Cascaded Fuzzy Logic for Adaptive Cruise Control

Milan Simic

School of Engineering, RMIT University, Australia

emails: *milan.simic@rmit.edu.au*

ARTICLE INFO

Article History:

Received: 16th November 2021
Revised: 18th May 2022
Accepted: 23rd May 2022
Published online: 26th June 2022

Keywords:

Fuzzy logic control
Adaptive cruise control
Cascade controller
Road vehicles
Driver assistance

ABSTRACT

Application of fuzzy logic is a powerful approach that could be applied in a large number of disciplines, starting with engineering control systems, as shown here, but also in other business areas. After a short introduction to fuzzy logic, its application for adaptive cruise control (ACC) is presented. ACC is a driver assistance feature that deals with the problem of speed control, while keeping the safe distance from the vehicle ahead. In the hierarchy of autonomous vehicles autonomy levels, as defined by Society of Automotive Engineers (SAE) International, adaptive cruise control appears in the vehicles at the level 1 and above. We developed a fuzzy logic controller where controlled variables are speed and distance. Input variables include weather conditions, style or mode of driving, vehicle speed and steering angle. A large number of input variables improve control but lead to a large fuzzy rules table. Because of that, in the design presented here, a tree of connected fuzzy inference systems (FIS) is applied. Fuzzy inference systems with a smaller number of variables are developed, algorithms explained, rule base defined, and obtained control surfaces presented. This approach requires less processing time enabling real time applications. Since the rules are defined based on drivers' experiences, fuzzy logic control systems make decisions in the same way as humans do, i.e., as experience drivers. This paper gives a comprehensive presentation of a novel cascaded fuzzy system development. This novel design also involves algebraic subtraction performed through a FIS subsystem.

1. INTRODUCTION

Fuzzy logic is a relatively new way of thinking and application development compared to well-known Boolean logic. It is established in the last century, by Lotfi A. Zadeh (Zadeh, 1965). That is 118 years after George Boole published, for the first time, his book *The Mathematical Analyses of Logic,* reprinted later (Boole, 2009). Boolean algebra is used to express and analyse operations of logic gates. Claude Shannon first applied Boolean algebra to analyse and design logic circuits (Shannon, 1938).

While in Boolean algebra, the values of variables could be just *true*, or *false*, expressed as integer values of 1 or 0, in fuzzy logic, values of variables are real numbers ranging between 0 and 1, including them. This enables fuzzy logic to be applied in control system design when input information is unreliable, or there is lack of certainty.

We could say that Boolean algebra deals with symbol manipulations and exact reasoning, leading to probability. We can quantify probability, i.e., assign a value. Further to that, Shannon has defined information as negative log of probability as given by Equation (1).

$$Information = -\log(p) \tag{1}$$

There is a difference between probability and possibility. Probability of an event, p, means that something may happen, and we believe that it is more likely than not. It can be quantified like p=0, 0.2, 0.5, …, 1. If we look at the system that can generate just 2 events, then in Equation (1) base of the log is 2. If events have the same probability of appearance, i.e., p=0.5, then the information carried by a single event is equal to 1bit.

On the other side, possibility means that something may happen, but we do not know how likely. Fuzzy logic applies symbol manipulations and numerical computations to come to approximate reasoning. It essentially deals with possibilities. Everything is a matter of degree. Fuzzy logic variables are represented by sets of values. That can be represented by mathematical formulations which give a degree of membership within the set. We can have unions and intersections of fuzzy sets. As humans, we make fuzzy logic decisions. Artificial Intelligence (AI) is effectively an application of fuzzy logic. Fuzzy logic is applied in many control engineering systems, when other strategies are not powerful enough to deal with uncertainties and unreliable data (Carter, Chiclana, Khuman, & Chen, 2021; Matía, Marichal, & Jiménez, 2014). It is also applied in business decision making, as shown in numerous references, like here (Bezděk, 2014) and here (M. Todorovic & M. Simic, 2019) in the process of *Transition to Electrical Vehicles Based on Multi-Attribute Decision Making. Managing Transition to Autonomous Vehicles Using Bayesian Fuzzy Logic* is also investigated and reported (Todorovic & Simic, 2019b).

At RMIT University, School of Engineering, research in autonomous vehicles (Elbanhawai, Simic, & Jazar, 2015; Elbanhawi & Simic, 2014; Elbanhawi, Simic, & Jazar, 2015a, 2015b; Elbanhawi, Simic, & Jazar, 2015; Elbanhawi, Simic, & Jazar, 2016; Elbanhawi, Simic, & Jazar, 2018), as well as, research in the process of new technology introduction (Aldakkhelallah, Todorovic, & Simic, 2021; Todorovic & Simic, 2019a, 2019b; Todorovic & Simic, 2019; Todorovic, Simic, & Kumar, 2017), are conducted concurrently.

Driving a car is a good example of fuzzy logic decisions that we make as drivers. We must take care of the speed, acceleration, path conditions and curvatures, visibility, our vehicle performances, especially available power in critical situations and many other. When we design a fully autonomous vehicle (AV), knowledge and skills of the best drivers should be embedded into the vehicle control system. On the road, AV decisions must be made in the real time. All of this was the motivation for the research conducted and presented here. The main contribution of this paper is a comprehensive report of an original cascaded fuzzy system development. Other publications available (Emmanuel, 2017), (Panse, Singh, & Satsangi, 2015), (Basjaruddin, Kuspriyanto, Saefudin, & Nugraha, 2014), although very valuable, do not give details and algorithms on defining safe distance and speed or speed error. This original design also involves algebraic subtraction performed through a FIS subsystem.

2. RELATED WORKS

Adaptive cruise control is one of the driver assistance features introduced on level 1 and 2 of SAE AV ranking. This can be achieved by various methods of control such as using Proportional-Integral-Derivative (PID) controller, state-space controller, fuzzy logic, (Osman, Rahmat, & Ahmad, 2009), Internal Model Control (IMC), neural networks or other, like coordinated throttle and brake control for ACC (Bala, Sadiq, Aibinu, & Folorunso, 2021). PID and IMC are not suitable for nonlinear operations on the road, such as aerodynamic drag and friction (Bala et al., 2021), or slippery road. All of those are effected by weather conditions. Further to that, it is shown that the fuzzy logic controller has faster response than neural network solutions and is less complex to implement (Panse et al., 2015).

Another recent publication, (Nchena, 2020) considers type of the road, i.e., downhill, and uphill driving. In this report on fuzzy logic controller (FLC) application in AV control, two types of controllers were investigated. It is shown that FLC outperforms a Proportional Integral (PI) controller. Since fuzzy logic can easily handle nonlinear conditions and shows superiority compared to other controller approaches, our investigation has proceeded in that direction. Large number of input variables are included in our novel design to enable better control. All stages and algorithms of the controller design are presented, what cannot be found so comprehensively given in existing works. The presented controller covers all road types and weather conditions.

Example of a generic fuzzy logic controller is shown in Figure 1.

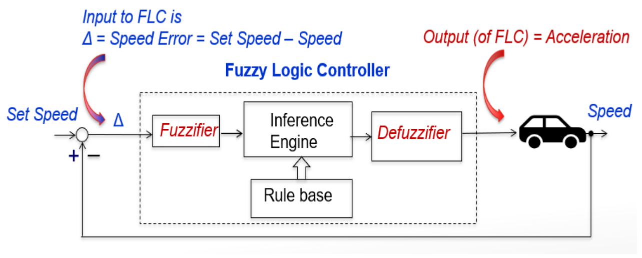


Figure 1: Generic fuzzy logic controller for cruise control

Basic components of this control approach, i.e., of fuzzy inference system, are presented. Fuzzifier block is converting an input variable into fuzzy sets through membership functions as shown later by Equations (5) and (6). Transfer of input variable values to outputs is performed using rules. Rule base, or database of rules, is defined based on human operator expertise, or drivers experience. It has numerous *if then* statements, as following

$$\boldsymbol{If}\ speed\ is\ below\ the\ set\ speed, \boldsymbol{then}\ speed\ up$$

Inference engine is applying fuzzy rules to the input variables, to generate fuzzy output. In the next step, defuzzification, output fuzzy set is converted to a crisp set. Center of gravity method is often used defuzzification method. When plain cruise control is implemented, the driver is responsible for maintaining the safe distance behind the car ahead. In our case, that function is performed by the fuzzy controller, which is now becoming more comprehensive.

3. ADAPTIVE CRUISE CONTROL SYSTEM

An adaptive cruise control, as a driver assistance feature, is monitoring and controlling the speed of our Cruise Car (CC) while simultaneously keeping the safe distance from the Car Ahead (CA). The speed of the CC is labeled as S_{CC} and the Speed of CA is S_{CA}. Safe distance is defined by the traffic authorities. In addition to that, it also depends on the vehicle speed, traffic, weather conditions and the driver, or driving mode. An additional variable, *Distance Error (DE)*, is introduced as the difference between *Measured Distance* and *Safe Distance,* as given by Equation (2).

$$DE = Distance - Safe\ Distance \qquad (2)$$

Different scenarios on the road are given in Figure 2.

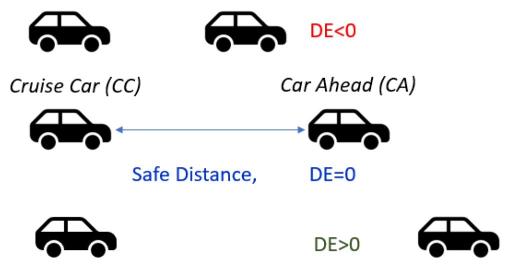


Figure 2: Defining distance error (DE) cases

We also need to define speed error, which is calculated as given by Equation (3).

$$SE = Speed_{Set} - Speed_{CC} \tag{3}$$

Considering distance error and speed error we can define a new rule such as the following:

***If** distance error is positive **and** speed error is positive, **then** accelerate*

The rule base consists of all possible rules, i.e., has responses for all scenarios that could appear on the road.

4. FUZZY LOGIC VARIABLES

According to SAE defined levels of driving automation, if adaptive cruise control, or lane centering, appears as driver support feature, the vehicle is level 1 automated. If both features are included, that vehicle is categorized as level 2. Manufacturers have different names for ACC and different control strategies are applied. For any control system we need to define input variables, control strategy, controller, and output variables. In this project, a cascaded fuzzy logic approach is applied in controller design.

In 1965 Zadeh has introduced *fuzzy set* as a class of objects with a continuum of membership grades (Zadeh, 1965). They are foundation of any logic regardless of truth levels assumed. For fuzzy variables we have continuum of logic levels between 0, associated to *false*, to 1 for completely *true*. We can label a space of points as *X*. Equation (4) defines an *x* as a generic element of space *X*.

$$x \in X \tag{4}$$

A *fuzzy set A* in the universe *X* is defined by a membership function $\mu_A(x)$, which associates point *x* to a real number in the interval [0, 1]. Value $\mu_A(x)$ represents *grade of membership* of *x* in *A*. As an illustration, membership functions' plots for the measured *Speed* input variable are shown in Figure 3.

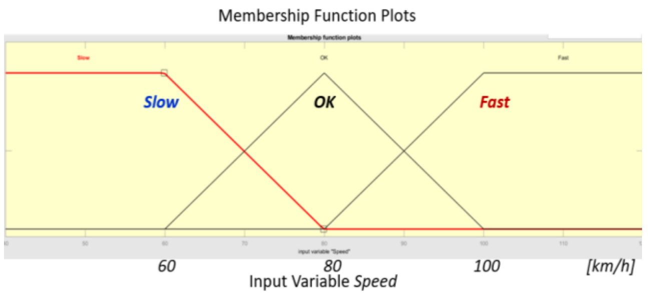


Figure 3: Membership functions' plots for the *Measured Speed* input variable

The most popular membership functions are Triangular, Trapezoidal, Piecewise linear, Gaussian and Singleton. Triangular membership function labeled as μ_{Tri} is given by Equation (5).

$$\mu_{Tri}[a,b,c](x) = \begin{cases} 0 & x < a \\ \frac{x-a}{b-a} & a \le x \le b \\ \frac{c-x}{c-b} & b < x \le c \\ 0 & x > c \end{cases} \tag{5}$$

As shown in Figure 3, our membership function *OK* is a triangular type, where braking points are $a = 60km/h$, $b = 80km/h$ and $c = 100km/h$.

Functions *Slow* and *Fast* are trapezoidal type μ_{Trap} and they are defined as given by Equation (6). In our case, both are particular cases of a generic trapezoidal function. For membership function *Slow*, we have $a = b = 0$, $c = 60km/h$ and $d = 80km/h$. For membership function *Fast*, $a = 80km/h$, $b = 100km/h$, and $c = d \rightarrow \infty$.

$$\mu_{Trap}[a,b,c,d](x) = \begin{cases} 0 & x < a \\ \frac{x-a}{b-a} & a \le x < b \\ 1 & b \le x \le c \\ \frac{d-x}{d-c} & c < x \le d \\ 0 & x > d \end{cases} \tag{6}$$

Our control system has six input variables and two outputs. Inputs, with associated membership functions in brackets, are given here:

- *Driving mode* (*Eco, Comfort, Sport*),
- *Weather* conditions (*Good, Bad*),
- Measured *speed* (*Slow, OK, High*),
- *Distance* from the vehicle ahead (*Big, OK, Too close*)
- *Speed error* (ranges from very negative to very positive, i.e., *NN, N, Z, P, PP*)
- *Steering angle* (ranging from very high, to moderate and zero, *PP, P, Z*).

Outputs of our control system, with associated membership functions in brackets, are as following:

- *Corrected acceleration* (from negative to zero and positive *NN, N, Z, P, PP*)
- *Braking* (from hard braking, *BB*, braking, *B*, to no braking, *Z*).

When the number of inputs to a fuzzy system is large than that requires large number of rules. In our case it is $3*2*3*3*5*3 = 810$ per output, i.e., 1620 in total for two outputs. Because of that complexity, controller is designed as a cascaded fuzzy inference system (FIS), i.e., it has four interconnected FIS subsystems inside.

5. FUZZY CONTROL SYSTEM STRUCTURE

In order to overcome the need for a large number of rules, fuzzy inference system is implemented as a hierarchical tree of smaller interconnected subsystems. Following that, block diagram of the fuzzy logic based adaptive cruise control is shown in Figure 4. It is an incremental structure.

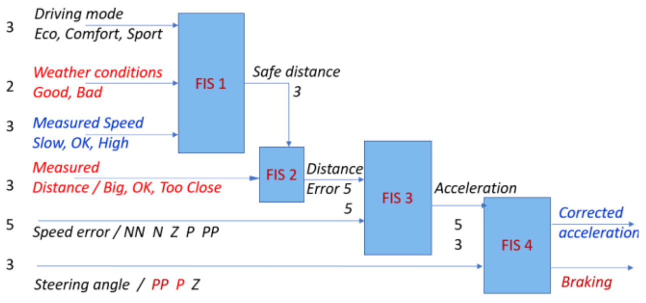


Figure 4: Incremental structure of the fuzzy control system. Number of membership functions for each input is given and all input / output variables are labelled

Each FIS has its own inputs and outputs, where outputs of the lower-level fuzzy systems are used as inputs to the higher-level fuzzy systems.

As shown, there are four interconnected levels. Fuzzy tree structure approach is more computationally efficient and easier to follow than a single FIS with the same number of inputs.

Level 1 in the tree has 3 inputs with $3 + 2 + 3 = 8$ membership functions in total. This gives number of rules $3 * 2 * 3 = 18$.

In level 2 we have 2 inputs, both with 3 membership functions. It creates 9 fuzzy rules.

On level 3 we have 2 inputs with 5 membership functions each, which leads to 25 rules.

Finally, on the level 4 we have 2 inputs, with 5 and 3 functions, creating 15 rules, for each output, i.e., 30 rules for the final, output level FIS.

The total number of rules is now $(18 + 9 + 25 + 2 * 15) = 82$, compared with 1620 which we would have if we did not use hierarchical tree structure.

6. FIS SYBSYSTEMS DESIGN

The following section presents design steps for each of the FIS subsystems included in the cascaded controller.

A. Level One FIS Design

Inputs at the first level in the tree, with associated membership functions in brackets, are given here:

- *Driving Mode* (*Eco, Comfort, Sport*)
- *Weather* conditions (*Good, Bad*)
- Measured *Speed* (*Slow, OK, High*)

Membership functions for variable *Speed* are already shown in Figure 3. Membership functions for *Weather* variable are given in Figure 5. It is a dimensionless variable displayed in the range 0 to 10, on x axes. Value 0 corresponds to bad weather, while 10 corresponds to good weather. In this initial stage we have defined just two membership functions, but later, if needed, we could include more to better cover the real weather conditions on the road.

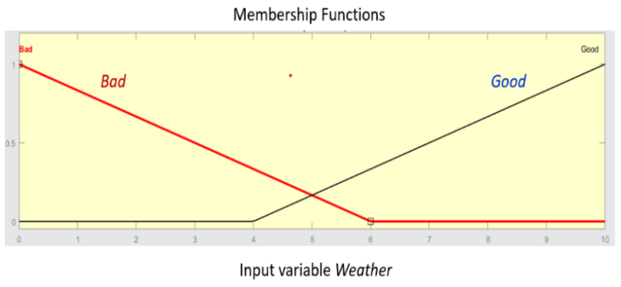


Figure 5: Membership functions for input variable *Weather*

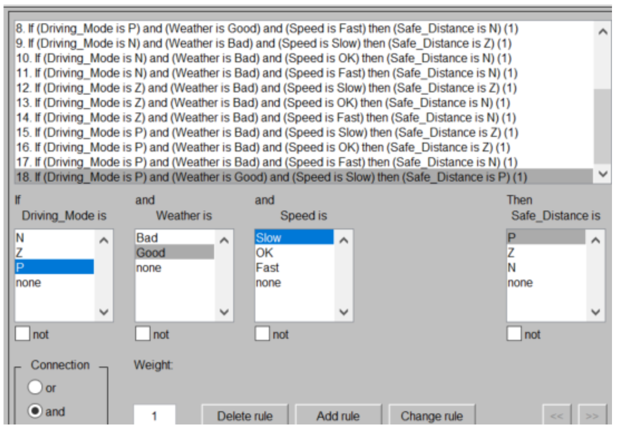
Our *Weather,* as an input variable here, could be an output variable from another fuzzy inference system that should consider onboard sensors' readings of temperature and detections of snow, rain, fog, and strong winds. This could also be a task for onboard Artificial Intelligence (AI).

Inference engine is using the rule base to define output, i.e., *Safe Distance* in this case. Rule base is shown in Figure 6 – which is a screen capture from the MATLAB Rule Editor. There are 18 rules because there are 3 membership functions for *Driving Mode*, 2 for *Weather* and 3 for the measured *Speed*.

FIS responses can be seen through the control surface. Since we have 3 input variables and one output, it makes 4 dimensions. That could not be presented on a single 3D graph. Following that, MATLAB can visualize three control surfaces

- S_1 = *Safe Distance (Speed, Weather)*
- S_2 = *Safe Distance (Driving mode, Speed)*
- S_3 = *Safe Distance (Weather, Driving mode)*

We are presenting here just one, but all of them are available for investigation. The first one is selected because drivers are usually selecting driving mode and do not change it that often. At the same time *Speed* and *Weather* are extremely important inputs, i.e., variables for real time processing and autonomous driving.


```
8. If (Driving_Mode is P) and (Weather is Good) and (Speed is Fast) then (Safe_Distance is N) (1)
9. If (Driving_Mode is N) and (Weather is Bad) and (Speed is Slow) then (Safe_Distance is Z) (1)
10. If (Driving_Mode is N) and (Weather is Bad) and (Speed is OK) then (Safe_Distance is N) (1)
11. If (Driving_Mode is N) and (Weather is Bad) and (Speed is Fast) then (Safe_Distance is N) (1)
12. If (Driving_Mode is Z) and (Weather is Bad) and (Speed is Slow) then (Safe_Distance is Z) (1)
13. If (Driving_Mode is Z) and (Weather is Bad) and (Speed is OK) then (Safe_Distance is N) (1)
14. If (Driving_Mode is Z) and (Weather is Bad) and (Speed is Fast) then (Safe_Distance is N) (1)
15. If (Driving_Mode is P) and (Weather is Bad) and (Speed is Slow) then (Safe_Distance is Z) (1)
16. If (Driving_Mode is P) and (Weather is Bad) and (Speed is OK) then (Safe_Distance is Z) (1)
17. If (Driving_Mode is P) and (Weather is Bad) and (Speed is Fast) then (Safe_Distance is N) (1)
18. If (Driving_Mode is P) and (Weather is Good) and (Speed is Slow) then (Safe_Distance is P) (1)
```

If Driving_Mode is	and Weather is	and Speed is	Then Safe_Distance is
N	Bad	Slow	P
Z	Good	OK	Z
P	none	Fast	N
none		none	none
not	not	not	not

Connection: or [✓ and]   Weight: 1

Figure 6: Fuzzy rules for variable Safe Distance as seen in MATLAB Rule Editor

Figure 7 shows S_1 control surface. We can see that the *Safe distance* is larger when the *Speed* is greater, and the *Weather* is Bad.

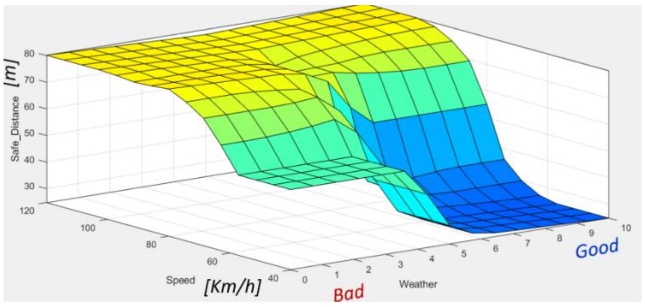


Figure 7: S_1 control surface for *Safe Distance*

B. Level Two FIS Design

Inputs for the second FIS level in the tree, with associated membership functions in brackets, are given here:

- *Safe Distance* (*P(Low), Z(OK), N(Higher)*),
- Measured *Distance* (*Big, OK, Too Close*)

Safe Distance comes out of the FIS at the level 1, as just shown. *Measured Distance* comes from the distance sensor readings (Radar or LIDAR). Control surface is shown in Figure 8. Output variable is *Distance Error (DE),* which has 5 membership functions. Visual presentation of the DE meaning is already given in the Figure 2. *Distance error* and *Speed error,* with five membership functions as well, are two key control variables in the feedback control loop. This, the most important logic, is realized on the next FIS level.

As an illustration of the functionality for this stage, let us analyze a scenario when the safe distance if 70m and our measurement shows 70m as well. Distance error is 0 as it can be seen from the location of the red point P on the surface.

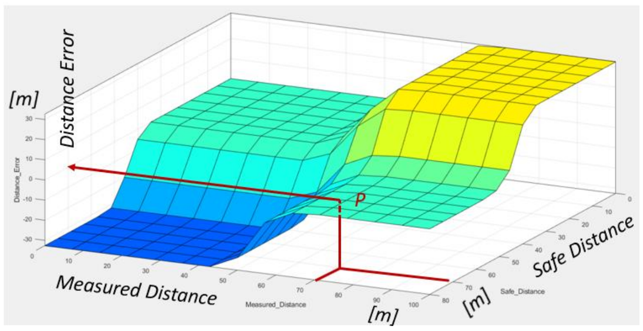


Figure 8: Control surface for FIS 2 i.e., *Distance Error*

Because FIS 2 output variable *Distance Error* is input variable for next level FIS 3, in order to follow the logic, this variable is shown in Figure 9 with its 5 membership functions.

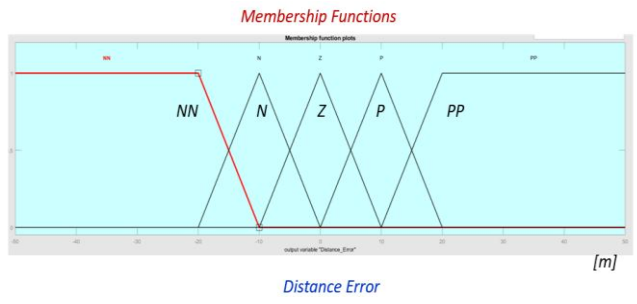


Figure 9: *Distance Error* variable membership functions

C. Level Three FIS Design

Inputs at the third level in the tree have five membership functions each. They are given here with associated membership functions in brackets:

- *Distance Error* (ranges from very negative to very positive, i.e., *NN, N, Z, P, PP*), as shown in Figure 9
- *Speed error* (ranges from very negative to very positive, i.e., *NN, N, Z, P, PP*)

Speed Error is calculated by Equation (3), as a difference between set speed and instant cruising speed, outside of this controller structure, and then fuzzified. These variable membership functions are shown in the Figure 10.

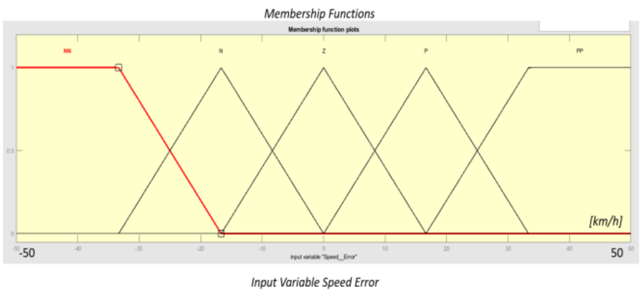
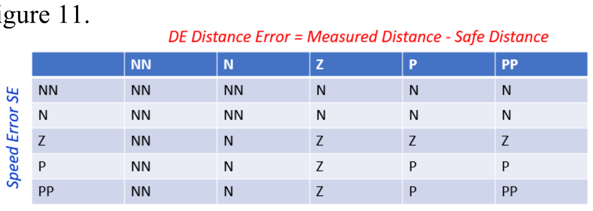


Figure 10: Input variable *Speed Error* membership functions

Since we have 5 functions for each variable, the total number of rules is 25. For easier tracking of rule base design, it is initially created in Excel and then transferred to MATLAB by Rule Editor. Excel rule base is given in Figure 11.



DE Distance Error = Measured Distance - Safe Distance

Speed Error SE

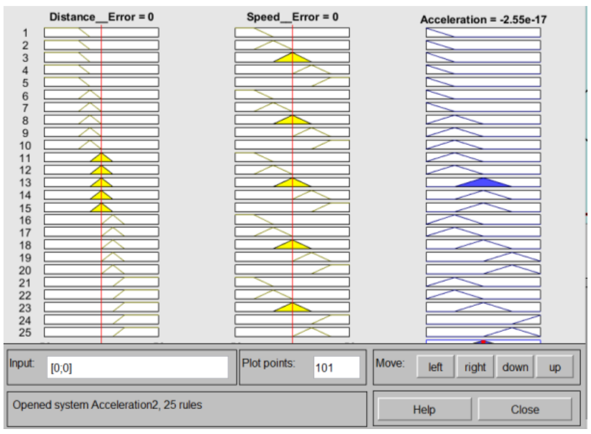
	NN	N	Z	P	PP
NN	NN	NN	N	N	N
N	NN	NN	N	N	N
Z	NN	N	Z	Z	Z
P	NN	N	Z	P	P
PP	NN	N	Z	P	PP

Figure 11: Whole rule base for FIS 3 *Acceleration*

For example, from the rule base shown in Figure 11 we crate rules as following.

***If** SE is NN, and DE is NN **then** Acceleration is NN, …,*
***If** SE is P, and DE is Z **then** Acceleration is Z,*

When the rules are in the system, using MATLAB Rule Viewer, we could see and verify all of them. Rule View for FIS 3 is shown in the Figure 12.



Distance__Error = 0 Speed__Error = 0 Acceleration = -2.55e-17

Input: [0;0] Plot points: 101

Opened system Acceleration2, 25 rules

Figure 12: Rule View for all FIS 3 i.e., for output variable *Acceleration*

When this is all set up properly, we can see the control surface for *Acceleration* as shown in Figure 13.

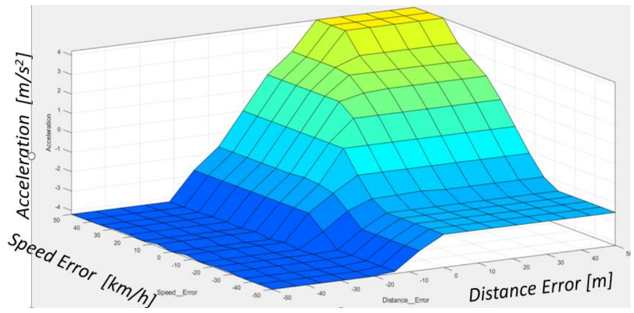


Figure 13: Control surface for *Acceleration*

Both input variables for FIS3, could be positive, zero or negative. As we can see from the Figure 13, *Acceleration,* as output, also could be positive, to speed up, and negative, when needed to slow down. For higher distance error or higher speed error we have higher acceleration. There should be a limit to acceleration and that is defined by the steering angle. The correction of the acceleration, as well as braking, is introduced by the last FIS4 in the tree structure.

D. Level Four FIS Design

Inputs to the fourth level FIS in the tree, are given here with associated membership functions in brackets:

- *Acceleration* (ranges from very negative to very positive, i.e., *NN, N, Z, P, PP*)
- *Steering Angle* (ranges from zero, positive, to very positive, i.e., *Z, P, PP*)

From this FIS we have two output functions:

- *Corrected Acceleration* (ranges from very negative to very positive, i.e., *NN, N, Z, P, PP*)
- *Braking* (ranges from zero to very positive, i.e., *Z, P, PP*)

Acceleration is output variable from FIS 3 and input for FIS 4. Its values, as FIS output, can be seen in the Figure 13. Membership functions are shown in the Figure 14.

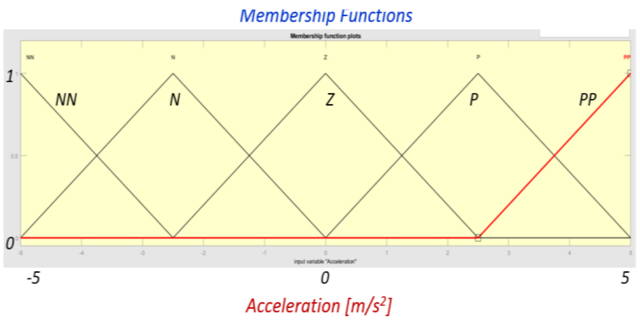


Figure 14: FIS 4 Input variable *Acceleration* membership functions

When defining the range for this variable we have taken values that could be found for a relatively powerful vehicles available to the market, like up to 5m/sec^2. For the *Steering Angle* vales are also relatively widely used, in the range of -30^0 to +30^0, expressed in degrees. In defining the rule base, we have considered absolute values because from the point of vehicle dynamics constraints, it does not matter if we have to turn left, or right.

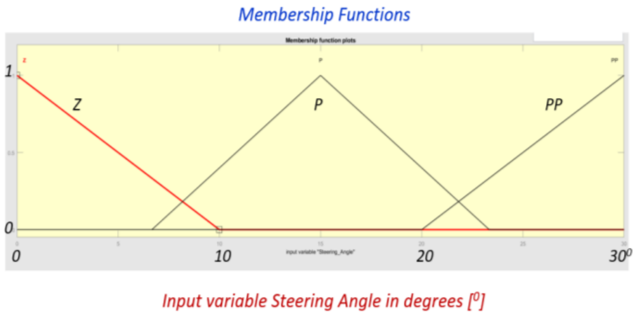


Figure 15: FIS 4 Input variable *Steering Angle* membership functions, shown as absolute value of real angle

Total number of rules is $2 * (3 * 5) = 30$, or 15 integrated rules, for two outputs, as in MATLAB Rule Editor. All rules, i.e., whole rule base, are shown in Figure 16. Based on that, we have control surfaces as shown in Figure 17 and Figure 18. Comparing Figure 13 and Figure 17 we can see the difference, i.e., acceleration corrections. If the path curvature is too large, we, as drivers, must adjust acceleration, or deceleration, accordingly.

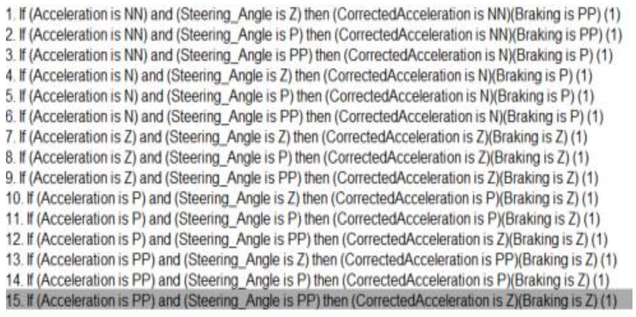

1. If (Acceleration is NN) and (Steering_Angle is Z) then (CorrectedAcceleration is NN)(Braking is PP) (1)
2. If (Acceleration is NN) and (Steering_Angle is P) then (CorrectedAcceleration is NN)(Braking is PP) (1)
3. If (Acceleration is NN) and (Steering_Angle is PP) then (CorrectedAcceleration is N)(Braking is P) (1)
4. If (Acceleration is N) and (Steering_Angle is Z) then (CorrectedAcceleration is N)(Braking is P) (1)
5. If (Acceleration is N) and (Steering_Angle is P) then (CorrectedAcceleration is N)(Braking is P) (1)
6. If (Acceleration is N) and (Steering_Angle is PP) then (CorrectedAcceleration is N)(Braking is P) (1)
7. If (Acceleration is Z) and (Steering_Angle is Z) then (CorrectedAcceleration is Z)(Braking is Z) (1)
8. If (Acceleration is Z) and (Steering_Angle is P) then (CorrectedAcceleration is Z)(Braking is Z) (1)
9. If (Acceleration is Z) and (Steering_Angle is PP) then (CorrectedAcceleration is Z)(Braking is Z) (1)
10. If (Acceleration is P) and (Steering_Angle is Z) then (CorrectedAcceleration is P)(Braking is Z) (1)
11. If (Acceleration is P) and (Steering_Angle is P) then (CorrectedAcceleration is P)(Braking is Z) (1)
12. If (Acceleration is P) and (Steering_Angle is PP) then (CorrectedAcceleration is Z)(Braking is Z) (1)
13. If (Acceleration is PP) and (Steering_Angle is Z) then (CorrectedAcceleration is PP)(Braking is Z) (1)
14. If (Acceleration is PP) and (Steering_Angle is P) then (CorrectedAcceleration is P)(Braking is Z) (1)
15. If (Acceleration is PP) and (Steering_Angle is PP) then (CorrectedAcceleration is Z)(Braking is Z) (1)

Figure 16: FIS 4 rule base for output variables *Corrected Acceleration* and *Braking*

For example, if the angle is extreme, such as 30^0, *Corrected Acceleration* is 0. If *Steering Angle* is 0 then there is no correction as shown clearly in Figure 17.

From Figure 18 we can see that there is no braking while *Acceleration* is positive, and that it increases as deceleration increases.

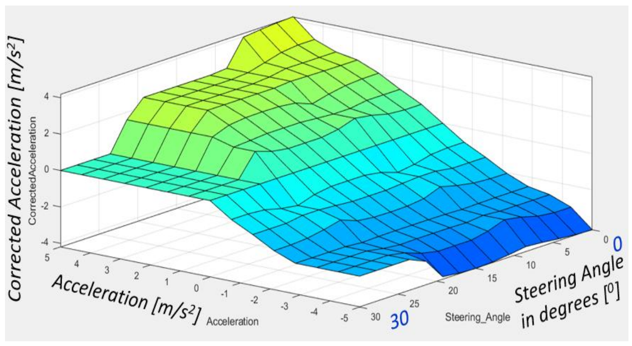


Figure 17: FIS 4 control surface for output variable *Corrected Acceleration*

Intensity of braking is defined in the range from 0 to 10, but for the particular car it will be converted to the braking force intensity in the range of 0 to *xx* N.

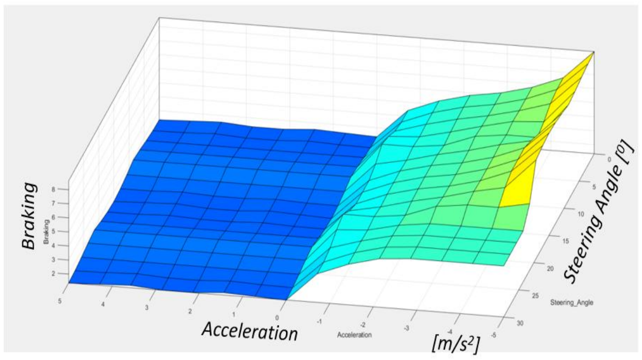


Figure 18: FIS 4 control surface for output variable *Braking*

In summary, membership functions, fuzzy rules and control surfaces for all control levels are presented. The system has 6 inputs, with 19 membership functions and two outputs with a total of 8 functions. When the number of inputs, and membership functions increase the number of rules increase as a product of the number of functions per input.

Compared to single fuzzy inference system, not cascaded, which must have 1620 rules, the presented design has just 82 rules. With 6 input variables, through 3 cascaded levels, the presented design enables better control, more akin to human operator's actions. Other publications do not give as much detail in the FIS subsystem design. Additionally, instead of calculating *Distance Error* with Boolean logic, as shown in other research reports (Basjaruddin et al., 2014), (Chen, Zhang, & Liu, 2016), in our controller, the output variable from FIS 2, derived through fuzzy logic, is further appearing as input variable on level 3 FIS. The same could be conducted for the *Speed Error* but was not presented here for the simplicity of design and presentation.

7. CONCLUSIONS

Design of a cascaded, incremental fuzzy control system is presented. At the beginning, the basics of two logic systems are explained: Boolean and Zadeh's Fuzzy logic. They both have their application areas and deal with uncertainties through handling probability, or possibility in the systems' design and control. Fuzzy logic and Artificial Intelligence are now widely used in all types of engineering systems, in business, medicine and other areas.

Adaptive cruise control is one of the driver assistance features. Car manufacturers have different controller solutions and names for this active safety system. The presented cascaded, incremental fuzzy logic controller uses data collected from all available vehicle onboard sensors. Membership functions, fuzzy rules and control surfaces for all control levels are presented. When the number of inputs, and membership functions increase, the number of rules increases as a product of functions per input. A large number of rules increase computation time and is not suitable for real time processing. The presented approach using a fuzzy tree is simplified and uses an order of magnitude less fuzzy rules.

In the further research, both distance and speed errors will be derived through fuzzy logic, and then the complete vehicle dynamic system modeling will be conducted. Through the model testing, fine tuning of the fuzzy logic controller will be conducted, the same as we improve our driving through experience.

The final step is to go from MATLAB code, to hardware description language (HDL) code, and then to field-programmable gate array (FPGA) that could be used for field testing.

ACKNOWLEDGEMENTS

The author would like to express his gratitude to the RMIT University, Australia. He also would like to thank the Editors and anonymous reviewers of the article for their insightful suggestions and comments to improve the content of the manuscript.

REFERENCES

Aldakkhelallah, A., Todorovic, M., & Simic, M. (2021). *Investigation in Introduction of Autonomous Vehicles.* Paper presented at the IASTEM - 1082nd International Conference on Control, Automation, Robotics and Vision Engineering(ICCARVE), Riyadh, Saudi Arabia.

Bala, J. A., Sadiq, T., Aibinu, A. M., & Folorunso, T. A. (2021, 15-16 July 2021). *A Fuzzy Super Twisting Sliding Mode Control Scheme for Velocity Regulation in Autonomous Vehicles.* Paper presented at the 2021 1st International Conference on Multidisciplinary Engineering and Applied Science (ICMEAS).

Basjaruddin, N. C., Kuspriyanto, K., Saefudin, D., & Nugraha, I. K. (2014). Developing Adaptive Cruise Control Based on Fuzzy Logic Using Hardware Simulation. *International journal of electrical and computer engineering (Malacca, Malacca), 4*(6), 944. doi:10.11591/ijece.v4i6.6734

Bezděk, V. (2014). Using Fuzzy Logic in Business. *Procedia - Social and Behavioral Sciences, 124*, 371-380. doi:https://doi.org/10.1016/j.sbspro.2014.02.498

Boole, G. (2009). *The Mathematical Analysis of Logic: Being an Essay Towards a Calculus of Deductive Reasoning*. Cambridge: Cambridge University Press.

Carter, J., Chiclana, F., Khuman, A. S., & Chen, T. (2021). *Fuzzy Logic: Recent Applications and Developments*. Cham: Springer International Publishing AG.

Chen, X.-w., Zhang, J.-g., & Liu, Y.-j. (2016). Research on the Intelligent Control and Simulation of Automobile Cruise System Based on Fuzzy System. *Mathematical problems in engineering, 2016*, 1-12. doi:10.1155/2016/9760653

Elbanhawai, M., Simic, M., & Jazar, R. N. (2015). Improved Manoeuvring of Autonomous Passenger Vehicles: Simulations and Field Results. *Journal of Vibration and Control*, 35. doi:10.1177/1077546315605666

Elbanhawi, M., & Simic, M. (2014). Sampling-Based Robot Motion Planning: A Review. *IEEE Access, 2*, 56-77. doi:10.1109/ACCESS.2014.2302442

Elbanhawi, M., Simic, M., & Jazar, R. (2015a). In the Passenger Seat: Investigating Ride Comfort Measures in Autonomous Cars. *IEEE Intelligent Transportation Systems Magazine, 7*(3), 4-17. doi:10.1109/MITS.2015.2405571

Elbanhawi, M., Simic, M., & Jazar, R. (2015b). Randomized Bidirectional B-Spline Parameterization Motion Planning.

Intelligent Transportation Systems, IEEE Transactions on, PP(99), 1-1. doi:10.1109/TITS.2015.2477355

Elbanhawi, M., Simic, M., & Jazar, R. (2015). The Role of Path Continuity in Lateral Vehicle Control. *Procedia Computer Science, 60*, 1289-1298. doi:http://dx.doi.org/10.1016/j.procs.2015.08.194

Elbanhawi, M., Simic, M., & Jazar, R. (2016). Solutions for Path Planning Using Spline Parameterization. In N. R. Jazar & L. Dai (Eds.), *Nonlinear Approaches in Engineering Applications: Advanced Analysis of Vehicle Related Technologies* (pp. 369-399). Cham: Springer International Publishing.

Elbanhawi, M., Simic, M., & Jazar, R. (2018). Receding horizon lateral vehicle control for pure pursuit path tracking. *Journal of Vibration and Control, 24*(3), 619-642. doi:10.1177/1077546316646906

Emmanuel, I. (2017). Fuzzy Logic-Based Control for Autonomous Vehicle: A Survey. *International Journal of Education and Management Engineering, 7*(2), 41-49. doi:10.5815/ijeme.2017.02.05

Matía, F., Marichal, G. N. s., & Jiménez, E. (2014). *Fuzzy modeling and control : theory and applications* (1st ed. 2014. ed.). Paris, France]: Atlantis Press.

Nchena, L. (2020, 16-18 Sept. 2020). *Fuzzy Logic Application in Automation Control.* Paper presented at the 2020 10th International Conference on Advanced Computer Information Technologies (ACIT).

Osman, K., Rahmat, M. F., & Ahmad, M. A. (2009, 6-8 March 2009). *Modelling and controller design for a cruise control system.* Paper presented at the 2009 5th International Colloquium on Signal Processing & Its Applications.

Panse, P., Singh, A., & Satsangi, C. (2015). Adaptive Cruise Control using Fuzzy Logic. *International Journal of Digital Application & Contemporary research, 3*, 7.

Shannon, C. (1938). *Symbolic Analysis of Relay and Switching Circuits.* (Masters). MIT, Boston USA. Retrieved from https://dspace.mit.edu/handle/1721.1/11173

Todorovic, M., & Simic, M. (2019a, 2019//). *Current State of the Transition to Electrical Vehicles.* Paper presented at the Intelligent Interactive Multimedia Systems and Services, Cham.

Todorovic, M., & Simic, M. (2019b, 2019//). *Managing Transition to Autonomous Vehicles Using Bayesian Fuzzy Logic.* Paper presented at the Innovation in Medicine and Healthcare Systems, and Multimedia, Singapore.

Todorovic, M., & Simic, M. (2019, 13-15 Feb. 2019). *Transition to Electrical Vehicles Based on Multi-Attribute Decision Making.* Paper presented at the 2019 IEEE International Conference on Industrial Technology (ICIT).

Todorovic, M., Simic, M., & Kumar, A. (2017). Managing Transition to Electrical and Autonomous Vehicles. *Procedia Computer Science, 112*, 2335-2344. doi:https://doi.org/10.1016/j.procs.2017.08.201

Zadeh, L. A. (1965). Fuzzy sets. *Information and Control, 8*(3), 338-353. doi:https://doi.org/10.1016/S0019-9958(65)90241-X