Automatic Code Generation of Real-Time Nonlinear Model Predictive Control for Plug-in Hybrid Electric Vehicle Intelligent Cruise Controllers

by

Sadegh Tajeddin

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Systems Design Engineering

Waterloo, Ontario, Canada, 2016

Abstract

Control systems have always been a vital part of the novel technological advancements of human being in any industry, especially transportation. With the introduction of the idea of autonomous driving, classical control systems are not effective anymore and the need for intelligent control systems is inevitable. Advanced Driver Assistance Systems (ADASs), which are systems proposed to help drivers improve the process of driving, and Intelligent Transportation Systems (ITS), which are proposed to provide information that promotes more coordinated and more ecological driving, require novel intelligent controllers that are adaptive to driving conditions. Therefore, the development of different strategic vehicle control systems by employing state-of-the-art intelligent control methods has been an active field of research in recent years.

The highly variant nature of transportation implies that an effective intelligent control technique must be able to handle a large multi-input multi-output (MIMO) system with nonlinear complex dynamics. It must also store and analyse a large amount of data and information about the vehicle, its environment and traffic conditions in the process of decision-making. Nonlinear Model Predictive Control (NMPC), as a unique optimal model-based approach to intelligent control systems design, is a promising candidate that comprises all of these characteristics. The ability to solve constrained multi-objective optimization problems with a predictive approach has made this technique powerful. However, NMPC controller developers face real-time implementation challenges as this method suffers from huge computational loads. Hence, fast Real-Time Optimization (RTO) methods are proposed to overcome this drawback. Optimization methods based on Generalized Minimum Residual (GMRES) method are examples of these RTO algorithms that have shown great potential for real-time applications such as vehicle control.

This thesis investigates the potential of employing GMRES-based RTO algorithms to design intelligent vehicle control systems, in particular intelligent cruise controllers. Plug-in Hybrid Electric vehicles (PHEVs) are introducing themselves as the future solutions for green and ecological transportation, the thesis also introduces an intelligent cruise controller for the Toyota Prius 2013 PHEV. To this end, an automatic multi-solver NMPC code generator based on GMRES-based RTO algorithms is developed in MATLAB. The user-friendly environment of this code generation tool allows the user to easily generate NMPC controller codes for further model-in-the-loop (MIL) and hardware-in-the-loop (HIL) simulations. Simulations are performed for two different driving scenarios: driving on hilly roads and a car-following scenario. In the case of driving on hilly roads, a comparative study is conducted between different real-time optimizers and it is concluded that the Newton/GMRES algorithm is faster than the Continuation/GMRES algorithm.

A novel adaptive prediction horizon length approach is also developed to enhance the performance of the NMPC controller. Simulation results demonstrate a minimum of 3.4% energy consumption improvement as compared to a PID controller performance as well as improvement of reference speed tracking when using an adaptive prediction horizon length. In case of the car-following scenario, the effect of several tuning parameters and adaptive gains on the performance of the proposed NMPC controller is studied. Then the ecological adaptive cruise controller was tested on urban and highway driving cycles, and resulted in 3.4% and 1.2%, respectively, improvement in the cost of the trip. Finally, the proposed NMPC controllers for both intelligent cruise control systems are tested on an HIL platform for rapid control prototyping. The HIL results on a dSPACE prototype Electronic Control Unit (ECU) indicate that the real-time optimizers and the proposed NMPC controllers are fast enough to be implementable on an actual ECU for a certain range of prediction horizon sizes.

Acknowledgements

I would like to thank my supervisors, Prof. Azad and Prof. Fraser who guided me through this research.

I acknowledge the NSERC and Toyota for financially supporting this work.

I would also like to thank all of my friends in my research group, specially Dr. Vajedi, who helped me and made this possible.

Dedication

This thesis is dedicated to my brilliant, loving and supportive wife, Mina, and to my wonderful encouraging parents.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Ever since the invention of machines, searching for and employing effective control techniques has been an inevitable part of technology. Today, it is the key to the creation of fully automated machines. Transportation, the inseparable need of men, is one of the essential areas where control plays a vital role in achieving great technological advancements in terms of safety, efficiency and comfort. The growing automotive industry is today intertwined in everyone's life and directly affecting it. The increase in the number of vehicles and demands for travelling at higher speeds has made concerns about safety greater than ever. Moreover, rising efficiency expectations demand more fuel-efficient vehicles than conventional gasoline-powered ones. Engine emissions that contribute to air pollution are another concern about such vehicles. All of these demands require more-advanced intelligent control techniques to be integrated into future vehicles to enhance different aspects of transportation including safety, efficiency, drivability and comfort.

American Corporate Average Fuel Economy (CAFE) as well as other standards require automakers to drastically reduce the emission level of their vehicles and increase their efficiency. The common Internal Combustion Engine (ICE) vehicles have an efficiency of at most 30%, which has brought most of the major automakers to the conclusion that they must develop electric-powered vehicles with energy efficiencies of as high as 70%. An Electric Vehicle (EV) utilizes one or more electric motor for propulsion instead of an ICE and consumes electric power from the grid. This solution, in addition to achieving higher energy-efficiency levels, reduces the need for fossil-based energy sources as the electric energy can be generated from renewable sources. However, EVs have not captured a proper market share because of their low operational range. Therefore, Hybrid Electric Vehicles (HEVs) and recently Plug-in Hybrid Electric vehicles (PHEVs) have attracted the attention of the automotive companies. The advantages of driving fully electric (zero emissions) alongside the capability of going longer ranges with low emission levels have made such vehicles promising sustainable solutions towards environmentally-friendlier transportation.

1.1 Motivation and Challenges

Recently, Intelligent Transportation Systems (ITS) have been proposed to provide transportation information to users and vehicles by employing several communication and information technologies for safer, more-coordinated and, in general, "smarter" driving. However, all such helpful information about modes of transportation and traffic is useless without a powerful, robust and fast decision maker on-board to analyse it. Such an intelligent strategic module can be achieved through the use of state-of-the-art control design techniques and theories. Therefore, only with the existence of a fast strategic high level controller that receives all the information, analyses it and generates intelligent control actions in real-time, can we move towards the goal of autonomous vehicles.

Intelligent control is a term mostly used when a control method is utilized to emulate the essential characteristics of human intelligence, including adaptation, anticipation and learning, in the process of generating control actions [1]. The term has been used in many areas, including neural network control, fuzzy control, genetic algorithms and machine learning [2]. In vehicle control, intelligent control is used in the sense that the controlled vehicle is aware of and adaptive to its environment [3]. Optimality has always been an ideal goal in engineering, and prediction is a basic principle that helps track optimal behaviour. Model Predictive Control (MPC) is an exceptional control method that aims to optimize current actions by taking future events into account. MPC has the ability to anticipate the future states of a system based on a dynamical model of it and generate an optimal input action sequence for a finite prediction horizon [4].

MPC technique is a superior method in optimal and intelligent control of future vehicles, as compared to easily-implementable rule-based controllers. This superiority rises from its capability of integrating current and future information about traffic and environments into the process of optimal decision-making. Nevertheless, accuracy is a vital factor in the stability and reliability of MPC controllers. A non-accurate prediction of the system caused by a non-accurate model of the system can lead to catastrophic consequences. Therefore, scientists and engineers are investigating the employment of more-accurate and nonlinear system models inside MPC controllers. Nonlinear MPC (NMPC) is the term used for such controllers [5, 6]. Unfortunately, NMPC controllers are very challenging to implement on fast systems such as automotive systems due to computational speed restrictions. Thus, considering the volume of data and computational load involved in the process of optimization inside an NMPC controller, searching for a fast and efficient optimizer is essential to overcoming the implementation challenges of NMPC. This study investigates the potential of GMRES-based optimization methods, as fast optimizers for use inside NMPC controllers for automotive applications. The multi-objective nature of the optimization problems of interest, i.e. intelligent cruise control systems, has encouraged the author to investigate the potential of applying MPC controllers in real-time.

The principles of a typical MPC framework are based on the optimization of a sequence of control actions in a finite-time prediction horizon. MPC uses a control-oriented model of the system for this purpose. To this end, a state feedback loop provides the MPC with a feedback signal of the states at each sampling time, and the optimizer written at the heart of the controller code solves a constrained optimization problem, repeatedly. Only the first optimal control action is then applied to the system. This control loop repeats over and over [7].

Despite the many advantages of the MPC technique, it suffers from extensive computational load, which has made it popular primarily among scholars dealing with chemical processes because of these processes relatively slower dynamics. With the introduction of faster processors and more efficient computation algorithms, MPC has found its way into other applications with faster dynamics. Automotive systems are not excluded from this trend and, recently, there have been increasing studies on MPC-based control systems for automotive applications. For practical implementation of MPC, different methods have been proposed, among them explicit MPC (eMPC) and Real-Time Optimization (RTO) methods. This thesis targets the RTO algorithms for synthesizing MPC controllers for nonlinear cruise control systems.

1.2 Problem Statement and Proposed Approach

One of the essential tools required for the design of real-time MPC controllers for nonlinear systems is an automatic NMPC code generator. A tool that, once the optimal control problem is defined, can automatically generate the controller and optimizer code for the user, thus saving significant amounts of time and effort for coding and debugging. In this thesis, a MATLAB-based automatic code-generation tool for intelligent NMPC controllers with GMRES-based real-time optimizers is developed with the purpose of designing intelligent cruise control systems for vehicles and presenting a comparison between the performance of different real-time optimizers. The code generator is user-friendly and the generated MATLAB code can be easily programmed onto hardware using the MATLAB C-code generation tool. The effect of applying adaptive prediction horizon length on the performance of the NMPC controllers is investigated as well. To this end, a set of the following actions must be considered:

1.2.1 Automatic Code Generation of Real-Time NMPC

An essential tool required for synthesizing real-time MPC controllers for nonlinear systems is an automatic code generation tool. Real-Time Optimization (RTO) methods for NMPC combine offline and online calculations and impose the preprocessing of an optimal control problem to transform it into a set of iterative online computations. Therefore, synthesizing, calibrating and enhancing an efficient real-time NMPC controller is practically impossible without the existence of an automatic code generator that generates numerous codes in a short time. Thus, the development of a valid automatic code generator for real-time NMPC is a vital task for this research. The developed code generator must possess certain characteristics as follows:

- Generates real-time NMPC code for several different optimal control problems regardless of its specifications, such as the number of states, inputs, objectives, etc.
- Generates an implementable controller code separate from the simulation code for further controller evaluations in Model-in-the-Loop (MIL) simulations.
- Handles various constant and time-varying user-defined parameters
- Has good compatibility with the MATLAB and Simulink modelling and simulation environments
- Allows users to generate an embeddable C code for practical testing of the controller, such as in Hardware-in-the-Loop (HIL) tests
- Has a user-friendly interface for problem definition

1.2.2 Control-oriented Modelling

Since the MPC technique requires a sufficiently accurate model of the dynamics of the controlled system, for each multi-objective optimal control problem, a control-oriented model of the system must be developed. This task must be done considering the dynamics of the problem of interest and the specifications of the controlled vehicle. It is important to note that a control-oriented model is real-time applicable as long as is simple and fast enough. In this work, simple longitudinal vehicle dynamics equations are employed for development of the control-oriented models.

1.2.3 Control Design

With a control-oriented model of the system and a code generator, we can move forward to design an NMPC controller. A suitable objective function with different cost terms representing different optimization objectives must be defined. Also, any equality or inequality constraint on the inputs or states must be taken into account. Last but not least, several weighting and tuning factors must be defined for the purpose of calibration in the later stages of the design. In order to make the controller adaptive to different circumstances and increase the robustness of the design, some of the weighting factors are made adaptive in this work.

1.2.4 Control Scheme Evaluation

In the last phase of the work, the generated NMPC controllers need to be evaluated. In this thesis, this task is performed through two sets of simulations. At first, the performance of the NMPC controllers are evaluated in an MIL simulation. In this phase, different optimization methods and real-time solvers are compared in terms of accuracy and speed. Also investigated are the effects of different factors on the performance of the controllers. Secondly, the controller codes are embedded on an HIL device to check the implementability of the developed NMPC controllers in a more practical test. For both of these simulations, a high-fidelity model of the intended PHEV is used as the controlled plant.

1.3 Thesis Organization

This thesis is organized in 5 chapters. Chapter 2 gives a brief on the background and a literature review on the topics covered in the rest of the thesis. Chapter 3 provides a discussion on the theory of optimal control and the real-time optimization methods and algorithms. The rest of the third chapter is dedicated to the description and instructions of the developed automatic code generation tool. Additionally, the code generator performance is validated with the results of another known code generator.

Chapter 4 consists of two parts and each part describes details of development of an intelligent cruise control system for a specific optimal control problem. The first part is specified to an Ecological Cruise Controller (ECC) for a PHEV driving on hilly roads. In this part, a comparison between different optimizers are also presented. In the second part, an Ecological Adaptive Cruise Controller (EcoACC) is designed for a PHEV in a car-following scenario. Two different sets of simulations are performed in this chapter: MIL simulations for controller performance evaluations and calibration; and HIL simulations for investigating real-time implementability of these systems. Finally, conclusions and future work are presented in Chapter 5.

Chapter 2

Literature Review and Background

2.1 Model Predictive Control: Theory and Methods

MPC is a closed-loop implementation of constrained optimal control. Figure 2.1 illustrates the block diagram of a system with an MPC controller. Basically, an MPC controller includes a simple yet efficient internal dynamic model of the system, called control-oriented model. It receives a history of past states and control actions through the feedback loop and aims to minimize a cost function over a finite prediction horizon. Moreover, ability to control Multi-Input Multi-Output (MIMO) systems, ability to handle equality and inequality constraints and ability to integrate the future information into the calculation of the control sequence, make MPC a powerful tool in controlling complex systems.

As shown in Figure 2.2, an MPC controller, at each sampling time, calculates the values of the manipulated variables (typically control sequence) at the next N_c timesteps in a way that the error between the predicted outputs and the reference trajectory is minimized for the next N_p timesteps, where N_c and N_p are the lengths of Control Horizon and Prediction Horizon, respectively. Then, the first calculated control input is implemented and the prediction horizon moves one sampling time forward and then this control process repeats. Therefore, MPC is also usually referred to as Receding Horizon Control [8].

Obviously, the accuracy and speed of an MPC controller greatly depends on the control-oriented model built within its structure. The more accurately the control-oriented model represents the dynamics of the controlled system, the more accurate are the MPC predictions and hence its performance is improved. However, accurate control-oriented models add complexity, nonlinearity and eventually more computational load to the problem

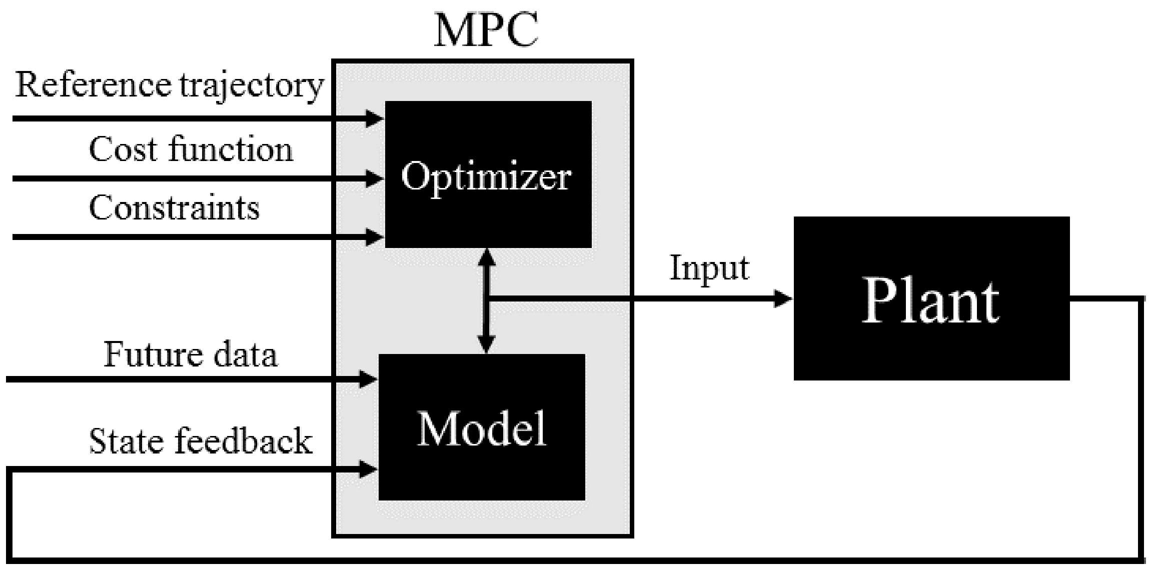


Figure 2.1: Block diagram representation of a Model Predictive Controller

which, in practice, makes the implementation of the MPC controller more challenging, specially in controlling fast processes.

Nonlinear MPC (NMPC) is a modification of MPC that uses a nonlinear control-oriented model at its heart for prediction [8, 5]. Sensitivity of the stability and performance of the NMPC highly depends on the accuracy of the prediction and therefore this matter encouraged scientists to develop several methods of NMPC. However, as mentioned earlier, huge computational load of NMPC in each sampling time makes it greatly challenging to implement in fast applications. With recent improvements in computational hardware, engineers are hopeful that NMPC technique can be employed in faster applications [9, 10], specially, automotive control [11, 12, 13]. For a recent solution, in order to overcome the implementation challenges of NMPC researchers proposed explicit MPC (eMPC) as an alternative to online approaches [14]. In an explicit approach, the nonlinear optimization problem is solved offline to eliminate the need for an online optimization solver. Thus, the controller can be generated as a piecewise affine function [15, 16]. In this method, though, the optimal solution is outputed as a look-up table of linear gains that must be stored and engineers must face data storage challenges rather than computational ones to meet the limitations of the automotive electronic control unit (ECU) [17].

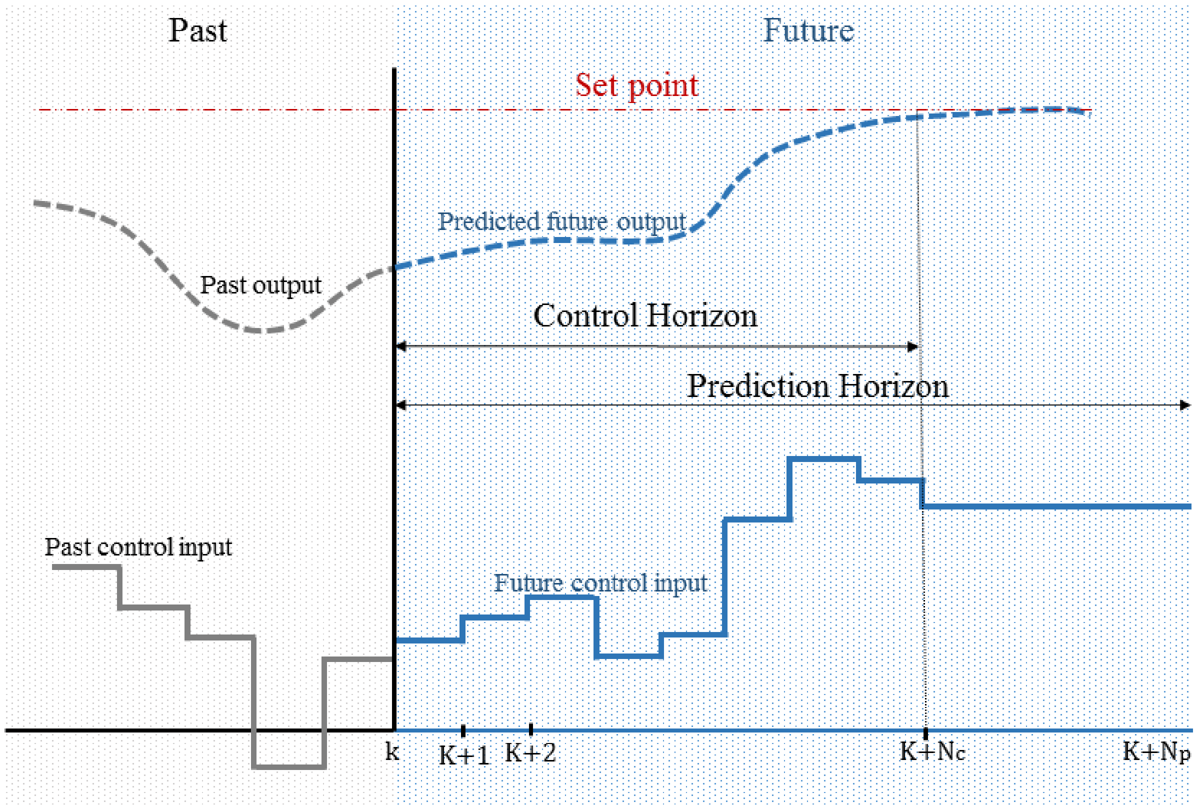


Figure 2.2: Principle of a Model Predictive Control

2.2 Automatic Code Generation for NMPC

Fast online optimization methods are another solution towards making MPC implementable in practice [18]. In such techniques, combination of offline calculations and fast online solvers are used instead of classical iterative optimizers to search for and find the optimal point. To exploit online optimization methods for NMPC controller, automatic code-generation tools are required. AutoGenU [19], based on Continuation/Generalized Minimal Residuals (C/GMRES) method, and ACADO [20], based on Real-Time Iteration (RTI), are the two existing automatic NMPC code generators. GMRES method is a Krylov subspace method for solving nonsymmetric systems which has a significant advantage in computation time [21]. Therefore, GMRES-based optimization methods have been introduced as potentially promising methods for use in NMPC [22].

As described in the the following chapter, generally, an NMPC controller is a closed-loop feedback control for nonlinear systems that solves a finite-horizon nonlinear optimal control problem over a finit predicition window in real-time at each sampling time. Such a problem can be solved through the Euler-Lagrange equations [23]. Since these nonlinear equations cannot be solved analytically, numerical methods have been proposed for solving them [24, 25]. In recent years, many works have studied the numerical methods for Real-Time Optimization (RTO) of NMPC controllers [26, 27, 28].

For real-time implementation of these methods, different automatic code generation software tools have been developed in order to avoid significant repetitive effort of coding and programming for various problems. AutoGenU [29], developed by Cybernet in collaboration with Ohtsuka, is an NMPC code generation tool that utilizes symbolic computation language, Mathematica, in Maple environment to generate a C code. This software tool works based on C/GMRES RTO algorithm proposed by Ohtsuka in [28]. In this program, once the user defines the optimal problem and related settings, AutoGenU can process the settings and generate the C code for simulation. The toolkit for automatic control and dynamic optimization (ACADO) is another software that can be used to generate Gauss-Newton real-time iteration algorithm codes for NMPC controllers [30]. ACADO is based on C++ and has a user-friendly MATLAB interface.

2.3 Hybrid Powertrain: Architecture and Control

Newly commercialized hybrid vehicles including HEVs and PHEVs have a unique powertrain architecture. In this section, a brief introduction is given on some variety of structure of HEVs. Generally, there are three different common known architectures of HEVs,

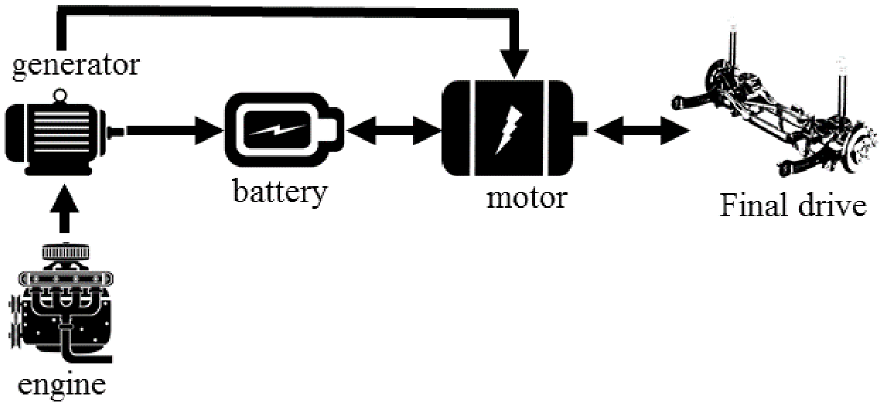


Figure 2.3: Series hybrid architecture

namely series, parallel and power-split. Each of these architectures have advantages and disadvantages while the power-split architecture claims to gather the best aspects of the other two.

In series configuration, as shown in Fig. 2.3, the electric motor directly drives the wheels whereas the engine is disconnected from the final drive and runs a generator to recharge the battery upon depletion. This way, since the engine is separated from the wheels, it can be operated in the high efficiency working points and therefore minimize emissions and fuel consumption. Moreover, during braking, portion of the kinetic energy usually dissipated through heat can be restored to the battery through the means of Regenerative Braking System (RBS). Having multiple stages of energy conversion, this architecture's most concerning drawback is its relatively low energy efficiency.

Despite series architecture, in parallel HEVs, a mechanical coupling connects the two sources of propulsion, the engine and the motor. As the engine is connected to the final transmission in this type of HEVs, currently, in commercialized cars the motor only assists the engine and is not the main source of motive power. Although utilizing an electric motor during propulsion and RBS during braking has increased the efficiency of parallel hybrid systems as compared to other non-hybrid vehicles, the coupled engine does not allow the controllers to freely operate it mostly inside its sweet spot.

The power-split architecture is believed to capture the advantages of the other two architectures described above. As shown in Fig. 2.5, the engine, the generator and the electric motor are all connected to each other with a planetary gear set, called the power-

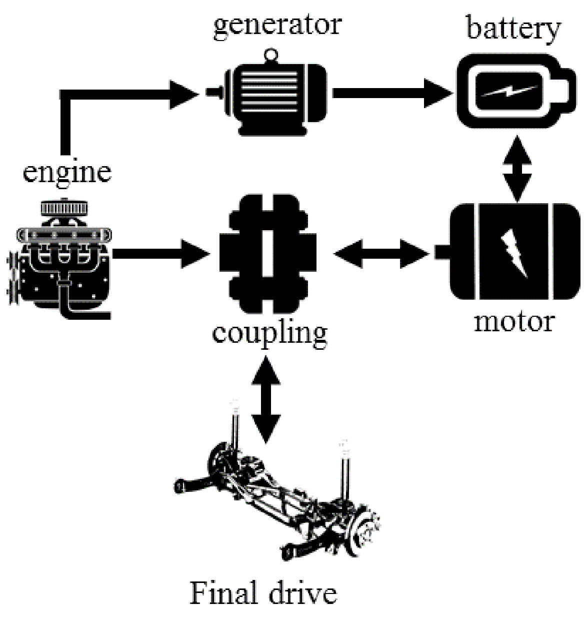


Figure 2.4: Parallel hybrid architecture

split device. The power-split device divides the engine power along two paths: one directly goes to the wheels and the other runs the generator to recharge the battery. Decoupling the engine from the motor in this configuration has led to an additional freedom in control which gives superiority to this architecture.

PHEVs combine two sources of power: battery and gasoline, which adds a complexity to the powertrain and requires optimization of decision-making process. Development of optimal supervisory controllers for improvement of PHEV's efficiency and taking the best advantage out of its integrated structure [31] has defined various research topics in the field of control and energy management [32, 33, 34]. For example, several different control techniques (rule-based, stochastic, model-based, etc.) have been conducted to design an optimal Energy Management System (EMS) for a HEV/PHEV [35, 36, 37]. In another work, Dynamic Programming (DP) has been employed to optimally manage battery charge-depletion in a PHEV using trip information [38]. Also, a recent study has targeted optimization of the battery management system of such vehicles using online model-based approaches [10]. While all of these brilliant control strategies demonstrate promising improvements in simulations, their developers are faced up against implementation challenges and limitations. Today's vehicle hardware limitations and very high speed of events in driv-

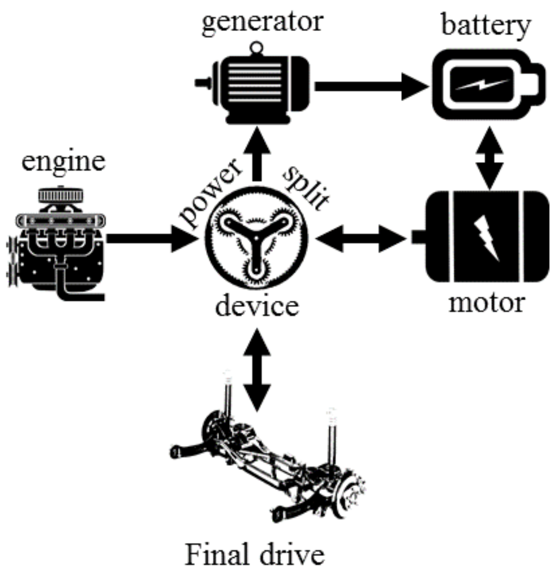


Figure 2.5: Power-split hybrid architecture

ing causes severe implementation challenges for intelligent controllers with huge loads of computation. This study addresses these challenges in the way of developing fast and efficient model-based controllers for automotive applications.

2.4 Intelligent Cruise Control Systems

With the introduction of ITS and integrating it with intelligent control for ecological driving, namely eco-driving, designing ecological vehicle controllers that drive the vehicle in its ecological state has always been in the center of attention [39, 40, 41]. Applying ITS and optimal control to improve efficiency is not bounded only to vehicle control as, for instance, in [42], traffic flow information collected from ITS has been used to optimally control time duration of the green phase of traffic signals. Additionally, Ecological Cruise Controllers (ECCs) and Ecological Adaptive Cruise Controllers (EcoACCs), as examples of application of intelligent control systems in eco-driving, have been an active area of research in recent years. For example, the authors in [43] utilize traffic signal and road slope information to design a predictive ecological strategic controller for an HEV. In a similar study, upcoming traffic signal information is used to develop an Adaptive Cruise

Control (ACC) to minimize fuel consumption by reducing idling time at stop lights [40].

There has been much literature on development of ECC and EcoACC systems for different types of vehicles with the MPC technique [44, 45, 46]. Ability of the MPC controllers to include future traffic information in optimization of control actions has encouraged many scholars to pursue the development of EcoACC systems using this technique. An analysis of the Advanced Driving Assistant Systems (ADAS) and simulation comparisons between EcoACC and ACC systems are presented in [47] which show higher fuel efficiency for an EcoACC system. A C/GMRES RTO approach has been conducted to design an MPC controller for improved fuel economy in [48]. Kamal *et al.* simulated their MPC system in a comparative study using traffic simulator to demonstrate a significant improvement in overall fuel consumption of an ICE vehicle driving an urban drive cycle. In another study by Kamal *et al.*, it has been shown that using C/GMRES based MPC controller to make one single host vehicle smart and cooperative with its human-driven preceding and following vehicles through the means of Cooperative Adaptive Cruise Control (CACC) will result in a significant improvement of the traffic flow in the following traffic [49].

2.5 Summary

The latter technologies and advancements become more vital in EVs, HEVs and PHEVs as fuel-dependency and range anxiety are two important concerns on the way of development and commercialization of such vehicles, see [50, 51, 52]. In the case of HEVs and PHEVs, complexity and integration of different sources of power requires state-of-the-art optimal control strategies in order to optimize driving and distribute power optimally between the existing sources. Although the amount of research work done in this area is not abundant like ICE vehicles, some studies have investigated application of MPC for nonlinear systems for optimal and smart control of HEVs and PHEVs [44, 16, 53]. In a recent study, an EcoACC system is developed based on NMPC technique for a PHEV with a trip planing module [44]. The presented simulation results for two different driving scenarios; car-following and driving over a hill, demonstrated up to 19% energy cost improvement. The rarity of reports on the application of real-time implementable NMPC-based EcoACC systems for PHEVs in the literature has driven the author to investigate the applicability of GMRES-based RTO algorithms to the design of EcoACC systems for these vehicles.

Chapter 3

Automatic Code Generation Tool

This chapter is dedicated to the developed Automatic Multi-solver NMPC Code Generator, named AuMuSoN. AuMuSoN is a mathematical program in MATLAB that collects all the necessary information about the optimal control problem, such as dynamics of the system, objective function, constraints, etc., plus the solver options set by the user and then automatically generates an NMPC controller for both Model-In-the-Loop (MIL) and Hardware-In-the-Loop (HIL) purposes.

3.1 Optimal Control Theory

According to Optimal Control theory, an optimization problem of the form shown in (3.1), can be transformed into a root finding problem of a set of nonlinear equations through the means of applying necessary conditions for optimality, see [28]. Consider the following objective function in terms of states vector, $\mathbf{x}(t)$, and inputs vector, $\mathbf{u}(t)$,

$$J = \Phi(\mathbf{x}(t+T), \mathbf{p}(t+T)) + \int_{t}^{t+T} Ł(\mathbf{x}(\tau,t), \mathbf{u}(\tau,t), \mathbf{p}(\tau,t))d\tau. \tag{3.1}$$

subjected to

$$\dot{\mathbf{x}}(\tau,t) = \mathbf{f}(\mathbf{x}(\tau,t), \mathbf{u}(\tau,t), \mathbf{p}(t+\tau))$$
$$\mathbf{g}(\mathbf{x}(\tau,t), \mathbf{u}(\tau,t), \mathbf{p}(t+\tau)) = 0.$$

Where T is the prediction window, t is time, $\phi(.)$ is the terminal cost at the end of the prediction horizon, $Ł(.)$ is the trajectory cost, $\mathbf{f}(.)$ denotes dynamics of the system and $\mathbf{g}(.)$

refers to the equality constraints. Since it is clear that in a dynamic problem states, inputs and time-varying parameters are functions of time, hereafter, t is excluded in notation for simplicity. Let us define τ as the prediction time domain and discretise the problem over the prediction horizon into N timesteps. With $\Delta\tau$ as the stepping time, the objective function and the system dynamics (control-oriented model) can be rewritten as

$$J = \Phi(\mathbf{x}_N, \mathbf{p}_N) + \sum_{i=0}^{N-1} \text{Ł}(\mathbf{x}_i, \mathbf{u}_i)\Delta\tau. \tag{3.2}$$

subjected to

$$\mathbf{x}_{i+1} = \mathbf{x}_i + \mathbf{f}(\mathbf{x}_i, \mathbf{u}_i, \mathbf{p}_i)\Delta\tau$$
$$\mathbf{g}(\mathbf{x}_i, \mathbf{u}_i, \mathbf{p}_i) = 0.$$

Defining the *Hamiltonian*, H, as

$$H(\mathbf{x}, \mathbf{u}, \lambda, \nu, \mathbf{p}) = \text{Ł}(\mathbf{x}, \mathbf{u}, \mathbf{p}) + \lambda'\mathbf{f}(\mathbf{x}, \mathbf{u}, \mathbf{p}) + \nu'\mathbf{g}(\mathbf{x}, \mathbf{u}, \mathbf{p}) \tag{3.3}$$

where λ and ν denote co-states and Lagrange multipliers, respectively, results in a Two-Point-Boundary-Value-Problem (TPBVP), as the necessary conditions for optimality, shown in (3.4) are applied.

$$\begin{cases} \textit{State eq.}: \mathbf{x}_{i+1} = \mathbf{x}_i + f(\mathbf{x}_i, \mathbf{u}_i, \mathbf{p})\Delta\tau \\ \textit{Costate eq.}: \lambda_i = \lambda_{i+1} + H'_x(\mathbf{x}_i, \mathbf{x}_i, \lambda_{i+1}, \nu_i, \mathbf{p})\Delta\tau \\ H_u(\mathbf{x}_i, \mathbf{u}_i, \lambda_{i+1}, \nu_i, \mathbf{p}) = 0 \\ \mathbf{g}(\mathbf{x}_i, \mathbf{u}_i, \mathbf{p}) = 0 \end{cases} \tag{3.4}$$

The finite-horizon optimal control problem defined in (3.1) does not include any inequality constraints. However, handling inequality constraints is one of the challenging tasks in implementing MPC for nonlinear systems. In practice, an optimal control problem for nonlienar systems usually includes one or more inequality constraints on states or inputs, as follows:

$$\mathbf{h}(\mathbf{x}(t), \mathbf{u}(t), \mathbf{p}(t), t) \leq 0 \tag{3.5}$$

In the problem solved in [28], a *dummy input* or *auxiliary variable* method is employed to handle such constraints. The proposed constraint handling method helps us transform inequality constraints into equality constraint by defining a set of auxiliary variables or dummy inputs [54]. If $\mathbf{h}(\mathbf{x}, \mathbf{u}, \mathbf{p}, t) \in \mathbb{R}^m$, then m new equality constraints can be defined as

$$h_j(\mathbf{x}(t), \mathbf{u}(t), \mathbf{p}(t), t) - \mu_j(t)^2 = 0 \quad \text{for} \quad j = 1, ..., m \tag{3.6}$$

where $[\mu_j] \in \mathbb{R}^m$ is a vector of dummy inputs. In using this method, the trajectory cost function must be modified by adding a small dummy penalty term to it in order to avoid singularity at any $\mu_j = 0$:

$$\tilde{Ł} = Ł(\mathbf{x}(\tau, t), \mathbf{u}(\tau, t), \mathbf{p}(\tau, t)) + r'\mu(\tau, t)) \tag{3.7}$$

where $r \in \mathbb{R}^m$ is the penalty factor with small elements. Other inequality handling methods, as well as auxiliary variable method, have been implemented and compared in [13]. According to Huang et. al. and also the author's simulation tests, the auxiliary variable method, proposed by Ohtsuka, turns out to be very sensitive to the weighting factors that makes the tuning and calibration task very effortful. In a more robust approach towards handling inequality constraints, *Exterior Penalty* method can be used. In this method, a penalty cost is added to the cost function for any violated constraint, meaning that, the cost function is modified into the following:

$$\begin{aligned} J = \Phi(\mathbf{x}(t+T), \mathbf{p}(t+T)) + \int_t^{t+T} & \{Ł(\mathbf{x}(\tau, t), \mathbf{u}(\tau, t), \mathbf{p}(\tau, t)) \\ & + \sum_{j=1}^{m} \psi_j(\mathbf{x}(\tau, t), \mathbf{u}(\tau, t), \mathbf{p}(\tau, t)\} d\tau \end{aligned} \tag{3.8}$$

where

$$\psi_j(\mathbf{x}, \mathbf{u}, \mathbf{p}) = \begin{cases} 0, & h_j(\mathbf{x}, \mathbf{u}, \mathbf{p}) \leq 0 \\ r_j h_j(x, u)^2, & h_j(\mathbf{x}, \mathbf{u}, \mathbf{p}) > 0 \end{cases} \tag{3.9}$$

where r_j is the weighting factor and must be tuned. In this work, we use the Exterior Penalty method for handling inequality constraints.

In order to solve Euler-Lagrange equations or (3.4), let us calculate the states of the system inside the prediction window, recursively, forward in time using *State* equation in (3.4), starting from $x_0(t) = x(t)$ and then similarly, co-states of the system backward in time using *Costate* equation in (3.4), starting from $\lambda_N = \Phi'_x(\mathbf{x}_N, \mathbf{p}_N)$. This will result in $(\mathbf{x}_i)_{i=0}^{N}$ and $(\lambda_i)_{i=0}^{N}$ sequences formed in terms of $(\mathbf{u}_i)_{i=0}^{N-1}$ and $(\nu_i)_{i=0}^{N-1}$ sequences which in fact together form a vector of unknowns as follows

$$\mathbf{U}(t) = [\mathbf{u}'_0(t), \nu'_0(t), \mathbf{u}'_1(t), \nu'_1(t), ..., \mathbf{u}'_{N-1}(t), \nu'_{N-1}(t)] \tag{3.10}$$

The latter vector which includes the sequence of optimal control actions, $(\mathbf{u}_i)_{i=0}^{N-1}$, can be found through solving the rest of the equations of the necessary optimality conditions introduced in (3.4), here regarded as one equation

$$F(\mathbf{U},\mathbf{x},t) = \begin{bmatrix} H_u(\mathbf{x}_0,\mathbf{u}_0,\lambda_1,\nu_0,\mathbf{p}) \\ \mathbf{g}(\mathbf{x}_0,\mathbf{u}_0,\mathbf{p}) \\ \vdots \\ H_u(\mathbf{x}_i,\mathbf{u}_i,\lambda_{i+1},\nu_i,\mathbf{p}) \\ \mathbf{g}(\mathbf{x}_i,\mathbf{u}_i,\mathbf{p}) \\ \vdots \\ H_u(\mathbf{x}_{N-1},\mathbf{u}_{N-1},\lambda_N,\nu_{N-1},\mathbf{p}) \\ \mathbf{g}(\mathbf{x}_{N-1},\mathbf{u}_{N-1},\mathbf{p}) \end{bmatrix} = 0 \tag{3.11}$$

Solution to the last equation is the key to the optimal answer of the problem at each sampling time. Performance of the NMPC is significantly depended on the accuracy and speed of the process of solving this nonlinear equation. Newton's method might be used to solve this equation [55]:

$$F_{\mathbf{U}}(\mathbf{U}^k(t),\mathbf{x}^k(t),t)\delta\mathbf{U}(t) = -F(\mathbf{U}^k(t),\mathbf{x}^k(t),t) \tag{3.12}$$

$$\mathbf{U}^{k+1} = \mathbf{U}^k(t) + \delta\mathbf{U}(t) \tag{3.13}$$

However, as the problem gets larger, calculation of the Jacobian becomes computationally more expensive. Moreover, the linear algebraic equation (3.12) requires a fast linear solver. Newton-iterative methods are potentially fast promising solvers that can be used to solve nonlinear equations in real-time. The question is how much the progress in Newton's iteration is affected when the exact solution to the linear equation for the Newton's step is replaced with an approximate solution. The answer is that an approximate solution is accepted as long as the relative residual is small. This useful statement helps us employ iterative methods such as GMRES to solve the linear equation. It is common to refer to the iterations on solving the linear equation for Newton's step as *inner iterations* and to the Newton's iterations as *outer iterations* [55].

3.2 Newton/GMRES Method for Solving Nonlinear Equation F(U,x,t)

As stated in the previous section, for each Newton's iteration we must solve linear equation (3.12). If the linear iterative method used to solve (3.12) is the GMRES method, then each inner iteration requires at least one evaluation of the product of the Jacobian $F_{\mathbf{U}}$ and a vector. To avoid such expensive computation, the action of $F_{\mathbf{U}}$ on a vector is approximated by a forward difference approximation as below:

$$F_{\mathbf{x}}(\mathbf{x})\mathbf{w} \approx D_h F(\mathbf{x} : \mathbf{w}) = \frac{F(\mathbf{x} + h\mathbf{w}) - F(\mathbf{x})}{h} \tag{3.14}$$

Where $\mathbf{x} \in \mathbb{R}^n$ and $\mathbf{w} \in \mathbb{R}^n$ are vectors of length n and h is some small positive value. In fact, (3.14) approximates the product of the Jacobian $F_{\mathbf{x}}$ and the vector $\mathbf{w}$. Such an approach is called Forward-Difference GMRES algorithm or *fdgmres*. We use *fdgmres* algorithm from [55], presented below, to solve for Newton's step. Note that for the initial iterate zero vector is used as the initial guess.

```
Result: $\delta\mathbf{U} = \texttt{fdgmres}(\delta\mathbf{U}, \mathbf{U}, \mathbf{x}, \mathbf{p}, F, k_{max}, \eta, \rho)$
$\delta\mathbf{U} = 0, r = -F(\mathbf{U}, \mathbf{x}, t), v_1 = r/\|r\|, \beta = \rho, k = 0;$
while $\rho > \eta\|F(\mathbf{U}, \mathbf{x}, t)\|$ and $k < k_{max}$ do
    $k = k + 1;$
    $v_k + 1 = D_h F(\mathbf{U}, \mathbf{x}, t : v_k, 0, 0);$
    for $j$=1,...$k$ do
        $h_{j,k} = v_{k+1}^T v_j;$
        $v_{k+1} = v_{k+1} - h_{j,k} v_j;$
    end
    $h_{k+1}, k = \|v_{k+1}\|;$
    $v_{k+1} = v_{k+1}/\|v_{k+1}\|;$
    $e_1 = (1, 0, ..., 0)^T \in \mathbb{R}^{k+1};$
    $H_k = [hij] \in \mathbb{R}^{(k+1)\times k}$ (if $i > j + 1$ then $h_i j = 0$);
    Minimize $\|\beta e_1 - H_k y^k\|$ to find $y^k \in \mathbb{R}^k$;
    $\rho = \|\beta e_1 - H_k y^k\|;$
    $V_k = [v_i] \in \mathbb{R}^{mN\times k};$
end
$\delta\mathbf{U} = V_k y^k;$
```

Algorithm 1: `fdgmres` algorithm for finding Newton's step $\delta\mathbf{U}$

3.3 Continuation/GMRES Method for Solving Nonlinear Equation F(U,x,t)

Another approach to solving a set of nonlinear equations is proposed by Ohtsuka in [28] by combining Continuation method [56] and forward difference GMRES method [55]. According to [28], in Continuation method we trace the solution curve of a nonlinear equation by integrating a differential equation in continuous time. Therefore, instead of solving (3.11), $\dot{\mathbf{U}}$ is determined so that the following differential equation is valid:

$$\dot{F}(\mathbf{U},\mathbf{x},t) = A_s F(\mathbf{U},\mathbf{x},t) \tag{3.15}$$

where A_s is a stable matrix. Differentiating the left side of (3.15) and rearranging, results in the following linear equation in terms of $\dot{\mathbf{U}}$:

$$F_{\mathbf{U}}(\mathbf{U},\mathbf{x},t)\dot{\mathbf{U}} = A_s F(\mathbf{U},\mathbf{x},t) - F_x(\mathbf{U},\mathbf{x},t)\dot{\mathbf{x}} - \dot{F}(\mathbf{U},\mathbf{x},t) \tag{3.16}$$

In order to solve the recent equation, one can employ the forward-difference approximation in (3.14) to eliminate the calculation of the Jacobians and replace (3.16) with (3.17):

$$D_h F(\mathbf{U},\mathbf{x}+h\dot{\mathbf{x}},t+h : \dot{\mathbf{U}},0,0) = A_s F(\mathbf{U},\mathbf{x},t) - D_h F(\mathbf{U},\mathbf{x},t : 0,\dot{\mathbf{x}},1) = b(\mathbf{U},\mathbf{x},\dot{\mathbf{x}},t) \tag{3.17}$$

Now, an *fdgmres* algorithm can be used to solve (3.17) for $\dot{\mathbf{U}}$. Also, by generalizing *fdgmres* algorithm, Ohtsuka suggests that the solution for $\dot{\mathbf{U}}$ in previous sampling time can be used as the initial guess for current timestep's first inner iteration [28]. The modified `FDGMRES` algorithm is presented below.

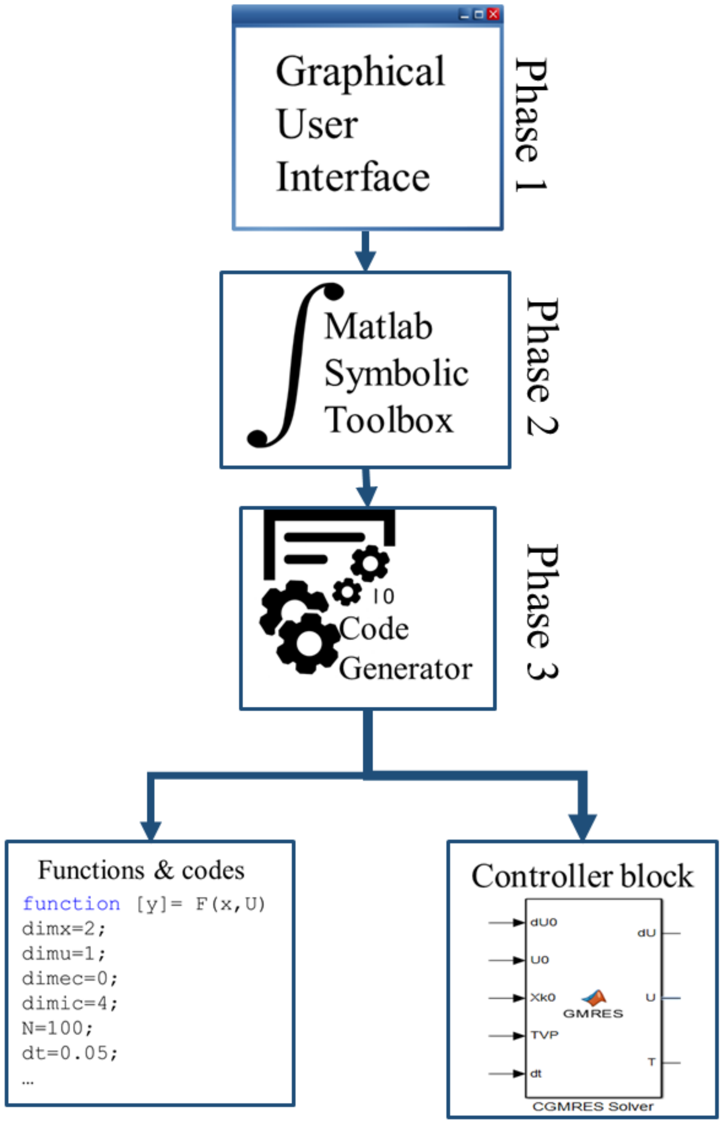
```
Result: $\dot{\mathbf{U}} = \texttt{FDGMRES}(\mathbf{U}, \dot{\mathbf{U}}, \mathbf{x}, \dot{x}, F, kmax, \eta, \rho)$
$r = b(\mathbf{U}, \mathbf{x}, \dot{\mathbf{x}}, t) - D_h F(\mathbf{U}, \mathbf{x} + h\dot{\mathbf{x}}, t + h : \dot{\mathbf{U}}, 0, 0), v_1 = r/\|r\|, \beta = \rho, k = 0;$
while $\rho > \eta\|F(\mathbf{U}, \mathbf{x}, t)\|$ and $k < kmax$ do
    $k = k + 1;$
    $v_k + 1 = D_h F(\mathbf{U}, \mathbf{x} + h\dot{\mathbf{x}}, t + h : v_k, 0, 0);$
    for $j$=1,...$k$ do
        $h_{j,k} = v_{k+1}^T v_j;$
        $v_{k+1} = v_{k+1} - h_{j,k} v_j;$
    end
    $h_{k+1}, k = \|v_{k+1}\|;$
    $v_{k+1} = v_{k+1}/\|v_{k+1}\|;$
    $e_1 = (1, 0, ..., 0)^T \in \mathbb{R}^{k+1};$
    $H_k = [hij] \in \mathbb{R}^{(k+1)\times k}$ (if $i > j + 1$ then $h_i j = 0$);
    Minimize $\|\beta e_1 - H_k y^k\|$ to find $y^k \in \mathbb{R}^k;$
    $\rho = \|\beta e_1 - H_k y^k\|;$
    $V_k = [v_i] \in \mathbb{R}^{mN\times k};$
end
$\dot{\mathbf{U}} = \dot{\mathbf{U}} + V_k y^k;$
```

Algorithm 2: FDGMRES algorithm for finding $\dot{\mathbf{U}}$ in C/GMRES method

As it can be induced from the FDGMRES algorithm, in C/GMRES method, the information about $\dot{\mathbf{x}}$ must be provided to the solver in addition to a state feedback. To this end, one can approximate $\dot{\mathbf{x}}$ at each sampling time as $\dot{\mathbf{x}}(t) \approx (\mathbf{x}(t + \Delta t) - \mathbf{x}(t))/\Delta t$.

3.4 Automatic Multi-Solver NMPC Code Generator

Basically, AuMuSoN consists of two separate set of calculations: offline and online; meaning that the mathematical process of solving an optimal control problem, explained in section 3.1, is categorized into two parts. The first part including forming the Hamiltonian and calculation of the derivatives are performed, symbolically offline, with the help of MATLAB symbolic toolbox and the resulted functions and expressions are outputted as MATLAB standard function codes. Then, these functions are repeatedly called during the online calculations inside the controller's function. This procedure helps significantly reduce the online computational load by performing most of the calculations offline and leaving nothing but a bunch of numerical evaluations for online computations.



```
function [y]= F(x,U)
dimx=2;
dimu=1;
dimec=0;
dimic=4;
N=100;
dt=0.05;
…
```

Figure 3.1: Principle of the Automatic Multi-solver NMPC Code Generator (AuMuSoN)

AuMuSoN works in three consecutive phases as shown in Fig. 3.1. In the first phase, a Graphical User Interface (GUI), illustrated in Fig. 3.2, receives the optimal control problem definition from the user, interactively. In the different predefined fields of the GUI, the user can enter the expressions for the control-oriented model, objective function, equality or inequality constraints and terminal cost. Moreover, the user can define his/her own User-Defined Parameters (UDP) to be used inside the expressions. Additionally, depending on the control problem, the control-oriented model or the objective function might have one or more Time-Varying Parameters (TVP) which the user can also define them in the GUI and later provide the relevant signals in the simulations. For example, in order to solve an optimal tracking problem, one can define a reference signal as a TVP in the problem definition section and then provide the reference signal in the SIMULINK file.

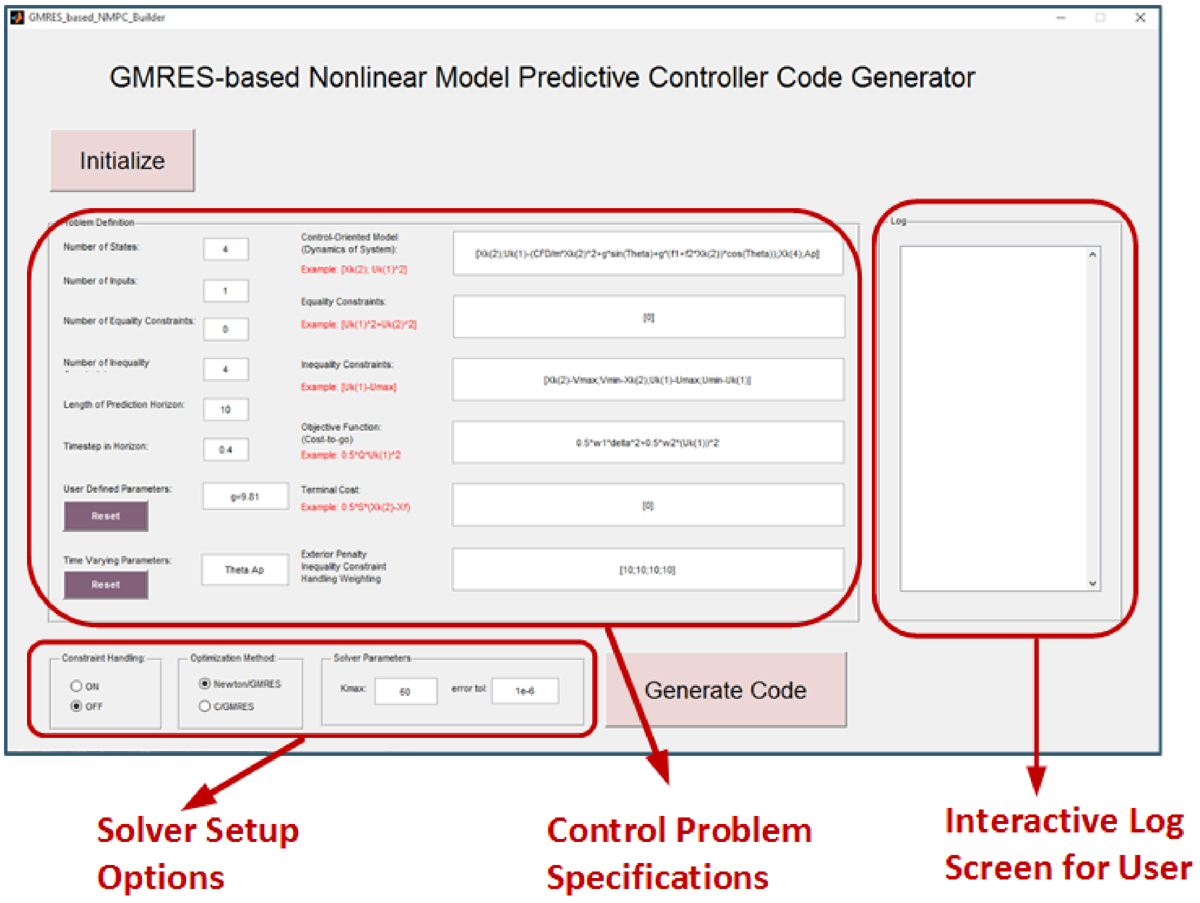
The second phase of the code generation includes the symbolic mathematical operations on the expressions, parameters and functions that the user has defined using the GUI. In this phase, the Hamiltonian is formed and partial derivatives of it with respect to input and state vectors are derived. Lastly, in the third phase, the generated symbolic expressions and equations in the previous symbolic phase are rearranged and written in several different files as MATLAB functions. All of these MATLAB functions contribute to forming the nonlinear equation F(U,x,t) in real-time. Also, based on the solver options that the user selects in the GUI, an NMPC main code is generated in a SIMULINK block that calls all of the other generated functions as required during the simulations.

For simulation purposes, AuMuSoN also generates a SIMULINK file that the user can employ for MIL simulations with small modifications. As shown in Fig. 3.3, the generated SIMULINK block diagram includes a block of the generated NMPC controller and a block of the controlled system. By default, the program uses the defined control-oriented model as a model of the controlled plant, though the user can replace this block with his/her own high-fidelity model. It is important to note that if the problem includes some time-varying parameters, the user must provide a block that generates the values for the parameters as a function of the time in the same order as defined previously in the GUI.

For handling inequality constraints, as described in Section 3.1, Exterior Penalty method has been implemented in AuMuSoN. The ease of implementation and calibration as well as robustness has encouraged the author to employ this method of constraint handling.

3.4.1 Graphical User Interface (GUI)

The GUI consists of three main input sections and two command buttons, see Fig. 3.2. The "*Initialize*" button resets the program and clears the values for the variables. The


GMRES_based_NMPC_Builder

GMRES-based Nonlinear Model Predictive Controller Code Generator

Problem Definition

Number of States:	4	Control-Oriented Model (Dynamics of System): Example: [Xk(2); Uk(1)^2]	[Xk(2);Uk(1)-(CFD/m*Xk(2)^2+g*sin(Theta)+g*(f1+f2*Xk(2))*cos(Theta));Xk(4);Ap]
Number of Inputs:	1	Equality Constraints: Example: [Uk(1)^2+Uk(2)^2]	[0]
Number of Equality Constraints:	0		
Number of Inequality	4	Inequality Constraints: Example: [Uk(1)-Umax]	[Xk(2)-Vmax;Vmin-Xk(2);Uk(1)-Umax;Umin-Uk(1)]
Length of Prediction Horizon:	10		
Timestep in Horizon:	0.4	Objective Function: (Cost-to-go) Example: 0.5*Q*Uk(1)^2	0.5*w1*delta^2+0.5*w2*(Uk(1))^2
User Defined Parameters:	g=9.81	Terminal Cost: Example: 0.5*S*(Xk(2)-Xf)	[0]
Time Varying Parameters:	Theta Ap	Exterior Penalty Inequality Constraint Handling Weighting	[10;10;10;10]

Constraint Handling: ON [✓ OFF]

Optimization Method: [✓ Newton/GMRES] C/GMRES

Solver Parameters: Kmax: 60 error tol: 1e-6

Log

Initialize
Reset
Reset
Generate Code
Solver Setup Options
Control Problem Specifications
Interactive Log Screen for User

Figure 3.2: Graphical User Interface (GUI) of AuMuSoN program

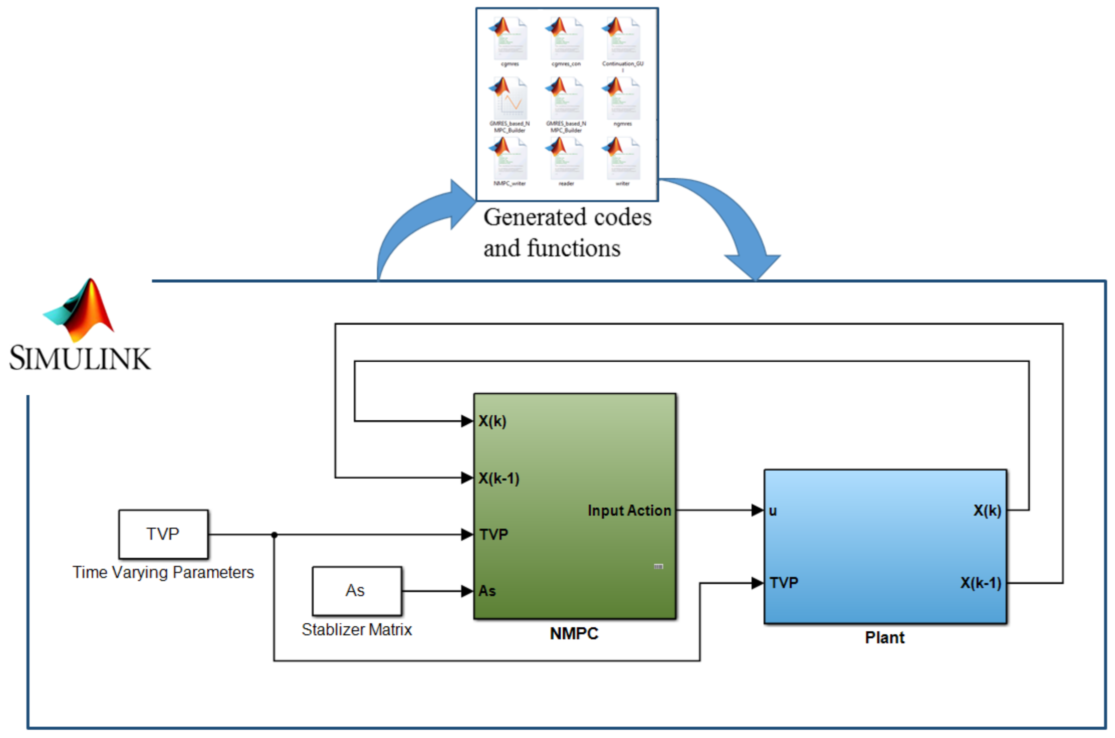


Figure 3.3: NMPC simulation model in Simulink and its interaction with the codes and functions generated by AuMuSoN

"*Problem Definition*" section contains input fields necessary to define an optimal control problem. A list of definable problem parameters and expressions in this section is presented in Table 3.1.

In the "*User Defined Parameters*" (UDP) field, several parameters can be defined in order to promote the process of problem definition and avoid entering the same values repeatedly. To do so, one must type in the parameters and their values sequentially and press "Enter" after each parameter. For example, one must type in "`g=9.81`", press "Enter", clear the box, type in "`R=8.315`" and then press "Enter" again in order to define these two parameters. Later, the user can use these parameters inside the rest of the input fields with the same names. A "Reset" button is placed next to this field for clearing all of the UDPs. Another input field is designated to the definition of Time-Varying Parameters (TVPs). To do so, the user simply just has to type in the name of the TVPs separated by a Space and then press "Enter". Similar to the UDP part, a "Reset" button clears all of the defined TVPs.

The rest of the input fields of the "Problem Definition" section are specified to the fixation of main functions and equations related to the corresponding problem. The first input field must be used to define the control-oriented model. Here, control-oriented model must be in the form of a MATLAB vector representing the right side of the vector equation $\dot{x} = f(x(t), u(t), p(t), t)$. It is important to note that AuMuSoN notation requires the user to type "`Xk(i)`" and "`Uk(i)`" for referring to the i^th element of the states and inputs vectors, respectively. Also, as AuMuSoN is a MATLAB-based program, the mathematical notation of the defined expressions must be compatible to this software syntax. Equality constraints and inequality constraints must be defined as MATLAB vectors representing the left side of the equations $g(x(t), u(t), p(t), t) = 0$ and $C(x(t), u(t), p(t), t) \leq 0$ in the same manner, as well, in the following fields. The *Trajectory* and *Terminal* costs can be defined in the next two input fields. According to the application, the problem may exclude one or more of the predefined fields. In this case, the user may input "`[0]`" for the unnecessary fields. Lastly, if the problem contains one or more inequality constraints, the exterior penalty method's weighting factors must be defined in the last input field as a vector.

The second main section of the AuMuSoN GUI is the "*Solver Settings*" section which consists of two sets of radio buttons and two input fields. In the "*Constraint Handling*" part, the user can switch ON or OFF this feature, regardless of the problem definition. Secondly, in the "*Optimization Method*" part, the user can switch between the two available GMRES-based real-time optimizers: Newton/GMRES and C/GMRES. Last but not least, in the "*Solver Parameters*" part, the two main parameters necessary for `fdgmres` solver, i.e. "k_max" and "η", can be set.

Table 3.1: Definable design and problem parameters in AuMuSoN

Design Parameter	Field Caption	Input Example
dimension of x(t)	Number of States	`2`
dimension of u(t)	Number of Inputs	`1`
dimension of $g(x, u, p)$	Number of Equality Constraints	`0`
dimension of $C(x, u, p)$	Number of Inequality Constraints	`4`
N	Prediction Horizon Length	`10`
$\Delta\tau$	Discretization Timestep	`0.1`
UDP	User-Defined Parameters	`g=9.81`
$p(t)$	Time-Varying Parameters	`Theta Grade Acc`
$f(x(t), u(t))$	Control-oriented Model (Dynamics of System)	`[Xk(2);Uk(1)-K*Xk(2)^2]`
$g(x, u, p)$	Equality Constraints	`[Uk(1)^2+Uk(2)^2-10]` or `[0]`
$C(x, u, p)$	Inequality Constraints	`[Uk(1)-Umax;Xmin-Xk(1)]` or `[0]`
Ł(x, u, p)	Trajectory Cost	`S*(Xk(1)-Xf)^2+R*Uk(1)^2`
$\Phi(x, u, p)$	Terminal Cost	`P*(Xk(1)-20)^2`
$[\beta_j]$	Exterior Penalty Weighting Factors	`[10;3;15;0.1]`

In addition to the two previous sections, the GUI has a Log screen that interactively displays to the user, the parameters and expressions that already have been defined. Finally, after the problem definition is complete and the solver settings are done, the "*Generate*" button automatically generates a set of MATLAB .m files and a SIMULINK .mdl files explained in the next section.

3.4.2 The Automatically Generated Files

The generated files by AuMuSoN include five .m files that are basically problem-dependent MATLAB functions that together with three other fixed (problem-independent) functions form a set of necessary functions to run the SIMULINK simulation file. The SIMULINK .mdl file included in the package generally gives the idea of how the NMPC system works. It has two main blocks: the NMPC controller and the plant. By default, the plant block contains the control-oriented model of the system provided by the user in the GUI. This block can be replaced with a high-fidelity model of the system in MIL simulations. The NMPC controller block calls the 8 MATLAB files (5 generated + 3 fixed), explained later, during the simulation.

Table 3.2 lists the MATLAB .m files used during the simulations. Among the problem-independent functions, `fdgmres.m` is the main forward-difference GMRES solver function receiving state feedback and TVPs at each sampling time and outputting Newton's step. `FDGMRES.m` works in the same manner, generating $\dot{\mathbf{U}}$, if the user is using C/GMRES algorithm for his/her NMPC controller. These two main functions use their own specific fixed functions for forward-difference approximation: `dhf.m` for `fdgmres.m` and `DHF.m` for `FDGMRES.m`. Also, the minimization of $\|\beta e_1 - H_k y^k\|$ in both of the algorithms are performed using k Givens Rotations [55] in `kGivens.m` function.

In addition to the fixed functions, 5 automatically generated functions contribute to the process of the NMPC controller, among which, the most important one is the `FxU.m`. `FxU.m` is the main function creating the basic nonlinear equation $F(U, x, t)$ in (3.11) and evaluating it at each sampling time using the rest of the problem-dependent functions. Two other generated files are `dHdu.m` and `dHdx.m`, which correspond to the partial derivatives of the Hamiltonian with respect to $\mathbf{u}(t)$ and $\mathbf{x}(t)$, respectively. The `CxU.m` file checks for inequality constraint violations in each sampling time and returns the values violated from the constraint boundaries. Lastly, the `f.m` file contains the dynamics of the system and contributes to the prediction of the future states.

Table 3.2: Definable design and problem parameters in AuMuSoN

File name	Type	Description
`fdgmres.m`	Fixed	runs *fdgmres* algorithm for a given equation in Newton/GMRES method
`FDGMRES.m`	Fixed	runs *FDGMRES* algorithm for a given equation in C/GMRES method
`dhf.m`	Fixed	applies forward-difference approximation for a given function and vector in Newton/GMRES method
`DHF.m`	Fixed	applies forward-difference approximation for a given function and vector in C/GMRES method
`kGivens.m`	Fixed	applies k sequential Givens rotations for minimization
`FxU.m`	Generated	Contains the function $F(U, x, t)$ and evaluates it at each iteration
`dHdu.m`	Generated	Contains the function for $H_u(x, u, t)$ and evaluates it at each iteration
`dHdx.m`	Generated	Contains the function for $H_x(x, u, t)$ and evaluates it at each iteration
`CxU.m`	Generated	Contains the function for $C(x, u, t)$ and evaluates it at each iteration
`f.m`	Generated	Contains the function for $f(x, u)$ and evaluates it at each iteration

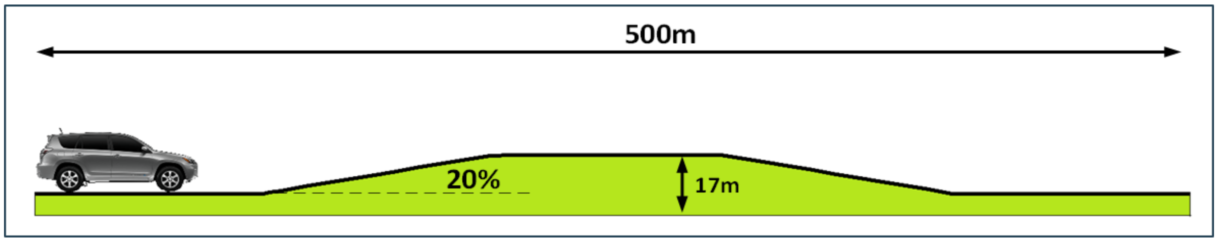


Figure 3.4: Schematic illustration of the code validation problem

3.5 Code Generation Evaluation

Recently, a software package is developed and published by Cybernet that provides a symbolic computation tool in Maple for generating NMPC controller code based on C/GMRES algorithm. The so called "AutoGenU for Maple" code generating system requires Maple software and a C compiler to automatically generate a C code, run the simulation and show the results in graphs. This package works based on Maple and because the generated controller code is in C, it requires a connector to generate a SIMULINK block for the controller for use in MATLAB simulations. On the other hand, since the generated C code is actually a code for running simulations, it is challenging for the user to modify the code and extract the sole NMPC controller code from the whole simulation code. For example, if a user tends to use a high-fidelity model of the plant in SIMULINK instead of the control-oriented model for control evaluations and then perform HIL tests, he/she must face two difficulties: one is extracting the controller code from the simulation code and adding the high-fidelity model to the system, the other is importing the generated C code into SIMULINK and then again generating a C code of the whole SIMULINK system including the high-fidelity model for uploading onto the designated HIL setup.

Consequently, the developed automatic code generator, AuMuSoN, is all based on MATLAB and every single phase of the code generation from defining the optimal control problem to running the simulation are performed within MATLAB software. It is important to note that in this program, replacing the control-oriented model with a high-fidelity model of the system is as easy as deleting a block in SIMULINK file and inserting another one. Also, for HIL testing, the user can use the MATLAB built-in C code generation tool to generate an embeddable C code from any part of the SIMULINK file. Consequently for a SIMULINK user, an empirical task involving numerous trials and modification in the control design, such as calibration and tuning of the designed controller, can be done in a shorter time and a more efficient way with the developed MATLAB program.

In order to check the validity of the developed code generator, an optimal control problem is solved using AuMuSoN and AutoGenU and the results are compared, accordingly. The control problem is defined as the optimal cruise control of a vehicle travelling over a hill. This cruise controller must take into account the road elevation ahead and then minimize the following objective function in order to minimize the control input as well as the error in the velocity. Equations (3.18) and (3.19) represent the definition of the optimal control problem:

$$\begin{aligned} J = {} & \frac{1}{2}\omega_1(v(T) - v_{ref}(T))^2 \\ & + \int_0^T \{\frac{1}{2}\omega_2(v(\tau) - v_{ref}(\tau))^2 + \frac{1}{2}\omega_3 u(\tau)^2\} d\tau. \end{aligned} \tag{3.18}$$

Subjected to

$$\begin{gathered} \begin{bmatrix} \dot{x}(t) \\ \dot{v}(t) \end{bmatrix} = \begin{bmatrix} v(t) \\ u - \frac{1}{m}F_r \end{bmatrix} \\ F_r = \frac{1}{2}\rho A_f C_d v^2(t) + mg\sin\theta(x(t)) + \mu mg\cos\theta(x(t)). \end{gathered} \tag{3.19}$$

Where θ is the road grade which is a TVP in this problem and is assumed to be fully known inside the prediction horizon. ρ is the air density, A_f is the vehicle's frontal area, C_d is the drag coefficient and μ represents rolling resistance coefficient. The input to the control-oriented model is obtained by dividing the demanded wheel torque by $(m.r)$ where m and r are the vehicle's mass and wheel radius, respectively. In the objective function defined in (3.18), ω_is are the different weighting factors related to different cost terms. v_{ref}, here assumed constant 15m/s, refers to the reference speed set by the driver.

The performance of the generated NMPC controller codes is tested on simple control-oriented models of the vehicle as the main purpose of these simulations is to evaluate the accuracy of the developed solver code. As illustrated in Fig. 3.5, the results of the simulations for both AutoGenU and AuMuSoN code generators are identical. Results show less than 0.01% difference between the responses of the two generated NMPC controllers.

Please note that in this example, the optimization problem is not subjected to any inequality constraint.

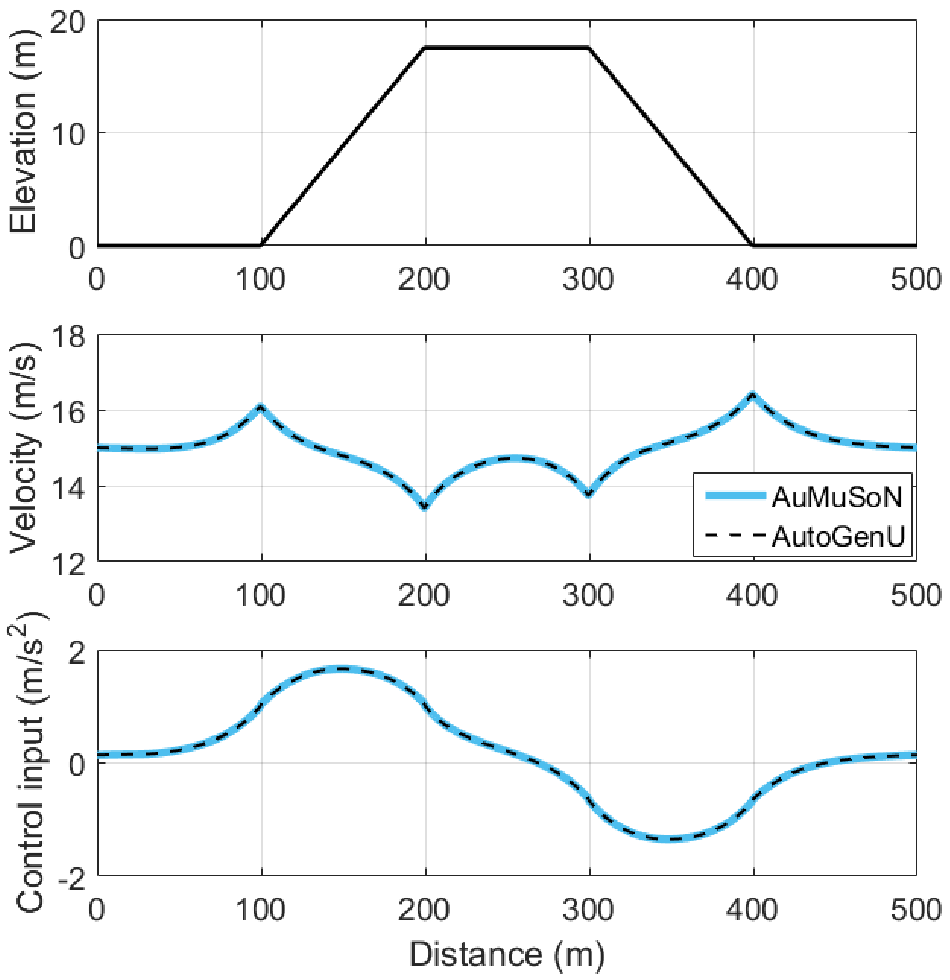


Figure 3.5: Simulation results for validation of AuMuSoN NMPC controller against AutoGenU controller

Chapter 4

Controller Design and Evaluation

The previously developed code generation tool enables us to solve several finite-horizon optimal control problems and implement them as MPC controllers for nonlinear systems. With that in mind, various optimization problems can be defined for improvement of automotive systems. In this study, we are interested in application of NMPC controllers in ecological cruise control of PHEVs.

4.1 High-fidelity Model

In this research, a high-fidelity model of Toyota Prius 2013 PHEV is used for both MIL and HIL simulations. The principles of the powertrain of this PHEV is illustrated schematically in Fig. 4.1. In the power-split hybrid structure of this PHEV, MG-1 and MG-2 are the electric motors that can operate as generators as well. These two electrical components are connected with an engine through two sets of planetary gears whose ring gears are coupled to the final drive.

The high-fidelity model used in this research is developed in Autonomie software by Argonne National Lab. The validity of this model is verified through various simulations in [57]. The main components of the high-fidelity model include models of the engine, MG-1, MG-2 and the battery pack. Figure 4.2 displays the Autonomie model in Simulink environment.

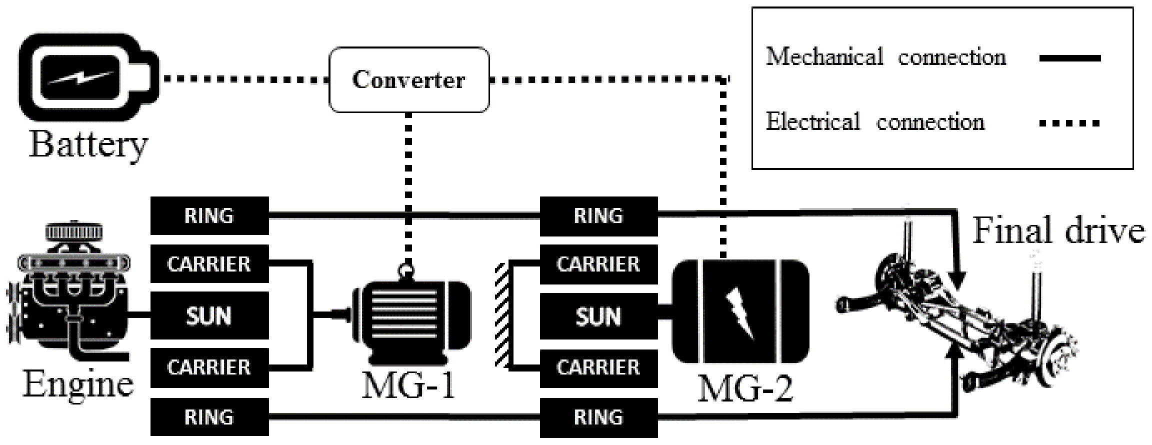


Figure 4.1: Schematic representation of the Toyota Prius 2013 powertrain structure

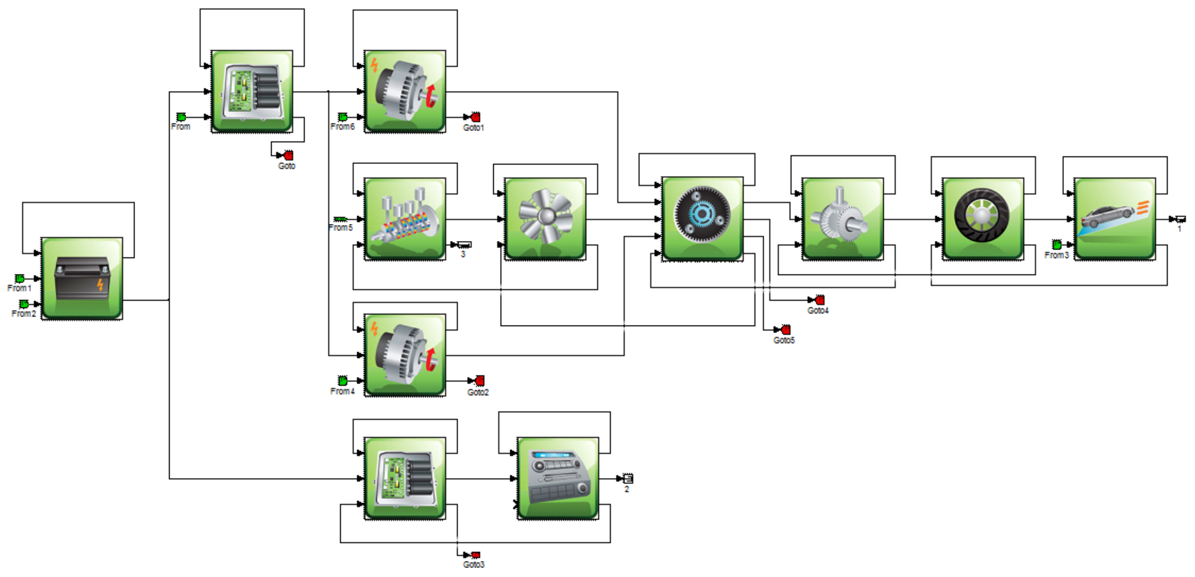


Figure 4.2: Autonomie high-fidelity model of the PHEV in Simulink

4.2 Case 1: Driving on Hilly Roads

A cruise controller can be modified and made more ecological considering many different driving conditions and aspects. One of these driving conditions that makes a potentially resolvable optimal control problem is driving on roads with up/down slopes where cruising over a hill with a constant speed is not the most ecological approach. In this case, one may save energy and fuel by providing the road grade information to the controller and benefiting from the gravitational force for accelerating. This idea is implemented with a predictive approach based on C/GMRES solver in [45] for an ICE vehicle.

Amongst different types of the vehicles, PHEVs as promising options for future transportation, make good candidates for eco-driving. Their complex drivetrain structure with two sources of power, creates a multi-objective optimization problem that bolds the effect of the use of intelligent control. In the rest of this chapter, first an ECC is designed for a PHEV, namely Toyota Prius 2013. Then, the familiar cruising over a hill problem described in Section 3.5 is solved with the purpose of presenting a comparison between different RTO methods, particularly C/GMRES and Netwon/GMRES. Finally, the designed ECC is tested on an actual road profile in order to investigate the potential of Dynamic Prediction Horizon Length (PDHL) on the improvement of the NMPC controller performance.

The proposed control system is intended to regulate the cruising speed of a PHEV on hilly roads such that it minimizes the energy cost. We make use of receding horizon predictive control to collect the road elevation profile ahead of the vehicle and generate an optimal velocity trajectory. Here, we are dealing with a trade-off between having a lower fuel consumption and tracking the demanded cruising velocity by the driver which defines an optimal control problem. A control-oriented model is developed and used at the heart of the controller that collects future data and current states to generate the future states. The GMRES-based optimizers have been utilized to solve the nonlinear root finding problem in real-time. Also, Exterior Penalty method is employed for handling inequality constraints. The codes for these simulations are generated by AuMuSoN.

4.2.1 Controller Design

Figure 4.3 shows the schematic demonstration of a vehicle approaching a hill. Considering the vehicle longitudinal dynamics, one can obtain a simple yet effective mathematical control-oriented model of the form as given in (3.19) repeated below:

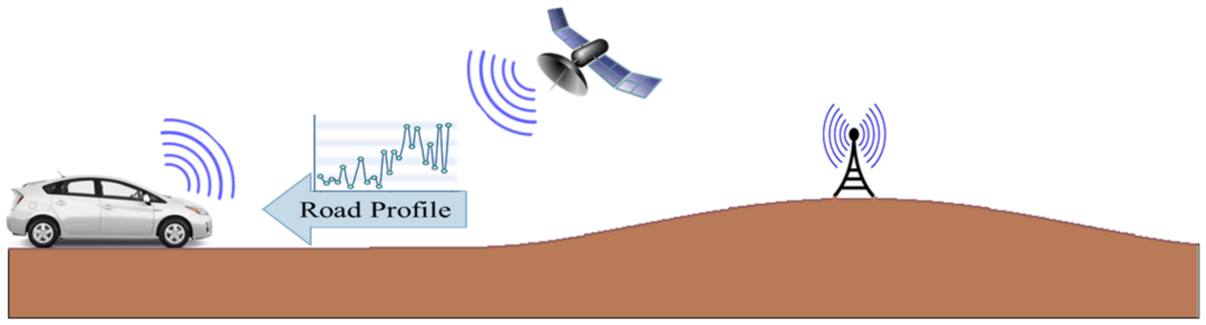


Figure 4.3: Schematic of a vehicle equipped with ECC cruising over a hill- Road profile information is provided to the vehicle through V2I

$$\begin{aligned} \begin{bmatrix} \dot{x}(t) \\ \dot{v}(t) \end{bmatrix} &= \begin{bmatrix} v(t) \\ u - \frac{1}{m}F_r \end{bmatrix} \\ F_r &= \frac{1}{2}\rho A_f C_d v^2(t) + mg\sin\theta(x(t)) + \mu mg\cos\theta(x(t)). \end{aligned} \tag{4.1}$$

Vehicle's position and velocity are the states of the system and $\theta(x(t))$, the road grade, is assumed to be fully known as a function of vehicle's position by means of road elevation maps and GPS. ρ is the air density, A_f is the vehicle's frontal area, C_d is the drag coefficient and μ represents rolling resistance coefficient. The input to the control-oriented model is obtained by dividing the demanded wheel torque by $(m.r)$ where m and r are the vehicle's mass and wheel radius, respectively. This control-oriented model is used at the heart of the NMPC in order to minimize the objective function in (3.18) repeated below:

$$\begin{aligned} J = &\frac{1}{2}\omega_1(v - v_{ref})^2 \\ &+ \int_0^T \{\frac{1}{2}\omega_2(v(\tau) - v_{ref}(\tau))^2 + \frac{1}{2}\omega_3 u(\tau)^2\}d\tau. \end{aligned} \tag{4.2}$$

Subjected to

$$\begin{cases} u_{min} \leq u(\tau) \leq u_{max} \\ v_{min} \leq v(\tau) \leq v_{max} \end{cases} \tag{4.3}$$

Where v_{ref} is the reference speed set by the driver on cruise control, which in this work $v_{ref} = 54km/h$ for all time. ω_1,ω_2 and ω_3 are the weighting factors. The trajectory cost function consists of two terms. The first term corresponds to the tracking of the reference velocity set by the driver and the second term contributes to minimizing the demanded torque and hence reducing the energy cost. The inequality constraints shown in (4.3) are applied to the problem by adding the following penalty functions to the performance index for the j^{th} inequality function:

$$\psi_j(x,u) = \begin{cases} 0, & h_j(x,u) \leq 0 \\ \beta_j C_j(x,u)^2, & h_j(x,u) > 0 \end{cases} \tag{4.4}$$

4.2.2 Controller Evaluation with MIL Simulations

Figures 4.4 to 4.6 represent the performance of the control strategy on a road section with a hill for two NMPC controllers with two different tunings and a PID controller. Figure 4.4 illustrates the velocity profile of the vehicle during cruising over a hill. For a better demonstration of the ecological aspect of the proposed NMPC (with Newton/GMRES), its performance is evaluated against a classic PID controller. One essential tuning parameter in the designed NMPC is the (ω_3/ω_2) ratio denoted as α hereafter. It is observed in Fig. 4.4 that the predictive strategy attempts to minimize the propelling effort by increasing the speed prior to reaching the uphill, letting the speed drop during uphill, and then regulating the speed back to the reference value by taking advantage of the gravitational force during downhill. Therefore, NMPC strategy decreases energy consumption, as shown in Fig. 4.6, by smoothing the demanded torque, as shown in Fig. 4.5, along the whole track and avoiding abrupt torque demands.

From the viewpoint of tuning, α, in fact determines the superiority of energy-saving feature over reference-speed- tracking feature which can be made adaptive in a separate study. As it is shown in the Fig. 4.7, increasing α results in greater deviation from the reference speed while decreasing the energy consumption. This leaves us with a trade-off. Figure 4.8 demonstrates the effect of increasing α on the vehicle speed trajectory and energy consumption. In this paper, we set $\alpha = 0.1$ hereafter as it gives the best energy-saving performance within 2km/h deviation from the reference speed.

As one of the main objectives of the current work, we focus on the comparison of different real-time optimization methods employed inside the NMPC, mainly C/GMRES and Newton/GMRES. To this end, the NMPC problem is solved with Newton/GMRES, C/GMRES, Fsolve and Interior-Point (IP) methods and the simulation results are compared in Figures 4.9 to 4.11. Here, Fsolve refers to a method where MATLAB's `fsolve` command is used to solve the root finding problem in (3.11). It should be also noted that

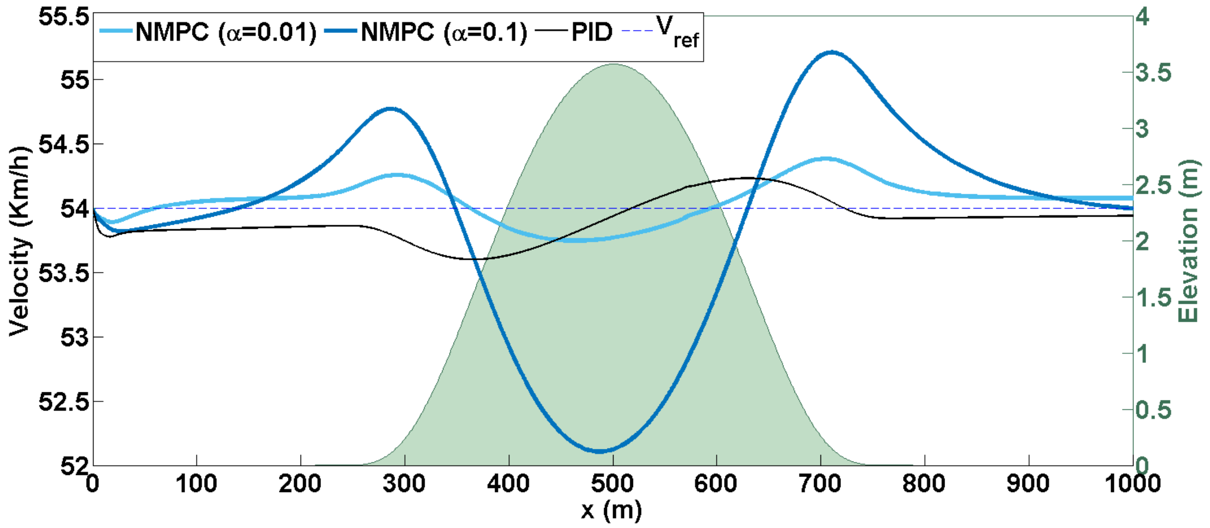


Figure 4.4: Velocity profile of the vehicles cruising over a hill by PID and NMPC controllers

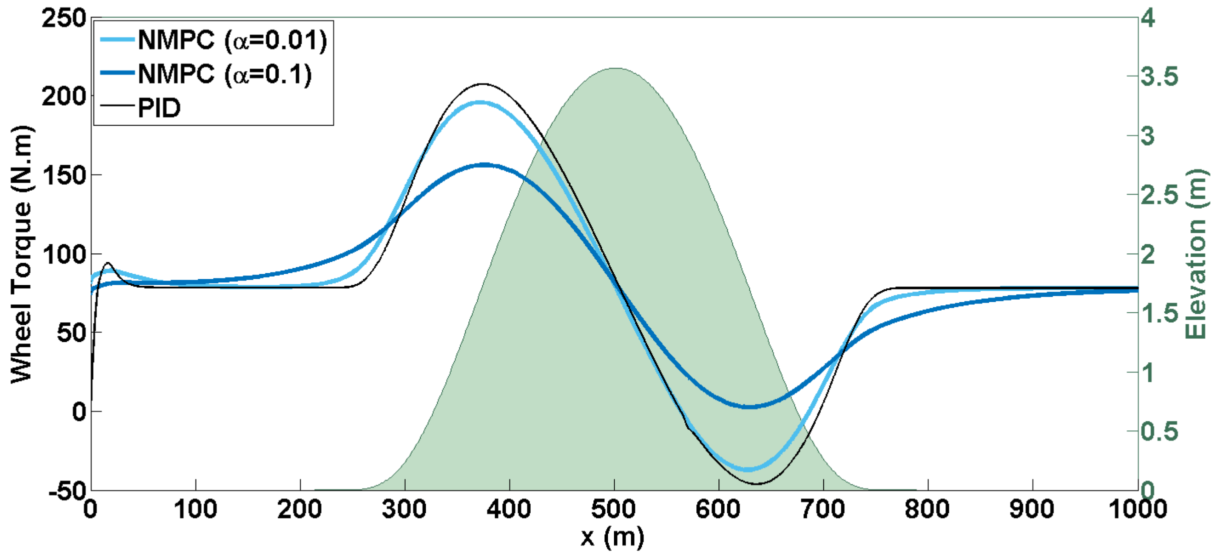


Figure 4.5: Torque demand of the vehicles cruising over a hill by PID and NMPC controllers

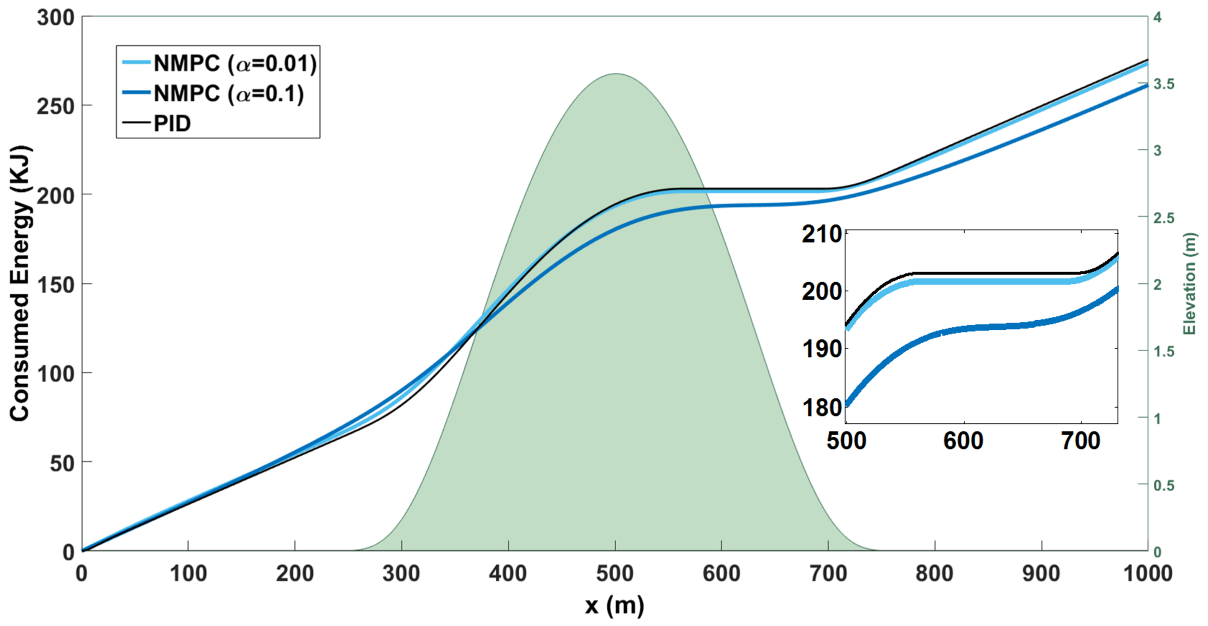


Figure 4.6: Energy consumption of the vehicles cruising over a hill by PID and NMPC controllers

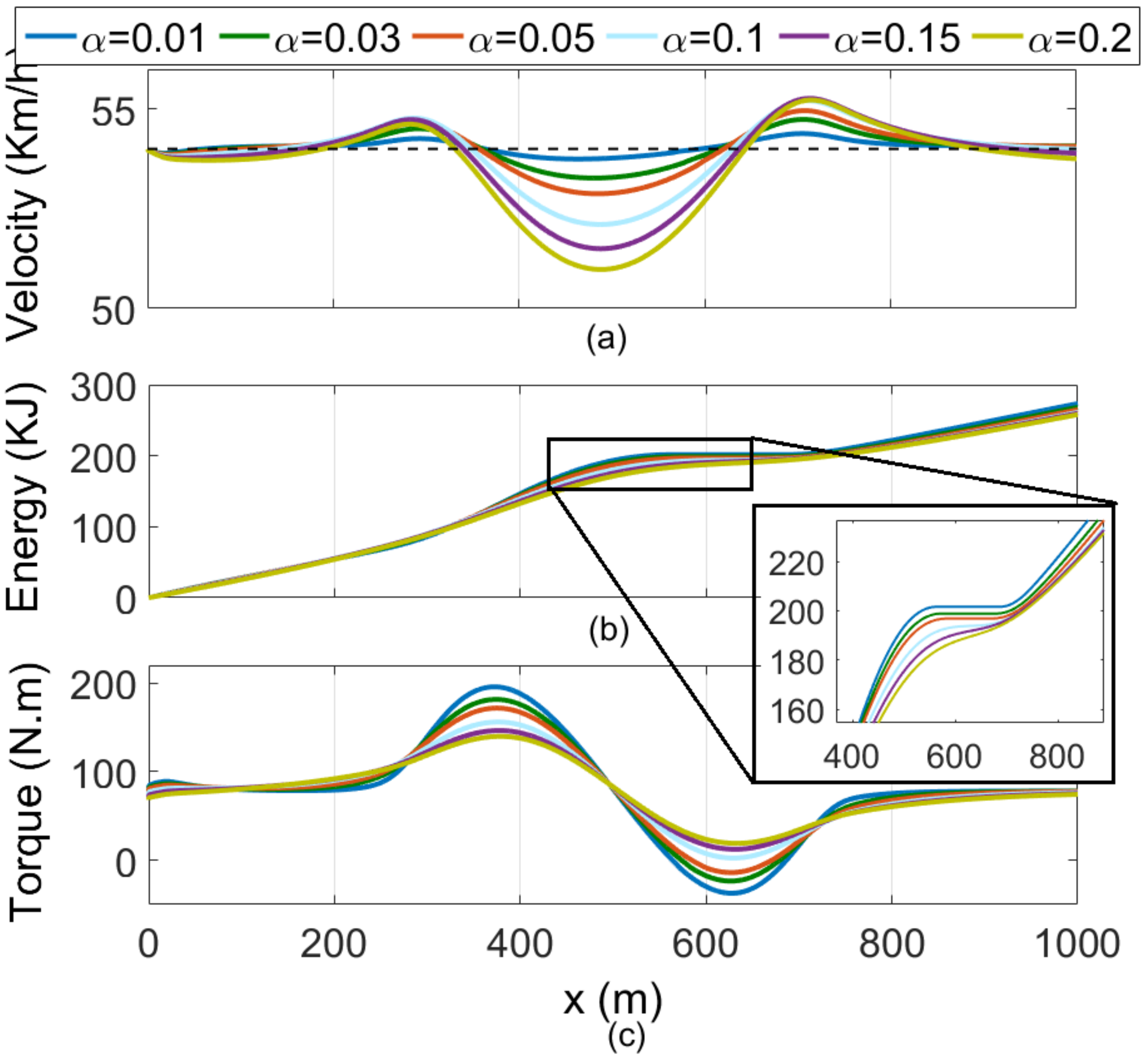


Figure 4.7: Effect of the tuning factor, α, on the performance of the NMPC controller

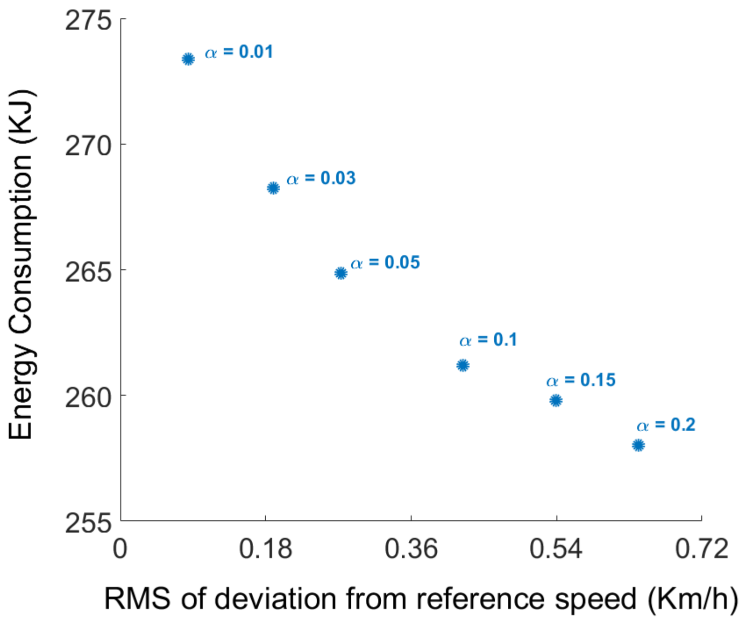


Figure 4.8: Effect of increasing α on the two performance factors; energy saving and reference speed tracking

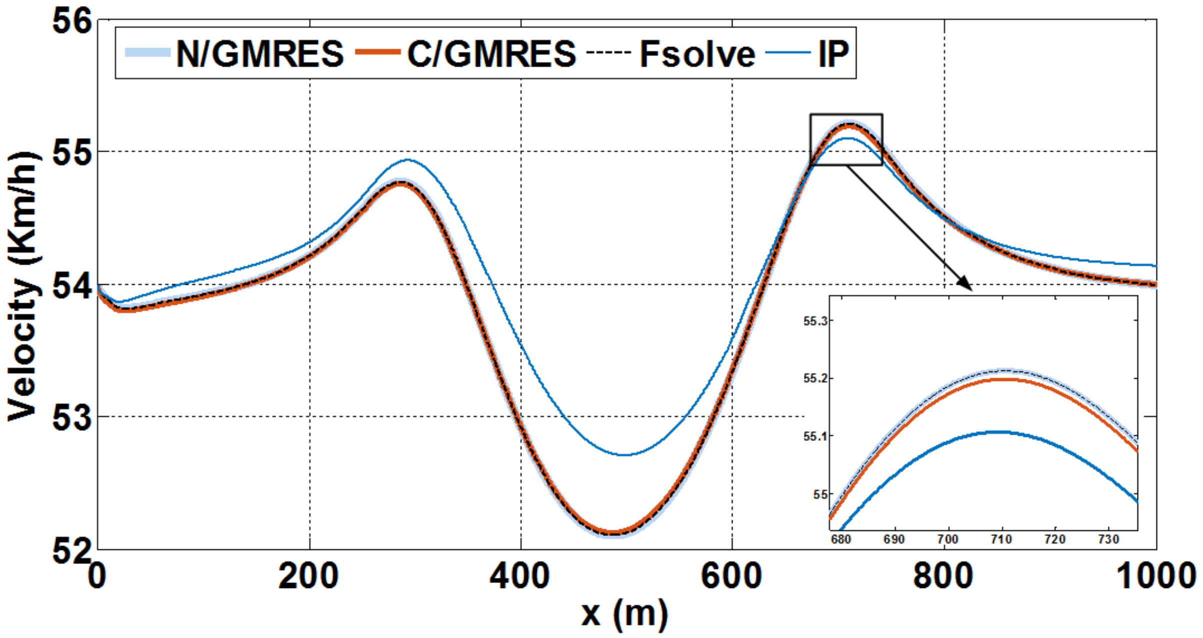


Figure 4.9: Velocity profile of the vehicles cruising over a hill by NMPC controllers with different solvers- C/GMRES, Newton/GMRES, Matlab Fsolve and Interior Point methods

for the interior-point method, MATLAB's optimization toolbox is used. Simulations are carried out with a maximum of 8 inner iterations in C/GMRES and Newton/GMRES, a prediction horizon length of 15 timesteps and 1s step time inside the prediction horizon. From an optimal performance perspective, it can be observed that the methods based on necessary optimality conditions have similar results and outperform the IP method. However, a more precise investigation of these three methods reveals the fact that C/GMRES method has a slightly different solution than the other two methods. This is mainly because of the fact that C/GMRES solves the root finding problem in continuous time and hence takes the dynamics of the system into account through the use of the predictor term ($F_x\dot{x}$) in (3.16). This improves the performance of the C/GMRES method by 0.015 % as compared to its rivals as demonstrated in Fig. 4.11. On the other hand, comparing the simulation computation times of Newton/GMRES (2.3ms) and C/GMRES (4.3ms) methods, presents the Newton/GMRES method as a superior method from a computation perspective as it is almost twice faster. Therefore, it can be concluded that Newton/GMRES is a better choice since the integration of Continuation method severely slackens the forward difference GMRES method for an insignificant improvement. According to the recent

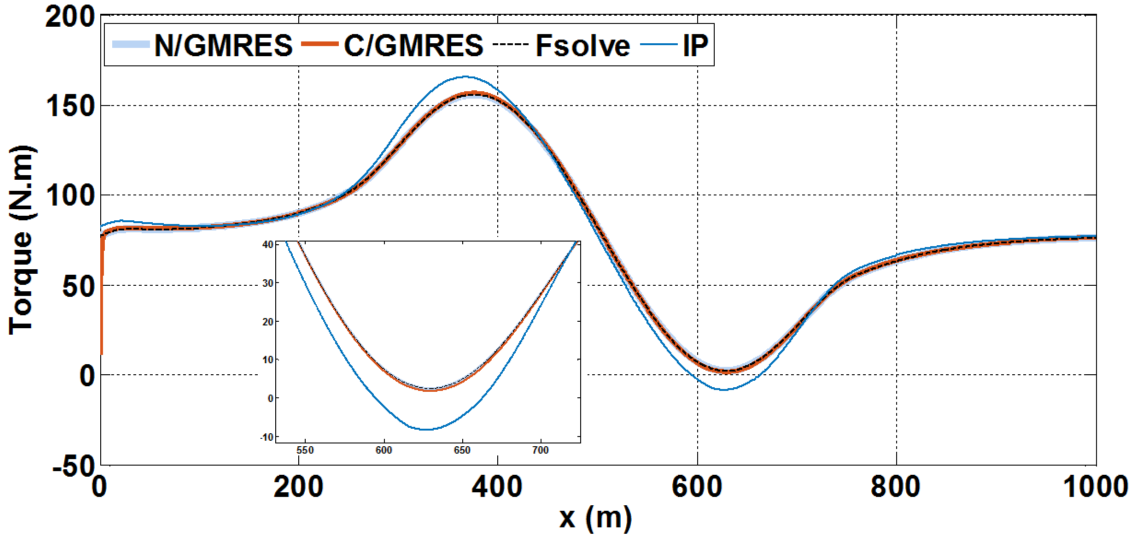


Figure 4.10: Torque demand of the vehicles cruising over a hill by NMPC controllers with different solvers- C/GMRES, Newton/GMRES, Matlab Fsolve and Interior Point methods

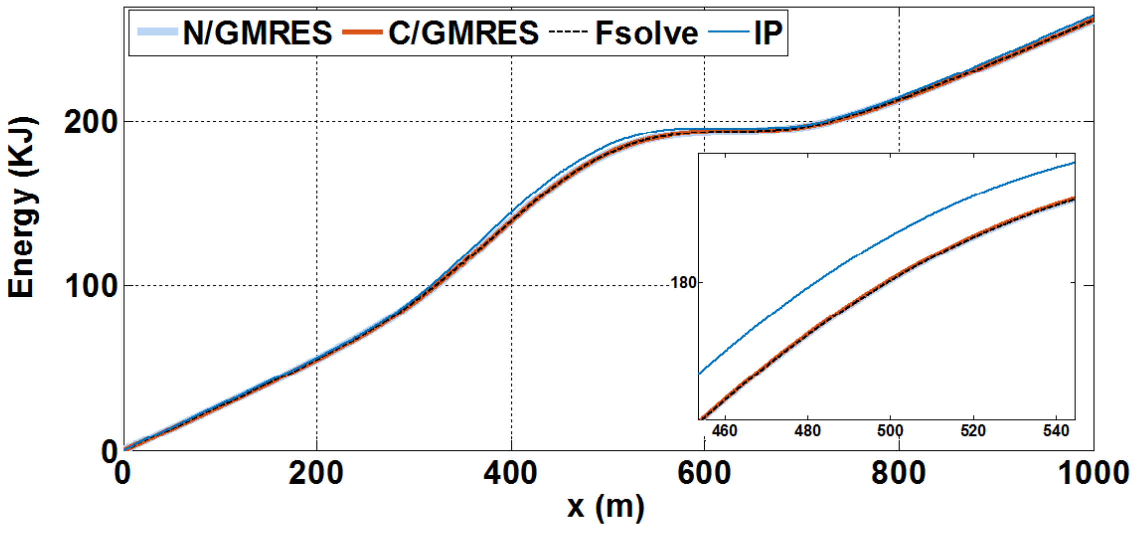


Figure 4.11: Energy consumption of the vehicles cruising over a hill by NMPC controllers with different solvers- C/GMRES, Newton/GMRES, Matlab Fsolve and Interior Point methods

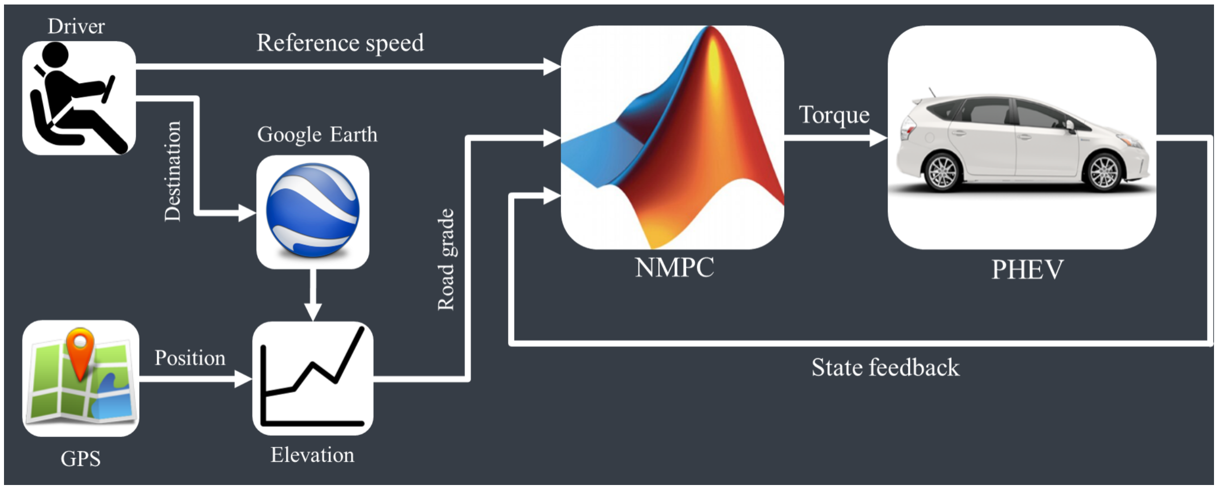


Figure 4.12: Block diagram representation of the Ecological Cruise Controller system with access to road grade information

conclusion, the developed strategic control idea is extended to a real road trip scenario. In this system, as shown in Fig. 4.12, the driver sets the destination in the beginning of the trip and then the road elevation data is immediately downloaded from a server. Here in this work, Google Earth elevation data is used for simulations. Then, according to the signal received from an onboard GPS device, road grade information is passed on to the NMPC controller. Figure 4.13 illustrates the road elevation data for a trip from Calgary to Vancouver, in Canada. Here for simulation purposes, only a part of this road is selected as demonstrated in Fig. 4.13.

Simulation results for the pre-described ECC system are presented in Fig. 4.14. As expected from the previous simulation results on a single hill, as compared to a PID controller, the NMPC controller reduces the energy consumption more than 3.5% at the cost of approximately 10% deviation from the reference speed set by the driver. Moreover, the demand torque profile is smoother with an NMPC controller which improves the ride comfort as well. However, variation in the lengths of the hills forms the following question that whether the performance of the NMPC controller is deeply dependent on the length of the prediction horizon and if yes, how it is related to the length of the hills. To answer this question, the system is simulated for several different prediction horizon lengths and results are presented in the Figures 4.15 and 4.16. Clearly, increasing the prediction horizon length has improved the reference speed tracking characteristic of the NMPC controller and on the other hand has decreased the energy-saving feature of it.

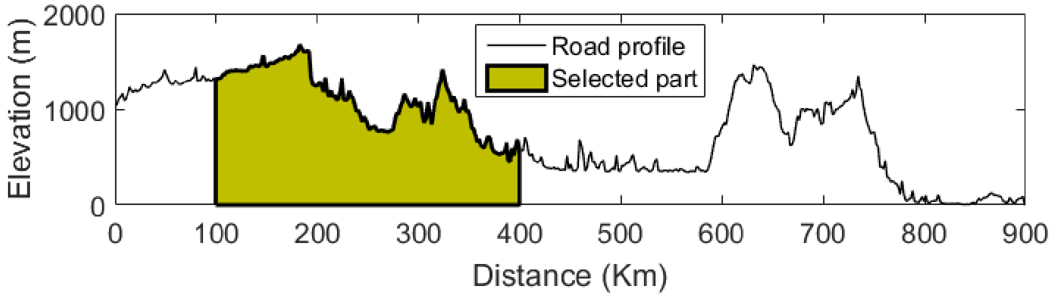


Figure 4.13: Road elevation profile for a trip from Calgary to Vancouver and the selected section for simulations

The existing trade-off between a better reference speed tracking behaviour in shorter prediction horizons and a better energy-saving behaviour in longer prediction horizons establishes the idea of an NMPC controller with an adaptive prediction horizon length (APHL). To this end, the control system shown in Fig. 4.12 is modified into a new system, shown in Fig. 4.17 and a look-up table is added to it which gives the prediction horizon length adaptive to the vehicles position read from the GPS. This look-up table is driven from the road elevation map and the values of the prediction horizon length is proportional to the derivative of the road grade as follows:

$$PH = N\frac{\kappa}{\dot{\theta}} \tag{4.5}$$

Where PH is the prediction horizon length and κ is a proportional tuning factor. Equation (4.5) adaptively regulates prediction horizon length based on the current variation of the road grade. This means that if the road grade is approximately a constant value for a relatively long portion of the trip, the prediction horizon expands in order to capture the changes in the road grade in the farther distances.

Simulation results for an NMPC controller with an APHL module show that the adaptive strategy tends to handle the trade-off in an optimal way where the energy improvement, the mean speed and the maximum speed values lie somewhere in between those of the maximum and minimum prediction horizon lengths. However, there is a significant improvement in the minimum speed of the vehicle that has been pushed up to the reference speed value. This matter can be seen in Figures 4.18 and 4.19.

A controversial argument can state that according to the optimal control theory, an infinite-horizon NMPC controller performs the best and most optimal behaviour whilst in

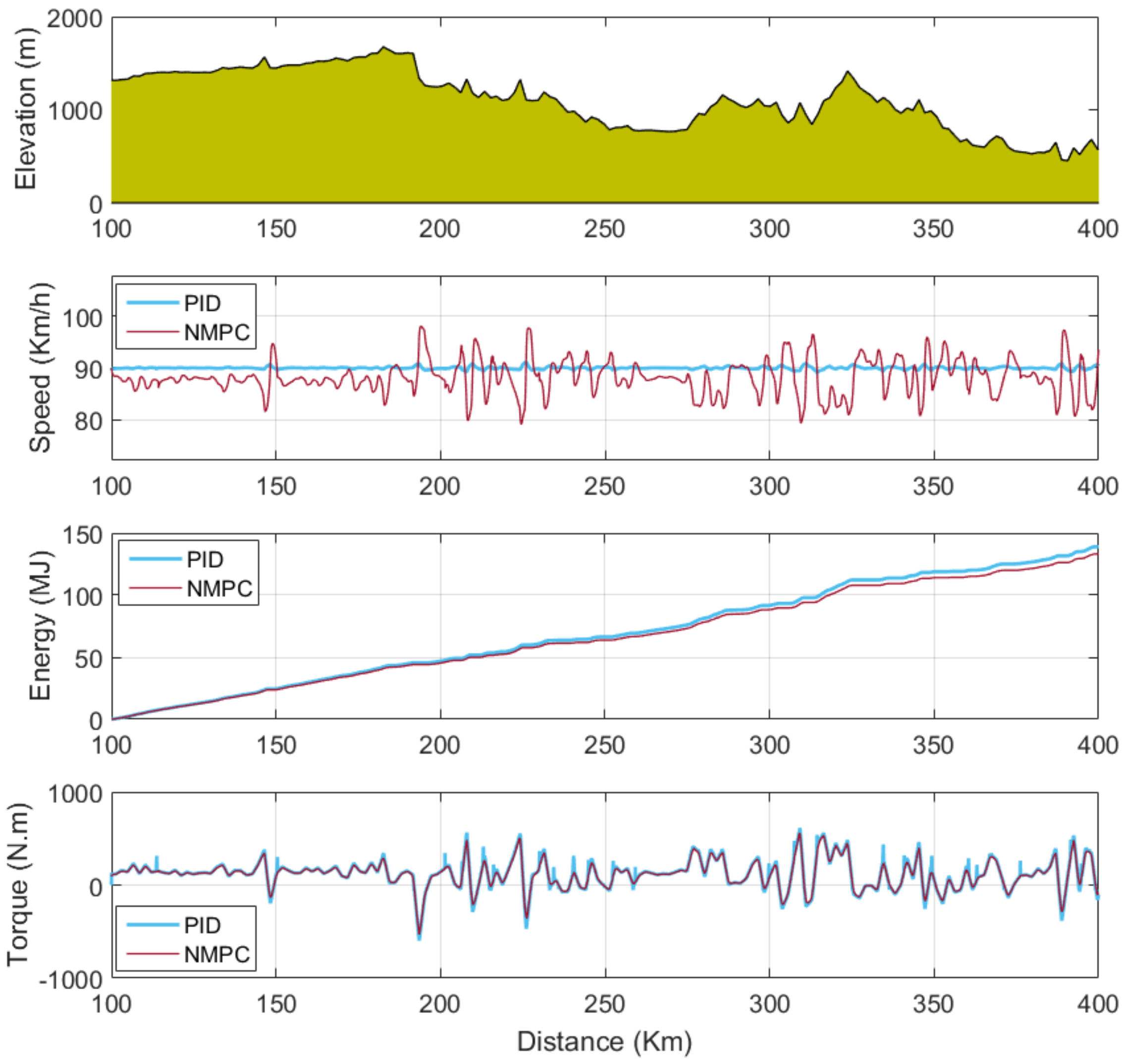


Figure 4.14: Simulation results for a vehicle travelling a section of Calgary-Vancouver road

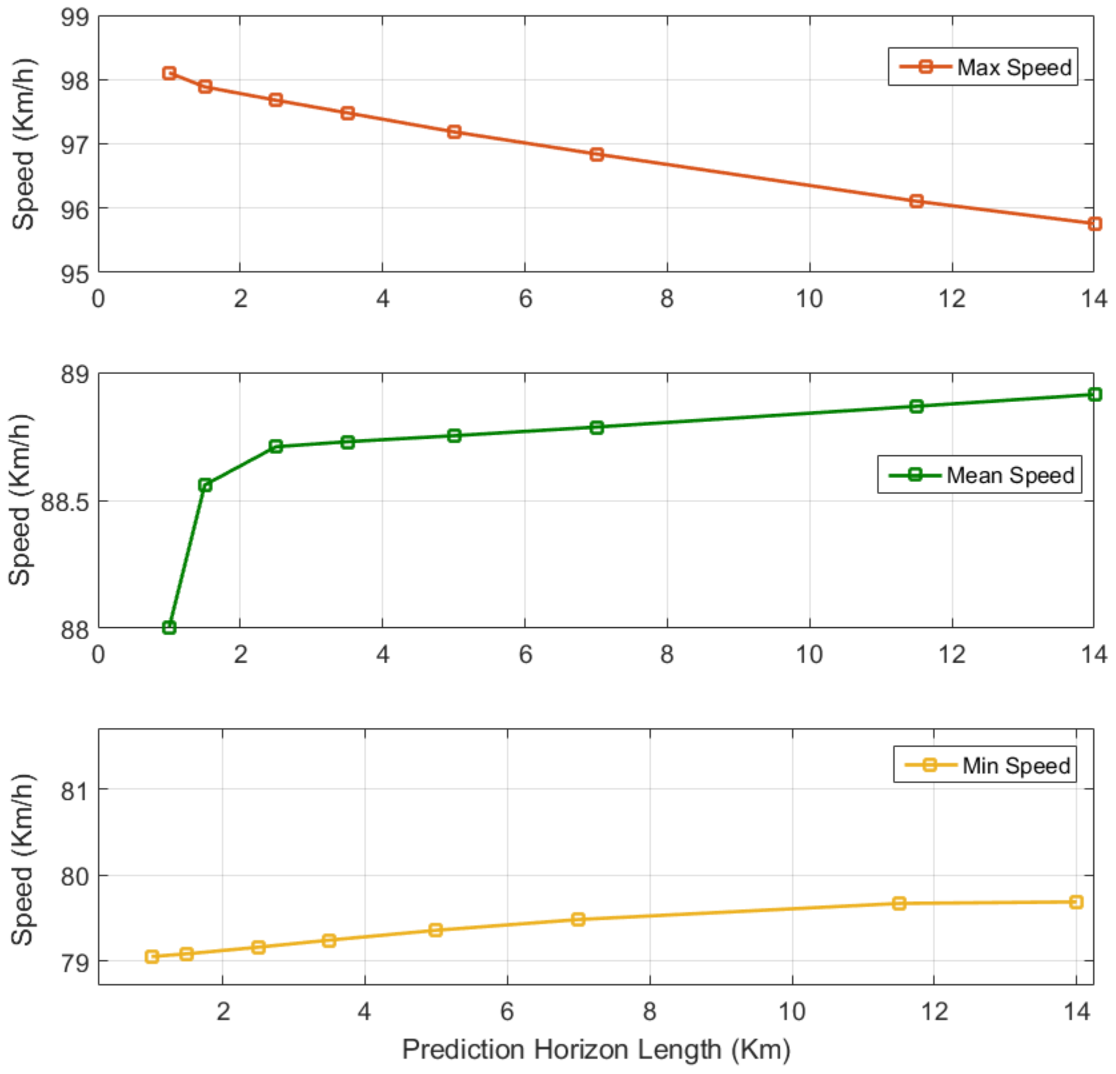


Figure 4.15: Effect of Prediction Horizon (PH) length on reference speed tracking behaviour-minimum, mean and maximum speed of the vehicle

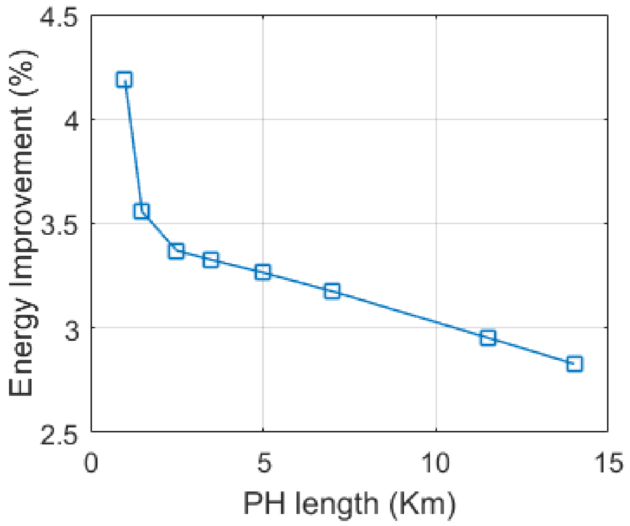


Figure 4.16: Effect of Prediction Horizon (PH) length on energy-saving behaviour

the presented simulation results this is not the case. In other words, there is a superficial contradiction in the simulation results, indicating the fact that increasing prediction horizon length can worsen some aspects of the NMPC performance, with the common expectation based on optimal control theory. To resolve this contradiction, an essential application issue must be addressed that we can only expect improvement of the NMPC controller behaviour by increasing the number predicted steps, not length of the steps. In other words, although increasing the length of the stepping time inside the prediction window broadens its length, simultaneously decreases its resolution and precision with respect to future data. For example, in a situation where the distance between two adjacent road grade sign changes are smaller than the stepping distance inside the prediction horizon, this leads to missing important data on a downhill or uphill. Therefore, in fact, the APHL module tends to dynamically handle the resolution of the prediction horizon throughout the trip. For instance, APHL decreases the prediction horizon resolution, thus increasing its length, when the future information indicates no or very small changes in the road grade value ahead. With this approach, the NMPC controller can capture data in a more informative manner and therefore the optimal trajectory planning is enhanced.

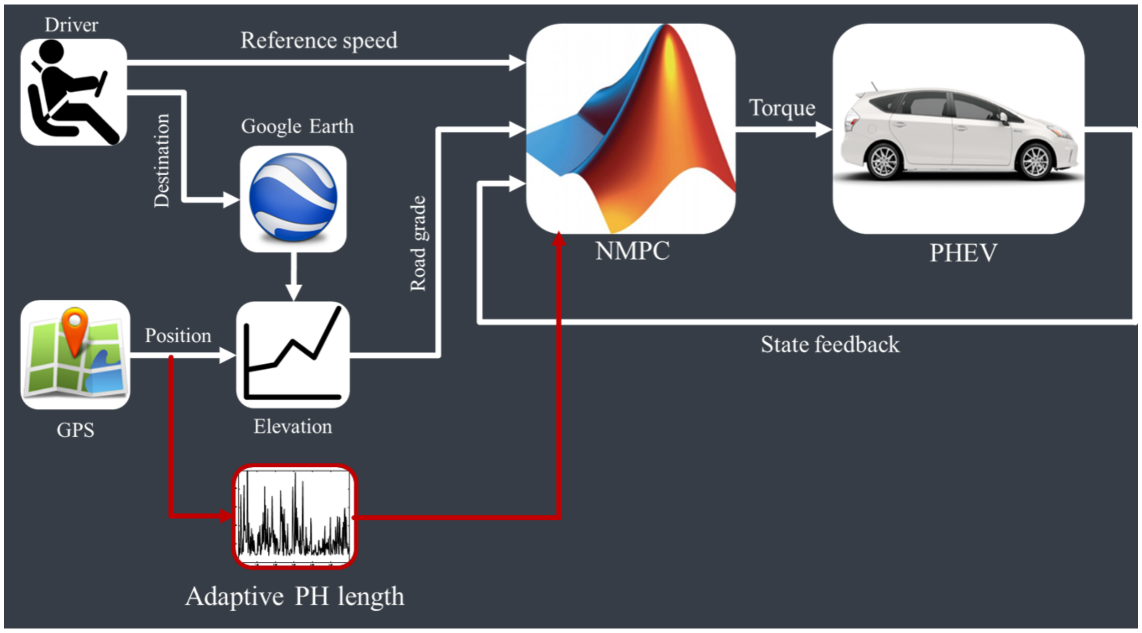


Figure 4.17: Block diagram representation of the Ecological Cruise Controller system with prediction horizon length adaptive to road information

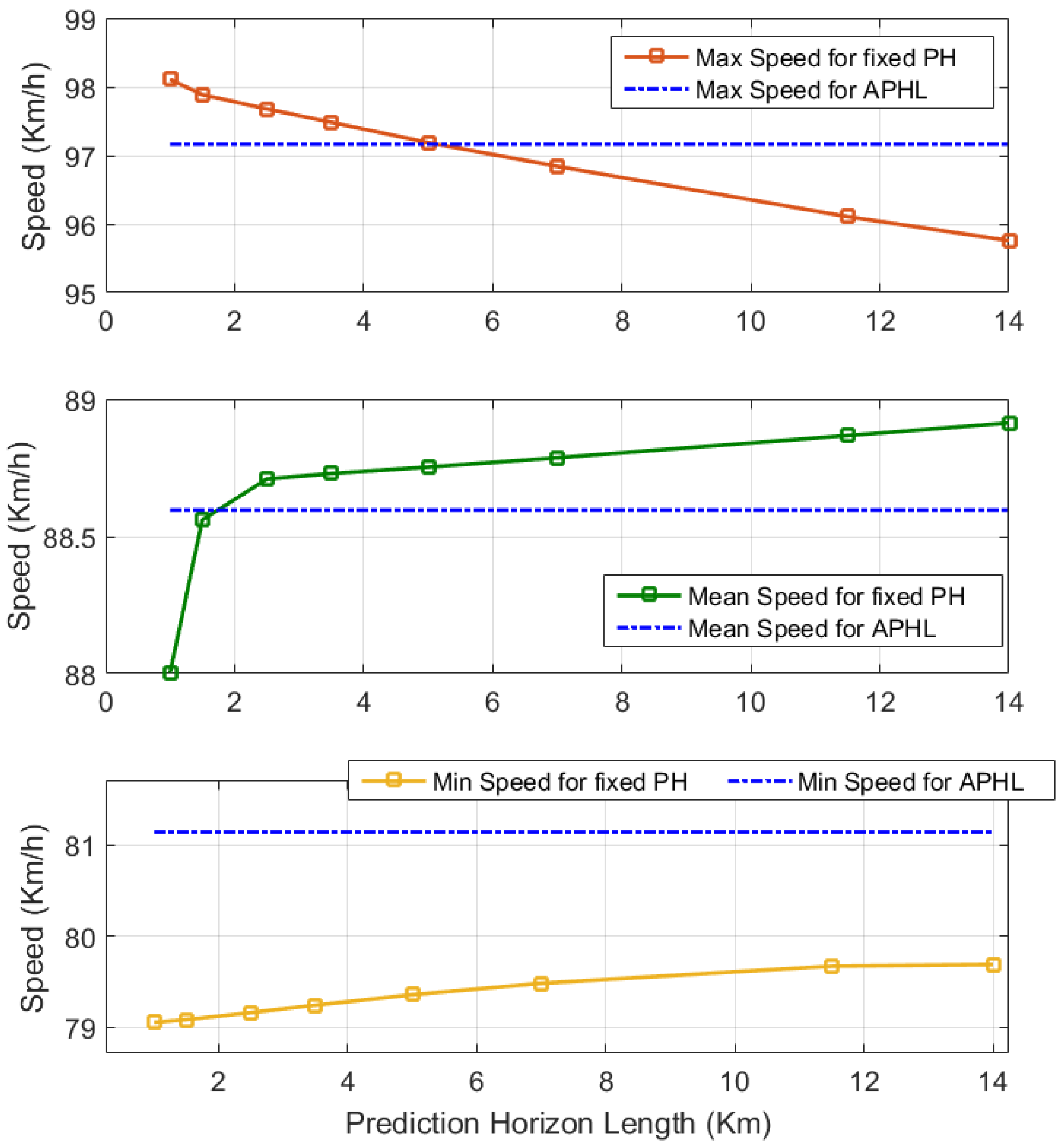


Figure 4.18: Effect of Adaptive Prediction Horizon Length (APHL) on reference speed tracking behaviour-minimum, mean and maximum speed of the vehicle

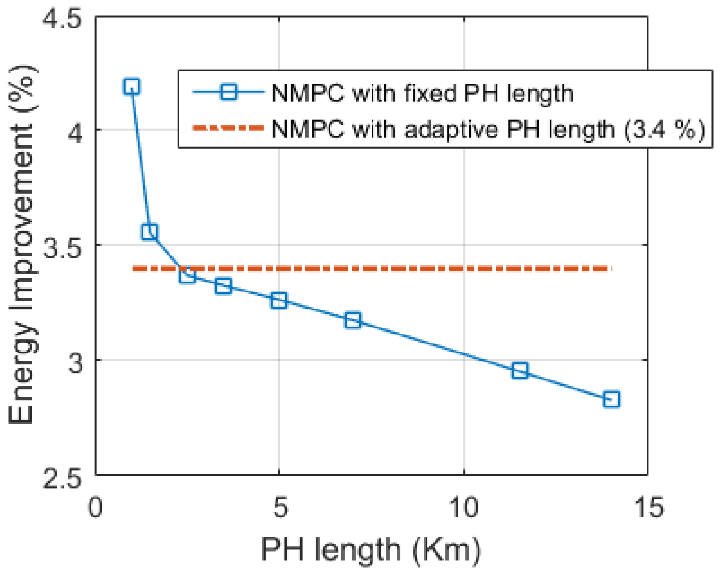


Figure 4.19: Effect of Adaptive Prediction Horizon Length (APHL) on energy-saving behaviour

4.3 Case 2: Car-following Scenario

Another cruise control problem that has received a lot of attention, is the car-following problem. In a car-following scenario, as shown schematically in Fig. 4.20, the preceding vehicle simply tracks a standard driving cycle and the host vehicle follows it with a safe distance in behind. An ACC controller utilizes onboard radar and sensors to measure the inter-vehicular distance and the speed and acceleration of the preceding vehicle and accordingly adjusts the cruising speed of the vehicle, safely. Here, the control objective is to maintain a safe distance with the preceding vehicle.

A more advanced ACC system, EcoACC, has been proposed for this problem in which the traffic condition information is collected and used to drive ecologically. The term Eco-driving is commonly used for such an optimal driving behaviour. Generally, in an EcoACC there are two main control objectives; one is minimizing the fuel consumption and the other is driving in a safe distance from the preceding vehicle at all time. Model-based approaches such as sliding mode [58], optimal control [59] and MPC [60], alongside classic rule-based approaches [61] have been used to develop ACC systems. Ability to deal with multi-input multi-output (MIMO) systems, handling inputs and states constraints and real-time multi-objective optimization have made MPC technique a suitable approach for automotive control applications as the vehicles hardware computational capacity increases. For example, integrating traffic data into MPC has shown potential in enhancing fuel economy as in [62]. Multi-input multi-output nature of the EcoACC problems along with presence of several system constraints have motivated research into applications of MPC to optimize fuel consumption, trip time, ride comfort and safety [63]. In the literature, numerous works have investigated application of MPC to EcoACC [40]. With recent increasing interest in employment of NMPC in the development of EcoACC systems [48, 5], in this work, we design and implement a Newton/GMRES-based algorithm to develop an NMPC for an EcoACC system of a PHEV, namely Toyota Prius 2013.

As stated above, MPC-based EcoACC controllers for conventional ICE vehicles have been vastly studied in the literature. However, complexity of the power-split PHEVs architecture in addition to having two sources of energy and interaction with Energy Management System (EMS) defines a whole new control-oriented model completely different from gasoline-powered vehicles. A novel control-oriented model of PHEV is developed for use at the heart of the proposed EcoACC which receives information about the optimum power distribution between ICE and Electric Motor from EMS. The main goal of the designed EcoACC is to improve total energy costs while considering safety. It is important to note that other important performance factors, such as emission levels and ride comfort, are consequently improved as well due to minimization of trip cost and avoiding sharp

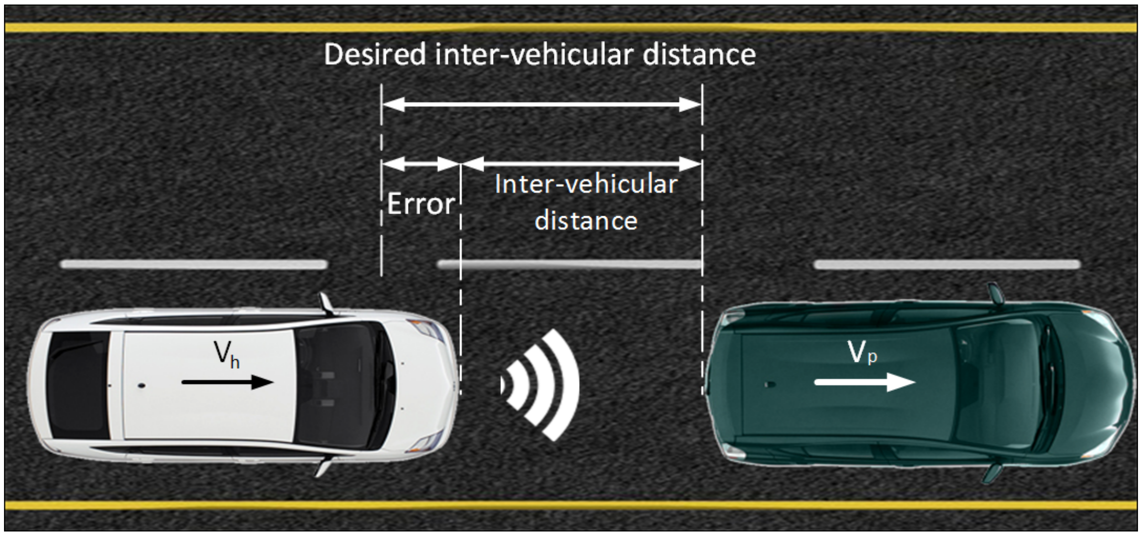


Figure 4.20: Schematic representation of a car-following problem

accelerations and decelerations. Having the advantage of using trip information previews, including road geometry and traffic flow, as well as using a nonlinear control-oriented model strengthens the proposed EcoACC and enhances the prediction. The originality of NMPC controller based on Sequential Quadratic Programming (SQP) algorithm is investigated in [44]. Moreover, in the previous section it was formerly concluded that Newton/GMRES, considering real-time implementation, is a faster solver than C/GMRES with approximately the same accuracy. Therefore, in this section, the potential of Newton/GMRES algorithm, as a promising fast optimizer for real-time implementation is examined for this EcoACC.

4.3.1 Controller Design

Any MPC controller requires a simple and computationally efficient model of the system at its heart that can be used for prediction in real-time. This section describes the steps required to develop a fast and sufficiently accurate control-oriented model for use inside the structure of the proposed NMPC controller. As shown in Fig. 4.20 which schematically illustrates the control problem, the proposed EcoACC is intended to adjust the optimal velocity of the host vehicle such that it is driven safely within a desired inter-vehicular distance, $\mathcal{L}_d$, from the preceding vehicle and also the fuel consumption is minimized. In our formulation, δ denotes the error between inter-vehicular distance, $\mathcal{L}$, and the desired inter-vehicular distance. Let us take speed and distance of both vehicles as the states and the wheel torque of the host vehicle as the input to the system. Therefore, using vehicle longitudinal dynamics theory, the system can be modelled as follows:

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{x}_h(t) \\ \dot{v}_h(t) \\ \dot{x}_p(t) \\ \dot{v}_p(t) \end{bmatrix} = \begin{bmatrix} v_h(t) \\ u(t) - \frac{1}{m}F_{res}(t) \\ v_p(t) \\ a_p(t) \end{bmatrix} \tag{4.6}$$
$$F_{res}(t) = \frac{1}{2}\rho A_f C_d v_h^2 + mg\sin\theta(x_h) + \mu mg\cos\theta(x_h).$$

Where $(v_h \ , \ x_h)$ and $(v_p \ , \ x_p)$ represent the speed and position of the host vehicle and preceding vehicle, respectively, u is the input, F_{res} is the resistance force, a_p is the acceleration of the preceding vehicle, m is vehicle mass, ρ is the air density, C_d is the drag coefficient, A_f is the host vehicle frontal area, μ denotes the rolling resistance, and θ is the road grade. It should be noted that the input, u, is in fact the wheel torque T_d divided by $(m.r)$ where r is the wheel radius. Plus, the inter-vehicular distance error is calculated as follows:

$$\delta = \mathcal{L}_d - \mathcal{L} = (\mathcal{L}_0 + hv_h) - (x_p - x_h) \tag{4.7}$$

Where $\mathcal{L}_0$, the desired distance at stop and h, the headway time, are used to adaptively change $\mathcal{L}_d$ in order to constantly drive within a safe distance. It is assumed that the speed and acceleration of the preceding vehicle are measured at each time or collected through V2V. For the purpose of prediction, an acceleration behaviour must be anticipated inside the horizon for the preceding driver. An investigation of several driving cycles reveals that it is mostly probable that a vehicle is driven at low accelerations and constant speed, [44]. Thus, the anticipated acceleration is calculated as below which gradually decreases with time:

$$\tilde{a}_p(\tau) \approx e^{-\xi\tau} a_p(t) \tag{4.8}$$

Where $\hat{a}_p$ is the predicted future acceleration, $a_p(t)$ is the measured acceleration of the preceding vehicle at current time, τ represents the prediction time, and α is a constant positive tuning parameter. Inside the prediction window, some quantities are calculated using Equations (4.9) through (4.12) as part of the control-oriented model. The PHEV (2013 Toyota Prius) has a power-split architecture with an energy management system that operates the engine at its optimal working point. In this work, Adaptive Equivalent

Consumption Minimization Strategy (A-ECMS) is employed to distribute power optimally between the energy sources [64]. Thus the predicted engine power, $\hat{P}_e$, and the battery power, $\hat{P}_b$, are obtained as fractions of the power demand, P_d, using:

$$\tilde{P}_b(\tau) = (P_b(t)/P_d(t))P_d(\tau) \tag{4.9}$$

$$\tilde{P}_e(\tau) = (P_e(t)/P_d(t))P_d(\tau) \tag{4.10}$$

where $P_b(t)$ and $P_e(t)$ are the power values, respectively, for battery and engine in the last timestep. For calculating the trip cost, the fuel consumption is approximated in kilograms per second by the following function:

$$\dot{f}_c = \alpha_0 + \alpha_1 P_e + \alpha_2 P_e^2 + \beta_1 v_h \tag{4.11}$$

Where P_e is the engine power, and a_0, a_1, a_2, and b_1 are constant parameters. Therefore, the cost of a trip is calculated and reported per distance travelled using:

$$\mathbb{C} = K_f \frac{\dot{f}_c}{v_h} + K_e \frac{P_b}{\eta_b \eta_{ch} v_h} \tag{4.12}$$

Where K_f and K_e are the unit cost of fuel and electrical energy, $\dot{f}_c$ is the rate of fuel consumption, P_b is the battery power, and η_b, η_{ch} are the battery and charger efficiencies, respectively.

A high-fidelity model of the Toyota Prius PHEV, developed in Autonomie, is employed for controller evaluation. The parameters of the current control-oriented model are identified using Least Square method and also the validity of the model is tested against this high-fidelity model by other members of the author's research group, [44].

With a control-oriented model in hand, we define the objective function and constraints and complete the design of the NMPC. Equation (4.13) gives the NMPC optimal control problem formulation where ω_i for $i = [1, \ldots, 4]$ are the weighting factors for the different terms in the objective function:

$$\begin{aligned} \underset{u}{\text{minimize}} \quad \mathcal{J} = \int_0^T [\omega_1\, \delta^2(\mathbf{x}) + \omega_2\, \mathbb{C}(\mathbf{x}, u) + \ldots \\ \omega_3\, (v_h - v_{ref})^2 + \omega_4\, (u - u_{ref})^2] d\tau \end{aligned} \tag{4.13}$$

Where v_{ref} is the reference speed to follow the preceding vehicle and u_{ref} is the input value that keeps the vehicle at a constant speed. Also, T is the prediction horizon length. The objective function is subjected to the following system model described in the modelling section, see (4.6),

$$\dot{\mathbf{x}} = f(\mathbf{x}, u) \tag{4.14}$$

as well as the following constraints that are mainly noted as one inequality equation, $g(\mathbf{x}, u) \leq 0$ in vector form:

$$v_{min} \leq v_h \leq v_{max} \tag{4.15}$$

$$u_{min} \leq u \leq u_{max} \tag{4.16}$$

As shown in (4.13), the objective function includes four terms which respectively correspond to minimizing the distance error, trip cost, reference speed tracking error and acceleration. In order to make sure that the host vehicle is continuously kept within a safe distance from the preceding vehicle ω_1 is adaptively changed, as follows:

$$\omega_1 = \begin{cases} e^{\gamma_1(\delta - 0.6\delta_{max})} & 0.6\,\delta_{max} \leq \delta \\ 1 & -0.6\,\delta_{max} < \delta < 0.6\,\delta_{max} \\ e^{\gamma_2(-\delta - 0.6\delta_{max})} & \delta \leq -0.6\,\delta_{max} \end{cases} \tag{4.17}$$

4.3.2 Controller Evaluation with MIL Simulations

To check the efficacy of the proposed GMRES-based NMPC controller, its performance is simulated for a car-following scenario where the preceding vehicle tracks a standard driving cycle. As stated earlier, the objective is to ultimately minimize the fuel consumption while maintaining a safe distance from the preceding vehicle. Simulations are carried out for two different driving cycles: three consecutive Urban Dynamometer Driving Schedule (3xUDDS)and three consecutive Highway Fuel Economy Driving Schedule (3xHWFET). The proposed predictive controller is a detailed strategy that includes several weighting factors which make it more robust and adaptive to various driving conditions. Hence, it is essential to analyse the effect of different design parameters on the behaviour of the controller. To this end, several simulations are performed for various values of a parameter while keeping the rest at their nominal values ($\omega_1 = 10, \omega_2 = 1, \omega_3 = 1, \omega_4 = 10, T = 10, \xi = 0.3, \gamma = 1$).

From a viewpoint of MPC technique, the length of the prediction horizon always plays a vital role in the performance of the controller. In general, a larger prediction window results in a more optimal control action as for an infinite prediction horizon the solution converges to that of an offline optimization method, e.g. Dynamic Programming. However, widening the prediction window makes it challenging to predict the time-varying

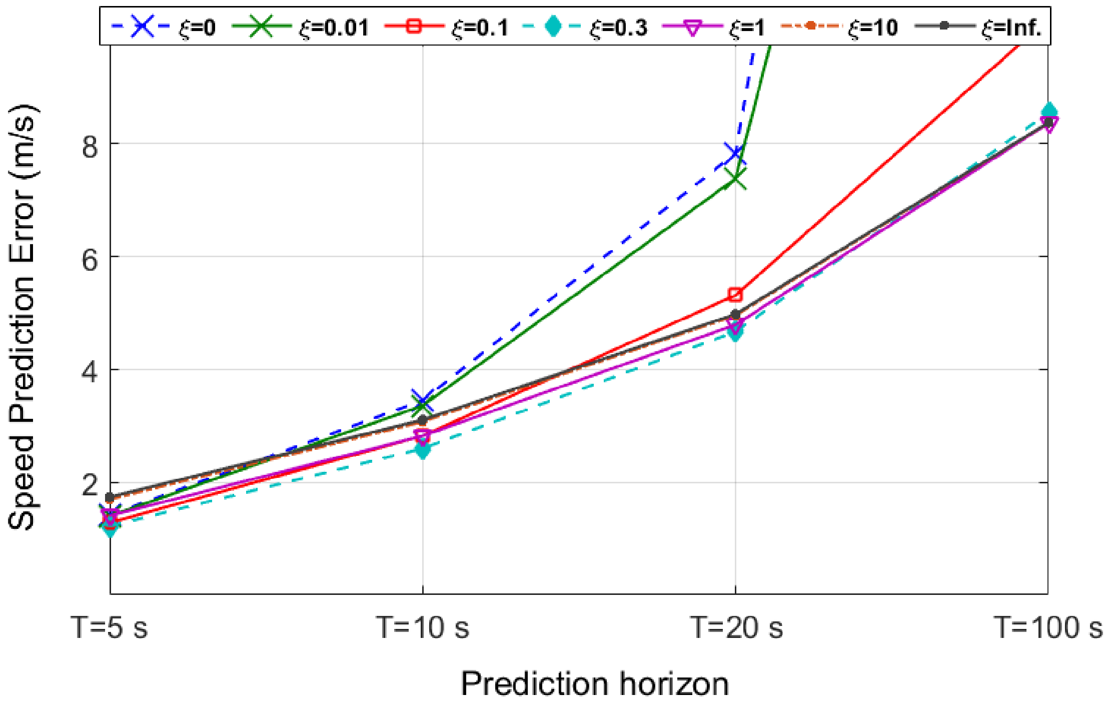


Figure 4.21: Effect of prediction horizon length on the speed prediction error for different values of ξ

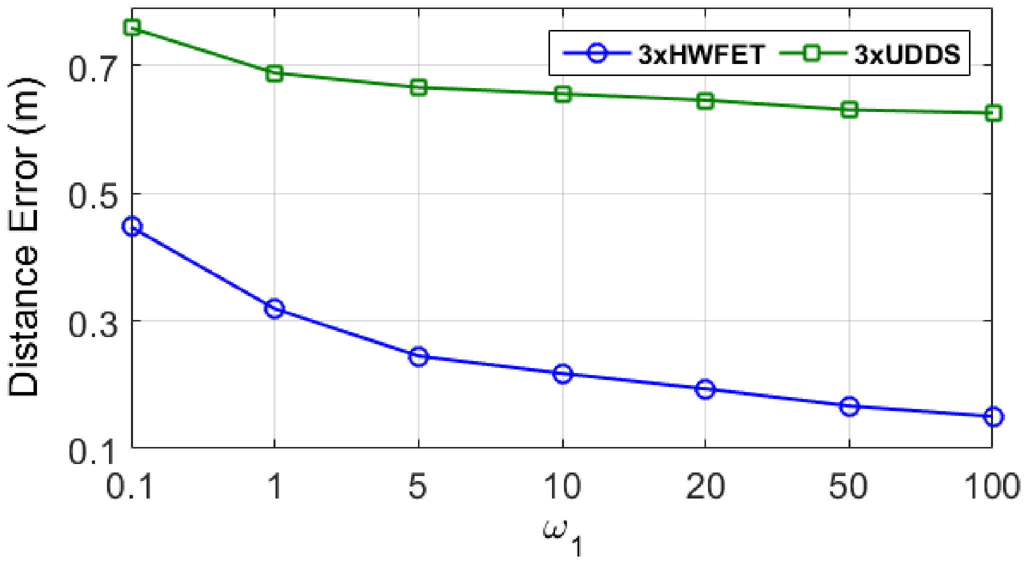


Figure 4.22: Effect of ω_1 on inter-vehicular distance error

parameters in future, due to arising uncertainties. For example, in this case, predicting the driving behaviour of the preceding vehicle, namely its future speed, becomes more and more difficult as the length of the prediction horizon increases. Figure 4.21 demonstrates the error in speed prediction of the preceding vehicle for different horizon lengths meaning that using (4.8) for this purpose encounters a significant rise in error for large prediction horizons ($T > 15s$). Moreover, increasing the prediction horizon length enlarges the computational load significantly that makes the problem impossible to solve in real-time, in practice. Accordingly, this leaves us with a trade-off situation for selecting the prediction horizon length with an optimum value.

Different terms of the objective function contribute individually to the performance of the NMPC controller, however, a preliminary investigation showed that the influence of ω_1 and ω_4 is more significant on the safety. Figures 4.22 and 4.23 illustrate the effect of increasing these two design parameters on the distance error. In other words, although increasing ω_1 reduces the error in driving within a desired distance from the preceding vehicle, the controller's sensitivity to this factor is saturated beyond a certain value ($\omega_1 \approx 5$) and further increase in ω_1 only makes the controller inresponsive to the other three terms. Despite ω_1, increasing ω_4, which has a positive effect on the energy saving, increases the distance error as shown in Fig. 4.23.

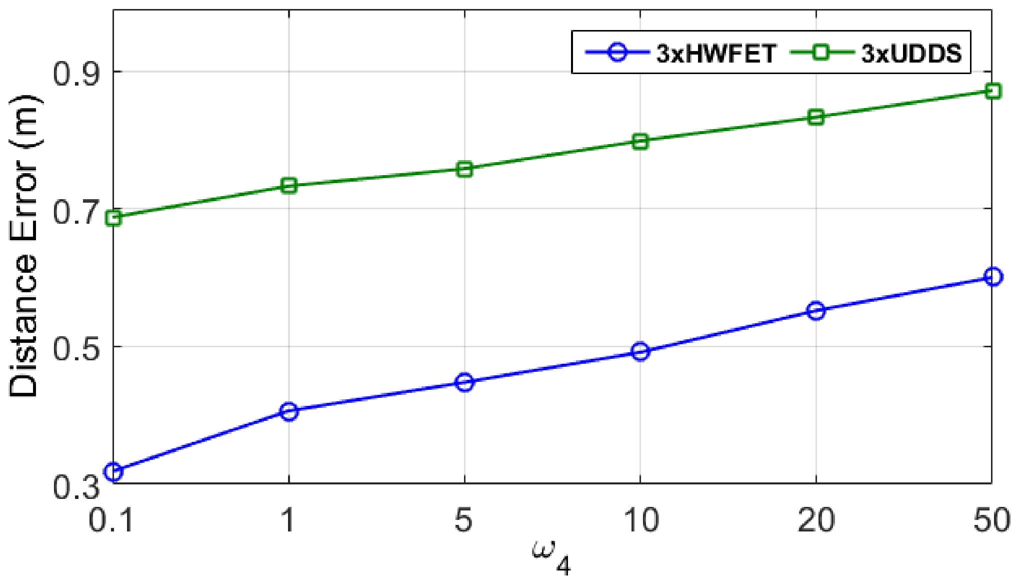


Figure 4.23: Effect of ω_4 on inter-vehicular distance error

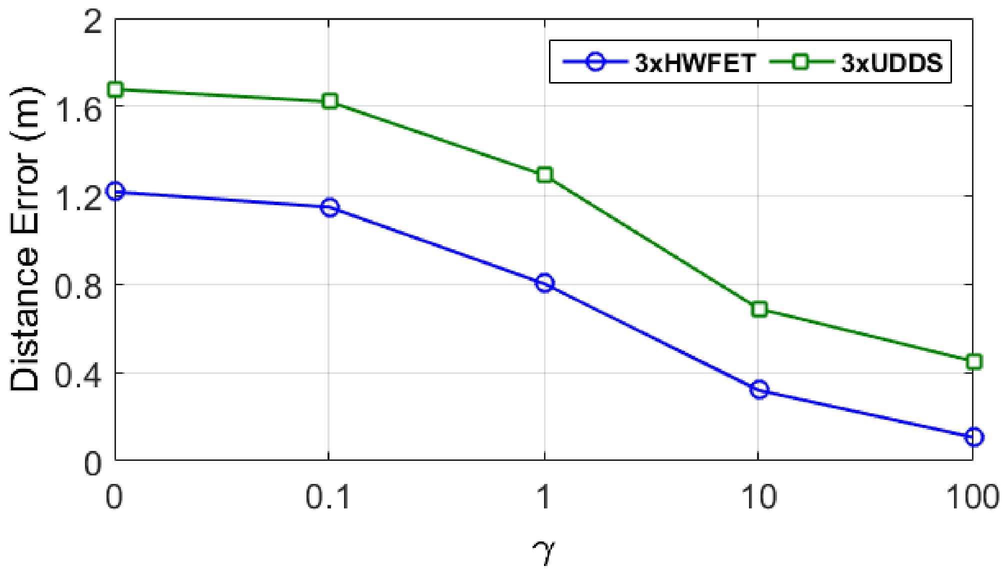


Figure 4.24: Effect of γ on inter-vehicular distance error

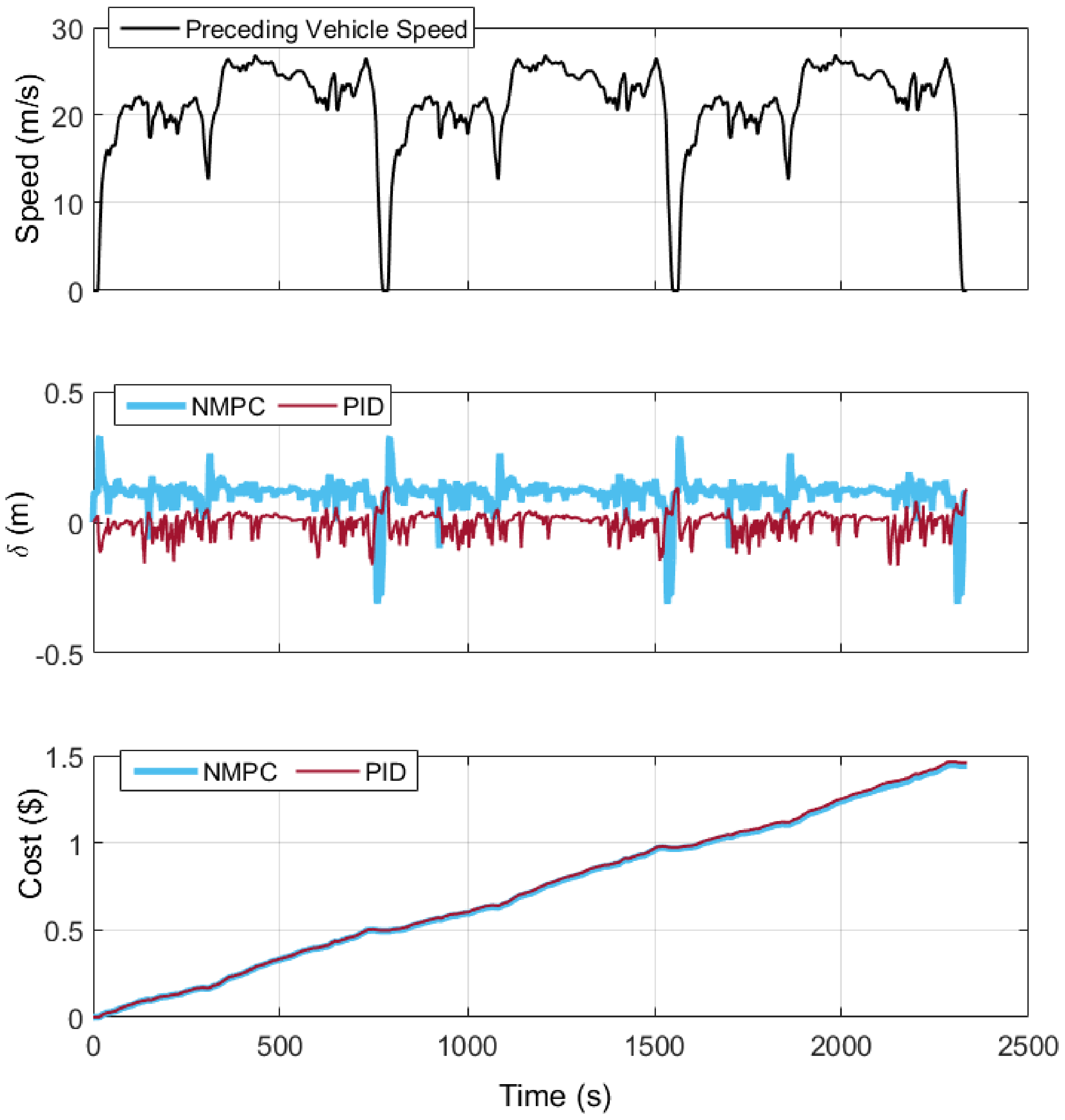


Figure 4.25: Simulation results for following a car driving 3xHWFET cycle

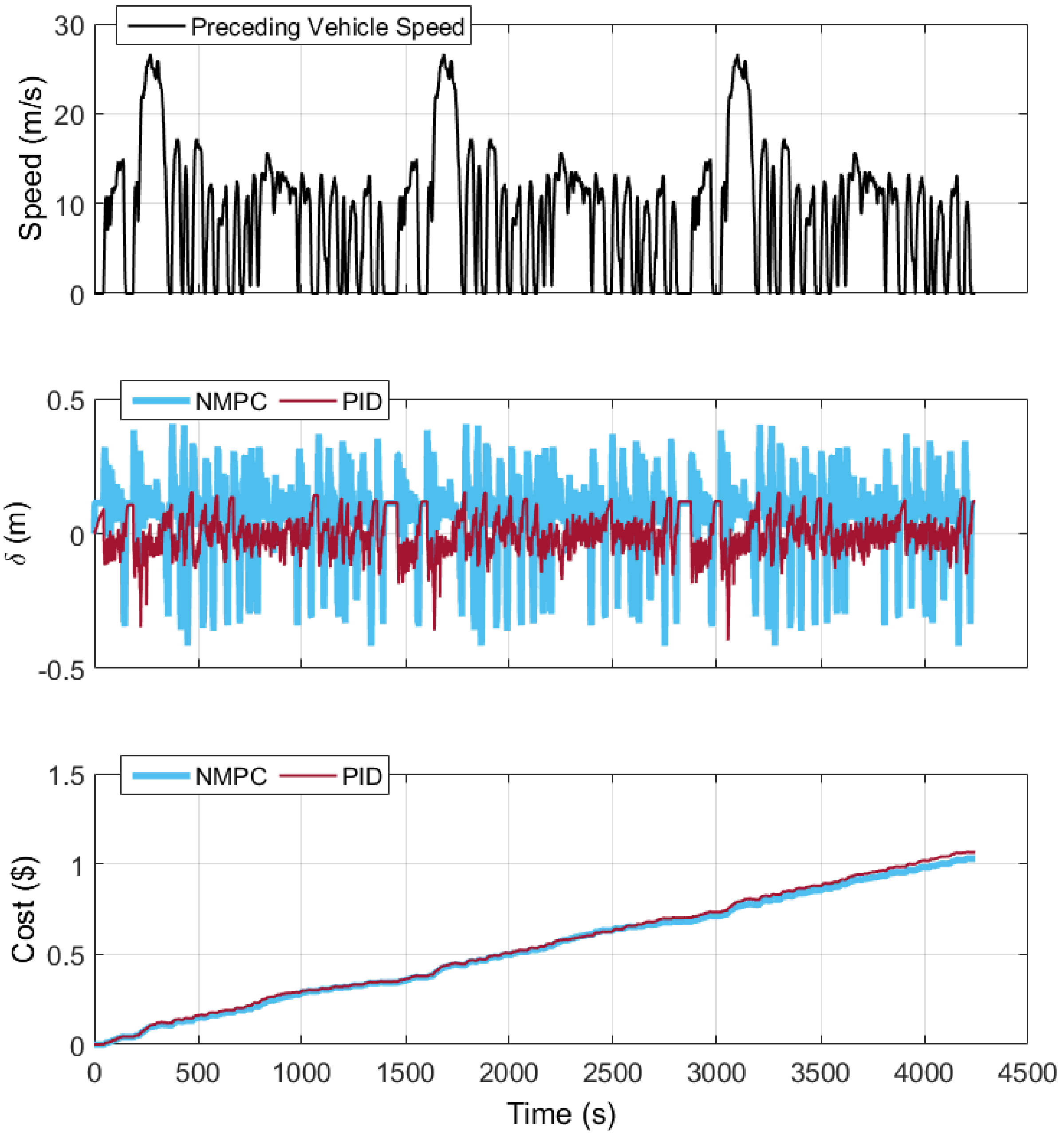


Figure 4.26: Simulation results for following a car driving 3xUDDS cycle

As stated earlier, ω_1 plays an essential role in keeping the vehicle at a safe distance. However, selecting a large value for ω_1 can result in disregarding the main goal of this work, i.e. driving ecologically. Therefore, adaptively changing this weighting factor using (4.17) is a brilliant solution toward optimally handling this trade-off. The two design parameters in (4.17), γ_1 and γ_2, are symmetrically selected, simply denoted as γ, hereafter. Figure 4.24 shows the impact of increasing γ on the reduction of the distance error, where similar to ω_1, its influence is significant in a certain range ($0.1 < \gamma < 10$) and then saturates in higher values ($\gamma > 10$). In this case, the simulation results show that at very high values of γ ($\gamma > 100$) the controller aggressively involves abrupt accelerations or decelerations.

For a better demonstration of the ecological behaviour of the proposed NMPC controller, its performance is simulated on a high-fidelity model of the Toyota Prius PHEV 2013 and evaluated against a PID controller. Figures 4.25 and 4.26 illustrate the MIL simulation results for 3xHWFET and 3xUDDS, respectively. These results indicate that the proposed NMPC-based E-ACC system can outperform a classic PID controller by 1.2% for driving at highway and a 3.4% for urban driving in terms of trip cost by slightly (less than 0.5m) deviation from the desired inter-vehicular distance.

4.4 Hardware-in-the-loop Simulations

This section describes the necessity of performing hardware-in-the-loop (HIL) testing, specially in the development procedure of a control system. The designed intelligent cruise controllers based on NMPC technique, in the previous chapter, are tested through HIL testing and the results are presented, accordingly.

In the early stages of the design and development of vehicle control systems, physical prototyping can be very expensive and time-consuming. In this case, one of the well-recognized rapid prototyping methods, widely used these days by the automakers, is the hardware-in-the-loop testing. In a situation where purely virtual simulations are not adequate to capture great fidelity levels of performance evaluation (often because of inability to model components whose dynamics are not fully understood), HIL techniques become very advantageous by prototyping only some complex and essential components of the system. This, not only enhances the fidelity of the performed tests, but also reduces the validation time and cost as the HIL simulations often require much less hardware components than the actual physical prototyping. Moreover, repeatability is an essential principle in a design process and in the systems with highly variable conditions, physical prototyping can not be an option. Whereas, through HIL simulations, a wide range of operating conditions can be simulated over and over in a controlled lab environment. In addition to repeatability, HIL testing also reduces the risk of danger and the costs associated with destructive tests. For example, HIL testing can simulate the severe environment conditions such as winter test drives to avoid a high risk of collision. All of these will result in a more rapid pace in the process of design, saving significant amount of time, effort and cost, as well as preparing a solid calibration foundation for further control system development phases.

Electronic Control Units (ECUs) are a category of embedded systems that read information form various sensors and use actuators to control different systems in vehicles. ECUs are now widely used to control various vehicle subsystems including engine, transmission, brake and suspension systems. Some modern advanced vehicles include up to 80 ECUs [65]. With the rapid increase in the number of ECUs used in today's vehicles, development and validating them has become a crucial task in the automotive industry. HIL testing is the most effective technique to develop and evaluate new advanced ECUs in shortest time. Therefore, in this thesis, the author investigates the implementability of the proposed intelligent cruise control systems on an ECU through HIL testing.

Typically an HIL platform for Rapid Control Prototyping (RCP)consists of three main components: a user interface for setting up the simulation, programming the hardware and reading and storing the outputted variables; a real-time simulator which is actually a super fast processor for running the high-fidelity model in real-time; and a prototype ECU for running the control law in real-time. These three components are shown in Fig. 4.27. Connections between these three components are made through a Controller Area Network (CAN). CAN is widely used within automobiles to allow different microcontrollers and ECUs to communicate without the need for a host computer. Special I/O ports of the HIL components compatible with CAN interface allows an efficient system of sending commands and receiving feedbacks. In this research, dSPACE HIL setup is utilized for simulations. dSPACE Inc. provides multiple products for design, development, RCP and HIL testing of embedded control systems which is well-recognized by many praised automakers such as Toyota, General Motors, Honda, Ford and BMW. High compatibility of dSPACE integrated development environment with MATLAB/Simulink creates a straight-forward tool chain for model-based development. Also, adding MATLAB-based AuMuSoN code generator for nonlinear MPC controllers to this chain, completes the necessary components for design, development, test and evaluation of real-time NMPC controllers for automotive applications.

4.4.1 Hardware Description and Programming

As shown in Fig. 4.27, dSPACE prototyping system includes the prototype ECU (MicroAutoBox II) and the real-time simulator (DS-1006 Processor). The specifications for

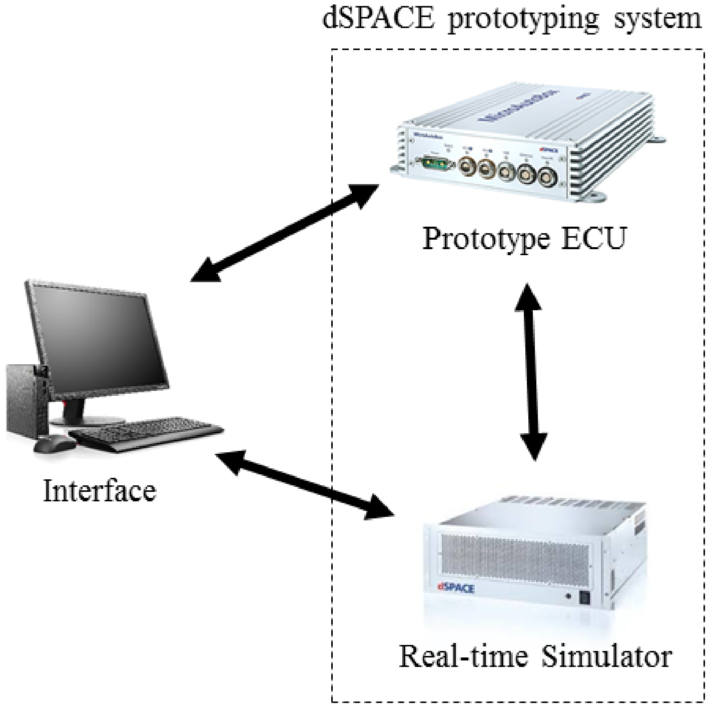


Figure 4.27: Principle of HIL testing

Table 4.1: dSPACE HIL specifications

Specification	Real-time simulator	Prototype ECU
Hardware	DS-1006 Processor board	MicroAutoBox II
Processor	DS-1006 Quad-Core AMD, 2.8 GHz	DS-1401 PowerPC 750GL 900 MHz
Memory	1GB local, 4x128 MB global	16 MB main, 16 MB nonvolatile
I/O	DS-2202	DS-1511

these two components are presented in Table 4.1. The real-time simulator runs a high-fidelity model of the vehicle and environment and communicates with the prototype ECU, which calculates and sends command signals, through CAN bus.

Fortunately, dSPACE provides related MATLAB libraries to generate compatible and embeddable C codes for its hardware. Any Simulink model can be compiled using Real Time Workshop code generator with the corresponding complier of the target hardware. For example, here we use rti1401.tlc and rti1006.tlc, repectively, to generate C codes for MicroAutoBox II and DS-1006 Processor board. It is important to note that the real-time simulator executes one part of the Simulink model (high-fidelity model) and the prototype ECU runs another (NMPC controller). Therefore, the Simulink model used for MIL simulations must be modified and divided into two separate *.mdl files. To do so, one should use the dSPACE Real Time Interface blocks, for instance RTI CAN Controller Setup Block, to configure the CAN terminals for the communication between these two components. The communicating signals between the two Simulink models must have the same CAN ID in both the high-fidelity model file and the controller file.

After preparing the two Simulink files and generating the corresponding C codes, system description *.sdf files are generated. These files can be located in dSPACE ControlDesk, which is a software that allows the user to use a computer as an interface to interact with the hardware, to upload the codes, run the test and record the desired signals. Once the codes are loaded onto the specific platforms, desired variables from both of the platforms, i.e. the prototype ECU and the real-time simulator, can be selected from the variables panel for recording. One of the most important variables that must be measured is the turnaround time of the prototype ECU which determines whether the designed controller can actually be implemented in real-time on an ECU or it is not fast enough to meet the computational speed limits. Turnaround time is the time taken by the controller to generate an input action upon receiving a feedback from the system. Thus, in the following sections we perform HIL simulations for the proposed intelligent cruise controllers and report the turnaround time on the prototype ECU.

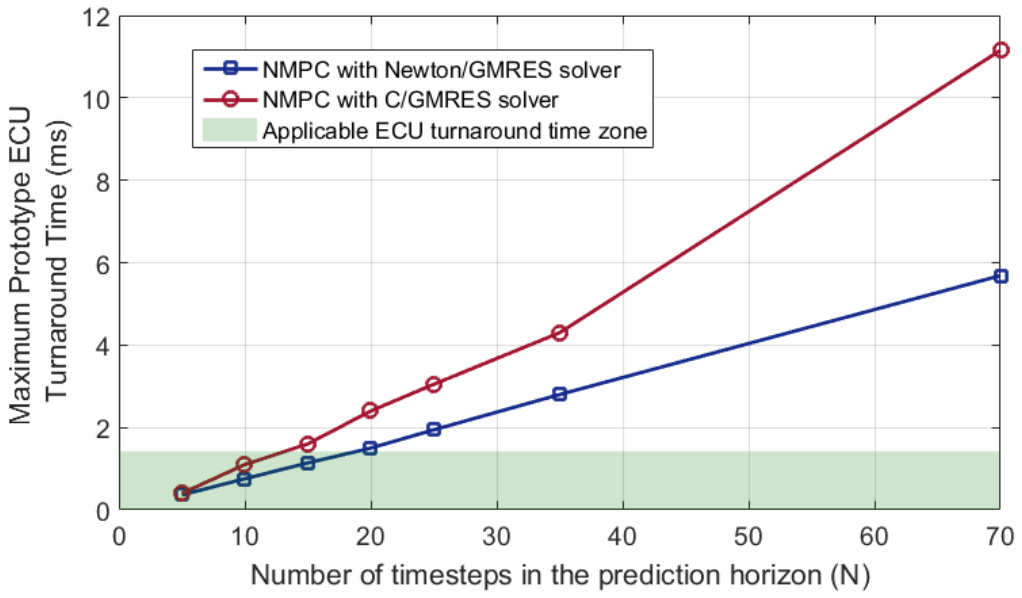


Figure 4.28: Comparison of prototype ECU turnaround time for different optimizers and prediction horizon sizes

4.4.2 Case 1: Driving on Hilly Roads

HIL simulations for the ECC controller designed earlier in this chapter are performed for a part of the Calgary-Vancouver road trip, whose road elevation profile is shown in Fig. 4.13. The simulations are carried out for Newton/GMRES and C/GMRES RTO algorithms with several different number of timesteps in the prediction horizon and the results for the prototype ECU maximum turnaround time are presented in Fig. 4.28. In this figure, the green area indicates the range of prototype ECU turnaround time for a practically applicable control strategy. Compliant with the results previously observed in the MIL simulations, the C/GMRES RTO algorithm is slower than the Newton/GMRES optimization method. However, here, another important outcome is observable about the computational speed of these two GMRES-based algorithms. The ratio of the C/GMRES turnaround time to the Newton/GMRES turnaround time increases as the size of the prediction horizon gets larger. In other words, the C/GMRES RTO algorithm is more sensitive to the size of the prediction horizon than the Newton/GMRES method. Therefore, it can be concluded that for the smaller problems with small prediction horizons, utilizing

Table 4.2: Estimated turnaround times for Prius ECU based on measured prototype ECU turnaround times for different prediction horizon sizes

PH size	Prototype ECU turnaround time	Estimated Prius ECU turnaround time
N=5	0.1 ms	0.7 ms
N=10	0.12 ms	0.85 ms
N=20	0.9 ms	6.6 ms
N=100	10 ms	70 ms

a C/GMRES optimizer can be a better choice since this method is slightly more accurate than the Newton/GMRES method. However, the computational time for this method rapidly increases with the size of the problem and the prediction horizon. So, in the case of a problem with larger prediction horizon, the Newton/GMRES method is a better choice.

In our work, for the proposed ECC controller, prediction horizon sizes of up to 14 and 19 are practically applicable on an actual ECU, respectively, for C/GMRES and Newton/GMRES methods. For the ECC controllers with the prediction horizon size of 14 or less, a C/GMRES algorithm is implementable on the Toyota Prius ECU. Similarly for Newton/GMRES method, ECC controllers with the prediction horizon size of 19 or less can be implemented on the Toyota Prius ECU. It is important to note that, utilizing simpler real-time optimization methods for the NMPC controller or even employment of a linear MPC controller instead of a nonlinear MPC controller will increase the range of applicable prediction horizon size. While increasing the size of the prediction horizon enhances the task of prediction and increases the accuracy, simplifying the controller by linearising or using less accurate optimizers will cripple the performance of the controller. This is a trade-off that must be investigated for every designed controller.

4.4.3 Case 2: Car-following Scenario

A set of HIL simulations for determining the applicable prediction horizon size is preformed, as well, for the EcoACC controller designed for a car-following scenario. These simulations are performed for a controller based on the Newton/GMRES algorithm. The resulted turnaround times are listed in Table 4.2 that show promising values for implementing controllers with prediction horizons less than 20 timesteps.

The HIL testing results ,reported in this section for the two proposed intelligent cruise controllers, show promising data in terms of computational speed and implementability of real-time NMPC controllers for automotive applications.

Chapter 5

Conclusion

This thesis has described the development of a mathematical program based on MATLAB for the purpose of automatically generating MPC control codes for nonlinear systems, in particular automotive systems. The code generation tool developed in this work is mainly used for the design and development of intelligent cruise control systems for vehicles. The automatic code generator, called AuMuSoN, has a user-friendly MATLAB interface for defining different optimal control problems and generates controller codes based on GMRES-based RTO algorithms for various MIL and HIL simulations and tests. The automatically generated NMPC codes by AuMuSoN were tested and validated with NMPC codes generated by other well-known C/GMRES-based code generation tool, AutoGenU. The presented results indicated a good agreement between AutoGenU and AuMuSoN controller performances.

Then, the code generator was used to develop two intelligent cruise controllers with a main goal of minimizing the energy consumption for a PHEV, namely Toyota Prius. The first proposed control strategy, called an ECC controller, was developed for a scenario in which a vehicle travels a road with up/down slopes. In this scenario, a comparative study was conducted in order to investigate the differences between the two GMRES-based RTO algorithms: Newton/GMRES and Continuation/GMRES. Results indicated that the Newton/GMRES algorithm is a faster optimizer than the C/GMRES method with approximately the same performance. Hence, Newton/GMRES algorithm was chosen for the rest of the control evaluations. Control strategy was tested on a real-life road elevation profile and a 3.5% reduction in the energy consumption, compared to a PID controller, was observed when the ECC controller regulates the speed of the vehicle by approximately 10% deviation from the reference speed set by the driver. In addition to that, the originality of the idea of using an adaptive prediction horizon length module to dynamically change the resolution of the prediction horizon was investigated to improve the NMPC controller performance by capturing the environment data in a smarter manner.

The second intelligent cruise controller developed in this thesis studied a strategic vehicle control system in a car-following scenario. The main objective of the designed EcoACC controller was to reduce the cost of a trip while maintaining the host vehicle within a safe distance of the preceding vehicle. After running numerous simulations in order to study the effect of different adaptive weighting factors on the performance of the controller, simulation results for urban and highway drive cycles demonstrated 3.4% and 1.2% respectively, as compared to a PID controller.

Lastly, a set of HIL tests was performed on a dSPACE MicroAutoBox II platform to check the applicability of the designed intelligent cruise controllers. The HIL simulation results indicated that the C/GMRES solver speed is more sensitive to the size of the prediction horizon than the Newton/GMRES solver. Therefore, it was concluded that in order for the ECC controller to be applicable on the Toyota Prius ECU, the maximum size of the prediction horizon for a C/GMRES-based NMPC controller is 14 and for a Newton/GMRES-based NMPC controller is 19.

5.1 Summary of Contributions

The major contributions of this research are summarized as follows:

1. Development of an automatic multi-solver NMPC code generation tool
   - First GMRES-based NMPC controller code generation tool based on MATLAB
   - Features a user-friendly graphical interface for defining optimal control problem
   - Allows the user to choose the two GMRES-based solvers, C/GMRES and Newton/GMRES, for the use inside the controller
   - Enables the user to apply several equality and inequality constraints to the problem with Exterior Penalty Inequality Constraint Handling method
   - Generates MATLAB codes and Simulink files for MIL simulations that can be easily compiled to C codes for HIL simulations
2. Presentation of a comparison between Continuation/GMRES and Newton/GMRES Real-Time Optimization (RTO) algorithms in case of accuracy and speed

3. Design and development of intelligent cruise control systems for a PHEV
    - Development of an ECC controller that minimizes the energy consumption of the PHEV with small deviations from the reference speed set by the driver in trips on hilly roads
    - The proposed ECC controller features an Adaptive Prediction Horizon Length (APHL) module to enhance the NMPC controller performance
    - Development of an EcoACC controller for a PHEV in car-following scenarios that minimizes the trip cost and maintains the vehicle within a safe distance
    - Calibration of the EcoACC controller and presenting the effects of different adaptive gains on the performance of the controller
4. Presentation of the HIL testing of the proposed intelligent cruise controllers and investigating the implementability of the GMRES-based RTO algorithms in real-time

5.2 Future Work

The recommended future work to expand this research is as follows:

- Add other fast real-time optimizers to the developed code generation tool
- The proposed approach works based on a discrete time-step formulation. Alternatively, a parametrized continuous function approach for calculating the optimal input must be studied and the corresponding results must be compared.
- Investigate the effect of using multiple-shooting method in real-time optimization
- Clearly, the chosen drive cycle had an impact on the proposed NMPC controller. Similarly, the effect of other different drive cycles, traffic and speed limits must be investigated in future.
- Extensive research is needed in the area of robustness and stability analysis of the proposed control strategies
- Extend the evaluation of the proposed intelligent cruise controllers in a more complex traffic system using a traffic simulator
- Perform component-in-the-loop simulations on facilities with several dynamometers

References

[1] A. Zilouchian and M. Jamshidi, *Intelligent control systems using soft computing methodologies*. CRC Press, Inc., 2000.

[2] J. S. Albus, "Outline for a theory of intelligence," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 21, no. 3, pp. 473–509, 1991.

[3] L. Vlacic, M. Parent, and F. Harashima, *Intelligent vehicle technologies: theory and applications*. Butterworth-Heinemann, 2001.

[4] E. F. Camacho and C. B. Alba, *Model predictive control*. Springer Science & Business Media, 2013.

[5] L. Grüne and J. Pannek, *Nonlinear model predictive control*. Springer, 2011.

[6] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. Scokaert, "Constrained model predictive control: Stability and optimality," *Automatica*, vol. 36, no. 6, pp. 789–814, 2000.

[7] J. B. Rawlings and D. Q. Mayne, *Model predictive control: Theory and design*. Nob Hill Pub., 2009.

[8] F. Allgöwer and A. Zheng, *Nonlinear model predictive control*, vol. 26. Birkhäuser, 2012.

[9] K. Yang, Y. Kang, and S. Sukkarieh, "Adaptive nonlinear model predictive path-following control for a fixed-wing unmanned aerial vehicle," *International Journal of Control, Automation and Systems*, vol. 11, no. 1, pp. 65–74, 2013.

[10] K. Yu, M. Mukai, and T. Kawabe, "A battery management system using nonlinear model predictive control for a hybrid electric vehicle," in *Advances in Automotive Control*, vol. 7, pp. 301–306, 2013.

[11] C. E. Beal and J. C. Gerdes, "Model predictive control for vehicle stabilization at the limits of handling," *Control Systems Technology, IEEE Transactions on*, vol. 21, no. 4, pp. 1258–1269, 2013.

[12] P. Shakouri, A. Ordys, and M. R. Askari, "Adaptive cruise control with stop&go function using the state-dependent nonlinear model predictive control approach," *ISA transactions*, vol. 51, no. 5, pp. 622–631, 2012.

[13] M. Huang, H. Nakada, K. Butts, and I. Kolmanovsky, "Nonlinear model predictive control of a diesel engine air path: A comparison of constraint handling and computational strategies," *IFAC-PapersOnLine*, vol. 48, no. 23, pp. 372–379, 2015.

[14] K. I. Kouramas, C. Panos, N. P. Faísca, and E. N. Pistikopoulos, "An algorithm for robust explicit/multi-parametric model predictive control," *Automatica*, vol. 49, no. 2, pp. 381–389, 2013.

[15] S. D. Cairano, A. Bemporad, I. Kolmanovsky, and D. Hrovat, "Model predictive control of magnetically actuated mass spring dampers for automotive applications," *International Journal of Control*, vol. 80, no. 11, pp. 1701–1716, 2007.

[16] A. Taghavipour, N. L. Azad, and J. McPhee, "Real-time predictive control strategy for a plug-in hybrid electric powertrain," *Mechatronics*, vol. 29, pp. 13–27, 2015.

[17] A. Alessio and A. Bemporad, "A survey on explicit model predictive control," in *Nonlinear model predictive control*, pp. 345–369, Springer, 2009.

[18] Y. Wang and S. Boyd, "Fast model predictive control using online optimization," *Control Systems Technology, IEEE Transactions on*, vol. 18, no. 2, pp. 267–278, 2010.

[19] H. Seguchi and T. Ohtsuka, "Nonlinear receding horizon control of an underactuated hovercraft," *International journal of robust and nonlinear control*, vol. 13, no. 3-4, pp. 381–398, 2003.

[20] B. Houska, H. J. Ferreau, and M. Diehl, "An auto-generated real-time iteration algorithm for nonlinear mpc in the microsecond range," *Automatica*, vol. 47, no. 10, pp. 2279–2285, 2011.

[21] Y. Saad and M. H. Schultz, "Gmres: A generalized minimal residual algorithm for solving nonsymmetric linear systems," *SIAM Journal on scientific and statistical computing*, vol. 7, no. 3, pp. 856–869, 1986.

[22] T. Ohtsuka, "Continuation/gmres method for fast algorithm of nonlinear receding horizon control," in *Decision and Control, 2000. Proceedings of the 39th IEEE Conference on*, vol. 1, pp. 766–771, IEEE, 2000.

[23] A. E. Bryson, *Applied optimal control: optimization, estimation and control.* CRC Press, 1975.

[24] J. Betts and I. Kolmanovsky, "Practical methods for optimal control using nonlinear programming," *Applied Mechanics Reviews*, vol. 55, p. 68, 2002.

[25] T. Ohtsuka and H. Fujii, "Stabilized continuation method for solving optimal control problems," *Journal of Guidance, Control, and Dynamics*, vol. 17, no. 5, pp. 950–957, 1994.

[26] D. DeHaan and M. Guay, "A real-time framework for model-predictive control of continuous-time nonlinear systems," *Automatic Control, IEEE Transactions on*, vol. 52, no. 11, pp. 2047–2057, 2007.

[27] M. Diehl, H. G. Bock, and J. P. Schlöder, "A real-time iteration scheme for nonlinear optimization in optimal feedback control," *SIAM Journal on control and optimization*, vol. 43, no. 5, pp. 1714–1736, 2005.

[28] T. Ohtsuka, "A continuation/gmres method for fast computation of nonlinear receding horizon control," *Automatica*, vol. 40, no. 4, pp. 563–574, 2004.

[29] T. Ohtsuka, "A tutorial on c/gmres and automatic code generation for nonlinear model predictive control," in *Control Conference (ECC), 2015 European*, pp. 73–86, IEEE, 2015.

[30] B. Houska, H. J. Ferreau, and M. Diehl, "Acado toolkitan open-source framework for automatic control and dynamic optimization," *Optimal Control Applications and Methods*, vol. 32, no. 3, pp. 298–312, 2011.

[31] A. Emadi, Y. J. Lee, and K. Rajashekara, "Power electronics and motor drives in electric, hybrid electric, and plug-in hybrid electric vehicles," *Industrial Electronics, IEEE Transactions on*, vol. 55, no. 6, pp. 2237–2245, 2008.

[32] S. G. Wirasingha and A. Emadi, "Classification and review of control strategies for plug-in hybrid electric vehicles," *vehicular Technology, IEEE Transactions on*, vol. 60, no. 1, pp. 111–122, 2011.

[33] P. Pisu and G. Rizzoni, "A comparative study of supervisory control strategies for hybrid electric vehicles," *Control Systems Technology, IEEE Transactions on*, vol. 15, no. 3, pp. 506–518, 2007.

[34] S. Stockar, V. Marano, M. Canova, G. Rizzoni, and L. Guzzella, "Energy-optimal control of plug-in hybrid electric vehicles for real-world driving cycles," *Vehicular Technology, IEEE Transactions on*, vol. 60, no. 7, pp. 2949–2962, 2011.

[35] S. J. Moura, H. K. Fathy, D. S. Callaway, and J. L. Stein, "A stochastic optimal control approach for power management in plug-in hybrid electric vehicles," *Control Systems Technology, IEEE Transactions on*, vol. 19, no. 3, pp. 545–555, 2011.

[36] H. Banvait, S. Anwar, and Y. Chen, "A rule-based energy management strategy for plug-in hybrid electric vehicle (phev)," in *American Control Conference, 2009. ACC'09.*, pp. 3938–3943, IEEE, 2009.

[37] A. Taghavipour, N. L. Azad, and J. McPhee, "An optimal power management strategy for power split plug-in hybrid electric vehicles," *International Journal of Vehicle Design*, vol. 60, no. 3/4, pp. 286–304, 2012.

[38] Q. Gong, Y. Li, and Z.-R. Peng, "Trip-based optimal power management of plug-in hybrid electric vehicles," *Vehicular Technology, IEEE Transactions on*, vol. 57, no. 6, pp. 3393–3401, 2008.

[39] E. Hellström, M. Ivarsson, J. Åslund, and L. Nielsen, "Look-ahead control for heavy trucks to minimize trip time and fuel consumption," *Control Engineering Practice*, vol. 17, no. 2, pp. 245–254, 2009.

[40] B. Asadi and A. Vahidi, "Predictive cruise control: Utilizing upcoming traffic signal information for improving fuel economy and reducing trip time," *Control Systems Technology, IEEE Transactions on*, vol. 19, no. 3, pp. 707–714, 2011.

[41] M. Barth and K. Boriboonsomsin, "Energy and emissions impacts of a freeway-based dynamic eco-driving system," *Transportation Research Part D: Transport and Environment*, vol. 14, no. 6, pp. 400–410, 2009.

[42] B. Zhou, J. Cao, X. Zeng, and H. Wu, "Adaptive traffic light control in wireless sensor network-based intelligent transportation system," in *Vehicular technology conference fall (VTC 2010-Fall), 2010 IEEE 72nd*, pp. 1–5, IEEE, 2010.

[43] K. Yu, J. Yang, and D. Yamaguchi, “Model predictive control for hybrid vehicle ecological driving using traffic signal and road slope information,” *Control Theory and Technology*, vol. 13, no. 1, pp. 17–28, 2015.

[44] M. Vajedi and N. L. Azad, “Ecological adaptive cruise controller for plug-in hybrid electric vehicles using nonlinear model predictive control,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 1, pp. 113–122, 2016.

[45] M. A. S. Kamal, M. Mukai, J. Murata, and T. Kawabe, “Ecological vehicle control on roads with up-down slopes,” *Intelligent Transportation Systems, IEEE Transactions on*, vol. 12, no. 3, pp. 783–794, 2011.

[46] K. Yu and J. Yang, “Performance of a nonlinear real-time optimal control system for hevs/phevs during car following,” *Journal of Applied Mathematics*, vol. 2014, 2014.

[47] M. Wang, W. Daamen, S. Hoogendoorn, and B. Van Arem, “Driver assistance systems modeling by model predictive control,” in *2012 15th International IEEE Conference on Intelligent Transportation Systems*, pp. 1543–1548, IEEE, 2012.

[48] M. A. S. Kamal, M. Mukai, J. Murata, and T. Kawabe, “Model predictive control of vehicles on urban roads for improved fuel economy,” *IEEE Transactions on Control Systems Technology*, vol. 21, no. 3, pp. 831–841, 2013.

[49] M. A. S. Kamal, J.-i. Imura, T. Hayakawa, A. Ohata, and K. Aihara, “Smart driving of a vehicle using model predictive control for improving traffic flow,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 15, no. 2, pp. 878–888, 2014.

[50] Q. Gong, Y. Li, and Z.-R. Peng, “Optimal power management of plug-in hev with intelligent transportation system,” in *2007 IEEE/ASME international conference on advanced intelligent mechatronics*, pp. 1–6, IEEE, 2007.

[51] T. Ivens, S. Spronkmans, B. Rosca, and S. Wilkins, “Model-based eco-driving and integrated powertrain control for (hybrid) electric vehicles,” in *Electric Vehicle Symposium and Exhibition (EVS27), 2013 World*, pp. 1–9, IEEE, 2013.

[52] A. Sciarretta and L. Guzzella, “Control of hybrid electric vehicles,” *IEEE Control systems*, vol. 27, no. 2, pp. 60–70, 2007.

[53] X. Huang and J. Wang, “Model predictive regenerative braking control for lightweight electric vehicles with in-wheel motors,” *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering*, vol. 226, no. 9, pp. 1220–1232, 2012.

[54] D. H. Jacobson and M. M. Lele, "A transformation technique for optimal control problems with a state variable inequality constraint," *Automatic Control, IEEE Transactions on*, vol. 14, no. 5, pp. 457–464, 1969.

[55] T. KelleyC, "Iterative methods for linearand nonlinear equations," *RaleighN. C.: NorthCarolinaStateUniversity*, 1995.

[56] S. L. Richter and R. A. DeCarlo, "Continuation methods: Theory and applications," *Systems, Man and Cybernetics, IEEE Transactions on*, no. 4, pp. 459–464, 1983.

[57] A. Taghavipour, R. Masoudi, N. L. Azad, and J. McPhee, "High-fidelity modeling of a power-split plug-in hybrid electric powertrain for control performance evaluation," in *ASME 2013 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, pp. V001T01A008–V001T01A008, American Society of Mechanical Engineers, 2013.

[58] X.-Y. Lu, J. K. Hedrick, and M. Drew, "Acc/cacc-control design, stability and robust performance," in *Proceedings of the 2002 American Control Conference (IEEE Cat. No. CH37301)*, vol. 6, pp. 4327–4332, IEEE, 2002.

[59] C.-Y. Liang and H. Peng, "Optimal adaptive cruise control with guaranteed string stability," *Vehicle System Dynamics*, vol. 32, no. 4-5, pp. 313–330, 1999.

[60] D. Corona and B. De Schutter, "Comparison of a linear and a hybrid adaptive cruise controller for a smart," in *Decision and Control, 2007 46th IEEE Conference on*, pp. 4779–4784, IEEE, 2007.

[61] K. J. Åström and B. Wittenmark, *Adaptive control.* Courier Corporation, 2013.

[62] N. J. Kohut, J. K. Hedrick, and F. Borrelli, "Integrating traffic data and model predictive control to improve fuel economy," *IFAC Proceedings Volumes*, vol. 42, no. 15, pp. 155–160, 2009.

[63] Y. Luo, T. Chen, S. Zhang, and K. Li, "Intelligent hybrid electric vehicle acc with coordinated control of tracking ability, fuel economy, and ride comfort," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 4, pp. 2303–2308, 2015.

[64] S. Onori, L. Serrao, and G. Rizzoni, "Adaptive equivalent consumption minimization strategy for hybrid electric vehicles," in *ASME 2010 dynamic systems and control conference*, pp. 499–505, American Society of Mechanical Engineers, 2010.

[65] C. Ebert and C. Jones, "Embedded software: Facts, figures, and future," *Computer*, vol. 42, no. 4, pp. 0042–52, 2009.

[66] E. L. Allgower and K. Georg, *Numerical continuation methods: an introduction*, vol. 13. Springer Science & Business Media, 2012.