Autonomous Car Platoon

Shahwar Saleem

I. Introduction

Autonomous Car

An autonomous car is a self driving car which has no dependency on human intervention to drive. This idea has been developing ever since computer and data scientists have started to think about intelligent systems. In late 90's there has been some research regarding how intelligent cars can be. In 1999, Laugier [1] talks about an "Automated Urban Vehicle" which came into ideas because of socio-economic and safety factors in the modern society. Recently, there has been a lot of discussion and word about Google's self-driving Car [2].

Autonomous Car Platoon

This project is basically an implementation of an Autonomous Car Platoon. The idea of the platoon is such that three self-driving cars on a straight piece of road. All these cars start with some initial velocity and some initial distances between them. Velocities and distances have to be specified by the user. Now, this platoon should be able to make every car drive with the 'Desired Velocity' and 'Desired Distance' specified by the user should be maintained between all cars. The acceleration or deceleration of the cars should be controlled with fuzzy logic. The system should use Fuzzy Inferencing System to decide for the amount of acceleration or deceleration produced in every car. With the help of suitable input values Fuzzy logic system should be able to output an acceleration value to adjust the velocity and distance between cars according to the user inputs.

Problem Statement

As explained before, goal of this project is to have a series of three cars maintain a fixed distance between each other while moving at a constant velocity down a straight piece of road.

The following initial conditions must be specified for the system to run:

- $V_{desired}$ - The desired velocity of the cars
- $D_{desired}$ - The desired distance between the cars
- D_{12} - The initial distance between the first two cars
- D_{23} - The initial distance between the second and third car
- $V_{initial}$ - The initial velocity of all cars

Initial acceleration is 0 for all cars. Once these conditions are specified, the system can be set in motion. The first car will adjust its rate of acceleration (or deceleration) trying to reach its desired velocity V_D. The second and third cars will observe the distance of the car in front, and adjust their acceleration accordingly so that the distance D_{12} and D_{23} match the desired distance D_D.

Requirements

This project requires to be implemented in following ways and some of other requirements are described in this section:

1) Fuzzy Logic Implementation: All adjustments will occur through the use of fuzzy rules. The following fuzzy variables will need to be maintained:

- V_N - The velocity of car n, where n=0, 1, 2. (The n=0 is the pilot car). Possible values: Very slow, slow, just right, fast, very fast. Here, "Just Right" denotes the case where V_n is the same as V_D. Likewise, slow denotes the case where V_N is slower than V_D, and acceleration needs to be applied.
- D_N - The distance between cars n and (n-1), where n = 1, 2. Possible values: Very far, far, just right, close, very close. Similar to the case with Velocity, "Just Right" denotes the case where the distance between cars n and (n-1) is the desired distance D_D.

Based on the values of these variables, a fuzzy output can be determined. For all cars, the output will be in the form of acceleration or deceleration. The following fuzzy output variables are needed:

- A_N - The acceleration of car n, where n=0, 1, 2. Possible values: Brake hard, brake, none, accelerate, accelerate hard

2) Neural Network Implementation: Once the fuzzy model for the system is developed, for which the control parameter is the acceleration component, build a data set out of this model and extract 1000 data points, where the input is the component input of the fuzzy system and the output is the acceleration value. Use 70% of the data points for training (under various placements of the cars), and 30% for testing and validation. Make sure to utilize various parameters of the neural network to get the best possible outcome.

3) GUI and Animation: The Java applet or MATLAB GUI is one of the deliverable for the project. It is a fully-functional autonomous car platoon visualization system that uses the discussed fuzzy logic inferencing engine to control a platoon of cars, and allow the positions of cars in that platoon to be graphically visualized. In their design, and to visualize the dynamics of the system under fuzzy inferencing, a dynamic graphing module that would show the velocity and acceleration of each of the cars as time progresses. This graph will update automatically and dynamically as each time step triggered and the cars were redrawn.

At run-time, a user manipulating the Java GUI should be able to change the value of V_D and D_D, and the system will react and adjust accordingly.

II. Fuzzy Logic Implementation

This section briefly discusses Fuzzy Logic and the approach applied to solve car platoon problem using Fuzzy Inference

System.

Fuzzy Logic

Let us get a little bit idea of Fuzzy Logic based system. We know that problems with such kind of control systems can be accurately solved using PID controllers or other control techniques. But when it comes to intelligent systems there are a lot of other complexities which may prove very significant while designing such problem such that complexity of such PID controllers could be a burden. In 1960's L.A Zadeh [3] introduced a very efficient way to override the complexity of the system but it deals with approximate knowledge. Usually a system should be dealing with crisp values such that a system must respond to exact values of an input value. Fuzzy logic deals with 'membership functions' such that a membership function corresponding to a crisp value assigns a certain value between 1 and 0 which describes how much the value is close to that crisp value. A very self explanatory and thorough discussion of Fuzzy Sets and Fuzzy logic is being presented by F. Karray and C. de Silva in 2004 [4]. In some studies [5], fuzzy logic is considered to be universal generalizing technique.

Approach towards Car Platoon

As section I discusses that the first car will adjust its velocity according to the desired velocity V_D entered by the user. Before moving further, a 'Fuzzifier' function needs to be decided. This project compares and utilizes two fuzzification techniques. Let us go through some of the assumptions considered before diving into implementation details of this project.

Assumptions for parameters: There are certain assumptions required to get started with the problem. Assuming all of these cars are having following aspects:

- Velocity Range = $0 - 80ms^{-1}$
- Maximum Acceleration = $50ms^{-2}$
- Maximum Deceleration = $-50ms^{-2}$
- Range of distance between cars = $0 - 20m$
- Time step for which simulation works = $0.01sec$

A car with that much acceleration values is very ideal but we are assuming for some reasons which need explanation and will be tackled later in another section. Distance between cars have been restricted to stick to a realistic value. A lot of distance between cars driving in a platoon would not be a good idea.

Another assumption and consideration for distance between cars to be calculated must be defined here. For every car, there needs to be a distance traveled parameter should be maintained. According to the initial conditions specified, system makes an assumption regarding the distance traveled by cars. This will help the system calculate the distance between the cars at every time step. With initial conditions specified, if $D_t(n)$ is the distance traveled by n^{th} car. Then initially:

- $D_t(1) = D_{12} + D_{23}$
- $D_t(2) = D_{23}$
- $D_t(3) = 0$

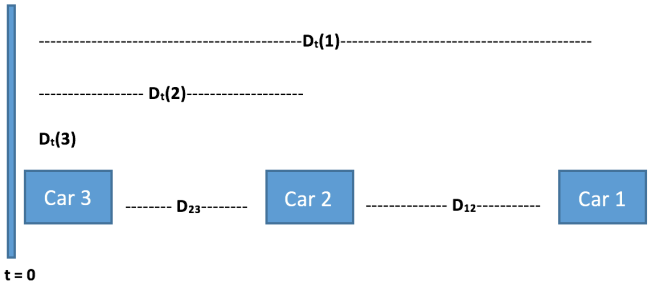

Fig. 1: Initial Conditions: Distance Traveled by Cars

Where D_{12} is initial distance between car 1 and 2. D_{23} is the initial distance between car 2 and 3.

Figure 1 helps the user visualize the distances between cars when they start. These distances traveled by cars will be helpful in simulation of the system.

Assumptions for linguistic variables: As section I-1 discusses about 5 linguistic variables being used for the membership functions. This assumption will play a significant role in performance of the project. Let us visit these assumptions. These assumptions are differently described on the basis of the type of membership function we are using.

1) Triangular Function Method: The simplest technique for fuzzification is Triangular Function method. This triangular curve surrounds the crisp values with 2 straight lines placed in a triangular fashion. Here is an expression which mathematically describes the triangular function:

$$\mu_A(v) = \begin{cases} 1 - \frac{|v - v_d|}{w}, & \text{for } |v - v_d| \leq w \\ 0 & \text{otherwise} \end{cases}$$

where w is the width of the triangle curve. Above membership function will have value 1 when v is exactly v_d. So if the velocity reaches desired velocity it is going to have maximum value i.e 1.

For simplicity, the values are described in the this fashion:
MF (Meaning): First minimum; Centre; Last minimum

Velocity membership functions can be arranged as described below.

- **VS (Very slow):** 0; 0; $V_D/2$
- **SL (Slow):** 0 ; $V_D/2$; V_D
- **JR (Just Right):** $V_D/2$; V_D; $V_D + V_D/2$
- **FS (Fast):** V_D; $V_D + V_D/2$; $2V_D$
- **VF (Very Fast):** $V_D + V_D/2$; $2V_D$; $2V_D$

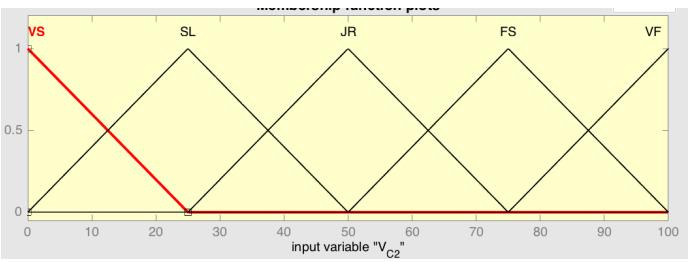

Fig. 2: Triangular Method: Fuzzy Membership functions for cars. Desired Velocity = 50m/s

Figure 2 makes it clear that how membership functions will arrange themselves on the basis of desired velocity set by user. Every time a user sets new desired velocity, the graphs of membership functions are updated immediately to adjust their peaks around V_D. In Figure 2 $V_D = 50ms^{-1}$.

Distance membership functions.

- **VC (Very Close):** 0; 0; $D_D/2$
- **CL (Close):** 0 ; $D_D/2$; D_D
- **JR (Just Right):** $D_D/2$; D_D; $D_D + D_D/2$
- **FR (Far):** D_D; $D_D + D_D/2$; $2D_D$
- **VF (Very Far):** $D_D + D_D/2$; $2D_D$; $2D_D$

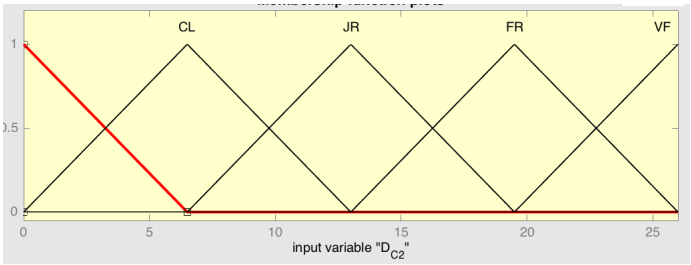

Fig. 3: Triangular Method: Fuzzy Membership functions distance between Car 1 and 2. Desired distance = 13m

Figure 3 shows the arrangement of membership functions when desired distance is 13 meters. These distances will keep adjusting there membership functions as soon as user inputs the desired distance. We would only have to worry about distance for Car 2 and 3.

Acceleration membership functions. 50 being the maximum acceleration allowed to the cars and -50 being the maximum deceleration allowed.

- **BH (Brake hard):** A_{min}; A_{min}; $A_{min}/2$
- **BR (Brake):** A_{min}; $A_{min}/2$; 0
- **NB (No brake):** A_{min}; 0; A_{max}
- **AC (Accelerate):** 0; $A_{max}/2$; A_{max}
- **AH (Accelerate hard):** $A_{max}/2$; A_{max}; A_{max}

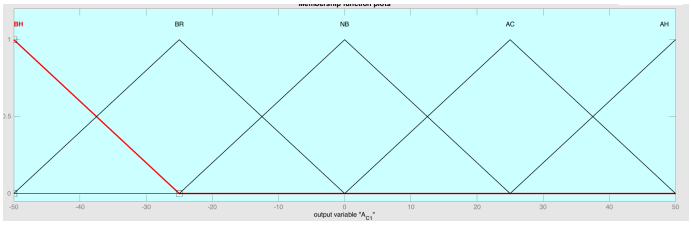

Fig. 4: Triangular Method: Fuzzy Membership functions for cars. Acceleration

Figure 4 shows how the acceleration function is distributed over various membership function. This is exactly in the same fashion as explained for velocity curves. This is much balanced and neat because of the assumptions made above have paid off just to make implementation smoother.

2) Gaussian Function Method: Another famous technique used for fuzzification of crisp value is Gaussian Function. This Gaussian curve is bell shaped famous curve being enormously used in mathematical calculations. Here is the expression in terms of velocity and desired velocity:

$$\mu_A(v) = e^{[\frac{v-v_d}{w}]^2}$$

where w denotes the parameter which controls the spread of the curve. If w is smaller, function is more sharp and hence less fuzzy and vice versa. The sharpness factor for these velocity membership functions have been decided by keeping in mind such that when for example velocity is exactly JR then other membership functions are zero. This is done using hit and trial method. Sharpness factor for every membership function and every parameter is also given below in terms of their desired value or the maximum value of the parameter.

The values are described in the this fashion:

MF (Meaning): Sharpness Parameter i.e w; Centre of bell curve.

Velocity membership functions can be arranged as described below.

- **VS (Very slow):** $V_D/6$; 0
- **SL (Slow):** $V_D/6$; $V_D/2$
- **JR (Just Right):** $V_D/6$; V_D
- **FS (Fast):** $V_D/6$; $V_D + V_D/2$
- **VF (Very Fast):** $V_D/6$; $2V_D$

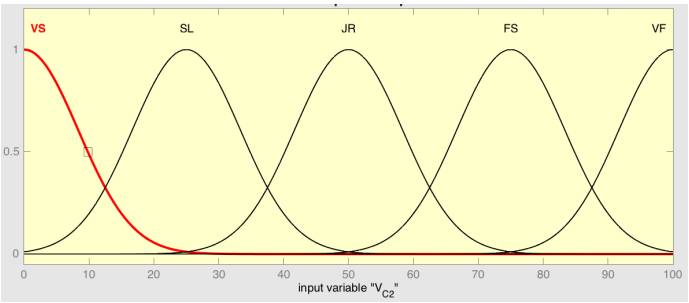

Fig. 5: Gaussian Method: Fuzzy Membership functions for cars. Desired Velocity = 50m/s

Figure 5 shows that membership functions utilizing the Gaussian curve have arranged themselves according to the assumption scheme described above.

Distance membership functions.

- **VC (Very Close):** $D_D/6$; 0
- **CL (Close):** $D_D/6$; $D_D/2$
- **JR (Just Right):** $D_D/6$; D_D
- **FR (Far):** $D_D/6$; $D_D + D_D/2$
- **VF (Very Far):** $D_D/6$; $2D_D$

Figure 6 shows the arrangement of membership functions using Gaussian Curves when desired distance is 13 meters. These distances will keep adjusting there membership functions as soon as user inputs the desired distance. We would only have to worry about distance for Car 2 and 3.

Acceleration membership functions. 50 being the maximum acceleration allowed to the cars and -50 being the maximum deceleration allowed.

- **BH (Brake hard):** $A_{max}/6$; A_{min}
- **BR (Brake):** $A_{max}/6$; $A_{min}/2$
- **NB (No brake):** $A_{max}/6$; 0
- **AC (Accelerate):** $A_{max}/6$; $A_{max}/2$
- **AH (Accelerate hard):** $A_{max}/6$; A_{max}

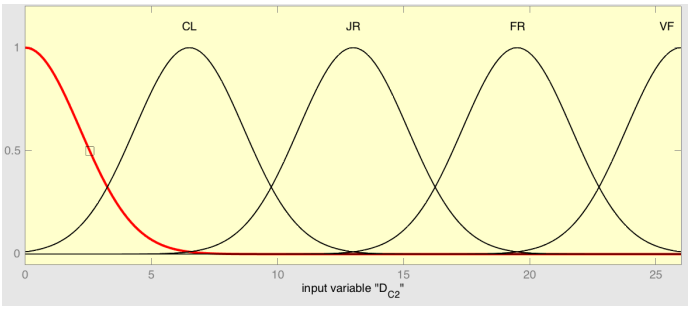

Fig. 6: Gaussian Method: Fuzzy Membership functions distance between Car 1 and 2. Desired distance = 13m

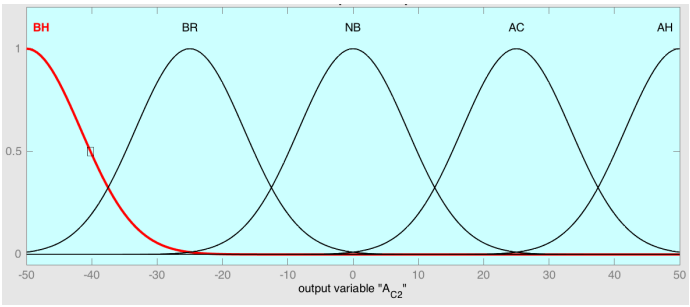

Fig. 7: Gaussian Method: Fuzzy Membership functions for cars. Acceleration

Figure 7 shows how the acceleration function is distributed over various membership function. This is exactly in the same fashion as explained for velocity curves.

Pilot Car FIS: The car in lead will have to adjust its velocity according to the desired velocity V_D entered by the user because there is no car in front of this car. This car will not be looking at any distances as input. There for there is only one input and one output to this Fuzzy Logic Inference system which Lead Car is going to have. Here are the rules which Fuzzy Logic Inference System will be applying to adjust the velocity of the car:

1) When velocity is VS; acceleration is AH
2) When velocity is SL; acceleration is AC
3) When velocity is JR; acceleration is NB
4) When velocity is FS; acceleration is BR
5) When velocity is VF; acceleration is BH

When these rules are programmed into a Fuzzy Logic system we can take a look at the rule vew that how these rules are going to look like graphically.

Figure 8 and 9 show how the rules will be initiated according to the rules defined above. These figures both implementations of Triangular Method and Gaussian Method respectively. Both of these figures show that when velocity is JR then only rule 3 is in action which is also highlighted in the figures. Acceleration is almost zero in this case because JR in velocity initiated NB in acceleration for this simple FIS.

Car 2 and Car 3 FIS

For car 2 and 3 the fuzzy inference engine is going to differ from what observed for car 1. This is because of the fact

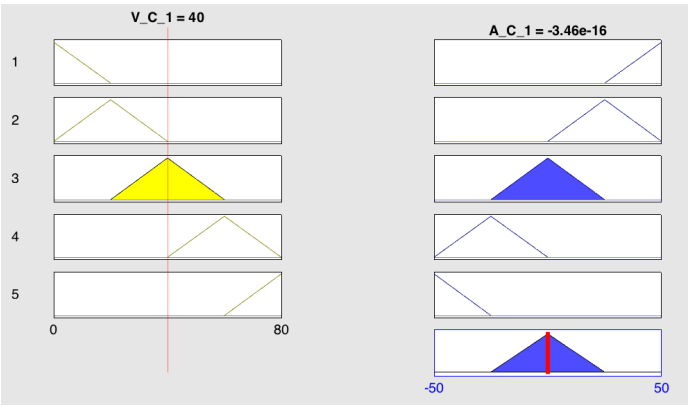

Fig. 8: Triangular Method: Rule View for Car 1

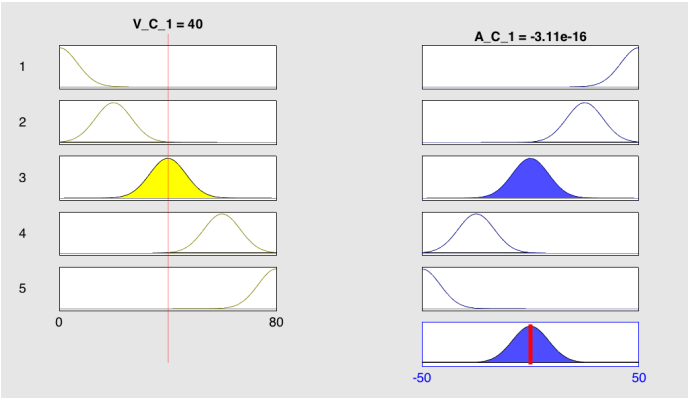

Fig. 9: Gaussian Method: Rule View for Car 1

that car 2 has to manage the desired velocity as well as the distance from car 1. In the same manner, car 3 has to manage distance between itself and car 2. Therefore, rules for these cars are going to have 2 inputs i.e desired velocity and desired distance. When these inference systems receive these V_D and D_D they apply some rules just like explained for car 1. But if we start writing combinations of these rules, depending on 2 inputs there are going to be 25 rules. For example:

1) When distance is VC **AND** velocity is VS; acceleration is BR
2) When distance is VC **AND** velocity is SL; acceleration is BR
3) When distance is VC **AND** velocity is JR; acceleration is BH

and so on. Therefore we employ Fuzzy Associative Memory (FAM) to summarize all the rules in the form of a Matrix or table. Table I describes the detailed FAM which mentions the association of Acceleration membership functions with the combinations of input velocity and distance ahead of the respective cars.

The rules summarized in the form of Table I can be visualized when fed to the inference systems for Triangular and Gaussian type of methods.

Figure 10 and 11 clearly show the rules which are fed inside the fuzzy inference system for car 2 and 3.

TABLE I: Fuzzy Associative Memory for Car 2 and Car 3 Inference System

Distance \ Velocity	VS	SL	JR	FS	VF
VC	BR	BR	BH	BH	BH
CL	BR	BR	BR	BR	BH
JR	AH	AC	NB	BR	BH
FR	AH	AC	AC	AC	AC
VF	AH	AH	AH	AC	AC

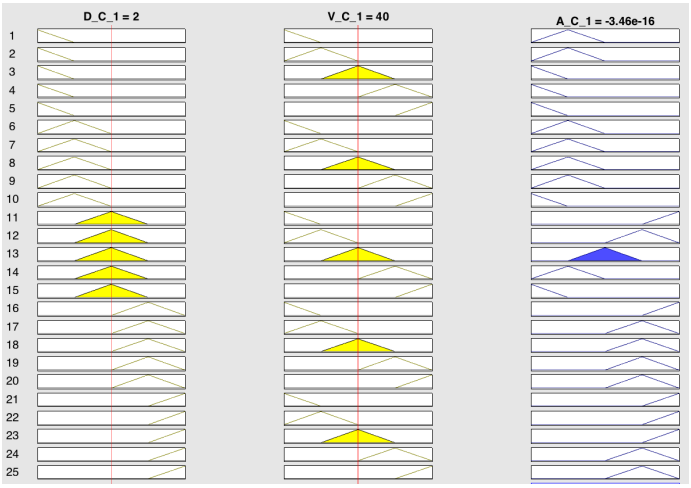

Fig. 10: Triangular Method: Rule View for Car 2 and 3

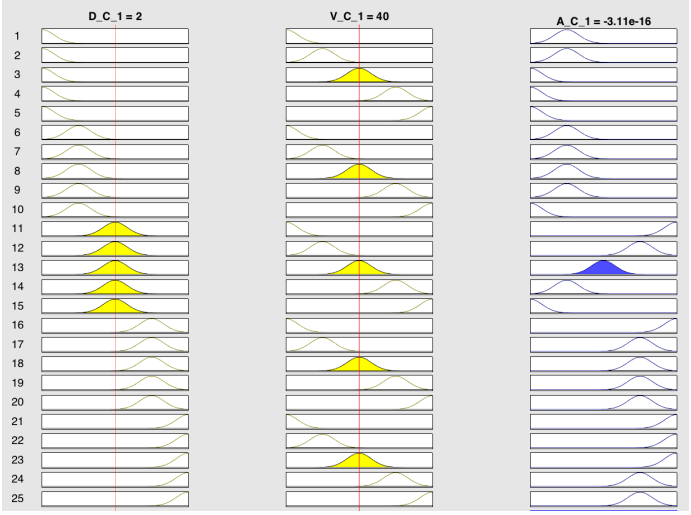

Fig. 11: Gaussian Method: Rule View for Car 2 and 3

Properties of FIS used

This project employs Mamdani FIS because of the requirements of this project. Mamdani is the simplest FIS system which can be used to solve problems like this. The main properties of this inference systems are:

- AND Method: min
- OR Method : max
- Aggregation : max

Mamdani FIS is a min-max approach which should be fine for the scope of this project.

Defuzzification Methods

This project experiments with 3 defuzzification methods. The experiments will be conducted in terms of time execution and accuracy of the results obtained. On the basis of these results in time, it will be decided which defizzufication technique proved best for this project. When the results of the rules are ORed, the output is in the form

Centroid Method: Centroid method takes the final function aggregated by the rules and it finds the exact centroid of the curve. So if the fuzzy inferencing control is given by $\mu_C(c)$ and it support set for which the membership grade is greater than zero is given by $S = c|\mu_C(c) > 0$ [4]. Then the centroid method of defuzzification can be given by:

$$cen = \frac{\int_{C\in E} c\mu_C(c)dc}{\int_{C\in E} \mu_C(c)dc}$$

Here cen is the defuzzified control action, which is given by the center of gravity of the membership function of control inference.

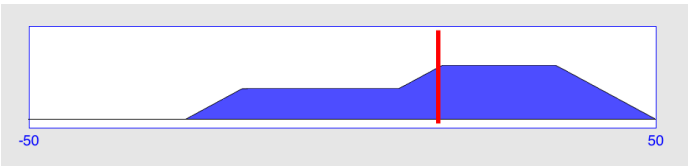

Fig. 12: Defuzzified Control Action (Acceleration) using Centroid Method.

Figure 12 shows when centroid method applied to the Acceleration curve obtained through FIS. It calculates the exact centroid of the area under the acceleration curve shown.

Mean of Maxima Method (MOM): This is relatively simple method for defuzzification. It takes the center of the maximum of the curve and utilizes the value of the acceleration corresponding to the centre of the curve. Figure 13 shows the concept. The crisp value is calculated corresponding to the red line on the acceleration area.

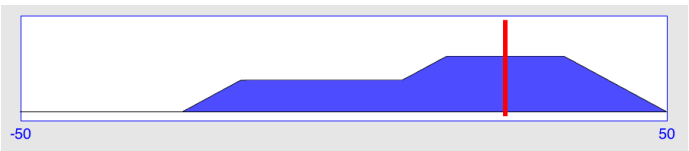

Fig. 13: Defuzzified Control Action (Acceleration) using MOM Method.

Bisector: Bisector method is a little bit close to the centroid but it is not the exact method. Bisector method finds the exact centre of the area distribution. So it lies more or less close to centroid but if the area is balanced or symmetrical then centroid and bisector do lie on the same point. Figure 14 shows where the bisector method calculates the acceleration value, as shown by the red line.

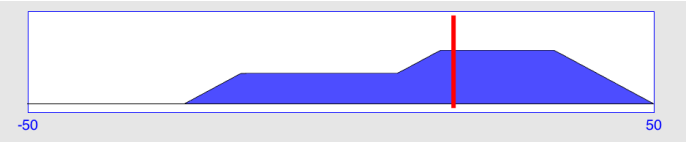

Fig. 14: Defuzzified Control Action (Acceleration) using Bisector Method.

System Dynamics

Now that Fuzzy Inference systems for 3 cars are ready, these systems will help in getting some acceleration values which is going to be fed to the respective cars. The main strategy here is that, the system builds the fuzzy systems according to every car when desired velocity and desired distances are fed to them according to their requirements. The system then calculates the acceleration value for every car which is fed to the System Dynamics which calculates the distance calculate on with new acceleration value. The expression for distance can be given by:

$$D(t + \delta t) = D(t) + v(t)\delta t + 0.5a(\delta t)^2$$

Along with distance, system also needs to calculate the value of the resultant velocity for the specific time chunk. This is the velocity which is achieved in the time interval after applying the acceleration attained in the previous time chunk. Here is the expression to calculate the velocities:

$$V_f = V_i + a(\delta t)$$

When these velocities and distances are calculated for every car. Note that distance calculated in the equation above equation for distance would be the distance traveled by the car. This distance traveled must be converted into inter-car distances so that the information could be again fed to the fuzzy inferencing system to complete the feed-back loop. Now if distance traveled by n^{th} car at time intance 't' is denoted by $D_n(t)$, then:

- $D_{12}(t) = D_1(t) - D_2(t)$
- $D_{23}(t) = D_2(t) - D_3(t)$

where $D_{12}(t)$ is the distance between car 1 and 2 at instance 't' and $D_{23}(t)$ is the distance between car 2 and 3 at instance 't'.

Integration of the modules

Now that every individual part of the system is thoroughly visited and discussed, it is of vital importance how these parts are going to be joined in order to give the required results. The information or data flow diagram in Figure 15 explains how the values of acceleration and velocity are going to circle around this closed loop system to provide the feedback to FISs of every car and then converge the system to required velocities and distances of the cars.

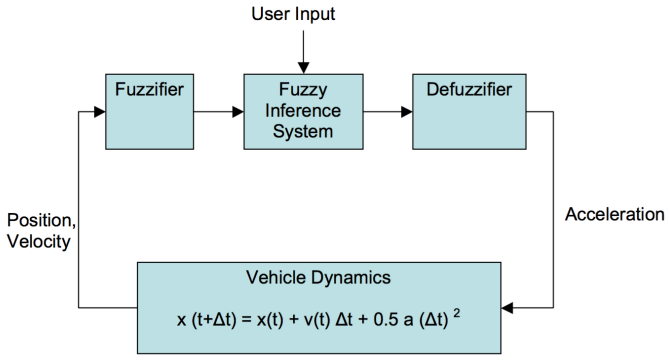

Fig. 15: System Dynamics closed system overview.

API and Code Files description

To implement the system described above there are 3 files present:

- carPlatoonFISGen.m
- Platoon_UI_With_Time.m
- CarPlatoon.m

'carPlatoonFISGen' is the generic implementation of FIS Generation function. It receives following parameters:

carPlatoonFISGen(fisFName, car_number, Vd, Dd, gauss):

where fisName is a 'String' value which is the name of the FIS. car_number is the number for car to make FIS. For lead car it take '1'. V_d is the desired velocity around which we want to construct our MFs for velocity. D_d is the desired distace value around which we will construct our distances values. 'gauss' is a parameter which is used to select the fuzzification system that will be used. If the user wants to construct MFs based on 'Gaussian Curve' if $gauss = 1$ then Gaussian curves will be used otherwise it will use 'Triangular' Fuzzification method.

NOTE: For car 1, user must pass D_d = -1, because distance does not make sense for car 1.

Platoon_UI_With_Time.m: This is the file which handles the behavior of the UI. Shortly we will have a look at how the GUI looks like. This file calls CarPlatoon.m and passes all the 'handles' in order to transfer control to the back end for GUI.

CarPlatoon.m: This file is basically actual back-end for simulation that would be visible in the UI. It controls everything. It gets handles from **Platoon_UI_With_Time.m** and manipulates the handles to control the UI values and graphs being plotted on UI. This file calls carPlatoonFISGen function multiple times to adjust the MFs according to the desired velocity and desired distances being mentioned.

Other API used: This project utilized very low level usage of API for Fuzzy Inferencing system. Every parameter in

the fuzzy logic has been programmed in detail by using CarPlatoon and carPlatoonFISGen files.

III. GUI and Simulation

GUI for this project has been programmed using MATLAB GUIDE. This is relatively simple to program because of the drag and drop option to generate the required GUIDE. GUI is being designed in textbfPlatoon_UI_With_Time.fig file which is also along with this project.

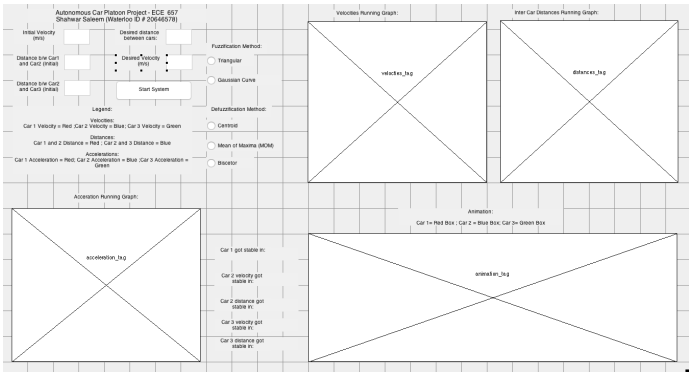

Fig. 16: GUIDE Implementation of GUI

Figure 16 shows how the GUI is being designed in GUIDE. This file automatically generates the **Platoon_UI_With_Time.m** file which when run, starts the GUI. Figure 17 gives a view of how the GUI looks like when running. It shows the system which has gone through the steps to reach the stable state. It can be seen that velocities have reached desired velocities and acceleration for every car has gone to zero.

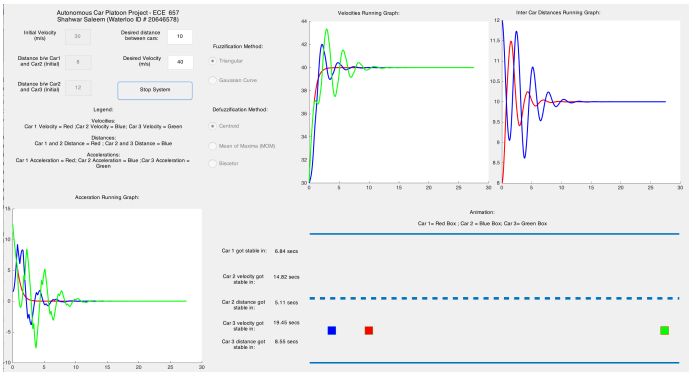

Fig. 17: GUIDE Implementation of GUI

The graphs which are shown in Figure 17 are running graphs for Acceleration, Velocities and inter car distances. The legends are being described in the GUI. The GUI is self explanatory. At the lower right corner of the GUI, there are 3 boxes moving on a simulated road. These are the boxes which represent cars which are moving on a straight road. The road is wrapped around because of the infinite simulation time being allotted to the system. The boxes color legend is also present on the GUI.

On the upper left corner of the GUI, there are edit boxes present for user to modify the values accordingly. When the values are specified completely, the user is free to press the 'Start System' push button in order to start the simulation. This will turn the graphs ON and the graphs will start updating themselves just according to the simulation.

Between the acceleration and simulation box, there are some timings mentioned. This information is basically self explanatory. It describes when any car's velocity or acceleration got stable. This module of the project was necessary because some results had to be compared between different techniques. This part would be explained more in the next section.

At just the center of the GUI, there are Fuzzification and Defuzzification method selectors. These selectors let user to select among different fuzzification methods which are described earlier and also defizzification methods can be selected just as described earlier. The control is fully given to the user to test any fuzzification or defuzzification method combination along with desired values specified by the user.

IV. Results - Fuzzy Logic Approach

In this section, results obtained from different fuzzy logic techniques will be explained. Before jumping into results, there is time measuring technique which needs explanation.

Measuring Stability Time

Measuring accuracy parameter for this particular is not so straightforward. Therefore, time taken for a car to get to stable the parameter would be considered as a success parameter for this project. Let us explain how to calculate this time for say velocity of car 1, other time calculations are done in exactly the same manner.

Code inside carPlatoon.m defines a running window called v_stable of 3 velocity values. This window is running on every velocity value that is calculated such that at time instance 'n' this window contains the values of velocities :

- v_stable(1) = $V_1(n-2)$
- v_stable(2) = $V_1(n-1)$
- v_stable(3) = $V_1(n)$

Therefore, whenever V_D or D_D values are changed by the user, this means that some instability has been introduced because the cars are of course not at the desired velocities and distances at that time. At this time, the system starts a timer in the form of a variable which keeps track of the running window v_stable. This check keeps checking at every time instance if these 3 values are equal to the V_D upto 4 decimal places. When all these values converge to V_D, the timer reports the time value to the UI and resets the timer at every other velocity or distance change in the UI.

This approach is implemented not only for velocities but also for distances in the same manner. For distances we keep a matrix d_stable for distances running 2D window. For velocities we keep v_stable matrix which is a 2D running window on all 3 velocity values of the cars. Figure 18 helps one imagine how these windows will be running on the velocity values. Hence, this running window technique is used to calculate the time for the system to get stable at different points. Window size can be increased or kept as desired according to the amount of precision required.

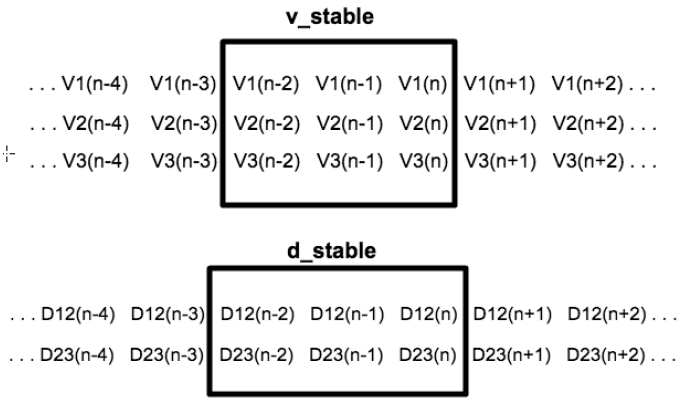

Fig. 18: Running windows illustrated

Stability Time statistics

This section describes in detail the results obtained with the help of the fuzzy logic approach. The system was run for different configurations possible because there are quite different parameters which vary as the user selects some techniques which are available in the UI. The parameters which can be changed are:

1) Fuzzification Method (Triangular Vs. Gaussian)
2) Defuzzification Method (Centroid Vs. MOM Vs. Bisector)

Also, there are different parameters whose timing are required to measure. Let us get familiar with some of short terms or symbols used for these parameters before we interpret them. These 5 parameters are:

- Velocity of Car 1: **V1**
- Velocity of Car 2: **V2**
- Velocity of Car 3: **V3**
- Distance between Car 1 and 2: **D12**
- Distance between Car 2 and 3: **D23**

Initial Conditions: While running this system for time measurements the initial conditions were kept same. These are randomly selected initial conditions used just to initiate the system and observe the behavior of the system.

Here are the initial conditions which will govern all of the comparisons that follow:

- $V_{initial} = 30ms^{-1}$
- $D_{12} = 8m$
- $D_{23} = 12m$
- $V_{desired} = 40ms^{-1}$
- $D_{desired} = 15m$

1) Fuzzification Methods Comparison: For fuzzification methods comparison, there needs to be one fixed defuzzification technique used. For only this method we only use Centroid defuzzification technique. The results are arranged in the form of tables and bar graphs which utilize short terms for the graphs to be concise. Table II summarizes some of the values obtained under different situations. This table mentions every value in seconds. Here

- **T1, T2, T3** signify that Triangular method was used.
- **G1, G2, G3** signify that Gaussian Fuzzification technique was used.
- Columns are stated for Velocites and Distances separately.
- T1(Initial Conditions): means that how much time when initial conditions were applied.
- T2(V=40-> 30): signifies that when system got stable. Velocity was changed to 30 to 40.
- T3(D=15-> 10): signifies that when system got stable from above disturbance. Distance was changed to 10 from 15.
- To generalize this transition, system started with T1 goes to T2 and then T3 separately for velocities and distances. In the same manner this variation is done for G1 -> G2 -> G3

In the same manner, every other condition is mentioned inside the table. We can see that for **T3** When the distance was changed, Car 1 got stable in 0 Seconds which is quite logical because the cars following have to adjust according to the requirement and Car 1 need no change itself because velocity is not changed. So car 1 keeps moving with the same velocity and Car 2 and 3 go stable in 11.41 and 13.54 seconds respectively for Triangular Method.

TABLE II: Comparison of Fuzzification Techniques

	V1	V2	V3		D12	D23
T1(Initial Conditions)	6.84	15.51	19.85	T1(Initial Conditions)	8.17	12.59
T2(V=40->30)	5.18	5.18	5.18	T2(D=15->10)	6.54	10.12
T3(D=15-> 10)	0	11.41	13.54	T3(V=40-> 30)	0	0
G1(Initial Conditions)	26.65	60.99	87.5	G1 (Initial Conditions)	18.48	34.92
G2(V=40->30)	19.64	32.44	55.01	G2(D=15->10)	24.67	30.77
G3(D=15-> 10)	0	65.54	93.91	G3(V=40-> 30)	0	0

Figure 19 segregates velocity values and summarizes them. In this figure, we can clearly observe that the Triangular Curve is a lot better than Gaussian Curve in terms of stability gain for this problem.

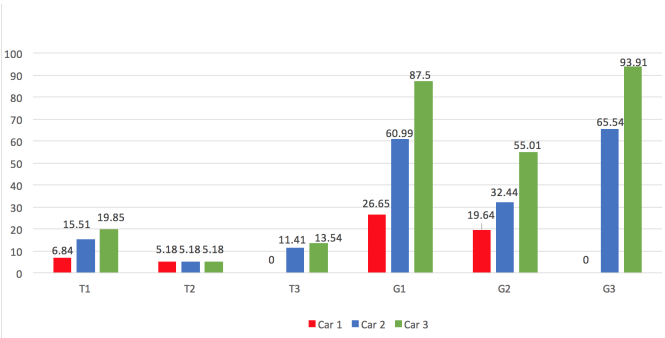

Fig. 19: Velocity Stability Timing (Fuzzification Method Comparison)

Figure 22 in the same manner summarizes the results in terms of distances getting stable. In this respect, again Triangular Method has performed a lot better than Gaussian Curve. For T3 and G3 both distances get stable in no time because of the obvious reason. For T3 and G3 velocity was changed when the system got stable in T2 and G2 condition. So when velocity of a stable system is changed, all of the

cars increase or decrease their velocities at same rates which is why their distances do not get unstable in this state and distance stability is achieved in no time.

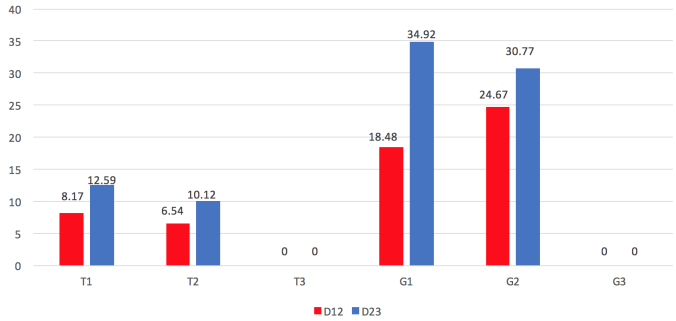

Fig. 20: Distance Stability Timing (Fuzzification Method Comparison)

2) Defuzzification Methods Comparison: For defuzzification, again we keep the fuzzification parameter constant. As 'Triangular Method' performed better in Fuzzification Comparison, therefore, for this comparison we only select Triangular Method for fuzzification and vary Defuzzification Techniques one by one.

Now that we are already familiar with the conditions technique. Here are the short terms used for every defuzzification method:

- Centroid Method: C1, C2, C3
- Mean of Maxima: M1, M2, M3
- Bisector Method: B1, B2, B3

For every technique the state transition is done from 1 to 2 and 2 to 3 after getting stability in every state. These transitions were performed again for velocity and distance separately. Table III summarizes all the results in seconds. There are 3 techniques being compared here. We can see a lot of infinity signs there in this table. This is because for the corresponding values, the system never got stable and did not achieve the desired values. Mean of maxima is the most unstable technique found in this problem, because it never lets the system get stable. For bisector method we see some distances get stable but velocities behave the same way.

TABLE III: Comparison of Defuzzification Techniques

	V1	V2	V3		D12	D23
C1(Initial Conditions)	6.84	15.51	19.85	C1(Initial Conditions)	8.17	12.59
C2(V=40->30)	5.18	5.18	5.18	C2(D=15->10)	6.54	10.12
C3(D=15-> 10)	0	11.41	13.54	C3(V=40-> 30)	0	0
M1(Initial Conditions)	∞	∞	∞	M1 (Initial Conditions)	∞	∞
M2(V=40->30)	∞	∞	∞	M2(D=15->10)	∞	∞
M3(D=15-> 10)	∞	∞	∞	M3(V=40-> 30)	∞	∞
B1(Initial Conditions)	∞	∞	∞	B1(Initial Conditions)	11.78	17.59
B2(V=40->30)	∞	∞	∞	B2(D=15->10)	10.28	∞
B2(V=40->30)	∞	∞	∞	B3(V=40-> 30)	0	0

Figure 21 summarizes the results in the Table III in terms of velocities only. So for M1, M2, M3 and B1, B2, B3 there is only infinite time required for the system to get stable.

Figure 22 in the same manner summarizes the results in terms of distances stability. For some transitions of Bisector

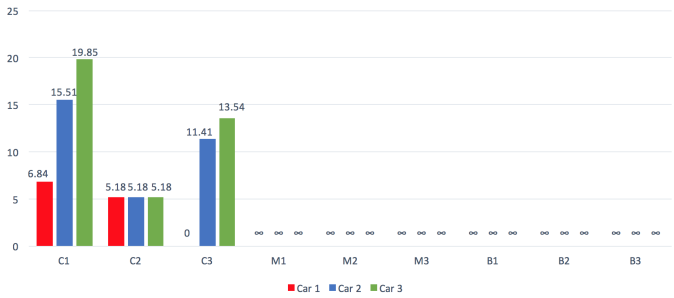

Fig. 21: Velocity Stability Timing (Defuzzification Method Comparison)

method, distances got stable but again this method is not so reliable because velocities never get stable.

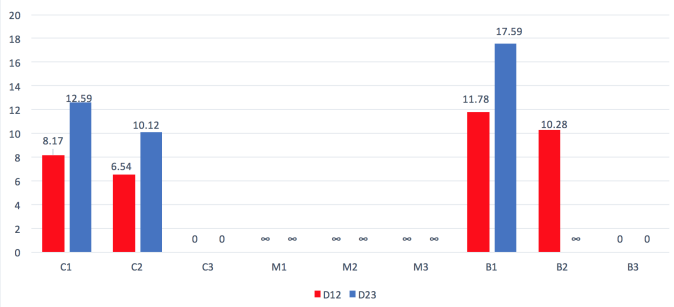

Fig. 22: Distance Stability Timing (Defuzzification Method Comparison)

This behavior was expected though. It is because of the idea that lies behind these techniques. Centroid Method is very complex but the results are really stable in that case. This is because Centroid method goes through all the pain to calculate the exact acceleration required. While MOM and Bisector method are relatively straightforward or more accurately intuitive techniques. Which is why these techniques do not calculate acceleration values that much precisely.

Interpretation of Results

We have compared different Fuzzification and Defuzzification techniques. According to numbers obtained in the research being carried out the best combination seems to be Triangular Method as Fuzzification technique and Centroid method as Defizzification method.

V. Neural Networks Implementation

In this method according to the requirements a Neural Network is required to be trained on 1000 data points with following division:

- Training : 70%
- Validation: 15%
- Testing: 15%

A little background for Neural Network would be helpful.

Neural Networks

Perceptron was invented in 1957 by Frank Rosenblatt. [6] This perceptron was supposed to be a machine but it was first realized as a computer program and thus it is an algorithm which classifies linearly separable data. Neural networks faced a lot of decline when perceptron limitations to classify limited type of classes came to light. Later, in 1982 Stephen Grossberg [7] gave a boost to Neural Network studies by publishing a series of papers exhibiting the capabilities of a perceptron in differential and XOR functionality.

Artificial neural networks (ANN) are large parallel systems consisting of an array of ”neurons” connected together by weighted connections. Each neuron performs a simple function of its inputs and generates a single output. This output is then propagated to other neurons via weighted connections. The information that a neural network learns is stored in the weighted connections which control how ”strong” certain neurons are connected to the rest. These connection weights are modified during the learning stage. Of course, there are variations of artificial neural networks that behave slightly differently, such as the radial basis function network. All ANNs are similar in that they all contain neurons with weighted connections to other neurons. ANNs are different from one another in how they are structured, the function performed by a neuron, and how they are trained.

Approach towards problem

In this approach we will use Artificial Neural Networks as a curve fitting tool because we need to achieve the acceleration values which are not binary. When used as a classification method, output of the values are normally in binary method, which is not the case here. For this approach we are going to vary the number of hidden neuron layers in order to compare the results when these parameters are varied.

API For Neural Network Approach: For Neural Network, there is a file called **PlatoonNN.c** which implements this approach. First we need to collect data for training Neural Network, therefore, we use the carPlatoonFISGen function used previously to collect data. Initial conditions used for collecting this data are given as follows:

- $V_{initial} = 40ms^{-1}$
- $D_{12} = 5m$
- $D_{23} = 7m$
- $V_{desired} = 20ms^{-1}$
- $D_{desired} = 10m$

These are the static hard coded initial conditions and can be seen coded inside PlatoonNN.m file. This simulation has nothing to relate to the previous simulation created. This mechanism only collects data in the form of matrices and arranges them in a form to be fed to Neural Networks.

At the end of data collection, there are 2 matrices formed which will be fed to the Neural Networks:

- Input_Mat - Input Data Set
- Acceleration - Target Values

Figure 23 shows first eight columns of the input matrix being calculated when data is collected inside this matrix.

	1	2	3	4	5	6	7	8
1	40	39.5800	39.1820	38.8016	38.4360	38.0828	37.7403	37.4071
2	40	39.5800	39.1820	38.8016	38.4360	38.0828	37.7403	37.4071
3	40	39.5902	39.2046	38.8375	38.4850	38.1441	37.8128	37.4894
4	5	5	5	5.0000	5.0000	5.0000	5.0000	5.0000
5	7	7	6.9999	6.9998	6.9995	6.9991	6.9985	6.9978
6	20	20	20	20	20	20	20	20
7	10	10	10	10	10	10	10	10

Fig. 23: First eight columns of ANN Input Matrix

The first three rows are velocities of respective cars. Next 2 rows contain changing distances between car 1,2 and 2,3 respectively. Finally rows 6 and 7 represent desired velocity and desired distance between cars respectively. Number of columns in this matrix are the number of instances for which this simulation is run, which by requirements is 1000.

The input matrix and acceleration matrix are not exactly input to the neural network just like that. Before we train neural network. We need to normalize the data. Therefore, we convert Input Matrix into Feature matrix and Acceleration Matrix into Target matrix. This data normalization is being achieved using Min-Max appoach.

VI. Results - Neural Network Approach

In this section let us review the results obtained will be varied over different hidden layers of neuron. Let us have a look.

A. 3 Hidden Layers

First the number of hidden layers allowed to the system were 3. Figure 24 shows the neural network when 3 hidden layers were used.

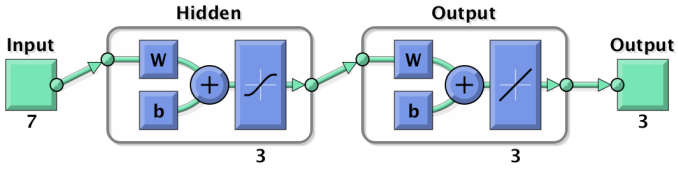

Fig. 24: Neural Network visualization with 3 Hidden Layers

Now the performance curve in Figure 25 shows that best performance was reached at 127 epochs which was really quite less error. The system at this level performs more than expectation.

Regression curves in Figure 26 show that the system closely follows the 45 degree line. Which indicates that there is significantly good performance shown by this method even with 3 layers. Regression value for 3 layered Neural Network in Training, Validation and Testing collectively is 0.99596 which is significantly satisfying.

B. 5 Hidden Layers

First the number of hidden layers allowed to the system were 5. Figure 27 shows the neural network when 5 hidden layers were used.

Now the performance curve in Figure 28 shows improvement in the number of epochs. Here the best performance was

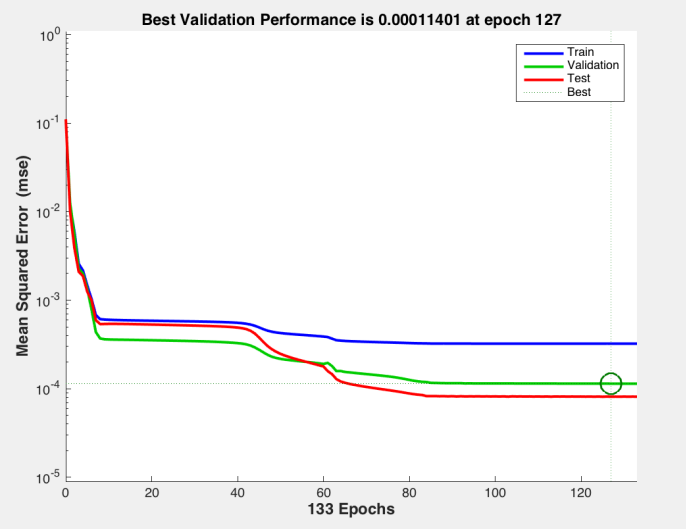

Fig. 25: Performance curve for Hidden Layers = 3

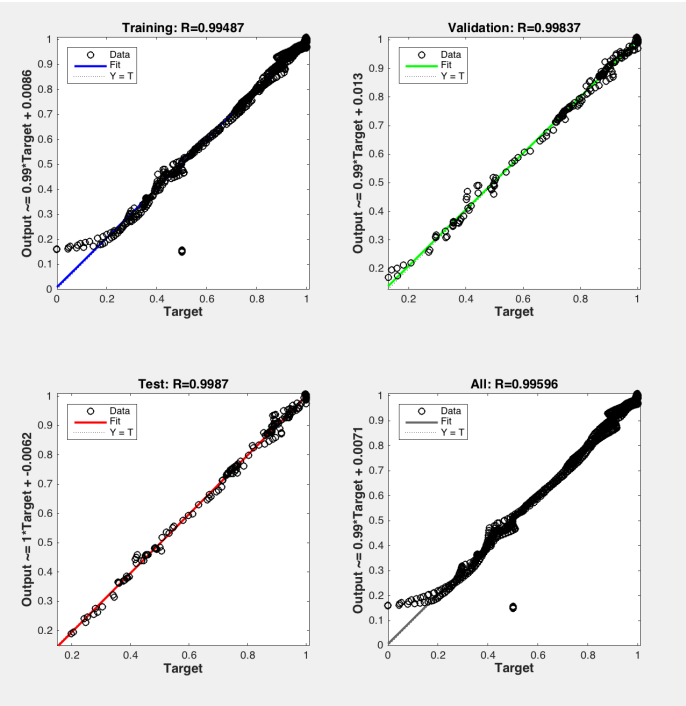

Fig. 26: Neural Network regression curves with 3 Hidden Layers

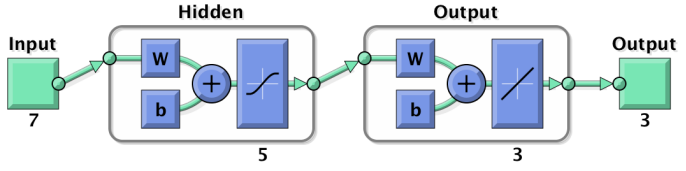

Fig. 27: Neural Network visualization with 5 Hidden Layers

found at epochs = 73. This has further reduced the number of epochs in this case. Though these epochs will vary every time we run this program with same parameters.

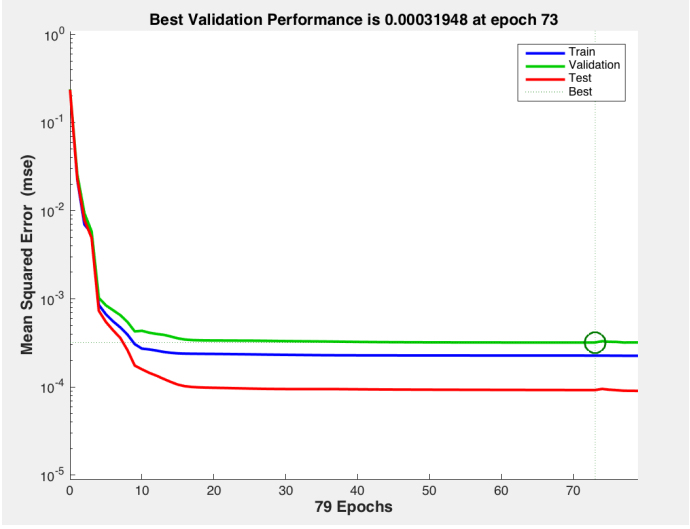

Fig. 28: Performance curve for Hidden Layers = 5

Regression curves in Figure 29 show more improvement such that R value for training, testing and validation has increased to 0.99654. Therefore, for now increasing the number of layers in this problem has somehow helped.

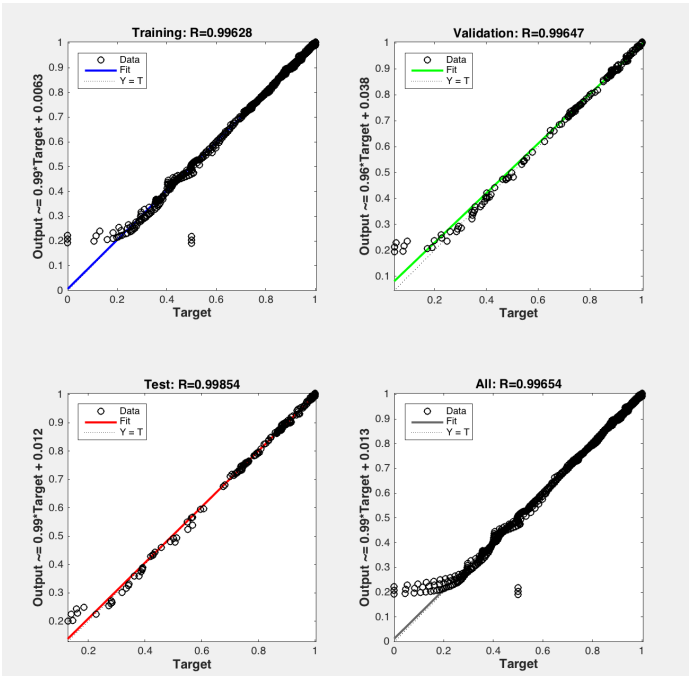

Fig. 29: Neural Network regression curves with 5 Hidden Layers

C. 7 Hidden Layers

First the number of hidden layers allowed to the system were 7. Figure 30 shows the neural network when 7 hidden layers were used.

Here Figure 31 shows that the best performance is shown to be at 121 epochs. We see that the number of epochs has

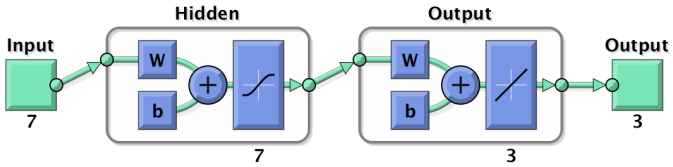

Fig. 30: Neural Network visualization with 7 Hidden Layers

increased in this case but performance is much better as can be that error value is much more less shown in Figure 31.

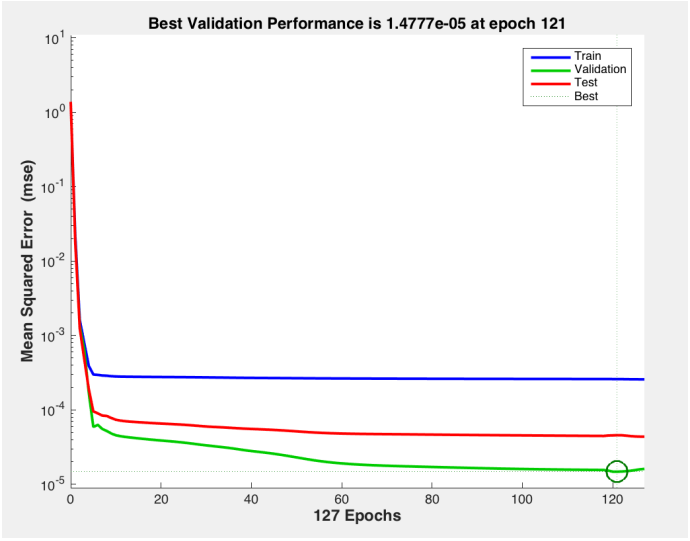

Fig. 31: Performance curve for Hidden Layers = 7

Regression value in Figure 32 is more improved in this case i.e 0.99699 which is better than before. Now let us proceed to summarize these results.

Interpretation of Results

We have seen that neural nets have been trained and the performances values are really promising in case of neural networks. As for training we have already used the best method from fuzzy logic technique i.e Triangular-Centroid combination of Fuzzification-Defuzzification technique which is why we can very aptly assume that even if this system follows the ideal system with 0.99 regression value it will show really good results.

To summarize, Neural Network performance with 5 layers is more beneficial because with less number of epochs it gave us the same performance.

Of course, there are some still some issues with the approaches used even though very exceptional performance achieved with the fuzzy logic approach. We will discuss these issues in the next section.

VII. Conclusion

Going through all the project, we have seen that with Fuzzy logic systems we have achieved a very convincing performance because the velocities and distances get really nicely stable.

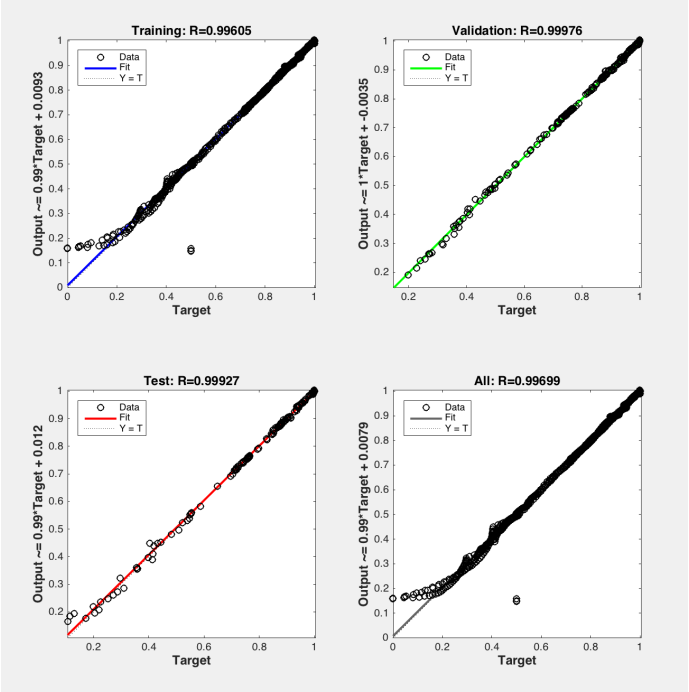

Fig. 32: Neural Network regression curves with 7 Hidden Layers

Achievements

In this study, fuzzy logic has shown itself to be a very powerful technique. Even without going deep into the system dynamics and without any complex integrator or differentiation module the system is very closely following a PID Controller curve. A PID controller is an electronics based circuit with very complex mathematics to deal with system dynamics . Figure 33 shows how the velocity graphs are smoothly mimicking a PID controller.

Goel, Arpit and Uniyal [8] talk about a fuzzy logic system designed to perform better than the ZN tuned PID controller or Fine tuned PID controller. So, a performance really close to a PID controller in this project is worth noticing.

Another achievement in this project was that, going through all the steps and testing with different fuzzification and defuzzification techniques to conclude that best technique among Fuzzification method in this particular problem turns out to be Triangular curve. Though Wang [5] talks about Gaussian and Centroid Combination to be a universal generalizing technique.

Even though the performance of the system was over all very good but still there are imperfections with this design. Let us review them.

Imperfections and Improvements

Gaussian performance: We have concluded Gaussian curve to be inferior than triangular fuzzification method. But it is rarely the case, because a study by Wang [5] showed better performances with Gaussian-Centroid combination.

Improvement: If we look at the design critically, there were assumptions made while deciding the limits for Gaussian

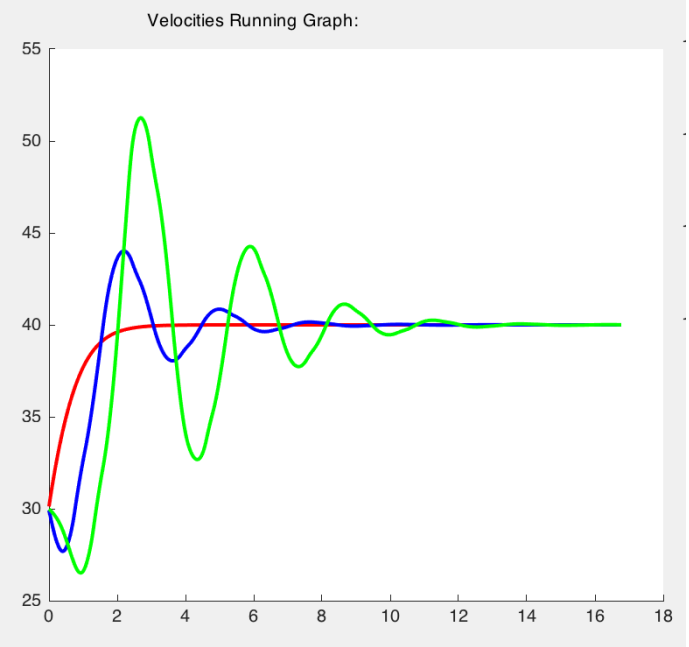

Fig. 33: PID Curve Follower

Curves being used as membership functions. The sharpness parameter decided on hit and trial basis can be improved to a more logical value so that the Gaussian curve overlapping in membership functions of input values can get closer to an optimal point such that the performance can surpass the Triangular Curve.

System Crash: Another issue with the implementation is that if the difference between desired distance and current distance between the cars is above than some certain value, then after few time instances the cars tend to collide. Due to car collision, the value of distance between those cars goes negative. When a negative distance value is fed to a constructed FIS to get acceleration it gives error that the input value has gone out of range. This is the point where system crashes. Because the velocity stops improving and distance starts going negative which results in the car surpassing its front car.

Improvement: The above system crash can be improved with the help of introducing an error checking mechanism. For example, if the distance is going negative it should sense at some meters before and add an error checking before the crash. This error checking will make a more realistic and real time system possible. If system senses some negative distance it should make the velocity constant instantly whatever the current value is and after making the velocity stable it should then start decreasing the velocity from that stable state. In this way, system going to unstable state can be avoided.

Crazy Acceleration/Deceleration: If we look at the fuzzy membership functions and acceleration and deceleration values allowed, these allowed values are way too much. This is basically making the system very ideal, but in real life cars with such big accelerations are not designed for daily use.

Improvement: Assigning a threshold to acceleration or deceleration value will make the system achieve stability quite later but it will be more realistic and applicable to real world autonomous car platoons.

With these improvements we can certainly improve the performances of the system. This system was designed on very simpler level and yet the performances were really convincing. In 2003 [9] a study has shown a nice and robust solution to inter vehicle gap management using fuzzy logic technique. Recently in 2013, Driankov [10] describes some challenges involved in modeling fuzzy logic systems for Autonomous Vehicles specifically. For such systems, real time feedback and real time solution of the velocities and accelerations matters a lot. Such improvements and the ones mentioned in this study can certainly help the system improve the performance to an optimal and realistic level.

REFERENCES

[1] C. Laugier, I. Paromtchik, and M. Parent, "Developing autonomous maneuvering capabilities for future cars," in *Intelligent Transportation Systems, 1999. Proceedings. 1999 IEEE/IEEJ/JSAI International Conference on*. IEEE, 1999, pp. 68–73.

[2] E. Guizzo, "How google's self-driving car works," *IEEE Spectrum Online, October*, vol. 18, 2011.

[3] L. A. Zadeh, "Fuzzy sets," *Information and control*, vol. 8, no. 3, pp. 338–353, 1965.

[4] F. O. Karray and C. W. De Silva, *Soft computing and intelligent systems design: theory, tools, and applications*. Pearson Education, 2004.

[5] L.-X. Wang, "Fuzzy systems are universal approximators," in *Fuzzy Systems, 1992., IEEE International Conference on*. IEEE, 1992, pp. 1163–1170.

[6] F. Rosenblatt, *The perceptron, a perceiving and recognizing automaton Project Para*. Cornell Aeronautical Laboratory, 1957.

[7] S. Grossberg, "Contour enhancement, short term memory, and constancies in reverberating neural networks," in *Studies of mind and brain*. Springer, 1982, pp. 332–378.

[8] A. Goel, A. Uniyal, A. Bahuguna, R. S. Patwal, and H. Ahmed, "Performance comparison of pid and fuzzy logic controller using different defuzzification techniques for positioning control of dc motors," *Journal of Information Systems and Communication*, vol. 3, no. 1, pp. 235–238, 2012.

[9] J. E. Naranjo, C. González, J. Reviejo, R. García, and T. De Pedro, "Adaptive fuzzy control for inter-vehicle gap keeping," *IEEE Transactions on Intelligent Transportation Systems*, vol. 4, no. 3, pp. 132–142, 2003.

[10] D. Driankov and A. Saffiotti, *Fuzzy logic techniques for autonomous vehicle navigation*. Physica, 2013, vol. 61.