rightarrow \frac{0-1}{5/3 - 0} = \frac{\mu_{SS}(S) - 1}{S - 0}$$



$$\Rightarrow \mu_{SS}(S) = \frac{5 - 3S}{5} \; ; \quad 0 \le S \le (5/3) \quad \text{——— (9) eq.}$$

② for $\mu_{MS}(S)$ (i.e, 'Medium-Speed' descriptor).

$$\frac{y_2 - y_1}{x_2 - x_1} = \frac{\mu_{MS}(S) - y_1}{S - x_1}$$

→ For ① line;
here, range is [0, 5/3].

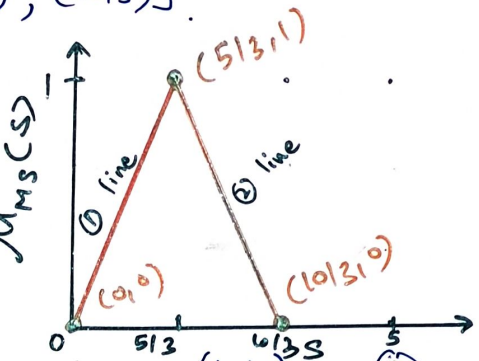
$$\frac{1-0}{(5/3)-0} = \frac{\mu_{MS}(S) - 0}{S - 0} \Rightarrow \frac{\mu_{MS}(S)}{S} = \frac{3}{5}$$

$$\Rightarrow \mu_{MS}(s) = \frac{3s}{5} \quad ; \quad 0 \le s \le (5/3) \qquad \text{---------- (10) eq.}$$

→ For (2) line, here range is $[(5/3), (10/3)]$.

$$\Rightarrow \frac{0-1}{\frac{10}{3}-\frac{5}{3}} = \frac{\mu_{MS}(s)-1}{s-\frac{5}{3}}$$

$$\Rightarrow \frac{-3}{5} = \frac{3(\mu_{MS}(s)-1)}{3s-5}$$



$$\Rightarrow \mu_{MS}(s) = \frac{10-3s}{5} \quad ; \quad (5/3) < s \le (10/3) \qquad \text{— (11) eq.}$$

(3) for $\mu_{FS}(s)$ (i.e, 'Fast Speed' descriptor)

$$\frac{y_2-y_1}{x_2-x_1} = \frac{\mu_{FS}(s)-y_1}{s-x_1}$$

→ for (1) line, here range is $[(5/3), (10/3)]$.

$$\Rightarrow \frac{1-0}{\frac{10}{3}-\frac{5}{3}} = \frac{\mu_{FS}(s)-0}{s-\frac{5}{3}}$$

$$\Rightarrow \mu_{FS}(s) = \frac{3s-5}{5} \quad ; \quad (5/3) \le s \le (10/3)$$

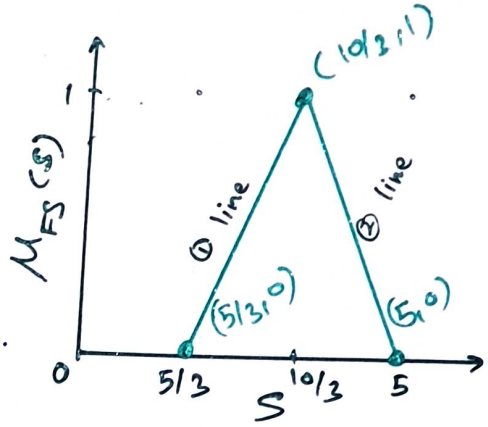
(12) eq.

→ for (2) line, here range is $[(10/3), 5]$.

$$\Rightarrow \frac{0-1}{5-\frac{10}{3}} = \frac{\mu_{FS}(S)-1}{S-\frac{10}{3}}$$

$$\Rightarrow \mu_{FS}(S) = \frac{15-3S}{5}$$

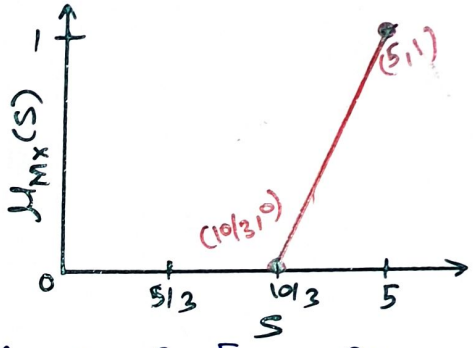
; $(10/3) < S \leq 5$ —— (13) eq.



(4) For $\mu_{MX}(S)$ (i.e, 'Maximum-Speed')

$$\frac{y_2-y_1}{x_2-x_1} = \frac{\mu_{MX}(S)-y_1}{S-x_1}$$

$$\Rightarrow \frac{1-0}{5-\frac{10}{3}} = \frac{\mu_{MX}(S)-0}{S-\frac{10}{3}}$$



$$\Rightarrow \mu_{MX}(S) = \frac{3S-10}{5} \quad ; \quad (10/3) \leq S \leq 5 \text{ —— (14) eq.}$$

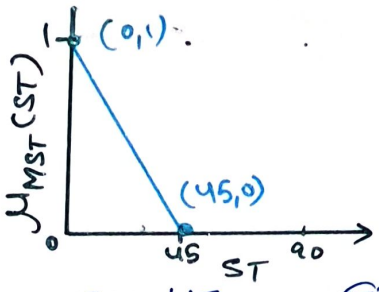
Obtaining Membership function for 'ST' Output.

Here, there are 3 descriptors, thus, the range scale will be divided into 3-1=2 parts, which will provide ranges as [0, 45] and [45, 90]. Therefore, applying 'trimf', their fuzzy subsets will be obtained as:-

① For $\mu_{MST}(ST)$ (ie, 'Mild steering Turn')

$$\frac{y_2 - y_1}{x_2 - x_1} = \frac{\mu_{MST}(ST) - y_1}{ST - x_1}$$

$$\Rightarrow \frac{0-1}{45-0} = \frac{\mu_{MST}(ST) - 1}{ST - 0}$$



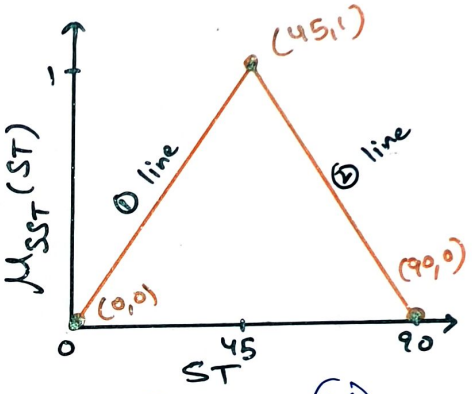
$$\Rightarrow \mu_{MST}(ST) = \frac{45 - ST}{45} \; ; \quad 0 \le ST \le 45 \quad \text{---(15) eq.}$$

② For $\mu_{SST}(ST)$ (ie 'Sharp Steering Turn').

$$\frac{y_2 - y_1}{x_2 - x_1} = \frac{\mu_{SST}(ST) - y_1}{ST - x_1}$$

→ For first line; here range is $[0, 45]$.

$$\Rightarrow \frac{1-0}{45-0} = \frac{\mu_{SST}(ST) - 0}{ST - 0}$$



$$\Rightarrow \mu_{SST}(ST) = \frac{ST}{45} \; ; \quad 0 \le ST \le 45 \quad \text{---(16) eq.}$$

→ For Second line; here range is $[45, 90]$.

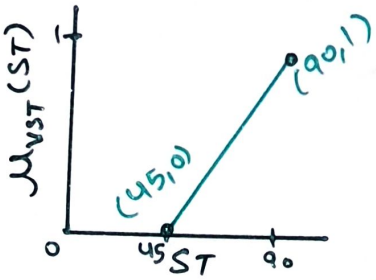
$$\Rightarrow \frac{0-1}{90-45} = \frac{\mu_{SST}(ST) - 1}{ST - 45}$$

$$\Rightarrow \mu_{SST}(ST) = \frac{90 - ST}{45} \; ; \quad 45 < ST \le 90 \quad \text{---(17) eq.}$$

③ For $\mu_{VST}(ST)$ (i.e., 'Very Sharp Turn')

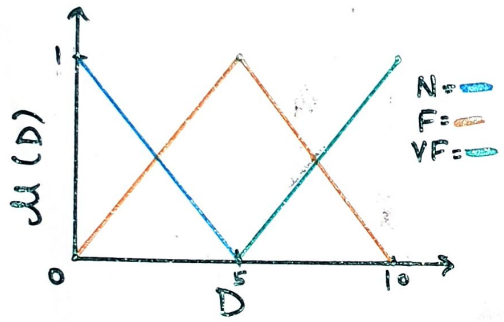
$$\frac{y_2 - y_1}{x_2 - x_1} = \frac{\mu_{VST}(ST) - y_1}{ST - x_1}$$

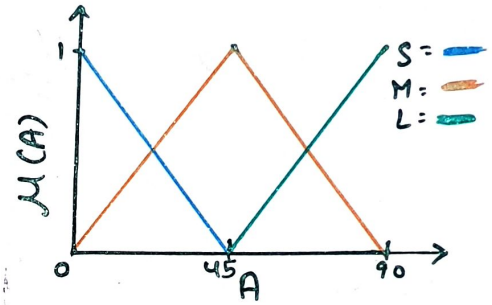
$$\Rightarrow \frac{1-0}{90-45} = \frac{\mu_{VST}(ST) - 0}{ST - 45}$$



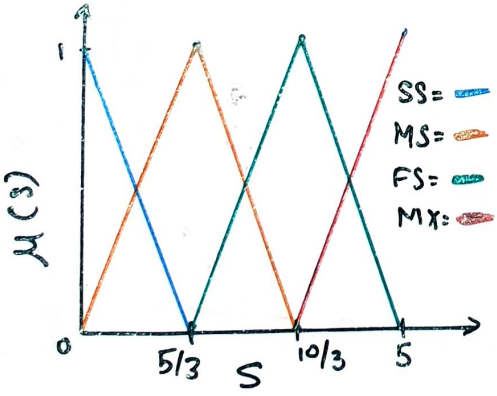
$$\Rightarrow \mu_{VST}(ST) = \frac{ST-45}{45} \; ; \quad 45 \le ST \le 90 \quad \text{---} \quad (18)\,eq.$$

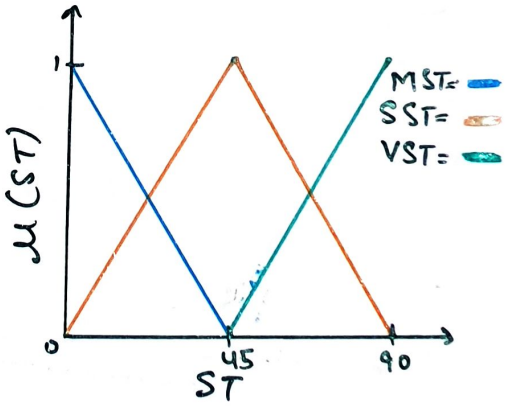
Plotting fuzzy sets of Antecedents.





Plotting fuzzy sets of Consequents.

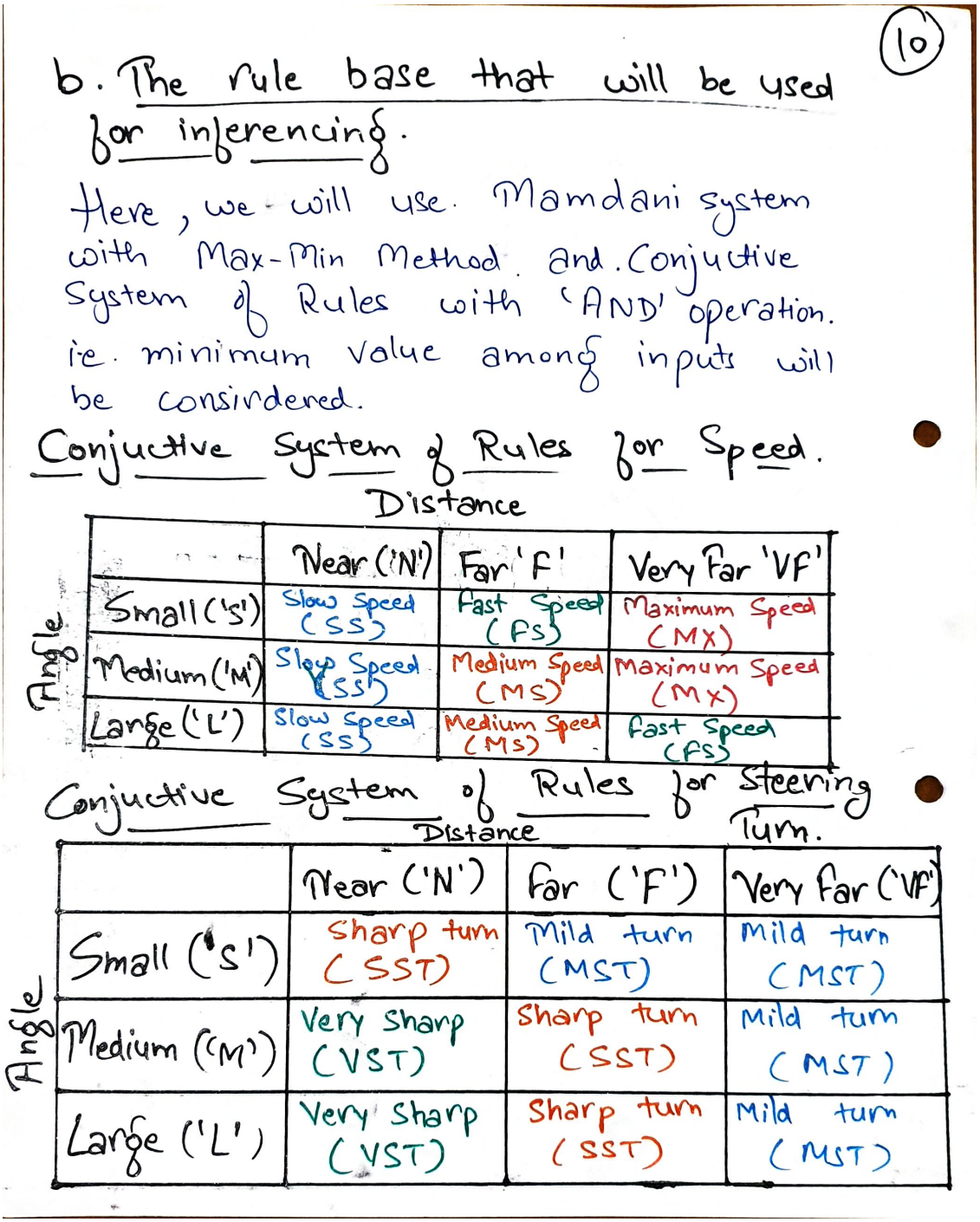




1.1.1 Implementing Inference System and Defining Rules:

- As, membership functions are already defined thus, now rule base will be created for 'Steering turn' and 'Speed' output.
- In case of Distance input , if the object is near , there is a need to take significant action, whereas if distance is far away , the situation is not that critical and minor change in speed of vehicle and angle of steering turn can avoid collision. Thus, it can be said , less is the distance , more crtical is the situation and there is immediate need of great change in outputs state, or vice-versa .
- Here, the more is the angle , the more critical the situation will be as , if angle is about 80 or 90 degree, then , collision will definately happen , but if angle is small , there may chances that the robot will not collide as hard as large angle and may pass by , just touching it. So, large is the angle , critical is the situation which arises immediate need of huge change in outputs , or vice-versa.

1.2 By hand implementation of Fuzzy Rule Base:



b. The rule base that will be used for inferencing.

Here, we will use Mamdani system with Max-Min Method and Conjuctive System of Rules with 'AND' operation. ie. minimum value among inputs will be considered.

Conjuctive System of Rules for Speed.

Angle \ Distance	Near ('N')	Far 'F'	Very Far 'VF'
Small ('S')	Slow Speed (SS)	Fast Speed (FS)	Maximum Speed (MX)
Medium ('M')	Slow Speed (SS)	Medium Speed (MS)	Maximum Speed (MX)
Large ('L')	Slow Speed (SS)	Medium Speed (MS)	Fast Speed (FS)

Conjuctive System of Rules for Steering Turn.

Angle \ Distance	Near ('N')	far ('F')	Very far ('VF')
Small ('S')	Sharp turn (SST)	Mild turn (MST)	Mild turn (MST)
Medium ('M')	Very Sharp (VST)	Sharp turn (SST)	Mild turn (MST)
Large ('L')	Very Sharp (VST)	Sharp turn (SST)	Mild turn (MST)

for 'Steering Turn':

- Rule 1: If Distance is 'Near' AND Angle is 'Large' , then Steering turn will be 'Very Sharp Turn'.
- Rule 2: If Distance is 'Near' AND Angle is 'Medium' , then Steering turn will be 'Very Sharp Turn'.

- Rule 3: If Distance is 'Near' AND Angle is 'Small' , then Steering turn will be 'Sharp Turn'.
- Rule 4: If Distance is 'Far' AND Angle is 'Large' , then Steering turn will be 'Sharp Turn'.
- Rule 5: If Distance is 'Far' AND Angle is 'Medium' , then Steering turn will be 'Sharp Turn'.
- Rule 6: If Distance is 'Far' AND Angle is 'Small' , then Steering turn will be 'Mild Turn'.
- Rule 7: If Distance is 'Very Far' AND Angle is 'Large' , then Steering turn will be 'Mild Turn'.
- Rule 8: If Distance is 'Very Far' AND Angle is 'Medium' , then Steering turn will be 'Mild Turn'.
- Rule 9: If Distance is 'Very Far' AND Angle is 'Small' , then Steering turn will be 'Mild Turn'.

Also, If maximum Distance =10 and minimum angle = 0 is given , then it will take it as that noi obstacle is detected,thus , there will be no change in Steering turn angle.. Else, in all other conditions, obstacle is detected.

[11]:
```
# Function that returns the output of Fuzzy system for 'ST' Consequent
def Inference_system_turn(ST,d,a):
    Rule1_turn = ctl.Rule(D['N'] & A['L'], ST['VST'])
    Rule2_turn = ctl.Rule(D['N'] & A['M'], ST['VST'])
    Rule3_turn = ctl.Rule(D['N'] & A['S'], ST['SST'])
    Rule4_turn = ctl.Rule(D['F'] & A['L'], ST['SST'])
    Rule5_turn = ctl.Rule(D['F'] & A['M'], ST['SST'])
    Rule6_turn = ctl.Rule(D['F'] & A['S'], ST['MST'])
    Rule7_turn = ctl.Rule(D['VF'] & A['L'], ST['MST'])
    Rule8_turn = ctl.Rule(D['VF'] & A['M'], ST['MST'])
    Rule9_turn = ctl.Rule(D['VF'] & A['S'], ST['MST'])

    base_turn = ctl.ControlSystem([Rule1_turn,Rule2_turn,Rule3_turn,Rule4_turn,
                                   ␣
↪Rule5_turn,Rule6_turn,Rule7_turn,Rule8_turn,Rule9_turn])
    system_turn = ctl.ControlSystemSimulation(base_turn)

    system_turn.input['D'] = d
    system_turn.input['A'] = a
    system_turn.compute()
    Turn_output=system_turn.output['ST']

    # Displaying the computed output
    print('The computed steering turn is: {}'.format(Turn_output))
    ST.view(sim = system_turn)

    return Turn_output
```

for 'Speed':

- Rule 1: If Distance is 'Near' AND Angle is 'Large' , then Speed will be 'Slow'.
- Rule 2: If Distance is 'Near' AND Angle is 'Medium' , then Speed will be 'Slow'.
- Rule 3: If Distance is 'Near' AND Angle is 'Small' , then Speed will be 'Slow'.
- Rule 4: If Distance is 'Far' AND Angle is 'Large' , then Speed will be 'Medium'.

- Rule 5: If Distance is 'Far' AND Angle is 'Medium' , then Speed will be 'Medium'.
- Rule 6: If Distance is 'Far' AND Angle is 'Small' , then Speed will be 'Fast'.
- Rule 7: If Distance is 'Very Far' AND Angle is 'Large' , then Speed will be 'Fast'.
- Rule 8: If Distance is 'Very Far' AND Angle is 'Medium' , then Speed will be 'Maximum'.
- Rule 9: If Distance is 'Very Far' AND Angle is 'Small' , then Speed will be 'Maximum'.

Also, If maximum Distance =10 and minimum angle = 0 is given , then it will take it as that no obstacle is detected,thus , Speed can be increased upto 'Maximum'. Else, in all other conditions, obstacle is detected.

[12]:
```
# Function that returns the output of Fuzzy system for 'S' Consequent
def Inference_system_S(S,d,a):
    Rule1_S = ctl.Rule(D['N'] & A['L'], S['SS'])
    Rule2_S = ctl.Rule(D['N'] & A['M'], S['SS'])
    Rule3_S = ctl.Rule(D['N'] & A['S'], S['SS'])
    Rule4_S = ctl.Rule(D['F'] & A['L'], S['MS'])
    Rule5_S = ctl.Rule(D['F'] & A['M'], S['MS'])
    Rule6_S = ctl.Rule(D['F'] & A['S'], S['FS'])
    Rule7_S = ctl.Rule(D['VF'] & A['L'], S['FS'])
    Rule8_S = ctl.Rule(D['VF'] & A['M'], S['MX'])
    Rule9_S = ctl.Rule(D['VF'] & A['S'], S['MX'])


    base_S = ctl.
↪ControlSystem([Rule1_S,Rule2_S,Rule3_S,Rule4_S,Rule5_S,Rule6_S,Rule7_S,Rule8_S,Rule9_S])
    system_S = ctl.ControlSystemSimulation(base_S)

    system_S.input['D'] = d
    system_S.input['A'] = a
    system_S.compute()
    S_output=system_S.output['S']

    # Displaying the computed output
    print('The computed S is: {}'.format(S_output))
    S.view(sim = system_S)

    return S_output
```

2 Part 2)

2.0.1 The Used Inferencing System - Mamdani:

- The used Inference system is Mamdani in above implementation and is suitable for Decision Support applications.
- The main distinguishable factor among them is that Mamdani use defuzzification for crisp outputs whereas , the Sugeno uses the weighted average to compute crisp output.
- Apart from this, Mamdani has output membership functions whereas Sugeno has no output memership functions. In our problem, output membership function place an important role

for decision making.

- Due to more human-like , intutive nature, expressive power and interpretibility, Mamdani is widely used and better.
- Also, Sugeno system returns the output as the function of input x and y, which is not suitable for given problem.Thus, Mamdani metthod is implemented here.

2.1 Implementation of Different Defuzzification methods:

2.1.1 Defuzzification method: 'centroid'

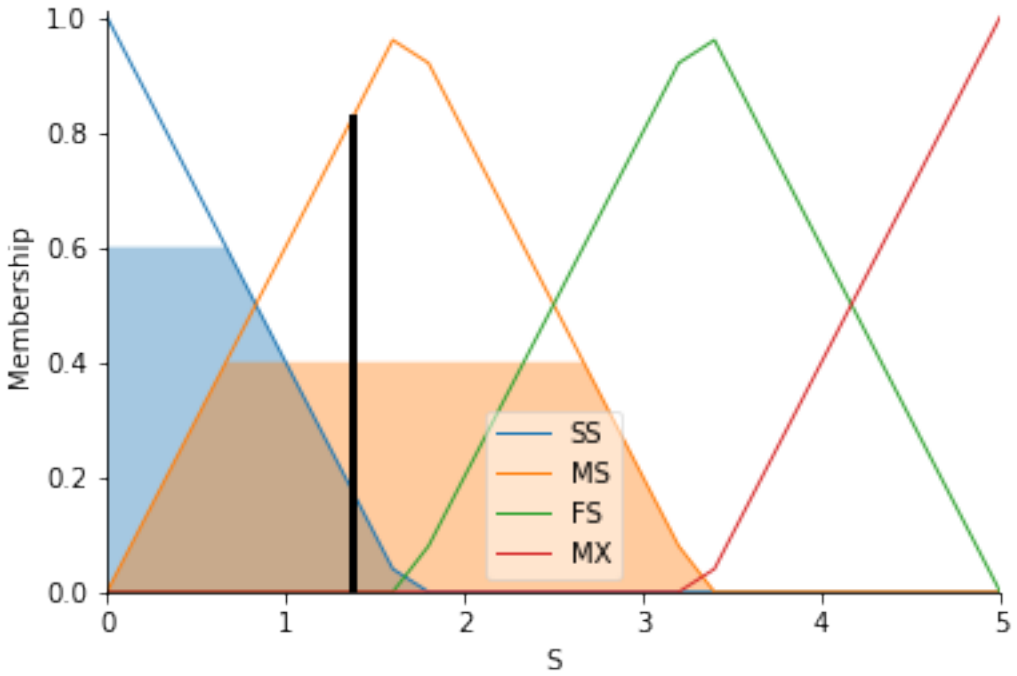
[13]:
```
# Change defuzzification method to centroid
S.defuzzify_method = 'centroid'
ST.defuzzify_method ='centroid'
```

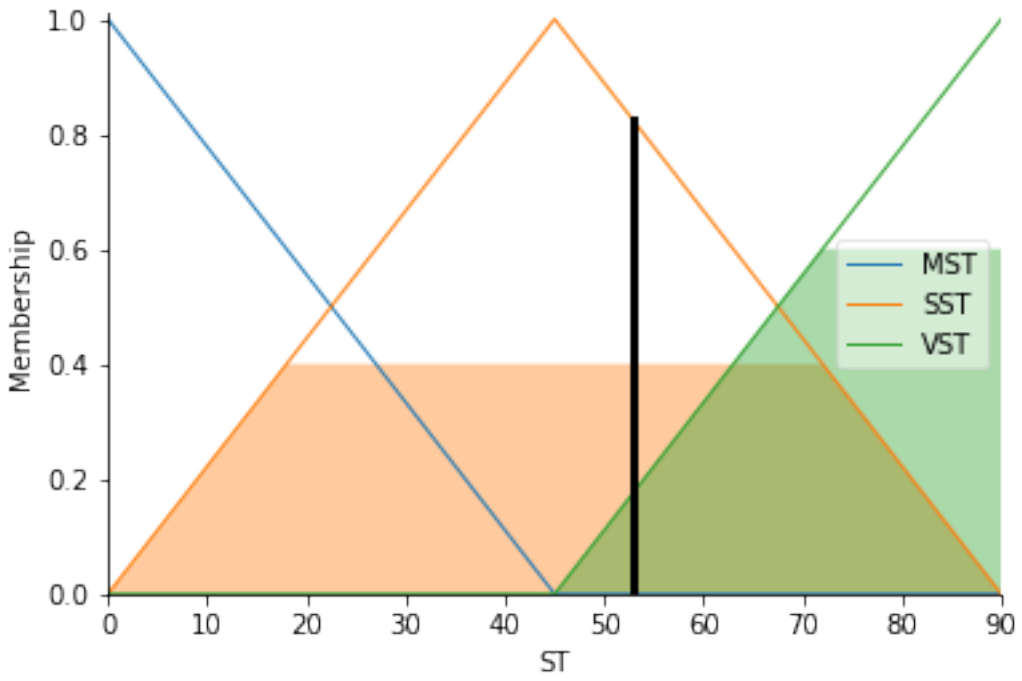
For input where object is detected:

[14]:
```
# Calling the functions
Inference_system_S(S,2,80)
Inference_system_turn(ST,2,80)
```

```
The computed S is: 1.3777561397814566
The computed steering turn is: 52.902439024390226
```

[14]: 52.902439024390226



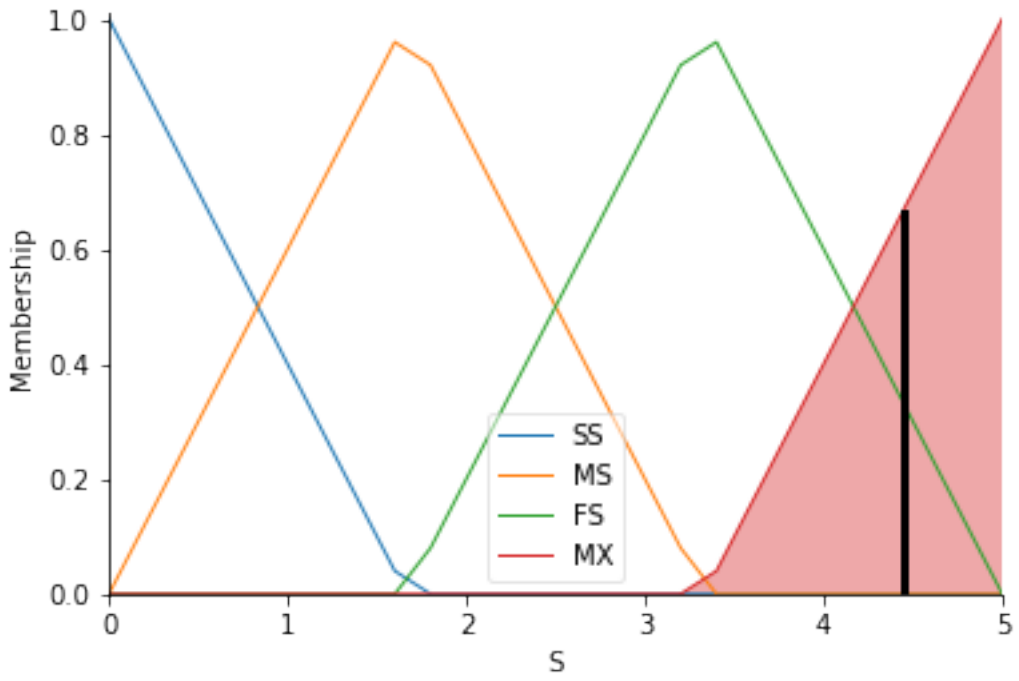


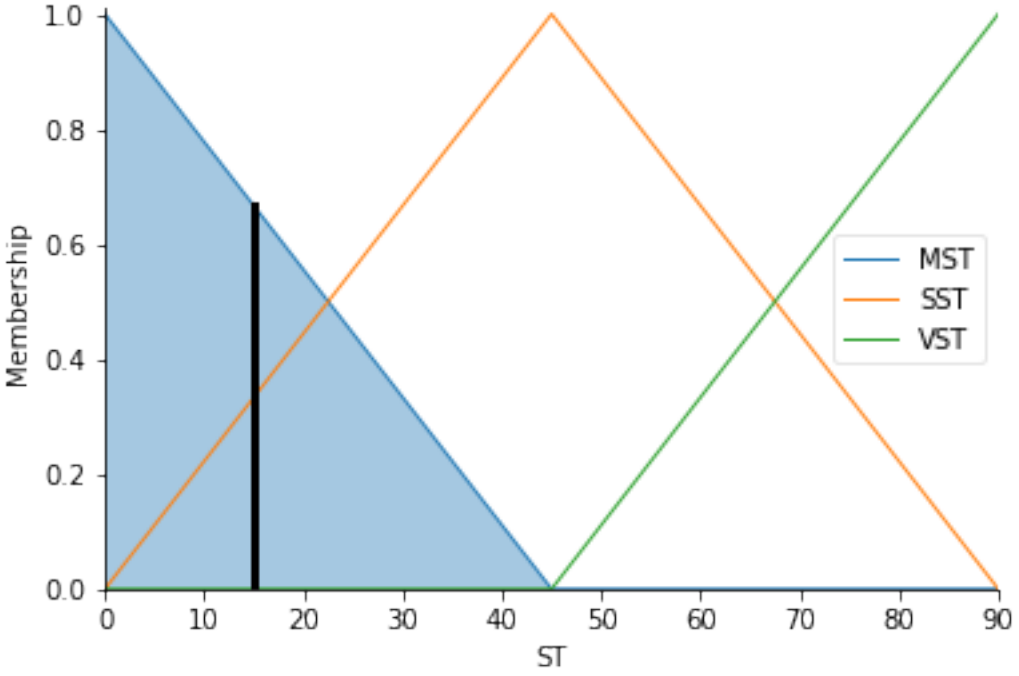
For input where object is not detected:

[15]:
```
Inference_system_S(S,10,0)
Inference_system_turn(ST,10,0)
```

```
The computed S is: 4.440829346092504
The computed steering turn is: 14.999999999999991
```

[15]: 14.999999999999991


1.0
0.8
0.6
0.4
0.2
0.0
Membership
SS
MS
FS
MX
0
1
2
3
4
5
S


1.0
0.8
0.6
0.4
0.2
0.0
Membership
MST
SST
VST
0
10
20
30
40
50
60
70
80
90
ST

2.1.2 Defuzzification method: 'bisector'

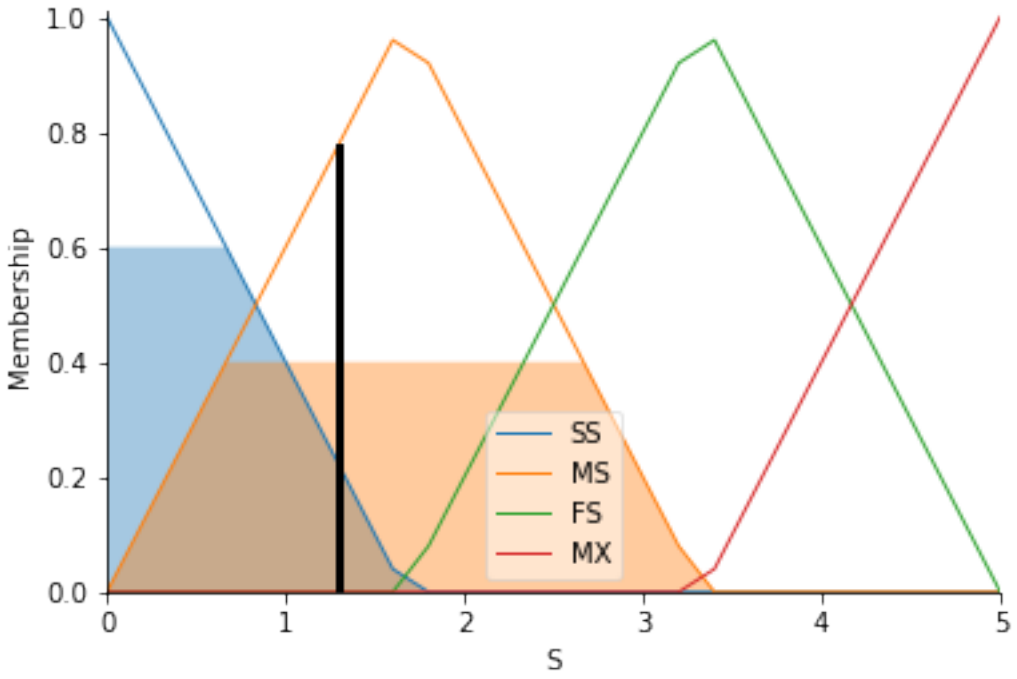
[16]:
```
# Change defuzzification method to bisector
S.defuzzify_method = 'bisector'
ST.defuzzify_method ='bisector'
```

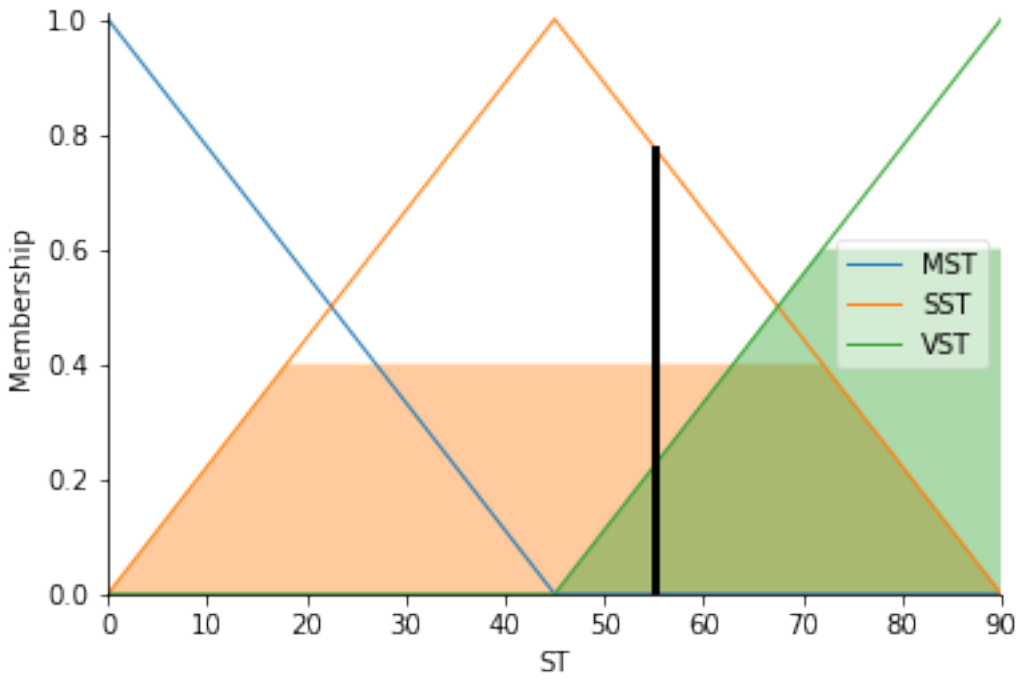
For input where object is detected:

[17]:
```
# calling the functions
Inference_system_S(S,2,80)
Inference_system_turn(ST,2,80)
```

```
The computed S is: 1.2949999999999997
The computed steering turn is: 55.125000000000014
```

[17]:
```
55.125000000000014
```



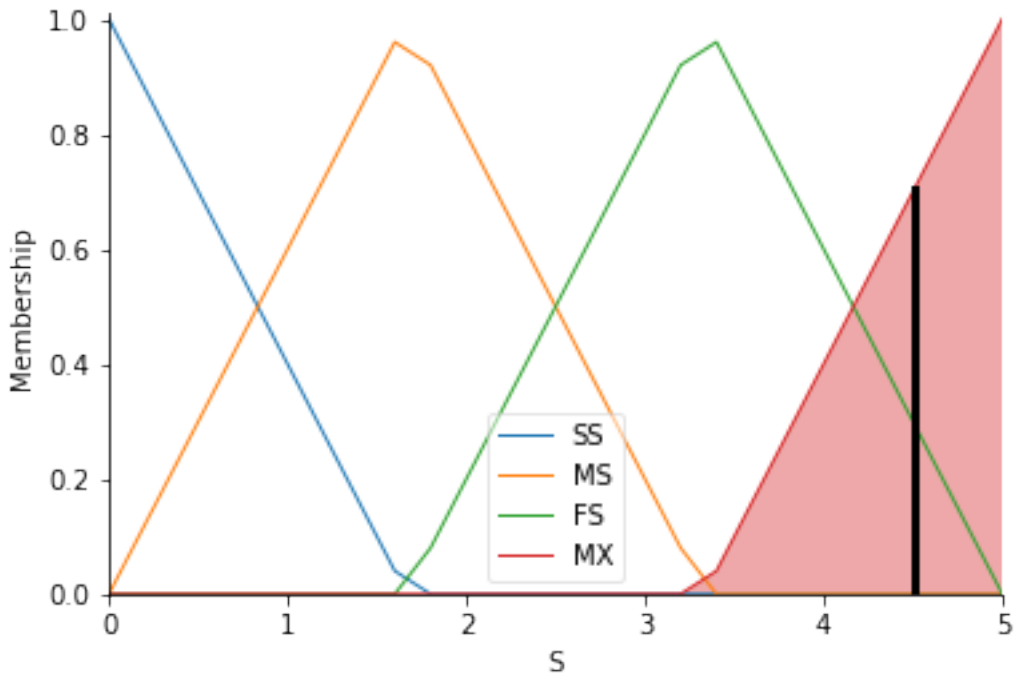


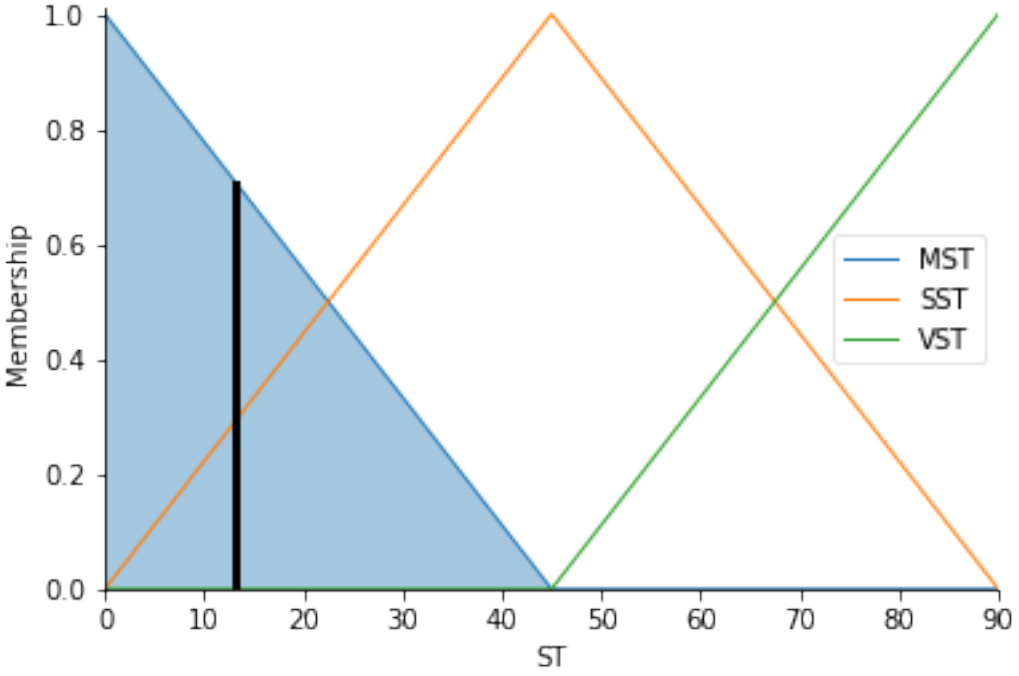
For input where object is not detected:

[18]:
```
# calling the functions
Inference_system_S(S,10,0)
Inference_system_turn(ST,10,0)
```

```
The computed S is: 4.509957506314853
The computed steering turn is: 13.180194846605374
```

[18]: 13.180194846605374


1.0
0.8
0.6
0.4
0.2
0.0
Membership
0
1
2
3
4
5
S
SS
MS
FS
MX


1.0
0.8
0.6
0.4
0.2
0.0
Membership
0
10
20
30
40
50
60
70
80
90
ST
MST
SST
VST

2.1.3 Defuzzification method: 'som'

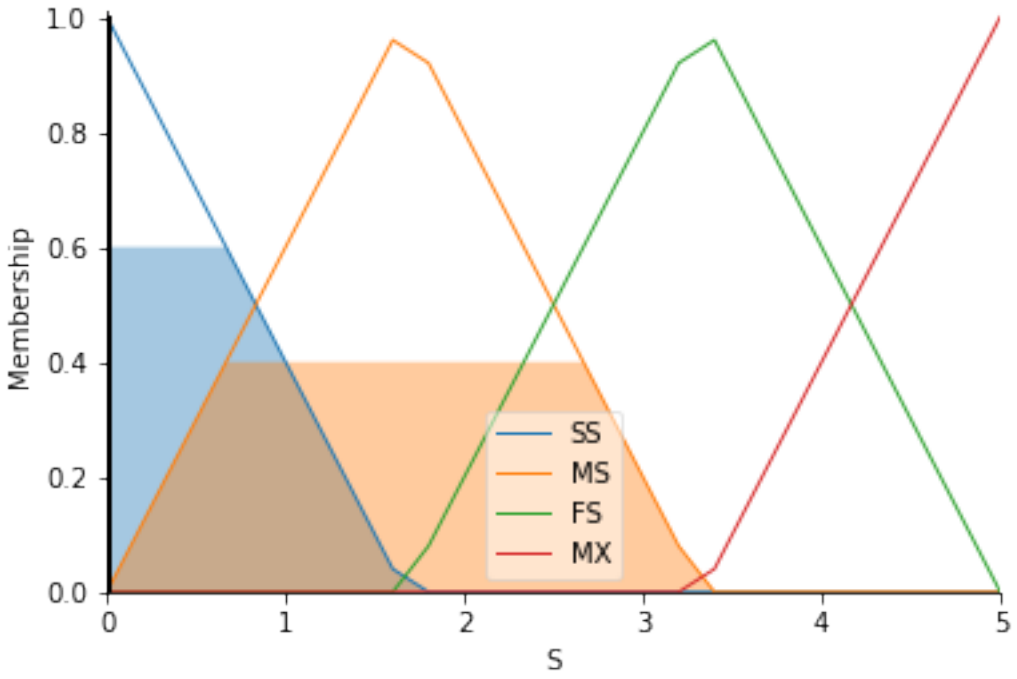
[19]:
```
# Change defuzzification method to som
S.defuzzify_method = 'som'
ST.defuzzify_method ='som'
```

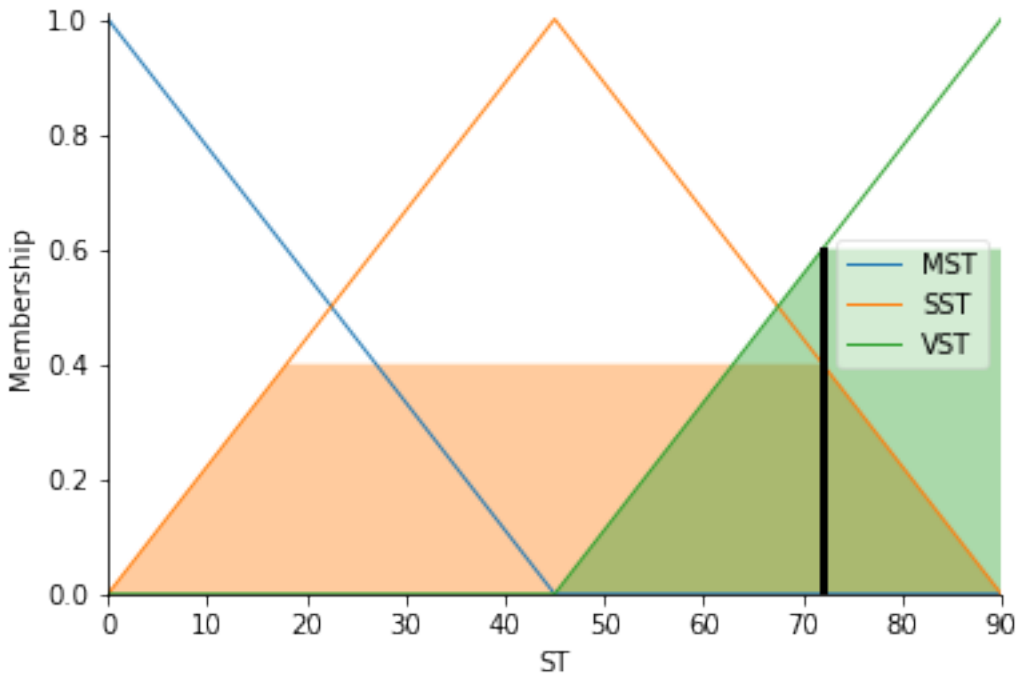
For input where object is detected:

[20]:
```
# Calling the functions
Inference_system_S(S,2,80)
Inference_system_turn(ST,2,80)
```

```
The computed S is: 0.0
The computed steering turn is: 72.0
```

[20]: 72.0



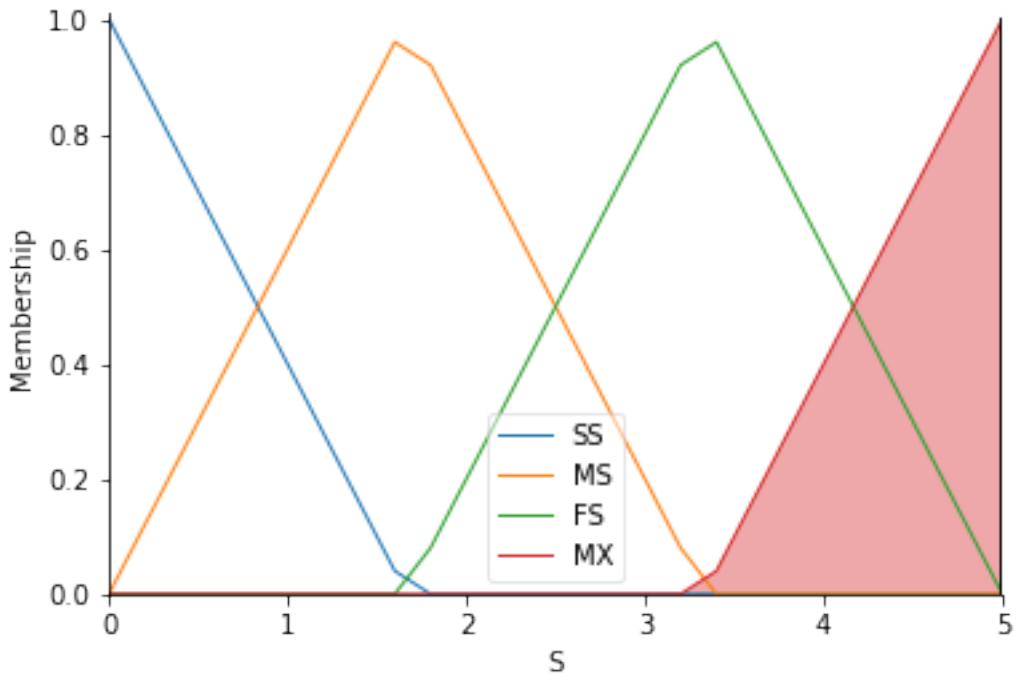


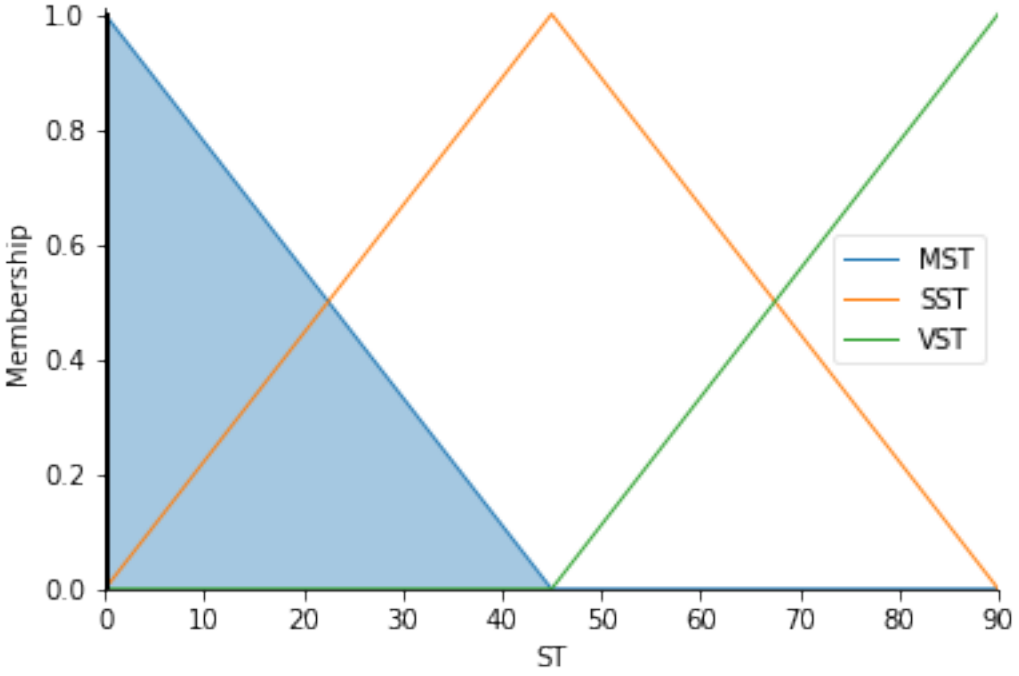
For input where object is not detected:

[21]:
```
# Calling the functions
Inference_system_S(S,10,0)
Inference_system_turn(ST,10,0)
```

```
The computed S is: 5.0
The computed steering turn is: 0.0
```

[21]: 0.0


1.0
0.8
0.6
0.4
0.2
0.0
Membership
SS
MS
FS
MX
0
1
2
3
4
5
S


1.0
0.8
0.6
0.4
0.2
0.0
Membership
MST
SST
VST
0
10
20
30
40
50
60
70
80
90
ST

2.1.4 Defuzzification method: 'lom'

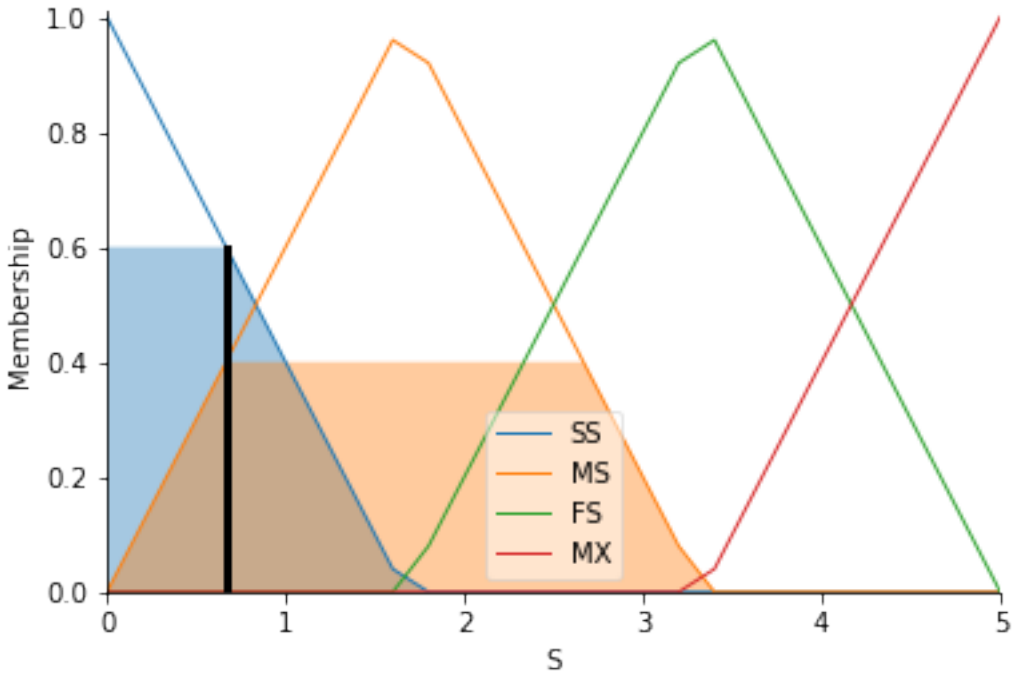
[22]:
```
# Change defuzzification method to lom
S.defuzzify_method = 'lom'
ST.defuzzify_method ='lom'
```

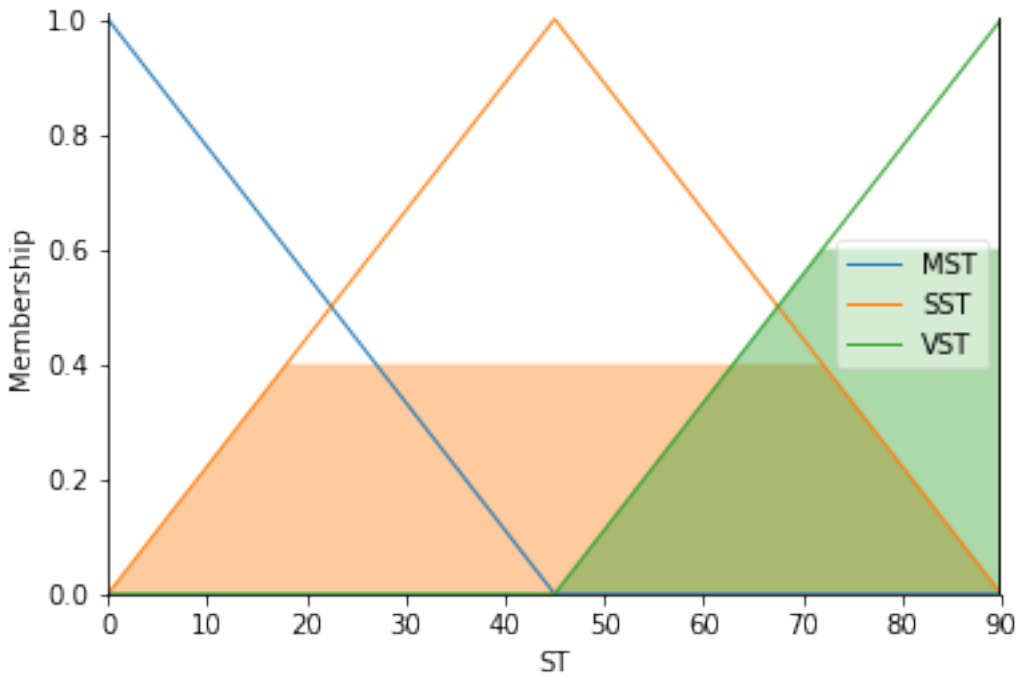
For input where object is detected:

[23]:
```
# calling the functions
Inference_system_S(S,2,80)
Inference_system_turn(ST,2,80)
```

```
The computed S is: 0.6666666666666667
The computed steering turn is: 90.0
```

[23]: 90.0



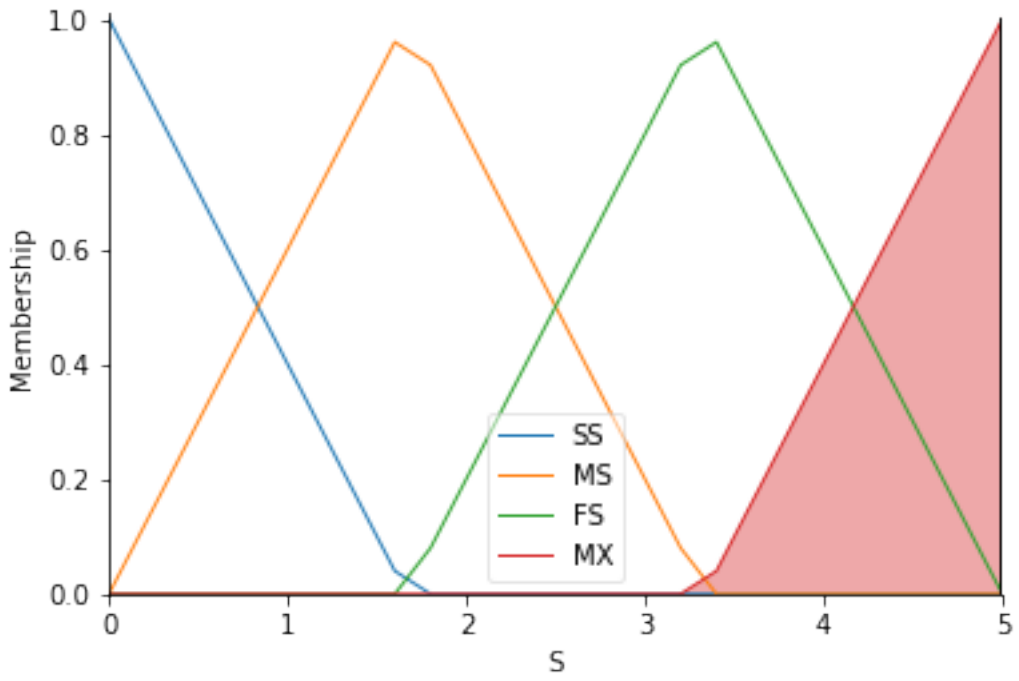


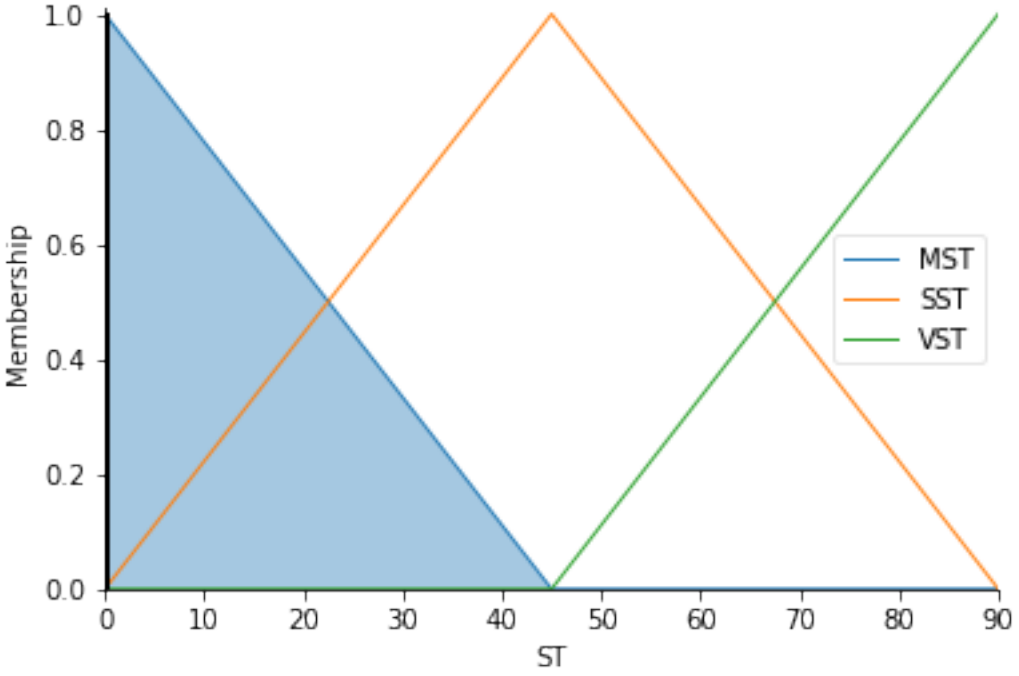
For input where object is not detected:

[24]:
```
# calling the functions
Inference_system_S(S,10,0)
Inference_system_turn(ST,10,0)
```

```
The computed S is: 5.0
The computed steering turn is: 0.0
```

[24]: 0.0


1.0
0.8
0.6
0.4
0.2
0.0
Membership
SS
MS
FS
MX
0
1
2
3
4
5
S


1.0
0.8
0.6
0.4
0.2
0.0
Membership
MST
SST
VST
0
10
20
30
40
50
60
70
80
90
ST

2.1.5 Defuzzification method: 'mom'

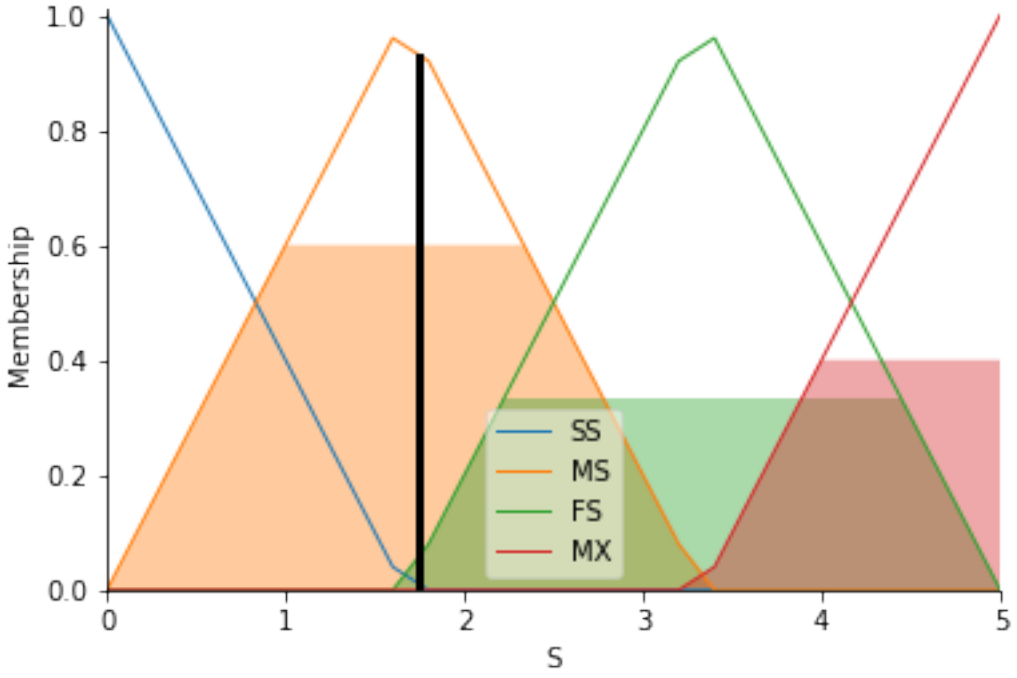
[25]:
```
# Change defuzzification method to mom
S.defuzzify_method = 'mom'
ST.defuzzify_method ='mom'
```

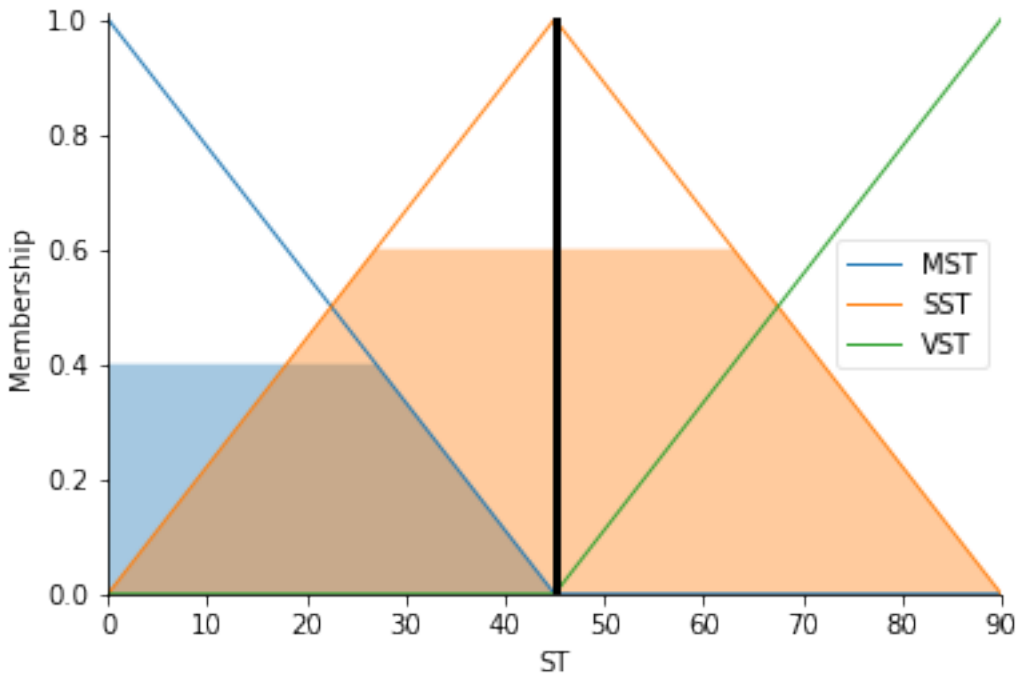
For 1) input where object is detected:

[26]:
```
# calling the functions
Inference_system_S(S,7,60)
Inference_system_turn(ST,7,60)
```

```
The computed S is: 1.750617283950617
The computed steering turn is: 45.0
```

[26]: 45.0



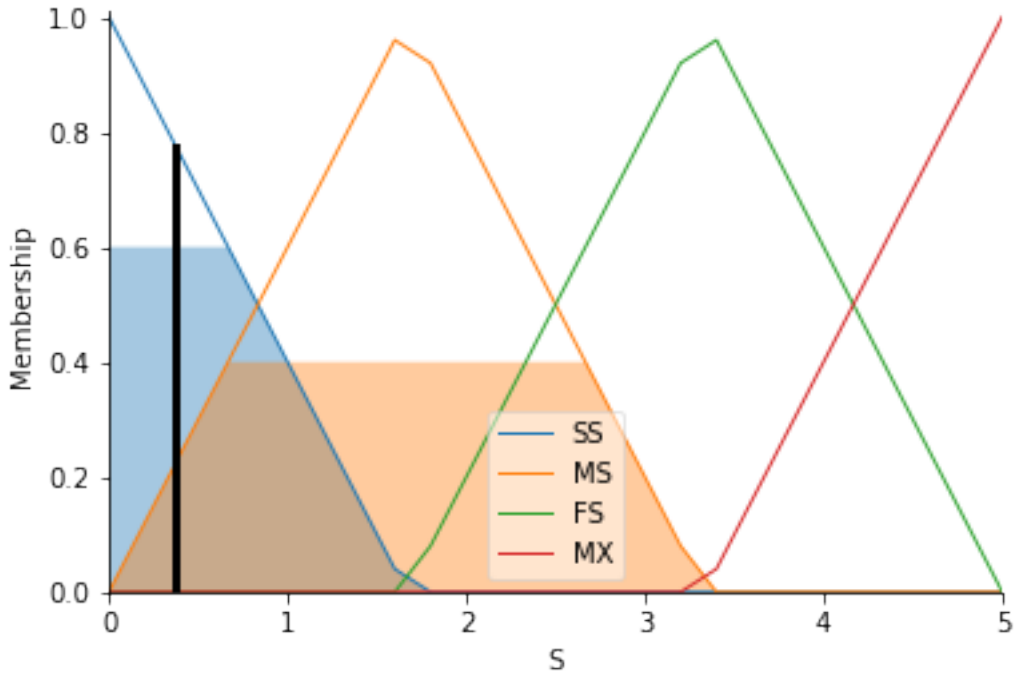


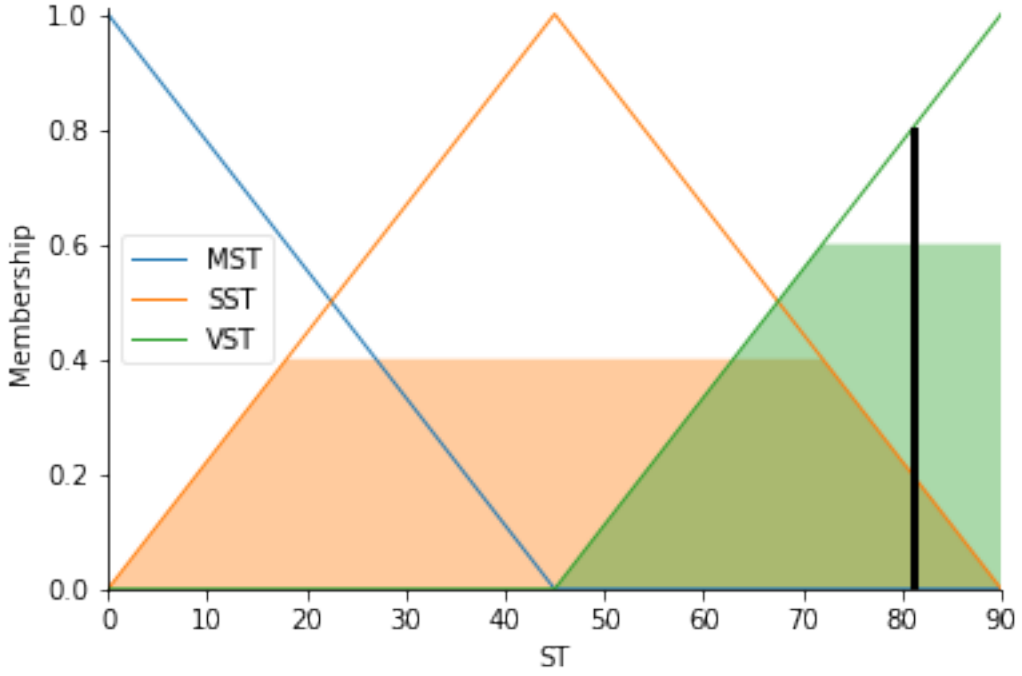
For 2) input where object is detected:

[27]:
```
# calling the functions
Inference_system_S(S,2,80)
Inference_system_turn(ST,2,80)
```

```
The computed S is: 0.3733333333333334
The computed steering turn is: 81.0
```

[27]: 81.0



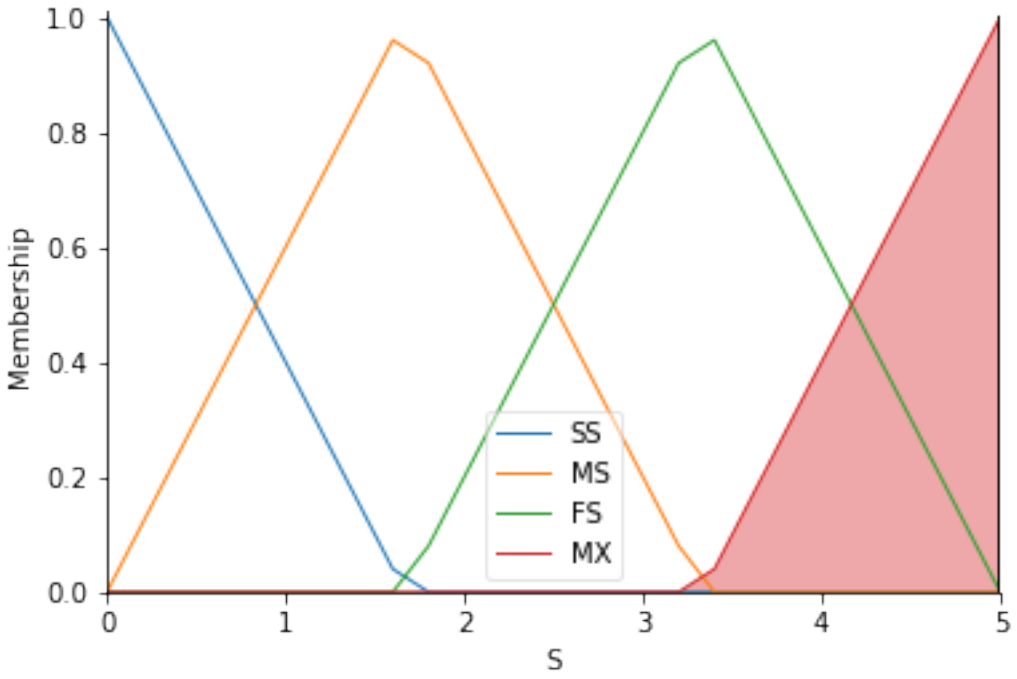


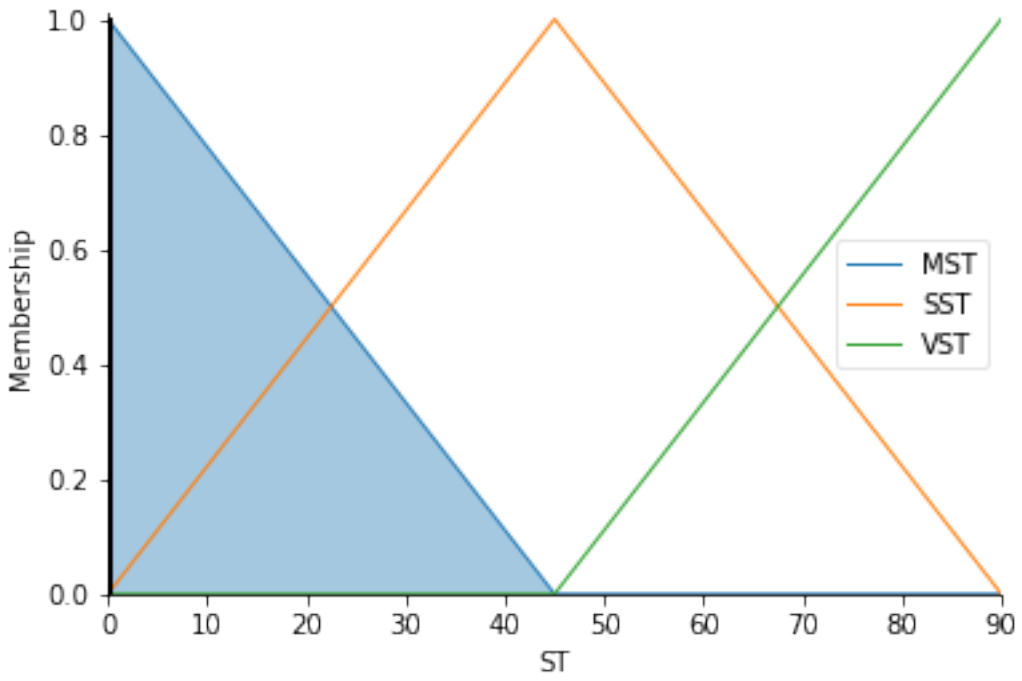
For input where object is not detected:

[28]:
```
# Calling the functions
Inference_system_S(S,10,0)
Inference_system_turn(ST,10,0)
```

```
The computed S is: 5.0
The computed steering turn is: 0.0
```

[28]: 0.0





2.1.6 Comparisions between Defuzzification methods:

- Here, the default method for defuzzification is 'centroid' and default Inferencing system is 'Mamdani'.
- Centroid : It resturns center of gravity of fuzzy set as the crisp output.
- Bisector : It returns the value of verticle line that divides the fuzzy set area into two equals.
- MOM , SOM, LOM : It returns the middle, smallest and largest of maximum value. If aggregate function has unique maximum, then they will return the same output.
- In our case, When object is detected , then bisector and Centroid both provided good results and so did MOM .
- But, when object is not detected , then bisector and centroid method still shows change in steering angle and even , speed is not cruised to maximum.
- Also, SOM and LOM are giving desired results when no obstacle is detected but, in other other case when obstacle is near, it reduces the speed upto 0 in SOM and in LOM it takes the maximum of very sharp turn, which is not good. so, these both methods will also be discarded.
- Thus, for defuzzification method, MOM will be used .

2.2 By hand implementation of Mamdani In ference system and Defuzzification strategy with two examples:

(11)

c. The inferencing System Used.

Here, Mamdani inferencing System will be used with Max-Min Method. As, we are using Conjuctive System of Rules. with 'AND' Operation. Thus, the minimum Value among the provided inputs in Rule will considered and used for output. Apart from this, we didn't use. Sugeno as, it returns output as. function of input. So, Our requirement of Control System is satisfied by. Mamdani inferencing System.

- ### d. The defuzzification Method.

There are many defuzzification methods such as 'Centroid', 'bisector', Maxima method. So, in our control system, Maxima MoM method has provided the Good and real world applicable results. So. we will use MoM method. Although 'Centroid' also provides fine result, but, in 'No obstacle' detected. one, it returns change in

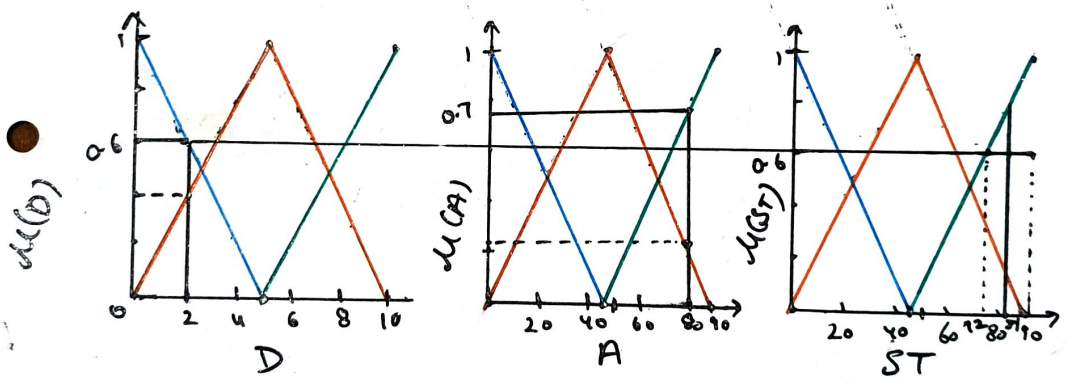
in both 'Speed' and 'Steering turn' output. whereas, MoM provided desired results. Also, MoM is easy to implement. Here, maximum membership function value is chosen and then, mean is calculated in that range.

Example ①

Suppose, we have. D = 2. and. A = 80. and Rule. is applied where

- if D = 'N' and. A = 'L', then. ST = 'VST'. and, in case of. Speed, Rule is, if D = 'N' and A = 'L', then. S = 'SS'. So, implementing them on graph.

Example:- for Steering turn Consequent.



Here, Among 0.7 and 0.6, 0.6 is the minimum and minimum is required in case of 'AND' operation. Therefore, now we will calculate. $\mu(ST) = 0.6$, ST value is required.

Also, here it intersects at line of 'VST'. So, Substituting $\mu(ST)$ in (18) eq. we get,

$$\mu_{VST}(ST) = \frac{ST-45}{45} \Rightarrow 0.6 = \frac{ST-45}{45}$$

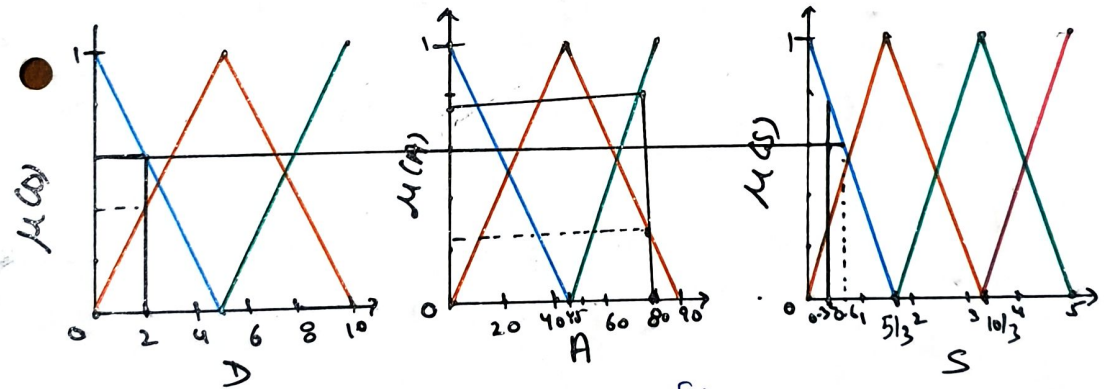
$$\Rightarrow \frac{3}{5} \times 45^{9} = ST-45 \Rightarrow ST = 72.$$

Hence, we found first intesecting point. value on x-axis. So, here a=72 and. it is clear from graph. that b = 90.

So, $MOM = \frac{a+b}{2} = \frac{72+90}{2} = \boxed{81}$

which is exactly as we calculated using liberaries.

for Speed Consequent.



Here, on applying only one given rule Among 0.7 and 0.6, 0.6 is minimum. So, its line will be extended. Thus now we have $\mu(S)=0.6$

and. aim is to find. S value on x-axis.
So, Substituting $\mu_{ss}(s) = 3/5$ in (9) eq. we get,

$$\mu_{ss}(s) = \frac{5-3s}{5} \Rightarrow \frac{3}{5} = \frac{5-3s}{5}$$

$$\Rightarrow s = 2/3 \Rightarrow s = 0.66.$$

Now, MoM for speed will be,

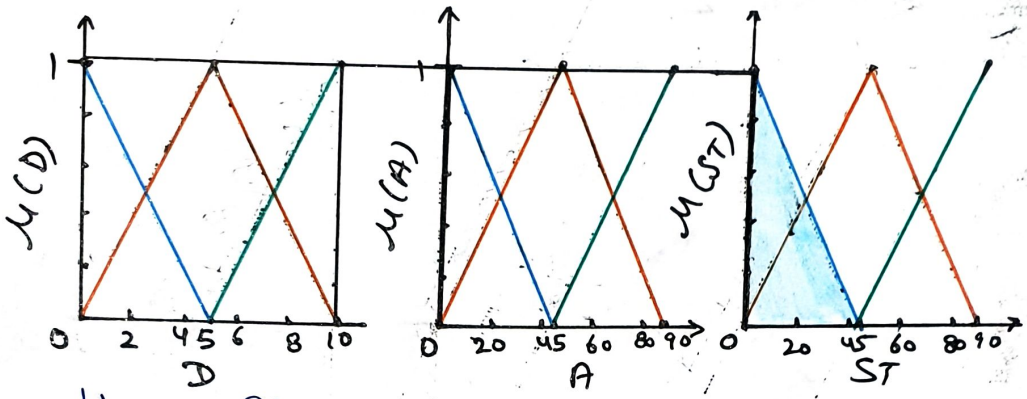
$$MoM = \frac{a+b}{2}$$

where $a = 0$ and $b = 0.66$.

Thus, $MoM = \boxed{0.33}$.

Example ②: 'No Object is detected' (D = 10 and A = 0).

For Steering Turn.

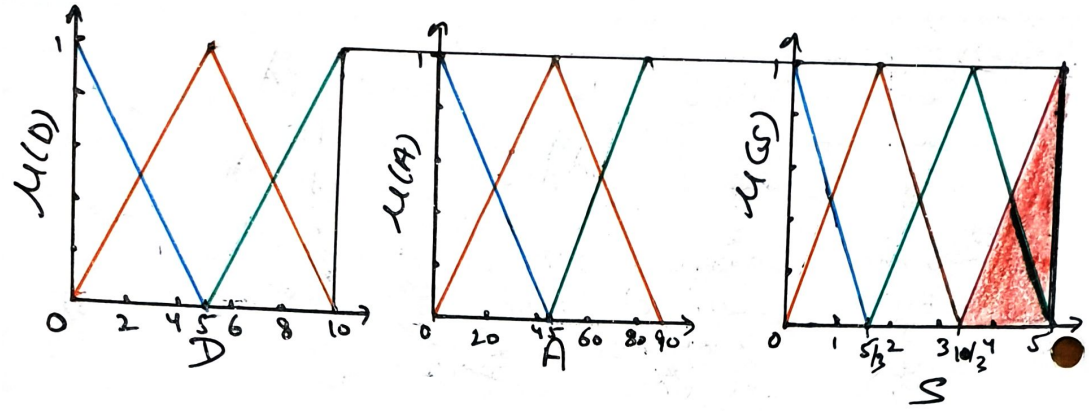


Here, MoM = $\frac{a+b}{2} = 0$

It means here. after applying rule. and when no obstacle is detected. then, ST=0 which is telling that there will be no change in Steering turn.

For Speed:

Similarly, for speed after applying the declared rules. we will see that in this condition, speed can be increased upto Maximum as no object will be detected.



(15)

Here, in Steering turn,

Rule : If D = 'Vf' and A = 'S'
, then ST = 'MST' is applied.

and, in Speed,

Rule : If D = 'Vf' and A = 'S'
, then S = 'Mx' is applied.

These rules are already mentioned. So,
here, MOM for Speed. = 5
which means if obstacle is not detected.
, Speed can be increased upto 'Maximum'.

3 Part 3)

3.0.1 Using 2 inputs with MOM Defuzzification method(Where obstacle is detected):

- First, the input is used as distance = 7 and angle = 60 , which provides the result of 1.75 in case of Speed and 45 in case of Steering turn.
- Then, another input is used where distance = 2 and angle = 80 , which provides the result of 0.37 in case of Speed and 81 in case of Steering turn.

- Thus, here we provided a normal input and the input where there are more chances of collision . The implementation also reveals how different defuzzification methods and choice of inference system reacted to the provided situation.

3.0.2 Using input with MOM Defuzzification method(Where obstacle is not detected):

Here, maximum distance is taken and minimum angle is given as input which is 10 and 0 respectively. On applying MOM, its returns no change that is 0 in Steering turn angle and speed as maximum that is 5. So, when no obstacle will be detected, then, steering turn angle will not change and speed will be increased.