FUZZY CONTROL PROJECT

Computational Intelligence

ABSTRACT

A controller based on fuzzy logic is presented in this work. The controller is used to operate a robot so that it can steer itself to any goals placed in an arena. This report focusses on the fuzzy logic rules and fuzzy sets used to build the controller. This work is submitted as a project for Computational Intelligence course of Dr. Timothy Havens at Michigan Technological university.

SHASHANK PATHRUDKAR

Michigan Tech

Contents

Problem statement 1

Q1 Design of Fuzzifier 2

Q2 Input fuzzy sets 6

Q3 Output fuzzy sets 8

Q4 Fuzzy rule system 9

Q5 Defuzzifier 11

Q6 Goals for the designed robot 12

Q7 **a.** Discussion regarding further improvements in the robot 14

b. Robot with 3 fuzzy sets 15

c. Learnings 17

Acknowledgements 18

References 19

Appendix 20

Fuzzy Control Project

Problem Statement

You are a robot in an arena (assume that the arena has no boundaries). The two control (action) parameters you have. are your rotation acceleration $\ddot{\theta}$ and your forward-drive acceleration $\ddot{r}$ (see the drawing below). The inputs to your system are the current rotational and forward-drive velocities ($\theta, \dot{r}$), the Boolean inputs of whether your rotational and radial accelerations are positive or negative, and the relative angle (ϕ) and distance (d) towards the center of a goal, which is parameterized by two coordinates ($x.$, $y.$) and a radius denoting the target area. There will be multiple goals that your robot must enter; however, you cannot achieve the next goal until you pass through the current goal (i.e., the goals must be met sequentially). The time-step of your controller is 100 milliseconds (i.e., you can compute inputs and outputs every 100 msec).

Specifications of the robot:
Range of radial velocity – [0, 1] m/sec (robot only goes forward)
Range of radial acceleration – 0.2 m/sec^2
Range of rotation velocity – 60 deg/sec
Range of rotation acceleration – 20 deg/sec^2
Spatial size – none (assume it is a point)

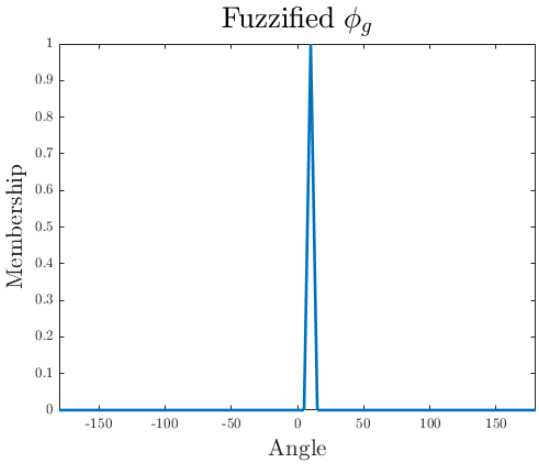
Specifications of arena:
Origin is at (0,0) (bottom left) and 0 degrees is positive *x*-axis

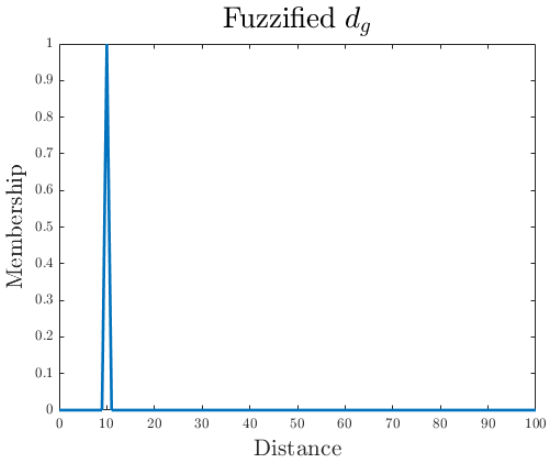
Robot always starts at (0,0) with velocity = (0 deg/sec, 0 m/sec) and angle = 0 degrees, acceleration state = (0 deg/sec2, 0 m/sec2). So, it is pointed to the right and at a stand-still. Goals are considered met when the robot is within the radius of the goal at the end of a time-step. Goals are presented sequentially (only one goal is known to the robot at a time).

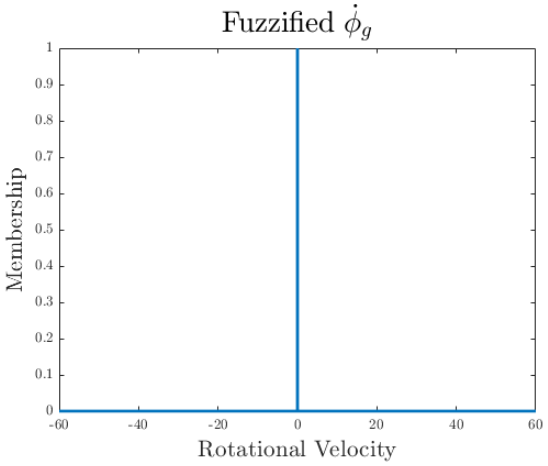
Question 1

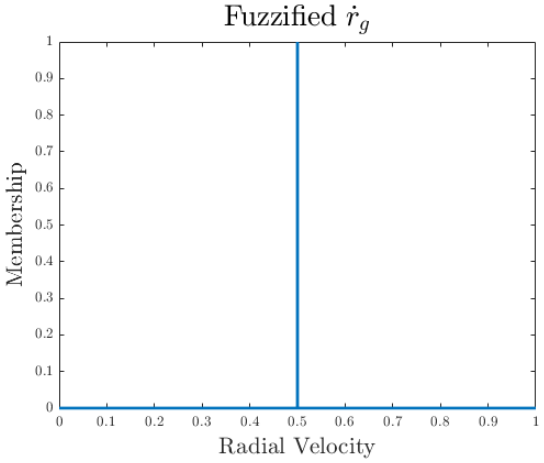
Fuzzified input for

1a) $\phi, d, \dot{\phi}, \dot{r} = (10, 10, 0, 0.5)$

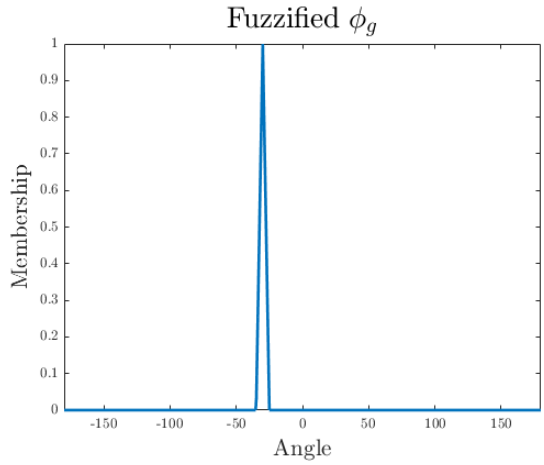


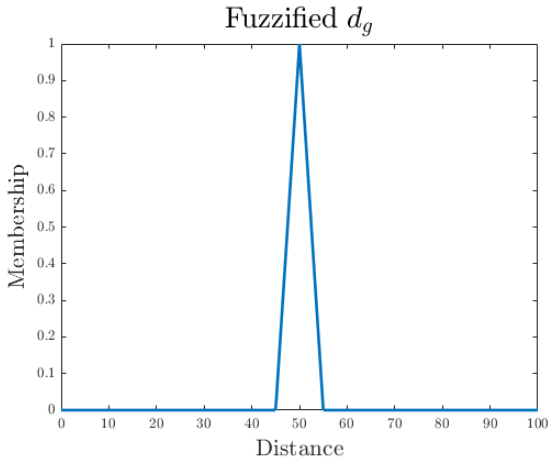


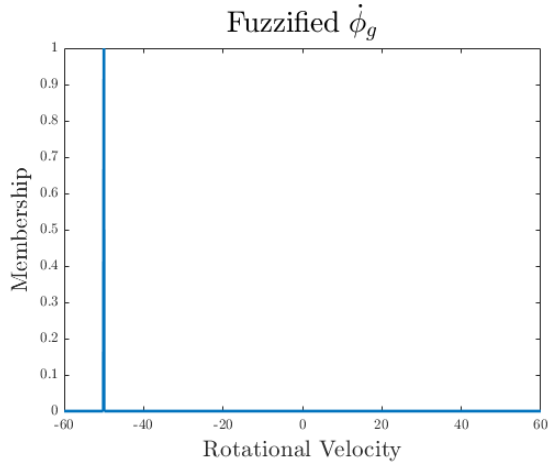


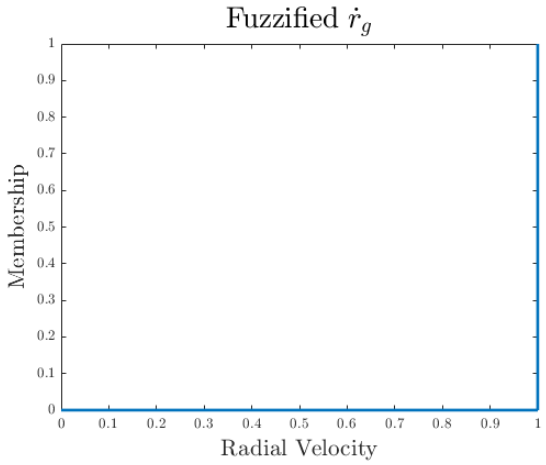


1b) $\phi, d, \dot{\phi}, \dot{r} = (-30, 50, -50, 1)$





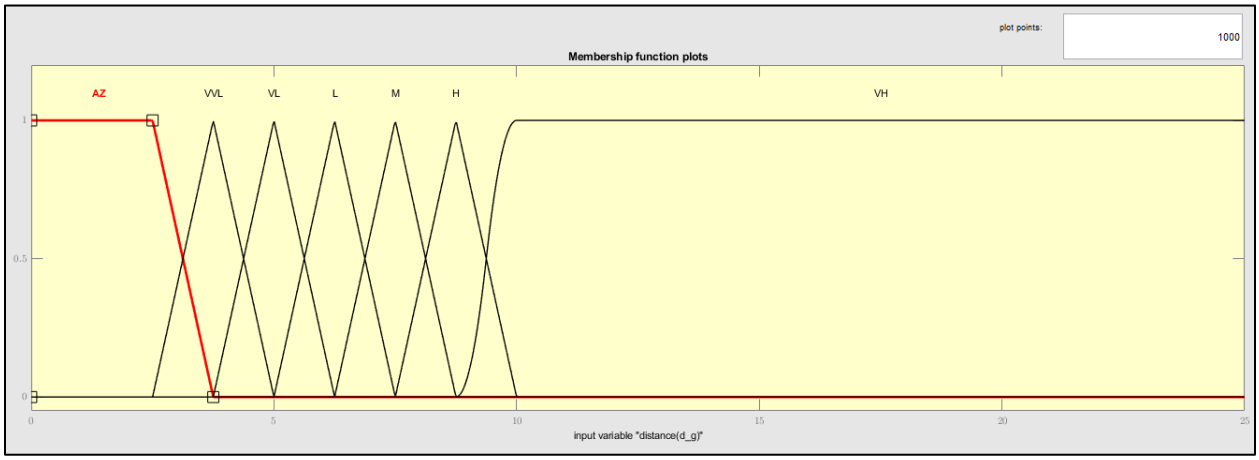




Published with MATLAB® R2019a

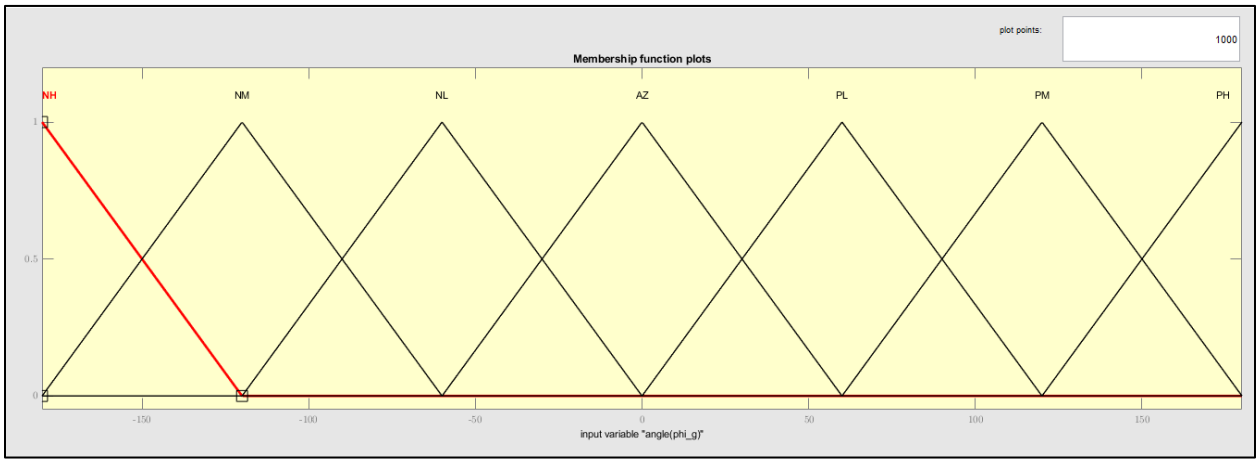
Question 2

Input I Fuzzified input for distance(d_g)



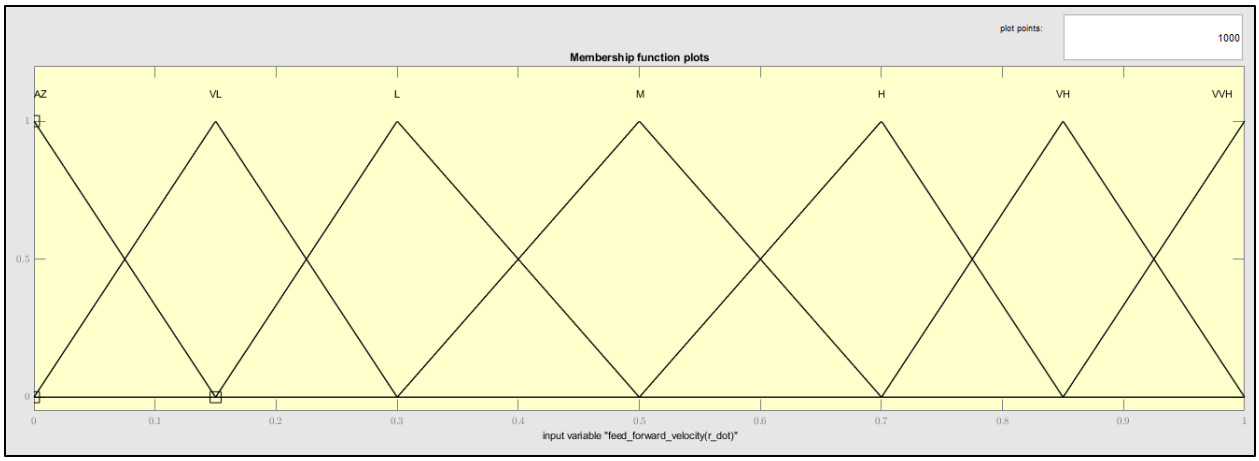
Fuzzy Set	Type	Interval
AZ: About zero	Trapezoidal	[0 0 2.5 3.75]
VVL: Very Very Low	Triangular	[2.5 3.75 5]
VL: Very low	Triangular	[3.75 5 6.25]
L: Low	Triangular	[5 6.25 7.5]
M: Medium	Triangular	[6.25 7.5 8.75]
H: High	Triangular	[7.5 8.75 10]
VH: Very High	Sigmoid	[8.75 10]

Input II Fuzzified input for Angle(ϕ_g)



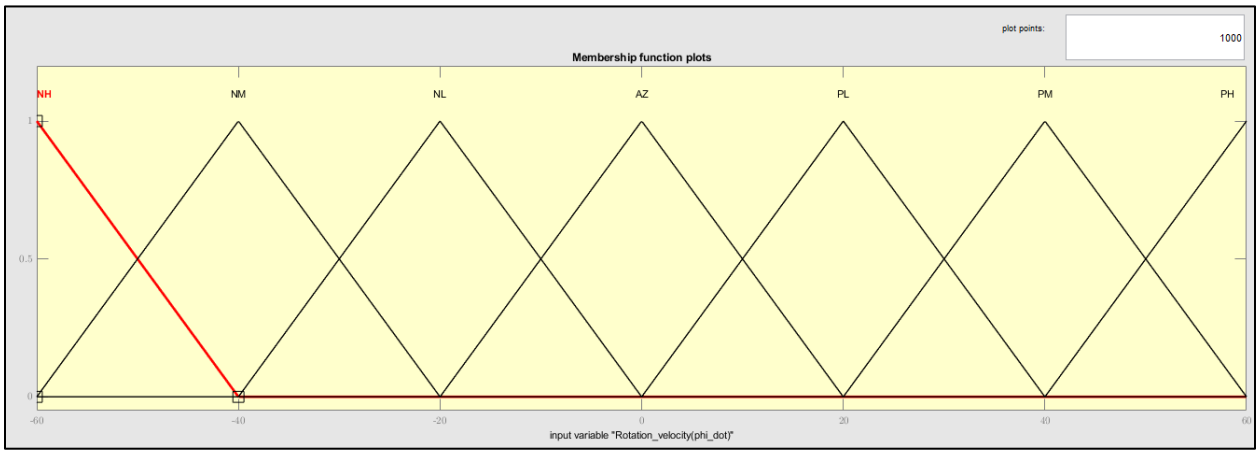
Fuzzy Set	Type	Interval
NH: Negative High	Triangular	[-180 -180 -120]
NM: Negative Medium	Triangular	[-180 -120 -60]
NL: Negative Low	Triangular	[-120 -60 0]
AZ: About Zero	Triangular	[-60 0 60]
PL: Positive Low	Triangular	[0 60 120]
PM: Positive Medium	Triangular	[60 120 180]

Input III Fuzzified input for forward velocity/radial velocity ($\dot{r}$)



Fuzzy Set	Type	Interval
AZ: About zero	Trapezoidal	[0 0 0.15]
VL: Very Low	Triangular	[0 0.15 3]
L: Low	Triangular	[0.15 0.3 0.5]]
M: Medium	Triangular	[0.3 0.5 0.7]
H: High	Triangular	[0.5 0.7 0.85]
VH: Very High	Triangular	[0.7 0.85 1]
VVH: Very Very High	Sigmoid	[0.85 1 1]

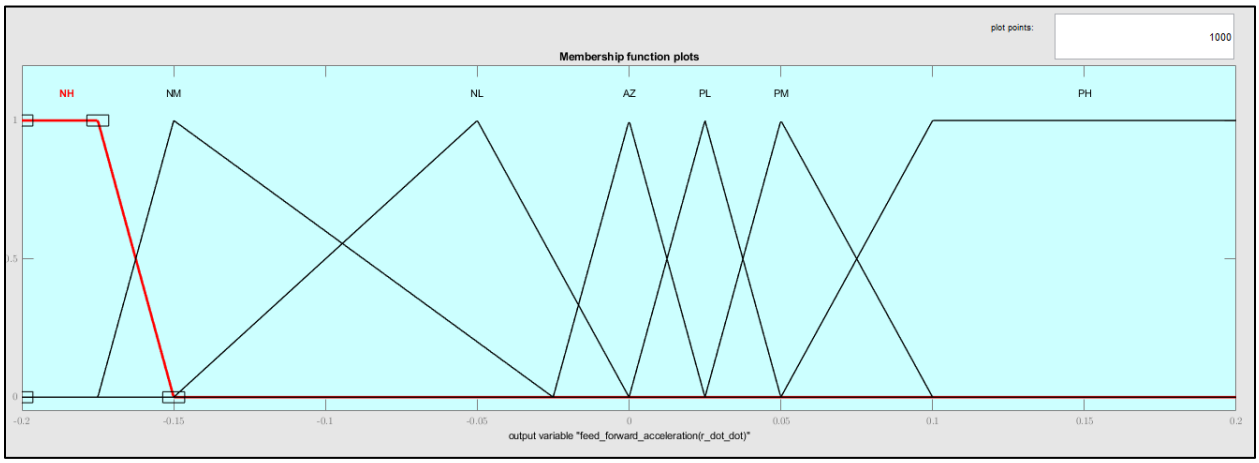
Input IV Fuzzified input for rotational velocity ($\dot{\phi}$)



Fuzzy Set	Type	Interval
NH: Negative High	Triangular	[-60 -60 -40]
NM: Negative Medium	Triangular	[-60 -40 -20]
NL: Negative Low	Triangular	[-40 -20 0]
AZ: About Zero	Triangular	[-20 0 20]
PL: Positive Low	Triangular	[0 20 40]
PM: Positive Medium	Triangular	[20 40 60]
PH: Positive High	Triangular	[40 60 60]

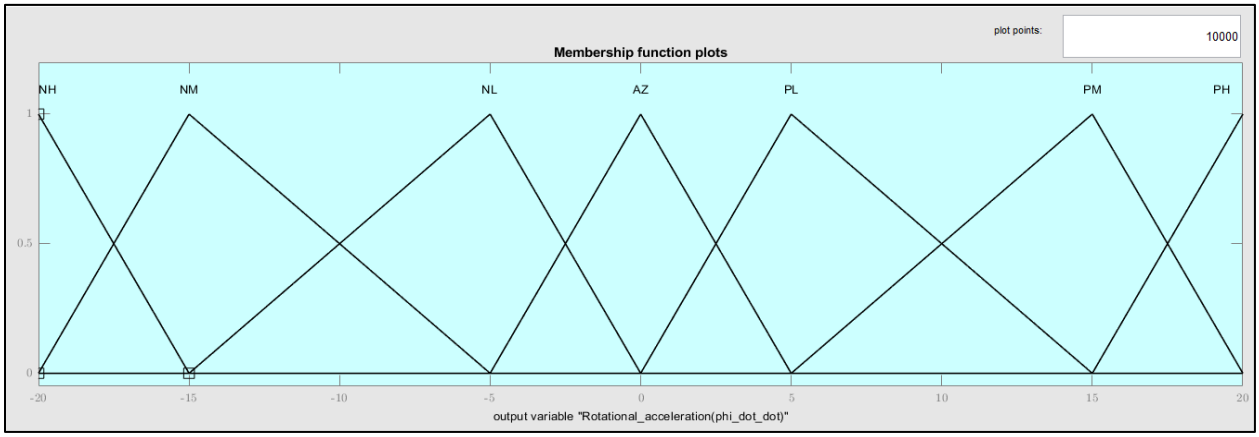
Question 3

Output I Fuzzified output for forward acceleration/radial acceleration ($\ddot{r}$)



Fuzzy Set	Type	Interval
NH: Negative High	Trapezoid	[-0.2 -0.2 -0.175 -0.15]
NM: Negative Medium	Triangular	[-0.175 -0.15 -0.025]
NL: Negative Low	Triangular	[-0.15 -0.05 0]
AZ: About Zero	Triangular	[-0.025 0 0.025]
PL: Positive Low	Triangular	[0 0.025 0.05]
PM: Positive Medium	Triangular	[0.025 0.05 0.1]
PH: Positive High	Trapezoid	[0.05 0.1 0.2 0.2]

Output II Fuzzified output for rotational acceleration ($\ddot{\phi}$)



Fuzzy Set	Type	Interval
NH: Negative High	Triangular	[-20 -20 -15]
NM: Negative Medium	Triangular	[-20 -15 -5]
NL: Negative Low	Triangular	[-15 -5 0]
AZ: About Zero	Triangular	[-5 0 5]
PL: Positive Low	Triangular	[0 5 15]
PM: Positive Medium	Triangular	[5 15 20]
PH: Positive High	Triangular	[15 20 20]

Question 4

Fuzzy Rules

I have made 8 sets of rules to drive my robot. Rule sets are as follows

I] Here I have related distance with radial acceleration directly. That means at high distances we want more acceleration, and when the robot approaches target, we want deceleration to slow it down.

- If distance is AZ, then radial acceleration is NH
- If distance is VVL, then radial acceleration is NM
- If distance is VL, then radial acceleration is NL
- If distance is L, then radial acceleration is AZ
- If distance is M, then radial acceleration is PL
- If distance is H, then radial acceleration is PM
- If distance is VH, then radial acceleration is PM

II] Here I have related angle with rotational acceleration. If we have positive angles, we have to angularly accelerate the robot in positive direction and if we have negative angles we have to angularly accelerate the robot in negative direction. The magnitude of the acceleration depends on the magnitude of the angle. Also, for zero angles we want zero angular acceleration.

- If angle is NH, then rotational acceleration NH
- If angle is NM, then rotational acceleration NM
- If angle is NL, then rotational acceleration NL
- If angle is AZ, then rotational acceleration AZ
- If angle is PL, then rotational acceleration PL
- If angle is PM, then rotational acceleration PM
- If angle is PH, then rotational acceleration PH

III] If we have feed forward velocity/radial velocity as zero we want the robot to accelerate otherwise it would get stuck.

- If radial velocity is AZ, then radial acceleration is PH

IV] Here, I have related rotational velocity inversely with rotational acceleration. These set of rules are just to control the rotational velocity in case it is becoming very large in magnitude.

- If rotational velocity is NH, then rotational acceleration is PH
- If rotational velocity is NM, then rotational acceleration is PM
- If rotational velocity is NL, then rotational acceleration is PL
- If rotational velocity is AZ, then rotational acceleration is AZ
- If rotational velocity is PL, then rotational acceleration is NL

- If rotational velocity is PM, then rotational acceleration is NM
- If rotational velocity is PH, then rotational acceleration is NH

V] These set of rules deaccelerate the robot if the angle is very high in magnitude. It is just to make sure that robot aligns itself first and then accelerate. Also, once the robot is aligned I have given a high positive radial acceleration.

- If angle is NH then radial acceleration is NH
- If angle is PH then radial acceleration is NH
- If angle is NM then radial acceleration is NH
- If angle is PM then radial acceleration is NH
- If angle is AZ then radial acceleration is PH

VI] These rules help to slow down the robot as it approaches the target. At lower distances we deaccelerate the robot.

- If distance is AZ and radial velocity is VVH then radial acceleration is NH
- If distance is AZ and radial velocity is VH then radial acceleration is NH
- If distance is AZ and radial velocity is H then radial acceleration is NH
- If distance is VVL and radial velocity is VVH then radial acceleration is NH
- If distance is VVL and radial velocity is VH then radial acceleration is NH

VII] Earlier we had a rule set where the rules slowed down the robot if the magnitude of angle was high. These set of rules speed up the robot in cases where magnitude of angle is high. If distance is high and angle is high we can give radial acceleration because robot would get ample time to realign its angle.

- If distance is VH and angle is PH then radial acceleration is PH
- If distance is VH and angle is NH then radial acceleration is PH
- If distance is H and angle is PM then radial acceleration is PM
- If distance is H and angle is NM then radial acceleration is PM

VIII] These set of rules deaccelerate robot radially if the rotational velocity is high. When rotational velocity is high we know that the robot is adjusting its angle, in this case we don't want the robot to accelerate.

- If rotation velocity is NH then radial acceleration is NH
- If rotation velocity is PH then radial acceleration is NH
- If rotation velocity is NM then radial acceleration is NM
- If rotation velocity is PM then radial acceleration is NM

Question 5

Defuzzification

When the rules get fired the resultant fuzzy sets are aggregated according to the activation levels. Conversion of these fuzzy sets to a crisp output that is single numeric value is defuzzification. There are many methods available for defuzzification. Out of which centroid defuzzification, which returns the center of area under the curve, is used in this work. MATLABs fuzzy logic toolbox already has a function to obtain defuzzified output. That function was used directly for defuzzification.

Formula for centroid defuzzification [1]

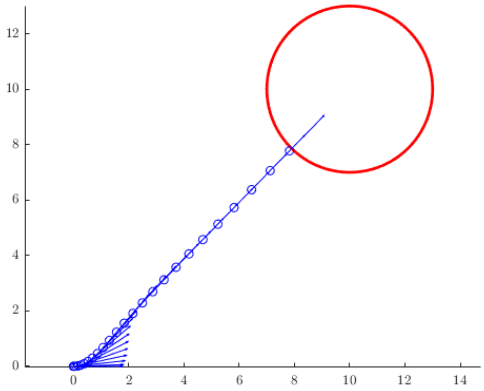
$$\bar{y} = \frac{\Sigma_{y \in Y}\ y * B'(y)}{\Sigma_{y \in Y}\ B'(y)}$$

Question 6

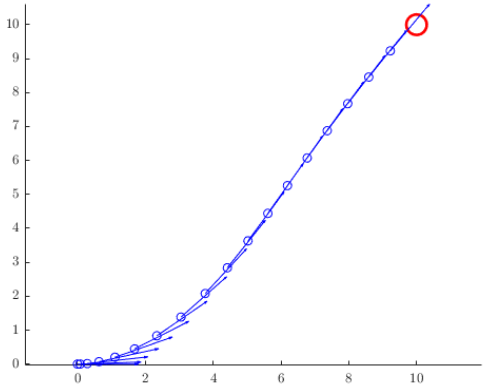
The controller is used to

Case	Goal	Goal D	Time steps taken
6a	[10 10]	[3]	243
6b	[10 10]	[0.3]	180
6c	[20 20; 5 20]	[1; 1]	576
6d	[0 10; 0 0; 20 0]	[4; 1; 5]	772
6e	[5 5; 4 4; 6 6; 3 6; 6 3]	[0.5; 0.5; 0.5; 0.5; 0.5]	1122

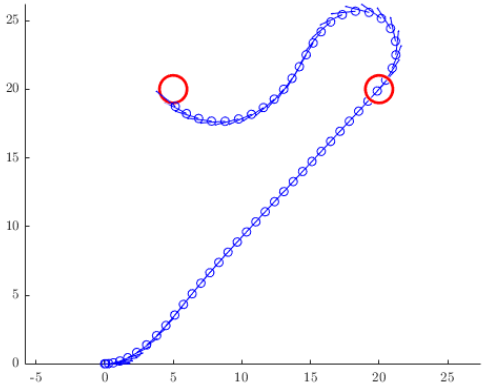
Question 6a



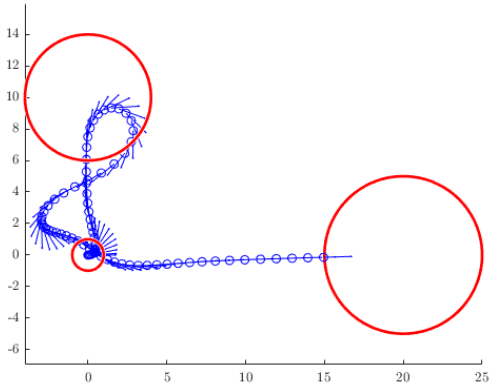
Question 6b



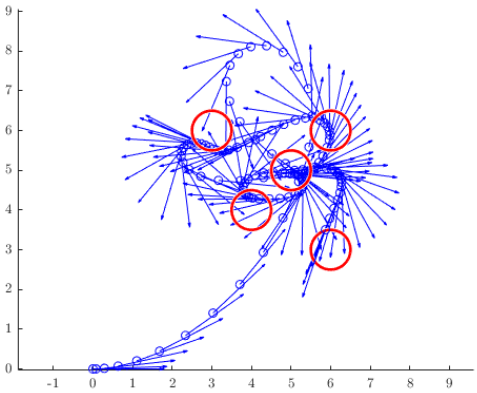
Question 6c

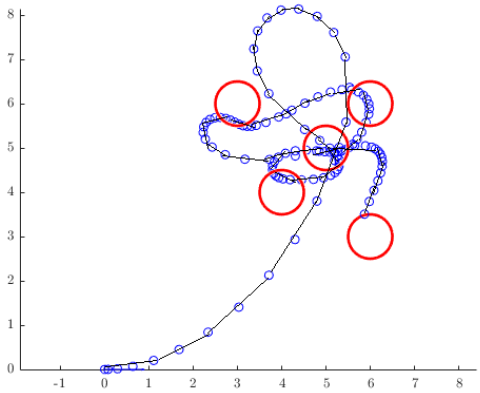


Question 6d



Question 6e





Question 7

Question 7a

Here are the proposed ideas which might improve performance of the controller.

1. Scaling of the distance is done using goal radius. In this case what happens is if the goal radius is high and distance is also high, scaled distance becomes low. The fuzzy rules slow down the robot in this case. On the other hand, if the goal radius is very low and distance is low then the scaled distance becomes high and the robot speeds up which is not wanted.

   This can be avoided using scale factor as original distance. Therefore, every time the scaled distance between robot and goal is between 0 and 1. In this case the fuzzy sets for distance can be created between 0 and 1.

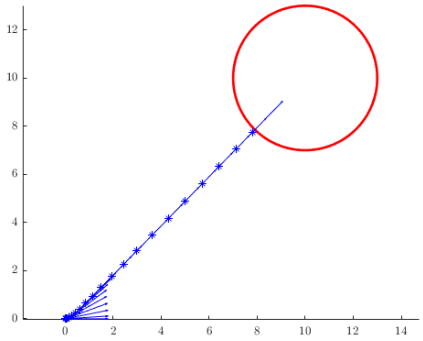
2. 7 fuzzy sets are not needed for all inputs and outputs. Desired functionality of the robot can be achieved even with lesser number of fuzzy sets.

3. Reverse motion can also be added to the robot which can reduce its time in aligning the angle in some cases.

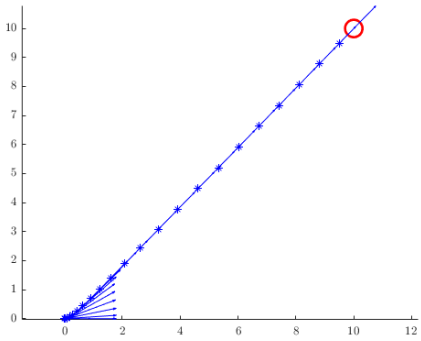
Question 7b

We have a controller which has 7 fuzzy sets for each input and output. But while designing rules, it was observed that not all 7 fuzzy sets are required to obtain desired operation of the robot. Hence, we try designing the **controller with just 3 fuzzy sets**. Performance of the robot can be observed through the path it takes and quantified using the time steps required. The paths new controller (3 fuzzy set controller) take for 5 defined cases are as follows:

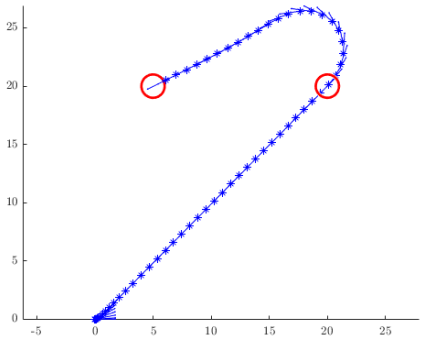
Case 6a



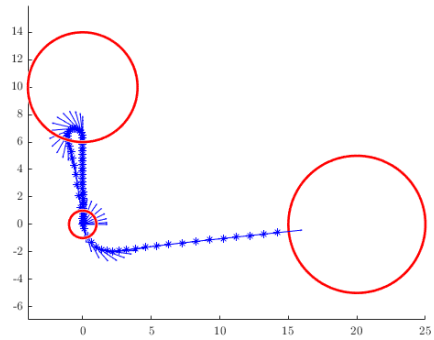
Case 6b



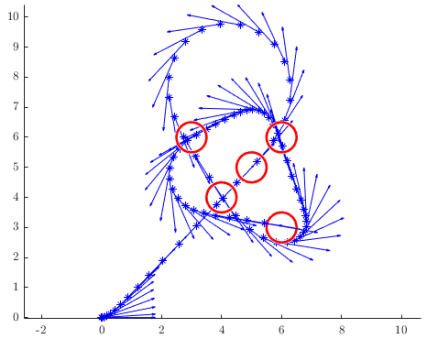
Case 6c



Case 6d



Case 6e



Comparison of the controllers with respect to the time steps taken to reach the goals is as follows

Case	Goal	Goal D	Time steps taken (7 fuzzy set controller)	Time steps taken (3 fuzzy set controller)
6a	[10 10]	[3]	243	203
6b	[10 10]	[0.3]	180	226
6c	[20 20; 5 20]	[1; 1]	576	594
6d	[0 10; 0 0; 20 0]	[4; 1; 5]	772	790
6e	[5 5; 4 4; 6 6; 3 6; 6 3]	[0.5; 0.5; 0.5; 0.5; 0.5]	1122	793

As it can be seen from the timings, 3 fuzzy sets controller is also working pretty well as compared to the 7 fuzzy sets controller.

Rules for 3 Fuzzy sets controller

The logic behind making rules for this controller is same as that was used earlier. Earlier we had 7 levels, now its three. Similar 8 set of rules are used, except now we have fewer rules in each set. To summarize the rules

- Distance should be directly proportional to radial velocity. For high distance we speed up, as distance becomes zero, we slow down.
- Magnitude as well as direction of angle is directly proportional to those of the angular acceleration.
- If we have zero radial velocity, we need to accelerate
- We don't want robot to have very high angular velocities, that's why we give an angular acceleration opposite to the angular velocity.
- If we have very high angle, we deaccelerate the robot radially, so that it gets time to realign its angle
- If the robot is very near to the target and we have high radial velocity, we slow it down
- We are giving time to robot to realign its angle before it speeds up, but if the target is far, we can speed it up. The robot will align itself along the path. These rules make the robot go in curved paths.
- If the robot has achieved high rotational velocity, we slow it down radially, because we don't want the robot to take longer curves to reach the target.

Along with the rules, fuzzy sets also play an important role in the working of controller. Fuzzy sets are also somewhat same as that used earlier. Previously used sets are merged together to form new sets. Example: About zero remains about zero, all positive fuzzy sets become just 1 fuzzy set as positive, all negative fuzzy sets become just 1 fuzzy set as negative. For cases where we had about zero to very very high, we cluster the sets as about zero medium and high.

Question 7c

Learnings

This work allowed me to understand working of a fuzzy controller. Fuzzy controller is designed using MATLAB fuzzy toolbox which is very easy to use. Working around the objectives of the project I have built a good intuition of fuzzy rules; fuzzy logic sets and their tuning. Operating a controller using fuzzy logic heavily depends on two things- fuzzy rules and fuzzy sets. The rules are to set such that the controller performs desired action in given scenario. Though rules guide the controller's action but tuning of the actions can be achieved by tuning of fuzzy sets. By doing exercise given in Q7b I got to observe that not always higher number of fuzzy sets and rules lead to better performance. Performance can be improved even with lesser rules and lesser fuzzy sets. I also understood that certain fuzzy controllers would be very good at achieving some goals and would be bad at achieving some goals. 2 fuzzy controllers designed to achieve same type of goals can differ in performance, depending on how the rules and fuzzy sets are optimized. On the whole, this work helped me understand working and tuning of fuzzy controller.

Acknowledgements

I would like to thank Dr. Tim Havens for designing this activity which helped me learn basics of fuzzy controller.

I would like to thank Michigan Tech IT services to provide me MATLAB License without which it was impossible to use Fuzzy toolbox to design this controller.

I am grateful towards J. Robert Van Pelt and John and Ruanne Opie Library to make the library computers available for this project.

I would also like to thank MathWorks for development of MATLAB Fuzzy toolbox.

Lastly, I would thank my fellow classmates who asked their doubts to Dr. Havens during classroom sessions. Responses of Dr. Havens to those doubts helped me understand the concepts further.

References

1. Fundamentals of Computational Intelligence: Neural Networks, Fuzzy Systems, and Evolutionary Computation – Keller, Liu, and Fogel (1st Ed)

2. https://www.mathworks.com/products/fuzzy-logic.html

Appendix

Question 1

Code to plot fuzzy sets

```
clc
clear all
close all

set(groot,'defaulttextinterpreter','latex');
set(groot, 'defaultAxesTickLabelInterpreter','latex');
set(groot, 'defaultLegendInterpreter','latex');
```

Question 1a

```
figure
x = -180:0.001:180;
y = trimf(x,[5 10 15]);
plot(x,y,'LineWidth',2)
xlim([-180 180])
title('Fuzzified ${\phi}_g$', 'Interpreter','latex','FontSize',20)
xlabel('Angle','FontSize',16)
ylabel('Membership','FontSize',16)

figure
x = 0:0.001:100;
dg = 10;
y = trimf(x,[dg-0.1*dg dg dg+0.1*dg]);
plot(x,y,'LineWidth',2)
xlim([0 100])
title('Fuzzified ${d}_g$', 'Interpreter','latex','FontSize',20)
xlabel('Distance','FontSize',16)
ylabel('Membership','FontSize',16)

figure
x = -60:0.001:60;
for i = 1:size(x,2)
    if x(1,i)==0
        y(1,i) = 1;
    else
        y(1,i)=0;
    end
end
plot(x,y,'LineWidth',2)
xlim([-60 60])
title('Fuzzified $\dot{\phi}_g$', 'Interpreter','latex','FontSize',20)
xlabel('Rotational Velocity','FontSize',16)
ylabel('Membership','FontSize',16)


figure
```

```
z = 0:0.0001:1;
for i = 1:size(z,2)
    if z(1,i)==0.5
        m(1,i) = 1;
    else
        m(1,i)=0;
    end
end
plot(z,m,'LineWidth',2)
xlim([0 1])
title('Fuzzified $\dot{r}_g$', 'Interpreter','latex','FontSize',20)
xlabel('Radial Velocity','FontSize',16)
ylabel('Membership','FontSize',16)
```

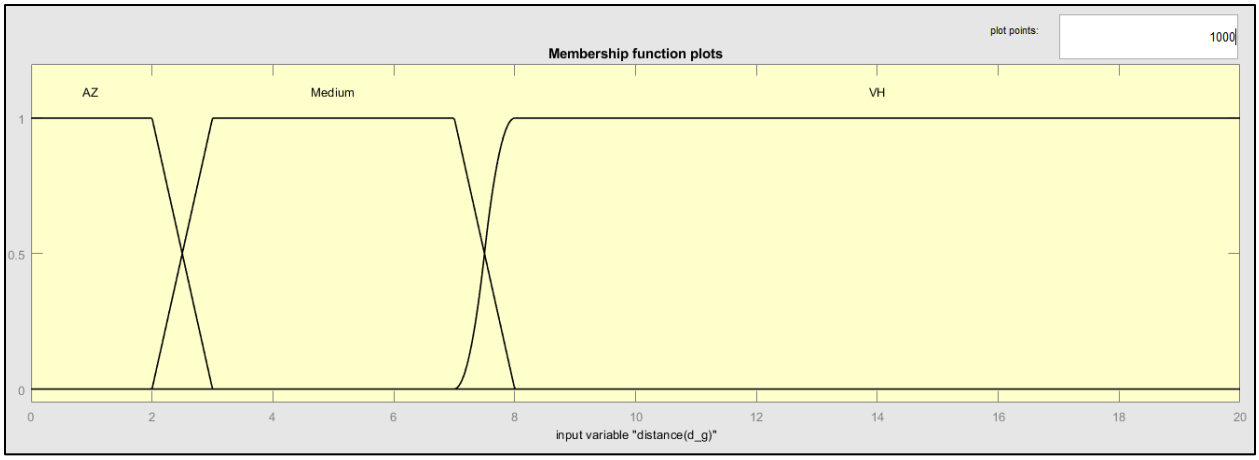
Question 1b

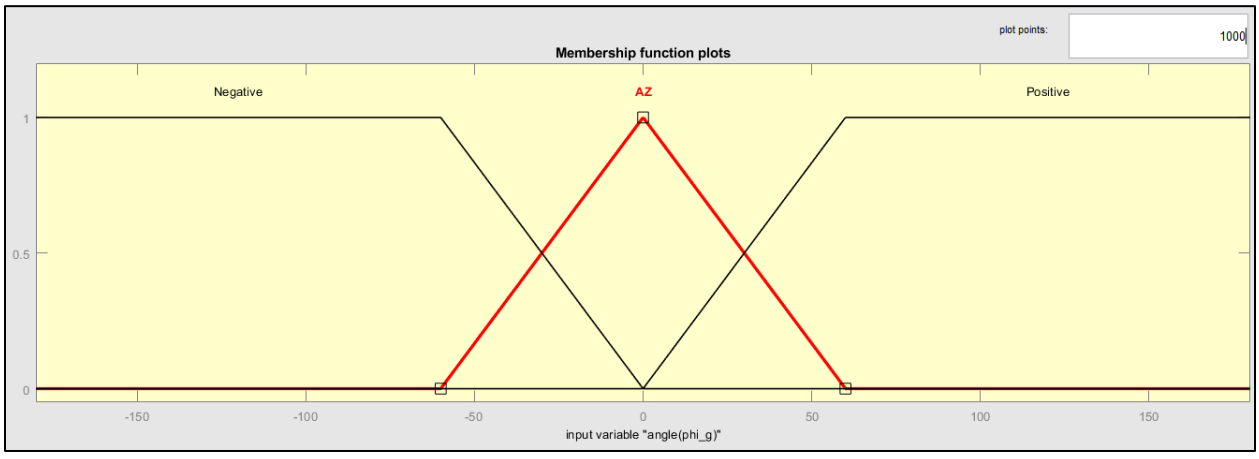
```
figure
x = -180:0.001:180;
y = trimf(x,[-35 -30 -25]);
plot(x,y,'LineWidth',2)
xlim([-180 180])
title('Fuzzified ${\phi}_g$', 'Interpreter','latex','FontSize',20)
xlabel('Angle','FontSize',16)
ylabel('Membership','FontSize',16)

figure
x = 0:0.001:100;
dg = 50;
y = trimf(x,[dg-0.1*dg dg dg+0.1*dg]);
plot(x,y,'LineWidth',2)
xlim([0 100])
title('Fuzzified ${d}_g$', 'Interpreter','latex','FontSize',20)
xlabel('Distance','FontSize',16)
ylabel('Membership','FontSize',16)

figure
x = -60:0.001:60;
for i = 1:size(x,2)
    if x(1,i)==-50
        y(1,i) = 1;
    else
        y(1,i)=0;
    end
end
plot(x,y,'LineWidth',2)
xlim([-60 60])
title('Fuzzified $\dot{\phi}_g$', 'Interpreter','latex','FontSize',20)
xlabel('Rotational Velocity','FontSize',16)
ylabel('Membership','FontSize',16)


figure
z = 0:0.0001:1;
```

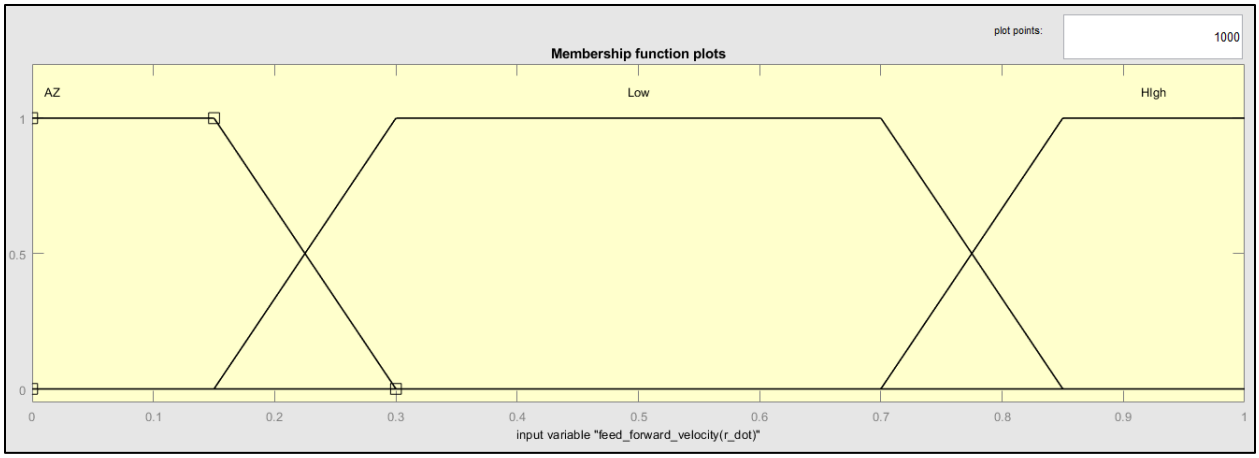
```
for i = 1:size(z,2)
    if z(1,i)==1
        m(1,i) = 1;
    else
        m(1,i)=0;
    end
end
plot(z,m,'LineWidth',2)
xlim([0 1])
title('Fuzzified $\dot{r}_g$', 'Interpreter','latex','FontSize',20)
xlabel('Radial Velocity','FontSize',16)
ylabel('Membership','FontSize',16)
```

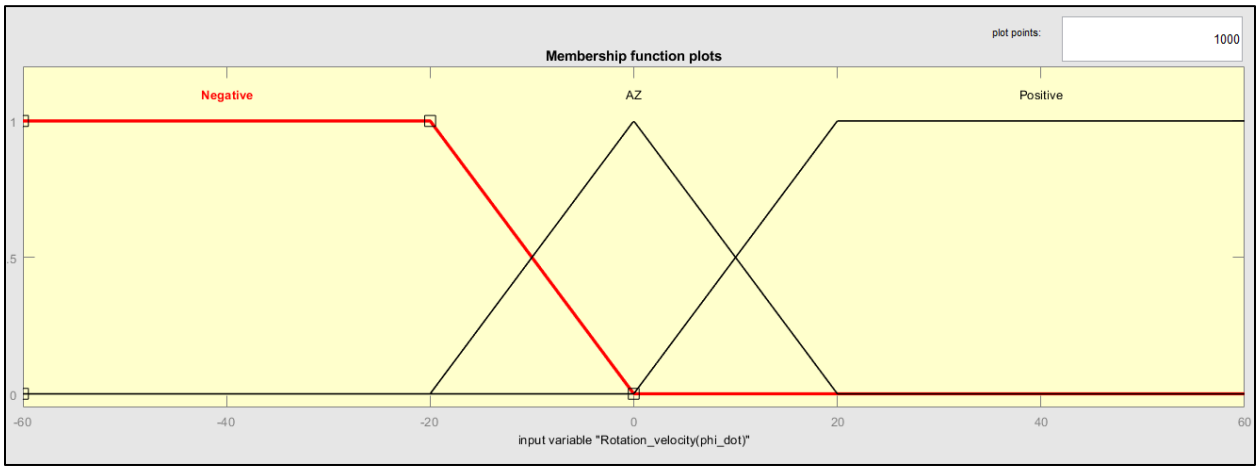
Question 7b

Following are fuzzy sets for inputs

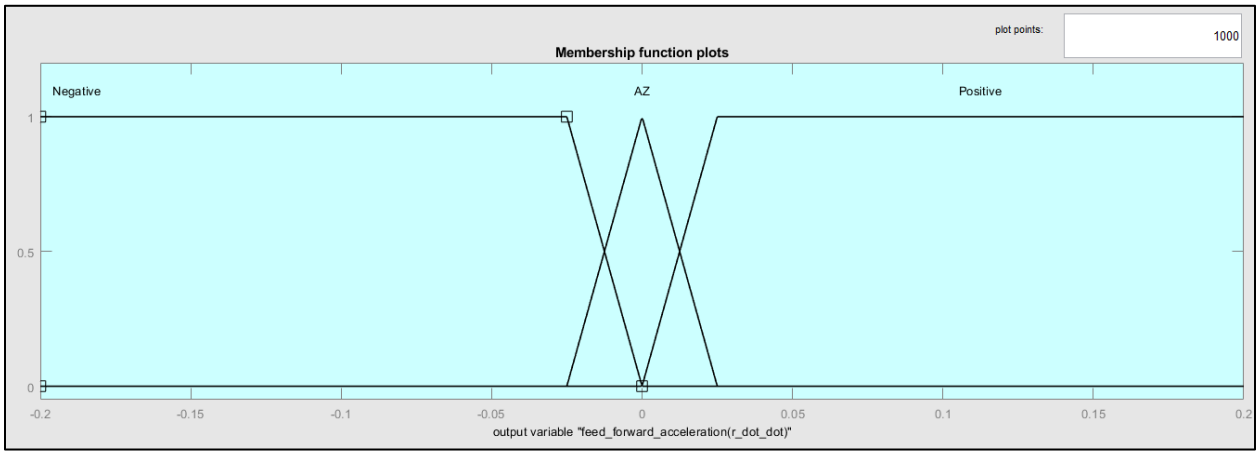


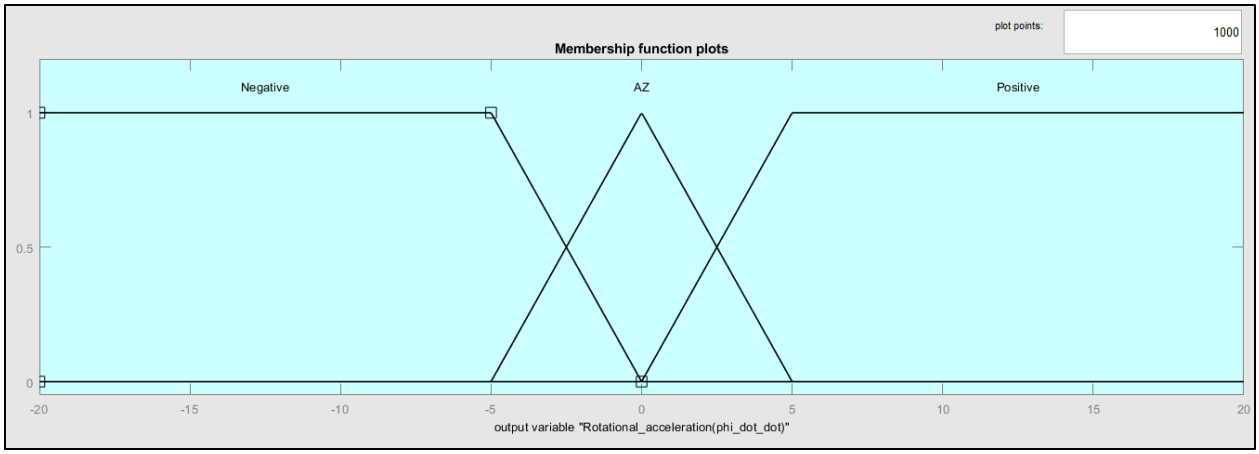






Following are fuzzy sets for outputs





Rules used for 3 fuzzy sets controller

1. If (distance(d_g) is AZ) then (feed_forward_acceleration(r_dot_dot) is Negative) (1)
2. If (distance(d_g) is Medium) then (feed_forward_acceleration(r_dot_dot) is AZ) (1)
3. If (distance(d_g) is VH) then (feed_forward_acceleration(r_dot_dot) is Positive) (1)
4. If (angle(phi_g) is AZ) then (Rotational_acceleration(phi_dot_dot) is AZ) (1)
5. If (angle(phi_g) is Positive) then (Rotational_acceleration(phi_dot_dot) is Positive) (1)
6. If (angle(phi_g) is Negative) then (Rotational_acceleration(phi_dot_dot) is Negative) (1)
7. If (feed_forward_velocity(r_dot) is AZ) then (feed_forward_acceleration(r_dot_dot) is Positive) (1)
8. If (Rotation_velocity(phi_dot) is Negative) then (Rotational_acceleration(phi_dot_dot) is Positive) (1)
9. If (Rotation_velocity(phi_dot) is Positive) then (Rotational_acceleration(phi_dot_dot) is Negative) (1)
10. If (Rotation_velocity(phi_dot) is AZ) then (Rotational_acceleration(phi_dot_dot) is AZ) (1)
11. If (angle(phi_g) is AZ) then (feed_forward_acceleration(r_dot_dot) is Positive) (1)
12. If (angle(phi_g) is Positive) then (feed_forward_acceleration(r_dot_dot) is Negative) (1)
13. If (angle(phi_g) is Negative) then (feed_forward_acceleration(r_dot_dot) is Negative) (1)
14. If (distance(d_g) is AZ) and (feed_forward_velocity(r_dot) is Low) then (feed_forward_acceleration(r_dot_dot) is Negative) (1)
15. If (distance(d_g) is AZ) and (feed_forward_velocity(r_dot) is Hlgh) then (feed_forward_acceleration(r_dot_dot) is Negative) (1)
16. If (distance(d_g) is VH) and (angle(phi_g) is Positive) and (feed_forward_velocity(r_dot) is Hlgh) then (feed_forward_acceleration(r_dot_dot) is Positive) (1)
17. If (distance(d_g) is VH) and (angle(phi_g) is Negative) and (feed_forward_velocity(r_dot) is Hlgh) then (feed_forward_acceleration(r_dot_dot) is Positive) (1)
18. If (Rotation_velocity(phi_dot) is Negative) then (feed_forward_acceleration(r_dot_dot) is Negative) (1)
19. If (Rotation_velocity(phi_dot) is Positive) then (feed_forward_acceleration(r_dot_dot) is Negative) (1)