JJEE Volume 10 | Number 1 | March 2024 | Pages 27-47

Jordan Journal of Electrical Engineering

ISSN (print): 2409-9600, ISSN (online): 2409-9619

Homepage: jjee.ttu.edu.jo

Comparison of Fuzzy Logic Control and Model Predictive Control for a Smart Adaptive Cruise Control Vehicle System

Adnan K. Shaout[1]*, Syed Adil Ahmad[2], Dan Osborn[3]

[1,2,3]Electrical and Computer Engineering Department, College of Engineering and Computer Science,
The University of Michigan - Dearborn, Dearborn, Michigan, USA
E-mail: shaout@umich.edu

Received: June 26, 2023 *Revised: August 11, 2023* *Accepted: August 17, 2023*

***Abstract*—** Adaptive cruise control (ACC), cruise control (CC), and automatic emergency braking (AEB) serve as the basis of longitudinal automated driving, and as such have been the subject of much research. Model predictive control (MPC) and fuzzy logic are often considered to be the next steps in improving the capability of these systems, but the two control strategies have not been compared to each other in the ACC, CC and AEB applications. Also, the three features (ACC, CC and AEB) have never been compiled into a single fuzzy logic controller. The purpose of this paper is to design a fuzzy logic-based ACC, CC, and AEB controller and compare it to an equivalent MPC controller. All three controllers control the desired longitudinal acceleration, and their functionality is tested using Matlab's Fuzzy Logic Designer and other Simulink toolboxes. Ultimately, the results of the analysis demonstrate that the proposed fuzzy controller operates just as well if not better than the MPC controller and that the fuzzy controller is able to operate well in all tested scenarios.

***Keywords*—** Adaptive cruise control; Cruise control; Automatic emergency braking; Model predictive control; Fuzzy logic controllers; Matlab; Simulink.

1. INTRODUCTION

Adaptive Cruise Control (ACC) is a common feature in new cars that vastly improves on the existing Cruise Control (CC) system. When a vehicle is in front of the user's car, the ACC maintains a correct headway distance between the two vehicles by altering either the brake pressure and throttle angle, or the wheel torques. When a vehicle is not in front of the user's car, the ACC system operates as a basic CC system. The correct headway distance is innately a 'fuzzy' variable as the driver makes numerous snap decisions to decide how close is too close to the vehicle ahead of them. As such the function is well-suited to be implemented using a fuzzy controller. ACC has been a popular field of research and as such has had a wide range of research. The ACC system is generally split into a higher level, supervisory, controller and a lower level, actuator, controller. Generally, the higher-level system takes in the measurement data of the vehicle in front and commands the lower controller to actuate in such a way, either through throttle angle and brake pressure or wheel torque, to keep the appropriate headway distance between the lead vehicle and ego vehicle. The bulk of research in ACC is focused on researching different control methodologies to improve the functioning of the higher-level controller.

There has been research published that looks to switch ACC modes depending on the situation the driver is in [1], create multi-loop switch strategies for the ACC system [2], and some researchers have even gone so far as to attempt to control the time gap between vehicles

* Corresponding author

rather than the distance [3]. As time has gone by higher level controller ACC research has been completed, MPC and fuzzy logic have become front runners to be used in the next generation of ACC systems. There has been MPC ACC research that looks to take fuel consumption into consideration when the ACC is operating [4] and there has been research that looks to tune the weights within the model in real time to improve performance [5].

On the fuzzy logic side there has been research published that discusses creating fuzzy if-then rules for ACC systems [6], research that looks to smooth out the transitions between ACC and CC modes [7], research that looks to guarantee string stability among many vehicles using ACC all in the same lane [8], and research has been done looking into neural network's ability to approximate the behavior of the leading vehicle [9, 10].

From our literature review before starting this paper we noticed there had not been much research comparing fuzzy logic ACC systems to MPC systems. As they seem to be the leading methodologies in the ACC field, we believe that comparing the performance of the systems to be a worthwhile addition to the current state of research. As such three scenarios were created to test the capability of both the fuzzy ACC systems and the MPC ACC systems. The three scenarios are as follows: following a lead vehicle with varying levels of acceleration, a cut in scenario, and a suddenly stopping lead vehicle. The proposed scenarios should cover a wide range of ACC operating conditions and test whether or not the controllers are capable of operating correctly during them. In creating the proposed fuzzy logic controller, we initially based the controller off the fuzzy ACC system within [8] and then improved upon it to get better performance. In our improvement we added a set of if-then rules to allow the controller to operate in an Automatic Emergency Braking (AEB) mode.

This paper is organized into sections. In section 2, we talk about the controller methodology, or the basic layout and parameters of our controller setup. In section 3, we talk about the controller design and plant dynamics. In section 4, we talk about implementation of controllers and the plant in Simulink. In section 5, we discuss the simulations and analyze them. In section 6 we give a conclusion and comment on future work.

2. CONTROLLER METHODOLOGY

ACC is a feature commonly available in new vehicles that allows the vehicle to maintain a safe follow distance between the lead car and the ego car with minimal input from the driver. The ego vehicle, or following vehicle, takes in a measurement of the relative distance between the lead vehicle and the ego vehicle typically using LIDAR. The system generally takes in a relative distance measurement and a relative velocity reading and outputs a desired acceleration to allow the vehicle to maintain a correct headway distance. The requirements for a successful ACC system allow the vehicle to always maintain a correct headway distance as well as never exceed the set desired speed of the vehicle. Fig. 1 shows the basic functioning of the ACC vehicle. The general ACC system functions are based on the following equations:

$$\text{Relative Distance: } d(t) = d_{lead}(t) - d_{ego}(t) \quad (1)$$

$$\text{Relative Velocity: } v(t) = v_{lead}(t) - v_{ego}(t) \quad (2)$$

$$\Delta d(t) = d(t) - d_{des}(t) \quad (3)$$

$$\Delta v(t) = v(t) - v_{des}(t) \quad (4)$$

$$d_{des}(t) = T_h v_{ego}(t) + d_o = T_h v_{ego}(t) + 2 \quad (5)$$

where d(t) is the relative distance and Δd(t) is the difference between the relative distance and the destination distance. The system functions are based on the relative distance and the relative velocity. The relative distance equation allows the system to gauge whether or not the ego vehicle is at a safe distance or not and the relative velocity equation allows the system to monitor the change in the headway distance. The whole system is predicated off the correct headway distance which is a function of the headway time ($T_h v_{ego}(t)$), 2 s for our application (d_0=2), two times the mass of the vehicle, and the speed of the ego vehicle.

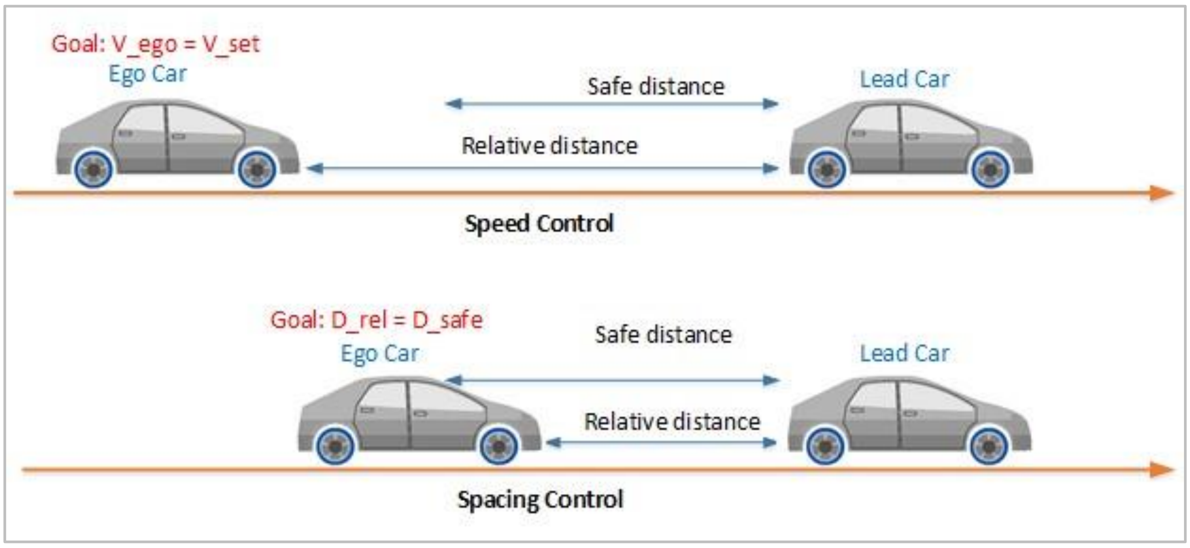


Fig. 1. Generalized function of ACC [11].

Generally, the ACC system is constructed of two different controllers, an upper-level controller and a lower-level controller as shown in Fig. 2. The upper-level controller is the system that detects the distance and relative speed between the vehicles. This upper-level controller sends a desired acceleration command to the lower-level controller to enact a wheel torque in order to satisfy the correct headway distance. The lower-level controller converts the desired acceleration input into a wheel torque using feed forward terms. Both these controllers working in tandem allow the system to operate correctly.

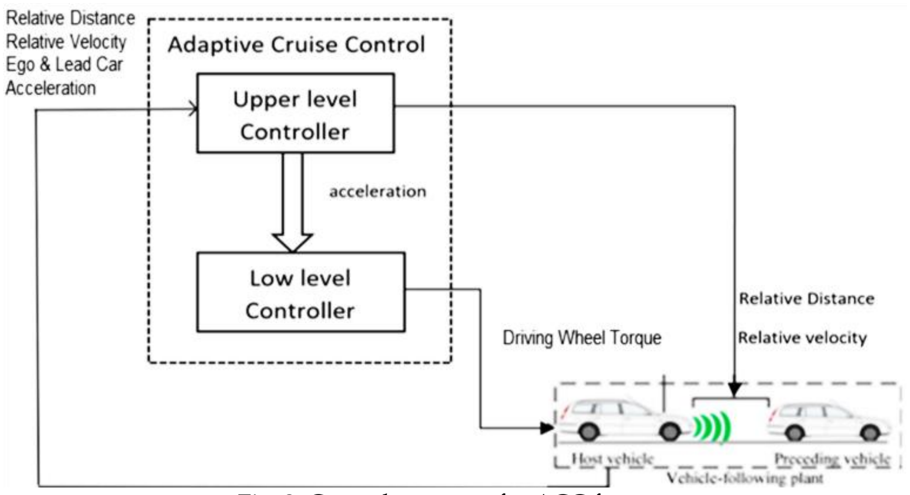


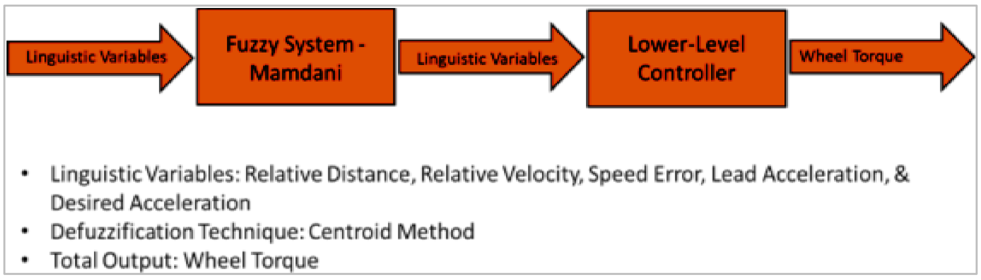
Fig. 2. Control structure for ACC feature.

3. CONTROLLER DESIGN AND PLANT DYNAMICS

The proposed controller is made of the following components: ACC Fuzzy Logic Based Controllers, ACC MPC Controller, Plant Model, and Lower-Level Controller.

3.1. ACC Fuzzy Logic Based Controllers

The fuzzy logic controller can be broken down into two main sections with three sets of data passing through it. The first main section is the fuzzy logic system. This system functions as the upper-level controller and it takes in a set of input linguistic variables and outputs a set of linguistic variables. The set of input linguistic variables was relative velocity, speed error, and relative distance for the basic fuzzy logic controller. The relative velocity, speed error, relative distance, and lead acceleration are the set of linguistic input variables for the improved fuzzy logic controller. The fuzzy logic controller takes those variables in, assigns them to output variable states based on the if-then rules and outputs a set of output linguistic variables based on the input. For both the basic and improved fuzzy logic controllers the fuzzy system uses the Mamdani method to computer the fuzzy output and the centroid method for defuzzification. Ultimately, the fuzzy system would output a set of data in the form of the only output linguistic variable in the system, desired acceleration. This would be sent to the lower-level controller as a defuzzified command signal that would enter the controller and be sent out that controller as a wheel torque command. The entire system is shown in the flowchart in Fig. 3.



- Linguistic Variables: Relative Distance, Relative Velocity, Speed Error, Lead Acceleration, & Desired Acceleration
- Defuzzification Technique: Centroid Method
- Total Output: Wheel Torque

Fig. 3. Flowchart of fuzzy upper-level controller.

In the course of our research two different fuzzy controllers were created and tested; one that included both Adaptive Cruise Control (ACC) and Cruise Control (CC) and the other that included ACC, CC, and Automatic Emergency Braking (AEB). Both of the fuzzy controllers were created and tested using MATLAB Simulink and the Fuzzy Logic System Designer Toolbox.

The fuzzy logic controller that only included ACC and CC functionality took in three linguistic variables, Relative Distance, Relative Velocity, and Speed Error, and output only one, Desired Acceleration. Relative Distance was defined as the distance difference from the correct headway distance and used Eq. (6), where x_{i-1} is the location of the lead vehicle; x_i is the location of the ego vehicle; A is the separation distance; T_h is the time headway and V is the ego vehicle's speed.

$$e_i = x_{i-1} - x_i - A + T_h V \tag{6}$$

Relative Velocity was defined as the velocity difference between the lead car and the ego car. Speed Error was defined as the speed difference between the ego car's speed and the speed the driver of the ego car set. Eqs. (7) and (8) were used for Relative Velocity and Speed Error, where v_{i-1} is the lead vehicle's speed and v_i is the ego vehicle's speed.

$$\frac{\Delta ei(t)}{\Delta t} = v_{i-1} - v_i \tag{7}$$

$$v_{i\ err} = v_{i\ des} - v_i \tag{8}$$

The fuzzy logic controller that included AEB functionality used the same three input linguistic variables but also included one additional input linguistic variable, Lead Acceleration. Lead Acceleration was defined as a direct sensor measurement of the lead vehicle's acceleration. This would allow the system to detect sharp decreases in speed and in turn activate the AEB system.

Relative Distance, Relative Velocity, and Lead Acceleration were all broken down into five different states while Speed Error was broken down into three. Utilizing these states, we were able to construct membership functions and a set of if-then rules to regulate the desired acceleration based on the state of the input linguistic variables. Relative Distance was broken down into five states (linguistic values): Close, A Little Close, Correct, A Little Far, and Far. The membership function was created using three triangle membership functions and two trapezoidal functions. The membership function graph is shown in Fig. 4.

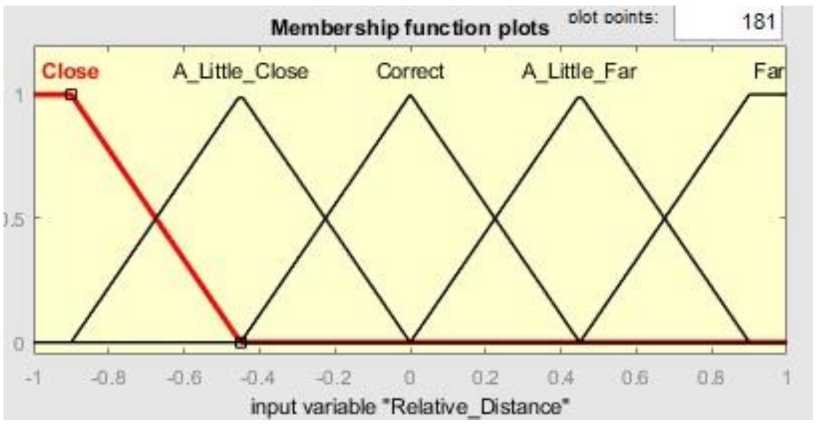


Fig. 4. Membership functions (linguistic values) for relative distance.

Relative Velocity was broken down into five states: Slow, A Little Slow, Zero, A Little Fast, and Fast. The membership function was created using three triangle membership functions and two trapezoidal functions. The membership function graph is shown in Fig. 5.

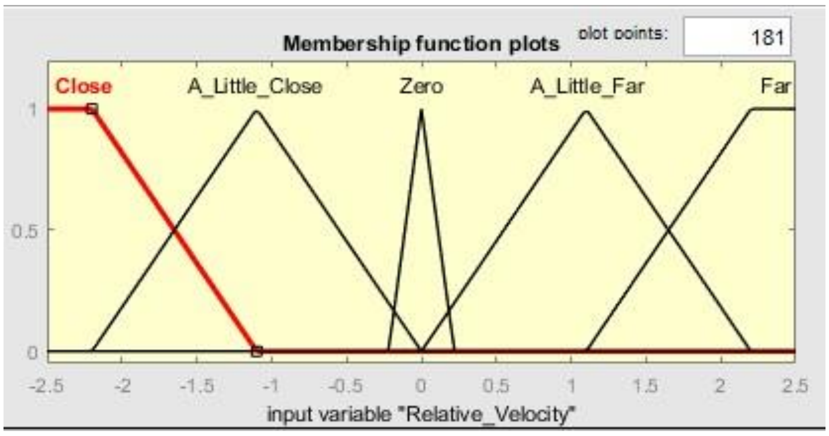


Fig. 5. Membership functions (linguistic values) for relative velocity.

Speed Error was broken down into three states: Negative, Zero, and Positive. The membership function was created using one triangle membership functions and two trapezoidal functions. The membership function is shown in Fig. 6.

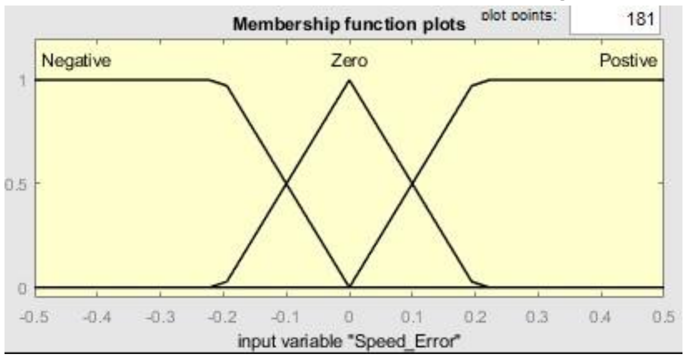


Fig. 6. Membership functions for speed error.

Lead Acceleration was broken down into five states: Slow Down, Slow Down A Little, Zero, Speed Up A Little, Speed Up. The membership function was created using three triangle membership functions and two trapezoidal functions. The membership function graph is shown in Fig. 7.

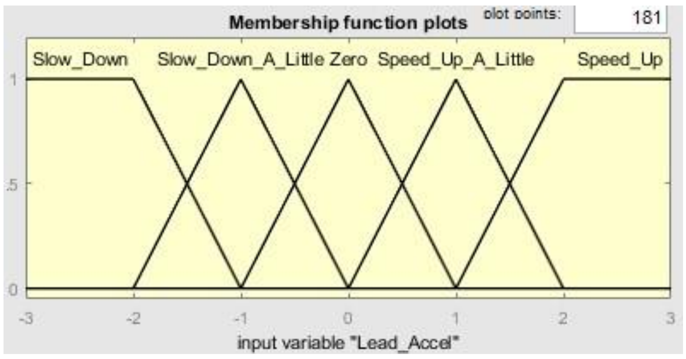


Fig. 7. Membership functions for lead acceleration.

The output linguistic variable consists of seven different states with a range of -6 m/s^2 to 6 m/s^2. The states of the output linguistic variable are as follows: Slow Down A Lot, Slow Down, Slow Down A Little, Zero, Speed Up A Little, Speed Up, Speed Up A Lot. The membership function consists of five triangle membership functions and two trapezoidal functions. The membership function graph is shown in Fig. 8.

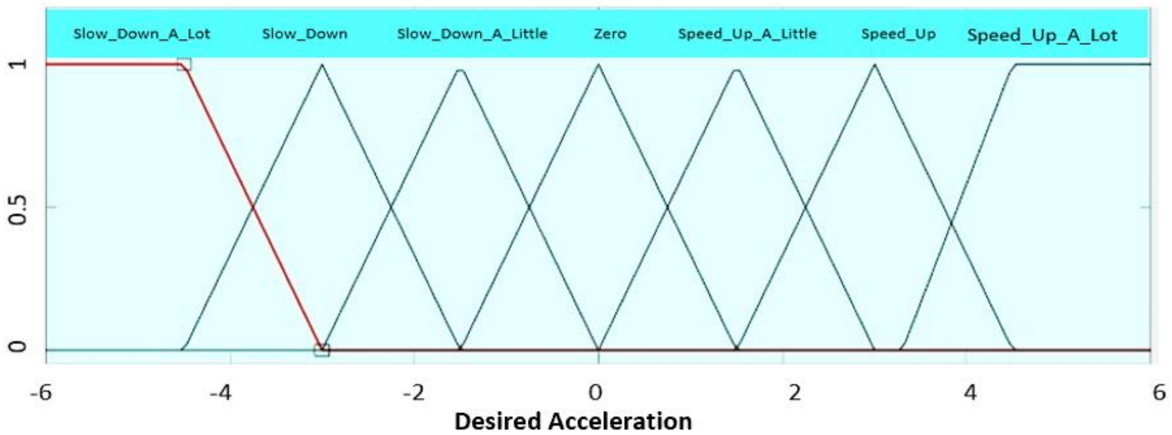


Fig. 8. Membership function plots for desired acceleration.

The membership states and their bounds were established based on a series of papers [6, 7, 11] that detailed a number of different ACC fuzzy controllers. Utilizing the membership functions, a series of thirty-three rules were developed for the fuzzy controller without AEB functionality and thirty-four rules were developed for the fuzzy controller with AEB functionality. Rules one through twenty-five encapsulate the ACC functionality, rules twenty-six through thirty encapsulate the CC functionality, and rules thirty-one through and thirty-four encapsulate the AEB functionality. The set of if-then rules are shown as follows:

<u>*ACC Rules:*</u>

1) RULE 1: IF *Relative Distance* is *Far* and *Relative Velocity* is *Slow* and *Speed Error* is *Negative* then *Desired Acceleration* is *Speed Up A Lot*.
2) RULE 2: IF *Relative Distance* is *A Little Far* and *Relative Velocity* is *Slow* and *Speed Error* is *Negative* then *Desired Acceleration* is *Speed Up*.
3) RULE 3: IF *Relative Distance* is *Correct* and *Relative Velocity* is *Slow* and *Speed Error* is *Negative* then *Desired Acceleration* is *Speed Up*.
4) RULE 4: IF *Relative Distance* is *A Little Close* and *Relative Velocity* is *Slow* and *Speed Error* is *Negative* then *Desired Acceleration* is *Speed Up A Little*.
5) RULE 5: IF *Relative Distance* is *Close* and *Relative Velocity* is *Slow* and *Speed Error* is *Negative* then *Desired Acceleration* is *Zero*.
6) RULE 6: IF *Relative Distance* is *Far* and *Relative Velocity* is *A Little Slow* and *Speed Error* is *Negative* then *Desired Acceleration* is *Speed Up*.
7) RULE 7: IF *Relative Distance* is *A Little Far* and *Relative Velocity* is *A Little Slow* and *Speed Error* is *Negative* then *Desired Acceleration* is *Speed Up*.
8) RULE 8: IF *Relative Distance* is *Correct* and *Relative Velocity* is *A Little Slow* and *Speed Error* is *Negative* then *Desired Acceleration* is *Speed Up A Little*.
9) RULE 9: IF *Relative Distance* is *A Little Close* and *Relative Velocity* is *A Little Slow* and *Speed Error* is *Negative* then *Desired Acceleration* is *Zero*.
10) RULE 10: IF *Relative Distance* is *Close* and *Relative Velocity* is *A Little Slow* and *Speed Error* is *Negative* then *Desired Acceleration* is *Slow Down A Little*.
11) RULE 11: IF *Relative Distance* is *Far* and *Relative Velocity* is *Zero* and *Speed Error* is *Negative* then *Desired Acceleration* is *Speed Up*.
12) RULE 12: IF *Relative Distance* is *A Little Far* and *Relative Velocity* is *Zero* and *Speed Error* is *Negative* then *Desired Acceleration* is *Speed Up A Little.*
13) RULE 13: IF *Relative Distance* is *Correct* and *Relative Velocity* is *Zero* and *Speed Error* is *Negative* then *Desired Acceleration* is *Zero*.
14) RULE 14: IF *Relative Distance* is *A Little Close* and *Relative Velocity* is *Zero* and *Speed Error* is *Negative* then *Desired Acceleration* is *Slow Down A Little*.
15) RULE 15: IF *Relative Distance* is *Close* and *Relative Velocity* is *Zero* and *Speed Error* is *Negative* then *Desired Acceleration* is *Slow Down*.
16) RULE 16: IF *Relative Distance* is *Far* and *Relative Velocity* is *A Little Fast* and *Speed Error* is *Negative* then *Desired Acceleration* is *Speed Up A Little*.
17) RULE 17: IF *Relative Distance* is *A Little Far* and *Relative Velocity* is *A Little Fast* and *Speed Error* is *Negative* then *Desired Acceleration* is *Zero*.
18) RULE 18: IF *Relative Distance* is *Correct* and *Relative Velocity* is *A Little Fast* and *Speed Error* is *Negative* then *Desired Acceleration* is *Slow Down A*

Little.

19) RULE 19: IF *Relative Distance* is *A Little Close* and *Relative Velocity* is *A Little Fast* and *Speed Error* is *Negative* then *Desired Acceleration* is *Slow Down*.
20) RULE 20: IF *Relative Distance* is *Close* and *Relative Velocity* is *A Little Fast* and *Speed Error* is *Negative* then *Desired Acceleration* is *Slow Down*.
21) RULE 21: IF *Relative Distance* is *Far* and *Relative Velocity* is *Fast* and *Speed Error* is *Negative* then *Desired Acceleration* is *Zero*.
22) RULE 22: IF *Relative Distance* is *A Little Far* and *Relative Velocity* is *Fast* and *Speed Error* is *Negative* then *Desired Acceleration* is *Slow Down A Little*.
23) RULE 23: IF *Relative Distance* is *Correct* and *Relative Velocity* is *Fast* and *Speed Error* is *Negative* then *Desired Acceleration* is *Slow Down A Little*.
24) RULE 24: IF *Relative Distance* is *A Little Close* and *Relative Velocity* is *Fast* and *Speed Error* is *Negative* then *Desired Acceleration* is *Slow Down*.
25) RULE 25: IF *Relative Distance* is *Close* and *Relative Velocity* is *Fast* and *Speed Error* is *Negative* then *Desired Acceleration* is *Slow Down A Lot*.

Cruise Control Rules:

1) RULE 26: IF *Speed Error* is *Positive* then *Desired Acceleration* is *Slow Down A Little*.
2) RULE 27: IF *Relative Distance* is *Far* and *Speed Error* is *Positive* then *Desired Acceleration* is *Speed Up A*

   Little.
3) RULE 28: IF *Relative Distance* is *A Little Far* and *Speed Error* is *Positive* then *Desired Acceleration* is *Speed Up A Little*.
4) RULE 29: IF *Relative Distance* is *A Little Close* or *Close* and *Relative Velocity* is *Zero* and *Speed Error* is *Zero* then *Desired Acceleration* is *Slow Down A*

   Little.
5) RULE 30: IF *Relative Distance* is *Close* and *Relative Velocity* is *Zero* and *Speed Error* is *Zero* then *Desired Acceleration* is *Slow Down A Little.*

Automatic Emergency Braking Rules:

1) RULE 31: IF *Relative Distance* is *Close* and *Lead Acceleration* is *Slow Down* then *Desired Acceleration* is *Slow Down A Lot*.
2) RULE 32: IF *Relative Distance* is *Close* and *Lead Acceleration* is *Slow Down A Little* then *Desired Acceleration* is *Slow Down*.
3) RULE 33: IF *Relative Distance* is *A Little Close* and *Lead Acceleration* is *Slow Down* then *Desired Acceleration* is *Slow Down A Lot*.
4) RULE 34: IF *Relative Distance* is *A Little Close* and *Lead Acceleration* is *Slow Down A Little* then *Desired Acceleration* is *Slow Down*.

3.2. ACC MPC Controller

The MPC design for the ACC system is based on [4, 12]. But the control design is not completely copied from these references and instead some modification is done by the addition of extra state and constraints. The control objective is to find the optimal acceleration that results in minimization of distance and velocity errors, while respecting other constraints.

To compute the optimal acceleration, the relationship between the ego and the lead car is established. However, to consider the time delay when applying the throttle and brake signal by the actuators, sensors etc. (the low-level controller), a first order time lag is considered in the construction of the vehicle following model as sown in Eq. (9).

$$\frac{d\alpha_e(t)}{dt} + \alpha_e(t) = \alpha_d(t) \tag{9}$$

where $d\alpha_e(t)$ is the first order lag caused by the actuators and sensors during the implementation; a_e is the acceleration of the ego car and a_d is the desired acceleration output from the upper-level controller.

Since the relative distance and relative velocity changes with the time we can represent them as shown in Eqs. (10) and (11).

Relative Distance: $d(t) = d_l(t) - d_e(t)$ (10)

Relative Velocity: $v(t) = v_l(t) - v_e(t)$ (11)

where d_l & v_l are distance and velocity of the lead car; d_e & v_e are the distance and velocity of the ego car, respectively.

Using the Eqs. (10) and (11), the error of relative distance and error of relative velocity can be respectively defined as in Eqs. (12) and (13).

$$\Delta d(t) = d(t) - d_{des}(t) \tag{12}$$

$$\Delta v(t) = v(t) - v_{des}(t) \tag{13}$$

The term d_{des} is desired distance, which can be calculated by the constant time headway spacing policy that is given by Eq. (14).

$$d_{des}(t) = T_h v_e(t) + d_0 \tag{14}$$

In Eq. (14), d_0 is the default spacing distance which should be maintained when both the ego and lead car are at a complete stop and T_h is the constant time headway that can be figured out within the range of human reaction time 1.5s to 2.5s.

Using the above equations, we can define the ACC MPC controller model in the form of a continuous time state space representation as shown in Eq. (15) as follows:

$$\begin{bmatrix} \Delta \dot{d} \\ \Delta \dot{v} \\ \dot{a}_e \\ a_e \end{bmatrix} = \begin{bmatrix} 0 & 1 & -T_h & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & -\frac{1}{\tau} & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \Delta d \\ \Delta v \\ a_e \\ v_e \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \frac{1}{\tau} \\ 0 \end{bmatrix} a_d + \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} a_l \tag{15}$$

And the Output equation is shown as follows:

$$\begin{bmatrix} \Delta d \\ \Delta v \\ a_e \\ v_e \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \Delta d \\ \Delta v \\ a_e \\ v_e \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} a_d + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} a_l \tag{16}$$

To form the above system into an optimization problem that can be solved in real-time to find the required input, we need to discretize the above state space model using a zero-order hold, giving matrices A_d, B_d, C *and* D. In MPC the current control action is obtained by solving an optimization problem, and the value of the solved incremental control signal $\Delta u(k)$ is firstly applied. Then the discrete time horizon moves one step ahead, and the process is repeated. Therefore, the incremental equation of the vehicle-following model is represented in Eqs. (17) and (18) as follows:

$$\Delta x(k+1) = A_d \Delta x(k) + B_{d,u} \Delta u(k) + B_{d,d} \Delta w(k) \tag{17}$$

$$\Delta y(k) = C\Delta x(k)+y(k-1) \tag{18}$$

where the change in the system state is given by, $\Delta x(k) = x(k) - x(k-1)$, and $x(k)$ is the present state and $x(k-1)$ is the previous predicated states. $\Delta u(k) = u(k)-u(k-1)$ is the change in the control input and the change in disturbance is given by $\Delta w(k) = w(k) - w(k-1)$.

If p is the prediction horizon, m is the control horizon, then the future system states are predicted by the following predicted performance vector:

$$Y_p(k+1 \mid k) = S_x\Delta x(k)+Iy(k)+S_d\Delta w(k)+S_u\Delta U(k)$$

where the predictive performance $Yp\ (k+1 \mid k)$ at the sample time k is given by Eq. (19) as follows:

$$Y_p(k+1\mid k) = \begin{bmatrix} Y_p(k+1\mid k) \\ Y_p(k+2\mid k) \\ \vdots \\ Y_p(k+p\mid k) \end{bmatrix}_{p*1} \tag{19}$$

The change in the control input vector $\Delta U\ (k)$ of the ACC system at the sample time k is given by Eq. (20).

$$\Delta U(k) = \begin{bmatrix} \Delta u(k) \\ \Delta u(k+1) \\ \vdots \\ \Delta u(k+m-1) \end{bmatrix}_{m*1} \tag{20}$$

where, S_x, S_d, S_u are the matrix parameters given by the following:

$$S_x = \begin{bmatrix} CA \\ CA+CA^2 \\ \vdots \\ \sum_{i=1}^{p} CA^i \end{bmatrix}_{p*1}, \quad S_d = \begin{bmatrix} CB_d \\ CAB_d+CB_d \\ \vdots \\ \sum_{i=1}^{p} CA^{i-1}B_d \end{bmatrix}_{p*1},$$

$$S_u = \begin{bmatrix} CB_d & 0 & \cdots & 0 \\ CB_d+CA_dB_d & CB_d & \cdots & 0 \\ \vdots & \vdots & \cdots & \vdots \\ \sum_{i=0}^{p-1} CA_d^iB_d & \sum_{i=0}^{p-2} CA_d^iB_d & \cdots & \sum_{i=0}^{p-m} CA_d^iB_d \end{bmatrix}_{p*m}$$

$$, I = \begin{bmatrix} I_{3*3} \\ I_{3*3} \\ \vdots \\ I_{3*3} \end{bmatrix}_{p*1}$$

At the sample time k, the relationship between the current disturbance and its previous state can be expressed as $\Delta w(k) = w(k-1 \mid k)$. Assuming the disturbance remains constant in the prediction horizon, which most of the researchers uses while modelling the disturbance, then the disturbance matrix is given by Eq. (21).

$$\Delta w(k) = \begin{bmatrix} w(k-1|k) \\ w(k-1|k) \\ \vdots \\ w(k-1|k) \end{bmatrix}_{p*1} \tag{21}$$

Cost function formulation for the performance of the vehicle following plant is given by Eq. (22).

$$J = \min_{\Delta U} \left\| \Gamma_y \left(S_u\Delta U(k) - E_p(k+1\mid k) \right) \right\|^2 + ||\Gamma_u \Delta U(k)||^2 \tag{22}$$

where the error matrix is given by the following:

$E_p(k+1 \mid k) = R(k+1) - S_x\Delta x(k) - Iy(k) - S_d\Delta w(k)$

And R(k+1) is the set point vectors associated with each output vector at the sample time k+1, Γ_y = is the weight matrix of the output and Γ_u is the weight scale of the input increment.

Eq. (22) is formulated based on the quadratic programming as shown in Eq. (23).

$$J = \Delta U(k)' H \Delta U(k) - G(k+1 \mid k)' \Delta U(k) \tag{23}$$

Subjected to $C_u \Delta U(k) \geq b(k+1 \mid k)$

where, C_u & $b(k+1 \mid k)$ are the constraint variables whose values are defined in Table 1 and the weight matrix is given by the following:

$$\Gamma_y = \begin{bmatrix} w_d & 0 & 0 & 0 \\ 0 & w_v & 0 & 0 \\ 0 & 0 & w_{a_e} & 0 \\ 0 & 0 & 0 & w_{v_e} \end{bmatrix} \; \& \; \Gamma_u = w_u$$

where w_d is the corresponding weight of its inter vehicle distance error; w_v is the corresponding weight of the velocity error; w_{ae} is the corresponding weight of acceleration and w_{ve} is the corresponding weight of the ego velocity.

Table 1. Constraints and parameters used in MPC optimization problem.

Parameter	Value	Parameter	Value
Prediction Horizon, p	30	w_{ve}	0
Control Horizon, m	5	τ	0.05
w_d	5	T_s	0.05
w_v	5	d_0	2m
w_a	1	U	$-0.61g \leq u \leq 0.5g$
w_u	10	a_d	$-0.61g \leq u \leq 0.5g$
g	9.8	Δj(jerk)	$-2 \leq \Delta j \leq 2$
T_h	2s	v_e	$v_{min} \leq v_e \leq v_{max}$

3.3. Plant Model

A double axle dynamic bicycle model is used to model the vehicle plant [13]. In this vehicle model, as shown in Fig. 9, there are 5 degrees of freedom (DOF) and the states, X, for the system are longitudinal velocity, V_x, lateral velocity, V_y, yaw rate, ω_y, front tire rotational velocity ω_f, and rear tire rotational velocity, ω_r. The inputs, U, to the plant are steering wheel angle, δ, and driving wheel torque, τ_d.

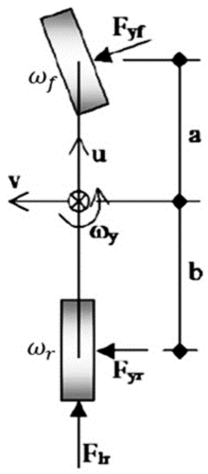


Fig. 9. Five-DOF bicycle plant model.

The schematic of the plant model and the respective equations of each state in the continuous time domain are provided in Eqs. (24) through (31). (The lateral portion of the model is not used in this work and is just provided for completeness of the model. Only the longitudinal portion of the above model is utilized.)

Nonlinear Dugoff Tire Model

$Fyi = Fyi(ai,si,Fzi,\mu)$ and $Fli = Fli(ai,si,Fzi,\mu)$

Where, F_{zi} is the normal force on front or rear tires and s_i is the longitudinal slip for front or rear tires given by:

Vehicle Body

$$m(\dot{U} - V\omega) = F_{lf}+F_{lr} \tag{24}$$

$$m(\dot{V} + U\omega) = F_{yf}+F_{yr} \tag{25}$$

$$J(\dot{\omega}) = aF_{yf}+bF_{yr} \tag{26}$$

Wheel Equations:

$$\tau f - RFlf = Jw\dot{\omega}f \tag{27}$$

$$\tau r - RFlr = Jw\dot{\omega}r \tag{28}$$

Tire Slip:

$$af = \delta - \frac{V+a\omega}{U} \tag{29}$$

$$ar = \frac{-V+b\omega}{U} \tag{30}$$

$$si = \frac{R_w\omega_i - U}{|R_w\omega_i} \tag{31}$$

The tire model we used is a nonlinear model which develops tire forces in terms of the available friction of the tire. Hence the friction circle concept is used, and the square sum of longitudinal and lateral tire forces should be equal to the available friction force/vertical load. The equations and details of the model are provided in [14]. The vehicle and tire parameters used in this plant model are for a sedan vehicle are provided in Table 2.

Table 2. Vehicle plant parameters.

Parameter	Value
a, Distance from the center of gravity to front axle [m]	1.029
b, Distance from the center of gravity to rear axle [m]	1.471
J, Yaw moment of inertia [kgm^2]	3100
m, Vehicle Mass [kg]	1700
R, Wheel radius [m]	0.334
Jw, Wheel moment of inertia [kgm^2]	3
Fnf, Front Wheel Vertical Load [N]	9812.7468
Fnr, Rear Wheel Vertical Load [N]	6864.2532
n, Steering gear ratio	16
Cornering stiffness, Cy [N/rad]	88964.4
Longitudinal stiffness, Cx [N/rad]	88964.4
Road Friction coefficient, μ	0.9

3.4. Lower-Level Controller

The lower-level controller (LLC) is composed of feedforward terms only. This choice is made in order to focus more on upper-level controller design. The main goal of LLC is to convert the controller acceleration to driving wheel torque for the autonomous vehicle, which is assumed to have inbuilt motors in each axle. The value of driving wheel torque is positive when the vehicle needs to move forward and negative when it needs to brake as shown in Eqs. (32), (33) and (34).

$$FD(N) = \frac{m}{\tau_1 s+1} \times T\ (m/s^2) - \frac{m\tanh(0.8Vx)}{\tau_2 s+1} \times T \times B(m/s^2) \tag{32}$$

$$F_{Df} = F_{Dr} = 0.5 \times F_D(N) \tag{33}$$

$$\tau_{Df}(Nm) = \tau_{Dr}(Nm) = F_{Df} \times R_w = F_{Dr} \times R_w \tag{34}$$

where, T= Throttle, B = Brake, F_{Df} & F_{Dr} are front and rear driving force, F_D= total driving force, τ_{Df} & τ_{Dr} are front and rear driving torque, V_x= ego vehicle speed and $\tau_1 = \tau_2 = 0.05s$ are the first order delay terms for throttle and brake.

In the equations above a first order delay term is utilized to depict a realistic delay in throttle or brake actuation when the driver presses the gas or brake pedal. We do not use more complex engine dynamics in LLC since the plant dynamics are kept simple, and we do not use complete engine models available in CarSim or other software due to licensing issues.

4. SOFTWARE IMPLEMENTATION

The controller and plant implementation is carried out in Simulink using Fuzzy logic toolbox, MPC Toolbox and the general Simulink toolboxes. The solver chosen for the simulations is ode23s, which is a stiff solver and required for correctly dealing with the stiff plant model used in our simulation. Fig. 10 shows the MPC implementation in Simulink and Fig. 11 shows the Fuzzy Logic Controller implementation in Simulink.

From both of these figures it can be seen that the lead car acceleration is a user defined input, which is done by using the signal generator block in Simulink. Furthermore, for calculating the relative distance between the lead and the ego car we use the Euclidean distance formula.

Also, we can see that the lead car subsystem is shown, which models it as a simple particle with a double integrator model. Eqs. (35), (36) and (37) shows this model as follows:

$$a_{lead} = \frac{a_{lead,desired}}{1+0.05s} \tag{35}$$

$$\dot{v}_{lead} = a_{lead} \tag{36}$$

$$\dot{X}_{lead} = v_{lead} \tag{37}$$

5. SIMULATION AND ANALYSIS

To check the functionality of the three controllers (Base Fuzzy Controller, Improved Fuzzy Controller ($Fuzzy_{AEB}$) and MPC), they were tested in three different scenarios:

1) Scenario 1: Verification simulation from [8], to check if the developed fuzzy controller has similar performance to the reference. This scenario has the lead car accelerating and decelerating from/to rest.

2) Scenario 2: Cut in scenario, where a lead car suddenly cuts in front of the ego car.

3) Scenario 3: Emergency Braking scenario, ego car encounters a lead vehicle that suddenly decelerates.

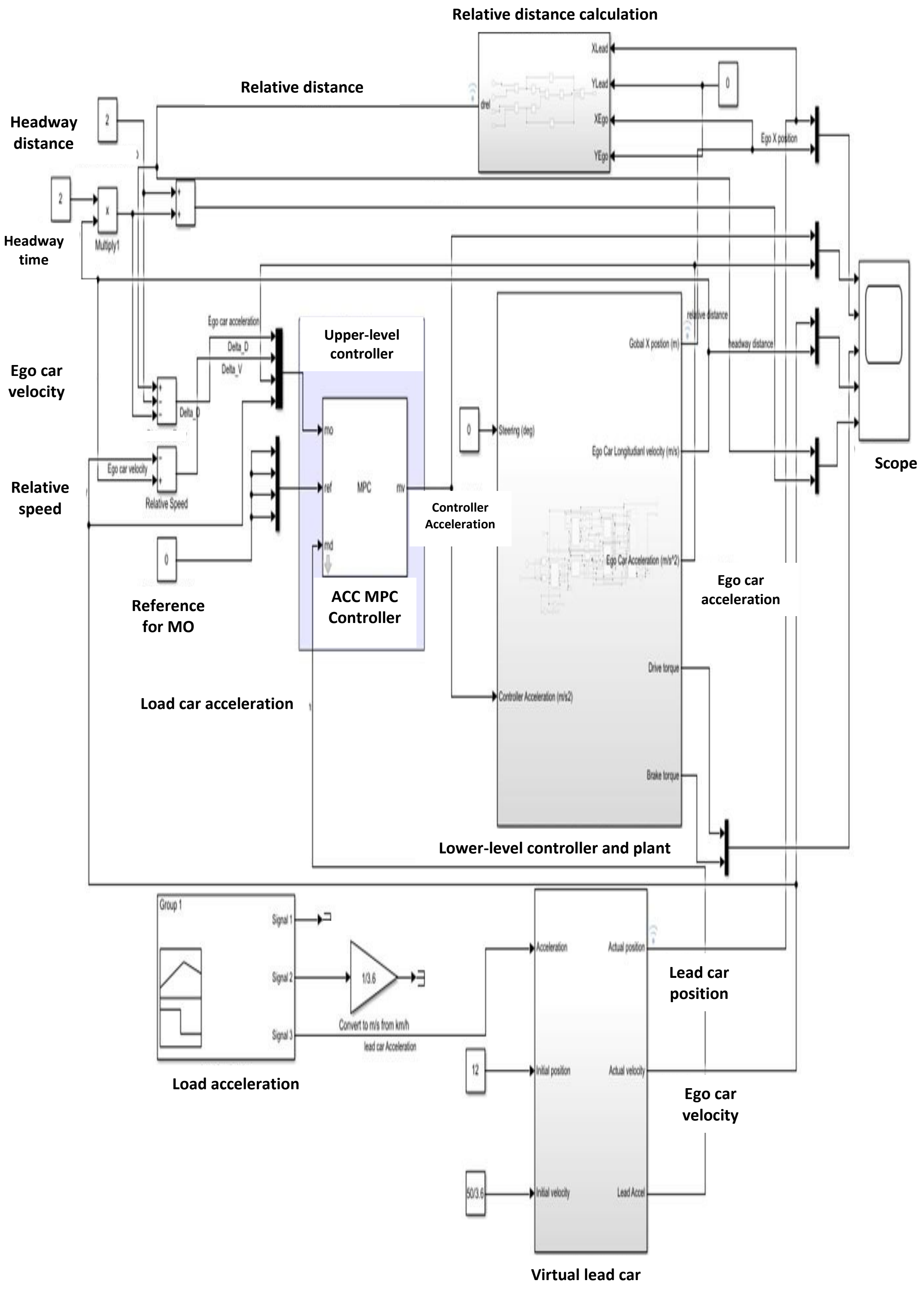


Fig. 10. MPC upper-level controller implementation.

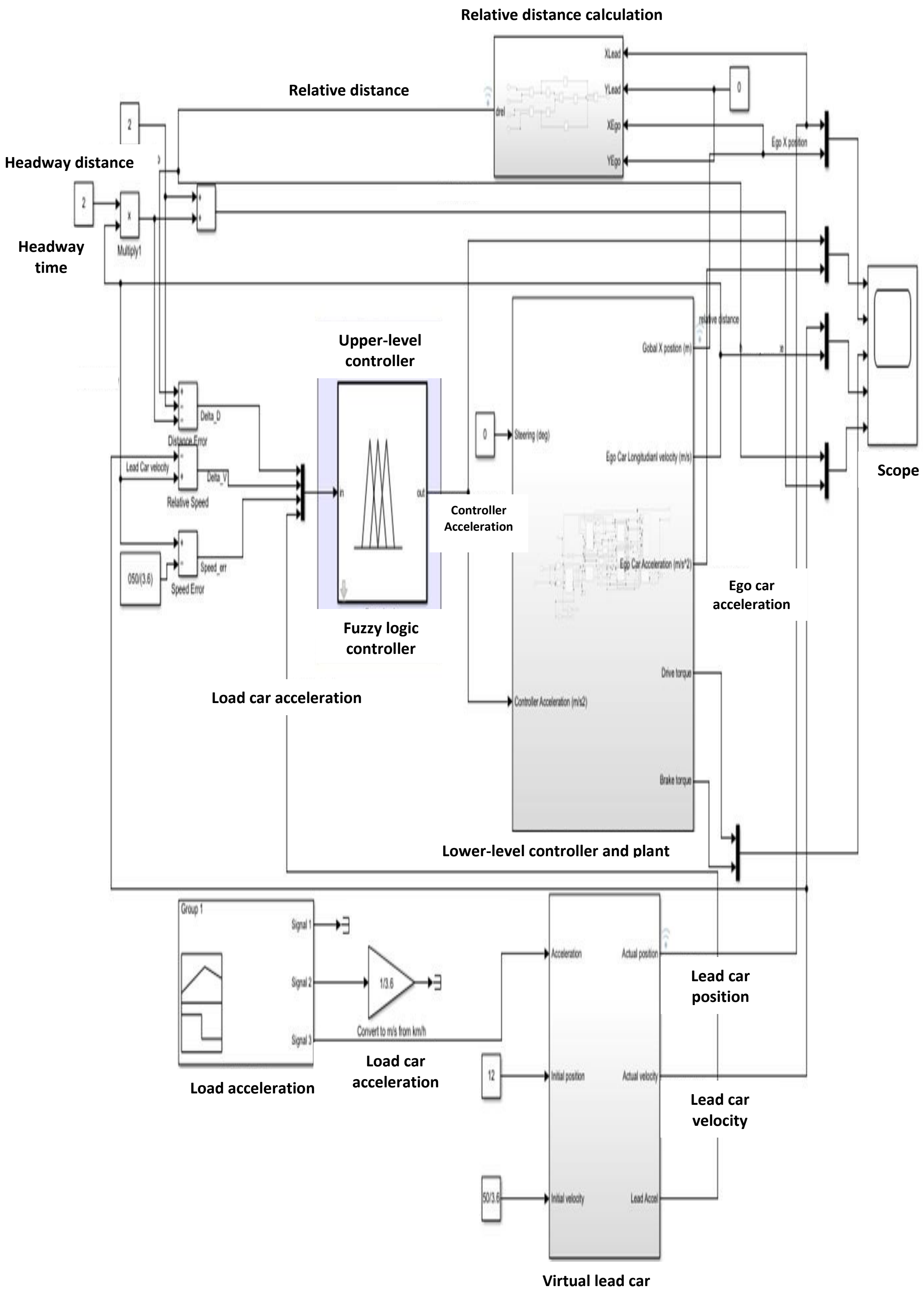


Fig. 11. Fuzzy upper-level controller implementation.

Details for each of these scenarios are provided in their respective sections. In each of the results that will be shown; Fuzzy = Base Fuzzy Controller; $Fuzzy_{AEB}$ = Improved Fuzzy Controller and MPC = ACC Model Predictive Controller.

5.1. Scenario1

Scenario 1 requires the ego vehicle to follow the lead vehicle under a varying level of accelerations and eventually results in the ego vehicle going into Cruise Control (CC) mode. In Fig. 12, the blue line represents the lead car velocity and the red line represents the velocity of the ego car with the controller developed in [8]. In this verification simulation our main aim is to replicate the fuzzy controller and see if its output is similar to the reference paper output.

The results of this scenario are shown in Fig. 13. The results show that all three controllers do a good job of keeping the distance error at zero. Furthermore, we also see that the velocity plot for the ego car is very similar to that shown in the reference paper plots. These results show that the fuzzy controllers developed work similarly to the reference paper and now they can be tested in other scenarios.

Furthermore, if we look at the plots, we can see that apart from the MPC, both fuzzy controllers undergo a big overshoot and undershoot in controller acceleration when we shift from ACC to CC mode at around 120 s. This may be due to the lack of fuzzy rules that represents the transition to CC mode.

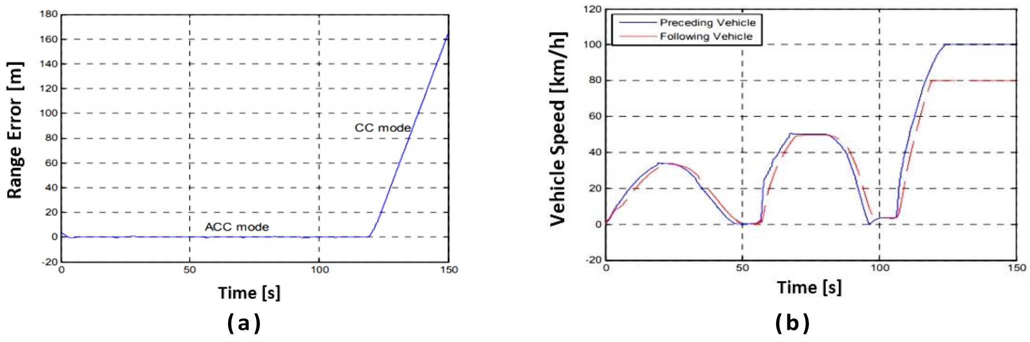


Fig. 12. a) Range error; b) speed plots that are to be replicated.

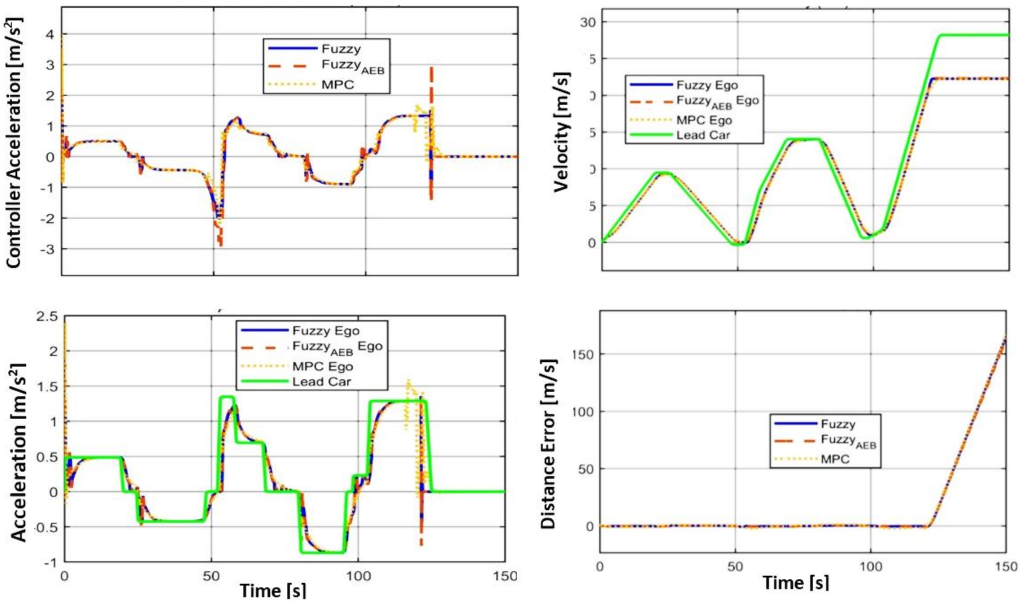


Fig. 13. Scenario 1 results for all three controllers.

5.2. Scenario 2

This scenario involves the lead car suddenly cutting into the ego car lane as shown in Fig. 14. This scenario is made to test the three controllers in the situation where deceleration is required to manage the spacing between the lead car.

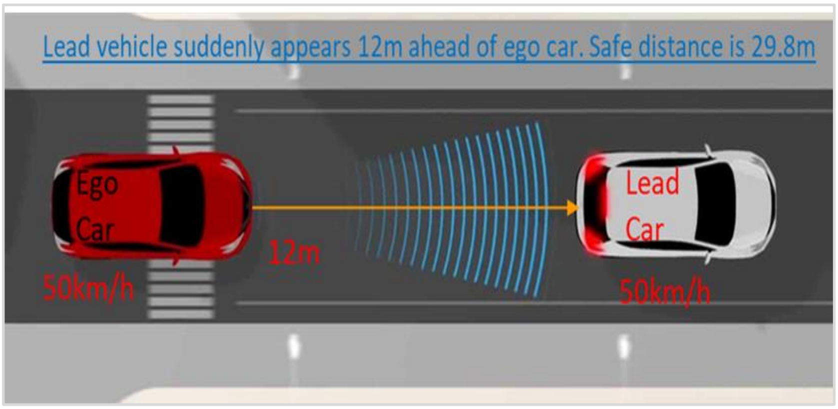

Lead vehicle suddenly appears 12m ahead of ego car. Safe distance is 29.8m
Ego Car
50km/h
12m
Lead Car
50km/h

Fig. 14. Scenario 2 schematic explanation.

The correct controller actuation should see the ego car immediately braking smoothly to get the distance error back to zero and then following the lead car speed. The results of this simulation are shown in Fig. 15. From the velocity and the distance error plots we can see that both the fuzzy controllers do a good job of minimizing the distance error and then tracking the lead car velocity. Also, we can see that they have a smoother acceleration output with less jerks (overshoots). On the other hand, the MPC undergoes large decelerations and accelerations at the start to rapidly bring the distance error back to zero. But due to this jerky acceleration the MPC has overshoots even after it has brought the distance error back to zero. This performance by the MPC is not desirable and will cause discomfort to the vehicle occupants. The reason for this behavior is going to be discussed in the controller comparison section.

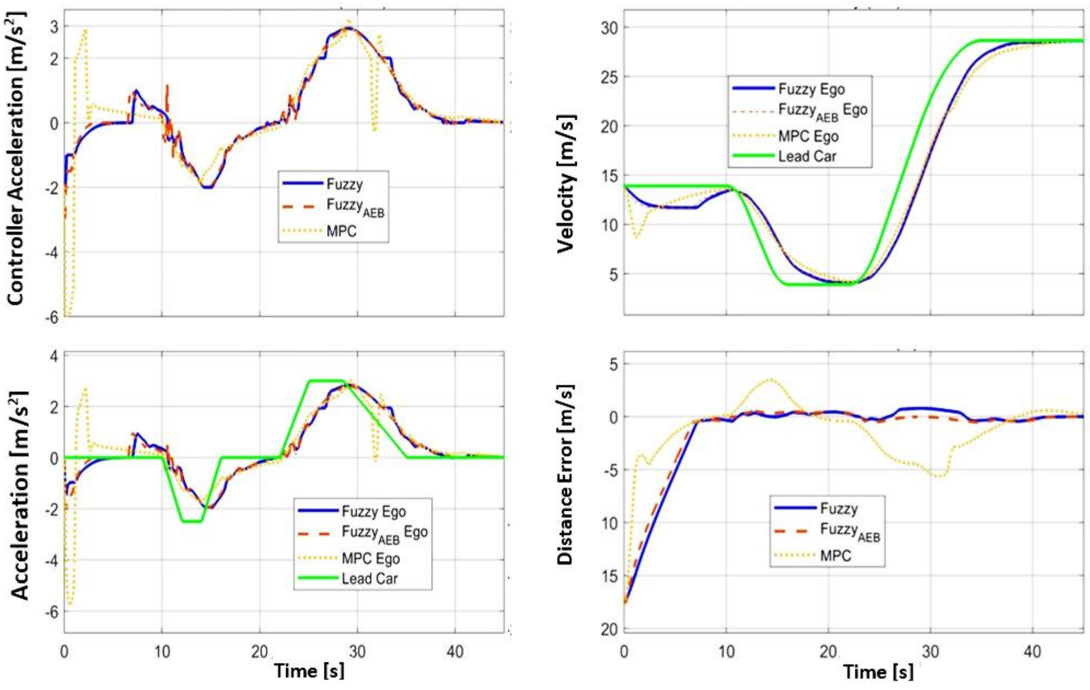


Fig. 15. Scenario 2 results for all three controllers.

5.3. Scenario 3

This scenario requires the ego vehicle traveling at 50 km/h to suddenly break due to lead vehicle, 12m ahead, suddenly deciding to break (Fig. 16). This scenario is part of the CCrb EURO NCAP test carried out for testing AEB controllers [15].

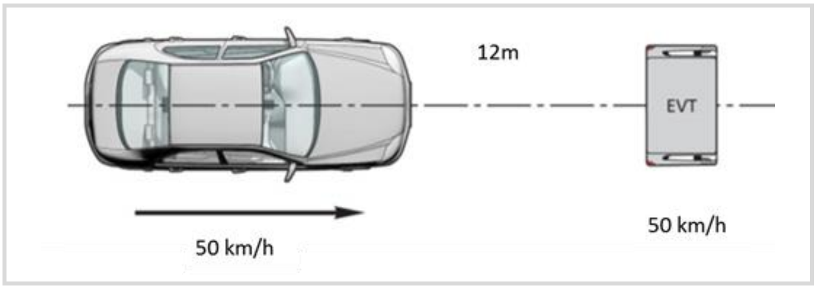


Fig. 16. Scenario 3 schematic explanation, taken from [15].

By using this scenario for testing our three controllers we aim to figure out if they are robust, that is, can they work at large accelerations and low speeds without any big performance drop? The results for this scenario are shown in Fig. 17. From the distance error and velocity graphs it can be seen that $Fuzzy_{AEB}$ is the controller that works the best and is able to bring the car to rest the fastest. The MPC does work for this scenario, but again it has a large jerky deceleration which does help reduce the distance error to zero very quickly. This though ends up giving a large overshoot in the distance error as the vehicle stops too much and ends up having a positive distance error. Therefore, the MPC then has to accelerate to bring the vehicle to zero distance error.

Furthermore, we can also see that the base fuzzy controller does not work for this scenario as it is locked into the CC mode. This is because it does not possess the rules required for emergency braking scenarios or the lead acceleration input.

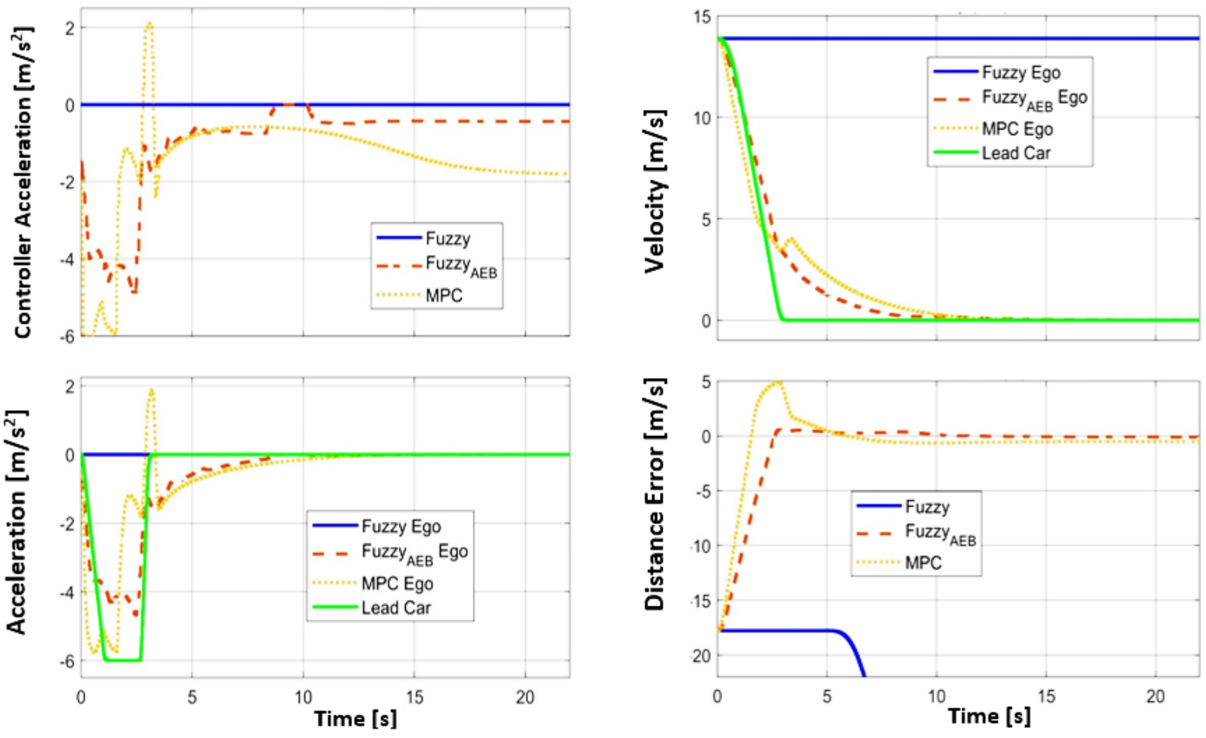


Fig. 17. Scenario 3 results for all three controllers.

5.4. Controller Comparison and Analysis

To summarize the above results, Table 3 is constructed to show the controller performance in each scenario. In the Table 3 the red box indicates controllers which do not work correctly in the scenario, while the green box indicates vice versa.

The improved fuzzy controller works in all scenarios. Furthermore, from the results we can also see that the improved fuzzy controller has smoother ego velocity and acceleration, which shows that the extra rules and inputs used serve their purpose well.

On the other hand, the base fuzzy controller fails to work in scenario 3 completely and is stuck in CC mode due to it not having rules for that scenario. Therefore, the base controller cannot be considered a 'smart ACC' controller.

Furthermore, the MPC is seen to work in all scenarios, but its performance in scenario 2 is not very good and hence it fails that scenario. But this performance is closely related to weight tuning of MPC and the control structure is not at fault for it. The current output weighting for the MPC is [5, 5, 1, 0]. This results in distance and velocity error having equal weighting and means that the controller is going to be very aggressively reducing the distance error by large accelerations [4]. Due to this reason, the authors in [4] mention that a real-time weight-tuning MPC is required for optimal performance in each scenario.

Table 3. Summary of controller pass/fail in each scenario.

Scenario	MPC Controller	Base Fuzzy Controller	Improved Fuzzy Controller
Scenario 1: Base ACC	🟩	🟩	🟩
Scenario 2: Lead vehicle Cut in	🟥	🟩	🟩
Scenario 3: Sudden stop	🟩	🟥	🟩

From the above comments, it can be concluded that the improved fuzzy controller and the MPC controller are the two controllers that can work in all scenarios. In the current form, the improved fuzzy controller is better than the MPC controller and this is also further emphasized by the comparison shown in Table 4. But, we cannot discount the MPC controller just yet, because we need further evidence with weight tuning of MPC weights for each scenario to see if performance improves and gets better than the fuzzy controller. This work is left for the future and not considered in this paper in order to keep the paper scope small.

Table 4. Comparison of improved fuzzy controller and MPC.

Improved fuzzy controller	MPC controller
Operates under all scenario conditions correctly - robust	Not very robust unless weight tuning is done for each scenario
Operates with less overshoot and less harsh acceleration commands	Trends towards jerky response
Easy to implement if-then rules	Requires complicated, high-fidelity models for better performance
Low memory cost when implementing	High memory cost generally, and increases with higher-fidelity models

6. CONCLUSIONS AND FUTURE WORK

This work aimed at comparing a fuzzy logic controller with a MPC in order to find a 'SMART' ACC controller that can work in all scenarios. In order to carry out this comparison we developed two fuzzy logic controllers, one based on [8] and the second an improvement on the first by the addition of more rules and an input. Furthermore, we developed a MPC controller based on [4] but improved with the addition of another state variable to the model. Simulink toolboxes were used to implement the three controllers, and their output was connected to a 5 DOF vehicle bicycle plant model via a simple lower-level controller. Three simulations were carried out which looked at different aspects of longitudinal vehicle motion. The results showed that the improved fuzzy controller and the MPC controller work in all the scenarios, with the improved fuzzy controller being better than the MPC controller in the current form. But, no outright claim can be made for the improved fuzzy controller being better than the MPC since more work with real-time weight tuning of MPC weights is required for fair comparison.

In the future, we would like to improve the behavior of the fuzzy controllers when switching between different modes or rules, since the current switching is jerky. For this purpose, additional inputs and rules pertaining to, acceleration rate or jerk, can be considered. Furthermore, we would like to compare the controllers with real-time weight tuning of MPC weights for each scenario, as this is expected to improve MPC performance. Lastly, for a realistic comparison, we should compare these controllers with a high-fidelity and realistic vehicle model that is available in software, like CarSim.

REFERENCES

[1] P. Nilsson, O. Hussien, A. Balkan, Y. Chen, A. D. Ames, J. Grizzle, N. Ozay, H. Peng, P. Tabuada, "Correct-by-construction adaptive cruise control: Two approaches," *IEEE Transactions on Control Systems Technology*, vol. 24, no. 3, pp. 1294–1307, 2016, doi:10.1109/TCST.2015.2501351.

[2] P. Ioannou, C. Chien, "Autonomous intelligent cruise control," *IEEE Transactions on Vehicular Technology*, vol. 42, no. 4, pp. 657–672, 1993, doi: 10.1109/25.260745.

[3] C. Liang, H. Peng, "Optimal adaptive cruise control with guaranteed string stability," *Vehicle System Dynamics*, vol. 32, no. 4-5, pp. 313–330, 1999, doi:10.1076/vesd.32.4.313.2083.

[4] R. Zhao, P. Wong, Z. Xie, J. Zhao, "Real-time weighted multi-objective model predictive controller for adaptive cruise control systems," *International Journal of Automotive Technology*, vol. 18, no. 2, pp. 279–292, 2017, doi:10.1007/s12239−017−0028−2.

[5] H. Xiong, L. Boyle, J. Moeckli, B. Dow, T. Brown, "Use patterns among early adopters of adaptive cruise control," *Human Factors: The Journal of the Human Factors and Ergonomics Society*, vol. 54, no. 5, pp. 722–733, 2012, doi: 10.1177/0018720811434512.

[6] R. Holve, P. Protzel, K. Naab, "Generating fuzzy rules for the acceleration control of an adaptive cruise control system," *Proceedings of North American Fuzzy Information Processing*, 1996, doi: 10.1109/NAFIPS.1996.534776.

[7] S. Sathiyan, S. Kumar, A. Selvakumar, "Optimized fuzzy controller for improved comfort level during transitions in cruise and adaptive cruise control vehicles," *International Conference on Signal Processing and Communication Engineering Systems*, 2015, doi: 10.1109/SPACES.2015.7058221.

[8] S. Ko, J. Lee, "Fuzzy logic based adaptive cruise control with guaranteed string stability," *International Conference on Control, Automation and Systems*, 2007, doi: 10.1109/ICCAS.2007.4406871.

[9] Y. Lin, H. Nguyen, C. Wang, "Adaptive neuro-fuzzy predictive control for design of adaptive cruise control system," *IEEE 14th International Conference on Networking, Sensing and Control (ICNSC)*, 2017, doi: 10.1109/ICNSC.2017.8000187.

[10] Y. Lin, H. Nguyen, "Adaptive neuro-fuzzy predictor based control for cooperative adaptive cruise control system," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 3, pp. 1054–1063, 2020, doi: 10.1109/TITS.2019.2901498.

[11] *MathWorks*, "Adaptive cruise control system using model predictive control," 2022, https://www.mathworks.com/help/mpc/ug/adaptive-cruisecontrol-using-model-predictive-controller.html.

[12] S. Ahmed, "Development, testing, and validation of ADAS systems in a scaled vehicle model," *Master's thesis, University of Michigan-Dearborn*, April 2021.

[13] S. Yu, E. Sheng, Y. Zhang, Y. Li, H. Chen, Y. Hao, "Efficient nonlinear model predictive control of automated vehicles," *Mathematics*, vol. 10, no. 21, p. 4163, 2022, doi: 10.3390/math10214163.

[14] R. Guntur, S. Sankar, "A friction circle concept for dugoff's tire friction model," International Journal of Vehicle Design, vol. 1, no. 4, pp. 373–377, 1980, doi: /10.1504/IJVD.1980.061234.

[15] *Test protocol–AEB systems*, Tech. Rep. Version 2, EUROPEAN New Car Assessment Program, Mar 2017, https://cdn.euroncap.com/media/26996/euro-ncap-aeb-c2c-test-protocol-v20.pdf.