Comparison of Model-Predictive control and PID control for Adaptive Cruise Control of UW EcoCAR vehicle

Yug Mukesh Jain

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

Master of Science in Mechanical Engineering

University of Washington

2020

Committee:

Brian Fabien

Per Reinhall

Robert Darling

Program Authorized to Offer Degree:

Mechanical Engineering

University of Washington

Abstract

Comparison of Model-Predictive control and PID control for Adaptive Cruise Control of UW EcoCAR vehicle

Yug Mukesh Jain

Chair of the Supervisory Committee:

Brian Fabien

Department of Mechanical Engineering

This work evaluates two control strategies for Adaptive Cruise Control (ACC), classical control (PID control), and Model Predictive Control (MPC) with linear-piecewise approximated engine fuel map as a part of cost function to penalize fuel consumption, both applied to UW EcoCAR vehicle model. The ACC system with MPC consists of hierarchical control architecture, a lower-level controller to track the acceleration command, and a higher-level ACC control. The control algorithms are tested on Model-In-Loop (MIL) using Simulink. The real-time Hardware-In-Loop (HIL) performance testing is done using the dSpace simulator which runs the vehicle model and the dSpace MicroAutoBox II which serves as a controller platform. A comparison of miles per gallon of fuel, average acceleration, and average jerk is provided for drive cycle runs by ACC PID and MPC.

Table of contents:

Acknowledgment

The UW EcoCAR lab has helped me develop important technical and interpersonal skills. The lab has provided equipment that was important for the successful completion of this thesis work. I would also like to thank lab members, who supported and helped me out whenever I faced any barrier.

It was a great pleasure to have Professor Brian Fabien as an advisor for my Master's thesis. He used to meet every week to solve any doubts while helping in overcoming the roadblocks that arise in the thesis. I would like to thank the thesis committee, Professor Per Reinhall, and Professor Robert Darling for accepting my request. I would like to thank Yudong Lin for helping me out in HIL set-up, Carlos Sama for providing his knowledge about the plant model of the system, Aman Kalia for guiding me on literature review, and Livia Anderson for her work to develop a lab access plan so that we could access the lab during the times of COVID.

Finally, I would like to thank my parents for the constant support that they have provided during my academic journey.

Dedication

to my mother Bharti and my father Mukesh

List of figures

List of tables

1. Introduction

Adaptive Cruise control is a driver assistance system that enables a vehicle to automatically control its longitudinal motion depending on the traffic conditions without the driver having to apply accelerator or brakes. This feature has been shown to reduce the driver workload and increase the driver's environmental awareness under certain circumstances [6]. This work compares two types of control strategies for ACC applied to the UW EcoCAR vehicle model.

The University of Washington EcoCAR 4 team is a participant in the GM-sponsored EcoCAR 4 competition aimed to convert a 2019 Chevrolet Blazer into a fully functional, electric-hybrid vehicle. The UW team is focused primarily on finalizing designs and integrating them within the vehicle. The team composes of three technical swimlanes or sub-teams namely, Propulsion Systems Integration (PSI), Connected and Automated Vehicles (CAV), and Controls and Systems Modeling Simulation (CSMS). The controls and system modeling swimlane is responsible for creating a mathematical model of the vehicle and using this model to design various vehicle control features like Hybrid Supervisory Control, Power-split controls, Adaptive Cruise Control.

Abdrakhmanov and Adouane, [7] developed a global optimization problem of energy-spilt over a predetermined path of a bus to be used as public transport and created an online database to be used as a sub-optimal solution in real-time for ACC. They found that the online sub-optimal control corresponds to the best 10% solutions of all the global optimization solutions in the database. But, for this solution, a predetermined path is required which is usually only applicable to public transport.

Park et. al., [8] used an adaptive LQR where LQR parameters are adaptively adjusted depending on the direction of acceleration of the lead vehicle and the relative distance between the host and lead vehicle. This approach uses weighting factors to maintain a safe distance and improve fuel economy by penalizing the relative distance, relative velocity, and acceleration command. But, this method does not put a constraint on collision avoidance. Also, it uses the acceleration command as a proxy parameter to penalize fuel consumption.

Schmied et. al., [9] implemented an MPC based Predictive ACC. Here, the cost function is the fuel map based on the acceleration command and the velocity of the host vehicle. This fuel map is derived from the engine fuel map for a single selected transmission gear. So, this entire work is based on a fixed transmission gear. This fuel map is non-linear and to use it in a linear MPC framework the map is piecewise-linearly approximated. Other constraint equations are used to incorporate other factors like driver comfort (acceleration and change in acceleration constraint), drivability, and safety (distance tracking constraint). This work has been shown to improve the fuel economy of the host vehicle as compared to the lead vehicle on a fixed gear.

Li et. al., [10] implements MPC based ACC with aims to tackle tracking capability, fuel economy, and driver desired response. They have designed a lower-level feedback controller and inverse dynamics to track the commanded acceleration. A higher-level MPC based controller has been designed to achieve the above-mentioned aims. The cost function consists of a 2-norm of velocity tracking error, distance tracking error, acceleration request, and corresponding jerk request. This controller was shown to perform better than LQR based ACC

on fuel consumption, safety, and tracking metrics. Here, the acceleration request and the jerk request are used as proxy parameters for penalizing fuel consumption.

Takahama et. al., [11] uses an Adaptive MPC to build an ACC. The main aim of this paper was to develop a computationally efficient MPC and improve vehicle response in stop-and-go situations. For improving the computational efficiency a lower order vehicle model is used. To improve stop-and-go response adaptive MPC weighting parameters are switched in real-time based on relative distance error and relative velocity states. The cost function penalizes relative distance error, relative velocity, acceleration command, and acceleration change. This work improved the response of the vehicle in stop-and-go and high-speed situations.

This work focuses on the Adaptive Cruise Control (ACC) algorithm keeping in mind key goals to develop a vehicle with good fuel economy and driver comfort. Various modifications are made to the vehicle like switching to a smaller engine and adding a motor to convert the vehicle to P4-hybrid. On the CSMS side algorithms like ECMS is being implemented to achieve optimal power-split between motors and engine so, lesser fuel is consumed. With this improved fuel economy goal, ACC algorithms in this work are compared on various drive cycles. The results of this work will be helpful to choose between the algorithms being compared. Other ACC performance factors related to driver comfort are also described in the Controller design section.

This writing is organized as follows. Section 2 describes the vehicle model used and the two controllers' design process. Section 3 describes the model-in-loop (MIL) runs and comparison of results. Section 4 describes the hardware-in-loop (HIL) setup, and results. Section 5 provides a conclusion and future work.

This work uses MATLAB and Simulink for plant modeling and controller design. MATLAB is a scripting language for scientific computing by MathWorks and Simulink is a dynamic system model simulation tool developed by MathWorks. The Simulink is also used for MIL simulations and auto-code generation. dSpace MicroAutoBox II (MABx) is used as a controller platform and dSpace Simulator for real-time plant simulation for HIL runs. Vector CANdb++ is used to create a dbc file for CAN messages transmitted between Simulator and MABx. CVXGEN quadratic solver library is used for the fast computation of MPC.

2. Controller Design

2.1 Description of the plant model

The UW EcoCAR vehicle is a modified GM Blazer. The main goal set by the team, that has driven the vehicle modification design decisions, is improved fuel economy [12]. The team has chosen P4 split-parallel hybrid powertrain architecture [12]. In this kind of architecture, the engine and motors are not directly connected thus, each of the power sources drives separate sets of wheels as shown in Fig. (1). In UW EcoCAR's case, the motors drive the rear wheels and the engine drives the front wheels. The vehicle model has been developed by MathWorks on Simulink.

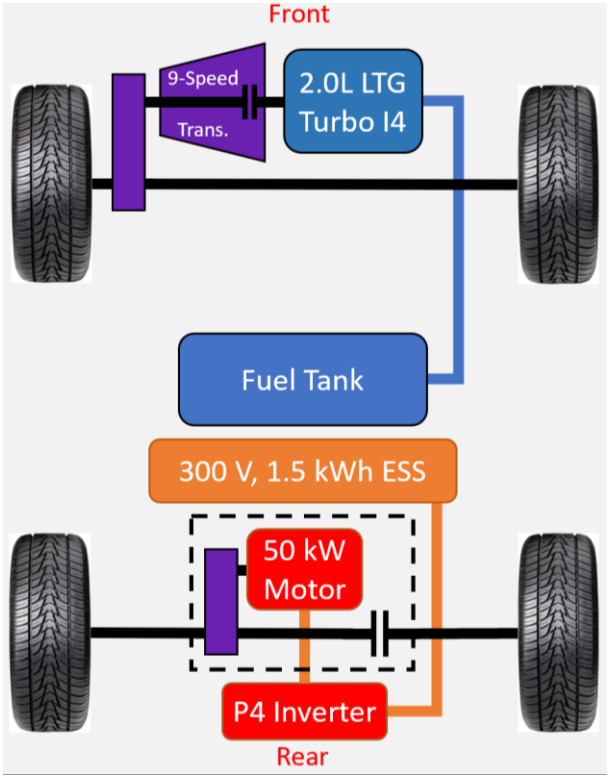

Figure 1: P4 split-parallel hybrid[12]

The controller design in this work is based upon this model with some modifications and additions. This thesis categorizes the blocks in the model into two parts depending on where they will be running in HIL. One part consists of the blocks that run on MABx and the other part that runs on the Simulator.

The higher-level blocks developed by MathWorks and used in this thesis are the UW Blazer model and Controller. The other blocks that were developed for this thesis will be described in later sub-sections. The UW Blazer model consists of the following sub-systems:-

SI Engine: This block simulates the engine. It is provided with various engine maps such as the Fuel rate-Torque-Speed map, efficiency map, etc. It also has first-order engine throttle dynamics and load dynamics modeled in it.

Electric power train block: This block contains a motor model with its maps like torque-speed envelope and efficiency map. This block also contains a Lithium-Ion Battery Pack model.

Drivetrain and vehicle dynamics: This block contains a transmission model and full-road load longitudinal vehicle dynamics model. Tire mechanics are not included for the simplicity of the model. So, it is assumed that the wheel strongly grips the road i.e. wheel slip is zero.

The ACC sends a wheel torque request to the power-split strategy controller i.e. ECMS which communicates with various controllers like Engine Control Module, Transmission Control Module, Body Control Module, Battery Management system, Motor Controller, and Electronic Brake Control Module. This produces the desired output as requested by ACC.

2.2 Control Architecture

ACC controller consists of two parts, namely, a higher-level control and lower-level control. The lower-level controller is responsible for tracking the acceleration requested by the higher-level controller. The lower-level controller outputs a wheel torque request which is taken as input by ECMS which then decides which actuators (Engine, Motors, and Brakes) to engage and by what amount to meet the wheel torque request by the lower-level controller. Fig(2) shows the ACC controller architecture. The ECMS control and vehicle model are combined in the plant model as this thesis focuses just on the development of ACC algorithms for which ECMS can be considered as a part of the plant. Note that even though the ECMS is shown as a part of the plant in the control architecture, in the HIL implementation the ECMS will be part of the MABx controller which will also be the case in the final vehicle.

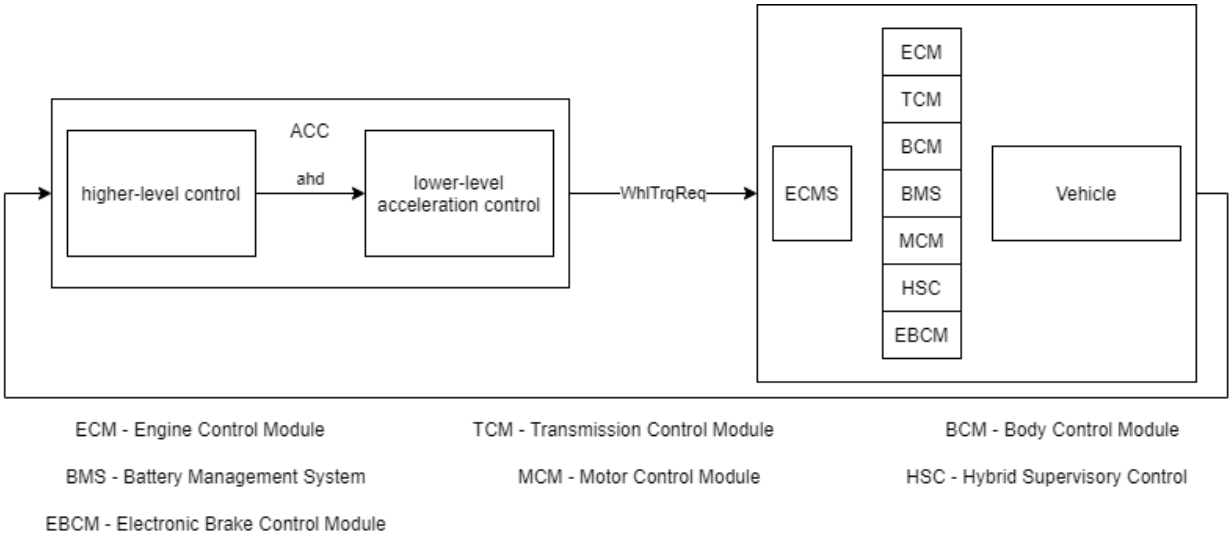

Figure 2: MPC Control Architecture

2.3 Description of performance metrics, tests, and operational limits for ACC

Basic Functional Test scenarios:
Tracking performance of ACC will be evaluated with a test scenario [1]. The result of the test qualitatively shows the capability of the ACC to track the lead vehicle and maintain a safe distance.

Performance for drive cycle run:
The driver comfort is measured using average acceleration and average jerk values in ISO standards [3]. The fuel economy is compared using Miles per gallon (MPG) of fuel consumed. Table 1. describes the expected performance.

Metric	Limits
Average jerk over 1 sec	2.5
Average deceleration over 2 sec	3.5
Acceleration	2
Distance tracking error (d_del)	>= 0

Table 1:Performance metrics

Operational limits:
Velocity range of ACC operation is 0 m/sec to 40 m/sec. The upper limit of 40 m/sec is set by considering highway driving speeds.

2.4 Lower-level acceleration tracking

The vehicle longitudinal model is highly non-linear. Non-linearities arise from the non-linear variation of road load with the velocity of the vehicle. Since, the goal is to use a linear-MPC for computational efficiency and ease of tuning, the vehicle model used in MPC is a simple linear kinematic model. To use this MPC for ACC there needs to be an intermediate controller that acts on the plant such that from MPC's perspective the plant behaves linearly. For this reason, a lower-level acceleration controller is designed.

This controller constitutes of two parts, namely, a reference feedforward control and feedback control. The feedforward control uses the actual non-linear road load equation to calculate the desired wheel torque. The feedback controller is a gain-scheduled PI-control used to improve the acceleration time-response of the vehicle. Figure 3. Shows the flow of signals in this controller.

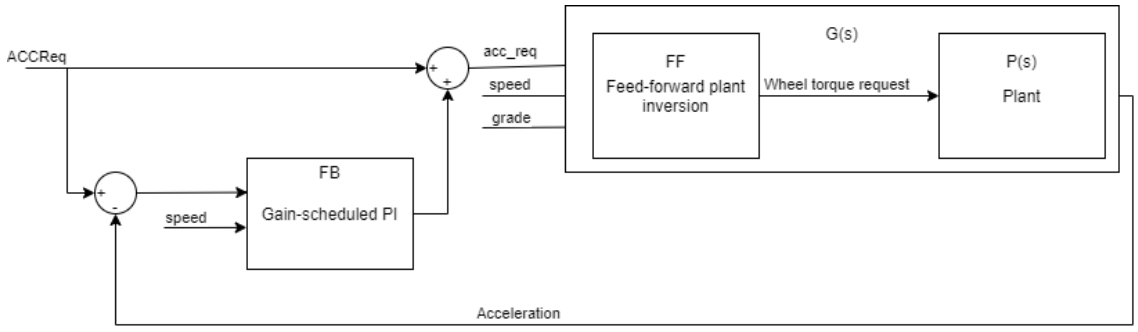

Figure 3: Lower-level controller structure

2.4.1 Design of feedforward plant-inversion control

The road load equation of the vehicle gives the total traction force required to accelerate the vehicle at the desired acceleration while accounting for all the resistances. There 4 types of resistances that the powertrain of a vehicle needs to overcome to accelerate it with the desired acceleration. These are as follows:

1. The inertia of vehicle: The mass of the vehicle causes resistance to change in motion of the vehicle which is given by Newton's first law. And this can be quantified by Newton's second law as, Fi = m.a, 'm' is the mass of the vehicle, 'a' is the desired acceleration and 'Fi' is the inertial resistance.

2. Rolling resistance: Tires of the vehicle have some resistance to rolling caused due to its contact patch which is formed by deformation of the tire under load. For purpose of simplicity, this rolling resistance is assumed to be a constant. 'A' is the rolling resistance in the UW EcoCAR model.

3. Driveline resistance: This is a resistance caused by the inertia of driveline components and losses in the driveline components. The driveline resistance model is usually boiled down to a linear dependency on the velocity of the vehicle. 'B' is the coefficient of driveline resistance in the UW EcoCAR model. Driveline resistance = B.v where, v =velocity of the vehicle.

4. Aerodynamic resistance: The motion of the vehicle is also resisted by the air surrounding the vehicle. This is caused because of aerodynamic effects like skin friction drag, form drag. These effects can be encapsulated in the aerodynamic drag coefficient 'C'. The aerodynamic resistance is C.v^2 thus, this resistance increases quadratically with vehicle velocity.

5. Road grade: The grade of the road causes the weight of the vehicle to have a component parallel to the longitudinal direction. Thus, this grade causes a force that may resist or support the motion of the vehicle. This is quantified by, Fgrade = Mass.g.sin(grade), where, 'g' is the acceleration due to gravity and grade is in radians.

6. Following is the complete road load equation used in reference feedforward control:
   $F_{road\_load} = Mass.a + A + B.v + C.v^2 + Mass.g.\sin(\theta)$

For the feedback control design in Section 2.4.2, the plant model is the feedforward control combined with the plant model denoted as G(s) in Fig. 3.

2.4.2 Design of feedback control loop

The powertrain of the vehicle has internal dynamics. The effect of these dynamics is carried on to the acceleration of the vehicle. The acceleration of the vehicle can be approximated as a second-order system using system identification. The feedback control loop is a gain-scheduled PI-controller used to improve the step response of the acceleration of the vehicle. The acceleration request from the ACC controller acts as a reference to this controller. The vehicle model and the reference feedforward control combined consist of a plant model for feedback control design. This combined plant is represented by G(s) in Fig. 3.

The UW EcoCAR vehicle model is used to collect acceleration data. This data is used in the MATLAB System Identification toolbox to fit a second-order model. Then, this model is used to tune PI-control using the MATLAB control design toolbox.

2.4.3 System Identification

System Identification (SysID) is used to get a mathematical model of a system from the input-output data collected from that system. MATLAB System Identification Toolbox is widely used for this purpose. Fig. 4 shows the SysID toolbox window for acceleration second-order model. The workflow of System Identification is as follows: Import input-output data, preprocess if required, select the model to fit the data and, estimate the model parameters.

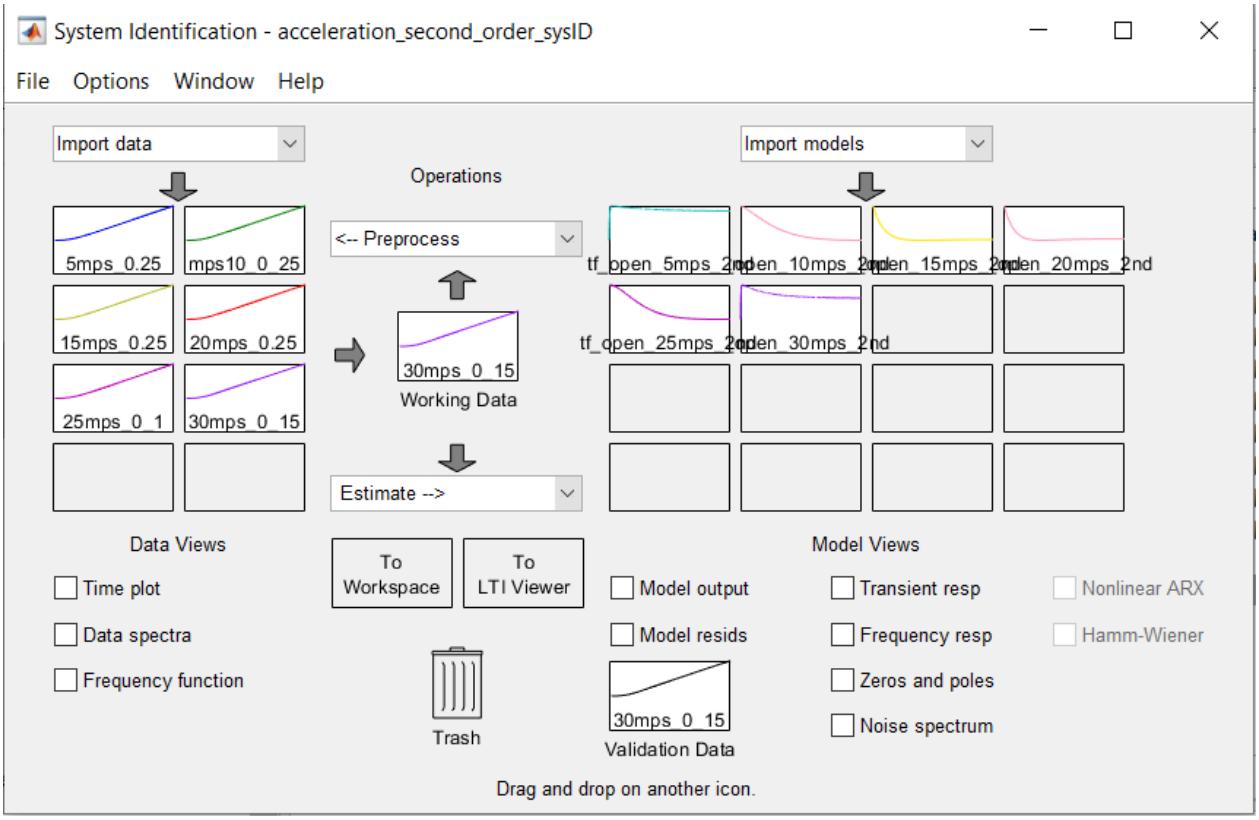

Figure 4: MATLAB System Identification Toolbox for estimating 2nd order transfer function

Before performing system identification appropriate data must be collected from the system, which in this case is the UW EcoCAR vehicle model. The acceleration response of the vehicle is limited by the actuators i.e. powertrain of the vehicle. And the output of the powertrain is velocity-dependent. So, the maximum achievable acceleration is dependent on the velocity of the vehicle. So, the vehicle model is run with maximum torque request from the powertrain at six equally spaced velocity values from 5 m/sec to 30 m/sec. So, we get maximum acceleration achievable in 0.1 sec at each velocity value as shown in Table 2. The reason for selecting 0.1 sec as the time for recording the acceleration is that the higher-level controller has a control

rate of 10 Hz (0.1 sec), so it is expected that the vehicle attains the commanded acceleration within 0.1 sec.

Velocity	Acceleration at 0.1 sec
5	0.45
10	0.4
15	0.37
20	0.28
25	0.13
30	0.18

Table 2: Maximum acceleration at 0.1 sec

The MPC designed for ACC has a constraint for acceleration change of 0.25 m/sec in 0.1 sec to satisfy the jerk constraint mentioned in section 2.3. So, data will be collected for 0.25 step acceleration response for velocities with a maximum acceleration greater than 0.25 in 0.1 sec. For velocities with maximum acceleration under 0.25, a value lower than their corresponding maximum values will be used as a step value to get the step response. Table 3 shows the step values of acceleration reference used for each response and Fig. 5 shows the data and corresponding 2nd order transfer function fit.

Velocity	Acceleration at 0.1 sec
5	0.25
10	0.25
15	0.25
20	0.25
25	0.1
30	0.15

Table 3: Step value of reference for acceleration response

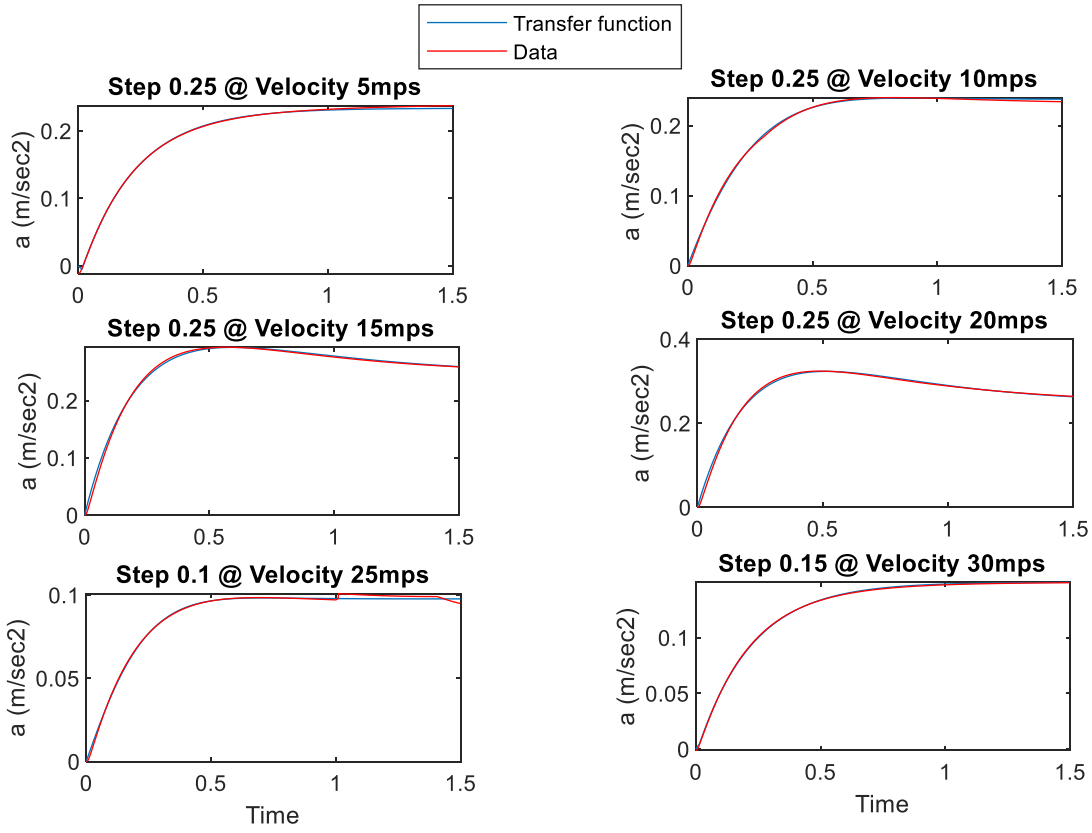

Figure 5: Acceleration response and system identification

2.4.4 Gain-scheduled PI-control design

To improve the acceleration response time over the velocity range of ACC operation a gain-scheduled PI-control is used. Gain scheduling simply means that the controller gains are adjusted in real-time according to the parameter that causes the plant model to change. In our case, the plant model i.e. the acceleration response changes with the velocity of the vehicle as shown in Section 2.4.2.1. A PI-control is tuned for each of the six transfer functions obtained in system identification from Section 2.4.2.1.

MATLAB PI-tuner used to tune the PI-control to achieve commanded acceleration within 0.1 sec. The gains obtained from PI tuning are then imported in the look-up table which adjusts the gains depending on the vehicle velocity using linear-point slope interpolation. Table 4 gives the PI gains obtained after tuning. Fig. 6 shows the comparison of acceleration response with and without feedback control.

Velocity	Kp	Ki
5	4.12702	27.84076
10	3.546919	41.65982
15	2.130638	31.0369
20	1.685717	27.32741
25	3.974324	44.60951
30	3.459197	39.7684

Table 4: Gains table for gain scheduled PI

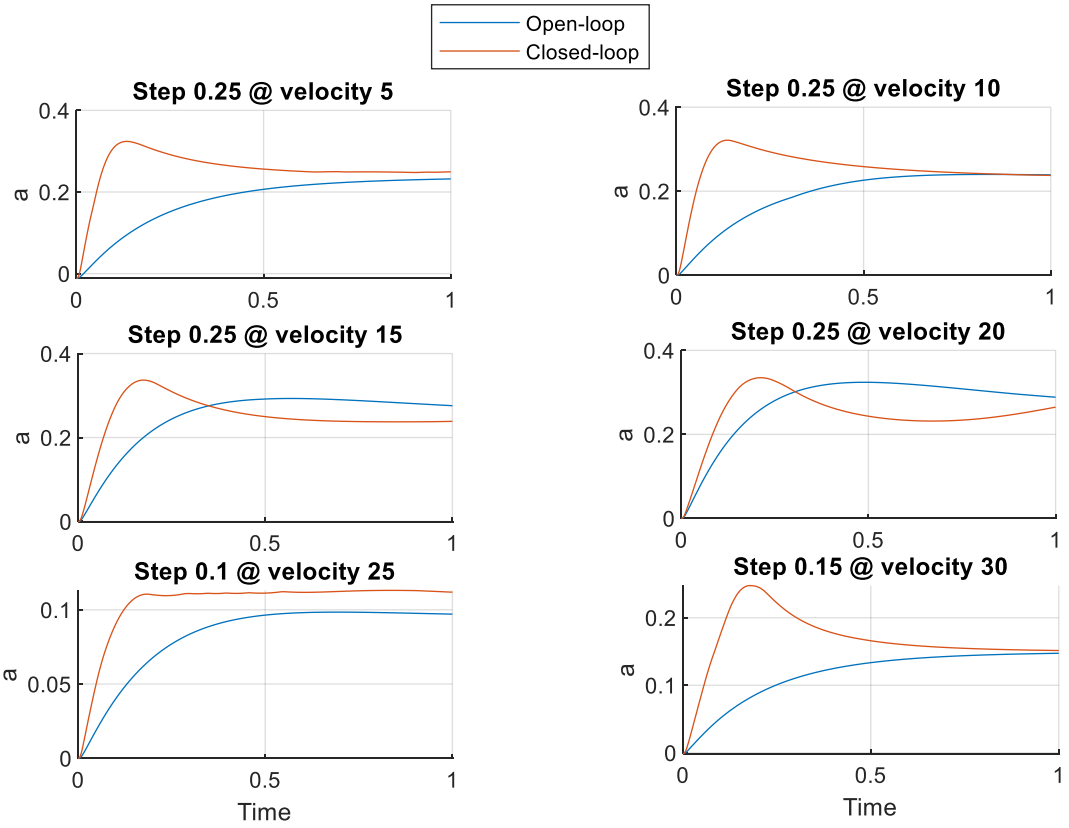

Figure 6: Open-loop and closed-loop acceleration response

2.5 Model Predictive Control for ACC

2.5.1 Model Predictive Control (MPC)

MPC is an online optimization-based control method. MPC comprises the optimization cost function, the plant model of the plant to be controller, and constraints over the plant states and control input. MPC solves a finite-time horizon optimal problem at each sampling time with the initial state as the most recently sampled or estimated state. The solution to the optimization problem is a sequence of control inputs. Only the first control value is applied to the plant.

The plant model describes the evolution of the states of the system. Plant models can be produced using various techniques like collecting the data from a physical plant by performing input-output tests then, using system identification methods to get the model, first-principles modeling of plant, or a combination of both the methods. Physical systems are usually continuous in time and thus, modeled by differential equations. The differential equation is

transformed into difference equations for the computation of optimal control. The difference equation plant model is:

$$x^+ = f(x,u) \quad f: \mathrm{X} \times U \to X$$

Here, $x \in X \subseteq R^n$ is the current state, $u \in U \subseteq R^m$ is the current control and, x^+ is the successive state.

The cost function which is to be minimized is a function of state and control over the entire finite-time horizon.

$$J = \left(\sum_{k=i}^{i+N-1} l\big(x(k), u(k)\big) \right) + V_f(x(N+i))$$

Here, l is stage cost, V_f cost on final state at time-horizon, i is the current time, k is time starting from i up to the time horizon, and N is the time-horizon.

MPC also can accommodate state and control constraints.

$$(x(k), u(k)) \in \mathbb{C}$$

Here, $\mathbb{C}$ is a set of state and control constraints.

Using the above expressions, the MPC problem can be stated as,

$$\min_{\hat{u}} J\,(\hat{x}, \hat{u})$$

$subject\ to,$

$x^+ = f(x,u)$

$(x(k), u(k)) \in \mathbb{C}$ for all $k \in \mathbb{Z}_{k \geq 0}$

MPC's capability of handling constraints makes it an attractive control method for applications like Adaptive cruise control. Unlike MPC, control methods like PID do not have constraints implicit to their formulation.

2.5.2 ACC state-space model

A prerequisite to designing a controller is the model of the plant. The main goal of ACC is to track the distance to the lead vehicle and track the lead vehicle velocity with minimum error. Also, the operation of ACC should be such that a chain of ACC vehicles following each other, i.e. platoon, is string stable. String stability ensures that the tracking error of the vehicle ahead is not amplified down the chain. For this purpose, a headway time tracking policy will be used as this method is inherently string stable [12].

Before describing the ACC state-space model it is important to mention a few terminologies that will be used further. The host/ego/ACC vehicle is the vehicle under consideration for ACC operation. The lead vehicle is the vehicle immediately ahead of the ACC vehicle which is sensed by the sensors onboard. Fig. 7 shows a pictorial representation of the ACC operation.

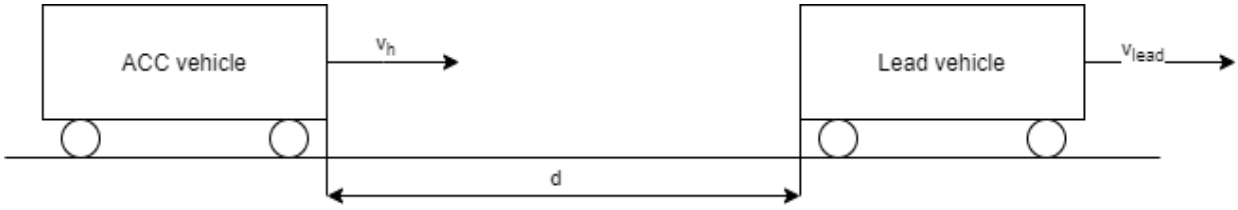

Figure 7: Depiction of ACC operation

In Fig. 7, v_h is host vehicle velocity, v_{lead} is lead vehicle velocity and, d relative distance from the leading edge of the host vehicle to trailing edge of the lead vehicle. A minimum use case of the ACC algorithm is to maintain a safe relative distance and track lead vehicle velocity.

The ACC state-space model is as follows:

$$\dot{x} = \begin{bmatrix} \dot{\Delta d} \\ \dot{v_h} \end{bmatrix} = \begin{bmatrix} 0 & -1 \\ 0 & 0 \end{bmatrix} . \begin{bmatrix} \Delta d \\ v_h \end{bmatrix} + \begin{bmatrix} -\tau_h \\ 1 \end{bmatrix} . a_{hd} + \begin{bmatrix} 1 \\ 0 \end{bmatrix} . v_{lead}$$

$$y = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} . \begin{bmatrix} \Delta d \\ v_h \end{bmatrix}$$

$where, a_{hd} = Acceleration\ command\ \ for\ host\ vehicle$

$\Delta d = Relative\ distance\ tracking\ error = d - d_{ref}$

$d_{ref} = Reference\ relative\ distance = d_0 + \tau_h . v_h$

$\tau_h = Headway\ time, d_0 = Minimum\ distance\ to\ maintain\ at\ stand\ still$

A headway-time policy is used for reference distance as it is inherently string stable[15].

This model when solved for the MPC prediction time step T_{ps} gives the following difference equation-based state-space model.

$$x(k+1) = A_d . x(k) + B_d . u(k) + \overline{B_d} . w(k)$$

$$where, A_d = \begin{bmatrix} 1 & -T_{ps} \\ 0 & 1 \end{bmatrix}$$

$$B_d = \begin{bmatrix} -\tau_h . T_{ps} - \frac{T_{ps}^2}{2} \\ T_{ps} \end{bmatrix}$$

$$\overline{B_d} = \begin{bmatrix} T_{ps} \\ 0 \end{bmatrix}$$

$u = Control\ input = a_{hd}$

$w = Measured/estimated\ disturbance\ input = v_{lead}$

This model is one of the constraints to the MPC optimization problem as mentioned in Section 2.5.1.

2.5.3 Constraints

Apart from the dynamic system model of ACC, other constraints need to be satisfied in the optimization problem. These constraints can be soft or hard depending on the system. For example, the constraints that are applied to the control inputs are usually hard because of actuator limits. While the constraints on the states of the system can be soft which allows some room for constraint violation but, with a huge penalty in the cost function if such violation is to occur. It is better to have a lesser number of hard constraints as it may lead to the infeasibility of the optimization problem.

The hard constraint used in the MPC optimization problem is on the control input and positive control input change i.e. acceleration/deceleration command, increase in acceleration command, and maximum velocity. Limit on maximum deceleration request is directly translated from the ISO performance metric of average deceleration. And limit on maximum acceleration request is directly translated from the ISO performance metric of maximum acceleration and actuator limits. Limit on increase in acceleration command is directly translated from ISO performance metric of maximum jerk allowance.

$Maximum\ acceleration\ constraint, u(k) \leq 2$

$Maximum\ deceleration\ constraint, u(k) \geq -3.5$

$Maximum\ increase\ in\ acceleration\ constraint, u(k+1) - u(k) \leq 0.25$

$Maximum\ velocity, v_h \leq\ v_{set}, v_{set}\ is\ cruise\ velocity\ set\ by\ driver$

Soft constraints are applied to a negative change in acceleration and limit relative distance error. A negative change in acceleration has a soft constraint as opposed to a positive change in acceleration which has a hard constraint because in case of emergency deceleration a high rate of decrease in acceleration is required. Soft constraints are accompanied by a slack variable which adds up as a cost in the cost function.

$Maximum\ decrease\ in\ acceleration\ constraint, u(k+1) - u(k) \leq -0.25 -\ \beta_2$

$Maximum\ relative\ distance\ error, \Delta d(k) \leq 10 + \gamma_1$

$Minimum\ relative\ distance\ error, \Delta d(k) \leq 0 - \gamma_2$

$where, \beta_2(k), \gamma_1(k), \gamma_2(k) \geq 0\ are\ slack\ variable$

2.5.4 Cost function

Apart from tracking another important consideration for ACC performance is fuel consumption relative to manual control by the driver. The cost function for Model Predictive Control is designed to minimize the fuel consumption by directly incorporating the fuel map of the engine as a part of the cost function. The overall cost function constitutes fuel rate and slack variables of soft constraints. Other performance factors, tracking error, and driver comfort (quantifies by acceleration and jerk) are kept within the allowable bounds (Section 2.3) by using constraint equations described in Section 2.5.3. Therefore, the cost function is given by,

$$\sum_{k=0}^{N_s-1} w_1.q(k) + w_2.\beta_2(k) + w_3.\gamma_1(k) + w_4.\gamma_2(k)$$

$where, N_s = \frac{Tp}{T_{ps}} = Number\ of\ time\ steps\ in\ optimization\ problem$

$T_p = Prediction\ time - horizon$

$q(k) = Fuel\ rate\ from\ approximation\ of\ fuel - rate\ map\ at\ time\ step\ k$

2.5.5 Fuel map

Data for engine fuel rate map is made available from the competition. Fuel maps are 3-D data points of fuel injection rate varying with engine rotational speed and torque command. Let's name this map speed-torque-fuel rate map (STFm). This fuel rate map can be converted to a map that estimates fuel rate from vehicle speed and acceleration command. Let's name this map as velocity-acceleration-fuel rate map (VAFm). This section describes equations used to get VAFm from STFm.

Maximum acceleration for each gear is a function of vehicle speed. The road load equation along with the gear ratio can be used to get the maximum acceleration curve at each gear. The maximum torque curve (MTC) data is already available. The engine speed range in MTC can be translated to vehicle speed by,

$VehSpd = EngSpd.\frac{pi}{30}.\frac{\mathrm{R_e}}{\mathrm{FD*N}}$,

$R_e\ is\ the\ wheel\ radius, N\ is\ the\ gear\ ratio, FD\ final\ drive\ ratio$

The maximum acceleration for a given gear and at the range of vehicle speeds can be calculated using (as shown in Fig. 9),

$$MaxAcc = \frac{\frac{EngTrq.N.FD}{R_e} - A - B.VehSpd - C.VehSpd^2}{M},$$

A, B, C *are resistence coefficients described in Section* 2.4.1, M *is the Mass of the vehicle*

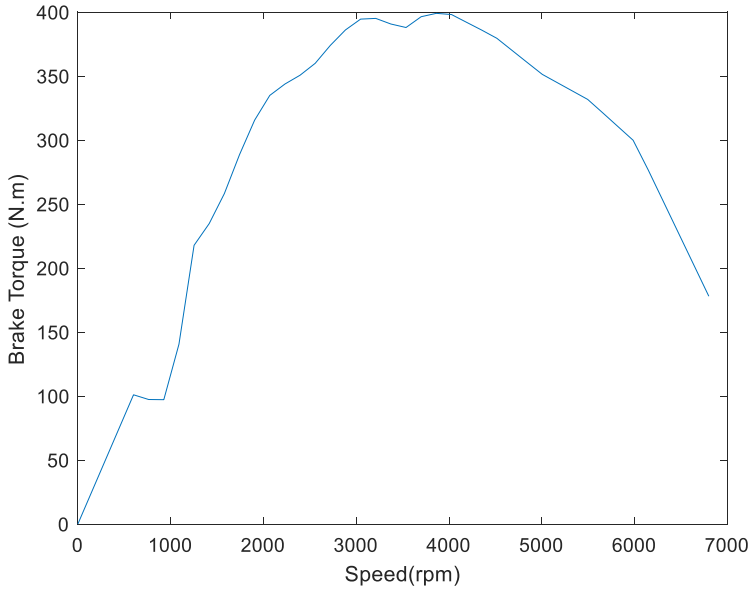

Figure 8: Engine torque curve

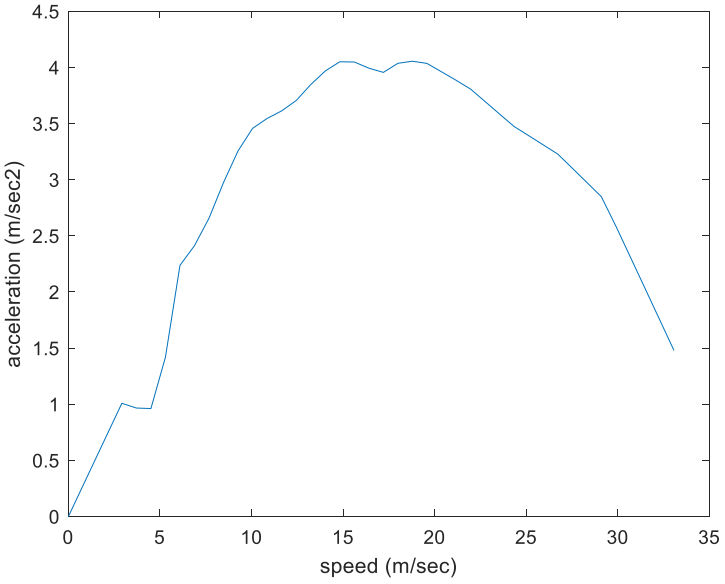

Figure 9: Ideal max acceleration vs vehicle speed curve at gear 4

Now that we have an acceleration curve, we can get VAFm. For automatic transmissions, any gear has a range of velocity of operation. This range can be found for the transmission gear shift map. Let us consider gear 4, the range of velocity of operation is 4.583 - 26.11 m/sec. So, we will find VAFm for a speed range of 4.583 - 26.11 m/sec.

The engine torque request will be calculated from the acceleration request and vehicle speed. Then, this engine torque and engine speed will be used to get the fuel rate from STFm. Thus, this fuel rate corresponds to the acceleration request and vehicle speed.

$$EngTrqReq = \frac{(M.AccReq + A + B.v + C.v^2).Re}{N.FD}$$

The fuel rate value can be interpolated from STFm by using EngTrqReq and EngSpd as the arguments. Fig. 10 shows an example fuel map along with a piece-wise linear approximation of the map with 4 planes.

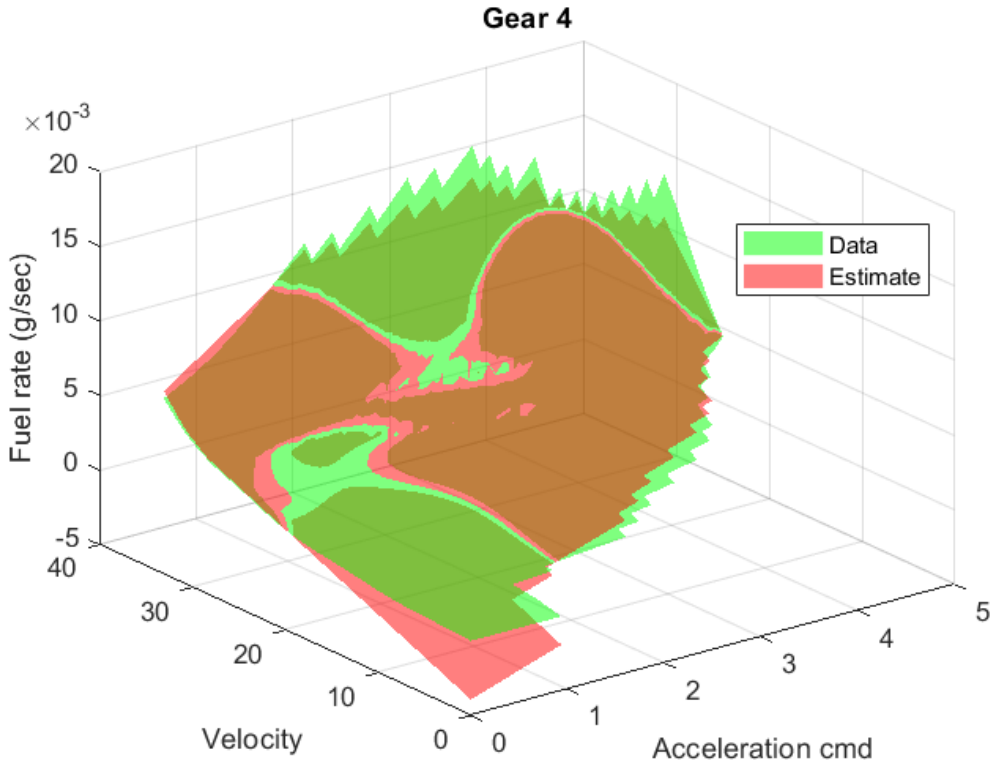

Figure 10: Fuel map data and 4-plane piece-wise affine estimate

2.5.6 Linear approximation of fuel map

Fuel rate is a convex non-linear function of acceleration command and velocity. To incorporate fuel rate in the cost function of the MPC a piece-wise affine approximation of the fuel map will be used.

Consider a convex non-linear function q(x). This function can be approximated by piece-wise max affine function as shown in Fig. 11.

$$q(x) \approx \max\left(a_1.x + b_1, a_2.x + b_2, a_3.x + b_3, a_4.x + b_4\right)$$

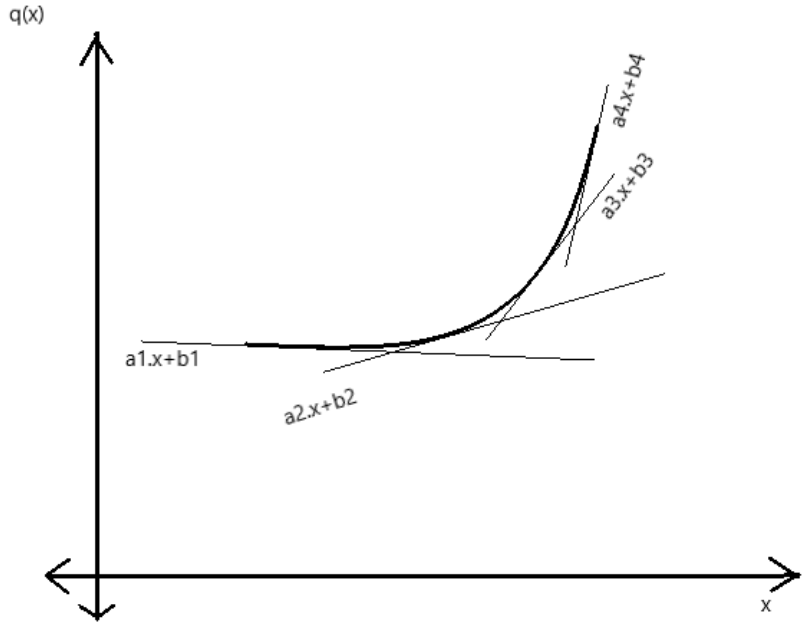

Figure 11: 1D example of piece-wise max affine function approximation of the non-linear convex function

If q(x) is represented by data at discrete points of x, then we can pose an optimization problem to estimate parameters of the set of affine functions such that it minimizes the overall root-mean-squared error of max-affine function from the actual data points[2]. This equation can be incorporated into the optimization objective by considering 'q' as the part of the cost function and adding constraints,

$$q \geq a_i.x + b_i, \qquad i = 1,2,3,4$$

The same technique can be used for higher-dimensional functions like fuel rate maps. Fig. 10 shows an example of the data and the corresponding 4 affine functions fitted to that data.

2.5.7 ACC MPC formulation

A complete ACC MPC optimization problem can be described as follows,

$$\min_u \sum_{k=0}^{N_s-1} w_1.q(k) + w_2.\beta_2 + w_3.\gamma_1 + w_4.\gamma_2$$

$subject\ to,$

$$x(k+1) = A_d.x(k) + B_d.u(k) + \overline{B_d}.w(k)$$

$$u(k) \leq 2$$

$$u(k) \geq -3.5$$

$$u(k+1) - u(k) \leq 0.25$$

$$x_2(k) \leq v_{set}$$

$$u(k+1) - u(k) \leq -0.25 - \beta_2$$

$$x_1(k) \leq 10 + \gamma_1$$

$$x_1(k) \leq -\gamma_2$$

$$\beta_2(k), \gamma_1(k), \gamma_2(k) \geq 0$$

$$q(k) \geq a_i . x_2(k) + b_i . u(k) + c_2, \qquad i = 1,2,3,4$$

$$q(k) \geq 0$$

2.5.8 MPC solver

For the real-time implementation of the optimization, the solver needs to be very efficient in solving the problem within the control step time to avoid controller overrun and potential instability. For this purpose, an off-the-self convex optimization CVXGEN[4] is used. From the HIL tests, it is found that with an MPC prediction time horizon of 10 sec, control step of 0.5 sec, and control update time of 0.1 sec the computation time is close to 0.003 sec which is 33.33 faster than the control update time.

2.6 ACC PID controller design

PID is a widely used control technique in the industry due to its simplicity of tuning and robustness. PID stands for proportional-integral-derivative control. The proportional gain improves the time response of the system. The integral gain reduces steady-state error. And the derivative gain helps in reducing the overshoot.

For tuning a PID control for ACC operation, a linear model of the plant is required. We will use the acceleration models from Section 2.4.3 to create multiple plant models varying with velocity. Then the PID will be tuned for each model on the Simulink PID tuner. Because the controller will be implemented as a discrete PID, the bandwidth of control is set at 9 rad/sec which is approximately 1/70th [13] of the control rate i.e. 628.3185rad/sec (i.e. control step time of 0.01 sec). And the robustness will be set to maximum to minimize the overshoot. This will give a set of tuned PID values for various velocity-dependent plant models. Then these parameters will be used for gain- scheduled PID for ACC operation. Since PID is not computationally heavy, a higher control rate of 100 Hz can be used. Fig. 12. show the closed-loop PID control, and plant model used for tuning PID. G(s) in Fig.12 is the 2nd order acceleration model obtained in Section 2.4.3 using system

identification.

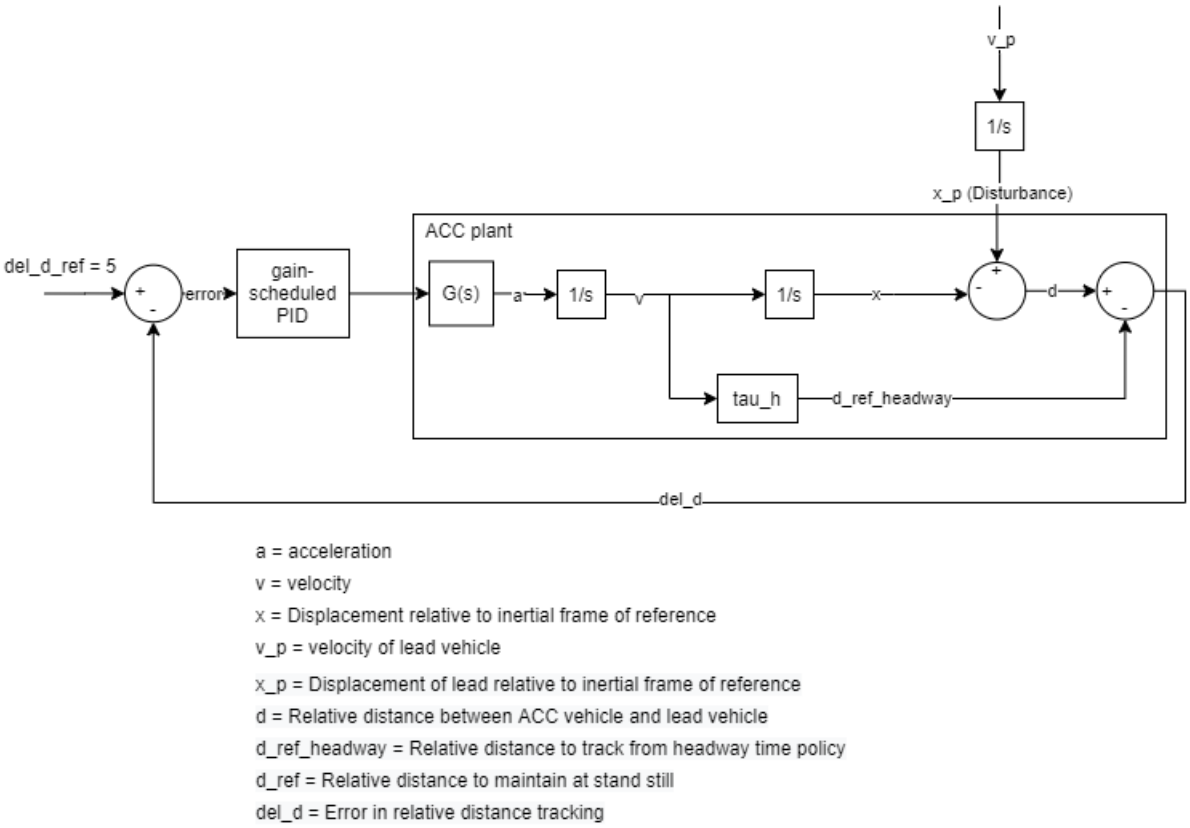

Figure 12: PID ACC control diagram

The plant model is dependent on velocity (since G(s) is dependent on velocity). So, the PID control is tuned for multiple values of velocity (same as in Section 2.4.3). This gives a lookup table for P, I, and D values which are used for gain-scheduled PID control. Table 5 shows all the PID values in the lookup table.

Speed	Kp	Ki	Kd
5	-3.024	-0.4356	-1.6405
10	-3.709	-0.5454	-1.5664
15	-4.125	-3.4134	-0.9649
20	-3.588	-3.044	-0.8169
25	-3.658	-0.5457	-1.2269
30	-4.425	-0.6595	-1.5099

Table 5: Tuned Kp, Ki, Kd values for gain-scheduling

3. Model-in-Loop Simulation results

To test the controllers on the plant model, Simulink Model-in-Loop (MIL) simulation is used. A test to visualize tracking performance is performed in MIL. Then, we also get the performance metrics by running tests on two drive cycles.

3.1 Lead vehicle tracking test

This test shows the ACC's capability to successfully track the lead vehicle. This test begins with the velocity of the lead vehicle maintained at 30 m/sec and the ACC vehicle following at about the same speed. At 40 seconds the lead vehicle starts decelerating at 0.25 m/sec^2 for 20 seconds. The deceleration event ends at 60 seconds. ISO requirements are satisfied by both MPC and PID ACC as seen from plot 2 and plot 3 in Fig. 13. The performance of both MPC and PID control is identical for the tracking test.

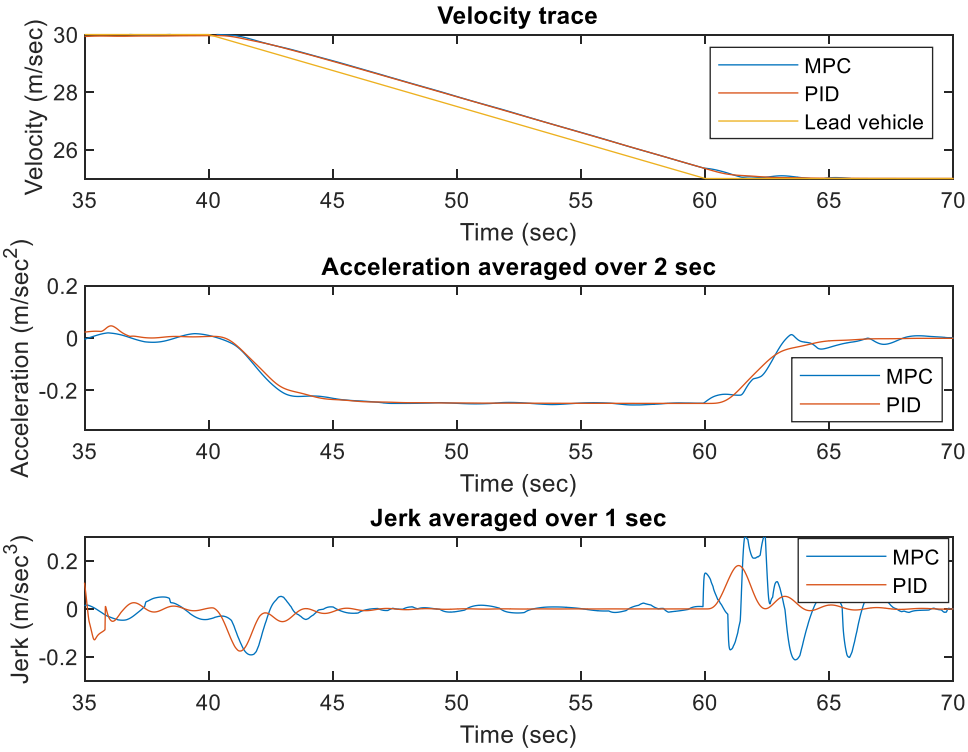

Figure 13: Tracking capability test

3.2 Drive cycle run

The ACC methods are tested on two drive cycles, namely, FTP and BVE (Baseline Vehicle Evaluation). FTP is a city drive cycle. Since the target market of the vehicle will be the Greater Seattle area so, FTP will be a good test cycle. The BVE drive cycle was produced by the team as a part of the Baseline Evaluation Report for the competition. This drive data was gathered by running the unmodified vehicle on Washington State Route 20. Only the section in the drive cycle where the ACC was active is used in this test. The ACC operation in the drive cycle ended at a non-zero velocity thus, to complete the drive cycle i.e. bring velocity back to zero the drive cycle is copied symmetrically.

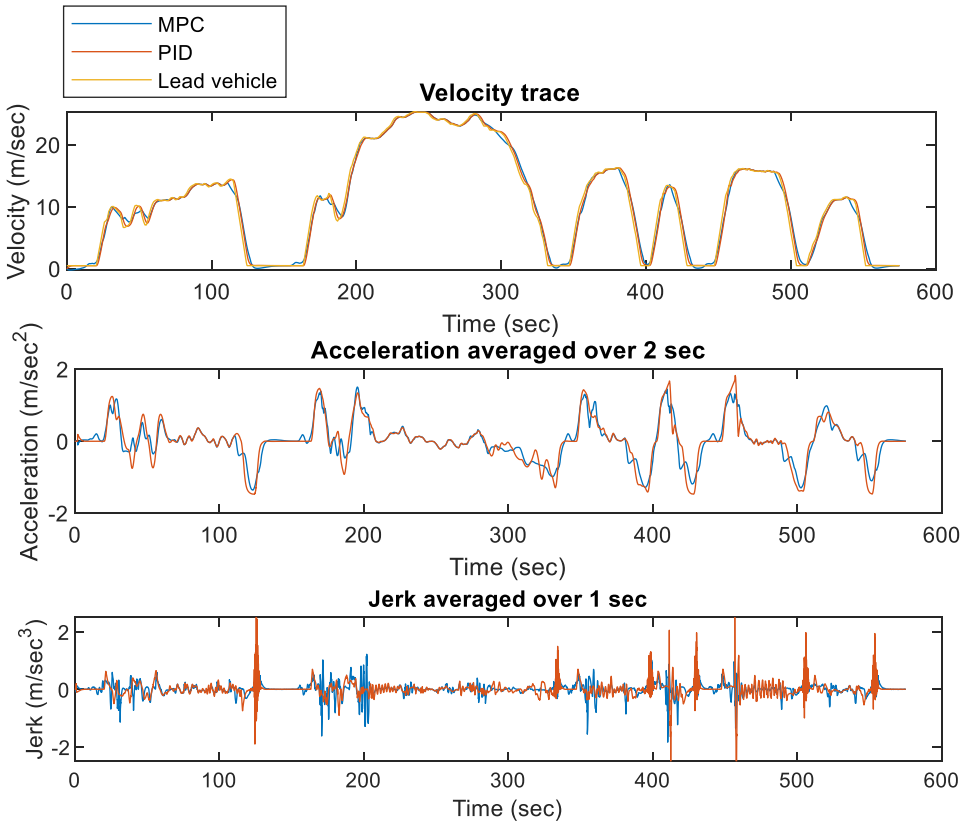

Figure 14: MIL FTP drive cycle run

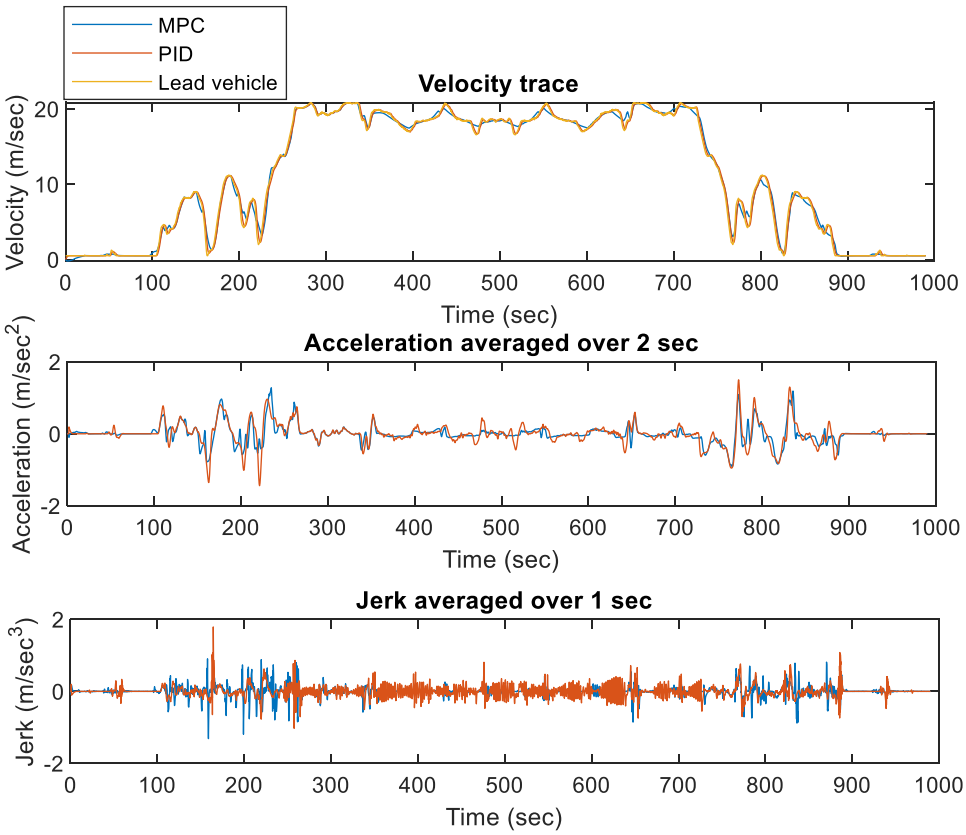

Figure 15: MIL BVE drive cycle run

As seen from the acceleration and jerk plots for FTP and BVE drive cycle in Fig 14, and Fig. 15 respectively, MPC operates with lower acceleration and jerk values, this is because of constraints set on acceleration and change in acceleration and a room 10 m provided for distance tracking error in the constraints set of MPC. At many points in the time of drive cycle run, PID control has a higher value of acceleration and jerk as it accurately tracks the distance from the lead vehicle which can cause it to accelerate or decelerate harshly if the lead vehicle does so. Lower and smoother acceleration behavior and inbuilt fuel consumption in the cost function of MPC also helps in conserving fuel as compared to PID control as shown in Table 6.

Drive cycle Controller	PID	MPC
FTP	43.2	44.2
BVE	48.6	49.4

Table 6: Fuel consumption in miles per gallon

4. Hardware-in-Loop (HIL)

HIL test is performed to test the real-time implementation of the controller on the actual hardware (dSpace MicroAutoBox II in this case). For running the plant model dSpace Simulator is used. The input data to the ACC is processed by the sensor fusion computer that provides velocity and distance data. In the actual vehicle, the communication between the ACC controller, and the sensor fusion computer will be using CAN lines. So, to get a realistic behavior of the system the communication between MABx and Simulator will be using CAN lines. And a simplified version of sensor fusion will run on Simulator.

4.1 Hardware description

MicroAutoBox II:

MABx is a controller platform by dSpace. It is a useful tool to test and debug real-time control software. Simulink also has support for auto code-generation for MABx. The RTI1401 platform is the Simulink setting used to generate code for MABx. The MABx has 6 CAN channels for communication. Technical specifications for MABx can be found on the link[14].

dSpace Simulator:

The plant model is run in the dSpace simulator for the real-time simulation. The RTI1006 platform is the Simulink setting used to generate code for MABx.

4.2 CAN communication

CAN stands for Controlled Area Network, which is a communication protocol widely used in the automotive industry. CAN uses message identifiers to decide the priority of a message i.e. which message is passed first. A simple CAN architecture is shown in Fig. 15. 0x0 (Hex form) has the highest priority in any CAN network. Then, as the identifier value increases, the priority level decreases. To implement CAN dSpace systems have Simulink block-set named CANMM.

We just need to provide a .dbc database file to the CANMM block-set for it to interpret the in-coming and encode the out-going CAN messages. DBC files are created using CANdb++ which is a database tool by Vector.

There are in-all 2 CAN lines used for the HIL test in this work. CAN line 1 is used for communication between powertrain controls and hybrid-supervisory control and CAN line 2 is used for communication between ACC and the sensor fusion model to get the required data for ACC operation.

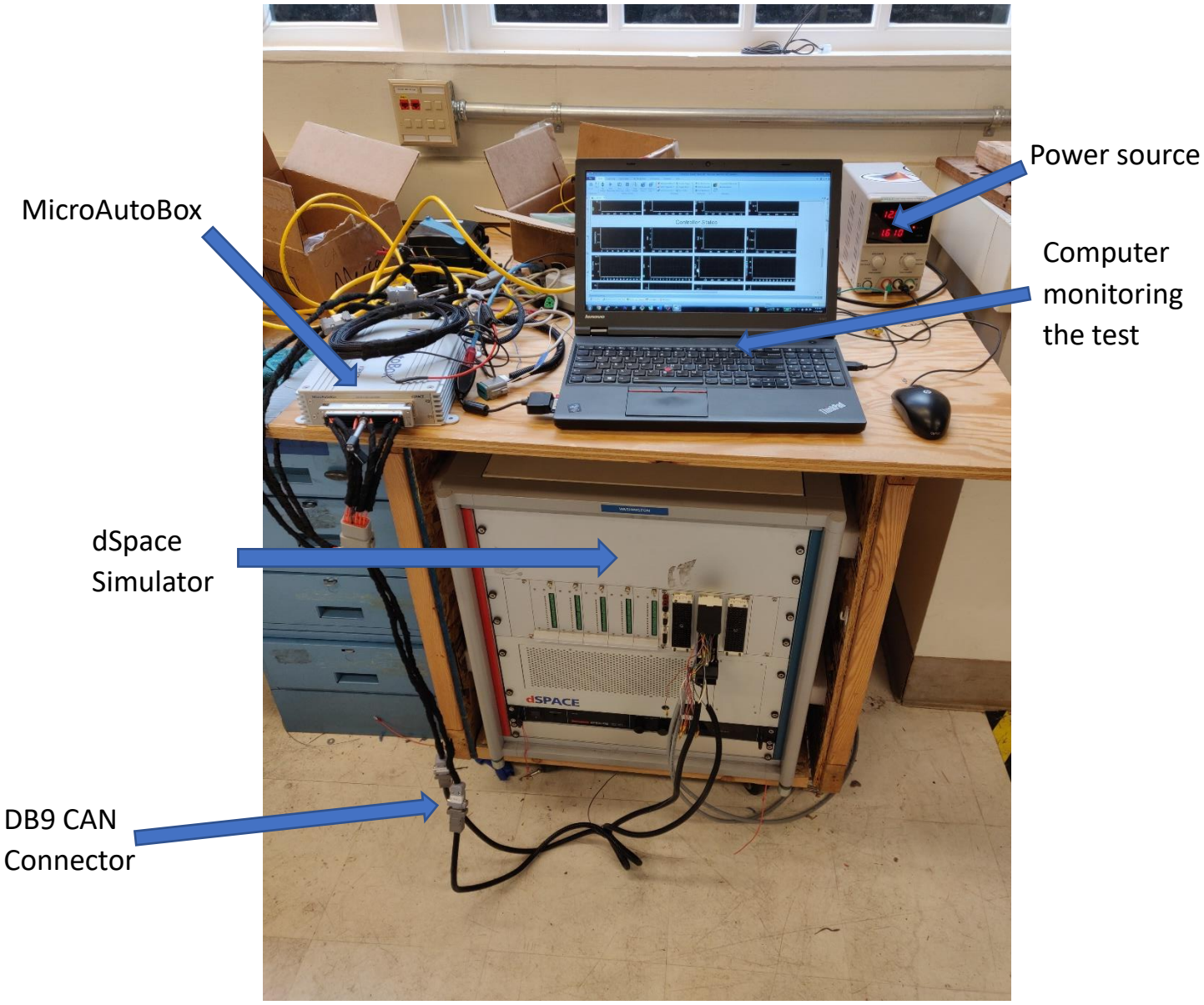

Figure 16: HIL set-up

4.3 HIL results

The purpose of the HIL tests is to show the real-time implementation ability of a controller. Fig. 16, 17, 18 and 19 shows real-time implementation of FTP PID, FTP MPC, BVE PID and BVE MPC (drive cycle controller) runs.

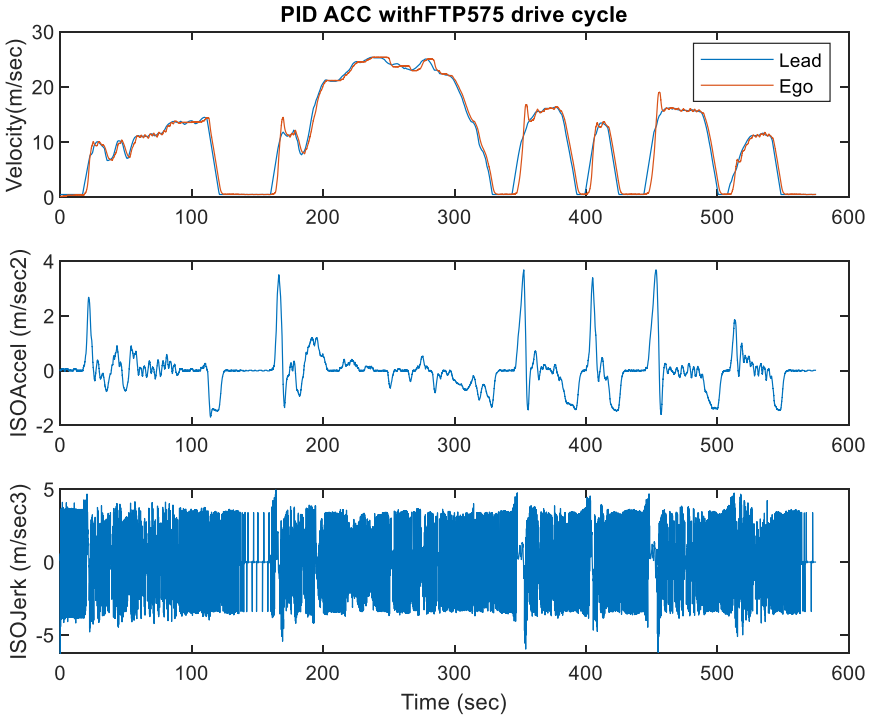

Figure 17: PID ACC run on FTP

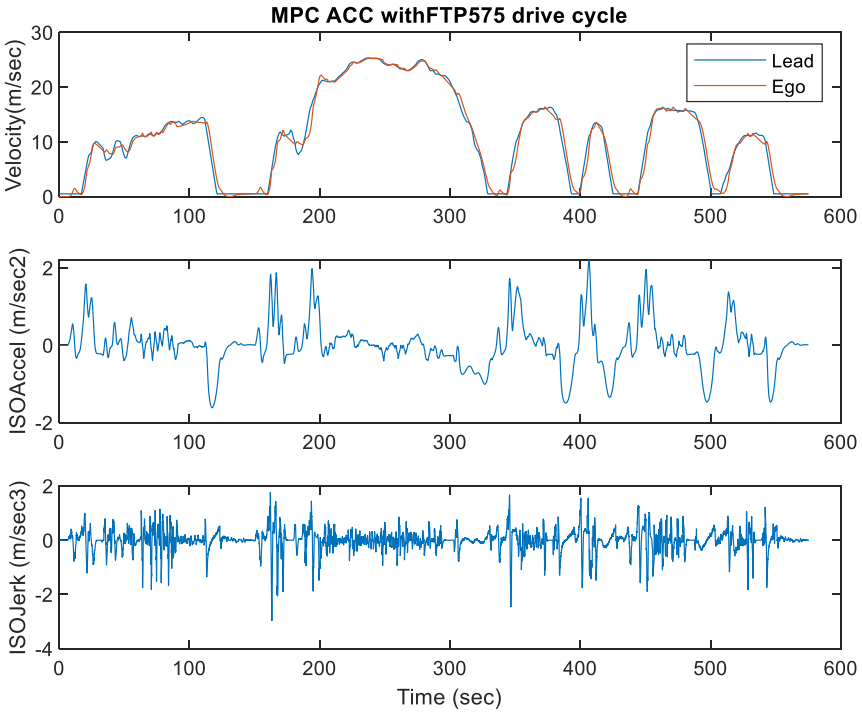

Figure 18: MPC ACC run on FTP

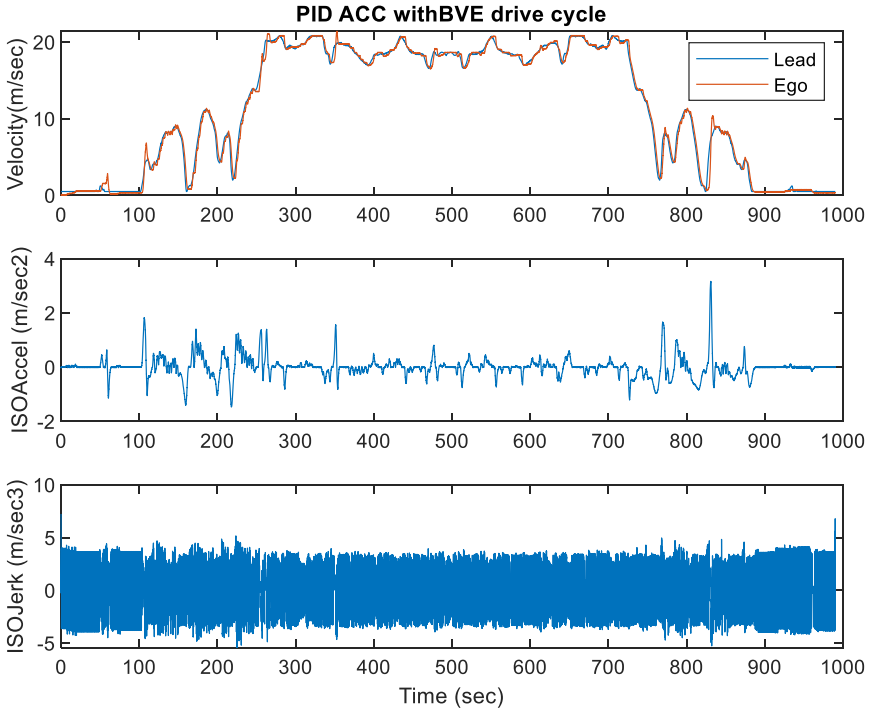

Figure 19: PID ACC run on BVE

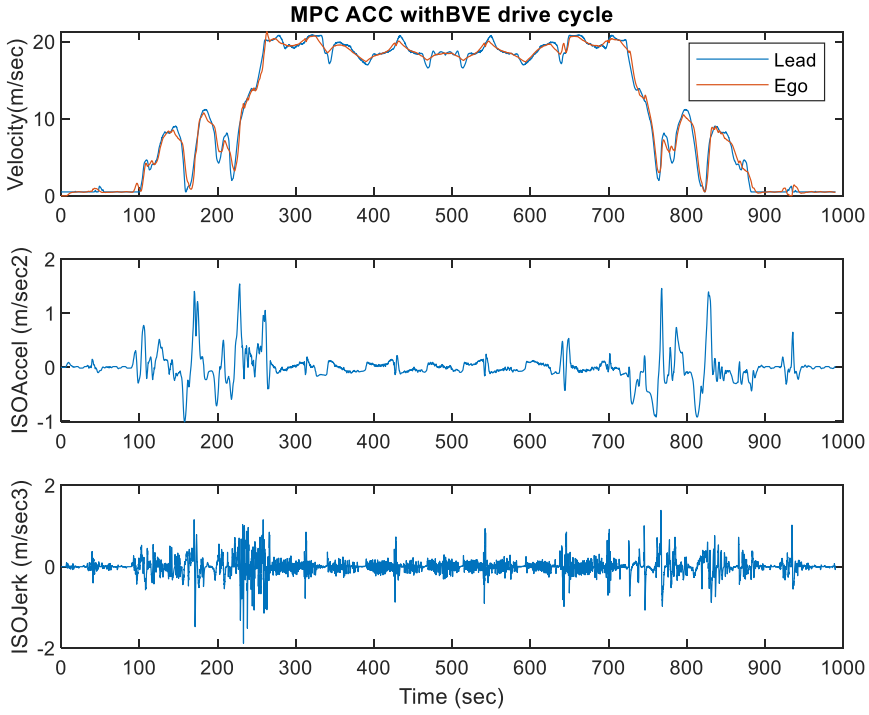

Figure 20: MPC ACC run on BVE

ISOAccel and ISOJerk correspond to acceleration calculated as per ISO standards mentioned in Section 2.3. Analyzing the first subplot in Fig. 16-19 it can be seen that PID is better than MPC at tracking the lead velocity. But, this tracking capability comes at the expense of reduced driver comfort as seen from ISOJerk subplots where the PID surpasses the ISO Average jerk whereas MPC has a jerk profile that is well below the ISO limits. And ISOAccel limits are obeyed in both the cases i.e. PID and MPC.

An important thing to note is that the high values of jerk in PID control can be an artifact of the method that is used to produce the jerk values, i.e. differentiating the acceleration vaues. Noisy acceleration data can be a reason for such a high values of jerk.

5. Conclusion and Future Work

The goal of this project was to compare the performance of PID control and MPC control for ACC of UW EcoCAR vehicle. The performance factors taken into consideration were acceleration, jerk (for driver comfort), tracking capability, and fuel consumption. The MPC has the flexibility for the trade-off between tracking response and fuel consumption. So, by giving some room for tracking error to vary within, MPC can make smart decisions to save fuel and improving driver comfort. PID has an excellent tracking response but, it cannot plan with lead vehicles estimated velocity so, it does not make any control decisions related to fuel savings neither does it perform well in the aspects of driver comfort. So, from MIL results we can conclude that by providing some room for tracking error we can provide an opportunity for MPC ACC to conserve fuel.

An important thing to note is the control rate of MPC which is 0.1 sec as compared to PID which is 0.01 sec. Due to computational limits the MPC control rate cannot be reduced to 0.01 sec unless some faster optimization algorithm is used. But the control rate of 0.1 sec is not sufficient for safety at high-speed. For example, if a vehicle cuts-in and decelerates to zero within 0.1 sec than the MPC with control rate 0.1 sec will not be able to respond. So further work is required to understand what the deceleration values is in the real-world emergency braking scenario to address this problem. One solution to this problem can be to incorporate another controller like collision avoidance which operates at higher control rates to ensure safety.

HIL runs show that even with a high computational load of MPC as compared to PID, MPC based control is achievable using powerful control hardware like MicroAutoBox II. It is important to note that the allowable size of the MPC problem is limited to the computational power available. From the HIL results, it can be concluded that even in the real-time implementation the MPC shows the driver comfort performance improvements as demonstrated in the MIL runs.

Next, it will be important to investigate how does the algorithm perform when the estimated future velocity of the lead vehicle (v_{lead}) is not exact or accurate as was in the case of this work. This can be done by developing non-linear estimation models like NARX models. The estimation errors in the disturbance (v_{lead}) is very likely to reduce the performance. The estimation of such models can be further improved and brought closer to the actual values by V2I and V2V communication architecture which is yet to be studied. Secondly, the MPC cost function can be updated by including transmission loss maps and motor energy consumption maps for possibly further improvement in fuel economy.

6. Bibliography

[1] P. Fancher, Tests characterizing performance of an adaptive cruise control system

[2] Alessandro Magnani, Convex piecewise-linear fitting

[3] ISO - Intelligent transport systems -- Adaptive Cruise Control systems -- Performance requirements and test procedures

[4] CVXGEN convex optimization solver - https://cvxgen.com/docs/index.html

[5] Washington_state_driver_guide

[6] Joost C.F. de Winter, Effects of adaptive cruise control and highly automated driving on workload and situation awareness A review of the empirical evidence

[7] Rustem Abdrakhmanov, Efficient ACC with Stop-Go Maneuvers for Hybrid Vehicle with online sub-optimal management

[8] Changwoo Park, A Study of Adaptive Cruise Control System to Improve Fuel efficiency

[9] Roman Schmied, Extension and Experimental Validation of Fuel Efficient Predictive

[10] Shengbo Li , Model Predictive Multi-Objective Vehicular Adaptive

[11] Taku TAKAHAMA, Model Predictive Control Approach to Design Practical Adaptive Cruise Control for Traffic Jam

[12] Aman Kalia, et.al Development of A Midsize Split-Parallel Hybrid Electric SUV for Shared Mobility

[13] M. Sami Fadali, Digital Control Text Book

[14] MABx technical specifications - https://www.dspace.com/en/inc/home/products/hw/micautob/microautobox2.cfm#144_31905

[15] Gerrit J. L. Naus, String-Stable CACC Design and Experimental

7. Appendix

Code:

- **Acceleration at Maximum torque request**
  https://drive.google.com/file/d/11YCx9R0KC_C3L2-g6dJvCtJbxGrwq6vj/view?usp=sharing

- **Open-loop Acceleration response**
  https://drive.google.com/file/d/1zyTfpiV6jGgurDiJzrNn_BATV1nHjD6g/view?usp=sharing

- **System Identification**
  https://drive.google.com/file/d/1TNNz-j4IIXlKfCqvfS1qP9ItoLfpjji-/view?usp=sharing

- **Closed-loop Acceleration response**
  https://drive.google.com/file/d/1PDzDxtzDRPCbEfdVv8nUk79tZguFQld9/view?usp=sharing

- **Convert engine fuel rate map (STFm) to vehicle fuel rate-acceleration-velocity map (VAFm)**
  https://drive.google.com/file/d/1yIfpzlN_4gfUy8gbp8L8ftjwlG739Oz-/view?usp=sharing

  Refer Flowchart_STFm_to_VAFm.drawio for explanation
  https://drive.google.com/file/d/1iU8zThCbFgzgcTvS87_1UDZdT69KWhXg/view?usp=sharing

- **Piece-wise linear approximation of VAFm for each gear**
  https://drive.google.com/file/d/1yN5Sh13SvLEZxQpqq3_pwMWhGv0FNDYs/view?usp=sharing

  Functions:-
  https://drive.google.com/file/d/1yIfpzlN_4gfUy8gbp8L8ftjwlG739Oz-/view?usp=sharing
  https://drive.google.com/file/d/1siFzqEJE-z41RN2Pn8MNdoizCDx5KbXe/view?usp=sharing
  https://drive.google.com/file/d/1sP00IpgvNNGDI1sQhmFTmxm5RY4qGHRr/view?usp=sharing
  https://drive.google.com/file/d/1sAN4sYbA3St-XyuvjFvtp1TcAh-bSb4P/view?usp=sharing
  https://drive.google.com/file/d/1D9ogHJPW5QqkJEL7DOhvm7XTe0WC1mnK/view?usp=sharing
  https://drive.google.com/file/d/1DVFMSHT07SV0VdzZYZTzJG1iLidykwiH/view?usp=sharing

- **S-function for MPC QP solver**
  https://drive.google.com/file/d/1fZpbdj7ijfwKnED0uBJ3Ha_MVy-PCgTY/view?usp=sharing

- **CVXGEN problem structure**
  https://drive.google.com/file/d/1xpazI4QxkvX_CiBqcnXxi79-GJxB9zCl/view?usp=sharing