2019

Intelligent Transportation Systems, Hybrid Electric Vehicles, Powertrain Control, Cooperative Adaptive Cruise Control, Model Predictive Control

Hadi Kazemi
West Virginia University, hakazemi@mix.wvu.edu

2019

Intelligent Transportation Systems, Hybrid Electric Vehicles, Powertrain Control, Cooperative Adaptive Cruise Control, Model Predictive Control

Hadi Kazemi

Information Driven Vehicle: CACC Interference Handling and HEV Powertrain Control

Hadi Kazemi

Dissertation submitted to the
Benjamin M. Statler College of Engineering and Mineral Resources
at West Virginia University

in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Electrical Engineering

Yaser P. Fallah, Ph.D., Chair
Parviz Famouri, Ph.D.
Muhammad A. Choudhry, Ph.D.
Vinod K. Kulathumani, Ph.D.
Scott W. Wayne, Ph.D.

Lane Department of Computer Science and Electrical Engineering

Morgantown, West Virginia
2019

Keywords: Intelligent Transportation Systems, Hybrid Electric Vehicles, Powertrain Control, Cooperative Adaptive Cruise Control, Model Predictive Control

Abstract

Information Driven Vehicle: CACC Interference Handling and HEV Powertrain Control

Hadi Kazemi

lnformation obtainable from Intelligent Transportation Systems (ITS) provides the possibility of improving the safety and efficiency of vehicles at different levels. In particular, such information has the potential to be utilized for prediction of driving conditions and traffic flow, which allows us to improve the performance of the control systems in different vehicular applications, such as Hybrid Electric Vehicles (HEVs) powertrain control and Cooperative Adaptive Cruise Control (CACC). In the first part of this work, we study the design of an MPC controller for a Cooperative Adaptive Cruise Control (CACC) system, which is an automated application that provides the drivers with extra benefits, such as traffic throughput maximization and collision avoidance. CACC systems must be designed in a way that are sufficiently robust against all special maneuvers such as interfering vehicles cutting-into the CACC platoons or hard braking by leading cars. To address this problem, we first propose a Neural- Network (NN)-based cut-in detection and trajectory prediction scheme. Then, the predicted trajectory of each vehicle in the adjacent lanes is used to estimate the probability of that vehicle cutting-into the CACC platoon. To consider the calculated probability in control system decisions, a Stochastic Model Predictive Controller (SMPC) needs to be designed which incorporates this cut-in probability, and enhances the reaction against the detected dangerous cut-in maneuver. However, in this work, we propose an alternative way of solving this problem. We convert the SMPC problem into modeling the CACC as a Stochastic Hybrid System (SHS) while we still use a deterministic MPC controller running in the only state of the SHS model. Finally, we find the conditions under which the designed deterministic controller is stable and feasible for the proposed SHS model of the CACC platoon. In the second part of this work, we propose to improve the performance of one of the most promising realtime powertrain control strategies, called Adaptive Equivalent Consumption Minimization Strategy (AECMS), using predicted driving conditions. In this part, two different real-time powertrain control strategies are proposed for HEVs. The first proposed method, including three different variations, introduces an adjustment factor for the cost of using electrical energy (equivalent factor) in AECMS. The factor is proportional to the predicted energy requirements of the vehicle, regenerative braking energy, and the cost of battery

charging and discharging in a finite time window. Simulation results using detailed vehicle power train models illustrate that the proposed control strategies improve the performance and Internal Combustion Engine (ICE). This is important because there is no other prior work addressing this problem using a controller which can adjust its decision to the driving of AECMS in terms of fuel economy by 4%. Finally, we integrate the recent development in reinforcement learning to design a novel multi-level power distribution control. The proposed controller reacts in two levels, namely high-level and low-level control. The high-level control decision estimates the most probable driving profile matched to the current (and near future) state of the vehicle. Then, the corresponding low-level controller of the selected profile is utilized to distribute the requested power between Electric Motor (EM)pattern. We proposed to use two reinforcement learning agents in two levels of abstraction. The first agent, selects the most optimal low-level controller (second agent) based on the overall pattern of the drive cycle in the near past and future, i.e., urban, highway and harsh. Then, the selected agent by the high-level controller (first agent) decides how to distribute the demanded power between the EM and ICE. We found that by carefully designing a training scheme, it is possible to effectively improve the performance of this data-driven controller. Simulation results show up to 6% improvement in fuel economy compared to the AECMS.

Dedication

This thesis is dedicated to my wife, parents, brothers and sister for their love, support, and sacrifice. Also to all of my teachers during my student life for their encouragement and motivation.

Acknowledgments

I want to especially thank my PhD advisor, Dr. Yaser P. Fallah, for his great support and all my labmates Hossein, Ehsan, Amin, Osman, Mehdi and Ahmad.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Problem Definition

The advent of vehicular communication networks [1] provides situational awareness of various degrees [2, 3, 4] and a new possibility of using traffic and road information for safety enhancement, traffic management, emissions reduction, and fuel efficiency. Information is usually obtained from Vehicle-to-Vehicle (V2V) and Vehicle-to-Infrastructure (V2I) communication networks and can include data from various sensors such as Global Positioning Systems (GPS) or other onboard vehicle sensors. A combination of such communicated data and information from maps (Geographical Information Systems (GIS)) provides vehicles with a degree of real-time awareness of their surroundings and upcoming driving conditions.

The Intelligent Transportation Systems (ITSs) are promising solutions, improving the efficiency and safety of vehicles, by the integration of information, telecommunication, and cyber technologies into transportation systems. This integration provides the vehicles with a real-time comprehensive awareness of their surroundings and situations. While safety-related applications, such as collision avoidance and collision warnings, which stop the vehicle in an imminent accident scenario or warn the driver in a non-imminent situation, were the primary goals of ITS, they can effectively advance the performance of other efficiency-related applications, such as fuel efficiency and traffic management systems. Such ITS applications generally are developed on top of connected and automated vehicle (CAV) systems connected through wireless communication technologies.

Cooperative vehicle safety (CVS) systems are examples of ITSs that employ communication

amongst vehicles for the sake of safety and efficiency improvement in the transportation system. CVS systems are designed to broadcast necessary information through a shared channel and specify required mechanisms to predict the state, e.g., speed, of other vehicles. Each vehicle transmits its state information within a limited neighborhood around it. Forward collision warning (FCW) and forward collision avoidance (FCA) are two common examples of CVS applications. FCW (FCA) is designed to warn the drivers (stop the vehicle) to avoid a rear-end collision. Even though CVS has been mostly explored for the situation in that a human user has control over the vehicle, it could be extended to many automated driving applications, such as Cooperative Adaptive Cruise Control (CACC).

Drivers are the most important and influential entities of non-autonomous vehicles in ground Intelligent Transportation System (ITS). A revolutionary age of modern driving has been initiated by the advent of safety and comfort driving applications that aim at assisting drivers in vehicle control. Forward collision warning [5, 6, 7, 8], lane keep assistance [9, 10, 11], automatic braking [12], adaptive cruise control [13, 14], efficiency [15, 16], and pedestrian safety [17, 18, 19] systems are amongst the most important automated driving applications. The first generation of safety applications was designed by virtue of local sensors such as radars and cameras. Local sensors provide a mediocre level of safety due to their limited sensing range and data processing complexity. Moreover, they noticeably underperform in the presence of occluding obstacles. In order to handle these issues, some other sources of information are required to provide more accurate situational awareness within a broader neighboring area. Vehicle-to-Vehicle (V2V) communication has been proposed to remove this barrier through its omnidirectional and non-line of sight connectivity capabilities. Consequently, the performance of safety applications is expected to be substantially improved by V2V communications. Currently, the most promising technology under consideration for V2V communication is the Dedicated Short Range Communication (DSRC) [20] technology.

In contrast, in automated driving applications, such as cooperative collision avoidance (CCA), and platooning, the control system dictates the optimal motion of the vehicle, in terms of safety and efficiency, based on the information about the current situation of the host vehicle and its neighboring vehicles and the road. Adaptive Cruise Control (ACC) is one of the earliest applications of ITS, which improves the safety and ease of driving. Similar to the ACC, Cooperative Adaptive Cruise Control (CACC), which utilizes V2x communication technology, is a more powerful CCA

system to prevent collision and maximize traffic throughput simultaneously. It takes proper action by means of the steering, engine, brakes, etc. Consequently, a comprehensive understanding of the vehicle surroundings and possible scenarios, and the likeliness of every single scenario, in the near future is necessary to design such a controller. In the CACC and platooning, which both are car-following scenarios, the control system design is confronted with similar challenges. In car platooning the primary objective is to keep vehicles in a very close and reasonable distance from each other (gap control) to maximize the number of vehicles on the road. Similarly, in CACC the goal is defined as keeping a safe distance while providing a comfortable ride for passengers. Extensive research has been carried out in the recent past in the area of platooning and CACC which can be categorized as follows:

- Interaction of driver and CACC control unit, such as turning the CACC on [21].
- The impact of CACC on traffic in terms of congestion and safety [22].
- CACC application design [23].
- CACC communication requirements [24].
- Platoon string stability [25].

A CACC system should be designed in a way to be completely robust to any unexpected event such as other vehicle maneuvers including cutting-into and cutting-out of the CACC platoons. Detecting of an unanticipated maneuver of a remote vehicle and determining an appropriate reaction to it are two of the most demanding tasks in the normal driving situations with and without CACC. Despite the comprehensive technical and theoretical investigation of CACC systems by researchers in the recent past, still, a universal framework for reaction to unexpected maneuvers of remote vehicles, such as cutting into a stable CACC platoon, needs more attention before reaching a complete CACC design.

Unsignaled lane change is one of the most critical situations among unexpected remote vehicle maneuvers to be addressed in CACC design as it can significantly affect the level of safety and platooning performance in this application. In this work, we specifically focus on a cut-in maneuver by a remote vehicle, due to its imminent threat to the safety and stability of the whole CACC

platoon. A vehicle in a stable CACC platoon needs to apply a hard brake in the case of a remote vehicle joining the platoon unexpectedly. The action of a hard brake increases the possibility of platoon deformation, or in the worst case, a crash. Consequently, predicting a possible upcoming cut-in maneuver by the neighboring vehicles, and having an estimation of their near future behavior can prevent the necessity of hard brakes and their consequences. In addition, a full CACC system needs to keep the platoon formation robust against entering and leaving vehicles. The performance of such CACC design in these detracting driving scenarios is highly dependent on the accuracy of vehicle trajectory prediction. Therefore, the prediction of a possible lane change is a crucial part of any platooning system design.

In the case of dealing with lane change scenarios, cut-out maneuvers are usually safe for CACC and can be handled without any risk. The CACC system only needs to speeds up the vehicle behind the departing vehicle to sync their distance to the new leader by filling the cut-out gap. However, in contrast, addressing a cut-in maneuver needs a mechanism for precise tracking of the remote vehicle joining the platoon. The new vehicle cannot only deform the platoon but can increase the risk of an accident. Thus, it is well-desired to first predict the probability of any remote vehicles in the adjacent lanes to show a cut-in intention beforehand, and then, the CACC system needs to take actions based on how probable a cut-in is to happen in front of the host vehicle. The action should not over-react to possible cut-in maneuvers with low probability to keep the comfort-drive goal of the CACC. At the same time, the controller should react fast to highly possible cut-in scenarios to avoid any possible crash.

In addition to incorporating the probability of a cut-in in CACC decision, the controller should consider how harsh is the upcoming cut-in. In other words, when a remote vehicle enters the CACC platoon smoothly, the controller should open up enough space for the entering vehicle as smooth as possible by reducing the speed of the host vehicle. However, in case of a harsh agile cut-in, the controller needs to make a brake as harsh as necessary to prevent an accident. Therefore, in this work, we address the problem of CACC system design, robust to interfering vehicles. We design a new MPC controller for CACC and prove the stability and feasibility of the controller. We demonstrate the effectiveness of this approach to control of CACC through several simulation studies. Then we propose an approach to consider an interfering vehicle in this problem.

ITS can also be used in efficiency related applications, such as power distribution management

in Hybrid electric vehicles (HEVs). HEVs have great impacts on fuel economy and are promising alternative means of transportation. In comparison with the conventional ICE powered vehicles, they are able to improve the fuel efficiency employing an engine smaller in size, and capturing the regenerative braking energy to recharge the batteries. Depending on the capacity of the batteries, which usually can range between 10 to 70 miles of driving, HEVs can exploit cheaper grid electricity to take over from fossil fuel. HEV power distribution control has been vastly studied, in the past decade, as an optimization problem. Fuzzy logic and other rule-based control techniques are investigated in HEV power distribution management, by splitting the driving conditions into multiple scenarios [26]. Their main advantage is their ease of implementation, while they neglect detailed vehicle dynamics. Others approached this problem as an Optimal control design [27], based on the current operation, which has a low computational complexity for real-time implementation.

Sliding mode control has also been investigated for the sake of robustness to the parameter and model variations or other external disturbances [28]. To find the global optimal solutions of the problem, one may use dynamic programming (DP) techniques. However, DP-based solutions need prior knowledge of the entire driving condition. Such dependency of a priori information makes DP approaches inappropriate for real-world implementation. As an alternatives adaptive equivalent consumption minimization strategy (AECMS) algorithm was proposed in [29] which changes the relative cost of using electrical energy compared to fossil fuel based on the current and history of the SOC.

A method based on mixed fuzzy classification and pattern learning was proposed in [30] which relies on learning about the driving conditions from the driving history information or optimization using the standard driving cycles. However, in real scenarios, the actual driving cycle can be completely dissimilar to the standard driving cycles, which results in a non-optimal power distribution between EM and ICE for different drivers or trips.

The recent developments in the area of Connected and Automated Vehicles (CAVs), such as GIS, GPS, vehicle-to-vehicle (V2V), and vehicle-to-infrastructure (V2I) communications, enables ITS to come up with new and better solutions to vehicle driving profile prediction. The road conditions, traffic lights, and speed limits can be actively utilized to have a precise prediction of the driving conditions in the near future (a short time prediction). This opens a new set of power distribution strategies which make use of limited information about the near future of the driving

conditions to improve the performance of real-time methods which only consider the instantaneous optimization of the powertrain components. To this end, we propose a sub-optimal strategy and a learning-based model-free controller based on Reinforcement Learning (RL) to improve the fuel economy of an HEV incorporating the information of the upcoming driving conditions.

1.2 Contributions and Dissertation Structure

1.2.1 Chapter 2

Vehicle to Vehicle (V2V) communication has a great potential to improve reaction accuracy of different driver assistance systems in critical driving situations. Cooperative Adaptive Cruise Control (CACC), which is an automated application, provides drivers with extra benefits such as traffic throughput maximization and collision avoidance. CACC systems must be designed in a way that is sufficiently robust against all special maneuvers such as cutting into the CACC platoons by interfering vehicles or hard braking by leading cars. To address this problem, a Neural-Network (NN)-based cut-in detection and trajectory prediction scheme is proposed in the first part of this dissertation. Next, a probabilistic framework is developed in which the cut-in probability is calculated based on the output of the mentioned cut-in prediction block. Finally, a specific Stochastic Model Predictive Controller (SMPC) is designed which incorporates this cut-in probability to enhance its reaction against the detected dangerous cut-in maneuver. The overall system is implemented, and its performance is evaluated using realistic driving scenarios from Safety Pilot Model Deployment (SPMD). Specifically, our contributions in this work are listed as follows:

- A learning-based driver behavior modeling sub-system is proposed to accomplish an accurate lane change prediction.
- A probabilistic framework is designed which employs the results of the lane-change monitoring block and translates it to a cut-in probability value.
- A new Model Predictive Controller (MPC) is developed and its stability is proved. As a future work we are going to design a Stochastic Model Predictive Controller (MPC) which takes the cut-in probability as its input. This SMPC controller is in charge of adjusting the dynamic parameters (mainly velocity and spacing error) of the vulnerable vehicles inside the

platoon. More specifically, it minimizes the spacing error (deviations from a predefined safe distance) between the vehicle and its immediate vehicle ahead, while keeping their velocity difference as close as possible to zero. Concurrently, it responds appropriately to a cut-in maneuver.

- The overall system architecture is designed and represented as a Time-Triggered Stochastic Impulsive System (TTSIS) model [31], originated from the emerging stochastic hybrid systems (SHS) methodology [32, 33, 31].

The Matlab codes of this chapter are listed in Appendix 6.

1.2.2 Chapter 3

Information obtainable from Intelligent Transportation Systems (ITS) provides the possibility of improving the safety and efficiency of vehicles at different levels. In particular, such information also has the potential to be utilized for prediction of driving conditions and traffic flow, which allows Hybrid Electric Vehicles (HEVs) to run their powertrain components in corresponding optimum operating regions. This dissertation proposes to improve the performance of one of the most promising realtime powertrain control strategies, called Adaptive Equivalent Consumption Minimization Strategy (AECMS), using predicted driving conditions. In this dissertation, three real-time powertrain control strategies are proposed for HEVs, each of which introduces an adjustment factor for the cost of using electrical energy (equivalent factor) in AECMS. These factors are proportional to the predicted energy requirements of the vehicle, regenerative braking energy, and the cost of battery charging and discharging in a finite time window. Simulation results using detailed vehicle powertrain models illustrate that the proposed control strategies improve the performance of AECMS in terms of fuel economy, the number of engine on/off events, and charge sustainability of the battery. Specifically, our contributions in this work are listed as follows:

- A framework is proposed to utilize the information about the predicted near future driving conditions in order to make smarter and more efficient power distribution in HEVs.

- The proposed method had low computational complexity compared to the global optimization techniques, and therefore, can be adopted for real-time implementation.

The Matlab codes of this chapter are listed in Appendix 6.

1.2.3 Chapter 4

An energy management strategy (EMS) plays a critical role in the efficiency of HEVs. However, the driver-specific and general variation of driving conditions affect the optimality of traditional EMSs. Majority of the EMSs are designed to track a set of pre-specified rules that are not adaptive to the variable driving conditions. Consequently, we found it beneficial to design an EMS framework, which we refer to as drive cycle aware EMS, that can adapt its rules to the current driving condition. This adaptation could be to a general pre-defined driving patterns, such as urban, highway, or harsh driving conditions, or drive-specific driving habits. To this end, we propose a deep Q-Network (DQN) based EMS such that it can switch between multiple policies based on the overall driving conditions in the past and near future. Similar to our proposed PECMS, the EMS output action is the ratio of demanded power distribution between the electric motor and the internal combustion engine. The effectiveness of the proposed method is studied with a set of simulation experiments. Experimental results validate the superiority of drive cyle aware EMS compared to our proposed PECMS. In summary, our contributions in this work are listed as follows:

- Unlike the previous adoption of reinforcement learning for HEV powertrain control, which assume that there is a single optimal policy for all the drive-cycles, we propose a new architecture which employs two different policies for different levels of control. This modification lets the proposed framework learn a different policy for each driving condition pattern.

- The proposed framework can converge to the optimal power management policy by learning a deep reinforcement learning (DRL) agent in an offline fashion. The proposed method then is considerably faster than the previous methods proposed in the literature.

- Unlike the global optimization policies, the proposed method does not rely on a priori knowledge of the driving condition. It is also a data-driven model which does not require any detailed and accurate HEV modeling.

The Matlab codes of this chapter are listed in Appendix 6.

Chapter 2

Stochastic MPC Design for Cooperative Adaptive Cruise Control to Handle Interfering Vehicle

2.1 Introduction

Adaptive Cruise Control (ACC) is one of the most demanding automated driving applications. In comparison with its predecessor, i.e. conventional cruise control which had been solely designed to provide a fluctuation-free driver-specified velocity, ACC is also responsible for sustaining a certain level of safety by continuously tracking the vehicle longitudinal distance from its immediate leader and keeping this distance within a safe range. One step ahead in cruise technology would result in Cooperative Adaptive Cruise Control (CACC), which also leverages the V2V communication (see Figure 2.1). This makes it more powerful to simultaneously preclude collision and maximize traffic throughput compared to ACC [34]. However, many CACC challenges still exist which need to be addressed. For instance, the CACC application should be robust against other vehicles' maneuvers such as unforeseen lane changes [34]. Detection and appropriate reaction to these unexpected vehicle maneuvers are among the most challenging tasks, even in the normal driving situations and without CACC imposed constraints. These challenging tasks reveal the criticality and complexity of a well-behaved CACC design for these scenarios. Even though different theoretical and technical aspects of CACC have been investigated by researchers [35], handling interfering vehicles needs more elaborations.

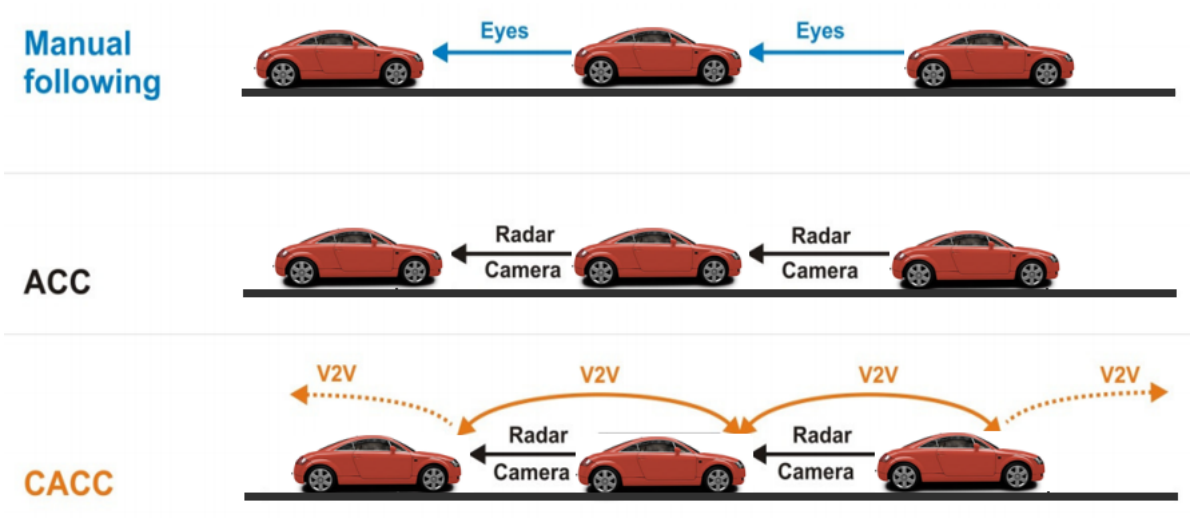


Figure 2.1: The terminal constraint set Ω.

In this work, we specifically concentrate on cut-in maneuvers due to their imminent threat, as a vehicle in a stable CACC platoon has to perform a hard brake reaction when another vehicle makes a sudden lane change just in front of it. This hard brake reaction is extremely dangerous and can result in a severe crash [36, 37]. Thus, it is well-desired to predict cut-in intention of other drivers in advance. Moreover, cutting into the platoon deforms the platoon structure which should be compensated by a proper CACC design. Therefore, a meticulous CACC system should be able to both prevent possible crashes and maintain the normal platoon formation against entering vehicles from adjacent lanes.

Based on the above discussion, the performance of CACC in these critical driving scenarios is extremely reliant on the accuracy of modeling other drivers behavior in the sense of detecting their lane change intentions and predicting cut-in path. This fact requires the introduction of a "lane-change monitoring block", which performs the aforementioned functions, as an inseparable and essential part of our CACC system. Specifically, our contributions in this work [**?**] are listed as follows:

- A learning-based driver behavior modeling sub-system is proposed to accomplish an accurate lane change prediction.
- A probabilistic framework is designed which employs the results of the lane-change monitoring

block and translates it to a cut-in probability value.

- A new Model Predictive Controller (MPC) is developed and its stability is proved. As a future work we are going to design a Stochastic Model Predictive Controller (MPC) which takes the cut-in probability as its input. This SMPC controller is in charge of adjusting the dynamic parameters (mainly velocity and spacing error) of the vulnerable vehicles inside the platoon. More specifically, it minimizes the spacing error (deviations from a predefined safe distance) between the vehicle and its immediate vehicle ahead, while keeping their velocity difference as close as possible to zero. Concurrently, it responds appropriately to a cut-in maneuver.

- The overall system architecture is designed and represented as a Time-Triggered Stochastic Impulsive System (TTSIS) model [31], originated from the emerging stochastic hybrid systems (SHS) methodology [32, 33, 31].

To the best of our knowledge, this is the first cut-in resistant CACC-SMPC design based on a real-time cut-in probability calculation in the literature.

The rest of this chapter is organized as follows. Section 2.2 is devoted to related works on proposed driver behavior modeling methods in the literature. The overall system description is explained in section 2.5 in which sections 2.5.1, 2.5.2 state the details of our learning-based cut-in monitoring block, proposed cut-in probability calculation approach based on that, respectively. The overall system performance is evaluated in section 2.6.

2.2 Related Work

The driver is the main source of system stochasticity in most of the ground ITS frameworks. Each maneuver of a vehicle is an immediate and direct consequence of its driver's intention, which is applied by a specific set of mechanisms, such as steering wheel, pedals, and handles [38]. The utilization of these tools can be directly measured through Controller Area Network (CAN). However, it is not possible to deterministically assign a maneuver to a specific pattern of these parameters as different maneuvers may have partially similar sections [39]. Therefore, a reliable approach is required to discriminate different driving maneuvers based on measured patterns of their param-

eters. The output of this stage could then be utilized to design an application-specific controller. This controller would obviously perform smarter compared to the controller which only acts based on the previous measurements without any predictive vision of driving scenario. The prevailing methods in the literature for driver behavior modeling, and some of the important proposed designs for adaptive cruise controller are mentioned in this section.

One of the important research mainstreams in driver behavior modeling is based on utilizing classification methods, such as Support Vector Machine (SVM) and Neural Network (NN), to differentiate between distinguishable driver behaviors. The main idea behind another major class of driver behavior modeling schemes in the literature, such as Hidden Markov Models (HMMs) and Dynamic Bayesian Networks (DBNs), is developing a probabilistic causal framework which tries to find the next most likely driving maneuvers using available data sequences from the driving history and then chain these predicted consecutive maneuvers to construct the most probable future scenario.

Authors in [40] developed a hierarchical classifier for observed scenes of the host vehicle from remote vehicle's lane change. These scenes were then assigned to the nodes of the hierarchy in the model to specify a pattern from the top nodes to the leaves. However, their overall scheme is not generalizable to other contexts, such as potential maneuver alternatives, since it is remarkably specialized.

SVM-based methods are proposed to classify lateral actions of drivers based on detection of preparatory behaviors, vehicle dynamics, and the environmental data prior to and during the maneuvers such as lane change [41]. A Relevance Vector Machine (RVM), was employed in [42] to distinguish between lane change and lane keeping maneuvers. In [43], feed forward artificial neural networks are used to predict the trajectory of the vehicle based on its movements history. The goal was to study the possibility of accurate movement prediction for a lane changing vehicle by an autonomous driving vehicle.

An Object-Oriented Bayesian Network (OOBN) is utilized to recognize special highway driving maneuvers, such as lane change [44]. This approach models different driving maneuvers as vehicle-lane and vehicle-vehicle relations on four hierarchical levels which can tolerate uncertainties in both the model and the measurements. A finite set of driving behaviors are classified and future trajectories of the vehicle are predicted based on currently understood situational context using a

DBN-based model [45].

Hidden Markov model (HMM) technique has been widely utilized to associate the observable time series of the vehicle to the unobserved driver intentions sequence during his maneuvers [46, 47, 39, 48, 49, 38, 50, 51]. Some pioneer works in driver behavior modeling, [46, 47], proposed a decomposition of driver behaviors into small scale and large scale categories. Time sequence of unobserved large scale driver actions are assumed to have Markovian property and HMM is suggested as an acceptable method to model this sequence. This modeling approach accuracy was validated by its results of the lane change maneuver prediction. Sensory collected information was used as the observation set in the designed HMM predictor.

Using the data from V2V communication, two HMMs were utilized to discriminate different types of driver lane change intent, namely dangerous and normal [39, 48]. A trajectory prediction stage and an MPC controller were mounted on top of the lane change prediction algorithm to manage reformation of a new CACC string after cut-in.

A controller for a CACC string which takes into account both V2V and non-V2V equipped vehicles was designed in [34]. This controller tries to handle cut-in and cut-out scenarios with a smooth reaction to the new condition of the host vehicles lane. No prediction is performed in this work to detect the cut-in or cut-out scenarios in advance. Another CACC design based on switched sampled-data model is presented in [35] which investigates the stability problem in the presence of sensor failures.

2.3 Model Predictive Control (MPC)

Model Predictive Control (MPC) is an optimal control strategy based on numerical optimization. This controller exploit the predicted future plant responses using the model of system to find the optimal future control inputs and optimize it at periodic intervals. In general, predictive control strategies are among the most popular advanced control techniques as computational methods for improving the performance of control system in various applications in the industry. Among them, MPC is the on with a logical theoretical ground whose optimality, stability, and robustness characteristics are well understood.

Even the design of an MPC control system is comparatively straightforward and it is easy to

implement, MPC algorithms are able to control large scale systems with a large number of control variables. Moreover, it provides a systematic approach of enforcing the physical and performance related constraints on system inputs and states. Satisfying the constraints is a critical part of all control design applications to meet the limitations of system actuators and states or physical, economic, or safety restraint. Taking these constraints into account, convert the MPC design problem into a real-time constrained optimization problem to determine the optimal control inputs to the system in the control horizon.

The predictive control feedback law can be computed by minimizing a performance-related cost over the prediction horizon. To this end, it predicts the future response of the closed-loop system using a dynamic model of the system. In this work, we use state-space representation of a discrete-time linear systems of our CACC system to design our MPC controller. Such system is generally defined as

$$x(k+1) = Ax(k) + Bu(k) \tag{2.1}$$

where $x(k)$ and $u(k)$ are the state and input vectors at the kth sampling time step. Then, to predict the sequence of system states having a predicted input sequence, we can simulate the model over the prediction horizon. For the sake of notational convenience, these predicted sequences are denoted by vectors $\boldsymbol{u}$, $\boldsymbol{x}$ defined as

$$\boldsymbol{u}(k) = \begin{bmatrix} u(k|k) \\ \vdots \\ u(k+N-1|k) \end{bmatrix} \qquad \boldsymbol{x}(k) = \begin{bmatrix} x(k+1|k) \\ \vdots \\ x(k+N|k) \end{bmatrix}, \tag{2.2}$$

where $u(k+i|k)$ and $x(k+i|k)$ represent input and state vectors at time $k+i$ that are predicted at time k. The predicted state vector $x(k+i|k)$ is derived as:

$$x(k+i+1|k) = Ax(k+i|k) + Bu(k+i|k), \quad i = 0, 1, \ldots \tag{2.3}$$

with the initial condition defined

$$x(k|k) = x(k) \tag{2.4}$$

To find the predictive control feedback law a predicted performance cost needs to be minimized, which is a function in terms of the predicted sequences $\boldsymbol{u}$, $\boldsymbol{x}$. In this dissertation we employ a quadratic cost, in which the predicted cost has the following general form:

$$J[k] = \sum_{i=0}^{N-1} [x^T[k+i|k]Qx[k+i|k] + u^T[k+i|k]Ru[k+i|k]] \tag{2.5}$$

where Q, R are positive definite matrices (note that the Q may be positive semi-definite). Obviously, the predicted cost $J(k)$ is a function of control input $u(k)$, and therefore, the optimal input sequence for the optimization problem of minimizing $J(k)$, denoted by $\boldsymbol{u}^*(k)$, is defined as:

$$\boldsymbol{u}^*(k) = \arg\min_u J(k) \tag{2.6}$$

Note that in the case of input and state constraints, they should be included in the optimization.

In this dissertation we use receding horizon implementation of MPC, in which despite calculation of the predicted input sequence over the entire control horizon, we only input to the system the first element of optimal predicted input sequence $\boldsymbol{u}^*(k)$. Then, this process is repeated by minimizing the predicted cost at the every next sampling time steps $k = 0, 1, \ldots$. This optimization technique is also known as an online optimization. Since we use the same prediction horizon length for the optimization at future time steps, this approach is called a receding horizon strategy.

The receding horizon approach helps in two ways. First, since the state predictions $\boldsymbol{x}$ depends on the current state measurement $x(k)$, the optimal input sequence $\boldsymbol{u}^*$ also depends on $x(k)$, which introduces feedback into the MPC law. This feedback provide a degree of robustness to the uncertainty and modeling errors. Second, by shifting the prediction horizon over the future inputs which are optimized, in fact we compensate the effect of having a finite horizon in our implementation. Note that, it can be shown under the perfectly designed cost and constraints, a receding horizon implementation of MPC have a closed-loop system performance of at least as good as that of the optimal prediction.

When we are dealing with linear systems, similar to our system, the relation between state predictions $x(k)$ and control input $u(k)$ is linear. Therefore, a quadratic predicted cost, similar to what we defined in 2.5, can be written as a sole quadratic function of the control input sequence $u(k)$. Thus, we can rewrite the predicted cost $J(k)$ as a function of u in the following form

$$J(k) = \boldsymbol{u}^T(k) H \boldsymbol{u}(k) + 2f^T \boldsymbol{u}(k) + g \tag{2.7}$$

where H is a constant positive definite (it also can be positive semidefinite) matrix, and f, g are vector and scalar terms, respectively, depending on the state variable vector $x(k)$. Then, the linear input and state constraints enforce linear constraints on $u(k)$ which can be expressed as

$$A_c \boldsymbol{u}(k) \leq b_c \tag{2.8}$$

where A_c is a constant matrix, and b_c is a vector which may be a function of $x(k)$, depending on the form of constraints. Consequently, the MPC optimization problem form the minimization of a quadratic objective cost, subject to a set of linear constraints, over u:

$$\min_u \boldsymbol{u}^T H \boldsymbol{u} + 2f^T \boldsymbol{u}, \qquad s.t. \qquad A_c \boldsymbol{u}(k) \leq b_c \tag{2.9}$$

This type of optimization problem is known as quadratic programming (QP). Given a positive definite matrix H and linear constraints, it is easy to show that this optimization problem is convex, which implies that both the objective function and the constraints are convex with respect to the optimization variable u. Such a convex quadratic programming can be solved efficiently employing well-known specialized algorithms.

However, when we are dealing with a nonlinear model, solving the MPC optimization problem is more difficult compared to the linear model case. The main reason is that due to the nonlinear dependency between the state variables $\boldsymbol{x}(k)$ and control input sequence $\boldsymbol{u}(k)$, the predicted quadratic cost in equation 2.5, which can be denoted by $J(u(k), x(k))$, and the problem constraints, $g(u(k), x(k)) \leq 0$, are, in general, nonconvex functions of control input $u(k)$. That means the optimization problem is a nonconvex nonlinear programming (NLP) problem under this condition. Consequently, one cannot guarantees that the solution converges to a global minimum of the pre-

dicted cost. Moreover, finding a local solution also is quite more complex, computationally, than QP problems of similar size.

The online optimization of MPC is usually solved periodically at discrete time steps $t = kT, k = 0, 1, \ldots$. For each time step k, the optimal control law $u = u^*(k|k)$ is applied to the system until the solution of the optimization at the next time step $t = (k+1)T$ becomes available. Clearly, the control time step T needs to be at least as large as the time which is required to to solve the online optimization. Ideally, T has to be very much larger than the computational time when we do not explicitly incorporate the computation delay in our predictions. Note that, this constraint on the control time step, does not apply to the sampling interval of our discretized state-space model, which is generally selected based on other considerations, such as the bandwidth of the system or frequency of disturbance signals.

Practically, it is feasible to employ a sampling interval, T_{samp}, which is smaller than the control time step T, i.e., $T_{samp} = T/m$. This enable us to apply the previously computed optimal control input sequence at the current time. In this dissertation we assume a discrete-time prediction model with $T_{samp} = T/m$.

2.3.1 Quadratic Programming

A general quadratic programming problem is formulated as a quadratic objective function subjects to a set of linear equality and inequality constraints:

$$\min_x \frac{1}{2} x^T Q x + q^T \quad s.t. \quad Ax = a, \quad Bx \leq b \tag{2.10}$$

The objective function needs to be selected carefully such that the vector q contains all of the linear terms and the matrix Q contains all of the quadratic terms. In other words, Q is the Hessian matrix of the objective function and q represent its gradient.

The matrix equation $Ax = a$ contains all of the linear equality constraints, and $Bx \leq b$ represents the linear inequality constraints. Then, we can write down the following Lagrangian to solve the QP problem:

$$L(x, \lambda, \mu) = \frac{1}{2} x^T Q x + q^T + \lambda^T (Ax - a) + \mu^T (Bx - b) = 0 \tag{2.11}$$

when our objective function is convex, then any local minimum is also the sole global minimum of the object. Note that a function is convex of its Hessian matrix, Q, is positive definite or equivalently, all its eigenvalues are positive. The solutions of this Lagrangian can be tested for optimality by checking if they satisfy the Karush-Kuhn-Tucker (KKT) conditions just as is done for other nonlinear problems:

Condition 1: sum of gradients is zero:

$$\nabla L(x^*, \lambda^*, \mu^*) = \frac{1}{2}x^T Q x + q^T + \lambda^{*T}(Ax - a) + \mu^{*T}(Bx - b) \tag{2.12}$$

Condition 2: all constraints satisfied:

$$Ax^* - a = 0, \qquad Bx^* \leq 0 \tag{2.13}$$

Condition 3: complementary conditions:

$$\mu^T Bx - b = 0 \qquad x^T, \lambda^T, \mu^T \geq 0 \tag{2.14}$$

Since the quadratic programming problems are a special form of nonlinear problems, they can be solved following the same approaches as in other nonlinear programming problems. For an unconstrained quadratic programming problem, it is most easy to find the solution; simply we derive the derivative (gradient) of the objective function and set it to be equal to zero. Then, we find the solutions to this equation. However, when we have a set of constraint to enforce in our optimization problem, we need a more practical approach to solve the constrained nonlinear problem. Conjugate gradient method is a common technique when the objective function is strictly convex and the problem has only equality constraints. However, when we deal with inequality constraints ($Bx \leq b$) as well, the *active set* and *interior point* methods are two possible approaches to find the solution. In addition, when the state variable vector x can takes values inside a range, i.e., $x^L \leq x \leq x^U$, *trust-region* methods are among the most frequently used techniques.

2.4 Hybrid Systems

This section reviews Hybrid Systems, in deterministic and stochastic forms, which is necessary to understand the rest of this chapter. A hybrid system (HS) is a dynamical system with both continuous and discrete behaviors. In other word, it has both continuous flow, expressed by a differential equation, and discrete jumps, described by a state machine or automaton.

A HS is useful in modeling various technological systems, in which continuous physical processes are incorporated with embedded control systems and logic decision making. To show how this type of systems evolve, it is required to combine the dynamics of the continuous parts with the dynamics of the discrete parts using mathematical models. These mathematical models basically comprise some form of differential or difference equations to model the continuous dynamics and discrete-event models for the discrete events.

Most of the different models which have been proposed in literature for hybrid systems share the same model ingredients. In general, every model of a hybrid system has to define at least the following elements:

- $\chi \in R^n$ is the continuous state space.
- Q is the discrete state space, for example $Q = 0, 1, \ldots, q$.
- f is a set of vector fields describing the continuous dynamics for all $qinQ$.
- $Init$ is a set of initial values (q_0, x_0) of the hybrid state.
- δ is the discrete state transition function.
- G is a set of guards determining when a discrete transition occurs.

However, since in most practical systems, uncertainty is inherent, stochastic hybrid system (SHS) models are required to model such systems. In stochastic hybrid systems, at least one of the continuous or discrete components of the state are stochastic processes. The stochastic continuous-state of a SHS is determined by a stochastic differential equation, while the probabilistic discrete-state is evolved by a transition or reset map. Theses transitions are triggered by stochastic events similar to the transitions between states in a continuous-time Markov chains. However, the transitions occur with a rate that may depend on the continuous-state.

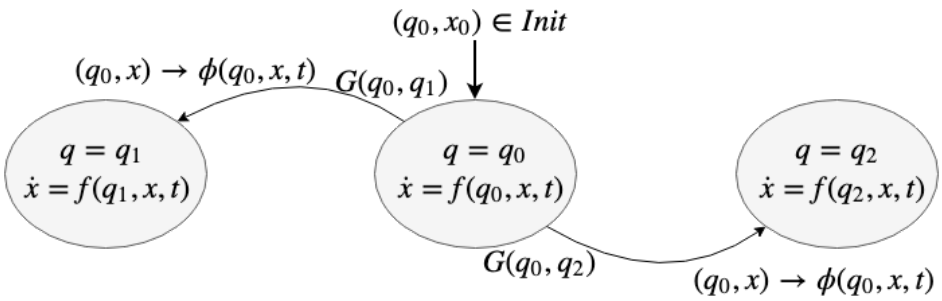


Figure 2.2: Graphical representation of a stochastic hybrid system.

One of the important classes of SHS is known as linear time-triggered SHS (TTSHS), where the continuous state evolves according to a linear dynamical system. Then, some stochastic events are triggered at random discrete times, where the intervals between these random events are independent random variables, and follow a general class of probability distributions.

2.5 System Description

In our framework, which is schematically depicted in Fig. 2.3, the vehicle inside the platoon, which is directly affected by the cut-in suspicious vehicle, is referred to as the host vehicle (see 2.4). The immediate vehicle in front of the host vehicle is known as the preceding vehicle, and the first vehicle of the platoon is the leading vehicle or leader. The dangerous area in front of the host vehicle is referred to as the bad-set. This area and its dimensions will be discussed in details later.

Although, detection of cut-in by the vehicle itself is beneficial to some applications such as lane keep assist system (LKAS) and blind spot warning (BSW), CACC and platooning need the lane change maneuver to be detected remotely by the host vehicle. The remote lane change detection is required because the host vehicle should react in a timely manner to avoid hazardous situations.

V2V communication periodically provides the parameters of the cut-in suspicious vehicles via broadcasting basic safety messages (BSM) [20, 52]. In our model, we assume that the host vehicle, which is in a stable condition in the platoon, periodically receives the BSMs of its surrounding vehicles and continuously traces them prior to any probable cut-in maneuver. From BSM part one of the SAE J2735 standard, [52], we utilize the following parameters for our behavior modeling: latitude, longitude, elevation, speed, heading, steering wheel angle, 4-way acceleration set, and the

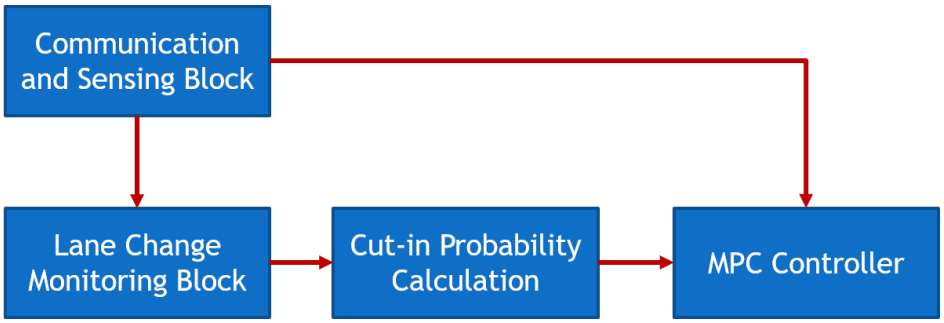


Figure 2.3: The overall schematic of the proposed framework.

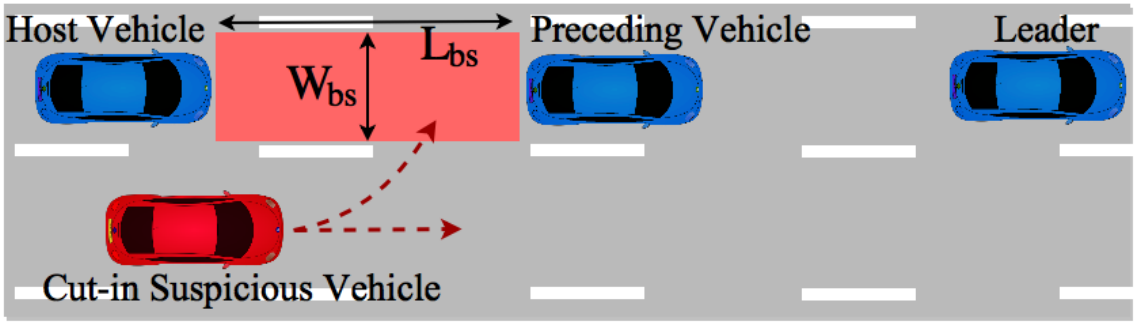


Figure 2.4: Host vehicle, cut-in suspicious vehicle and bad-set

vehicle size. The latitude, longitude, and elevation represent the location of the vehicles center of gravity in the WGS-84 coordinate system. The 4-way acceleration set consists of acceleration values in 3 orthogonal directions plus yaw rate, which are calculated based on the assumption that the front of the vehicle is toward the positive longitudinal axis, right side of it is the positive lateral axis, and clockwise rotation as the positive yaw rate.

In CACC platooning, a safe longitudinal gap must be continuously kept between every two consecutive vehicles. The deviation from the safe gap, which is known as spacing error, should remain as small as possible to reduce the risk of collision and take the advantages of platoon formation, such as lower fuel consumption and higher traffic throughput [53]. As mentioned, we define *bad-set* as the dangerous area in front of the vehicle in which the safe gap is violated. In other words, our bad-set is a rectangle aligned to the road surface in front of the host vehicle, while its longitudinal dimension, L_{bs}, depends on the platoon speed and is equal to the desired longitudinal safe gap and its lateral dimension, W_{bs}, is the lane width. The front bumper of the host vehicle

is always located at the center of the bad-set rear lateral edge. These definitions are illustrated in Fig. 2.4.

The goal of our lane-change monitoring block is tracking and predicting the trajectory of all of the vehicles in the adjacent lanes of the host vehicle. The model should not only predict the immediate kinematics of the vehicles, but also the high-level driving maneuvers. Therefore, the position of neighboring vehicles should be predicted for multiple future steps based on their current and previous communicated information. The number of required prediction steps is determined by the duration of a complete high-level maneuver and denoted by S_m. This multi-step prediction is then used to determine the probability of unsafe lane change which is passed to the SMPC for better estimation of the required inter-vehicle spacing gap.

2.5.1 Lane Change Monitoring Block Design

Each lane change maneuver consists of four separate phases: Intention phase, Preparation phase, Transition phase, and the Completion phase [54, 55]. It is worth mentioning that some more complicated maneuvers, such as overtaking, have also been investigated in the literature. For instance two-phase and five-phase overtake modeling frameworks are proposed in [56] and [57], respectively. The lateral acceleration and lateral speed in a lane change maneuver are bounded by the comfortable lateral acceleration threshold and the maximum tolerable lateral speed, respectively [58]. To safeguard a smooth transition of the vehicle between lanes, the acceleration is bounded by -0.2g and 0.2g [59]. Our model is designed to not only predict the immediate kinematics of the vehicles in the transition phase but also the complete four-phase lane change maneuvers. The trajectory of each remote vehicle is modeled as a time series. In our model, we separate the learning of lateral and longitudinal behaviors of the driver as they are influenced by different control inputs.

Artificial neural networks (ANNs) are one of the most famous tools for description and prediction of nonlinear systems [60, 61]. Neural networks with hidden units can principally predict any well-behaved function. In the case of time series, in order to handle the dependency of the prediction to a finite set of past values and time varying nature of the input signals, neural network topologies need to be equipped with a short term memory mechanism which is called the feedback delay. In this work, we used feedback delay- based ANNs, namely nonlinear autoregressive (NAR), nonlinear autoregressive exogenous (NARX) and recurrent neural networks (RNN) toward driver

behavior and lane change prediction.

NARX is a neural network with feedback delay that can be trained and used to predict a time series from its past values and an exogenous one, compared to NAR which does not rely on any external inputs. We use the NAR model to predict the future pattern of different system inputs, i.e. steering wheel angle, yaw rate, heading, speed, and longitudinal acceleration, based on their currently available values. A NARX model is employed to predict the longitudinal trajectory of the vehicle during the lane change using some of the previously estimated sequences of input signals as the exogenous input. The exogenous inputs in our framework are yaw rate, heading, speed, and longitudinal acceleration.

Finally, an RNN is adopted to model the lateral trajectory of the vehicle based on the predicted input signals. RNNs can use their internal memory to process arbitrary sequences of inputs. The input signals to our lateral position prediction RNN are steering wheel angle, yaw rate, and heading. Using the internal memory, the RNN can distinguish between different maneuvers with partially similar input signals. For example, a steering due to the road curvature might look partially similar to the one from lane change maneuver, but the RNN can learn to distinguish between these two maneuvers by looking at a longer history of the signals or other input signals, such as road curvature. In the former case, the RNN should also be trained on other maneuvers which share the same input signal patterns in a portion of their lifetime.

All of the ANN models (see Figure 2.5) are batch trained and the training phase is offline due to the low computational cost of batch training and insufficient accessible data for online training. In order to use the full capability of neural networks, the input signals for all ANNs are normalized to [-1, 1] range. Then, the input signals are differenced to remove the linearity and improve the nonlinearity prediction process. The resulting time series is known as integrated time series. The value of predicted location can be reconstructed by adding the first actual value to the estimated difference in the series.

To mitigate the effect of noise, small variations of input signals, based on the nature of the signal, are filtered to smooth the time series and mitigate the effect of noise. Variation smaller than 3 degrees, 0.1 rad, 0.1 m/s, and 0.1 m/s^2 are removed from steering wheel angle, heading, speed, and longitudinal acceleration, respectively. The resulting input signals during one maneuver are shown in Fig. 2.6.

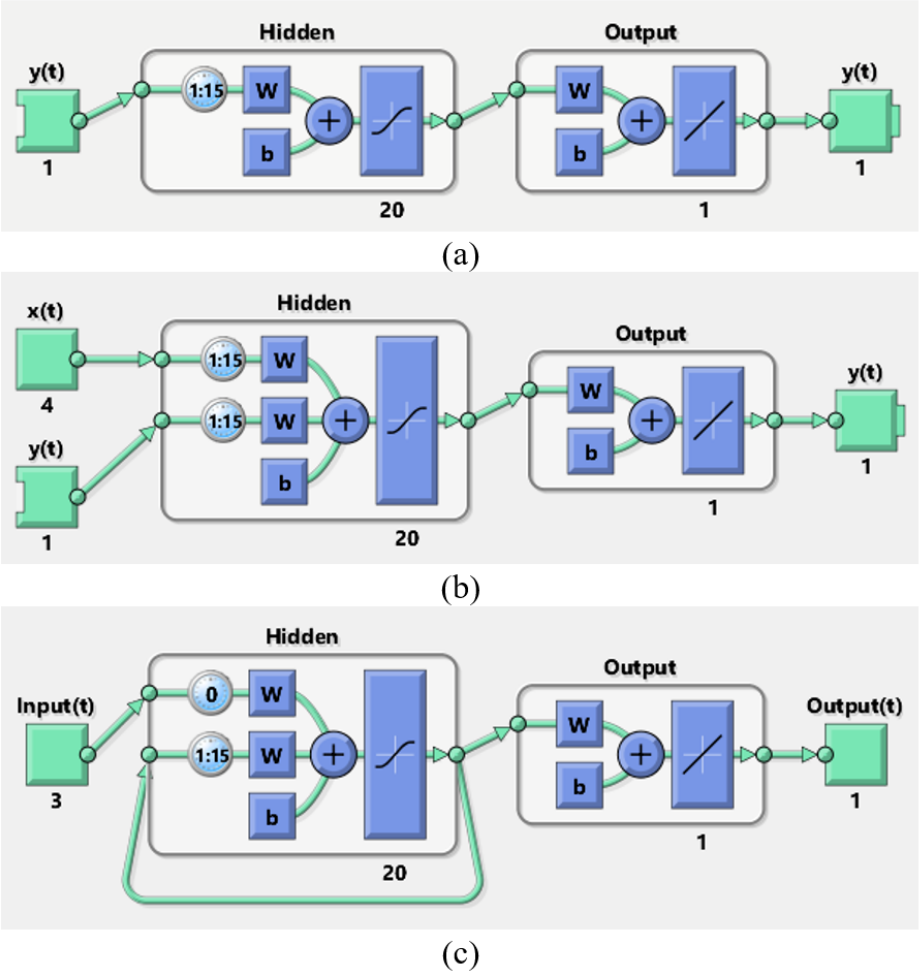


Figure 2.5: Layer structure of (a) NAR (b) NARX (c) RNN

All of our ANNs have a hidden layer with 20 nodes and 15 step short term memory, which means that they are using the past information of 1.5 seconds for future prediction. The required prediction steps are also set to 10 steps for all of the ANNs, $S_m = 10$, which means that the we are predicting the behavior of the driver for 1 second in the future, since the driver can change his decision and behavior beyond this time [62]. As mentioned before, the NAR is used to model the patterns of input signals to the system. The NARX and RNN are used to model and predict the longitudinal and lateral position of the vehicle, respectively. The longitudinal position of the vehicle is modeled based on the predicted values of heading, speed, and longitudinal acceleration as external inputs. On the other hand, the RNN should not only predict the future lateral position of the vehicle, but should also distinguish between different lateral maneuvers. Therefore, the lateral

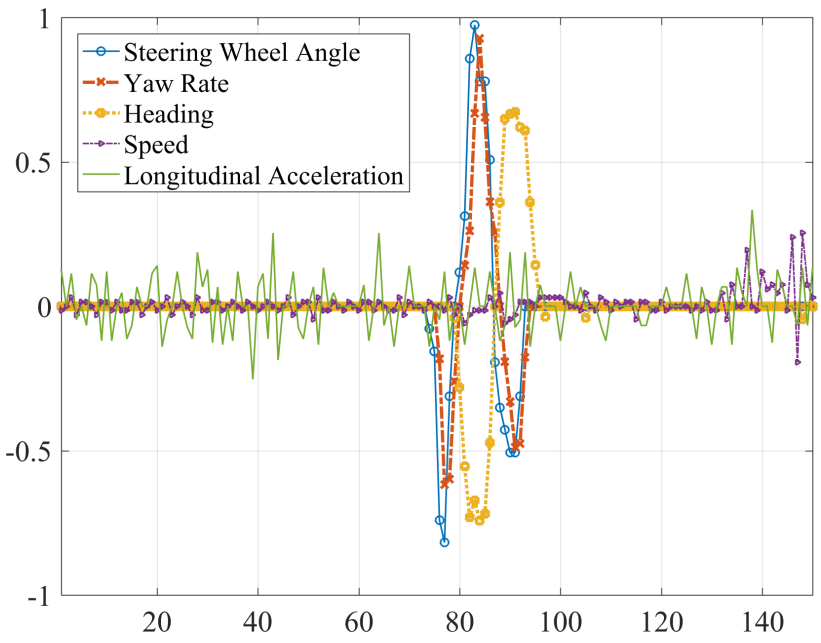


Figure 2.6: Smoothed, Normalized, and Integrated input signals of a single lane change maneuver

model is also trained with some road curve data to be able to differentiate between different lateral movements.

2.5.2 Cut-in Probability Calculation

The results of the proposed cut-in prediction scheme is now applied to find a single value between 0 and 1 which represents the overall cut-in probability. This probability, which is denoted by P_c from now on, will be fed to our SMPC as its input. SMPC design details are discussed in the following subsection. At each prediction cycle we have S_m predicted future values for each of the longitudinal and lateral relative positions of the suspicious cut-in vehicle. In our implementation, each of these $2 \times S_m$ predicted values comes with a specific 90 percent confidence level. Hence, we have S_m rectangular areas, each of them determines the predicted area for the position of the cut-in vehicle in the corresponding upcoming time step with 90 percent accuracy. We take the most conservative approach to define the cut-in probability, P_c as follows:

- Each of these S_m rectangles, (A in Fig. 2.7), is intersected with the host vehicle's bad-set at that moment and its intersection area, (A_1 in Fig. 2.7), is calculated.

- The resultant intersection area, (A_1), is normalized by dividing it by the corresponding

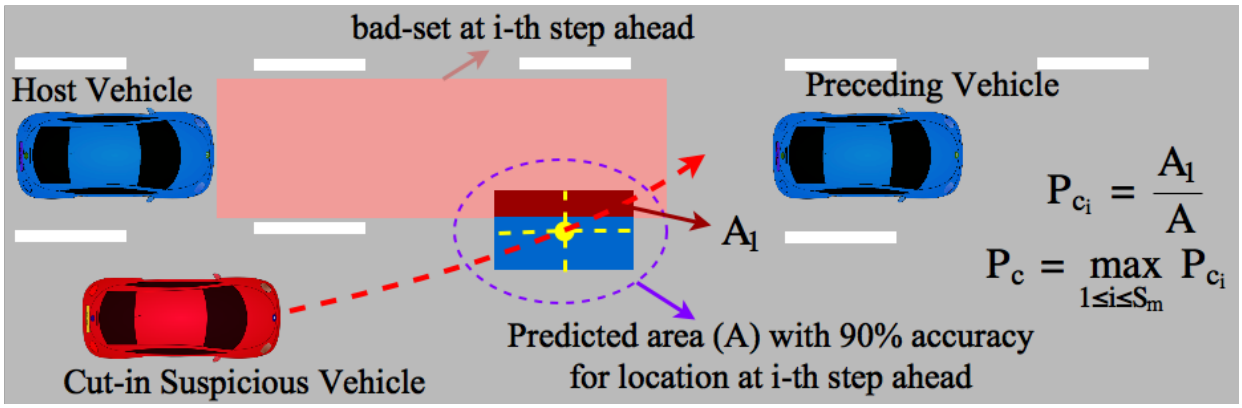


Figure 2.7: Procedure of cut-in probability (P_c) calculation

predicted area value, (A), to calculate the probability value of being inside the bad-set for each of these predictions.

- The maximum value amongst these S_m probabilities is selected as the P_c value for that prediction cycle.

For more clarification, this procedure for the i^{th} step prediction, ($1 \le i \le S_m$), is depicted in Fig. 2.7. In this dissertation, one second ahead prediction is targeted which is equivalent to $S_m = 10$, due to the DSRC baseline information broadcasting frequency (10 Hz). It is worth mentioning that this frequency could be easily supported by most of the currently available commercial GPSs like what is used in this work's dataset [63].

2.5.3 CACC Model Predictive Controller Design

Considering a CACC platoon of vehicles, the spacing error of the i^{th} following vehicle is defined as follows [59]:

$$\delta_i = x_{i-1} - x_i - hv_i - L_i - d_0 \tag{2.15}$$

for all $i \in \{1, 2, \ldots, n\}$, where x_i and v_i are longitudinal position and velocity of the i^{th} following vehicle, respectively (x_0 stands for the longitudinal position of the lead vehicle); h headway which introduces a speed dependent spacing policy in addition to d_0 which is a constant minimum desired distance between each vehicle and its preceding vehicle in the platoon, and L_i is the length of the

i^{th} vehicle. Based on these definitions, longitudinal dimension of the bad-set, L_{bs}, for i^{th} vehicle could be represented as:

$$L_{bs} = hv_i + d_0 \tag{2.16}$$

Then, the dynamics of the i^{th} following vehicle in the platoon is modeled as follows [59]:

$$\begin{aligned} \dot{\delta}_i &= v_{i-1} - v_i - h\dot{v}_i \\ \Delta\dot{v}_i &= a_{i-1} - a_i \\ \dot{a}_i &= f_i(v_i, a_i) + g_i(v_i)c_i \end{aligned} \tag{2.17}$$

where c_i is the control input of the vehicle, i.e. engine/brake, with $c_i \geq 0$ represents the engine throttle input and the $c_i < 0$ is the brake input. Moreover, f_i and g_i are given by

$$\begin{aligned} f_i(v_i, a_i) &= -\frac{1}{\tau_i}(\dot{v}_i + \frac{\sigma A_i c_{di}}{2m_i} v_i^2 + \frac{d_{mi}}{m_i}) - \frac{\sigma A_i c_{di} v_i a_i}{m_i} \\ g_i(v_i) &= \frac{1}{\tau_i m_i} \end{aligned}$$

where m_i is the vehicle mass, σ is the air specific mass, A_i is the cross-sectional area, d_{mi} is the mechanical drag, c_{di} is the drag coefficient, τ_i is the engine time constant, and $\frac{\sigma A_i c_{di}}{2m_i}$ is the air resistance. We adopt the following control law from [59]:

$$c_i = u_i m_i + \sigma A_i c_{di} v_i^2/2 + d_{mi} + \tau_i \sigma A_i c_{di} v_i a_i \tag{2.18}$$

where u_i is the control input that we need to design. Then we can rewrite the system as:

$$\dot{\delta}_i = v_{i-1} - v_i - h\dot{v}_i \tag{2.19}$$

$$\Delta\dot{v}_i = a_{i-1} - a_i \tag{2.20}$$

$$\dot{a}_i = -\frac{a_i}{\zeta_i} + \frac{u_i}{\zeta_i} \tag{2.21}$$

where ζ_i is the engine time constant, a_i is the acceleration of the i^{th} vehicle, and u_i is an input signal which comes from an MPC controller. However, due to the communication delay each vehicle receives the delayed version of its preceding vehicle's acceleration value. Denoting the

communicated acceleration of i^{th} vehicle at receivers by $\bar{a}_i(t)$, the state space equation of a vehicle in a CACC system could be represented as follows

$$\dot{x}_i(t) = A_i x_i(t) + B_i u_i(t) + G_i \bar{a}_{i-1}(t) \tag{2.22}$$

with state vector $x = [\delta_i \quad \Delta v_i \quad a_i]^T$, and

$$A_i = \begin{bmatrix} 0 & 1 & -h \\ 0 & 0 & -1 \\ 0 & 0 & -\frac{1}{\zeta_i} \end{bmatrix} \quad B_i = \begin{bmatrix} 0 \\ 0 \\ -\frac{1}{\zeta_i} \end{bmatrix} \quad G_i = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \tag{2.23}$$

However, delay of the communication network is not considered in this work, so $\bar{a}_{i-1}(t) = a_{i-1}(t)$.

An MPC controller with three primary objectives is required to control the platoon system described by (2.22). The controller must compute the input signal u_i to minimize the spacing error, keep the velocity of the host vehicle as close as possible to its preceding vehicle velocity, and finally, respond appropriately to a cut-in vehicle based on our prediction of the driver behavior. To this end, the system dynamics (2.22) is discretized and an optimal control problem, which satisfies the aforementioned control goals, is defined. The continuous time dynamics of the system is discretized using the Euler forward method with a time step T_s:

$$\dot{x}_i[k+1] = A_i^k x_i[k] + B_i^k u_i[k] + G_i^k \bar{a}_{i-1}[k] \tag{2.24}$$

where

$$A_i^k = \begin{bmatrix} 1 & T_s & -hT_s \\ 0 & 1 & -T_s \\ 0 & 0 & 1 - \frac{T_s}{\zeta_i} \end{bmatrix} \quad B_i^k = \begin{bmatrix} 0 \\ 0 \\ -\frac{T_s}{\zeta_i} \end{bmatrix} \quad G_i^k = \begin{bmatrix} 0 \\ T_s \\ 0 \end{bmatrix} \tag{2.25}$$

Hereinafter, for simplicity of notation, we use A, B, G, x, and u instead of A_i^k, B_i^k, G_i^k, x_i, and u_i, respectively. The cost function of the optimal control problem is defined based on the primary

objectives of the controller:

$$J[k] = \sum_{i=0}^{N-1} c_\delta \delta^2[k+i] + c_v \Delta v^2[k+i] + c_u \Delta u^2[k+i] \tag{2.26}$$
$$= \sum_{i=0}^{N-1} [x^T[k+i|k] Q x[k+i|k] + u^T[k+i|k] R u[k+i|k]]$$

where N is the control horizon, c_δ, c_v, and c_u are weighting coefficients reflecting the relative importance of each term and

$$\Delta u[k+n] = u[k+n] - u[k+n-1] \tag{2.27}$$

which is added to the cost function as an extra term to bound the jerk and prevent fast variations of the input signal. This constraint could be interpreted as comfort ride. The MPC law finds the optimal input sequence $\boldsymbol{u}^*[k]$ which minimizes the predicted cost function (2.26) at each time instant:

$$\boldsymbol{u}^*[k] = \arg\min_u J[k] \tag{2.28}$$
$$\text{subject to} \quad \begin{cases} x_{min} \le x[k+i|k] \le x_{max} \\ u_{min} \le u[k+i|k] \le u_{max} \end{cases} \quad i = 1, \ldots, N-1$$

To solve this MPC problem, the future values of the preceding vehicle's acceleration are required. These values are obtained from the aforementioned NAR neural network.

Conventional MPC Design

In this section, MPC design problem without incorporating the calculated cut-in probability, P_c, is investigated. We referred to this MPC design as conventional design in this dissertation. The values of $a_{i-1}[k]$ could be considered as a measured disturbance when its model is available to the MPC controller. Then, the system equations could be rewritten in the standard form as

$$\bar{\boldsymbol{x}}[k+1] = \bar{A}\bar{\boldsymbol{x}}[k] + \bar{B}u[k] \tag{2.29}$$

where $\bar{\boldsymbol{x}}[k] = \big[\boldsymbol{x}[k], \boldsymbol{z}[k]\big]^T$ is the augmented state vector and the measured disturbance state vector $\boldsymbol{z}[k] = \big[z_0, z_1, \dots z_{N-1}\big]$ is defined as

$$\begin{aligned} z_0[k+1] &= z_1[k] \\ z_1[k+1] &= z_2[k] \\ &\vdots \\ z_{N-2}[k+1] &= z_{N-1}[k] \\ z_{N-1}[k+1] &= z_{N-1}[k] \end{aligned} \qquad \begin{cases} z_0[0] = a_{i-1}[0] \\ z_1[0] = a_{i-1}[1] \\ \vdots \\ z_{N-2}[0] = a_{i-1}[N-2] \\ z_{N-1}[0] = a_{i-1}[N-1] \end{cases}$$

In the receding horizon implementation of the MPC problem (2.28), only the first element of the optimal input sequence $u^*[k]$ is selected as the input to the system and the whole process is repeated at each time step. However, designing a receding horizon controller based on a finite-horizon cost function does not guarantee the stability and optimality of the closed loop system [64]. This problem can be avoided by defining an infinite prediction horizon for the cost function:

$$J[k] = \sum_{i=0}^{\infty} [x^T[k+i|k]Qx[k+i|k] + u^T[k+i|k]Ru[k+i|k]] \tag{2.30}$$

However, to have finite number of variables in the MPC optimization problem, a dual-mode prediction approach can be utilized in which the predicted input sequence is defined as

$$u[k+i|k] = \begin{cases} \boldsymbol{u}^*[k+i|k] & i = 0, 1, ..., N-1 \\ Kx[k+i|k] & i = N, N+1, ... \end{cases} \tag{2.31}$$

Then, by choosing a terminal weighting matrix, denoted by $\bar{Q}$, in a way that $x^T[k+N|k]\bar{Q}x[k+N|k]$ is equal to the cost over the second mode of the predicted input sequence, the infinite cost J can be rewritten as (see Appendix A)

$$\begin{aligned} J[k] = &\sum_{i=0}^{N-1} [x^T[k+i|k]Qx[k+i|k] \\ &+ u^T[k+i|k]Ru[k+i|k]] + x^T[k+N|k]\bar{Q}x[k+N|k] \end{aligned} \tag{2.32}$$

Theorem 1 ***(Stability)*** *The state variables of system (2.29), $x[k]$, asymptotically converge to zero, i.e. the system is asymptotically stable, under the control law (2.31) if predicted cost $J[k]$ is*

an infinite cost, (A, Q^{12}) *is observable, and the tail* $\tilde{\boldsymbol{u}}[k]$ *is feasible for all* $k > 0$ *where*

$$\tilde{\boldsymbol{u}}[k+1] = \left[\boldsymbol{u}^*[k+1|k], \ldots, K\boldsymbol{x}^*[k+N|k]\right] \tag{2.33}$$

Proof: *see Appendix 6.*

Therefore, selecting Q and $\bar{Q}$ which satisfy the first two conditions, the stability and convergence of the closed loop system rely on the assumption that tail $\tilde{\boldsymbol{u}}[k]$ is feasible for all $k > 0$. To this end, a set of extra constraints on the state vector should be satisfied at each time instant k. These extra constraints that are introduced to enforce the feasibility of the tail are known as the terminal constraints since they apply to the second mode of the dual mode prediction approach.

Theorem 2 ***(Recursive feasibility)*** *The MPC optimization (2.28) with the cost function* $J[k]$ *defined in (2.32) is guaranteed to be feasible at all time* $k > 0$ *if a new constraint* $\boldsymbol{x}[k+N|k] \in \Omega$ *is met, provided it is feasible at* $k = 0$*, and terminal constraint set* Ω *(see Figure 2.8) satisfies*

- *The following constraints are satisfied for all points in* Ω*, (i.e.* $\boldsymbol{x}[k+N|k] \in \Omega$*)*

$$\begin{cases} u_{min} \leq K\boldsymbol{x}[k+N|k] \leq u_{max} \\ \boldsymbol{x}_{min} \leq \boldsymbol{x}[k+N|k] \leq \boldsymbol{x}_{max} \end{cases} \tag{2.34}$$

- Ω *is invariant in the second mode of (2.31) which means*

$$\boldsymbol{x}[k+N|k] \in \Omega \quad \Rightarrow \quad (A+BK)x[k+N|k] \in \Omega \tag{2.35}$$

It is shown that the largest possible Ω is derived by

$$\begin{aligned} \Omega = \{\boldsymbol{x} : \quad & u_{min} \leq K(A+BK)^i\boldsymbol{x} \leq u_{max}, \\ & \boldsymbol{x}_{min} \leq (A+BK)^i\boldsymbol{x} \leq \boldsymbol{x}_{max}, \quad i = 0, 1, \ldots\} \end{aligned} \tag{2.36}$$

To show that Ω is invariant over the infinite horizon of the second mode of (2.31), constraint satisfaction should be checked over a long enough finite horizon (N_c). Here, N_c is the smallest

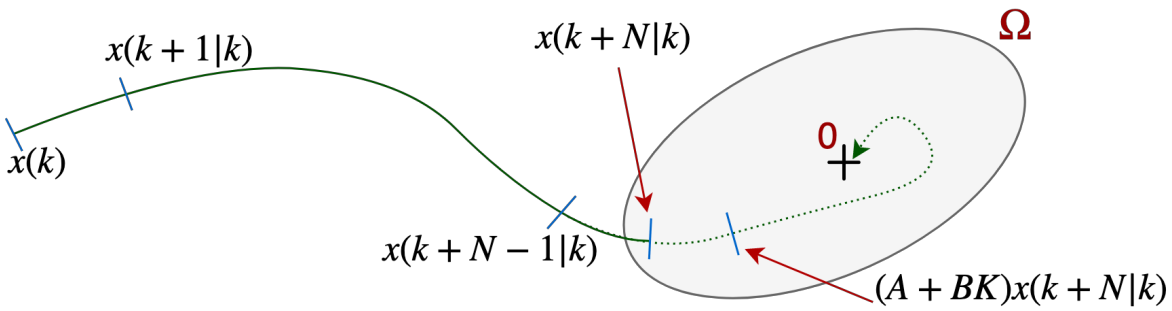


Figure 2.8: The terminal constraint set Ω.

number which satisfies the following equations

$$\begin{cases} \underline{u} = \min_x K(A+BK)^{N_c+1}\boldsymbol{x} \\ \overline{u} = \max_x K(A+BK)^{N_c+1}\boldsymbol{x} \end{cases} \tag{2.37}$$

such that

$$\begin{cases} u_{min} \leq K(A+BK)^i\boldsymbol{x} \leq u_{max} \quad i = 0,\ldots,N_c \\ u_{min} \leq \underline{u} \leq u_{max} \end{cases} \tag{2.38}$$

The algorithm of finding N_c is summarized in Figure 2.9. Having N_c found, adding the following constraints to the MPC problem guarantees the feasibility of the controller:

$$\begin{aligned} u_{min} \leq K(A+BK)^i\boldsymbol{x}[k+N|k] \leq u_{max}, \\ x_{min} \leq (A+BK)^i\boldsymbol{x}[k+N|k] \leq x_{max}, \\ i = 0,1,\ldots,N_c \end{aligned} \tag{2.39}$$

So far, we have designed an MPC controller for a CACC platoon and found the conditions to guarantee the stability and feasibility of the system. However, our main goal is to handle a possible cut-in scenario. Figure 2.10 shows how the designed controller react to a possible cut-in. We claim that if we expect a cut-in to happen in advance, then the controller can do better than Figure 2.10.

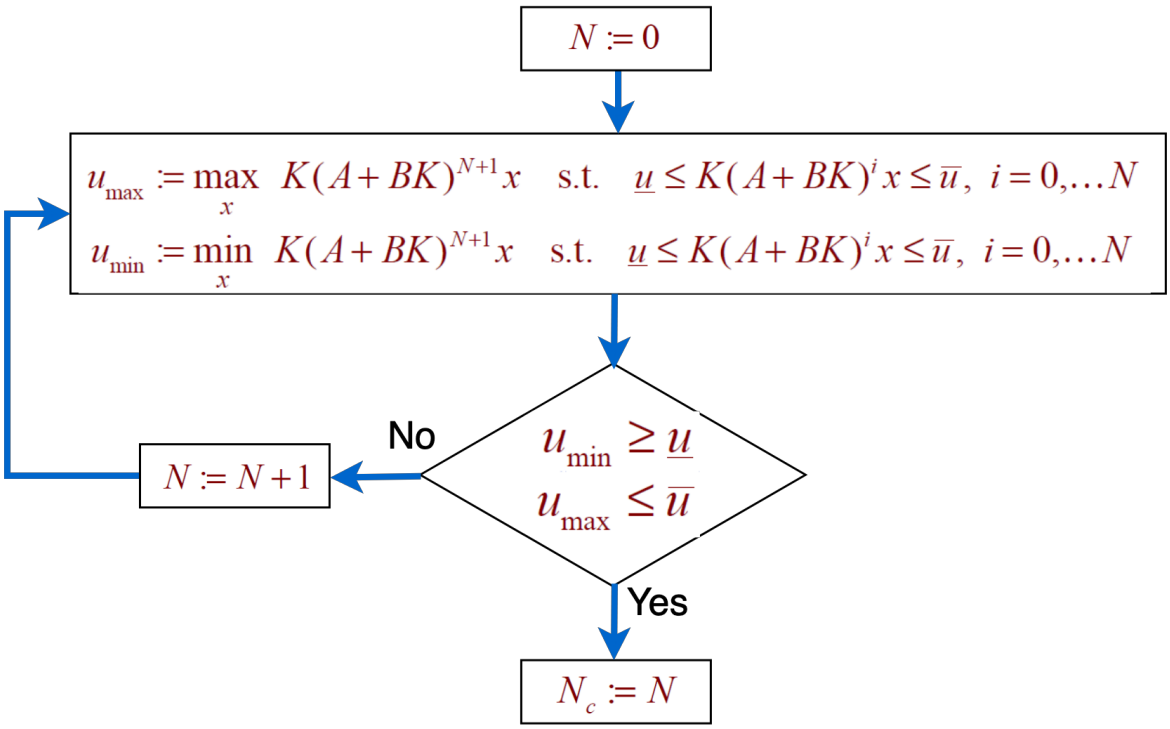


Figure 2.9: Algorithm for computing the mode 2 constraint checking horizon N_c.

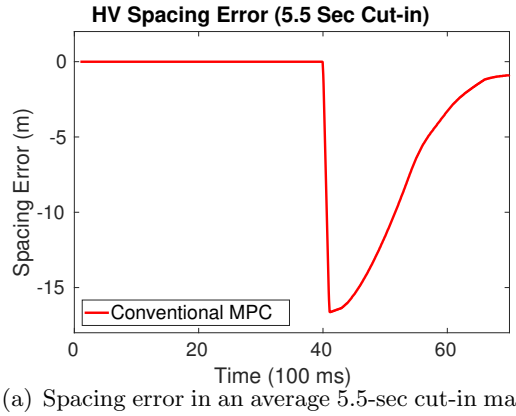
MPC design: Incorporating cut-in probability

The designed MPC controller in the previous section satisfies our first two primary goals, namely spacing error and velocity error minimization. However, the controller should be able to react appropriately if a cut-in suspicious vehicle enters the platoon unexpectedly and pushes the host vehicle to decelerate to reestablish the safe distance.

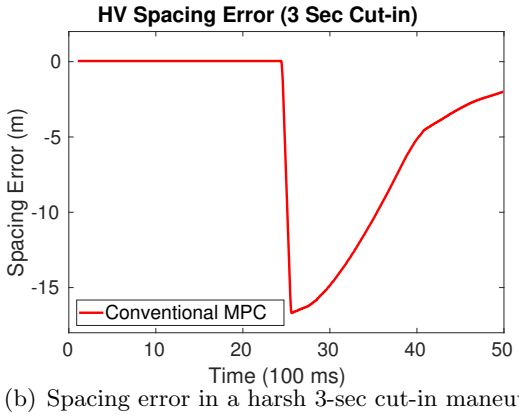
Heretofore, the probability of the suspicious vehicle's cut-in trajectory intersection with the host vehicle's bad-set has been determined. Based on this, we propose a new stochastic definition for the spacing error:

$$\delta_i = \frac{x_{i-1} - x_i}{2 - e^{-\alpha P_c}} - hv_i - L_i - d_0 \tag{2.40}$$

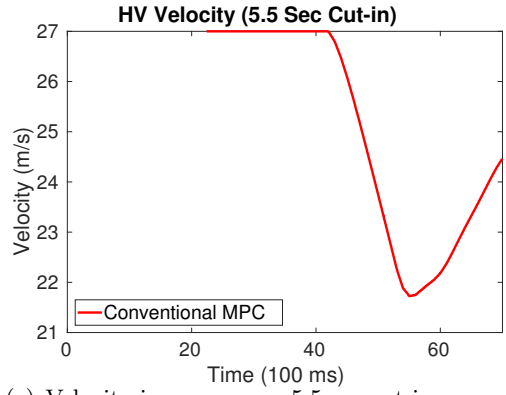
where P_c is the probability of the cut-in vehicle being in the bad- set of the host vehicle, and is a constant control parameter which adjusts the reaction sensitivity of the MPC controller to the cut-in probability. Clearly, when the probability is one, assuming α has been set to a sufficiently large



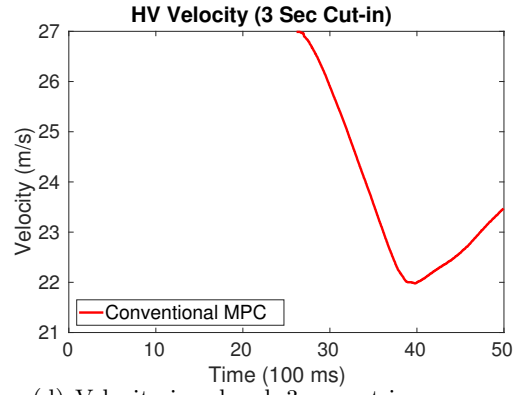
(a) Spacing error in an average 5.5-sec cut-in maneuver



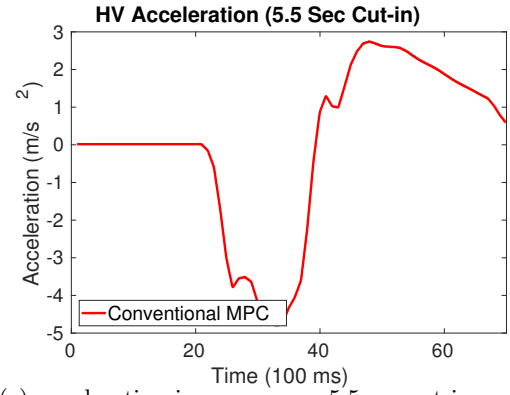
(b) Spacing error in a harsh 3-sec cut-in maneuver



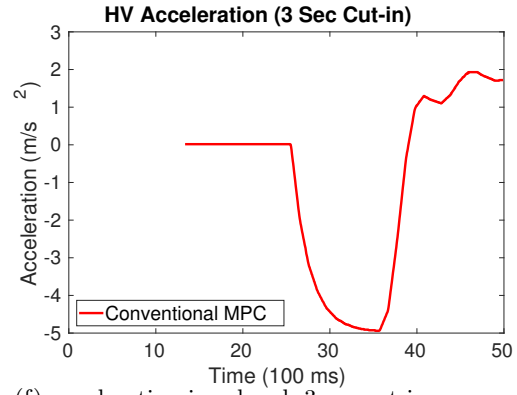
(c) Velocity in an average 5.5-sec cut-in maneuver



(d) Velocity in a harsh 3-sec cut-in maneuver



(e) acceleration in an average 5.5-sec cut-in maneuver



(f) acceleration in a harsh 3-sec cut-in maneuver

Figure 2.10: Spacing error, velocity, and acceleration of the designed conventional MPC

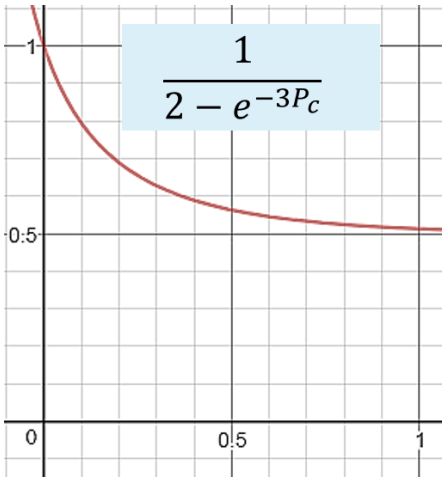


Figure 2.11: The factor goes to 0.5 when the cut-in probability, P_c, goes to one. Consequently, the controller underestimate the distance to the preceding vehicle and doubles the distance which provides enough space for the suspicious vehicle to joint the platoon.

number, the controller starts doubling the distance from its current preceding vehicle by halving the enumerator of the first term in the proposed equation for spacing error, (2.40). Consequently, the cut-in vehicle has enough safe gap to enter the CACC platoon. On the contrary, when the probability is zero, the suspicious vehicle is not expected to cut in or it has the safe distance from the host vehicle for its maneuver. This zero probability sets the denominator of (2.40) to one, which means the host vehicle keeps the normal safe distance, $hv_i + d_0$, from its preceding vehicle.

Although the stability and feasibility of the controller is already guaranteed for the MPC design with deterministic spacing error, it should be proved under the new circumstances due to the stochastic spacing error definition. However, in this work, we reformulate the stochastic MPC problem into a stochastic hybrid system and a robust MPC design. To this end, a Time-Triggered Stochastic Impulsive System (TTSIS) SHS model, [31], incorporating the probability of an upcoming cut, is utilized as shown in Fig. (2.12).

In this model, $p \in [0, 1]$ is a random variable which represents the cut-in probability, $G(p)$ is a guard condition that must be hold for the discrete transition (time trigger), and $R(x, p)$ is a reset function which describes the changes in the continuous states after the transition. The stochastic nature of our system comes from the dependency of the guard on the random variable p.

Proposition 1 *The TTSIS of Fig (2.12) is stable under the designed MPC controller if the fol-*

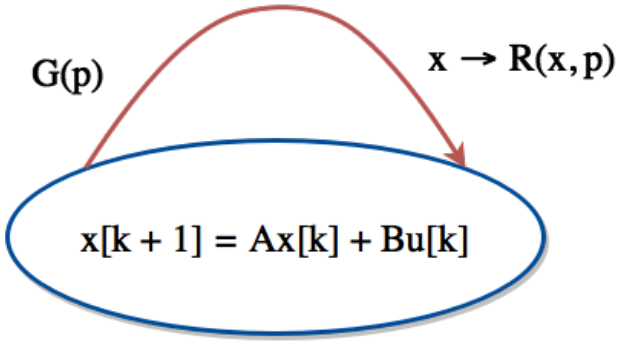


Figure 2.12: A hybrid model for the system incorporating the cut in probability

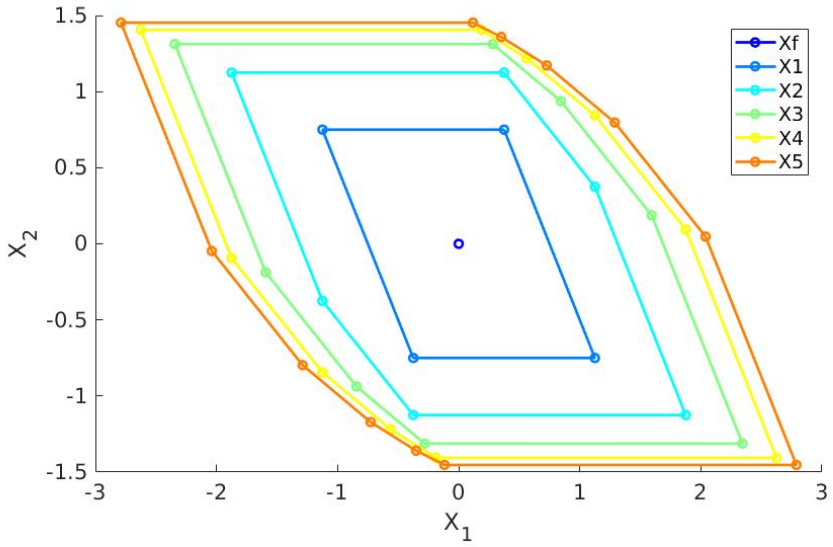


Figure 2.13: Region of attraction of MPC.

lowing condition is satisfied:

- *There is a finite invariant region $\mathcal{S}_R$ in which*

$$\begin{aligned} x[k] \in \mathcal{S}_R \quad &\rightarrow \quad R(x[k], p) \in \mathcal{S}_R \\ \forall p \in [0, 1]. \quad &x_{min} \leq x[k] \leq x_{max} \end{aligned} \tag{2.41}$$

The region $\mathcal{S}_R$ is a subset of the region of attraction for the MPC law, denoted by $\mathcal{S}_\Omega$ ($\mathcal{S}_R \subset \mathcal{S}_\Omega$). Here, $\mathcal{S}_\Omega$ is defined as the set of all initial states from which a sequence of inputs exists that forces the state predictions to reach the terminal constraint set Ω in the first mode of control law (2.31),

i.e.

$$\mathcal{S}_\Omega = \begin{cases} x[0]: & \exists \boldsymbol{u}[0], \quad \boldsymbol{x}[N|0] \in \Omega, \\ \text{s.t.} & \begin{cases} x_{min} \leq x[i|0] \leq x_{max}, & \forall i \leq N \\ u_{min} \leq u[i|0] \leq u_{max}, & \forall i \leq N-1 \end{cases} \end{cases} \tag{2.42}$$

Figure 2.13 shows the region of attractions for different values of control horizon N for a to example. Here, the x_1 represents the terminal constraint set, and x_i represent $\mathcal{S}_\Omega$ for $N = i$. This plot is for a system with $A = [2, 1; 0, 2]$, $B = [1, 0; 0, 1]$, $Q = [1e-5, 0; 0, 1e-5]$, $R = [1, 0; 0, 1]$, $x_{min} = [-5; -5]$, $x_{max} = [5; 5]$, $u_{min} = [-1.5; -1.5]$, and $u_{max} = [1.5; 1.5]$.

Note that since our system has 13 state variables, its region of attraction cannot be visualized. Therefore, to guarantee the stability of the system over a larger set of conditions, the constraints on the system states should be relieved as much as the safety is not violated. To this end, we set the constraint on the spacing error to $[\delta_{min}, \delta_{max}] \in (-hv_i - L_i - d_0 + \delta_s, +\infty)$ for the case of no cut-in detected, where δ_s is the minimum desired safe gap between the vehicles. Clearly, after each discrete transition, the value of spacing error can only jump with a value between $-hv_i - L_i - d_0$ and $hv_i + L_i + d_0$. However, for the case of a positive cut-in probability, we should choose a more conservative constraint, $[\delta_{min}, \delta_{max}] \in (-hv_i - L_i - d_0 + (x_{i-1} - x_{rv}) + \delta_s, +\infty)$ to assure the collision avoidance where x_{rv} is the position of the cut-in suspicious vehicle. Finally, if the is no constraint found which can guarantee the feasibility, the driving situation is considered as an unsafe or a harsh maneuver and the controller temporarily is overwritten with u_i set to the maximum possible deceleration (usually up to $-10m/s^2$ [65], to prevent the collision) or the maximum pre-defined acceleration of the vehicle till the feasibility of the controller can be guaranteed again.

2.6 Evaluation

In this section overall performance of the proposed MPC control design is evaluated using realistic driving scenarios from Safety Pilot Model Deployment (SPMD) dataset [63]. The practicality of our cut-in trajectory prediction method is also shown by its performance comparison versus the kinematic-based trajectory prediction as a ground truth.

A general cut-in maneuver duration is between 3.5-6.5 seconds for urban scenarios with the mean of 5 seconds and 3.5-8.5 seconds for highway scenarios with the mean of 5.8 seconds [66].

For the sake of comparison fairness, two types of cut-in maneuvers, i.e. the harshest type with 3.5 seconds duration, and the average class with 5.5 seconds duration, have been selected. In both cases, the platoon velocity is assumed 27 m/s or 60 mph.

2.6.1 Conventional MPC Performance Evaluation

In this section overall performance of the proposed system framework is evaluated using realistic driving scenarios from Safety Pilot Model Deployment (SPMD) dataset [63]. First, the practicality of our cut-in trajectory prediction method is shown by its performance comparison versus the kinematic-based trajectory prediction as a ground truth. Next, noticeable better behavior of the proposed SMPC controller versus the conventional MPC is discussed.

2.6.2 Cut-in Trajectory Prediction Performance Evaluation

To evaluate the performance of our method, we extracted 90 lane change maneuvers from the BSMs generated by participating vehicles in SPMD dataset in Ann Arbor, Michigan. BSM broadcast rate had been set to 10Hz in this dataset. Therefore, we have all of the time series recorded in this rate. The signals from all 90 maneuvers are concatenated to create a long univariate time series for each input signal. Finally, a 70-15-15 percent training, cross-validation, and testing data selection is used for ANNs training and performance evaluation.

Combination of the longitudinal and lateral predictions, for one second ahead of a cut-in scenario from SPMD dataset, is depicted in Figure 2.15, for more clarification. In this figure, prediction errors are shown with consecutive rectangles on the predicted path. The performance of the prediction methods, i.e. vehicle kinematics model and our model (trained RNN and NARX), are compared using their 90 percentile accurate predictions for each of the 10 prediction steps.

The performance of two trajectory prediction methods, vehicle kinematic model, and our trained RNN, for a lane changing vehicle in terms of lateral confidence levels at each time step ahead, averaged on all 90 scenarios, are shown in Fig. 2.14(a). The classic car model could be represented by kinematic-based differential equations as follows:

$$\dot{x}_i = v_i \cos\theta_i, \qquad \dot{y}_i = v_i \sin\theta_i, \qquad \dot{\theta}_i = \frac{v_i}{L_i} \tan\phi_i \tag{2.43}$$

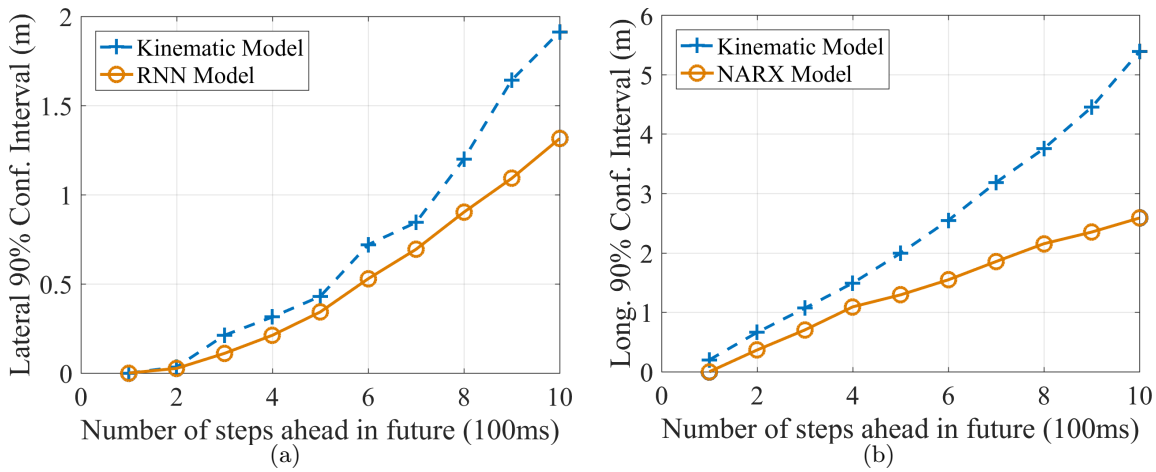


Figure 2.14: Comparison of 90-percentile conf. interval of the Kinematic and RNN models for different prediction steps. (a) Lateral and (b) Longitudinal predictions

where x_i, y_i, and v_i are longitudinal position, lateral position, and velocity of the i^{th} vehicle, respectively. Also, ϕ_i denotes the steering angle, θ_i stands for the angel between the vehicle's instantaneous heading and the road direction, and $L_i = 5$ is the length of the vehicle [67].

Fig. 2.14(b) shows the same comparison for the longitudinal position prediction.

2.6.3 Designed SMPC Performance Evaluation

In this section, superiority of designed SMPC versus conventional MPC is investigated. To this end, reactions of these two different designs to real cut-in maneuvers should be compared. A general cut-in maneuver duration is between 3.5-6.5 seconds for urban scenarios with the mean of 5 seconds and 3.5-8.5 seconds for highway scenarios with the mean of 5.8 seconds [66]. For the sake of comparison fairness, two types of cut-in maneuvers, i.e. the harshest type with 3.5 seconds duration, and the average class with 5.5 seconds duration, have been selected and the outputs of two aforementioned controllers are compared in each case. In both cases, the platoon velocity is assumed 27 m/s or 60 mph.

Cut-in probabilities, calculated based on the discussed method in section 2.5.2, for average and harsh maneuvers are depicted in Fig. 2.16. As mentioned before, these probabilities are fed into our SMPC controller at each prediction cycle. The vertical dashed lines in both figures stand for the moments at which the cut-in vehicles cross the road line between two adjacent lanes. It is clear that our cut-in detection starts around 1.25 seconds and 2 seconds ahead of this moment for

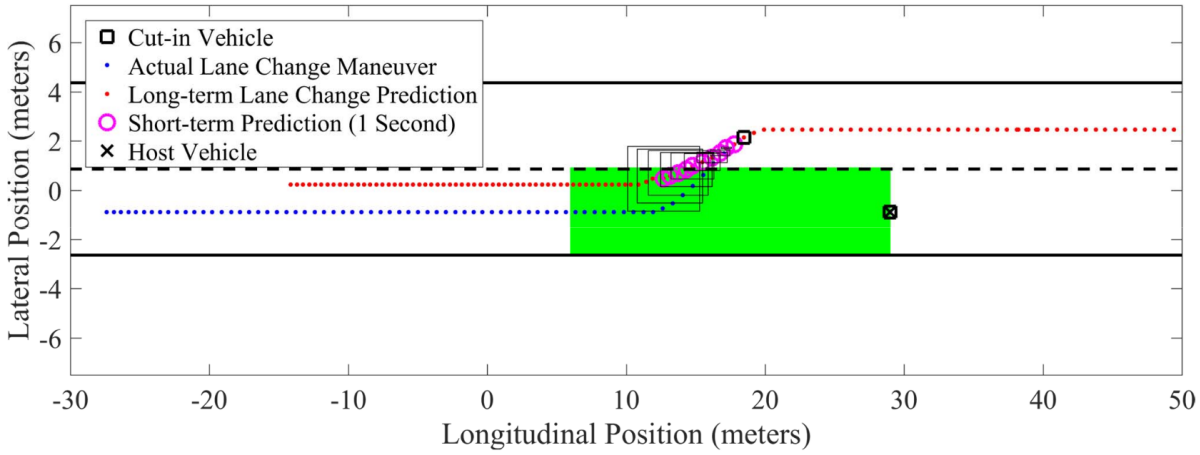


Figure 2.15: Joint perspective of longitudinal and lateral predictions

harsh and average maneuvers, respectively, which provides a noticeable extra reaction time for the controller.

Fig. 2.17, illustrates the changes in spacing error, velocity and acceleration of the host vehicle produced by two different controllers in response to the average cut-in maneuver. It is noteworthy that our controller starts its reaction to compensate the situation notably sooner than the conventional one which results in a noticeable smoother reaction. In addition, it highly increases the reaction safety as a consequence of its considerable lower maximum spacing error which is evident by comparing the spacing error in Figs. 2.17(a) and 2.17(b). For instance, as it is clear in Fig. 2.17(a), the worst SMPC spacing error reaches 10 meters, while its counterpart in conventional MPC system is around 17 meters. Moreover, the spacing error of the SMPC controller is around 6 meters when the suspicious vehicle entered the CACC lane while in conventional MPC it is on its maximum value of 17 meters. Finally, the SMPC cut-in detection starts around 2 seconds from the beginning of the scenario, which gives the controller about 2 seconds additional reaction time in comparison with the conventional system.

The same plots for harsh maneuver, which are depicted in Figs. 2.17(b), 2.17(d), and 2.17(f), demonstrate the dominance of the proposed SMPC performance in terms of sooner and safer reaction.

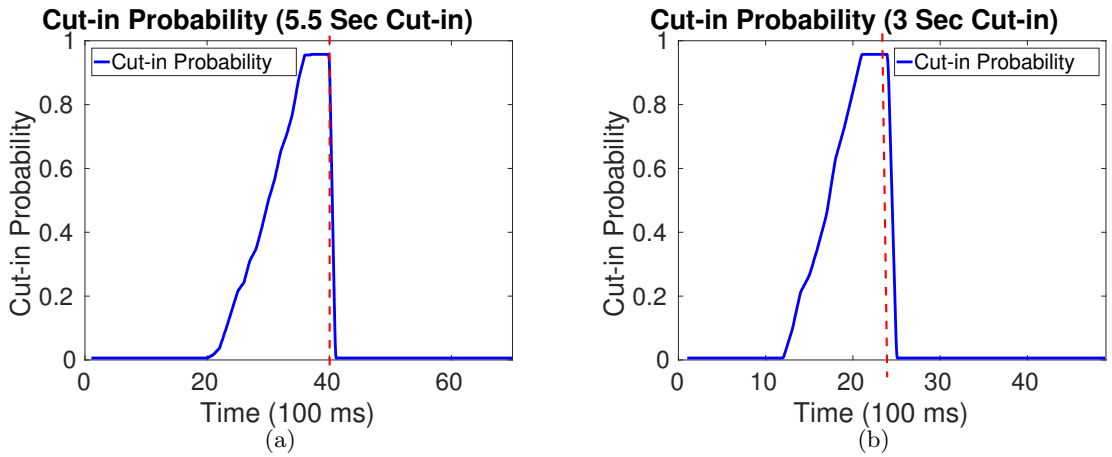
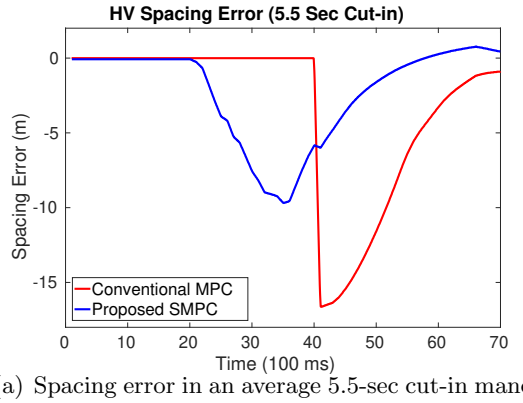


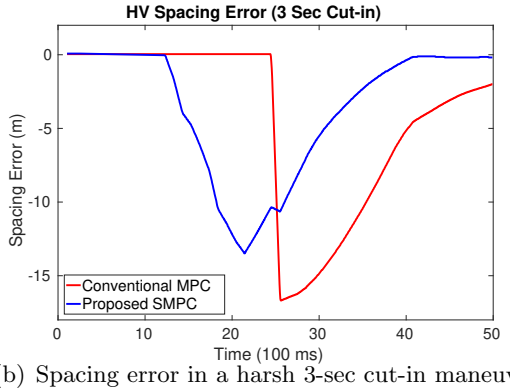
Figure 2.16: Cut-in Probability, P_c, for the proposed SMPC controller (a) an average 5.5-sec maneuver and (b) a harsh 3-sec maneuver

2.7 Conclusion

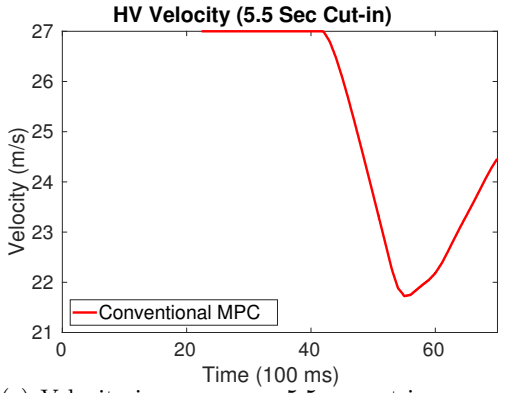
In this dissertation, a probabilistic framework for handling cut-in maneuvers into a CACC platoon is proposed and its better performance compared to the conventional controller design is demonstrated. At the first step of the designed procedure, a cut-in maneuver of an interfering vehicle is detected and its trajectory is predicted using a novel three-layer neural network-based approach. The high accuracy of this method is demonstrated by comparing its results against the state of the art Kinematic-based deterministic models. Afterwards, the output of this phase is utilized to calculate the probability of cut-in predicted trajectory overlap with the host vehicles bad-set area. This probability, which is referred to as cut-in probability, specifies the severity level of the dangerous situation caused by a sudden cut-in into the stable CACC platoon. Obviously, higher values of this probability need more urgent reactions from the host vehicles controller to prevent the possible collision with a smooth and safe reaction. This goal is achieved by giving this probability to a new stochastic MPC controller, designed based on the emerging SHS concept. The overall performance of the designed system is evaluated and its effectiveness for better regulation of the host vehicles reaction to dangerous cut-in situations is discussed using realistic cut-in driving scenarios from SPMD dataset.



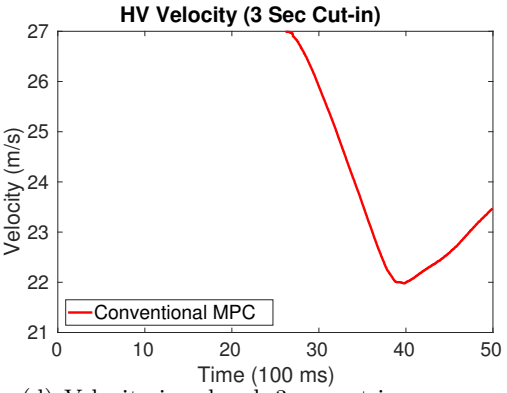
(a) Spacing error in an average 5.5-sec cut-in maneuver



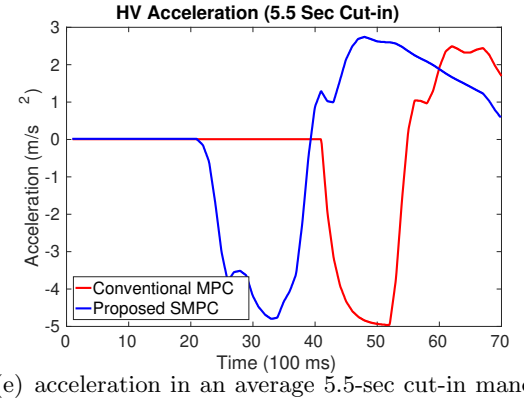
(b) Spacing error in a harsh 3-sec cut-in maneuver



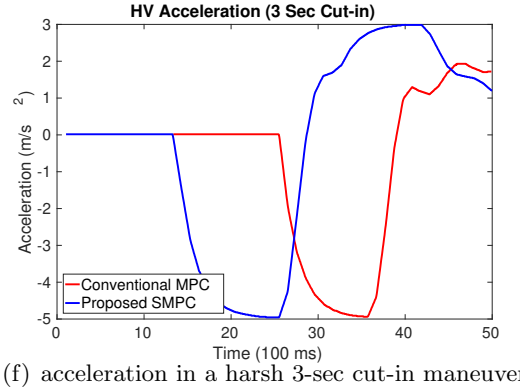
(c) Velocity in an average 5.5-sec cut-in maneuver



(d) Velocity in a harsh 3-sec cut-in maneuver



(e) acceleration in an average 5.5-sec cut-in maneuver



(f) acceleration in a harsh 3-sec cut-in maneuver

Figure 2.17: Comparison of spacing error, velocity, and acceleration of the proposed SMPC (Blue) and the conventional MPC (Red)

Chapter 3

Predictive ECMS for Hybrid Electric Vehicle Powertrain Control

3.1 Introduction

Smart control of powertrain have been studied in different levels to enhance the efficiency and safety of vehicles [68, 69, 70, 71, 72, 73, 70, 74]. Intelligent transportation system (ITS), however, improves the efficiency and safety of vehicles by providing a new level of understanding, known as situational awareness. The benefits of the ITS for conventional vehicles, in terms of the fuel consumption, is limited due to the strict constraints enforced by the driver's power demand and the Internal Combustion Engine (ICE) as the sole source of motive power. However, Hybrid Electric Vehicles (HEVs) have an extra degree of freedom which provides them more flexibility to fully exploit situational awareness for the improvement of their fuel economy. This flexibility comes from an additional source of power, i.e. Electric Motors (EM), which provides the supervisory controller with the opportunity of power distribution between ICE and EM for a better fuel economy. In other words, the Internal Combustion Engine (ICE) has a greater chance of operating at its optimum region when the EM can provide the rest or store its excessive energy in an energy storage system (battery). In addition, the kinetic energy of the vehicle can be harnessed through the regenerative braking. However, the improvements come at the expense of a complicated powertrain and corresponding controller which dictates operation points of powertrain components (such as EM, ICE, and transmission system). Consequently, research interests have been largely directed toward the study of the powertrain management strategies, with special attention being paid to

the power distribution between ICE and EM as one of the most critical decisions impacting HEVs performance.

In general, power distribution strategies can be categorized based on the level of prior knowledge of the drive cycle. First, real-time power distribution strategies only utilize real-time data [75, 76] and many of them are adaptive versions of Equivalent Consumption Minimization Strategy (AECMS) [29, 77, 78]. Real-time power distribution strategies distribute the driver's power demand (through acceleration and brake pedals) between the ICE and the EM based on the most efficient operating points of the powertrain components and the minimization of their total losses. Xu, et al. [79], designed three different cruising strategies which improved the efficiency by adding a new degree of freedom (velocity) to the optimization problem. However, the decisions made by the real-time strategies might not be the most efficient and feasible power distribution due to the system global constraints, such as the constraint on the battery State Of Charge (SOC), which are supposed to be satisfied at the very end of the drive cycle instead of at every single moment.

On the contrary, globally optimized power distribution strategies, such as Dynamic Programming (DP) based methods [80, 81, 82] and Equivalent Consumption Minimization Strategy (ECMS) [83], consider the entirety of the drive cycle to be known in advance. To reduce the computational complexity of this group of strategies, Zhang, et al. [84] found a sub-optimal solution of the optimization problem by utilizing a method called Power-weighted Efficiency Analysis for Rapid Sizing (PEARS). Even though the proposed approach is much faster than DP, this category of strategies still features two drawbacks making it inapplicable for effective real-time control of hybrid vehicle powertrains. First, despite the rich literature on vehicle velocity and trajectory predictions based on live traffic data and driving history [85, 86], an accurate prediction of the whole drive cycle is not practical due to its varying nature and unexpected events. Second, even in the presence of an accurate prediction, these strategies are not suitable to run on a supervisory controller in real-time due to the computational complexity of their optimization techniques.

The third class of power distribution strategies, which is also the main concern of this work, emerged to address the aforementioned drawbacks of the first two approaches. They employ a short-term prediction of the drive cycle [87, 88, 89] to make their decisions towards the control of powertrain components. Compared to the real-time strategies, accessing the short-term prediction of the drive cycle gives an extra degree of freedom to the controller to adjust the soft constraints

locally, e.g. level of SOC, based on its pre-knowledge of the future driving conditions with the aim of efficiency improvement, and still satisfies them globally (at the end of the drive cycle).

Sun, et al. [90] developed a neural network based predictor to forecast the feature velocity of the vehicle based on the driving history. The prediction was used for equivalent factor adaptation for an AECMS controller and more than three percent reduction in fuel consumption was reported. In [91], a new framework, called stochastic model predictive control with learning (SMPCL), is proposed which solves Model predictive control (MPC) problem utilizing quadratic programming with efficiency and SOC regularization included in the cost function. In addition, a Markov Chain Based model of the driver is used to improve the prediction of the drive cycle over the MPC horizon. However, each drive cycle needed to run twice before measuring the fuel economy, so the controller could learn the pattern of the drive cycle.

Utilizing a short-term prediction of the drive cycle can significantly reduce the computational cost of the last group of strategies; however, employing the same optimization techniques as the globally optimized strategies, increases the computational complexity to a level which is higher than the real-time computing power of the embedded systems. Therefore, in this work, despite the previous aformentioned works, three low-computational-cost, real-time predictive AECMS-based strategies are proposed which update the equivalent factor (cost of using electrical energy) according to the short-term prediction of the vehicle's future velocity. These methods are not built upon the global optimization techniques, and, therefore, they do not require a heavy computational load for the supervisory controller to run them in real-time. Each of the strategies introduces an adjustment factor, according to the prediction of the drive cycle in a finite time horizon, which updates the equivalent factor of AECMS for the sake of fuel economy improvement and battery charge sustainability. The first two methods adjust the equivalent factor regarding the energy requirements of the vehicle in the prediction horizon. However, the first method utilizes a fixed prediction time window, and the second one uses a variable time window. The variable-time window is utilized to address the lack of ability in the fixed-time window strategy to adjust the equivalent factor in the presence of consecutive acceleration and deceleration in the prediction window. Finally, the third control strategy, which is an extension of our previous work [92], incorporates the calculated sub-optimal cost of battery charging and discharging over the prediction horizon in addition to the energy requirements of the vehicle in the other two methods. All of the three proposed methods

are applicable to any parallel or series-parallel HEV in which there is a real-time power distribution between an EM and an ICE. However, for HEVs that have more than one EM operational at a time, these methods may need more consideration and modification.

Despite the common trend in the literature toward using simplified and idealized models of HEVs for simulation and validation of powertrain management strategies, in this work a high-fidelity model of a hybridized Chevrolet Camaro is used for validation purposes. This approach is mostly used in the literature to address the impact of incorporating details in powertrain dynamics on fuel economy [93, 94, 95]. The simulation results verify the effectiveness of the proposed methods on fuel economy improvement in comparison with a non-predictive AECMS. The rest of this chapter is structured as follows: the system is described generally and powertrain components are modeled in Section 3.2; the power distribution optimization problem, ECMS, and AECMS are introduced in Section 3.3; the proposed predictive ECMS strategies are formulated in Section 3.4; the simulation results are shown and discussed in Section 3.5; and finally, the chapter is concluded in Section 3.6.

3.2 System description and modeling

3.2.1 Hybrid Electric Vehicles (HEVs)

Hybrid electric vehicles incorporate multiple sources of energy to fulfill the requested propulsion by the driver. Since, in HEVs, there are a wide range of possible configurations of powertrain components, different vehicle architectures evolved by integrating the elements of the hybrid propulsion system. Weight consideration, cost, targeted customers satisfaction, and type of the application are among the most important factors of selecting a specific HEV configuration. Studying different techniques for designing hybrid vehicle powertrains is beyond the scope of this dissertation. This section aims to familiarize the reader with common terminology in the literature of HEVs and also different HEV configurations.

The propulsion system of a hybrid electric vehicle generally comprises an ICE, one or more electrical energy storage modules (e.g. batteries, super-capacitors), electric machines (EMs), a fuel tank, power converters, a transmission and various driveline linkages. These modules can be mixed up in different ways and orders to reach different objectives. The resulting configurations can be categorized under one of the following general divisions (see Figure):

- **Series Hybrids**: In this configuration , the tractive power is exclusively supplied directly by one or more EMs. The ICE, then, is responsible for running an electric generator in its efficient operating region to charge the electrical energy storage system (EESS). The series hybrid architectures are mostly deployed in heavy-duty vehicles, e.g., trucks. Other vehicles with wheel hub motors can also benefit from operating with this configuration.

- **Parallel Hybrids**: In this category of HEVs, the tractive power is provided by an optimal combination of the ICE and the EMs. In addition, a dedicated generator can be employed to maintain the level of SOC at its desired reference value. Parallel-hybrid vehicles are of two sub-categories according to the location of the EMs with respect to the ICE and transmission:

  - **Pre-transmission parallel hybrid**: In this class of HEVs, the EM is attached to the ICE prior to the transmission. This category includes dual-mode hybrids and mild-hybrids with integrated starter-generators.

  - **Post-transmission parallel hybrid**: Here, the EM is connected to either the driven axle (after the final drive) or the non-driven axle, which is referred to as a through-the-road hybrid.

- **Power-split Hybrids**: This class of HEVs employs electric-variable transmissions (EVTs). The EVT is a hybrid transmission which combines all functionalities of fixed and continuously variable transmissions into one single unit. An EVT is a single- or multiple-stage structure of planetary gears that enables the power provided by an ICE and two EMs to be distributed

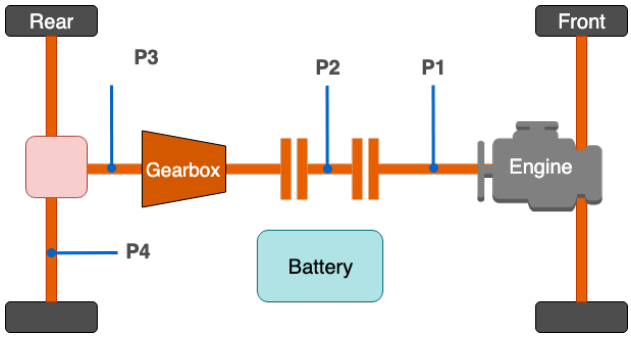


Figure 3.1: Different configurations of HEVs.

Table 3.1: Vehicle parameters

Component	Description
Engine	GM 2.4L I4 LEA E85, Peak Power: 136 kW
Electric Motor	Parker GVM 210-200S, Peak Power: 148 kW
ESS	A123 Systems 7x15s2p, Power Output: 40 kW - 118 kW peak
Transmission	GM 8L45, 8 Speed Automatic
Inverter	Rinehart PM150DX

in a continuously variable fashion. IT can acts in a way to deliver a portion of the generated power to the road, while the excess power is stored in the battery.

3.2.2 High-Fidelity Model - Validation

In this work, a post-transmission parallel HEV is selected for development and validation of the control strategy. The powertrain architecture of the studied HEV is shown in Figure 3.2. For the purpose of assessment, a high-fidelity model of the hybridized Chevrolet Camaro is used, which is developed by the West Virginia University EcoCAR 3 (an Advanced Vehicle Technology Competition (AVTC) series) team. The components of this vehicle are listed in Table 3.1. The model is created mostly based on Simulink and Simscape [96] where the required data regarding each component is provided to the team by the corresponding company. In addition to the high-fidelity model, a simplified and idealized version of the model is also developed to run inside the supervisory controller due to its limited computational capacity.

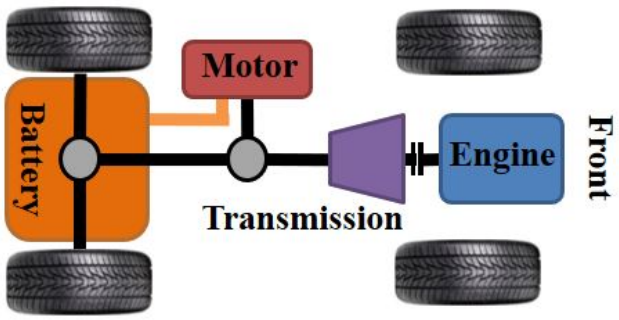


Figure 3.2: Post-transmission hybrid electric vehicle

3.2.3 Idealized Model - Controller

According to the level of detail utilized in modeling the powertrain components, a forward-facing quasi-static method is considered for the modeling of a post-transmission parallel HEV. In this approach, knowing the total required traction force, F_{trac}, the speed and acceleration of the vehicle are calculated by the vehicles longitudinal dynamic model:

$$\begin{aligned} m\frac{dv}{dx} =& F_{trac} - m.g.C_r\cos(\alpha(t)) - \frac{1}{2}\rho.C_d.A_d.v(t)^2 \\ & - m.g.\ sin(\alpha(t)) \end{aligned} \tag{3.1}$$

where m (kg) is the vehicle's mass, v (m/s) is the vehicle's speed, g (m/s^2) is the gravitational constant, C_r is the road friction constant, ρ (kg/m^3) is the air density, C_d is the aerodynamic drag coefficient, A_d (m^2) is the cross-sectional area of the vehicle, and α is the angle of the road slope. The wheel tractive force F_{trac} is the product of the road surface adhesion coefficient and the normal force acting on the wheel. To calculate the normal forces which act on each wheel, the dynamic weight transfer is modeled for the front and rear wheels in terms of the vehicle pitch and lever arms to the vehicle center of mass. Then we can calculate the road load torque for each wheel from the wheel diameter and tractive force:

$$T_{road} = 0.3048.F_{trac}.R_w \tag{3.2}$$

where R_w is the wheel diameter. The torques acting on a wheel include the road load torque, halfshaft torque, bearing friction torque, and the braking torque.

The fuel consumption of the ICE in the model is given by a static map:

$$\dot{m}_{fuel}(t) = f_1(T_{ICE}, \omega_{ICE}) \tag{3.3}$$

where $\dot{m}_{fuel}$ (g/s) is the fuel rate, T_{ICE} (Nm) is the crankshaft torque, and ω_{ICE} (rad/s) is the engine angular velocity. Then, the power consumed by the ICE is calculated as follows

$$P_{fuel}(t) = \dot{m}_{fuel}(t) \times Q_{fuel} \tag{3.4}$$

where Q_{fuel} (J/g) is the fuel energy density constant. Similarly, the battery terminal power P_e (watt) is given by two static maps $f_1(T_{em}, \omega_{em})$ and $f_2(T_{em}, \omega_{em})$ for the motoring and generating modes of the EM, respectively. Here, T_{em} (Nm) represents the motor shaft torque, and ω_{em} (rad/s) is the motor shaft angular velocity.

The battery is modeled as a resistive Thevenin equivalent circuit including a voltage source (V_{oc}) and a resistance (R_{bat}) which are nonlinear functions of the battery State Of Charge (SOC), indicated by $\zeta(t)$, and is calculated as follows

$$\zeta(t) = \zeta_0 - \frac{\int_0^t I_{bat} dx}{Q_{bat}} \tag{3.5}$$

where I_{bat} (A) is the battery current, that is positive when the battery is discharging, and it is negative when the battery is charging, and Q_{bat} (Ah) is the battery capacity. Taking time derivatives from the both sides of (3.5), the rate of change of SOC derived as a function of battery current:

$$\dot{\zeta}(t) = -\frac{I_{bat}}{Q_{bat}} \tag{3.6}$$

Finally, power at the terminals of the battery is found by:

$$P_e = V_{oc}.I_{bat} - R_{bat}.{I_{bat}}^2 \tag{3.7}$$

The power demanded by the accessories is only considered in the high fidelity model, but it is neglected in the simplified model for ease of calculation.

3.3 Optimal Power Management

3.3.1 Problem Statement

The total fuel consumption, emissions, or a combination of both can be considered as a performance measure for comparison of different power distribution strategies. In other words, the problem of designing a power distribution strategy can be formulated as an optimization problem with the aim of minimizing the performance measure subject to the local and global constraints such as satisfying demanded power, battery SOC sustaining, and the components power output limitations. In this work, the performance measure, which is also the cost function of the optimiza-

tion problem, is defined as the total fuel consumption over the whole drive cycle, or equivalently the integral of the fuel consumption rate, defined in Equation (3.8), over the drive cycle's time interval $[t_0, t_f]$:

$$J = \int_{t_0}^{t_f} \dot{m}_{fuel}(u(t), t)dx \tag{3.8}$$

where $\dot{m}_{fuel}$ (g/s) is the fuel rate of the ICE, and $u(t)$ represents the amount of torque which is provided by the EM and is considered as the control input variable in the optimization problem. Hence, the optimal torque generated by the EM at each time instant, $u^*(t)$, can be found by minimization of the cost function:

$$u^*(t) = \arg\min_u J(u(t), t) \tag{3.9}$$

The cost function is subject to a set of constraints which impose operational limitations on the powertrain components such as engine output power, electric motor output power, minimum and maximum permissible values of the battery state of charge, and the drivers power demand satisfaction:

$$0 < P_{ICE} < P_{ICE,max} \quad \forall t \in [t_0, t_f] \tag{3.10}$$

$$P_{em,min} < P_{em} < P_{em,max} \quad \forall t \in [t_0, t_f] \tag{3.11}$$

$$\zeta_{min} < \zeta(t) < \zeta_{max} \quad \forall t \in [t_0, t_f] \tag{3.12}$$

$$P_{dem} = P_{em} + P_{ICE} \quad \forall t \in [t_0, t_f] \tag{3.13}$$

where *max* and *min* indicate maximum and minimum permissible values, respectively, P_{ICE} is the output power of the internal combustion engine, P_{em} is the output power of the EM, $\zeta(t)$ is the SOC of the battery, and P_{dem} is the power demand requested by the driver. In addition to the local constraints, the optimization problem is subject to a global constraint which is the charge sustainability of the battery,

$$\zeta(t_0) = \zeta(t_f) \tag{3.14}$$

that indicates the SOC at the end of the drive cycle should be equal to its initial value.

3.3.2 Equivalent Consumption Minimization Strategy (ECMS)

Equivalent Consumption Minimization Strategy (ECMS) [83] offers an *equivalent factor* between electrical (battery) and chemical (fuel) energy consumption. This factor, denoted by $s(t)$, basically represents the virtual fuel energy consumption equivalent to the stored energy in the battery by ICE. To find the globally optimal solution of the problem (3.9) subject to (3.10)-(3.14), a Hamiltonian function H is defined as

$$H(\zeta, u, \lambda, t) = \dot{m}_{fuel}\big(u(t), t\big) + \lambda(t).\dot{\zeta}(u, \zeta) \tag{3.15}$$

where λ is the co-state. Substituting Equation (3.6) in (3.15), the Hamiltonian function can be rewritten as:

$$H(\zeta, u, \lambda, t) = P_{fuel} + s(t).P_e \tag{3.16}$$

where equivalent factor $s(t)$ is defined as

$$s(t) = -\lambda(t)\frac{Q_{thv}}{V_{oc,max}.Q_{bat}} \tag{3.17}$$

where Q_{thv} is the fuel lower heating value. The Hamiltonian function (3.16) is the summation of the fuel power P_{fuel}, and the battery power P_e transformed into an equivalent fuel power $\big(s(t)P_e\big)$ which is consumed or saved by the engine when the battery is charged or discharged, respectively. The equivalent factor plays a crucial role in the power distribution between the ICE and the EM using ECMS. The optimal value of the equivalent factor (s_{opt}) can be calculated when the drive cycle is known a priori as the Hamiltonian is a convex function according to the shape of the fuel curve demonstrated by [75]. However, in the real world having an accurate estimation of the whole drive cycle is impractical due to its stochastic varying nature. Therefore, the powertrain controller does not have enough information to derive s_{opt}. To overcome this issue with ECMS, an extension of it known as Adaptive ECMS (AECMS) emerged in the literature in which the equivalent factor is updated in real-time based on the main objectives of the control strategy.

3.3.3 Adaptive ECMS (AECMS)

For charge sustainability as one of the most critical goals of the control strategy, the equivalent factor can be updated so that it weights the electrochemical battery power according to the deviation of SOC from its reference value. In other words, when $(\zeta(t) - \zeta(t_0))$ is considerably positive, $s(t)$ has a small value, and therefore the electrochemical energy becomes less expensive than the fuel energy. On the contrary, when $(\zeta(t) - \zeta(t_0))$ is negative, $s(t)$ would be a large value to penalize the electrochemical energy consumption. This idea is the basis of the majority of AECMS strategies in which the equivalent factor is updated based on the deviation of SOC from its reference value [97] and information about the driving condition history [98].

Chasse et al. [78] has suggested to use a PI controller to update the equivalent factor at each time instance:

$$s(t) = s_0 + K_P(\zeta_{ref} - \zeta(t)) + K_I \int_0^t (\zeta_{ref} - \zeta(\tau))d\tau \tag{3.18}$$

where s_0 is the initial value of equivalent factor, K_P and K_I are proportional and integral gains of the PI controller, respectively, and ζ_{ref} is the reference value of the SOC. Using (3.18) to update the equivalent factor, when the deviation of SOC from its reference value $(\zeta_{ref} - \zeta(t))$ is negative, $s(t)$ decreases, and consequently the supervisory controller would prefer to discharge the battery by raising the proportion of electric power applied to meet the driver's power demand. On the contrary, positive SOC deviation leads to $s(t)$ increase, and accordingly the proportion of electric power in vehicle propulsion decreases. This adaptation law updates $s(t)$ only by taking the instantaneous value of SOC at each time instant into account.

3.4 Proposed Framework: Predictive ECMS (PECMS)

In this work, a predictive power distribution framework is proposed with the primary focus on reducing the computational complexity of the algorithm. In this scheme, a speed prediction block provides a short-term prediction of the drive cycle using the information acquired through the vehicular communication network. Design of the prediction block is beyond the scope of this work. However, several methods have been proposed in the literature for short-term and long-term predictions of traffic flow and a vehicle's speed profile and trajectory [99, 100, 101, 102, 103, 104].

Therefore, for the rest of this work, we assume that up to one minute ahead of the drive cycle is available to the controller at each time instant as the output of the velocity predictor. The proposed control strategies get the predicted speed profile and the driver's instantaneous power demand and decide how to distribute the requested power between the ICE and EM based on the minimization of the total loss, not only at each time instant but also over the prediction horizon. In the following, three different Predictive extensions of the AECMS (PECMS) are proposed.

3.4.1 Predictive ECMS: Method I

One of the key factors which directly affects the equivalent factor is the constraint on the SOC level. Moreover, the energy requirements of the vehicles in a near-future time window ($[t, t + t_h]$) determine how the SOC level is going to change in this window. In other words, a low or negative value reveals an upcoming regenerative braking, while its large value is interpreted as a forthcoming high demand of power, which might not be satisfied by the ICE, exclusively. Therefore, near future perspective of the energy requirements represents the average variation in the cost of using the EM in the upcoming seconds. This gives the opportunity to modify the equivalent factor in advance with the aim of improvements in fuel economy and charge sustainability.

In AECMS, variations of the equivalent factor are directly related to the energy requirements of the vehicle as the EM should come to the ICE's aid in high power demand situations, or save its excessive generated power during low demands. Therefore, the lack of information about the future requirements causes an unnecessary fluctuation in equivalent factors, which is a reason for excessive engine ON/OFF cycles and transient losses. Hence, considering the average energy requirements of the vehicle in the near future can decrease the unnecessary fluctuations and its corresponding losses. On this ground, we proposed a new definition for the equivalent as follows

$$s_I(t) = s(t) + \delta s(E_{req}) \tag{3.19}$$

where $s(t)$ is defined in Equation (3.18), and δs is the modification factor that is a function of the energy requirements of the vehicle in the prediction horizon ($E_{req} = E_{req}[t, t + t_h]$) and defined as

$$\delta s(E_{req}) = \begin{cases} m_p E_{req} & \text{if } E_{req} > 0 \\ m_n E_{req} & \text{otherwise} \end{cases} \tag{3.20}$$

in which m_p and m_n are variable factors that determine the extent to which required energy affects the equivalent factor for its positive and negative values respectively. These factors are defined as a function of the SOC deviation from its limits to guarantee that the SOC level will not overstep its pre-defined boundaries

$$m_p = (SOC_{max} - SOC)\frac{\delta s^{max}}{E_p^{max}} \tag{3.21}$$
$$m_n = (SOC - SOC_{min})\frac{\delta s^{min}}{E_n^{min}}$$

where SOC_{min} and SOC_{max} are the minimum and the maximum permissible values of SOC respectively, δs^{max} and δs^{min} are maximum and minimum possible variations of the equivalent factor in a period of time with the length of the prediction horizon, and E_p^{max} and E_n^{min} are the possible most positive and negative values of the energy requirements of the vehicle in the same period of time, respectively. With these definitions, m_p and m_n go to zero when the SOC approaches its boundaries, and as a consequence the future energy requirements will not affect the equivalent factor and it is left to be determined by the PI controller exclusively. However, when the level of SOC is not close to its boundaries, the modification factor is proportional to the required energy of the vehicle in the prediction horizon.

In fact, in AECMS the equivalent factor fluctuates around its optimal value (ECMS). Therefore, when the level of SOC is too low, the equivalent factor goes far beyond the optimal value to force the controller to discharge the battery despite the total loss. However, with the new definition of the equivalent factor (3.19), the controller still gives more weight to the efficiency of powertrain when the SOC level is too low if there is an upcoming regenerative braking opportunity. The same argument applies to the situation in which the SOC level is too high. Besides, the new formulation of the equivalent factor can also improve the fuel economy by decreasing the number of engine ON and OFF cycles. This decrease is a direct result of the fact that the modification factor decreases the cost of using the electrical energy when there is an upcoming regenerative braking and prevents the control strategy from shutting the EM down even if the SOC level is low. Similarly, the controller would not turn the EM on when there is an approaching high acceleration demand even if the level of SOC is high enough.

3.4.2 Predictive ECMS: Method II

Despite the improvements in fuel economy associated with Method I (Section 3.4.1), this method can fail in the case of subsequent regenerative braking and high power demand on account of near zero value for the integration of the energy requirements over the prediction horizon. To overcome this downside, a new method with a dynamic time horizon is developed. The definition of the equivalent factor in this method is the same as the Method I and is expressed by Equations (3.19)-(3.21). However, the modification factor is calculated over a dynamic time window. The length of the time window at each time instance is defined as the time to the next minimum of the vehicle's drive cycle (as long as the time window has a length of less than the prediction horizon), which is the last moment of the upcoming regenerative braking. Furthermore, to eliminate the small changes, a minimum drop in the velocity is considered when the algorithm is looking for the next minimum of the drive cycle. Finally, when the next minimum of the drive cycle is closer in time than a threshold, the algorithm neglects it to prevent a very short time window and looks for the next minimum of the drive cycle.

3.4.3 Predictive ECMS: Method III

So far, the proposed methods (I and II) take only the future energy requirements of the vehicle into account to adjust the equivalent factor. However, further improvement of the efficiency is expected, considering either it is more beneficial to charge or discharge the battery in the prediction horizon and to what extent. To this end, the equivalent factors update law should be not only a function of the energy requirements, E_{req}, but also the cost of charging, s_n, and discharging, s_p, of the battery in the near future. Therefore, we proposed a new formulation for equivalent factor's update law as follows:

$$s_I(t) = s(t) + \delta s(E_{reg}, s_n, s_p) \tag{3.22}$$

where $s(t)$ is defined in Equation (3.18), and δs is a modification factor that is a function of the energy compensated through the regenerative braking ($E_{reg} = E_{reg}[t, t+t_h]$) and the cost of battery charging (s_n) and discharging (s_p) over the prediction horizon t_h. Based on a method proposed in [105], we are able to calculate s_n and s_p in a specific time interval. However, due to the short

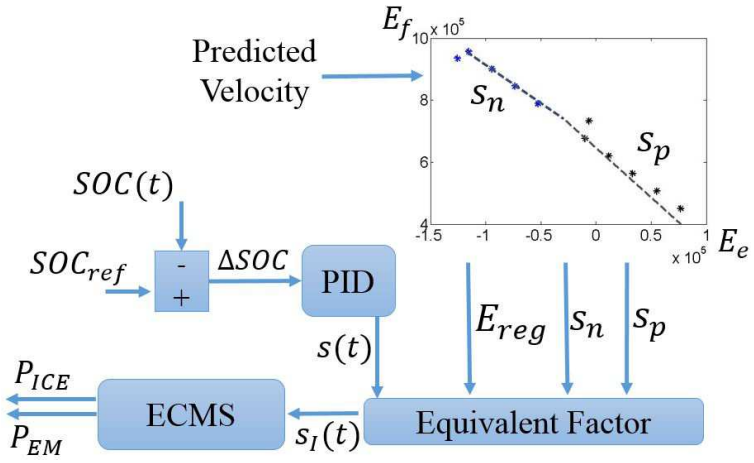


Figure 3.3: Schematic of the Method III

prediction horizon in our work, their proposed approach needs more consideration for possible special cases. To find E_{reg}, s_n and s_p in the prediction time horizon, the model of vehicle is run on the prediction time horizon for different constant values of torque distribution $u = \frac{\tau_m}{\tau_d}$ among the parallel paths where τ_m and τ_d are the torque of the EM and the requested torque by the driver, respectively. Torque distribution varies in a feasible range from a minimum value (u_{min}) to a maximum value (u_{max}) which are determined by the constraints of the problem. Then, for each u, the fuel energy use over the prediction horizon is plotted against the electrical energy use (depicted in Figure 3.3). It is shown in [105] that this plot is rather linear. Hence, the slope of the fitted line on the points with negative u gives the cost of battery charging (s_n), and the slope of the fitted line on the positive values of u gives the cost of using electrical energy (s_p) on that time interval. Now, the instantaneous value of the equivalent factor s_0 derives using the probability of charging and discharging the battery in the prediction horizon

$$s_0 = p(t)s_p + (1 - p(t))s_n \tag{3.23}$$

in which the probability factor $p(t)$ is the possibility that electrical energy consumption (E_e) is greater than zero at the end of the prediction horizon and is estimated as

$$p(t) = \frac{E_e^+(t)}{E_e^+(t) - E_e^-(t)} \tag{3.24}$$

where $E_e^+(t)$ and $E_e^-(t)$ are the maximum positive and negative values of electrical energy use over the prediction horizon, i.e. from time t to $t + t_h$. In addition, the electrical energy use of the vehicle when the ICE solely provides the propulsion power, $(u = 0)$ gives the possible regenerative braking energy E_{reg} over the prediction horizon.

Since in this work we apply the approach on a short horizon $[t, t + t_h]$, both s_n and s_p can get very large positive or negative values in the case that over the entire (or most) of the prediction horizon. The vehicle is braking, and, therefore, all points are placed exactly (or nearly) on each other. This situation can be detected as it happens simultaneously for both s_n and s_p. In other cases, a small value of s_n can be interpreted that in the near future the vehicle can benefit more by consuming fuel and recharging the battery. In addition, a large E_{reg} shows the possibility of a considerable amount of regenerative braking energy in the prediction horizon. Consequently, we defined s_I as follows

$$s_I = \begin{cases} s + m_e E_{reg} + m_s(s_n - s_{thld}) & \text{if } s_n < s_{thld}^n \\ & s_p > s_{thld}^p \\ \dfrac{s + \alpha s_0}{1 + \alpha} + m_e E_{reg} & \text{otherwise} \end{cases} \tag{3.25}$$

where α is a weighting factor, $m_e < 0$ and $m_s < 0$ are adjustment factors, s_{thld}^n is an upper threshold for s_n, and s_{thld}^p is a lower threshold for s_p. These thresholds and adjustment factors are selected by running the vehicle model with an AECMS controller on a long drive cycle (different than the drive cycles on the simulation section) which includes different driving conditions, namely highway and urban, and finding the maximum and minimum value of $s(t)$. Then, the best values of the adjustment factors are found while the threshold was set to the maximum and the minimum of $s(t)$. Therefore, using the new proposed evaluation for $s(t)$, the cost of electromechanical energy starts decreasing when there is a possible regenerative braking in the prediction time horizon, even when the SOC is below its target value, and the PI controller tries to increase the cost of electromechanical energy. On the other hand, the cost of electromechanical energy increases when s_n has a very small value which means it is more beneficial to charge the battery. Finally, the cost of electromechanical energy is adjusted when $s_n > s_{thld}^n$ using s_0 which represents the real cost of electromechanical energy over the prediction horizon.

3.5 Simulation results

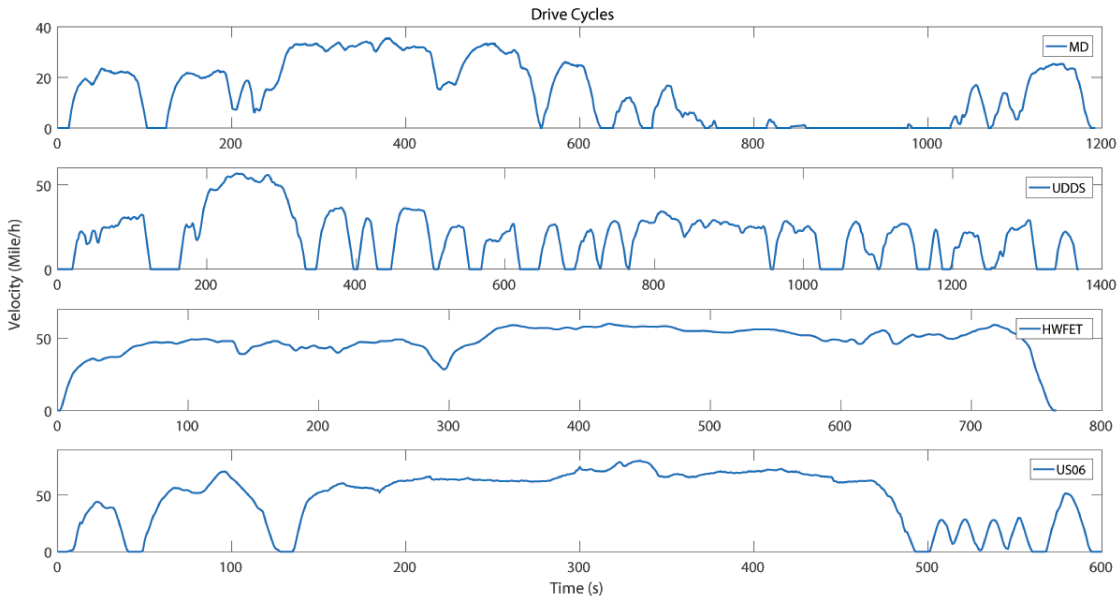


Figure 3.4: Drive Cycles

To investigate the effectiveness of the proposed methods on the fuel economy and charge sustainability, a hybridized Chevrolet Camaro was simulated for three dynamometer driving schedules, developed by the US EPA for the determination of fuel economy in urban (UDDS), highway (HWFET), and aggressive, high speed and/or high acceleration (US06) tests, as well as a real-world scenario from the Model Deployment (MD) dataset [106]. The velocity profiles of these scenarios are illustrated in Figure 3.4. During the simulation, next t_h seconds of the drive cycle ($[t_i, t_i + t_h]$) are considered to be available as the prediction of the driving conditions. The simulation was run for four different time windows including 5, 15, 30, and 60 seconds. In all cases, the PI controller has a proportional gain $P_p = 5$ and an integral gain $P_I = 0.01$. Also, to take the final value of SOC into account, the equivalent fuel economy is calculated by consideration of the electrical energy use

$$Fuel_E = Fuel_C - \alpha \Delta SOC$$

$$\alpha = \frac{Fuel_C}{\Delta SOC_E - \Delta SOC}$$

where $Fuel_C$ is the real fuel consumption, $Fuel_E$ is the equivalent fuel consumption considering the electrical energy use, ΔSOC_E is the change on SOC level using EM exclusively, and ΔSOC is

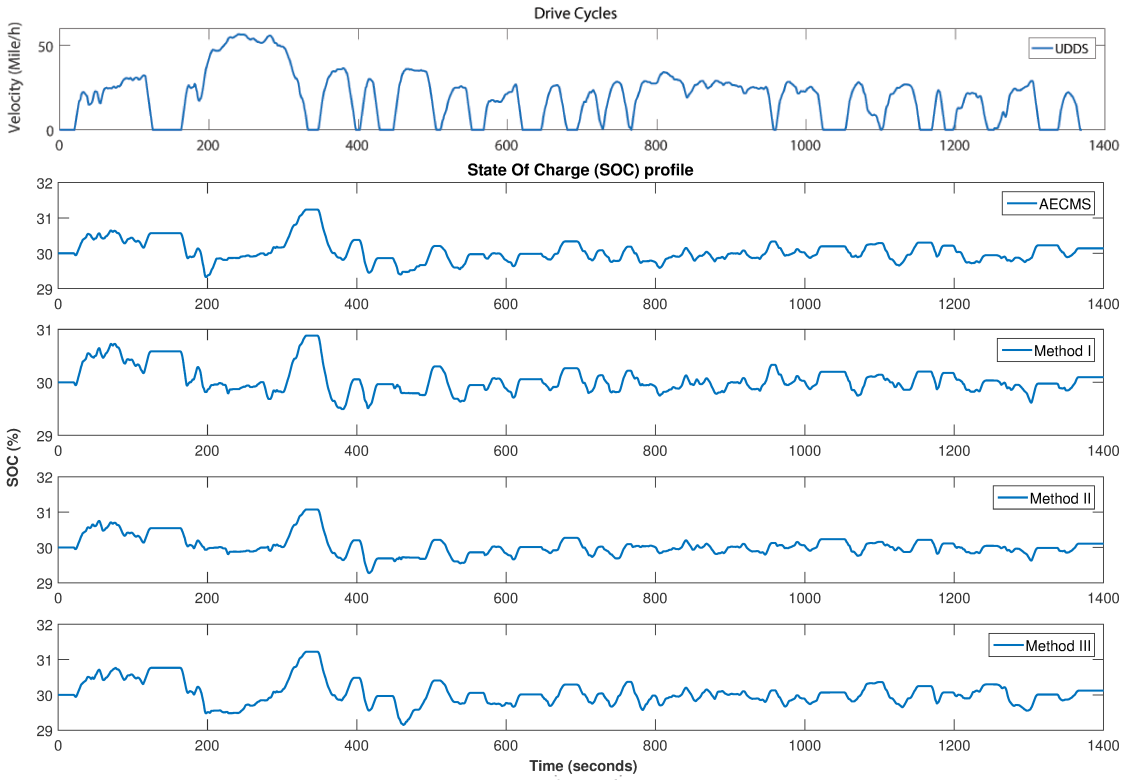


Figure 3.5: SOC for a UDDS drive cycle for AECMS and IECMS, 15s Time Window

the variation in SOC level when both EM and ICE are utilized.

Figure 3.5 illustrates the SOC profiles of AECMS and our proposed predictive versions (15 seconds prediction time window) for UDDS Drive cycle. Moreover, Figure 3.6 demonstrates their corresponding equivalent factors. In all three proposed PECMSs, the equivalent factor starts decreasing early before each regenerative braking.

The detailed simulation results of the proposed methods are listed in Table 3.2 for the sake of comparison. Among all controllers, method III provides the best average fuel economy. The underlying argument in its favor is that the controller has a realization of whether and to what extent it is beneficial to charge or discharge the battery over the prediction horizon. The poor performances of all the three methods, for a short time window, is due to the lack of enough information. Increasing the window size can improve their performance. However, the effectiveness of the first method decreases when the time window exceeds 15s. These results confirm that the integration of the energy requirements over a long horizon can eliminate the effect of its fluctuations and reduce the effectivness of the first method. However, the degradation in method III, is a

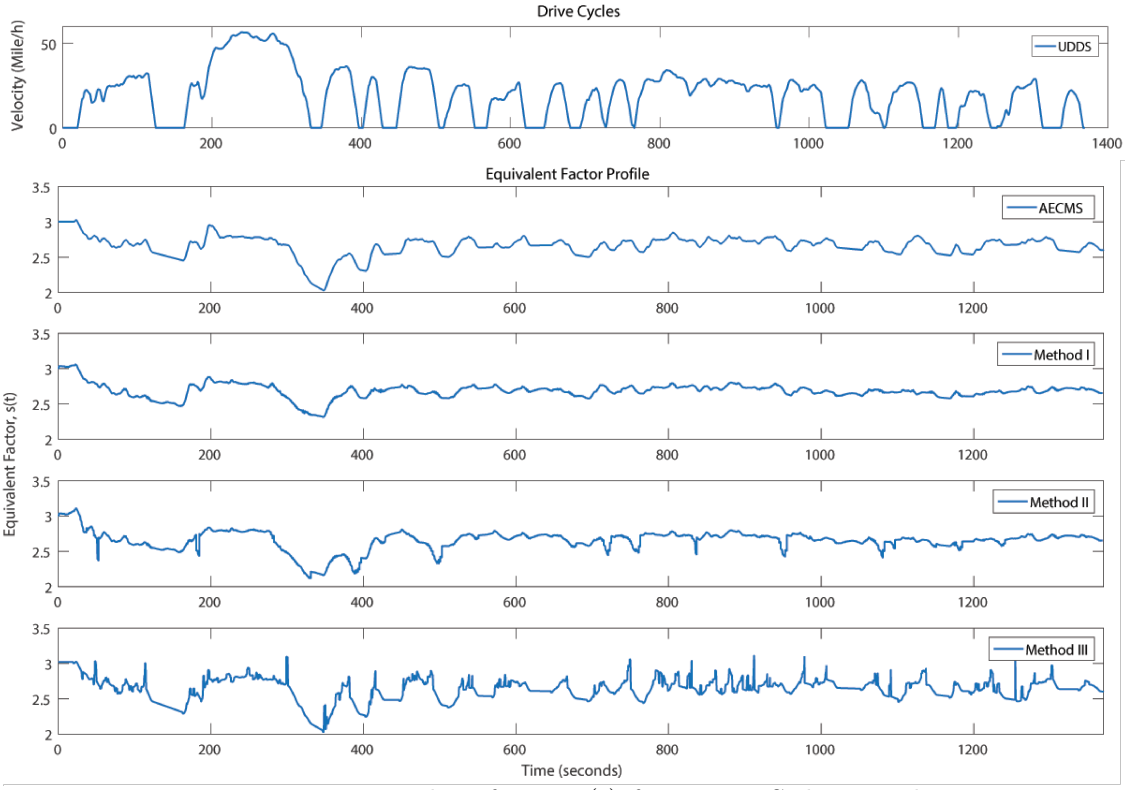


Figure 3.6: Equivalent factor $s(t)$ for a UDDS drive cycle, 15s Time Window

consequence of the fact that a simplified version of the vehicle's model is utilized inside the controller instead of the high fidelity model. This simplification is associated with an error in the calculation of the optimal equivalent factor. Thus, increasing the prediction window can result in a propagation of error in this calculation, and, therefore, the performance starts to degrade.

Strategy	MPG	# ON/OFF	Final SOC	Window	Improvement
UDDS					
AECMS	54.77	24	30.53	-	-
Method I	54.97	23	30.23	60 s	0.3 %
	55.59	23	30.36	30 s	1.5 %
	55.40	22	30.4	15 s	1.1 %
	55.15	24	30.42	5 s	0.7 %
Method II	55.33	21	30.27	60 s	1 %
	55.64	21	30.41	30 s	1.6 %
	55.60	22	30.43	15 s	1.5 %
	55.12	22	30.5	5 s	0.6 %
Method III	55.47	20	30.38	60 s	1.3 %
	55.75	20	30.53	30 s	1.8 %
	55.87	20	30.6	15 s	2 %
	55.61	21	30.62	5 s	1.5 %
HWFET					
AECMS	39.77	21	31.05	-	-
Method I	39.86	21	31.03	60 s	0.2 %
	39.98	20	30.99	30 s	0.5 %
	40.06	19	30.93	15 s	0.7 %
	39.95	21	30.91	5 s	0.5 %
Method II	39.86	20	31.01	60 s	0.2 %
	40.31	17	31	30 s	1.4 %
	39.95	19	30.95	15 s	0.5 %
	40.05	21	30.89	5 s	0.7 %
Method III	40.1	19	31.25	60 s	0.8 %
	40.93	17	31.23	30 s	2.9 %
	41.40	16	31.15	15 s	4.1 %
	40.91	19	31.02	5 s	2.9 %
US06					
AECMS	24.99	20	30.52	-	-
Method I	25.29	18	30.39	60 s	1.2 %

	25.27	18	30.21	30 s	1.1 %
	25.35	17	30.63	15 s	1.4 %
	25.21	18	30.73	5 s	0.8 %
Method II	25.32	17	30.25	60 s	1.3 %
	25.43	17	30.12	30 s	1.8 %
	25.41	16	30.42	15 s	1.7 %
	25.2	18	30.7	5 s	0.8 %
Method III	25.8	17	30.22	60 s	3.2 %
	25.92	14	30.35	30 s	3.7 %
	25.87	15	30.25	15 s	3.5 %
	25.76	18	30.46	5 s	3.1 %
Model Deployment					
AECMS	54.77	15	30.53	-	-
Method I	30.23	14	54.85	60 s	0.3 %
	55.59	13	30.36	30 s	1.5 %
	55.40	12	30.4	15 s	1.1 %
	55.15	13	30.42	5 s	0.7 %
Method II	55.33	12	30.27	60 s	1 %
	55.64	12	30.41	30 s	1.6 %
	55.60	12	30.43	15 s	1.5 %
	55.12	14	30.5	5 s	0.6 %
Method III	55.47	12	30.38	60 s	1.3 %
	55.75	11	30.53	30 s	1.8 %
	55.87	11	30.6	15 s	2 %
	55.61	13	30.62	5 s	1.5 %

Table 3.2: Fuel Economy, Final SOC, and Number of Engine ON/OFF for an UDDS drive cycle

3.6 Conclusion

Incorporating information collected through vehicular communication networks and local sensory data leads to an accurate prediction of driving conditions in a short time window. This

information enables the supervisory controller to make smarter decisions in terms of power distribution between ICE and EM. In this work, a framework is proposed which utilizes the predicted driving conditions and energy requirements of the vehicle to find the most optimal combination of EM and ICE power outputs which minimize the fuel consumption while sustaining the state of charge of the battery ESS. All three proposed methods are developed on top of AECMS, which is one of the most promising approaches for real-time fuel consumption minimization and charge sustainability in HEVs. The equivalent factor plays the most crucial role in this approach and has a critical impact on its performance. Therefore, in the proposed predictive schemes the equivalent factor was updated in accordance with the predicted driving conditions in the near future to decrease the total loss of ICE and EM. A comparison between the results of proposed predictive methods (PECMS) and real-time AECMS verified the effectiveness of PECMS regarding the fuel economy and the number of engine ON/OFF cycles. However, further investigation is required to study the impact of prediction uncertainty on the effectiveness of the proposed methods.

Chapter 4

Reinforcement Learning for Drive-Cycle Aware Power Distribution Control of HEVs

4.1 Introduction

The main goal of artificial intelligence (AI) is to approach complicated tasks with raw sensory inputs. In recent years, deep learning techniques have quickly become the state of the art in most of the AI realted applications, specially in computer vision [107, 108, 109, 110, 111, 112, 113, 114, 115]. In recent past, a significant advance has been made by integration of Reinforcement Learning (RL) and deep learning based raw data processing, which resulted in the advent of Deep Q Network (DQN) [116]. It is shown that DQNs are able to reach (or even surpass) human performance on different applications and tasks. To this end, a neural network is employed as a function approximator to estimate the action-value function. For a while, DQNs were developing to deal with low-dimensional discrete action spaces. However, many applications of interest, especially control systems, have high-dimensional continuous action spaces. Consequently, a large body of research seeks to design continuous DQNs. For example Lillicrap et al., [117] proposed a DQN framework which is based on an iterative optimization of action-value function by finding the optimal action.

To adapt deep reinforcement learning techniques to continuous domains, one can discretize the action space. However, this approach suffers from the curse of dimensionality, in which the

degree of freedom exponentially increases the possible actions. This condition gets worse, in the case of dealing with the tasks that need delicate control inputs, as they need accordingly finer discretization. Such an enormous action space is hard to be explored, and therefore, training the network gets intractable. To overcome this problem, one can use a model-free actor-critic to learn policies directly in continuous action spaces [118].

Prior to DQN, it was a well-known fact that estimating a value function by means of a non-linear function approximator is a challenging unstable problem. However, DQNs can, in a stable way, learn a value function employing such function approximators due to two novelties: first, an off-policy training scheme is employed to train the network with samples selected from a replay buffer. This modification can minimize correlations between samples; second, a target Q network is trained simultaneously with the network to give consistent targets during temporal difference backups.

Reinforcement learning algorithms are generally employed to solve each task independently and from scratch. However, one may consider the setting in which agents solve a set of related tasks, with the aim of learning more efficiently. One challenge is that sharing information between multiple different tasks fails to converge, due to the fact that different tasks may have distinct optimal policies. Consequently, it could be suboptimal to learn a unique common policy for all the tasks.

In this work, we make use of the same ideas to design a control strategy for the power-train of a HEV, while a short-term prediction of the drive cycle is available to the controller. In fact, this approach is a data-driven alternative to the PECMS which we have proposed in Section 3.4. Our work is a model-free framework based on deep deterministic policy gradient (DDPG) algorithm and can outperform the PECMS, and requires only an actor-critic architecture which is able to learn with very few iterations. Note that the PECMS has full access to the underlying simulated dynamics. Our framework is a multi-level deep reinforcement learning module running at different control levels. The controller takes decisions over two levels of abstractions: first, a top-level control element (meta-controller) estimate the most optimum drive-cycle pattern based on the newly available information about the near future driving condition; then, a lower-level element (controller) takes action based on both the chosen drive-cycle pattern and current state of the system. Our model utilizes stochastic gradient descent (SGD) at different temporal scales to maximize the expected

future rewards both in intrinsic (controller) and extrinsic (meta-controller) levels.

In summary, our contributions in this work are listed as follows:

- Unlike the previous adoption of reinforcement learning for HEV powertrain control, which assumes that there is a single optimal policy for all the drive-cycles, we propose a new architecture which employs two different policies for different levels of control. This modification lets the proposed framework learn a different policy for each driving condition pattern.

- The proposed framework can converge to the optimal power management policy by learning a DRL agent in an offline fashion. The proposed method then is considerably faster than the previous methods proposed in the literature.

- Unlike the global optimization policies, the proposed method does not rely on a priori knowledge of the driving condition. It is also a data-driven model which does not require any detailed and accurate HEV modeling. In fact, the underlying network learns all the details of the model that are important in making power distribution decision.

4.2 Related Works

The HEV power distribution strategy indicates the operation of ICE and EM, to satisfy the requested thrust with minimum fuel consumption. The rule-based HEV powertrain control strategies have been developed to follow a set of rules which are derived based on the human expertise or many-valued logic methods such as fuzzy logic [119, 26]. Despite the ease of running these kinds of approaches in real-time, they do not guarantee the optimality of the control actions.

To overcome their imperfection, one may employ the global optimization methods such as dynamic programming, to decide how to split the power between the ICE and EM [120, 121]. A decision made based on global optimization results in the most optimal fuel consumption for a given trip. However, they need a priori knowledge of the driving condition during the whole trip. Moreover, they heavily rely on an accurate and detailed model of the HEV. In contrast, real-time optimization techniques [122], such as the adaptive equivalent consumption minimization strategy (AECMS), emerged to transform the global optimization problem into a sub-optimal instantaneous optimization. Despite their vast adaptation of these techniques in real-time powertrain control

strategies, their performances depend on the driving conditions. Finally, the third group of power management strategies employ a short-term prediction of the drive cycle [87, 88, 89] to make their decisions towards the control of powertrain components. Compared to the real-time strategies, accessing the short-term prediction of the drive cycle gives an extra degree of freedom to the controller to adjust the soft constraints locally, e.g., level of SOC, based on its pre-knowledge of the future driving conditions with the aim of efficiency improvement, and still satisfies them globally (at the end of the drive cycle).

Machine learning provides a powerful tool for the decision-maker (i.e., the agent) to realize how to optimally take action when the highly accurate model of system dynamics is difficult, or even impractical, to obtain. The agent can sense the state of the environment to take action based on the current state. Then, a reward based on the taken action will be given to the agent. Inspired by the given reward, and being aware of the current state, the agent seeks a policy, which maps each possible state to an action, by learning from its taken actions at each state and the received reward values. A reinforcement learning (RL) framework has been already developed for the HEV energy management problem [123]. Their proposed policy for power distribution in HEVS does not rely on any knowledge of the future driving condition. In [124], the application of inverse reinforcement learning has been investigated for learning driver behavior. However, it is out of our focus.

RL has been applied to the problem of HEV power distribution to minimize fuel consumption, or total operation cost [123, 125, 126]. Even though RL techniques have been proven to convergence to the optimal policy; the convergence rate depends on the dimensions of the state and action space. With the success and popularity of neural networks in many machine learning applications, a few research works studied applications of learning techniques on HEVs. He et al. [127], proposed a learning vector quantization neural network to identify the driving cycle profile. A fuzzy neural network has been employed in [128] to detect urban driving conditions. In [129] a framework is introduced based on the ECMS and an adaptive neural network for driving cycle recognition to decrease the sensitivity of the algorithm to different driving patterns.

The original DPG paper demonstrated data efficiency for stochastic actor-critic in an off-policy fashion. They applied the method on multiple toy problems with linear approximators. However, they did not examine how robust is the approach to high-dimensional state spaces. Levine et al. [130] showed that the frameworks which are developed based on DPG could not deal with large

scale problems. A vast majority of the past work with actor-critic optimization approaches, due to instability in training, had the same problem with scaling up to more challenging tasks [131].

Recent works demonstrated that with a model-free policy search the whole framework is more robust to the scale of the problem [132]. Wawrzy et al. [133] proposed to use stochastic policies in actor-critic frameworks employing a replay buffer. The DPG algorithm has been extended [134] with an auxiliary network, namely deviator, which learns $\partial Q/\partial a$ explicitly.

However, they only train it on low-dimensional action spaces. To solve this issue, SVG(0) is proposed in [135] which employs a Q-critic to learn a stochastic policy. This method can be applied to stochastic policies using the re-parametrization trick. Schulman et al. [136] developed trust region policy optimization (TRPO), which directly incorporates stochastic neural network policies without braking up the problem into an optimal control phase and supervision phase. This approach does not need to learn an action-value function, and consequently, is significantly less data efficient.

To overcome the challenges of training an actor-critic framework, Levine et al. [137] proposed to employ guided policy search (GPS) algorithms and decompose the problem into three easy-to-solve stages: first, they utilize full-state observations to produce locally-linear approximations, and then use optimal control to find the locally-linear optimal policy; finally, they use supervised learning to learn a complex, non-linear policy (parameterized with a deep neural network) to regenerate the state-to-action mapping.

4.3 Preliminaries

In this section, we provide some rudiments of RL and Recurrent Neural Networks, necessary to understand the proposed framework.

4.3.1 Reinforcement Learning

To get benefit of the near future driving conditions, we can follow the standard reinforcement learning setup in which an agent interacts with an unknown environment E in discrete timesteps. At each timestep the agent senses the environment through an observation $s \in S$ (known as state) and takes an action $a \in A$. Consequently, it receives a scalar reward r from the environment.

Here, S is the set of possible states, and A is the set of plausible actions. In our setup, S consists of all the possible configurations of the HEV drivetrain which can affect our power distribution decision regarding the fuel economy improvement, namely battery State of the Charge (SOC), driver's demanded power, and ICE and EM speeds. Similarly, A is the set of actions we can take, which is also the output of our power distribution control unit. We define the action as the portion of the requested power which is supposed to be provided by the EM. Clearly, the action set should be in the range of $[\underline{P}_m/P_d, 1]$, where $\underline{P}_m$ is the minimum possible instantaneous power of the EM, and P_d is the demanded power by the driver.

A policy, π, defines the agents behavior and maps each states to the actions. The return from a state is defined as cumulative total of discounted future rewards $R(s,a) = r(s,a) + \gamma R(s',a')$, where $0 \leq \gamma \leq 1$ is the discount parameter. It should be noted that the value of the return is related to the actions chosen, and therefore depends on the policy π.

In Reinforcement Learning, we are looking for a policy $\pi : S \rightarrow A$ that maximizes the expected return $Q_\pi(s_0)$, which is defined as the total expected rewards, following a policy π and starting from an initial state s_0. To find such a policy, in most of the reinforcement learning algorithms the action-value function is employed. Action-value function describes the expected return in given state s after taking an action a, and thenceforth selecting the actions from policy π:

$$Q_\pi(s,a) = \sum_{s'} T(s,a,s')(r(s,a) + \gamma Q_\pi(s', \pi(s'))) \tag{4.1}$$

where $T(s,a,s')$ denotes the probability of transitioning from state s to s' when taking action a. An optimal policy π_{opt} necessarily satisfies the following condition:

$$\pi_{opt}(s) = \arg\max_{a \in A} Q_{\pi_{opt}}(s,a), \quad \forall s \in S \tag{4.2}$$

Therefore, an easy strategy to find the optimal policy is to estimate an optimal action-value function Q_{opt} and then find π_{opt} using Eq. (4.2). The action-value function depends only on the environment which means that we can learn Q_{opt} policy, using actions which are taken from a completely different stochastic behavior policy. In Q-learning [138], one of the most common off-policy algorithms, a greedy policy $\pi(s) = \arg\max_a Q(s,a)$ is utilized for this purpose.

In actuality, an optimal-policy agent is after collecting more rewards following the approximated

optimal strategy, and learning an accurate approximator of the action-value function, Q, across the state-action space. The latter needs a comprehensive exploration of the state-action space, $S \times A$, which commonly earned by employing an stochastic policy, called ϵ-greedy, which selects a random non-optimal policy with a probability ϵ. In other words, the ϵ-greedy policy is defined as follows

$$\pi_\epsilon(s) = \begin{cases} \hat{\pi}_{opt}(s), & \text{w. prob. } 1-\epsilon \\ \text{Random } a \in A, & \text{w. prob. } \epsilon \end{cases} \tag{4.3}$$

In deep Q-learning, to learn a function approximator of the action-value function, a neural network parameterized by θ^Q is used which can be optimized by minimizing the following loss

$$\begin{aligned} L(\theta^Q) &= \sum_{(s,a,r)} \big(\hat{Q}(s,a;\theta^Q) - y(s,a)\big)^2 \\ y(s,a) &= r(s,a) + \gamma \hat{Q}(s', \pi(s'); \theta^Q) \end{aligned} \tag{4.4}$$

where $y(s,a)$ is an approximation of the true value of the action-value function for the given state-action pair (s,a). However, the aforementioned Q-learning setup cannot be directly applied to the continuous action spaces, since in continuous spaces it is too complex to find the greedy policy with large, unconstrained function approximators. Therefore, an actor-critic approach proposed in [117] can be used to train the neural network using stochastic gradient descent updates.

In that work, they proposed to use another neural network, called actor function $\pi(s;\theta^\pi)$, parameterized by θ^π, which maps each state to a specific action in a deterministic fashion. The action-value function $Q(s,a)$, also called critic, is learned as explained, while the actor is updated applying the chain rule to the expected return [117]. In other words, defining the objective function J as the total discounted reward for a given policy, the actor estimator can be updated using the gradient of J with respect to the actor parameters

$$\bigtriangledown_{\theta^\pi} J = \frac{1}{N} \sum_{s_n} \bigtriangledown_a Q(s,a;\theta^Q) \mid_{s=s_n}^{a=\pi(s_n)} \cdot \bigtriangledown_{\theta^\pi} \pi(s;\theta^\pi) \mid_{s=s_n} \tag{4.5}$$

where N is the total number of samples in the training batch and s_n is the n^{th} state sample.

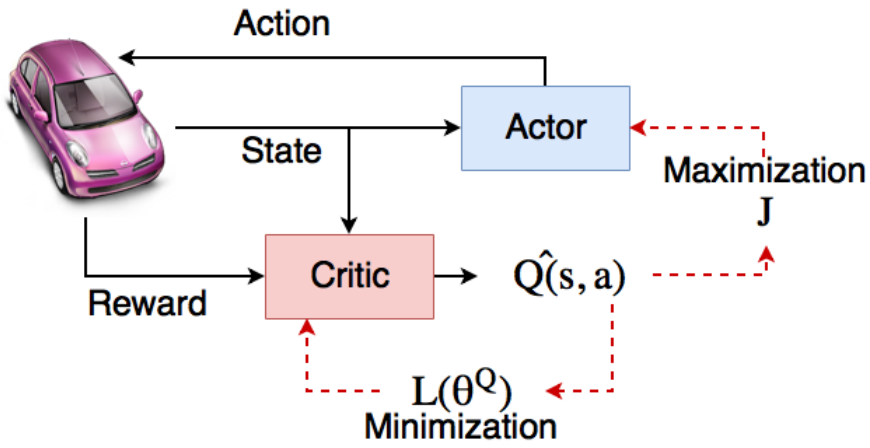


Figure 4.1: Schematic of the whole concept

4.3.2 Recurrent Neural Networks

Recurrent Neural Networks (RNNs) have a memory of the past actions. Therefore, the network is capable of modeling dynamic temporal and spatial dependencies of sequence data. Long short term memory (LSTM) networks are a variant of RNNs which allows capturing long term dependencies. They have been successfully applied in many applications such as speech recognition, time-series prediction, and natural language processing (NLP).

HEV power distribution strategies require looking over a long sequence of states and actions to make an optimal decision. Clearly, the optimal action in each timestep is not independent of the previous actions and states, since the dynamic responses of the vehicle's components have relatively large time constants compared to the control decision time steps. Consequently, observing just the instantaneous state of the powertrain is not sufficent to make the power split decision. Bakker [139] has shown that LSTM can outperform feed-forward networks in complex tasks where part of the environment's state is hidden from the agent. Therefore, our reinforcement learning agent gets the benefit from added memory, by utilization of LSTM networks, which results in making smarter decision which are way closer to the optimal solution.

An extended Deterministic Policy Gradient (DPG) algorithm has been proposed in [140], namely RDPG, to deal with the recurrent networks in reinforcement learning critic and actor updates. In this work we also follow the same technique to train our critic and actor networks.

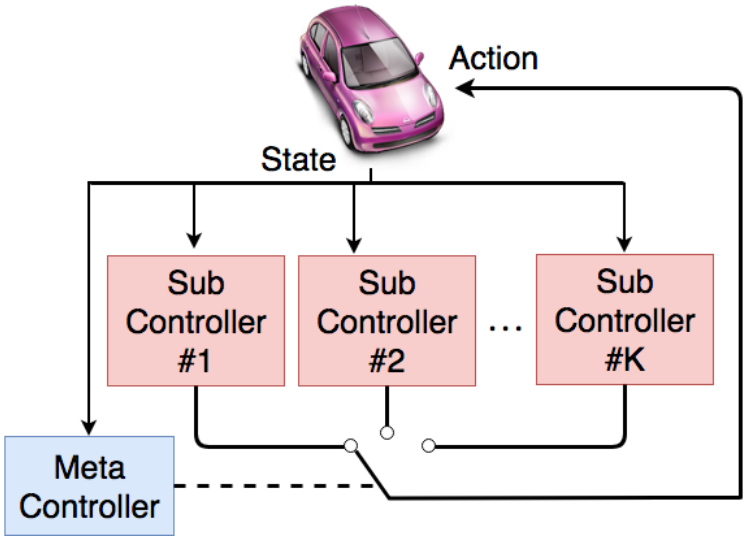


Figure 4.2: Schematic of Drive Cycle Aware Power Distribution

4.4 Drive-Cycle Aware Powertrain control

The explained reinforcement learning algorithm in Section 4.3.1 is able to learn a single policy in order to maximize the cumulative reward of a given task. In the problem of HEV power distribution though, learning a single policy is not optimal for all the driving conditions. It is proven that the powertrain control decision is a function of the long-term driving pattern, such as urban or highway, in addition to the instantaneous driving condition. Therefore, in this work, we propose to learn a distinct controller for each driving pattern, which we refer to as sub-controllers. Then, we also learn a separate DQN as a meta-controller which is responsible to choose the most optimal sub-controller based on the history of the driving information.

The agent, which in the new formulation has multiple controllers in different levels of goal abstraction, takes action based on the received sensory data. The underlying controllers of the agent are referred to as meta-controller and sub-controllers. Note that separate DQNs are employed inside each of these controllers. The meta-controller incorporates the raw information about the states and define a policy over driving patterns by estimating an action-value function $Q_m(s_t, a_t; \theta)$ which maximizes the expected future reward given the sub-controllers. The sub-controllers, in contrast, take in the states and generate a policy over actions by estimating their own action-value functions $Q_{si}(s_t, a_t; \phi_i)$ to solve their underlying optimization problem which is associated with a

specific driving pattern (see Figure 4.2).

We first define the underlying optimization problem to be solved. In our formulation, we have a distribution over drive cycles (tasks). The ultimate goal is to design a meta-controller which select among a set of drive cycle specific controllers. In other words, our framework is supposed to learn the tasks which are sampled from the drive cycle distribution.

Denoting S and A as the state space and action space, respectively, we can define a Markov Decision Process (MDP) with a transition function $P(s', r|s, a)$, where (s', r) represent the next state and reward pair, and (s, a) is the current state and action pair. The agent, here, is defined as a function which maps a series of state-action-reward history $(s_0, a_0, r_0, s_1, a_1, r_1, \ldots, s_{t-1})$ to the next optimal action a_t. To learn this reinforcement learning agent, we propose to iteratively learn high and low level control networks with the weights denoted by θ and $\phi = [\phi_i]$, respectively. In fact, each low level control network ϕ_i learns to be optimal for a specific type of driving condition. These low and high level control networks, define a stochastic policy $\pi_{\phi,\theta}(a|s)$. Our objective is to optimize the expected return (negative fuel consumption) during a low level control networks entire lifetime, over the sampled tasks (drive cycles).

There are different possible ways that one can integrate high level control parameters θ and drive cycle specific parameters ϕ. In this work, we suggest an architecture that is developed having the hierarchical reinforcement learning as motivation. More specifically, each sub-controller ϕ_i defines a sub-policy $\pi_{\phi_i}(a|s)$. The high level controller which switches between the sub-policies, is implemented as a separate neural function estimator. In other words, it defines a meta-policy which is parameterized by θ, and choose action from the set of sub-controller indices $\{i = 1, \ldots, k\}$. Here, the meta-policy takes actions at a fixed N timesteps.

4.4.1 Algorithm

Our framework is developed based on learning a set of sub-policies and a meta-policy, iteratively. Each sub-policy should be optimal to reach high performance for the assigned task by the meta-controller. At the same time, the distribution of tasks between the sub-controller should be in a way that the framework can learn the meta-policy rapidly. To this end, we let the meta-controller decides when to switch between the sub-policies based on our hand-defined drive cycle patterns. More specifically, we start learning the meta-policy using our own definition of drive cycle patterns.

This step is referred to as warm-up training of the meta-controller. However, the meta-controller has the flexibility to choose between the sub-policies in the main training loop.

Algorithm 1 describes how we train the sub and meta-policies. Starting from a random initialization, we update the two components, iteratively. We start with a meta-policy warm up training step. Note that during the warm up step, we use our own labeling for the driving patterns. In this work, we define three driving pasterns, namely urban, highway, and harsh. Consequently, we have three sub-controllers to be selected by the meta-policy.

During the warm-up stage, we first sample a new driving pattern (task) from the driving pattern distribution P_i. Then, we sample from the distribution of the selected driving pattern states P_{dc}. Finally, we update the corresponding sub-controller which is selected to maximize its expected return. In other words, in this stage, the meta-policy is not in the loop of sub-controllers' update. Instead, we update the meta-controller by forcing it to select the correct sub-controller given the information about the past, current, and future driving conditions.

Algorithm 1 Drive Cycle Aware Power Distribution

Randomly initialize θ and ϕ_i for $\forall i$
repeat ▷ (warm-up)
    Sample driving pattern index $i \sim P_i$
    Sample from the i^{th} driving pattern $D \sim P_{dc}$
    Update ϕ_i to maximize the expected return
    Update θ_i to select the i^{th} sub-controller
until convergence
repeat ▷ (joint optimization)
    for `m=1,...,M` **do**
        Sample driving pattern index $i \sim P_i$
        Sample from the i^{th} driving pattern $D \sim P_{dc}$
        Update θ_i to select the i^{th} sub-controller with the maximum expected return
    for `s=1,...,S` **do**
        Sample driving pattern index $i \sim P_i$
        Sample from the i^{th} driving pattern $D \sim P_{dc}$
        Update the selected ϕ_i by the meta-controller to maximize the expected return
until convergence

During the joint optimization stage of the algorithm, both parts (meta and sub controllers) get optimized simultaneously. In this stage, we do not use the pre-defined driving pattern labels to update the meta-controller weights. Instead, at each time step, we update the meta-controller weights in a way to select the sub-policy which maximizes the expected reward. Note that in the

training of the meta-controller we calculate the expected reward of the sub-policies over a larger window size including information from the history of the driving conditions. In this way, the meta-policy can gets the benefit of a larger window to select the most optimal sub-policy. We update the meta-controller for a pre-defined number of steps before updating the sub-policies. Preliminary results proved the effect of multiple step training on improving the performance of the learned policies, and the stability of the training process.

Finally, after completing the meta-controller update steps, we fixed the weights of the meta-policy and train the selected sub-controller by the meta-policy (at each step), and update its weight to maximize the underlying expected return. Since we have multiple sub-controllers, the number of sub-policy update steps is usually more than of the update steps of the meta-policy. We iteratively switch between training of sub and meta controllers until the framework converges.

4.4.2 Architecture and Training

The critic and actor networks comprise 4 fully-connected layers followed by an LSTM layer, 64 nodes each. Both networks take system state as their inputs while the critic network gets the performed action as well. As it is mentioned, we have a multi-dimensional state vector (instantaneous driver's demanded power, SOC, and ICE and EM speeds, as well as the predicted driving condition in the next 30 seconds) and a scalar action value (EM contribution on the drivers' requested power). The structure of the networks are illustrated in Fig. 2.

We trained the DQN by running the model of the HEV in Matlab/Simulink for multiple drive cycles. The reinforcement learning agent was implemented in TensorFlow/Python. The Simulink model and the reinforcement agent were communicating through TCP/IP. The Simulink model runs in a fixed-step setting to send vehicle's state to the agent with a fixed timestep. Subsequently, the agent sends back the action for that timestep and receives the corresponding reward in the next step. The instantaneous reward for each timestep is defined in section 4.4.3.

For exploration purposes, in training time, instead of using ϵ-greedy policy, an exploration normal noise is added to the action of the actor network before passing it to the HEV model. In other words, the action at each time is given by $a_t = \pi(s_t; \theta^\pi) + \eta_t$ where η_t is the normal noise.

To abate how highly correlated data affect the training process, we utilized a large replay buffer to train the neural networks. At each training step, the last five states, actions, and their immediate

rewards are saved as a single training sample in the replay memory. When the size of the replay memory exceeds the batch size, the training process is started by drawing a batch of samples uniformly at random. The replay memory has a fixed maximum size of 100,000 samples. If it reaches its maximum size, new training samples replace the oldest ones. The training is performed in episodic fashion. We used Adam optimization technique to update the networks' parameters. At the end of each drive cycle, the next drive cycle where chosen randomly. We trained the networks for 2.5 millions steps.

4.4.3 Reward Function

The most critical key to the success of reinforcement learning agents is the definition of a proper reward function. The reward should be a good representative of to what extend each action is aligned with the main goal of the problem. The given reward at each timestep, in general, could be not only a consequence of the last taken action but past several actions. The goal of our control system is to minimize the fuel consumption of the HEV over a drive cycle. However, since an HEV can also use electrical energy, the final trip battery State of Charge (SoC) also matters in the optimization. Therefore, instead of just minimizing the total fuel consumption of the vehicle over a drive cycle, we need to minimize the total weighted energy consumption of the vehicle:

$$E_{total} = E_{fuel} + \alpha E_{battery} \tag{4.6}$$

However, the we cannot employ this reward in its intrinsic form, and therefore we define a suitable reward function as

$$r(a_t) == \begin{cases} -r_{nf}, & a_t \text{ Not feasible} \\ -E^t_{fuel} - \alpha E^t_{battery}, & a_t \text{ Feasible} \end{cases} \tag{4.7}$$

where $r_{nf} > 0$ is a big penalty value to prevent the network from generating infeasible actions. In our problem, an infeasible action is an action which cannot satisfy one of the following conditions:

$$\begin{aligned} P_d &= P_m + P_i \\ \underline{P}_m \leq P_m &\leq \overline{P}_m \\ \underline{P}_i \leq P_i &\leq \overline{P}_i \end{aligned} \tag{4.8}$$

where P_d is the power demand, P_m is the power of the EM, and $P_i > 0$ stands for the power of ICE. The underlines and upperlines represent the minimum and maximum possible value of the power, respectively. In braking time, however, this condition should not be satisfied and a_t represents the portion of the maximum possible regenerative braking to be provided by the EM. Obviously, in the best action with this definition for braking time is $a_t = 1$.

To select the value of α, we ran the model of HEV for multiple drive cycles with different driving pattern (e.g. urban, highway, and harsh) both in hybrid (EM and ICE) and pure electric (EM only) modes. Then, the value of alpha is selected as follows

$$\alpha = \frac{E_f}{E_e - E_{eh}}$$

where E_f is the consumed fuel energy, E_e is the electrical energy consumption using EM in pure electric mode, and E_{eh} is the electrical energy consumption in hybrid mode, where both the EM and ICE are utilized. We ran the model using Adaptive Consumption Minimization Strategy (AECMS) controller to estimate an initial value for the alpha. During training. Since the value of α is related to the energy management strategy, we update the alpha in a moving average fashion at the end of each episode:

$$\alpha = \beta\alpha + (1 - \beta)\alpha_{episode}, \tag{4.9}$$

where $0 < \beta \ll 1$ is a real valued coefficient.

Learning the meta-controller to maximize the expected return may result in frequent switches between different controllers. To prevent frequent switches, we add a reward term to the objective function of meta-controller proportional to the time since the last control switch.

4.5 Simulation Results

The simulation results of the proposed framework for a P2 HEV power distribution are presented in this section. We follow the same HEV model as in the previous chapter. The proposed multi-level reinforcement learning framework for HEV power management is compared with the the Method III proposed in Section 3.4.3 and AECMS, based on both standard (UDDS, HWFET, US06) and real-word, i.e., model deployment (MD) dataset [106], driving cycles. To train the agents we used three hours of velocity pattern from MD dataset. The velocity profiles, then, clustered into three groups of highway, urban, and harsh driving conditions. We added a random smooth noise (generated by choosing a Fourier transform coefficient randomly) to augment the data. Due to the lack of harsh driving conditions in the dataset, we created a set of harsh driving profiles by adding relatively high frequency noise to the highway profiles.

To improve the stability of neural-network based reinforcement learning algorithm we use target network schemes. The algorithm maintains two copies of the value function and of the policy networks. We update the first copies, which are used in real-time simulation, with some delay. Different authors have investigated different methods to update the first copies. In this work, we use soft updates as in [117].

To investigate the effect of multi-level control strategy in the proposed framework, we also trained a single universal agent to deal with all the three driving conditions. Similar to the previous experiments in this work, we assumed that the next 30 seconds of the drive cycle is available to the controller. Table II compares the fuel consumptions of proposed methods in this work with the AECMS over different driving cycles. Note that the test MD drive cycle is created by creating a 2-hour driving profile from multiple profiles in MD dataset (different than the one used in the training phase).The fuel consumptions of the proposed multi-level DRL framework are always lower than that of the all other methods. It also always outperforms the single-level control strategy which confirm our assumption that a single-level control strategy could be sub-optimal for different driving conditions.

Strategy	MPG	# ON/OFF	Final SOC	Window	Improvement
UDDS					
AECMS	54.77	24	30.53	-	-

Method III	55.75	20	30.53	30 s	1.8 %
Multi-Level DRL	56.24	17	30.49	30 s	2.7 %
Single-Level DRL	56.19	17	30.38	30 s	2.6 %
HWFET					
AECMS	39.77	21	31.05	-	-
Method III	40.93	17	31.23	30 s	2.9 %
Multi-Level DRL	41.08	18	31.12	30 s	3.3 %
Single-Level DRL	41.04	20	31.09	30 s	3.2 %
US06					
AECMS	24.99	20	30.52	-	-
Method III	25.92	14	30.35	30 s	3.7 %
Multi-Level DRL	26.11	17	30.18	30 s	4.1 %
Single-Level DRL	25.96	17	30.25	30 s	3.9 %
Model Deployment					
AECMS	45.18	68	30.22	-	-
Method III	46.24	59	30.61	30 s	2.3 %
Multi-Level DRL	48.03	52	30.52	30 s	6.3 %
Single-Level DRL	47.76	56	30.57	30 s	5.7 %

Table 4.1: Fuel Economy, Final SOC, and Number of Engine ON/OFF of the proposed multi-level DRL control strategy compared to that of AECMS, Method III, and single-level DRL on different drive cycles

Figure 4.3 shows how the meta-controller split a drive cycle (extracted from MD dataset) between different low-level controllers. Note that, the labels for low-level controllers comes from the pre-training phase. It is clear from the figure that the high-level controller captured the patterns of different scenarios (urban, highway, and harsh).

Figure 4.4 compares the total reward achieved by Multi-Level DRL and Single-Level DRL through the training process. Clearly, Multi-Level DRL converges faster and gets higher rewards

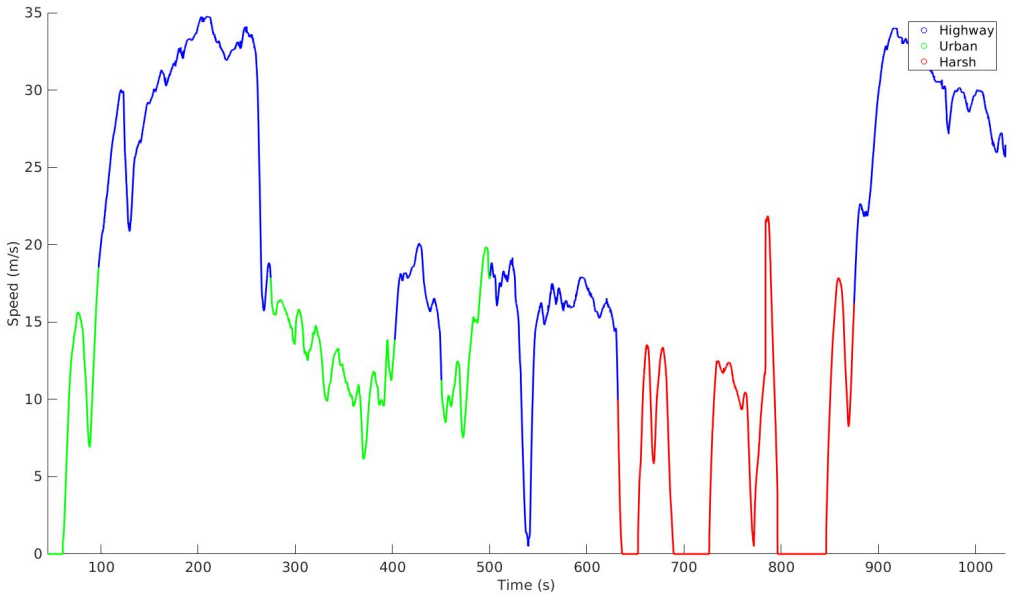


Figure 4.3: High level controller selects a low level controller for power distribution based on the history and future of the driving conditions.

at the end of training.

4.6 Conclusion

An energy management strategy (EMS) plays a critical role in the efficiency of HEVs. However, the driver-specific and general variation of driving conditions affect the optimality of traditional EMSs. In this work, in contrast to the previous developed EMSs in the literature which are designed to track a set of pre-specified rules that are not adaptive to the variable driving conditions, we propose to design an EMS framework that can adapt its rules to the current driving condition. We refer to this framework as drive cycle aware EMS. This adaptation could be to a general pre-defined driving patterns, such as urban, highway, or harsh driving conditions, or drive-specific driving habits. To this end, we proposed a deep Q-Network (DQN) based EMS such that it can switch between multiple policies based on the overall driving conditions in the past and near future. Similar to our proposed PECMS, the EMS output action is the ratio of demanded power distribution between the electric motor and the internal combustion engine. The effectiveness of the proposed

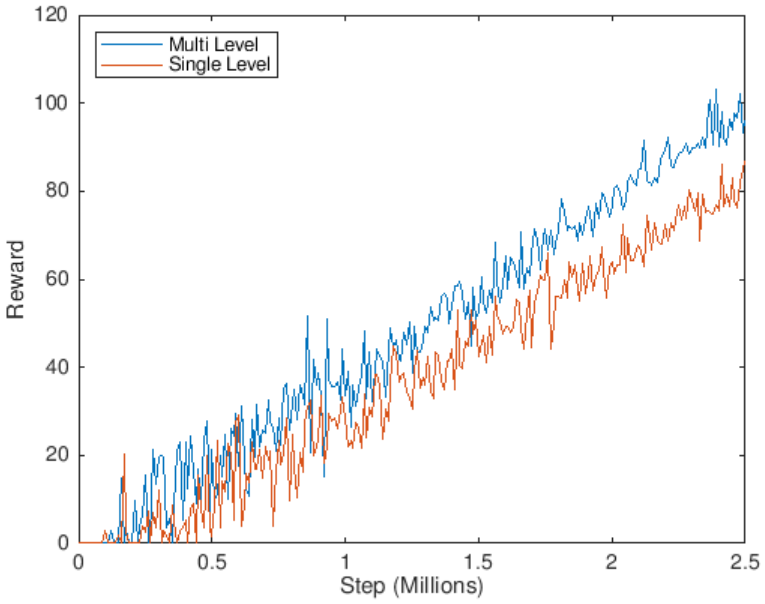


Figure 4.4: Comparison of total reward vs. training steps for Multi-Level DRL and Single-Level DRL.

method has been studied with a set of simulation experiments. Experimental results validate the superiority of drive cyle aware EMS compared to our proposed PECMS.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

In this dissertation, we explored several applications of ITS in improving the efficiency and safety of vehicles, by the integration of information, telecommunication, and cyber technologies into transportation systems. This integration provides vehicles with a real-time comprehensive awareness of their surroundings and situations for the purpose of improving safety and efficiency. In the first part of this work, we first designed a neural-network based trajectory predictor which can estimate the location of other vehicles from the information communicated through V2V communication. Then, we developed an MPC controller which employs future location of the surrounding vehicles to improve the stability and safety of a CACC platoon. The designed controller can react in time and properly to an unexpected cut-in scenario in the middle of the platoon. In the second part of this dissertation, we designed multiple power distribution strategies for HEVs which incorporate the predicted near-future velocity of the vehicle to decrease the fuel consumption by optimally distributing the requested power between the EM and the ICE. Then, we designed a controller which can detect different driving conditions, and use corresponding drive-cycle specific sub-controller to distribute the requested power between the ICE and EM.

Safety: Handling interfering vehicles in a CACC platoon.

Unsignaled lane change is one of the most critical situations among unexpected remote vehicle maneuvers to be addressed in CACC design as it can significantly affect the level of safety and platooning performance in this application. In this work, we specifically focused on a cut-in maneuver

by a remote vehicle, due to its imminent threat to the safety and stability of the whole CACC platoon. We first predicted the probability of any remote vehicles in the adjacent lanes to show a cut-in intention. Then, we developed a probabilistic framework to incorporate the probability of a possible cut-in, in the middle of a CACC platoon, in the control system design to improve the performance of the current conventional CACC controllers.

In our design, a cut-in maneuver of an interfering vehicle is detected, and its trajectory is predicted using a three-layer neural network-based approach. Then, the predicted path of the remote vehicle is used to calculate the probability of a possible cut-in maneuver. The probability is defined as the ratio of predicted trajectory overlap with the host vehicles bad-set area. This probability, which is referred to as cut-in probability, determines how likely a dangerous situation may be caused by a sudden cut-in into a stable CACC platoon. Clearly, higher values of this probability need more urgent reactions from the host vehicles controller to prevent the possible collision with a smooth and safe reaction. This goal is achieved by giving this probability to a new stochastic MPC controller, designed based on the emerging SHS concept. The overall performance of the designed system is evaluated, and its effectiveness for better regulation of the host vehicles reaction to dangerous cut-in situations is discussed using realistic cut-in driving scenarios from SPMD dataset.

Efficiency: HEV power distribution strategy.

We focused on the design of a power distribution strategy for splitting the instantaneous demanded power by the driver, between the EM and ICE in a hybrid electric vehicle. The proposed methods incorporate the prediction of the drive cycle in the near future, to improve the current real-time techniques. We introduced two novel approaches. First, we developed a method based on a linear estimation of the optimal equivalent factor of AECMS in the prediction horizon, namely PECMS. The proposed method then uses this sub-optimal solution to adjust the equivalent factor of AECMS toward a sub-optimal solution. Here, the AECMS, as of its original version, adjusts the equivalent factor to prevent the SOC from diverging too much of its nominal value. However, the PECMS relieves the constraint of AECMS on the SOC, based on the information about the near future driving condition, such as energy requirement or the optimal equivalent factor. We achieved up to 4% improvement in fuel economy using the proposed PECMS methods.

Second, we followed the recent development in reinforcement learning for a novel multi-level

power distribution control. This is important because there is no other prior work addressing this problem using a controller which can adjust its decision to the driving pattern. We proposed to use two reinforcement learning agents in two levels of abstraction. The first agent, select the most optimal low-level controller (second agent) based on the overall pattern of the drive cycle, i.e., urban, highway and harsh. Then, the selected agent by the high-level controller (first agent) decides how to distribute the demanded power between the EM and ICE. We found that by carefully designing a training scheme, it is possible to effectively improve the performance of this data-driven controller. We achieved up to 6% improvement in fuel economy using the proposed drive-cycle aware power distribution strategy.

5.2 Future Work

5.2.1 Stochastic MPC design for CACC

To handle a possible cut-in in a CACC platoon, we reformulated the problem as a deterministic MPC design by remodeling the system as a time-triggered hybrid system. However, as future work, the performance of CACC developed based on its original stochastic MPC problem can be compared with the designed controller. One may also adopt the proposed method in other scenarios than cut-in in a CACC.

5.2.2 HEV Powertrain Control

In Chapters 2 and 3, we proposed two methods of power distribution in HEVs. However, our design was only evaluated in the perfect estimation of the near future velocity of the vehicle. One may evaluate the effect of prediction error on the proposed methods. The design might also need to degrade the effect of prediction error. For future work, it would also be interesting to use a real velocity predictor and study different sources of imperfection. For the multi-level RL controller, a neural network based velocity predictor can be incorporated in the network, as it enables us to jointly optimize the velocity predictor and the controller.

References

[1] "Wireless access in vehicular environment (WAVE)."

[2] Y. P. Fallah, C. Huang, R. Sengupta, and H. Krishnan, "Design of cooperative vehicle safety systems based on tight coupling of communication, computing and physical vehicle dynamics," *Proceedings of the 1st ACM/IEEE International Conference on Cyber-Physical Systems*, pp. 159–167, 2010.

[3] The CAMP Vehicle Safety Communications Consortium consisting of BMW, Daimler-Chrysler, Ford, GM, Nissan, Toyota, and VW., "Vehicle safety communications project, task 3 final report: Identify intelligent vehicle safety applications enabled by DSRC," *Vehicle Safety Communications Consortium (VSCC)*, 2005.

[4] R. Sengupta, S. Rezaei, S. E. Shladover, J. A. Misener, S. Dickey, and H. Krishnan, "Cooperative collision warning systems: Concept definition and experimental implementation," *Journal of Intelligent Transportation Systems*, vol. 11, no. 3, pp. 143–155, 2007.

[5] R. Kiefer, M. Cassar, C. Flannagan, D. LeBlanc, M. Palmer, R. Deering, and M. Shulman, "Forward collision warning requirements project: refining the CAMP crash alert timing approach by examining" last-second" braking and lane change maneuvers under various kinematic conditions," Tech. Rep., 2003.

[6] S. M. Iranmanesh, E. Moradi-Pari, Y. P. Fallah, S. Das, and M. Rizwan, "Robustness of cooperative forward collision warning systems to communication uncertainty," in *Systems Conference (SysCon), 2016 Annual IEEE*. IEEE, 2016, pp. 1–7.

[7] J. Wang, L. Zhang, D. Zhang, and K. Li, "An adaptive longitudinal driving assistance system based on driver characteristics," *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 1, pp. 1–12, 2013.

[8] J. Wang, C. Yu, S. E. Li, and L. Wang, "A forward collision warning algorithm with adaptation to driver behaviors," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 4, pp. 1157–1167, 2016.

[9] F. Breyer, C. Blaschke, B. Farber, J. Freyer, and R. Limbacher, "Negative behavioral adaptation to lane-keeping assistance systems," *IEEE Intelligent Transportation Systems Magazine*, vol. 2, no. 2, pp. 21–32, 2010.

[10] C. Liu, A. Carvalho, G. Schildbach, and J. K. Hedrick, "Stochastic predictive control for lane keeping assistance systems using a linear time-varying model," in *American Control Conference (ACC), 2015*. IEEE, 2015, pp. 3355–3360.

[11] A. Benine-Neto, S. Scalzi, S. Mammar, and M. Netto, "Dynamic controller for lane keeping and obstacle avoidance assistance system," in *Intelligent Transportation Systems (ITSC), 2010 13th International IEEE Conference on*. IEEE, 2010, pp. 1363–1368.

[12] T. Wada, S. Hiraoka, S. Tsutsumi, and S. Doi, "Effect of activation timing of automatic braking system on driver behaviors," in *SICE Annual Conference 2010, Proceedings of*. IEEE, 2010, pp. 1366–1369.

[13] A. Vahidi and A. Eskandarian, "Research advances in intelligent collision avoidance and adaptive cruise control," *IEEE transactions on intelligent transportation systems*, vol. 4, no. 3, pp. 143–153, 2003.

[14] E. Moradi-Pari, H. N. Mahjoub, H. Kazemi, Y. P. Fallah, and A. Tahmasbi-Sarvestani, "Utilizing model-based communication and control for cooperative automated vehicle applications," *IEEE Transactions on Intelligent Vehicles*, 2017.

[15] C. F. Minett, A. M. Salomons, W. Daamen, B. Van Arem, and S. Kuijpers, "Eco-routing: comparing the fuel consumption of different routes between an origin and destination using field test speed profiles and synthetic speed profiles," in *Integrated and Sustainable Transportation System (FISTS), 2011 IEEE Forum on*. IEEE, 2011, pp. 32–39.

[16] H. Kazemi, Y. P. Fallah, A. Nix, and S. Wayne, "Predictive AECMS by utilization of intelligent transportation systems for hybrid electric vehicle powertrain control," *IEEE Transactions on Intelligent Vehicles*, vol. 2, no. 2, pp. 75–84, 2017.

[17] A. L. Rosado, S. Chien, L. Li, Q. Yi, Y. Chen, and R. Sherony, "Certainty and critical speed for decision making in tests of pedestrian automatic emergency braking systems," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 6, pp. 1358–1370, 2017.

[18] A. Tahmasbi-Sarvestani, H. Kazemi, Y. P. Fallah, M. Naserian, and A. Lewis, "System architecture for cooperative vehicle-pedestrian safety applications using DSRC communication," SAE Technical Paper, Tech. Rep., 2015.

[19] A. Tahmasbi-Sarvestani, H. N. Mahjoub, Y. P. Fallah, E. Moradi-Pari, and O. Abuchaar, "Implementation and evaluation of a cooperative vehicle-to-pedestrian safety application," *IEEE Intelligent Transportation Systems Magazine*, vol. 9, no. 4, pp. 62–75, 2017.

[20] J. B. Kenney, "Dedicated short-range communications (DSRC) standards in the united states," *Proceedings of the IEEE*, vol. 99, no. 7, pp. 1162–1182, 2011.

[21] C. Nowakowski, S. E. Shladover, D. Cody, F. Bu, J. O'Connell, J. Spring, S. Dickey, and D. Nelson, "Cooperative adaptive cruise control: Testing drivers' choices of following distances," Tech. Rep., 2011.

[22] B. Van Arem, C. J. Van Driel, and R. Visser, "The impact of cooperative adaptive cruise control on traffic-flow characteristics," *IEEE Transactions on intelligent transportation systems*, vol. 7, no. 4, pp. 429–436, 2006.

[23] J. Vander Werf, S. Shladover, M. Miller, and N. Kourjanskaia, "Effects of adaptive cruise control systems on highway traffic flow capacity," *Transportation Research Record: Journal of the Transportation Research Board*, no. 1800, pp. 78–84, 2002.

[24] M. Amoozadeh, H. Deng, C.-N. Chuah, H. M. Zhang, and D. Ghosal, "Platoon management with cooperative adaptive cruise control enabled by vanet," *Vehicular communications*, vol. 2, no. 2, pp. 110–123, 2015.

[25] G. J. Naus, R. P. Vugts, J. Ploeg, M. J. van de Molengraft, and M. Steinbuch, "String-stable cacc design and experimental validation: A frequency-domain approach," *IEEE Transactions on vehicular technology*, vol. 59, no. 9, pp. 4268–4279, 2010.

[26] B. M. Baumann, G. Washington, B. C. Glenn, and G. Rizzoni, "Mechatronic design and control of hybrid electric vehicles," *IEEE/ASME Transactions On Mechatronics*, vol. 5, no. 1, pp. 58–72, 2000.

[27] S. Delprat, J. Lauber, T.-M. Guerra, and J. Rimaux, "Control of a parallel hybrid powertrain: optimal control," *IEEE transactions on Vehicular Technology*, vol. 53, no. 3, pp. 872–881, 2004.

[28] M. Gokasan, S. Bogosyan, and D. J. Goering, "Sliding mode based powertrain control for efficiency improvement in series hybrid-electric vehicles," *IEEE Transactions on power electronics*, vol. 21, no. 3, pp. 779–790, 2006.

[29] C. Musardo, G. Rizzoni, Y. Guezennec, and B. Staccia, "A-ECMS: An addaptive algorithm for hybrid electric vehicle energy management," *European Journal of Control*, vol. 11, no. 4-5, pp. 509–524, 2005.

[30] J.-S. Won and R. Langari, "Intelligent energy management agent for a parallel hybrid vehicle-part ii: torque distribution, charge sustenance strategies, and performance results," *IEEE transactions on vehicular technology*, vol. 54, no. 3, pp. 935–953, 2005.

[31] J. P. Hespanha, "Modeling and analysis of networked control systems using stochastic hybrid systems," *Annual Reviews in Control*, vol. 38, no. 2, pp. 155–170, 2014.

[32] H. A. Blom and J. Lygeros, "Stochastic hybrid systems(theory and safety critical applications)," *Lecture notes in control and information sciences*, 2006.

[33] C. Cassandras and J. Lygeros, "Stochastic hybrid systems: Recent developments and research trends. number 24 in control engineering series," 2006.

[34] V. Milanés and S. E. Shladover, "Handling cut-in vehicles in strings of cooperative adaptive cruise control vehicles," *Journal of Intelligent Transportation Systems*, vol. 20, no. 2, pp. 178–191, 2016.

[35] G. Guo and W. Yue, "Sampled-data cooperative adaptive cruise control of vehicles with sensor failures," *IEEE Transactions on Intelligent Transportation Systems*, vol. 15, no. 6, pp. 2404–2418, 2014.

[36] J. D. S. Basav Sen and W. G. Najm, "Analysis of lane change crashes: (report no. dot hs 809 571)." *Washington, DC: National Highway Traffic Safety Administration*, 2003.

[37] S. E. Shladover, C. Nowakowski, X.-Y. Lu, and R. Ferlis, "Cooperative adaptive cruise control: definitions and operating concepts," *Transportation Research Record: Journal of the Transportation Research Board*, no. 2489, pp. 145–152, 2015.

[38] N. Oliver and A. P. Pentland, "Graphical models for driver behavior recognition in a smartcar," in *Intelligent Vehicles Symposium, 2000. IV 2000. Proceedings of the IEEE*. IEEE, 2000, pp. 7–12.

[39] P. Liu, A. Kurt *et al.*, "Trajectory prediction of a lane changing vehicle based on driver behavior estimation and classification," in *Intelligent Transportation Systems (ITSC), 2014 IEEE 17th International Conference on*. IEEE, 2014, pp. 942–947.

[40] S. Bonnin, F. Kummert, and J. Schmüdderich, "A generic concept of a system for predicting driving behaviors," in *Intelligent Transportation Systems (ITSC), 2012 15th International IEEE Conference on*. IEEE, 2012, pp. 1803–1808.

[41] H. M. Mandalia and M. D. D. Salvucci, "Using support vector machines for lane-change detection," in *Proceedings of the human factors and ergonomics society annual meeting*, vol. 49, no. 22. SAGE Publications Sage CA: Los Angeles, CA, 2005, pp. 1965–1969.

[42] B. Morris, A. Doshi, and M. Trivedi, "Lane change intent prediction for driver assistance: On-road design and evaluation," in *Intelligent Vehicles Symposium (IV), 2011 IEEE*. IEEE, 2011, pp. 895–901.

[43] R. S. Tomar, S. Verma, and G. S. Tomar, "Prediction of lane change trajectories through neural network," in *Computational Intelligence and Communication Networks (CICN), 2010 International Conference on*. IEEE, 2010, pp. 249–253.

[44] D. Kasper, G. Weidl, T. Dang, G. Breuel, A. Tamke, A. Wedel, and W. Rosenstiel, "Object-oriented bayesian networks for detection of lane change maneuvers," *IEEE Intelligent Transportation Systems Magazine*, vol. 4, no. 3, pp. 19–31, 2012.

[45] T. Gindele, S. Brechtel, and R. Dillmann, "A probabilistic model for estimating driver behaviors and vehicle trajectories in traffic environments," in *Intelligent Transportation Systems (ITSC), 2010 13th International IEEE Conference on*. IEEE, 2010, pp. 1625–1631.

[46] A. Pentland and A. Liu, "Modeling and prediction of human behavior," *Neural computation*, vol. 11, no. 1, pp. 229–242, 1999.

[47] N. Kuge, T. Yamamura, O. Shimoyama, and A. Liu, "A driver behavior recognition method based on a driver model framework," SAE Technical Paper, Tech. Rep., 2000.

[48] P. Liu and Ü. Özgüner, "Predictive control of a vehicle convoy considering lane change behavior of the preceding vehicle," in *American Control Conference (ACC), 2015*. IEEE, 2015, pp. 4374–4379.

[49] H. Berndt, J. Emmert, and K. Dietmayer, "Continuous driver intention recognition with hidden markov models," in *Intelligent Transportation Systems, 2008. ITSC 2008. 11th International IEEE Conference on*. IEEE, 2008, pp. 1189–1194.

[50] M. Mori, C. Miyajima, T. Hirayama, N. Kitaoka, and K. Takeda, "Integrated modeling of driver gaze and vehicle operation behavior to estimate risk level during lane changes," in *Intelligent Transportation Systems-(ITSC), 2013 16th International IEEE Conference on*. IEEE, 2013, pp. 2020–2025.

[51] D. Mitrovic, "Reliable method for driving events recognition," *IEEE transactions on intelligent transportation systems*, vol. 6, no. 2, pp. 198–205, 2005.

[52] J. B. Kenney, "Dedicated short-range communications (dsrc) standards in the united states," *Proceedings of the IEEE*, vol. 99, no. 7, pp. 1162–1182, 2011.

[53] S. Tsugawa, S. Jeschke, and S. E. Shladover, "A review of truck platooning projects for energy savings," *IEEE Transactions on Intelligent Vehicles*, vol. 1, no. 1, pp. 68–77, 2016.

[54] A. Kanaris, E. B. Kosmatopoulos, and P. A. Loannou, "Strategies and spacing requirements for lane changing and merging in automated highway systems," *IEEE transactions on vehicular technology*, vol. 50, no. 6, pp. 1568–1581, 2001.

[55] B. K. Chaurasia and S. Verma, "Haste induced behavior and VANET communication," in *Vehicular Electronics and Safety (ICVES), 2009 IEEE International Conference on*. IEEE, 2009, pp. 19–24.

[56] E. N. Barmpounakis, E. I. Vlahogianni, and J. C. Golias, "Decision trees and meta-algorithms for revealing powered two wheelers' overtaking patterns," *Transportation Research Board 96th Annual Meeting*, 2017.

[57] T. Wilson and W. Best, "Driving strategies in overtaking," *Accident Analysis & Prevention*, vol. 14, no. 3, pp. 179–185, 1982.

[58] D. Caveney, "Cooperative vehicular safety applications," *IEEE Control Systems*, vol. 30, no. 4, pp. 38–53, 2010.

[59] D. Yanakiev and I. Kanellakopoulos, "Nonlinear spacing policies for automated heavy-duty vehicles," *IEEE Transactions on Vehicular Technology*, vol. 47, no. 4, pp. 1365–1377, 1998.

[60] A. Khosroshahi, E. Ohn-Bar, and M. M. Trivedi, "Surround vehicles trajectory analysis with recurrent neural networks," in *Intelligent Transportation Systems (ITSC), 2016 IEEE 19th International Conference on*. IEEE, 2016, pp. 2267–2272.

[61] J. T. Connor, R. D. Martin, and L. E. Atlas, "Recurrent neural networks and robust time series prediction," *IEEE transactions on neural networks*, vol. 5, no. 2, pp. 240–254, 1994.

[62] A. Liu and D. Salvucci, "Modeling and prediction of human driver behavior," in *Intl. Conference on HCI*, 2001.

[63] D. Bezzina and J. Sayer, "Safety pilot model deployment: Test conductor team report (Report No. DOT HS 812 171)." *Washington, DC: National Highway Traffic Safety Administration*, 2015.

[64] B. Kouvaritakis and M. Cannon, *Model predictive control: Classical, robust and stochastic*. Springer, 2015.

[65] O. Gietelink, J. Ploeg, B. De Schutter, and M. Verhaegen, "Development of advanced driver assistance systems with vehicle hardware-in-the-loop simulations," *Vehicle System Dynamics*, vol. 44, no. 7, pp. 569–590, 2006.

[66] L. Tijerina, W. Garrott, M. Glecker, D. Stoltzfus, and E. Parmer, "Van and passenger car driver eye glance behavior during the lane change decision phase," in *Proceedings of the 84th Annual Meeting of the Transportation Research Board*, 2005.

[67] J.-P. Laumond, S. Sekhavat, and F. Lamiraux, "Guidelines in nonholonomic motion planning for mobile robots," *Robot motion planning and control*, pp. 1–53, 1998.

[68] A. Crowther and N. Zhang, "Torsional finite elements and nonlinear numerical modelling in vehicle powertrain dynamics," *Journal of Sound and Vibration*, vol. 284, no. 3-5, pp. 825–849, 2005.

[69] O. Hegazy and J. Van Mierlo, "Optimal power management and powertrain components sizing of fuel cell/battery hybrid electric vehicles based on particle swarm optimisation," *International Journal of Vehicle Design*, vol. 58, no. 2-4, pp. 200–222, 2012.

[70] A. Niknam and K. Farhang, "Vibration instability in a large motion bistable compliant mechanism due to stribeck friction," *Journal of Vibration and Acoustics*, vol. 140, no. 6, p. 061017, 2018.

[71] K. M. Lyon, "Mechatronic vehicle powertrain control system," Jul. 26 2005, uS Patent 6,920,865.

[72] A. Niknam and K. Farhang, "Friction-induced vibration due to mode-coupling and intermittent contact loss," *Journal of Vibration and Acoustics*, vol. 141, no. 2, p. 021012, 2019.

[73] S. D. Farrall, "Control of a vehicle powertrain," Aug. 12 1997, uS Patent 5,656,921.

[74] A. Niknam and K. Farhang, "On the passive control of friction-induced instability due to mode coupling," *Journal of Dynamic Systems, Measurement, and Control*.

[75] J. T. B. A. Kessels, M. W. T. Koot, P. P. J. van den Bosch, and D. B. Kok, "Online energy management for hybrid electric vehicles," *IEEE Transactions on vehicular technology*, vol. 57, pp. 3428–3440, 2008.

[76] X. Zhang, S. E. Li, H. Peng, and J. Sun, "Design of multimode power-split hybrid vehicles - a case study on the voltec powertrain system," *IEEE Transactions on Vehicular Technology*, vol. 65, no. 6, pp. 4790–4801, 2016.

[77] B. Gu and G. Rizzoni, "An addaptive algorithm for hybrid electric vehicle energy management based on driving pattern recognition," *ASME International Mechanical Engineering Congress and Exposition*, no. IMECE2006-13951, pp. 249–258, 2006.

[78] A. Chasse, G. Corde, A. D. Mastro, and F. Perez, "Online optimal control of a parallel hybrid with after-treatment constraint integration," *IEEE Vehicle Power and Propulsion Conference*, pp. 1–6, 2010.

[79] S. Xu, S. E. Li, H. Peng, B. Cheng, X. Zhang, and Z. Pan, "Fuel-saving cruising strategies for parallel hevs," *IEEE Transactions on Vehicular Technology*, vol. 65, no. 6, pp. 4676–4686, 2016.

[80] J. Liu and H. Peng, "Modeling and control of a power-split hybrid vehicle," *IEEE Transactions on Control Systems Technology*, vol. 16, no. 2, pp. 1242–1251, 2008.

[81] C. Lin, H. Peng, J. W. Grizzle, and J. Kang, "Power management strategy for a parallel hybrid electric truck," *IEEE Transactions on control systems technology*, vol. 11, no. 6, pp. 839–849, 2003.

[82] A. Brahma, Y. Guezennec, and G. Rizzoni, "Optimal energy management in series hybrid electric vehicles," *IEEE Proc. on American Control Conference*, vol. 1, pp. 60–64, 2000.

[83] L. Serrao, S. Onori, and G. Rizzoni, "Ecms as a realization of pontryagin's minimum principle for hev control," *IEEE Proc. on American Control Conference*, pp. 3964–3969, 2009.

[84] X. Zhang, S. E. Li, H. Peng, and J. Sun, "Efficient exhaustive search of power-split hybrid powertrains with multiple planetary gears and clutches," *Journal of Dynamic Systems, Measurement, and Control*, vol. 137, no. 12, 2015.

[85] F. A. Bender, M. Kaszynski, and O. Sawodny, "Drive cycle prediction and energy management optimization for hybrid hydraulic vehicles," *IEEE Transactions on Vehicular Technology*, vol. 62, no. 8, pp. 3581–3592, Oct 2013.

[86] D. Karbowski, V. Sokolov, and A. Rousseau, "Vehicle energy management optimisation through digital maps and connectivity," *22nd ITS World Congress*, Oct 2015.

[87] L. Fu, Ü. Özgüner, P. Tulpule, and V. Marano, "Real-time energy management and sensitivity study for hybrid electric vehicles," *IEEE Proc. American Control Conference*, pp. 2113–2118, 2011.

[88] D. Ambuehl and L. Guzzella, "Predictive reference signal generator for hybrid electric vehicles," *IEEE Transactions on Vehicular Technology*, vol. 58, no. 9, pp. 4730–4740, 2009.

[89] J. Junbo, "Vehicle fuel consumption optimization using model predictive control based on V2V communication," Master's thesis, Ohio State University, 2014.

[90] C. Sun, F. Sun, and H. He, "Investigating adaptive-ecms with velocity forecast ability for hybrid electric vehicles," *Applied Energy*, pp. –, 2016.

[91] S. D. Cairano, D. Bernardini, A. Bemporad, and I. V. Kolmanovsky, "Stochastic mpc with learning for driver-predictive vehicle control and its application to hev energy management," *IEEE Transactions on Control Systems Technology*, vol. 22, no. 3, pp. 1018–1031, May 2014.

[92] H. Kazemi, B. Khaki, Y. P. Fallah, A. Nix, and S. Wayne, "Utilizing situational awareness for efficient control of powertrains in parallel hybrid electric vehicles," *IEEE International Conference on Ubiquitous Wireless Broadband (ICUWB)*, pp. 1–5, 2015.

[93] F. Yan, J. Wang, and K. Huang, "Hybrid electric vehicle model predictive control torque-split strategy incorporating engine transient characteristics," *IEEE Transactions on Vehicular Technology*, vol. 61, no. 6, pp. 2458–2467, July 2012.

[94] M. Bidarvatan and M. Shahbakhti, "Energy management control of a hybrid electric vehicle by incorporating powertrain dynamics," *ASME 2015 Dynamic Systems and Control Conference*, vol. 1, Oct 2015.

[95] S. Yu, G. Dong, and L. LI, "Transient characteristics of emissions during engine start/stop operation employing a conventional gasoline engine for hev application," *International Journal of Automotive Technology*, vol. 9, no. 5, pp. 543–549, 2008.

[96] Mathworks, "Simscape user's guide," https://www.mathworks.com/help/pdf_doc/physmod/simscape/simscape_ug.pdf.

[97] M. Khodabakhshian, L. Feng, and J. Wikander, "Improving fuel economy and robustness of an improved ECMS method," in *2013 10th IEEE International Conference on Control and Automation (ICCA)*, 2013, pp. 598–603.

[98] S. Onori, L. Serraro, and G. Rizzoni, "Adaptive equivalent consumption minimization strategy for hybrod electric vehicles," *ASME Dynamic Systems and Control Conference*, vol. 1, no. DSCC2010-4211, pp. 499–505, 2010.

[99] J. Jing, A. Kurt, E. Ozatay, J. Michelini, D. Filev, and U. Ozguner, "Vehicle speed prediction in a convoy using v2v communication," *IEEE 18th International Conference on Intelligent Transportation Systems*, 2015.

[100] D. Moser, H. Waschl, R. Schmied, and H. Efendic, "Short term prediction of a vehicle's velocity trajectory using its," *SAE Int. J. Passeng. Cars Electron. Electr. Syst.*, pp. 364–370, 2015.

[101] I. Bosankic, L. Banjanovic-Mehmedovic, and F. Mehmedovic, "Speed profile prediction in intelligent transport systems exemplified by vehicle to vehicle interactions," *Cybernetics and Information Technologies*, vol. 15, no. 5, pp. 63–77, 2015.

[102] B. Jiang and Y. Fei, "Vehicle speed prediction by two-level data driven models in vehicular networks," *IEEE Transactions on Intelligent Transportation Systems*, vol. PP, no. 99, pp. 1–9, 2016.

[103] J. Huang and H.-S. Tan, "Vehicle future trajectory prediction with a dgps/ins-based positioning system," *American Control Conference*, pp. 6 pp.–, 2006.

[104] S. Qiao, N. Han, W. Zhu, and L. A. Gutierrez, "Traplan: An effective three-in-one trajectory-prediction model in transportation networks," vol. 16, no. 3, pp. 1188–1198, 2015.

[105] A. Sciarretta, M. Back, and L. Guzzella, "Optimal control of parallel hybrid electric vehicles," *IEEE Transactions on Control Systems Technology*, vol. 12, no. 3, pp. 352–363, 2004.

[106] DOT, "Saftey pilot model deployment data," https://www.its-rde.net/index.php/data/explore-rde-data/10018-safety-pilot-model-deployment-data.

[107] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[108] H. Kazemi, S. Soleymani, F. Taherkhani, S. Iranmanesh, and N. Nasrabadi, "Unsupervised image-to-image translation using domain-specific variational information bound," in *Advances in Neural Information Processing Systems*, 2018, pp. 10 348–10 358.

[109] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *European conference on computer vision*. Springer, 2016, pp. 21–37.

[110] H. Kazemi, S. M. Iranmanesh, and N. Nasrabadi, "Style and content disentanglement in generative adversarial networks," in *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 2019, pp. 848–856.

[111] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3431–3440.

[112] F. Taherkhani, N. M. Nasrabadi, and J. Dawson, "A deep face identification network enhanced by facial attributes prediction," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2018, pp. 553–560.

[113] F. Taherkhani and M. Jamzad, "Restoring highly corrupted images by impulse noise using radial basis functions interpolation," *IET Image Processing*, vol. 12, no. 1, pp. 20–30, 2017.

[114] V. Talreja, F. Taherkhani, M. C. Valenti, and N. M. Nasrabadi, "Using deep cross modal hashing and error correcting codes for improving the efficiency of attribute guided facial image retrieval," in *2018 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*. IEEE, 2018, pp. 564–568.

[115] F. Taherkhani, H. Kazemi, and N. M. Nasrabadi, "Matrix completion for graph-based deep semi-supervised learning," in *Thirty-Third AAAI Conference on Artificial Intelligence*, 2019.

[116] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529, 2015.

[117] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[118] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," in *ICML*, 2014.

[119] H. Banvait, S. Anwar, and Y. Chen, "A rule-based energy management strategy for plug-in hybrid electric vehicle (phev)," in *2009 American control conference*. IEEE, 2009, pp. 3938–3943.

[120] C.-C. Lin, H. Peng, J. W. Grizzle, and J.-M. Kang, "Power management strategy for a parallel hybrid electric truck," *IEEE transactions on control systems technology*, vol. 11, no. 6, pp. 839–849, 2003.

[121] L. V. Pérez, G. R. Bossio, D. Moitre, and G. O. García, "Optimization of power management in an hybrid electric vehicle using dynamic programming," *Mathematics and Computers in Simulation*, vol. 73, no. 1-4, pp. 244–254, 2006.

[122] G. Paganelli, M. Tateno, A. Brahma, G. Rizzoni, and Y. Guezennec, "Control development for a hybrid-electric sport-utility vehicle: strategy, implementation and field test results (i)," in *Proceedings of the American Control Conference*, vol. 2, 2001, pp. 5064–5064.

[123] X. Lin, Y. Wang, P. Bogdan, N. Chang, and M. Pedram, "Reinforcement learning based power management for hybrid electric vehicles," in *Proceedings of the 2014 IEEE/ACM International Conference on Computer-Aided Design*. IEEE Press, 2014, pp. 32–38.

[124] A. Vogel, D. Ramachandran, R. Gupta, and A. Raux, "Improving hybrid vehicle fuel efficiency using inverse reinforcement learning," in *Twenty-Sixth AAAI Conference on Artificial Intelligence*, 2012.

[125] Y. Wang, X. Lin, M. Pedram, and N. Chang, "Joint automatic control of the powertrain and auxiliary systems to enhance the electromobility in hybrid electric vehicles," in *2015 52nd ACM/EDAC/IEEE Design Automation Conference (DAC)*. IEEE, 2015, pp. 1–6.

[126] X. Lin, P. Bogdan, N. Chang, and M. Pedram, "Machine learning-based energy management in a hybrid electric vehicle to minimize total operating cost," in *2015 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, 2015, pp. 627–634.

[127] H. He, C. Sun, and X. Zhang, "A method for identification of driving patterns in hybrid electric vehicles based on a lvq neural network," *Energies*, vol. 5, no. 9, pp. 3363–3380, 2012.

[128] Y. Tian, X. Zhang, L. Zhang, and X. Zhang, "Fuzzy control strategy for hybrid electric vehicle based on neural network identification of driving conditions," *Control Theory & Applications*, vol. 28, no. 3, pp. 363–369, 2011.

[129] Y. Gurkaynak, A. Khaligh, and A. Emadi, "Neural adaptive control strategy for hybrid electric vehicles with parallel powertrain," in *2010 IEEE Vehicle Power and Propulsion Conference*. IEEE, 2010, pp. 1–6.

[130] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 1334–1373, 2016.

[131] M. P. Deisenroth, G. Neumann, J. Peters *et al.*, "A survey on policy search for robotics," *Foundations and Trends® in Robotics*, vol. 2, no. 1–2, pp. 1–142, 2013.

[132] P. Wawrzyński, "Real-time reinforcement learning by sequential actor–critics and experience replay," *Neural Networks*, vol. 22, no. 10, pp. 1484–1497, 2009.

[133] P. WawrzyńSki and A. K. Tanwani, "Autonomous reinforcement learning with experience replay," *Neural Networks*, vol. 41, pp. 156–167, 2013.

[134] D. Balduzzi and M. Ghifary, "Compatible value gradients for reinforcement learning of continuous deep policies," *arXiv preprint arXiv:1509.03005*, 2015.

[135] N. Heess, G. Wayne, D. Silver, T. Lillicrap, T. Erez, and Y. Tassa, "Learning continuous control policies by stochastic value gradients," in *Advances in Neural Information Processing Systems*, 2015, pp. 2944–2952.

[136] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," *arXiv preprint arXiv:1506.02438*, 2015.

[137] S. Levine, N. Wagener, and P. Abbeel, "Learning contact-rich manipulation skills with guided policy search," in *2015 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2015, pp. 156–163.

[138] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, no. 3, pp. 279–292, 1992.

[139] B. Bakker, "Reinforcement learning with long short-term memory," in *Advances in neural information processing systems*, 2002, pp. 1475–1482.

[140] N. Heess, J. J. Hunt, T. P. Lillicrap, and D. Silver, "Memory-based control with recurrent neural networks," *arXiv preprint arXiv:1512.04455*, 2015.

Chapter 6

Appendices

Appendix A

The model predictive control law finds the optimal input sequence $\boldsymbol{u}^*[k]$ which minimize a predicted cost function $J[k]$ at each time instant:

$$\boldsymbol{u}^*[k] = \arg\min_u J[k] \tag{6.1}$$

$$\text{subject to} \quad \begin{matrix} x_{min} \leq x[k+i|k] \leq x_{max} \\ u_{min} \leq u[k+i|k] \leq u_{max} \end{matrix} \quad i = 1, \ldots, N-1$$

In the receding horizon implementation of MPC, only the first element of this optimal input sequence is selected as the input to the system and the whole process repeats at each time instant. However, designing a receding horizon controller based on a finite-horizon cost function does not guarantee the stability and optimality of the closed loop system. This problem can be avoided by defining an infinite prediction horizon for performance evaluation cost:

$$J(k) = \sum_{i=0}^{\infty} [x^T(k+i|k)Qx(k+i|k) + u^T(k+i|k)Ru(k+i|k)] \tag{6.2}$$

However to keep the number of MPC optimization problem finite, a dual-mode prediction approach can be utilized in which the predicted input sequence is defined as follows

$$u(k+i|k) = \begin{cases} \text{optimization variables} & i = 0, 1, ..., N-1 \\ Kx(k+i|k) & i = N, N+1, ... \end{cases} \tag{6.3}$$

Then by choosing the terminal weighting matrix $\bar{Q}$ in a way that $x^T(k+N|k)\bar{Q}x(k+N|k)$ is equal to the cost over the second mode of the predicted input sequence, the infinite cost (6.2) can be rewritten as

$$\begin{aligned} J(k) = &\sum_{i=0}^{N-1} [x^T(k+i|k)Qx(k+i|k) + u^T(k+i|k)Ru(k+i|k)] \\ &+ x^T(k+N|k)\bar{Q}x(k+N|k) \end{aligned} \tag{6.4}$$

Theorem 3 *The performance cost (6.4) is equal to the infinite cost (6.2) under the control law (6.3) when $\bar{Q}$ and K are the solutions of the Riccati equations*

$$\begin{aligned} K &= (R + B^T\bar{Q}B)^{-1}B^T\bar{Q}A \\ \bar{Q} &- (A+BK)^T\bar{Q}(A+BK) = Q + K^TRK \end{aligned} \tag{6.5}$$

To solve the MPC problem (6.1), it should first converted into a Quadratic Programming (QP) Problem by rewriting the cost function (6.4) in a compact form

$$J(k) = \boldsymbol{u}^T[k]H\boldsymbol{u}[k] + 2x^T[k]F^T\boldsymbol{u}[k] + x^T[k]Gx[k] \tag{6.6}$$

where

$$H = \mathcal{C}^T\tilde{Q}\mathcal{C} + \tilde{R}, \qquad F = \mathcal{C}^T\tilde{Q}\mathcal{M}, \qquad G = \mathcal{M}^T\tilde{Q}\mathcal{M} + Q$$

with $\mathcal{M} = [A, A^2 \dots A^N]^T$, $\tilde{R} = diag[R, R \dots R]$, $\tilde{Q} = diag[Q \dots Q, \bar{Q}]$ and the convolution matrix $\mathcal{C}$ is defined by

$$\mathcal{C} = \begin{bmatrix} B & 0 & \dots & 0 \\ AB & B & \dots & 0 \\ \vdots & \vdots & \ddots & \\ A^{N-1}B & A^{N-2}B & \dots & B \end{bmatrix}$$

Then, the MPC problem (6.1) is equivalent to the following QP problem

$$\begin{aligned} \boldsymbol{u}^*[k] = \arg\min_{u} \quad & \boldsymbol{u}^T H \boldsymbol{u} + 2x^T[k] F^T \boldsymbol{u}[k] \\ \text{subject to} \quad & A_c \boldsymbol{u} \leq b_0 + B_x x[k] \end{aligned} \tag{6.7}$$

where

$$A_c = \begin{bmatrix} I_{n_u} \\ -I_{n_u} \\ \mathcal{C} \\ -\mathcal{C} \end{bmatrix}, \qquad b_0 = \begin{bmatrix} I_{n_u} u_{max} \\ -I_{n_u} u_{min} \\ I_N x_{max} \\ -I_N x_{min} \end{bmatrix}, \qquad B_x = \begin{bmatrix} \mathbf{0}_{n_u} \\ \mathbf{0}_{n_u} \\ -A \\ A \end{bmatrix}$$

Appendix B

As a direct consequence of the more general condition for Cauchys series convergence test, we know that for any infinite sequence $a(0), a(1), \ldots$, if $\sum_{k=0}^{n} a(k)$ tends to finite limit as $n \to \infty$, then $a(k) \to 0$ as $k \to \infty$.

From that, we can show that if there exists a continuously differentiable positive definite scalar function $V(x)$ such that $V(f(x)) - V(x) \leq -l(x) \leq 0$, then $l(x(k))$ as $k \to \infty$.

Proof: Since $V(f(x)) - V(x) \leq -l(x) \leq 0$ implies $l(x(k)) \leq V(x(k)) - V(x(k+1))$, and by summing the both sides of this inequality over $k = 0, 1, \ldots$ we have

$$\sum_{k=0}^{\infty} l(x(k)) \leq V(x(0)) - \lim_{k \to \infty} V(x(k)) \tag{6.8}$$

The right hand side of this inequality is finite as $V(x(k)) \geq 0$ and $V(x(k+1)) \leq V(x(k))$ imply that $V(x(k))$ converge to a finite limit when $k \to \infty$. Consequently, $l(x) \to 0$.

Now with the assumption of MPC optimization problem being feasible, i.e., at all time steps there exist predicted input and state trajectories that satisfy the constraints, then the MPC cost function is a function of $x(k)$ or equivalently $J^*(k) = V(x(k))$. Now, to satisfy the positive definiteness condition of $V(x(k))$, either of the following conditions should be held:

- Q is positive definite.
- $(A, Q^{\frac{1}{2}})$ is an observable pair, where $Q^{\frac{1}{2}}$ s any matrix with $Q^{\frac{1}{2}^T} Q^{\frac{1}{2}} = Q$.

When the second condition is met, then $J^*(k) = 0$ implies $\| Q^{\frac{1}{2}} x(k+i|k) \|_2^2 = 0, \quad i = 0, 1, \ldots$ and since $(A, Q^{\frac{1}{2}})$ is observable, we have $x(k) = 0$.

Appendix C

This section lists the codes developed in MATLAB for the designed MPC controller for CACC to handle interfering vehicles.

Listing 6.1: Matlab code for CACC

```
function [delta_out, delta_v_out, a_out, u_out, pr_out] =
    CACC_MPC_hybrid(pr, pr_last, h, d0, delta_in, delta_v_in, a_in, u_in
    , t, v_platoon, l_b)
% h: time gap
% pr: probability of a cut-in
% pr_last: last step probability of a cut-in
% delta_in: last step delta
% delta_v_in: last step delta_v
% a_in: last step a
% u_in: last step input
% [delta_out, delta_v_out, a_out] = CACC(pr, pr_last, h, d0, delta_in,
    delta_v_in, a_in, u_in, t)

ci = 0.25;
N = 30;
Nc = 23;
delta_t = 0.1;  %Sample Time
Len = 0;

%% Controller and System Modeling

% System is controllable since rank(ctrb(A0,B0)) = 3
A0 = [1,delta_t,-h*delta_t;0,1,-delta_t;0,0,1-delta_t/ci];
B0 = [0;0;delta_t/ci];
G0 = [0;delta_t;0];

G1 = [G0,zeros(3,N-1)];
G2 = [zeros(N,1),[eye(N-1);zeros(1,N-2),0.1]];

```

```
27 A = [A0,G1;zeros(N,3),G2];
28 B = [B0;zeros(N,1)];
29
30 c_b = [1,zeros(1,length(A)-1);0,1,zeros(1,length(A)-2);0,0,1,zeros(1,
       length(A)-3)];    %y = c_b * x
31
32 x_max = [5;5;5;100*ones(N,1)];
33 x_min = [-5;-5;-5;-100*ones(N,1)];
34 x_max = [l_b;15;9;100*ones(N,1)];
35 x_min = [-l_b/2;-5;-8;-100*ones(N,1)];
36 u_max = 3;
37 u_min = -5;
38
39 mu = [];
40 C = [];
41 Ac = [];
42 b0 = [];
43 Bx = [];
44
45 for i = 1:N
46         mu = [mu;A^i];
47         Bx = [Bx;-A^i;A^i];
48 end
49
50 for i = 1:N
51         Ci = [];
52         for j = 1:N
53                 if(j==i)
54                         Ci = [Ci;B];
55                 elseif(j>i)
56                         Ci = [Ci;(A^(j-i))*B];
57                 else
58                         Ci = [Ci;zeros(size(B))];
59                 end
60         end
```

```
61          b0 = [b0;x_max;-x_min];
62          C = [C,Ci];
63  end
64
65  Q = c_b'*c_b;
66  R = 0.01;
67
68  [Q_b,L,k] = dare(A,B,Q,R);
69
70  temp = repmat({Q}, 1, N-1);
71  temp(end+1) = {Q_b};
72  Q_t = blkdiag(temp{:});
73
74  temp = repmat({R}, 1, N);
75  R_t = blkdiag(temp{:});
76  H = C' * Q_t * C + R_t;
77  H = (H+H')/2;
78  G = mu' * Q_t * mu + Q;
79  F = C' * Q_t * mu;
80
81  %rank(obsv(A+B*k, c_b))
82
83  Ac = [[eye(N);-eye(N)];Ac];
84
85  for i = 1:N
86          start = 1 + (i-1) * length(B);
87          Ac = [Ac;C(start:start+length(B)-1,:);-C(start:start+length(B)
                -1,:)];
88  end
89
90  b0 = [ones(N,1)*u_max; -ones(N,1)*u_min; b0];
91  Bx = [zeros(2*N,size(Bx,2));Bx];
92
93  %% Run the platoon system
94
```

```
95  a0 = 0;                          %Initial Acceleration of the lead RV
96  v0 = v_platoon;                  %Initial Velocity of the lead RV %30
97  delta = [0, delta_in];           %Initial Spacing Error
98  delta_v = [0, delta_v_in];       %Initial Velocity Difference (V_i - V_(
        i-1))
99  a = [a0, a_in];                  %Initial Acceleration of all 4 vehicles
100 v = [v0,v0 - delta_v(2)];
101
102 index = 1;
103
104 A = [0,1,-h;0,0,-1;0,0,-1/ci];
105 B = [0;0;1/ci];
106 G = [0;1;0];
107 u = [0,0,0,0,0];
108 a(1) = 0;
109
110 %update velocity of the i-th HV vehicle
111 v(2) = v(2) + a(2) * delta_t;
112
113 if(v(2) < 0)
114         v(2) = 0;
115 end
116
117 delta_delta = (1 + pr) * (h * v(2) + Len + d0) - (1 + pr_last) * (h * v
        (2) + Len + d0);
118
119 delta(2) = delta(2) - delta_delta;
120 x_u = [delta(2),delta_v(2),a(2),a(1)*ones(1,N)]';
121
122 % Solving the QP problem
123 temp = quadprog(H, F*x_u, Ac, b0 + Bx * x_u);
124
125 %update states of the i-th HV vehicle
126 x = [delta(2),delta_v(2),a(2)]';
127 delta_x = (A * x + B * u_in + G * a(1))*delta_t;
```

```
128 x = x + delta_x;
129 y = [x;a(1)];
130
131 if length(temp)>0
132         u_out = temp(1);
133 else
134         if delta(2)< 0
135                 u_out = -5;
136         else
137                 u_out = 3;
138         end
139 end
140
141 delta_out = x(1);
142 delta_v_out = x(2);
143 a_out = x(3);
144 pr_out = pr;
145
146 end
```

Appendix D

This section lists the codes developed in MATLAB/Simulink for the HEV simulation.

Listing 6.2: Implementation of AECMS

```
function [P_em_opt, P_ice_opt, error] = AECMS(P_req, s, P_em_min,
    speed_em_RPM, P_em_max, P_ice_max, speed_ice_RPM, eng_spd, eng_trq,
    eng_map, mot_spd, mot_trq, mot_map, charge_sustaining, ess_volt,
    ess_max_curr_charge, ess_max_curr_discharge)

% P_req: requested power by the driver
% s: Equivalent factor
% P_em_min: Minimum instantaneous EM power
% speed_em_RPM: Instantaneous EM speed
% P_em_max: Maximum instantaneous EM power
% P_ice_max: Maximum instantaneous ICE power
% speed_ice_RPM: Instantaneous ICE speed
% eng_spd: Speed axis for ICE efficiency map
% eng_trq: Torque axis for ICE efficiency map
% eng_map: ICE efficiency map
% mot_spd: Speed axis for EM efficiency map
% mot_trq: Torque axis for EM efficiency map
% mot_map: EM efficiency map
% charge_sustaining: Check if we are in charge_sustaining mode to use
    AECMS
% ess_volt: ESS voltage
% ess_max_curr_charge: ESS max charging current
% ess_max_curr_discharge: ESS max discharging voltage

P_em_opt = 0;
P_ice_opt = 0;
error = 0;

if(charge_sustaining > 0)
delta_P = 200;
```

```
27 J_min = 1e10;
28
29 P_req_mot_max = P_req;
30 if (P_req > P_em_max)
31 P_req_mot_max = P_em_max;
32 end
33
34 P_em_vec = linspace(P_em_min,P_req_mot_max,delta_P);
35 P_ice_vec = P_req - P_em_vec;
36 P_em = P_em_vec(P_ice_vec <= P_ice_max);
37 P_ice = P_ice_vec(P_ice_vec <= P_ice_max);
38
39 if(speed_ice_RPM > 590 && ~isempty(P_ice))
40 [P_e, curr] = P_ech(P_em,speed_em_RPM, mot_spd, mot_trq, mot_map,
      ess_volt);
41 if (sum(curr >= ess_max_curr_discharge)==0)
42 P_e = 0;
43 P_f = 0;
44 error = 1;
45 P_em = P_em_min;
46 elseif(sum(curr <= ess_max_curr_charge)==0)
47 P_e = 0;
48 P_f = 0;
49 error = 1;
50 P_ice = min(P_ice_max,P_req);
51 %P_em = min([ess_max_curr_charge * ess_volt, P_req - P_ice, P_em_max]);
52 P_em = min([ess_max_curr_charge * ess_volt, P_req - P_ice, P_em_max]);
53 else
54 P_f = P_fuel(P_ice,speed_ice_RPM, eng_spd, eng_trq, eng_map);
55 P_e = P_e(curr <= ess_max_curr_charge & curr >= ess_max_curr_discharge)
      ;
56 P_f = P_f(curr <= ess_max_curr_charge & curr >= ess_max_curr_discharge)
      ;
57 P_em = P_em(curr <= ess_max_curr_charge & curr >=
      ess_max_curr_discharge);
```

```
58 P_ice = P_ice(curr <= ess_max_curr_charge & curr >=
       ess_max_curr_discharge);
59 end
60 J = P_f + s * P_e;
61
62 J_min = min(J);
63 P_em = P_em(J == J_min);
64 P_ice = P_ice(J == J_min);
65 if(~isempty(P_em))
66 P_em_opt = P_em(1);
67 P_ice_opt = P_ice(1);
68 else
69 P_em_opt = min(P_em_max,P_req);
70 P_ice_opt = 0;
71 end
72 else
73 P_em_opt = min(P_em_max,P_req);
74 P_ice_opt = 0;
75 end
76 end
77 end
78
79 function p_f = P_fuel(P_ice, speed_ice, eng_spd, eng_trq, eng_map)
80 speed_ice_rad = speed_ice * (2 * pi / 60);
81 T_ice = P_ice/(speed_ice_rad);
82
83 m_dot = interp2(eng_trq, eng_spd, eng_map, T_ice, speed_ice*ones(1,
       length(T_ice)));
84
85 fuel_density_val            = 0.801;          % kg/L
86 fuel_heating_val             = 29047000;     % (J/kg)Specific LHV
87 Q_fuel = fuel_density_val*fuel_heating_val/1000;    % J/g
88
89 p_f = m_dot * Q_fuel;            % m_dot is the fuel rate and Q_fuel is
       the fuel energy density
```

```
90 end
91
92 function [P_e curr] = P_ech(P_em, speed_em, mot_spd, mot_trq, mot_map,
       ess_volt)
93 if(speed_em > 0)
94 speed_em_rad = speed_em * (2 * pi / 60);
95 T_em = abs(P_em/(speed_em_rad));
96
97 efficiency = interp2(mot_trq, mot_spd, mot_map', T_em, speed_em*ones(1,
       length(T_em)))/100;
98 P_em(P_em >= 0) = P_em(P_em >= 0) ./ efficiency(P_em >= 0);
99 P_em(P_em < 0) = P_em(P_em < 0) .* efficiency(P_em < 0);
100 P_e = P_em;
101 curr = -P_em / ess_volt;
102 else
103 curr = 0;
104 P_e = 0 * P_em;
105 end
106 end
```

Listing 6.3: Implementation of PECMS

```
1 function [s_p,s_n,E0_all,t_last_out]= Equivalent_factor(t_start,
      gear_num , Sch_Cycle, pedal, E_0_last, eng_spd, eng_trq, eng_map,
      mot_spd, mot_trq, mot_map, eng_trq_max_map, eng_trq_max_spd,
      mot_trq_max_map, mot_trq_max_spd, ess_volt, t_last, s_n_last,
      s_p_last)%,E_old_in)
2 %%
3 all_E_e_p =[0 0 0 0 0 0];
4 all_E_f_p =[0 0 0 0 0 0];
5
6 all_E_e_n =[0 0 0 0 0 0];
7 all_E_f_n =[0 0 0 0 0 0];
8
9 E_e_p_old = [0 0 0 0 0 0];
10 E_e_n_old = [0 0 0 0 0 0];
```

```
11 E_f_p_old = [0 0 0 0 0 0];
12 E_f_n_old = [0 0 0 0 0 0];
13
14 m = 1700;
15 grade = 0;
16 R_Wheel = 0.353;
17 C_r = 0.009;
18 Drag_coefficient = 0.372;
19 g = 9.81;
20 Diff_ratio = 2.77;
21
22 t_lookahead = 5;
23 time_sps = 10;
24
25 i_p = 1;
26 i_n = 1;
27
28 s_p = s_p_last;
29 s_n = s_n_last;
30 E0_all = E_0_last;
31
32 %E_old_out = E_old_in;
33
34 if pedal > 0.005
35 %calculate s
36 if t_start - t_last> 1
37 t_last
38 t_last = t_start;
39 for u = -1 : 0.1 : 1
40 if((u>-0.6 && u < 0.6) || u == -1 || u == 1)
41 t_num = time_sps * t_lookahead;
42 gear = gear_num;
43 E_f = 0;
44 E_e = 0;
45
```

```
46 if(any(E_e_p_old) || any(E_f_p_old))
47 t_num0 = time_sps * (t_lookahead - (t_start - t_last));
48 else
49 t_num0 = 0;
50 end
51
52 %                  t_last = t_start;
53
54 for t_index = t_num0 : 1 : t_num
55 sim_t = t_start + t_index/time_sps;
56 vehicle_speed = interp1(Sch_Cycle(:,1),Sch_Cycle(:,2),sim_t) * 0.44704;
       % m/s
57 vehicle_speed_next = interp1(Sch_Cycle(:,1),Sch_Cycle(:,2),sim_t+1/
     time_sps)* 0.44704;% m/s
58 vehicle_accel = (vehicle_speed_next - vehicle_speed) * time_sps;
59 gear = transmission(gear,vehicle_speed/0.44704);
60
61 w_e = vehicle_speed*60*Diff_ratio*gear_selector(gear)/(R_Wheel*2*pi);
62 w_m = vehicle_speed*60*Diff_ratio*2.5/(R_Wheel*2*pi);
63
64 if vehicle_speed > 0
65 tau_d = (m * vehicle_accel + C_r*m*g*cosd(grade) + vehicle_speed^2 *
     Drag_coefficient)*R_Wheel;
66 else
67 tau_d = (m * vehicle_accel +  vehicle_speed^2 * Drag_coefficient)*
     R_Wheel;
68 end
69 tau_d = tau_d / (Diff_ratio);
70 tau_e = 0;
71
72 max_mot_trq = interp1(mot_trq_max_spd, mot_trq_max_map, w_m);
73
74 if(w_e > 550)
75 max_eng_trq = interp1(eng_trq_max_spd, eng_trq_max_map, w_e);
76
```

```
77 if (tau_d>=0)
78 tau_m = u * tau_d / 2.5;
79 tau_e = (tau_d - tau_m * 2.5)/gear_selector(gear);
80 if(tau_e > max_eng_trq)
81 continue
82 end
83 else
84 tau_m = tau_d;
85 tau_e = 0;
86 end
87
88 if (tau_m > max_mot_trq)
89 continue
90 end
91
92 P_f = 0;
93 if(tau_e ~= 0)
94 P_f = P_fuel(tau_e,w_e, eng_spd, eng_trq, eng_map)/time_sps;
95 end
96 else
97 P_f = 0;
98 if (tau_d>=0)
99 tau_m = min(tau_d/2.5,max_mot_trq);
100 else
101 tau_m = max(tau_d/2.5,-max_mot_trq);
102 end
103 end
104
105 [P_e, curr] = P_ech(tau_m, w_m, mot_spd, mot_trq, mot_map, ess_volt);
106
107 E_e = E_e + P_e/time_sps;
108 E_f = E_f + P_f;
109 end
110
111 if u == 0
```

```
112 all_E_e_p(i_p) = E_e;% + E_e_p_old(i_p);
113 all_E_e_n(i_n) = E_e;% + E_e_n_old(i_n);
114 all_E_f_p(i_p) = E_f;% + E_f_p_old(i_p);
115 all_E_f_n(i_n) = E_f;% + E_f_n_old(i_n);
116 i_p = i_p + 1;
117 i_n = i_n + 1;
118 E0_all = E_e;
119 elseif ( u > 0 && u < 0.7)
120 all_E_e_p(i_p) = E_e;% + E_e_p_old(i_p);
121 all_E_f_p(i_p) = E_f;% + E_f_p_old(i_p);
122 i_p = i_p + 1;
123 elseif( u < 0 && u > -0.7)
124 all_E_e_n(i_n) = E_e;% + E_e_n_old(i_n);
125 all_E_f_n(i_n) = E_f;% + E_f_n_old(i_n);
126 i_n = i_n + 1;
127 end
128 end
129 end
130 if all(all_E_e_p == all_E_e_p(1))
131 s_p = s_p_last;
132 s_n = s_n_last;
133 else
134 s_p_t = polyfit(all_E_e_p,all_E_f_p,1);
135 s_p = -s_p_t(1);
136 s_n_t = polyfit(all_E_e_n,all_E_f_n,1);
137 s_n = -s_n_t(1);
138 end
139 E_e_p_old = all_E_e_p;
140 E_e_n_old = all_E_e_n;
141 E_f_p_old = all_E_f_p;
142 E_f_n_old = all_E_f_n;
143 else
144 s_p = s_p_last;
145 s_n = s_n_last;
146 E0_all = E_0_last;
```

```
147 end
148 else
149 s_p = 0;
150 s_n = 0;
151 E0_all = 0;
152 end
153 t_last_out = t_last;
154 end
155
156 function ratio = gear_selector(gear)
157 ratios = [4.6200, 3.0400, 2.0700, 1.6600, 1.2600, 1, 0.8500, 0.6600];
158 ratio = ratios(gear);
159 end
160
161 function current_m = current_calc(elec_pow_m,batt_volt_oc,batt_r)
162 current_m = -elec_pow_m / batt_volt_oc;
163 batt_volt = batt_volt_oc + batt_r * current_m;
164 current_m = -elec_pow_m / batt_volt;
165 batt_volt = batt_volt_oc + batt_r * current_m;
166 current_m = -elec_pow_m / batt_volt;
167 end
168
169
170 function p_f = P_fuel(tau_e, w_e, eng_spd, eng_trq, eng_map)
171 m_dot = interp2(eng_trq, eng_spd, eng_map, tau_e, w_e);
172
173 fuel_density_val          = 0.801;          % kg/L
174 fuel_heating_val           = 29047000;    % (J/kg)Specific LHV
175 Q_fuel = fuel_density_val*fuel_heating_val/1000;   % J/g
176
177 p_f = m_dot * Q_fuel;          % m_dot is the fuel rate and Q_fuel is
    the fuel energy density
178 end
179
180 function [P_e curr] = P_ech(tau_m, w_m, mot_spd, mot_trq, mot_map,
```

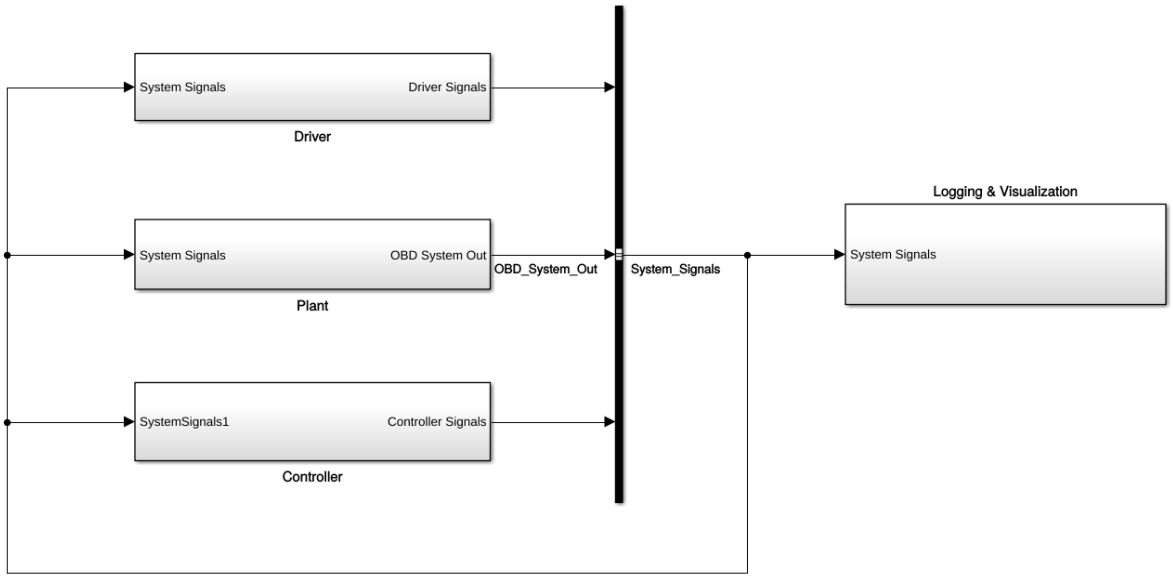


Figure 6.1: Simulink model of HEV module.

```
    ess_volt)
181 if(w_m > 0)
182 speed_em_rad = w_m * (2 * pi / 60);
183
184 efficiency = interp2(mot_trq, mot_spd, mot_map', abs(tau_m), w_m)/100;
185 if(tau_m >= 0)
186 P_em = tau_m * speed_em_rad / efficiency;
187 else
188 P_em = tau_m * speed_em_rad * efficiency;
189 end
190 P_e = P_em;
191 curr = -P_em / ess_volt;
192 else
193 curr = 0;
194 P_e = 0;
195 end
196 end
```

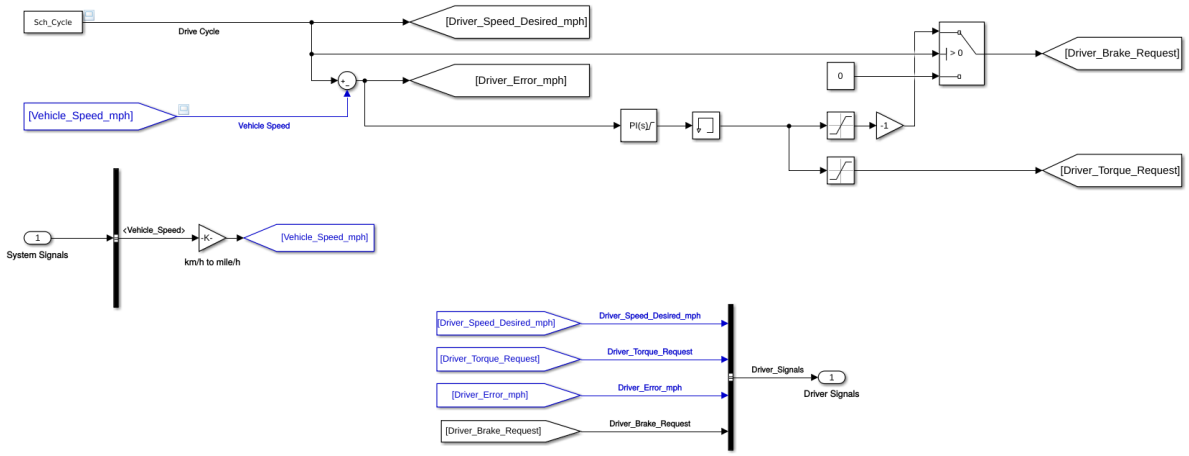


Figure 6.2: Simulink model of driver module.

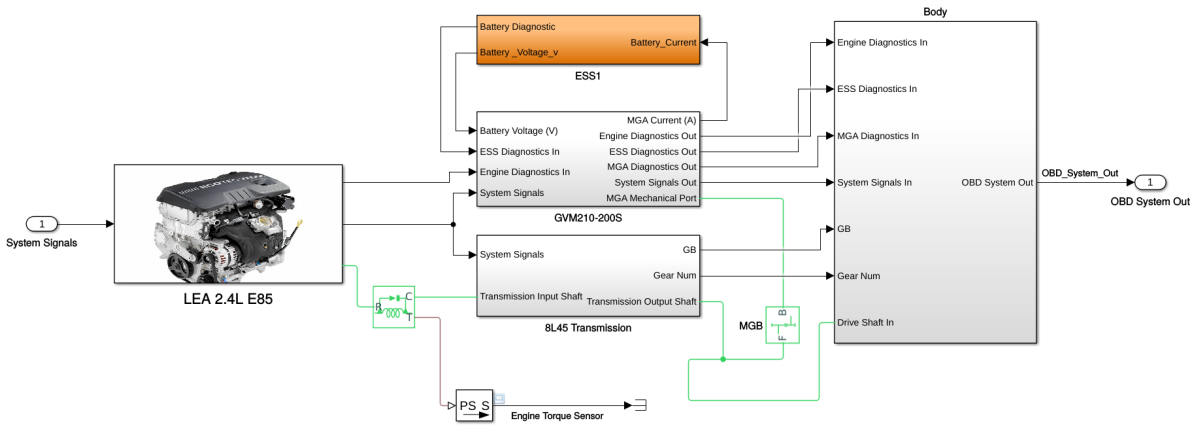


Figure 6.3: Simulink model of plant module.

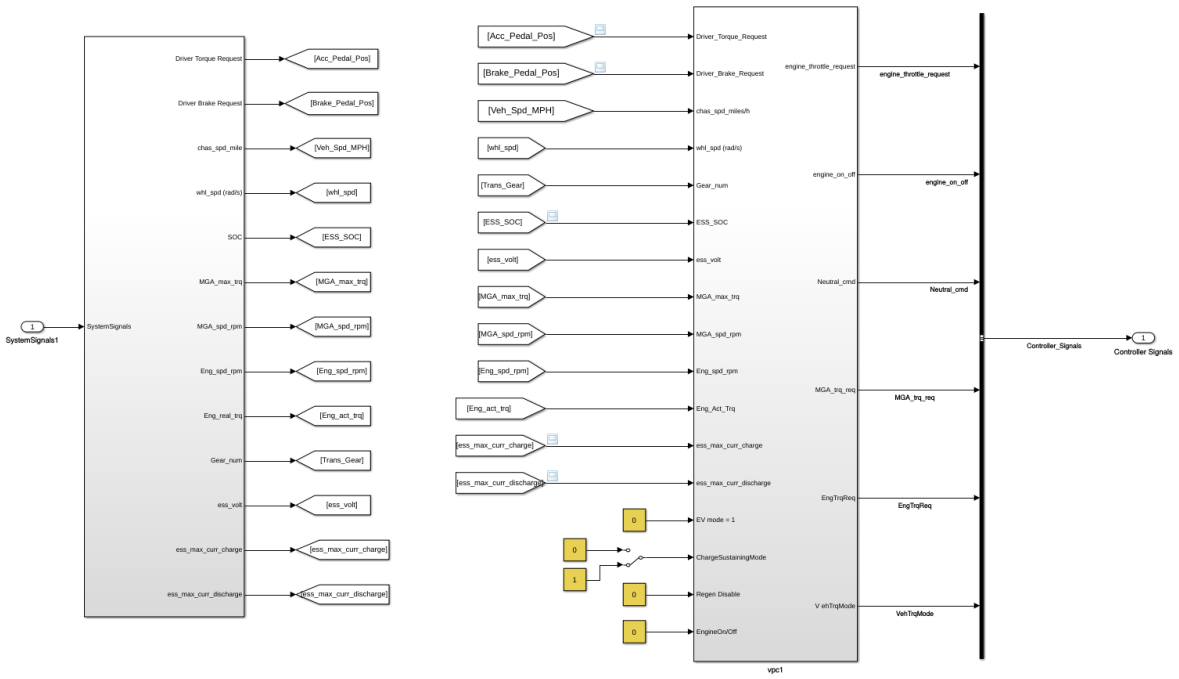


Figure 6.4: Simulink model of controller module.

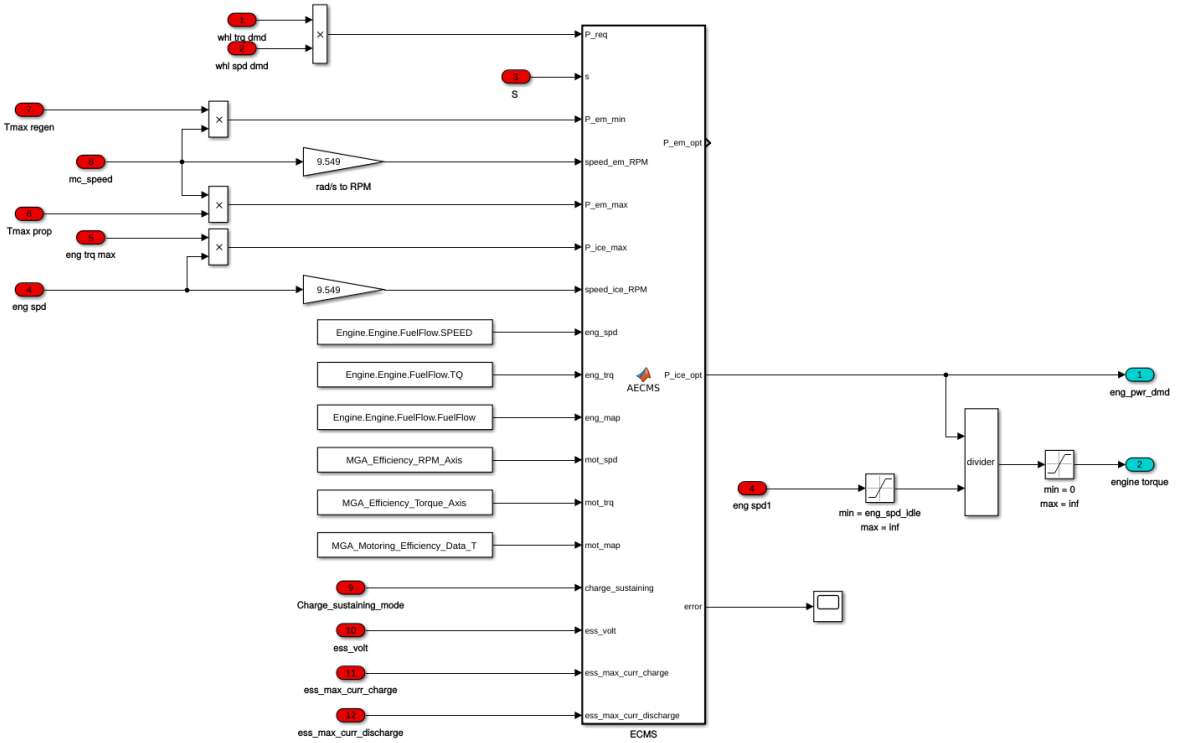


Figure 6.5: Simulink model of AECMS module.

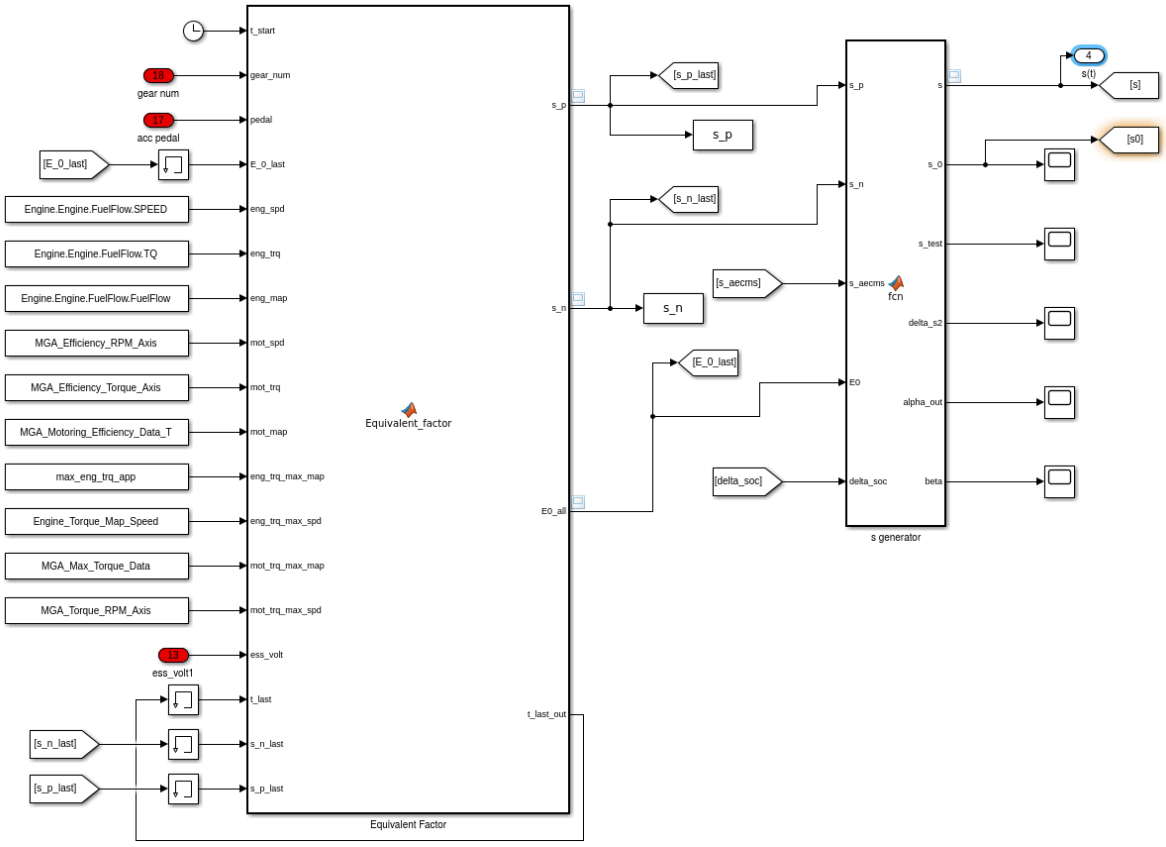


Figure 6.6: Simulink model of PECMS module.