Manik Daya

FCI

2017140008

AIM: To implement a Fuzzy controller.

THEORY: The term fuzzy refers to things which are not clear or are vague. In the real world many times we encounter a situation when we can't determine whether the state is true or false, their fuzzy logic provides a very valuable flexibility for reasoning. In this way, we can consider the inaccuracies and uncertainties of any situation. In the boolean system truth value, 1.0 represents absolute truth value and 0.0 represents absolute false value. But in the fuzzy system, there is no logic for absolute truth and absolute false value. But in fuzzy logic, there is intermediate value to present which is partially true and partially false.

What is Fuzzy Control?

1.It is a technique to embody human-like thinking into a control system.

2.It may not be designed to give accurate reasoning but it is designed to give acceptable reasoning.

3.It can emulate human deductive thinking, that is, the process people use to infer conclusions from what they know.

4.Any uncertainties can be easily dealt with the help of fuzzy logic.

Application:

1.It is used in the aerospace field for altitude control of spacecraft and satellites.

2.It is used in the automotive system for speed control, traffic control.

3.It is used for decision making support systems and personal evaluation in the large company business.

4.It has application in the chemical industry for controlling the pH, drying, chemical distillation process.

Code:

In [20]:
```
import numpy as np
import skfuzzy as fuzz
from skfuzzy import control as ctrl
import matplotlib.pyplot as plt
```

In [21]:
```
#setting scales
distance = ctrl.Antecedent(np.arange(0, 300, 1), 'distance')
speed = ctrl.Antecedent(np.arange(0, 15, 1), 'speed')
breakpower = ctrl.Consequent(np.arange(0, 75, 1), 'breakpower')
```

In [23]:
```
distance['SD']= fuzz.trimf(distance.universe,[0,100,200])
distance['MD']= fuzz.trimf(distance.universe,[100,200,300])
distance['LD']= fuzz.trimf(distance.universe,[200,300,400])
```

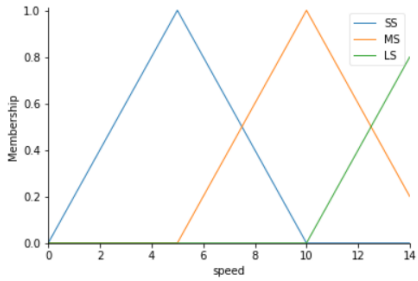
In [24]:
```
speed['SS']= fuzz.trimf(speed.universe,[0,5,10])
speed['MS']= fuzz.trimf(speed.universe,[5,10,15])
speed['LS']= fuzz.trimf(speed.universe,[10,15,20])
```

In [25]:
```
breakpower['SB']= fuzz.trimf(breakpower.universe,[0,25,50])
breakpower['MB']= fuzz.trimf(breakpower.universe,[25,50,75])
breakpower['LB']= fuzz.trimf(breakpower.universe,[50,75,100])
```

In [26]:
```
distance.view()
```

In [27]:
```
speed.view()
```

```
C:\Users\manik\anaconda3\lib\site-packages\skfuzzy\control\fuzzyvariable.py:122: UserWarning: Matplotlib is currently using mod
ule://ipykernel.pylab.backend_inline, which is a non-GUI backend, so cannot show the figure.
  fig.show()
```

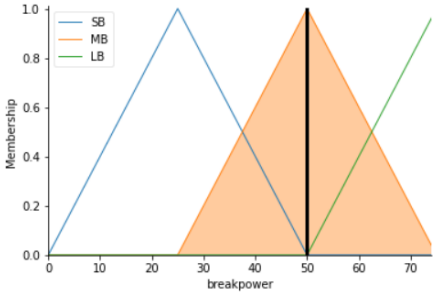


In [37]:
```
breakpower.view()
```

```
C:\Users\manik\anaconda3\lib\site-packages\skfuzzy\control\fuzzyvariable.py:122: UserWarning: Matplotlib is currently using mod
ule://ipykernel.pylab.backend_inline, which is a non-GUI backend, so cannot show the figure.
  fig.show()
```

```
In [29]: #setting rules
rule1 = ctrl.Rule(distance['SD'] & speed['SS'], breakpower['SB'])
rule2 = ctrl.Rule(distance['MD'] & speed["MS"], breakpower['MB'])
rule3 = ctrl.Rule(distance['LD'] & speed['LS'], breakpower['LB'])
rule4 = ctrl.Rule(distance['SD'] & speed['LS'], breakpower["LB"])
rule5= ctrl.Rule(distance['LD'] & speed['SS'], breakpower["SB"])
rule6= ctrl.Rule(distance['MD'] & speed['LS'], breakpower["LB"])
rule7= ctrl.Rule(distance['MD'] & speed['SS'], breakpower["SB"])
rule8 = ctrl.Rule(distance['LD'] & speed['MS'], breakpower["MB"])
rule9= ctrl.Rule(distance['SD'] & speed['MS'], breakpower["MB"])

In [30]: #controller
breaker_ctrl = ctrl.ControlSystem([rule1, rule2, rule3, rule4, rule5,rule6,rule7,rule8,rule9])
breakpowera = ctrl.ControlSystemSimulation(breaker_ctrl)

In [34]: breakpowera.input['distance'] = 100
breakpowera.input['speed'] = 10

In [35]: breakpowera.compute()

In [36]: print(breakpowera.output['breakpower'])
breakpower.view(sim=breakpowera)

49.98051774753136

C:\Users\manik\anaconda3\lib\site-packages\skfuzzy\control\fuzzyvariable.py:122: UserWarning: Matplotlib is currently using mod
ule://ipykernel.pylab.backend_inline, which is a non-GUI backend, so cannot show the figure.
  fig.show()
```


```
In [35]: breakpowera.compute()

In [36]: print(breakpowera.output['breakpower'])
breakpower.view(sim=breakpowera)

49.98051774753136

C:\Users\manik\anaconda3\lib\site-packages\skfuzzy\control\fuzzyvariable.py:122: UserWarning: Matplotlib is currently using mod
ule://ipykernel.pylab.backend_inline, which is a non-GUI backend, so cannot show the figure.
  fig.show()
```

Conclusion: Successfully implemented a system which calculates the break power needs to be applied using Fuzzy Logic.

There are three scales distance, speed and break power where speed and distance are Antecedent and break power as Consequent. Custom membership functions are made for all the scales and can be seen using view method, which will show the membership function graphically in triangular form. Then we define fuzzy relationship between input and output variables. For the purposes of our example, consider nine rules like, if the speed is super slow [SS] (< 5) and the distance is medium [MD] that is

between 100 to 200, then the break power applied is slow brake power [SB]. All other 8 rules are applied in same way but different configuration of speed and distance. Most people would agree on these rules, but the rules are fuzzy.

This is the kind of task at which fuzzy logic excels.

Now that we have our rules defined, we can simply create a control system. In order to simulate this control system, we will create a ControlSystemSimulation. We can now simulate our control system by simply specifying the inputs and calling the compute method. In this case the distance was 100 out of 300 and the speed is 10 out of max 15. Once computed, we can view the result as well as visualize it.
The results suggested break power of 49.98%. Therefore by changing speed and distance we will get different break power because fuzzy logic is applied and the outcome will be as per rules.