Michigan Technological University

CS5821

Fuzzy Control Project;

By

Nisarg Dave

Overview:

Introduction:

The Project is about designing the Fuzzy controller for movement of Robot. The controller considers the arena and related physical entities with some goals. The goals should be achieved by the robot in sequence.

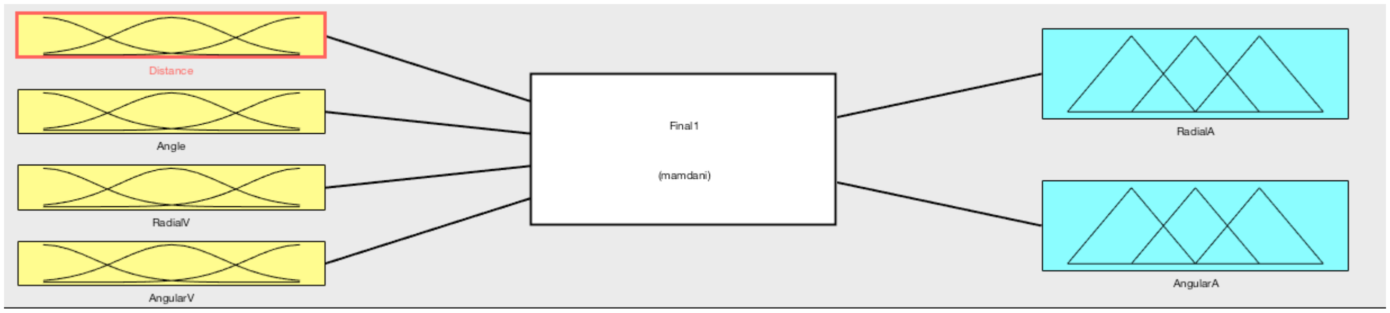
1. Mamdani Fuzzy Inference System:

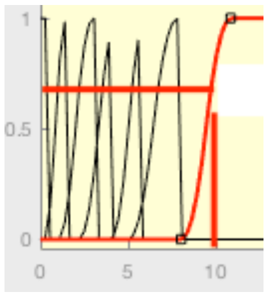
I've used Mamdani type fuzzy system for designing controller for robot's movement.

Basic Description of the system,
Controller Variables,
- Rotational and Forward acceleration

The Robot has defined parameters as per each state & those are,
- Coordinates (location)
- Angle
- Angular Velocity
- Radial Velocity

Overall Schematic for my FIS System is,

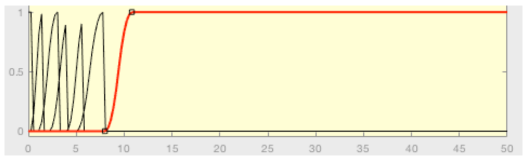




1a - (10,10,0,0.5) - Robot will move 10 degrees right 10 meters with forward drive velocity of 0.5

Membership value for this firing will be 0.65 (nearly) (see figure)

1b - (-30,50,-50,1) - Robot will move 30 degrees to left 50 meters with forward drive velocity of 1

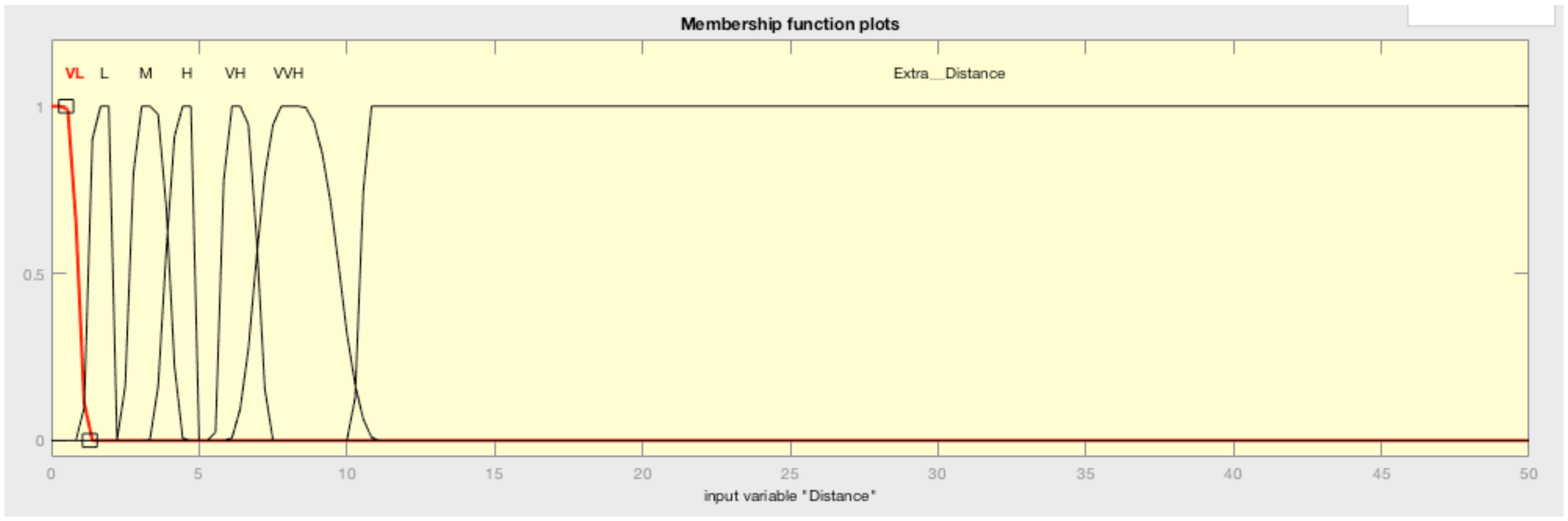


Membership value will be 1 (see fig)

2. Mapping of the Output space of the controller

The Input control variables are,

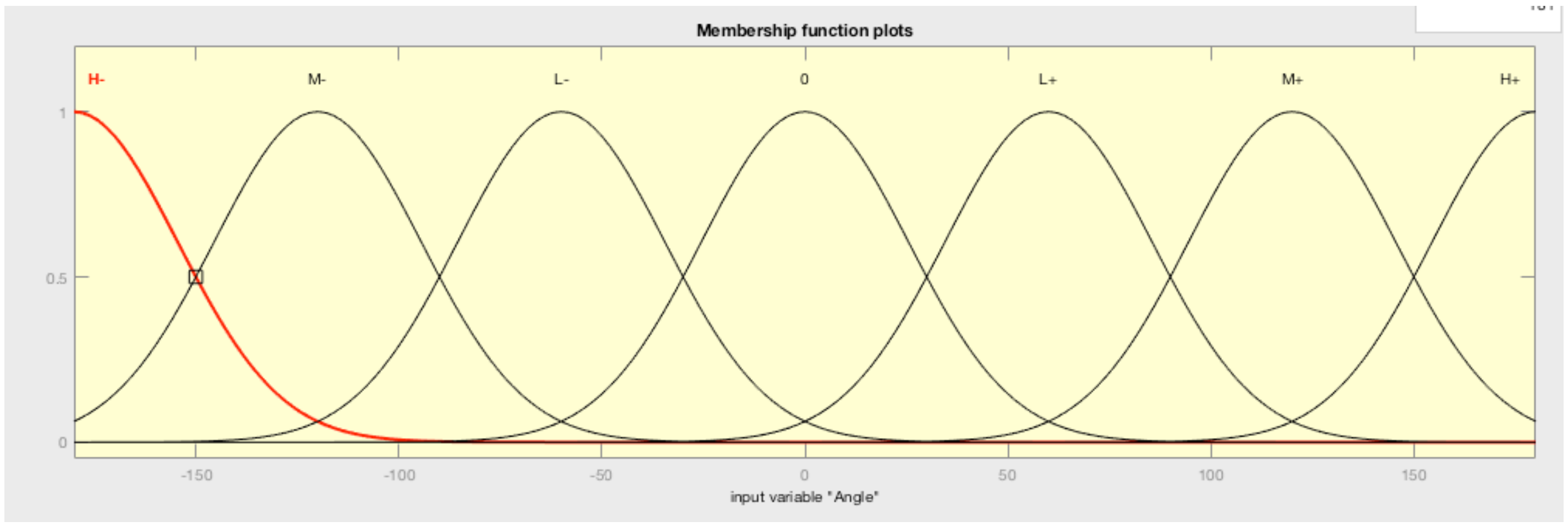
- Distance
- Angle
- RadialV (Radial Velocity)
- AngularV (Angular Velocity)



Distance: Distance is the variable for getting insight of detailed robot state and respective distance from the goal. The distance is the point in cartesian coordinate system, which indicates current and transition state of our Robot.
I took Pi function for 7 membership functions,

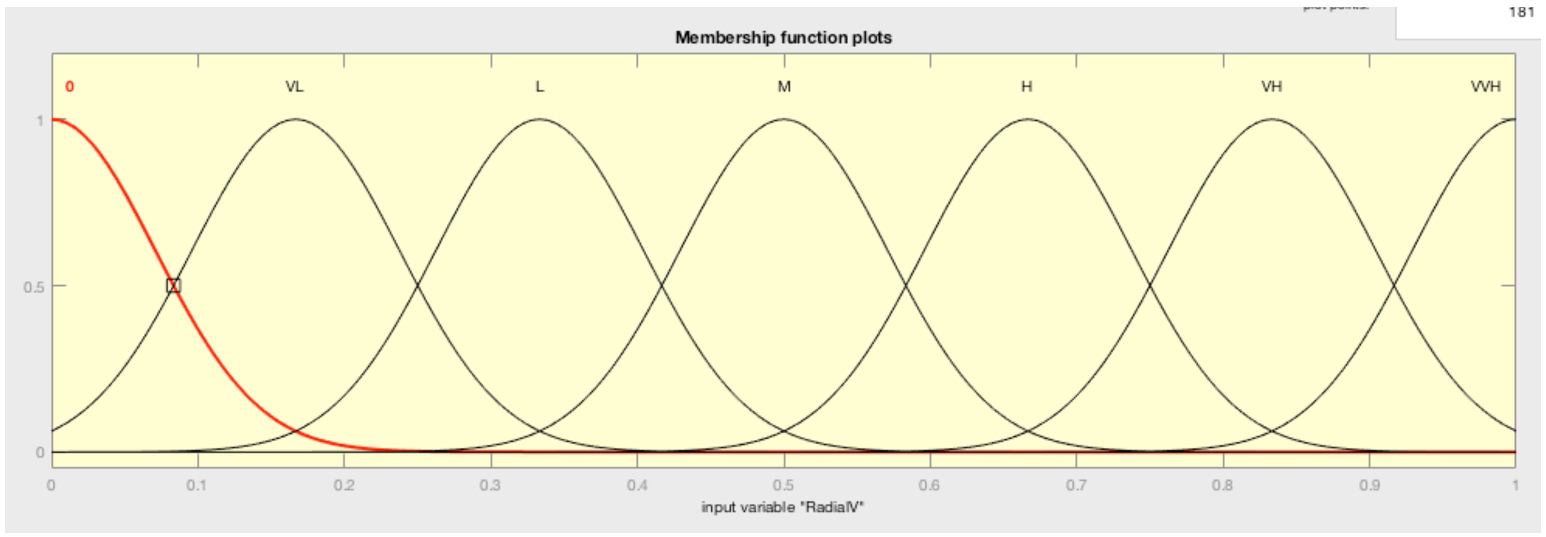
- VL (very low)
- L (Low)
- M (Medium)
- H (High)
- VH (Very high)
- VVH (Very very high)
- Extra__Distance (Very far distance in arena)

Special attention: The important thing in my 7 membership functions of Distance is that, I did scale down to 0-15 for better results. As per our points we have goals till 20 (Max) so it's good to scale down the system for better performance. So I took the most of Fuzzy membership within that range of 0-15.



Angle: Angle has it's value from -180 to 180. I took equally distributed 7 Gaussian membership functions for angle [-180,180]
The functions are,

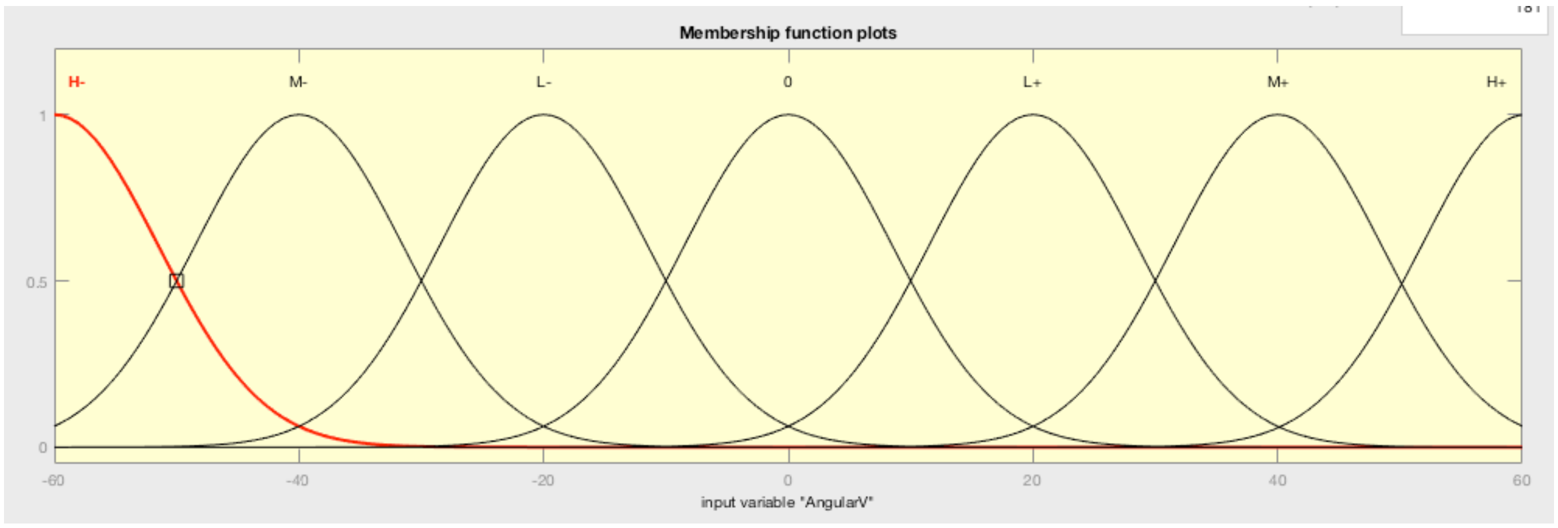
- H- (highly negative)
- M- (medium negative)
- L- (low negative)
- 0 (tends to zero)
- L+ (low negative)
- M+ (medium positive)
- H+ (high positive)



RadialV: Radial Velocity is from the range of [0,1] I took 7 equally spaced Gaussian membership functions.
The functions are,

- 0 (No velocity or tends to zero velocity)

- VL (Very low velocity)
- M (medium velocity)
- H (high velocity)
- VH (very high velocity)
- VVH (very very high velocity)



AngularV: The Radial acceleration is from [-60,60] I took again 7 equally spaced gaussian fuzzy membership functions,

The functions are,

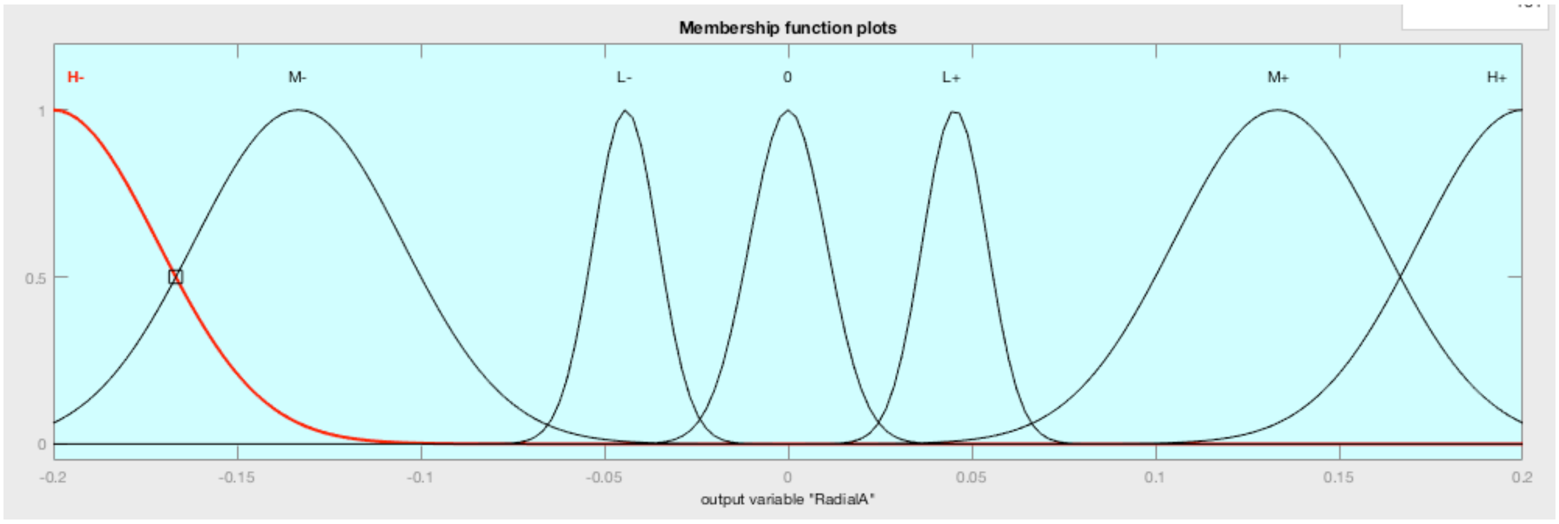
- H- (high negative)
- M- (medium negative)
- L- (low negative)
- 0 (No or tends to zero angular velocity)
- H+ (high positive)
- M+ (medium positive)
- L+ (low positive)

3. Mapping of the Output space of the controller

The output control variables are,

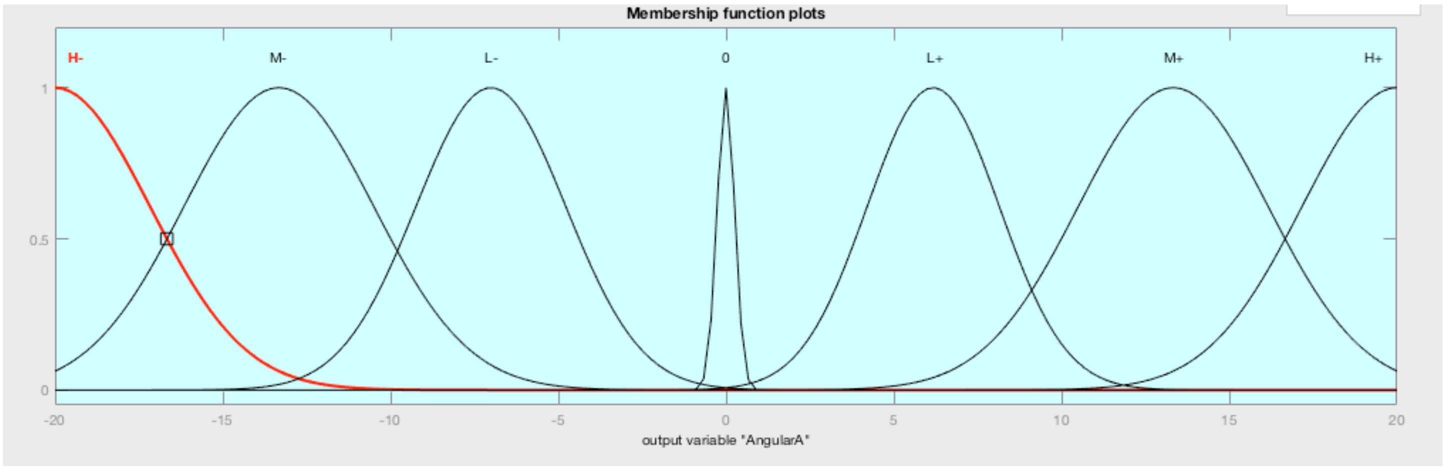
- RadialA (Radial Acceleration)
- AngularA (Angular Acceleration)

RadialA : Radial acceleration has the interval of [-0.2, 0.2]. I took 7 gaussian fuzzy membership functions,



Functions are,
- H- (high negative)
- M- (medium negative)
- L- (low negative)
- 0 (No or tends to zero Radial acceleration)
- H+ (high positive)
- M+ (medium positive)
- L+ (low positive)

I have tweaked the membership functions of L- & L+ for better performance, I did shrink those functions for tuning the overall working.



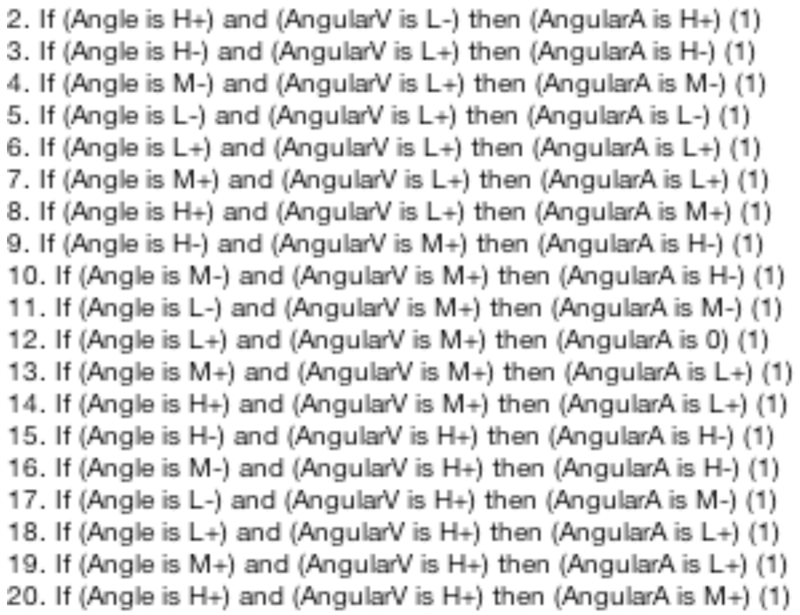
AngularA: Angular Acceleration has the value interval of [-20,20] with 7 gaussian fuzzy membership functions.

The functions are,

- H- (high negative)
- M- (medium negative)
- L- (low negative)
- 0 (No or tends to zero Angular acceleration)
- H+ (high positive)
- M+ (medium positive)
- L+ (low positive)

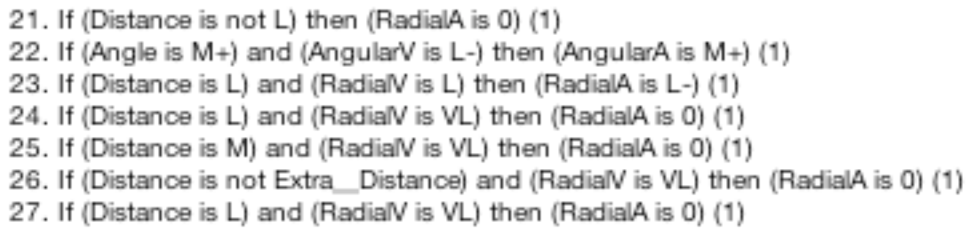
4. Fuzzy Rulebase:

The overall behavior of the system is very simple and made from several rule sets.



2. If (Angle is H+) and (AngularV is L-) then (AngularA is H+) (1)
3. If (Angle is H-) and (AngularV is L+) then (AngularA is H-) (1)
4. If (Angle is M-) and (AngularV is L+) then (AngularA is M-) (1)
5. If (Angle is L-) and (AngularV is L+) then (AngularA is L-) (1)
6. If (Angle is L+) and (AngularV is L+) then (AngularA is L+) (1)
7. If (Angle is M+) and (AngularV is L+) then (AngularA is L+) (1)
8. If (Angle is H+) and (AngularV is L+) then (AngularA is M+) (1)
9. If (Angle is H-) and (AngularV is M+) then (AngularA is H-) (1)
10. If (Angle is M-) and (AngularV is M+) then (AngularA is H-) (1)
11. If (Angle is L-) and (AngularV is M+) then (AngularA is M-) (1)
12. If (Angle is L+) and (AngularV is M+) then (AngularA is 0) (1)
13. If (Angle is M+) and (AngularV is M+) then (AngularA is L+) (1)
14. If (Angle is H+) and (AngularV is M+) then (AngularA is L+) (1)
15. If (Angle is H-) and (AngularV is H+) then (AngularA is H-) (1)
16. If (Angle is M-) and (AngularV is H+) then (AngularA is H-) (1)
17. If (Angle is L-) and (AngularV is H+) then (AngularA is M-) (1)
18. If (Angle is L+) and (AngularV is H+) then (AngularA is L+) (1)
19. If (Angle is M+) and (AngularV is H+) then (AngularA is L+) (1)
20. If (Angle is H+) and (AngularV is H+) then (AngularA is M+) (1)

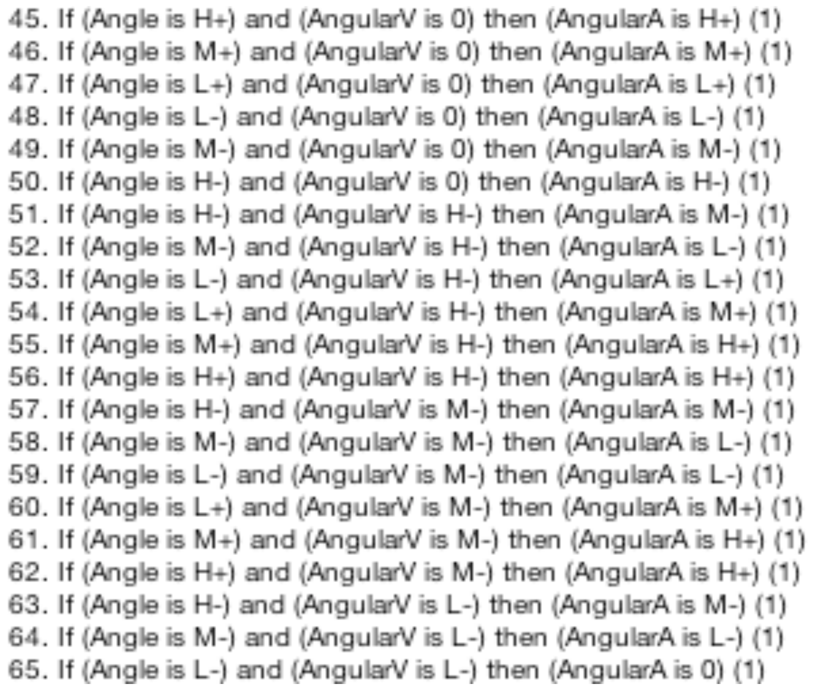
- here I'm taking angular velocity and angular acceleration into consideration and I'm trying to move robot in right angular direction by applying negative acceleration for positive velocity and vice versa.



21. If (Distance is not L) then (RadialA is 0) (1)
22. If (Angle is M+) and (AngularV is L-) then (AngularA is M+) (1)
23. If (Distance is L) and (RadialV is L) then (RadialA is L-) (1)
24. If (Distance is L) and (RadialV is VL) then (RadialA is 0) (1)
25. If (Distance is M) and (RadialV is VL) then (RadialA is 0) (1)
26. If (Distance is not Extra__Distance) and (RadialV is VL) then (RadialA is 0) (1)
27. If (Distance is L) and (RadialV is VL) then (RadialA is 0) (1)

- Here I'm trying to tweak angular & radial acceleration using distance and radial velocity.

Example: If distance is low or medium & radial velocity is low or very low then I'm tuning radial acceleration to 0 or negative low for stabilizing robot movement.


45. If (Angle is H+) and (AngularV is 0) then (AngularA is H+) (1)
46. If (Angle is M+) and (AngularV is 0) then (AngularA is M+) (1)
47. If (Angle is L+) and (AngularV is 0) then (AngularA is L+) (1)
48. If (Angle is L-) and (AngularV is 0) then (AngularA is L-) (1)
49. If (Angle is M-) and (AngularV is 0) then (AngularA is M-) (1)
50. If (Angle is H-) and (AngularV is 0) then (AngularA is H-) (1)
51. If (Angle is H-) and (AngularV is H-) then (AngularA is M-) (1)
52. If (Angle is M-) and (AngularV is H-) then (AngularA is L-) (1)
53. If (Angle is L-) and (AngularV is H-) then (AngularA is L+) (1)
54. If (Angle is L+) and (AngularV is H-) then (AngularA is M+) (1)
55. If (Angle is M+) and (AngularV is H-) then (AngularA is H+) (1)
56. If (Angle is H+) and (AngularV is H-) then (AngularA is H+) (1)
57. If (Angle is H-) and (AngularV is M-) then (AngularA is M-) (1)
58. If (Angle is M-) and (AngularV is M-) then (AngularA is L-) (1)
59. If (Angle is L-) and (AngularV is M-) then (AngularA is L-) (1)
60. If (Angle is L+) and (AngularV is M-) then (AngularA is M+) (1)
61. If (Angle is M+) and (AngularV is M-) then (AngularA is H+) (1)
62. If (Angle is H+) and (AngularV is M-) then (AngularA is H+) (1)
63. If (Angle is H-) and (AngularV is L-) then (AngularA is M-) (1)
64. If (Angle is M-) and (AngularV is L-) then (AngularA is L-) (1)
65. If (Angle is L-) and (AngularV is L-) then (AngularA is 0) (1)

-here I'm tweaking angular acceleration based on value of angle and angular velocity.
- If angle is positive and velocity is 0 then angular acceleration should be positive.
-If angle is negative and velocity is negative then saturate that negative with negative angular acceleration and so on!

Overall behavioral rational is to set the angle according to angular velocity and acceleration. After setting the desired angle move forward with radial velocity into that direction. Slowing down the robot near the goal and stabilize the rotational velocity according to desired angular acceleration and further accelerate towards the next goal with newly tuned velocity, distance and angle.

5. Defuzzifier:

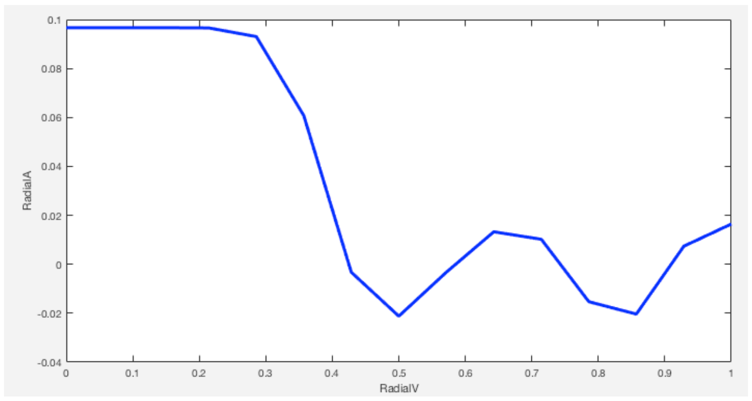
I've used the mamdani fuzzifier and MATLAB Fuzzy logic designer takes care for the defuzzification. I've set the defuzzifier type to the centroid based difuzzification.

Most prevalent and physically appealing method for this application,

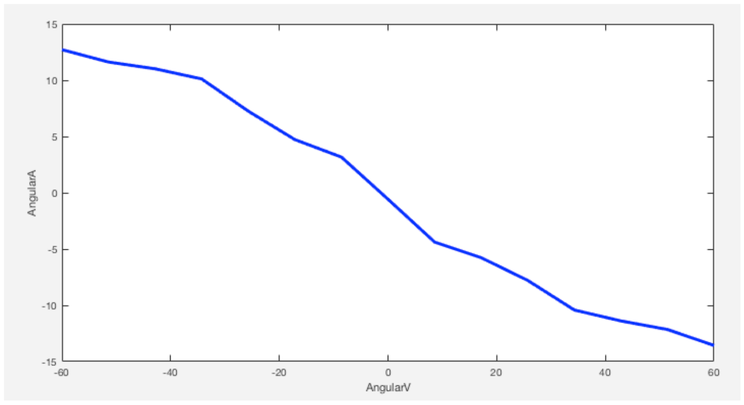
Ex: DeFuzzifier will take input of (10,10,0.1,5) then it'll give output that "Move Robot at 10 Point to the right with 5 unit of velocity in the direction of 10 degree relative angle"

Why that method?

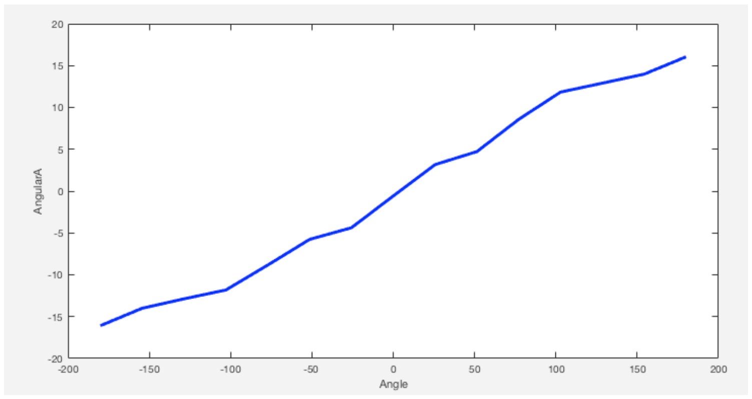
1. Continuity: I judged that Small change in the value of distance makes big change in overall behavior of system. So continuity should be maintained.
2. Disambiguity: The method is resulting into unique values so ambiguity is avoided with this method
3. Plausibility: Centroid is the middle of X axis and in the support region it has high degree of membership which is clearly desired here.
4. It's not that computationally intensive for this application.



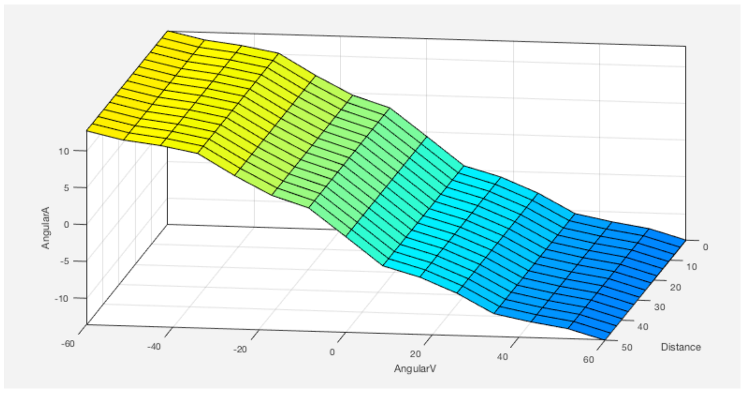
RadialA
RadialV



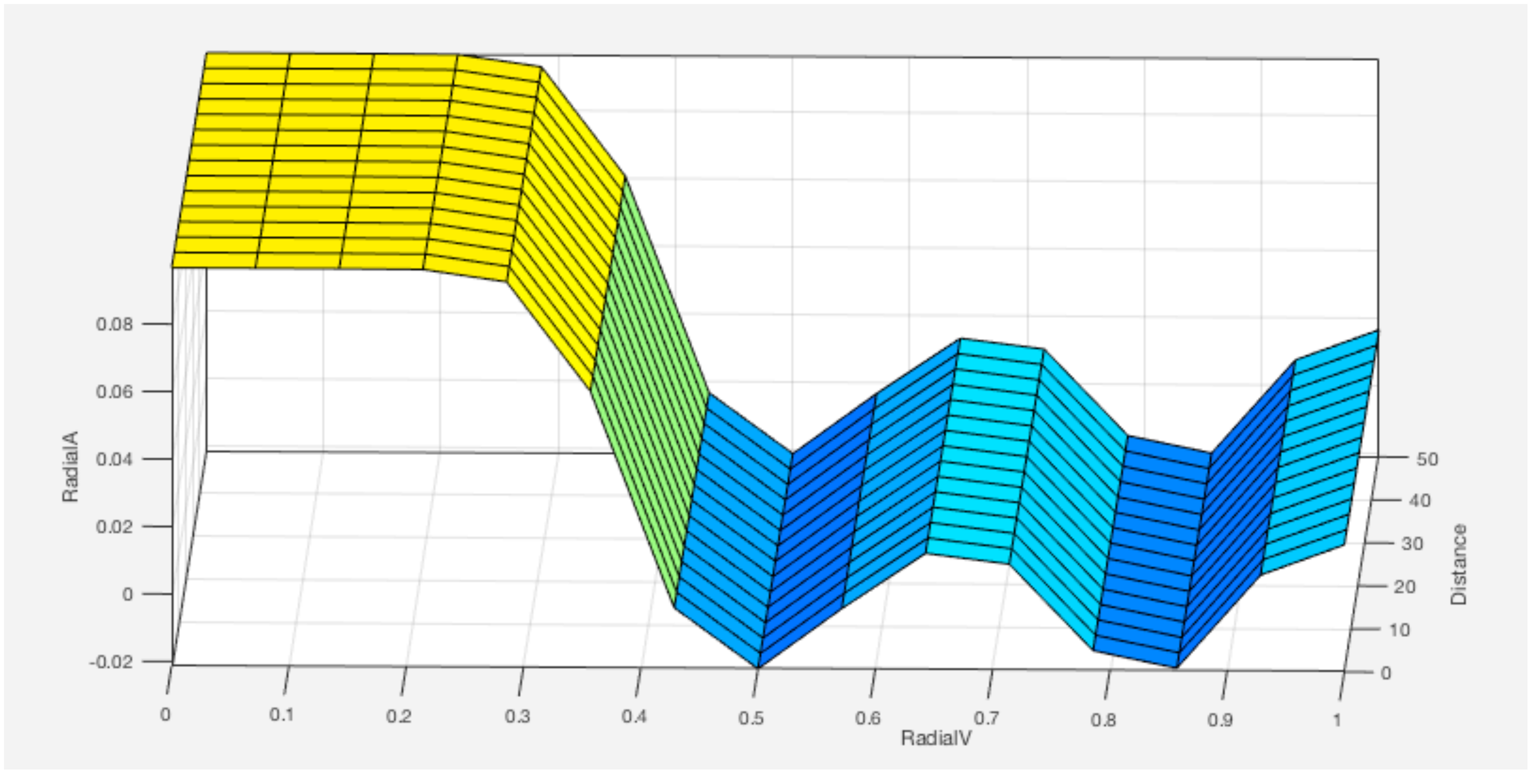
AngularA
AngularV



AngularA
Angle



AngularA
AngularV
Distance



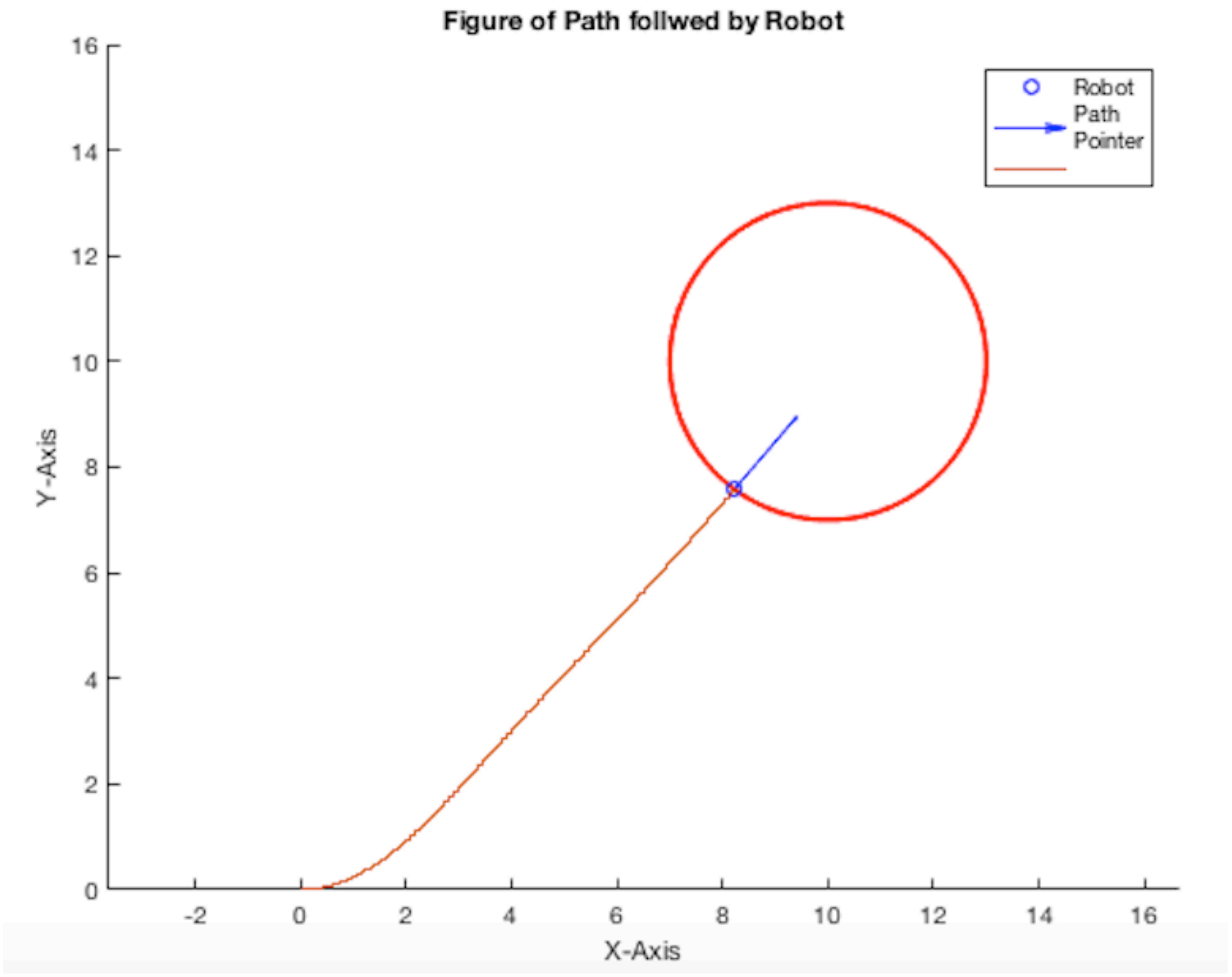
RadialA
RadialV
Distance

6. Visualization:

Time Step necessary to reach to each Goal

Path	Scenario	#Time steps
1	One goal at (10,10) m Radius: 3 m	311
2	One goal at (10,10) m Radius: 0.3 m	357
3	Two goals:1st goal at (20,20) m Radius: 1 m, 2nd goal at (5,20) m radius:1 m	1059
4	Three goals: (0,10) radius:4; (0,0) radius 1, (20,0) radius 5	926
5	Five goals: (5,5) radius 0.5; (4,4) radius 0.5; (6,6) radius 0.5; (3,6) radius 0.5; (6,3) radius 0.5	843

Path1:



Path2:

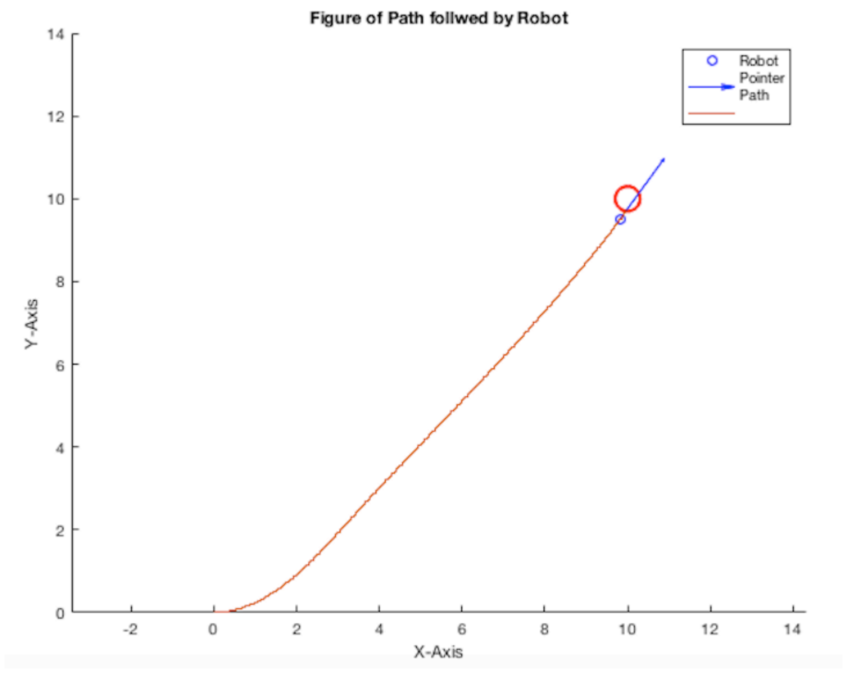


Figure of Path follwed by Robot
Robot
Pointer
Path
Y-Axis
X-Axis
14
12
10
8
6
4
2
0
-2
0
2
4
6
8
10
12
14

Path3:

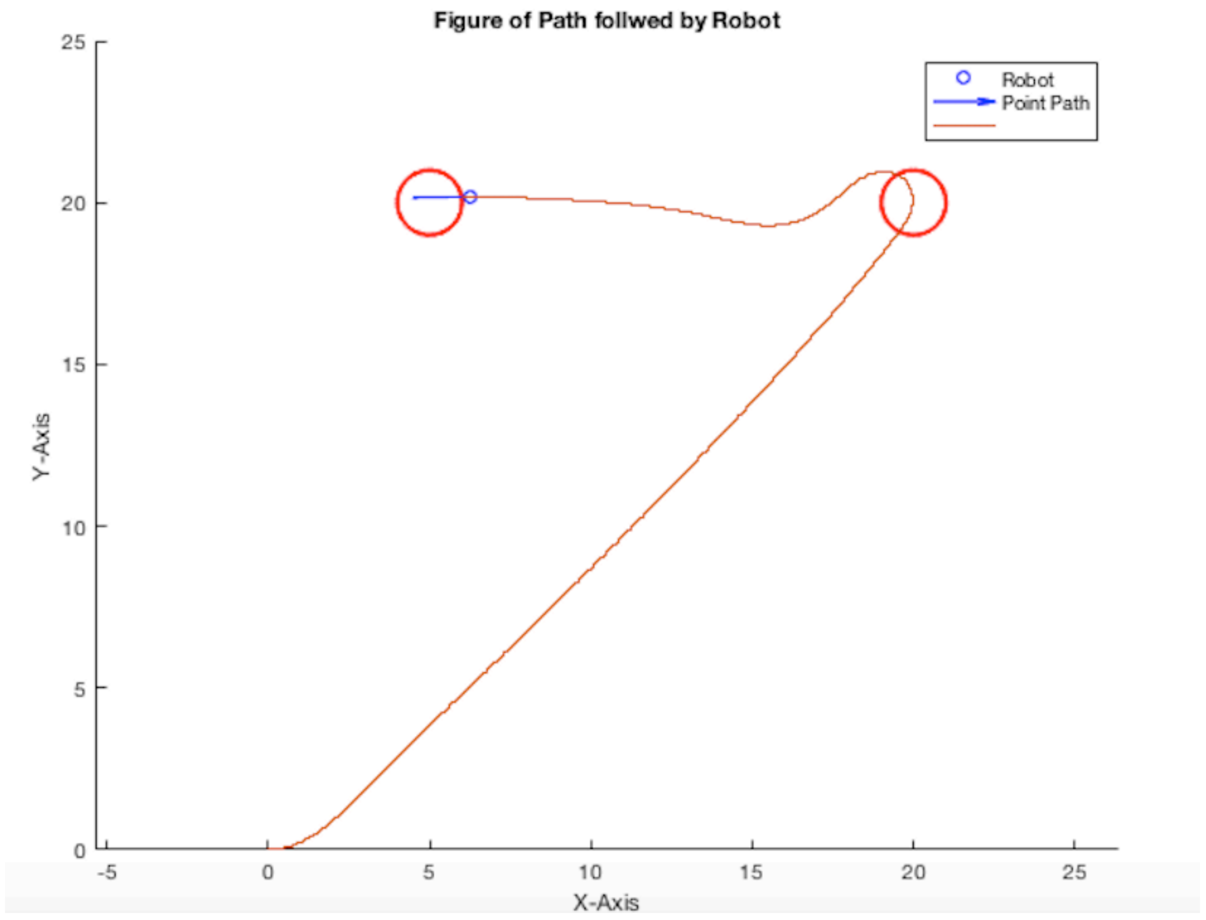


Figure of Path follwed by Robot
Robot
Point Path
Y-Axis
X-Axis
25
20
15
10
5
0
-5
0
5
10
15
20
25

Path4:

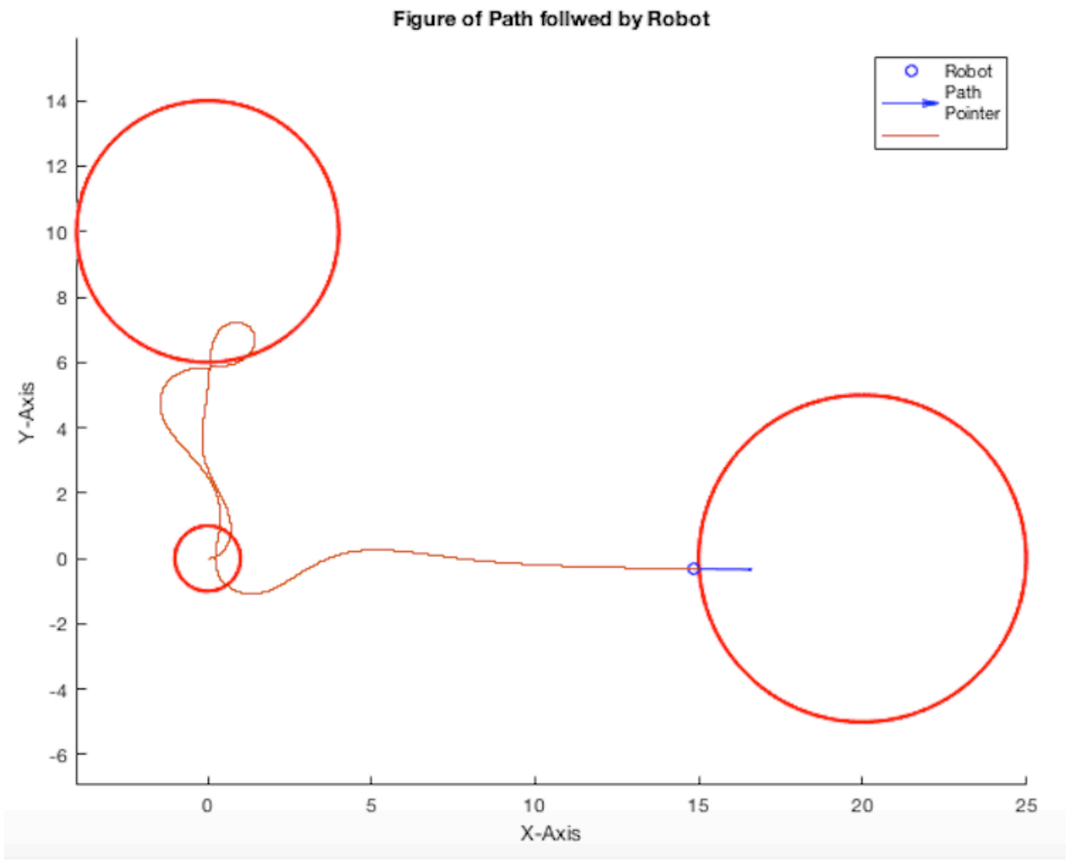


Figure of Path follwed by Robot
Robot
Path
Pointer
Y-Axis
X-Axis
14
12
10
8
6
4
2
0
-2
-4
-6
0
5
10
15
20
25

Path5:

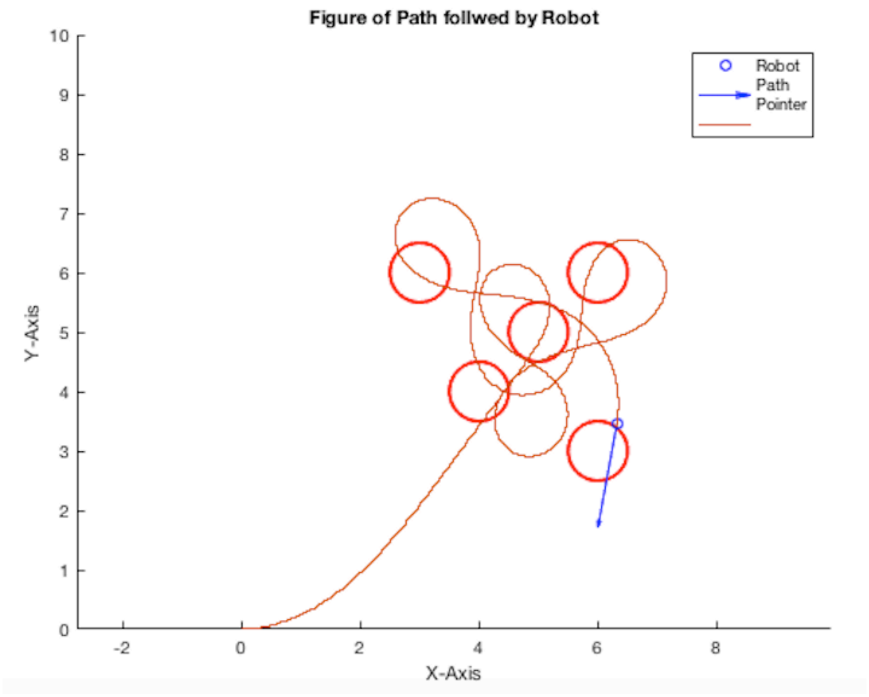


Figure of Path follwed by Robot
Robot
Path
Pointer
Y-Axis
X-Axis
10
9
8
7
6
5
4
3
2
1
0
-2
0
2
4
6
8

7. Changes/tweaks to the system and Final Results:

After getting the results, I looked over once again to my rule base. I made some tweaks into rules,

- First tuning correction was in setting radial acceleration, I was setting the acceleration to one variable before. I made tweak for constantly changing the radial acceleration in between goals so the steps will be reduced.
- After starting from one goal the acceleration will be medium and then certainly it will increase till the distance of nearby. After getting nearby it will be negative so velocity will get decreased and robot stops at the point.
- I even changed the distance input range for each fuzzy class membership function so each point can be classified as small influx region. I did shrink the each and every membership function. The smoothing of functions resulted in tremendous step decrease for the five goals target.
- I even did smooth velocity transition and angular acceleration transition for stability of the robot so now robot is more stable.

Final Results are,

Path	Scenario	#Time steps
1	**One goal at (10,10) m Radius: 3 m**	295
2	**One goal at (10,10) m Radius: 0.3 m**	357
3	**Two goals:1st goal at (20,20) m Radius: 1 m, 2nd goal at (5,20) m radius:1 m**	1055
4	**Three goals: (0,10) radius:4; (0,0) radius 1, (20,0) radius 5**	838
5	**Five goals: (5,5) radius 0.5; (4,4) radius 0.5; (6,6) radius 0.5; (3,6) radius 0.5; (6,3) radius 0.5**	790

I got good performance improvement after this corrections.

Now I performed one experiment by reducing number of membership functions for inputs and outputs.
Apparently I did experiment with 3,4,5,6 membership functions for each inputs and outputs. The results are worst for 3 & 4 number of membership functions for each inputs and outputs.

As membership function decreases the performance index and overall working efficiency of system decreases. Some input variables like, distance and angle requires higher number of membership functions such as 6-7. Other inputs such as angular and radial velocity are fine with 4-5 membership functions because their number of fuzziness isn't effecting the output. They aren't as sensitive as distance and angle. Slight change in membership fn of distance can impact very adversely to output. In one case the robot even stopped running! So I don’t think so using 3 membership fn for each input and output is good idea. Yeah, at least for some input/output variables we can use that but not for all of them!

Conclusive Summary:

The overall experience was really very nice. I learned a lot of things. Specially debugging for Fuzzy systems! In first attempt I failed, My robot was not even moving. The next I tried controlling the velocity with respect to distance and angle. Again it wasn’t moving. It started rotating around (0,0) then I figured out that I should control angular velocity with conjunction of others and after doing that It moved! But again that was really very worst, it moved by like in spiral mode, so I followed the another approach.

I check the surface and rule firing by varying inputs and analyzing the output accordingly. After doing that I did some debugging while simulation, I found out reason for one constant misalignment of velocity and acceleration and I fixed those rules. Either tweaked or added some new and here I’m with descent system. It’s really very good experience, sometimes frustrating but still interesting because that black box was attracting me and challenging me to solve that.