Investigation of Cooperative Adaptive Cruise Control with Experimental Validation

A Thesis
SUBMITTED TO THE FACULTY OF
UNIVERSITY OF MINNESOTA
BY

Pratik Mukherjee

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
MASTER OF SCIENCE

Zongxuan Sun, Adviser

July 2016

Acknowledgements

It is with great honor that I will like to thank my adviser Professor Zongxuan Sun for giving me the opportunity to conduct research under his guidance for my Master's Plan A thesis. His superior insight into the problem has allowed me to overcome many obstacles while working on the several problems that I have come across in the last two years. His consistent dedication of his time toward his students and focus on individual student's progress has kept me on track to finish my work. Under his guidance, I have learned to work and think independently and I am constantly improving my problem solving skills which will definitely be an asset that I take with me after I leave this place. Without his continuous support to the whole research team, it would not have been possible for me to graduate on time, with quality.

I will also like to thank the primary sponsors for the project, Federal Highway Administration(FHWA). Without their constant financial support and trust it would not have been possible to carry out various crucial tasks. Their constant interest in our progress motivated me to complete my work with dedication to provide quality solutions for every problem I solved, that will stand the test of time.

I will also like to thank my lab mates Azrin, Yunli, Abhinav, Chen, Yongsoon, Yaoying and Keyan for constantly motivating me to work hard. Observing their work ethics over these past two years has made me aware of my weaknesses that I need to work on to constantly improve my skills.

Pratik Mukherjee

Dedication

I dedicate this thesis to my motherland India, my mother and father

Abstract

Increasing effects of global warming have concerned scientists and engineers for quite some time now. The major contributor to global warming has been the inefficient use of energy to fulfill the need for the ever growing human population. One of the major sources of energy is oil which fuels one of its largest consumer, the transportation sector. The Energy Information Administration reported that the U.S transportation sector contributed 28% to the total energy consumption and 72% to the total petroleum consumption in the year 2010. With these concerning developments, it has become critical to find a solution to improve the efficiency of transportation. A solution to this problem can be connected vehicles. Connected vehicle environment paves the pathway for future road transportation. Researches in this area have specifically focused to improve traffic mobility and safety, and also vehicles' fuel consumption and emissions. A Hardware-in-the-Loop-System (HiLS) test-bed to evaluate the performance of connected vehicle applications has already been developed. A laboratory powertrain research platform, which consists of a real engine, a hydrostatic dynamometer and a virtual powertrain model to represent a vehicle, is connected using a software to a microscopic traffic simulator (VISSIM). Actual fuel and emissions measurements are obtained using this test-bed. This thesis documents the development of the software architecture that enables the different components of the HiLS to communicate with each other in real time. Different methodologies of software design are tested to demonstrate real time execution of HiLS with a 200ms time step. Further, using this test-bed a comprehensive evaluation of Cooperative Adaptive Cruise Control (CACC) application has been conducted to compare the fuel consumption and emissions of CACC vehicle and non-CACC vehicle in a traffic network simulated in VISSIM. In literature, CACC application is implemented using several different complex optimization methods based on prediction models derived from measurements of traffic information with the cost of computation power. In this thesis, a heuristic, averaged velocity approach to CACC is implemented using the information from the preceding vehicles which can be used in real time systems like the HiLS for a realistic evaluation of the CACC application. The algorithm designed uses information from multiple preceding vehicles to determine a velocity profile for the controlled vehicle which will obtain fuel benefits. The simulation and experimental results obtained using this CACC algorithm prove that using more preceding vehicle information provides higher fuel benefits. This phenomenon is described by the understanding that the accessibility of future information, with regards to the number of preceding vehicle velocity averages, by the controlled vehicle allows the CACC controller to obtain a smoothened velocity profile for the controlled vehicle with suppressed acceleration or decelerations which has a direct correlation with fuel savings. VISSIM traffic simulator is used to simulate different traffic conditions like city driving and highway driving. From the simulation, an individual vehicle is selected to be completely controlled by the CACC controller whereas the other vehicles are controlled byVISSIM's internal driver model based on the Wiedmann Car following model. Then an extensive study is done through simulation of different traffic scenarios on the fuel consumption of the controlled and the immediate preceding vehicle which is evaluated using the Vehicle Specific Power (VSP) requirement. Further, to validate these simulation results, the HiLS is used to conduct experiments with selected scenarios from the simulations and actual fuel and emissions results for the CACC controlled vehicle and immediate preceding vehicle are compared. From the results obtained, it is realized that CACC application provides significant fuel saving, between 15-18% approximately for both highway and city driving cases, for the controlled vehicle in comparison to the immediate preceding vehicle. Future work will focus on using the observations from the current CACC methodology and implementing a systematic method, possibly an online optimization method, to find out a general solution where the preceding vehicles' information can be used. Then CACC application will be extended to evaluate more than one vehicle in a platoon of vehicles. Consequently, HiLS will be used to obtain experimental results.

Table of Contents

List of Tables

List of Figures

Nomenclature

X_n	Distance of controlled vehicle from reference
X_{n-1}	Distance of immediate preceding vehicle from reference
Δx	Difference of distance between controlled and preceding vehicle
$\Delta x_{min,0}$	Standstill minimum distance gap between controlled and preceding vehicle
$\Delta x_{max,0}$	Standstill maximum distance gap between controlled and preceding vehicle
L_{n-1}	Length of preceding vehicle
Δx_{max}	Maximum distance corridor limit
Δx_{min}	Minimum distance corridor limit
Δx_{med}	Median of distance corridor bounds
h	constant time headway
v_t	instantaneous velocity of controlled vehicle
v_{pn}	instantaneous velocity of preceding vehicle
n	number of preceding vehicle
v_{des}	average of preceding vehicles velocities
t	time in seconds
J	Cost function
u	control law
a	acceleration
T_{ph}	Prediction horizon
T_{step}	gain with respect to the magnitude of time step
dt	Time derivative
k, k_a,	gain

Y_t	system output at instantaneous time
C_D	Drag coefficient
M	mass of vehicle
g	gravitational constant
A	vehicle frontal area
μ	rolling resistance for vehicle
ρ_a	density
$\emptyset(t)$	road grade angle
Δv	difference in speed [m/s]

Chapter 1 Introduction and Background

1.1 Connected Vehicles Application

Recent concerns with global warming have revealed alarming rise in the average temperature of planet Earth. Experts are concerned because use of fossil fuel and its byproducts have majorly contributed to this rise in temperature and serious measures have to be taken to curb this rise. At the same time, the availability of resources is decreasing year after year with the demand rising exponentially. Thus scientists have started to focus towards developing more energy efficient technology or find ways to make existing technology more efficient, and one such major area of field of technology is the transportation sector. The Energy Information Administration reported that the U.S transportation sector contributed 28% to the total energy consumption and 72% to the total petroleum consumption in the year 2010.

Researchers are currently exploring different avenues to reduce fuel consumption and one major field of application is connected vehicles. Connected vehicles research pertains to research in the field of traffic and most recently in vehicle powertrain. Early developments in this topic focused majorly on safety of on road vehicles and reducing traffic congestion. With the introduction of Cruise Control(CC) in production vehicles in the early 1990s, came the development of Adaptive Cruise Control (ACC). Compared to conventional CC in vehicles, which only control the vehicle speed, ACC allows drivers to maintain a desired distance behind a preceding vehicle as well as a desired velocity. From the perspective of traffic network operation, a stable ACC controller that can maintain a consistent desired gap between vehicles will improve traffic safety and capacity of the traffic network. In [1] ACC is reviewed with the perspective of safety. [2], [3]demonstrate one of the first ways to implement ACC with respect to safety as well as fuel benefits.

1.1.1 Inter Vehicle Communication & Vehicle Infrastructure Integration

ACC uses range sensors to determine the relative distance and velocity to the preceding vehicle. It uses just one of the preceding vehicle's information gathered from its onboard sensors to determine the relative distance and velocity , whereas many researchers in the field of traffic are curious to find out how more information from several other vehicles as well as traffic infrastructure in the same traffic network can be used to improve safety, reduce traffic congestion and increase fuel benefits .Hence the idea of vehicle to vehicle and vehicle to infrastructure communication has emerged.

Recent research in the field of transportation has focused extensively on Inter Vehicle Communication (IVC) and Vehicle Infrastructure Integration(VII). Initial developments of these concepts were formulated with the intention of increasing road safety, but there has been an increase in the research related to using the information obtained from IVC and VII for increasing fuel economy. The concept of IVC enables communication within vehicles whereas VII is an attempt to enable communication between vehicles and aspects of traffic infrastructure like traffic signal or other road side units as shown in figure 1.

IVC and VII have not only gained attention in research communities, but also the Department of Transportation who are proposing for installing communication devices in new vehicles as early as 2017. Traffic information communication between vehicles, known as connected vehicle, will improve traffic mobility and safety. Connected vehicle technology also enables better optimization of a vehicle's fuel economy and emissions by utilizing traffic information such as the traffic light Signal-Phase-and-Timing (SPaT) and other vehicle speed information in a traffic environment.

Technologies associated with IVC and VII[4] have shown potential to improve traffic safety and efficiency[5], [6]. Technologies are being developed to implement IVC and VII, especially in wireless traffic communications. Recent researches are focusing to validate the reliability of Dedicated-Short-Range-Communication (DSRC) [7], [8]which is now the standard for wireless vehicle communication [9]. The US Federal Communication Commission (FCC) has agreed to dedicate 75MHz spectrum from 5.85GHz to 5.925GHz bands for DSRC. Using these technologies, tests have been evaluated with respect to the scalability, security and interoperability of DSRC communications in a real world setting. IVC and VII technology enable use of more traffic information which helps improve fuel benefits and reduce emissions.

1.1.2 Vehicle Communication Devices

As an extension to ACC, Cooperative Adaptive Cruise Control (CACC) incorporates IVC and VII communication. CACC has been under development to utilize IVC and VII to conduct vehicle level optimization. It is manifested with the idea of using every possible information available in a traffic environment, to gain potential benefits with regards to fuel economy, safety and traffic congestion. Information such as traffic signal timing cycle, longitudinal and latitudinal vehicles speed, acceleration behavior are crucial to realizing the true potential of CACC. CACC vehicles can be designed to follow the preceding vehicles with significantly higher accuracy and faster response because of the availability of more information. Previous research has shown that CACC vehicles are better at following preceding vehicles and are much more stable than ACC vehicle [10].

The development of ACC/CACC controller is crucial in understanding the involvement of different levels of vehicle autonomy. The three levels of vehicle control currently known are non-autonomous, semi-autonomous and autonomous. Non-autonomous vehicles are everyday vehicles that are controlled by humans. The vehicles are not provided with any external traffic information except for what is perceived by the human driver. Semi-autonomous is a higher penetration of autonomy in vehicle control where information is provided to the driver in the form of an advisory. An autonomous vehicle scenario occurs when the ACC/CACC controller takes complete control of the vehicle and uses parameters fed to it as input from the available traffic data. These traffic information are based on the detection and measurements from range sensors, internal states measurements from vehicle state sensing position, engine speed, vehicle speed and extending it to accessing the preceding vehicles' information through wireless communication. A typical traffic network setup to enable a complete or semi-autonomous vehicle control is shown in figure 1.

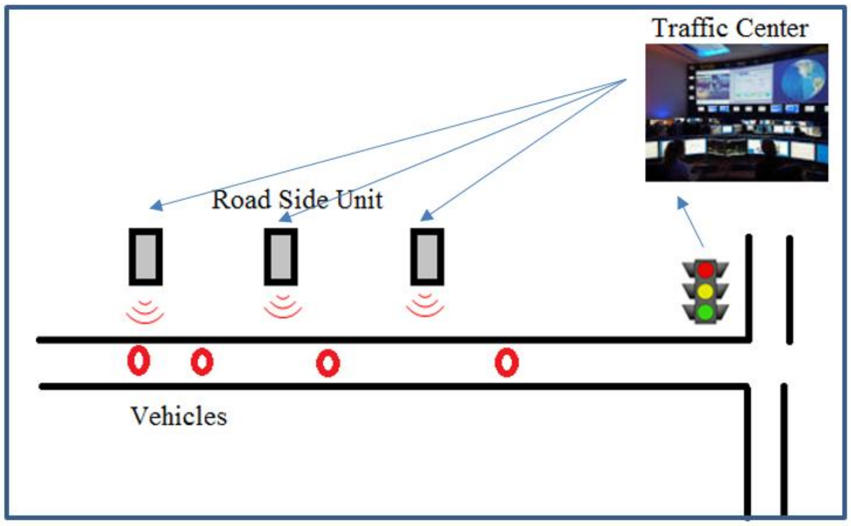


Figure 1. Traffic Information processing through IVC & VII

One of the key components in determining the extent of the success of autonomy of vehicles is the capability of range sensor equipment. The range sensor’s basic, core responsibility is to detect relative distance and velocity between the controlled and the immediate preceding vehicle. Commonly used range sensors are radar, vision sensors, and light detection sensors for the application of ACC and wireless communication for CACC. The biggest challenge for such devices is to provide reliable and accurate information to the controlled vehicle under any circumstances because the information these devices provide are crucial to the stability of the ACC/CACC controllers implemented. For CACC application long range wireless communication, such as a network with a specific bandwidth, is more relevant. With the availability of wireless communication more information can be sent because through wireless communication both IVC and VII communication can be achieved.

1.2 CACC Controller

Many optimization tools like Model Predictive Control, as shown in [11], Pontryagin’s Minimum Principle as shown in [12] or Pulse-and-Gliding (Png) methods [13], [14]have been used for both ACC and CACC applications. However, many of the optimization algorithms implemented are computationally heavy and difficult to implement in real time systems. In [15] , the behavior observed from the optimized driving cycle of a target vehicle is intuitively used to develop a simplified ACC approach that leads to fuel savings and maintains safety as well as can be implemented in real time systems. This idea can further be extended to develop a CACC controller

1.2.1 CACC Controller Development

In this thesis, the discussion will be on several case studies and approaches taken towards CACC implementation with respect to fuel consumption savings and its evaluation using a real engine. Using the simplified method mentioned in [15], the behavior of vehicles is studied. The study is focused on using only the information from just one preceding vehicle which demonstrates the application of ACC, but in this thesis a similar simplified method is implemented to demonstrate the capability of CACC approach to achieve fuel benefits as well as maintain safety of vehicles in traffic network. Therefore, information from more than one vehicle is obtained to formulate an algorithm that has potential for obtaining fuel benefits without any prediction of the future velocity profile of the preceding vehicles. The approach aims to prove that using the preceding vehicles' velocities will give significant fuel benefits. The idea is to incorporate the dynamics of the preceding vehicles, based on the reaction of the driver in the lead vehicles, to any traffic situation in the velocity calculation of the controlled vehicle. This will allow the controlled vehicle to make crucial judgments to modify its own dynamics which will lead to fuel benefits.

Different scenarios are tested with respect to the ability to maintain safety of vehicles, and reduce fuel consumption. One of the constraints is to maintain the controlled vehicle within the specific distance corridor bounds which means the controlled vehicle must have a limited range of distance to follow from the immediate preceding vehicle within which it can traverse freely to obtain maximum fuel benefits. The controlled vehicle must not approach the preceding vehicle dangerously close as well as recede too far away from the preceding vehicle. Either of these situations will jeopardize the safety or traffic flow respectively. To tackle such a situation, different methods are simulated with respect to the deviation of the controlled vehicle's distance from the distance corridor limits. In

[16],distance constraints are added to assure the safe travelling of vehicles within specific distance corridor bounds

1.2.2 A Simplified CACC Methodology

The objective of the Simplified Fuel Efficient Predictive Cruise Control Approach from literature is to reduce fuel consumption relative to a preceding vehicle by following the predecessor within a safe distance corridor. This approach implements an Adaptive Cruise Control (ACC) with a prediction of the velocity profile of the preceding vehicle over a horizon. A prediction model is used which is derived based on measurements of a velocity profile of a real vehicle on road. Using this prediction model, the approach conducts a prediction of the velocity of the preceding vehicle for a specific time horizon and takes the average of the velocity over the horizon. This average velocity is then assigned as the speed of the controlled vehicle for the next time step. However it also implements a velocity correction term to maintain the maximum and minimum distance corridor constraints. This approach is much more simplified than the other methods as it does not need to use any computationally heavy optimization algorithm. The approach of a velocity trajectory generator is based on the observations made from the results of other optimization methods that is; an averaged velocity profile of the controlled vehicle compared to the preceding vehicles has high potential of fuel benefits. Therefore, using a prediction model for a specific time horizon, the future velocity of the preceding vehicle is obtained which is averaged over the time horizon to give a smoothened velocity profile for the controlled vehicle.

Although the above approach presented is simplified, it uses a prediction model for the prediction of the immediate preceding vehicle's velocity profile which is not applicable for all traffic scenarios. Therefore, like any other prediction model it has inaccuracies associated with it. A common problem with prediction model is that it may be specific for one road condition and different for a different road condition, making it tedious to determine a prediction model for every specific road condition. Thus, the approach presented in this thesis aims to use the idea of a smoothened velocity profile for the controlled vehicle but without using a prediction model. Instead, the controlled vehicle is able to access the velocity of multiple preceding vehicles and incorporate the dynamics of the preceding vehicles in its own dynamics by making its velocity the function of the average of the preceding vehicles' velocities. The algorithm is not based on offline measurements but one which can be determined in real time .This algorithm is implemented to obtain a velocity profile for the controlled vehicle that can lead to reduced fuel consumption and maintain a safe distance between vehicles. In this averaged velocity approach an online CACC algorithm is implemented with the assumption that the velocities of the preceding vehicles are available to the controlled vehicle at each time step.

This approach of CACC control implemented complements the idea of CACC as it uses more than one preceding vehicles' velocities to take the average and add a correction term to maintain the distance corridor. The method of implementation makes this algorithm applicable in real time to obtain a velocity profile for the controlled vehicle that can achieve up to 17% of fuel benefit. Vehicle Specific Power in [12] is first used to measure the fuel consumption of the controlled as well as for the first preceding vehicle for simulation purposes assuming that the preceding and the controlled vehicles are same type of vehicles. Then the simulation results are validated with experimental results obtained from the powertrain-research-platform. Simulations are carried out with different number of preceding vehicles for different driving cycles to understand the influence of number of vehicle velocity averages on the velocity profile of the controlled vehicle.

1.2.3 CACC Simulation with HiLS

For simulation purposes, two different approaches to apply the constraints have been simulated. The two methods simulated are the buffer zone and the median method. The case study with the best results is used to develop a CACC controller in SIMULINK which is integrated with the HiLS. It requires sophisticated software interaction to achieve the simulation objectives. The powertrain-research-platform consists of sophisticated hardware such as a full-sized diesel engine, a hydrostatic dynamometer on which the engine is loaded for tests. A state of the art laboratory is used to house the engine. To successfully run and test the engine, a control environment has to be created. This is only achievable through the use of software integration. A middleware is designed to integrate all the different components of the HiLS. This middleware is developed in C# platform and integrated with Matlab/SIMULINK, and VISSIM Microscopic Traffic Simulator to enable the simulation of a selected virtual vehicle to test the application of CACC. The information from this simulator is transferred over a wired or wireless network to the powertrain-research -platform to obtain experimental, quantitative fuel consumption and emissions measurements. Therefore the collaboration of the software and the hardware plays a vital role in the successful execution of the complete HiLS.

1.3 Experimental Validation of CACC Application

Methods to measure the performance of a vehicle's fuel economy and emissions in traffic include conducting simulation [3]or instrumenting a vehicle .However both approaches have drawbacks. A simulation-based approach implements steady-state fuel-use and emission maps as a function of the engine torque and speed, which are deemed inaccurate compared to actual measurements whereas instrumenting a vehicle is a cumbersome process because it requires modifying the vehicles. Therefore HiLS offers the flexibility and accuracy of evaluating the performance of connected vehicle applications.

HiLS, a laboratory powertrain-research-platform, consists of a real engine, a hydrostatic dynamometer and a virtual powertrain model to represent a vehicle, is connected using a software to a microscopic traffic simulator (VISSIM). HiLS utilizes the powertrain-research-platform, which consists of a real engine for fuel and emission measurements in real-time. VISSIM traffic simulator allows different vehicles to be tested with different driving profiles by altering the engine and the load settings on the dynamometer. The lab set up for HiLS accommodates large precision measurement devices. Therefore, making connected vehicle applications testing in a simulated but realistic traffic more economical, without having to instrument multiple vehicles. This method is much safer and it overcomes the legal issues associated with testing the application in real traffic .

1.3.1 Components of HiLS

With HiLS, different vehicles can be tested flexibly by changing the engine and the load settings on the dynamometer. Large precision measurement devices can be fitted in large vehicles [17] , but it will be cumbersome and time consuming for smaller passenger vehicles. Smaller portable measurement devices have been used, but are less accurate, and require calibrations for different driving cycles [18].

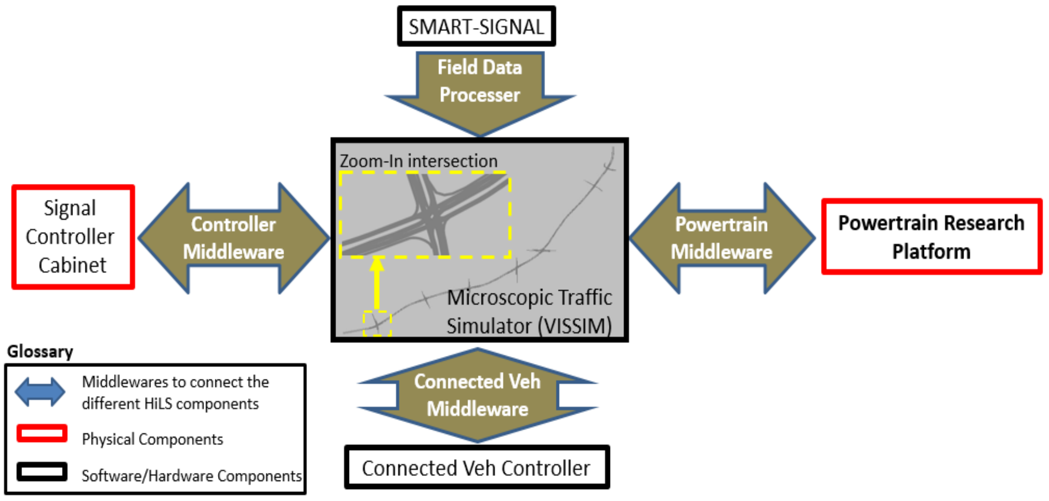

SMART-SIGNAL
Field Data Processer
Zoom-In intersection
Signal Controller Cabinet
Controller Middleware
Microscopic Traffic Simulator (VISSIM)
Powertrain Middleware
Powertrain Research Platform
Glossary
Middlewares to connect the different HiLS components
Physical Components
Software/Hardware Components
Connected Veh Middleware
Connected Veh Controller

Figure 2. HiLS Componenets

The HiLS consists of several different components as shown in figure 2. Each component is connected to the other with the help of a middleware developed specifically to serve the purpose of connecting the components in real time. The complete integration of all the components in a synchronized manner is required to demonstrate real time application. The three components of focus for this thesis are the Connected Vehicles Controller which pertains to CACC , the powertrain-research-platform which consists of a real engine with a virtual powertrain equipped with measurement devices for the purpose of obtaining accurate and precise measurements of fuel consumption and emissions, and VISSIM traffic simulator to simulate a realistic traffic environment.

1.3.2 Powertrain Research Platform

The engine set up in the Thomas E. Murphy lab is a John Deere diesel engine. The powertrain research platform is developed with the intention of using a real engine with a virtual powertrain to provide the user with the flexibility of using any powertrain model developed in commercial software like MATLAB. In particular, this platform is implemented with a power-split Hybrid Electrical powertrain derived from the Toyota Prius hybrid architecture, as given in[19], [20]. This power split architecture divides the power provided to the powertrain which is partially shared by the internal combustion engine and an electric motor or generator. The power split architecture allows higher efficiency at the fuel consumption level as it allows two degrees of freedom to the engine; providing power through the engine or through the motor.

1.3.3 Virtual Hybrid Powertrain

The hybrid powertrain research platform is designed to carry out investigation of fuel efficiency on hybrid vehicles. One of the major benefits of using such a set up for investigation is the flexibility with which experiments can be conducted. This set up employs a high bandwidth hydrostatic dynamometer to emulate the dynamic behaviors of the hybrid power sources like the electric motor/generator and vehicle loads, and interact with a multi cylinder engine in real time. The idea of virtual hybrid is used to emulate the transmission, driveline and load of a hybrid powertrain using a hydrostatic dynamometer as described in [21]. This is shown in figure 3.

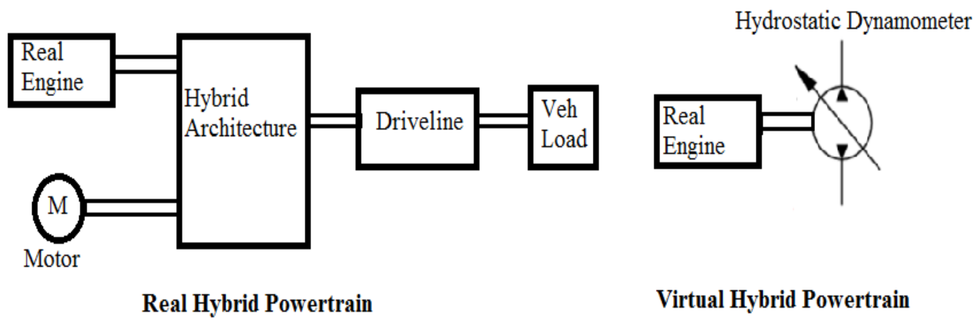


Figure 3. Virtual Hybrid Powertrain[19], [21]

This idea of virtual hybrid powertrain saves the cost of building a physical powertrain system and expedites the speed of hybrid powertrain research. The design of the hydrostatic dynamometer based research platform is a complete hybrid powertrain control and simulation system .

1.3.4 Fuel and Emissions Measurement

The HiLS utilizes an existing powertrain research platform that has been developed as shown in Figure 4. A real engine overcomes the inaccuracies associated with the modelling of combustion and emission behavior of an engine accurately for real-time application, while the dynamics of powertrain is obtained using accurately developed models. Therefore, the control and simulation is defined by a three-level closed-loop architecture[19], [21]. The high-level controller, given the power demanded from the vehicle, selects a reference engine operating point that optimizes fuel consumption and emissions. For the middle-level controller, the virtual-torque-controller controls the powertrain torques that utilizes the reference engine torque from the high-level controller. Highly accurate models are used to simulate the dynamic responses of the powertrain components which include the desired engine loading torque. In the low level controller, the dynamometer tracks the desired engine loading torque from the middle-level controller. Fuel consumption and emissions from the engine are measured by precision measurement devices.

Fuel consumption is measured using AVL's Fuel Measurement System Model P402 with a measurement uncertainty of 0.1% and output frequency of up to 80kHz, and the emissions are measured using AVL's SESAM-FTIR, which measures 25 components of exhaust gas from engine combustion, for example NO, NO_2, CO and CO_2, with a sampling rate of 1Hz.

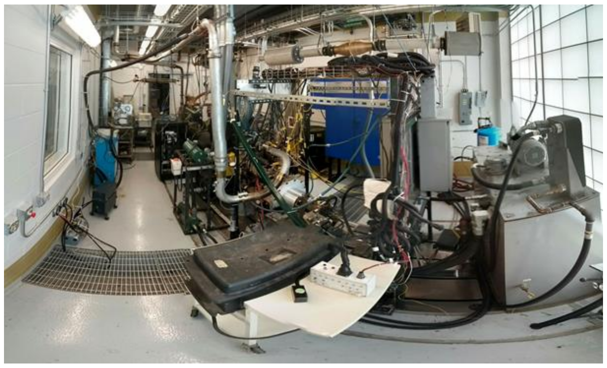


Figure 4.Powertrain Research Platform

1.3.5 VISSIM Traffic Simulator

For the current research under progress, VISSIM, a microscopic traffic simulation software is used to carry out real time simulation of traffic on different types of road architectures. VISSIM provides the user with the flexibility to design their own traffic scenario using calibration based on actual measurement data from on road traffic or a hypothetical set up of simulation scenarios for experimental purposes. It is a microscopic simulator that allows the user to focus on individual vehicle and also allows the user to acquire individual speed, location and various other vehicle attributes.

For this project, VISSIM is a vital component that is used to run the simulation. However, it is important to extract this information from VISSIM software and pass it on to the powertrain-research-platform for obtaining real time experimental data. The information sent by VISSIM will be used as a reference parameter by the powertrain-research-platform . VISSIM is used to provide the real time simulation data and emulate a realistic traffic scenario. To achieve all these tasks of sending vital information of a specific vehicle amongst all other vehicles in the traffic and then use that information in the powertrain-research-platform, different software need to communicate with each other. The proposed idea is to use C# and MATLAB to communicate with VISSIM where C# acts as the Component Object Model (COM) client and server [22], [23]. The complete synchronization of the software should be designed keeping in mind the implementation of various applications. For this thesis, the implementation of CACC application is crucial.

1.3.6 Integration of Various Software Platforms using Middleware

The software communication has to be executed three way. From the previous discussions, it can be inferred that the major pathway between the software communications has to be between VISSIM and a programming software compiler like MATLAB, C, C# or any other compatible language compiler that allows the user to carry out analysis on the data collected from VISSIM. Previous work [24] shows successful integration of VISSIM traffic simulator with COM objects like MATLAB. For HiLS, MATLAB is used for various other applications like simulating the Hybrid Electric Vehicle powertrain and also the CACC controller. Creating a COM outside the MATLAB environment gives the user the freedom of extracting information from VISSIM and makes the whole architecture less cumbersome. Therefore it is realized that it will be more efficient to have a middleware to communicate between VISSIM and other software. Hence ,it is decided that a powerful programming language will be most appropriate for the application of the COM and so C# is used to communicate both between MATLAB and VISSIM. The COM client or server, in this case C#, communicates all the data that it receives from VISSIM to MATLAB. VISSIM and C# are extremely compatible and it is easy to integrate the VISSIM software using C# programming language. More about programming with C# and VISSIM is discussed in [25]. In this thesis we are focusing on sending the extracted data to the powertrain-research-platform through a network, the internet.

1.4 Thesis Contribution

In this thesis, a heuristic CACC controller is developed, a software or middleware is designed to integrate the HiLS and finally CACC methodology is evaluated using experiments in integration with the HiLS.

CACC Controller Design

The objective is to develop a connected-vehicle controller and integrate it with a microscopic traffic simulator (VISSIM) to replicate real traffic dynamics for fuel and emissions measurements. Hence, a simplified, averaged preceding vehicles' velocities method is used to design a CACC controller to implement the controller in real time with HiLS. Instead of using a prediction model the controller uses the vehicle information of multiple preceding vehicles to determine a velocity profile for the controlled vehicle which provides potential fuel benefits.

Design of Software Architecture for HiLS

In the HiLS, the powertrain research platform calculates the demanded power of a target vehicle in VISSIM which is then used to load a real engine. Real fuel consumption and emissions are then measured using state-of-the-art measurement devices. To integrate the powertrain research platform with the traffic simulator and CACC application, a robust software or middleware is designed and implemented. The software must flawlessly conduct real-time data transfer from and to different components of the HiLS within a time-step of 200ms.

Evaluation of CACC controller through Experiments.

The final objective is to use the powertrain-research-platform to evaluate CACC approach for connected vehicles applications. Several simulation case studies are obtained using the developed CACC controller. A real-time CACC application is evaluated with the powertrain-research-platform in integration with the CACC controller to validate the simulation results .

Chapter 2 Implementation and Evaluation of Cooperative Adaptive Cruise Control Using Simulation

2.1 Introduction and Background

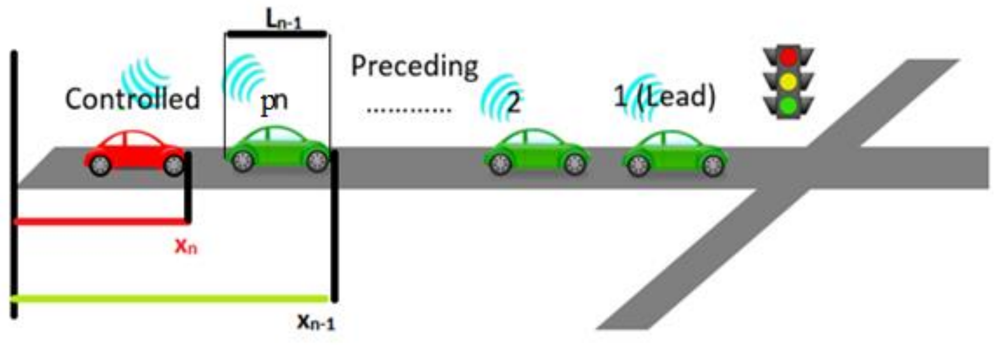


Figure 5. CACC concept sketch

Recent developments in CACC have demonstrated the implementation of various optimization methods. Optimization is done with respect to multiple objectives like to increase safety, to increase comfort and most recently to reduce fuel consumption for better fuel economy. In [2], [3] an application of multi-objective Model Predictive Control (MPC) is implemented. The MPC algorithm is designed to fulfill the objective of reducing fuel consumption while taking into consideration mobility and comfort of the controlled vehicle. In [26],CACC is implemented with the perspective of increasing traffic flow by inducing string stability so that the inter-vehicle distance can be reduced while keeping a safe distance between consecutive vehicles in front of the controlled vehicle.

CACC takes into account a sophisticated IVC or VII which has opened up a whole new avenue for research in vehicle dynamics. In [4] the velocity of controlled vehicle is influenced based on the information obtained from the traffic signal timings to reduce the time of travel as well as to obtain fuel benefit by reducing the occurrence of stops at traffic signals. Different optimization methods have served to explore ways to obtain fuel benefit, reduce travel time, increase comfort, but these complex optimization methods come with the cost of complex computations. Such optimization methods make the implementation of an effective controller in real time very difficult. However, such algorithms can be used to give insight and understanding of the vehicle dynamics which may further lead to methods that are computationally less complex and can be implemented in real time for real systems. One such algorithm developed is the Simplified Predictive Cruise Control [15] discussed in chapter 1.

2.1.1 The Distance Corridor Constraints

The general CACC implementation methodology includes a string of vehicles in a specific traffic network where the controlled vehicle follows the immediate preceding vehicle at a specific distance, maintaining the distance corridor. The controlled vehicle must maintain the constraints given by

$$\Delta x_{min} \leq \Delta x \leq \Delta x_{max}$$

The constraints Δx_{min} and Δx_{max} are given by the constant time headway policy which are stated as

$$\Delta x_{min} = \Delta x_{min,0} + hv$$

$$\Delta x_{max} = \Delta x_{max,0} + hv$$

$$\Delta x = x_{n-1} - L_{n-1} - x_n$$

Where $\Delta x_{min,0}$ and $\Delta x_{max,0}$ are the minimum and maximum inter-vehicle distance when the vehicles are at stand still with h as the constant time headway and v the velocity of the controlled vehicle at that instant. The value of h is typically equal to one or more than one. Since constant time headway is assumed, it is decided that h will be equal to one based on the initial condition of the controlled vehicle in VISSIM traffic simulator. The controlled vehicle's initial speed is around 20m/s and it is approximately 22m away from the immediate preceding vehicle. Therefore, the controlled vehicle will take approximately 1.1 seconds to reach the position of the immediate preceding vehicle, which makes the selection of h as one very feasible. Also, x_{n-1} and x_n are the positions of the preceding vehicle and the control vehicle respectively from the starting point of the simulation. L_{n-1} is the length of the preceding vehicle. In VISSIM, the length of each vehicle type is given as a default and the length of vehicle type cars is between 4.11 and 4.76 m. Therefore, for simulation purposes, the average value of 4.435m is chosen for the length of cars. This is clearly depicted in figure 5.

As mentioned earlier, an ideal communication between the controlled vehicle and all the preceding vehicles in front of it is assumed. This assumption is crucial for the implementation of the controller, as the average of the velocities of all the preceding vehicles is required to determine the velocity of the controlled vehicle for the next time step.

2.1.2 A Simplified Approach to CACC

In [11] ,the approach chosen to determine the velocity profile of the controlled vehicle for a reduced fuel consumption over the complete driving cycle is to implement an optimization method to minimize the fuel cost which is a function of the vehicle speed and acceleration as stated below. This method of optimization with respect to fuel consumption is widely used in literature. However, the simplified approach will not use any optimization with respect to fuel consumption.

$$\min \int_0^{t_{end}} q_f(v, a)\, dt$$

The approach in this thesis is to implement a simplified method without doing any sort of optimization with respect to fuel consumption as shown above . The intention is purely to showcase the advantage of using multiple vehicle information on fuel savings for a controlled vehicle. Taking the average of the velocities of the preceding vehicles provides a velocity profile for the controlled vehicle such that the high accelerations and decelerations are suppressed which in turn lead to fuel savings. By taking the average of the velocities of a certain number of preceding vehicles, the idea is to incorporate the dynamics of the preceding vehicles in the velocity of the controlled vehicle as future information. For instance, in figure 5, as a platoon of vehicles is shown, if the lead vehicle makes a stop or decelerates at a traffic signal junction and the velocity of the controlled vehicle is a function of the lead preceding vehicle, then the deceleration of the lead vehicle will cause the controlled vehicle's velocity to decelerate too but not at the same rate because the controlled vehicle's velocity is also the function of the average of the several other preceding vehicles in front of it. At the same time, the preceding vehicle number two, not equipped with CACC controller, will not have obtained this deceleration and continue to accelerate until it realizes it has approached the first preceding vehicle and then decelerate abruptly.

This abrupt deceleration will lead to high fuel consumption for the second preceding vehicle and also for the other preceding vehicles without CACC. However, the controlled vehicle with CACC benefits because it has already slowed down due to the incorporation of the velocities, which are a function of the dynamics of the preceding vehicles. Due to this ahead of time incorporation of the preceding vehicle's dynamics in the controlled vehicle, the controlled vehicle somewhat emulates the reaction of the preceding vehicles. If the lead vehicle reacts to a signal by decelerating, this reaction of deceleration takes some time to reach the other following vehicles and the most amount of time to reach the controlled vehicle. However, with the average velocity method, the reaction flow time is drastically reduced and the controlled vehicle reacts as instantly as the lead vehicle does. Due to this slight deceleration, the controlled vehicle recedes behind the immediate preceding vehicle, increasing the distance gap between each other.

Now, by the time the deceleration reaction reaches the immediate preceding vehicle, the controlled vehicle has sufficient distance between itself and the immediate preceding vehicle to not decelerate abruptly, whereas the immediate preceding vehicle will experience aggressive acceleration or deceleration to maintain a safe distance between itself and the consecutive preceding vehicle. Since, the controlled vehicle will not decelerate abruptly or accelerate abruptly, over a driving cycle, this trend of the velocity profile of the controlled vehicle will look like an averaged profile of the immediate preceding vehicle with reduced stops compared to any non-CACC preceding vehicle. If this smoothened profile is repeated over the complete driving cycle then the controlled vehicle will benefit from reduced fuel consumption.

2.2 CACC Controller Design

The method in [15] uses Adaptive Cruise Control technology where it is assumed the controlled vehicle has a radar sensor attached to it which determines the velocity of the forward vehicle at the specific time step, and uses this velocity with a prediction model to predict the future velocity of the forward vehicle for ten seconds time horizons. Using this future velocity of the forward vehicle, the author claims that taking the average over the ten seconds prediction horizon gives the desired velocity trajectory or reference velocity. The idea is to average the velocity of the forward vehicle and assign it as the desired velocity of the controlled vehicle so that the controlled vehicle has less fluctuations in acceleration or deceleration. In this thesis, a similar approach is taken where a reference velocity or velocity trajectory is derived based on the average of the velocities of the multiple preceding vehicles. However, no prediction of the velocity profile was conducted. The states used for this dynamic system are $[\Delta x, v]$.

The basic dynamic equations in continuous time are as follows:

$$\Delta x' = v_p - v$$

$$v' = a$$

Where Δx is the absolute distance between the preceding vehicle and the controlled vehicle, v_p and v are the instantaneous velocities of the immediate preceding and the controlled vehicle respectively and a is the instantaneous acceleration of the controlled vehicle

The above dynamic equations are used with constraints. The major constraint is on the mobility of the controlled vehicle with respect to the preceding vehicle. The intention is to make sure that whatever the velocity is derived for the controlled vehicle, it must not lead the controlled vehicle to exceed the minimum distance limit such that it hits the preceding vehicle and crashes or it must not exceed the maximum distance limit such that it recedes behind the preceding vehicle, disrupting the flow of traffic. It is important to maintain the traffic flow for achieving a high mobility as well as maintaining a safe distance from the forward vehicle. A simple linear spacing policy is used, as mentioned earlier, to make sure the above criteria of safety and mobility is fulfilled. Where $\Delta x_{min,0}$ *and* $\Delta x_{max,0}$ are the minimum and maximum stand still positional difference between the controlled and the immediate preceding vehicle.

In discrete form the dynamic equations become:

$$\Delta x(t+1) = \Delta x(t) + dt * (v_p(t) - v(t))$$

$$v(t+1) = v(t) + dt * (a(t))$$

Δx(t) is the measured difference of position between the controlled vehicle and preceding vehicle. To measure Δx(t) correctly at every time step from VISSIM traffic simulator, it is important to consider the length of the preceding vehicle and include it in the calculation for Δx(t) as follows

$$\Delta x(t) = x_{n-1}(t) - L_{n-1} - x_n(t)$$

2.2.1 Velocity Trajectory Generator Equation

The velocity trajectory equation is derived as

$$v = v_{des} + v_{cor}$$

Where

$$v_{cor} = \mathbf{f_{cor}}\ (\Delta x, \mathbf{v}, \mathbf{v_p}, \Delta x_{\mathbf{med}}, \Delta x_{\mathbf{min}}, \Delta x_{\mathbf{max}})$$

The discussion for v_{cor} with regards to constraints is done in the next section. In this section, a CACC approach without using a prediction model is evaluated to determine a velocity trajectory. In this approach the equation as above is modified to give a velocity trajectory equation, but the derivation of v_{des} is based on the average of the velocities of a certain number of preceding vehicles in a platoon. V_{des} is derived as shown in the equation below

$$\mathrm{v_{des}}(t) = \frac{v_{p1}\,(t) + v_{p2}(t) + v_{p3}(t) + ..\, v_{pn}(t)}{n}$$

Where n is the number of preceding vehicles in front of the controlled vehicle and v_{p1} to v_{pn} are the velocities obtained from the n preceding vehicles at each time step. This is based on the assumption that CACC enabled vehicles have ideal communication either through IVC or VII.

The second term in equation above, the velocity correction term v_{cor}, is not utilized until the v_{des} determined leads the controlled vehicle to exceed the distance Δx constraints mentioned in equation above. Since the second term applies the distance corridor constraints to correct the v_{des} derived from average of the velocities of the preceding vehicles, this second term is given the name velocity correction v_{cor} .

In cases when the controlled vehicle's assigned v_{des} leads it to exceed the maximum or minimum distance corridor constraints, the second term v_{cor} comes into play. The second term is utilized to make sure the instantaneous velocity of the controlled vehicle is reduced by a deceleration term or increased by an acceleration term if the controlled vehicle exceeds the minimum or maximum distance corridor constraint respectively. Therefore the combined equation given to derive the velocity of the target vehicle is nothing but a velocity trajectory generator. It is expected that following the trajectory generated from this equation will give significant fuel benefits and also give more intuition into how averaging the velocity leads to possible fuel savings. Therefore, paving the path for a much more realistic CACC implementation without using any prediction model.

2.3 Applying Constraints – Derivation of the Velocity Correction term

The averaged velocity method to generate a trajectory utilizes the average of the velocities of a number of preceding vehicles in front of the controlled vehicle. Using this averaged velocity as the controlled vehicle's velocity is not sufficient to control the vehicle for achieving significant fuel benefits and maintaining safety of the vehicle. This is because an averaged velocity is a function of all the preceding vehicles in front of the controlled vehicle. Therefore the averaged velocity obtained may approach values that will cause the controlled vehicle to come dangerously close to the immediate preceding vehicle or even hit the preceding vehicle. On the contrary, the assigned velocity of the controlled vehicle cannot be much lower than the immediate preceding vehicles such that the controlled vehicle recedes far behind the preceding vehicle. Therefore, it is crucial to add constraints to the velocity of the controlled vehicle based on the velocity and the distance gap from the preceding vehicle. A velocity correction term is added to the average velocity term, as a function of the distance corridor bounds and also as a function of the immediate preceding vehicle's velocity. [16] proposes a similar control strategy where the correction term is an acceleration term which is a function of the distance corridor bounds and the immediate preceding vehicle's velocity. It also uses a piecewise method to increase the magnitude of the correction as the controlled vehicle approaches the upper or lower bounds of the distance corridor. This method is simulated in the next section and it is referred to as the buffer zone method. These correction terms ensure that the combined velocity derived for the controlled vehicle complements the objective of saving fuel as well as safe driving for all the vehicles in the traffic network.These constraints, in the form of velocity correction, are categorized as soft and hard constraints.

2.3.1 The Hard Constraints

The two basic hard constraints are categorized as case 1 and 2. The hard constraints are added to make sure the controlled vehicle does not exceed Δx_{max} and Δx_{min} which have already been defined before. The cases are explained as follows:

Case#1

If the controlled vehicle exceeds the maximum distance corridor limit, that is:

$$\Delta x \geq \Delta x_{max}$$

$$v_{cor} = dt * a = dt * k_a * (\frac{\Delta x - \Delta x_{max}}{dt^2})$$

Where $k_a = t_{step}$*k

Case#2

If the controlled vehicle happens to exceed the minimum distance corridor limit, that is:

$$\Delta x \leq \Delta x_{min}$$

$$v_{cor} = dt * a = dt * k_a * (\frac{\Delta x - \Delta x_{min}}{dt^2})$$

Where $k_a = t_{step}$*k

Where t_{step} must be equal in magnitude to the time step used in simulation. In both cases, 1 and 2, v_{cor} will take care of the velocity correction to bring the controlled vehicle within the constraints of the distance corridor. Case 1 specifically compensates for the deviation of the distance between the controlled vehicle and the immediate preceding vehicle as it exceeds the maximum distance corridor. Case 2 compensates the acceleration term when the same distance exceeds the minimum distance corridor limit. Therefore ,the equations mentioned above compensate for the deviation of the difference of distance between the controlled vehicle as an added acceleration term to the desired velocity term of the controlled vehicle. The weight k is designed to vary in relation to the change in distance corridor. The magnitude of k will decide the degree to which a correction is needed. Figure 6 depicts both the cases with the red vehicle as the controlled vehicle and the green vehicle as the preceding vehicle.

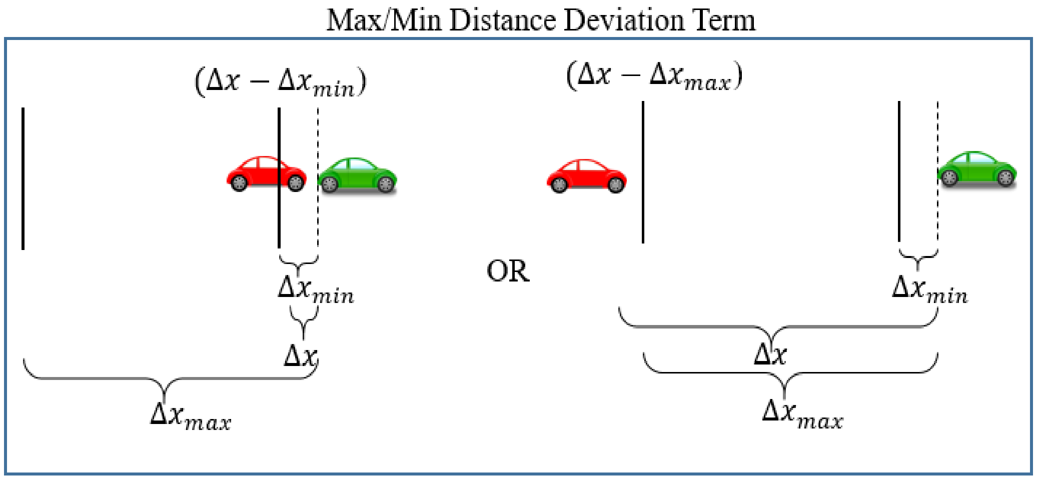

Max/Min Distance Deviation Term
$(\Delta x - \Delta x_{min})$
$(\Delta x - \Delta x_{max})$
Δx_{min}
OR
Δx_{min}
Δx
Δx
Δx_{max}
Δx_{max}

Figure 6. Minimum and Maximum Distance Corridor

2.3.2 The Median Method as Soft Constraints

Unlike the hard constraints, which impose drastic changes to the velocity of the controlled vehicle to strictly keep it within the limits, the soft constraints are added to reduce the impact of the drastic changes to the velocity of the controlled vehicle. The main objective of the soft constraints is to make the controlled vehicle realize that it is approaching the bounds and it must take action before the hard constraints are implied. It is highly preferable that the hard constraints are never utilized and instead the soft constraints are used to avoid the use of hard constraints because the hard constraints are detrimental to the reduction of acceleration of the controlled vehicle. Therefore the soft constraints are added as a compensation to the deviation of the distance between the controlled and the immediate preceding vehicle and the median of the maximum and minimum distance corridor limits. The idea is to directly compensate for this deviation so that by the time the controlled vehicle approaches the extreme limits, its velocity profile is such that a drastic change is not required to keep the target vehicle within the limits. This also implies that the hard constrains will not act very strictly on the velocity correction term for the target vehicle leading to less aggressive acceleration and deceleration behavior of the controlled vehicle. Figure 7 depicts the derivation of the median distance term based on Δx_{max} and Δx_{min}.

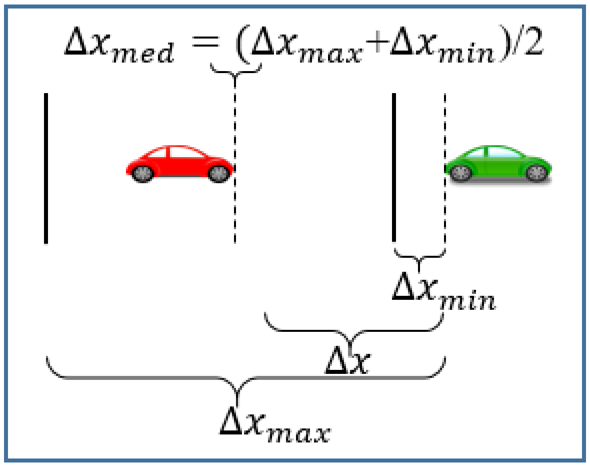


Figure 7. Median Distance Corridor

The dotted line, shown in figure 7, on the red controlled vehicle is the median of Δx_{max} and Δx_{min}. The term Δx_{med} can be used to compensate for the deviation of Δx as an acceleration term in v$_{cor}$, as v$_{cor}$ is a function of the acceleration of the vehicle. Thus the deviation term is derived as shown in figure 8 and implemented in the equations below.

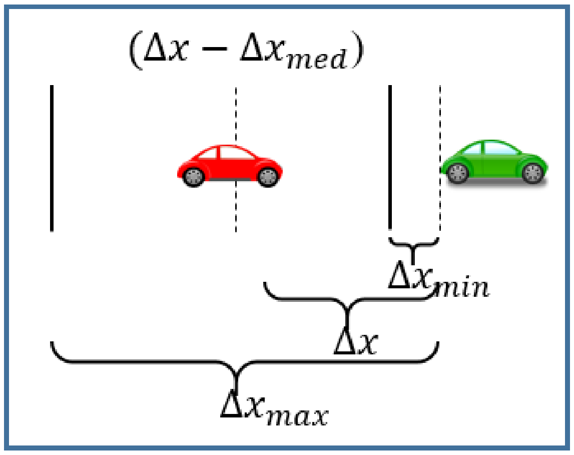


Figure 8. Derivation of Median Distance Corridor

Case#1

$$\Delta x \geq \Delta x_{med}$$

$$v_{cor} = dt * a = dt * k_a * (\frac{\Delta x - \Delta x_{med}}{dt^2})$$

Where k_a= t_{step}*k

Case#2

$$\Delta x \leq \Delta x_{med}$$

$$v_{cor} = dt * a = dt * k_a * (\frac{\Delta x - \Delta x_{med}}{dt^2})$$

Where k_a= t_{step}*k

Unlike in the hard constrain cases, 1 and 2 , where v_{cor} is derived with respect to Δx_{max} *and* Δx_{min}, in this case v_{cor} is a function of Δx_{med}. The complete v_{cor} piecewise function with both hard and soft constraints is given below

$$a = \begin{cases} k_{1\min}(\frac{\Delta x - \Delta x_{med}}{dt^2}), \\ \boldsymbol{\Delta x - \Delta x_{med} < 0, \Delta x - \Delta x_{min} > 0, \Delta x - \Delta x_{max} < 0} \\ k_{1\max}(\frac{\Delta x - \Delta x_{med}}{dt^2}), \\ \boldsymbol{\Delta x - \Delta x_{med} > 0, \Delta x - \Delta x_{min} > 0, \Delta x - \Delta x_{max} < 0} \\ k_{1\min}(\frac{\Delta x - \Delta x_{med}}{dt^2}) + k_{2\min}((v_p - v)/dt) \\ + k_{3\min}(\frac{\Delta x - \Delta x_{\min}}{dt^2}), \\ \boldsymbol{\Delta x - \Delta x_{med} < 0, \Delta x - \Delta x_{min} < 0, \Delta x - \Delta x_{max} < 0} \\ k_{1\max}(\frac{\Delta x - \Delta x_{med}}{dt^2}) + k_{2\max}(v_p - v)/dt \\ + k_{3\max}(\frac{\Delta x - \Delta x_{max})}{dt^2}), \\ \boldsymbol{\Delta x - \Delta x_{med} > 0, \Delta x - \Delta x_{min} > 0, \Delta x - \Delta x_{max} > 0} \end{cases}$$

2.3.3 Deriving the Weight Function

The weights used with the v$_{cor}$ function are crucial in deciding the degree to which the correction is needed to be added to v$_{des}$ so that the final velocity of the controlled vehicle provides a profile that has some potential fuel benefits as well as maintains the controlled vehicle within the distance corridor limits based on the smoothening of the velocity profile. The weight is decided by the k function which again is a function of the distance corridor Δx.

$$k_{1\min} = t_{step} * e^{-\frac{\Delta x}{\Delta x_{med}}}$$

$$k_{1\max} = t_{step} * e^{-\frac{\Delta x_{med}}{\Delta x}}$$

The k$_1$ weights are applied to the soft constraints as an exponential function .The objective of using an exponential function is to apply varying weight based on the deviation of Δx from Δx_{med}. Thus as the magnitude of difference between $\Delta x\ and\ \Delta x_{med}$increases, the weights k$_1$ increase exponentially depending on either $\Delta x > \Delta x_{med}\ or\ \Delta x < \Delta x_{med}$.

The functionality of the exponential function is depicted using the exponential graph in figure 9. It can be seen in the graph below that as the x-axis value increases the weight function's magnitude decreases and as the x-axis value decreases, the weight function value increases. This change in the magnitude is applied appropriately to the different scenarios as shown in the equation below. Thus this is how the weight k$_1$ is made to vary based on the change in the deviation of Δx from Δx_{med}.

$$f\left(\frac{\Delta x_{med}}{\Delta x}\right) = e^{\frac{-\Delta x_{med}}{\Delta x}} \quad when\ \ \Delta x - \Delta x_{med} > 0\ , \frac{\Delta x_{med}}{\Delta x} > 0$$

$$f\left(\frac{\Delta x}{\Delta x_{med}}\right) = e^{\frac{-\Delta x}{\Delta x_{med}}} \quad when\ \ \Delta x - \Delta x_{med} < 0, \frac{\Delta x}{\Delta x_{med}} > 0$$

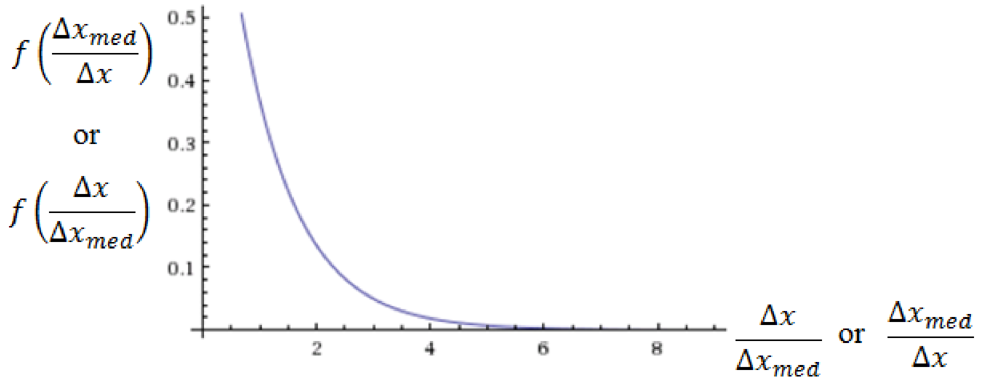


Figure 9. Exponential function with negative power

Similarly for the hard constraints the weights k$_2$ and k$_3$ are applied as an addition to the weights k$_1$ to add an extra measure to make sure the controlled vehicle remains within the bounds of the distance corridor. The hard constraints have a linear relation with Δx, Δx_{max}, Δx_{min} . Unlike the weight k$_1$, k$_2$ and k$_3$ change linearly with respect to the change between $\Delta x\ and\ \Delta x_{max}\ or\ \Delta x_{min}$ depending on $\Delta x > \Delta x_{max}\ or\ \Delta x < \Delta x_{min}$ The k$_2$ and k$_3$ functions are defined as follows:

$$k_{2min} = \frac{|\Delta x - \Delta x_{min}|}{\Delta x_{min}}$$

$$k_{2max} = \frac{|\Delta x - \Delta x_{max}|}{\Delta x_{max}}$$

$$k_{3min} = t_{step} * \frac{|\Delta x - \Delta x_{min}|}{\Delta x_{min}}$$

$$k_{3max} = t_{step} * \frac{|\Delta x - \Delta x_{max}|}{\Delta x_{max}}$$

2.3.4 Applying Different Methods of Constraints

There were two different methods that were simulated, before deciding to use the median method as mentioned in the previous section. These methods are intended to study the influence of averaging the velocities of preceding vehicles over the distance corridor limits, the acceleration of the vehicle based on the velocity correction determined to keep the vehicle within the distance corridor limits and the fuel consumption. The two methods are defined as: buffer and the non-constraint method.

First, for observation purposes, a non-constraint method is simulated where the distance corridor limits are kept sufficiently large to ensure that the controlled vehicle has enough space to purely use the average of the preceding vehicles as the velocity for the controlled vehicle in the next time step and not implement the correction factor. In other words, no constraints are implemented. Hence, the controlled vehicle is directly assigned with the average of the preceding vehicles' velocities. This method particularly studies the dynamics of the average of the preceding vehicles, allowing the controlled vehicle to freely move in the expanse of the enlarged limits of the distance corridor. It is anticipated that the target vehicle under the complete influence of the average of the preceding vehicles will have some fuel benefits. However, since the average velocity can be drastically different from the immediate preceding vehicle in front of the controlled vehicle , the controlled vehicle may not be able to follow the preceding vehicle in a small limit of the distance corridor. This is further discussed in the results.

The buffer zone method limits the expanse of the distance corridor between 10 and 50 meters of distance from the preceding vehicle, and has the velocity correction term added to the average velocity for the specific buffer zone [16]. This means that at a specific buffer zone the weight of the velocity correction term will increase based on an exponential function to prevent the controlled vehicle to surpass the maximum or minimum distance corridor limits. The objective of implementing a buffer zone is to reduce the unnecessary and abrupt acceleration and deceleration when the controlled vehicle suddenly is made to realize that it has crossed the distance corridor bounds. With the buffer zones implemented, it is anticipated that the abrupt acceleration or deceleration will be smoothened, leading to fuel benefits.

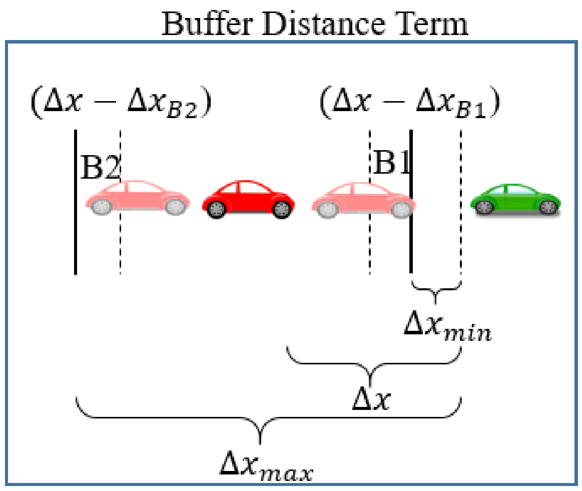


Figure 10. Buffer Zone Method

Next, in the median method, as already discussed in the previous section, the objective is to compensate for the deviation of the controlled vehicle’s position from the median of the minimum and maximum distance corridor limits. It is anticipated that with this added weight over the weights that compensate for the deviation of the controlled vehicle’s distance when it crosses the maximum and minimum limits, the controlled vehicle can decide a trajectory for itself which will be much more smoothened than the buffer method .This is because the weights added to compensate for the deviation will decide the velocity trajectory of the controlled vehicle based on the deviation from the median . In the other methods, the weights are abruptly added when the controlled vehicle crosses the limits which leads to sudden acceleration and deceleration causing the controlled vehicle to consume more fuel.

To begin obtaining simulation results, the several ways mentioned above are implemented. The first method is to simply assign the average velocity v_{des} as the velocity of the controlled vehicle at each time step and then observe the trend in the velocity profile of the controlled vehicle. The second method is to add buffer zones before the vehicle exceeds the maximum and minimum distance corridor limits. The idea is to increase the weights according to the buffer zone; that is as the target vehicle crosses the first minimum and maximum buffer limits, the weight must increase to compensate for exceeding the buffer limits. Then, if the target vehicle happens to further exceed the absolute maximum and minimum distance corridor limits, the weights must increase even more. The third method tested is to directly compensate for any deviation of relative distance from the median value of the maximum and minimum distance corridor as shown in figure 8. The idea is to allow the vehicle to smoothly add the velocity correction values to v_{des}, without abruptly changing the velocity when the target vehicle exceeds these bounds. This will help reduce the sudden acceleration and deceleration the target vehicle will experience upon realization that it has exceeded the distance corridor bounds.

2.3.4.1 Non-constraint

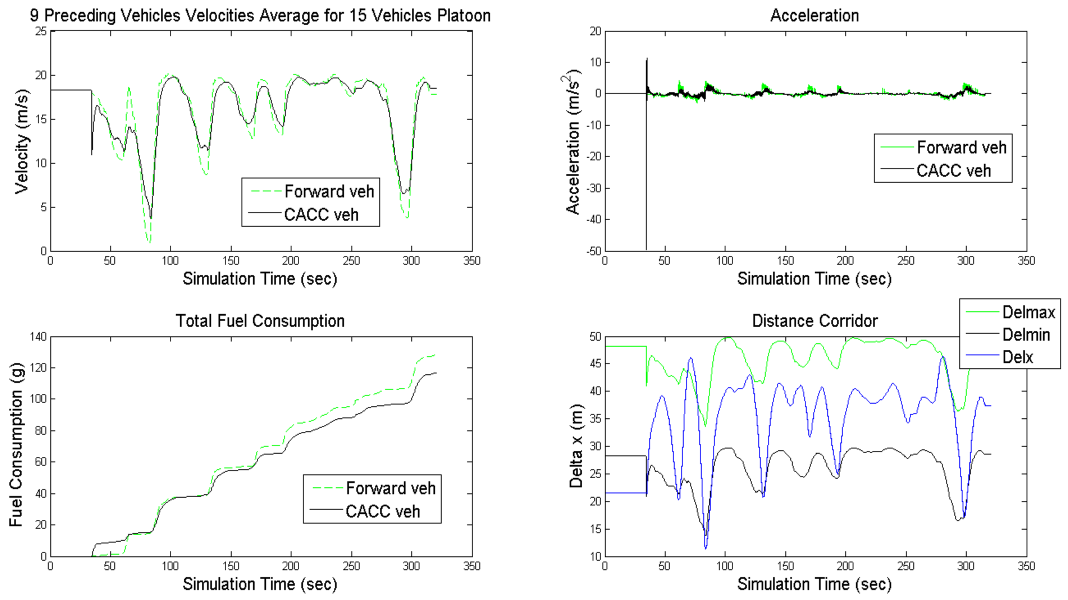


Figure 11.Simulation results for non-constraint method

As shown in figure 11 , with the non-constraint case, the CACC controlled vehicle velocity profile shown in the top left plot is time shifted on the left side. This time shifted velocity profile of the controlled vehicle,compared to the immediate preceding vehicle, makes it seem like as if the controlled vehicle is acting ahead in time to any situation in the traffic network. This means that the reaction of the controlled vehicle is occurring ahead of time when compared to that of the immediate preceding vehicle. However, with time shifted reaction, the controlled vehicle requires a larger limit of the distance corridor for it to maintain a safe distance from the preceding vehicle. This can be seen in the top right plot, where the difference in the distance between the preceding vehicle and the controlled vehicle varies in the range 10 to 150 m. In real traffic scenarios, such vast distances between vehicles will hamper mobility drastically which will be a huge cost in terms of travel time. However, the fuel consumption plot proves that there is potential space for fuel savings with averaging the preceding vehicles' velocities and using it as the velocity for the controlled vehicle. This is complemented by the comparison of the acceleration profile of the controlled vehicle and the preceding vehicle where the controlled vehicle has lower variations in its acceleration compared to the preceding vehicle's acceleration.

2.3.4.2 Buffer Zone

The results from implementing the buffer zone method are shown below. Similar to the non-constraint method, the buffer method is using the averaged velocity profile of the preceding vehicles. However there are observable peaks in the velocity profile of the controlled vehicle in the buffer method and these occurrences of the peaks are explained by the reduction of the distance corridor limits shown in the top right plot of figure 12. It can be seen that there are two buffer zones bordered with the blue line and the maximum and minimum distance corridor limits bordered by green and red colored lines respectively. The v_{cor} term comes in to action as soon as the controlled vehicle exceeds these buffer zones with a certain weight k. The weightage k increases further when the controlled vehicle exceeds the upper or lower limits of the distance corridor. It can be clearly seen in the velocity profile plot that the v_{cor} successfully tries to correct the velocity of the controlled vehicle by matching the velocity to the preceding vehicle whenever the limits are approached or crossed because the best way to correct the velocity at the limits is by following the preceding vehicle at its exact velocity. However, the corrections occur abruptly causing sudden accelerations and decelerations shown in the acceleration plot. This is complemented by the fuel consumption plot as the fuel consumption for the controlled vehicle soars higher than that of the preceding vehicle.

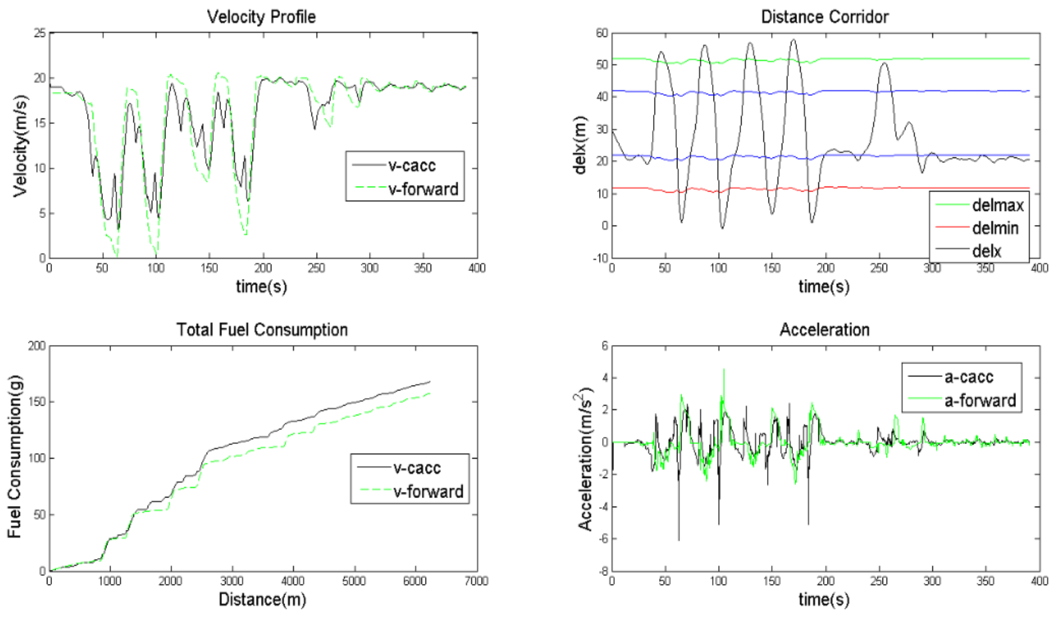


Figure 12. Simulation results for buffer zone method

2.3.4.3 Median Method

Figure 13 shows the results for the median method for 14 vehicles' velocities average with a constant time headway of 0.5 seconds. The fuel savings are approximately 8% and the distance corridor limits are much smaller than the previous methods. The fuel savings can be increased with a larger constant time headway or larger distance corridor bounds. With smaller distance corridor bounds the control effort increases, therefore the fuel benefits also decrease. As already discussed earlier, the median method uses the averaged velocity of the preceding vehicles. However, unlike the buffer zone method, the velocity profile of the controlled vehicle has no abrupt changes in its profile during instances when the vehicle surpasses the constraints. This can be seen from the comparison of the velocity plot of figure 12 and 13. The velocity profile derived using the median method behaves similar to buffer zone method where when the controlled vehicle approaches the limits, the controlled vehicle's velocity tries to approach the velocity of the immediate preceding vehicle. However, the advantage of using this median method over the buffer zone method is that an averaged velocity profile can be achieved without the occurrence of any sudden jerks as well as the distance corridor limits can be kept small. This is shown in the top right plot for the distance corridor in figure 13. From the plot, it is clearly visible that the controlled vehicle is made to remain within the bounds of the distance corridor as well as maintain a velocity profile which takes into account the average of the preceding vehicles without any abrupt changes in the velocity profile .This behavior of the controlled vehicle is definitely very beneficial in terms of fuel consumption as shown in the fuel consumption plot which is again complemented by the reduced magnitude of the controlled vehicle's acceleration.

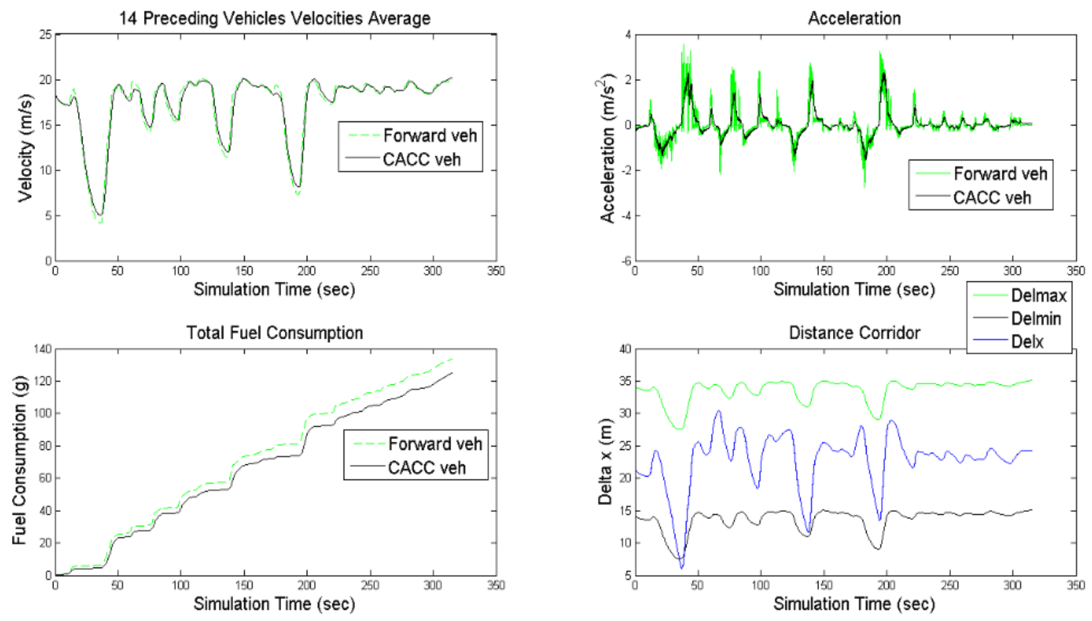


Figure 13. Simulation results for median method

There are three prominent conclusions that can be drawn from the above results. One is that , although taking the average of the preceding vehicles and using it as the reference velocity for the controlled vehicle gives fuel benefits, it compromises with safety of the vehicle if small distance corridor limits are used. To solve this problem, a correction term needs to be added. This is implemented in the buffer zone method , where the correction terms are added in piece wise manner. However, the implementation of correction term in a piecewise manner with respect to the buffers introduces sudden peaks in the velocity profile of the controlled vehicle which leads to sudden changes in acceleration. This behavior is undesirable if fuel consumption savings is one of the crucial objectives. The implementation of the median method takes care of these sudden changes in the velocity by applying a compensation in the reference velocity profile of the controlled vehicle in the form of the correction factor , to smoothly change the reference velocity of the controlled vehicle to cater to the soft and hard constraints mentioned earlier. The results obtained from the median method give the best fuel benefits while taking care of the distance corridor constraints.

2.4 Problem Formulation- Real-Time Implementation of CACC

From the above simulation results with the velocity trajectory equations, using the median method, highest fuel benefit was achieved at the same time the distance corridor limits were also maintained. The potential of the application of CACC using this heuristic method has been clearly shown. However, for real-time implementation, a dynamic model is necessary and not a velocity trajectory generator as mentioned in the previous section. The above formulation is designed to provide a velocity profile or trajectory for a controlled vehicle based on the averaged velocity v_{des} and corrected velocity v_{cor}. At every time step an averaged velocity v_{des} is calculated based on the instantaneous velocities of the preceding vehicles and then a v_{cor} is obtained based on the constraints. Using this velocity trajectory or reference equation is problematic when the median method is applied and the controlled vehicle’s initial position with respect to the immediate preceding vehicle is outside the bounds of the distance corridor. The controlled vehicle is made to operate at this initial speed for a time period before the CACC application is initiated as shown in figure 14. When the CACC application is initiated and applied to the control vehicle, the control vehicle experiences a surge in its velocity with a very high acceleration or deceleration depending on whether it is beyond the maximum or minimum distance corridor limits respectively. This occurs because the CACC controller with the velocity trajectory generator does not consider the initial dynamics of the vehicle. It realizes that the controlled vehicle is outside the bounds and immediately assigns a large v_{cor} value to bring the controlled vehicle within the bounds. The gain factor for the velocity correction term when the vehicle is outside the maximum and minimum limits is set to be high or in other words when the vehicle is outside the maximum or minimum distance corridor limits the hard constraints are implemented. Hence, in this specific scenario because the initial condition of the vehicle's position is beyond the minimum distance corridor limit, the hard constraints are implied directly leading the vehicle to suddenly reduce its velocity to bring itself within the bounds. However, in this process of bringing the vehicle within the bounds, the controller assigns a large velocity correction term which leads to drastic surges in the acceleration as shown in figure 14. The velocity plot in figure 14 shows a constant velocity for a period of approximately 50 seconds. During this period the CACC application has not started. At around 50 seconds when the median method CACC application is implemented, the vehicle's velocity suddenly dips to compensate for the initial Δx that is beyond the minimum Δx_{min} as shown in the distance corridor plot in figure 14. This sudden change in velocity is shown, in the acceleration plot, by the unrealistic acceleration of approximately 10 to -50m/s^2 in a matter of milliseconds.

2.4.1 The Dynamic Model Equations

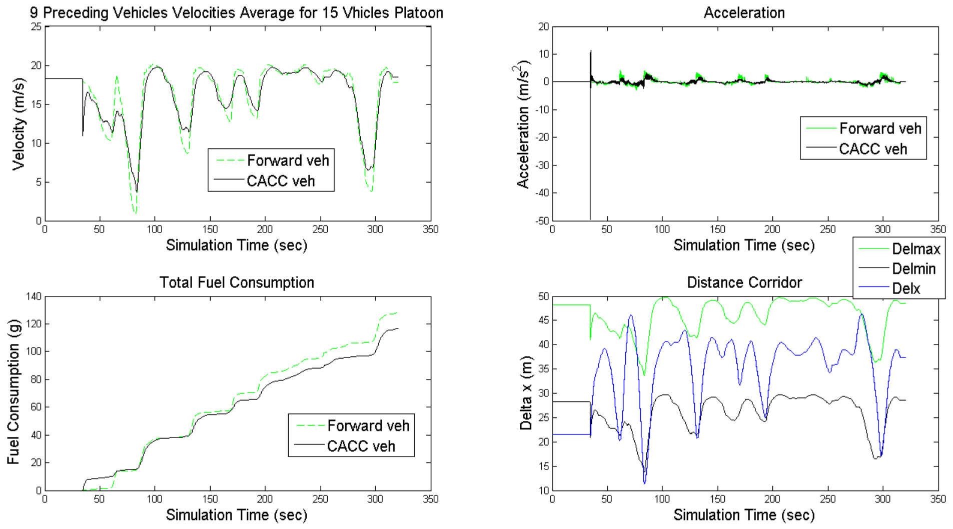


Figure 14.9 vehicle average using reference generator model with initial $\Delta x < \Delta x_{min}$

To tackle this challenge, the formulation is changed to implement a dynamic model.

The dynamic model equations are given below in discrete time form.

$$\Delta x(t+1) = \Delta x(t) + dt * (v_p(t) - v(t))$$

$$v(t+1) = v(t) + dt * u(t)$$

In the above dynamic equations, a method has to be determined to solve for u(t), the acceleration for the vehicle, at every time step. The method used to determine the acceleration at every time step is done using an objective function as explained below . The objective function is designed using the intuition obtained from the reference velocity generator simulation results from the previous section.

2.4.2 The First Objective of the Function

Given the dynamic model of the system, an objective function with specific objectives has been designed. The ultimate objective of the cost function J is to provide u, the acceleration for the controlled vehicle which will track the reference speed v_{des}, the averaged velocities of preceding vehicles. From the non-constraint method simulation results it is known that using the averaged velocity of the preceding vehicles provides potential fuel benefits. A simple cost function representing this is shown in the equation below.

$$J_t = (v_{des}(t) - v(t+1))^2$$

Where after expansion

$$J_t = \left(v_{des}(t) - v(t)\right)^2 - 2dt * u(t)\left(v_{des}(t) - v(t)\right) + dt^2 u(t)^2$$

Differentiating the above cost function with respect to u(t) and solving for u(t) will give

$$u(t) = \frac{v_{des}(t) - v(t)}{dt}$$

Substituting the above relation in the dynamic equation will provide the following relation

$$v(t+1) = v_{des}(t)$$

Thus using the simple objective function will fulfill the objective of following v_{des} as a reference. However, it is not desirable that the velocity of the controlled vehicle completely track v_{des} because from previous results of the non-constraint method, it is observed that although following v_{des} has potential for fuel benefits, the vehicle requires large distance corridor boundaries to function safely in a traffic network. Thus the objective function must be modified to include distance corridor constraints. Therefore, the objective function J has constraints with weights added to it as a part of its objective. These constraints are considered with regards to the distance corridor boundaries to provide a suitable acceleration u(t) for every time-step such that the velocity of the controlled vehicle tracks v_{des} but also takes into account the distance corridor boundaries. The two other objectives of the cost function with respect to maintaining the constraint are that the controlled vehicle must track the immediate preceding vehicle's velocity depending on the weight W_1 and the acceleration of the controlled vehicle u(t) must compensate for the deviation of Δx from Δx_{med}.

2.4.3 The Second Objective of the Function

The objective of the controlled vehicle tracking the immediate preceding vehicle's velocity is intended to add a high cost when the weight W_1 increases. The weight W_1 is an exponential function similar to k_1 mentioned previously. The idea is to increase the weight W_1 exponentially to the deviation of Δx from Δx_{max}, Δx_{min} and Δx_{med}. This objective of the function acts like a constraint that was being added in a piece wise manner previously. The weight W_1 only increases when the controlled vehicle approaches the distance corridor limits very closely. Increasing weight causes the cost of the difference of velocities between the preceding and the controlled vehicle to increase. In other words, when W_1 is high, the objective function J will provide an acceleration u(t) which will attempt to follow the immediate preceding vehicle's velocity. This is desirable for two specific situations. In a situation when the controlled vehicle is travelling at a velocity higher than the preceding vehicle and it is approaching the preceding vehicle, the only way to prevent the controlled vehicle from hitting the preceding vehicle is to bring the controlled vehicle's velocity to be at the same level as the preceding vehicle before the controlled vehicle crosses Δx_{min}. In the same way if the controlled vehicle is travelling at a velocity lower than the preceding vehicle which leads it to recede from the preceding vehicle , then the only way to stop this recession is by following the preceding vehicle's exact velocity. Therefore, with this objective, the function is altered as follows:

$$J_t = (v_{des}(t) - v(t+1))^2 + \left(v_{pn}(t) - v(t+1)\right)^2 * W_1$$

Where v_{pn} is the instantaneous velocity of the immediate preceding vehicle.

2.4.4 The Third Objective of the Function

From the two terms in the above objective function , an acceleration can be derived that will balance the vehicle's velocity to follow between the averaged velocity and the preceding vehicle's velocity. However, it is observed from preliminary results that the second term is not enough to keep the controlled vehicle within the distance corridor bounds. Hence a third term with the objective of compensating the deviation of Δx from Δx_{med} is added as an extra constraint. This term is added to the objective function to implement the median method which has been already discussed previously. The idea is to add the cost of the deviation based on a weight W_2 which is a constant value of small magnitude. As the deviation of Δx from Δx_{med} increases the cost due to the third term increases. The weight W_3 is added with the purpose of keeping the units consistent through out the cost function. The first two objective functions have a unit of m^2/s^2 and the third objective has units of m^2/s^4. Therefore W_3 is $1s^2$, to keep the unit m^2/s^2 uniform throughout the cost function

Cost Function

$$J_t = (v_{des}(t) - v(t+1))^2 + \left(v_{pn}(t) - v(t+1)\right)^2 * W_1$$
$$+ W_3\left((u(t) - W_2(\frac{\Delta x(t) - \Delta x_{med}(t)}{dt^2})\right)^2$$

The first two objectives of the function are evident but the last term in the above function needs more explanation. When the last objective in the above function is expanded it becomes:

$$(u(t) - W_2(\frac{\Delta x(t) - \Delta x_{med}(t)}{dt^2})) = u(t) - W_2(\Delta x(t) - \frac{\Delta x_{min,0} + \Delta x_{max,0} + 2hv(t)}{2})/dt^2$$

The substitution for Δx and Δx_{med} are obtained from the dynamic equation and definition of Δx previously mentioned. Rearranging the above equation, it gives the following:

$$(u(t) - W_2(\frac{\Delta x(t) - \Delta x_{med}(t)}{dt^2})) = u(t) - W_2(\Delta x(t) - \frac{\Delta x_{min,0} + \Delta x_{max,0}}{2} - hv(t))/dt^2$$

The above expansion clearly shows the physical meaning of the objective of the function. The last term in the function is responsible for adding a cost when the Δx deviates from the median of the maximum and minimum distance bounds when the vehicles are not in motion and plus the second term in the above equation compensates for any deviation from the assumed constant time head way distance to the preceding vehicle. This means that this complete term will make sure that the vehicle lies within a specific range of the median of the stand still distance between the controlled and the preceding vehicle as well as within the constant time headway of one second which was previously chosen.

The ultimate objective of the combined function J is to provide the acceleration u for the least cost that will satisfy each of the objectives at every time step. Unlike any other cost function which is designed with the objective of optimization over a specific prediction horizon, this function's sole objective is to provide an acceleration for every time step. There is no prediction used over a time horizon, because the data available for v_{des} , v_p , Δx and Δx_{med} are for the current time step only. Therefore, this objective function does not use any prediction horizon. To determine the least cost J ,the function is differentiated with respect to u and equated to zero for every individual time step. The derivative is then solved for u online.

The equation for u is given below.

$$u(t) = \frac{\left(dt * \left(v_{des}(t) - v(t)\right)\right) + \left(W_1 * dt * \left(v_{pn}(t) - v(t)\right)\right) + \left(W_3 * W_2 * \left(\frac{\Delta x(t) - \Delta x_{med}(t)}{dt^2}\right)\right)}{(dt^2 + W_1 * dt^2 + W_3)}$$

Where

$$v_{des}(t) = \frac{v_{p1}\,(t) + v_{p2}(t) + v_{p3}(t) + ..\, v_{pn}(t)}{n}$$

$$v_{pn} = velocity\ of\ immeidate\ preceding\ vehicle$$

Looking at the objective function equation J, it can be argued that if the objective is to reduce fuel consumption, then the objective function must be a function of the fuel consumption. However, although the objective is to reduce fuel consumption, the approach taken here is to understand the dynamics of the controlled vehicle based on the average velocities of the preceding vehicles with regards to improvements in fuel economy. The objective function implemented is just used as a tool to prove that following the averaged velocity with some constraints has potential fuel benefits. The objective function is also able to handle any drastic surges in the velocity because it provides an acceleration u fulfilling the three objectives. Therefore, it means that there will not be any drastic changes in u from one time-step to another, if the three objectives have to be maintained.

In figure 15, it is clearly seen that using the dynamic model with the objective function mentioned above, prevents any aggressive acceleration or deceleration behavior of the preceding vehicle. The distance corridor plot shows that even though the initial $\Delta x < \Delta x_{\min}$, when the controlled vehicle enters the VISSIM Traffic simulator, the controlled vehicle is gradually brought back within the distance corridor boundaries without any unusual surge or dip in the vehicle's velocity unlike in the previous model shown in figure 14.

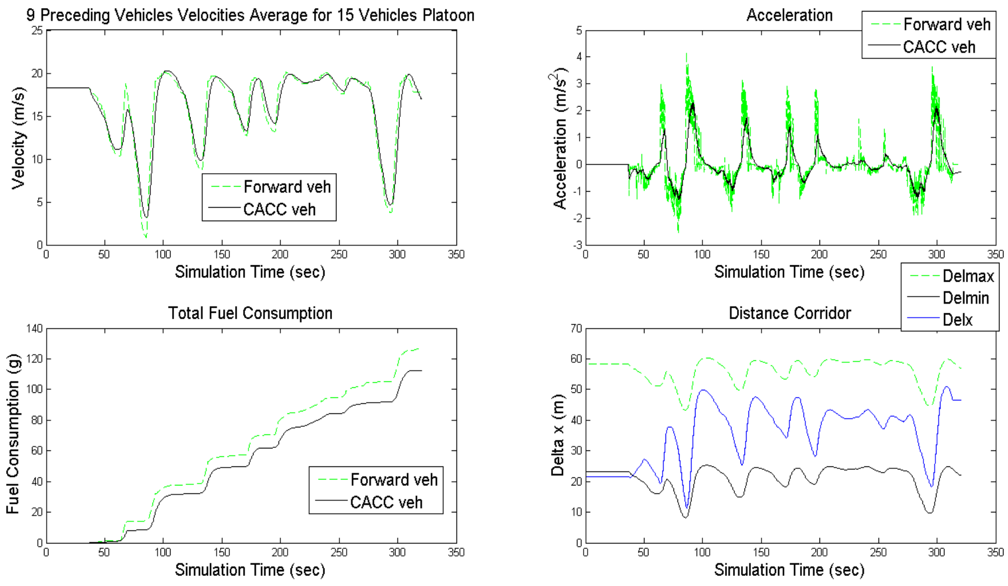


Figure 15. 9 vehicle average using dynamic model with initial $\Delta x < \Delta x_{min}$

2.4.5 Evaluation of the Effectiveness of Objective Function

The effectiveness of the objective function J can be arbitrated based on the comparison of the actual velocity profile of the vehicle and the averaged velocity profile v_{des}. As already known, the objective function must provide an acceleration based on the weightage of each objective. Therefore, comparing v_{des} (averaged preceding vehicles' velocities),tracking which is one of the objectives of the function, with the actual velocity of the vehicle will showcase the extent to which the different objectives have contributed in determining the actual velocity of the controlled vehicle. Figure 16 is used to analyze the operational functionality of the objective function. The four plots in figure 16 compare the actual velocity profiles of 14 and 2 vehicles velocities averages, actual velocity profile and 2 vehicles average velocities, actual velocity profile and 14 vehicles velocities average and the comparison of the actual velocity profiles derived from 14 and 2 vehicles velocities averages. The difference in the average velocity profiles for 14 and 2 vehicles profile is very evident. From the second plot of actual velocity profile comparison with 2 vehicles average, it can be said the velocity profiles are very similar because taking the average of just 2 preceding vehicles is almost like following the immediate preceding vehicle. However, the trend completely changes in the third plot comparing the actual velocity profile with 14 vehicles' velocities averages. In the third plot the 14 vehicles average velocities profile is very different from the actual velocity profile of the controlled vehicle. This plot clearly shows that the objective function is effectively influencing the velocity profile of the controlled vehicle depending on the weightage of each objective of the function. The fourth plot compares the velocity profiles from the two different vehicle averages of 2 and 14 vehicles. This plot clearly demonstrates that the intuition previously gained that averaging more preceding vehicles' velocities clearly provides a smoothened velocity profile for the controlled vehicle. These plots clearly exhibit the effort the objective function puts in determining the final velocity profile of the controlled vehicle in two different cases of 2 and 14 preceding vehicles' velocities averages. These results clearly show that the objective function is using the different weightages on each of the objectives to determine the actual velocity of the controlled vehicle at every time step.

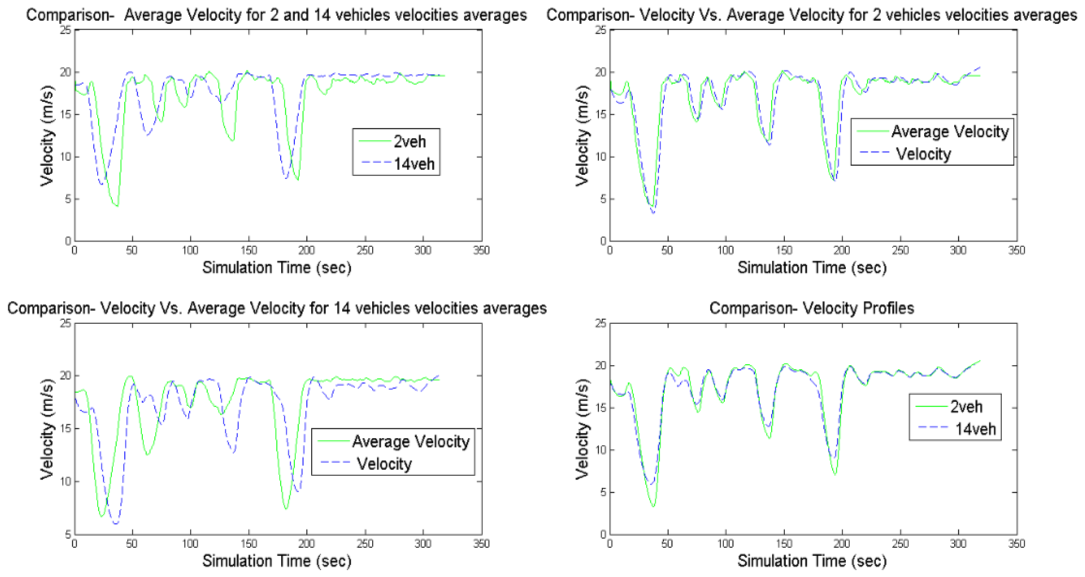


Figure 16. Comparison of averaged and actual velocity profiles for 2 and 14 preceding vehicles.

2.5 Simulation Results of Scenarios

The evaluation of CACC is conducted for several different scenarios. V_{des} is the average of the preceding vehicles' velocities

$$\mathrm{v}_{\mathrm{des}}(t) = \frac{v_{p1}\,(t) + v_{p2}(t) + v_{p3}(t) + ..\, v_{pn}(t)}{n}$$

a. Average of 14 preceding vehicles' velocities for city driving
b. Average of 8 preceding vehicles' velocities for city driving
c. Average of 14 preceding vehicles' velocities for highway driving
d. Average of 8 preceding vehicles' velocities for highway driving
e. Varying platoon size – city driving

The simulation results are presented in four different plots: vehicle velocity, distance corridor, total fuel consumption and acceleration. The calculation for cumulative fuel consumption is obtained by taking the summation of fuel consumption at each time step for the whole driving cycle. The fuel consumption calculated for the simulation of different scenarios is based on vehicle power request as shown in the equation below

$$P_{req}(t) = \left(ma(t) + mgsin\emptyset(t) + \ \mu mgcos\emptyset(t) + \frac{1}{2} * C_D \rho_a A v(t)^2\right) * v(t)$$

Where g is gravity , $\emptyset(t)$ is terrain slope of zero, μ is the rolling resistance constant, C_D is the drag coefficient, ρ_a is the density of air, A is the vehicle frontal area. The above equation represents the combined power expended over the driving cycle. In[12], a linear equation of power request against fuel consumption is determined. The fuel consumption for different engine operating points was found by using the engine map[20].

The objective in this approach is to use more than one forward vehicles' information to evaluate CACC and determine if there are any potential fuel consumption savings. V_{des} in this case takes the average of the preceding vehicles' velocities at every time step. From literature, it can be inferred that providing future information to the controlled vehicle has some potential benefits. Similarly, if the velocity of the controlled vehicle is a function of the preceding vehicles' velocities, then it can be said that taking the average over the preceding vehicles' velocities is like taking the average of the prediction of the velocity of the controlled vehicle. This is based on the assumption that the controlled vehicle will more or less follow the trajectory of the preceding vehicles. For instance, if the lead vehicle in a platoon stops at a traffic signal, then it can be assured that the controlled vehicle may not completely stop at the signal but it will at least slow down to maintain a safe distance from the immediate preceding vehicle which itself will slowdown following the reaction of the first lead vehicle. Thus it is inferred that in a string of vehicles the reaction of the lead vehicle to any change in traffic is passed on to all the other lagging vehicles. By taking the average of the instantaneous velocity of the forward vehicles at every time-step, and using it as the velocity of the controlled vehicle incorporates these reactions of the preceding vehicles in the controlled vehicle ahead of time .The simulation is conducted for several different cases. In one case, the controlled vehicle is placed behind eight preceding vehicles as shown in figure 17 and in the second case, the controlled vehicle is placed behind 14 preceding vehicles as shown in figure 23. The intention of obtaining simulation results for two different cases is to see the impact of different velocity profiles of the preceding vehicles on the velocity profile and fuel consumption of the controlled vehicle. These two cases are varied between city and highway driving scenarios. In the final case, a varying platoon size is simulated.

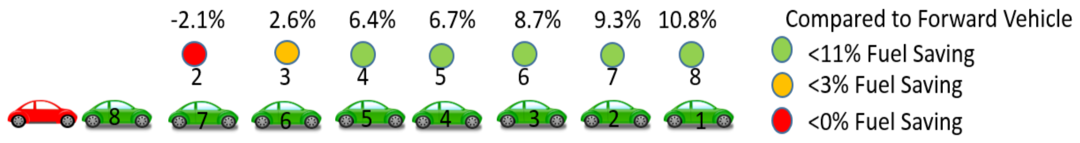


Figure 17. Fuel economy trend based on number of vehicle averages for 8 preceding vehicles-city driving

2.5.1. 8 Vehicles' Velocities Average- City Driving with 0.1s constant time headway

The controlled vehicle is placed in the ninth position in the string of vehicles and the fuel consumption for the corresponding number of vehicles averaged is shown above each vehicle. From the results it is observed that at only two vehicles' averages, there is no fuel benefit. For this case specifically, the constant time headway is chosen to be 0.1seconds for observation purposes. For all the cases after this the constant time headway is chosen to be one second as decided earlier. With a smaller constant time headway, the fuel consumption is expected to be higher as the control effort will be higher. The effect of constant time headway on the control effort is discussed in detail [27]. However, as the number of vehicle averages increase the fuel benefit also increases proportionally. The specific plots for eight vehicles' velocities average are plotted in figure 18. Eight vehicle's average has the highest fuel economy when compared to fewer vehicles velocities averages. The limits of distance corridor is between 10 to 30 m, which is a reasonable value for a realistic traffic network, and it can be seen for a better fuel economy, the controlled vehicle seems to utilize this limited space extensively throughout the driving cycle. This tells that the role of the correction factor is vital in making sure the vehicle remains within these bounds. The fuel consumption plot clearly shows that there is considerable fuel benefit, about 10.8%, which is complemented by the lower magnitude of the controlled vehicle's acceleration compared to that of the immediate preceding vehicle .

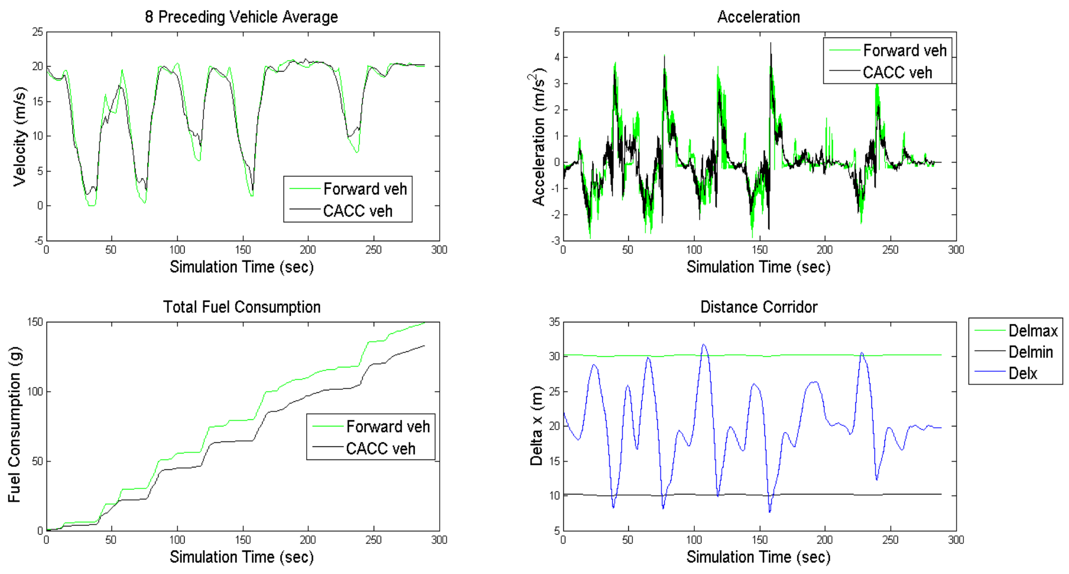


Figure 18. 8 vehicles' velocities average simulation results- city driving

2.5.2 8 Vehicles' Velocities Average- City Driving

The eight preceding vehicles' average simulation is repeated for a less aggressive preceding vehicles profile with city driving conditions and a highway situation with no stops. Here the constant time headway is chosen to be one. Fuel consumption savings are expected to increase with a higher constant time headway as control effort for the vehicle reduces. Figure 19 shows the fuel benefit trend for the city driving case and the velocity, acceleration, fuel consumption and distance corridor plots are shown in figure 20. It can be clearly seen that, the trends in fuel benefits are similar to the earlier case where the fuel benefit increases with increasing vehicle velocity averages.

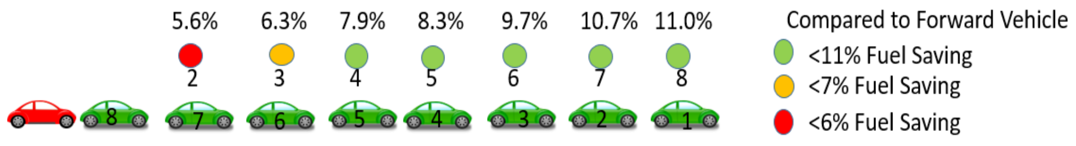


Figure 19. Fuel economy trend based on number of vehicle averages for 8 preceding vehicles–city driving

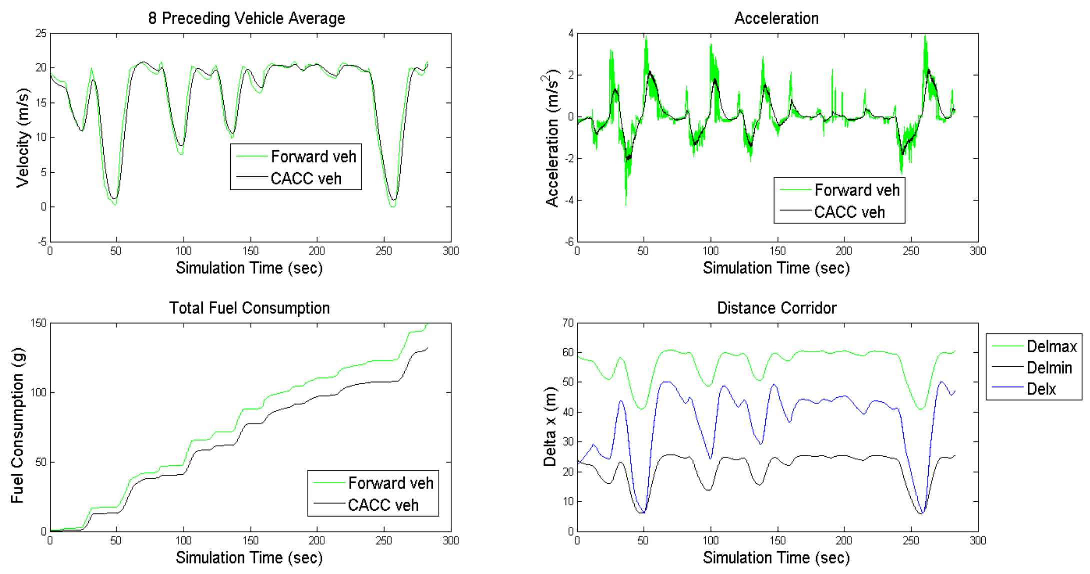


Figure 20. 8 vehicles' velocities average simulation results -city driving

2.5.3 8 Vehicles' Velocities Average- Highway Driving

For the highway case, the eight vehicles' average fuel benefit trends remain the same. However, the magnitude of fuel benefit decreases as expected. With less aggressive driving trajectory , where the preceding vehicles do not make a stop and keep tracking an almost uniform velocity for the whole driving cycle, there is not enough room for improvements in fuel consumption for the controlled vehicle. From the fuel consumption plot in figure 22, the magnitude of fuel consumption (approximately 85g) compared to figure 20 (approximately 140g) is much lower and so are the fuel benefits. Figure 21 shows that like the previous two cases, the fuel benefit increases with the increasing number of averages but the maximum benefit achieved is 4.8%.

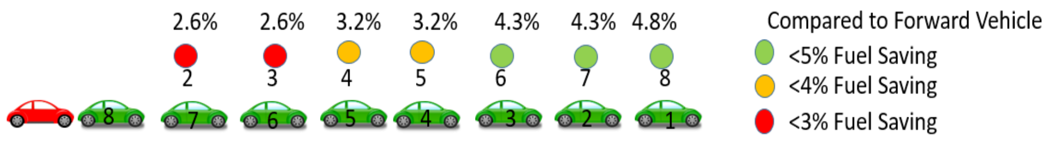


Figure 21. Fuel benefits for 8 vehicles' velocities average - highway case

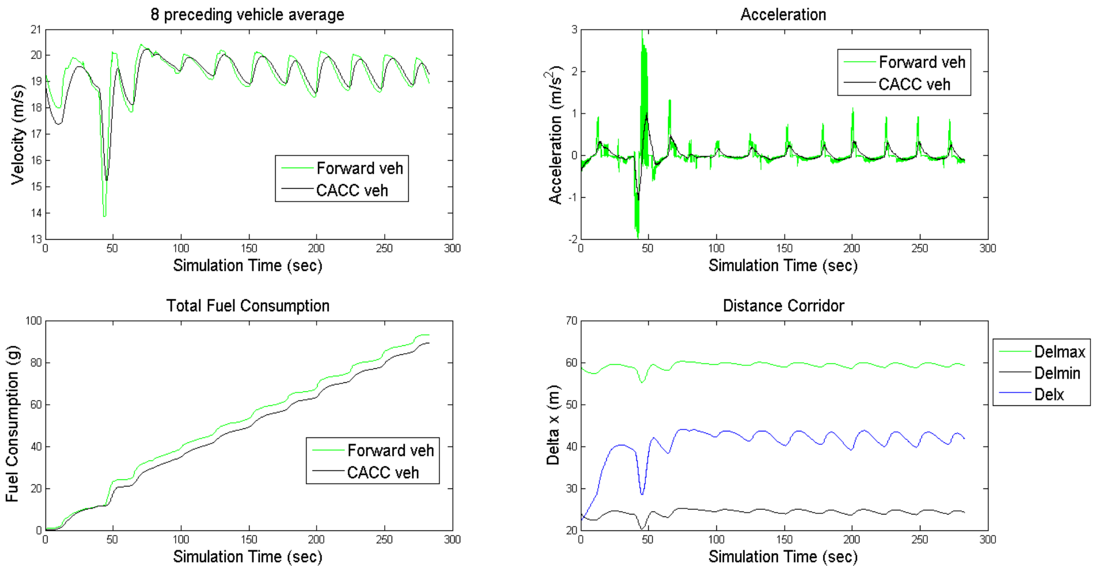


Figure 22.Plots for 8 vehicles' velocities average -highway case

2.5.4 14 Vehicles' Velocities Average- City Driving

For the 14 preceding vehicles scenario, the same cases of simulation were conducted: city and highway driving cases. For the city driving case, from figure 23, it can be clearly seen, that the fuel economy trend is similar to what was seen in 8 vehicles' average; that is the fuel economy has an increasing trend as the averages increase. Similar to 8 vehicles' averages, the controlled vehicle tends to use the given distance corridor space to move itself within the expanse of it extensively, meaning that the correction factor is consistently working to bring the vehicle to its median position whenever the controlled vehicle tends to deviate. The fuel consumption plot shows that there is significant fuel benefit which is complemented by the reduced acceleration of the controlled vehicle shown in the acceleration plot. The maximum magnitude of fuel benefit increases to 17.5% for 14 vehicles' average compared to 11.0% savings for the eight vehicles' average shown in figure 19.

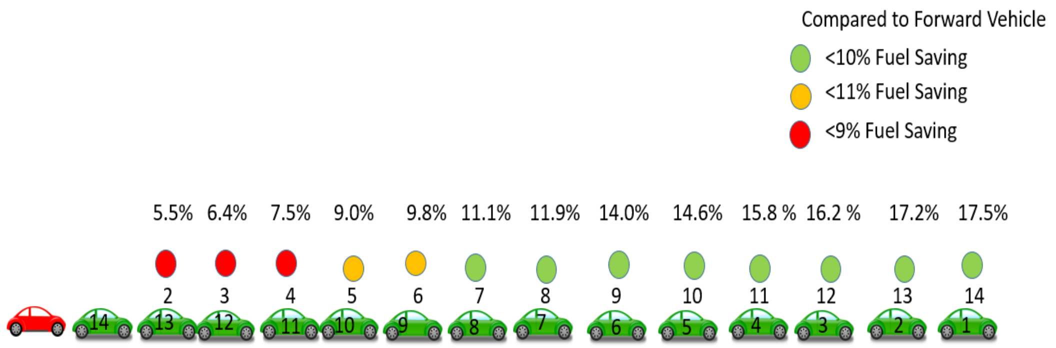


Figure 23. Fuel economy trend based on number of vehicle averages for 14 preceding vehicles –city driving

The reason for the increase in fuel benefit is because of the reduced magnitude of acceleration for the controlled vehicle shown in figure 24. The acceleration comparison for the controlled vehicle and the preceding vehicle shows that the controlled vehicle has a suppressed acceleration. From the velocity plot in figure 24, it can be seen that the controlled vehicle never stops completely whereas, the preceding vehicle stops once and decelerates a second time around 250 seconds. These drastic changes in vehicle velocity are the reason why the acceleration of the preceding vehicle is much more aggressive than the controlled vehicle's acceleration.

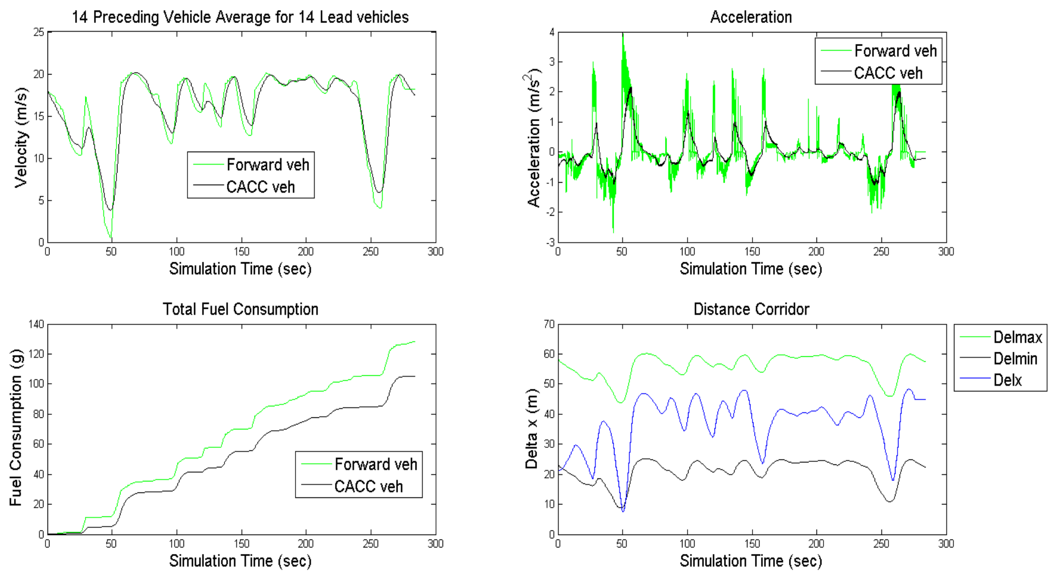


Figure 24. 14 vehicles' velocities average simulation results –city driving

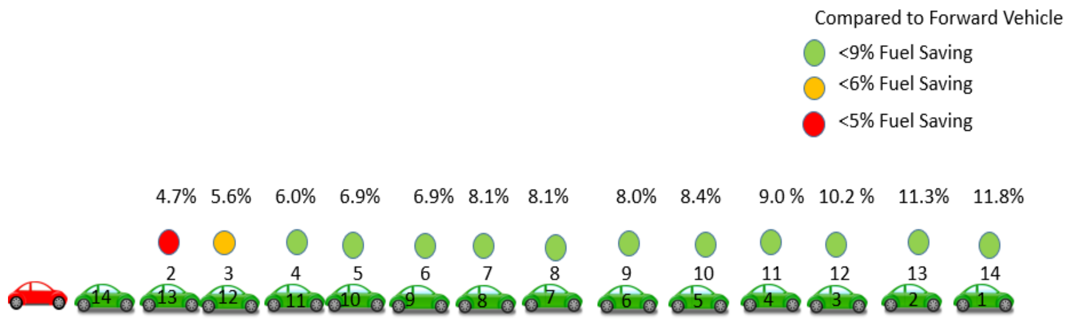


Figure 25. Fuel economy trend based on number of vehicle averages for 14 preceding vehicles - highway case

2.5.5 14 Vehicles' Velocities Average- Highway Driving

Figure 25 shows the fuel economy trend for 14 vehicles' averages for a highway situation. A highway situation is different from city driving cases because it has no major deceleration or acceleration trends and the velocity of the vehicles in the platoon fluctuate around a specific range of velocities. The idea is to determine the fuel benefit for the controlled vehicle when it is running in a highway situation for its whole driving cycle. From the previous two cases , it is known that stop and go scenarios give significant fuel benefits. Thus it can be inferred that for a highway situation, fuel benefits will exist but they will be limited due to the less aggressive behavior of the vehicles. This inference is proved by the fuel benefits shown in figure 25 where the maximum fuel benefit achieved is 11.8% .From figure 26, the acceleration plot clearly shows that the acceleration behavior of the controlled vehicle as well as the immediate preceding vehicle is much suppressed compared to the earlier cases of simulation, and hence the fuel benefits are reduced.

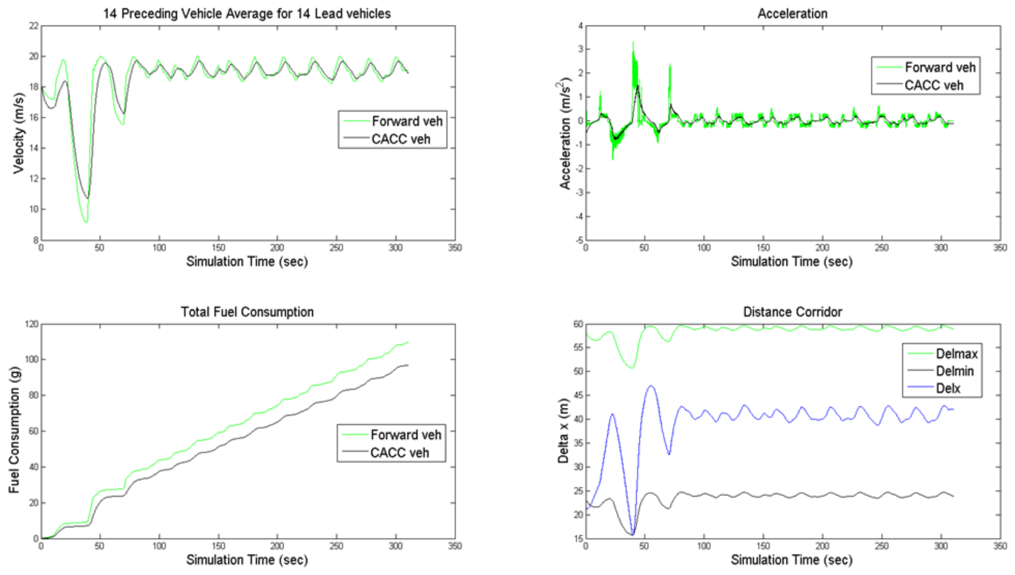


Figure 26. 14 vehicles' velocities average simulation results - highway case

2.5.6 Preceding Vehicles' Velocities Average- Varying Platoon Size

A separate case is simulated where the size of the platoon of vehicles vary randomly. This means that depending on the traffic signal timing, the number of preceding vehicles change. There are instances when 14 preceding vehicles are in the platoon, and instances when just two preceding vehicles exist because the other twelve did not stop at the signal. This case is simulated on the same traffic network as shown in figure 29. VISSIM-COM was modified to accommodate the varying number of preceding vehicles in the platoon. The logic used in the code was such that the controlled vehicle can be fed in with a maximum of 14 preceding vehicles' information as well as the state of the next traffic signal is also continuously fed as an added information to the controlled vehicle. If at any point the state changes to red, then the controlled vehicle can obtain information for only those preceding vehicles who have stopped at the signal. On the contrary, if the signal turns green, the controlled vehicle can access information for preceding vehicles who are within 300m of distance from the controlled vehicle. The figure below shows that even with varying platoon size, the fuel savings are approximately 10.4% compared to the immediate preceding vehicle. The velocity plots show the velocity trajectory for the controlled and immediate preceding vehicle as well as the average of the preceding vehicles' velocities in the platoon. It can be seen that the profile for the controlled vehicle is very different from the averaged velocity. This clearly tells that the controller is working to make sure that the constraints are maintained for a safe ride. If the controlled vehicle is to follow the averaged velocity profile completely, then a safe driving cannot be assured, as the averaged velocity profile can lead the controlled vehicle to exceed the distance corridor bounds. One such close counter with the preceding vehicle can be seen in the distance corridor plot. At around 170seconds , the controlled vehicle happens to cross the minimum distance corridor bounds slightly. However, the controller manages to pull back the vehicle within the bounds quickly, making sure there is a minimum safe distance between the preceding and the controlled vehicle.

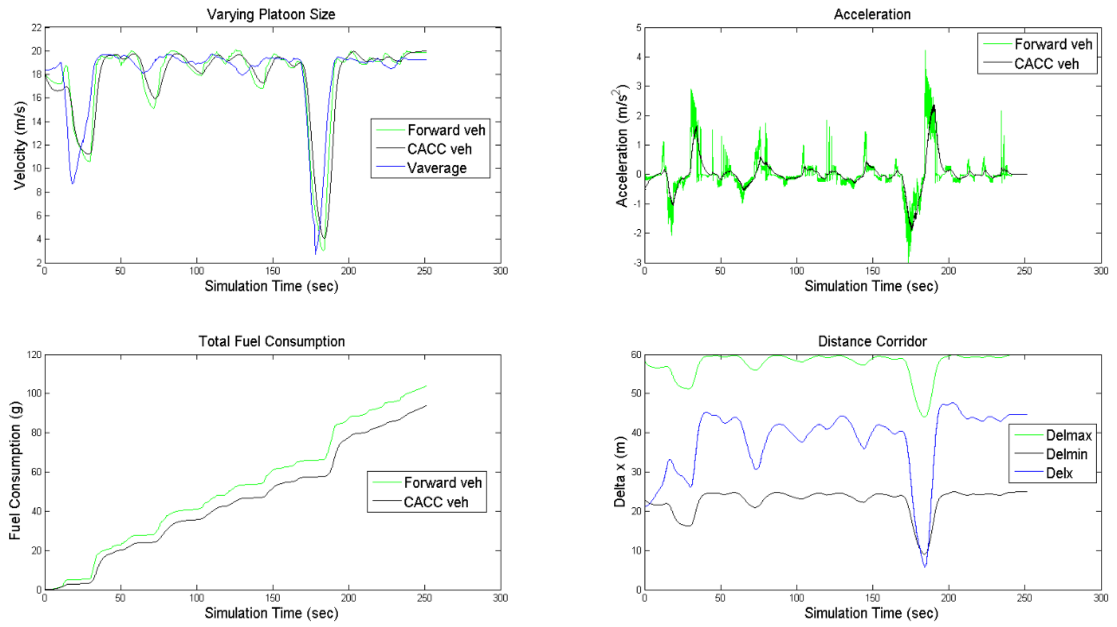


Figure 27. Varying Platoon Size Vehicle dynamics and fuel consumption

2.6 Conclusion

After conducting simulations for the average velocity method, the observations from the results are compared in this section. It is to be noted that the median method is the common methodology implemented across all the scenarios.

Averaged Velocity Method

1. **a) speed average with 14 preceding vehicles, without prediction-city**

   b) speed average with 8 preceding vehicles, without prediction -city

   c) speed average with 14 preceding vehicles, without prediction -highway

   d) speed average with 8 preceding vehicles, without prediction –highway

   e) Varying Platoon size- city

Table 1. Fuel Economy Comparison Table

Scenario	Highest Fuel Economy(%)
1a	17.5
1b	11.0
1c	11.8
1d	4.8
1e	10.4

From the table it is clear that scenario 1a achieves high fuel economy. All the scenarios, with averaged velocity approach, give promising results where average is taken with respect to the velocities of the preceding vehicles without any sort of prediction mechanism. From the results, it can be concluded that there is potential fuel benefit in taking the average of increasing number of preceding vehicles. However, only taking the average of the preceding vehicles' velocities is not sufficient to maintain the constrains. Therefore constraints are taken care by adding objectives to the objective function as discussed earlier.

From the above results obtained for the simplified, averaged velocity approach it is observed that across all the cases simulated, the fuel benefit increases with the increasing number of preceding vehicles' velocities average. This is clearly seen in the comparison between 8 and 14 vehicles' velocities average that the fuel savings of 14 vehicles' velocities average is higher. This is the result of the phenomenon previously discussed that as more traffic information is accessible by the controlled vehicle, the potential for fuel savings also increases. In this method, providing the vehicle speeds of the preceding vehicles to the controlled vehicle incorporates the preceding vehicles' dynamics in the controlled vehicle's dynamics, which allows the controlled vehicle to plan its trajectory ahead of time and obtain high fuel economy. This heuristic method does not implement any prediction model but assumes that complete information from preceding vehicles is available either through IVC or VII at every time step. Both results obtained from constant platoon and varying platoon size showcase that there is significant potential for fuel savings. However there is variance in the magnitude of fuel savings from the highway and the city driving cases. This is clearly because the velocity profiles of the city driving case are much more aggressive than the highway driving case where the vehicle velocity fluctuation is minimal. Thus the room for fuel savings is also lower for the highway case. In conclusion from the simulation results, it can be claimed that the heuristic, averaged velocity approach can provide fuel savings in the ranges of 10-17% across all scenarios of traffic by incorporating preceding vehicles' information.

Chapter 3 Software Architecture Development for HiLS Integration

3.1 Middleware Structure between Powertrain Research Platform and VISSIM

The structure of communication between Powertrain-Research-Platform and VISSIM is shown in figure 28 below. It can be seen that the left half of the structure consists of the research platform and the right half consists of the VISSIM traffic simulator. To execute real-time experiment, the powertrain-research-platform needs to communicate over a network with the traffic simulator in synchronization to obtain vehicle speed information for the powertrain-research-platform.

The remote machine running VISSIM is more specifically used to run traffic simulation to communicate the vehicle dynamics with the powertrain-research-platform. As it has been emphasized earlier that all simulations should be in real time, it is important to integrate the software in such a way that the data transfer will be highly efficient, enabling the whole simulation to be carried out in real time. To achieve this, major calculations and processes will have to take place online.

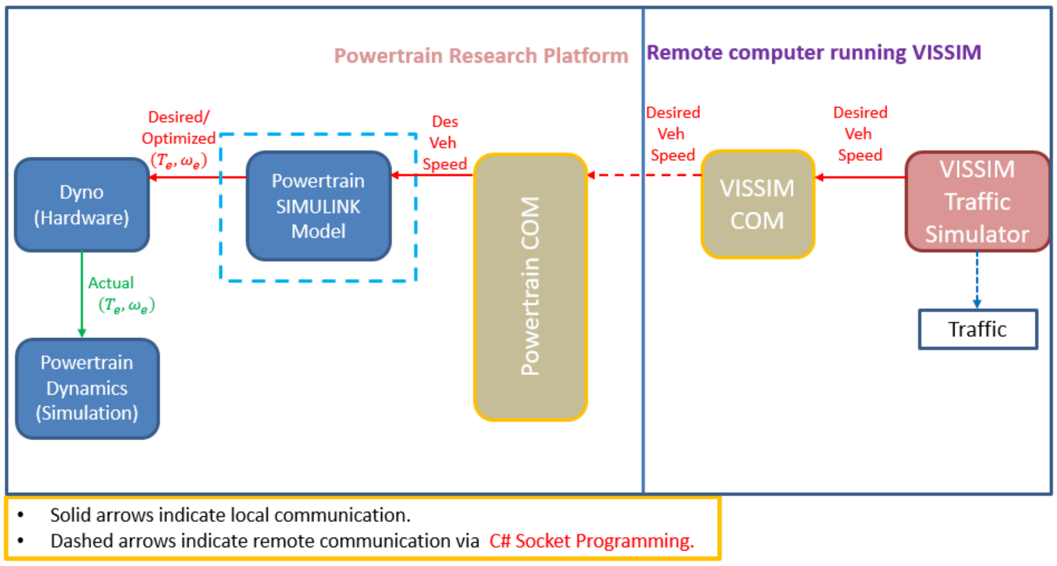

Powertrain Research Platform
Remote computer running VISSIM
Desired/ Optimized (T_e, ω_e)
Dyno (Hardware)
Powertrain SIMULINK Model
Des Veh Speed
Powertrain COM
Desired Veh Speed
VISSIM COM
Desired Veh Speed
VISSIM Traffic Simulator
Traffic
Actual (T_e, ω_e)
Powertrain Dynamics (Simulation)
• Solid arrows indicate local communication.
• Dashed arrows indicate remote communication via C# Socket Programming.

Figure 28. Structure for collaborating Powertrain Research and VISSIM

VISSIM is set up for different simulation parameters. The simulation used is a replica of a straight road with seven traffic signal junctions, as shown in figure 29. The simulation run time is set for five minutes and VISSIM is commanded to run a single step of 200 milliseconds to collect vehicle data like vehicle speed. VISSIM simulation also designates an unique identification number to the vehicle during the simulation which helps determine if the vehicle is within the traffic network. This vehicle identification number can be accessed by C# and all the details related to this specific vehicle can be extracted as well.

The current set up in the program is to run single step at 200 milliseconds. However, it has to be investigated that whether the vital tasks of sending and receiving data are reaching a stage of completion before the 200 millisecond time constrain. If a delay occurs in any of the processes, it will lead the single step time to exceed 200 milliseconds. To figure out the solution to this problem, it is important to analyze the time taken for VISSIM to run single step and extract data individually as well as figure out how much time it takes C# to pass data to MATLAB.

For analyzing the time for VISSIM running single step, simulations were run for different traffic densities on the road. The traffic density was increased for each simulation so that the total number of vehicles increased from 200 to 800 vehicles. The reason for varying the traffic density each time is to observe the change in execution time for VISSIM software with large number of vehicles in the simulation. Before running any analysis, it is hypothesized that as the traffic density increases the run time for VISSIM also increases.

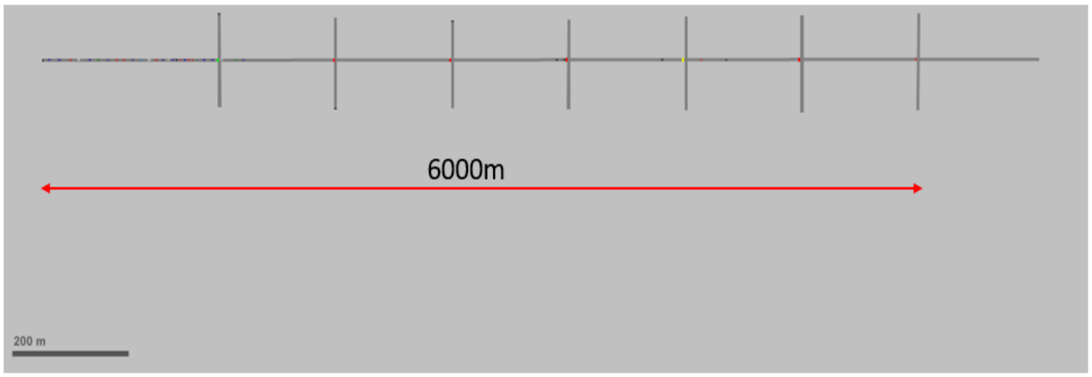


Figure 29. Single lane, seven traffic junctions, Traffic network

3.1.1 Basic COM Communication with VISSIM

The Component Object Model (COM), in this case C# program, will be local to VISSIM. It will be the pathway to communicate with VISSIM as well as communicate to other platforms as shown in figure 30 with the blue and green arrows respectively. The idea is to have several other platforms like the signal controller cabinet, connected vehicle controller to communicate with the COM effectively. In the initial stages of the project, the middleware or the COM was not included. Hence, VISSIM was communicating directly with the different platforms. However, it is realized that having a middleware gives the flexibility to run each simulation independently with respect to each platform at the same time.

C# program is solely responsible for initiating VISSIM and extracting data from it. C# program is written in such a way that it can initiate a particular VISSIM simulation which has already been set up according to the specifications needed for the research, and then use that simulation to run for specific time and collect data. This program is most concerned about three specific processes; VISSIM single step run, VISSIM extraction and sending and receiving data from different components of the complete architecture. The initial time step was kept as 100 milliseconds. However, from initial tests, this small time step of 100 milliseconds could not be maintained as sometimes the time taken for the three processes to run one single step would take more than 100 milliseconds. So, 200 milliseconds was the new time step for s single step run of the simulation.

The program is further modified to check the time for execution of each process and if the total time is under 200 milliseconds, the program will ask the processes to wait for the remaining time till 200 milliseconds has elapsed. Thus it becomes necessary to analyze the time taken for each individual process to run, and figure out which process is taking the longest so that a more efficient code can be implemented to reduce this time consumption.

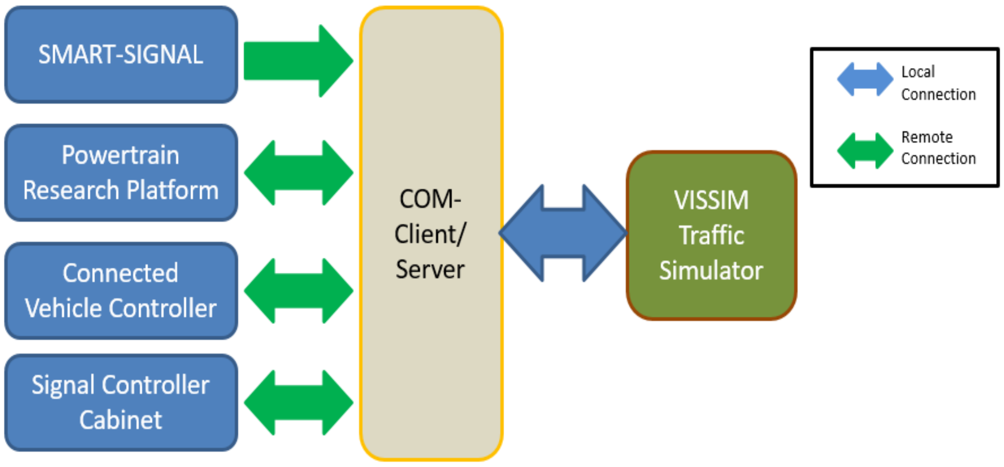


Figure 30.Overall Proposed Middle ware Integration Structure

3.1.2 Basic COM Communication with SIMULINK

The previous section discussed about the COM interaction with VISSIM. In this section the discussion pertains to the communication of the COM, C# program, with SIMULINK program environment. Unlike the previous case, where VISSIM needs to be initiated by C#, SIMULINK is already running and it needs access by C# to pass on the data. To directly access SIMULINK, C# needs to access the workspace of MATLAB and then update the workspace. The challenge is to update the workspace of MATLAB while SIMULINK is running. So, it is necessary that when MATLAB /SIMULINK environment is initiated manually by the user, the simulation reads some default values of the desired data based on the initial conditions. Once C# program is executed, it will initiate VISSIM which then will pass on the required information to the workspace of MATLAB through C# and update it for the run time. It is very important to note that for C# to communicate with MATLAB environment, it is necessary to make MATLAB an automation server. This allows the COM, C#, to access the server MATLAB.

Figure 28 clearly gives an idea about the functioning of C#, MATLAB and VISSIM. C# works independent of the MATLAB environment as well as the VISSIM simulation environment. Figure 28 shows the data flow only one way; from VISSIM traffic simulation to the hardware. This is just the first half of the simulation which has been completed and tested. Further discussion of possibly implementing two way communication is done later.

3.1.3 Network Programming- TCP/IP & UDP

Network programming or socket programming [28] is crucial to the development of the middleware architecture. Primarily network programming is a method used to send data across a network, specifically the internet. It is network programming that enables communication between the remote computer running VISSIM and the powertrain-research-platform which is represented by dashed arrows in figure 28. There are several applications that use network programming to send data across a network and the most common ones are any online chatting software that allow person to person chatting, video conferencing or voice conferencing. Implementing network programming with the HiLS increases its usability and accessibility of the powertrain- research-platform. With the availability of network communication, the powertrain- research-platform can be used by anyone with a simple internet connection. Therefore anyone can test their application with the powertrain- research-platform. Further in this paper, network programming will be crucial in enabling the evaluation of CACC application .

There are two approaches to network programming: TCP (Transmission Control Protocol) and UDP (User Datagram Protocol). Most network programming over the internet is done using TCP as its features ensure that data is sent over the network from server to client unlike UDP, where data transfer is not assured. However, UDP is much faster at transporting data over a network in comparison to TCP. The table below summarizes the different features of TCP and UDP.

Table 2.TCP and UDP features

TCP	**UDP**
• Connection based using port	• No concept of connection
• Guaranteed reliable and ordered data transfer	• No guarantee of reliability or ordering of packets, they may arrive out of order, be duplicated, or not arrive at all.
• Automatically breaks up data into packets	• Data has to be broken into packets
• Makes sure it doesn't send data too fast for the internet connection to handle (flow control)	• User has to make sure it doesn't send data too fast
• Easy to use: read and write data like it is a file	• If a packet is lost, user needs to devise some way to detect this, and resend that data if necessary

TCP uses features like flow control, data sequencing, retransmission to make sure that data is surely transferred unlike UDP which transfers data over a network without a connection. In other words, UDP does not care if the data packet is surely sent or dropped during transfer. Data packets are not sequenced hence data duplication occurs very often. However, UDP is faster because it does not implement features such as flow control, data sequencing and retransmission and directly sends the data packet to the client.

UDP is usually used in applications where fast data transfer is given higher priority than data transfer reliability. Most multimedia data transfer like streaming a video through a network uses UDP; as a data lag such as in case of TCP is not desired. However, TCP is used in most cases where reliable data transfer over long distances is higher priority than fast data transfer such as in the case of emails.

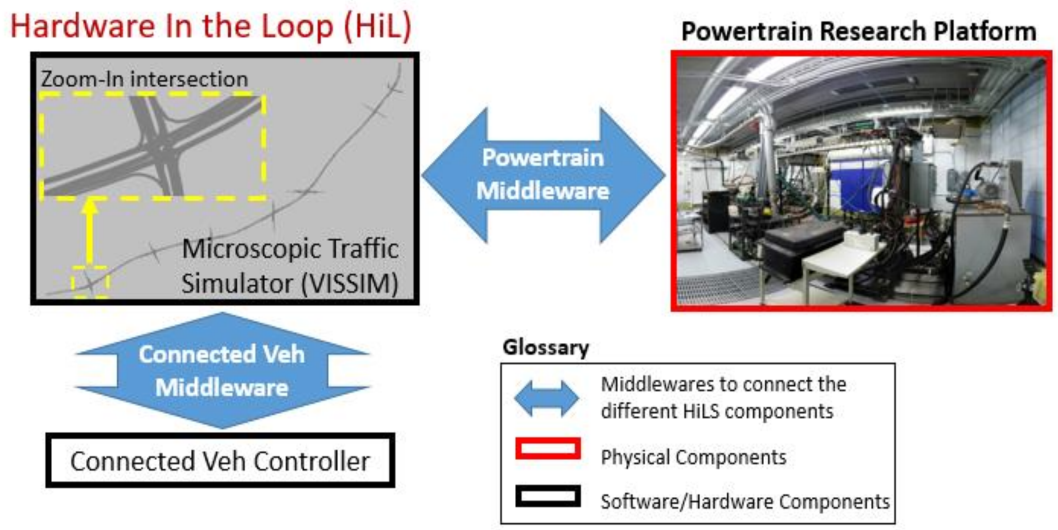
For the middleware, TCP is used as the network communication transport protocol because of the reliability of data transfer and ordered data delivery. In TCP, buffer memories are allocated on VISSIM-COM side for data sending and on Powertrain-COM side for data retrieval. These buffers ensure that data is not lost during the transfer. TCP also ensures the order of data is preserved on the receiving side, which is important in the HiLS application to distinguish the traffic data contents. UDP transport protocol is faster than TCP, but is not used due to unreliable data transfer and disordered data. Reliable and ordered data delivery is important because data loss will affect the accuracy of the tests. Although TCP is relatively slower than UDP, it is fast enough for the HiLS application.

The communication from VISSIM-COM to Powertrain-COM is one-directional, as shown in Figure 28. At start-up, Powertrain-COM is designed to continuously send requests for a connection with VISSIM-COM while the hardware(powertrain-research-platform) is running. In order to establish a connection, VISSIM-COM opens a port in the socket of the remote computer running VISSIM to accept connection request from the Powertrain-COM. The socket address is defined by the internet protocol (IP) address of the remote computer and the port number. Therefore, socket connection is established as soon as VISSIM-COM opens the port.

Utilizing TCP, VISSIM-COM and Powertrain-COM sends and retrieves data from their respective buffer memories. However, since network connection is established between the two buffers, the COMs will not be informed if interruption occurs in the internet network. It is therefore a common practice in TCP applications to include a keep-alive data to check the status of the internet connection between the buffers. Utilizing the keep-alive data, the Powertrain-COM will throttle down the engine if it detects a severe network interruption to ensure the engine is at a suitable operating point before shutting down for safety purposes and to avoid hardware damage. This safety feature is discussed in detail in a different section.

When VISSIM simulation is completed, the network disconnection is initiated by VISSIM-COM by closing the socket port and notifying the Powertrain-COM, where the engine will then throttle down in preparation for hardware shutdown.

3.2 COM Operation Development



Hardware In the Loop (HiL)
Zoom-In intersection
Microscopic Traffic Simulator (VISSIM)
Powertrain Middleware
Powertrain Research Platform
Connected Veh Middleware
Connected Veh Controller
Glossary
Middlewares to connect the different HiLS components
Physical Components
Software/Hardware Components

Figure 31. Components of Hardware in the Loop System (HiLS)

The middleware is designed to serve the purpose of linking different components of the HiLS, as shown in figure 31. It is based on COM a specific platform which enables inter-process communication and dynamic object creation in different programming languages[25]. The HiLS is made up of different components each based on different software platforms. The powertrain-research-platform and the Connected Vehicle Controller are in SIMULINK and the standalone Microscopic VISSIM Traffic Simulator, linked by the middleware , written in C#. For this thesis, the objective is to synchronize powertrain-research-platform and the connected vehicle controller in real time by developing a middleware that will handle data transfer with efficacy.

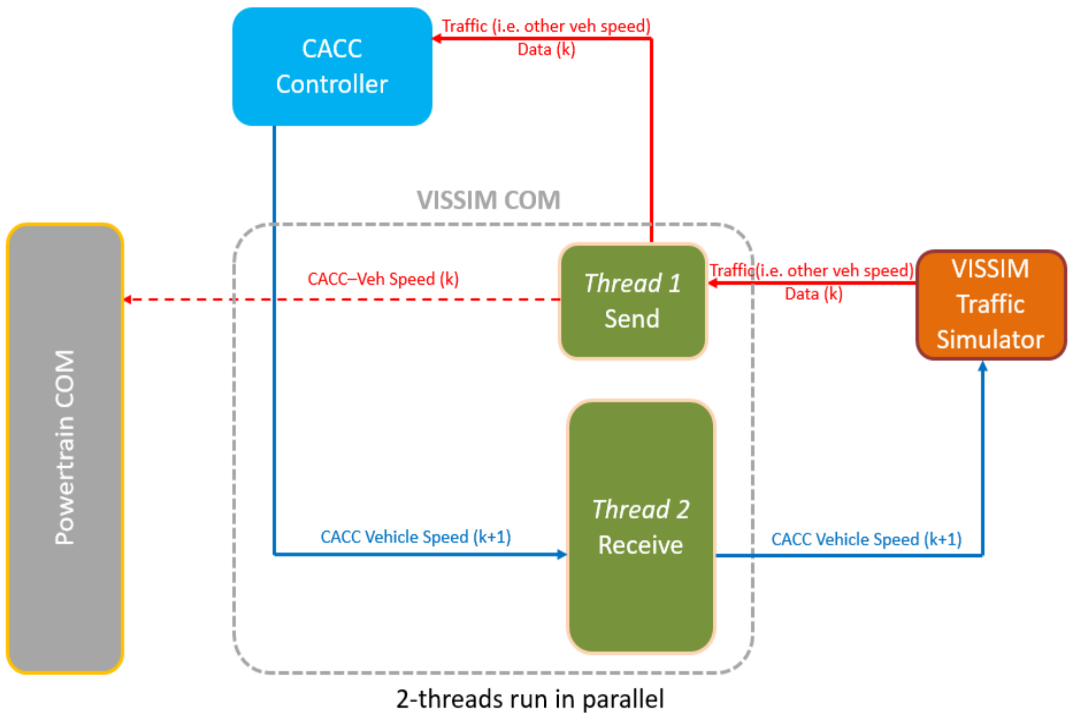


Figure 32. One way CACC Architecture

3.2.1 One Way Middleware Architecture

The middleware architecture for one way communication with CACC controller is shown in figure 32. The complete route for the data extracted from VISSIM traffic simulator is sent to a designated port in a network from thread one of VISSIM -COM and on the other side of the network, Powertrain-COM's thread one receives this data and sends it to Powertrain Simulink Model. Thread two of VISSIM -COM then receives this data and sends it to VISSIM Traffic Simulator before the next time step to use this data. The data here is specifically vehicle speed of the target vehicle. Extracting and sending any data to VISSIM Traffic Simulator requires to access the VISSIM simulation attributes. These attributes are predefined variables in VISSIM that may or may not be accessed from VISSIM by calling them during simulation. The attributes important for COM are related to vehicle dynamics and they are specifically vehicle speed, vehicle desired speed and acceleration. However, as mentioned earlier, the access to various attributes differ. So, in this case, VISSIM allows the user to access all three attributes of speed, desired speed and

acceleration during each time step of simulation. However only speed and desired speed are editable during simulation and acceleration is only a readable attribute. The different characteristics of VISSIM attributes are clearly defined in VISSIM.
Therefore, knowing that acceleration is a readable attribute and not an editable one, it is decided to use speed attribute of the controlled vehicle to update the speed of the vehicle for the next time step. Since the speed attribute needs to update the speed of the target vehicle for the next time step, it is crucial for COM to execute the task within the designated time step of 200ms.

This COM structure is the most basic structure. With the target vehicle's speed, other vehicles' speeds can also be extracted. This capability of multiple vehicle data extraction like speed, vehicle number, acceleration, position on link and link or lane number enables the possibility of implementing an application like CACC. CACC is an individual entity, that works independent of the COM but it communicates with the COM to access and send vehicle data for implementing the application in the best possible way. For CACC application, the extracted target vehicle data and other vehicle data is sent to the CACC controller. Providing all the information to CACC controller gives the user the capability to select and organize the data as per their own requirement. The COM is designed to send the complete set of data as one long array to CACC in a format of speed, position, lane and vehicle number. The method of selecting and sorting the useful data from this long array is completely left to the CACC controller. Once the CACC controller has sorted the data and processed the sorted data. It sends it back to COM for updating this processed data back in VISSIM for the preselected controlled vehicle.

The processed data from CACC controller is sent back to VISSIM-COM thread two which is responsible for sending data back to VISSIM before the next time step has elapsed. The synchronization between thread one and thread two of VISSIM-COM is very crucial. It must be made sure, that thread two only extracts data from CACC controller after thread one has sent a new data at every time step. Thread one is made to run at a designated time step of 200ms to make sure that the whole architecture maintains this specific time step throughout the simulation. However, it is not necessary for thread two to maintain the time step and that is why thread two is made to run as fast as it can. Therefore thread two extracts data much faster than the time step of 200ms, but only sends the data back to VISSIM when thread one tells it to. The communication between thread one and two must ensure when to send the updated data and this is done using a flag number that is sent from thread one. Every time a time-step is elapsed, this flag number increments like an index number. In thread two, there is a check done to make sure the flag number changes, and once it recognizes that the flag number has changed, it sends the extracted data from CACC controller instantly to VISSIM within the same time step. The timing results later will prove that the data extracted from VISSIM in thread one is sent back to VISSIM through the CACC controller before the 200ms time step.

The communication across the network to Powertrain-COM is one way as shown in figure 32. The current requirement is to send the CACC obtained velocity of the vehicle to the Powertrain-COM and then to the powertrain-research-platform to emulate the virtual vehicle that is running in VISSIM controlled by CACC controller. The engine in the powertrain-research-platform will be used to obtain actual measurements of fuel and emissions to compare it with a non-CACC vehicle running in VISSIM traffic simulation under the influence of VISSIM's internal driver model.

The structure of both the COMs can be divided in to sending and receiving data. For the first stage of the software development a sequential code was written where the sending and receiving of data would take place consecutively. However, it was discovered from timing results that such a method could not maintain the specific time step desired for the real time simulation. Therefore, the thread approach was taken where it allowed the two crucial processes to be divided into two threads. In figure 32, the structure emphasizes on the data flow through VISSIM-COM and Powertrain-COM. Thread one plays the role of sending the extracted data from VISSIM traffic simulator and at the same time Powertrain-COM serves to receive the data sent by VISSIM-COM over a network at a specific port . Once the extracted data from VISSIM Traffic simulator is received by Powertrain-COM, Powertrain-COM then sends it to Powertrain Simulink Model and then goes into the listening phase. Listening phase is unique to Powertrain-COM and thread two of VISSIM-COM. During the listening phase, the Powertrain-COM will check for any new data at the ports over the network. Any data sent to these ports will immediately be received for processing. The receiving of the extracted data at the Powertrain-COM is the mark of the end of one way communication over a network. In alignment with the current objective of one way communication, Powertrain-COM need not send the data back to VISSIM-COM over the network . Hence Powertrain-COM is independent of any threads and is running on sequential code which is capable of handling a time step of 200ms. Therefore, the data sent from thread one of VISSIM-COM at every time step is received by Powertrain-COM and sent to the powertrain-research-platform to complete the successful functioning of the structure shown in figure 32.

3.2.2 Basic Threads Structure

Before it was decided that for current development only one way communication across the network was sufficient, the focus was to develop a middleware structure that could handle two way communication across a network with high efficacy. This structure for both two and one way are shown in figure 33.It is deemed important to send the processed data back across the reverse path, from Powertrain-COM to VISSIM-COM and to VISSIM traffic Simulator . The idea is to update the next time step in the traffic simulator with the actual tracked velocity of the vehicle obtained from powertrain-research-platform. All these tasks need to be completed in a single time step of 200ms. Any minor delays can cause the whole program to be aborted from running as it will not be safe for the engine to run. Detailed discussion of safety features is done later.

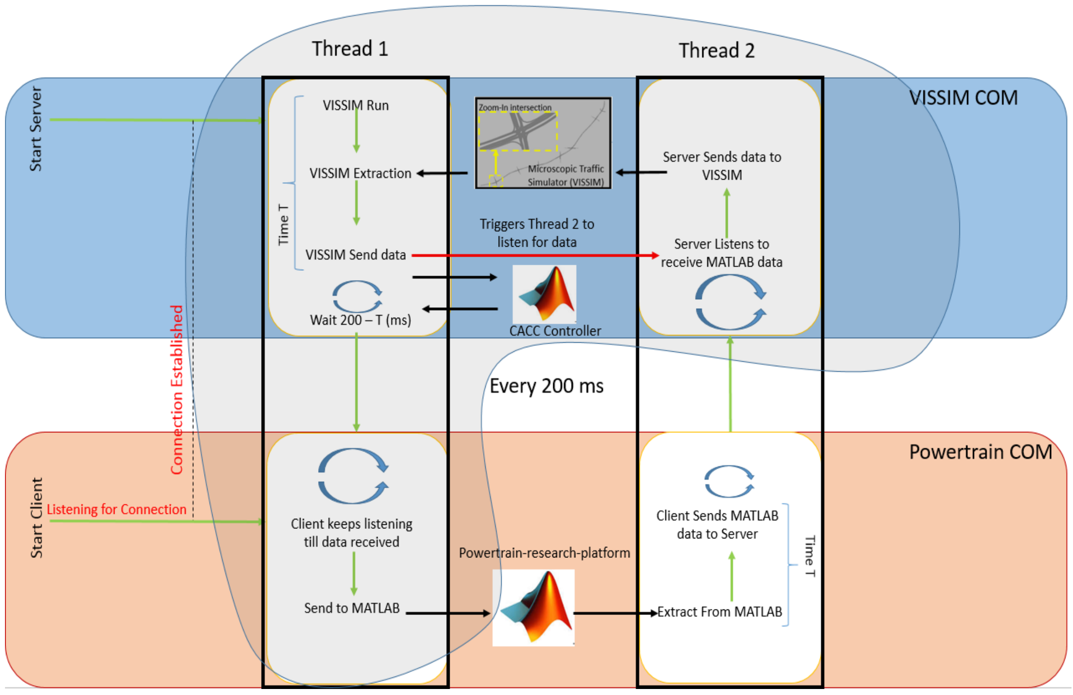


Figure 33. COM One and Two Way Thread structure – One way is shown under shaded area

The first stage of development for this middleware was to successfully conduct one way communication. One way communication is the complete data flow from the traffic simulator through the VISSIM-COM, across a network and to the Powertrain-COM and finally to the powertrain Simulink model for vehicle speed tracking purpose as shown in figure 33. However, to complete the loop, a back tracking of this processed data from the Powertrain Simulink model is also essential. Although two way communication was not implemented, it can be shown that the thread structure can handle two way communication as smoothly as it does one way communication. Keeping this intention of influencing the velocity of the target vehicle in the traffic simulator, the idea of a two way communication is developed. The challenge is to determine an efficient way to conduct this whole process of data transfer across a network in real time. One serious concern with real time application software is any sort of delays, especially in this case where the probability of a delay increases with so many different software platforms synchronized together. A literature survey was conducted to determine an efficient way to implement such a real time application middleware that will enable different software platforms to communicate with each other and successfully update the velocity status of the target vehicle in the traffic simulator after all the processing.

Sequential code implements various tasks one after the other and does not allow different tasks to run in parallel. It was immediately discovered that there were significant delays in the data processing and synchronization was difficult because if one part of the code had a delay, this delay would be propagated to the other tasks. Also, the load of computation was much higher on the code. One such bottleneck was extracting multiple data from the traffic simulator for multiple vehicles existing in the simulation. It was noticed that major delays were occurring at around 50 vehicles' information extraction which clearly made the augmented two way communication middleware inefficient. The results for timing are discussed later in detail.

In [22], [29] it is discussed that for real time application software, one of the ways to increase the performance of a code is by separating the processing of information and realizing different components of the middleware as different processes in the system. To decentralize the different tasks of the middleware, the methodology implemented is using threads. Using threads, different objectives of a software can be run independently from one another and each thread is recognized as a separate process where the computer processing power is dedicated depending on the power required to execute the specific process. Using this idea of threads both the Powertrain-COM's and the VISSIM –COM's structure were reorganized. Multi-threads can run in parallel as well as independently of one another. The benefit of parallel running of threads is that it allows different tasks to be executed at the same time which is crucial for the middleware. Threads are prominently used for multiplayer games played over network. They are known to reduce lag or delay between players who are playing the same game over the internet and this application of thread is significant in the running of middleware.

The most crucial part of running this thread structure without any delay is to have the threads synchronized. Synchronization is assured at the starting of the threads. As it can be inferred from the above figure, not all the threads start at the same time. The first thread is initiated in VISSIM-COM and Powertrain-COM. As Thread one in VISSIM-COM is initiated it starts the general process of extracting information from VISSIM Traffic Simulator .Once it has extracted the data and sent it across the network to Powertrain-COM thread one within a time step of 200ms, it sends a trigger value to start thread two of VISSIM-COM. In the case of one way communication the thread structure will look like the second diagram in figure 33. Therefore, with Powertrain-COM receiving data from the network port will mark the end of one way communication. However, this trigger alerts thread two of VISSIM-COM that at any moment it should expect a data on the specified port if two way communication is concerned. Hence it should start listening as fast as it can, without any specific time step. The reason behind not using a time step for thread two of VISSIM-COM is that the data in VISSIM simulator must be updated before the next time step as mentioned earlier. Hence, it does not matter when the data is updated until and unless it is before the next time step is executed because the simulator uses the latest data that has been updated. For instance, if thread two updates the same value for 100 times before the next time step, the simulator will use the 100th data that is updated. If in the next time step, the simulation tends to become slower and thread two updates the data only 50 times, then the simulator will use the 50th data that is updated before the next time step. At the same time Thread one of Powertrain-COM has been listening and also receiving the first data over the network. It then sends the data to MATLAB and simultaneously sends a trigger to its thread two to initiate the listening and sending process. So thread two of Powertrain-COM will try to extract processed data from Simulink Model and send it across the network to VISSIM-COM's thread two in the similar fashion VISSIM -COM's thread two is described to be functioning, without any time step. Since VISSIM -COM's thread two has already been listening, it is ready to receive any data from the port and send it back to VISSIM Traffic Simulator to update the value for the next time step. All these tasks need to be completed in 200ms for the next time step data to be updated.

In figure 33, the complete structure of the threads for the respective COMs clearly shows that the crucial tasks are running in parallel unlike previously when the tasks were in sequential order. The benefits achieved from this structure are that the tasks are recognized as two separate processes in the memory of the system which allows the system to dedicate processing power separately to the different threads making it more efficient than the sequential way of implementing the tasks in the code. However there is still a possibility that with the thread structure, maintaining synchronization of data transfer can be a challenge as delays may not just occur due to the specific task assigned in the thread, but also due to other tasks that the system is simultaneously running like all other applications of the operating system. A deep dive into the details of how delays can be reduced is done with an elaborate timing analysis on each section of the code written to identify any existing bottlenecks and possibilities that can lead to a delay.

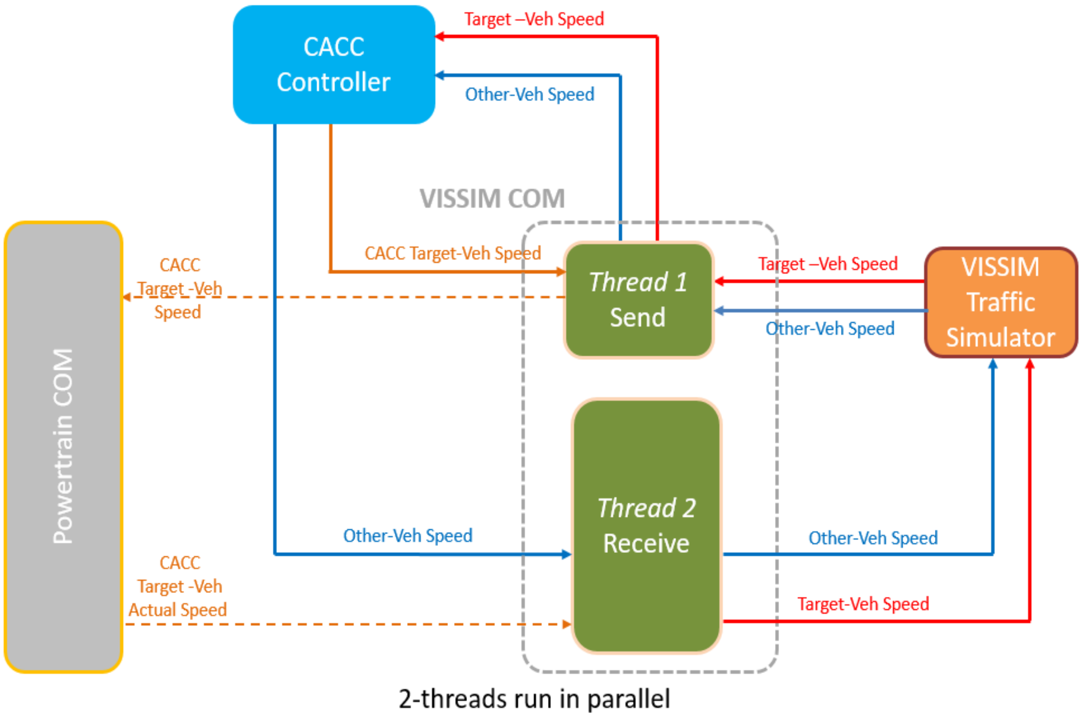


Figure 34. Implementation of CACC application with 2 way communication

3.2.3 Two Way Middleware Architecture

One of the benefits of having two way communication is that various applications can be tested and one of them is the Connected Vehicle Controller application which in this case is the CACC application. A much more sophisticated structure of the two COMs in synchronization with the CACC component is shown in Figure 34. The architecture shown in figure 34 conveys the complete loop structure, however figure 32 depicts the current functional structure that has been tested. All the different components, VISSIM Traffic Simulator, VISSIM-COM, CACC controller and the Powertrain-COM and Powertrain Model are linked in a synchronized manner. The CACC controller is a separate model created in SIMULINK platform and it is linked to the VISSIM -COM in the similar manner the Powertrain Simulink Model is linked to the Powertrain-COM. In this figure more emphasis is given to the VISSIM -COM structure with the CACC controller and VISSIM as it requires more sophistication in the code written, in terms of maneuvering multiple data extracted from VISSIM.

VISSIM Microscopic Traffic Simulator is a piece of software created by PTV group in Germany. The word microscopic is a key feature that the traffic simulator possesses and that is it allows the user to zoom into the simulation details of an individual driver or vehicle. VISSIM uses different driver models to realistically simulate a traffic situation. These models are based on years of traffic research and statistical data. The model particularly used in the simulations is the Weidman model. More technical details can be found in [30]. The important feature that makes the use of VISSIM in this architecture significant is that it allows the user to use an internal driver model that is predefined by the software developers of this simulator as well as allows the user to implement their own driver model by writing their driving model in the form of a Dll (Dynamic Link Library)structure . This external driver model completely overtakes the internal driver model of VISSIM to give the user the freedom to change the vehicles' behaviors in the simulation based on their requirement.

Based on this feature of the traffic simulator, it is decided that processed vehicle speed from the CACC controller will be sent back as feedback to the simulator to influence the next time step velocity of the vehicles in the simulation during run time. Among the two approaches, COM and DLL, the COM approach is chosen as it gives more freedom to link other components like the CACC controller explicitly with the COM whereas the DLL acts like a library to each piece of software and it is difficult to connect the different components to the DLL and make them work in synchronization without using communication through a network. In other words, the DLL is a library that can be simultaneously used with any piece of software. However, when two pieces of different software are linked to a DLL, the DLL creates a copy of itself to provide its service to both the software at the same time, but it does not recognize that it is connected to both the software. The DLL works completely independent on both the software as if they are not connected to the DLL together. Hence, the problem arises when the requirement is to pass information from one software to another through the DLL. So updating a variable in the DLL from one software and passing it to the other software for further processing is not possible without a network. However there is a workaround to this problem and that is to connect the DLL to a TCP or UDP [28] port and then pass the data through a network, but this makes the task more tedious for the system and may increase the time for data processing depending on the type of network in use and the traffic of information in the network. Thus to keep this local on the computer and the structure simple, it is decided to go with a COM. However, the tradeoff is with speed of processing, as DLL works like an internal component of any software it is linked to and carries out tasks efficiently in comparison to COM which has to be executed explicitly. The COM however allows the code to be much more sophisticated in terms of functionality and has much more flexibility in terms of linking to other software paradigm .Thus, for this architecture shown in figure 34 or in figure 32, the COM is chosen to be the mediator of information between different software pieces.

3.2.4 Thread one and two VISSIM-COM

The structure of both the Powertrain-COM and VISSIM -COM consist of two threads for two way communication and no threads in the Powertrain-COM for the one way communication. One thread is responsible for sending information to the network and the other thread is responsible for receiving information from the network. Their synchronized functioning is enabled by the capability of threads to work in parallel. As shown in figure 33, thread one in VISSIM-COM is responsible for initiating VISSIM Traffic Simulator, and running the simulator for every single time step as specified or required. In this case, all the simulations are run with a time step of 200 milliseconds. A for loop is implemented which is controlled to run for 200 ms using a pause methodology available in C# language. The for loop, if finishes an iteration before 200ms, is made to pause for the 200 minus the time elapsed to keep the time step consistent at every simulation step. Similar methodology is used for thread two of the VISSIM -COM but without a specific time step as explained earlier. Within that loop structure of thread one, exists the feature of extracting multiple data information from VISSIM. Data types that are relevant for the purpose of this middleware are vehicle speed, vehicle number, link number and position of vehicle. The code is written to extract all information for all vehicles existent in simulation at every single time step in the form of a single one dimension array. Once all this information from VISSIM traffic simulator have been extracted, the arrays are sorted for only selected data that need to be passed on to the other components of the HILs, in this case the CACC controller. The sorted data is arranged in another one dimensional single array and sent across the network. The idea of using an array makes data sending and receiving much easier as the selected data is in a compact form. Previously, a for loop was being used to sort the arrays or fill the elements of the arrays and it was discovered that using a for loop was causing the runtime to exceed the designated time step of 200 ms and leading to major delay in loop. Thus a method where all elements from an array could be copied and pasted to a new array was used to make the sorting of data into compact arrays much faster. Thread two on the other hand is responsible for listening to the network after it receives the trigger signal from thread one. Thread two is designed to continuously check for new data at the designated port in the network. If any data is received it will send the data to VISSIM, otherwise the previous data will be sent in case there is any data loss due to network disconnection. However, for one way communication thread two of VISSIM-COM will not listen at a network port, but it will access the data locally from the CACC controller. One important fact to note on the VISSIM -COM side is that any sort of internet disconnection or network loss does not need to be dealt with explicitly because the simplest thing to do in such cases is to just end the simulation at the moment network disconnection occurs. However, the matter is dealt in a different way on the Powertrain-COM side as discussed further.

3.2.5 Thread one and two Powertrain-COM

Thread one of Powertrain-COM plays the exact same role of listening like thread two of VISSIM-COM but with added features. Since the Powertrain-COM is directly dealing with Powertrain Simulink Model which is directly connected to a real engine, there are some safety precautions taken to make sure during times of disconnection or network loss, the engine does not run uncontrolled and a way to shut the engine down safely is assured. The engine currently in use has an idling speed of 900 RPM. Thus the safest way to shut the engine is to bring the engine to its idling speed of 900RPM and very low engine torque. It is not completely necessary to shut the engine at 900RPM but definitely safe. However, the engine torque must be maintained very low for shut down procedures. Thus, keeping these parameters in consideration a safety check methodology is implemented using a check value. It is observed that every time a disconnection occurs, the data received from the network in the thread one of Powertrain-COM will read the previous data from the previous time step as no data has been updated.

The first approach is to check for the repeated value over one second which means that with a time step of 200ms, if the value repeats for over a time of one second, the code will be instructed to reduce the speed of the vehicle from the last updated speed to zero. However, it is realized that this cannot be achieved with just the vehicle speed information because it can be possible that the target vehicle is at a stop and it is sending zero velocity to the network repeatedly, and this zero velocity with the current safety method can cause the code to abort even though there is no actual disconnection. Thus a check-alive data is sent in an array with the starting value of one. This assures that the check-alive data can never be the single digit zero and it will send incremented data, until and unless there is a network disconnection of some sort. Thus, within one second of elapsed simulation time, if the check-alive data is repeated, the code will assume that there has been a network disconnection and proceed to use the last updated vehicle velocity to reduce the speed to zero and send it to the powertrain telling the engine to come down to its idling speed of about 900 RPM. This way it becomes safe for the engine to shut down.

Such a safety feature is not repeated in thread two of the Powertrain-COM, as the thread two is only responsible for sending the processed data extracted from Powertrain Simulink model and send it back to VISSIM -COM across the network. The threads structure of the middleware make two way communication possible. Currently, two way communication is achieved but not completely executed as the focus is to implement CACC application with one way network communication. Using threads makes it possible for future development of two way communication with the CACC application integrated . However for now, since the engine can track the speed sent from VISSIM traffic simulator very well, it is not desired to implement two way communication to update the actual vehicle speed from powertrain back to VISSIM for the next time step.

Two way communication is a crucial feature of this structure because it enables feedback of data completing the Hardware in the Loop. For instance, for the CACC controller, the Microscopic Traffic Simulator and VISSIM-COM, it is very important that the vehicle information extracted from the traffic simulator is efficiently passed on through the controller and back to the simulator within one time step. In this case, logic of the data flow fulfills the objective of emulating an individual vehicle using a real engine and a virtual powertrain. The information of an individual vehicle is extracted from a microscopic simulator, and sent across a network to the powertrain research platform for running a real engine with a virtual powertrain. Since this virtual traffic simulator is a microscopic simulator, it is possible to extract the information of one individual vehicle. Information obtained from the simulator is specific to vehicle dynamics like the speed, acceleration and location of the vehicle and other attributes may be extracted if desired. This information obtained from the simulator is then accessed by the COM which is identified as VISSIM-COM. It is responsible for routing selected data to the different components associated with this structure.

Figure 31 shows the different applications of HiLS and one of them is the Connected Vehicle Controller. The Connected Vehicle Controller compliments various applications and one of them is the CACC. CACC application is implemented with the HiLS to investigate the effect of the controller for one vehicle now and multiple vehicles in the future. VISSIM-COM must manage the routing of the data to CACC controller which is running on MATLAB software platform and the Powertrain-COM which is another middleware on a different computer, connected using a network (i.e. internet).

The Powertrain-COM's responsibilities are analogous to that of VISSIM-COM. VISSIM-COM is the middleware that creates routes for data from VISSIM Traffic Simulator to the other components of HiLS whereas Powertrain-COM provides data to the powertrain-research-platform. The data received on the Powertrain-COM, is routed to Powertrain Simulink model for processing. The virtual powertrain, in Simulink , calculates the dynamometer torque which is then sent to the dynamometer associated with the lower level controller for tracking . The tracked engine speed and torque can be received back in the model to recalculate the actual vehicle speed. Based on the tracking performance of the controller ,the vehicle speed can be sent to the Powertrain-COM which can send it back to the VISSIM-COM. Once VISSIM-COM acknowledges the receiving of the data, it sends it to the traffic simulator intended to influence the next time step velocity of the target vehicle. The traffic simulator assigns the speed of the target vehicle as the actual speed received from the Powertrain model before the next time step is updated. This single loop of data flow displays the emulation of one vehicle in the traffic simulator. This is one very significant application of two way communication.

3.2.6 General Car Following Model

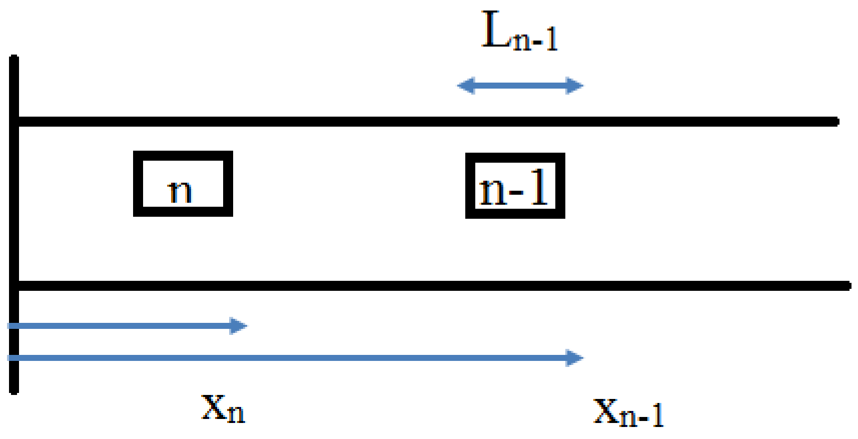


Figure 35. Driver Following Model

Microscopic VISSIM Traffic Simulator emulates different traffic scenarios based on various parameters. One of the key features of VISSIM Traffic Simulator is that an internal driver model is used by the software to emulate driving behavior of individual vehicles. Like most other traffic simulation software, VISSIM too uses a driver following model. The driver following model pertains to the dependence of the characteristic of the target vehicle's driver behavior on the behavior of the preceding vehicle. The main objective of a car-following model is to maintain a safe distance between consecutive vehicles in a traffic situation so that the following vehicle does not hit the preceding vehicle and cause an accident. Scientists have developed several car-following models. However , no model till date can be claimed to perfectly emulate the human behavior. It is extremely difficult to emulate the behavior of a human driver because the behavior of each individual human is not only unique from other humans but also unique to a specific situation. Researchers claim that there are two basic groups of parameters which influence the time of drivers' reactions. The individual characteristics are age, sex, driving skills, tiredness, stress, alcohol, drugs, psychological pressure and the characteristic of vehicle, and other external factors like time of day, road conditions and visibility . Taking all these variable conditions into consideration makes it very difficult to design an accurate driving model

.This makes it very difficult to derive models that emulate human driving behavior. For the application of VISSIM software, it is important to have a car following model as each vehicle in the simulation must uniquely behave because of the software's feature of a microscopic simulation package. Most existing car-following models are designed with the objective of collision evasion and one such model that is closely used in VISSIM traffic Simulator is the Wiedemann Car following Model[30].

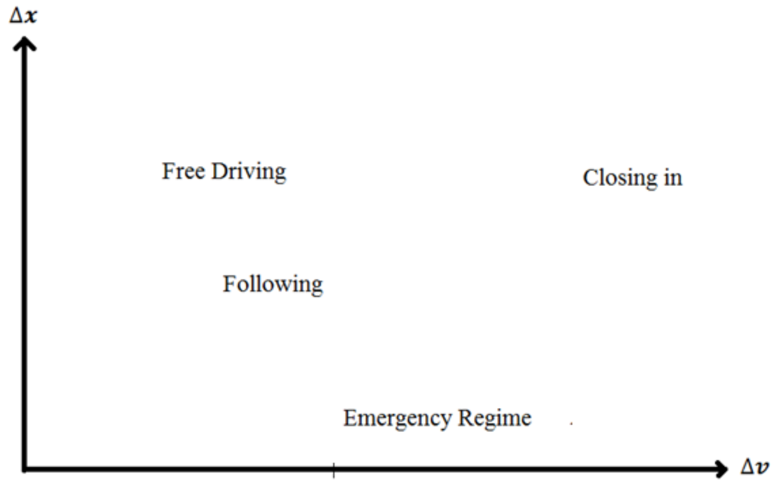


Figure 36. The regimes in the Wiedemann car following model

3.2.6.1 Wiedemann Car Following Model

The Wiedemann psycho-physical car-following model as shown in figure 36 above is responsible for modelling the longitudinal dynamics of the vehicles in the simulation. The drivers' behavior is described with four distinct regimes as a function of $\Delta x\ and\ \Delta v$. The regimes are as follows: free driving, approach to a car queue, driving in a queue and braking. All these regimes are decided based on the difference of absolute position between the preceding vehicle and the following vehicle given by Δx and the difference in the velocities of the preceding and following vehicles given by Δv . Based on these parameters the decision for the velocity of the following vehicle, for the next time is taken. The decision is to either accelerate, decelerate or keep the same velocity. The model used in VISSIM derives the desired distance between the following vehicle and the preceding vehicle based on the relation shown in [30] which further denominates regions as SDX,ABX,SDV,CLDV and OPDV.

SDX gives the maximum following distance for the following vehicle with respect to the leading or the preceding vehicle. This distance is about 1.5 to 2.5 times the minimum following distance ABX. SDV is the regime where the following vehicle is approaching the preceding vehicle. The region CLDV is SDV in VISSIM Traffic simulator. These conditions described above give rise to the car-following regimes: Following, Free driving, and Closing in as shown in figure 36 . Following occurs when the following vehicle approaches another vehicle in front of it. Free driving occurs when all the other vehicles, although they exist in the traffic network, are not in the vicinity of the following vehicle. Closing in occurs when the following vehicle approaches another vehicle in front of it with a lower velocity than its own.

Following:

The regimes SDV,SDX,OPDV and ABX, fall under the following regime. In this region the target vehicle is clearly influenced by the preceding vehicle which it is following. As the target, or the following vehicle crosses the SDV or the ABX regime, the following vehicle has a negative acceleration or deceleration. However, in the OPDV and SDX region, the acceleration is positive.

Free Driving:

In this regime the following vehicle is technically not following any other vehicle as it is not influenced by the other vehicles in the traffic network. This is the regime where it can be claimed that the driver model is not acting anymore. The following or the controlled vehicle now tries to achieve its desired velocity. The vehicle uses its maximum acceleration to reach its desired velocity.The maximum acceleration is a function of the v_{max} of the target vehicle and not a function of the preceding vehicles' acceleration.

Closing In:

This region occurs when the driver is passing the SDV regime. The target vehicle, more appropriately known as the following vehicle in this case, is closing in to a leading vehicle that is travelling at a slower velocity than the following vehicle. It is very obvious, that to prevent any sort of collision , the following vehicle has to decelerate. From the deceleration function in [30] it can be clearly seen that the deceleration in the closing in regime has no relation to desired velocity or maximum velocity of the following vehicle unlike in the case of free driving. In fact, the deceleration is clearly influenced by the preceding vehicle's driver's behavior as there is a clear relation of the deceleration function to the preceding vehicle's acceleration.

3.2.6.2 Overthrowing the VISSIM Internal Driver Model

The above background study on the internal driver model of VISSIM gives an insight into when the driver's behavior is directly influenced by the preceding vehicle's driver behavior and when it is not. This piece of information is very important to implement autonomous vehicle applications like CACC. Essentially there are two ways to implement autonomous vehicle applications. One is to completely take control of the vehicle which will require overthrowing the VISSIM internal driver model discussed above for individual vehicles or a platoon of vehicles. The other option is to implement a semi-autonomous application, like a driver advisory. Since in the future it is highly desired to implement completely autonomous application, the approach is taken to remove the internal driver model of VISSIM. The microscopic feature of VISSIM Traffic Simulator is especially helpful in this case as it allows to select individual vehicles and remove the functionality of the internal driver model.

In [31] , to implement a cruise control approach to an individual vehicle , it is realized that the vehicle can be completely controlled by an external cruise controller when the individual vehicle is in the free driving regime. From figure 36 and previous discussion it is clear that in the free driving region the vehicle is not influenced by other vehicles in its vicinity . Hence, this is the only regime where an external controller, as mentioned in [31], can take full control of the vehicle. However, [31] acknowledges that when the vehicle passes other regions like SDV or SDX, then the VISSIM internal driver model takes control of the vehicle overthrowing the external controller. Hence, this scenario is a typical depiction of a driver advisory. However, to fully realize the benefit of an external controller like CACC with respect to fuel consumption, reduced emissions and reduced traffic congestion, it will be beneficial to provide full control of the vehicle to the external controller under any circumstances because unlike a driving advisory where the driver's reaction will play a critical role in determining the vehicle's velocity profile, the controller can implement its optimal control output to the vehicle. This is given that the controller is robust in several driving situations and does not become unstable and cause dangerous maneuvering of the vehicle. With this intention of giving full control to the external controller, it is desirable to overthrow the VISSIM internal model of the driver. To achieve this , it is necessary to realize the method used to determine the acceleration and velocity of each individual vehicle in the simulation. Although, VISSIM's official manual claims that the driver model is a depiction of the Wiedemann Driving model, there are some hidden differences in VISSIM's model which are not released to the public due to commercial reasons of protecting intellectual property. Hence, it is difficult to exactly determine the internal driver model of VISSIM. However, since it is known that the driver model in VISSIM is close to Wiedemann driving model , an approximate idea of the model can be made.

From figure 36 and [32], it is known that only in the free driving region VISSIM's driver model is not influenced by the preceding vehicle. From the equations in [30], it can be seen that except for the Free Driving regime, in every other region , the acceleration is a function of distance from the preceding vehicle as well as the speed or the acceleration of the preceding vehicle. The free driving region only occurs when $\Delta x\ and\ \Delta v$ are large in magnitude which means when the following vehicle is either too far away from the preceding vehicle or has a large velocity difference. Now, it is important to figure out a way to implement this free driving region at smaller magnitudes of $\Delta x\ and\ \Delta v.$ The approach taken is to study a "dummy" driver model DLL file provided by VISSIM. In the driver model DLL, there are two specific functions defined as "SetValue" and "GetValue". The role of these functions are to set the parameters as output from VISSIM and send the parameters updated after a single time step to VISSIM for the next time step respectively. It is realized that the GetValue function is solely responsible for providing VISSIM with the updated parameters of the vehicles like the desired velocity, desired acceleration, and desired lane. However, there is no specific parameter for setting the speed and acceleration of the vehicles. This means that VISSIM's internal model is dependent on the desired velocity and desired acceleration to determine the current speed and acceleration of the vehicle. From the equations it is also known that the acceleration in the car following models are a function of the preceding vehicle's velocity and the distance from the preceding vehicle, which means that other than the free driving regime, the vehicle's acceleration will be largely influenced by the preceding vehicle's behavior. In fact , in the closing in regime, the deceleration has a direct term of deceleration of the preceding vehicle added to it. Thus, it can be said that disabling the desired velocity and desired acceleration parameters in the DLL can possibly get rid of the internal model for a specific vehicle.

The first approach is to disable sending desired speed parameter to VISSIM from the external driver model DLL. By disabling it simply means to put an "if" condition that if the vehicle identification number of a specific vehicle is recognized, don't send the parameter. Hence, for the first test vehicle number 16 was selected and the "if" condition was implemented to disable sending the desired velocity to the internal driver model. After running the simulation it was realized that disabling the desired speed showed up as zero desired velocity of the target vehicle, and the vehicle in the simulation directly approached that zero velocity and halted in the middle of the road. This tells that the internal driver model was still acting and in full control of the vehicle. The next approach was to enable the desired speed and disable the desired acceleration. After running the simulation, it was observed that the vehicle starts to accelerate without any consideration of the surrounding traffic. The vehicle even hits the other preceding vehicles and over takes them in the virtual simulation. This clearly means that disabling the desired acceleration parameters overthrows the driver model to some extent. However, the question is why does the vehicle accelerate throughout the simulation and reach a velocity of more than 100km/h by the end of the simulation which is higher than its designated desired velocity of 75km/h.

First, it is inferred that the acceleration is probably due to the vehicle trying to achieve the desired velocity which is enabled in this case, but the inference is proved wrong from the observation that the vehicle exceeds the desired velocity . This is where an approximation of the model is attempted. It is determined from the general acceleration function for most car following models that the acceleration is a function of the preceding vehicle's driving behavior. Hence, disabling the driving behavior automatically gets rid of the VISSIM driver model which was observed when the vehicle accelerated without considering the vehicles surrounding it and surpassing them in the virtual traffic network. However, the vehicle still accelerated, and it seemed like that the vehicle was accelerating at its maximum acceleration. Hence, it can also be inferred that the acceleration is a function of the maximum acceleration. From the background study above, it is explicitly stated in [30] that the vehicle achieves its maximum acceleration to achieve its desired velocity and that too only when the vehicle dynamics is in the free driving regime. Therefore, in this case it can be claimed that the vehicle is behaving as if it is in the free driving regime but it is not accelerating to achieve the desired velocity . This again proves that the driver model is disabled because to stop the vehicle from accelerating beyond its desired velocity , the driver model needs to provide feedback but since the driver model is disabled, it cannot provide any feedback.

Taking the above observation into consideration, for the next simulation run, both the desired velocity and desired acceleration terms were disabled. It was observed that disabling these terms the vehicle did not accelerate any more but instead traveled at its initial velocity assigned as it entered the traffic network. The vehicle speeded at this initial velocity for the whole simulation without paying any regards to the surrounding vehicle. This clearly tells that the vehicle was no more influenced by any desire to achieve any speed or acceleration and it was not influenced by any regimes. However, to confirm this inference, it is important to determine if the vehicle is still following a physical, dynamical model. A comparison is conducted to make sure that the VISSIM model, even though has a disabled internal driver model , follows the fundamental laws of physics. The figures below clearly show that the distance obtained by integrating the speed over the simulation time is the same as the distance obtained from VISSIM simulation which is directly extracted as a vehicle position attribute and stored in a text file for each time step. Since both the plots in figure 37 are the same, it can be claimed that even with the driving model disabled , the simulator follows the speed and distance relationship based on the fundamental laws of physics.

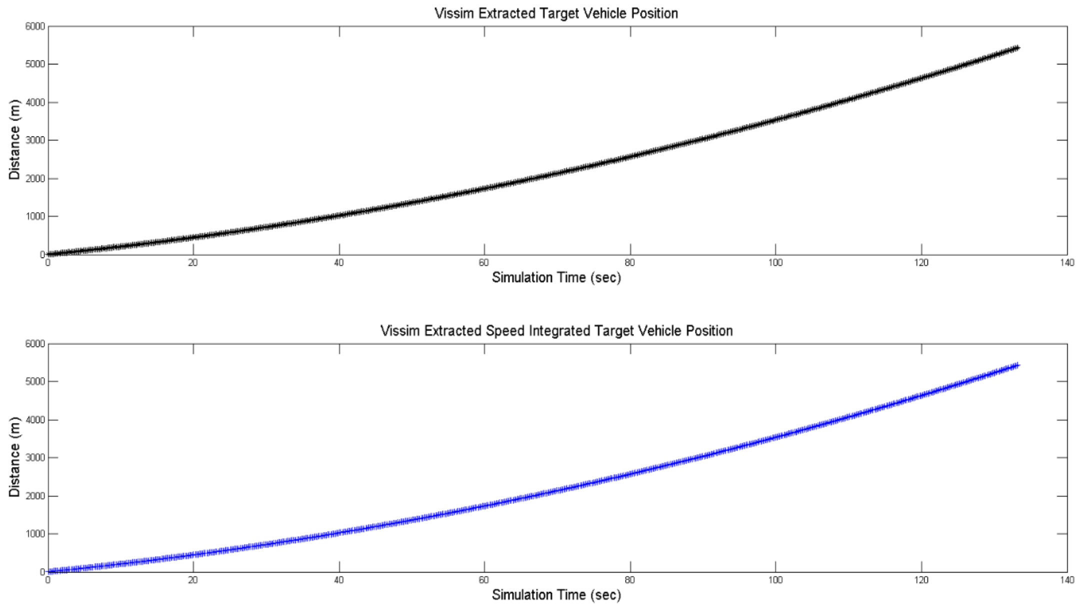


Figure 37. Comparison of Δx from Matlab and VISSIM simulation

3.3 Timing and Synchronization of CACC architecture with Simulation Results

This section will give an overview of the time execution analysis of the three vital processes that take place in the program. The three processes under discussion are: Time taken for VISSIM to run single step, time for C# to extract data from VISSIM for different number of vehicles extracted and the time taken for C# to receive and send data to the other components of HiLS.

As mentioned earlier, it is decided that the total time taken for these processes will be set to be below 200 milliseconds; that is the three processes will have to be completed within 200 milliseconds. With the assumption that the individual time for each process takes less than 50 milliseconds, the initial program is written such that if the total time taken for the three processes takes less the 150 milliseconds, the program will be paused for the 200 milliseconds minus the time elapsed to complete the three processes to maintain the desired time step.

In the following sections the results for the time for execution of each of the three processes are discussed.

3.3.1 Time for VISSIM to run single step

Figure 38 shows the time taken for VISSIM to run single step over a simulation time of 300 seconds and also combines the graph of the total vehicle count at the end of each simulation. The simulation is carried out for six different situations with regards to the total number of vehicles present. The green cluster of points represents the increasing presence of vehicles in the simulation. The different cases for which the simulation is run are: 600, 700, 800, 900, 1000 and 1100 vehicles.

The blue cluster in figure 38 represents the time taken for VISSIM to run single step over the total simulation period. There are about 3000 different points plotted which form the blue cluster shown in figure 38. It can be seen that as time passes by, the execution time for VISSIM has an increasing trend. For example, in the graph for 600 vehicles, it shows a very slight increase in time as the simulation time reaches the end of 300 seconds. The increasing trend is not very prominent in this case. If the case of 1100 vehicles is considered, then the increasing trend of time is much more prominent. This increasing trend in time is expected with the increasing density of vehicles. Comparing the plot for 600 vehicles with the plot of 1100 vehicles, the plot for 1100 vehicles has slightly higher initial execution time and the time increases as the total vehicle count increases.

3.3.2 Time for C# to transfer data to MATLAB

Figure 39 represents the relation of run time for MATLAB with respect to the total runtime of the simulation. It also shows the vehicle count for the total simulation time which is kept constant to investigate the effect of changing the number of vehicle information extracted. The six plots shown below correspond to the extraction of 1, 50,100,150,200 and 300 vehicles.

From figure 39 it can be seen that the points plotted in blue represent the run time for sending individual data to MATLAB. The simulation was run for different conditions as it was done for analyzing VISSIM execution time for running one single step shown in figure 38. The average time for sending data to MATLAB is approximately in the range of 0 to

15 milliseconds. Like VISSIM running single step, it does not take C# much time to send data to MATLAB. However, unlike VISSIM running single step, the transfer rate of data to MATLAB remains almost constant even as the number of vehicles extracted increase in the simulation.

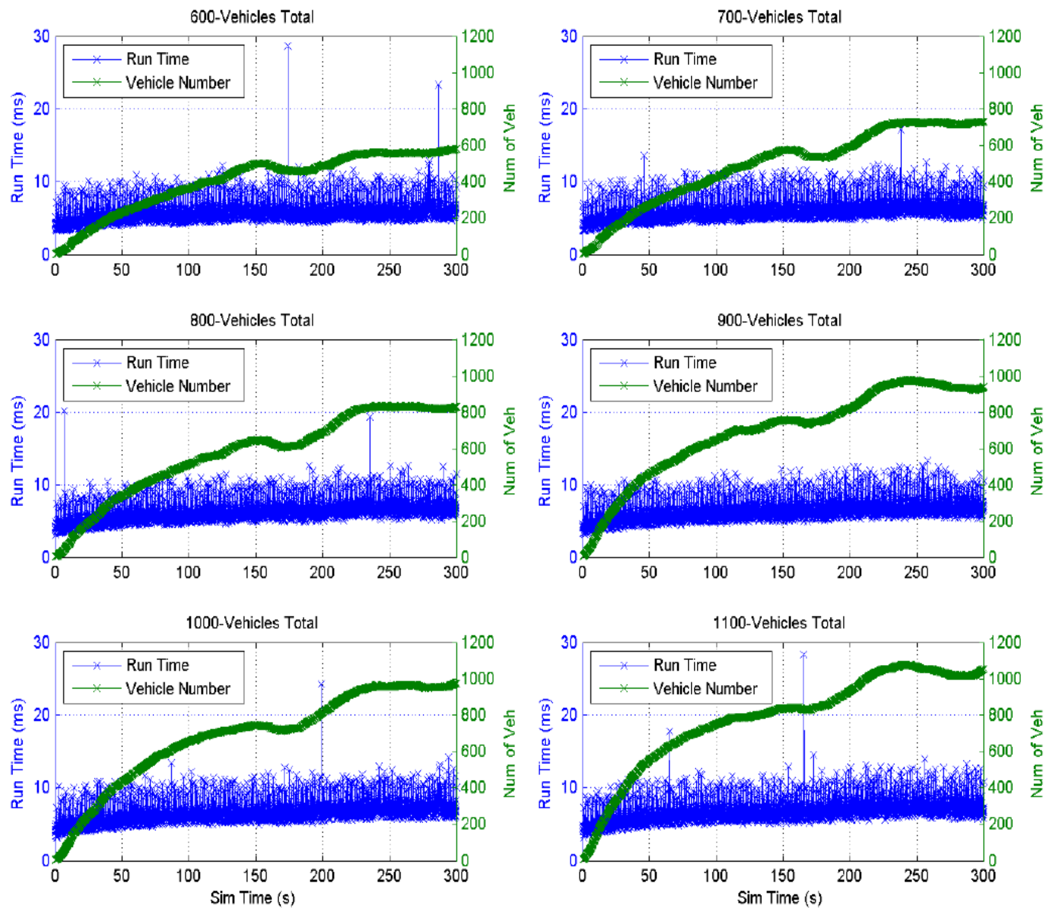


Figure 38. Time graph for VISSIM to run single step

This trend is expected because the C# program sends an array of vehicle information instead of sending individual information one by one. Implementing an array makes the process faster, because then C# sends all the data at once. MATLAB only has to receive the array which is of a particular size for every single run of the different simulation conditions. Since the array size does not change, the time for execution is almost same for all the graphs. If the array size changed for every single simulation, then the time plot would have an increasing trend. The insignificant difference in execution time for vehicle extraction is only because of the extra time taken to load the array as the number of extraction vehicle increases from one simulation to the next.

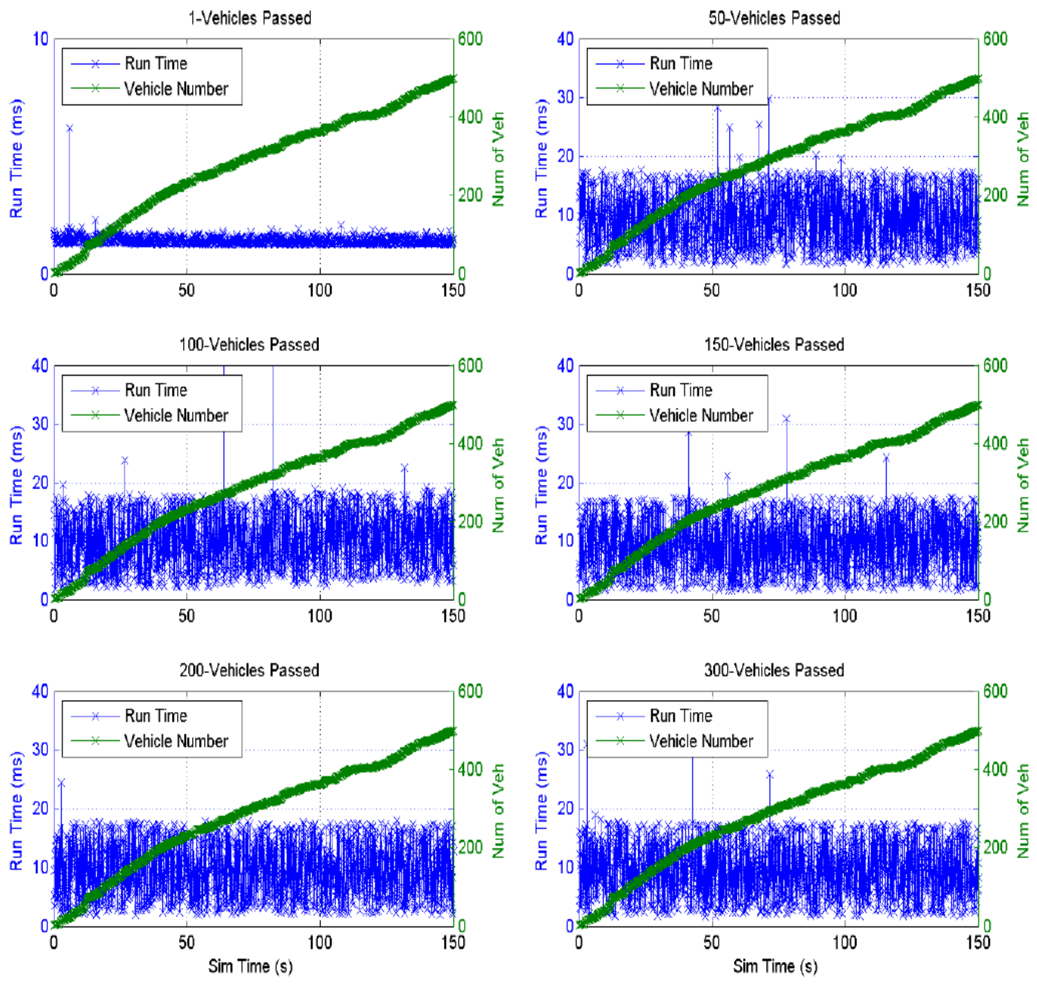


Figure 39. Time graph for sending data to MATLAB

3.3.3 Time for C# to extract data from VISSIM

The results for extracting vehicle data for 50 and 100 vehicles using the preliminary code are shown in figure 40. It shows the time for execution increases significantly as the number of vehicles increase in the simulation. In this case for extracting data for 50 vehicles, the time increase in the beginning is till the vehicle extraction is below 50 vehicles, but as it reaches 50, the time stays constant for the rest of the simulation.

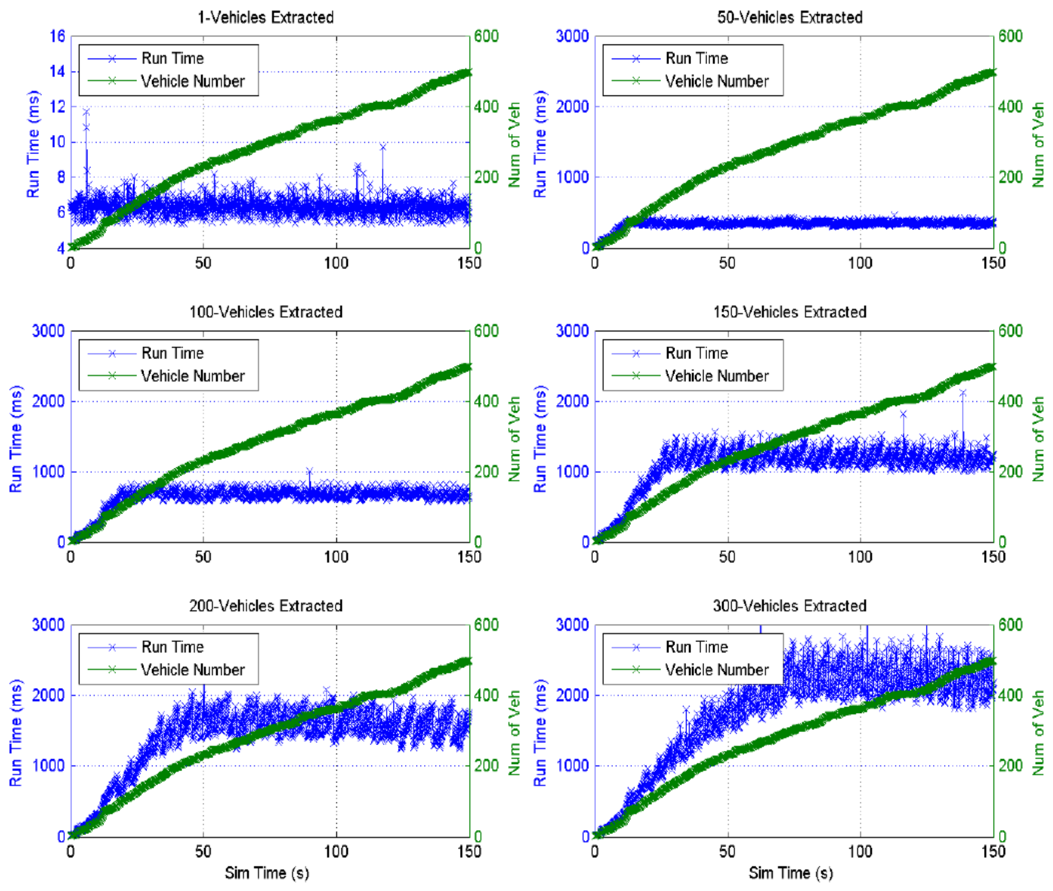


Figure 40. Execution time for VISSIM extraction

From figure 40, for the plots of different vehicle extraction values, the time exceeds 200 milliseconds by a large margin. This significant delay in the simulation with the increase in vehicle is contributed by the for loop approach which was previously used to extract multiple data from VISSIM. The most recent approach to multiple data extraction implements arrays which reduce the time for extraction drastically. The results with the array approach are discussed later. Thus it is known that VISSIM extraction is the reason why the 200 millisecond time step is not working when implemented in the C# code with the for loop structure. These preliminary results show that when a for loop is implemented instead of arrays, the time consumption for extracting multiple data from VISSIM simulation during runtime is significantly high. This situation is improved by implementing large arrays to extract multiple data from VISSIM without significantly impacting the time even when the data size is increased fourfold.

Next, a complete simulation of the COM middleware architecture is executed to examine the synchronization of the different components as well as investigate the timing of specific components to figure out any bottlenecks. The simulation is carried out to record the timing for both thread structures, Thread one and two for the Powertrain-COM and the VISSIM -COM as shown in figure 33. The functionality of threads one and two corresponding to VISSIM -COM and Powertrain-COM respectively, are the same as well as for threads two and one corresponding to VISSIM -COM and Powertrain-COM.
Threads one and two , VISSIM -COM and Powertrain-COM respectively , have the functionality of extracting data from an external software that is linked to the COM and then send it across a network to a specific port. Thread one in VISSIM -COM extracts vehicle information like vehicle number, position, speed and link and lane location at once in array form and sends it to the network as well as to the CACC controller built in Simulink that is simultaneously running locally. On the other hand, Thread two in powertrain-COM extracts processed information from Powertrain Simulink model and sends it back to VISSIM -COM. The running of Thread one VISSIM -COM is crucial in determining the complete synchronization of the structure as its performance is dependent on how much information is being extracted from VISSIM traffic simulator, whereas for the other threads like thread two of VISSIM -COM and threads one and two in powertrain-COM are responsible for passing the data as quickly as possible to maintain the desired time step for the whole architecture .

A complete synchronization of all the threads in both the COMs can be achieved by making sure the data extracted from VISSIM Traffic simulator is sent back with the processed data before the next time step for which the simulation is run. The idea is to provide VISSIM's internal model with the updated data so that it can use the updated data to run the next time step. Looking at the structure shown in figure 33, it can be said that it is crucial for thread one VISSIM -COM to maintain the desired time step in this case a time step of 200ms. This is because thread one is the main control thread which decides when to run the traffic simulator for one time step as well as when to extract different types of data from the simulator. Once a piece of data is extracted , the next time it should be extracted from VISSIM must be exactly after one time step is elapsed to maintain uniformity of time. However, the extracted data from VISSIM must now travel across all the other threads through a network, through different software like CACC controller in Simulink and Powertrain Simulink Model for processing and then back to thread two of VISSIM -COM before the thread one of VISSIM -COM runs the current simulation for another single step. To achieve this, all the threads through which the data travels must be synchronized .

Initially it was realized that all threads must maintain a time step of 200ms. However, after running a few simulation tests, it is realized that allowing all the threads to execute at the 200ms time step holds the data unnecessarily for long time and causes delay in the transmission. After many considerations it was decided that it is only logical to keep the control thread (Thread one VISSIM -COM) in a specific time step, and all the other threads must send the data received as soon as possible , so that the processed data can reach VISSIM traffic simulator before the next single step run is executed in the control thread. This is because the time step is completely dependent on the control threads as the control thread decides when to run VISSIM depending on the set time step. Therefore, following this logic, all the other threads follow a logic of listening for data over a continuous while loop, and as the data is received, the threads pass it on to the next thread or software. For this to occur, all the other threads must run the loop at a faster rate then the control thread which is made to run its loop at the desired times step. This means that if all the other threads loop over faster than the control threads, there is a possibility for thread two of VISSIM -COM to receive the same data over and over again if one time step is not complete. Since VISSIM -COM thread two is responsible for delivering the processed data back to the VISSIM simulation, it is fine if repeated data is sent to VISSIM simulation before the time step is over. However if thread two sends a repeated data exactly at the time when control thread runs a single step, then it will cause an error. Thus it is decided to put a flag value attached to the actual data extracted from VISSIM simulation. This flag value acts like an index which will always increment when a new data is extracted from VISSIM in the control thread. This flag value is then checked in all the other threads to realize a new data has been extracted which means a new time step has elapsed. Using this flag value, thread two of VISSIM -COM can make sure it does not send repeated data to the simulation software and instead send the updated data once and wait till the flag value increments for the next data to be updated in VISSIM simulation software. This check for the increment in the flag value allows the correct data to be replaced back in VISSIM Software before a time step is elapsed. To enable the actual data extracted from VISSIM traffic simulation and the flag value to travel together through the threads, they are placed as an element in an array. The maximum array size of data in all threads except the control thread is two elements. For control thread in VISSIM -COM the information array size can vary depending on the requirement of the user. The current control thread set up extracts multiple data from VISSIM traffic simulator as mentioned earlier and sorts the large array of data to extract the most important data. Since the elements in the array vary at every time step with increasing or decreasing number of vehicles, depending whether vehicles enter or exit the traffic simulation network at every time step, it is easier to set a constant large size for the array and let the data occupy or vacant the element space depending on the number of vehicles present during one single time step. It can be easily inferred that more the information is extracted from VISSIM, slower the running speed of control thread will become. It is crucial for the control thread to maintain the desired time step to keep the other threads synchronized. Any delay in the control thread will propagate the delay in the upcoming time steps in all the threads leading to break the simulation in VISSIM. It is observed that anytime a major delay occurs, for reasons like several vehicles running in simulation at a time step can increase the load on VISSIM Software, then the data processed through the threads update VISSIM in the wrong time step causing the internal model of VISSIM to loose control. This can lead to either the vehicles crashing in VISSIM simulation or stopping abruptly in the middle of the road causing all other vehicles to stop too. Therefore it is vital to determine the upper limit of the quantity of information that can be extracted from VISSIM and processed without any significant delay. Thus, the simulation was carried out for two different scenarios, for 200 and 800 vehicle information extraction. The results are discussed below.

3.3.4 Vehicle Information Extraction

As mentioned earlier, the quantity of vehicle information extraction plays a major role in maintaining the timing and synchronization of the middleware and its components. As per the current requirement, one simulation time step must be within 200ms precisely. To maintain such a tight time step it means that any delay in any of the components will cause the whole synchronization to fall behind in time. Taking into account that there are so many components , the threads structure is implemented as the core design of the middleware. The parallel running feature of the threads allows the middleware to divide its task in different parts based on the requirement and run each component responsible for a specific part to run as a separate process on the computer system. This design of decentralizing the working of the middleware, instead of using sequential code, definitely helps more in synchronizing the different components of the code. As mentioned earlier, the major tasks of the threads are to extract information from a third party software running simultaneously and send or receive the information across a network or locally.

Extraction of data starts in Thread one of VISSIM-COM as it is the initiator of the whole software loop. This is where VISSIM is initiated and run for every single time step. Thread one of VISSIM -COM is the busiest in terms of work load as it has to extract and send much more information than all the other threads . This thread has direct association with VISSIM, from where it extracts multiple data information as mentioned earlier. So for example, if 200 vehicle information has to be extracted then it extracts about 800 single data elements in the form of array as there are four data types for which it has to extract 200 elements each. The four data types are vehicle number, speed, lane and position. Thus for 800 vehicle information, it extracts 3200 individual elements in the form of array at every time step. Further, these multiple data elements have to be processed before they can be sent because not all of them are required for processing. Thread one's primary responsibility is to deliver information to CACC model running locally and also to a network.

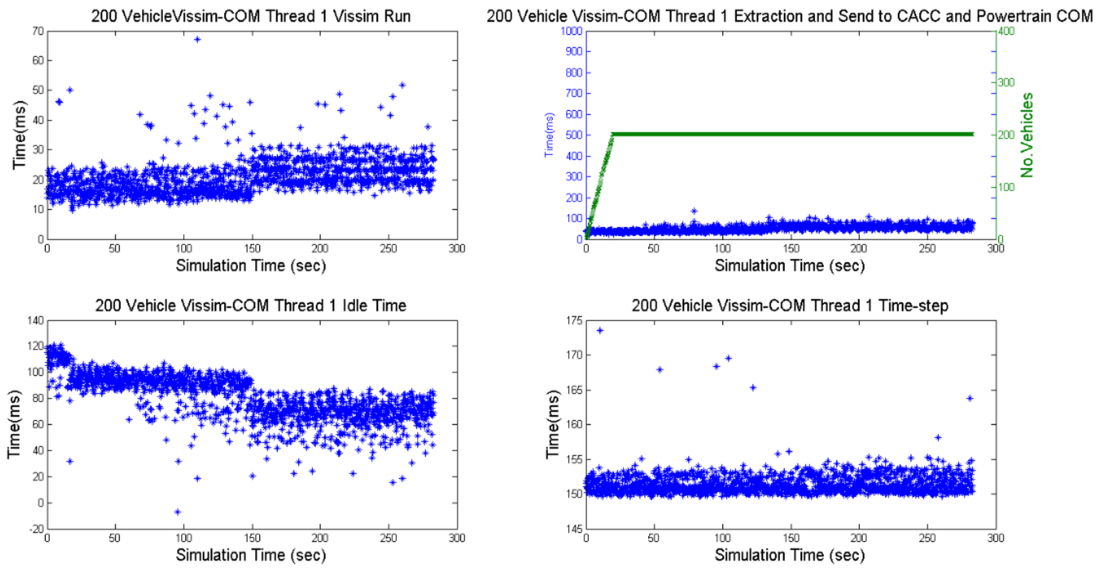


Figure 41.Timing for Thread one Vissim-COM for 200 vehicle information extraction

3.3.4.1 VISSIM-COM Thread 1 -200 Vehicle Extraction

The timing results shown in figure 41 are for 200 vehicle extraction of Thread one of VISSIM -COM. It is to be noted that 200 vehicle information includes four different data types; vehicle speed, position, number and lane, which means the extraction is for 800 individual data points in the form of array. The graph with the timing for VISSIM run clearly shows that it takes less than 50 ms for Thread one to run VISSIM for one time step. Most of the timing points are cluttered around 20-25ms. However, it is inferred that these delays which cause the timing to increase can be due to increasing number of vehicles in the simulation as simulation time increases. This is clearly depicted in the plot titled Extraction Timing. This plot shows the general increase of vehicles over simulation time till 200 vehicles are extracted. It takes about 25 seconds of simulation time for the traffic simulation to have 200 vehicles. Although the plot shows the vehicle number saturated at 200, it is to be noted that actual number of vehicles in the simulation reach to about 850 by the end of the simulation. That is the reason why at a later time, the timing for extraction increases slightly. Although COM is programmed to extract 200 vehicles, VISSIM simulation has more processing to execute with more number of vehicles in the network. The third plot gives the idea of how much idle time is left for thread one to elapse. Since the time step of 200 ms has to be maintained for the purpose of synchronization with other threads, it is important to pause thread one for the extra time till the one loop of thread one takes up 200ms. From the plot titled Idle time, it can be seen that at the beginning of the simulation when the total number of vehicles in the simulation are few, the idle time is in the range of 10-160 ms. However, over simulation time, there is a distinctive decreasing trend in the idle time, and this is again explained by the fact that the number of vehicles in the simulation increase. The last plot shows the total time for the thread to run one time step. This plot clearly showcases the capability of the thread structure by maintaining the time step values within 200 ms for the whole simulation . From these set of plots it can be guaranteed that the crucial tasks of thread one in VISSIM -COM , of running VISSIM single step and extracting 200 vehicle information, does not exceed the time step.

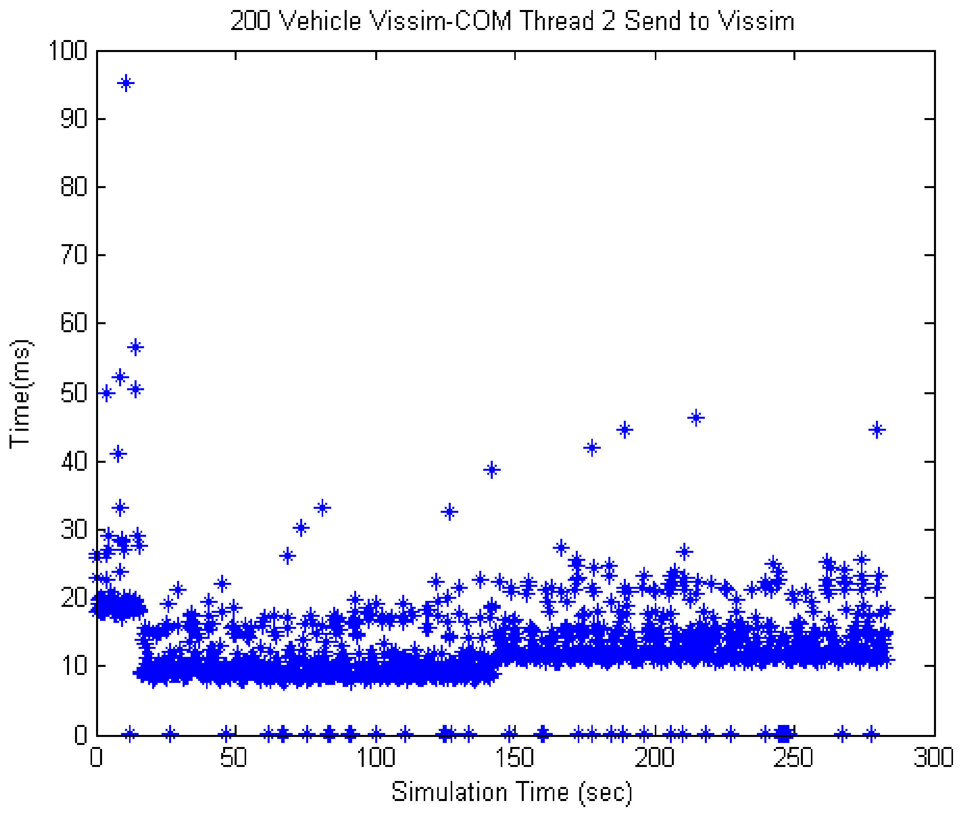


Figure 42. Timing for thread two Vissim-COM

3.3.4.2 VISSIM-COM Thread 2 - 200 Vehicle Extraction

Timing data for thread two of VISSIM -COM is recorded and displayed in figure 42. Thread two's crucial tasks are receiving the data from the network and sending it to VISSIM traffic simulator locally. It is also responsible for continuously looking for any new data when it is not sending the received data to VISSIM. This is clearly depicted in figure 42. Looking at the plot titled Send to VISSIM, conveys that it does take thread two of VISSIM -COM time for it to receive and send the data to VISSIM. Most of the cluster of points are around 20-50 ms. These set of plots clearly convey that the crucial task of receiving data from the network or the CACC SIMULINK model and sending to VISSIM Traffic simulator is definitely not a bottle neck.

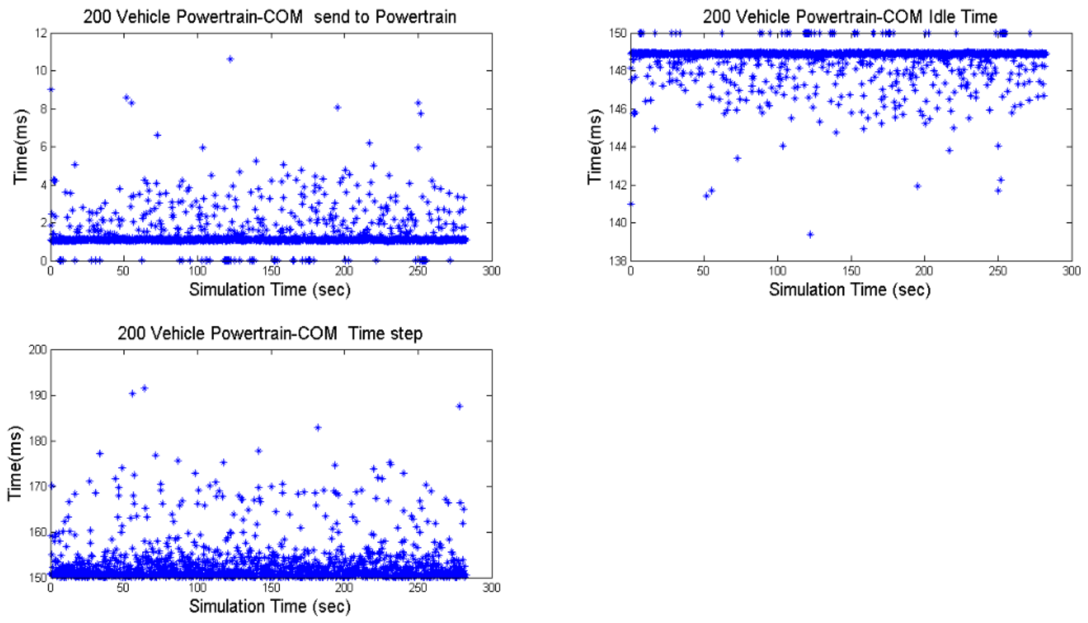


Figure 43. Timing for thread one Powertrain-COM

3.3.4.3 Powertrain-COM Thread 1 -200 Vehicle Extraction

Similar timing analysis was conducted for Powertrain-COM. For the timing analysis of Powertrain-COM without any threads, figure 43 shows three plots: time for sending data to Powertrain, idle time and time step plots. From the first plot titled send to Powertrain , it can be seen that the timing for sending data to Powertrain Simulink model is less than 10ms which is a significantly small timing value. The idle time and the time step plots clearly show that the Powertrain-COM code is capable of maintaining a time step of 200ms. The timing data points are consistently spread and do not show any particular trend in time which suggests that over simulation time, the time to execute the code increases. This is accounted for my the uniform size of array that is been received from network and sent to Powertrain through out the simulation run. Hence, with this plot it can be concluded that for one way network communication , the complete architecture of the middleware is validated to work in real-time.

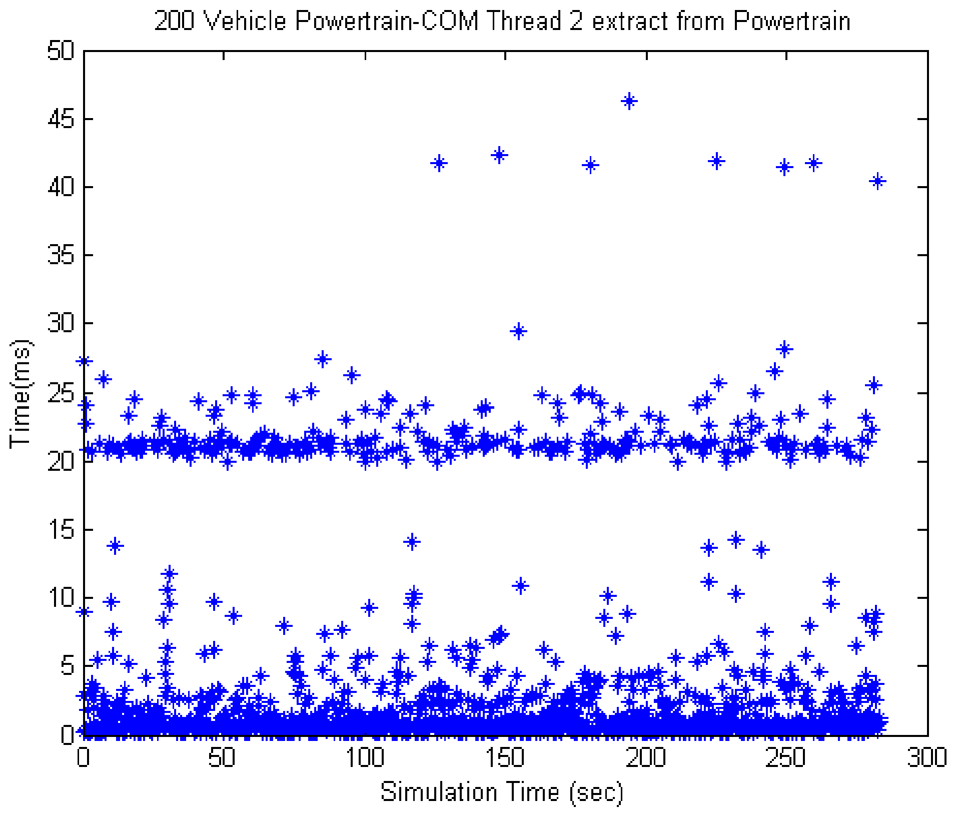


Figure 44. Timing for thread two Powertrain-COM

3.3.4.4 Powertrain-COM Thread 2 -200 Vehicle Extraction

Figure 44 shows the timing analysis plots for thread two of Powertrain-COM in the case of two way communication. It can be seen that the main task of extracting data from the Powertrain-Simulink model takes time in the range from 5ms to 25 ms with some outliers that exceed 30 ms. Like the other threads in the architecture, except the main control thread controlling VISSIM Traffic simulator, the major task of thread two of powertrain-COM does not consume too much time to cause any significant delay . All the above plots are for 200 vehicle information extraction, but the number of vehicle information extraction is only relevant for thread one of VISSIM -COM, the control thread, because all the other threads pass an array with two elements no matter how many vehicles information are extracted in one single time step.

3.3.4.5 VISSIM-COM Thread 1 -800 Vehicle Extraction

To compare the results with increasing number of vehicle information ,figure 45 is plotted. Figure 45 shows the timing results for VISSIM -COM thread one for 800 vehicle information extraction, which means 3200 elements are extracted at each time step. The extraction and send plot clearly show an increase in time consumption as the number of vehicle increases to 800 but compared to the extraction and send plot for 200 vehicles in figure 41, the increase in time is not significantly different. The reason behind this is the change in VISSIM's method of accessing attributes from the previous versions where for accessing multiple data information during simulation run, a for loop had to be implemented which definitely took more time compared to extracting multiple data all at the same time using an array. Due to this added feature the time step plot and the ideal time plot also do not differ much from that of 200 vehicle information extraction. This is very advantageous for real time applications such as the COM architecture, in this case, because even a fourfold increase in the vehicles population does not change the processing time significantly . Comparing the timing analysis for 200 and 800 vehicle information in figure 41 and 45 with the timing analysis for up to 300 vehicle extraction in figure 40, it can be concluded that the efficacy of the program has drastically improved with the change in the method of extracting information. Using the array method to extract multiple data has reduced the timing for extracting 300 vehicle information from 2000ms to 50-70 ms for 800 vehicle information extraction. This feature of the code has given the user the leverage to modify the VISSIM traffic simulation software and increase its complexity not only in terms of increasing the number of vehicles in the network but also to make the traffic network more complex with several branches and traffic signal junctions. Further investigation with respect to extracting more than 800 vehicles information was not deemed necessary at this point because the current application of CACC does not need even 200 vehicle information. However, with 800 vehicle extraction it is showcased that the middleware has a robust architecture. With this drastic decrease in time for extraction, now more realistic simulations can be tested. Although the plots below show that the respective tasks of the threads do not contribute to any delay, it can be seen in the time-step plot that there exists a slight delay after 250 seconds of simulation. The delay occurs for two time-steps for about 400ms and 1600ms, but they are not deemed significant, since the time step is 200ms which is very small. Therefore, one or two time step delays are negligible. However, if there were frequent occurrences of delays, then the problem would be more serious. This particular delay is accounted for by the functioning of the operating system because from previous time analysis plots it is proved that the particular tasks of the thread do not contribute to the delay.

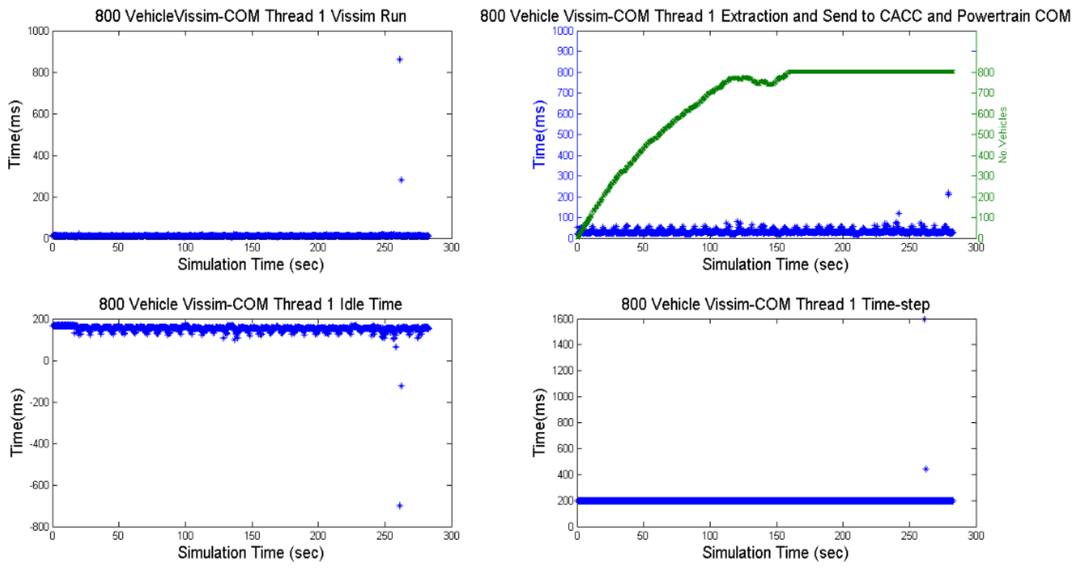


Figure 45. Timing for thread one Vissim-COM for 800 vehicle information extraction

3.3.5 Synchronization of VISSIM-COM Thread one and Thread two

When CACC application is incorporated in the software structure, synchronization of the different component becomes vital for real time simulation. The synchronization of the data travelling from VISSIM traffic simulator, to thread one, to CACC Simulink model , to thread two and back to VISSIM traffic simulator is a very crucial path for maintaining the synchronicity of the whole software structure. Keeping this in mind, a simulation test was conducted to determine if the data extracted from VISSIM is sent back to VISSIM in time so that the simulation can be updated in the next time step. The figure below shows the superimposition of the difference in the distance between a target vehicle and its preceding vehicle. The graph compares the distance calculated in MATLAB, and also extracted as an attribute from VISSIM. The idea is that if the two distances completely superimpose on each other with respect to simulation time on the x-axis, then it can be said that the data extracted from VISSIM , sent for processing to CACC controllers is appropriately updated for the next time step and hence the components of the middleware architecture are synchronized. These two superimposed plots also indicate that the CACC controller is indirectly able to influence the target vehicle in the simulation. If there was any discrepancy, the plots would not superimpose leading to the conclusion that either CACC controller is not able to influence the specific target vehicle in the simulation or there is some problem in the path of data which is leading to some sort of delay. This figure further supports the conclusion that the internal driver model of VISSIM can be completely overthrown by using the method mentioned earlier because the distance obtained from the CACC Simulink model and from VISSIM attributes match exactly .

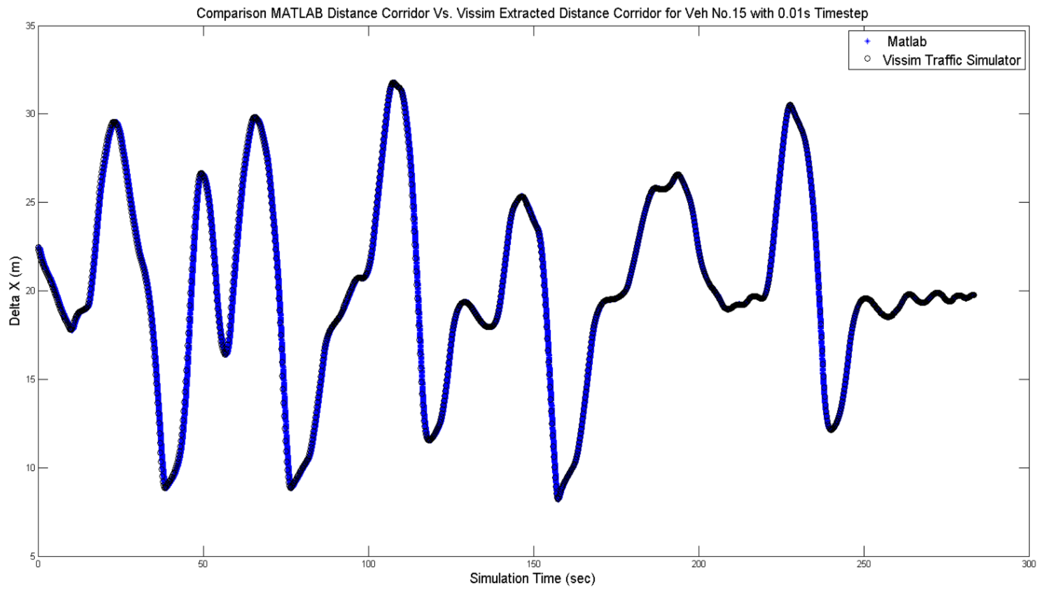


Figure 46. Distance Corridor comparison extracted from CACC controller and Vissim Simulation

3.4 Conclusion

From the above analysis of the different components of the thread structure, it can be concluded that the current architecture is validated to conduct real-time execution of the CACC architecture shown in figure 32. The thread method gives the leverage to synchronize several different software platforms and execute the simulation in real time. For this thesis, the focus is to implement the CACC methodology described in Chapter 2 with the HiLS. The timing analysis for the different components of the architecture clearly show that a simulation time-step of 200ms can be easily achieved with all of the crucial tasks finished within a range of 150ms to 170ms. Most of the crucial tasks of the architecture are conducted in control thread, VISSIM-COM thread one, and each of the timings for each of the tasks in thread one have proved to obey the 200ms designated time-step. The timing results clearly showcase the versatility of the middleware handling multiple software and communicating locally as well as through a network like the internet. The robustness of the middleware has been tested for 200 vehicle information and 800 vehicle information extraction from VISSIM with minimal difference in the timing results. Therefore, it can be concluded that the current middleware can successfully handle CACC application in integration with the powertrain-research-platform and execute in real-time.

Future work will focus on implementing other complex applications and attempt will be made to make the structure even more efficient so that a smaller time-step of approximately 100ms can be implemented. As mentioned in the report earlier, the current structure does one way communication with respect to network communication, sending data from VISSIM-COM to Powertrain-COM, but two way communication is a possibility with a slight change in the structure. If timing is concerned, the current architecture, with threads, can handle one and two way communication with efficacy.

Chapter 4 Evaluation of CACC Using Experimental Results

From the various vehicle level simulation cases observed in Chapter 2, it is determined that the median method with the averaged velocities of the preceding vehicles has potential fuel benefits for a controlled vehicle in a traffic network. In this section, the focus is to validate the CACC algorithm using a Hardware in the Loop Simulation(HiLS) where powertrain and engine dynamics will be included using a virtual powertrain as well as a real engine. As mentioned in chapter 3, the CACC software architecture is implemented with the powertrain-research -platform. The different components of the HiLS are explained briefly.

4.1 HiLS Components

The HiLS consists of the powertrain-research-platform which represents the CACC controlled vehicle in VISSIM, and VISSIM traffic simulator which provides the controlled vehicle dynamics and road conditions to the powertrain-research-platform. The powertrain-research-platform allows real-time measurement of fuel using the AVL's Fuel Measurement System Model P402. Unlike the method of fuel measurement mentioned in simulation results previously which used a static fuel map, the fuel measurements using AVL are much more accurate as it has a high bandwidth that takes care of the transient driving maneuvers effectively.

The emissions measurement is conducted using AVL's SESAM-FTIR. Each constituent of the exhaust gas is measured in terms of concentrations (ppm). Therefore, the formula below is used to convert the unit to mass-rate in grams per second.

$$\frac{g}{s} = \left(\frac{PPM}{10^6 moles\ of\ exhaust\ gas}\right) \times \left(\frac{exhaust\_mass\_rate}{exhaust\_molar\_mass}\right) \times constituent\_molar\_mass$$

PPM is the measurement unit for micromole concentration per mole of exhaust gas. The exhaust gas mole-rate (in moles per second) is obtained using the exhaust mass-rate (in grams per second) and the exhaust molar-mass for diesel fuel (29.4 grams per mole). The exhaust mass-rate can be determined by summing the measured intake-air and fuel mass-rates. The mole-rate of the constituent can be determined by multiplying the micromole concentration of the constituent with the exhaust mole rate. The mass-rate of the constituent is determined using the product of the mole-rate and the molar-mass of the constituent.

4.1.1 VISSIM Microscopic Traffic Simulator

VISSIM is a microscopic traffic simulator which uses the Wiedemann's car following model (Wiedemann, 1974) . The software allows users to simulate realistic traffic scenarios and access information related to individual vehicles. The attributes that VISSIM allows users to access are related to vehicle dynamics like the vehicle speed and acceleration.

4.1.2 HiLS Middleware

The HiLS middleware consists of the Powertrain-COM and VISSIM-COM. Local communication between the COM structures and the different software entities are discussed in Chapter 3 in detail[20]. Internet network communication between Powertrain-COM and VISSIM-COM is also explained in Chapter 3.

4.2 Test Results and Discussions

4.2.1 Highway Driving

First an offline test for simulation scenario, shown in figure 25, 14 vehicle average for highway driving is conducted using the powertrain-research-platform. The velocity profiles of the controlled vehicle and the preceding vehicle are stored offline by simulating the CACC architecture without the powertrain-research-platform in the loop. The idea is to compare the results of the simulation with an actual test using the powertrain-research-platform. The powertrain-research-platform emulates a Hybrid Electric Vehicle(HEV) which uses a Power Sharing Transmission (PST) which is controlled by rule-based optimization method [20]. Without going into much of the details of an HEV, to obtain fair and comparable CACC simulation results, it is necessary to maintain the battery state of charge (SOC) the same at the beginning and at the end of the simulation. Maintaining the SOC at the same level will imply that both the controlled vehicle and target vehicle have used the same battery power provided by the battery through the motor/generator . This will make the comparison fair in a powertrain that emulates HEV .For the experimental analysis, the complete plots of generator, motor torque and the battery SOC are included in Appendix A for all cases. The SOC plots clearly show the initial and final SOC are maintained at 0.6 at the beginning and end of simulation.

Experimental results for a highway driving case are obtained for the 14 vehicles' velocities average simulation case as shown in figure 25,. This showcases the advantage of using HiLS with a traffic simulation package that can simulate variety of realistic driving cycles for vehicles. Figure 47 shows reference and actual velocity profile, engine speed , engine torque and the fuel consumption of the controlled vehicle for 14 preceding vehicles' velocities averages. The total fuel consumption is around 180g which is lower than the total fuel consumption of the city driving case, discussed next, because of the less aggressive driving behavior in a highway situation. This trend was also depicted in the simulation cases for 14 vehicles' velocities averages. The driver does not tend to vary the velocity of the vehicle too much over the driving cycle. Since the preceding vehicles in the platoon too do not follow an aggressive behavior, the averaged velocity for the controlled vehicle is less aggressive.

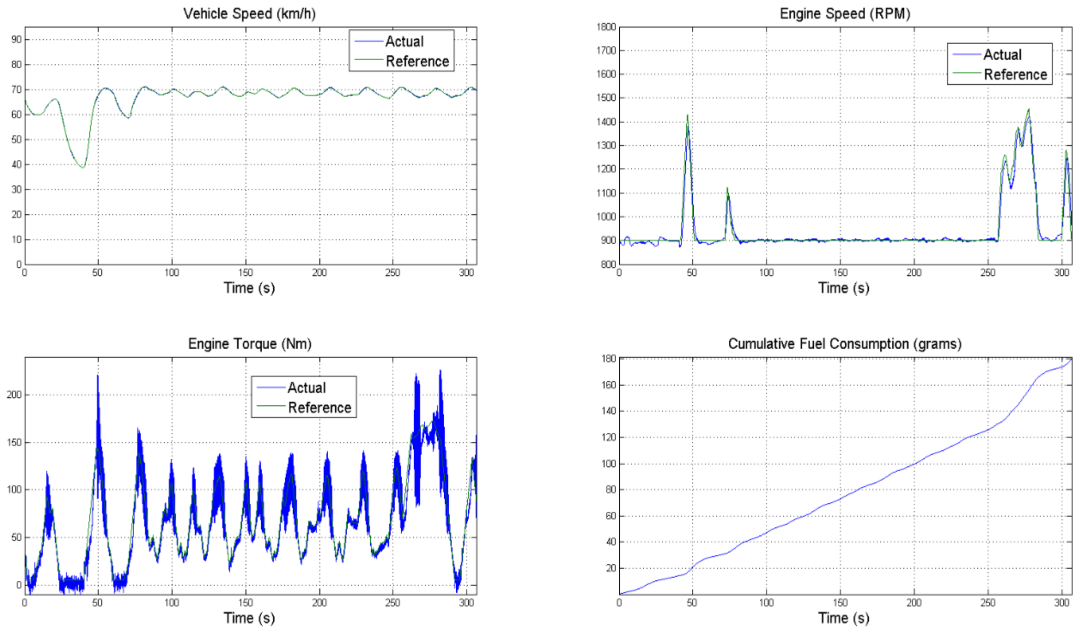
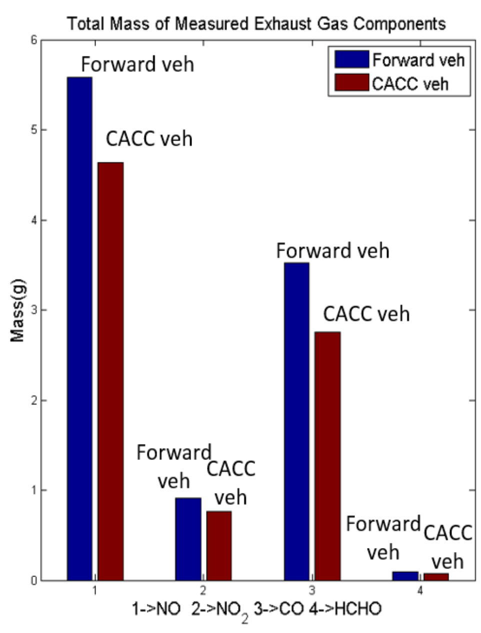


Figure 47. Powertrain and Vehicle Dynamics for Controlled Vehicle- 14 vehicles' velocities average for highway driving



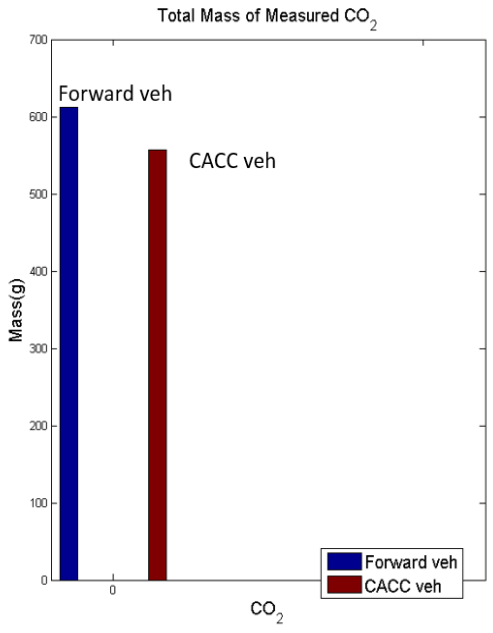


Figure 48. Emissions Measurements for Controlled Vehicle Vs. Preceding Vehicle- 14 vehicles' velocities average

Figure 48 compares the emissions measurements for the controlled and the immediate preceding vehicle. It is clearly seen from the plots that for all the pollutants emitted, the quantity emitted with combustion gas for the controlled vehicle is lower than that from the preceding vehicle. This is analogous to the engine torque magnitudes of the two vehicles. From figure 47 and 49, the engine torque magnitudes of controlled vehicle are much lower than that of the preceding vehicle.

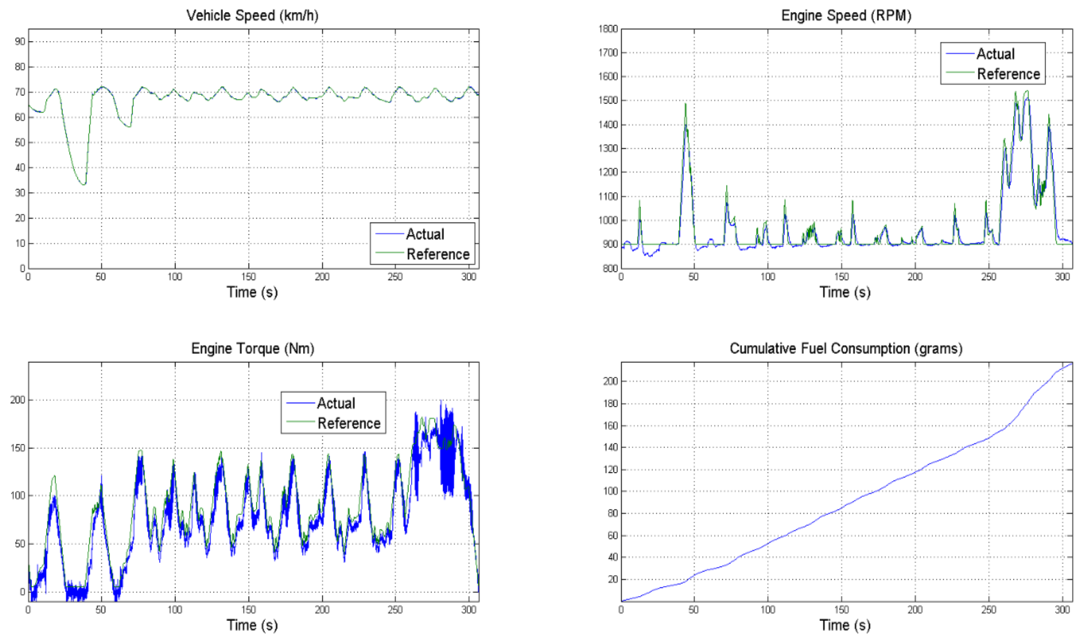


Figure 49. Powertrain and Vehicle Dynamics for immediate preceding vehicle-highway driving

Figure 49 shows the powertrain and vehicle dynamics for the immediate preceding vehicle in a highway situation. From the plots the actual seems to track the reference engine speed and torque very well except for the last portion of the simulation where the actual engine torque fluctuates. The total fuel consumption for the complete driving cycle is 216g for the preceding vehicle.

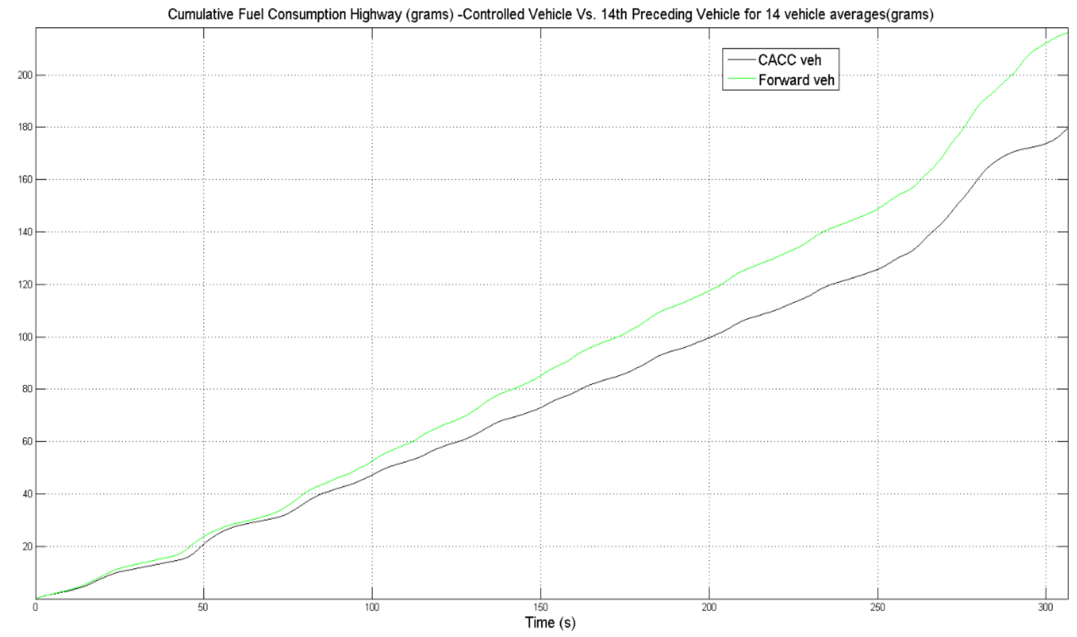


Figure 50. Total fuel consumption comparison- Controlled Vs. Preceding Vehicle highway driving

Figure 50 shows the total fuel consumption comparison for the CACC controlled and the immediate preceding vehicle. The fuel saving for the controlled vehicle is around 16.6% for 14 vehicles' velocities averages in a highway situation as depicted in figure 25. The fuel measurements using the powertrain-research-platform can be deemed accurate, and tend to provide higher fuel savings compared to the vehicle level simulations because the HEV powertrain is controlled using rule-based optimization methods.

Ultimately these results for the highway case validate the previously obtained simulation results that taking the average of the preceding vehicles' velocities provides potential fuel benefits. This also shows that the CACC approach is not only able to achieve significant fuel benefits from a city driving case but even from the highway driving case where the room for any fuel benefit is lower than that of the city driving case due to the less aggressive driving behavior of vehicles on highways.

4.2.2 Experimental results of real-time CACC architecture- Local driving with constant and varying platoon size

Unlike the previous experimental results where the vehicle velocity trajectories were stored offline, the results below are obtained from the implementation of real-time CACC architecture as shown in figure 32. The results are for 14 preceding vehicles' velocities averages in a city driving scenario. Throughout the traffic network, a platoon of 15 vehicles travel together where the 15th vehicle is the controlled vehicle whose velocity is derived by taking the average of the 14 preceding vehicles and the 14th vehicle is the immediate preceding vehicle.

4.2.2.1 City Driving

Figure 51 shows the vehicle dynamics and the fuel consumption respectively for the controlled vehicle. Figure 52 compares the emissions measurements from the controlled and the preceding vehicle, whereas figure 53 shows the vehicle dynamics and fuel consumption for the immediate preceding vehicle to the controlled vehicle. The fuel saving for the controlled vehicle, relative to the preceding vehicle, are approximately 18.5% .

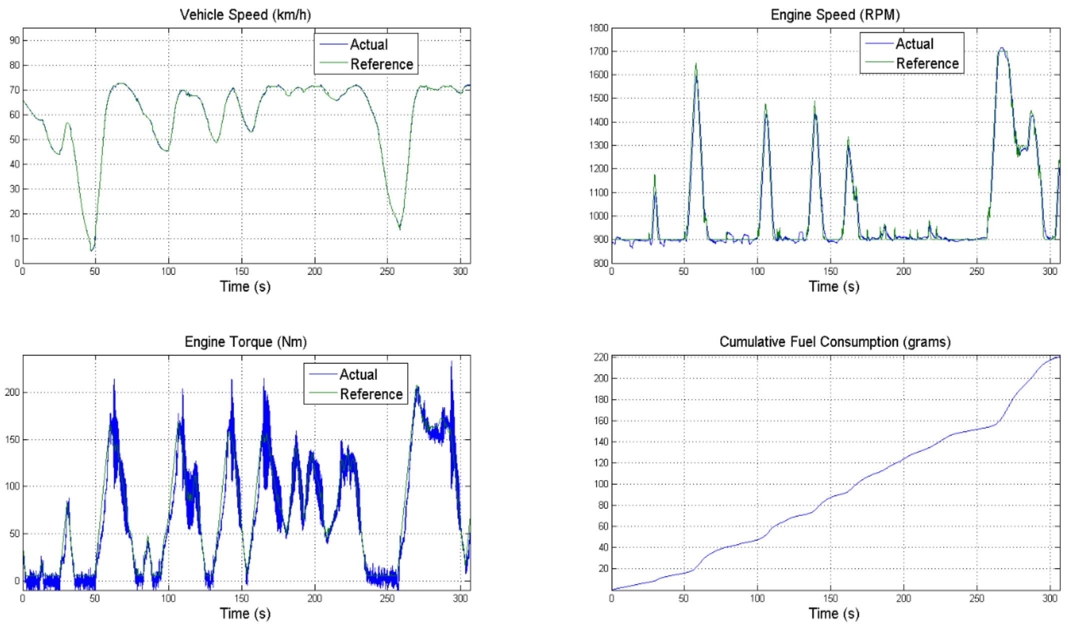


Figure 51. Powertrain and Vehicle Dynamics for Controlled Vehicle- 14 vehicles' velocities average city driving

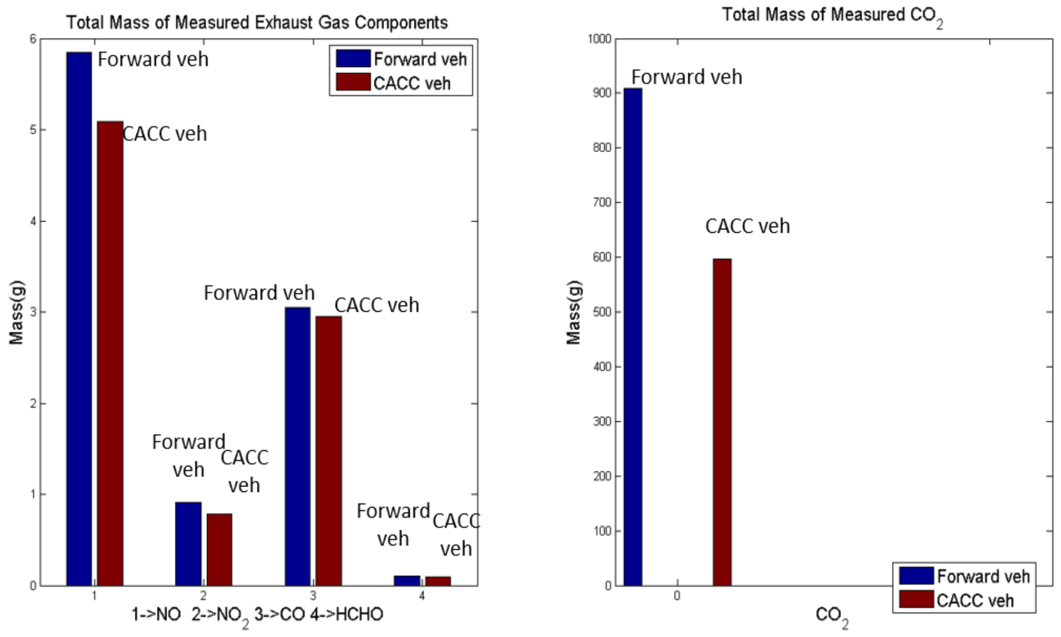


Figure 52. Emissions Measurements for Controlled Vehicle Vs. Preceding Vehicle- 14 vehicles' velocities average

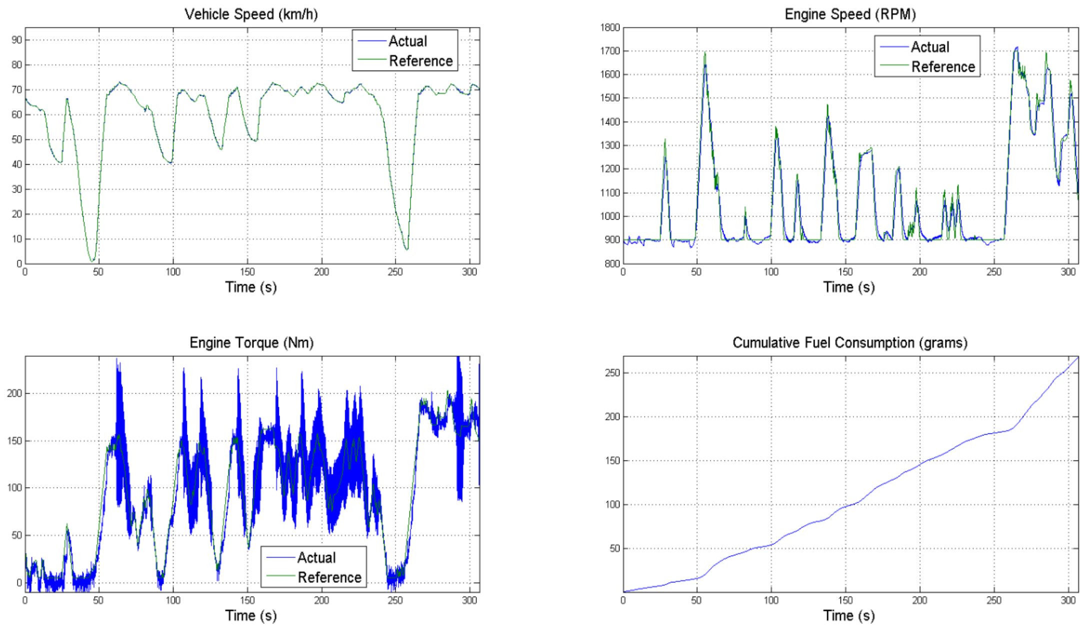


Figure 53. Powertrain and Vehicle Dynamics for immediate preceding vehicle- city driving

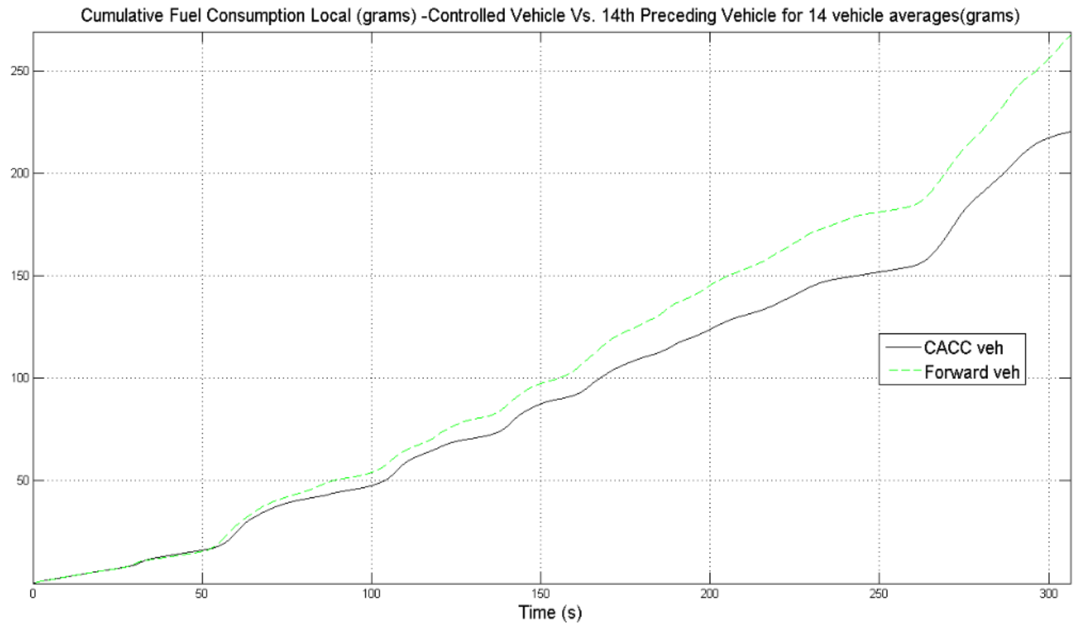


Figure 54. Total fuel consumption comparison- Controlled Vs. Preceding Vehicle city driving

The experimental tests provide strong evidence that the CACC method implemented has significant potential to not only improve fuel consumption but also reduce emissions which is validated using results from a very accurate measurement device, the AVL's SESAM-FTIR.

4.2.2.2 Varying Platoon Size

A second online CACC experimental result is obtained but for a more realistic scenario. In real traffic scenarios it is not necessary that the size of the platoon will remain the same for the whole driving cycle. There are several traffic junctions at which vehicles will get added and subtracted from the platoon depending on whether vehicles join or leave at a junction. Hence, a similar simulation is prepared in VISSIM traffic simulator to determine the effect of vehicles leaving and joining a platoon on the fuel consumption of the controlled vehicle. This scenario has already been simulated as shown by the simulation results in figure 27. The vehicle level fuel savings obtained in simulation are about 10.4% which is lower than the city driving case for a constant platoon size.

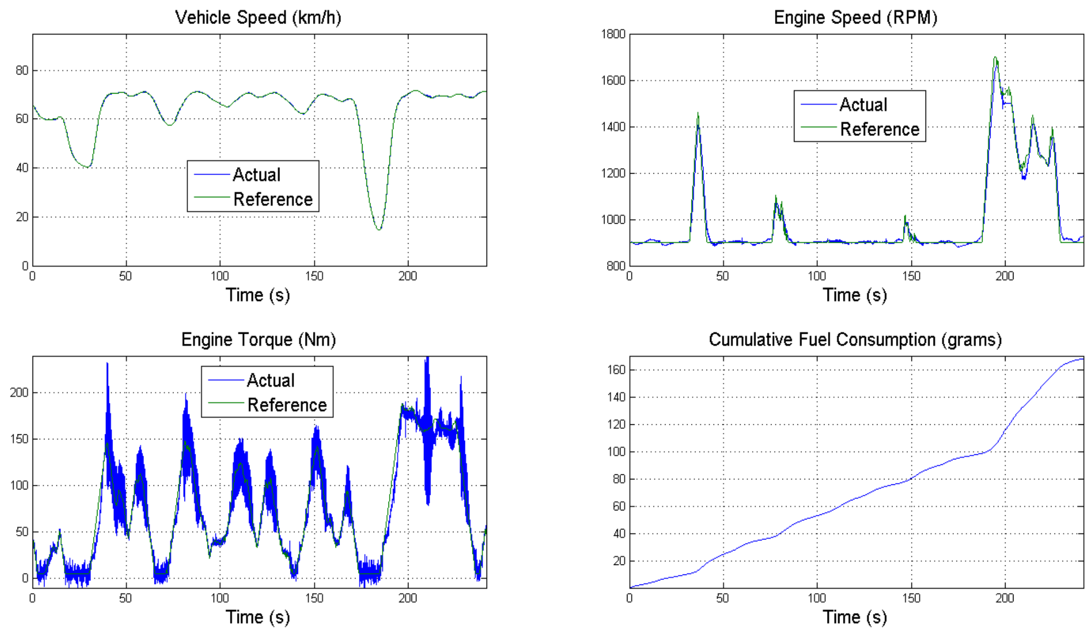


Figure 55. Powertrain and Vehicle Dynamics for Controlled Vehicle- Varying Platoon Size

Figure 55 shows the dynamics for the controlled vehicle and it can be seen that the engine speed is very well tracked. However, the engine torque experiences some occurrences of actual and reference torque discrepancies. The fuel consumption is approximated to be 168g for the complete driving cycle of the controlled vehicle.

Figure56 compares the emissions for the controlled and the immediate preceding vehicle. Like the previous results from the city driving case shown in figure 52, in this varying platoon size case too the emissions are higher for the immediate preceding vehicle.

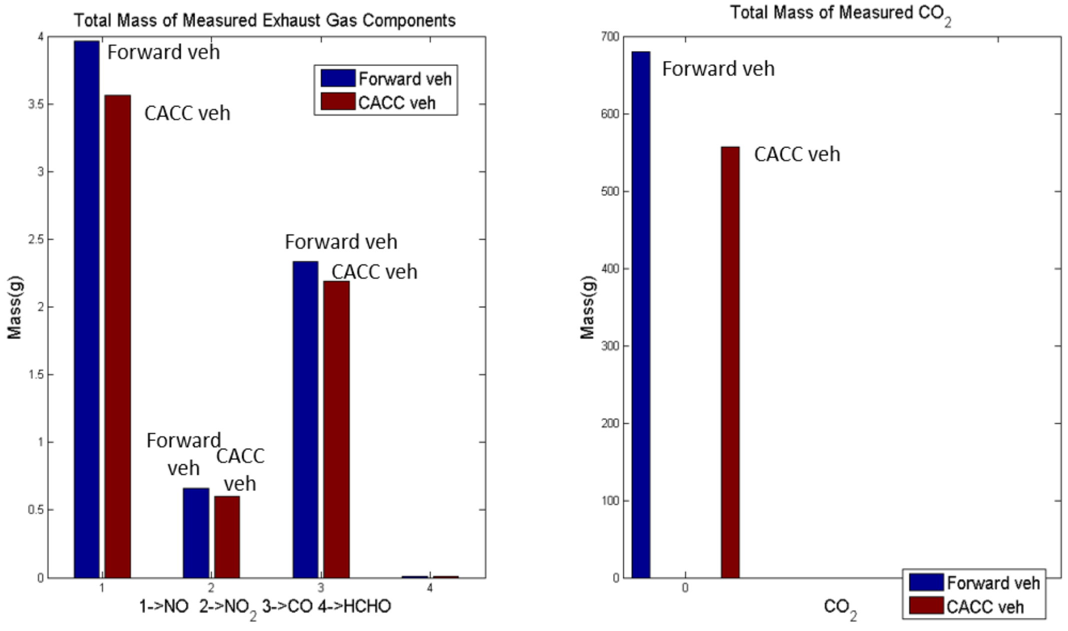


Figure 56. Emissions Measurements for Controlled Vehicle Vs. Preceding Vehicle-Varying Platoon Size

Figure 57 shows the vehicle dynamics for the immediate preceding vehicle with the actual engine speed tracking the reference speed very well. However, the engine torque tracking has some discrepancy. The fuel consumption for the preceding vehicle is approximately 200g for the whole driving cycle.

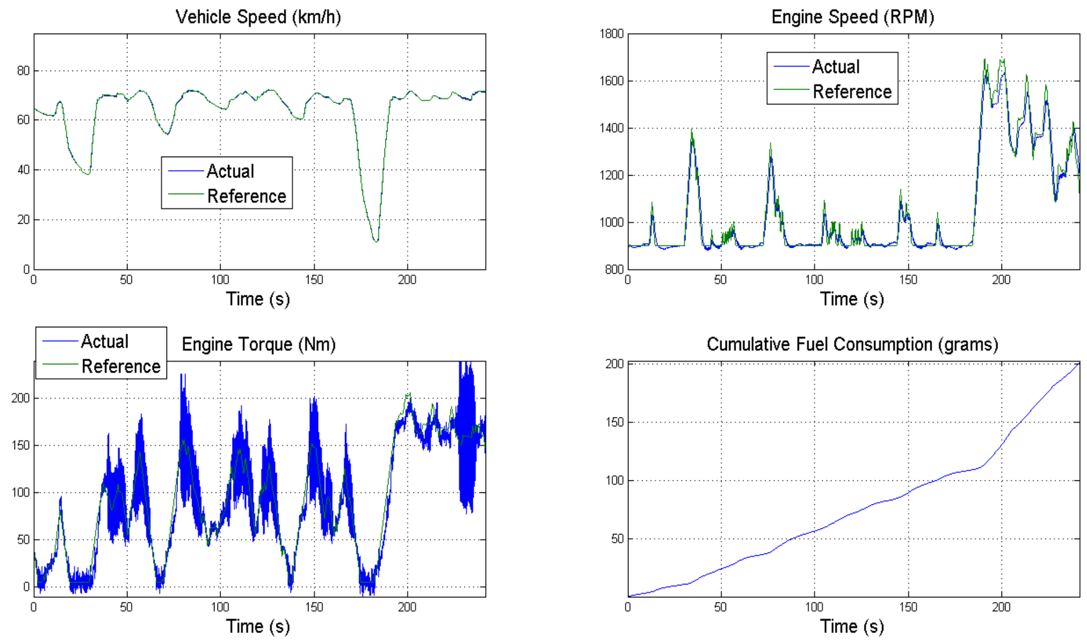


Figure 57 Powertrain and Vehicle Dynamics for immediate preceding vehicle

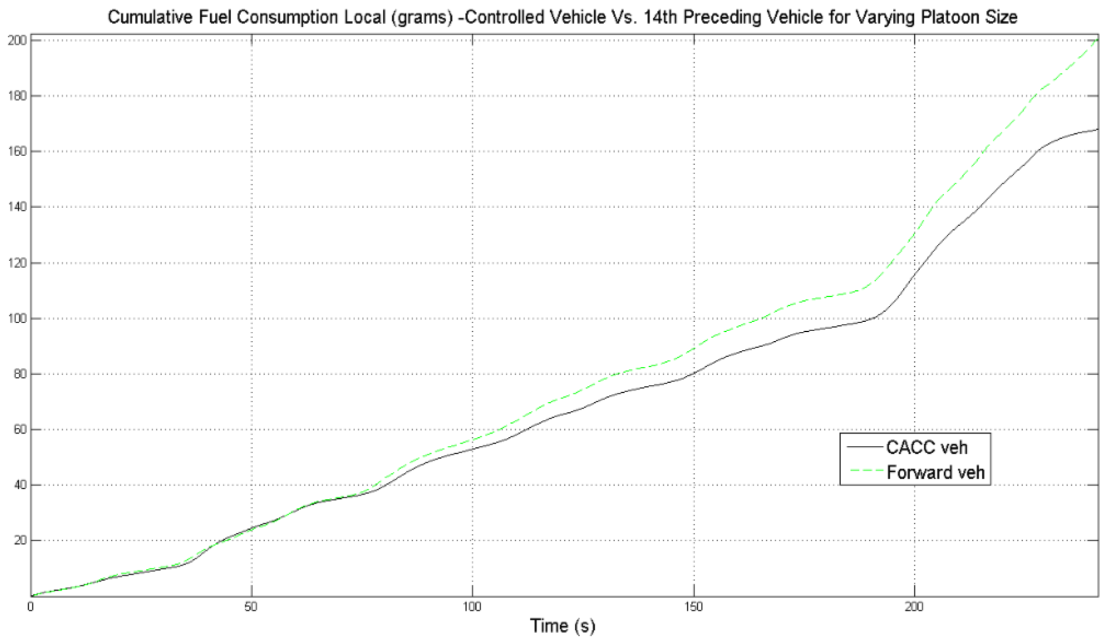


Figure 58 Total fuel consumption comparison- Controlled Vs. Preceding Vehicle Varying Platoon size

Figure 58 shows the fuel consumption comparison for the controlled and the immediate preceding vehicle. It must be noted that the fuel consumption comparison between the controlled and the preceding vehicle is viable in this case because for the whole driving cycle the simulation was designed in such a way that the immediate preceding vehicle never leaves the platoon. Therefore the controlled vehicle follows the immediate preceding vehicle for the whole driving cycle. Also, the battery state of charge is maintained to be at the same level as it was initially, to provide a fair comparison of vehicles on a HEV powertrain. The plots for battery state of charge are shown in Appendix A figures A5 and A6 for both vehicles.

From the above fuel consumption plot it is determined that the fuel benefit from both the vehicle level and the optimized[20] powertrain level is approximately 16% .The fuel benefit obtained from the vehicle level simulation of the exact scenario is 10.4% as shown in figure 27. The fuel consumption results obtained from powertrain-research-platform for HEV clearly validate the simulation results that even with a varying platoon size, taking the average of the preceding vehicles' velocities in a platoon provides significant fuel benefits.

4.3 Conclusion

The experimental results obtained for both the real time execution of the CACC architecture, with(online) and without(offline) HEV powertrain-research-platform in the loop ,very clearly support the simulation results that using the controlled vehicle's velocity as a function of the preceding vehicles' velocities average gives potential fuel benefits for the controlled vehicle. The table below summarizes the results obtained from the experiment.

Table 3.Summary of Experimental and Simulation Results

Case	Experimental Fuel Benefit %
Offline Highway	16.6
Online City	18.5
Online Varying Platoon size	16.0

The offline experimental results for the highway driving case had fuel saving of approximately 16.6% whereas the online city driving case had 18.5%. The real-time execution of the CACC architecture gave 16% fuel saving for the city driving case with varying platoon size. This consistency in the fuel gains demonstrates the versatility of the controller for different traffic conditions that it is used in. This phenomenon is a result of providing more information to the controlled vehicle. Using the preceding vehicles' velocities provide the controlled vehicle with dynamics of the preceding vehicles for the different traffic situation the preceding vehicles confront. This ahead of time reaction of the preceding vehicles helps the controlled vehicle make crucial judgements with regards to its dynamics for the upcoming path of road in front of it . For instance if the lead vehicle , approximately at 300m from the controlled vehicle, stops at a traffic signal, then the controlled vehicle can decide to reduce its own velocity to give it sufficient time between the traffic signal state change and distance between itself and the immediate preceding vehicle for it to not completely stop at the signal and continue moving at that speed. If this behavior is repeated over a complete driving cycle, as already demonstrated by the simulations and the experimental results, the controlled vehicle obtains a smoothened velocity profile compared to the preceding vehicle which reduces the dynamics in its engine torque and speed that helps reduce the overall fuel consumption. Therefore, from both the simulation and experimental results it can be claimed that using the preceding vehicles' velocities and incorporating them in the dynamics of the controlled vehicle has potential fuel benefits.

Future work will primarily consist of implementing CACC application for a large platoon of vehicles where all the vehicles in the platoon will be controlled by CACC. Further investigation will be conducted using a communication model where the success and failure rates of IVC and VII communication will be included in the algorithm to determine the effect on fuel consumption. With the current results it is proved that using the preceding vehicles' velocities provides potential fuel benefits. Therefore an approach to design a systematic method where a possible online optimization method with direct relation to fuel consumption will be developed.

References

[1] S. Seshagiri and H. K. Khalil, “Longitudinal adaptive control of a platoon of vehicles,” *Proc. 1999 Am. Control Conf. (Cat. No. 99CH36251)*, vol. 5, pp. 291–296, 1999.

[2] S. Li, K. Li, R. Rajamani, and J. Wang, “Model predictive multi-objective vehicular adaptive cruise control,” *IEEE Trans. Control Syst. Technol.*, vol. 19, no. 3, pp. 556–566, 2011.

[3] L. Luo, H. Liu, P. Li, and H. Wang, “Model predictive control for adaptive cruise control with multi-objectives: comfort, fuel-economy, safety and car-following,” *J. Zhejiang Univ. Sci. A*, vol. 11, no. 3, pp. 191–201, 2010.

[4] B. Asadi and A. Vahidi, “Predictive cruise control: Utilizing upcoming traffic signal information for improving fuel economy and reducing trip time,” *IEEE Trans. Control Syst. Technol.*, vol. 19, no. 3, pp. 707–714, 2011.

[5] Ma, Yongchang, "A Real Time Traffic Condition Assessment And Prediction Framework Using Vehicle Infrastructure Integration With Computational Intelligence”. *Clemson Univ.* pp.42-62, 2008.

[6] E. Paikari, S. Tahmasseby, and B. Far, “A simulation-based benefit analysis of deploying connected vehicles using dedicated short range communication,” *IEEE Intell. Veh. Symp. Proc.*, pp. 980–985, 2014.

[7] F. Bai and H. Krishnan, “Reliability Analysis of DSRC Wireless Communication for Vehicle Safety Applications,” *IEEE Intell. Transp. Syst. Conf.*, pp. 355–362, 2006.

[8] X. Chen, H. H. Refai, and M. Xiaomin, “A quantitative approach to evaluate DSRC highway inter-vehicle safety communication,” *GLOBECOM - IEEE Glob. Telecommun. Conf.*, pp. 151–155, 2007.

[9] J. B. Kenney, “Dedicated short-range communications (DSRC) standards in the United States,” *Proc. IEEE*, vol. 99, no. 7, pp. 1162–1182, 2011.

[10] B. Van Arem, C. J. G. Van Driel, and R. Visser, “The impact of cooperative adaptive cruise control on traffic-flow characteristics,” *IEEE Trans. Intell. Transp. Syst.*, vol. 7, no. 4, pp. 429–436, 2006.

[11] T. Stanger and L. del Re, “A model predictive Cooperative Adaptive Cruise Control approach,” *Am. Control Conf. (ACC)*, pp. 1374–1379, 2013.

[12] J. Hu, Y. Shao, Z. Sun, M. Wang, J. Bared, and P. Huang, “Integrated optimal eco-driving on rolling terrain for hybrid electric vehicle with vehicle-infrastructure communication,” *Transp. Res. Part C Emerg. Technol.*, vol. 68, pp. 228–244, 2016.

[13] S. E. Li and H. Peng, “Strategies to minimize the fuel consumption of passenger cars during car-following scenarios,” *Proc. Inst. Mech. Eng. Part D J. Automob. Eng.*, vol. 226, no. 3, pp. 419–429, 2012.

[14] S. E. Li, H. Peng, K. Li, and J. Wang, “Minimum fuel control strategy in automated car-following scenarios,” *IEEE Trans. Veh. Technol.*, vol. 61, no. 3, pp. 998–1007, 2012.

[15] Schmied, R., Waschl, H., and del Re, L., "A Simplified Fuel Efficient Predictive Cruise Control Approach," SAE Technical Paper ,2015.

[16] D. Lang, T. Stanger, and L. Del Re, "Fuel efficient quasi optimal adaptive cruise control by control identification," *Proc. IEEE Int. Conf. Control Appl.*, pp. 229–234, 2013.

[17] H. Hu, Z. Zou, and H. Yang, "On-board Measurements of City Buses with Hybrid Electric Powertrain , Conventional Diesel and LPG Engines," *Power*, vol. 4970, pp. 1–7, 2009.

[18] B. Daham, G. Andrews, H. Li, B. Rosario, M. Bell, and J. Tate, "Application of a portable FTIR for measuring on-road emissions," *Eprints.Whiterose.Ac.Uk*, vol. 724, 2005.

[19] Y. Wang and Z. Sun, "A Hydrostatic Dynamometer Based Hybrid PowertrainResearch Platform," *Proc.International Symposium on Flexible Automation.*,pp. 1–8, 2010.

[20] M. Azrin, M. Zulkefli, J. Zheng, Z. Sun, and H. X. Liu, "Hybrid powertrain optimization with trajectory prediction based on inter-vehicle-communication and vehicle-infrastructure-integration," *Transp. Res. Part C*, vol. 45, pp. 41–63, 2014.

[21] Z. Sun, Y. Wang, and K. A. Stelson, "Nonlinear Tracking Control of a Transient Hydrostatic DynamometerFor Hybrid Powertrain Research," *Proc.ASME Dynamic Systems and Control Conference.*,pp. 1–8, 2010.

[22] Ciubotaru, Bogdan. *Advanced Network Programming – Principles and Techniques*. London: Springer, 2013.

[23] Mosher, Microsoft Outlook Programming: Jump Start for Administrators, Developers , and Power Users ,ISBN 1-55558-286-9, pp.624, 2002.

[24] T. Tettamanti and I. Varga, "Development of road traffic control by using integrated vissim-matlab simulation environment," Periodica Polytechnica, vol. 56, no. 1, pp. 43–49, 2012.

[25] PTV Group, "VISSIM 5.10-06 COM Interface Manual," *PTV AG*, pp. 260, 2009.

[26] J. Ploeg, B. T. M. Scheepers, E. Van Nunen, N. Van De Wouw, and H. Nijmeijer, "Design and experimental evaluation of cooperative adaptive cruise control," *Conf. Intell. Transp. Syst.*, pp. 260–265, 2011.

[27] D. Swaroop and K. R. Rajagopal, "A review of constant time headway policy for automatic vehicle following," *ITSC 2001. 2001 IEEE Intell. Transp. Syst. Proc. (Cat. No.01TH8585)*, pp. 65–69, 2001.

[28] Comer D. E., Stevens, D. L., 2001. Internetworking with TCP/IP Vol III: Client-Server Programming And Applications Linux/POSIX Sockets Version. Prentice Hall, New Jersey.

[29] Stone, J., Stewart, R. R., and Otis, D. "Stream Control Transmission Protocol (SCTP) implementations, and techniques. New topics include:Checksum Change," RFC 3309, 2002.

[30] PTV Group, "PTV VISSIM 7 User Manual," PTV AG, pp. 240, 2013.

[31] A. Mihaly, “Look-ahead cruise control design in VISSIM simulation environment,” Models and Technologies for Intelligent Transportation Systems,. June, pp. 3–5, 2015.

[32] M. Aycin and R. Benekohal, “Comparison of Car-Following Models for Simulation,” *Transp. Res. Rec.*, vol. 1678, no. 1, pp. 116–127, 1999.

Appendix A

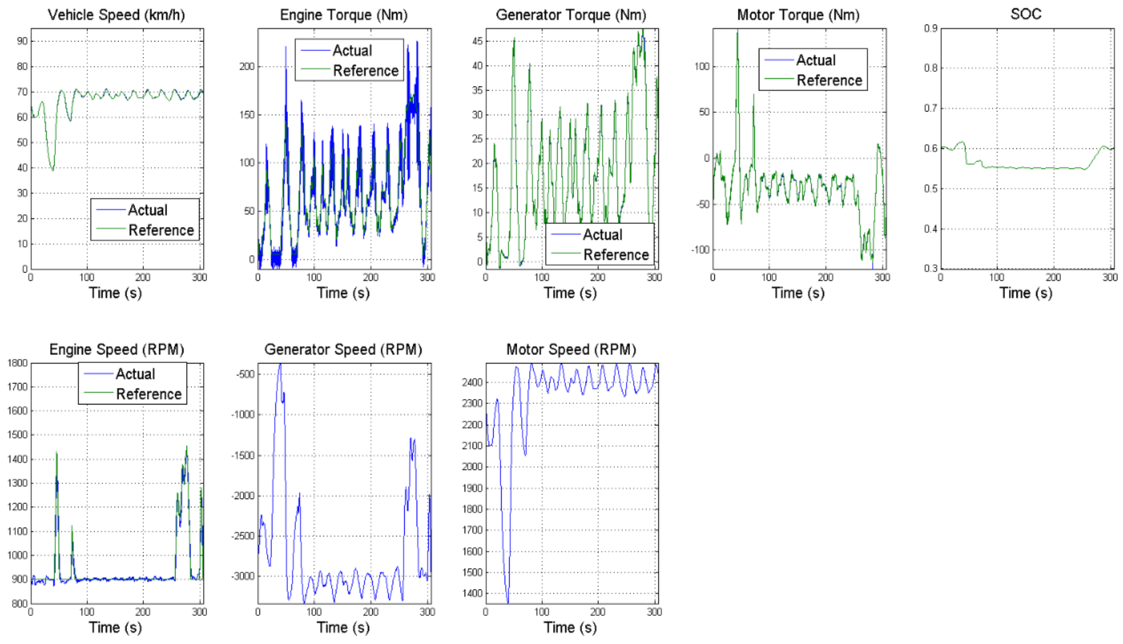


Figure A1.Vehicle Dynamics of Controlled Vehicle- highway driving

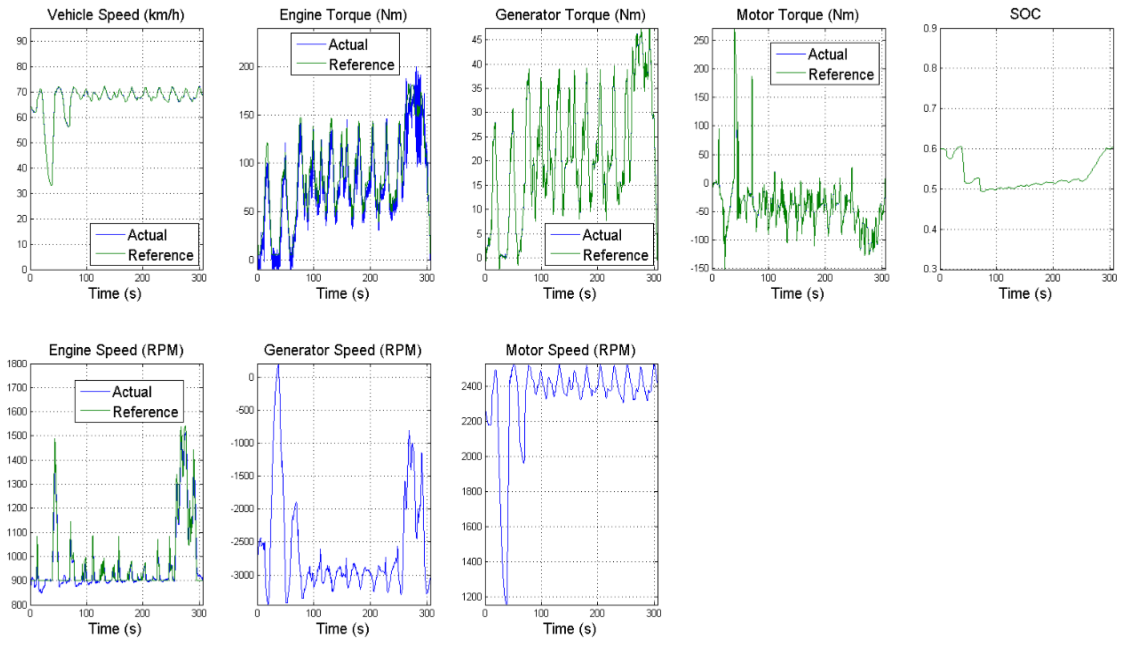


Figure A2.Vehicle Dynamics of Preceding Vehicle- highway driving

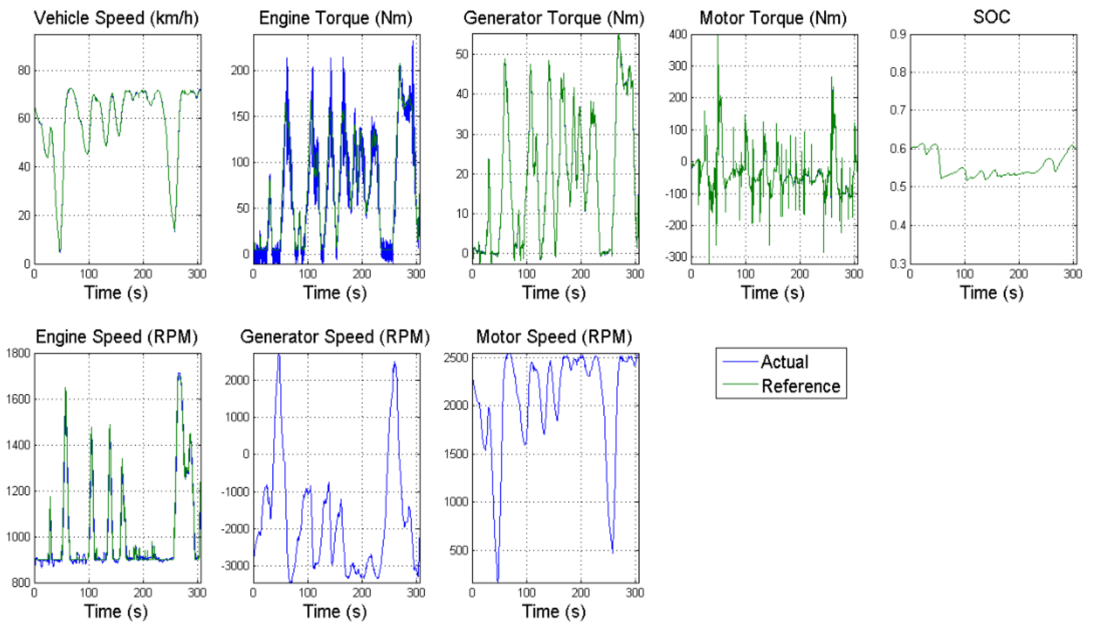


Figure A3.Vehicle Dynamics of Controlled Vehicle- city driving

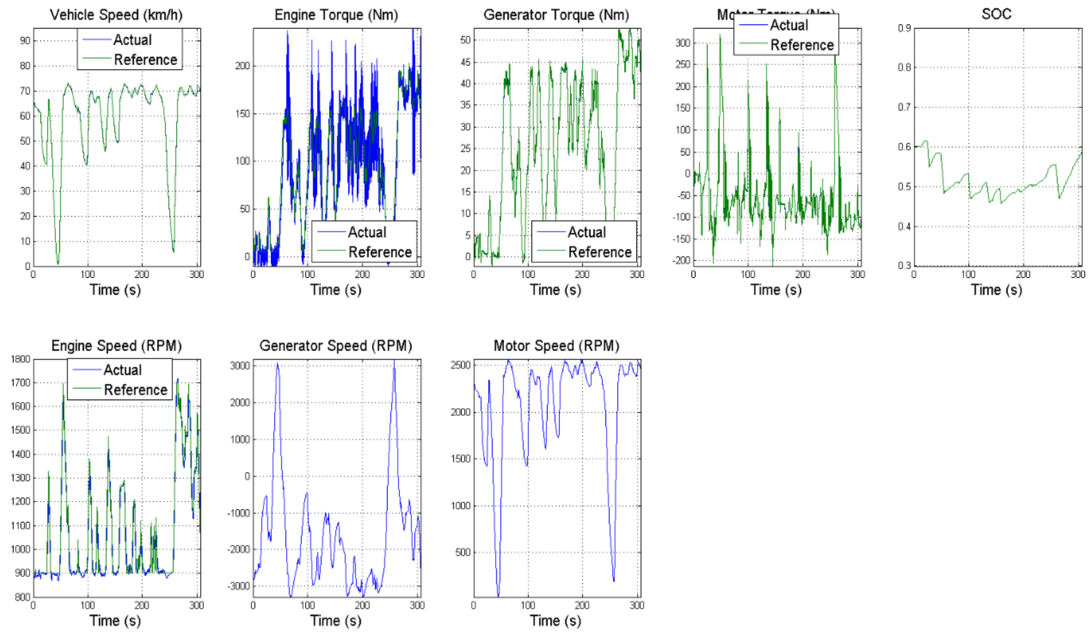


Figure A4.Vehicle Dynamics of Preceding Vehicle- city driving

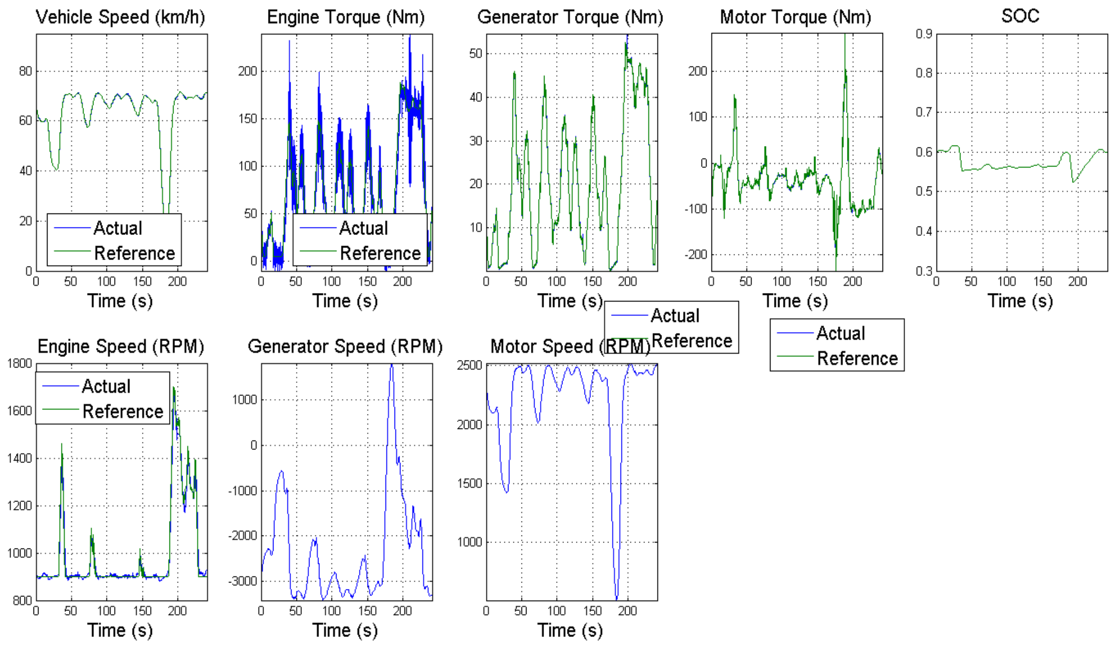


Figure A5.Vehicle Dynamics of Controlled Vehicle using real-time CACC- Varying Platoon size

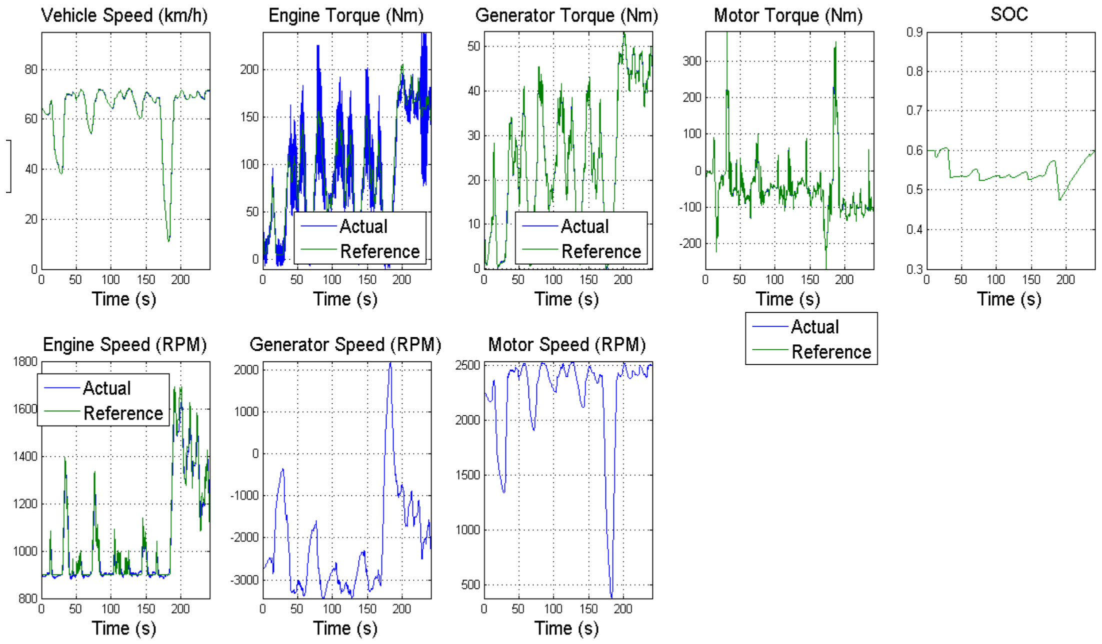


Figure A6. Vehicle Dynamics of Preceding Vehicle using real-time CACC-Varying Platoon size

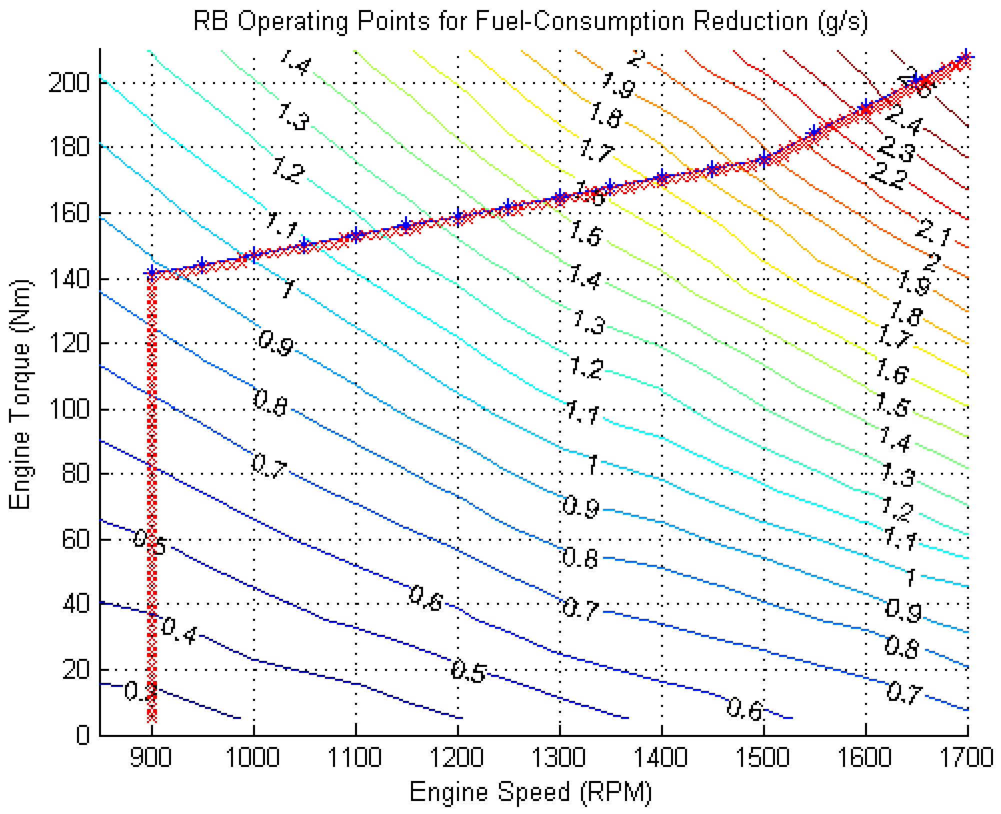


Figure A9.Fuel Map