SAE Level 2 ADAS - Controller Design for UW EcoCAR

Naveen Mathan Sundaram

A thesis
submitted in partial fulfillment of the
requirements for the degree of

Master of Science in Mechanical Engineering

University of Washington

2021

Committee:

Per Reinhall

Santosh Devasia

Xu Chen

Program Authorized to Offer Degree:

Mechanical Engineering

University of Washington

Abstract

SAE Level 2 ADAS - Controller Design for UW EcoCAR

Naveen Mathan Sundaram

Chair of the Supervisory Committee:

Per Reinhall

Department of Mechanical Engineering

This work focuses on building a supervisory controller for the UW EcoCAR to incorporate SAE level 2 autonomy features. It comprises Adaptive Cruise Control (ACC), Automatic Emergency Braking (AEB) and Steering Assistance. It uses a separate controller for Velocity Tracking and Distance Tracking. A switching algorithm is used to intelligently shift between the Conventional Cruise Control (CCC) and 'Stop & Go' Adaptive Cruise Control (S&G ACC). Complete control algorithm has been verified in simulation using Model-In-the-Loop (MIL) testing on MATLAB/Simulink. Further, real-time Hardware-In-the-Loop (HIL) testing has been done using MicroAutoBox II (MABx) and dSPACE Simulator using Controller Area Network (CAN) Communication. The main goal is to improve vehicle safety and performance, considering the constraints on acceleration, deceleration and jerk related to passenger comfort.

Table of Contents

Table of Figures

Table of Tables

Acknowledgment

I would like to express my sincere gratitude to my advisor, Professor Per Reinhall for providing his valuable time, continuous support and motivation throughout my master's thesis. His guidance and stimulating discussions helped me carry out my research in the best possible way.

Besides my advisor, I would like to thank the rest of my thesis committee Prof. Santosh Devasia and Prof. Xu Chen for their valuable support and for accepting my request during my thesis preparation.

I am grateful to the University of Washington for giving me a wonderful opportunity to take part in the UW EcoCAR program. I thank the university for offering me Lab support to gain hands-on experience and practically execute my thesis experiments.

My sincere thanks to my PCM teammates Yudong, Yug, Carlos and Aidan who have encouraged and supported me during challenging times. Their insights on the plant model, HIL setup and testing were very helpful in completing my thesis work.

Finally, I would like to thank my family members for supporting me throughout my life.

Dedication

To my parents

Mr. Mathan Sundaram and Mrs. Josephine Rita,

and my sister

Ms. Nisha

I. Introduction

1. Automobile Industry

Major manufacturers in the Automobile Industry are currently transforming to the field of Electric and Autonomous Vehicles. Electric vehicles increase energy efficiency and reduce the carbon footprint. Global Warming is an alarming problem and all countries are working to resolve it. Electrification helps reduce emissions and have sustainable development. Automation increases safety and has the potential to reduce about 94% of the accidents which are caused by human errors according to US DoT [1].

The automotive industry has always been working towards increasing the passenger's safety, convenience and comfort. Advanced Driver Assistance System or ADAS makes driving easier and safer for the passengers. Adaptive Cruise Control (ACC) is an ADAS that enables the ego vehicle (Connected/Automated Vehicle being tested) to automatically control its longitudinal velocity. Depending on the traffic condition, the vehicle maintains a safe distance from a vehicle ahead or maintains the driver set speed limit. This feature decreases the driver workload significantly in certain cases such as monotonous highway driving.

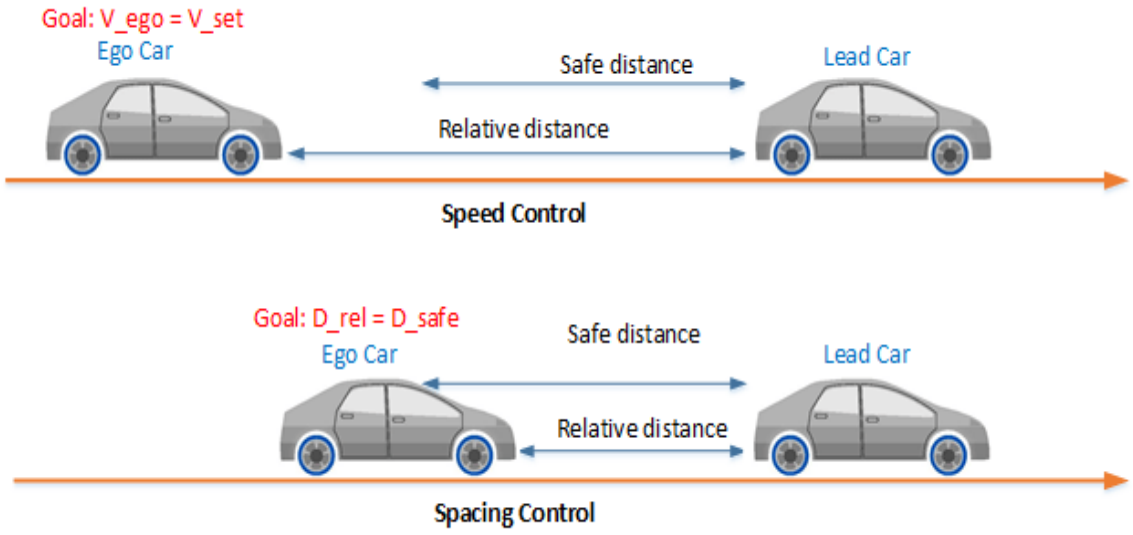


Figure 1: MathWorks, Adaptive Cruise Control System [2]

2. EcoCAR

The goal of the EcoCAR Mobility Challenge is to convert a 2019 Chevrolet Blazer (sponsored by GM) into a fully functional Hybrid-Electric Vehicle (HEV) and improve the safety and efficiency of the vehicle. The University of Washington EcoCAR 4 team comprises of 3 major technical sub-teams – Propulsion System Integration (PSI), Propulsion Controls and Modeling (PCM) and Connected and Automated Vehicles (CAV) in addition to Human Machine Interface (HMI) and User Interaction/ User Experience (UI/UX) [3].

The PCM team focuses on developing a safe and robust software for the vehicle. A broad category of systems include Adaptive Cruise Control (ACC), Energy Management System (EMS) and Power-Splitting between the Internal Combustion Engine (ICE) and Electric Motors. This involves developing a model of the vehicle, building the control algorithm for these systems, testing, verifying, validating and deploying the final algorithm on the vehicle.

Figure 2: UW EcoCAR [4]

3. ACC

This work focuses on developing the ACC algorithm keeping in mind the safety and performance of the control algorithm in adverse situations. It includes developing the controller model, building test cases, MIL and HIL testing. The nonlinear vehicle model is available in Simulink and the controller model is developed for MIL testing. After developing the test cases or driving scenarios, HIL testing is done using the dSPACE platform. The MABx emulates the controller, the Simulator runs the vehicle model and CAN communication is used to pass the signal between them. This setup is used to mimic the vehicle running the control hardware in real-time. The data collected is used to analyze the results.

II. Objective

Many new autonomous features have been introduced in automobiles. In recent years, there has always been concern regarding safety and robustness. This work focuses on building a supervisory controller for the vehicle to provide safe next-gen mobility solutions. All the features such as Adaptive Cruise Control (ACC), Automatic Emergency Braking (AEB) and Steering Assistance, together will qualify as SAE Level 2 Autonomy.

A Literature review in the field of ACC for autonomous vehicles will be conducted to understand the different ACC algorithms available and their evolution. The control architecture will be decided and modified based on new insights from the literature review. The control signals generated will be used to control the engine and motors dynamically.

A Simulink model will be developed with the help of the existing vehicle model as a reference. Safety features such as AEB and necessary constraints such as maximum acceleration, deceleration, jerk for passenger comfort are to be included. The required steering angle will be determined based on the planned trajectory for lateral control.

Driving scenarios will be designed to complete algorithm testing based on ISO standards. A variety of test cases will be covered to ensure systematic validation. The first step of validating the model will be in Simulation or MIL testing. Parameters must be tuned to modify the response of the ACC algorithm on the vehicle as per the requirements. A final check has to be made and determined whether all constraints are satisfied even in extreme cases for increased safety and robustness.

After necessary improvements, HIL testing will be done on MABx and simulator using CAN communication between these hardware components. This setup will be closer to the actual vehicle running with various Electronic Control Units (ECUs) and controlling the whole vehicle system. The response of the real hardware will be verified with real-time conditions such as limited computation power and delays.

The overall control algorithm will be designed, developed, tested, verified, and validated.

III. Literature Review

1. Definition

90% of the accidents occur due to human error such as distraction, poor judgment, or situation awareness [1]. Thousands of people die, millions of people are injured and billions of dollars are lost in crashes every year [5]. Advanced Driver Assistance Systems – ADAS reduce driving errors, improve safety, enhance driving comfort, increase traffic capacity and reduce energy consumption [5].

A crucial component of ADAS is an ACC system which is commercially available in many passenger vehicles. ACC is an extension of Conventional Cruise Control (CCC) that adjusts the reference speed if there is a vehicle in front of it. The ACC algorithm should be able to intelligently shift between user-set velocity and headway time gap. Various range sensors like Lidar, Radar or Camera are added to measure the distance 'd' and velocity 'V' of the lead vehicle with respect to the ego vehicle. The velocity is generally limited to a range from 40 – 160 km/h and maximum braking deceleration of around 0.5 g [5]. The driver always has precedence over the ACC controller to ensure ultimate control of the vehicle.

2. Timeline

The 1st gen ACC systems were included in luxury vehicles to enhance driving comfort and convenience, with a potential increase in safety. But there have always been concerns among consumers, automotive industries and the government. Researchers have categorized issues into 3 types: Human, Traffic and Social. Human issues include driver behavior, user acceptance and human-machine interface. Traffic issues include safety, capacity and stability. Social issues include environmental, marketing and legal [5, 6]. Also, improvement in traffic flow, capacity and fuel consumption is still debatable [6].

ACC systems had been in Japan since 1995, Europe since 1998, North America since 2000. Initially, it started as Adaptive Intelligent Cruise Control (AICC) adopted from the PROMETHEUS program in Europe. The term ACC was then applied by the ISO 15622 [7].

The ACC ECU generally consists of a range sensor and a Controller. This together with the other ECUs for Engine, Brake and Transmission control help the vehicle run as a whole Mechatronic system. Cameras, curve sensors, velocity and lateral acceleration sensors are used to help the vehicle drive along curves and during lane changes [8]. The sensors and ECUs are connected via a CAN Bus to facilitate efficient and error-free communication [8]. The evolution of ACC systems can be broadly categorized into 3 phases – early preparation, intermediate implementation and latest product phase.

3. Phases

Early preparation phase (1950-1985)

In the 1950s, proportional feedback provided full throttle when the vehicle's velocity dropped 6 to 10 mph below set velocity. This caused discomfort and did not improve safety. Automatically Controlled Highway System (ACHS) was developed to deal with traffic congestion in 1966 [5]. By the 1980s, there was a huge increase in the number of vehicles on road leading to more congestion and accidents. The transportation infrastructure could not keep pace with the rising number of vehicles. The increase in driving stress created discomfort for the people. This phase built a strong technical foundation, but due to lack of technology, unavailability of radars that could satisfy performance needs, complete implementation was not possible.

Intermediate implementation phase (1986-2000)

By end of the 1980s, due to an increase in availability, functionality, flexibility, reliability and decrease in cost of microchips [5], Conventional Cruise Control (CCC) became easy to operate, more robust and reliable. Improved communication, computer and sensor technologies helped in the implementation of ACC in different countries.

In Europe, an 8-year program called PROMETHEUS was initiated in 1986 by automotive companies cooperating with suppliers and research institutes to improve their competitiveness all over the world. Urban Drive Control (UDC) project was created to evaluate ACC systems [5].

In Japan, many automotive companies had been working on developing ACC systems under the guidance of Vehicle, Road and Traffic Intelligence Society (VERTIS), created in 1994. Major companies like Mazda, Mitsubishi and Toyota had developed and tested vehicles focussing on Crash Avoidance System under the ASV-1 program [5]. ASV-2 program focused on full-speed ACC system from 1996 to 2000.

In the USA, the California Department of Transportation initiated a study on the automation of vehicles plying on highways. Ford also joined to design ACC controllers. In 1988, Mobility 2000 grouped Intelligent Vehicle Highway System (IVHS) was created to increase collaboration among federal and private players. In 1997, Intelligent Vehicle Initiative (IVI) program was created to prevent highway crashes, fatalities and injuries. In 1999, Automotive Collision Avoidance System (ACAS) was created to assess the performance of rear-end collision warning system [5]. A large-scale ACC system field operations test was performed in Michigan with 10 ACC-equipped vehicles to assess safety, performance, acceptance among users and deployment issues of ACC systems.

Latest product phase (2001-2020)

Automakers introduced ACC in luxury vehicles and are extending this feature to mid-range vehicles. ACC has become a popular marketing term. A sustainable transport system requires an eco-driving style. Different transportation challenges such as limited road infrastructure, driver errors and new emission laws further boosted the progress towards Autonomous Driving. Technical problems have been fixed, but marketing and legal issues are yet to be tackled [5].

Intelligent Transportation Systems (ITSs) and Connected & Autonomous Vehicles (CAVs) have created new opportunities for improvement in ACC [9]. Stop & Go Cruise Control is a more sophisticated variant of ACC and has started gaining importance. With vehicle-to-vehicle communication (V2V), we can have Cooperative ACC (CACC) [9] with more extensive preceding vehicle information. This helps reduce inter-vehicle distance and thereby increases road throughput. Wireless communication links introduce new dynamic subsystems into the control loops. Challenges may arise due to communication uncertainties like time delay, packet loss and errors, which may severely deteriorate driving safety.

Fast and effective update of a hybrid electric powertrain and cruise driving systems requires consistency, collaboration and interoperability. MicroAutoBox provided by dSPACE and Matlab/Simulink is commonly used to implement the control algorithm and test it on the experimental vehicle. Various control strategies are being developed to implement ACC. Common ones include PID Feedback/Feedforward Control, Model Predictive Control (MPC) and Fuzzy Logic Control (FLC) [9].

Proportional Integral Derivative (PID) Feedback/Feedforward Control

PID controller uses the error term, that is the difference between the reference and actual value to track the reference value. In the ACC system:

- Distance error e = actual distance – desired distance = $\Delta x - \Delta x_d$.
- Derivative is $\dot{e}$.
- Acceleration is determined as a function of error $'e'$ and relative speed Δv.

Proportional or proportional-derivative feedback/feedforward control can also be used instead of PID. A detailed review of the PID Controller for ACC will be discussed later.

Model Predictive Control (MPC)

MPC algorithm calculates the current control action by solving an online, iterative, finite-horizon optimization of the plant model. Steps involved in the implementation are:

- Predicts the future states based on the current states
- Computes cost function for the first step of the solved control sequence
- Applies feedback control loop to compensate for predictive errors and model inaccuracies.
- Samples new current plant states to iterate the above process till the final goal is reached.

MPC is generally used for nonlinear kinematic models. An inherent assumption is made that the preceding vehicle will drive at constant yaw rate and acceleration up to time horizon 'T'. The cost function, integrating a weighted sum of squares of three terms, is minimized to determine the optimal control command (desired acceleration of the following vehicle). Weighting factors reduce distance error penalizing excessive acceleration and diminishing relative speed. Desired distance 'r' and control input 'u' is the acceleration of the ego vehicle.

Fuzzy Logic Control (FLC)

FLC can achieve both functions of normal ACC and Stop & Go. S&G ACC systems are designed for lower velocity ranges <40 kmph, smaller deceleration and require better object detection sensors. The inputs are distance error 'e' and relative speed Δv like the PID Controller. The output is a Pulse-width-modulation (PWM) command for throttle position and braking pressure.

Three conditions are used to calculate safety distance. If a preceding vehicle is detected, the safe distance Δx should be estimated based on the preceding vehicle speed V_p and the driver desired time headway t_h.

- When no preceding vehicle is sensed, the safety distance is Δx_{s1}.
- When the driver has specified the desired time headway t_h and speed V_{set} the safety distance Δx_{s2} is their product.
- The final safety distance is defined as $\Delta x_s = \min\{\Delta x_{s1}, \Delta x_{s2}\}$

4. System Design

Designing an ADAS system and implementing it successfully is a huge challenge. The major reason is that each vehicle has a unique dynamic response due to various physical factors like an electric motor and engine torque splitting, transmission, hybrid-electric powertrain management, tires and braking characteristics. Therefore, the parameters of the ACC algorithm must be tuned according to each unique vehicle. The difficulty increases as control strategies adopted by commercially available ACC systems are the closely guarded intellectual property of their industrial developers [9]. Each automotive company develops its own ACC system which is not available to the public. Therefore, there is no uniform standard leading to different dynamic responses for each vehicle manufacturer.

Spacing Policy is used to calculate the desired steady-state distance between the two vehicles. The design of the spacing policy is crucial in building an ACC algorithm. Constant distance, Constant Time Headway (CTH), constant safety factor spacing are commonly used for Personal Rapid Transit (PRT) [9]. Considering feasibility, stability, safety, capacity and reliability, CTH is most commonly used by automakers. The control architecture is generally hierarchical, where the upper-level ACC controller decides the desired target acceleration and transmits to the lower-level longitudinal controller which determines throttle and brake values. Performance is evaluated based on Individual, String and Traffic Flow Stability. Test procedure includes simulation, lab platform and field operation.

The overall problem can be divided into three subproblems. Sensing refers to perceiving states of the surrounding environment and host vehicle. Data Fusion is used to mitigate any sensor inaccuracies. Finally, Control is used to adjust the acceleration command using the improved state estimate. The Low-level controller gives input signals to powertrain components like the internal combustion engine, electric motor and braking system, to track the desired acceleration a_{des}. The dynamics are comparable to a 3rd order model with a time lag τ_l and an actuation delay τ_d.

Control Architecture – PID

Paper 1

Stop & Go Adaptive Cruise Control Systems for Vehicle require a robust control architecture. 2013 paper [10] mentions a controller architecture for a PID control algorithm. Three major inputs of the ACC system are speed of host vehicle, headway time set by the driver and actual gap measured by the radar scanner. The driver sets the desired speed of the car. The actuator

adjusts the throttle position according to the command of the controller. The changes in the throttle position lead to the change in the speed of the car traveling and obtain the desired speed.

Two distinguished loops were used to perform cascade control. The Outer Loop Control (OLC) switches between CCC and ACC mode depending on the distance in front of the vehicle. The Inner Loop Control (ILC) is provided to track the reference velocity calculated by OLC. A simple switching rule coordinates the brake and throttle operations.

Paper 2

In a 2003 paper [11], a Robust Design of PID Based ACC S&G Systems was mentioned. It consisted of 3 separate blocks – Reference Generator, Compensator and Vehicle Inverse Model. The inputs of the system were the speed of the host vehicle V_F, relative distance d_R and relative velocity V_R between the controlled and preceding vehicle. The outputs were throttle opening angle α and brake circuit pressure β. α and β here are analogous to the acceleration pedal value and brake pedal value in the UW EcoCAR Blazer Model.

The reference acceleration $a_{ref} = f(V_R, d_R, V_F)$ is generated based on the precomputed map between α and the target velocity V_{tar}. Measured signal V_F is used for reference generation while the other two acquired signals d_R and V_R are considered for switching between distance and velocity tracking. The ACC Controller has four different modes of operation.

- GO corresponds to transients required to reach subsequent modes.
- CRUISE follows the set target velocity V_{tar}.
- TRACK follows target distance $d_s = s_0 + T_H V_F$, where s_0 is a fixed safety distance and $T_H = d_R/V_F$ is the headway time.
- STOP is used to decelerate the vehicle or during emergency braking.

A simple 1st order transfer function was assumed for throttle command response $G_\alpha(s) = \frac{K_\alpha}{(1+\tau_\alpha s)}$. The brake circuit pressure response was modeled as an integrator $G_\beta(s) = \frac{K_\beta}{s}$. All the parameters mentioned above were found using standard identification procedures for a given step response. The compensator was modeled as a simple PI controller $C(s) = \frac{K_c(1+\tau_c s)}{s}$. The paper demonstrated successful control with the following constraints: Maximum acceleration limited to 3 m/s^2 and maximum deceleration limited to 4 m/s^2.

IV. Research Methodology

1. Linearization:

The initial approach was to check if a simple PID controller can be used to build a controller for the vehicle. The complete Simulink model for the vehicle has been provided by MathWorks and the parameters specific to GM have been updated. The challenge here is that the model is highly nonlinear due to various factors like gear changes, torque splitting between the IC engine and the Electric Motors. A simple approach is to linearize the model and build a controller for the vehicle. The dynamics of the systems are comparable to a 1^{st} order transfer function. The response of the vehicle can be found out for different velocity reference values.

Another challenge here is the operating range of the vehicle. Since linearization works well only around the set operating point, a single linearized model cannot be used to operate the vehicle over the entire span of velocities. The velocity range can be split into 2 to 3 different categories depending on the similarity of the linearized model. A separate controller can be designed for each group of linearized models. A continuous switching mechanism can be used to transition smoothly between the different velocity segments.

The step response for each velocity reference was imported from Simulink to MATLAB. This data was then used to find its 1^{st} order linearized transfer function using the system identification function in MATLAB. A simple PID controller was used to design the Velocity tracking controller. The Proportional, Integral and Derivative gains were initially tuned manually to understand the response of the vehicle. Later, the PID tuner available in MATLAB was used to generate the PID gains. Both the results were compared to see the effect of manual tuning. They were similar to each other which shows that tuning the system manually gives us the same result.

The resulting PID controller was able to perfectly track the lead vehicle's velocity, provided the velocity profile is possible under actuation constraints. There is an issue with implementing a simple PID controller and tuning it. The goal here is to optimize the performance of the controller. Common factors considered while tuning includes Overshoot, Rise time, Settling Time and Steady State Error. Any constraints that must be imposed on the control variable or actuation cannot be incorporated. This limits the capability of the controller.

Adding constraints on the rate of change of the control variable, like acceleration and jerk are not possible with this type of control architecture. Since the dynamics of the vehicle is slow, the control signal generally saturates trying to achieve the reference value as fast as possible. The overall control algorithm works as an On/Off controller. This is not a desired response because it creates passenger discomfort and worsens fuel economy.

A sophisticated control algorithm like Model Predictive Control (MPC) can be used to incorporate the different constraints on the state variables and control variables. All the 3 factors – Safety, Passenger Comfort and Fuel Economy can be taken care of in such an architecture. The major problem with an MPC architecture is that it requires heavy online computation power, and the control update rate is far lower than other comparable control algorithms. This increases the risk in terms of the safety aspect.

2. Control System Design

A novel approach in building a Controller is therefore required to balance the various factors like safety, performance, comfort and efficiency. This section explains the overall process of designing, building and testing the new controller with a unique approach.

Plant Model:

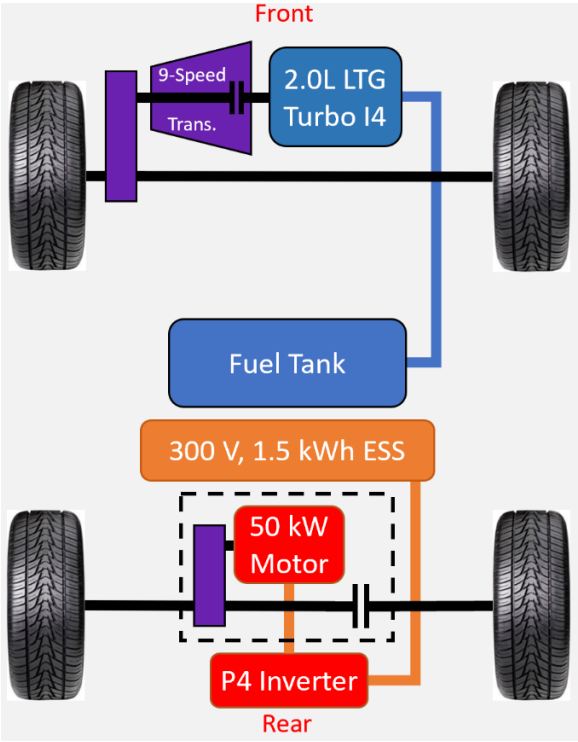


Figure 3: UW EcoCAR P4 HEV [12]

The UW EcoCAR project aims to modify a 2019 Chevrolet Blazer into a hybrid electric vehicle. The main goal of the team is to improve the safety and efficiency of the vehicle. The ADAS features will increase the safety of the system and the Energy Control Management System (ECMS), will help improve the fuel economy. A P4 split-parallel hybrid powertrain architecture has been chosen [12]. The IC engine and electric motors are not directly connected. Both power sources are connected to different axles and drive a separate set of wheels. At UW EcoCAR, the smaller IC engine drives the front wheels, and the traction motors drive the rear wheels.

A P4 hybrid electric vehicle model is available in Matlab [13]. But the dynamics of each vehicle are different, and the parameters must be uniquely tuned according to the modifications made. MathWorks has developed the vehicle model on Simulink and necessary parameters have been modified according to GM's specifications. It consists of different blocks or subsystems connected to form the entire vehicle model. This modular design allows individuals to work on different parts of the model simultaneously.

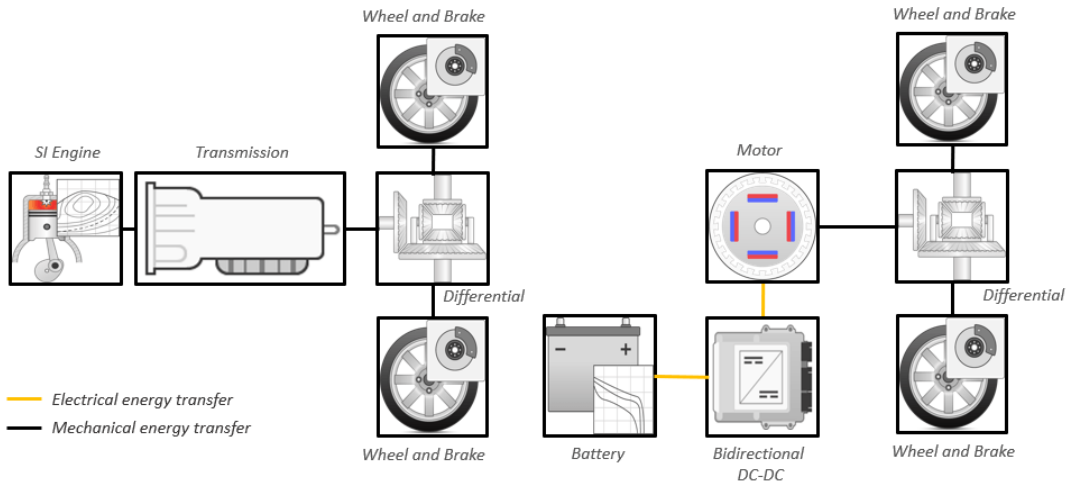


Figure 4: P4 hybrid electric vehicle (HEV) powertrain configuration [13]

The major subsystems include Energy Control Management System (ECMS), Engine Control Module (ECM), Transmission Control Module (TCM), Body Control Module (BCM), Battery Management System (BMS), Motor Control Module (MCM), Hybrid Supervisory Controller (HSC), Electronic Brake Control Module (EBCM) and vehicle dynamics. These control modules control the vehicle dynamics by sending the right control signal to the end actuators like the Spark Ignition IC Engine, Electric Motors and Braking Systems.

ECMS is responsible for managing the torque request generated due to Acceleration Pedal Position (APP) and Brake Pedal Position (BPP). It should distribute the torque required between the IC Engine and Electric Motors efficiently. It works along with HSC to minimize the total energy consumption of the system in the long run. ECM is responsible for controlling the combustion process within the IC Engine depending on the torque requested. APP, BPP, the velocity of the vehicle and the current gear together are used to determine the required engine speed. ECM finally makes the engine reach the requested speed.

TCM is responsible for transmitting the power from the engine to the drive wheels efficiently. It also decides and manages the timing of gear changes required. BCM controls the components connected to the body of the vehicle like power windows, locking system and air conditioning. BMS is responsible for monitoring the critical functions of the battery like maintaining within a range of SOC, charging, discharging and temperature control.

MCM is responsible for varying the motor speed depending on the torque requested. It works as an inverter, converting DC power from the batteries into AC power of varying frequency. The speed of the wheel is directly controlled in proportion to the gear ratio. EBCM distributes the braking torque requested between the four wheels of the vehicle. This is done in addition to the regenerative braking achieved by the motors.

The vehicle dynamics block is modeled as a force acting on the vehicle. It captures the actual response of the vehicle when all the forces/torque act on the vehicle in a combined fashion. When the vehicle is accelerating, the torque from the engine and motors generate a net forward force. Similarly, when the vehicle is decelerating, all the resistive forces from engine braking, regenerative braking, and drag forces cause the braking action.

Control Architecture:

The control architecture is crucial for any control algorithm design. It starts with choosing the inputs and outputs of the controller based on the objective. The goal of the PCM team is to incorporate SAE level 2 autonomy features to improve the safety and robustness of the control algorithm. ACC is a key component that controls the velocity of the vehicle depending on the surrounding situation. Human beings control the vehicle by using the acceleration and brake pedals. They get feedback from the environment necessary to take these control actions. They have a quantitative idea of the distance of the vehicle in front of them and what speed they are traveling at. Eyesight and speedometer are the sensors through which we get this feedback.

Figure 5: ExtremeTech, ACC [14]

For both a human driver and an ACC controller, the velocities of the lead vehicle and ego vehicle are the feedback received from the environment. The velocity of the ego vehicle is controlled using the acceleration and brake pedals. An architecture like this will be easy to understand, visualize, design and implement. In addition to this, the approach mentioned in the PID-ACC literature review [10, 11, 8, 15] have been used to arrive at the optimal implementation.

The whole system can be broadly divided into Plant, Controller and Sensor/Perception model. To test the ACC algorithm, the perception part has been added to the plant model as a simulated drive cycle for the lead vehicle's velocity. The details of the control architecture used in the ACC algorithm are as follows:

Control Variables:

- Velocity of the ego vehicle V_F
- Relative Distance between the lead and the follower d_{rel}

Inputs:

- Velocity of the ego vehicle V_F
- Minimum safe distance d_0
- Velocity set by the driver V_{set}
- Headway time set by the driver t_H
- Relative Distance between the lead and the follower vehicle measured by the Radar d_{rel}

Outputs:

- Acceleration Pedal Position APP
- Brake Pedal Position BPP

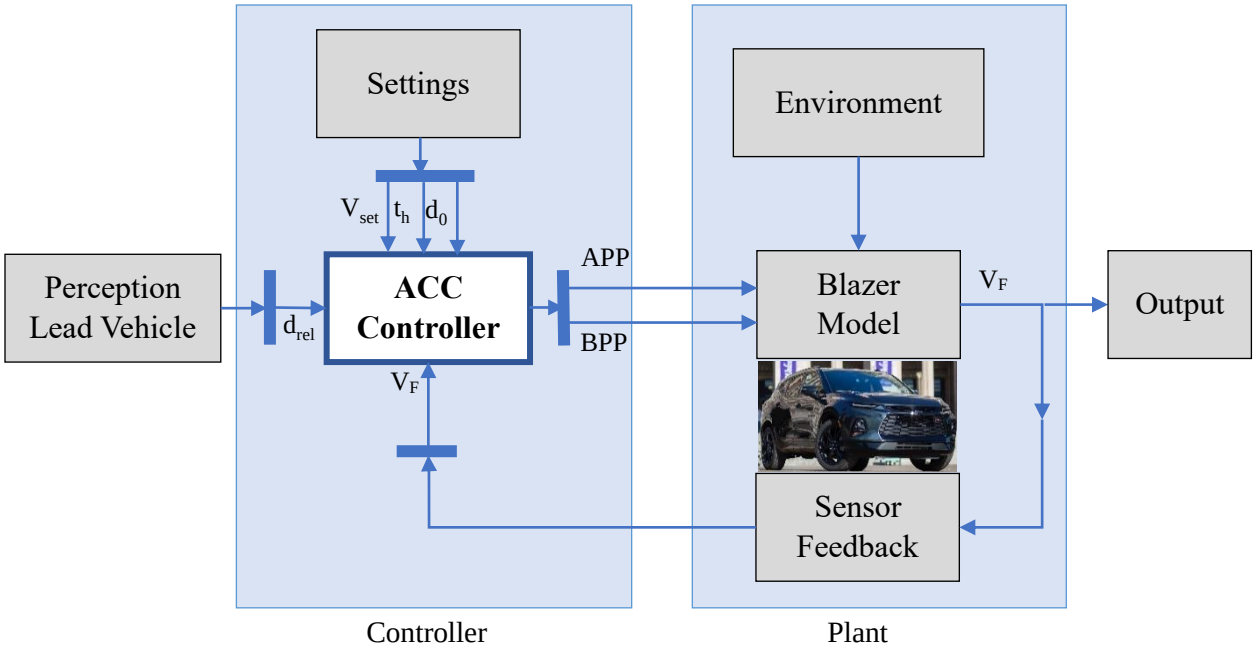


Figure 6: ACC Control Architecture

The controlled variable here is the velocity of the vehicle. Depending on the control scenario it will use the driver set speed or the relative distance between the vehicles to determine the control action. The velocity of the ego vehicle is received as feedback from the plant model. The minimum safe distance can be increased to improve the buffer distance to the lead vehicle. The Cruise Velocity is set by the driver to limit the maximum velocity of the vehicle. It should be less than or equal to the allowed speed limit. The Headway time is also set by the driver. This is used to give the driver, control over the aggressiveness of the ACC control algorithm.

Finally, the relative distance measurement is required for the ACC algorithm to decide whether it must follow the driver set speed or the vehicle in front of it. The data should ideally come from the Perception algorithm which processes the data from the sensors. The Connected and Automated Vehicles (CAV) team is currently working with Bosch Radar systems to develop signal processing and sensor fusion algorithms. These are crucial to get a reliable estimate of the distance from the preceding vehicle. In the development of the ACC control algorithm, Drive Cycles have been designed to mimic the actual sensor data from the perception algorithm. Details about these Drive Cycles will be discussed later.

Most commonly, the ACC controller consists of two parts in series, a higher-level control to generate the velocity or acceleration to be tracked and a lower-level control to track the acceleration or velocity requested. The ACC controller developed here has a parallel control architecture. Two blocks generate the APP and BPP commands independently. An intelligent switching algorithm is used to shift between the two controllers depending on the driving conditions. The outputs of the ACC controller, APP and BPP reach the ECMS. It decides which actuators - Engine, Motors or Brakes and the amount by which they must be engaged depending on the request. APP and BPP therefore indirectly control the velocity of the vehicle.

Preferably the development of ECMS controls should be done independently from the plant model. But the focus of this thesis is to develop the ACC algorithm. So, the ECMS and vehicle model that captures the dynamics have been combined to form the entire plant model. APP and BPP are considered inputs to this plant model. A different group under the PCM team is working on ECMS currently. In the final controller design, ECMS will be part of the MABx controller.

Parameters:

Velocity limit of the ego vehicle V_F generally starts with a non-zero value because the normal ACC does not work for velocities close to 0. The controller designed here is an S&G ACC system. The operating range covers the entire span of velocity the vehicle can reach. The algorithm has been tested for the range of 0 – 33 m/s. This corresponds to 0 – 120 kmph or 0 – 75 mph.

Minimum safe distance d_0 set in the algorithm design is 5m which is equal to a car's length. This can be changed later depending on the requirement. Velocity set by the driver V_{set} can generally be varied in increments of 5 mph. Headway time set by the driver t_H can be qualitatively classified as near, medium, far, corresponding to a headway time of 1, 1.5 and 2 seconds. Relative Distance 'd' between the lead and the follower vehicle is measured by the Radar. The distance to be maintained is given by $d_{des} = d_0 + t_H V_X + t_S V_X$, where the Stopping time $t_S = \frac{V_X}{dec_{max}}$.

APP α and BPP β range between 0 – 1. This corresponds to rest and fully pressed positions, respectively. Based on ISO standard 15622 [7], the maximum comfortable acceleration acc_c= 2 m/s^2 and maximum comfortable deceleration dec_c= 3.5 m/s^2 for a smooth ride. The jerk must be less than 3.5 m/s^3. The maximum possible deceleration for an average car is dec_{max}= 8 m/s^2, for safety reasons a lesser dec_{max}= 6 m/s^2 has been used to calculate t_S.

Simulink Model Development:

The controller model has been built on Simulink to simulate and test the ACC control algorithm. The development of the controller was done by considering the plant model as an isolated system. The whole control system can be categorized into 4 different parts – GM Blazer, Environment, Drive Cycle and Controller blocks.

GM Blazer

GM Blazer is the plant model which broadly comprises the ECMS, a group of control modules and vehicle dynamics block. The details of these blocks have been discussed in the Plant Model section. It calculates the total force acting on the vehicle as a whole and finds the acceleration of the vehicle. Using acceleration, it finds the velocity and displacement. This subsystem takes APP and BPP command, simulates the vehicle dynamics and provides vehicle states - displacement and velocity.

Environment

Environment block is used to emulate the environmental conditions. These include various external factors like temperature, pressure and wind. Apart from these factors, road conditions such as coefficient of friction and road grade changes are also modeled within this block. This can also be considered as a disturbance to the vehicle. During ACC algorithm testing, all these parameters have been kept at zero or in their respective default mode.

Drive Cycles

Drive Cycles are used to emulate the driving behavior of the lead vehicle. In real-time testing of the vehicle, the data should ideally come from the perception algorithm. To conduct MIL and HIL tests in a lab, different scenarios must be manually designed. These Drive Cycles contain velocity data points with timestamps. This discretized data is used to interpolate and generate a plot of the lead vehicle's velocity like a continuous function.

There are two ways to generate a Drive Cycle in Matlab. A function script can be coded to generate the velocity values as a function of time. This can be used to create simple drive cycles which consist of continuous line segments or polynomial functions. But the Drive Cycle we need for testing is way more complex to mathematically generate it depending on time values.

The other way to create a Drive Cycle is by manually designing the velocity values with respect to time. A '.mat' or '.xlsx' file can be used to design the data points and import them into Matlab/Simulink. In this controller model, separate Excel files were used to construct 10 different Driving Scenarios and were imported to the final Simulink model. Based on a Webinar from GM for testing an ACC algorithm [16], 10 different driving scenarios or test cases were manually designed. The description of the Driving Scenarios are as follows:

ACC Testing Driving scenarios summary

1. Acceleration and Deceleration for Open Lane (No Target)
2. Approaching a Slower Moving Target
3. Approaching a Decelerating Target
4. Follow Vehicle to a Stop
5. Resume after a Stop
6. Stop and Go
7. Subject Vehicle Cut-Out to Open Lane
8. Subject vehicle cut-in from open lane
9. Target Cut-In with Subject vehicle in an Open Lane
10. Target cut-out revealing open lane

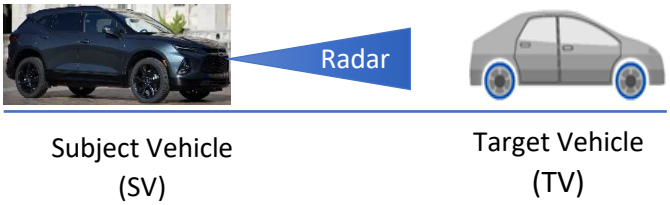


Figure 7: SV following TV

The Ego Vehicle or the Subject Vehicle (SV) is following the Lead Vehicle or the Target Vehicle (TV). The details of the Driving Scenarios are given below:

1. Acceleration and Deceleration for Open Lane (No Target)

The ego vehicle travels at an ACC set speed of 40 mph on a flat straight road without any preceding vehicle. The driver may decrease or increases the set speed to the desired value without an excessive amount of jerk and should reach that value within a pre-determined amount of time. Brake control may be used to reach the deceleration set speed.

2. Approaching a Decelerating Target

Test scenario states that the subject vehicle is traveling at 30 mph. The target vehicle is also traveling at 30 mph. But before the subject vehicle gets into headway, the target vehicle begins to decelerate to a stop. We need to test in all gap settings and different speeds.

3. Approaching a Slower Moving Target

Test scenario states that the subject vehicle is traveling at a speed of 40 mph. The target vehicle is traveling at a speed of 30 mph. The relative distance between them is 150 meters. The subject vehicle approaches the target vehicle and the ACC system detects the target vehicle and begins slowing down. The subject vehicle comes into headway smoothly and maintains constant headway behind the target vehicle. We need to test in all gap settings and different speeds.

4. **Follow Vehicle to a Stop**

The test must be conducted on a flat straight road. The subject vehicle is set to 35 mph. The target vehicle is set at 30 mph. The subject vehicle is settled in headway and is following the target vehicle at 30 mph. The target Vehicle begins slowing down and comes to a complete stop. The subject vehicle comes to a complete stop behind the target vehicle and maintains a comfortable distance to the target vehicle. We need to test in all gap settings and different speeds.

5. **Resume after a Stop**

The test must be conducted on a flat straight road. The ACC set speed is 30 mph. 5 seconds after a complete halt, the target vehicle resumes and accelerates to 30 mph. Just before the target vehicle starts moving, the subject vehicle presses the resume switch. The above test is repeated by using the accelerator pedal to resume. The test should be done in all gap settings.

6. **Stop and Go**

The test must be conducted on a flat straight road. The subject vehicle is set to 40 mph and the target vehicle is set at 35 mph. The subject vehicle follows the target vehicle to a stop. Then Target vehicle creeps forward up to 10 mph and decelerates to a stop and repeats multiple times. The test should be done in all gap settings.

7. **Subject Vehicle Cut-In from Open Lane**

The test must be conducted on a 2-lane flat straight road. The subject vehicle is set to 40 mph and the target vehicle is set at 30 mph. The Subject vehicle changes from the left lane to the right lane using a turn signal such that it does not cut it too close to the Target while changing lanes. When the subject vehicle changes lane and picks up the target, the vehicle should either coast or brake to match the target speed. The transition should be done smoothly and with a minimum jerk and settle into headway. The test should be done in all gap settings.

8. **Subject Vehicle Cut-Out from Open Lane**

The test must be conducted on a 2-lane flat straight road. The subject vehicle is set to 40 mph and the target vehicle is set at 30 mph. The subject vehicle changes from the right lane to the left lane using the turn signal. As the subject vehicle changes lane and accomplishes open lane, the vehicle should accelerate to a set speed of 40 mph smoothly and maintain the set speed. The test should be done in all gap settings.

9. **Target Cut-In with Subject Vehicle in Open Lane**

The test must be conducted on a 2-lane flat straight road. The subject vehicle is traveling in the left lane and the speed is set to 40 mph, the Target vehicle is set at 30 mph. Target vehicle performs cut-in and moves to the left lane such that it does not cut-in too close while changes lanes. The Subject vehicle will pick up the target and will either coast or brake to match the target speed. The transition should be done smoothly and with a minimum jerk and settle into headway. The test should be done in all gap settings.

10. **Target Cut-Out with Revealing Open Lane**

The test must be conducted on a 2-lane flat straight road. The subject vehicle is traveling at an ACC set speed 10 mph faster than the target speed. The subject vehicle is following the Target vehicle. The target vehicle changes from the left lane to the right lane. Once the target vehicle leaves the lane, the subject vehicle will accelerate to set speed. The transition should be done smoothly and reach set speed in an appropriate amount of time. The test should be done in all gap settings and at different ACC set speeds.

Apart from the 10 manually designed Driving Scenarios, a standard FTP75 Drive Cycle and a constant block with zero velocity were used to complete the ACC algorithm testing.

FTP75

FTP-75 is an EPA Federal Test Procedure. It is a Standard Drive Cycle used for emission certification and testing the fuel economy. It consists of Cold start transient, Stabilized and Hot start transient phases [17]. It mimics a combination of city driving scenarios for light-duty vehicles in the US.

AEB

The ego vehicle is at a very high velocity. A constant block with zero velocity is used to simulate a lead vehicle at a complete stop. The ego vehicle detects a lead vehicle at rest in front of it. The ego vehicle should immediately activate AEB mode and brake until it reaches a safe situation. The ego vehicle should successfully come to a complete halt without a collision.

The table below summarizes the details of the driving scenarios:

DS	V_init (mph)	V_set (mph)	d_rel (m)	V_L (mph)	Driving Scenario Description	Total Cases (50)	**Test Cases (14)**
1	**40**	**45, 35**	-	-	Acceleration, Deceleration	2	**2**
2	**40**	**40**	**150**	**30**	Approach Slow target	1	**1**
3	**30**	**30**	**50**, 100, 150	**30**	Approach Decelerating target	3	**1**
4	**30**	**35**	**50**, 100, 150	**30**	Follow to Stop	3	**1**
5	**0**	**30**	**50**, 100, 150	**30**	Resume after Stop	3	**1**
6	**40**	**40**	**50**, 100, 150	**35**	Stop and Go	3	**1**
7	**40**	**40**	**50**, 100, 150	**30**	Cut out to open lane	3	**1**
8	**40**	**40**	**50**, 100, 150	**30**	Cut in from open lane	3	**1**
9	**40**	**40**	**10,** 20, 30, 50	**30**	Target cut in from open lane	4	**1**
10	**40**,35,30	**40**,35,30	**10,** 20, 30, 50	**30**,25,20	Target cut out to open lane	12	**1**
11	**0**	**25**	5	**FTP75**	FTP Drive Cycle	1	**1**
12	30**,** 45**, 60, 75**	30, 45**, 60, 75**	**30, 50**, 100	**0**	Automatic Emergency Braking	12	**2**

Table 1: Driving Scenarios Testing Conditions Table

Totally, a combination of 50 test cases were generated from this table. It was not feasible to test each one of them individually. Only the extreme cases marked in bold contribute to the final set of 14 test cases that have been tested.

Controller Model

Individual Controllers

The ACC controller is being designed to incorporate the features of both CCC and S&G ACC. A separate PID controller was designed for CCC and S&G ACC. The velocity tracking controller corresponds to CCC, and the distance tracking controller corresponds to S&G ACC. The PID values of the controllers were tuned individually because the control variables are different in both these cases. The dynamics of the control variable will vary for each controller. Since one controller cannot be used to control two variables of different types, at least two different controllers must be designed.

The velocity tracking controller takes the V_s set by the driver as the reference velocity. The ego vehicle's velocity is received as a feedback measurement. The difference between these two variables is the PID error term which is used by the controller to determine the control action. The output signal will be positive if the vehicle must accelerate and negative if the vehicle must decelerate. Since both positive and negative values are combined into one single output, it must be separated using two parallel saturation blocks. The positive value will be the APP signal and the negative value will be the BPP signal.

The distance tracking controller calculates the desired distance which the ego vehicle must follow. It is determined based on d_0, t_H, V_X and t_s. Parameter d_0 is inbuilt into the control algorithm, t_H is set by the driver. V_X is the ego vehicle's velocity received as a feedback measurement. Stopping time $t_S = V_x/dec_{max}$ is calculated to add safety buffer distance.

The desired distance $d = d_0 + t_H V_X + t_S V_X$ is calculated and passed on to the controller as the reference distance to be tracked. The relative distance measurement which should come from the Radar is calculated based on the lead vehicle's velocity within the controller. The difference between the desired and measured distance is the PID error term which is used by the controller to determine the control action. Similar to the velocity tracking controller, the output's positive value will be the APP signal and the negative value will be the BPP signal.

Both the controllers were tuned separately. The velocity tracking controller was tuned by checking the response of the vehicle for different reference velocities. Depending on the different characteristics like Overshoot, Rise time, Settling Time and Steady-State Error, the PID gains were adjusted to reach the target velocity. Velocity tracking can be done slowly because there are no safety threats like colliding with a vehicle in front of it. A set of PID values were obtained based on this response.

A similar approach was used to tune the distance tracking controller. But this controller must be more aggressive as compared to the velocity tracking controller. This is because the ego vehicle will be following a lead vehicle moving in front of it and a delay or error in tracking the desired distance might lead to collisions. The PID values were tuned based on the qualitative response for different lead vehicle velocity profiles.

Switching Algorithm

The velocity tracking controller works when there is no vehicle in front of it or when it is very far away. When a vehicle is detected in front of the vehicle and is close to the ego vehicle, the distance tracking controller should come into action. These two controllers were combined using a switching algorithm. The ACC system must switch between velocity tracking and distance tracking depending on the distance from the vehicle it is following.

We can use the same desired distance ‘d’ calculated in the distance tracking algorithm. But there arises an issue. The distance tracking PID controller works based on the principle or error. There must be a difference between the reference and the actual signals for the controller to work in a dynamic situation. Disturbances from the environment will make the actual distance higher than the desired distance. In that case, the switching algorithm will shift to velocity tracking. The car will try to reach the set velocity, but as it increases its speed, the measured distance will become less than the desired distance. The switching algorithm will now shift to distance tracking.

This oscillatory behavior will continue as long as the lead vehicle is traveling at a velocity lower than the driver set velocity. This will cause unnecessary acceleration and deceleration of the vehicle which will lead to passenger discomfort as well as loss in fuel efficiency. The switching algorithm should therefore use a distance that is greater than the desired distance calculated by the distance tracking controller. Since this condition happens only when V_s is higher than the lead vehicle's velocity, we can use V_s to calculate the distance at which the algorithm should switch.

The switching distance $d_{switch} = d_0 + t_H V_S + t_S V_X$ can be calculated. V_S can be used instead of V_X to calculate the headway gap. Now the switching algorithm will work correctly. If the measured distance $d_{rel} \geq d_{switch}$, then it will shift to velocity tracking. Otherwise, it will shift to distance tracking.

One more issue arose during the switching part. Both the controllers kept generating the control signals parallelly. Due to the 'I' term in the PID controller, the errors kept building up when the switching algorithm shifted to the other controller. When the switch took place again, this large error term built-up created an unexpected behavior that creates discomfort to the passengers. Also, these type of sudden actions increases the associated safety risks. The PID controller, therefore, had to be changed into a PD Controller.

3. Control Algorithm Testing

Requirements / Constraints

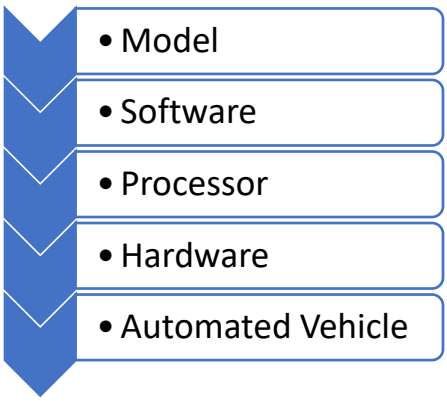
The parameters considered while designing are:

- Relative Distance: ≥ 0
- Relative Velocity:
- Set Velocity: $\leq$ 75 mph
- Headway Time (Near / Medium / Far): 1 / 1.5 / 2 seconds
- Maximum Acceleration: ≤ 2 m/s^2
- Maximum Deceleration: ≤ 3.5 m/s^2
- Jerk for passenger comfort: ≤ 2.5 m/s^3

Model-Based Design

The ACC Controller was built using the Blazer model available on Simulink. This model captures the dynamics of the entire system. Starting the control algorithm design with the model as a reference is called Model-Based Design and development. The requirements of the control system are incorporated during the development process. The controller is generally modeled in simulation using a graphical programming platform like Simulink.

Verification and Validation of the designed algorithm are very much essential for a Model-Based Design. This will ensure that the developed software will work as expected when deployed on the actual vehicle. There are various types of testing available for Model-Based Design [18]. Testing is done in the below order:



Testing categories

MIL: Model-In-the-Loop

The plant model must be developed in a simulation environment like Simulink. It should capture most of the important features of the hardware system. The Controller model must be developed and verified if it can control the Plant. The logic of the control algorithm will be verified.

SIL: Software-In-the-Loop

Once the model is verified in MIL, code must be generated from the Controller model. The Controller block in the model must be replaced with this code. The simulation must be run with the Controller block (which contains the C code) and the Plant. This step verifies that the Controller model can be converted to code and is hardware implementable.

PIL: Processor-In-the-Loop

The Controller model must be flashed on an embedded processor/FPGA board. A closed-loop simulation must be run with the simulated Plant. The Controller must be replaced with a PIL block which will run the Controller code. This will verify that the Processor board can run the developed Control logic without any issues.

HIL: Hardware-In-the-Loop

Once the model has been verified using PIL, the plant model must be replaced with the hardware that simulates the plant. The plant model must run in real-time with proper Analog/CAN communications between the Plant and Controller. The plant simulator is usually a production board/controller. HIL testing will help verify that there are no issues related to communication or propagation delays.

Automated Vehicle Testing

Finally, the controller model is deployed on the hardware that goes on the vehicle. The actual vehicle is the plant here. This test requires the vehicle to be driven in real-world conditions and verified. The vehicle should be allowed to navigate through different scenarios including the ones not tested previously. If it can complete all the tasks as per the requirements, the algorithm has been validated.

At UW EcoCAR, only MIL and HIL are done to expedite the testing process. Once the controller model passes the HIL testing, it can be deployed on the vehicle for real-time testing.

MIL Testing:

Once the model building has been completed on Simulink, testing must be done to verify and validate the developed algorithm in simulation (Model-In-Loop). This model must work considering the various constraints such as headway time, minimum safety distance, acceleration and deceleration to minimize jerk for passenger comfort. Based on a Webinar from GM for testing an ACC algorithm [16], 10 different driving scenarios or test cases were manually designed. Using these drive cycles, tests were conducted to verify the response in MIL simulation.

For each Driving Scenario to be tested, many different parameters like the Initial Velocity V_0, Driver Set Velocity V_s, Initial Relative distance d_{rel}, must be modified. This is a tedious process while testing many different scenarios multiple times while developing the control algorithm. A Matlab script file was written to ease this testing process. All the parameters which must be changed for each Driving Scenario were coded and selected using a switch case condition.

The script file was designed to handle the entire testing process from initializing variables, opening the Simulink model to importing data, plotting and saving them. It can be modified to run only the required Driving Scenarios or different parameters. The script follows the subsequent set of instructions to accomplish this task:

1. Load the variables required to run the Blazer model
2. Open the controller design Simulink file
3. Initialize parameters required to run the controller model
4. Loop through the Driving Scenarios
5. Change the parameters using switch case according to the selected Driving Scenario
6. Modify any other parameters if required
7. Run the simulation for the set time
8. Import the state variables data from Simulink simulation run and store it in a variable
9. Plot the Relative Distance, Lead and Ego vehicle Velocities, Acceleration and Jerk
10. Save the plots for future reference

Depending on the response of the model, the parameters have been modified iteratively. Once all the constraints imposed on the model have been satisfied, MIL testing has been verified. The results from the MIL testing will be discussed later.

HIL Testing:

The next step of testing is to perform the HIL or Hardware-In-Loop testing. This type of testing is closest to the real-time implementation of the controller on the actual vehicle hardware. It helps in verifying that the communication is proper and there are no issues like instability due to propagation delays. The Testing has been done on dSPACE platforms – MABx and Simulator.

Hardware Setup

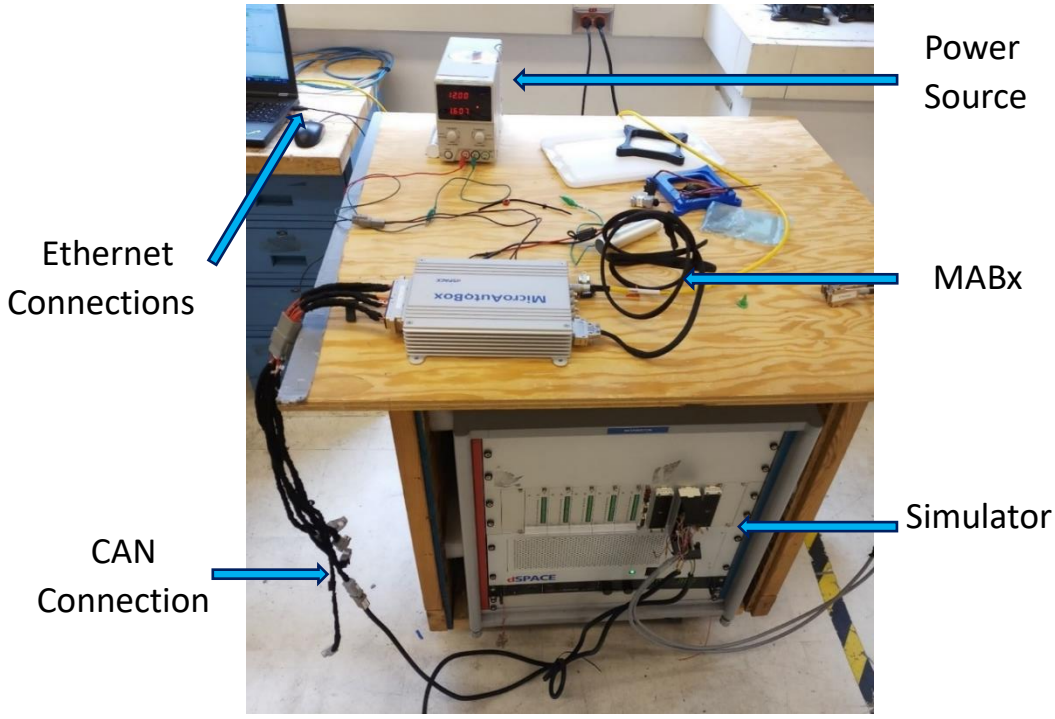


Figure 8: HIL Hardware Setup

The communication interface has been set up between the various hardware. CAN is the most commonly used communication protocol in the automobile industry. CAN is also called a message-based communication protocol because they send messages with identifiers that are used to prioritize their transmission. These messages travel through CAN at a specific frequency depending on the cycle time. All the nodes on the CAN network can receive the message without any issues.

The various hardware has been physically connected for the CAN communication to take place. Two CAN lines, two ethernet cables, and two laptops were used for the HIL testing. CAN 1 from MABx and CAN 2 from Simulator were used for communication between them. MABx was connected to the Control Laptop via an ethernet cable. The simulator was connected to the other Laptop via a different ethernet cable. The communication loop has now been set up completely.

Software Setup

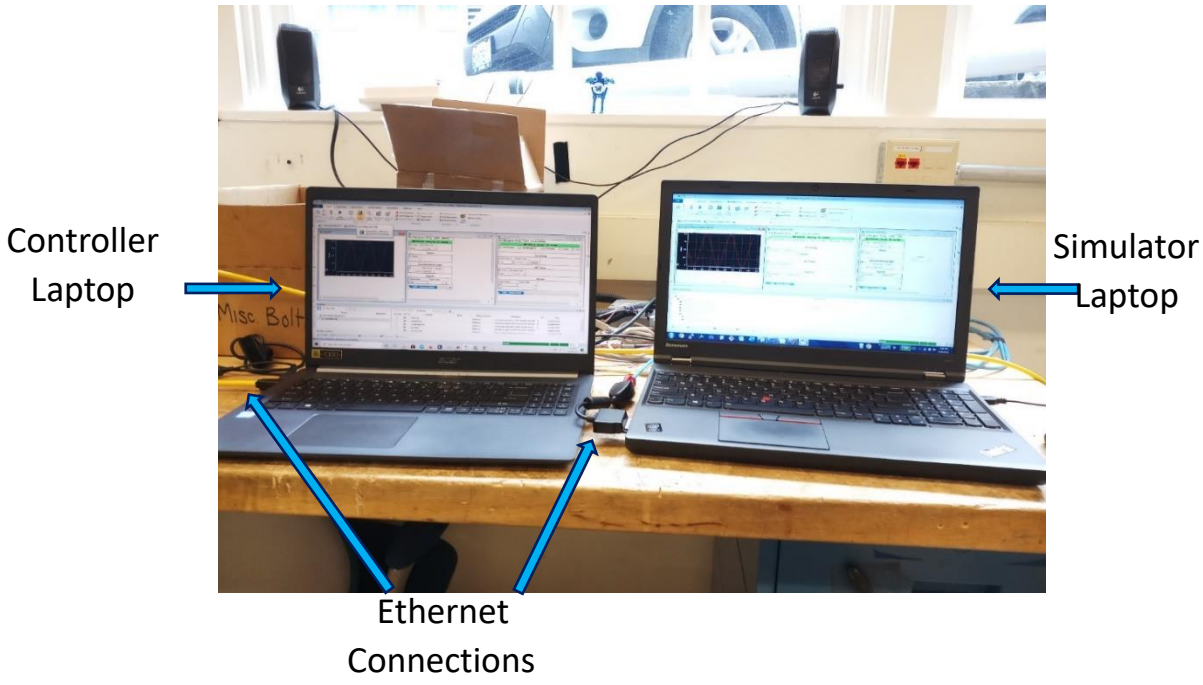


Figure 9: HIL Software Setup

Real-Time-Interface Controller-Area-Network Multi-Message (RTICANMM) Simulink block-sets have been provided by dSPACE to implement the CAN communication interface. Two separate Simulink models have been created for MABx and Simulator. The RTICANMM Controller Setup was used to set up the interface with the processor boards on the hardware. MABx has an RTI1401 board that supports 6 CAN channels. dSPACE Simulator has an RTI1006 board that supports 2 CAN channels.

The RTICANMM Main Block has been set up to link the Simulink model with the Hardware. Messages containing a set of signals have been defined using CANdb++. A DataBase Container (dbc) file is created. It contains the description of various CAN messages and signals within them in a format understood by CAN. The dbc file has been imported into the Simulink model using the Main Block. After all the necessary signals have been selected in the Main Block, an S-function has been created.

The perception data should come from separate hardware developed by the CAV team. To mimic this situation, the designed Drive Cycles have been included within the Simulator model along with the plant model.

The developed Simulink model has been split into Controller and Plant model. A file describing the variables or a System Description File (sdf) was generated during the build process. This sdf file has been imported into ControlDesk to create a layout of the variables required. If the hardware is connected during the build process, the files converted into an embedded C code are flashed on the hardware. The controller model code file will be flashed on the MABx and the plant model code file will be flashed on the Simulator.

When both Hardware and Software setup has been completed, the control algorithm has been tested and data from the simulation has been stored. The data has been processed later and used to plot the various state variables similar to MIL testing. The plots have been compared to verify if the controller is working as per the requirements.

V. Results & Discussion

1. MIL Testing:

The ACC algorithm Simulink model is run for each drive cycle after modifying the necessary variables. Depending on the Driving Scenario, we get different velocity profiles. The data from these velocity profiles are imported into MATLAB workspace from Simulink. All the data received are stored together in a structure variable 'state'. This set of data includes Displacement, Velocity of the Lead Vehicle and Displacement, Velocity, Acceleration, Jerk of the Ego Vehicle. All these data are processed from the Velocity data of Lead and Ego vehicles within the Simulink model.

The required data is then plotted to verify whether the model has performed according to the constraints imposed on the various parameters. The four crucial variables plotted are Relative Distance between the two vehicles, Velocity of both the vehicles, Acceleration and Jerk of the Ego Vehicle. All these plots together provide us with a quantitative idea of how well the ACC algorithm is working.

These plots were also used during the algorithm development process. Once the development process is completed, the same plots are later used to verify that the algorithm works according to the requirements. The results from the 12 different Driving Scenarios (DS) are plotted and explained below:

1) Acceleration and Deceleration for Open Lane (No Target)

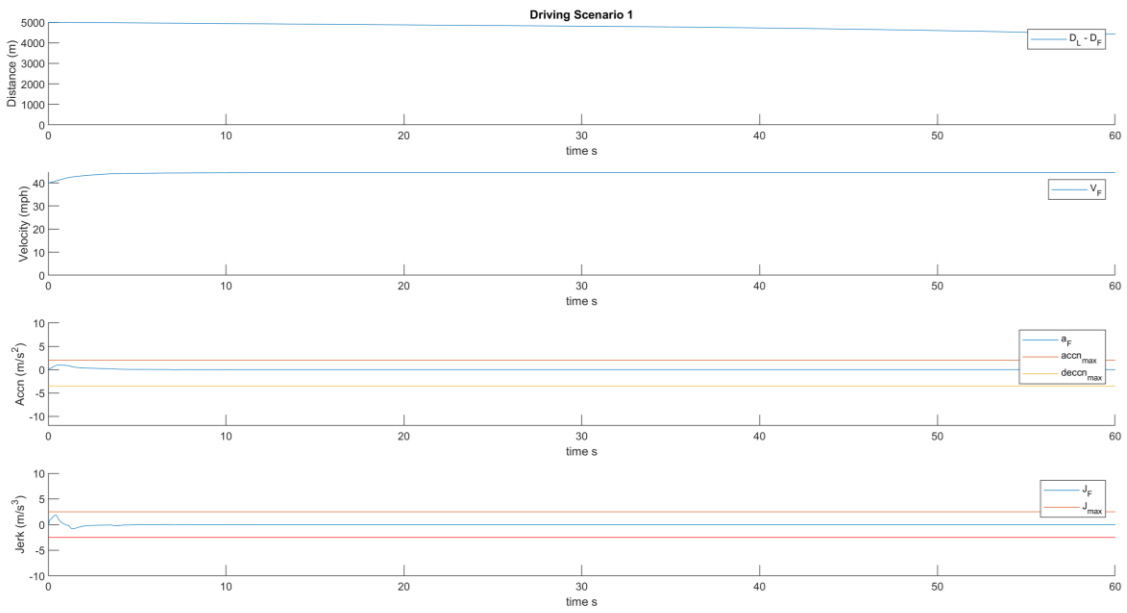


Figure 10: DS 1_1

The ego vehicle is initially at 40 mph. The driver changes the set velocity 5mph higher (45 mph) or lower (35 mph). Since there is no vehicle in front of the ego vehicle, it stays in the velocity tracking mode and tracks the driver set velocity smoothly.

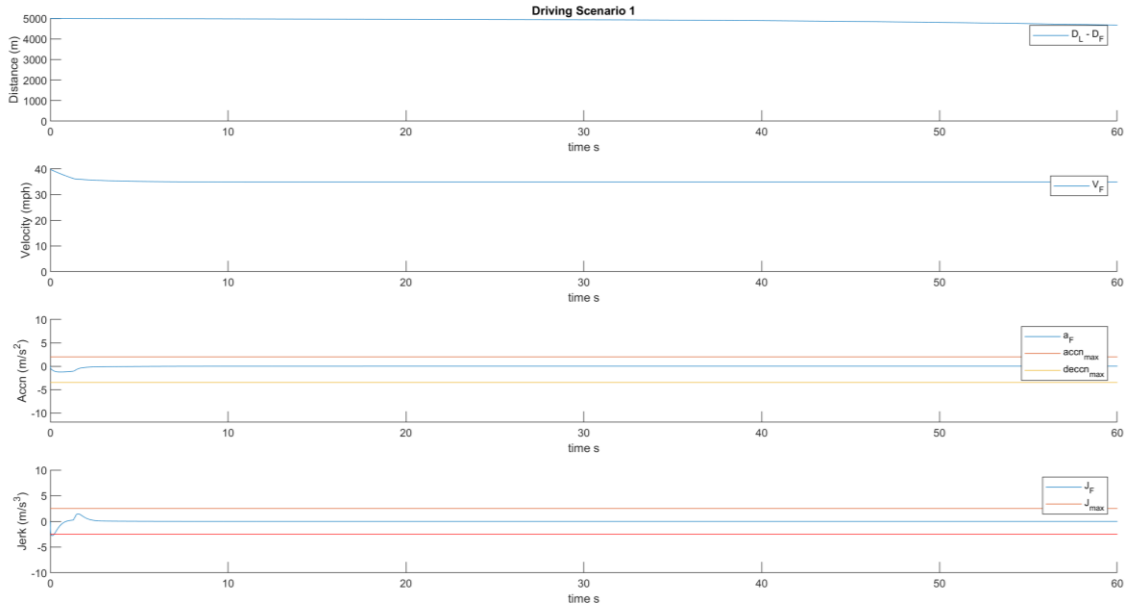


Figure 11: DS 1_2

2) Approaching a Slower Moving Target

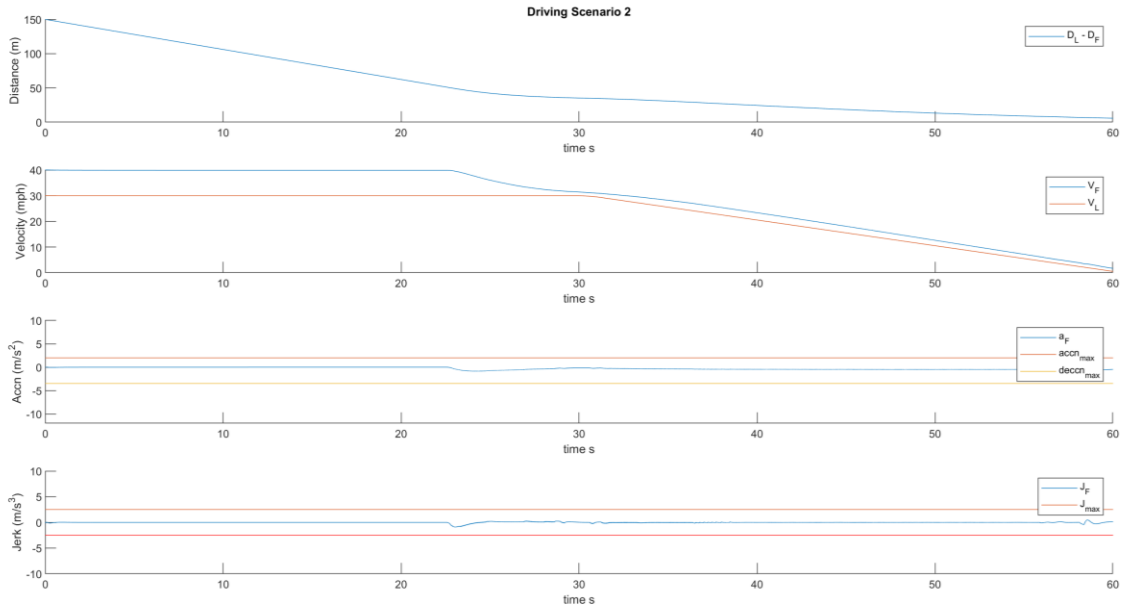


Figure 12: DS 2

The ego vehicle is initially at 40mph in velocity tracking mode. The distance of the lead vehicle goes below the distance calculated in the switching algorithm, so it shifts to distance tracking mode. The ego vehicle slows down to the velocity of the lead vehicle and keeps following it.

3) Approaching a Decelerating Target

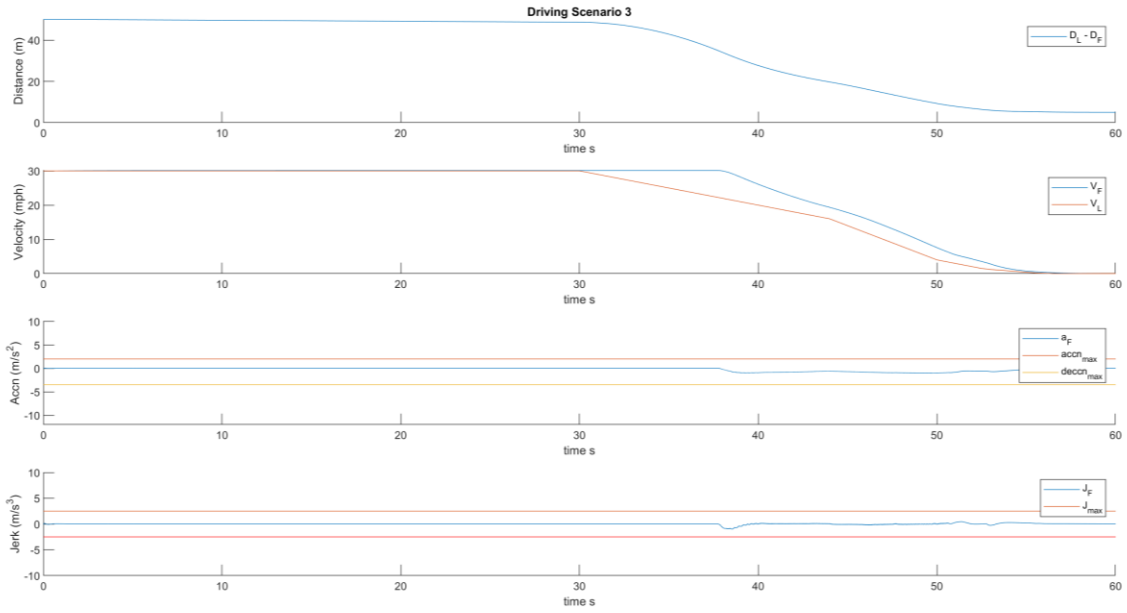


Figure 13: DS 3

The ego vehicle is initially at 30 mph set velocity in velocity tracking mode. It shifts to distance tracking mode and follows the lead vehicle. When it starts decelerating, the ego vehicle slows down similar to the deceleration of the lead vehicle and keeps following it.

4) Follow Vehicle to a Stop

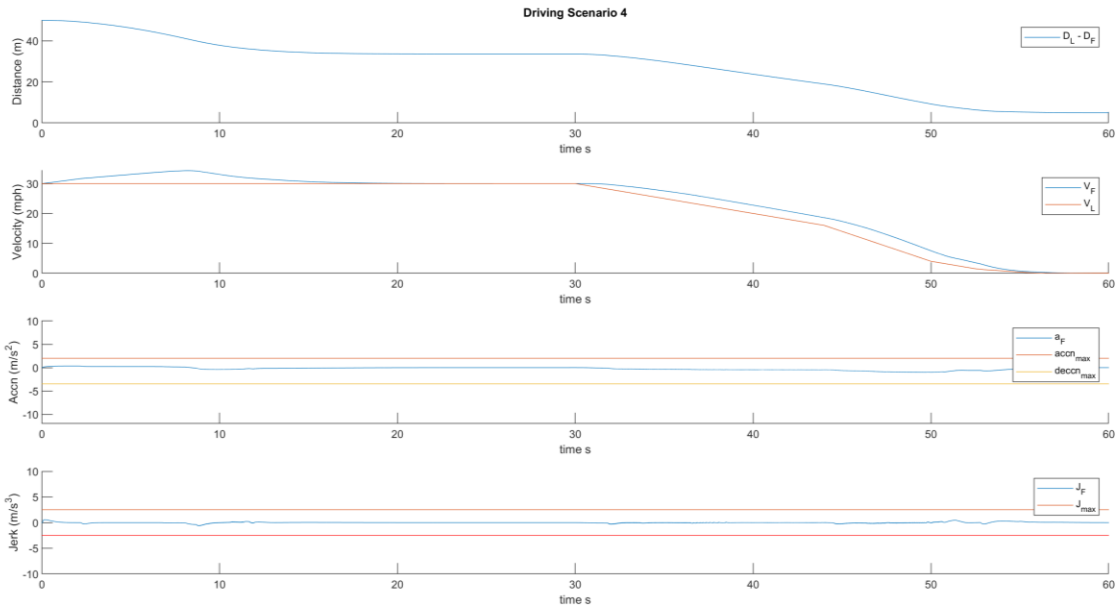


Figure 14: DS 4

Similar to the previous case, the ego vehicle is initially at 30 mph in velocity tracking mode. But the set velocity is 35 mph, so it initially starts accelerating. When it approaches the decelerating lead vehicle, it shifts to distance tracking mode. The ego vehicle slows down similar to the deceleration of the lead vehicle and keeps following it. By the end of the drive cycle, both vehicles come to a complete stop.

5) Resume After a Stop

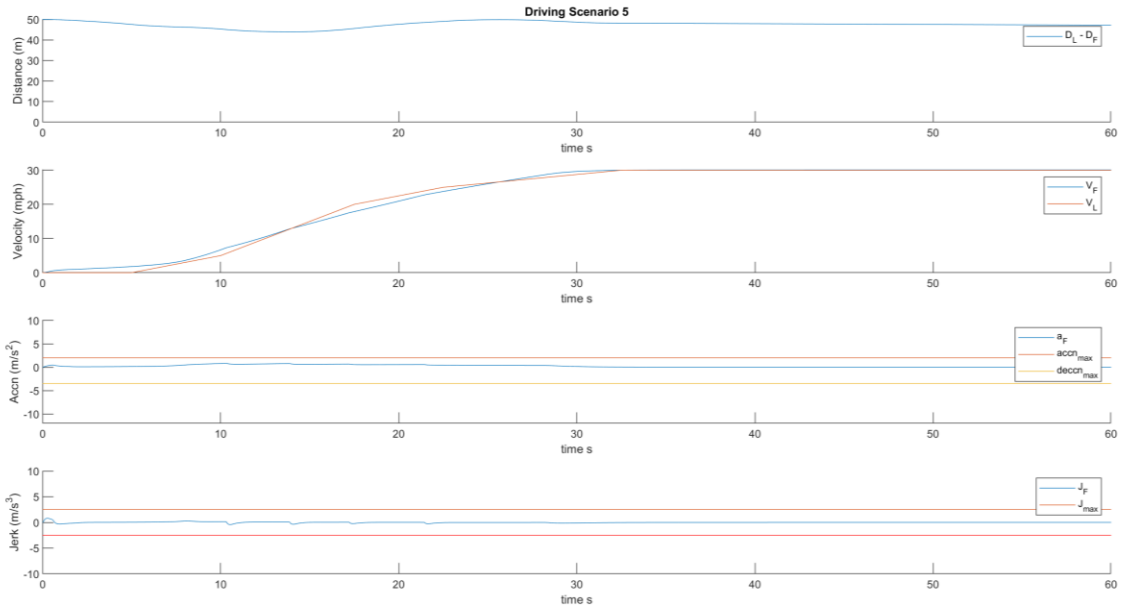


Figure 15: DS 5

This is the next step after the previous case. The ego vehicle is initially at rest. The lead vehicle starts accelerating to 30 mph. The ego vehicle starts following the lead vehicle in distance tracking mode. The distance between them gradually increases and reaches a steady state when both start traveling at the same velocity.

6) Stop and Go

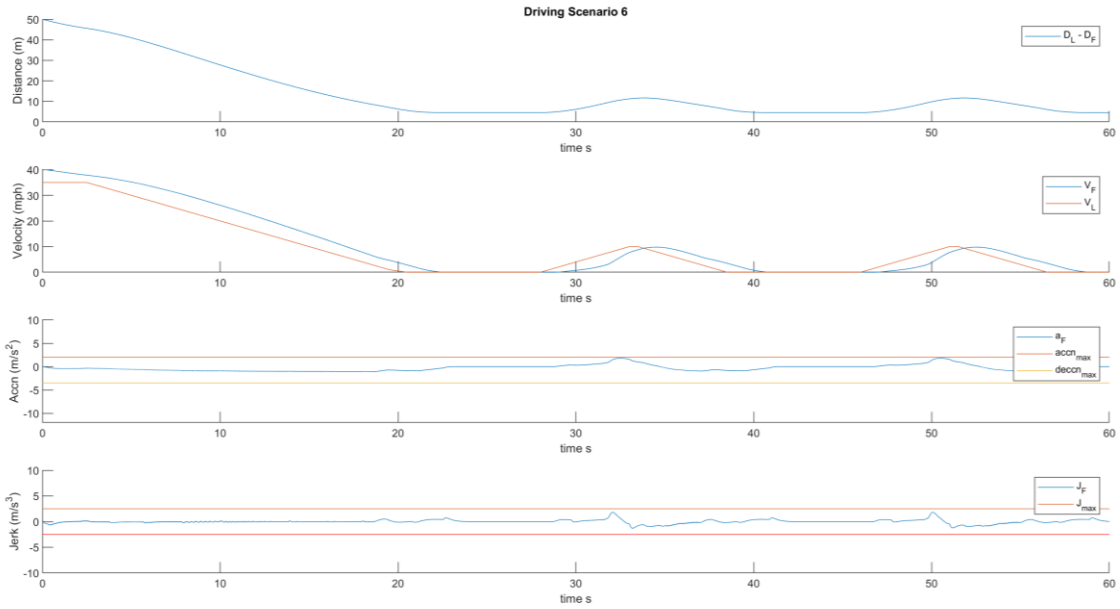


Figure 16: DS 6

This Driving Scenario is a combination of the last 3 cases. The ego vehicle is initially at 40 mph in distance tracking mode and a slower lead vehicle decelerates to stop. After few seconds, it accelerates to 10 mph and stops again. This cycle is repeated two times to mimic vehicles traveling in traffic. The ego vehicle should always follow the lead vehicle smoothly.

7) Subject Vehicle Cut-Out to Open Lane

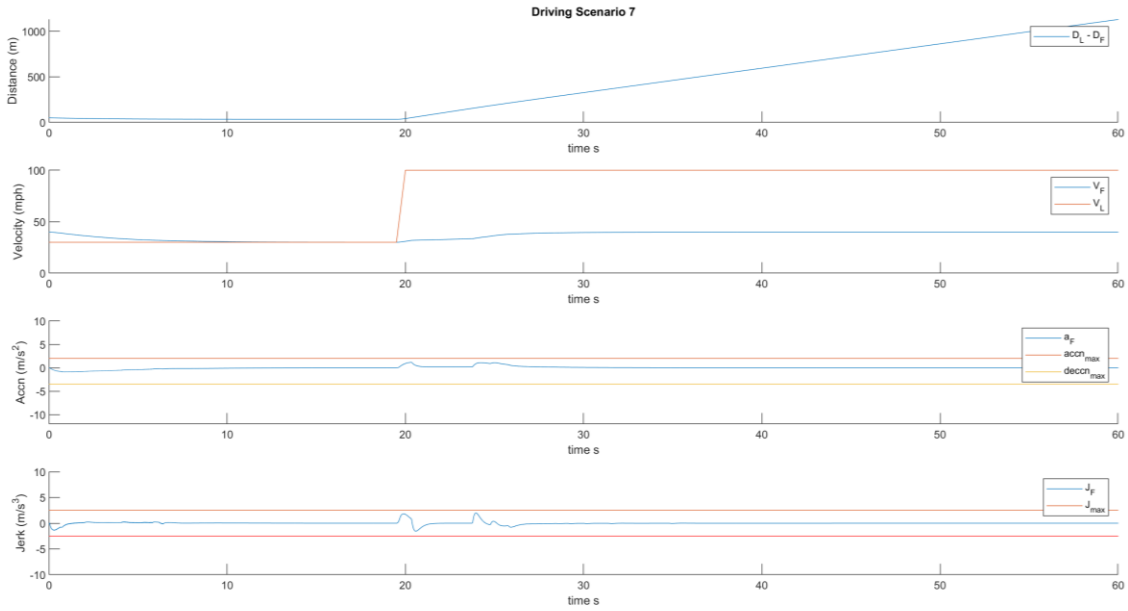


Figure 17: DS 7

The ego vehicle is initially at 40 mph and starts following the lead vehicle at 30 mph in distance tracking mode. It moves to the open lane to overtake the slower vehicle in front of it. Since the controller model developed can only handle data from one lane, the lead vehicle velocity is set to 100. This mimics the situation where there is a sudden increase in the relative distance measurement similar to the open lane situation.

8) Subject Vehicle Cut-In From Open Lane

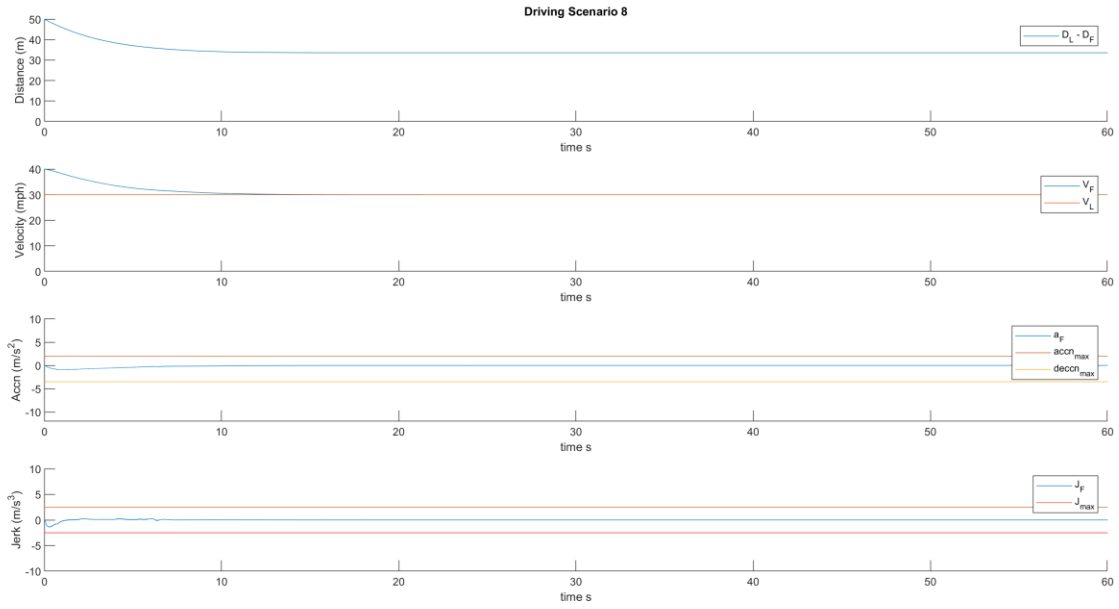


Figure 18: DS 8

The ego vehicle is initially at 40 mph in velocity tracking mode in an open lane. It moves into a lane and starts following a slower vehicle in front of it. Just before t=0, the distance measured will be safe to continue. At t=0, the distance measurement should detect a vehicle. This Driving Scenario tries to mimic this situation where there is a sudden detection of a lead vehicle similar to a cut-in from an open lane situation.

9) Target Cut-In With Subject Vehicle in Open Lane

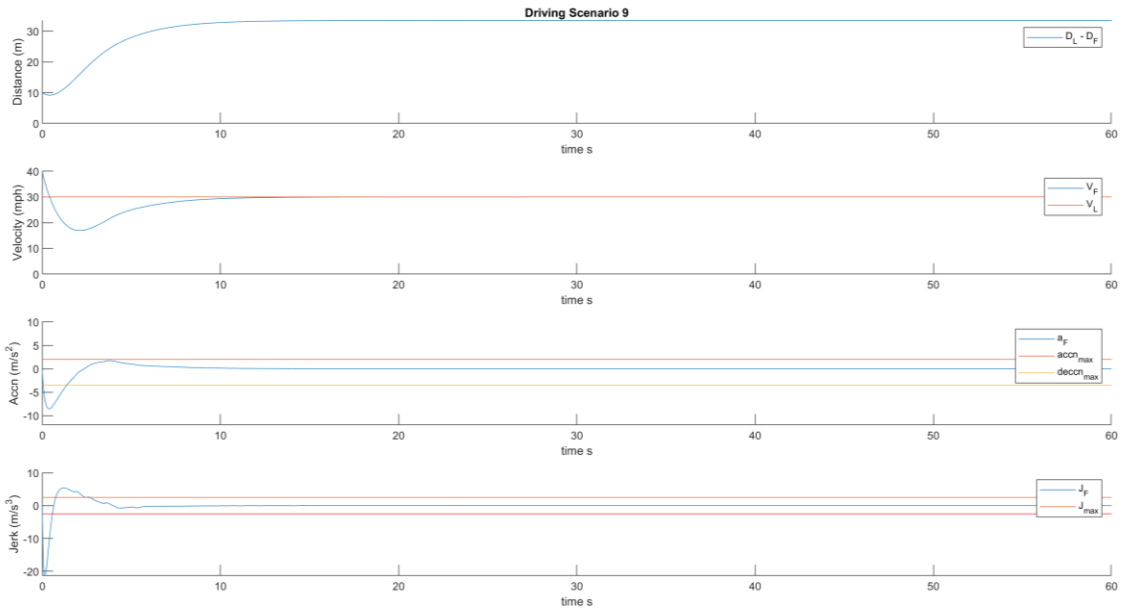


Figure 19: DS 9

Although the scenario is completely different, the lead velocity data is similar to the previous case. The ego vehicle is initially at 40 mph in velocity tracking mode in the open lane. A slower vehicle moves into the open lane just in front of the ego vehicle. At t=0, there is a sudden detection of a vehicle. Since the distance measured is very less compared to the desired distance, the vehicle goes into Automatic Emergency Braking (AEB) mode for a few seconds. Once the distance measurement is safe it goes back to distance tracking.

10) Target Cut-Out Revealing Open Lane

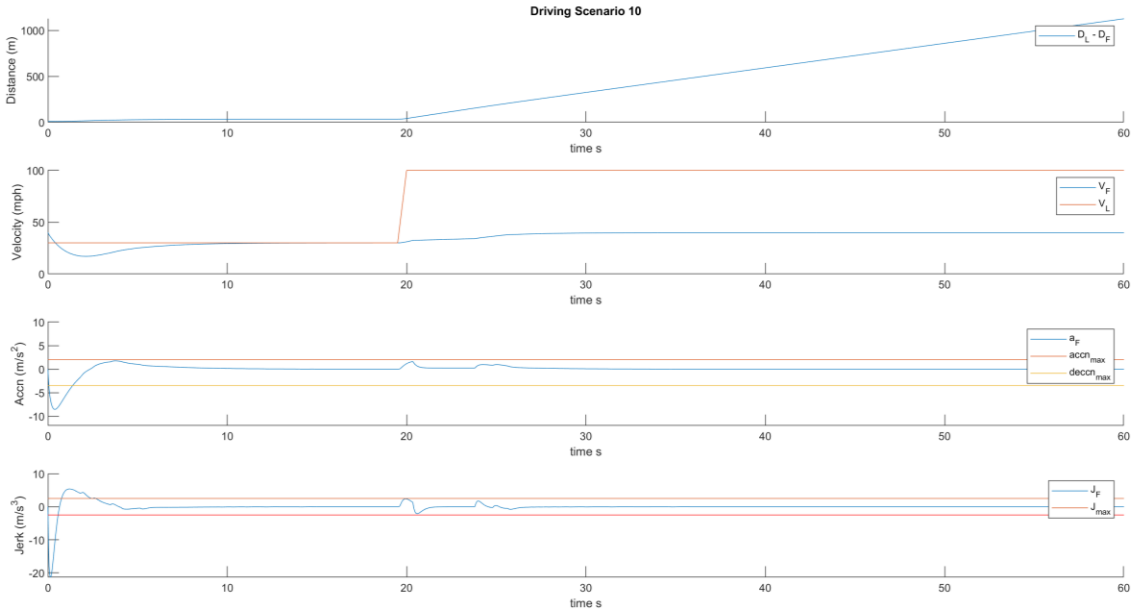


Figure 20: DS 10

The lead velocity data for this case is similar to the previous Driving Scenario 7. The ego vehicle is initially at 40 mph in velocity tracking mode. It detects a lead vehicle and shifts to distance tracking mode. While trying to follow the lead vehicle, it suddenly moves to the open lane leaving the current lane open. The lead vehicle velocity is set to 100 to mimic this situation where there is a sudden increase in the relative distance due to the open lane. The ego vehicle goes back to velocity tracking mode once the lead vehicle leaves the current lane.

11) FTP75

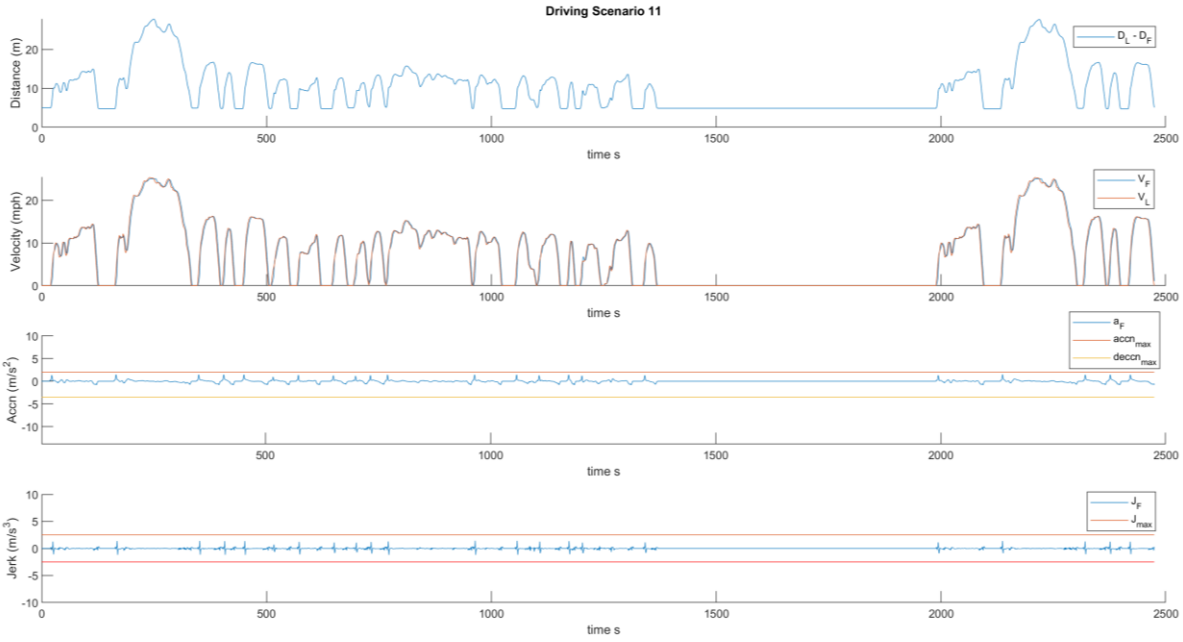


Figure 21: DS FTP75

FTP-75 is a Standard Drive Cycle used which mimics a combination of city driving scenarios for light-duty vehicles in the US. From the plot, we can clearly see that the ACC algorithm can track the lead vehicle perfectly. All the transitions are smooth and stay within the constrained parameters.

12) Automatic Emergency Braking (AEB)

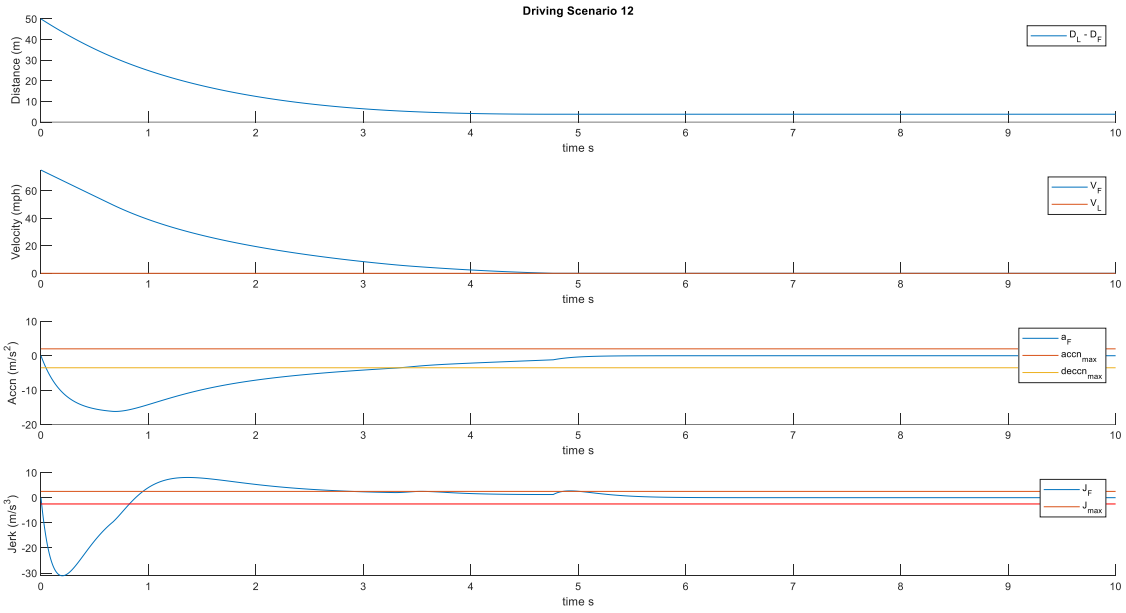


Figure 22: DS AEB_1

The ego vehicle is at a very high velocity of 75 mph. It detects an object at rest, 50m in front of it. The ego vehicle immediately activates AEB mode and brakes till it reaches a safe distance. The acceleration and jerk constraints are not considered here due to an emergency. The ego vehicle can successfully come to a complete halt without a collision.

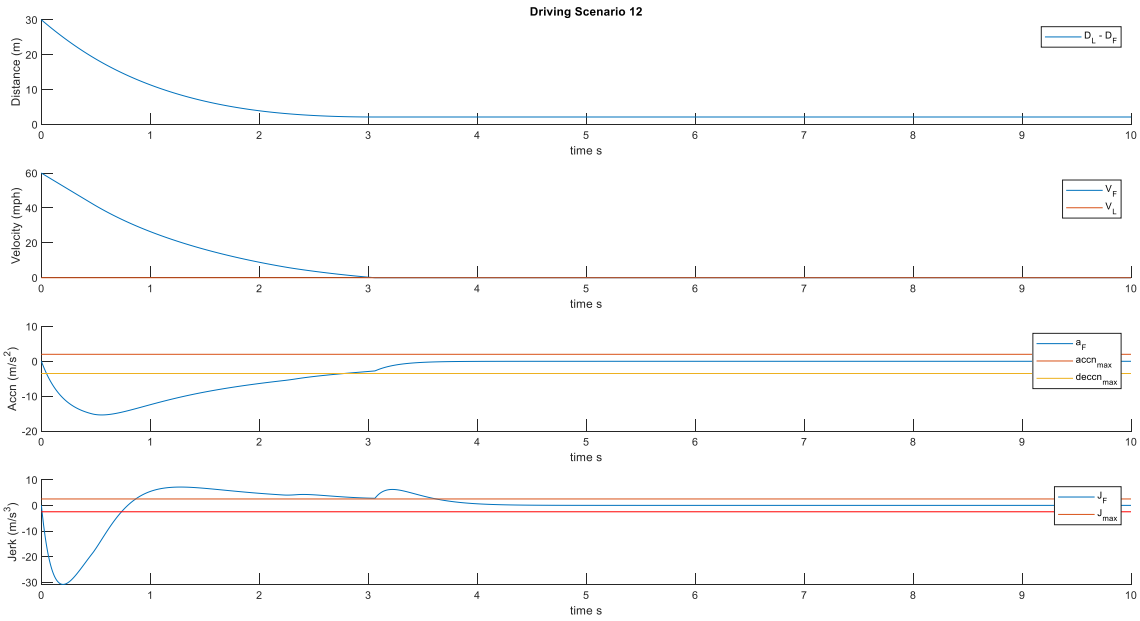


Figure 23: DS AEB_2

The ego vehicle is at a high velocity of 60 mph. It detects an object at rest, 30m in front of it. Although the velocity is slightly lower than before, the distance has been reduced to check the controller's effectiveness. The ego vehicle immediately activates AEB mode and brakes like the previous situation. The ego vehicle can successfully come to a complete halt without a collision.

2. HIL Testing:

HIL testing has been done using the Controller model on MABx and Blazer model on Simulator. The hardware communication has been established through CAN. Since only one license is available to flash the code on the hardware, there is always a delay in the propagation of the right control signals. The starting data acquired from the simulation will always have errors.

To overcome this situation, the drive cycles designed were set to run in a cyclic mode. The simulation has been kept running in a loop. The data from the second run of the drive cycle has been captured. The issue here is that the simulator will consider the initial velocity command only for the first run of the cycle. When the drive cycle restarts again, the ego vehicle velocity starts with the velocity value at which it was at the end of the previous drive cycle.

New CAN signals were added to the communication bus to ensure that the initial velocity command was being continuously transmitted to the simulator. But this error could not be rectified because the initial velocity set in the simulator works only when it is switched on for the first time. The initial velocity signal received by the simulator later does not affect the simulator velocity. Therefore, only the drive cycles which start and end at the same velocity will be discussed here.

The data from the simulation has been stored using ControlDesk. This data has been exported into a file readable by Matlab. The data was then loaded into Matlab workspace. This data has then been processed similar to the MIL testing done previously using Simulink. The plots below are obtained after the data has been processed on Simulink.

FTP75 - MIL vs HIL Comparison

MIL test plot for 500 s

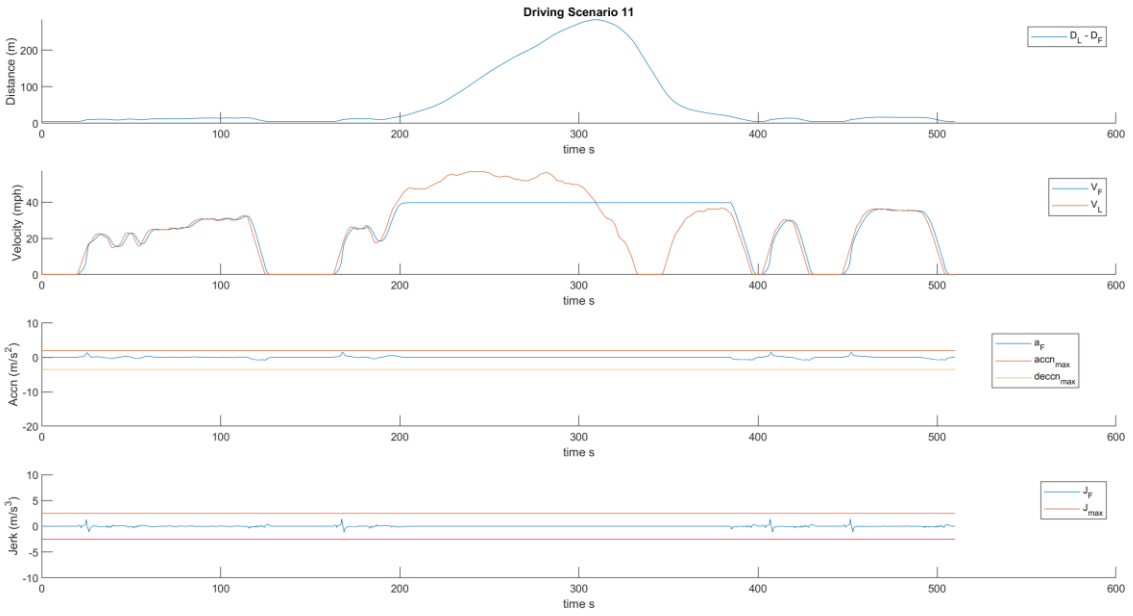


Figure 24: DS FTP75_MIL

HIL test plot for 500 s from cyclic data

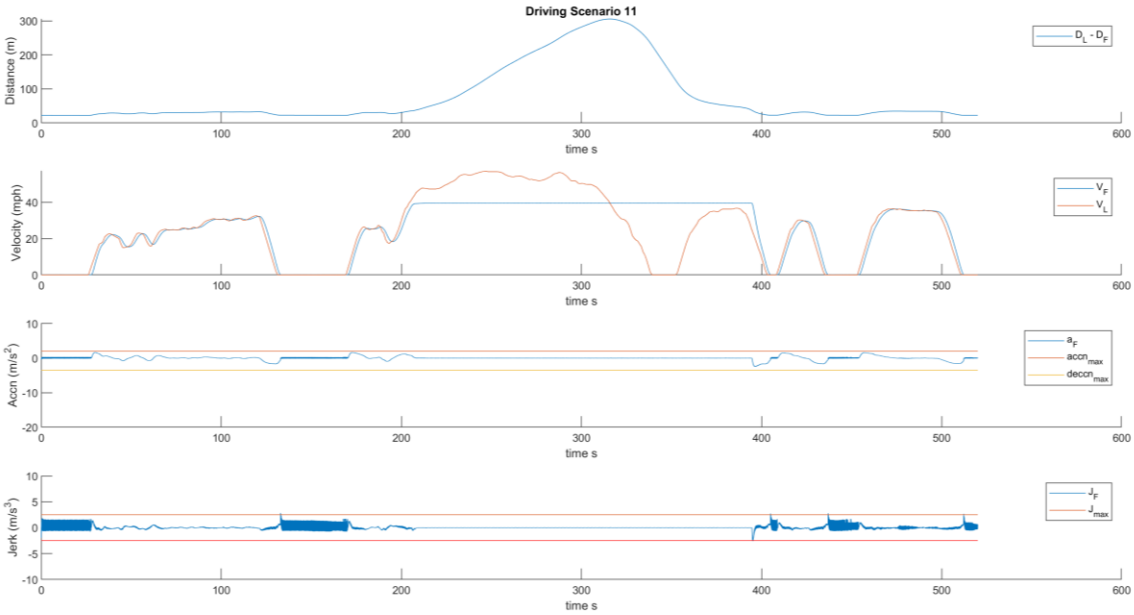


Figure 25: DS FTP75_HIL

Here the ego vehicle is initially at rest and has been set to a lower speed of 18 mph. It follows the lead vehicle until the velocity is below the set limit. When the lead vehicle exceeds the speed limit, the ego vehicle shifts into velocity tracking mode. Once again when it detects the lead vehicle within the range, it shifts back into distance tracking mode.

In the MIL and HIL plots above, there are a few minor variations in the velocity profile. They are not the same. These discrepancies were expected due to the inherent nature of HIL testing. They are prone to have noise in the data due to communication delays, errors and limited precision in data, that is the number of bits by which the data is represented in the CAN signal.

Due to the discretization of data, there is a high frequency added due to the fluctuation of values between the representable values. This issue can be resolved to a certain extent by increasing the precision by using more bits to represent the data. This comes at a cost of using a higher bandwidth for communication through CAN. This trade-off will be required when many signals are trying to simultaneously pass through the CAN network.

But as a whole ACC control system can track the lead vehicle perfectly. the acceleration, deceleration and jerk values are still below the constrained limit. The ego vehicle can start, follow, stop and resume again which corresponds to S&G ACC. We can therefore conclude that the HIL testing has been completed successfully and the designed ACC algorithm has been verified.

VI. Conclusion

The objective of this thesis work was to build a supervisory controller for the UW EcoCAR to provide safe next-gen mobility solutions. The focus of this work was to improve the safety and performance of the control algorithm. ACC, AEB and Steering Assistance together qualify as SAE Level 2 autonomy. ACC and AEB features have been implemented here, taking care of the necessary constraints on Acceleration, Deceleration, Jerk and Actuators.

The control architecture and Simulink model have been designed and developed to implement the control algorithm. Various Driving Scenarios have been designed to complete testing based on ISO standards. The ACC controller has been verified using MIL testing.

Necessary parameters have been tuned in the controller model and deployed on MABx. The plant model has been deployed on the simulator. HIL testing has been done using CAN communication between these hardware components. The computation power of the MABx was sufficient to handle the control load. The response of the model deployed on real hardware has been verified with real-time conditions such as measurement noise and communication delays.

This setup mimics the actual vehicle running different Electronic Control Units (ECUs) using CAN communication and controlling the vehicle as a whole system. Both MIL and HIL testing results show that the designed ACC algorithm can handle a variety of driving scenarios. Hence it is concluded that the entire control system design, development, testing and verification process has been successfully completed.

VII. Limitations

The Blazer plant model used here simulates the vehicle dynamics based on the net force acting on the vehicle. The brake force applied is considered as a backward force acting on the vehicle. Therefore, when the brake is applied after the vehicle stops, it starts moving backward. A switch block has been added to prevent braking action once the ego vehicle stops. This works perfectly on flat roads, but when the road is uphill or downhill, the vehicle starts to slide. The controller counters this action, which causes unnecessary oscillation creating passenger discomfort. The plant model can be modified to tackle this issue.

During initial MIL testing, the control signals to the actuators saturated and behaved as an On/Off controller. Acceleration Pedal Maps and Brake Gains were included to keep issues under control. A different algorithm can be designed to solve this issue completely.

The switching mechanism sometimes tries to be aggressive in the distance tracking mode. The ego vehicle continues following the lead vehicle even after the driver set speed limit is reached. Another block has been introduced to limit the actuation signal. A better switching algorithm can be designed to ensure an appropriate shifting process.

HIL testing had issues like initializing the velocity, delay in transmission and data acquisition. Hence, all the HIL tests planned earlier could not be performed to validate the control algorithm completely.

The steering Assistance feature could not be developed due to time constraints. The plan was to use a reference path profile consisting of (x, y) coordinates in a plane. The steering angle can be calculated based on the turning radius and the longitudinal velocity of the vehicle which limits the lateral acceleration. Steering Assistance can be incorporated within the ACC algorithm developed here.

VIII. Future Work

The last step in Model-Based Design is completing the Automated Vehicle Testing. The refined ACC control module can be deployed on the actual vehicle and tested in a real-time driving environment. This will ensure that the ACC control algorithm developed can handle unexpected situations that might not have been covered during the testing process.

Roads generally have multiple lanes. Vehicles equipped with an ACC system must be able to navigate through different lanes depending on the situation. Automatic lane changes using blinkers can further increase the driver's comfort and safety. The perception data from other lanes can be used by the vehicle to safely maneuver smoothly.

The lead vehicle's velocity can also be used in the desired distance calculation under distance tracking. In cases where the relative distance is less than desired and the lead vehicle is traveling faster than the ego vehicle, the ACC system can prevent unnecessary braking actions. This can further increase the driver's comfort by minimizing the extent of braking.

Data from other sensors such as GPS can be used to change the behavior of the ACC system. Using the current location of the vehicle, the maximum speed limit can be determined by overruling the driver set limit. This feature can increase the driver's comfort and safety.

Real-Time HD Maps can be used to update traffic situations in real-time. An alternate path can be planned if there are congestions or issues observed in the current route. This can save traveling time and reduce stress for the passengers.

Vehicle to Infrastructure (V2I) communication like traffic signals can be utilized to minimize the amount of acceleration or braking required. Optimized control actions can increase driver comfort and fuel efficiency.

Vehicle to Vehicle (V2V) communication can provide reliable velocity data from the lead vehicle. This can optimize control actions and implement Cooperative Adaptive Cruise Control (CACC). The headway gap between the vehicles can be reduced to achieve energy efficiency and increase traffic capacity.

IX. Bibliography

[1] "NHTSA, Automated Vehicles for Safety".

[2] "MathWorks, Adaptive Cruise Control System".

[3] "EcoCAR, About the Team".

[4] "UW EcoCAR".

[5] "Xiao, L., & Gao, F. (2010). A comprehensive review of the development of adaptive cruise control systems. Vehicle system dynamics, 48(10), 1167-1192.".

[6] "Vahidi, A., & Eskandarian, A. (2003). Research advances in intelligent collision avoidance and adaptive cruise control. IEEE transactions on intelligent transportation systems, 4(3), 143-153.".

[7] "ISO 15622:2018(en) Intelligent transport systems — Adaptive cruise control systems — Performance requirements and test procedures".

[8] "Winner, H., Witte, S., Uhler, W., & Lichtenberg, B. (1996). Adaptive cruise control system aspects and development trends. SAE transactions, 1412-1421.".

[9] "He, Y., Ciuffo, B., Zhou, Q., Makridis, M., Mattas, K., Li, J., ... & Xu, H. (2019). Adaptive cruise control strategies implemented on experimental vehicles: A review. IFAC-PapersOnLine, 52(5), 21-27.".

[10] "Sivaji, V. V., & Sailaja, M. (2013). Adaptive cruise control systems for vehicle modeling using stop and go manoeuvres. Int. J. Eng. Res. Appl, 3, 2453-2456.".

[11] "Canale, M., & Malan, S. (2003). Robust Design of PID Based ACC S&G Systems. IFAC Proceedings Volumes, 36(18), 333-338.".

[12] "Jain, Y. M. (2020). Comparison of Model-Predictive control and PID control for Adaptive Cruise Control of UW EcoCAR vehicle (Doctoral dissertation, University of Washington).".

[13] "Mathworks, Explore the Hybrid Electric Vehicle P4 Reference Application".

[14] "ExtremeTech, What is ACC, and how it works?".

[15] "Prestl, W., Sauer, T., Steinle, J., & Tschernoster, O. (2000). The BMW active cruise control ACC (No. 2000-01-0344). SAE Technical paper.".

[16] "GM Webinar, Adaptive Cruise Control (ACC) Algorithm Testing".

[17] "DieselNet, Emission Test Cycles".

[18] "Erkkinen, T., & Conrad, M. (2008). Verification, validation, and test with model-based design (No. 2008-01-2709). SAE Technical Paper.".

X. Appendix

MATLAB files:

- Automated MIL Simulation on Simulink
  - ACC_MIL_Script
- Velocity measurement data processing using Simulink
  - ACC_HIL_Data_Processing_Script
- FTP Drive Cycle MIL Data
  - DS_FTP75