Statistical Learning and Stochastic Process for Robust Predictive Control of Vehicle Suspension Systems

by

Ahmad Mozaffari

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Statistics

Waterloo, Ontario, Canada, 2017

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

Predictive controllers play an important role in today's industry because of their capability of verifying optimum control signals for nonlinear systems in a real-time fashion. Due to their mathematical properties, such controllers are best suited for control problems with constraints. Also, these interesting controllers can be equipped with different types of optimization and learning modules. The main goal of this thesis is to explore the potential of predictive controllers for a challenging automotive problem, known as active vehicle suspension control.

In this context, it is intended to explore both modeling and optimization modules using different statistical methodologies ranging from statistical learning to random process control. Among the variants of predictive controllers, learning-based model predictive controller (LBMPC) is becoming more and more interesting to the researchers of control society due to its structural flexibility and optimal performance. The current investigation will contribute to the improvement of LBMPC by adopting different statistical learning strategies and forecasting methods to improve the efficiency and robustness of learning performed in LBMPC. Also, advanced probabilistic tools such as reinforcement learning, absorbing state stochastic process, graphical modelling, and bootstrapping are used to quantify different sources of uncertainty which can affect the performance of the LBMPC when it is used for vehicle suspension control. Moreover, a comparative study is conducted using gradient-based as well as deterministic and stochastic direct search optimization algorithms for calculating the optimal control commands.

By combining the well-established control and statistical theories, a novel variant of LBMPC is developed which not only affords stability and robustness, but also surpasses a wide range of conventional controllers for the vehicle suspension control problem. The findings of the current investigation can be interesting to the researchers of automotive industry (in particular those interested in automotive control), as several open issues regarding the potential of statistical tools for improving the performance of controllers for vehicle suspension problem are addressed.

Acknowledgements

I would like to thank my supervisor, Professor Shoja'eddin Chenouri for his support and guidance during the preparation of the thesis. Also, I am grateful to Professor Greg Rice and Professor Mu Zhu for providing constructive and valuable comments on my thesis.

Dedication

To my loving family whom without their support it would not be possible for me to do the current research.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Background

Automotive industry can undoubtedly be enumerated as one of the most brilliant achievements of modern society. Due to the market demand and quality control issues, over the past decades, tremendous amount of investment has been done to promote the fundaments of automotive industry. Interestingly, the creative mind of designers has produced so many idealistic schemes which have instigated automotive engineers to seek for advanced technologies to improve the performance of vehicles. The research on designing new efficient vehicles and improving the existing vehicles is abound, and that would definitely not be convenient to categorize them into a certain number of branches. However, as a general view, one can say that the improvements are made on hardware and software devised on vehicles. The main focus of this thesis is on reviewing and analyzing the important progresses and improvements made to automotive software. By using the general term software, we mainly refer to those controlling and fault detection algorithms used either to make the functioning of automobiles possible or to improve the performance of different components of automobiles. To come up with efficient algorithms for improving the performance of vehicles, programmers should use the well-established theories and ideas from the fields of sensing [1], optimization [2], estimation [3], modeling [4], and control [5].

The final goal of any algorithm designer is to develop an algorithmic frame which is much more robust and accurate compared to the exiting techniques or at-least has the same efficiency of the existing methods but with less computational complexity [6]. Nowadays, such concerns are even more pivotal as the performance of modern vehicles mostly relies on algorithms used for monitoring and controlling of vehicle components.

Fortunately, remarkable computational and theoretical developments have made it possible for algorithmic designers to conduct comprehensive simulation, and test the compatibility of their methods with computational resources available in vehicles. The detailed review of the conducted researches will be given later in Chapter 2 so that readers can easily follow the streamline of conducted researches.

As a general remark, it can be stated that the most successful algorithms are those which efficiently automate the data acquisition and decision making processes to make the most optimal control decisions. So, a proficient algorithm designer is the one with a decent knowledge of fields like data mining, machine learning, and control [7]. Such prerequisites are inline with the interest for designing smart vehicles [8] which should simultaneously make efficient inference from the sensed data, and calculate proper controlling commands to steer the vehicle. It is also worth mentioning that the final goal of automotive industry is to gradually elevate the smartness of vehicles to come up with fully-automated vehicles, technically known as autonomous vehicles [9].

The existing traditional and advanced controlling / monitoring algorithms are designed for different objectives, and are mounted into different components of vehicles. To name only a few, controlling / monitoring algorithms can be used for passengers' safety [10], riding comfort [11], tire stability control [12], vehicle body vibration control [13], adaptive cruise control [14], fuel consumption minimization [15], and *etc.* Hence, ongoing research activities mainly concern with designing better controlling techniques to improve the quality of controllers for the abovementioned control objectives. As can be inferred, tremendous amount of investigation has been devoted to each of the abovementioned control fields. However, due to the mentioned reasons, recently, a remarkable trend has been emerged towards designing interdisciplinary controlling algorithms which not only take advantage from well-established control theories to guarantee the robustness and accuracy of the actuation signals, but also can use the incoming sensory information to make proper decisions regarding the vehicle status [16].

Together with the mentioned interest, there is also another trend to seek for a class of real-time optimal controllers which are capable of calculating the control commands based on the future behavior of vehicles. Such class of controllers are known as model predictive controllers (MPCs) which have several modules in their architecture that enables them to calculate optimal actuation signals in a real-time fashion based on the future behavior of vehicles [17]. In the next sub-section, the main motivations of the current thesis are stated, and it is clarified how the thesis is connected to the state-of-the-art.

1.2 Motivation

In this thesis, a comprehensive investigation is conducted to come up with an efficient and robust variant of MPC which can perform both control and decision making in tandem. The main idea, in terms of the algorithmic architecture, has previously been proposed in [18]. The controller which is a variant of the mentioned interdisciplinary controllers is called learning-based model predictive control (LBMPC). So far, a large number of learnable MPCs have been proposed which mainly replace the physics-based control-oriented state-space model with a black-box oracle [19, 20]. However, due to the best knowledge of the author, LBMPC is among the rare existing learnable controllers (if not the only one) which has a firm theoretical foundation, and deals with the uncertainties of the learning model, as well as the measurement noises, and also satisfies the required conditions to ensure the robustness of the controlling commands [21].

In the current investigation, LBMPC is used for coping with one of the most important controlling objectives of vehicles, i.e. suspension system control. The final goal of LBMPC is to guarantee the ride comfort and also decrease the vibration of the suspension system. Through a comprehensive experimental and theoretical investigation, different statistical and black-box learning systems are contrived at the heart of LBMPC to explore the power of its learning modules. Also, in this thesis, some important strides are taken towards introducing a novel algorithm that uses stochastic process and reinforcement learning theory to develop a real-time desired trajectory building mechanism. It is shown that well-established statistical theories are not only beneficial for developing reliable and robust learning modules, but also can play pivotal role for designing a desired trajectory building mechanism which considers the main sources of uncertainty to optimally calculate the desired tracking profile [22].

In spite of the fact that the road roughness is one of the major disturbances affecting the performance of suspension controllers, there exist rare reports in the literature for developing realistic models for its estimation. Indeed, most of the conducted researches make simple assumptions, e.g. considering sinusoid bumps, predefined waves, or Gaussian uncertainties, to simulate the road roughness [23]. This is when the author's own experiments indicate that the road roughness does not follow a Gaussian distribution. Instead, it is usually a non-stationary spatial profile for which the first and second difference orders are mean-stationary. Also, in a remarkable number of researches, the road roughness is estimated by point-estimators without considering any confidence band for the predicted values. Such simulation-based studies result in unsatisfactory outcomes. Hence, in this thesis, an independent chapter is devoted to the detailed explanation of the steps that should be taken to design a data-driven road roughness prediction module. To do so,

experimental data are captured using an accelerometer sensor, and then statistical techniques are adopted to perform data analysis, and finally (after making proper inferences), statistical prediction techniques are used to come up with a practical model for predicting the road roughness. To the best knowledge of the author, this is one of the rare existing reports which tries to develop a data-driven road roughness predictor in time-domain to be used at the heart of suspension controller as road disturbance estimator.

One of the other important aspects considered in the thesis is to explore the computational complexity of LBMPC for suspension control problem. Mainly, it is tried to find out whether the structure of LBMPC is compatible with the computational units used in vehicles. Also, several existing controlling methods alongwith authentic performance evaluation indices are adopted from the literature, and are applied to the same control problem to evaluate the performance of LBMPC.

In general, by performing a throughout simulation and rigorous theoretical analysis, it is tried to fulfill the following objectives:

1. Since the mathematical state-space model used for suspension control is complicated, it is quite difficult to train a surrogate model with acceptable accuracy. Thus, that would be logical to have a considerable modeling error. Using learning modules without considering a mechanism to handle the uncertainty drastically undermines the performance of general learnable MPCs. Here, it is shown that the robustness considerations and the dual-modeling module devised in this specific variant of learnable controller (i.e. LBMPC) is best suited for suspension control problem. The findings of the simulation are organized in the form of a research paper and is going to be submitted to a control engineering journal.

2. By using different well-established statistical techniques and theories, it is indicated that one can come up with a realistic algorithm to predict the road roughness beforehand, and feed it to the state-space model as one of the main sources of disturbance. Also, through experimental data acquisition and data analysis, it is revealed that some common assumptions usually used for simulation-based numerical experiments need much more careful considerations. Due to the fact that the obtained results contribute to the available literature (to the best knwoledge of the author, this is the first time that a statistical analysis is conducted using the real road data for road roughness forecasting), the findings are gathered in the form of a research paper and is submitted for publication to a Journal.

3. A novel desired trajectory building mechanism is developed which uses the concepts of reinforcement learning and stochastic process Markov chains to deal with the

uncertainties in a real-time fashion, and to optimally calculate the desired trajectory. The obtained results are editted in the form of a book chapter and will be published.

4. Some novel constraints and optimization remarks are presented based on the author's experiment which facilitate the implementation of LBMPC, and also ensures the controlling commands are logical.

5. Several controlling and learning modules are tested to find out the most efficient ones for suspension control. Also, it is tried to find out whether LBMPC has the potential of surpassing the other rival controllers. Finally, the advantages and downsides of LBMPC are reported. Due to the importance of this issue and also the interesting findigs, the results are prepared in the form of a research paper and will be submitted to a journal.

1.3 Outline

The thesis is organized in 9 chapters. Chapter 2 is devoted to a detailed review of the existing controllers for suspension control problem. The mathematical formulation of the considered vehicle suspension system is given in Chapter 3. In this chapter, the main nonlinear terms playing part in the vehicle model are presented, and it is discussed how one can come up with an equivalent linearized model to be used in a real-time fashion. In Chapter 4, the steps and procedures required for developing an experimentally derived and provably robust predictor for road roughness are scrutinized. Also, through data analysis, it is indicated which common assumptions for developing road roughness models should be avoided to have an authentic predictor. Chapter 5 is devoted to the detailed description of Markov models for developing desired trajectory builder. The mathematical formulation and corresponding details for the implementation of LBMPC are given in Chapter 6. Chapter 7 encompasses the detailed studies regarding the selection and evaluation of statistical and intelligent oracles as a part of LBMPC. Some recommendations on using proper oracles are given based on the conducted experiments. The comparative simulation results are presented in Chapter 8. In this chapter, the pros and cons of LBMPC are given based on the results. Finally, the thesis is concluded in Chapter 9. Moreover, some outlines are given that deserve further investigation to reveal the potentials of LBMPC for advanced applications. Also, it is tried to explain why LBMPC can be a good choice for some of the emerging applications.

Chapter 2

Literature Review

In this chapter, a detailed review of the main streamline of researches on designing controllers and learning algorithms for vehicle suspension system is provided, and the main findings are reported. Another objective of this chapter is to categorize the conducted researches into a certain classes so that the readers can find out which issues are important topics for vehicle suspension control. It is worth pointing out that in the literature, the term active suspension control is used to represent designing controllers for vehicle suspension systems. The controllers can be designed to satisfy versatile objectives such as road holding, vehicle stability, passenger comfort, regenerating power, and *etc.*

For a long period of time, engineers have focused on using well-established control theories to satisfy different control objectives within the context of optimal and robust vehicle suspension control [24]. Although traditional controllers show very promising results at the simulation level, they usually encounter functioning problems when implemented as hardware-in-the-loop controllers. For most of such controllers, the problem was that despite of their computational and theoretically proven power, there were no decision making / data processing modules to do perception and update the control strategies to produce proper actuation signals. This has been a remarkable motivation for automotive control engineers to switch to using more complicated controlling algorithms which are capable of performing perception and control simultaneously. In line with such interest, machine learning and decision making tools have extensively been explored, and several useful techniques from data mining have been fused with controlling modules [25]. By the introduction of learnable control algorithms, new challenges have emerged. Obviously, complicated controlling algorithms consume a considerable computational power which is not a good feature from automotive control prospective in which the available computational processors are not so powerful [26]. Hence, different statistical and sensing theorems

have been adopted to somehow reduce the computational complexity of complicated suspension controllers. In what follows this chapter, it is tried to review some of the most promising traditional and recent active controllers which have been developed to satisfy different control objectives for vehicle suspension control.

Prior to proceeding with the literature review, it is worth mentioning that other than active suspension controllers, there exist another variant of controllers which fall within the category of semi-active suspension control. Semi-active controllers such as sky hook controller (SHC), ground hook controller (GHC) and magnetorheological damper (MRD) [27, 28] can only change the viscous damping coefficient of the shock absorber, and are not capable of adding any external energy in the form of actuation force to suspension system. Such simple heuristically designed controllers use a small number of rules to change the damping coefficient on road, and have been proven to be practical choices given their relatively cheap implementation. In the light of promising feedbacks on the applicability of semi-active controllers, researchers have put a considerable effort to further improve their performance, in particular by incorporating the power of well-established offline controlling schemes. In [29], a modified SHC was proposed in which the parameters were optimally verified via equating the control force of the feedback system to that obtained by linear quadratic regulator (LQR). The simulation using Monte Carlo indicated that the resulting controller is as good as other well-known optimal controllers. In [30], a model-free control structure for the online tuning of semi-active suspension of a passenger car is proposed. The controller was developed based on some simple physical insights from the vehicle and semi-active suspension dynamics beneficial for linearizing and decoupling the system, and decentralized linear feedback. The most interesting asset of the controller is its independency from any model for calculating the controller which reduces its computational complexity. In [31], a comparative analysis was conducted using two types of heuristic semi-active controllers, i.e. a global suspension controller and a semi-active controller which used four independent controllers for each quarter of vehicle. The simulation indicated that both of the considered semi-active controllers have the same performance accuracy; however, more hardware including measuring and actuation devices should be used for the one with four independent controllers. In [32], a novel adaptive semi-active suspension control scheme was proposed which used a linear time varying model to estimate any change in the mass of the vehicle body (mainly due to any change in the number of passengers). To do so, a gain scheduled parameterization to adaptively capture the nonlinear behavior of dampers as well as the variation of sprung mass was implemented. The results indicated that this adaptive semi-active controller can show promising results, and its performance surpasses those controllers considering a fixed nominal model for suspension control. In [33], an interesting open loop variational feedback controller (VFC) was proposed based on variational

optimal control theory for the semi-active control of suspension system. The proposed optimal semi-active controller was compared to different well-known semi-active controllers such as SHC and GHC. The simulation results indicated that VFC based controller enjoys a higher generalization and is more appropriate for practical usage. In [34], an optimal semi-active preview control law was formulated for a four-degree of freedom half-car model with constant velocity. The model includes a hysteretic nonlinear suspension spring which was linearized for deriving the control law. The road roughness was also considered as a stochastic signal which was simulated by means of Monte Carlo. The simulation indicated that the control law can be viewed as a promising scheme for suspension control. In [35], the control synthesis was proposed for a class of semi-active suspension systems with norm-bounded parameter uncertainties, time-varying input delays and actuator saturation. The existence conditions of the desired state-feedback controller were proven with the aid of linear matrix inequality (LMI) and delay-range-dependent Lyapunov function and also exploring the property of the underlying saturation nonlinearity. Simulation indicated the usefulness and advantageous performance of the developed theoretical model.

As can be inferred, researchers are vividly trying to develop novel semi-active controllers. However, most of such controllers are functioning based on a predefined logic (heuristic or optimal) to regulate the damping coefficient, and may not be completely authentic for practical uses. Therefore, the majority of modern vehicles are equipped with active controllers which are able to estimate the states of suspension systems in a real-time fashion, and produce actuation forces using external energy production systems to satisfy predefined control objectives. In some senses, research activities on active suspension control have attracted considerable attentions after the publication of seminal research papers by Balzer [36], Yoshimura and Ananthanarayana [37], and Alleyne and Hedrick [38]. In [36], a theoretical optimal control strategy resulting from the concept of partial preview of disturbance with generalized rate penalties and measurements is solved and used for vehicle suspension control. In [37], a relatively same idea has been used for designing an optimal stochastic controller for suspension vibration control with preview on an irregular surface. In this way, the controller has been fused with a Kalman filter to estimate the dynamic behavior of suspension system on irregular surfaces. By considering a numerical example, it was endorsed that the controller can be very effective for active control of suspension systems with preview. In [38], an adaptive nonlinear sliding control law was formulated to control an electrohydraulic suspension system. The primary goal of this nonlinear controller was to guarantee the robustness in the presence of different types of uncertainties and disturbances. A comparative study was also carried out considering the standard sliding mode controller (SMC) as well as the conventional passive suspension system. Through simulation, it was shown that the proposed active controlling scheme

can remarkably improve the performance of passive suspension system. At the end of 20^{th} century, a comprehensive review has been published which precisely investigated the state-of-the-art of active suspension control [39]. Several clues and future research outlines had been proposed which have been later used for technical progresses in the field and have served as a motivation for an exhaustive research on designing advanced active controllers for vehicle suspension systems.

Following the streamline of conducted researches, in the 21^{st} century, traditional/classical theories within the realm of optimal, adaptive, robust, hybrid switching and nonlinear control have come to the aid of practitioners for designing active suspension controllers. In [40], a LQR was proposed for the active control of vehicle suspension. In this research, LQR control strategy was used for the optimal calculation of external force to regulate the vibration of vehicle suspension system. By comparing the performance of the resulting active suspension to the passive system, the efficacy of the proposed controller was verified. In [41], a multi-objective synthesis framework which corporates the generalized L_2-norm, H_2-norm, and generalized H_2-norm were implemented in such a way that the poles of the closed-loop system were constrained within the sub-region of the left-half of s-plane. Based on the simulation, it was observed that the proposed multiobjective controller retains a proper trade-off between vehicle ride comfort and suspension travel objectives. In [42], a novel structural condition on the controller was derived that guaranteed appropriate disturbance response decoupling and achievable performance with application to vehicle active suspension control. By numerical simulation on a half-car model, it was observed that the proposed theorems on disturbance response decoupling can be used for standard quarter and half-car models with various different choices of measurements. In [43], an innovative nonlinear backstepping active suspension controller was proposed for half-car suspension model. The main goal of the proposed controller was to improve the inherent trade-off between ride quality and suspension travel. The numerical results indicated that the proposed controller could effectively guarantee the improvement of trade-off, and thus, is appropriate for multitask control of suspension system. In [44], a throughout analysis of various principles of suspensions with variable dampers and springs as well as active components was carried out, and several existing models for suspension systems were discussed. Based on the models, different variants of controllers such as adaptive, feedback based, and *etc.* were presented, and their applicability to advanced tasks such as fault diagnosis and sensor fault detection were scrutinized. It was shown that the model-based fault detection of vehicle suspension is possible by the classification of symptoms generated by parameter estimation and parity equations. In [45], a robust non-chattering SMC for a full-vehicle without suspension gap loss was developed. By considering a seven degree of freedom model and a periodic surface, the efficacy of the proposed controller, and its resis-

tance against unmeasured noises were demonstrated. In [46], a switching robust controller comprised of three independent control strategies were developed to satisfy three different objectives, namely road holding ability, passenger comfort, and suspension deflection. It was shown that by considering a control switching logic, one can improve the trade-off capability of the implemented multi-task suspension control scheme. In [47], a load dependent controller was designed by means of linear matrix inequality (LMI) to solve the problem of multiobjective control for vehicle active suspension system. A quarter-car model with active suspension was considered to test the performance of the proposed controller. In the formulation of this controller which was based on a parameter-dependent Lyapunov function, the gain matrix was updated using the online available information about the body mass. The simulation results indicated that the proposed controller could yield much less conservative results compared to a number of rival robust controllers formulated within the quadratic frame. In [48], an active suspension controller was proposed which used an adaptive filter (called William's filter) for decoupling vehicle response to load and inertial disturbance from the vehicle response to road disturbance. One of the other advantages of devised filter was its independency from the values of vehicle parameter. The efficacy of the proposed controller was examined on a model with the abrupt variation of parameters such as vehicle load. The results elaborated the power of the proposed method for adaptive suspension control. In [49], a novel impedance controller was proposed to control the dynamic behavior of a vehicle subjected to road disturbance. The role of the impedance rule was to achieve passenger comfort and vehicle handling using a hydraulically actuated suspension system. The salient asset of the proposed controller was its model-free property as well as its applicability to a broad range of suspension systems. In [50], a H_∞ controller was designed for active vehicle suspension control. The acceptable robustness of the proposed controller against different sources of uncertainties were studied through simulation. In [51], a gain-scheduled H_∞ controller was developed for active suspension control considering two different objectives, i.e. attitude and handling improvement. Simulation on a complex nonlinear full-vehicle model as well as on experimental data coming from a real model demonstrated the efficacy of the proposed controller. In [52], a generalized predictive control (GPC) algorithm was proposed for embedded active control of vehicle suspension system. The software-only simulation indicated that the proposed GPC could speed-up the calculation of optimal control command compared to rival GPCs with alternative parameter tuning strategies while satisfying tight constraints on the actuation force and states. In [53], a H_∞ controller was proposed for robust control of suspension systems while taking the uncertainty of the model parameter into account. The simulation results indicated that the proposed model can efficiently cope with the undesired effects of disturbances while calculating optimal actuation commands for active suspension system control. In [54], a novel stochastic optimal control law was formulated for stochastic

active control of a half-car nonlinear suspension with random road roughness profile. The randomness of the road roughness was simulated by considering a first order filter with Gaussian white noise. Simulation based on Monte Carlo indicated the applicability of the proposed optimal controller for suspension system being exposed to stochastic road profile. In [55], an innovative robust optimal controller was proposed using Pontryagin's minimum principle, Lyapunov theory and affine quadratic stability to improve vehicle ride comfort and handling performance. The results of the carried out simulation indicated the withstanding property of the controller against road bump disturbance. In [56], an active vehicle suspension controller with input output feedback linearization was proposed considering actuator delay. Stability proofs for the zero-dynamics and the closed-loop system were also presented. Based on numerical results, it was shown that the proposed controller is efficient for suspension control. In [57], a multiplexed model predictive control (MMPC) was proposed for active suspension control. The optimization problem in MMPC was formulated such that each control input was optimized independently, and thereafter, a cyclic process was repeated to improve the optimality of the control command. The simulation results demonstrated that the proposed method possesses less computational complexity compared to standard model predictive control (MPC) which solved a multivariate optimization problem at each updating stage. In [58], a predictive control strategy was proposed based on dynamic matrix control (DMC) and variable structure control (VSC) for the active control of suspension systems. By a throughout comparative study considering a large number of well-established controllers and periodic / non-periodic disturbances, it was shown that the proposed controller has acceptable control power. Also, theoretical results from VSC side indicated that the method can achieve the zero-dynamics provided that the uncertainty of suspension system model is bounded.

Fortunately, research on using standard / classical control theories for vehicle suspension control is ongoing and more and more algorithms are being invented to handle the mentioned task. However, notwithstanding the obvious advantages of such techniques, they have some drawbacks which needs further investigation and thoughts. One can strongly claim that standard control algorithms are not smart enough to make inference in a real-time fashion. At most, such techniques follow a pre-determined controlling rule (both stochastic and deterministic versions) to calculate the actuation command. As discussed previously, the fast evolution of computational hardware and software has made it feasible for engineers to take ambitious steps towards designing autonomous systems. One of the most pragmatic and initial stages to fulfill this goal is the adoption of intelligent knowledge based methods and using them at the heart of control algorithms. In this fashion, intelligent techniques can conduct data-driven inference and share their findings with controllers (stochastic and deterministic) to come up with data-driven control commands.

The abovementioned idea has been pursued for designing vehicle suspension controllers, and in particular, computational intelligence techniques such as neural networks and fuzzy models have been used to satisfy the mentioned objectives. Here, we will report some of the most remarkable findings pertaining to the implementation of intelligent controllers for vehicle suspension control. In [59], an intelligent optimization technique called genetic algorithm (GA) was adopted and hybridized with LMI for optimal design of static output feedback and non-fragile output feedback H_∞ controller for active vehicle suspension control. The proposed intelligent robust controller was examined using a quarter-car model with active suspension. The simulation results indicated that the resulting controller significantly improved the non-fragility characteristics over controller gain variations. In [60], two different variants of intelligent controllers, namely fuzzy controller and adaptive fuzzy controller were proposed for active vehicle suspension control. The control objective was to achieve the road holding and riding comfort over a wide range of road profiles. Numerical simulation were conducted to demonstrate the efficacy of the proposed controllers. By comparing the intelligent controllers with LQR, it was observed that intelligent controllers have superior performance. In [61], GA was utilized to optimize a complex design task involving the control of electric damper and model parameters such that the power dissipation of the electric damper be minimized while maintaining acceptable comfort and road-holding capabilities. The results of the simulation endorsed the effectiveness of GA meta-optimization of the architecture of the considered controller. In [62], a fuzzy controller was proposed based on the vehicle dynamics and control theory to control a six degree of freedom suspension system subjected to irregular excitation from road surface. The simulation results indicated the acceptable performance of the proposed controller for vehicle suspension control. Also, an innovative simulation frame comprising of Matlab/Simulink and ADAMS was developed for the visualization of the performance of controller. In [63], a neural network-based MPC (NNMPC) was developed for controlling a servo-hydraulic vehicle suspension system. The proposed intelligent controller was applied to a two degree of freedom quarter car servo hydraulic vehicle suspension system. Neural network made it possible to model the nonlinear dynamics of the servo-electric actuator. The proposed nonlinear optimal controller was compared to proportional-integral-derivative (PID) controller tuned by Ziegler-Nichols method. The simulation results unveiled the superior performance of NNMPC over PID with respect to different performance indices such as adaptation to deterministic road disturbance. In [64], a novel intelligent controller was developed to control the dynamic behavior of vehicle suspension system subjected to road disturbance. The proposed intelligent controller includes a model-free impedance control law with two interior loops that were force control of the actuator by feedback linearization and fuzzy control loop to track a desired body displacement. The system stability was analyzed to ensure the controller does not experience any sort of collapse on road. The

proposed controller was applied to a nonlinear suspension system with a nonlinear model of hydraulic actuator. The simulation results indicated that the proposed model-free controller is as good as model-based optimal control strategies, and also can be applied to a broad range of road conditions. In [65], GA was adopted for multiobjective optimization of semi-active LPV suspension controller which used H_∞ control law for comfort and road holding. The simulation indicated that the Pareto front obtained by GA could give the designer versatility to find a trade-off between potential optimal solutions. Also, from a control prospective, it was observed that the proposed intelligent controller can be viewed as an effective technique for suspension control. In [66], an adaptive impedance controller was designed using particle swarm optimization (PSO) for the vibration control of a hydraulic suspension system. The inner loop of the controller was a force controller and the outer loop was a robust model reference adaptive controller. The outer loop enabled the controller to handle different sources of uncertainties. By simulating the proposed intelligent controller on a quarter-car model in the presence of disturbance, the effectiveness of the control law was proven. In [67], a novel inverse adaptive neuro-fuzzy inference system (ANFIS) based MR damper hybridized with linear quadratic Gaussian (LQG) controller was developed for the vibration control of vehicle suspension system. Simulation demonstrated that the resulting controller can accurately track the desired force using ANFIS, and, at the same time, LQG could show competitive performance compared to rival active controllers. In [68], a fuzzy logic controller combined with PID was implemented for the modeling and control of a nonlinear half-car suspension system with quadratic tire stiffness, cubic suspension stiffness, and coulomb friction. By comparing the proposed controller with standard PID and fuzzy controller, it was observed that the hybrid controller with coupled rules could show promising results. In [69], a robust ANFIS-based controller was proposed for the vibration control of a vehicle active suspension system. A comparative study considering PID and NN-based controller was carried out to evaluate the power of the proposed control scheme. The simulation results indicated that the proposed controller can outperform the rival techniques in terms of travelling comfort when the suspension system is subjected to random disturbance. In [70], an intelligent semi-active controller with SCH and ANFIS with Gaussian membership functions and back-propagation learning was proposed for controlling a quarter-car suspension system with semi-active MR damper. By comparing the performance of the proposed semi-active controller with SHC and GHC as well as passive system, the efficacy of the proposed intelligent controller was shown. In [71], a concise review of the well-known suspension control methods including both classical and intelligent versions was conducted. In that report, the main advantages of using intelligent controllers for vehicle suspension system control were explained, and it was justified that computational intelligence can serve as a good mean for vehicle suspension control. In [72], an intelligent semi-active controller including a fuzzy logic based SHC and ANFIS model

was proposed for the control of vehicle suspension system in order to improve passenger comfort and road holding, and also to stabilize the vehicle movements. The simulation results indicated that the proposed intelligent controller provides a better isolation performance compared to passive suspension system. In [73], an intelligent controller was proposed which used simulated annealing (SA) for the optimization of the weight matrix of LQR controller. The performance of the intelligent controller was tested against standard LQR and passive suspension system. It was observed that GA-LQR can outperform the standard LQR. In [74], an adaptive controller with an augmented NN was proposed for the active control of uncertain suspension system with prescribed performance. An efficient optimization algorithm was also considered to estimate the unknown weights of NN, and also to estimate the sprung mass in a real-time fashion. Simulation results indicated the effectiveness of the proposed controller for uncertain suspension control. In [75], an intelligent multiobjective optimization method called, non-dominated sorting genetic algorithm (NSGA-II), was adopted to optimize a hybrid electromagnetic suspension system for ride comfort, regenerated power and road holding. The proposed technique was applied to a two degree-of-freedom quarter-car model. The results of the simulation indicated an improvement with respect to the mentioned control objectives. In [76], a robust and adaptive intelligent controller was proposed which comprised of H_∞ controller, GA and wavelet-based support vector machine (SVM) for nonlinear suspension control with time delay actuation signal. Also, a mixture of wavelet and radial-basis functions (RBFs) were considered to design another intelligent controller. Time varying sprung and unsprung masses and the suspension performances with actuator delay were considered to construct the model. Simulation results demonstrated the robustness of the proposed intelligent controller against any variation on sprung mass in the presence of actuator time delay.

The abovementioned intelligent controllers were mostly combined with existing soft intelligent tools without performing a firm statistical analysis or having a deep insight into their structural properties. As far as the author is concerned, more elaborated studies should be conducted to come up with theoretically cherished data-mining and modeling methods to be used at the heart of controllers. Such a philosophy is called learning-based control, which is a general term which comprises techniques from soft computing and statistical computing. The main contribution of the thesis is to perform a deep investigation into the existing theories and techniques mostly raised from statistics to answer some important questions for designing smart controlling algorithms [77]. It is a fact that to come up with a proper data-driven learning module, several important design questions concerning the probabilistic behavior of the signal of interest, sampling from objects / original signals, sparse representations, reduction of the undesired effects of over-fitting and

under-fitting, and balancing the trade-off between variance and bias of the model should be answered. Obviously, such important issues cannot be handled simply by extracting a model (either soft or statistical) and fit it to a set of captured data. What is actually important here is to conduct some theoretical analysis and find out how to design a model. In [77], most of the abovementioned issues was neatly discussed, and several solutions to solve them were presented. Also, in general studies, statistical learning theories have been demonstrated to be very beneficial for designing controllers. In [78], the potentials of randomization algorithms and well-established statistical learning theories were explored for robust control synthesis. It was theoretically indicated that several problems such as robust stabilization and weighted H_2/H_∞ minimization can be well treated using the proposed statistical method. In [79], the capabilities of statistical theories for designing a learning paradigm for humanoid robots were investigated. In this way, techniques such as locally weighted projection regression were used for the online learning of inverse dynamics models for model-based control, the learning of inverse kinematics of redundant manipulators, and the learning of oculomotor reflexes. The numerical simulation indicated that the learning method has a fast convergence rate with a highly accurate performance. In [80], a novel statistical learning controller was developed for the energy management of a fuel cell electric vehicle. The learning module comprised of a bank of neural networks designed using different statistical learning theories. The simulation results comparing the proposed method with a learning controller with a single NN indicated the superiority of the proposed statistical learning algorithm with a high confidence level. It was also probabilistically indicated that the statistical learning strategy almost surely converged to an optimal solution. In [81], a statistical machine learning method was proposed to make the automatic control practical for internet datacenters. It was argued that common shortcomings can be handled using statistical analysis and modeling theories for automatic controller design. In [82], a probabilistic direct statistical learning algorithm was proposed for designing a learning-based controller for robots. Also, it was argued that how such techniques can be used to improve the manoeuvrability of robots for practical applications. It was argued that in near future, statistical theories will play pivotal role in improving the performance of robots. In [83], the theoretical foundation of several statistical algorithms have been established to be used for designing controllers for uncertain systems. The simulation results indicated that the resulting techniques can reduce the computational complexity of some optimal control strategies by turning them to decision making problems. Based on comprehensive probabilistic analysis, it was indicated that such decision making based statistical controller can find approximated solutions with high probability. In [84], a statistical learning based technique was proposed for the optimal control of hybrid systems. Based on the performed simulation, it was indicated that the optimization problem which is treated from a statistical viewpoint is near global optimum. In [85], a provably stable

learning adaptive controller was formulated with the aid of statistical learning techniques. Also, a nonparametric regression technique was adopted to approximate the unknown dynamical system, and it was shown that the resulting tracking controller has an acceptable convergence speed.

In spite of obvious progresses made in the field of statistical computing and the existence of versatile theoretical results, so far, they have not been given full consideration for designing active suspension controllers. Therefore, the author tries to take strides towards designing a learning based controller to find out whether such methodologies are promising and deserve a thorough investigation.

Chapter 3

Suspension System Control Problem

In this chapter, a detailed investigation into the common existing suspension system models is carried out. Also, a nonlinear suspension system model is implemented for the current simulation, and its equivalent linearized version is determined to be used at the heart of learning-based model predictive controller (LBMPC). In general, the society of control engineering tries to adopt one of the well-known mathematical models (including the one formulated in this chapter) to design controlling algorithms for suspension system control. However, the fact that mathematical models are just approximations of real suspension systems, and several unknown factors and disturbances can affect the performance of controllers is neglected. Therefore, in an attempt to come up with a more realistic model as well as practical conditions for designing a controlling algorithm, in this chapter, different sources of uncertainties which can affect the performance of controller and the precision of mathematical model are taken into account, and proper strategies / arguments are presented for the realistic calibration of the considered model.

3.1 Literature Review of Commonly Used Models

In this section, a concise and consistent review concerning the progress of the mathematical modeling of suspension systems is provided. The literature on the modeling, analyzing and designing of surrogate models for vehicle suspension dynamics has been expanding significantly, thanks to the availability of computational resources and mathematical tools. There are several standard archived research papers and terminologies released by Society of Automotive Engineers (SAE) which endorse on this claim. Fortunately, all of the researchers working within the realm of vehicle dynamics analysis and modeling use the

same standard terminologies (suggested by SAE), and this helps them to conveniently share their information and conduct breakthrough progress in recent decades. Having said that although the most of the conducted researches and analysis as well as standard terminologies pertain to passenger vehicles, their results can be easily extended to heavy vehicles, articulated vehicles, and *etc.* [86].

By using simplified models of vehicle suspension systems, the main goal is to assess the potential effects of vibration on human whole-body health, comfort and perception. At the same time, some engineering / economical goals such as reducing the cost of final product, energy harvesting and *etc.* may be pursued. In a very near future, the increasing demands on improving vehicle dynamics and stability along with the extensive production of hybrid and electric vehicles might make the mathematical modeling of suspension systems quite challenging. This implies the importance of paying attention to mathematical modeling of vehicle systems (in particular the suspension system and tires) and searching for the most reliable mathematical models for designing controlling algorithms, decision making systems, or learning paradigms. It is also worth mentioning that despite of the significant progress on active suspension design and model-based analysis, most of the vehicles on road still use traditional passive suspension systems. This is mainly due to the expense of equipping vehicles with model-based controllers and decision making algorithms. However, the tight governmental regulations and market demand have inclined the automotive industry towards upgrading vehicles' suspension systems, and use the findings of automotive research and development societies.

From the abovementioned information, one can realize the role of mathematical model in the modification and precise analysis of vehicle dynamics and stability on roads. Therefore, here, some of the most important archived research papers concerning the mathematical modeling of vehicle suspension systems are briefly reviewed, which will be then followed by giving some more details about real suspension models.

In a pioneering research [87], a through discussion and analysis was conducted to unveil some new aspects of vehicle dynamics. The accuracy of the results suggested in the paper further endorsed on the veracity of using mathematical models for analyzing the performance of vehicles. In [88], a through investigation was carried out to analyze the computational aspects of different mathematical models of vehicle dynamics, and also to find out their accuracy / robustness to be used for practical applications. By means of a model-based analysis, the author studied the effect of suspension system on the safety and performance of vehicles. In [89], a thorough investigation was carried out for the model-based analysis of heavy vehicle ride dynamics. Based on the simulation, it was revealed how important a mathematical model can be for the accurate analysis of heavy vehicles. In [90], a novel mathematical model was developed for the simulation of the suspension

system of a vehicle with pneumatic tire modelling. Also, the mathematical modeling of vehicle dynamic has been extended to some advanced applications, such as the simulation and measurement of human perception as a function of vehicle vibration. In [91], a review of existing mathematical models for the measurement, assessment and evaluation of the human perception of vehicle vibration was provided. It was argued that mathematical models are pivotal for such an analysis, and provided that the models be designed in a reliable fashion, their feedbacks can be used for real-life applications. In [92], the compatibility of safety-based performance measures with vehicle dynamic mathematical models including suspension system was studied for the development of safety-oriented vehicles. Some theoretical extensions to traditional active suspension mathematical models were proposed in [93]. In [94], a comprehensive review was conducted to discuss the role of mathematical suspension system modeling for analyzing the effect of heavy commercial vehicles on pavement loading. Also, some practical examples of vehicles were provided to endorse the efficacy of mathematical models for reliable analysis. A relatively same analysis was conducted in [95] to elaborate the potential of mathematical modeling for heavy vehicles such as trucks. In [96], model-based analysis was taken into account for a generic design procedure of pneumatic and air suspension systems. In an innovative research [97], a multiobjective optimization problem was coupled to a mathematical model for the efficient and robust design of suspension systems with regard to a set of conflicting objective functions. In [98], a comprehensive review of various advanced mathematical models of chassis system was carried out for commercial vehicles aiming at enhancing vehicles traffic safety. The derived conclusions indicated that a model-based analysis can afford very promising outcome for guaranteeing the safety of vehicles.

As seen, the model-based analysis of vehicle dynamics in particular suspension system has come to the aid of researchers to improve the performance of vehicles with respect to different objectives. Almost all of the abovementioned research reports have conceded that without having a robust, computationally efficient and accurate model, it would be impossible to ensure the simulation results can be used in practice. Indeed, the selection of the best fitted mathematical model is problem / case specific and depends on the type of vehicle as well as the objective of analysis. Suspension systems vary in configuration and type of components used for operation, and thus, that would be important to realize the type of suspension architecture / components when selecting a model for analysis, design, and control.

Indeed, providing a detailed review of the types of existing suspension system architectures and actuators is beyond the scope of this research (and somehow impossible due to versatility of such systems). However, there are some certain types of suspension systems which are commonly used by researchers for vehicle dynamics analysis. Here, we would like

to name some of the most important variants which have been used by automotive research community and consequently have played tremendous role in improving the performance of suspension systems. Keeping in mind that there are also an independent through literature dedicated to the detailed high-fidelity modeling of vehicle tire (e.g. Magic model and LuGre friction model) which will not be reviewed here.

In general, before starting to excavate the existing mathematical models for analysis, that would be a wise choice to ensure which type of vehicle is going to be analyzed. This aids one to select the optimum configuration of suspension system which is the prerequisite of selecting a mathematical model. As an example, it is well-known that (1) hydrogass slow-active suspension systems can be a logical option for racing cars, some passenger cars, public service vehicles and off-road vehicles, (2) Lotus electro-hydraulic active suspension system has proven its potential to be used for racing cars and passenger cars, and (3) vehicles such as Leyland Trucks are compatible with the suspension systems using pneumatic actuators in their architecture [99]. Also, it is obvious that when analyzing the vehicle dynamics and suspension systems of non-traditional vehicles such as electric vehicles and hybrid electric vehicles, attention should be paid to the specific configuration of such systems. This is because, due to their particular configuration, the results coming from standard model-based analysis are unreliable for such vehicles.

In [100], three types of suspension systems were proposed, and their advantages for real-life implementation were studied. The considered systems were (1) suspension system with oil damper mounted in parallel with a compression helical spring, (2) suspension system with colloidal damper without attached compression helical spring, and (3) colloidal damper mounted in parallel with a compression helical spring. For the first suspension system, a Kelvin-Voigt model was presented which included a dashpot and an elastic element connected in parallel. For the second system, a Maxwell model with a dashpot and an elastic element connected in series was formulated. For the third suspension system, a standard linear model consisted of Maxwell unit connected in parallel with an elastic element was considered. The efficacy of the mathematical models and the suspension systems were studied independently by the calculation of vibration transmission from the rough road to vehicle body, considering the constraint that damping coefficient varies with respect to the excitation frequency. It was numerically indicated that with the aid of the presented mathematical models, that would be possible to minimize the vibration transmissibility and consequently maximize the vehicle's ride comfort. In [101], a review of common actuators used in active suspension design was conducted, and some of their physical properties were analyzed by means of mathematical models. It was reported that, among the existing actuators, Oleo-pneumatic actuators, hydraulic actuators, magnetic actuators and electromagnetic actuators have attracted the attention of industrialists and

are widely used in todays' products.

Also, there exist other types of suspension systems which are capable of energy harvesting. Such suspension systems are often called energy-regenerative suspension systems. The main reason behind using such suspension systems is to harvest the dissipated energy due to the vibration and convert it to regenerative energy to improve the vehicle fuel efficiency. The harvested energy can be used for the improvement of suspension performance, for power vehicle electronics, and also for increasing the vehicle fuel efficiency. In general two types of such suspension systems have gained value for automotive industrialists which are mechanical regenerative systems and electromagnetic regenerative systems. Mechanical regenerative suspension systems are those using hydraulic and pneumatic suspensions. Due to their considerable weight, their complexity and the possibility for leaks and raptures, they are scarcely used in the architecture of advanced modern vehicles [102]. On the other hand, electromagnetic regenerative suspension systems are becoming popular in market. The most important variants of such systems are: direct-drive electromagnetic suspension, ball screw electromagnetic suspension, rack-pinion electromagnetic suspension, planetary gear electromagnetic suspension, hydraulic transmission electromagnetic suspension, and self-powered magnetorheological suspension.

In the current research, it is intended to study the suspension system of a traditional passenger vehicle, which is one of the most important products in todays' automotive markets. Also, since it can somehow be viewed as a base suspension system, the results of analysis performed in this thesis can be extended to the other variants of suspension models, such as those stated previously.

Typically, the existing models for vehicle suspension systems have been implemented for time-domain and frequency domain analysis [103]. Traditionally, frequency-domain analysis has been performed for the active control of suspension systems. However, due the fact that most of the modern controlling algorithms operate based on time-domain state space models, mathematical models formulated in time-domain have been becoming more and more popular. Arguably, there exist a limited number of well-consensused time-domain mathematical models available in the literature, which have been used for analyzing the suspension systems of standard passenger cars. These models are quarter-car [104], half-car [58], and full-car models [105]. The main difference of the models used in the literature is the values of model parameters which fit the mentioned mathematical models to the suspension system of particular vehicles, e.g. different models of Toyota, Ford, BMW, and *etc.* Also, another difference which may exist in the modeling of suspension systems is the type of spring and dampers used to come up with the mathematical model. Some researchers have advocated using simple linear state-space models with linear dampers, linear springs and linear model of tire, while some other researchers have fostered the

introduction of a certain amount of nonlinearity to suspension system, typically by using nonlinear springs, nonlinear dampers, asymmetric damping coefficients and modeling tires by means of damping and spring systems.

For the current simulation, based on trial and error, and given the recommendations of the abovementioned papers as well as the author's own assessment [58], a half-car model with nonlinear damper, nonlinear spring as well as the spring and damper model of tires is used for analyzing the vibration of suspension systems. Also, a piece-wise linear representation of the nonlinear model is presented which will be then used at the heart of LBMPC for the calculation of optimal controlling commands.

It is worth pointing out that a number of performance metrics should be formulated together with a mathematical model for the calculation of optimal actuation signals from the actuator devised in active suspension systems. The most commonly used variants of such performance metrics are (1) suspension displacement constraint, (2) road holding metric, and (3) ride comfort index.

The detailed implementation of the mathematical model used here is given in the next section. Also, the performance metrics used in this investigation is given later when formulating the controlling law of LBMPC.

3.2 Mathematical Modeling of Suspension System

For our simulation, a 4 degree of freedom (4-DOF) half-car suspension system with nonlinear damping and nonlinear stiffness is considered. A schematic illustration of the 4-DOF active suspension system is given in Figure 3.1. The notations and symbols used in Figure 3.1 are defined one-by-one to make the interpretation of the mathematical model given later possible. In the figure, M_b represents the body (sprung) mass, J_b represents the body mass moment of inertia, m_{u_1} and m_{u_2} represent the front and rear unsprung masses, c_{s_1} and c_{s_2} are the front and rear suspension nonlinear damping coefficients, k_{s_1} and k_{s_2} are the front and rear suspension nonlinear stiffness coefficients, F_{a_1} and F_{a_2} are the front and rear actuation forces which should be inflicted on suspension system by active controllers, c_{t_1} and c_{t_2} are the front and rear tire damping coefficients, and k_{t_1} and k_{t_2} are the front and rear tire stiffness coefficients.

Given the above notations and definitions, it is now intended to formulate the nonlinear dynamics of the considered suspension system. The 4-DOF half-car model comprises various components including front and rear tires, front and rear unsprung masses, and body masses working altogether. Apparently, potholes, bumps, road roughness and other

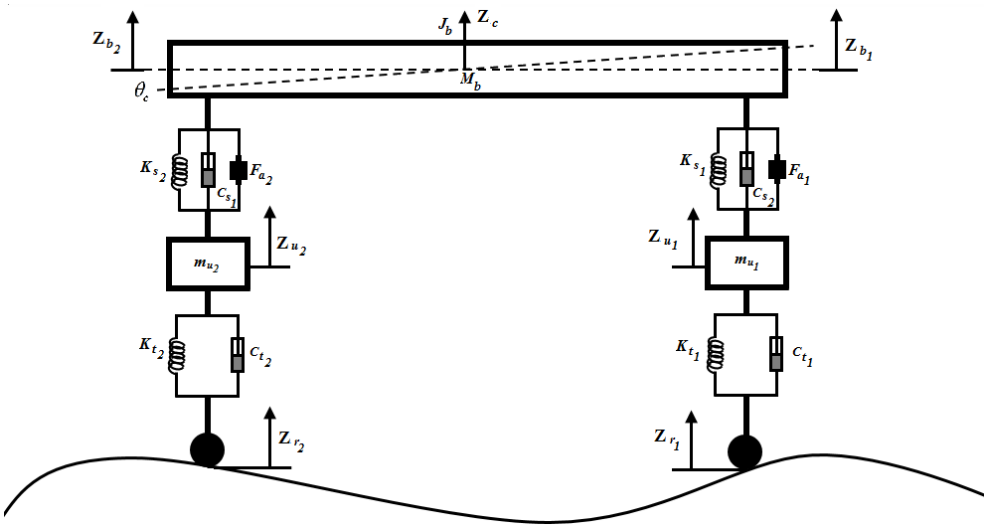

Figure 3.1: Schematic illustration of 4-DOF half-car active suspension system on road.

sources of unknown disturbances excite the vehicle. In such conditions, suspension system serves as a filter which tries to mitigate the abrupt and continuous deflections to reduce the possible damages, and also improve the ride comfort. Under excitation / vibration, each of the components of suspension system are affected and follow specific equations. To have a sense regarding the overall behavior of suspension system, the governing equations for each of the components should be written done separately and combined to simulate the dynamics of suspension system. In the presented mathematical model, the Newton's law of motion is used to formulate the nonlinear dynamics of the half-car suspension system. In this context, one can formulate the front and rear body mass displacements as:

$$\begin{cases} z_{b_1} = z_c + l_1 \sin\theta \\ \dot{z}_{b_1} = \dot{z}_c + l_1 \dot{\theta} \cos\theta \\ z_{b_2} = z_c - l_2 \sin\theta \\ \dot{z}_{b_2} = \dot{z}_c - l_2 \dot{\theta} \cos\theta \end{cases}, \tag{3.1}$$

where l_1 and l_2 are the distances of front and rear axles from the center of gravity of body mass, z_{b_1}, z_{b_2}, and z_c are the front, rear, and center of gravity of body mass vertical displacements, $\dot{z}_{b_1}$, $\dot{z}_{b_2}$, and $\dot{z}_c$ are the front, rear, and center of gravity of body mass velocities, θ is the body mass pitch angle, and $\dot{\theta}$ represents the angular velocity.

As mentioned, in the considered model, the nonlinearity of damping component is

taken into account to increase the accuracy of simulation. This nonlinearity is imposed by an additive term as a function of linear damping, asymmetric damping, and nonlinear damping coefficients. The mathematical formulation of the damping force of suspension system can be given as:

$$\begin{cases} F_{bs_1} = c_{s_1}^{l}\left(\dot{z}_{u_1} - \dot{z}_{b_1}\right) - c_{s_1}^{sym}\left|\dot{z}_{u_1} - \dot{z}_{b_1}\right| + c_{s_1}^{nl}\left(\sqrt{\left|\dot{z}_{u_1} - \dot{z}_{b_1}\right|}\right) sgn\left(\dot{z}_{u_1} - \dot{z}_{b_1}\right) \\ F_{bs_2} = c_{s_2}^{l}\left(\dot{z}_{u_2} - \dot{z}_{b_2}\right) - c_{s_2}^{sym}\left|\dot{z}_{u_2} - \dot{z}_{b_2}\right| + c_{s_2}^{nl}\left(\sqrt{\left|\dot{z}_{u_2} - \dot{z}_{b_2}\right|}\right) sgn\left(\dot{z}_{u_2} - \dot{z}_{b_2}\right) \end{cases},$$

where F_{bs_1} and F_{bs_2} are the front and rear damping forces, $\dot{z}_{u_1}$ and $\dot{z}_{u_2}$ are the front and rear unsprung mass vertical velocities, and c_s^l, c_s^{sym}, and c_s^{nl} are the linear, asymmetric and nonlinear damping coefficients of dampers, respectively.

The nonlinearity of the springs of suspension system is formulated as follows:

$$\begin{cases} F_{ks_1} = k_{s_1}^{l}\left(z_{u_1} - z_{b_1}\right) + k_{s_1}^{nl}\left(z_{u_1} - z_{b_1}\right)^3 \\ F_{ks_2} = k_{s_2}^{l}\left(z_{u_2} - z_{b_2}\right) + k_{s_2}^{nl}\left(z_{u_2} - z_{b_2}\right)^3 \end{cases},$$

where F_{ks_1} and F_{ks_2} are the front and rear spring forces, z_{u_1} and z_{u_2} are the front and rear unsprung mass vertical displacements, and k_s^l and k_s^{nl} are the linear and nonlinear stiffness coefficients of springs, respectively.

Given the recommendation of [48], tires are modelled by a linear spring and damper system rather than a simple spring to improve the accuracy of simulation. The mathematical formulation of the tire damping force is given below:

$$\begin{cases} F_{bt_1} = c_{t_1}\left(\dot{z}_{u_1} - \dot{r}_1\right) \\ F_{bt_2} = c_{t_2}\left(\dot{z}_{u_2} - \dot{r}_2\right) \end{cases},$$

where F_{bt_1} and F_{bt_2} are the front and rear tire damping forces, c_{t_1} and c_{t_2} are the rear and front tire damping coefficients, and $\dot{r}_1$ and $\dot{r}_2$ are the derivatives of front and rear tires' disturbances with respect to time, respectively.

The mathematical formulations of the front and rear tire damping forces are given below:

$$\begin{cases} F_{kt_1} = k_{t_1}\left(z_{u_1} - r_1\right) \\ F_{kt_2} = k_{t_2}\left(z_{u_2} - r_2\right) \end{cases},$$

where F_{kt_1} and F_{kt_2} are the front and rear tire spring forces, k_{t_1} and k_{t_2} are the rear and front tire stiffness coefficients, and r_1 and r_2 are the front and rear tires' disturbances, respectively.

The total forces in the front and rear parts of suspension system can be calculated as:

$$\begin{cases} f_1 = F_{ks_1} + F_{bs_1} - F_{a_1} \\ f_2 = F_{ks_2} + F_{bs_2} - F_{a_2} \end{cases}.$$

By combining the equations of motion for each component, the equations of motion for half-car suspension system are obtained, as given below:

$$\begin{aligned} M_b\ddot{z}_c = {} & k_{s_1}^l\left(z_{u_1} - z_{b_1}\right) + k_{s_1}^{nl}\left(z_{u_1} - z_{b_1}\right)^3 + c_{s_1}^l\left(\dot{z}_{u_1} - \dot{z}_{b_1}\right) - c_{s_1}^{sym}\left|\dot{z}_{u_1} - \dot{z}_{b_1}\right| + c_{s_1}^{nl}\left(\sqrt{\left|\dot{z}_{u_1} - \dot{z}_{b_1}\right|}\right) sgn\left(\dot{z}_{u_1} - \dot{z}_{b_1}\right) \\ & + k_{s_2}^l\left(z_{u_2} - z_{b_2}\right) + k_{s_2}^{nl}\left(z_{u_2} - z_{b_2}\right)^3 + c_{s_2}^l\left(\dot{z}_{u_2} - \dot{z}_{b_2}\right) - c_{s_2}^{sym}\left|\dot{z}_{u_2} - \dot{z}_{b_2}\right| + c_{s_2}^{nl}\left(\sqrt{\left|\dot{z}_{u_2} - \dot{z}_{b_2}\right|}\right) sgn\left(\dot{z}_{u_2} - \dot{z}_{b_2}\right) \\ & - F_{a_1} - F_{a_2} \end{aligned}$$

$$\begin{aligned} J_b\ddot{\theta} = {} & -l_1 k_{s_1}^l\left(z_{u_1} - z_{b_1}\right)\cos\theta - l_1\cos\theta k_{s_1}^{nl}\left(z_{u_1} - z_{b_1}\right)^3 - l_1\cos\theta c_{s_1}^l\left(\dot{z}_{u_1} - \dot{z}_{b_1}\right) + l_1\cos\theta c_{s_1}^{sym}\left|\dot{z}_{u_1} - \dot{z}_{b_1}\right| \\ & - l_1\cos\theta c_{s_2}^{nl}\left(\sqrt{\left|\dot{z}_{u_2} - \dot{z}_{b_2}\right|}\right) sgn\left(\dot{z}_{u_2} - \dot{z}_{b_2}\right) + l_2 k_{s_2}^l\left(z_{u_2} - z_{b_2}\right)\cos\theta + l_2\cos\theta k_{s_2}^{nl}\left(z_{u_2} - z_{b_2}\right)^3 \\ & + l_2\cos\theta c_{s_2}^l\left(\dot{z}_{u_2} - \dot{z}_{b_2}\right) - l_2\cos\theta c_{s_2}^{sym}\left|\dot{z}_{u_2} - \dot{z}_{b_2}\right| + l_2\cos\theta c_{s_2}^{nl}\left(\sqrt{\left|\dot{z}_{u_2} - \dot{z}_{b_2}\right|}\right) sgn\left(\dot{z}_{u_2} - \dot{z}_{b_2}\right) \\ & + F_{a_1} l_1\cos\theta - F_{a_2} l_2\cos\theta \end{aligned}$$

$$\begin{aligned} m_{u_1}\ddot{z}_{u_1} = {} & -k_{t_1}\left(z_{u_1} - \omega_{r_1}\right) - c_{t_1}\left(\dot{z}_{u_1} - \dot{\omega}_{r_1}\right) - k_{s_1}^l\left(z_{u_1} - z_{b_1}\right) - k_{s_1}^{nl}\left(z_{u_1} - z_{b_1}\right)^3 - c_{s_1}^l\left(\dot{z}_{u_1} - \dot{z}_{b_1}\right) \\ & + c_{s_1}^{sym}\left|\dot{z}_{u_1} - \dot{z}_{b_1}\right| - c_{s_1}^{nl}\left(\sqrt{\left|\dot{z}_{u_1} - \dot{z}_{b_1}\right|}\right) sgn\left(\dot{z}_{u_1} - \dot{z}_{b_1}\right) + F_{a_1} \end{aligned}$$

$$\begin{aligned} m_{u_2}\ddot{z}_{u_2} = {} & -k_{t_2}\left(z_{u_2} - \omega_{r_2}\right) - c_{t_2}\left(\dot{z}_{u_2} - \dot{\omega}_{r_2}\right) - k_{s_2}^l\left(z_{u_2} - z_{b_2}\right) - k_{s_2}^{nl}\left(z_{u_2} - z_{b_2}\right)^3 - c_{s_2}^l\left(\dot{z}_{u_2} - \dot{z}_{b_2}\right) \\ & + c_{s_2}^{sym}\left|\dot{z}_{u_2} - \dot{z}_{b_2}\right| - c_{s_2}^{nl}\left(\sqrt{\left|\dot{z}_{u_2} - \dot{z}_{b_2}\right|}\right) sgn\left(\dot{z}_{u_2} - \dot{z}_{b_2}\right) + F_{a_2} \end{aligned}$$

$$\begin{aligned} \ddot{z}_{b_1} = {} & \zeta\left[k_{s_1}^l\left(z_{u_1} - z_{b_1}\right) + k_{s_1}^{nl}\left(z_{u_1} - z_{b_1}\right)^3 + c_{s_1}^l\left(\dot{z}_{u_1} - \dot{z}_{b_1}\right) - c_{s_1}^{sym}\left|\dot{z}_{u_1} - \dot{z}_{b_1}\right| + c_{s_1}^{nl}\left(\sqrt{\left|\dot{z}_{u_1} - \dot{z}_{b_1}\right|}\right) sgn\left(\dot{z}_{u_1} - \dot{z}_{b_1}\right) - F_{a_1}\right] \\ & + \mu\left[k_{s_2}^l\left(z_{u_2} - z_{b_2}\right) + k_{s_2}^{nl}\left(z_{u_2} - z_{b_2}\right)^3 + c_{s_2}^l\left(\dot{z}_{u_2} - \dot{z}_{b_2}\right) - c_{s_2}^{sym}\left|\dot{z}_{u_2} - \dot{z}_{b_2}\right| + c_{s_2}^{nl}\left(\sqrt{\left|\dot{z}_{u_2} - \dot{z}_{b_2}\right|}\right) sgn\left(\dot{z}_{u_2} - \dot{z}_{b_2}\right) - F_{a_2}\right] \end{aligned}$$

$$\begin{aligned} \ddot{z}_{b_2} = {} & \mu\left[k_{s_1}^l\left(z_{u_1} - z_{b_1}\right) + k_{s_1}^{nl}\left(z_{u_1} - z_{b_1}\right)^3 + c_{s_1}^l\left(\dot{z}_{u_1} - \dot{z}_{b_1}\right) - c_{s_1}^{sym}\left|\dot{z}_{u_1} - \dot{z}_{b_1}\right| + c_{s_1}^{nl}\left(\sqrt{\left|\dot{z}_{u_1} - \dot{z}_{b_1}\right|}\right) sgn\left(\dot{z}_{u_1} - \dot{z}_{b_1}\right) - F_{a_1}\right] \\ & + \gamma\left[k_{s_2}^l\left(z_{u_2} - z_{b_2}\right) + k_{s_2}^{nl}\left(z_{u_2} - z_{b_2}\right)^3 + c_{s_2}^l\left(\dot{z}_{u_2} - \dot{z}_{b_2}\right) - c_{s_2}^{sym}\left|\dot{z}_{u_2} - \dot{z}_{b_2}\right| + c_{s_2}^{nl}\left(\sqrt{\left|\dot{z}_{u_2} - \dot{z}_{b_2}\right|}\right) sgn\left(\dot{z}_{u_2} - \dot{z}_{b_2}\right) - F_{a_2}\right] \end{aligned}$$

where $\zeta = \left(\frac{1}{M_b} + \frac{l_1^2}{J_b}\right)$, $\mu = \left(\frac{1}{M_b} - \frac{l_1 l_2}{J_b}\right)$, and$\gamma = \left(\frac{1}{M_b} + \frac{l_2^2}{J_b}\right)$.

The nominal values of the model parameters are given in Table 3.1. It should be mentioned that due to external disturbances, and some unknown factors and unseen random events, it would be a wise choice to treat some of the model parameters as random variables, and perform statistical analysis to get more reliable answers. The discussion on the mentioned concern will be given later in this chapter, and it is discussed how to treat with some of the model parameters via uncertainty analysis.

As mentioned before, for calculating the control commands of LBMPC, a piece-wise linear version of the presented nonlinear model should be determined. In what follows this section, the details of the steps taken for piece-wise linearization of the nonlinear mathematical model are presented.

Table 3.1: Model parameters for half-car vehicle suspension system

Parameters	**Value**	**Parameters**	**Value**
V	20 (m/s)	$c^{l}_{s_2}$	700 (Ns/m)
M_b	580 (kg)	$c^{sym}_{s_1}, c^{sym}_{s_2}$	400 (Ns/m)
m_{u_1}	40 (kg)	$c^{nl}_{s_1}, c^{nl}_{s_2}$	400 (Ns/m)
m_{u_2}	35.5 (kg)	$k^{l}_{s_1}, k^{l}_{s_2}$	14000 (N/m)
L_1	1.5 (m)	$k^{nl}_{s_1}, k^{nl}_{s_2}$	2.35e4 (N/m)
L_2	1 (m)	k_{t_1}, k_{t_2}	190e3 (N/m)
J_b	1100 $(kg.m^2)$	c_{t_1}	80 (Ns/m)
$c^{l}_{s_1}$	800 (Ns/m)	c_{t_2}	70 (Ns/m)

Let's suppose a nonlinear function $g(\boldsymbol{\beta}, t)$, with $\boldsymbol{\beta} = (\beta_1, \beta_2, ..., \beta_d)^T$ is going to be linearized. Then, the piece-wise linear function can be approximated around the working point $\boldsymbol{\beta_0}$, as follows:

$$g(\boldsymbol{\beta}, t) \simeq \boldsymbol{\wp}(\boldsymbol{\beta} - \boldsymbol{\beta_0}),$$

where $\boldsymbol{\wp}$ is a $1 \times d$ vector, and its i^{th} array is calculated as follows:

$$\wp_i = \left. \frac{\partial g\left(\boldsymbol{\beta}, t\right)}{\partial \beta_i} \right| \boldsymbol{\beta} = \boldsymbol{\beta_0} \quad , \quad i = 1, ..., d.$$

The nonlinearity of the model comes from the characteristics of the dampers and springs of suspension system. Both of the dampers and springs consist of a linear part and a nonlinear part, and an equivalent linear function should be obtained for the nonlinear terms to make the model suit for LBMPC. Suppose that $\delta_1 = z_{u_1} - z_{b_1}$ and $\delta_2 = z_{u_2} - z_{b_2}$, then the piece-wise linear format of spring model can be given as:

$$\begin{cases} k^{nl}_{s_1} \left(\delta_1\right)^3 = \wp_{ks_1} \left(\delta_1 - \left(\delta_1\right)_0\right) \\ k^{nl}_{s_1} \left(\delta_2\right)^3 = \wp_{ks_2} \left(\delta_2 - \left(\delta_2\right)_0\right) \end{cases},$$

where $\wp_{ks_1} = 3k^{nl}_{s_1} \left(\delta_1\right)^2_0$ and $\wp_{ks_2} = 3k^{nl}_{s_2} \left(\delta_2\right)^2_0$.

Also, suppose that $\dot{\delta}_1 = \dot{z}_{u_1} - \dot{z}_{b_1}$ and $\dot{\delta}_2 = \dot{z}_{u_2} - \dot{z}_{b_2}$, then the piece-wise linear format

of damping coefficient can be given as follows:

$$\begin{cases} c_{s_1}^{sym}\left|\dot{\delta}_1\right| = \wp_{s_1}^{sym}\left(\dot{\delta}_1 - \left(\dot{\delta}_1\right)_0\right) \\ c_{s_1}^{nl}\left(\sqrt{\dot{\delta}_1}\right) sgn\left(\dot{\delta}_1\right) = \wp_{s_1}^{nl}\left(\dot{\delta}_1 - \left(\dot{\delta}_1\right)_0\right) \\ c_{s_2}^{sym}\left|\dot{\delta}_2\right| = \wp_{s_2}^{sym}\left(\dot{\delta}_2 - \left(\dot{\delta}_2\right)_0\right) \\ c_{s_2}^{nl}\left(\sqrt{\left|\dot{\delta}_2\right|}\right) sgn\left(\dot{\delta}_2\right) = \wp_{s_2}^{nl}\left(\dot{\delta}_2 - \left(\dot{\delta}_2\right)_0\right) \end{cases},$$

where

$$\begin{cases} \wp_{s_1}^{sym} = c_{s_1}^{sym}\frac{\left(\dot{\delta}_1\right)_0}{\left|\left(\dot{\delta}_1\right)_0+\varepsilon\right|} \\ \wp_{s_1}^{nl} = c_{s_1}^{nl}\left[\frac{1}{2}\frac{\left(\dot{\delta}_1\right)_0}{\left|\left(\dot{\delta}_1\right)_0+\varepsilon\right|}\left|\left(\dot{\delta}_1\right)_0+\varepsilon\right|^{-0.5} sgn\left(\left(\dot{\delta}_1\right)_0\right)\right] \\ \wp_{s_2}^{sym} = c_{s_2}^{sym}\frac{\left(\dot{\delta}_2\right)_0}{\left|\left(\dot{\delta}_2\right)_0+\varepsilon\right|} \\ \wp_{s_2}^{nl} = c_{s_2}^{nl}\left[\frac{1}{2}\frac{\left(\dot{\delta}_2\right)_0}{\left|\left(\dot{\delta}_2\right)_0+\varepsilon\right|}\left|\left(\dot{\delta}_2\right)_0+\varepsilon\right|^{-0.5} sgn\left(\left(\dot{\delta}_2\right)_0\right)\right] \end{cases}.$$

Based on the above piece-wise linearization, the equivalent model of the suspension system dynamics can be given as follows:

$$\begin{aligned} M_b\ddot{z}_c &= k_{s_1}^*\left(z_{u_1} - z_{b_1}\right) + c_{s_1}^*\left(\dot{z}_{u_1} - \dot{z}_{b_1}\right) + k_{s_2}^*\left(z_{u_2} - z_{b_2}\right) + c_{s_2}^*\left(\dot{z}_{u_2} - \dot{z}_{b_2}\right) - F_{a_1} - F_{a_2} \\ J_b\ddot{\theta} &= -l_1\cos\theta k_{s_1}^*\left(z_{u_1} - z_{b_1}\right) - l_1\cos\theta c_{s_1}^*\left(\dot{z}_{u_1} - \dot{z}_{b_1}\right) + F_{a_1}l_1\cos\theta \\ &\quad + l_2\cos\theta k_{s_2}^*\left(z_{u_2} - z_{b_2}\right) + l_2\cos\theta c_{s_2}^*\left(\dot{z}_{u_2} - \dot{z}_{b_2}\right) - F_{a_2}l_2\cos\theta \\ \ddot{z}_{b_1} &= \zeta\left[k_{s_1}^*\left(z_{u_1} - z_{b_1}\right) + c_{s_1}^*\left(\dot{z}_{u_2} - \dot{z}_{b_2}\right) - F_{a_1}\right] + \mu\left[k_{s_2}^*\left(z_{u_2} - z_{b_2}\right) + c_{s_2}^*\left(\dot{z}_{u_2} - \dot{z}_{b_2}\right) - F_{a_2}\right] \\ \ddot{z}_{b_2} &= \mu\left[k_{s_1}^*\left(z_{u_1} - z_{b_1}\right) + c_{s_1}^*\left(\dot{z}_{u_2} - \dot{z}_{b_2}\right) - F_{a_1}\right] + \gamma\left[k_{s_2}^*\left(z_{u_2} - z_{b_2}\right) + c_{s_2}^*\left(\dot{z}_{u_2} - \dot{z}_{b_2}\right) - F_{a_2}\right] \\ m_{u_1}\ddot{z}_{u_1} &= -k_{t_1}\left(z_{u_1} - r_1\right) - c_{t_1}\left(\dot{z}_{u_1} - \dot{r}_1\right) - k_{s_1}^*\left(z_{u_1} - z_{b_1}\right) - c_{s_1}^*\left(\dot{z}_{u_1} - \dot{z}_{b_1}\right) + F_{a_1} \\ m_{u_2}\ddot{z}_{u_2} &= -k_{t_2}\left(z_{u_2} - r_2\right) - c_{t_2}\left(\dot{z}_{u_2} - \dot{r}_2\right) - k_{s_2}^*\left(z_{u_2} - z_{b_2}\right) - c_{s_2}^*\left(\dot{z}_{u_2} - \dot{z}_{b_2}\right) + F_{a_2} \end{aligned} \tag{3.2}$$

where $k_{s_1}^* = k_{s_1}^l + \wp_{ks_1}$, $k_{s_2}^* = k_{s_2}^l + \wp_{ks_2}$, $c_{s_1}^* = c_{s_1}^l - \wp_{s_1}^{sym} + \wp_{s_1}^{nl}$, and$c_{s_2}^* = c_{s_2}^l - \wp_{s_2}^{sym} + \wp_{s_2}^{nl}$.

To make sure the piece-wise linear mathematical model operates properly, after calculating the control commands via LBMPC, a validation test is performed by feeding the calculated actuation signals to the original nonlinear model and comparing the obtained responses.

3.3 Uncertainty Sources for Suspension Unit Control

As mentioned before, a considerable portion of research reports have used the nominal mathematical models (e.g. 4-DOF suspension system model without disturbance) for model-based decision making and control. At most, some of the conducted researches have considered the road roughness as the dominant source of uncertainty, while neglecting so many other sources of uncertainty which can play part in practice [54]. This is mainly due to the fact that automotive engineer's interest lies in the design of control algorithms rather than performing statistical analysis to come up with realistic solutions. For more information regarding the performed research on uncertainty handling, one can refer to the literature review conducted in Chapter 2. It is quite clear that unless the effects of dominant sources of uncertainties have not been considered, it would be impossible to ensure the obtained results can be reliably used in practice. Therefore, based on the opinion of experts, published reports, and the author's own assessment, the most dominant sources of uncertainty which can affect the dynamics of vehicle on road are identified and categorized, and proper strategies are taken into account to capture them [106, 107].

In this section, the emphasis is exerted on categorizing and introducing the sources of uncertainty, and in the next section, it is mentioned how these stochastic events are simulated. Fortunately, LBMPC has a distinct module (oracle) which is best suited for simulating the uncertainties. Through numerical experiments and theoretical analysis, the usefulness of LBMPC for our simulation scenario (in which a remarkable attention is paid to the uncertainty of the formulated mathematical model) will be justified.

The sources of uncertainty which can affect vehicle dynamics and suspension system are as below:

1. Driver's behavior: It is quite clear that due to several behavioral and environmental factors, drivers' driving style can change on road. This is quite justifiable and realistic since it is, for example, possible that driver abruptly reduces the vehicle speed due to an accident or adjust the speed of vehicle when seeing a pothole or bump on road. Also, cognitive and psychological factors can affect the driver's driving style. Therefore, to have a realistic simulation, the effect of uncertainty due to

the driver's behavior should be taken into account. Unfortunately, the majority of conducted researches consider a constant cruise speed for the vehicle, as given in nominal simulation condition in Table 3.1. Given the information provided, it can be easily inferred that conducting the simulation with a constant speed does not have practical implication, and affords unrealistic simulation-based results and misjudgement. To the best of the author's knowledge, the current investigation is among the rare researches which pay a considerable attention to uncertainty of the driver's behavior, and its effect on the dynamics of the vehicle. It sounds logical to simulate the uncertainty of driver's behavior by taking the cruise speed as a random variable. The speed of the vehicle has an indirect relation with the excitation of suspension system. To be more precise, the more the speed of the vehicle, the faster the tires are exposed to the road profile. Thus, at varying speeds (which results from driver's behavior uncertainty), the controller should be powerful enough to adaptively calculate the controlling commands, and also to filter the shocks due to the time varying excitation of tires.

2. Number of passengers and load: Another constant factor in the nominal simulation scenario is the vehicles body mass (M_b) which directly affects the dynamics of suspension system. Just like the vehicle speed, the body mass can undergo remarkable variations due to several factors, e.g. number of passengers in the vehicle and the loads in the vehicle. However, this very important element has retained constant during the simulation in a large number of researches. To be more precise, the variation of body mass on road has been given more attention compared to driver's behavior uncertainty. There exist seminal reports in the literature in which the researchers have taken ideas from Kalman filtering and estimation theory to adaptively estimate the vehicles total body mass [108, 109]. In the current investigation, a statistical analysis is carried out to deal with the uncertainty of body mass. It sounds more logical to recline on the results with uncertain (varying) body mass compared to considering a constant vehicle mass.

3. Model-plant mismatch: For many of the existing systems, obtaining a high-fidelity model requires an exhaustive trial and error effort to extract usually a considerable number of variables, and precisely calculate the DOF of the considered system. High-fidelity models are often too complex and take a considerable computational power for the simulation of real systems' behavior. However, for applications such as model-based control and decision making, it is essential to have an abstract model which can represent the most important aspects of the real system. Within the context of control, such abstract models are called control-oriented models. To develop

a control-oriented model, one is allowed to consider the most prominent variables affecting the systems behavior. Also, the DOF of control-oriented models is less than that of high-fidelity models, and identification algorithms are used to identify the flexible parameters of control-oriented models by maximizing their correlation with high-fidelity models. The 4-DOF half-car model is an example of control-oriented models which is much simpler than a high-fidelity model of real vehicle suspension system. The salient asset of such a model lies in its computational speed which suits it to be used at the heart of LBMPC for the online calculation of control commands. So, it is obvious that there is a mismatch between the real suspension system and the considered model. On the other hand, there is a possibility that even the nominal values of the parameters of real-plant, such as damping coefficient, stiffness of springs, and *etc.*, change for the same product [106]. Therefore, that would be a wise choice to consider a distribution for some of the parameters of the control-oriented model rather than using the nominal values of model parameters. Such a consideration will result in a more realistic analysis and increases the reliability of the controller proposed for active suspension control.

4. Road roughness: Arguably, road roughness is one of the foremost challenging factors when designing a controlling algorithm for suspension systems. So far, tremendous efforts have been made to cope with the random excitation of suspension system due to the road roughness. In this research, a thorough investigation is conducted to come up with efficient statistical methods for forecasting the road roughness. To the best knowledge of the author, this investigation remarkably contributes to the research activities concerning the prediction of road roughness. To do so, well-known statistical theories and forecasting techniques are taken into account, and a confidence band is obtained to predict the roughness of the road with an acceptable robustness rate.

5. Information preview uncertainty: Modern vehicles are equipped with many sensors and communication systems which enable them to read information from global positioning systems (GPS) and radars. This actually paves the way of coming up with intelligent transportation system. Due to so many factors such as communication delay, rapture of vehicle sensors, and the noisy information transmitted to vehicle from radars, a certain degree of uncertainty is accompanied with information preview. So, for ITS, that would be mandatory to take the advantage of statistical techniques to cope with the effects of uncertainty.

For better understanding of the above details, Figure 3.2 represents a schematic illustration of a vehicle on road which encounters different sources of uncertainty.

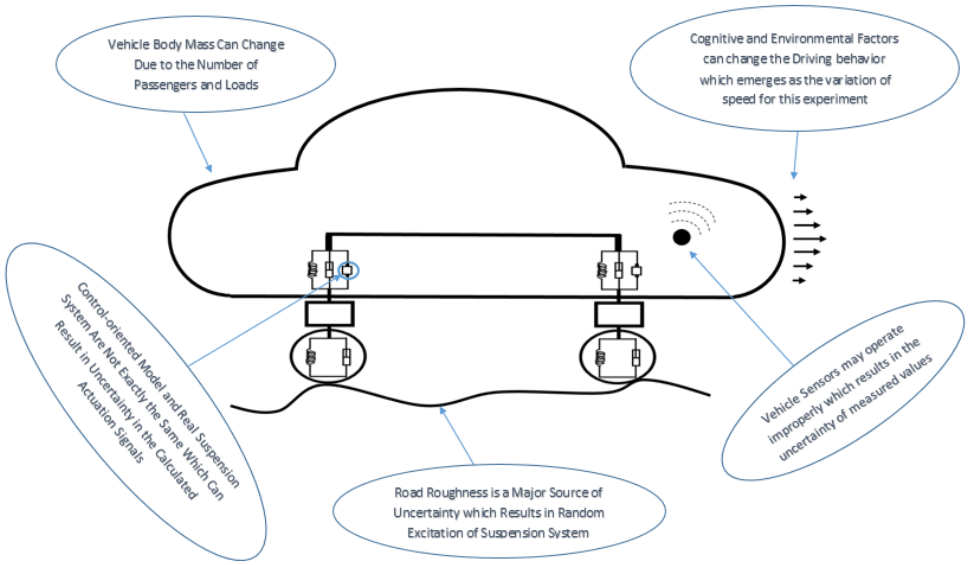

Figure 3.2: Schematic illustration of a vehicle with possible sources of uncertainty.

3.4 How to Treat / Simulate Uncertainties

In this section, it is intended to point out how the sources of uncertainty mentioned previously are handled in this investigation. Having said that detailed simulation and numerical experiments will be given in corresponding chapters.

Let's call the sources of uncertainties (SUs) as: (1) SU-1: driver's behavior, (2) SU-2: number of passengers and load, (3) SU-3: model-plant mismatch, (4) SU-4: road roughness, (5) SU-5 information preview uncertainty and the uncertainties due to the rapture of sensors and internal components of vehicles.

SU-1 is captured by implementing an absorbing stochastic Markov chain and designing a proper transition probability matrix in Chapter 5. In this way, a distribution is obtained for absorption time, and this is equivalent to the time that driver can manage the condition to get back to the cruise speed after a random event during the driving cycle. For SU-2 and SU-3, a thorough discussion and analysis is performed by considering a distribution for unknown model parameters in Chapter 8 after the implementation of the state-space model in Chapter 6. For SU-4, a separate investigation is carried out in Chapter 4 and several well-known forecasting models and statistical theories / techniques are considered to come up with an authentic and robust way of predicting the road roughness. For SU-

5, we have two scenarios of uncertainties. Actually, coping with the uncertainties due to the noisy information transmission from radar and GPS falls within the category of ITS, which is out of the scope of this study, as we neither use an intelligent car nor any satellite based information preview. However, the uncertainty due to the rapture of sensors and internal components of vehicles lies within the scope of the current investigation, and has an unknown and unpredictable nature. For the current simulation, we consider it as an additive disturbance and took the advantage of oracle module of LBMPC to tame it. The simulation results concerning the use of oracle for SU-5 are presented in details in Chapter 7.

The information provided in this chapter serves as a base for the extensions and contributions of the next chapters. So, some of the mathematical details presented in this chapter are considered as known truth in the coming chapters, and the author may simply refer the readers to Chapter 3 for some basic information about the control-oriented model, sources of uncertainties, and *etc.*

Chapter 4

Prediction of Road Roughness

This chapter is devoted to the design and evaluation of statistical methods for predicting the road roughness (also called road disturbance). After designing an appropriate predictor which holds a desirable trade-off between robustness and efficiency, it will be used as a model to predict the disturbances resulting from road roughness at the heart of the control-oriented state-space model used for designing learning based model predictive controller (LBMPC). At the end of the chapter, one can find informative clues on how well statistical methods and data analysis can guarantee the both robustness and efficiency of road disturbance prediction module.

4.1 Problem Statement and Review

Arguably, the rapid development of computational processors has made it possible for practitioners to adopt mathematical models for hardware-in-the-loop online applications. Automotive industry is one of such fields that has instigated engineers and mathematicians to develop sophisticated prediction and controlling modules for performance improvement. A detailed review of the conducted research on designing efficient predictive controllers and prediction tools was given in Chapter 2. In this chapter, we focus on the most remarkable achievements pertaining to the development of prediction modules for predicting the future roughness of road. In the context of vehicle suspension control, road roughness is regarded as a source of uncertainty and is usually included in the state-space model as an additive disturbance. This is mainly due to the fact that for any kind of vehicle suspension control (including passenger comfort and deflection control), road roughness results in the deflection of tires which imposes an external force to spring and dampers. Therefore, it is

vital for any kind of active controller to have a good sense of road deflection for a proper reaction in a right time. This is even more crucial when the active controller is of predictive type in which the optimum controlling command is determined using the future behavior of the plant. Apparently, using a model which cannot precisely predict the future road roughness is very offensive, as under such a condition, the controlling command is determined based on a wrong disturbance profile. The exact mathematical formulation of the predictive state-space, which clearly states that how the controlling command is related to the future road roughness profile, will be given in Chapter 6.

The idea of searching for well-established prediction models have come to the mind of researchers working on designing predictive control algorithms for vehicle suspension system control. Here, we will only concentrate on citing and analyzing those research papers which have rational logics and convincing reasons for selecting a predictive method for determining the future road roughness. Having said that the literature is full of research reports in which either an arbitrary predictor is used without any justification or a trial-and-error procedure is conducted to design a road roughness prediction module. Also, there exist a vast number of research papers in which the road roughness is replaced with some basic sinusoids or spikes which results in unrealistic controlling commands [110, 111, 112, 23]. In [113], several intelligent techniques including neuro-fuzzy and fuzzy logic based methods were used for predicting the road roughness. These methods did not consider any assumption on the distribution of data, and only developed an interpolator based on a number of input/output pairs. In [114], a set of regression techniques, including artificial neural network (ANN), were evaluated, and it was concluded that intelligent methods are good choices for roughness estimation. For the analysis of the results, the authors merely reclined on the correlation between model output and data metric, and left the other important issues untouched. For example, they did not mention the fact that ANN simply develops a nonlinear map between the input-output pairs used for training, and also, the simulation did not contain any information regarding the robustness of the prediction. In [115], radial basis neural network (RBNN) was used to predict the road displacement (in the form of acceleration) for different road conditions such as concrete, waved stone block paved and country roads. The veracity of the predictor was verified using mean-squared error (MSE). One of the main flaws associated with this study was the relatively low number of training/testing data. Indeed, the authors evaluated RBNN using a limited test data, and then, conducted a sensitivity analysis (based on the assumption that RBNN is precise and robust). There also exist a number of research papers considering the whole suspension system model together with the road disturbance as a unique uncertain system. In [68], both modeling and control of a nonlinear half-car suspension system were carried out by means of a hybrid fuzzy logic method, obtained by hybridizing proportional-

integral-derivative (PID) and fuzzy logic controller (FLC). The uncertainty and unknown parameters in the nonlinear model were treated with arbitrary membership functions, and a comparative study was carried out to ascertain the properness of the controller. Since, the shape of the considered membership functions were designed randomly, and had no meaningful tie with the underlying nature of the model (including suspension model uncertainty and road profile disturbance), the controller can be viewed as a black-box which is not appropriate for interpretation and inference. In [116], a FLC was developed for the seat vibrations of a nonlinear vehicle model. However, the simulation suffered exactly from the same flaws, and there were neither a firm statistical analysis nor any knowledge coming from experimental data, and thus, the veracity of the model could not be assured.

There also exist a number of research papers in the literature which are based on more realistic scenarios to predict the road roughness. In [27, 56, 117], it was assumed that a multi-harmonic input which is close to an actual road profile can be used for simulation. To attain this, one way is to consider pseudo-random process resulting from the summation of several non-commensurately related sine waves. It is recommended to select spatial frequencies of certain forms to achieve desirable pseudo-random profiles. In spite of the fact that such a scenario borrows some statistical ideas to model the randomness, it still has a set of parameters which should be manipulated. However, usually these tunable parameters are set based on priori information and judgment of the user. For example, in [27], the final pseudo-random processes generated a profile which was similar to Gaussian white noise. However, the real experiments indicate that mean-stationary Gaussian profile is not a realistic model for simulating the random roughness of roads. In [54], a stochastic process was proposed to generate time histories of the front wheel road input. The model used power spectral density function, and also had a uniformly distributed additive noise to avoid periodicity. Although the use of statistical theories and methods for predicting the road roughness is appreciated, the considered model was just a simulation technique, design based on the authors' judgement, and did not have anything to do with the real characteristics of road profile.

By precise reading and contemplating the results and conclusions of the above research papers, two key questions come to one's mind which need further analysis and simulation, at-least from statistical view point. Firstly, in the majority of the conducted research, the road-roughness prediction is simply treated as a point estimation problem, without any experimental quantification of the distribution of data, or using any statistical theorem to come up with a confidence interval for the predictions. This hinders the engineers from having a proper inference by means of the developed prediction model, which makes such a blind point estimation unacceptable. This is mainly because at the end of the day, the prediction module is going to be used at the heart of real-time predictive controller.

Therefore, making inference by the developed prediction model, as well as the boundedness of prediction error are very important. In some of the conducted research (such as [27, 56, 117, 54, 118, 119]), a much more logical procedure was taken into account using the concept of Monte-Carlo simulation with some assumptions on the disturbance of road roughness. Despite the fact that considering a source of uncertainty and performing statistical analysis for designing an efficient predictor is appreciable, there are still some important questions regarding the authenticity of the conducted research, which should be clearly addressed. To the best of the author's knowledge, most of the papers which have tried to use statistical techniques to design a practical predictor have never taken strides beyond software based simulation (see for example [54, 118]). In these papers, it was simply assumed that by considering the road roughness as a random variable and assigning a certain type of distribution to it, one can derive acceptable statistical conclusions regarding the robustness and accuracy of the predictor. For example, in [27, 56, 117, 54, 118, 119], it was assumed that the road roughness approximately follows the Gaussian distribution. This is when, as discussed later, our analysis on experimentally captured data indicates that the distribution of road roughness is not Gaussian at all. However, its 1^{st} and 2^{nd} order differenced versions can be considered as Gaussian (1^{st} order stationary) time series.

Unfortunately, such simple assumptions which can facilitate the computer-based simulation have hindered the majority of automotive engineering researchers from deriving appropriate conclusions, and some of the concluded results may not be practical.

To cope with the abovementioned flaws, and also to have a fare judgment on the veracity of the existing reports, a throughout analysis is conducted using different well-established statistical theorems and techniques. Also, all of the comparative and theoretical analysis are conducted on experimental data which are acquiesced using an accelerometer sensor. By doing so, it is tried to ensure that the recommendations and conclusions derived are authentic, and can be used by other practitioners who intend to develop a predictor for determining the road roughness.

It should be mentioned that the predictor designed in this section will be finally used at the heart of predictive controller for predicting the road disturbance and will be included in the state-space representation as an additive noise. For a better vision, the schematic illustration of the process is given in Figure 4.1.

4.2 Statistical Prediction Strategies

In this section, the methods used for predicting the road roughness are formulated. The considered models are among the most applicable statistical techniques which have suc-

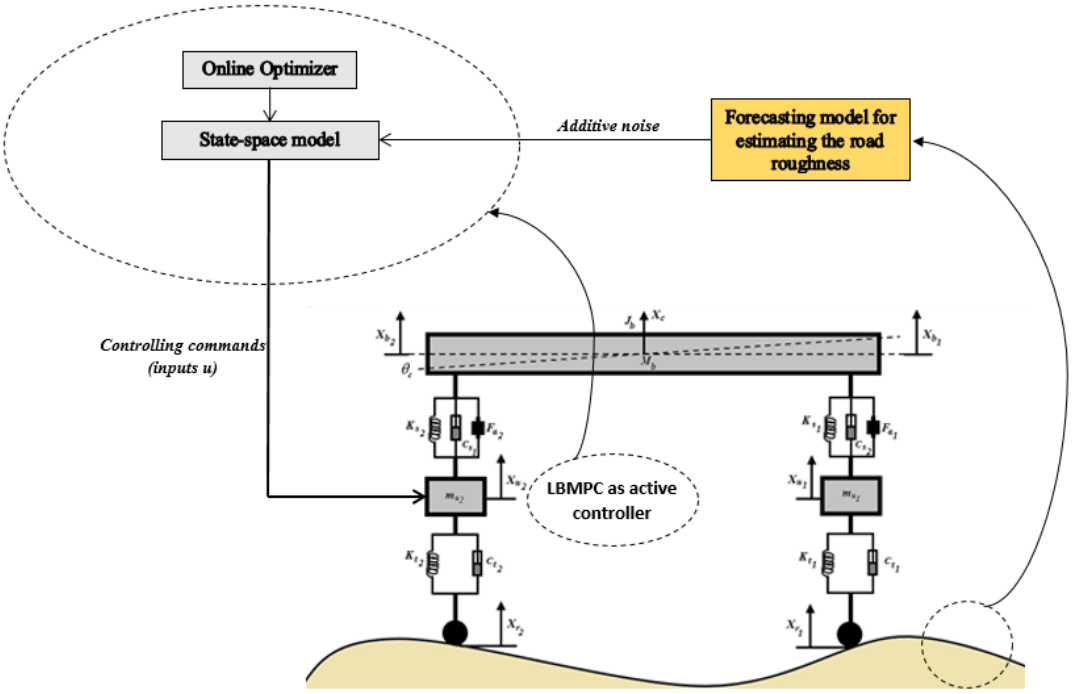

Figure 4.1: LBMPC with prediction module for suspension system control.

cessfully been applied to different prediction and forecasting problems [120, 121]. Also, they are efficient, easy to implement and have firm theoretical background. Since the road roughness is included in the suspension system state-space model as an additive noise (see Figure 4.1), selecting predictors with simple structure is quite advantageous. This is because such methods do not take a considerable computational time for performing the prediction, and are good fits for real-time applications, such as the control problem at hand in which the prediction and control are done in a real-time fashion, and computational speed is of highest importance.

4.2.1 Notations and preliminaries

Before starting the formulation of considered methods, there are a few notes which should be pointed out. All vectors are shown by lower case bold fonts ($\mathbf{a}$, $\mathbf{b}$, and *etc.*). Sets are presented by calligraphic upper case italic format ($\mathcal{A}$, $\mathcal{B}$, and *etc.*), and matrixes are given by upper case italic format (A, B, and *etc.*). The vectors are presented in columns, i.e.

$\mathbf{a} = col(a_1, a_2, ..., a_m)$, where $\mathbf{a} \in \mathbb{R}^m$. The transpose of matrix A is shown by A^T. Also, a vector of zeros of appropriate size (e.g. $n \times 1$) is shown by $\mathbf{0}_{n\times 1}$.

4.2.2 Moving average process

Suppose that the time series $\{\mathbf{x}_t\}_{t\in\mathbb{Z}}$ is collected. Moving average (MA) process is a forecasting method which estimates the value of a time series at time t (x_t) as a linear combination of previously captured noises plus the noise at time t. Let's define the backward shift operator B as below:

$$\mathrm{B}^l x_t = x_{t-l},\ l \in \mathbb{Z}\ .$$

Also, let's define the operator $\boldsymbol{\theta}(\mathrm{B})$, as follows:

$$\boldsymbol{\theta}(\mathrm{B}) := 1 + \sum_{l=1}^{q} \theta_l \mathrm{B}^l\ ,$$

such that when $\boldsymbol{\theta}(\mathrm{B})$ operates on x_t, we get:

$$\boldsymbol{\theta}(\mathrm{B})x_t = x_t + \theta_1 x_{t-1} + \cdots + \theta_q x_{t-q}\ .$$

Definition 4.1 Let $\{\boldsymbol{\omega}_t\}_{t\in\mathbb{Z}}$ be an independent and identically distributed sequence of random noises, i.e. $\omega_t \overset{iid}{\sim} N(0, \sigma^2)$, where $\sigma^2 < \infty$. Then, the time series obtained by q^{th} order moving average processes, MA(q), is defined as:

$$x_t := \boldsymbol{\theta}(\mathrm{B})\omega_t = \omega_t + \theta_1\omega_{t-1} + \theta_2\omega_{t-2} + \cdots + \theta_q\omega_{t-q}\ .$$

By simple calculations, the mean and auto-covariance of MA(q) can be respectively obtained as:

$$E[x_t] = 0\ ,$$

$$\gamma(t,s) = \begin{cases} \sigma^2 \sum\limits_{j=0}^{q-|t-s|} \theta_j\theta_{j+|t-s|} & \text{if } \mid t-s \mid \leq q \\ 0 & \text{otherwise} \end{cases}\ ,$$

where $E\big[\,\cdot\,\big]$ is the expectation operator. It can be shown that MA process is stationary.

Definition 4.2 A process $\{\mathbf{x}_t\}_{t\in\mathbb{Z}}$ is causal if x_t is only a function of $\omega_s \overset{iid}{\sim} N(0, \sigma^2)$, such that $s \leq t$.

As seen, MA(q) has a very simple formulation, and yet is among the most powerful prediction / forecasting methods. One the salient assets of MA(q) process is that it predicts based on the previously captured noises (it is a causal process), which makes its use for practical and real-time applications possible, thanks to the sensors and noise measurement tools at-hand.

4.2.3 Autoregressive process

The idea behind using autoregressive (AR) process is to estimate the value of a time series at time t (x_t) as a linear combination of previously captured values of time series plus the noise at time t.

Let's define the operator $\boldsymbol{\phi}(\mathrm{B})$, as below:

$$\boldsymbol{\phi}(\mathrm{B}) := 1 + \sum_{l=1}^{p} -\phi_l \mathrm{B}^l \ ,$$

where B is the backward shift operator, and also, when $\boldsymbol{\phi}(\mathrm{B})$ operates on x_t, we get:

$$\boldsymbol{\phi}(\mathrm{B})x_t = x_t - \phi_1 x_{t-1} - \cdots - \phi_p x_{t-p} \ .$$

Definition 4.3 Let $\{\boldsymbol{x}_t\}_{t\in\mathbb{Z}}$ be the time series and $\omega_t \overset{iid}{\sim} N(0, \sigma^2)$, where $\sigma^2 < \infty$. Then, the p^{th} order autoregressive processes, AR(p), is the stationary solution to the differenced equation below:

$$\boldsymbol{\phi}(\mathrm{B})x_t = x_t - \phi_1 x_{t-1} - \cdots - \phi_p x_{t-p} = \omega_t \ .$$

It can be shown that AR(p) is stationary if the polynomial below:

$$\boldsymbol{\phi}(z) = 1 - \phi_1 z - \phi_1 z^2 - \cdots - \phi_p z^p \ ,$$

has no root on the unit circle. Also, if all of the roots strictly lie outside the unit circle, by some calculation, it can be shown that the AR(p) is causal and can be presented as:

$$x_t = \sum_{j=0}^{\infty} \psi_j \omega_{t-j} \ .$$

The mean and auto-covariance of a causal and stationary AR(p) process can, respectively, be given as:

$$E[x_t] = 0 \ ,$$

$$\gamma(t,s) = \sigma^2 \sum_{j=0}^{\infty} \psi_j \psi_{j+|t-s|} \ .$$

AR(p) is a very useful forecasting / prediction method given its simple formulation, as well as its capability to forecast based on the previous measured states of the system. This is especially beneficial for state-space model (hidden Markov model) based controllers, which evolve the system dynamics based on the variation of states, as a function of the previous values of the states.

4.2.4 Autoregressive moving average process

Autoregressive moving average (ARMA) process is an efficient forecasting method which takes advantage from both MA and AR process. Given a fine tuning of its parameter, it can usually yield much better estimation for real-word problems, compared to MA and AR. By means of $\boldsymbol{\theta}$(B) and $\boldsymbol{\phi}$(B) operators defined in the previous sections, the formulation of ARMA process can presented.

Definition 4.4 Let $\{\boldsymbol{x}_t\}_{t\in\mathbb{Z}}$ be the time series and $\{\boldsymbol{\omega}_t\}_{t\in\mathbb{Z}}$ be an independent and identically distributed sequence of random noises, i.e. $\omega_t \overset{iid}{\sim} N(0,\sigma^2)$, where $\sigma^2 < \infty$. Then ARMA(p, q) process is the stationary solution to the differenced equation below:

$$\boldsymbol{\phi}(\mathrm{B})x_t = \boldsymbol{\theta}(\mathrm{B})\omega_t \ .$$

Note that when forming ARMA(p, q) process, it is assumed that polynomials $\boldsymbol{\theta}(z)$ and $\boldsymbol{\phi}(z)$ possess no common roots. Just like AR(p) process, ARMA is stationary when the polynomial $\boldsymbol{\phi}(z)$ has no root on the unit circle. Also, if all of the roots strictly lie outside the unit circle, the process is causal.

There is no unique / exact way to find the proper values of ARMA process (unlike AR and MA which could be tuned by looking at the graphical auto-correlation function (ACF) and partial auto-correlation function (PACF) plots) for a given data. Therefore optimization methods such as maximum likelihood and least-square technique are usually used to find the values of p and q. Also, it is usually the case that the obtained ARMA is such that $p + q << n$, where n is the number of training data points (actually the same holds for AR and MA). So, such methods can be viewed as short-term forecasting techniques, and in long-term, the forecasted value is just the mean.

Therefore, it would be a wise choice to seek for more advanced forecasting methods and use them together with MA, AR, and ARMA.

4.2.5 APARCH model

It is possible that, due to the complexity of time series data, some type of periodicity or volatility clustering be observed. In such cases, the abovementioned methods do not work efficiently, and usually the trained model's residual sequence is not white noise. APARCH is an interesting method which tries to take advantage from the potentials of ARMA process and a nonlinear model called generalized autoregressive conditional heteroskedastic (GARCH) process [120]. GARCH is best suited for cases in which a volatility clustering is observed in the time series. The beauty of the considered APARCH model is that it tries to use GARCH to reduce the possible dependency of the residual errors obtained by a fitted ARMA model. In this context, after training an ARMA model and conducting model diagnosis, if the ACF plot of residual error has significant values (due to several reasons such as nonlinearity of the original time series), GARCH is applied to the residual errors with the hope of capturing the dependency between the residual errors. So, it is expected that APARCH can effectively cope with forecasting of complicated time series in a fashion that the residual errors are just white noise and follow normal distribution, and also the ACF of residual error sequence does not have periodicity, and its elements always remain within the predefined confidence interval.

Definition 4.5 Let $\{\boldsymbol{x}_t\}_{t\in\mathbb{Z}}$ be the time series and $\{\boldsymbol{\omega}_t\}_{t\in\mathbb{Z}}$ be an independent and identically distributed sequence of unit variance random noises, i.e. $\omega_t \overset{iid}{\sim} N(0,1)$. Also, let the estimated solution of ARMA(p, q) process be denoted by $\hat{x}_t$. Then, the residual error $e_t = \hat{x}_t - x_t$ is fed to GARCH(1, 1) model for further processing, using the equation below:

$$e_t = \sigma_t \, \omega_t \ ,$$

$$\sigma_t^2 = \alpha_0 + \alpha_1 e_{t-1}^2 + \beta_1 \sigma_{t-1}^2 \ ,$$

with initial condition $\sigma_1 = \dfrac{e_1}{\omega_1}$.

The trained model can be used then for forecasting the unseen values of $\hat{e}_t$, which together with $\hat{x}_t$ gives the APARCH model. The model can be simulated in R using either *tseries* or *rugarch* packages.

4.2.6 Dynamic linear model

Dynamic linear model (DLM) is a state-space model which is based on the concept of Bayesian regression. In this context, the parameters of the model are not only tuned

by using the captured data, but also depend on any form of information, such as expert knowledge, which can be represented probabilistically, as prior distribution. Consider a univariate time series $\{\boldsymbol{x}_t\}_{t\in\mathbb{Z}}$. Assume that the information available for prediction / forecasting at time t be represented as $\mathcal{I}_t$, and the model parameters (called states) vector at time t be $\boldsymbol{\mu}_t \in \mathbb{R}^n$. To take advantage from Bayesian regression concept, we need a probability model for time series, i.e. $\mathcal{P}r(x_t \mid \boldsymbol{\mu}_t, \mathcal{I}_{t-1})$, and a prior probability model $\mathcal{P}r(\boldsymbol{\mu}_t \mid \mathcal{I}_{t-1})$, then we need to find the posterior probability model $\mathcal{P}r(\boldsymbol{\mu}_s \mid x_t, \mathcal{I}_t)$, where $s > t$ for prediction scenario. The mathematical formulation of DLM can be given based on the mentioned concepts.

Definition 4.6 Let $\{\boldsymbol{x}_t\}_{t\in\mathbb{Z}}$ be the sequence of univariate collected data, in which at each time t, $\mathbf{x}_t \in \mathbb{R}$, and $\boldsymbol{\mu}_t$ be the states / parameters vector of the model. Then, the general DLM can be presented as:

$$x_t = \mathbf{f}_t^T \boldsymbol{\mu}_t + \nu_t \ ,$$

$$\boldsymbol{\mu}_t = G_t \boldsymbol{\mu}_{t-1} + \boldsymbol{\omega}_t \ ,$$

where the first equation is called the observation equation, and the second equation is called the system equation. Here, the $n \times 1$ vector $\mathbf{f}_t$ and $n \times n$ matrix G_t are known values, $\boldsymbol{\omega}_t \sim N(\mathbf{0}_{n\times 1}, W_t)$ and $\nu_t \sim N(0, \sigma_t^2)$, where W_t is a $n \times n$ time-varying covariance matrix, σ_t^2 is a time-varying variance, and also the prior distribution of system state is $\boldsymbol{\mu}_0 \mid \mathcal{I}_0 \sim N(\mathbf{m}_0, M_0)$, where $\mathbf{m}_0$ is a known $n \times 1$ vector, and M_0 is a known $n \times n$ covariance matrix . In this model, the covariance matrix W_t and σ_t^2 can be known or unknown, and the system state vector $\boldsymbol{\mu}_t$ is unknown.

Upon fine tuning, DLM can efficiently model a given time series, even when the series is not stationary. Considering this method together with ARMA, AR and MA will help us to conduct a good comparative study, and obtain reliable results for forecasting the road disturbance for vehicle suspension system control.

4.3 Data Acquisition Experiment

Collecting detailed experimental data for estimating the road roughness is a very time consuming and burdensome task. This is due to the fact that, to make sure the experimental data are of acceptable values, one should use highly accurate sensors, and also repeat the data collection for the same road through different independent trials to reduce the effects of uncertainty and sensors measurement noise as much as possible. Such facts have hindered engineers and practitioners from collectecting experimental data for road roughness. As mentioned, one of the main goals of this investigation is to come up with practical road

roughness forecasting models, and to do so, it is needed to have experimental dataset. In this regard, two different sets of data are used for the simulation.

In the rest of the section, the properties of the considered datasets as well as the required pre-processing for using them are explained. The first dataset is a well-known experimental road roughness data collected by engineers working on a Queensland governmental project, and the second dataset is obtained by the author's research group at Waterloo by using an accelerometer sensor.

4.3.1 Data obtained in Queensland

The data has been collected by engineers working on a Queensland governmental project, and contains averaged road roughness data over different road segments. The collected data are available at two levels of granularity on different road segment. For each of the data bases, the average roughness with respect to international roughness index (IRI) for the segment lengths of interest (100 m) has been reported. The detailed procedures and steps taken to collect data can be found in the website https://data.qld.gov.au/case-studies. Also, a related analytical study regarding the evaluation of pavement deflection values was conducted in [122]. Our assessment of the initial dataset indicated that there were some unreasonable spikes in the time series.

The initial time series which exactly shows what reported on the database is given in Figure 4.2. It can be seen that in some points, the IRI index goes to -100 which is not a standard value. By further investigation, it was discerned that those values represent the data-pairs for which the sensor could not measure the IRI index. So, those have been discarded to get the reasonable data in the form of time series. It is clear that the collected data is spatial as the IRI values are reported based on some distance measure. Keeping this in mind, for the sake of convenience, we rename the x-axis and consider the unit to be time, and the values series are treated as time series.

The filtered time-series is depicted in Figure 4.3 (a). As expected, the initial time series does not satisfy the first order and second order stationarity conditions, and thus, is not prepared to be used, and can degenerate the performance of standard forecasting systems which do their best when there is stationarity in the series. There are two main approaches to come up with a stationary times series. The first way is to decompose the stochastic time series into a trend (main signal) and a noise sequence. However, for practical time series such as road roughness, it is a very optimistic assumption to seek for an accurate trend and extract it from the original series. Therefore, the differencing strategy is taken into account to wash out the trend and make the series stationary for analysis. In this

way, different orders of differencing are performed on the time series. The results indicated that the first order differencing can acceptably wash out the trend and make the series stationary.

Figure 4.3 (b) indicates the first order stationary series, and as seen, it can be viewed as stationary time series. Therefore, it is used for training the forecasting models. Also, note that to make sure the first order differenced times series is the best choice, the original time series was differentiated for more times, and it was observed that differentiation orders greater than one increase the variability of the resulting differenced signal which is not desirable.

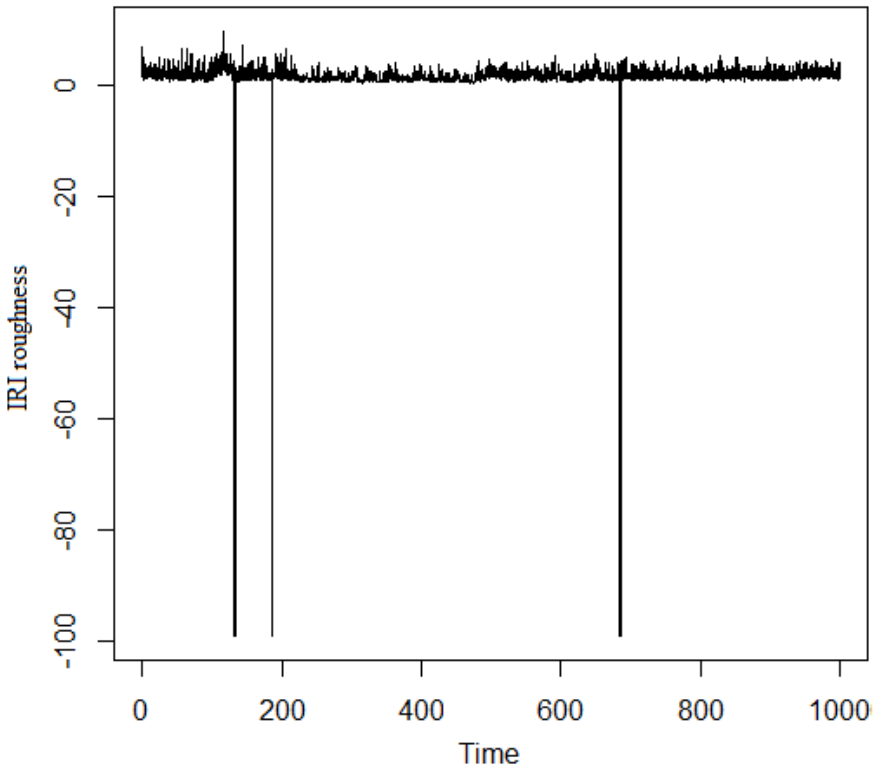

Figure 4.2: The initial time series with spikes (due to lack of information).

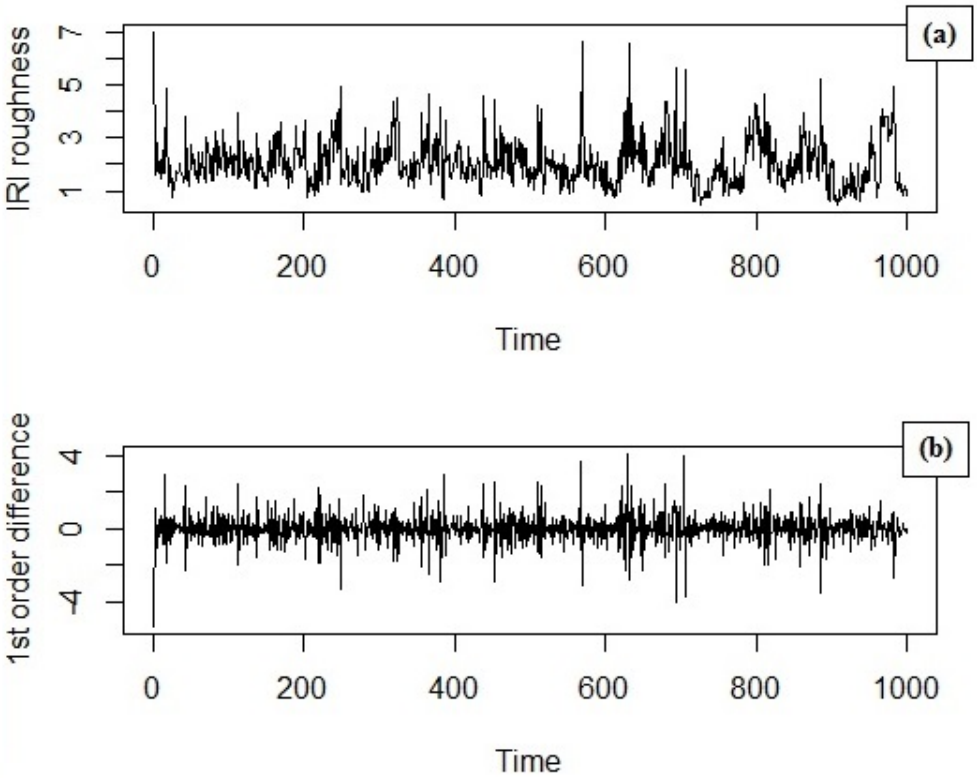

Figure 4.3: Time series of (a) IRI road roughness, and (b) first order differenced version of time series.

4.3.2 Data obtained in Waterloo

One of the difficulties of the Queensland experimental data is that the road roughness is reported in terms of IRI index, for which there is not an inverse function to get the road displacement information (which is required for the controller). Therefore, the author's research group conducted a complementary experimental study at Waterloo using an accelerometer sensor, which measured the vertical acceleration of the vehicle, and could be used to get the displacement profile.

The dataset of Waterloo experiment was collected on a Honda Civic automobile, using an accelerometer software installed in a Black-Berry passport mobile, with Black-Berry OS 10.3.3.1 operating system, and 3 GB memory in a 160 *sec* period which will be used to evaluate the performance of forecasting systems.

Figure 4.4 (a) indicates the measured vehicle acceleration over 160 *sec* (from 11:53:52 to 11:56:32). After reducing a constant value (0.25) from the profile, the mean adjusted time series seems to be approximately stationary (see Figure 4.4 (b)). To have a more stationary time series, different differenced version of the mean adjusted profile is determined. Based on the experiment, the author realized that first order differenced profile can be viewed as stationary time series. Therefore, the first order differenced profile which is shown in Figure 4.4 (c) is used for training the forecasting methods.

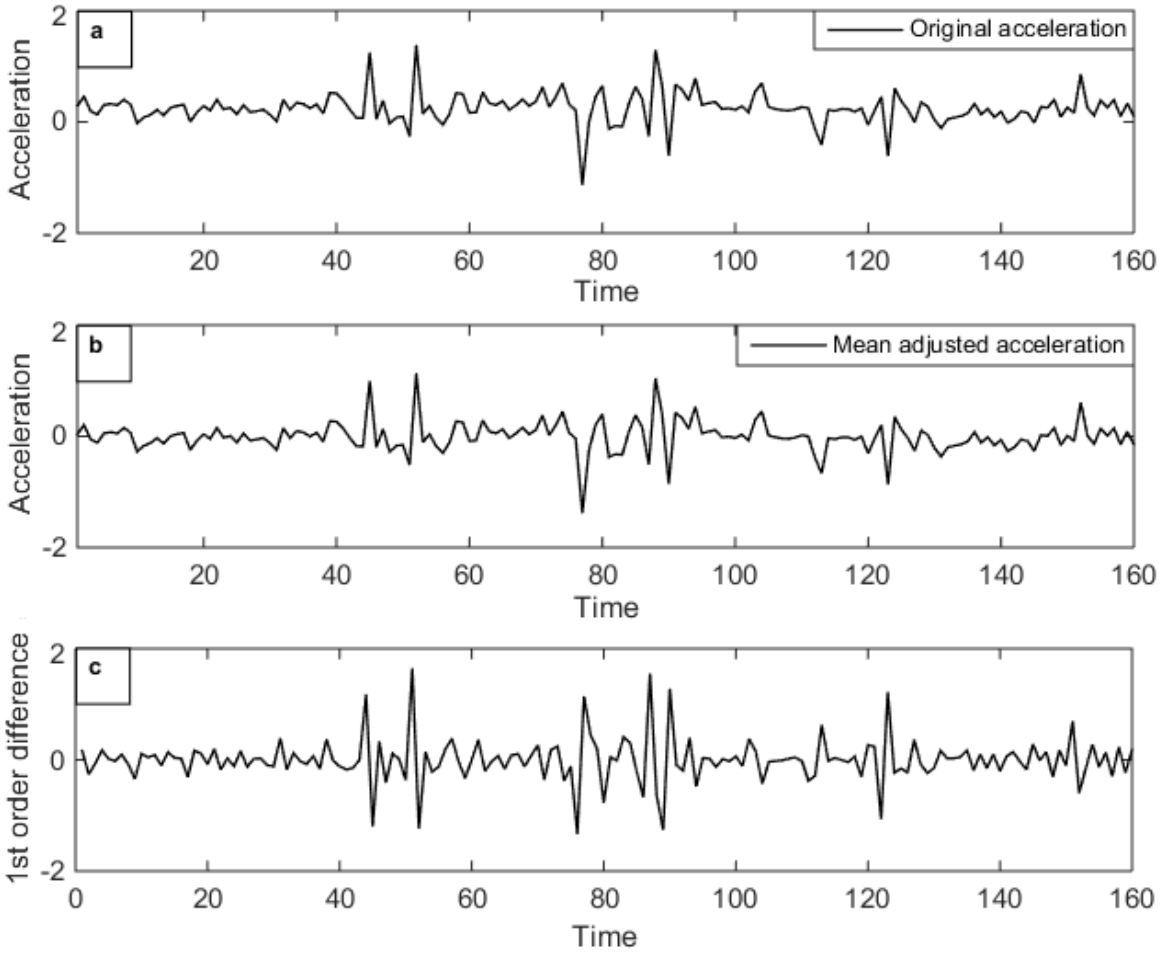

Figure 4.4: Time series of (a) original vehicle acceleration, (b) mean adjusted vehicle acceleration, and (c) first order differenced version of acceleration.

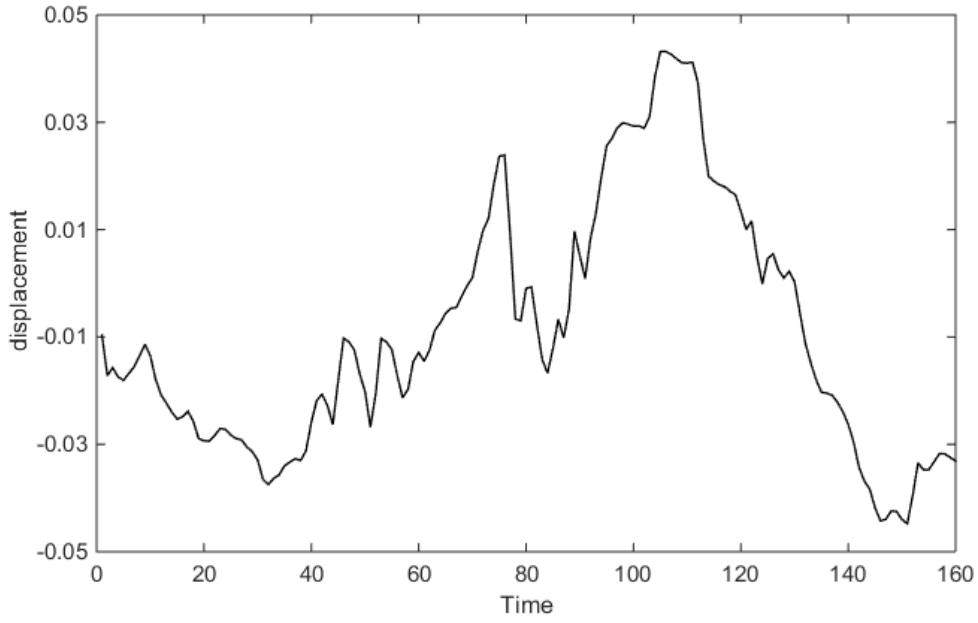

Figure 4.5: The corresponding road displacement.

Also, the vehicle displacement for the measured acceleration profile is depicted in Figure 4.5. In this way, after forecasting the vertical acceleration, the estimated value will be transformed to displacement, and used at the heart of control state-space model, as road disturbance. Interestingly, the obtained value matches with standard road displacements which are assumed by designers for developing controllers.

4.4 Simulation Results

This section is given in two sub-sections. Firstly, the detailed description of the parameter settings as well as the steps taken to conduct the numerical study are presented. Thereafter, the numerical results obtained through simulation are scrutinized.

4.4.1 Simulation setup and parameter settings

As mentioned in Chapter 3, the suspension system senses the road disturbance on both the front and rear axles. Indeed, the front and rear tires sense exactly the same disturbance

profile with a time delay, which can be calculated as a function of vehicle cruise speed and acceleration. Thus, the forecasting method is just used once, to forecast the future road roughness profile which will be sensed by the front tire, and the same value is fed to the rear tire (and also to the state-space model of the rear axles) with a time delay.

For using AR, MA and ARMA methods, the Box-Jenkins strategy is taken into account, by following the model identification / model selection, parameter estimation, and model checking / diagnosis stages. Note that, as stated before, these methods work perfectly when times series are stationary. So, since for the data used in this case study, the observed non-stationary time series can be viewed as the linear transformation of stationary processes, it would be possible to take differences and wash out the trends, and then, use the above methods as the variants of general autoregressive integrated moving average, ARIMA(p, d, q), process.

For model-order selection, graphical visualization by means of autocorrelation function (ACF) and partial autocorrelation function (PACF) can be used. Although such a strategy can be useful for selecting the order of MA and AR processes, it cannot afford an exact result for ARMA process. So, making inference regarding ARMA model is difficult, and even if an approximate model can be taken, it is impossible to claim that this is the best fit ARMA for a given time series. Therefore, here, the author uses the Akaike information criterion (AIC) to get the best model for ARMA. AIC can be formulated as below [121]:

$$AIC := -2\ell(\hat{\boldsymbol{\beta}}) + 2\kappa \ ,$$

where $\ell(\hat{\boldsymbol{\beta}})$ is the maximum log-likelihood value, κ is the number of parameters in the fitted model, and $\hat{\boldsymbol{\beta}} \in \mathbb{R}^k$ represents the estimated parameter vector. For the current simulation, AIC objective function is optimized by Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm [2].

Also, for comparing the performance of the rival considered methods, mean square estimate (MSE) and standard deviation (*std.*) are taken into account. For reporting the MSE values, it is common to divide the data into training and testing sets and check the performance of the fitted model for testing set. A better way is the use of k-fold cross-validation. However, the current case study deals with time series, which are intrinsically ordered data. This makes the use of cross-validation problematic because some patterns may emerge at a certain point (which could potentially be neglected at some of the stages of k-fold cross-validation training process), and be crucial for forecasting the step-ahead points. Therefore, an approach which is more principled for time series, and is called k-fold forward chaining is taken into account. Let's say $k = 3$, then the procedure includes dividing the time series into $k = 3$ segments, and doing (a) fold 1 (training segment 1 and

testing N-step ahead), (b) fold 2 (training segments $\{1, 2\}$, and testing N-step ahead), (c) fold 3 (training segments $\{1, 2, 3\}$, and testing N-step ahead), and finally reporting the average MSE and *std.* values. Such a process also lets us evaluate the dependency of the forecasting models on the required size of training dataset. For this study 10-fold forward chaining is taken into account. Also, due to the properties of the dataset and the nature of the predictive control problem at hand, $N = 5$ is selected for simulation.

For model diagnosis, some important visualization tools such as quantile-quantile plot (QQ-plot) of residual error, correlation of fitted model and data, and confidence interval (CI) are taken into account [120]. For each of the models, the future predicted values are reported in the form of a mean value and a 95% CI bound to check the robustness of the system. Note that, during the analysis, the time lag between two measured values at times t and s is denoted by $h =\mid t - s \mid$.

Also, for training APARCH model, after fitting a proper ARIMA model, the residual errors are calculated, and the resulting sequence is fed to GARCH(1, 1) for further processing. In the current study, Quasi Newton-Raphson algorithm is used for calculating the optimal model parameters of GARCH.

As pointed out, it is necessary to use both of the considered datasets for simulation and evaluation of the potential of the adopted forecasting / prediction methods, since they are experimental and represent two real road profiles. Note that, since the second dataset, (which has been collected by the author's research group at Waterloo) has been reported based on vertical acceleration measure, and can be transformed to road displacement, it is more suited to be used at the heart of controller. Therefore, the main focus of the current simulation is to find out an authentic predictor for that dataset, and eventually use it as the road disturbance prediction module at the heart of LBMPC in Chapter 8.

All of the simulation are conducted in MATLAB and R software on a Pentium IV DELL laptop, with Windows 7 operating system, Intel Dual core 2.2 GHz, and 2 GBs RAM.

4.4.2 Analysis of data obtained in Queensland

As mentioned, before proceeding with fitting procedure, that would be a wise choice to carry out some primary analysis on the dataset. This gives us some clues to seek for more appropriate versions of forecasting algorithms. Therefore, instead of blindly using of autoregressive (AR), moving average (MA) and autoregressive moving average (ARMA) methods, here, the behavior of the times series is analyzed in terms of auto-covariance function (ACF) and partial auto-covariance function (PACF) to see which model is a good fit for analysis and fitting. First of all, the resulting difference time series has been tested in

terms of quantile-quantile correlation to find out to what extend it shares properties with Gaussian distribution. Figure 4.6 illustrates the QQ-plot of the time series. As seen, QQ-plot shows a good correlation between data and theoretical quantiles, which is a favorable phenomenon, and helps us conveniently proceed with the analysis.

Figure 4.6 also unveils the ACF and PACF values for the road roughness data. It can be concluded from the obtained information that PACF has a tail-off type decaying property (which makes the data a good fit for MA process). Also, the ACF plot approximately cuts-off at lag $h = 2$ (at $h = 2$ still coincides the CI bound), which is interestingly another property of MA process. Given the mentioned facts, one can infer that more emphasis should be put on designing a forecasting model which is based on the linear combination of previous white noise signals (MA(1) or MA(2) may be proper choices). Also, to be more accurate and carry out a more comprehensive investigation, it should be taken into account that one cannot precisely claim that the ACF values entirely match the properties of MA model, as the ACF value does not exactly cuts-off at a point to zero. That is to say, after $h = 1$, there is a strong suppression in the ACF value. But, it is not exactly equal to 0 (maybe due to the noises or maybe due to the nature of the data), rather it gives some values in the vicinity of 0. Such a fact also suggests considering ARMA model for this dataset, provided that it is better to seek for ARMA(p, q) processes with order $q = p$ or $q > p$. As mentioned, for ARMA, an optimization is conducted using AIC and considering MSE error. Other than that, to have more comprehensive experiment, a sensitivity analysis by considering some possible ARMA(p, q) variants is carried out, and the results are compared to the one obtained by optimization, keeping in mind that due to the properties of the data, that would not be a wise choice to increase the order of AR part significantly.

In view of the derived conclusions, for the trial and error experiment, MA(1), MA(2), MA(3), and MA(4) are used to fit the data. Also, ARMA(1, 1), ARMA(1, 2), ARMA(2, 2) and ARMA(2, 3) are considered for further analysis (AIC optimization for ARMA is also used). Firstly, the performance of MA for fitting a good forecasting model is evaluated, and then, the results pertaining to fitting an ARMA model is presented.

The statistical results obtaining after training MA models are summarized in Figure 4.7. As seen, QQ-plots of all of the considered models are acceptable in the sense that they are in good correlation with theoretical quantiles. This means that MA is a good choice for interpolation on the captured data. Each of the MAs predict 5-step ahead values of the series (as mentioned, MAs are short-term forecasters). Also, the other reason is that LBMPC controller usually uses a prediction horizon length of 5 (i.e. the five future signals) to make an optimal controlling decision. The second plot for each of MAs shows the correlation between the model values and the experimental data. As seen, the model

and experimental values are in good agreement. Also, MA models are measured with respect to MSE performance metric obtained by 10-fold forward chaining. It can be seen that MSE index for MA(2) is less than the other MA counterparts.

Also, the last two figures show the performance of MAs for the estimation of future unseen values. For each of the models, the future values are predicted and reported in the form of a mean value and a 95% CI bound to check the robustness of the system. The fourth (last) subplot of each model includes the considered intervals and means as well as the actual experimentally evaluated data. It can be seen that all of the models could efficiently forecast the future values of the time series, and also the actual values are very close to the mean values of estimated intervals.

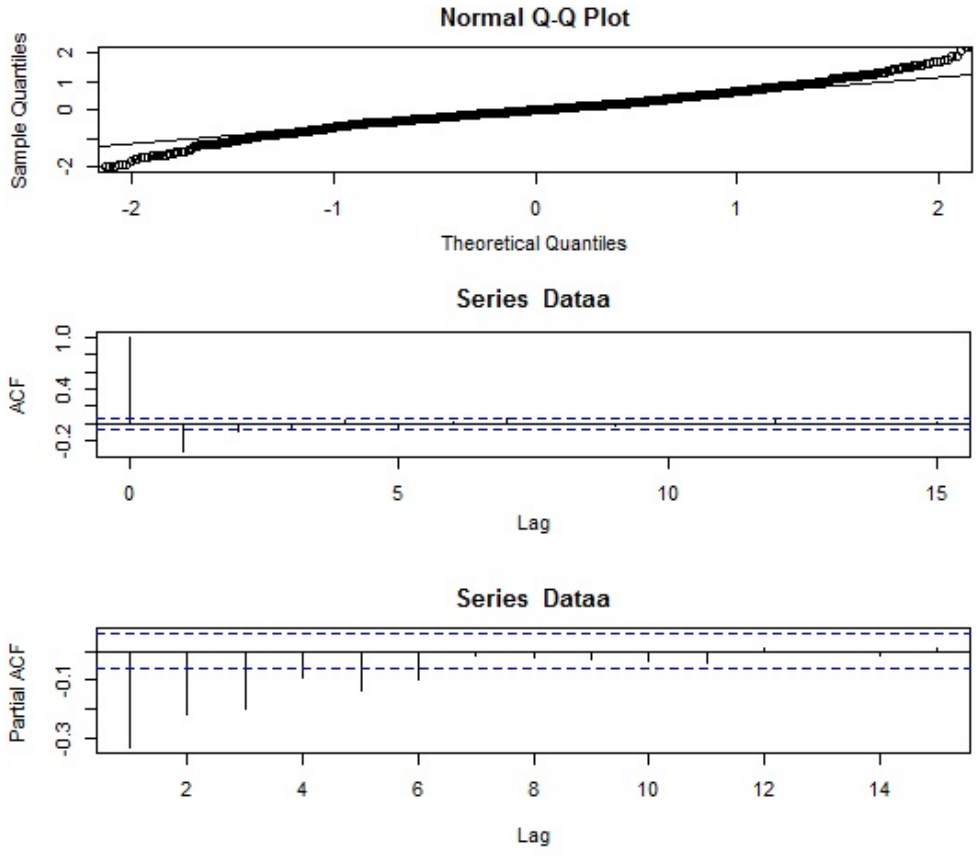

Figure 4.6: Analyzing the Queensland roughness data in terms of (a) quantile-quantile plot, (b) autocovariance function, and (c) partial autocovariance function.

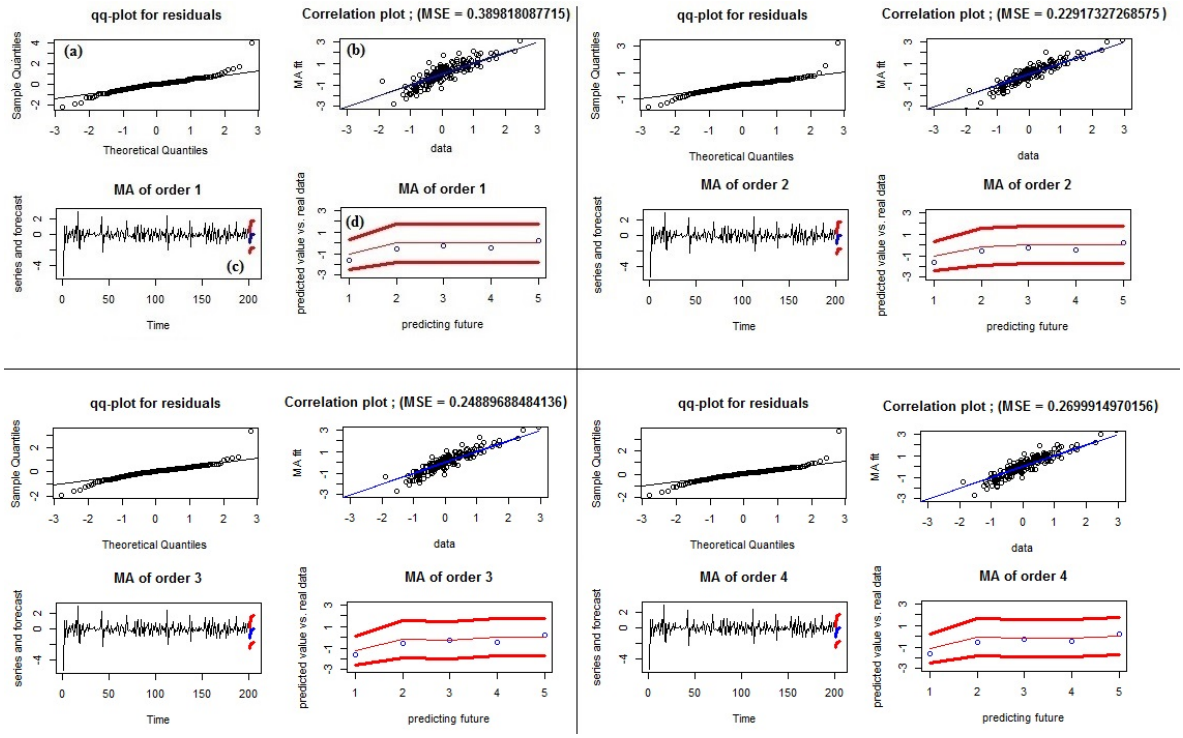

Figure 4.7: Model diagnosis results for fitting models by different MAs for Queensland dataset.

By repeating the experiments, it was observed that the actual future signals rarely violate the estimated CI. This implies the good performance of MA models for the considered forecasting task (as expected). Obviously, the ACF values in Figure 4.6 approximately cuts-off to zero after lag 2, which somehow augurs the standard behavior of MA(2). This can also be verified in Figure 4.8 which compares the actual and model estimated signals. As seen, the model has a very good performance, and its estimated values are in a very good agreement with actual experimental signals.

Also, some standard statistical tests on the residuals of MA(2) are conducted for further analysis. Figure 4.9 (a) indicates the residuals of MA. Figure 4.9 (b) evaluates the performance of MA(2) by measuring the ACF values for residual signals. It can be seen that the values are in most of the times within the 95% CI. This shows that the captured residuals are approximately Gaussian white noises and MA(2) does an acceptable fitting on time series.

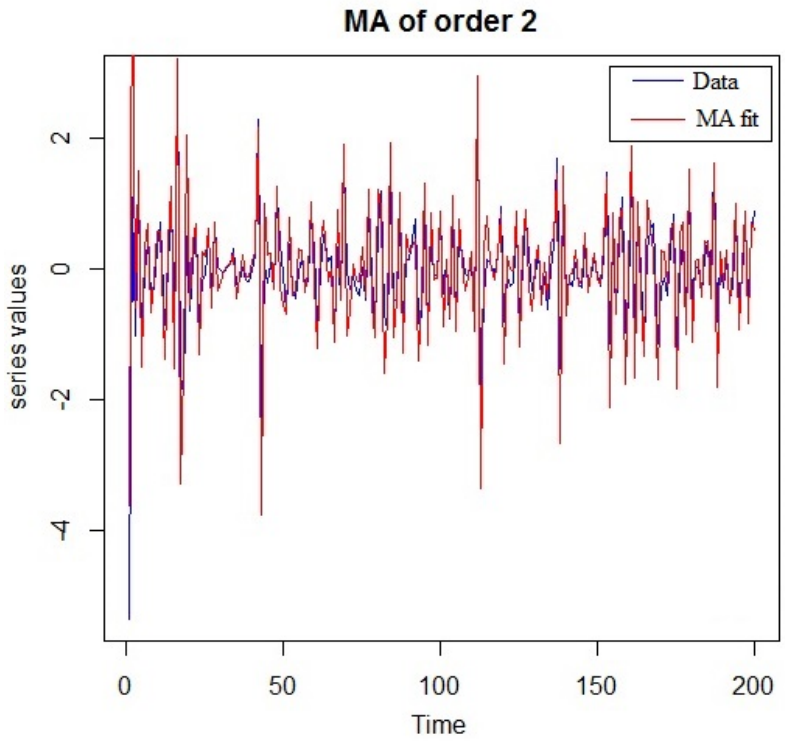

Figure 4.8: Experimental time series data vs. MA(2) prediction for Queensland dataset.

The last test is known as Ljung-Box test. This test can be performed on residuals of any variant of fitted ARIMA process, including MA. The test is performed under the null hypothesis that the residuals from any ARIMA (p, d, q) have no autocorrelation. The rejection criterion (critical value) comes from Chi-squared distribution with degree of freedom equals the considered lag. It can be seen that the obtained values approximately do not violate the threshold (except $h = 9$ and 10) which means that the calculated errors have no (or at-least trivial) dependence, and there is no strong evidence to reject the null hypothesis.

Also, the estimation errors for MA processes of different orders are obtained in an additional experiment to get some clues about the veracity of the conducted ACF / PACF analysis (results of Figure 4.6), and the derived conclusions. As mentioned previously, based on the properties of ACF and PACF plots, it was concluded that Queensland dataset is best suited to be modeled by MA(2), as the ACF approximately cuts-off after lag $h = 2$.

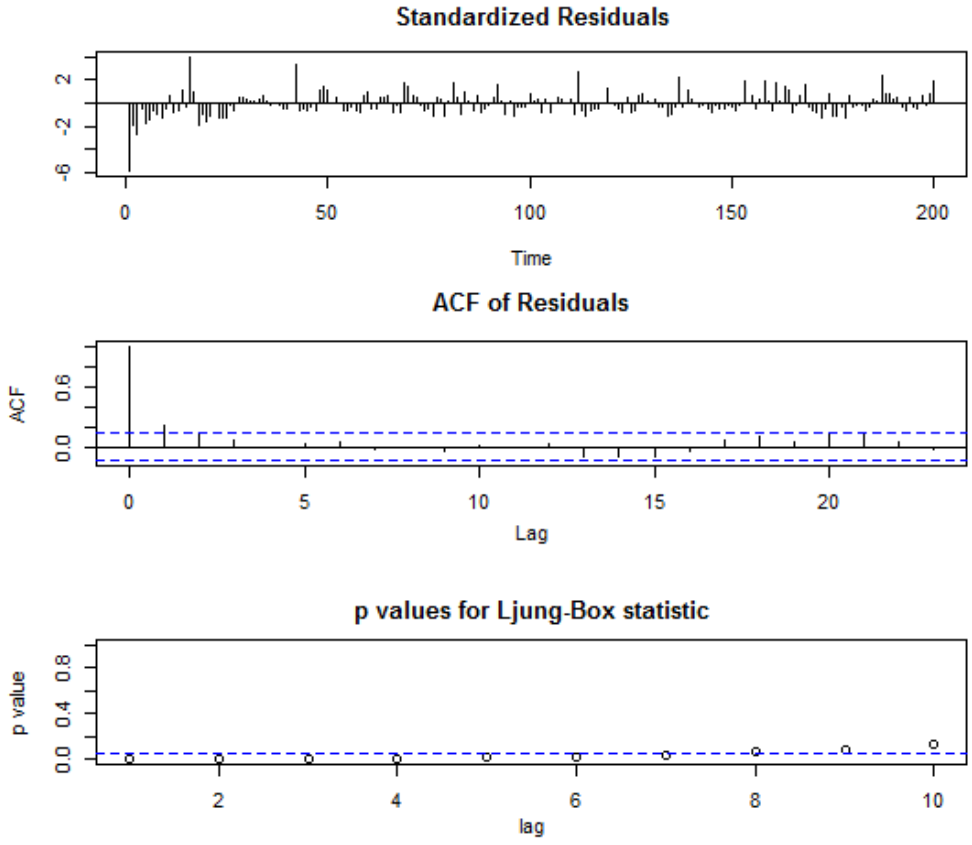

Figure 4.9: Further model diagnosis results for MA(2).

The results of sensitivity analysis in terms of MSE and the order of MA is shown in Figure 4.10. It can be easily verified that the most optimum performance in terms of MSE index is achieved for MA(2). Also, the sensitivity results reveals that for MA of orders greater than 4, the estimation error increases significantly. This also is in consensus with our conclusion from data ACF-PACF analysis that the order of MA for this specific data should not be large (it should be something around 1 to 3).

To further investigate the potential of linear process predictors, and also based on the results of data analysis, it is intended to fit a number of ARMA models to the data. The same experiments are repeated using the aforementioned ARMA models. Figure 4.11 presents the results of statistical test on the ARMA models.

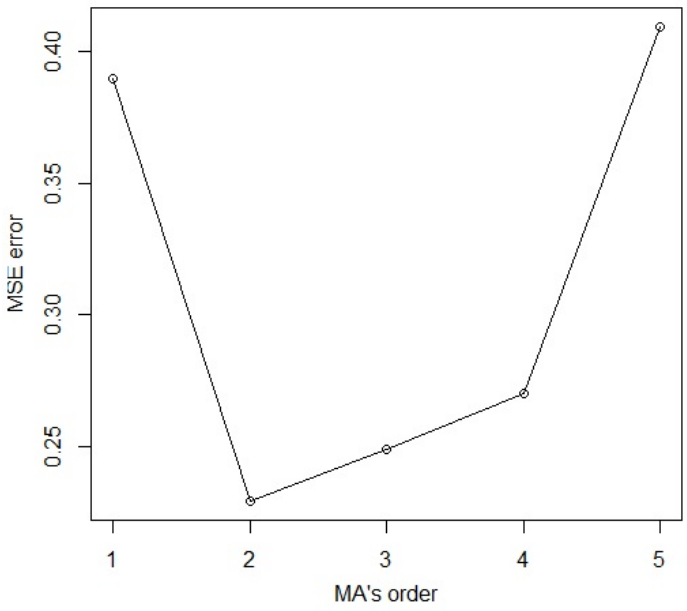

Figure 4.10: MSE error of MA models of different orders obtained by 10-fold forward chaining.

It can be seen that the considered ARMA models are also capable of handling the fitting task. However, by taking a fair comparison with MA counterparts, it can be inferred that their MSE error are higher. Also, they have a much more complex structure compared to MA models. This leads to the conclusion that, for Queensland dataset, MA could be a better choice to be used as prediction module for LBMPC, as online calculation speed and less computational complexity is of highest priority.

The obtained AIC, MSE and *std.* values for ARMA model selection is shown in Table 4.1. The AIC optimization suggests ARMA(1, 2) as the most optimum structure with $\ell(\hat{\boldsymbol{\beta}}) = -210.6$ and AIC $= 427.26$. Also, the 10-fold forward chain MSE and *std.* values for ARMA(1, 1) and ARMA(1, 2) models are better than the other two models, i.e. ARMA(2, 2) and ARMA(2, 3). As seen, the estimation power of this two models are very close to each other, and based on the AIC results, ARMA(1, 2) dominates ARMA(1, 1) because of having lower AIC value. The selection of ARMA(1, 2) is also in agreement with MA(q) model selection experiments, and it seems that $q = 2$ is the best choice for Queensland dataset. Indeed, this is inline with the findings of ACF / PACF analysis of the time series (results of Figure 4.6).

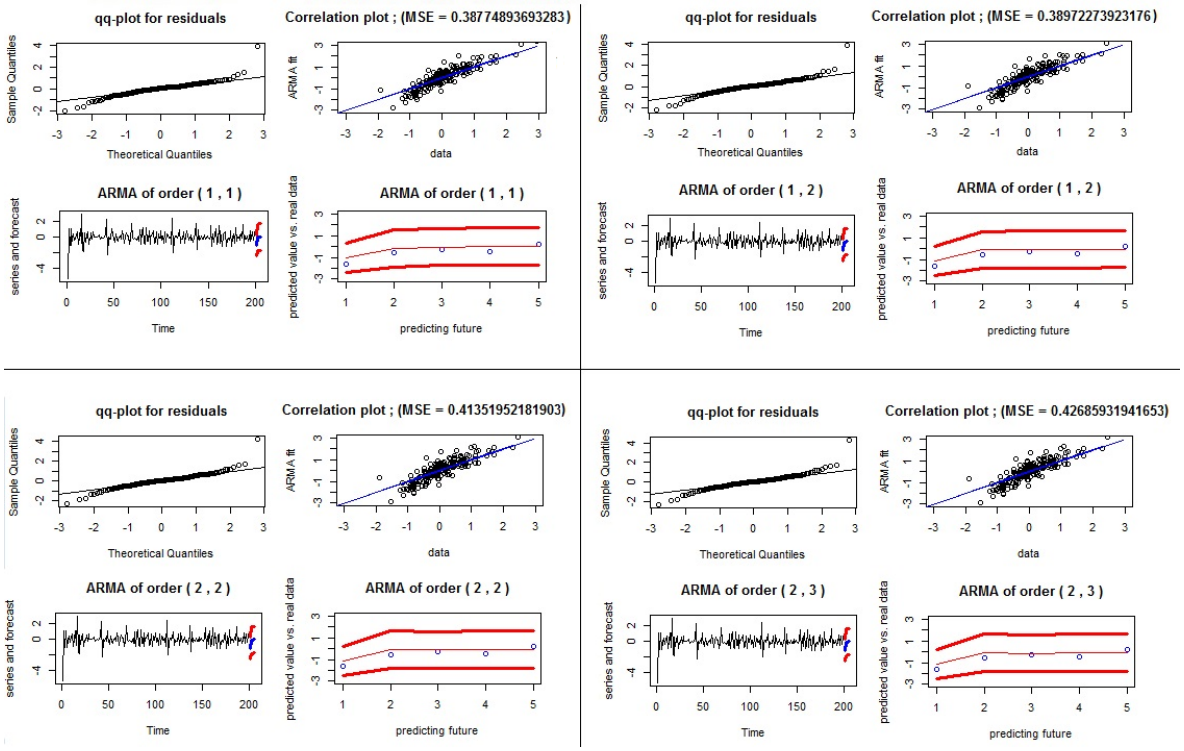

Figure 4.11: Model diagnosis results for fitting models by different ARMA processes for Queensland dataset.

Table 4.1: Results of optimization for ARMA models for Queensland dataset

Parameters	$\ell(\hat{\boldsymbol{\beta}})$	**AIC**	**MSE**	**std.**
$p = 1, q = 1$	-212.6	429.20	0.3877	0.0440
$p = 1, q = 2$	-210.6	**427.26**	0.3897	0.0440
$p = 2, q = 2$	-210.6	429.20	0.4135	0.0454
$p = 2, q = 3$	-210.6	431.20	0.4268	0.0454

Figure 4.12 compares the values estimated by ARMA(1, 2) and the actual time series. It can be seen that the signals acceptably match each other. Figure 4.13 shows the statistical tests on residuals obtained by ARMA(1, 2). By comparing the results of Ljung-Box test for ARMA and MA models, it can be seen that MA does a better job, as ARMA(1, 2) violates the threshold in more points. This means that the residuals of ARMA(1, 2) model

have higher dependency compared to those of the MA(2) model.

To complete the experiment, more advanced forecasting models are also taken into account. For Queensland road roughness time series, some sort of volatility clustering can be observed in the original data as well as the first differenced version (see Figure 4.3). Also, to be very precise, it can be stated that ACF values of the model diagnosis results for both MA(2) and ARMA(1, 2) are at some lags significant or very close to the confidence bound (this can be also verified by the results of Ljung-Box test). Such observations suggests the need for using a much more sophisticated modeling scenario (maybe a nonlinear model) to get better results.

To cope with such a diminishment, we seek for a remedy to improve the statistical results of forecasting. This is done by using APARCH model which takes advantage from both volatility and linear process theories.

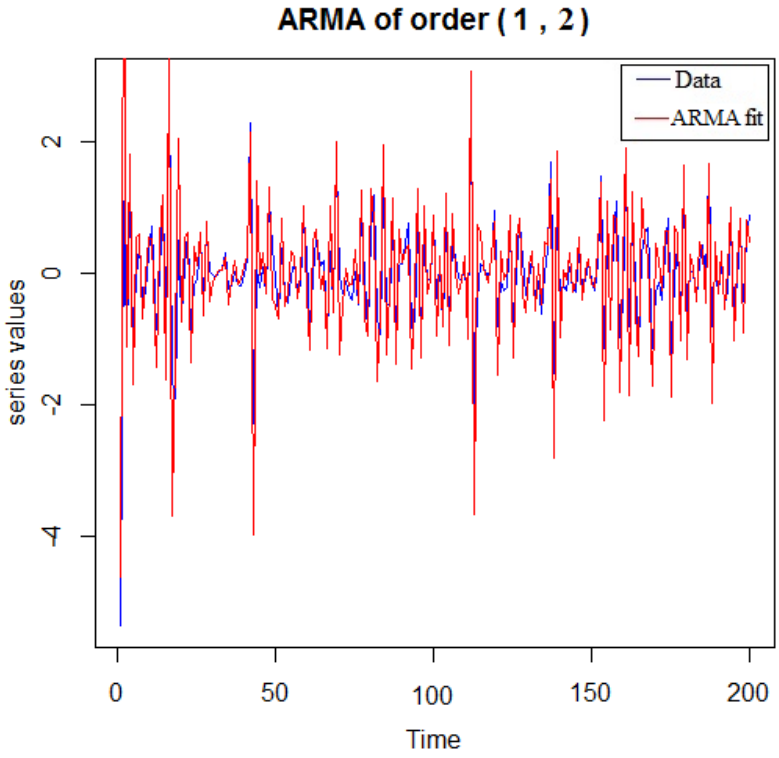

Figure 4.12: Experimental time series data vs. ARMA(1, 2) prediction for Queensland dataset.

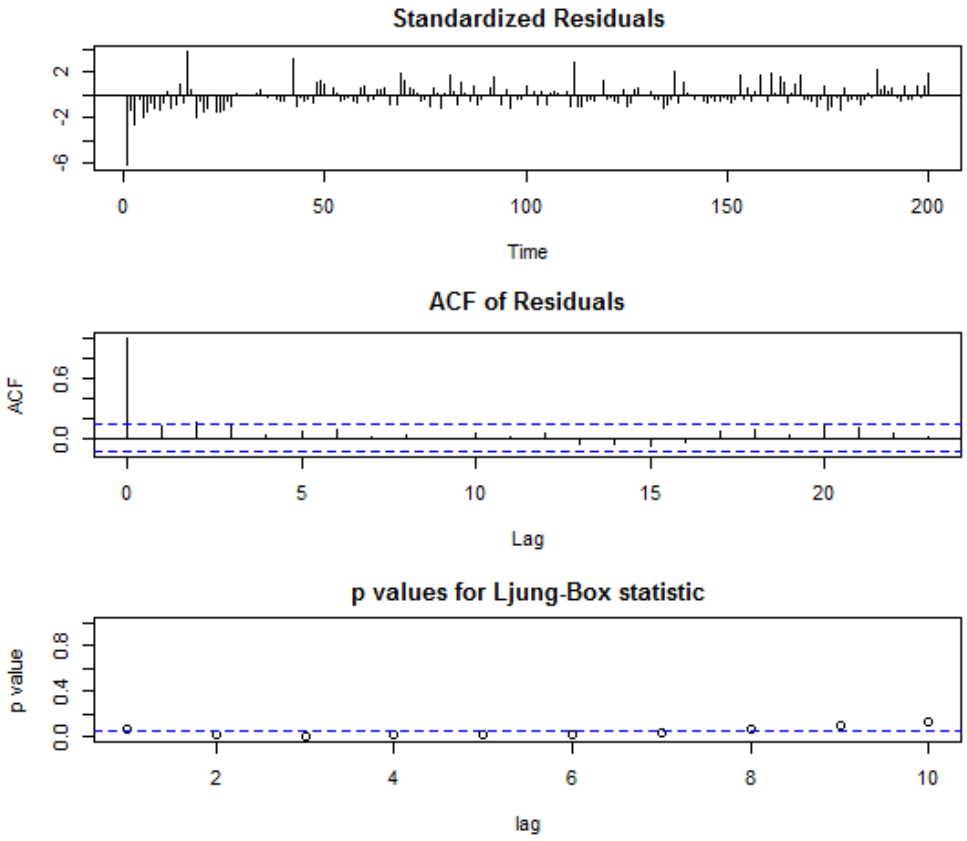

Figure 4.13: Further model diagnosis results for ARMA(1, 2).

Thanks to the powerful tseries tool box of R software, a model is developed which firstly uses ARMA(1, 2) to fit a model, and then, applies GARCH(1, 1) to residual errors. Note that just like DLM, upon fine tuning of APARCH, it can be used for long-term forecasting, and could be advantageous for the current application. The ACF-square of the model obtained by GARCH(1, 1) is shown in Figure 4.14 (*a*). It obvious that the ACF values always remain within the CI bound. Also, the ACF of APARCH model is shown in Figure 4.14 (*b*). The obtained result indicates that except a significant value at lag $h = 1$, the ACF at the other lags always falls within the CI and is not even close to the bounds. By comparing the ACF of residuals of APARCH to ARMA(1, 2) and MA(2), it can be seen that APARCH has a superior performance in terms of removing the dependency of residual sequence. Also, APARCH yields MSE of 0.3678 and *std.* of 0.0428, which are both better than the previous obtained results.

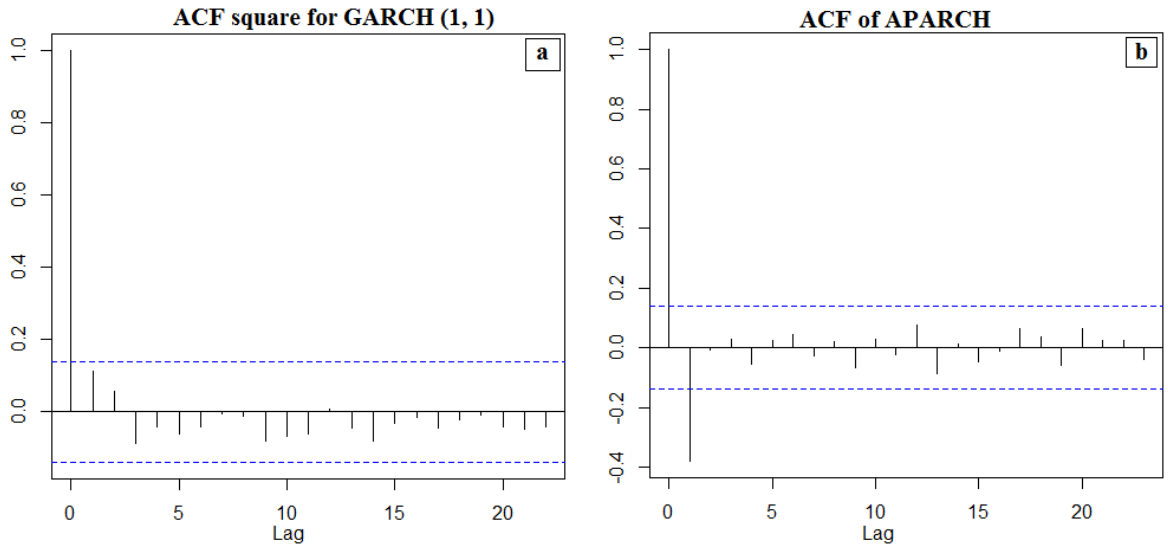

Figure 4.14: ACF plots for (a) GARCH and (b) APARCH for Queensland dataset.

Figure 4.15 compares the values estimated by APARCH and the actual time series. All in all, the results of conducted simulation indicate that APARCH can also be a good choice for our application. It is worth pointing out that it has a much complex structure compared to ARIMA methods, and also requires more computational time for estimating the parameters, and due to the nonlinearity of GARCH, more sophisticated optimization process is needed for its fine tuning. Such observations make the applicability of APARCH for online applications and incremental learning a little bit questionable, and care should be taken when using it.

The final model used for forecasting of Queensland road roughness time series is DLM. Note that for DLM, the original version of the time series is taken into account, which is non-stationary. Based on the experiments, it was realized that the best variant of DLM is the simple DLM with one parameter and $f_t = g_t = 1$ (the original vector $\mathbf{f}_t$ and matrix G_t are simplified to one array for our case). Also, the optimization suggests using $V = 2.59$ and $W = 3$. Note that the initial distribution of the parameter is set by taking m_0 equal to mean of data, and M_0 equal to variance of dataset. The obtained MSE error and *std.* values are 0.5936 and 0.0544, respectively, which are not as good as the other rival methods. The model diagnosis results are summarized in Figure 4.16. As seen, the residual error sequence of the fitted model has an acceptable behavior, and seems to approximately follow normal distribution.

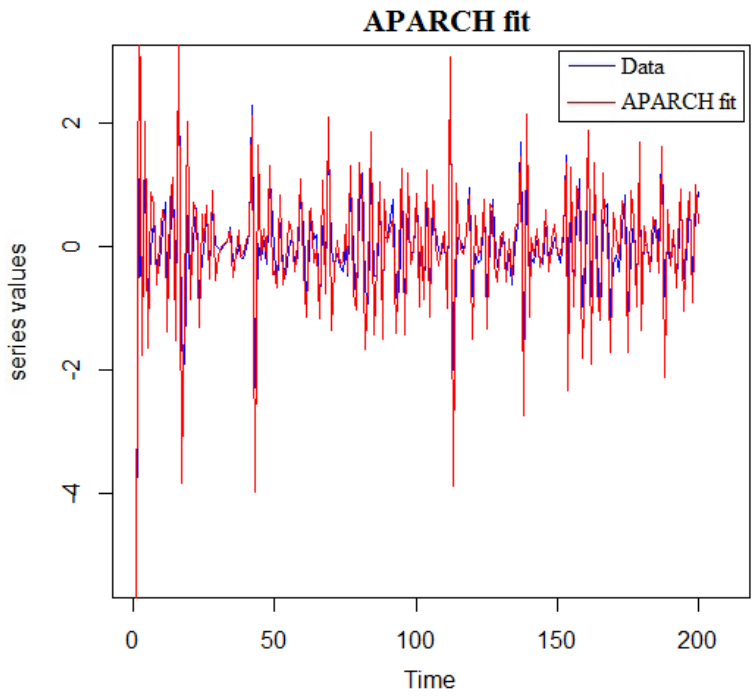

Figure 4.15: Experimental time series data vs. APARCH prediction for first order differenced version for Queensland dataset.

Also, the ACF of residuals remains within the CI for most of the considered lags. The weakness of the model lies in its estimation capability, and as can be seen from the correlation plot, the DLM estimation and time series data are not in a very good correlation. Also, for having a fair comparison with the other methods (for which the estimated series are reported in terms of first order sequence), the first order differenced version of the data and DLM prediction is depicted in Figure 4.17.

By comparing the result with Figure 4.8 and Figure 4.12, one can find out that the estimation power of the other rival methods is better than that of DLM, and at-least for this dataset, the other methods can be more beneficial. One reason could be the lack of enough information for efficient determination of prior distribution of the system state, which is a common drawback of all Bayesian regression methods. Having said that DLM may not outperform the other methods, but still does an acceptable prediction job, and in particular, its results are better than the other methods (except APARCH) in terms of capturing the dependency of the residual error.

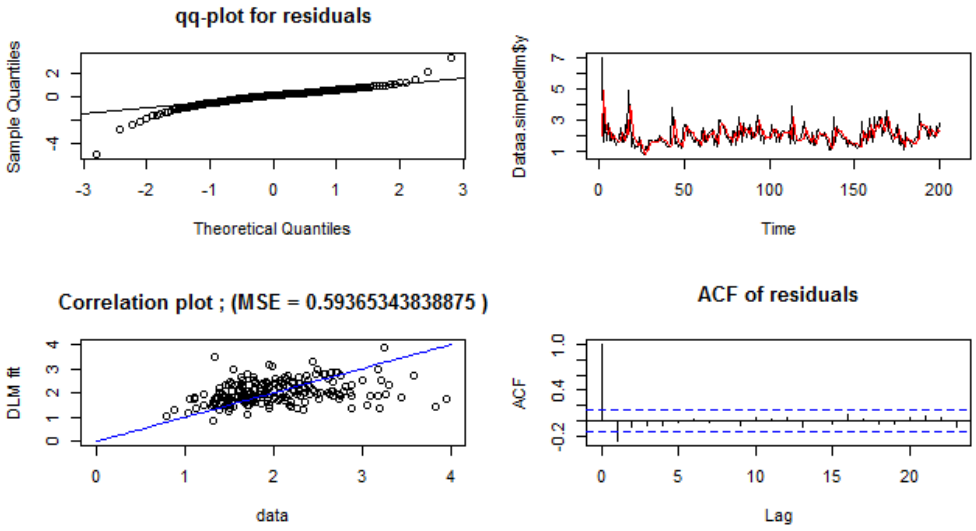

Figure 4.16: Model diagnosis results of DLM fit for Queensland dataset.

4.4.3 Analysis of data obtained in Waterloo

To find out which types of forecasting models are more appropriate for Waterloo dataset, and also, to have a visual inference regarding the proper (or relevant) orders of ARIMA processes, a statistical test is carried out on the first order differenced version of dataset. The conducted experiment includes quantile-quantile, ACF and PACF analysis. The results of the experiment are summarized in Figure 4.18. It seems that the theoretical and sample quantiles are in an acceptable correlation with some deviation on both tails. Our experiment indicated that further differencing can drastically increase the variability of series as well as the deviation on both tails, and thus, the first order difference is an acceptable choice. Also, the ACF plot cuts-off at lag $h = 2$, and clearly after lag $h = 1$, the other ACF values are within the confidence bound. Also, PACF shows an exponential decay which together with ACF results suggests the use of MA process for fitting. Also, it can be seen that at some point, the ACF and PACF values raise from 0 (but still remain within the CI) and again cut-off to 0. This could be because of the measurement noises or other sources of disturbance which contaminate the dataset. Such a fact substantiates the need for using other versions of forecasting methods to get a better view on the most appropriate model for Waterloo dataset. Therefore, ARMA process can also be a good choice for the current experiment, again, with focus on selecting models with order $q = p$ or $q > p$, and also not setting large values for p. Also, APARCH and DLM models are

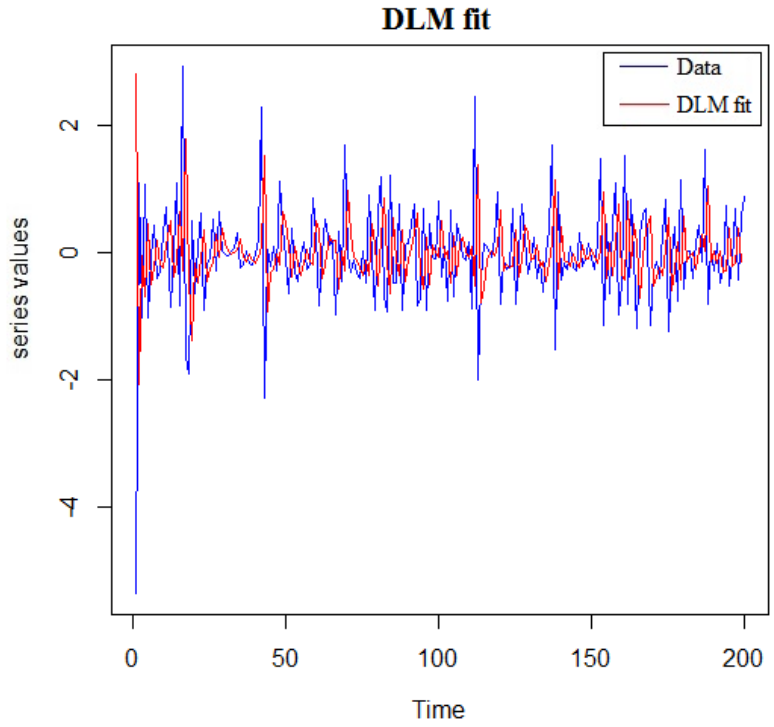

Figure 4.17: Experimental time series data vs. DLM prediction for first order differenced version of Queensland dataset.

good choices for Waterloo dataset.

As mentioned, for the considered variants of ARIMA processes, i.e. MA and ARMA, the analysis is conducted in the light of Box-Jenkins procedure which includes, model selection, parameter estimation, and model diagnosis. Different variants of MA and ARMA, namely MA(1), MA(2), MA(3), and MA(4), ARMA(1, 1), ARMA(1, 2), ARMA(2, 2) and ARMA(2, 3) are used for analysis and the best models are selected with respect to AIC, MSE, *std.*, and other model diagnosis experiments. Also, due to priority of using simple and efficient models at the heart of LBMPC controller, and the possible need for incremental learning in a real-time fashion, simple DLM is selected as the most appropriate version of Bayesian regression models.

The statistical results obtaining after training MA models are presented in Figure 4.19. By checking the QQ-plots of the simulation, one can realize that the correlation between theoretical and sample quantiles for residual error sequence is not that satisfactory. In

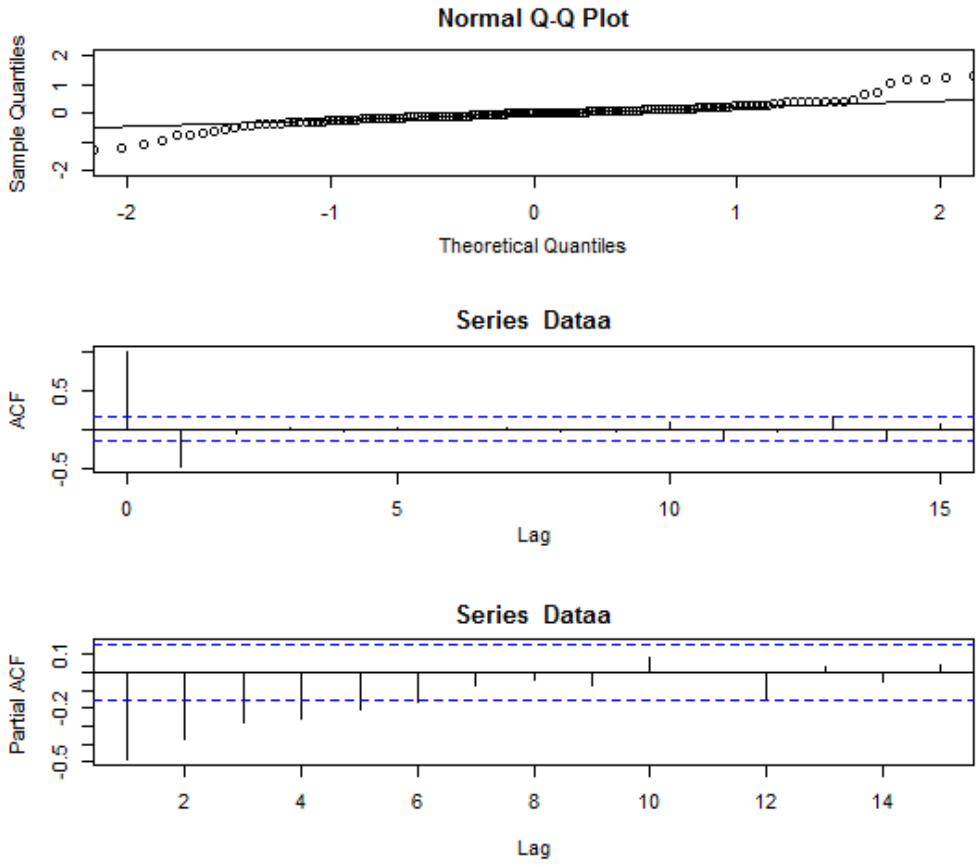

Figure 4.18: Results of statistical tests on Waterloo roughness data, (a) quantile-quantile plot, (b) autocovariance function, and (c) partial autocovariance function.

particular, remarkable deviation can be observed at the both tails for all of the considered MA models. The MSE values obtained by 10-fold forward chain suggest that MA(3) has a superior performance comparing to the other values. Also, for all of the cases, the N-step ahead forecast violates the 95% percent CIs. Putting the results altogether brings us to the conclusion that MA is not a very efficient model for Waterloo dataset. Also, it can be stated that among the considered MAs, MA(3) outperforms the other variants.

Figure 4.20 indicates the complementary model checking results for MA(3). There is an obvious pattern in the residual error sequence, which shows the dependency of the obtained errors. This pattern can be also observed in the ACF of residuals plot, though it seems that the obtained values remain within the confidence bound. Also, the Ljung-Box test

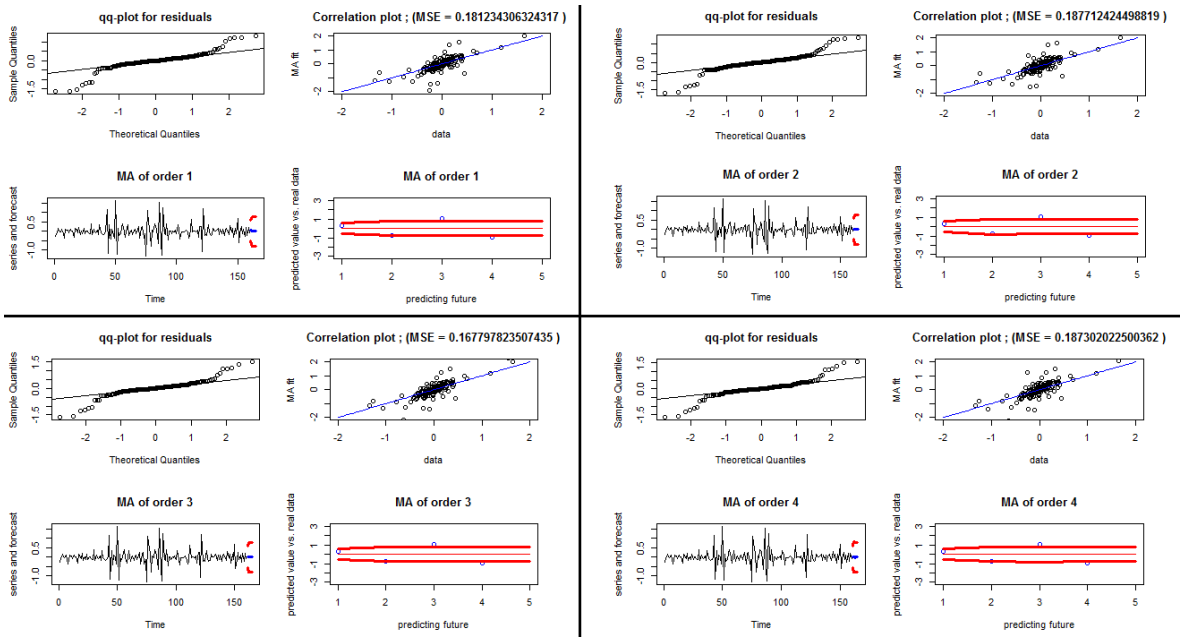

Figure 4.19: Model diagnosis results for fitting models by different MAs for Waterloo dataset.

rejected the null hypothesis of independency of residual errors.

As a complementary experiment, a sensitivity analysis based on error-bar values for MAs of different orders is performed in Figure 4.21. It can be seen from Figure 4.21 (a) that the variation of error is relatively the same for all of the MAs, and as mentioned, the best MSE belongs to MA(3). Figure 4.21 (b) indicates the estimation of MA(3) vs. real data. As a conclusion, it seems that MA process cannot be considered as a reliable forecaster for the Waterloo road roughness, and the other models should be checked.

The experiment is followed by using ARMA process for forecasting task. Figure 4.22 depicts the model diagnosis results for the fitted ARMA models. In general, the obtained results are not in favor of using ARMA for this process. In particular, QQ-plot indicates that the residual errors have deviation from the theoretical quantiles in both tails, and also, the prediction for unseen data falls outside the 95% CI at one point. At the same time, it seems that ARMA successfully trained its structure with respect to the training data because the correlation of the models output and real data are very good. However, this cannot be an enough condition since the performance for unseen data is more important.

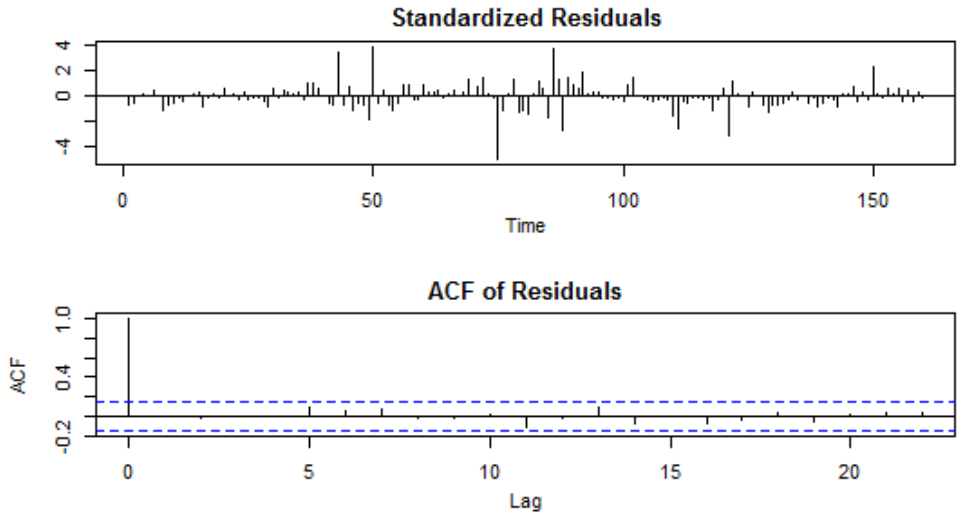

Figure 4.20: Model diagnosis results for MA(3).

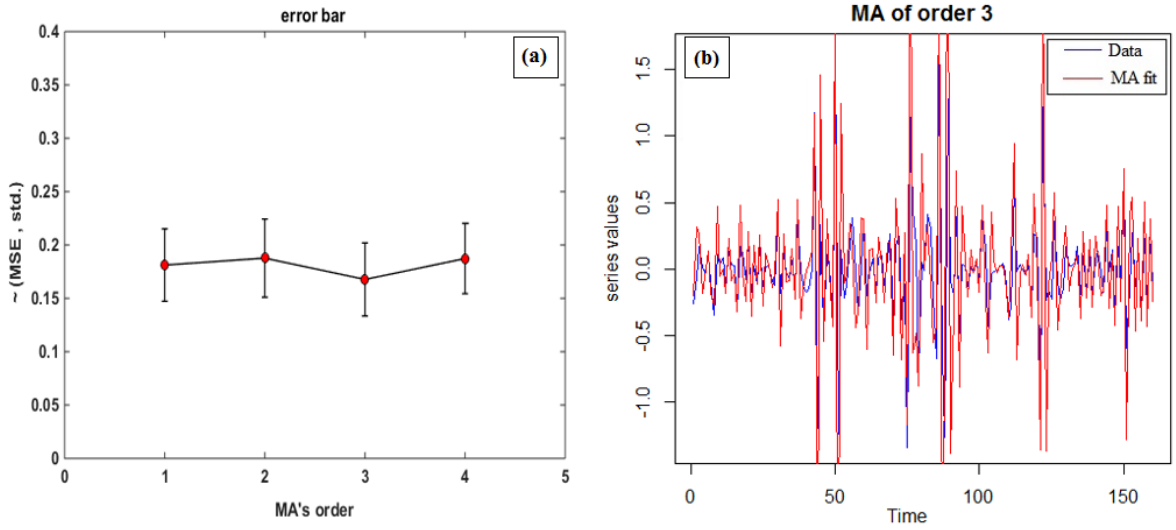

Figure 4.21: (a) Error bar sensitivity analysis of MA models of different orders obtained by 10-fold forward chaining, and (b) experimental time series data vs. MA(3) prediction for Waterloo dataset.

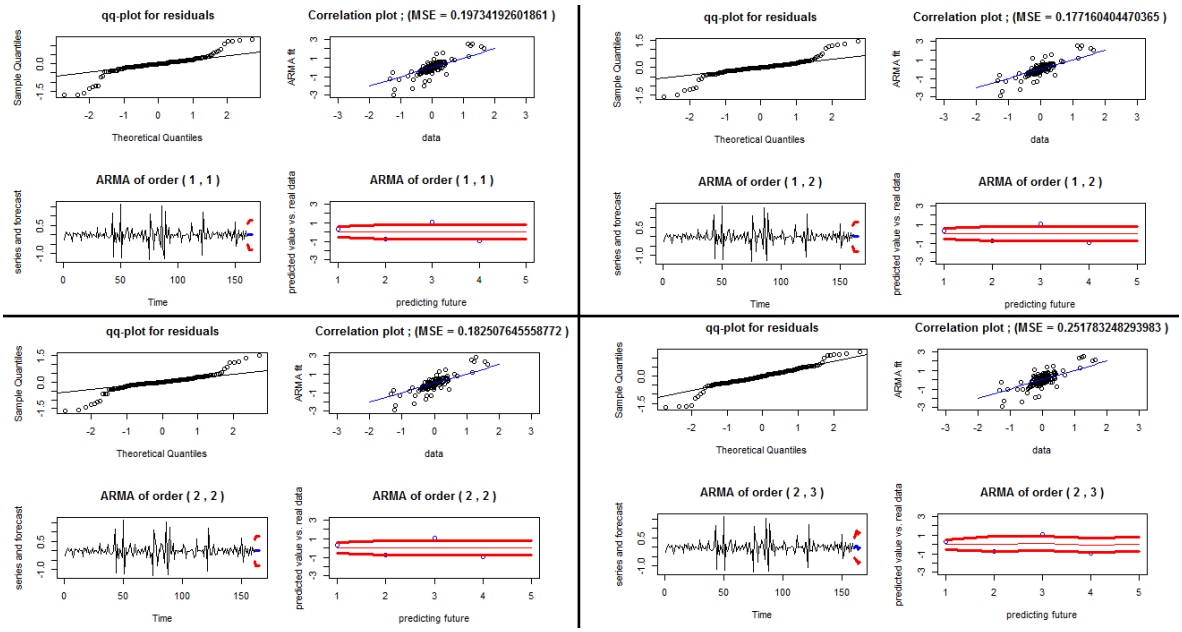

Figure 4.22: Model diagnosis results for fitting models by different ARMA processes for Waterloo dataset.

Table 4.2: Results of optimization for ARMA models for Waterloo dataset

Parameters	$\ell(\hat{\boldsymbol{\beta}})$	**AIC**	**MSE**	**std.**
$p = 1, q = 1$	-27.96	**59.92**	0.1973	0.0351
$p = 1, q = 2$	-28.03	62.06	0.1771	0.0332
$p = 2, q = 2$	-27.00	62.00	0.1825	0.0337
$p = 2, q = 3$	-25.48	62.48	0.2517	0.0396

Table 4.2 lists the obtained results for the trained models. As seen, the MSE and *std.* values of the first three models are really close to each other, and thus, the best model is ARMA(1, 1) because of having the lowest AIC. Therefore. Further analysis are conducted on this model.

Figure 4.23 depicts the model diagnosis results for ARMA(1, 1) process. As seen, ACF values of the residual errors remain within the CI bound, which is promising. Also, the prediction of ARMA(1, 1) vs. the experimental data is shown in Figure 4.24. All in all, it can be concluded that ARMA(1, 1) does a good job at the training level, but cannot be expected as a reliable model because of its performance at the testing stage. It seems that

more advanced models should also be tried for Waterloo dataset.

Figure 4.25 indicates the model diagnosis results for APARCH model. The MSE and *std.* values of APARCH model are 0.2648 and 0.0406, respectively. The comparison of the obtained results with the previous results indicate that APARCH shown a better performance in terms of QQ-plot, and the deviation from the theoretical quantiles are decreased a little bit. Also, the correlation plot is acceptable. Note that one of the APARCH predictions for the unseen data violates the 95% CI which undermines its reliability.

The results of ACF analysis for the model are shown in Figure 4.26. As seen, the ACF of both APARCH and GARCH(1, 1) model significantly violate CI, which means there could be a significant correlation between residual error values at different lags. This is clearer from QQ-plot of GARCH(1, 1) residuals, which has heavy tails and makes the normality of errors assumption not quite satisfactory. This offers checking GARCH(1, 1) with student t distribution for possible performance improvement. The Q-Q plot for GARCH(1, 1) with student t distribution seems quite satisfactory, and can further improve the performance of APARCH.

Figure 4.27 compares the predicted values of APARCH and the experimental data. It can be seen that APARCH does a good job in identification of the series; however, it is obvious that ARMA(1, 1) model has a lower MSE value which could be a result of overfitting because the performance of APARCH and ARMA are very close for N-step ahead forecast (unseen data).

The next model used for Waterloo data experiment is DLM. The model diagnosis results are shown in Figure 4.28. Note that the non-differenced non-stationary version of dataset is used for training DLM. Again the simple DLM is selected for the current study with $\mu_0 \mid \mathcal{I}_0 \sim N(-0.0045, 0.0810)$, $\nu \sim N(0, 2.59)$, and $\omega \sim N(0, 3)$. As seen, the QQ-plot of residual errors shows a deviation on both tails, but the deviation is not as significant as those of the other methods. Also, the ACF values of residuals mostly remain within the CI, and the correlation between data and the DLM outputs is acceptable. The obtained MSE is 0.2065 which is quite less than that of APARCH model and is close to those of ARMA processes. To have a better comparison between DLM and the models presented previously, the MSE and *std.* values of the differenced version of DLM are also reported, that are 0.2330 and 0.0381, respectively. It seems that the results are comparable to ARMA(1, 1).

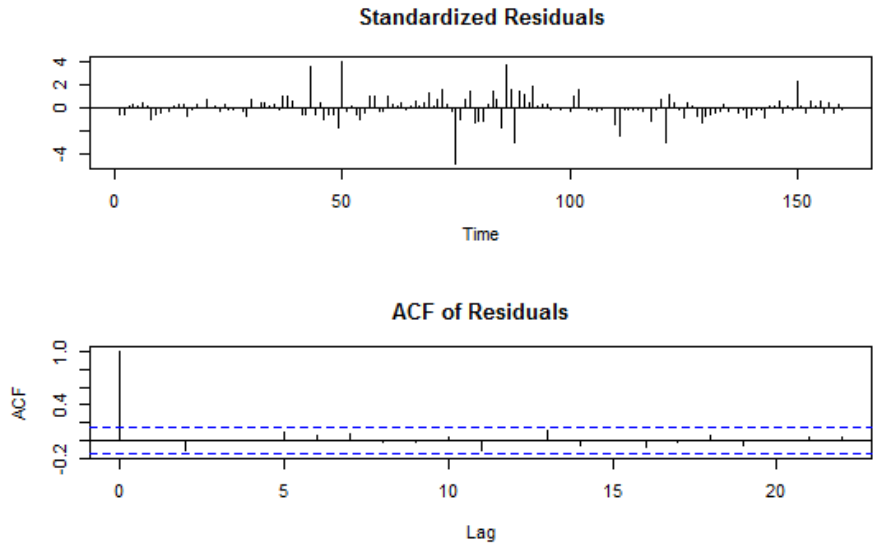

Figure 4.23: Model diagnosis results for ARMA(1, 1).

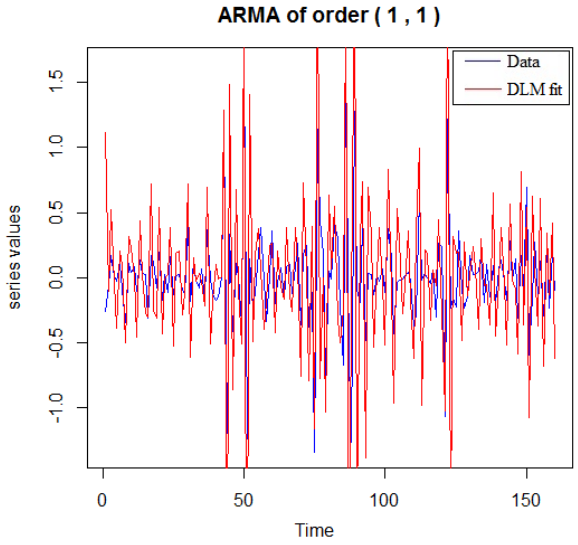

Figure 4.24: Experimental time series data vs. ARMA(1, 1) prediction for Waterloo dataset.

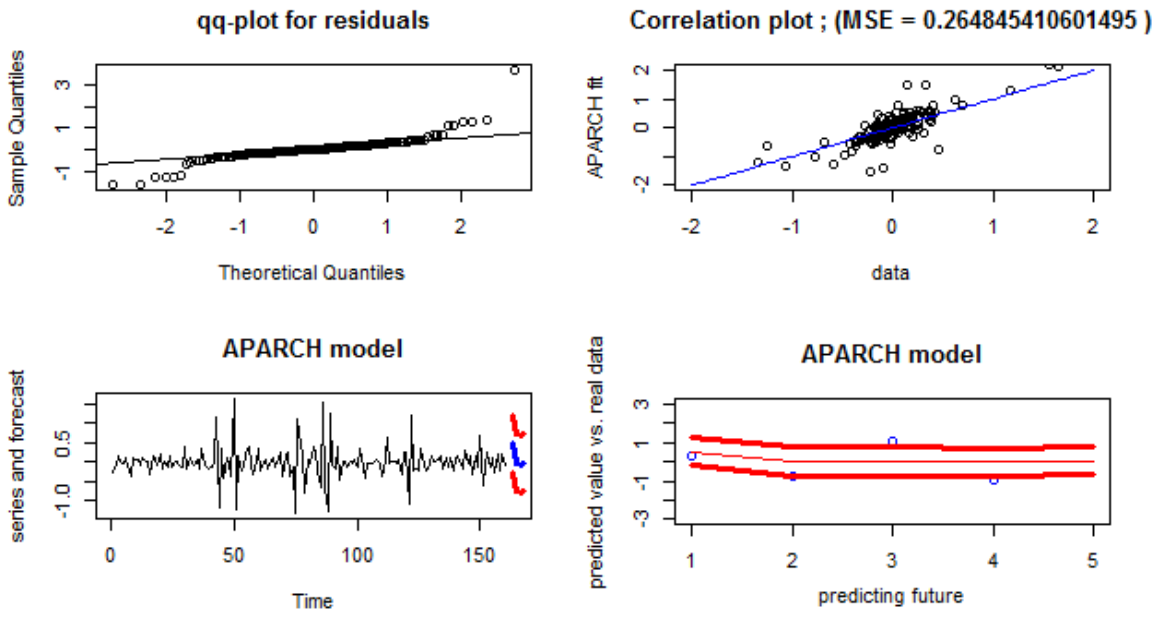

Figure 4.25: Model diagnosis results for fitted APARCH model for Waterloo dataset.

Also, the comparison of the predicted values of DLM with real data for both original and first order differenced versions are shown in Figure 4.29. It is obvious that DLM does a very good job and the estimated and real data profiles are very close to each other.

Among the models used for Waterloo dataset, it seems that DLM and ARMA(1, 1) have a good performance at the training stage. Also, DLM shows improvement in terms of independency of the obtained residual errors, and can be selected as the final choice for this study. Note that, since the simple version of DLM is used, it has a good computational speed, and does not take a significant time for forecasting which is acceptable for real-time applications, as is the case in this study. Also, as a Bayesian regression, once more data and information are available, the prior estimates can be improved which results in the improvement of the model, and it seems that DLM has an aptitude to be considered as an incremental learning technique, which is another required feature for the efficient design of the learning module of LBMPC.

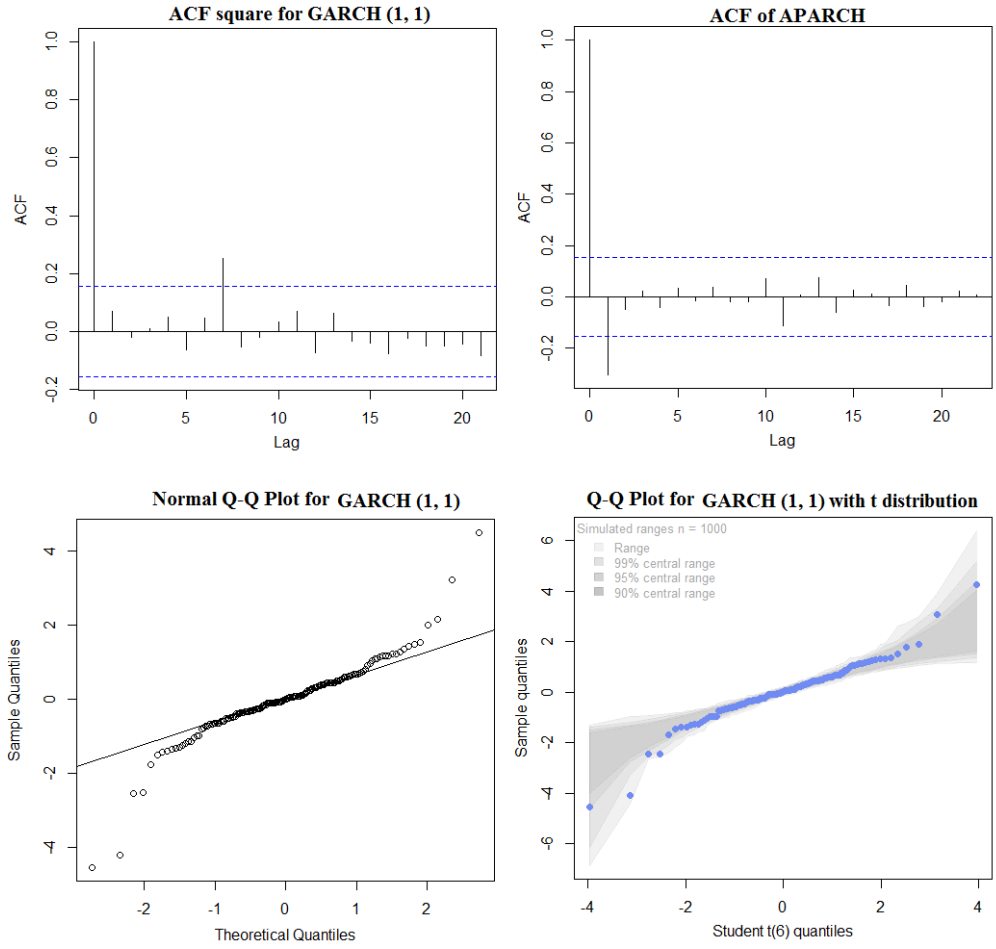

Figure 4.26: (a) ACF plot for GARCH(1, 1), (b) ACF plot for APARCH (c) Q-Q plot for GARCH(1, 1), and (d) Q-Q plot of GARCH(1,1) with t distribution for Waterloo dataset.

4.5 Remarks on Using Statistical Techniques for Road Roughness Prediction

Based on the conducted simulation, the following remarks are observed for the current case study:

1. One of the prominent features of the considered statistical forecasting methods is their capability to predict with a CI. The boundedness of disturbance sources, including

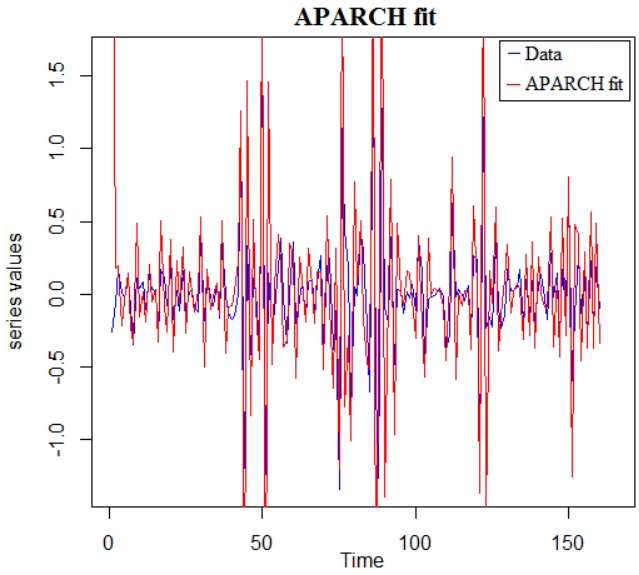

Figure 4.27: Experimental time series data vs. APARCH prediction for Waterloo dataset.

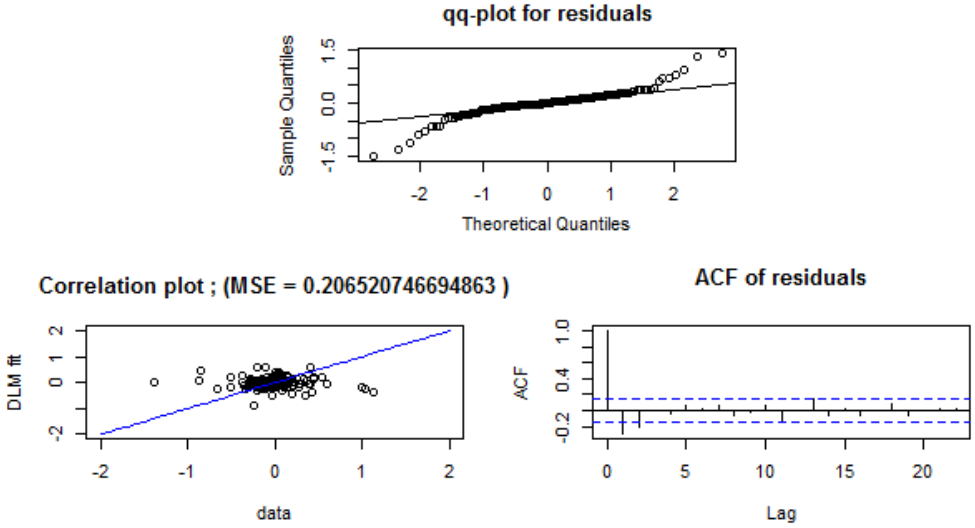

Figure 4.28: Model diagnosis of DLM for Waterloo dataset.

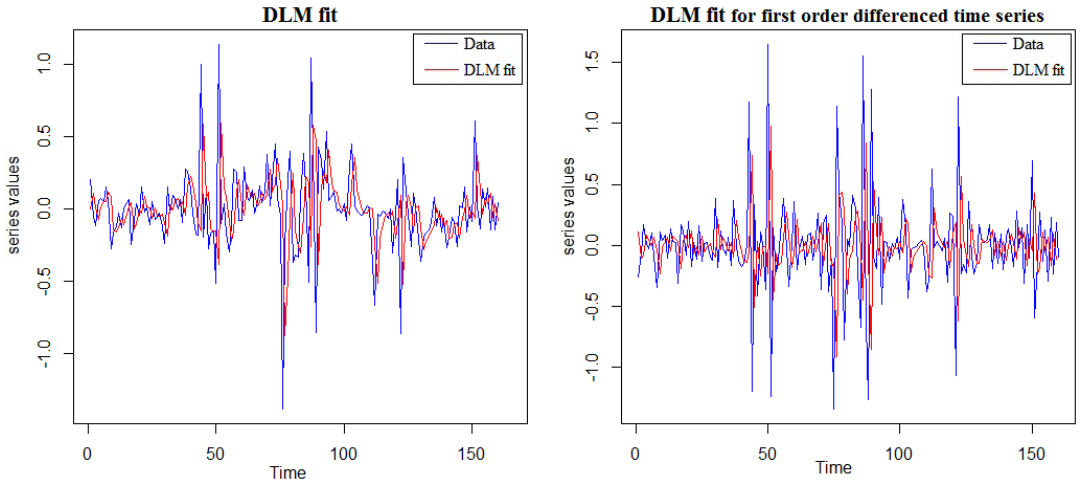

Figure 4.29: Experimental time series data vs. DLM prediction for of Waterloo dataset.

road disturbance, is necessary for proving the stability and robustness of LBMPC. This will be proven in Chapter 6.

2. The statistical forecasting methods have usually simple yet efficient formulations, which is very beneficial when using them for real-time applications. As observed in the previous sections, most of the obtained models had at-most 2 to 4 structural parameters, and could be tuned in a short period of time. This desirable feature together with the intrinsic robustness of the resulting forecasting systems make such techniques good fits to be used at the heart of any type of learnable predictive controllers including LBMPC.

3. As observed, Bayesian forecasting methods have a very good potential to be used for forecasting the road roughness profiles. Such techniques can be always improved by collecting more information about the road roughness, and choosing more appropriate prior distributions for the system parameters. This predisposes such techniques for incremental learning and online forecasting / prediction, which is another desirable feature for the learning module of LBMPC.

Note

All MATLAB and R codes pertaining to the simulation performed in this chapter can be found in Appendix.

Chapter 5

Reinforcement Learning and Stochastic Process for Desired Trajectory Design and Driver's Behavior Perception

This chapter is devoted to theoretical analyzing and explaining the potentials of stochastic process and reinforcement learning (RL) for estimating uncertain / unmeasured parameters such as drivers behavior on road (which is represented by vehicle speed) as well as designing a desired trajectory for learning-based model predictive controller (LBMPC), taking several sources of uncertainty into account. The rest of the chapter is organized as follows. Firstly, a review of the state-of-the-art pertinent to the application of stochastic process to control theory is presented, and some promising aspects of using probabilistic methods for optimal and robust controllers design are discussed. The review is followed by two distinct sections which argue why stochastic process and RL are useful for our control problem. Thereafter, the mathematical formulations of an absorbing state stochastic process and RL for vehicle speed estimation and desired trajectory design under uncertainty are given. After formulating the problems, some simulation are conducted to validate the performance of the considered methods. Finally, the findings of the chapter are presented.

5.1 Review of Advances from Control Prospective

Stochastic process and RL are among the most fruitful and promising techniques which have come to the aid of control engineers to design robust, optimal and adaptive control systems. Now a days, there is a wide consensus among control theoreticians that theorise and tools from stochastic calculus are rigid enough to be used for designing efficient and robust filtering and control algorithms [123]. For example, techniques like RL are very close in their implementation to traditional optimal control approaches like dynamic programming (DP). However, unlike DP which is based on a grid-based partitioning of state-space and control input space, and checking all possible outcomes to find the global solution, RL takes the advantage of Monte-Carlo Markov Chain (MCMC) procedure, transition probability matrix, a finite set of actions, and a reward function to approximately find the optimal solution, with remarkably less computational effort compared to DP [124].

Also, combination of stochastic process methods, e.g. absorbing state processes, point processes, Gaussian graphical model processes with other statistical learning and hypothesis testing techniques enables us to efficiently calibrate uncertainties associated with dynamic systems to be controlled [125]. These sorts of uncertainties usually appear in the form of measurement noises on sensors which read the system output, as well as un-modeled uncertainties on internal states which results in imperfect state information models [126, 127].

A majority of the existing controllers are implemented to track a desired trajectory which steers the system's evolution in an appropriate fashion. The controllers can have both open loop and closed-loop forms. It is apparent that online closed-loop controllers which update their performance based on the real functioning of the system are more interesting than offline controllers. On the other hand, online trajectory tracking controllers require efficient tools to deal with uncertainties of the environment and efficiently recognize the optimal and robust desired trajectory for tracking. To attain this goal, there is no better way than taking advantage from stochastic process techniques such as hidden Markov models, Monte Carlo with expectation maximization (MC-EM) and RL by defining a reward function and an appropriate action space which yields the desired trajectory [128, 129, 22]. The use of stochastic process methods for designing optimal trajectory for controllers have found its place within the society of robotics and autonomous systems [130], and in this sense, it is related to the topic of the current investigation.

In the light of the abovementioned information, in the next two sections, more details regarding the use of stochastic process for the current control problem will be given. In particular, a RL combined with graphical modeling concept and an absorbing state stochastic process are used for the online design of desired trajectory and realizing driver's behavior on road, respectively.

5.2 Why RL for Desired Trajectory Design?

RL can be viewed as a kind of cost-to-go function evaluation which tries to estimate the reward of the future evolution of the system based on the previous experiment. RL has been implemented in both deterministic and stochastic formats; however, the most interest goes upon the stochastic format as it has a deeper tie with real-world problems that are often associated with uncertainty. The motivation behind the proposition of RL was to facilitate solving tedious problems raised within the fields of big data, artificial intelligence (AI) and deep learning. In particular, RL has found its place among practitioners after being used (together with deep learning) by a company called Google Deep Mind for designing a computer program for playing an ancient Chinese game called Go. Because of the exponential increase of the possible events, it was previously impossible to write a program for this game which could compete with human professional player. However, it was demonstrated that RL, which is based on position evaluation and maximizing the reward of future movements at each point instead of evaluating all possible movements, is the key success of this program [131]. It has been shown that RL shares remarkable similarities with the computational strategies and mechanisms of decision making at the cellular and circuit level in the brain [132].

The approximate optimality of RL as well as its potential to be implemented in a stochastic fashion have placed it among the most distinct learning / decision making methods, and have made it reputable among practitioners. As reviewed in the previous section, the conducted research on using RL for the generation of desired trajectory for robots is abound in literature. In this investigation, RL is used for the real-time design of desired trajectory under uncertainty so that the vehicle suspension controller (LBMPC) can damp the vibration of suspension system on road in a realistic fashion.

In almost all of the conducted research in the literature of active suspension controller design, the desired trajectory is simply taken to be the body displacement of 0 (i.e. $z_{b_1} = z_{b_2} = 0$ in Eq. 3.1), and controller should always try to create actuation signals in a way that the vehicle body displacement be damped as fast as possible. However, in practice, there is a possibility that vehicle encounters big bumps which results in the considerable deflection of vehicle body. In such cases, the desired trajectory of $z_{b_1} = z_{b_2} = 0$ forces the controller to produce big actuation signals to damp the inflicted displacement in shortest possible time. This results in an inconvenience for passengers, and can harm the global interest for purchasing the vehicle in markets. Also, there are many other sources of uncertainty which can affect the displacement of vehicle body, and make the authenticity of choosing a constant desired trajectory regardless the environmental conditions more questionable.

To the author's opinion, using RL for the adaptive and smart online designing of desired

trajectory can be beneficial in the following senses:

1. RL has the potential to be fused with well-known statistical tools and concepts, such as transition probability matrix, expectation maximization and maximum likelihood estimation (in the form of maximum reward) to come up with an online trajectory builder which captures the effects of uncertainty, and at the same time can make robust and optimal decisions to steer the displacement of vehicle body towards 0 in a smooth fashion. By doing so, one can further increase the possibility of guaranteeing the comfort of passengers on road.

2. One of the other important features of RL is that it is an optimal decision making paradigm. It means that one can manually adapt the objective function (in the form of reward) to have the decision of interest. For the case of vehicle suspension control, for example, the reward function can be the weighted summation of different objectives, such as the speed of vibration damping and the smoothness of damping (which are obviously in confliction with each other). By tuning the weight of each objective, RL can build specific desired trajectories to meet the real pragmatic requirements.

3. Recently, in [133], model predictive control (MPC) was used for designing optimal trajectory for an aerially towed cable system. The findings of the research was very promising, and it was shown that using a desired trajectory generation scheme can be very beneficial compared to having a predetermined desired trajectory. However, as can be inferred, desired trajectory building by MPC requires remarkable calculations, and at-least, for our case study, it was impossible to use the same approach. This is because, for our problem, desired trajectory builder is only a sub-module in the bigger controller frame (LBMPC), and it is not computationally tractable to devise independent controllers for both the calculation of desired trajectory and control commands. However, it will be shown that by using an interesting graph theoretic based reward maximization function at the heart of RL, one can calculate the maximum likelihood trajectory with the worst-case complexity scenario of $O(poly(k) + k\ logN)$, where N and k represent the horizon length and the number of system states, respectively.

The detailed formulation of RL for this case study will be given later in this chapter, and its performance is validated by some examples.

5.3 Why Stochastic Perception of Driver's Behavior?

In this investigation, a stochastic process with absorbing state [134] is used for modeling the driver's behavior uncertainty (SU-1, see Chapter 3) on road. Within the context of vehicle suspension control, the most influential parameter which can be affected by driver's behavior is the speed of vehicle. In general, the state-space model of suspension system is formulated by considering a cruise speed for vehicle. This means that, regardless of the elements and operating conditions which can affect the vehicle speed on road, a constant speed is considered for vehicle.

This is when having a realistic estimation of vehicle speed is crucial for precise vehicle suspension control. Note that the suspension system under study has control and actuation systems on both the front and rear tires. So, by having the road profile (which includes the roughness information that vibrates the suspension system) and also the distance between the front and rear tires, one can exactly estimate the real-time external forces imposed on both the front and rear suspension units as a function of vehicle speed. Recall from the state-space model given in Eq. 3.2 that both the front and rear tires deflection depend on time, and thus, formulating the exact time of deflections as a function of vehicle speed is important. From the physical laws of motion (1^{st} order Taylor expansion), the following approximate time for the deflection of rear and front tires can be obtained:

$$t_{r_2} \simeq t_{r_1} + \frac{l_1 + l_2}{\nu_t}, \tag{5.1}$$

where l_1 and l_2 are the distances of the front and rear axles from the center of gravity of body mass, t_{r_1} and t_{r_2} are the time of deflection for front and rear tires, and ν_t is the real-time estimated speed of vehicle. Note that the variation of speed between time-steps is negligible, as the time portion between each set-point of the estimated profile is very small. So, that is logical to approximate the vehicle speed without considering acceleration (i.e. $\frac{1}{2}at^2 \simeq 0$).

The reason behind using absorbing state stochastic process lies in the fact that after any sudden deviation in driver's perception (which appears as a random change in vehicle speed), driver can manage to return to the nominal speed after a portion of time. Figure 5.1 visualizes the mentioned concept. So, it would logical to consider the nominal speed of vehicle as an absorbing state so that after any deviation in the vehicle speed, the absorbing state MC finally converges to the state delegating the standard speed, and stays there until another sudden change affects the drivers perception.

To the best knowledge of the author, such a concept has not been used so far by

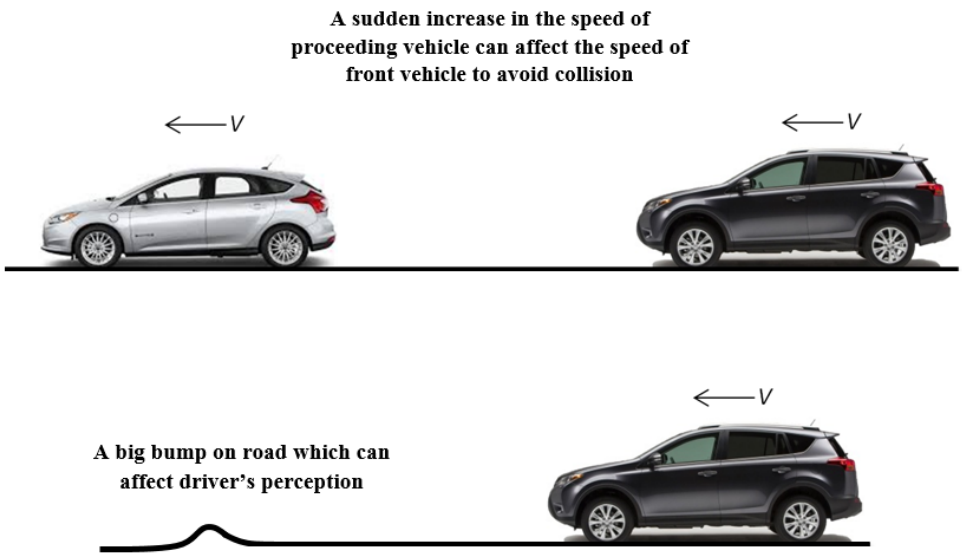

Figure 5.1: Two typical examples of conditions which can affect driver's perception.

any other researcher, and can be viewed as a very promising paradigm for the stochastic modeling of driver's perception on road.

The mathematical details of the abovementioned process will be given in the next section, and its authenticity will be validated numerically.

5.4 Problem Formulation

This section is organized in three sub-sections. Firstly, some standard notations are presented which will be used through the rest of the chapter. Thereafter, the considered RL method which is combined with a graph theoretic based optimization procedure for maximizing the reward function is presented. In the last sub-section, the absorbing state stochastic process that is used for the estimation of vehicle speed under uncertainty is presented.

5.4.1 Notations and preliminaries

Throughout the chapter, vectors are shown by lower case bold fonts ($\mathbf{a}, \mathbf{b}$, and *etc.*). Sets are presented by calligraphic upper case italic format ($\mathcal{A}$, $\mathcal{B}$, and *etc.*), and matrixes are given by upper case italic format (A, B, and *etc.*). The vectors are presented in columns, i.e. $\mathbf{a} = col(a_1, a_2, ..., a_m)$ where $\mathbf{a} \in \mathbb{R}^m$ (unless something else be mentioned). A^T is the trnspose of A. The Euclidian (L_2) norm is shown by $\| \cdot \|_2$. Also, $\| \cdot \|_A$, where A is a matrix, represents the weighted norm. For example, $\|\mathbf{a}\|_A = \sqrt{\mathbf{a}^T A \mathbf{a}}$. A vector of one's of appropriate size (e.g. $n \times 1$) is shown by $\mathbf{1}_{n\times 1}$. An identity matrix of appropriate size (e.g. $n \times n$) is shown by $\mathbb{I}_n$. The cardinality of set $\mathcal{A}$ (number of elements in set $\mathcal{A}$) is shown by $\#\mathcal{A}$.

5.4.2 RL for desired trajectory estimation

Before going into the details of the RL considered for this study, the general formulation of RL is given for better understanding. In general, RL includes 4 components $\{\mathcal{S}, \mathcal{A}, \mathcal{R}, f\}$ which are defined below.

Definition 5.1 [124] $\mathcal{S}$ (where $\#\mathcal{S} = k$) is the set of possible states (k is the number of states that the Markov chain can jump to), $\mathcal{A}$ is the set of actions, $\mathcal{R}$ is the set of reward signals, and $f : \mathcal{A} \times \mathcal{S} \to \mathcal{S}$ is a transition function which determines the jump from one state to another state given an action $\tilde{a} \in \mathcal{A}$, i.e. $s_{t+1} = f(s_t, \tilde{a}_t)$.

Definition 5.2 [124] A policy $\pi : \mathcal{S} \to \mathcal{A}$ is a function which determines the actions based on the current state, i.e. $\tilde{a}_t = \pi(s_t)$. Given the action at set-point t, the new state can be calculated using the transition function which in turn lets us calculate the conditional reward signal $\tilde{r}_{t+1} \mid_{s_{t-1},\tilde{a}_t,s_t} \equiv \tilde{r}(s_t, \tilde{a}_t, s_{t+1})$.

The goal of RL is to find the optimal policy sequences to maximize the accumulated reward at the end of process, given the initial state s_0. To comply with this, one plausible choice is to define a value function $V_t(.)$ and a cost fucntion $J_t(.)$, as below:

$$V(s_t) = \max_{\boldsymbol{\pi}} J(\boldsymbol{\pi}, s_{t+1}, ..., s_{t+N}, \tilde{a}_t, ..., \tilde{a}_{t+N-1}) = \max_{\boldsymbol{\pi}} \sum_{i=1}^{N} \gamma^i \ \tilde{r}(s_{t+i-1}, \tilde{a}_{t+i}, s_{t+i}) \mid_{s_t}$$

where $0 \leq \gamma \leq 1$, and N is the horizon length (the length of the chain created by stochastic process). The cost function of RL is defined such that the *Bellman principle of optimality* be satisfied. Let's denote the optimal value function by $V^*(s_t)$, due to the Bellman equation

of optimality, we have that:

$$V^*(s_t) = \max_{\boldsymbol{\pi}}\{\tilde{r}(s_t, \tilde{a}_t, s_{t+1}), \gamma\ V^*(s_{t+1})\}\ .$$

Remark 5.1 We can interchangeably consider the action-value function for finding the optimal policy. The resulting RL is called Q-learning, and the optimization problem and Bellman principle of optimality can be reformulated as:

$$Q(s_t, \tilde{a}_t) = \max_{\boldsymbol{\pi}} \sum_{i=1}^{N} \gamma^i\ \tilde{r}(s_{t+i-1}, \tilde{a}_{t+i}, s_{t+i})\ |_{s_t, \tilde{a}_t}\ .$$

$$Q^*(s_t, \tilde{a}_t) = \tilde{r}(s_t, \tilde{a}_t, s_{t+1}) + \gamma \max_{\boldsymbol{\pi}}\{Q^*(s_{t+1}, \tilde{a}_{t+1})\}.$$

Note that since we are dealing with the stochastic process, the transition function f is replaced with transition probability matrix, as follows:

$$\mathcal{P}r(s_{t+1} \mid s_t, \tilde{a}_t) = \mathcal{P}r(s_{t+1} = j \mid s_t = i, \tilde{a}_t) = p_{ij}\ .$$

where $i, j \in \{1, 2, ..., k\}$. The reward function and the value function are calculated as:

$$\tilde{r}(s_t, \tilde{a}_t, s_{t+1}) = E_{\boldsymbol{\pi}}[\tilde{r}_{t+1}\ |_{s_{t-1}, \tilde{a}_t, s_t}]\ ,$$

$$V(s_t) = E_{\boldsymbol{\pi}}\Big[\sum_{i=1}^{N} \gamma^i\ \tilde{r}(s_{t+i-1}, \tilde{a}_{t+i}, s_{t+i})\ |_{s_t}\Big]\ , \tag{5.2}$$

where $E_{\boldsymbol{\pi}}$ is the expectation operator under policy sequence $\boldsymbol{\pi}$.

The stochastic implementation of RL is best suited for our problem, and will be used to formulate the desired trajectory production problem.

Remark 5.2 Although the state-space of the controller is continuous, the stochastic process used for desired trajectory generation is set to be discrete which can be modelled by a finite-time Markov chain. Otherwise, the desired trajectory building complexity can make the problem computationally intractable, as the continuous desired trajectory space could require an infinite possible elements in the transition probability matrix. Thus, for the desired trajectory production, which is used for steering vehicle body displacement (i.e. z_{b_1} and z_{b_2} in Eq. 3.1 that have continuous domain), the original continuity of the state-space is replaced with a Markov chain which has a finite transition probability matrix (as the discrete skeleton of the chain) and a sojourn time after jumping to a new state. Indeed

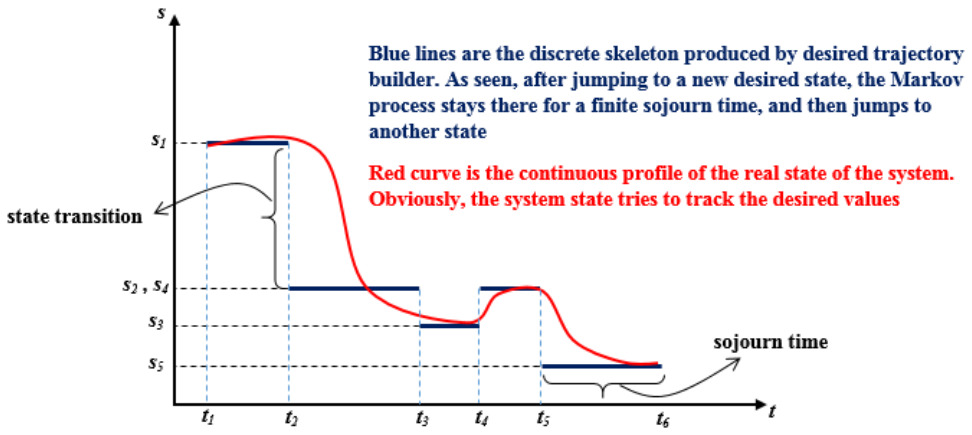

Figure 5.2: Schematic illustration of the considered desired trajectory production strategy.

using a discrete skeleton transition probability matrix together with a sojourn time is a common strategy to cope with continuous stochastic processes.

For better understanding of the above remark, Figure 5.2 visualizes the concept.

The stochastic RL with transition probability matrix, probabilistic reward signal and probabilistic value function presented above is used for producing desired trajectory for vehicle's rear and front body displacement. As mentioned, the trajectory produced by RL is used instead of simply taking the value 0 for the desired deflection of vehicle body along the road.

Remark 5.3 The same desired trajectory will be used for both controllers mounted at the rear and front axels of vehicle body. Eq. 5.1 is used for calculating the required time delay for sending the desired trajectory signal to front and rear controllers.

The remaining point is to use an algorithm to find the optimal policy by maximizing the accumulated reward presented in Eq. 5.2 to get the best possible trajectory. To do so, a very efficient method is used which reformulates the reward signal maximization as the *maximum likelihood estimation of trajectories in a Markov chain* (MLE-TMC) [135]. The salient asset of the method lies in its potential to generate the most practical trajectory based on the transition probability matrix of the stochastic process.

MLE-TMC works based on a predetermined transition probability matrix. Firstly, the method is presented for a given probability matrix, and after that, it is mentioned how to determine the transition probability matrix of the discrete skeleton and sojourn time for a

continuous time Markov chain (CTMC).

Remark 5.4 MLE-TMC represents the Markov chain stochastic process in graphical modeling fashion. So, there are, say k nodes ($\#\mathcal{S} = k$), which create a complex directed graph based on a $k \times k$ transition probability matrix P.

Remark 5.5 MLE-TMC assumes that no node repetition is allowed in the stochastic process, and at each transition, Markov chain should jump from one state to another. Indeed, this condition is compatible with CTMC, as the transition probability matrix of discrete skeleton does not allow any state (node) repetition, by definition (which will be given right after the formulation of MLE-TMC).

Let $\{\mathbf{s}(t)\}_{t\in\mathbb{Z}^+}$ be a finite-time discrete stochastic process with $k \times k$ transition probability matrix P, such that $\forall i, j : p_{ij} \geq 0$ and $P.\mathbf{1}_{k\times 1} = \mathbf{1}_{k\times 1}$. Let s_0 and s_f be the pre-defined initial and final states of the chain. Note that $\mathbf{s} = (s_0, s_1, ..., s_{t+N-1}, s_f)$ can be viewed as a random walk, where N is the horizon length, and $\mathbf{s}(i) = s_i$ is a random variable representing the state of the chain at i^{th} step.

Definition 5.3 [135] The MLE of a trajectory $\{\mathbf{s}(t)\}_{t\in\mathbb{Z}^+}$ is defined as any trajectory $(s_0, s_1, ..., s_f)$ which satisfies:

$$MLE(\mathbf{s}) = \max \mathcal{P}r(s_1, s_2, ..., s_{t+N-1} \mid s_0, s_f, \tilde{a}_t) = \max \mathcal{P}r(s_1, s_2, ..., s_{t+N-1} \mid s_0, \tilde{a}_t) \ .$$

Remark 5.6 The above problem can also be viewed as maximizing a reward signal based on a sequence of actions.

Definition 5.4 The maximum likelihood estimation of trajectory can be represented as a minimum-weight walk problem of finite length $t + N - 1$ from node s_0 to node s_f on a weighted directed graph $\mathcal{G}(\mathcal{V}, \mathcal{E}, W)$, where $\mathcal{V}$ is the vertices set with cardinality k (i.e. $\#\mathcal{V} = k$), $\mathcal{E} = \{(i, j) \mid \forall i, j \in \{1, 2, ..., k\} \ \& \ p_{ij} > 0\}$ is the set of edges in the graph, and W is a $k \times k$ matrix of weights such that $\forall i, j : W(i, j) = w_{ij} = -\log p_{ij}$.

Definition 5.5 Let denote a simple cycle in a directed graph $\mathcal{G}$ by $\mathbf{c}$, then $\mathbf{c}^{(m)}$ is a walk in a graph that is the concatenation of m repetitions of cycle $\mathbf{c}$. See Figure 5.3 for more details.

Definition 5.6 Let $\mathbf{s}$ be an arbitrary random walk on graph $\mathcal{G}$. The length of the walk is measured by the number of edges in $\mathbf{s}$ and is denoted by $|\mathbf{s}|_{\mathcal{E}}$, and $\mathbf{s}(i)$ represents the state of the chain at i^{th} step. See Figure 5.4 for more details.

Definition 5.7 Let $w(\mathbf{s})$ be the sum of the weights assigned to the edges of $\mathbf{s}$, then, the mean weight of $\mathbf{s}$ is defined as $\bar{w}(\mathbf{s}) = \frac{w(\mathbf{s})}{|\mathbf{s}|_{\mathcal{E}}}$. See *Example 5.1* for more details.

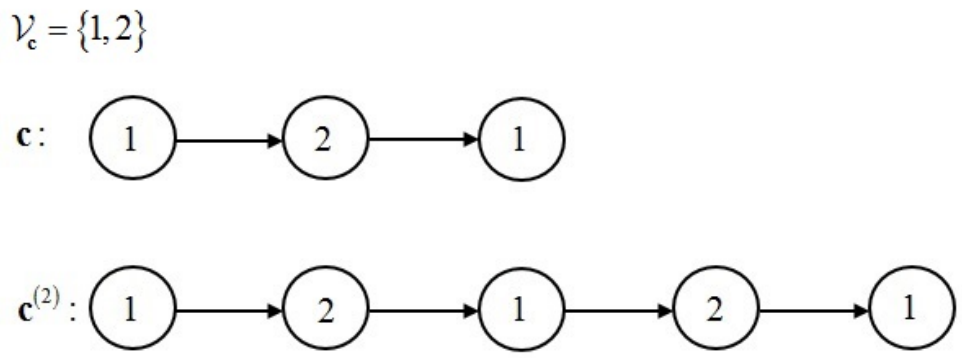

Figure 5.3: A random walk consisting of two time repetitions of $\mathbf{c}$.

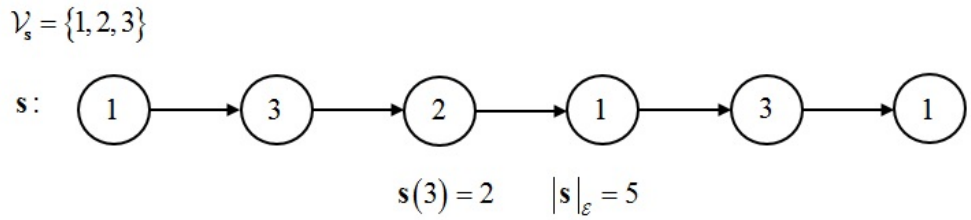

Figure 5.4: The length and state representation of a given random walk.

Example 5.1 Let $\mathbf{s}$ with $\mathcal{V}_{\mathbf{s}} = \{1, 2, 3\}$, be a random walk, generated by the transition probability matrix given in Figure 5.5, then we get:

$$|s|_{\mathcal{E}} = 5 \ ,$$
$$w(\mathbf{s}) = -log\ p_{13} - log\ p_{32} - log\ p_{23} - log\ p_{31} - log\ p_{13} = 4.021 \ ,$$
$$\bar{w}(\mathbf{s}) = \frac{w(\mathbf{s})}{|\mathbf{s}|_{\mathcal{E}}} = 0.8042 \ .$$

Remark 5.6 It can be immediately inferred that two random walks, say $\mathbf{s}_1$ and $\mathbf{s}_2$, that share the same collection of edges are "equivalent", as they have the same length and weight summation.

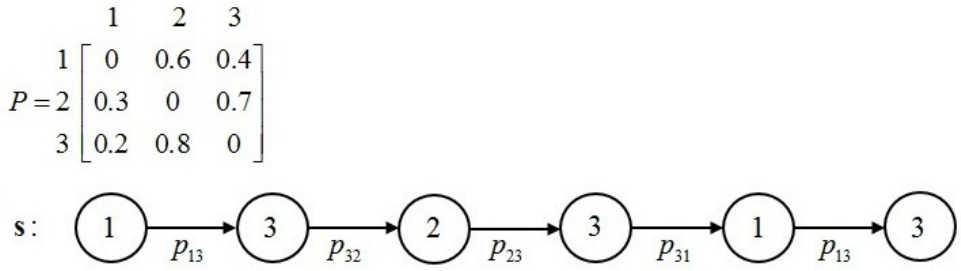

Figure 5.5: The transition probability matrix and resulting random walk for *Example 5.1*.

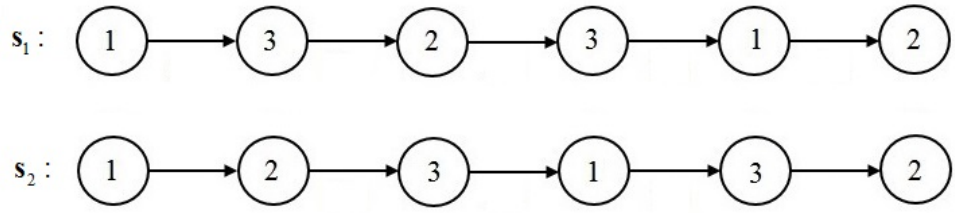

Figure 5.6: The two random walks studied in *Example 5.2*.

Example 5.2 The random walks depicted in Figure 5.6 are equivalent since (1) $\mathbf{s}_1$ and $\mathbf{s}_2$ possess the same collection of edges, i.e. $\mathcal{E}_{\mathbf{s}_1} = \mathcal{E}_{\mathbf{s}_2} = \{e_{13}, e_{32}, e_{23}, e_{31}, e_{12}\}$, (2) $|s_1|_\mathcal{E} = |s_2|_\mathcal{E} = 5$, and (3) $w(\mathbf{s}_1) = -log\ p_{13} - log\ p_{32} - log\ p_{23} - log\ p_{31} - log\ p_{12} = -log\ p_{12} - log\ p_{23} - log\ p_{31} - log\ p_{13} - log\ p_{32} = w(\mathbf{s}_2)$.

Definition 5.8 Let $\mathbf{s}_1$ and $\mathbf{s}_2$ be two arbitrary random walks on graph $\mathcal{G}$, and $\mathcal{E}_\mathbf{s}$ be the collection of edges in $\mathbf{s}$, then $\langle \mathbf{s}_1 + \mathbf{s}_2 \rangle$ is all set of walks resulting from the combination of edges obtained by $\mathcal{E}_{\mathbf{s}_1} \cup \mathcal{E}_{\mathbf{s}_2}$, and $\langle \mathbf{s}_1 - \mathbf{s}_2 \rangle$ is all set of walks resulting from the combination of edges obtained by $\mathcal{E}_{\mathbf{s}_1} \setminus \mathcal{E}_{\mathbf{s}_2}$ if $\mathcal{E}_{\mathbf{s}_1} \subseteq \mathcal{E}_{\mathbf{s}_2}$, and $\emptyset$ otherwise. Based on the given definitions, some properties can be immediately extracted: (1) $\langle \cdot \pm \cdot \rangle$ can be extended to $\langle \cdot \pm \langle \cdot \pm \cdot \rangle \rangle$ and *etc.*, and (2) $w(\langle \cdot \pm \cdot \rangle)$ and $|\langle \cdot \pm \cdot \rangle|_\mathcal{E}$ denote the weight summation and length of such walks, respectively.

Example 5.3 The properties presented in *Definition 5.8* can be calculated as below for two graphs $\mathbf{s}_1$ and $\mathbf{s}_2$ with $\mathcal{V}_{\mathbf{s}_1} = \mathcal{V}_{\mathbf{s}_2} = \{1, 2, 3, 4\}$, shown in Figure 5.7:

$$\mathcal{E}_{\mathbf{s}_1} = \{e_{12}, e_{24}, e_{41}, e_{13}, e_{32}\}\ ,\ \mathcal{E}_{\mathbf{s}_2} = \{e_{14}, e_{42}, e_{23}, e_{31}\}\ ,$$

$$\langle \mathbf{s}_1 + \mathbf{s}_2 \rangle = \{\mathbf{s} \mid \mathcal{E}_{\mathbf{s}} = \{e_{12}, e_{13}, e_{14}, e_{23}, e_{24}, e_{31}, e_{32}, e_{41}, e_{42}\}\} \ ,$$
$$\langle \mathbf{s}_1 - \mathbf{s}_2 \rangle = \emptyset \ .$$

Based on the above definitions and remarks, it would be possible to implement a computationally efficient heuristic-type search to get the minimum weight walk trajectory (the one which maximizes the reward signal). The considered algorithm works based on the decomposition of a walk into a group walk [135]. A group walk possesses the same collection of edges as those of original walk $\mathbf{s}$. In a group walk, the edges of original walk $\mathbf{s}$ are divided into two categories of anchor walk $\mathbf{a}$, and a set of simple cycles which is called free cycle $\mathbf{f}$.

Definition 5.9 [135] The anchor walk is a bounded length walk that contains anchor nodes as the first occurrence of the unique nodes in original walk $\mathbf{s}$. Each simple cycle between anchor nodes is removed and added to the collection of free cycles. Based on the given definition, one can infer that $w(\mathbf{s}) = w(\mathbf{a}) + w(\mathbf{f})$ and $|\mathbf{s}|_{\mathcal{E}} = |\mathbf{a}|_{\mathcal{E}} + |\mathbf{f}|_{\mathcal{E}}$.

Remark 5.7 If there is a simple cycle at the beginning of the chain, it cannot be removed from the anchor walk since it completely changes the nature of the chain. The starting point of the anchor walk is the same as that of original walk.

Figure 5.8 depicts a schematic illustration of representing a walk $\mathbf{s}$ by anchor walk and free cycle.

One interesting point about the group decomposition method is that it can represent a given walk of arbitrary length by a finite length anchor walk and a number of simple cycles. It will be shown that this interesting feature allows us to come up with an optimal minimum-weight trajectory with low computational effort which is quite desirable for being used as online desired trajectory generation module in LBMPC.

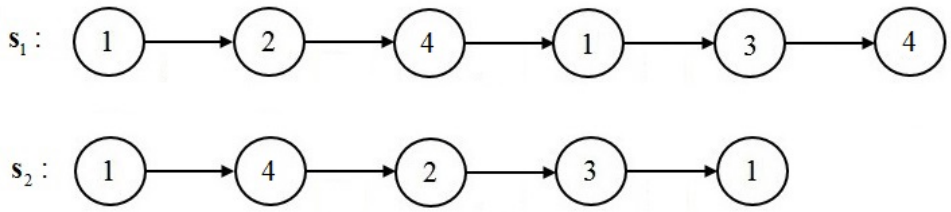

Figure 5.7: The two random walks studied in *Example 5.3*.

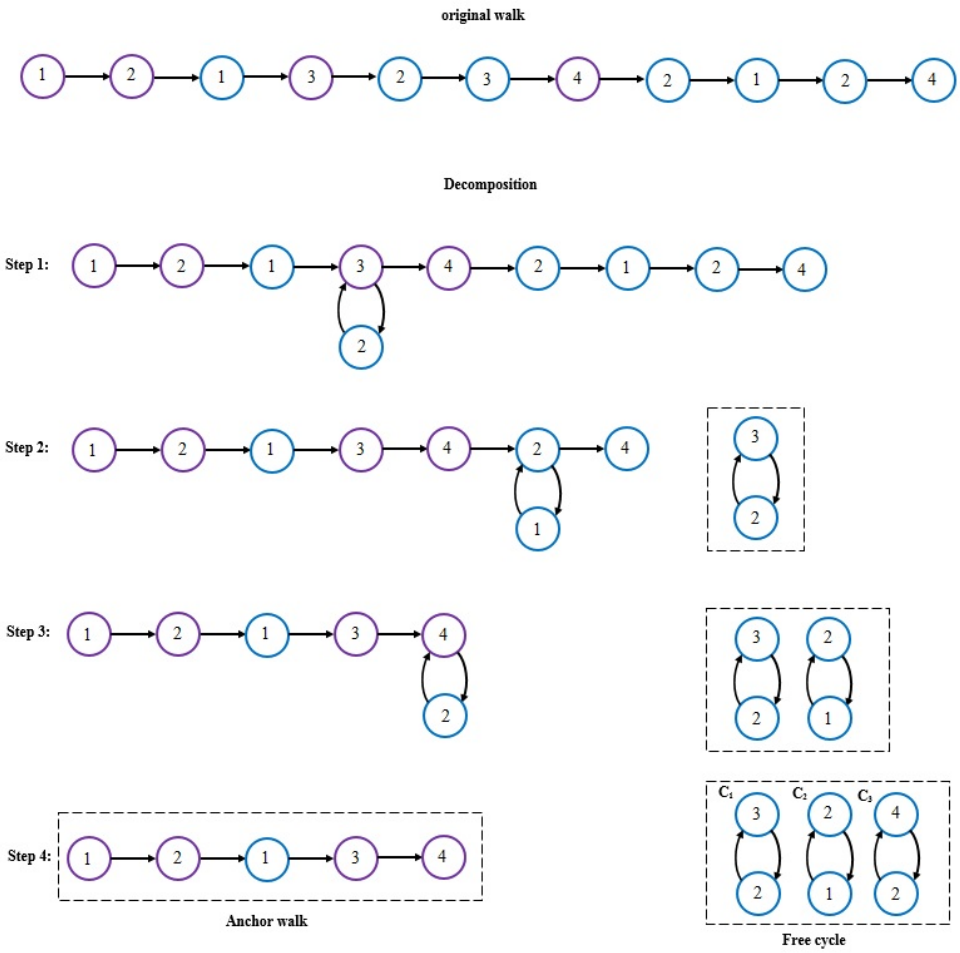

Figure 5.8: A schematic illustration of group walk decomposition process.

Theorem 5.1 *Let $h < k$ be the number of distinct nodes in random walk* $\mathbf{s}$*. Then, the length of anchor walk* $\mathbf{a}$ *is bounded by* $|\mathbf{a}|_{\mathcal{E}} \leq \frac{h(h+1)}{2}$.

Proof. The proof is rather simple following the information given in *Definition 5.9*. As mentioned, anchor walk includes anchor nodes which are the first occurrence of unique nodes in original walk $\mathbf{s}$. So, it holds that between i^{th} and $(i+1)^{th}$ anchor nodes, once can place at-most i distinct nodes. This means that, in the worst-case scenario, the maximum length of anchor walk is $|\mathbf{a}|_{\mathcal{E}} = 1 + 2 + \cdots + h = \frac{h(h+1)}{2}$. $\square$

Definition 5.10 [135, 136] In a weighted directed graph $\mathcal{G}(\mathcal{V}, \mathcal{E}, W)$, a mean-weight cycle cover refers to a weight-wise ordered set of cycles $\mathcal{C} = \{\mathbf{c}_1, \mathbf{c}_2, ..., \mathbf{c}_q\}$ if for $\forall j \in \{1, 2, ..., q\}$, by removing all vertices appearing in $\{\mathbf{c}_1, \mathbf{c}_2, ..., \mathbf{c}_{j-1}\}$ from graph $\mathcal{G}$, the cycle $\mathbf{c}_j$ has the smallest mean-weight in the resulting graph, and also the graph obtained by removing all vertices appearing in $\mathcal{C}$ from $\mathcal{G}$ is acyclic. Note that the remaining graph can be something like anchor walk.

Now it is the time to mention how the algorithm works. Prior to that, one proposition and one Lemma should be given for facilitating the understanding of the performance of the method.

Lemma 5.1 *For two numbers $x, y \in \mathbb{Z}^+$, if $z = xy + 1$ objects are distributed among y sets, then at-least one of the sets contains $x + 1$ objects.*

Proof. It can be easily shown that $x + 1 = \lfloor (z - 1)/y \rfloor + 1$, where $\lfloor \cdot \rfloor$ represents the floor function. □

Proposition 5.1 [135] *Let $\mathcal{D}$ be a set including a finite number of integers with cardinality $\#\mathcal{D}$. Then, for any integer $0 < k \leq \#\mathcal{D}$, there exists a non-empty set $\mathcal{I} \subseteq \mathcal{D}$ such that $(\sum_{x \in \mathcal{I}} x) \mod k = 0$.*

Proof. Assume that $\mathcal{D} = \{d_1, d_2, ..., d_{\#\mathcal{D}}\}$, and let $S_j = \sum_{i=1}^{j} d_i$ be the summation of the first j^{th} elements in $\mathcal{D}$. If $\exists j \ni S_j \mod k = 0$, then $\mathcal{I} = \{d_1, d_2, ..., d_j\}$. Otherwise, for all j, we get $S_j \mod k \in \{d_1, d_2, ..., d_j\}$. Thus, based on *Lemma 5.1*, $\exists j_1 < j_2 \ni S_{j_1} \mod k = S_{j_2} \mod k$, which means $\mathcal{I} = \{d_{j_1+1}, d_2, ..., d_{j_2}\}$. □

Theorem 5.2 [135] *Let $\mathcal{C} = \{\mathbf{c}_1, \mathbf{c}_2, ..., \mathbf{c}_q\}$ denote the mean-weight cycle cover of a weighted graph $\mathcal{G}(\mathcal{V}, \mathcal{E}, W)$. Let $\mathbf{s}_t^*$ be a minimum-weight walk of length $|\mathbf{s}_t^*|_{\mathcal{E}} = N > k^2 + \frac{k(k+1)}{2}$ (where N is the horizon length and k is the number of states), satisfying the initial and final conditions $\mathbf{s}_t^*(t) = s_0$ and $\mathbf{s}_t^*(t + N) = s_f$. Then, $\exists i \in \{1, 2, ..., q\}$ and an arbitrary finite length walk $\mathbf{s}$ such that:*

$$|\langle \mathbf{s} + \mathbf{c}_i^{(m)} \rangle|_{\mathcal{E}} = N \ ,$$

$$w(\langle \mathbf{s} + \mathbf{c}_i^{(m)} \rangle) = w(s_t^*) \ ,$$

where $\langle \mathbf{s} + \mathbf{c}_i^{(m)} \rangle$ is a non-empty set, $|\mathbf{s}|_{\mathcal{E}} \leq k|\mathbf{c}_i|_{\mathcal{E}} + \frac{k(k+1)}{2}$, and $m = \frac{N - |\mathbf{s}|_{\mathcal{E}}}{|\mathbf{c}_i|_{\mathcal{E}}}$.

Proof. Let's represent $\mathbf{s}_t^*$ by group decomposition method in the form of anchor walk $\mathbf{a}$ and free cycle $\mathbf{f}$. Based on *Theorem 5.1*, it holds that $|\mathbf{a}|_{\mathcal{E}} \leq \frac{k(k+1)}{2}$. Thus, based on the assumption made in the theorem and also based on *Definition 5.9*, we get $|\mathbf{f}|_{\mathcal{E}} = |\mathbf{s}_t^*|_{\mathcal{E}} - |\mathbf{a}|_{\mathcal{E}} = N - |\mathbf{a}|_{\mathcal{E}} > k^2$, which means that free cycle possesses more than k^2 edges on $\mathbf{s}_t^*$. Also, based on the properties of mean-weight cycle cover given in *Definition 5.10*, there is at-least one cycle in $\mathcal{C}$ intersecting all cycles in $\mathbf{f}$ (they should have at-least one

common node). Let $c_i \in \mathcal{C}$ be the cycle with smallest mean-weight which shares common node with some members of $\mathbf{f}$. To construct the optimal walk, one should remove cycles from $\mathbf{f}$, and replace it with the repetitions of $\mathbf{c}_i^{(m)}$. Based on *Proposition 5.1*, as long as $\mathbf{f}$ includes at-least $\mathbf{c}_i$ cycles, it is possible to choose a number of members of $\mathbf{f}$ (resulting in a smaller free cycle container $\mathbf{f}'$) such that the length of $|\mathbf{f}'|_{\mathcal{E}}$ becomes a multiple of $\mathbf{c}_i$. Removing cycles from $\mathbf{f}$ should continue until the number of free cycles in $\mathbf{f}$ becomes less than $|\mathbf{c}_i|_{\mathcal{E}}$. Thereafter, the remaining cycles in $\mathbf{f}$ are added to anchor walk $\mathbf{a}$ to form $\mathbf{s}$. By forming $\mathbf{s}_t^*$ in the abovementioned manner, we get $|\langle \mathbf{s}_t^* \rangle|_{\mathcal{E}} = |\langle \mathbf{s} + \mathbf{c}_i^{(m)} \rangle|_{\mathcal{E}} = N$. Also, since $\mathbf{c}_i$ is the smallest mean-weight cycle in $\mathcal{C}$ that intersects with members of $\mathbf{f}$, no cycle in $\mathbf{f}$ can have smaller mean edge weight than $\mathbf{c}_i$, and thus $w(\langle \mathbf{s} + \mathbf{c}_i^{(m)} \rangle) \leq w(s_t^*)$. On the other hand, the optimality of $\mathbf{s}_t^*$ implies $w(s_t^*) \leq w(\langle \mathbf{s} + \mathbf{c}_i^{(m)} \rangle)$, and combining the two inequalities gives $w(\langle \mathbf{s} + \mathbf{c}_i^{(m)} \rangle) = w(s_t^*)$. Also $|\mathbf{s}|_{\mathcal{E}} \leq k|\mathbf{c}_i|_{\mathcal{E}} + \frac{k(k+1)}{2}$ can be inferred from *Theorem 5.1* which states that $|\mathbf{a}|_{\mathcal{E}} \leq \frac{k(k+1)}{2}$, and also from the fact that the number of remaining cycles in $\mathbf{f}$ is less than $|\mathbf{c}_i|_{\mathcal{E}}$, and the length of a simple cycle cannot exceed k. This completes the proof. $\square$

At this point, based on *Theorem 5.2*, it would be possible to calculate the most probable desired trajectory, provided that a graph $\mathcal{G}$ with discrete finite-time transition probability matrix be available.

However, as mentioned, in our case, the model states (including front and rear vehicle body deflections) follow a continuous time Markov chain (CTMC), and thus, it is necessary to point out how to turn the process to a discrete finite-time random event.

Definition 5.11 A stochastic process $\{\mathbf{s}(t)\}_{t \in \mathbb{Z}^+}$ is called CTMC if (1) for $t \geq 0$, $\mathbf{s}(t)$ takes values from a countable set $\mathcal{S}$, and (2) the Markov property holds, i.e. for $t_1, t_2 \geq 0$ and $\forall i, j \in \mathcal{S}$:

$$\mathcal{P}r(s_{t_1+t_2} = j \mid s_{t_1} = i, \forall s_u; 0 \leq u \leq t_1) = \mathcal{P}r(s_{t_1+t_2} = j \mid s_{t_1} = i) \ .$$

Remark 5.8 It is common to consider the CTMC to be time-homogenous, i.e. $p_{ij}(t_2) = \mathcal{P}r(s_{t_1+t_2} = j \mid s_{t_1} = i) = \mathcal{P}r(s_{t_2} = j \mid s_0 = i)$.

In general, two important questions should be answered when dealing with CTMC:

1. How long does Markov chain stay in a state i ?
2. When does Markov chain leave the current state, and how to decide which state it will enter?

Theorem 5.3 *Let T_i be the sojourn time, i.e. the random portion of time that the Markov chain stays in state i. Then, T_i has an exponential distribution.*

Proof. The proof requires the properties given in *Definition 5.11* and *Remark 5.8*, and follows the argument below:

$$\forall t_0, t_1, t_2 \in \mathbb{R}^+ \ :$$

$$\mathcal{P}r(T_i > t_1 + t_2 \mid T_i > t_1) = \mathcal{P}r(s_u = i, t_0 \leq u \leq t_0 + t_1 + t_2 \mid s_u = i; \forall s_u : t_0 \leq u \leq t_0 + t_1)$$

$$= \mathcal{P}r(s_u = i, t_0 + t_1 \leq u \leq t_0 + t_1 + t_2 \mid s_{t_0+t_1} = i; s_u = i; \forall s_u : t_0 \leq u \leq t_0 + t_1)$$

$$= \mathcal{P}r(s_u = i, t_0 + t_1 \leq u \leq t_0 + t_1 + t_2 \mid s_{t_0+t_1} = i)$$

$$= \mathcal{P}r(s_u = i, t_0 \leq u \leq t_0 + t_2 \mid s_{t_0} = i) = \mathcal{P}r(T_i > t_2) \ .$$

So, $\forall t_1, t_2 \in \mathbb{R}^+ : \mathcal{P}r(T_i > t_1 + t_2 \mid T_i > t_1) = \mathcal{P}r(T_i > t_2)$ that in-turn satisfies the memoryless property. This means that T_i follows and exponential distribution. This completes the proof. $\square$

Definition 5.12 Let θ_i be the intensity of exponential process, then $T_i \sim exp(\theta_i)$. This means that once the stochastic process enters state i, the time it spends there before jumping to another state is exponentially distributed with mean $\frac{1}{\theta_i}$.

For the 2^{nd} question, it is obvious that the transition probability is independent of sojourn time once stochastic process leaves i, and only depends on the current state (and obviously the current state carries no information about time). So, the probability of jumping from i to j only depends on p_{ij} which can be read from transition probability matrix. $\{p_{ij}\}_{t \geq 0}$ can be viewed as the transition probability matrix of discrete-time stochastic process (DTMC) which is the discrete skeleton of CTMC. Also, not that based on the presented definitions and properties of CTMC, it holds that $p_{ii} = 0$.

Remark 5.9 For the current study, it is a logical choice to consider a constant discrete-skeleton for CTMC, i.e. $\forall t_1, t_2 \in \mathbb{R}^+ : p_{ij}(t_1) = p_{ij}(t_2)$. This means that the finite-time discrete transition probability matrix does not change over time.

So, CTMC stays in state i for an exponential amount of time T_i, which is then followed by a jump from i to j based on the p_{ij} of discrete skeleton, and then stays in j for an exponential amount of time T_j.

5.4.3 Absorbing state stochastic process for speed estimation

The estimation of vehicle speed on road is performed using an absorbing state stochastic process. As mentioned before, this is because it is rational to assume that driver can regain the control of vehicle after each perturbation (which means that vehicle speed finally converges to absorbing state).

Definition 5.13 Absorbing state stochastic process $\{\mathbf{s}(t)\}_{t\in\mathbb{Z}^+}$ refers to a process which possesses a number of transient states and a number of absorbing states ($\mathcal{S} = \{s \mid s \in \mathcal{S}_{abs} \cup \mathcal{S}_{trans}\}$). Such a stochastic process almost surely converges to one of the absorbing states and remains there.

Example 5.4 Assume that $\{\mathbf{s}(t)\}_{t\in\mathbb{Z}^+}$ is a stochastic process with $\mathcal{V}_{\mathbf{s}} = \{1, 2, 3, 4, 5\}$, where states $\{1, 5\}$ are absorbing and states $\{2, 3, 4\}$ are transient. The fully connected graph of such a process is shown in Figure 5.9. Note that the process should start from one of the transient states to have a random walk.

Definition 5.14 Time until absorption is denoted by $T_{abs} := \min\{t \in \mathbb{Z}^+ \mid \mathbf{s}(t) \in \mathcal{S}_{abs}\}$.

Consider a finite state-space Markov chain with $\mathcal{S}_{abs} = \{0, 1, ..., p-1\}$ and $\mathcal{S}_{trans} = \{p, ..., q\}$. Figure 5.10 shows the probability matrix obtained by canonical decomposition.

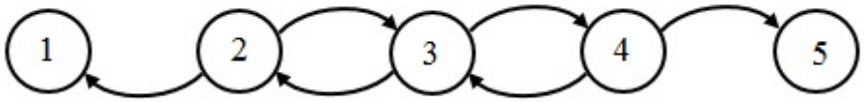

Figure 5.9: The corresponding graph of *Example 5.4*.

$$P = \left[\begin{array}{c|c} Q & R \\ \hline 0 & I \end{array}\right]$$

Figure 5.10: Probability matrix obtained by canonical decomposition.

In the formed matrix, the $p \times p$ matrix Q represents the probability of switching from one transient state to another transient state, the $p \times q-p+1$ matrix R includes the probabilities of switching from a transient state to an absorbing state. I is a $q-p+1 \times q-p+1$ identity matrix stating the fact that the process stays in an absorbing state once it enters the state. Finally, the zero matrix in the down left block represents the fact that the probability of switching from an absorbing state to a transient state is 0.

Remark 5.10 Once the probabilities are obtained by canonical decomposition, based on the initial transient state, one can sample to create the random walk and finally converge to one of the absorbing states. There is no unique way to sample from the matrix. So, an empirical algorithm is implemented to generate the random walk.

Proposition 5.2 *The algorithm implemented works based on the simple fact that the column summation of the elements of canonically decomposed probability is 1. Once starting from one of the transient states, say $i \in \mathcal{S}_{trans}$, it holds that $1 = \sum_{j=0}^{q} P_{ij} = \sum_{j=0}^{p-1} Q_{ij} + \sum_{j=p}^{q} R_{ij}$.In this context, we partition a line of unit length based on the probabilities of each state. So, each partition represents one of the states. Then, draw a random variable from $Unif(0,1)$, and see the number falls in which of the segments, and based on that, jump to the corresponding state.*

Proof. To prove the proposed algorithm works properly, and samples based on the probabilities assigned to each of the states in a specific row of transition probability matrix, it is sufficient to show that by means of uniform distribution, we can sample from a set of discrete random numbers. Let z be a random variable that can take values from a discrete set $\{a_1, a_2, ..., a_k, ...\}$, with corresponding probabilities $\{p_1, p_2, ..., p_k, ...\}$, satisfying the condition $\sum_{k=1} p_k = 1$. By dividing the closed unit interval $[0, 1]$ into $\mathcal{H}_1, \mathcal{H}_2, ..., \mathcal{H}_k, ...$ with $\mathcal{H}_k = (h_{k-1}, h_k]$, such that $h_0 = 0$, $h_k = p_1 + p_2 + ... + p_k$, and $\cup_{k=1} \mathcal{H}_k = [0, 1]$, it can be interpreted that each sub-interval represents a unique value for random variable z. So, it holds that by generating a random variable $u \sim Unif(0, 1)$ and verifying the interval $\mathcal{H}_k$ that the observed value belongs to, one actually assign z to a_k from the actual set, because:

$$\mathcal{P}r(z = a_k) = \mathcal{P}r(u \in \mathcal{H}_k) = \mathcal{P}r(h_{k-1} \leq u \leq h_k) = h_k - h_{k-1} = p_k \ .$$

This completes the proof. □

Example 5.5 Assume that an absorbing state stochastic process with the following information are available:

$$\mathcal{S}_{trans} = \{1, 2\} \ , \mathcal{S}_{abs} = \{3, 4\},$$

$$Q = \begin{bmatrix} 0.2 & 0.3 \\ 0.1 & 0.4 \end{bmatrix} \ , R = \begin{bmatrix} 0.4 & 0.1 \\ 0.3 & 0.2 \end{bmatrix} \ .$$

Let's say the process starts from initial state $s_0 = 2$. Then, $P_{21} = 0.1, P_{22} = 0.4, P_{23} = 0.3$ and $P_{24} = 0.2$. To take a random walk s_1, a random variable u is drawn from $Unif(0, 1)$. Let's say the random variable is $u = 0.56$. Since $0.56 \in (0.5, 0.8]$, we conclude that $s_1 = 3$. Also, note that 3 is an absorbing state, so the process will stay there and no other jump can take place.

Remark 5.11 Let's say $j \in \mathcal{S}_{abs}$. By using the algorithm proposed in *Proposition 5.2* (which is originally implemented to generate a random walk), one can also calculate $\mathcal{P}r(s_{T_{abs}} = j \mid s_0 = i)$. Note that R_{ij} only represents the probability of jumping from i to j, and is different from the probability of absorbing to state j, conditioning on starting from state i.

To ascertain the acceptable precision of the proposed algorithm, a validation test is performed. Fortunately, there is a theoretical formulation to determine the exact value of $\mathcal{P}r(s_{T_{abs}} = j \mid s_0 = i)$ for a given absorbing state stochastic process. By comparing the theoretically obtained value with the empirical one obtained from the proposed algorithm, we can check the performance of the method. Note that since the proposed method is empirical and stochastic, the expected value over independent simulation is compared to the theoretical value.

Theorem 5.4 *Let $\{\mathbf{s}(t)\}_{t\in\mathbb{Z}^+}$ be a finite state absorbing stochastic process with $\mathcal{S}_{abs} = \{0, 1, ..., p-1\}$, $\mathcal{S}_{trans} = \{p, ..., q\}$ and canonically decomposed probability matrix P which has block matrixes Q, R, I and 0, as stated previously. Also, let T_{abs} be the time until absorption. Define $o_{ij} := \mathcal{P}r(s_{T_{abs}} = j \mid s_0 = i)$, and the matrix O such that $O(i, j) = o_{ij}$. Then, it holds that $O = (I - Q)^{-1}R$.*

Proof. The proof is followed by the first step analysis strategy, and the fact that $\{\mathbf{s}(t)\}_{t\in\mathbb{Z}^+}$ has Markov property. By the first step analysis, we get:

$$o_{ij} := \mathcal{P}r(s_{T_{abs}} = j \mid s_0 = i) = \sum_{k=0}^{q} \mathcal{P}r(s_{T_{abs}} = j \mid s_0 = i, s_1 = k)\mathcal{P}r(s_1 = k \mid s_0 = i) \text{ , where}$$

$$\mathcal{P}r(s_{T_{abs}} = j \mid s_0 = i, s_1 = k) = \begin{cases} \mathcal{P}r(s_{T_{abs}} = j \mid s_0 = k) & \text{if } k \in \mathcal{S}_{trans} \\ 1 & \text{if } k = j \\ 0 & \text{if } k \in \mathcal{S}_{abs}, k \neq j \end{cases} \text{ , that implies:}$$

$$o_{ij} = \sum_{k=0}^{p-1} \mathcal{P}r(s_{T_{abs}} = j \mid s_0 = k).p_{ik} + p_{ij} + 0 \times \sum_{\substack{k=p \\ k\neq j}}^{q} p_{ik} = R_{ij} + \sum_{k=0}^{p-1} o_{kj}Q_{ik}.$$

In the matrix form, it gives $O = R + QO$, and thus, $O = (I - Q)^{-1}R$. $\square$

The only remaining issue is to determine the time of absorption, i.e. T_{abs}. It is possible to theoretically determine the expected value of T_{abs}, based on the properties of a given absorbing state stochastic process.

Definition 5.15 The expected time until absorption, condition on starting from transient state $i \in \mathcal{S}_{abs}$ is denoted by $e_i := E[\ T_{abs} \mid s_0 = i\ ]$.

Theorem 5.5 *Let $\{\mathbf{s}(t)\}_{t\in\mathbb{Z}^+}$ be a finite state absorbing stochastic process with $\mathcal{S}_{abs} = \{0, 1, ..., p-1\}$, $\mathcal{S}_{trans} = \{p, ..., q\}$ and canonically decomposed probability matrix P which has block matrixes Q, R, I and 0. Also, let T_{abs} be the time until absorption. Form the vector $\mathbf{e} = \{e_0, ..., e_{p-1}\}$. Then, it holds that $\mathbf{e} = (I - Q)^{-1}\mathbf{1}_{p\times 1}$.*

Proof. The proof can be done by means of the first step analysis, as follows:

$$e_i := E[\ T_{abs} \mid s_0 = i\ ] = \sum_{k=0}^{q} E[\ T_{abs} \mid s_0 = i, s_1 = k\ ]\ \mathcal{P}r(s_1 = k \mid s_0 = i)\ \text{, where}$$

$$E[\ T_{abs} \mid s_0 = i, s_1 = k\ ] = \begin{cases} 1 + E[\ T_{abs} \mid s_0 = k\ ] & \text{if } k \in \mathcal{S}_{trans} \\ 1 & \text{if } k \in \mathcal{S}_{abs} \end{cases}\ \text{, therefore}$$

$$e_i = \sum_{k=0}^{p-1} E[\ T_{abs} \mid s_0 = i, s_1 = k\ ]\ \mathcal{P}r(s_1 = k \mid s_0 = i) + 1 \times \sum_{k=p}^{q} p_{ik} = 1 + \sum_{k=0}^{p-1} e_k Q_{ik}\ .$$

This implies that $\mathbf{e} = Q\mathbf{e} + \mathbf{1}_{p\times 1}$, which means $\mathbf{e} = (I - Q)^{-1}\mathbf{1}_{p\times 1}$. $\square$

Based on the given information, it is possible to simulate the driver's behavior on road using the proposed algorithm, and also to validate it by the result of *Theorem 5.4* and *Theorem 5.5*.

5.5 Simulation Results

This section is organized in two sub-sections. Firstly, the results of desired trajectory generation is given, and thereafter, the simulation pertinent to speed estimation is provided.

5.5.1 Desired trajectory generation

In this sub-section, based on the details given for desired trajectory generation, an algorithm is developed for estimating a desired reference to steer the control commands

towards decreasing vehicle body displacement on road. As mentioned, the most important step is to design a realistic / reasonable graph $\mathcal{G}(\mathcal{V}, \mathcal{E}, W)$ comprising of some nodes / states (representing the possible values of vehicle displacement), and an appropriate set of edges indicating the possibility of jumping from a node to the others (each jump is equivalent to the variation of vehicle displacement), and also to assign proper weights to each of the edges of the considered graph. The second step is to form the mean-weight cycle cover $\mathcal{C} = \{\mathbf{c}_1, \mathbf{c}_2, ..., \mathbf{c}_q\}$. Based on the mentioned information, MLE-TMC can find an optimal desired trajectory. It is worth noting that in all of the cases, when applying the algorithm for generating a trajectory, the initial state can be read from the sensor (the current displacement of vehicle body) and the final state is the stable point 0.

In what follows this sub-section, MLE-TMC is used for a possible simulation scenario to get a desired trajectory. Note that, the goal of this experiment is to validate the performance of the algorithm, which makes it possible to use it in Chapter 8 at the heart of the trajectory building module of LBMPC.

Consider a vehicle which traverses a bumpy road, and at some point passes a pothole or bump which results in the vibration of axels and consequently the displacement of vehicle body. Let's assume that at the time of passing a pothole, the sensor reads the displacement of $-0.05m$ (note that for bump, it could be $0.05m$) and the goal is to make a plausible desired trajectory which takes into account the other unmeasured sources of uncertainty, and guides the tracking control system (LBMPC in our case) such that the final vehicle displacement becomes 0. So, for the considered scenario, $s_0 = -0.05m$ and $s_f = 0m$. Also, let's consider a state set with cardinality of 11 ($\#\mathcal{S} = k = 11$), and a transition probability matrix calibrated based on the prior knowledge of expert, and subjective information regarding the conditional factors affecting the vehicle vibration on road [58], as given below:

$$P = \begin{bmatrix} 0 & 0.3 & 0.3 & 0.2 & 0.2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.3 & 0.3 & 0.2 & 0.2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.3 & 0.3 & 0.2 & 0.2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.3 & 0.3 & 0.2 & 0.2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.1 & 0 & 0.7 & 0.2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.5 & 0 & 0.5 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.2 & 0.7 & 0 & 0.1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.2 & 0.2 & 0.3 & 0.3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.2 & 0.2 & 0.3 & 0.3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.2 & 0.2 & 0.3 & 0.3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.2 & 0.2 & 0.3 & 0.3 & 0 \end{bmatrix},$$

$$\mathcal{S} = \{-0.05, -0.04, -0.03, -0.02, -0.01, 0, 0.01, 0.02, 0.03, 0.04, 0.05\} \ .$$

Also, based on the given information, a graph $\mathcal{G}(\mathcal{V}, \mathcal{E}, W)$ with the following properties is considered for generating optimal desired trajectory:

$$\mathcal{V} = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11\} \ ,$$

$$\mathcal{E} = \{e_{12}, e_{13}, e_{14}, e_{15}, e_{23}, e_{24}, e_{25}, e_{26}, e_{34}, e_{35}, e_{36}, e_{37}, e_{45}, e_{46}, e_{47}, e_{48}, e_{54}, e_{56}, e_{57}, e_{65}, e_{67}, e_{75}$$
$$e_{76}, e_{78}, e_{84}, e_{85}, e_{86}, e_{87}, e_{95}, e_{96}, e_{97}, e_{98}, e_{10,6}, e_{10,7}, e_{10,8}, e_{10,9}, e_{11,7}, e_{11,8}, e_{11,9}, e_{11,10}\} \ ,$$

$$W = \begin{bmatrix} 1.2 & 1.2 & 1.6 & 1.6 & & & & & \\ & 1.2 & 1.2 & 1.6 & 1.6 & & & & \\ & & 1.2 & 1.2 & 1.6 & 1.6 & & & \\ & & & 1.2 & 1.2 & 1.6 & 1.6 & & \\ & & 2.3 & & 0.3 & 1.6 & & & \\ & & & 0.7 & & 0.7 & & & \\ & & & 1.6 & 0.3 & & 2.3 & & \\ & & 1.6 & 1.6 & 1.2 & 1.2 & & & \\ & & & 1.6 & 1.6 & 1.2 & 1.2 & & \\ & & & & 1.6 & 1.6 & 1.2 & 1.2 & \\ & & & & & 1.6 & 1.6 & 1.2 & 1.2 \end{bmatrix} \ .$$

Also, the corresponding graph is shown in Figure 5.11, which is used to form the mean-weight cycle cover $\mathcal{C}$.

Prior to forming the set of mean-weight cycle cover, it is necessary to extract all of the existing cycles from the graph. Figure 5.12 indicates the cycles of the graph, and the resulting acyclic graph.

The mean weight of extracted cycles and acyclic graph are given in Table 5.1.

Now, based on the conditions given in *Definition 5.10*, and matrix W, the ordered mean-weight cycle cover set $\mathcal{C}$ is obtained which is $\mathcal{C} = \{\mathbf{c}_1, \mathbf{c}_{17}, \mathbf{c}_{12}\}$. The first two members fulfill the first condition, and the last member, i.e. $\mathbf{c}_{12}$, is the lowest weight cycle which completes $\mathcal{C}$, by fulfilling the second condition. The graph resulting from removing the nodes of $\mathcal{C}$ is shown in Figure 5.13 . As seen, the resulting graph is directed and acyclic, and satisfies the conditions.

Now that the set $\mathcal{C}$ is achieved, based on *Theorem 5.2*, it is possible to proceed with the calculation of minimum-weight walk $\mathbf{s}_t^*$ using MLE-TMC. As mentioned, the initial state is

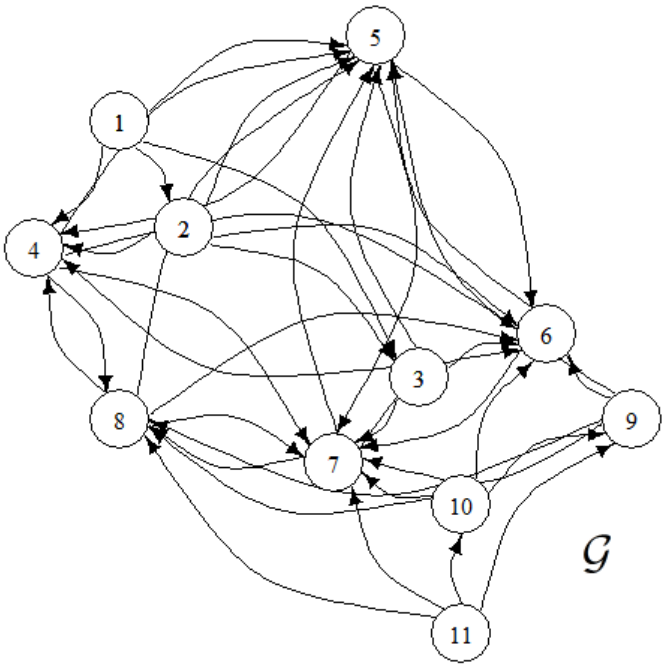

Figure 5.11: The corresponding graph for the considered case study.

$s_0 = -0.05m$. Due to the properties of the considered transition probability matrix, and the fact that the body vibration starts after passing a pothole, the anchor walk including nodes $\mathcal{V} = \{1, 2, 3, 4, 5, 6, 7, 8\}$ and the edge set of $\mathcal{E}_{\mathbf{a}} = \{e_{12}, e_{23}, e_{34}, e_{45}, e_{56}, e_{67}, e_{78}\}$, $w(\mathbf{a}) = 8.1$, and $|\mathbf{a}|_{\mathcal{E}} = 7$ would be a logical choice. Based on that, the transition probability matrix of interest reduces to a 8×8 matrix, and the minimum required edges of the walk to make the use of MLE-TMC possible is 100. Following *Theorem 5.2*, and looping over all members in $\mathcal{C}$, the optimum weight walk is determined, as below:

$$(\mathbf{I}) : \begin{cases} (45 \times |\mathbf{c}_1|_{\mathcal{E}}) + |\mathbf{a}|_{\mathcal{E}} + |\mathbf{c}_{10}|_{\mathcal{E}} = (45 \times 2) + 7 + 3 = 100 \\ \big(45 \times \bar{w}(\mathbf{c}_1)\big) + \bar{w}(\mathbf{a}) + \bar{w}(\mathbf{c}_{10}) = (45 \times 1.75) + 1.15 + 0.86 = 80.76 \end{cases},$$

$$(\mathbf{II}) : \begin{cases} (45 \times |\mathbf{c}_{17}|_{\mathcal{E}}) + |\mathbf{a}|_{\mathcal{E}} + |\mathbf{c}_{10}|_{\mathcal{E}} = (45 \times 2) + 7 + 3 = 100 \\ \big(45 \times \bar{w}(\mathbf{c}_{17})\big) + \bar{w}(\mathbf{a}) + \bar{w}(\mathbf{c}_{10}) = (45 \times 0.5) + 1.15 + 0.86 = 24.51 \end{cases},$$

$$(\mathbf{III}) : \begin{cases} (22 \times |\mathbf{c}_{12}|_{\mathcal{E}}) + |\mathbf{a}|_{\mathcal{E}} + |\mathbf{c}_{16}|_{\mathcal{E}} = (22 \times 4) + 7 + 5 = 100 \\ \big(22 \times \bar{w}(\mathbf{c}_{12})\big) + \bar{w}(\mathbf{a}) + \bar{w}(\mathbf{c}_{16}) = (22 \times 1.45) + 1.15 + 1.22 = 34.27 \end{cases}.$$

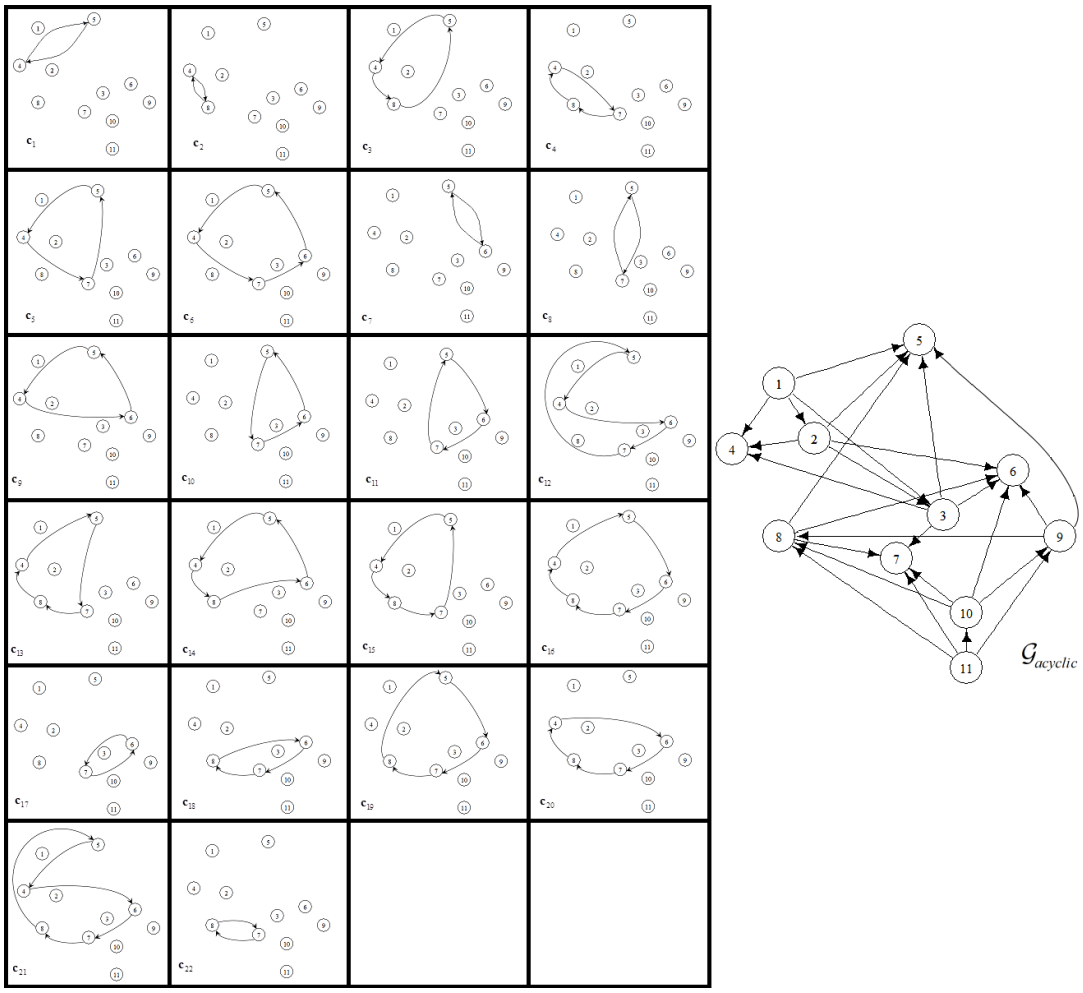

Figure 5.12: (a) Extracted cycles and (b) acyclic graph obtained from $\mathcal{G}$.

From the above scenarios, the walk obtained by (**II**) possesses the minimum mean weight, and is therefore selected. The corresponding desired trajectory is shown in Figure 5.14.

As can be seen, the trajectory resulting from vibration sounds logical from mechanical viewpoint. At the beginning steps, it tries to mitigate the significant oscillation and gradually, after getting rid of remarkable vibration, it mitigates the amplitude of vibration and remains in the safe bound of $0.01m$ which steers the controller's commands such that the vehicle body displacement be damped over the control horizon.

Table 5.1: The mean weight of extracted cycles and acyclic graph

	Sum of weights	**Number of edges**	**Mean weight**
$\mathbf{c}_1$	3.5	2	1.75
$\mathbf{c}_2$	3.2	2	1.60
$\mathbf{c}_3$	5.5	3	1.83
$\mathbf{c}_4$	5.5	3	1.83
$\mathbf{c}_5$	5.5	3	1.83
$\mathbf{c}_6$	4.9	4	1.22
$\mathbf{c}_7$	1	2	0.50
$\mathbf{c}_8$	3.2	2	1.60
$\mathbf{c}_9$	4.2	3	1.40
$\mathbf{c}_{10}$	2.6	3	0.86
$\mathbf{c}_{11}$	2.6	3	0.86
$\mathbf{c}_{12}$	5.8	4	1.45
$\mathbf{c}_{13}$	6.7	4	1.67
$\mathbf{c}_{14}$	5.8	4	1.45
$\mathbf{c}_{15}$	6.7	4	1.67
$\mathbf{c}_{16}$	6.1	5	1.22
$\mathbf{c}_{17}$	1	2	0.50
$\mathbf{c}_{18}$	4.2	3	1.40
$\mathbf{c}_{19}$	4.9	4	1.22
$\mathbf{c}_{20}$	5.8	4	1.45
$\mathbf{c}_{21}$	8.1	5	1.62
$\mathbf{c}_{22}$	3.5	2	1.75
$\mathcal{G}$	33.6	24	1.40

Note that, for this simulation, we consider a similar sojourn time for all of the jumps which can be simply changed based on the results of *Theorem 5.3*. Using such a trajectory also ensures the controller designer that the effect of some unmeasured sources of uncertainty is taken into account, which is quite desirable comparing to considering a constant reference trajectory of 0 for all points of the process. Also, the method has a remarkable advantage from computational point of view. As seen, once the anchor walk, free cycles and mean-weight cycle are achieved, the underlying optimization problem is simplified to some trivial calculations, and there is no need for using a complex optimization algorithm or heuristic searching process. This is very beneficial, as LBMPC is a real-time controller, and requires fast computation in all of its modules (including the desired trajectory building module).

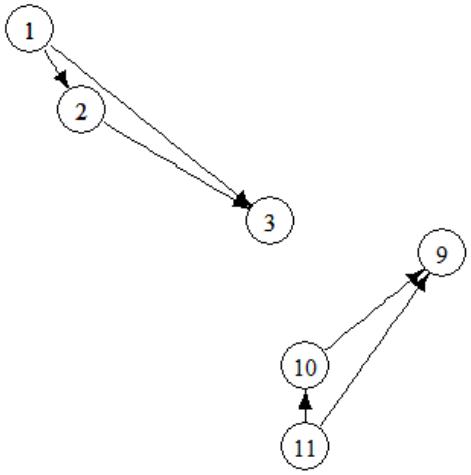

Figure 5.13: Graph resulting from removing nodes of $\mathcal{C}$ from $\mathcal{G}$.

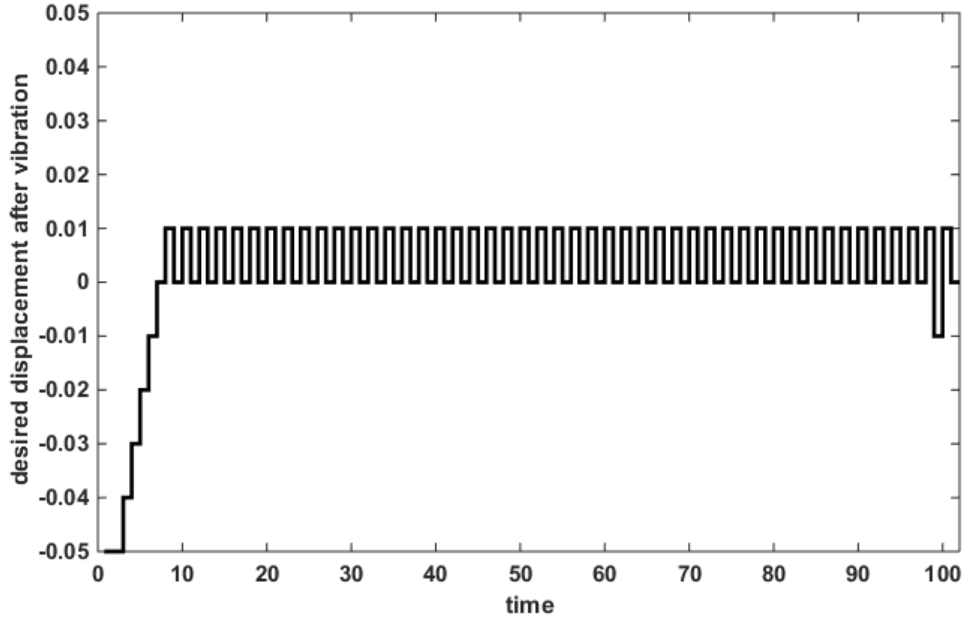

Figure 5.14: Desired trajectory obtained by MLE-TMC.

5.5.2 Vehicle speed estimation

As mentioned, for the current simulation, driver's behavior is modelled by the variation of speed on road, since it is the most important factor related to vehicle suspension vibration control. Driver may change the speed due to different external conditions, such as passing a significant bump or pothole, sudden change in the driving behavior of adjacent vehicles, and *etc.*

Among the mentioned issues, the most realistic external condition which is available and can be considered for simulation is the first element, i.e. passing a significant bump or pothole. In this context, during the movement of vehicle on road, if the road roughness estimated in Chapter 4 exceeds a pre-defined threshold, a random perturbation is imposed to vehicle, which deviates the speed, and consequently results in a set of actions from driver to re-gain the control of vehicle (return to the standard cruise speed). This process is modelled by absorbing state stochastic process. The details of the roughness threshold for perturbing vehicle speed are given in the simulation setup of controller in Chapter 8. Here, the goal is to validate the performance of the considered process for simulating driver's behavior after the perturbation of speed.

Case 1: Let's consider the case in which the cruise speed of vehicle is $V = 40\ m/s$, and the vehicle encounters a pothole on road which causes the perturbation of speed. The event is modelled by an absorbing state stochastic process with $\mathcal{S}_{trans} = \{1, 2, 3, 4\}$ and $\mathcal{S}_{abs} = \{5, 6\}$, for which the following information is obtained after canonical decomposition:

$$Q = \begin{bmatrix} 0.1 & 0.2 & 0.2 & 0.1 \\ 0.1 & 0.2 & 0.3 & 0.1 \\ 0.2 & 0.2 & 0.2 & 0.2 \\ 0.1 & 0.3 & 0.2 & 0.1 \end{bmatrix} , R = \begin{bmatrix} 0.3 & 0.1 \\ 0.1 & 0.2 \\ 0.1 & 0.1 \\ 0.1 & 0.2 \end{bmatrix} .$$

Also, the states represent the following speeds:

$$\mathcal{V}_{trans} = \{V_1 \equiv 35\ m/s,\ V_2 \equiv 38\ m/s,\ V_3 \equiv 43\ m/s,\ V_4 \equiv 45\ m/s\} ,$$
$$\mathcal{V}_{abs} = \{V_5 \equiv 40\ m/s,\ V_6 \equiv 41\ m/s\} .$$

Also, right after perturbation, assume that the process starts from the initial transient state 2, i.e. $s_0 = 2$. Now, based on the algorithm proposed in *Proposition 5.2*, the process is simulated. Also, to make the results compatible with the predictive controller, assume that we are interested in the first $N = 10$ walks of the stochastic process, where N is the horizon length for the predictive controller. Figure 5.15 shows the obtained walks for 4 different simulation.

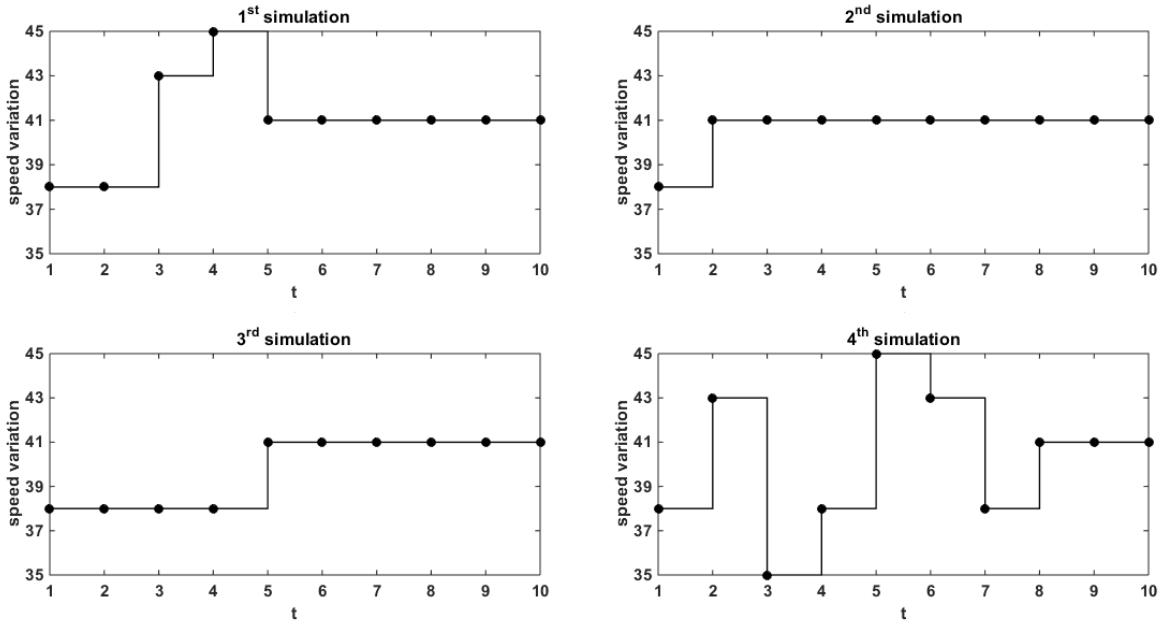

Figure 5.15: Speed estimation for the first considered scenario over 4 independent runs.

It can be observed that the proposed algorithm does a very good job, and estimated the vehicle speed after perturbation until absorption. In all of the plotted scenarios, the absorption occurs within the horizon length. The obtained results shown that the considered algorithm is a good choice to be used at the heart of controller for stochastic estimation of vehicle speed after a random perturbation. The simulation was terminated in 0.001063 *sec* which also shown the computational efficiency of the proposed algorithm, and its compatibility with online control algorithms, such as LBMPC. Note that the simulation for the considered scenario was also conducted by taking the other 3 transient states as the initial state, and in all of the cases, a similar promising result was achieved.

To further examine the performance of the proposed algorithm, it is applied to a more complicated scenario with more states, and a more sophisticated probability matrixes.

Case 2: Let's consider a vehicle with cruise speed of $V = 40\ m/s$. Again, as a result of an external disturbance, vehicle speed is perturbed. For this case, the event is modelled by an absorbing state stochastic process with $\mathcal{S}_{trans} = \{1, 2, 3, 4, 5, 6, 7, 8\}$ and $\mathcal{S}_{abs} = \{9, 10, 11\}$, for which the following information is obtained after canonical decomposition:

$$Q = \begin{bmatrix} 0.1 & 0.2 & 0.1 & 0.1 & 0.05 & 0.05 & 0.05 & 0.1 \\ 0.1 & 0.1 & 0.05 & 0.1 & 0.1 & 0.15 & 0.1 & 0.15 \\ 0.1 & 0.05 & 0.05 & 0.1 & 0.1 & 0.05 & 0.1 & 0.05 \\ 0.1 & 0.1 & 0.1 & 0.1 & 0.05 & 0.1 & 0.05 & 0.1 \\ 0.05 & 0.05 & 0.05 & 0.1 & 0.1 & 0.05 & 0.1 & 0.1 \\ 0.05 & 0.1 & 0.05 & 0.1 & 0.1 & 0.05 & 0.05 & 0.1 \\ 0.1 & 0.1 & 0.1 & 0.1 & 0.05 & 0.05 & 0.05 & 0.05 \\ 0.15 & 0.05 & 0.1 & 0.1 & 0.05 & 0.15 & 0.1 & 0.05 \end{bmatrix}, \; R = \begin{bmatrix} 0.05 & 0.1 & 0.1 \\ 0.05 & 0.05 & 0.05 \\ 0.1 & 0.2 & 0.1 \\ 0.1 & 0.1 & 0.1 \\ 0.1 & 0.2 & 0.1 \\ 0.15 & 0.15 & 0.1 \\ 0.15 & 0.1 & 0.15 \\ 0.05 & 0.15 & 0.05 \end{bmatrix}.$$

Also, the states represent the following speeds:

$$\mathcal{V}_{trans} = \begin{Bmatrix} V_1 \equiv 35\ m/s, & V_2 \equiv 36\ m/s, & V_3 \equiv 37\ m/s, & V_4 \equiv 38\ m/s, \\ V_5 \equiv 42\ m/s, & V_6 \equiv 43\ m/s, & V_7 \equiv 44\ m/s, & V_8 \equiv 45\ m/s \end{Bmatrix},$$

$$\mathcal{V}_{abs} = \{V_5 \equiv 40\ m/s,\ V_6 \equiv 41\ m/s\}\ .$$

Also, right after perturbation, assume that the process starts from the initial transient state 2, i.e. $s_0 = 2$. Also, let $N = 20$. Figure 5.16 shows the obtained walks for 4 different simulation.

As seen, even after considering a more complicated scenario with more states, and more uniform-like distribution of probabilities among the states (including transient and absorbing ones), the proposed algorithm can efficiently estimate the variation of speed after perturbation untill absorption happens. Again, it can be seen that the absorption occurs before the end of horizon length, which is a desired issue from control view point. This is because converging to stable cruise speed reduces one of the sources of disturbances and helps the controller make a better decision. The average simulation time for this case is 0.000487 *sec*, which is very promising. Also, it can be seen that in most of the simulated walks, the jump between the states is logical and there is not a big variation in the speed (except for the very initial point of the 1^{st} simulated chain). Note that this can happen, since in the considered case scenario, the probability to jump from one state to all of the other states is non-zero. Also, even for the biggest jump (which happens at the very beginning of the 1^{st} simulated chain), the variation of speed is 7 m/s which is still completely realistic. All in all, the results of the performed simulation are promising, and are in favor of using absorbing state stochastic process for estimating the variation of speed after a sudden perturbation on road.

In the final part of the simulation, it is intended to compare the results of the proposed absorbing chain generation algorithm with those obtained by theoretical probabilistic model. Assume that $j \in \mathcal{S}_{abs}$, the goal is to estimate $\mathcal{P}r(s_{T_{abs}} = j \mid s_0 = i)$. As

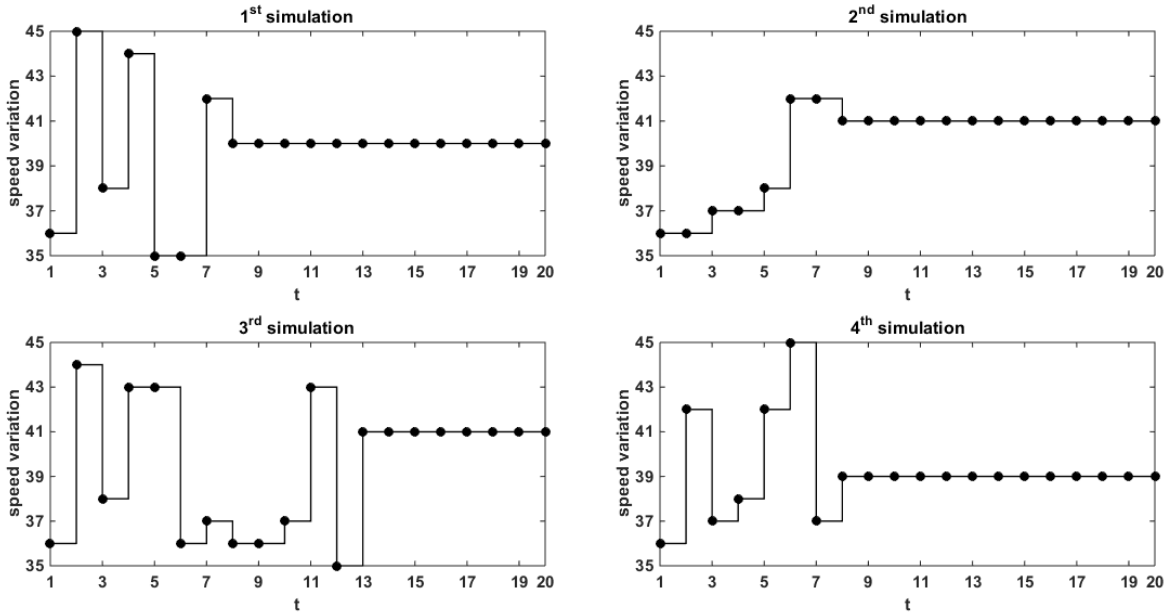

Figure 5.16: Speed estimation for the second considered scenario over 4 independent runs.

mentioned, since the proposed method is empirical and stochastic, the expected values over 1000 independent simulation are compared to the theoretical values. The theoretical values of expected absorption time and the probability of absorption to a certain state given starting from a certain transient state can be calculated by means of *Theorem 5.4* and *Theorem 5.5*. In this way, there are 8 conditions for *Case 1* and 24 conditions for *Case 2* which should be checked. Table 5.2 summarizes the result of simulation for all of the considered cases. As it can be seen from the obtained results, the expected theoretical and empirical values are very close to each other in most of the cases. The acceptable accuracy holds for both simulation cases, though the second case can be viewed as a challenging case with a considerable number of transient and absorbing states. Note that, during the simulation, it was observed that the empirical method can sometimes yield inaccurate results, which is logical due to the probabilistic nature of the both algorithm and problem at hand. But, all in all, the obtained results are very interesting, and especially, in almost all of the cases, the results indicate the generated chain is absorbed to one of the absorbing states (which is of high interest when the algorithm is used at the heart of LBMPC). Also, Table 5.2 reports the theoretical expected time until absorption. The simulation indicate that, to generate a more accurate chain, the proposed algorithm usually requires more

than the theoretical absorbing time. For our simulation, it was observed that for some of the cases, the algorithm requires even 20 steps to converge to one of the absorbing states. However, in general, it is possible to generate an absorbed chain in a time close to the theoretical expected value.

Table 5.2: The simulation result for the considered scenarios

Case No.	$\mathbf{T_{abs}}$	**Initial transient state**	**Destined absorbing state**	**Theoretical value**	**Empirical value**
1	3	1	5	0.5860	0.5520
1	3	1	6	0.4140	0.5050
1	3	2	5	0.4348	0.4300
1	3	2	6	0.5652	0.5800
1	4	3	5	0.4876	0.4860
1	4	3	6	0.5124	0.4440
1	3	4	5	0.4295	0.5330
1	3	4	6	0.5705	0.6420
2	3	1	9	0.2688	0.3580
2	3	1	10	0.4089	0.4940
2	3	1	11	0.3223	0.3070
2	4	2	9	0.3010	0.3770
2	4	2	10	0.3990	0.3430
2	4	2	11	0.2999	0.3420
2	3	3	9	0.2762	0.1770
2	3	3	10	0.4438	0.4800
2	3	3	11	0.2799	0.2020
2	3	4	9	0.3049	0.3720
2	3	4	10	0.3890	0.4410
2	3	4	11	0.3061	0.3810
2	3	5	9	0.2764	0.2820
2	3	5	10	0.4461	0.3470
2	3	5	11	0.2775	0.2710
2	3	6	9	0.3252	0.3300
2	3	6	10	0.3987	0.5020
2	3	6	11	0.2761	0.1960
2	3	7	9	0.3250	0.3160
2	3	7	10	0.3463	0.2820
2	3	7	11	0.3286	0.4700
2	3	8	9	0.2722	0.2790
2	3	8	10	0.4540	0.4760
2	3	8	11	0.2738	0.3040

5.6 Findings

Based on the conducted simulation, the following points were inferred regarding the potential of stochastic process for handing desired trajectory generation and speed estimation tasks:

1. The conducted simulation indicated that the algorithm used for designing the most likely desired trajectory has a promising performance from mechanical point of view, which is quite important for the design of suspension controller. It was observed that at the very beginning of the vibration where significant oscillations are expected, MLE-TMC tries to force the controller to send significant actuation signals (force) to suspension system to damp the vibration. Thereafter, the desired trajectory imposes a safe oscillation around the equilibrium point until the vehicle displacement becomes 0.

2. From the computational point of view, it was proven that MLE-TMC has a quite efficient performance. Unlike most of the optimization algorithms which require a considerable computational time, MLE-TMC can find the most likely optimum trajectory with trivial heuristic-type calculations, provided that the graph information including anchor walk, mean-weight cycle cover, and free cycles be available.

3. The algorithm proposed for speed estimation can be easily generalized for whatever scenario that is designed for simulating the driver's perception on road. As seen, for both of the considered scenarios, the algorithm acceptably produced chains which converged to one of the absorbing states within the predetermined horizon length (N). Also, the computational time required for generating absorbing chains was very short. Such facts prove the compatibility of the method to be used at the heart of LBMPC for the estimation of vehicle speed in an online fashion.

4. The salient asset of the absorbing chain generation algorithm was its simple implementation concept. By comparing the results of the method with those obtained from theoretical probabilistic model, it was observed that the empirical chains suggested by the algorithm have enough accuracy. Hence, the method can be accepted as a tool for modeling the unknown driver's perception on road.

Note

All MATLAB and R codes pertaining to the simulation performed in this chapter can be found in Appendix of the thesis.

Chapter 6

Learning Based Model Predictive Control

In this chapter, the mathematical formulation of LBMPC together with the mathematical proofs for guaranteeing the safety of its performance in real-time are presented. The chapter is organized in different sections. Firstly, a precise and concise review of the state-of-the-art of LBMPC is presented, and it is mentioned how this control strategy differs from the existing learning based predictive controllers proposed so far. Thereafter, the general formulation of LBMPC is presented. An effective method for the approximation of invariant set is presented. Also, conditions and theorems required for satisfying the epi-convergence of oracles (in both parametric and non-parametric forms) used in LBMPC are presented. At the next step, an investigation is carried out into the potential optimization algorithms and encoding strategies for the calculation of optimum controlling policy. Finally, the state-space representation of suspension control is given.

Throughout the chapter, the terms *controlling policy*, *controlling command*, and *actuation signal* are used interchangeably. It should be stressed that it is a common tradition within the control engineering society to call the system to be controlled as *plant*. So, the words *plant* and *real-system* are also used interchangeably.

6.1 A Concise Review

As far as the author is concerned, this is the first time that a complete review of the state-of-the-art of LBMPC is presented. Over the past two decades, learning based adaptive and

predictive controllers have attracted an increasing interest from the researchers of control engineering society. Before starting the literature review, it should be mentioned that some research papers have been published under the name of learning based predictive controllers and even learning based model predictive control [137, 138, 139]. Although that proposals may share some similarity in their structure and implementation with LBMPC, they are not as comprehensive. Indeed, a deliberate and wise manipulation is embedded into the structure of LBMPC which distinguishes its functioning from all of the other variants of learning based model predictive controllers.

LBMPC was initially proposed by a number of control theoreticians working at Hybrid System Laboratory at University of California, Berkeley (UC Berkeley) [21]. The salient asset of LBMPC lies in its theoretical foundation and rigorous analysis which enables it to (a) handle state and input constraints, (b) take advantage from statistical learning theory to identify model uncertainty, (c) search for optimal control policies with respect to a convex cost function, and (d) be provably convergent [18, 140]. From what stated, one can realize that LBMPC is a kind of robust, adaptive, and optimal predictive controller. LBMPC uses a decoupled architecture which enables it to handle the performance and safety criteria separately using well-known theories from reachability analysis [141, 142]. To be more specific, the performance is optimized by determining actuation signals as solutions of a convex optimization problem, while the safety and robustness of LBMPC comes from the well-established methods from the field of robust model predictive control [143, 144]. One of the other advantages which is associated with this innovative decoupled structure is that LBMPC is robust to mis-learning. By robustness against mis-learning, we mean that even if the oracle used in LBMPC is badly tuned or poorly designed, the provably robustness performance of LBMPC provides safety. Fortunately, this distinct feature of LBMPC was proven theoretically [18] and experimentally [21, 145].

Since LBMPC is rather a recent spotlighted controller, it has not been vigorously utilized by practitioners, and a limited number of published reports is available in the literature. However, the existing published papers are authorized by some of the most well-known scholars, and due to its provably safe performance and rational computational complexity, it can be viewed as a very good choice for practical applications.

In [18], a seminal investigation was carried out and different theories from statistical learning, robust MPC, discrete stability analysis, optimal control and probabilistic theories were taken into account to prove the safe performance of LBMPC. A supplementary paper to [18] were also published in [140] which completed the theoretical results on epi-convergence of non-parametric and parametric oracles. To elaborate on the theoretical findings, three real-time control problems, i.e. energy-efficient building automation, quadrotor helicopter flight control, and Moore-Greitzer compressor control, were taken

into account. By comparing the performance of LBMPC against linear MPC and nonlinear MPC (NMPC), it was observed that the accuracy of LBMPC is as good as that of NMPC, and is also quite better than MPC. However, unlike NMPC, LBMPC did not account for the nonlinearities of the model, which was quite promising. This is because accurate modeling of the nonlinearities of real systems is very arduous, and increases the computational complexity of the control-oriented model. Based on the obtained results, some open issues were also reported which could be a topic for further investigation. Most importantly, it was stated that selecting the most efficient learning technique among the existing parametric and non-parametric methods remained an open problem. It is worth pointing out that one of the main objectives of the current thesis is to answer this open issue. To do so, in Chapter 7, a wide range of learning methods are adopted from literature, and are used as oracles in the architecture of LBMPC. Also, it will be tried to extend the theoretical proofs to the adopted learning algorithms for guaranteeing the epi-convergence.

After the publication of the seminal paper, investigation on LBMPC has been continued by considering different architectures for LBMPC and applying them to more complicated control problems. In [146], hybrid systems modeling as well as hybrid systems control theory were used for developing a hybrid system LBMPC to improve the energy-efficiency of heating, ventilation, and air-conditioning (HVAC) systems. Through simulation, it was indicated that hybrid modeling enabled the authors to simplify the original model of HVAC, and hybrid system LBMPC used the hybrid simplified model to improve the energy-efficiency. Significant improvement was achieved through simulation while the occupancy comfort constraints were not violated.

Research on using LBMPC for HVAC control was continued by applying the controller to reducing the transient and steady state electricity consumption of HVAC [147]. To do so, experimental data was collected using Berkeley retrofitted and inexpensive HVAC testbed for energy efficiency (BRITE), and a novel parameter identification algorithm was combined with the physics based model for measuring the electrical energy consumption of BRITE. Also, a convex relaxation method was adopted to render the original function to a convex format which could be used by LBMPC for calculating the controlling commands. Random sampling was used to capture an appropriate signal for developing the oracle. The comparative simulation results indicated that LBMPC was a very good candidate to be used for BRITE control.

In [148], an extension of LBMPC was proposed which used an efficient numerical algebraic optimizer together with an extended dual Kalman filter for the estimation of unmeasured states. The efficacy of LBMPC was proven by applying it to real-time control of a quadrotor helicopter. Also a technique was proposed for the approximation of maximal output admissible disturbance invariant set. In [148], the simulation was continued by ap-

plying the proposed variant of LBMPC to a very challenging robotics problem, known as ball catching. Through repeated experiments, it was observed that quadrotor with LBMPC can catch the thrown balls for more 90% of trials. In [144], a comprehensive report was provided to elaborate on the design, implementation and experiments for quadrotor control.

In [149], numerical experiments were conducted to improve the optimization problem at the heart of LBMPC. In this context, two benchmark control problems called hovering control for the quadrotor and on-board control experiments of dynamic quadrotor flight were taken into account. Also, three optimization algorithms, i.e. primal-dual infeasible start interior point method and two variants of dense active set solvers, were adopted. The results of the simulation indicated that each of optimization techniques were suited for certain applications, and it was recommended that for any given control problem, a number of optimization techniques should be considered to extract the best one.

All of the recommendations and findings of the above researches are considered by the author for designing LBMPC for suspension system control. In this regard, this investigation tries to answer some of the open questions and enrich the literature of LBMPC.

6.2 General Architecture and Formulation of LBMPC

In this section, the general architecture of LBMPC along with the theoretical foundation required for the safe operation of LBMPC are presented. Understanding these general theories and concepts, along with the definitions and lemmas behind the development of LBMPC is essential for the extensions done in this research as well as for specific adjustment of LBMPC for suspension control. Therefore, it is tried to precisely formulate the general LBMPC to make it clear how the combination of different modules enables this controller to be robust, optimal, stable and probabilistically convergent. To give a clear vision of the block-diagram architecture of LBMPC and its formation, a schematic illustration of LBMPC is presented in Figure 6.1. As can be seen, LBMPC consists of (1) a nominal module with disturbance, (2) a learnable module, (3) an optimization module, and (4) a reference trajectory builder. Having said that the trajectory builder is an extension to the basic LBMPC, and is required since the suspension control problem can be viewed as a trajectory tracking problem. The details of reference trajectory builder were given in Chapter 5, and is not related to this chapter. So, it will be viewed as an abstract component during the analysis.

Note that, for the sake of completeness, the proofs of the important theorems of LBMPC are presented in details. Before starting the formulation of different modules of LBMPC and

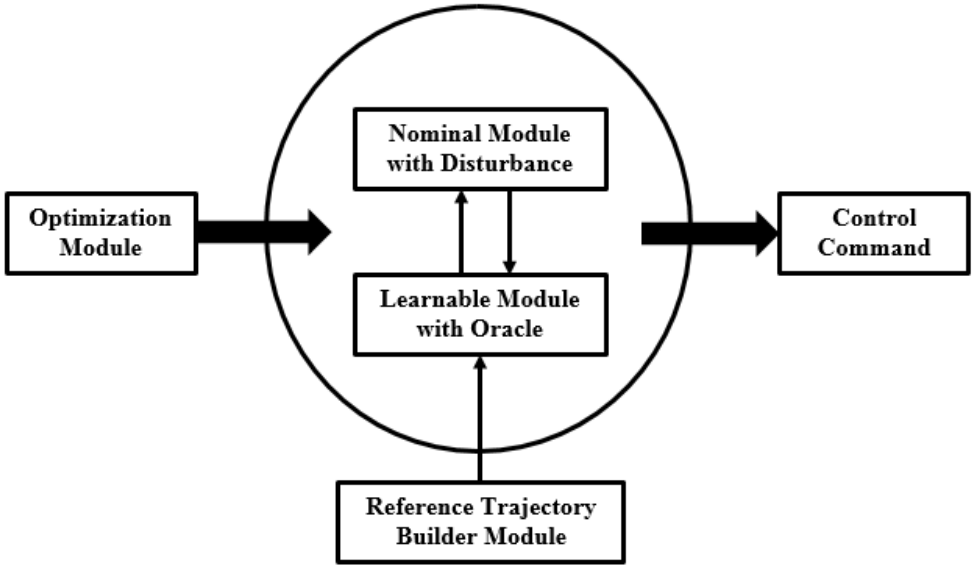

Figure 6.1: Schematic illustration of LBMPC architecture.

definitions, lemmas and theories associated with them, that would be necessary to present some notations and well-known definitions in order to be consistent with the literature of automation and control.

6.2.1 Notations and preliminaries

Firstly, one should notice that vectors are shown by lower case bold format ($\mathbf{a}$, $\mathbf{b}$ and *etc.*), matrixes are shown by upper case italic format (A, B and *etc.*), and sets are shown by calligraphic upper case italic format ($\mathcal{A}$, $\mathcal{B}$ and *etc.*). As mentioned, LBMPC works with (1) true system dynamics, (2) nominal system dynamics, and (3) learned system dynamics, and these three scenarios are distinguished in the carried out analysis. In this context, $\mathbf{u} \in \mathbb{R}^m$, $\mathbf{x} \in \mathbb{R}^n$, and $\mathbf{y} \in \mathbb{R}^q$ indicate the control input vector, state vector and output vector of the true model, respectively. $\bar{\mathbf{u}} \in \mathbb{R}^m$, $\bar{\mathbf{x}} \in \mathbb{R}^n$, and $\bar{\mathbf{y}} \in \mathbb{R}^q$ denote the control input vector, state vector and output vector of the nominal model with bounded disturbance, respectively. $\tilde{\mathbf{u}} \in \mathbb{R}^m$, $\tilde{\mathbf{x}} \in \mathbb{R}^n$, and $\tilde{\mathbf{y}} \in \mathbb{R}^q$ denote the control input vector, state vector and output vector of the model with oracle, respectively. Sub-scripts denote the time. For example, $\mathbf{u}_t$ represents the value of control input vector at set-point t. All vectors are column vectors with the corresponding number of elements, e.g. $\mathbf{u}_t = col(u_{1,t}, u_{2,t}, ..., u_{m,t})$.

Also, A^T is used to show the transpose of matrix A. Let's assume that $\big(\mathbf{a};\mathbf{b}\big)$ concatenates the two vectors $\mathbf{a}$ and $\mathbf{b}$. For example, if $\mathbf{z} = \big(\mathbf{x};\mathbf{u}\big)$, then $\mathbf{z} \in \mathbb{R}^{n+m}$. Moreover, $[A\ B]$ concatenates two matrixes A and B with the same number of rows, and possibly different number of columns. A positive definite matrix A is shown by $A \succ 0$, and for two matrixes A and B, the condition $A \succ B$ implies $A-B \succ 0$. The term $\|\cdot\|_2$ is the standard Euclidian (L_2) norm, and $\|\cdot\|_A$, where A is a matrix, represents the weighted norm. For example, $\|\mathbf{a}\|_A = \sqrt{\mathbf{a}^T A\mathbf{a}}$. The term $\|\cdot\|_{Frobenius}$ represents the Frobenius matrix norm which is used when it is intended to calculate the norm of a matrix. A vector of zeros of appropriate size (for example $n \times 1$) is shown by $\mathbf{0}_{n\times 1}$. Also, an identity matrix of appropriate size (e.g. $n \times n$) is shown by $\mathbb{I}_n$. The operator $int(\mathcal{T})$ represents the interior points of the set $\mathcal{T}$. For an open set $\mathcal{P}$, the closure of the set is denoted by $clo(\mathcal{P})$. Finally, assume that $\mathbf{a} \in \mathbb{R}^{n_a}$ and $\mathbf{b} \in \mathbb{R}^{n_b}$, and the set $\mathcal{T} \subset \mathbb{R}^{n_a+n_b}$, then, the projection operator is defined as $Proj_{\mathbf{a}}(\mathcal{T}) := \{\mathbf{a} \in \mathbb{R}^{n_a} \mid \exists \mathbf{b} \in \mathbb{R}^{n_b}, \big(\mathbf{a};\mathbf{b}\big) \in \mathcal{T}\}$.

Definition 6.1 [150, 151] A given function $\alpha : \mathbb{R}^+ \to \mathbb{R}^+$ is called *type-*$\mathcal{K}$ if $\alpha(0) = 0$, and α be continuous and strictly increasing. Having said that function α differs from the class of *type-*$\mathcal{K}_\infty$ functions, in the sense that the condition $\alpha(h) = \infty$ does not hold as $h \to \infty$, and thus, there exists a finite upper bound for the range of α.

Definition 6.2 [151, 152] A given function $\beta : \mathbb{R}^+ \times \mathbb{R}^+ \to \mathbb{R}^+$ is called *type-*$\mathcal{KL}$ if for fixed $h \geq 0$, $\beta(\cdot, h)$ becomes a *type-*$\mathcal{K}$ function, and for each fixed $s \geq 0$, $\beta(s, \cdot)$ becomes non-increasing, and satisfies the asymptotic property $\lim\limits_{h\to\infty} \beta(s, h) = 0$

Definition 6.3 [153] Let's assume that the state space is constrained by polytope $\mathcal{X}$ (i.e. $\mathbf{x} \in \mathcal{X}$), and denote the steady-state point by $\bar{\mathbf{x}}_{\mathbf{stable}}$, then the function $V_t : \mathcal{X} \to \mathbb{R}^+$ is said to be a Lyapunov function for a discrete-time process if for every set-point $t \in \mathbb{Z}^+$, $\bar{\mathbf{x}}_{\mathbf{stable}}$ lies in the interior of the domain of V_t, $V_t(\bar{\mathbf{x}}_{\mathbf{stable}}) = 0$, $V_t(\mathbf{x}_t) > 0$ for $\forall \mathbf{x}_t \neq \bar{\mathbf{x}}_{\mathbf{stable}}$ and $V_{t+1}(\mathbf{x}_{t+1}) - V_t(\mathbf{x}_t) < 0$ for $\forall \mathbf{x}_t \neq \bar{\mathbf{x}}_{\mathbf{stable}}$ of dynamics system. Also, $\alpha_1\big(\|\mathbf{x} - \bar{\mathbf{x}}_{\mathbf{stable}}\|_2\big) \leq V_t(\mathbf{x}) \leq \alpha_2\big(\|\mathbf{x} - \bar{\mathbf{x}}_{\mathbf{stable}}\|_2\big)$, where α_1 and α_2 are two deliberately selected *type-*$\mathcal{K}$ functions.

Remark 6.1 The steady-state point is defined by a bar on the top of $\mathbf{x}$ (note that using bar is the standard notation for the representation of the nominal model with bounded disturbance), as all of the robustness and stability conditions are verified and proven by the dynamical behavior of the nominal model with bounded disturbance. Later, it will be indicated that this innovative strategy results in the retaining of robust and stable performance even under deficient mis-learning conditions.

Definition 6.4 [154] Let $\mathcal{A}$, $\mathcal{B}$, and $\mathcal{C}$ be three different sets defined in some vector space, then, the Minkowski sum operator is defined as $\mathcal{A} \oplus \mathcal{B} := \{\mathbf{a} + \mathbf{b} \mid \mathbf{a} \in \mathcal{A}; \mathbf{b} \in \mathcal{B}\}$, and the Pontryagin (Minkowski) set difference is defined as $\mathcal{A} \ominus \mathcal{B} := \{\mathbf{a} \mid \mathbf{a} \oplus \mathcal{B} \subseteq \mathcal{A}\}$. Also, the linear mapping of the set $\mathcal{A}$ by matrix T is $T\mathcal{A} = \{T\mathbf{a} \mid \mathbf{a} \in \mathcal{A}\}$. In the proofs of robustness, we will use these sets together with some of their important properties. In particular, we use the properties

$(\mathcal{A} \ominus \mathcal{B}) \oplus \mathcal{B} \subseteq \mathcal{A}$, $\mathcal{A} \ominus (\mathcal{B} \oplus \mathcal{C}) \oplus \mathcal{C} \subseteq \mathcal{A} \ominus \mathcal{B}$, and $T(\mathcal{A} \ominus \mathcal{B}) \subseteq T\mathcal{A} \ominus T\mathcal{B}$. Also, it holds that $\mathcal{A} \subseteq \mathcal{B} \iff \mathcal{A} \oplus \mathcal{C} \subseteq \mathcal{B} \oplus \mathcal{C}$.

Definition 6.5 [120] For a random sequence $\hbar_t$, a constant η, and a rate r_t, the mathematical term $\|\hbar_t - \eta\|_2 = O_p(r_t)$ means that there exists $\epsilon > 0$ such that for $\exists M > 0$ and $\exists\ t_f \in \mathbb{Z}^+$ and $t_f > 0$, we have $\mathcal{P}r\big(\|\hbar_t - \eta\|_2 / r_t > M\big) < \epsilon$, for $\forall t > t_f$. Now, let's define $\hbar_t \xrightarrow{p} \eta$ which represents $\|\hbar_t - \eta\|_2 = O_p(r_t)$, as $r_t \to 0$.

Based on the above definitions and notations, we can start writing down the formulation of LBMPC.

6.2.2 Formulation of true model dynamics

In this sub-section, the true model of system dynamics is presented. It is assumed that for a given plant which is going to be controlled, there exists an exact representation of system dynamics. Note that this true model is just considered as an ideal reference, not a model to be used at the heart of LBMPC for the calculation of control commands.

As we know, it is almost impossible to calibrate a system of ordinary differential equations (ODEs) in either linear or nonlinear formats and assert that it exactly follows the real-system's dynamical behavior. Unfortunately, such a mistake is common within the society of control engineering who use model-based schemes for designing their controlling algorithms. As mentioned, the beauty of LBMPC is that the true model is just considered as an ideal reference, and the formulation of controller is carried out keeping in mind that it is very optimistic to postulate that one can calibrate an exact model for designing a model-based controller. Actually, such a realistic consideration results in the use of two alternative models, i.e. the nominal model with disturbance and the learnable model with oracle which are approximations of true model.

Taking the above facts into account, in what follows this section, a general model is formulated as the representative of plant's true dynamics.

Let's assume that both control input and state vectors can be represented by convex and compact polytopes $\mathcal{U} \in \mathbb{R}^m$ and $\mathcal{X} \in \mathbb{R}^n$, respectively ($\mathbf{u} \in \mathcal{U}$ and $\mathbf{x} \in \mathcal{X}$). This implies that the constraints on the system dynamics can be represented by a finite number of half-spaces. The compactness and convexity assumptions make the system amenable for numerical optimization which is required for the calculation of optimal control policies. The true system dynamics can be formulated by the composition of a linear term with an un-modeled (usually nonlinear) term, as below:

$$\begin{cases} \mathbf{x}_{t+1} = A\mathbf{x}_t + B\mathbf{u}_t + g(\mathbf{x}_t, \mathbf{u}_t) \\ \mathbf{y}_t = C\mathbf{x}_t \end{cases}, \tag{6.1}$$

where A and B are appropriate $n \times n$ and $n \times m$ matrixes, and function $g : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ determines the un-modeled (usually nonlinear) dynamics. Obviously, $g(\mathbf{x}_t, \mathbf{u}_t)$ is used to

compensate the uncertainty resulting from modeling error. Assume that $g(\mathbf{x}_t, \mathbf{u}_t)$ is bounded and lies within a polytope, say $\mathcal{W}$, for $\forall \mathbf{u}_t \in \mathcal{U}$ and $\forall \mathbf{x}_t \in \mathcal{X}$. Fortunately, there are lots of statistical and computational inference techniques which enable us quantifying $\mathcal{W}$.

Remark 6.2 Other than the modelling error, we can add the effect of measurement noise to Eq. 6.1, and assuming that measurement noise is bounded and is confined within the polytope $\mathcal{D} \in \mathbb{R}^n$. In such a condition (which is the case for the control problem at hand), we get the following state updating rule:

$$\mathbf{x}_{t+1} = A\mathbf{x}_t + B\mathbf{u}_t + g(\mathbf{x}_t, \mathbf{u}_t) + e(\mathbf{x}_t) \ , \tag{6.2}$$

where $e : \mathbb{R}^n \to \mathbb{R}^n$ is either a linear or nonlinear function depending on the measured states of the system. In such a condition, uncertainty is encapsulated by the set $\mathcal{W} \oplus \mathcal{D}$.

Based on the abovementioned reference model, in the next sub-section, the formulation of nominal and learnable state-space models are presented.

6.2.3 Nominal model and learnable model dynamics

It is well-known that, in practice, we cannot formulate the function $g(\mathbf{x}_t, \mathbf{u}_t)$ which exactly captures the un-modeled system dynamics. Thus, an approximate state-space model is used as a surrogate model to Eq. 6.1 which uses its linear part together with a prescribed disturbance term, as below:

$$\bar{\mathbf{x}}_{t+1} = A\bar{\mathbf{x}}_t + B\bar{\mathbf{u}}_t + \mathbf{d}_t \ , \tag{6.3}$$

where $\mathbf{d}_t$ represents the disturbance. Since, it is assumed that $g(\mathbf{x}_t, \mathbf{u}_t)$ is quantified by statistical techniques, the disturbance term is selected in such a way that $\mathbf{d}_t \in \mathcal{W}$. Apparently, in the case that the measurement noise also contaminates the system dynamics, we get $\mathbf{d}_t \in \mathcal{W} \oplus \mathcal{D}$. The simulation of the nominal model with disturbance would be convenient as we just need to sample from $\mathbf{d}_t$. Also, we ensure that the nominal model is a logical approximation of the true model dynamics as the disturbance term is constrained by the same polytope encapsulating $g(\mathbf{x}_t, \mathbf{u}_t)$. By imposing the constraints (based on the quantified polytope) on the nominal states and inputs, we can analyze the feasibility of the states / control signals, and thus, prove the robustness of the system dynamics.

On the other hand, an independent model called learnable model is defined which operates independently, and is responsible for improving the performance and convergence of the LBMPC control law to the ideal control law obtained by using linear MPC with true model (Eq. 6.1) at its heart. As mentioned, this independency not only prones a good rate of robustness and efficiency in tandem (which are usually in confliction with each other), but also makes the robustness of LBMPC independent from possible mis-learning (which is very common in real applications).

The state updating rule of learned model is given as:

$$\tilde{\mathbf{x}}_{t+1} = A\tilde{\mathbf{x}}_t + B\tilde{\mathbf{u}}_t + \mathcal{O}_t(\tilde{\mathbf{x}}_t, \tilde{\mathbf{u}}_t) \ , \tag{6.4}$$

where $\mathcal{O}_t(\tilde{\mathbf{x}}_t, \tilde{\mathbf{u}}_t)$ is a time-varying black-box type identification tool, known as oracle [155].

Through theoretical investigation, it will be proven that the desirable convergence is not related to the complexity of oracle, and thus, one can use black-boxes with complicated structures for system identification. Also, the theoretical results remain valid for time-varying and adaptive oracles. This is a very nice feature as time-varying oracles can be re-trained (using incremental learning [156] tools such as recursive least square, Gaussian filters, and *etc.*) to better understand the system behavior. It is just needed to contrive a sensor in the plant to collect new data, and re-train the oracle.

Remark 6.3 There are a few issues related to feasibility and convergence which should be ascertained when designing oracles for the learnable state-space model. Firstly, an oracle should be selected in such a way that its outputs be bounded and lie in $\mathcal{W}$, i.e. $\mathcal{O}_t(\tilde{\mathbf{x}}_t, \tilde{\mathbf{u}}_t) \in \mathcal{W}$. Secondly, since the learned model is responsible for the efficiency of LBMPC, it is important to make sure the gradients of $\mathcal{O}_t(\tilde{\mathbf{x}}_t, \tilde{\mathbf{u}}_t)$ are computable, as it is required by numerical optimization algorithm used for the calculation of optimal control policies [157].

Remark 6.4 LBMPC is a real-time optimal control algorithm. Therefore, the computational time required for the calculation of control command is of paramount importance. This implies that though we are free to select whatever architecture for $\mathcal{O}_t(\tilde{\mathbf{x}}_t, \tilde{\mathbf{u}}_t)$, cares should be taken to make sure the structure is not too complex, and the estimation and consequent calculation of optimal actuation signal can be conducted in a reasonable time. Satisfying this criterion is beyond the results inducing from theoretical proofs, and often needs a careful empirical investigation and problem-specific experiments.

At this point, the presentation of the architectural formulation of LBMPC is completed. In the coming sub-sections, we go into the details required for the correct implementation of the controller.

6.2.4 Construction of invariant set

For discrete time predictive controllers, a well-known strategy for guaranteeing the robust and safe performance is to determine an invariant set and make sure the terminal state and control input lie in this set. This can be done by adjusting the constraints on terminal states and control inputs to ensure they fall within this invariant set $\Omega \in \mathbb{R}^{n+m}$. Obviously, some conservative conditions and criteria should be satisfied to make sure the invariant set Ω has the required properties to make the systems dynamics stable, safe and robust. Here, the required conditions that any invariant set should possess are presented.

Let's assume that the pair (A, B) is stabilizable, then one can show that the steady-state point and steady input are solutions to the following equation [158]:

$$\begin{bmatrix} A - \mathbb{I}_n & B \\ C & 0_{n\times m} \end{bmatrix} \cdot \begin{bmatrix} \bar{\mathbf{x}}_{\mathbf{stable}} \\ \bar{\mathbf{u}}_{\mathbf{stable}} \end{bmatrix} = \begin{bmatrix} \mathbf{0}_{n\times 1} \\ \mathbf{y}_t \end{bmatrix} \ .$$

It is possible that this system of equations has a unique solution $(\bar{\mathbf{x}}_{\mathbf{stable}}; \bar{\mathbf{u}}_{\mathbf{stable}})$ or multiple solutions which result in sets, i.e. $\bar{\mathbf{x}}_{\mathbf{stable}} \in \mathcal{X}_{\mathbf{stable}}$ and $\bar{\mathbf{u}}_{\mathbf{stable}} \in \mathcal{U}_{\mathbf{stable}}$. Apparently, $\mathcal{X}_{\mathbf{stable}} \subseteq \mathcal{X}$ and $\mathcal{U}_{\mathbf{stable}} \subseteq \mathcal{U}$. By calculating the full column-rank matrixes ϕ and ψ of size $m \times m$ and $n \times m$, respectively, using the null-space computation below:

$$range\Big([\psi^T \ \phi^T]^T\Big) = null\Big([(A - \mathbb{I}_n) \ -B]\Big) \ ,$$

we can parameterize the steady-state points and steady control inputs by $\bar{\mathbf{u}}_{\mathbf{stable}} = \phi\boldsymbol{\theta}$ and $\bar{\mathbf{x}}_{\mathbf{stable}} = \psi\boldsymbol{\theta}$, where $\boldsymbol{\theta} \in \mathbb{R}^m$ and characterizes each solution.

Definition 6.6 [18] A squared matrix is said to be Schur stable if all of its eigenvalues be strictly less than 1.

Definition 6.7 [159] A squared matrix is said to be Hurwitz stable if all of its eigenvalues be strictly negative.

Assume that we determine the $m \times n$ feedback gain matrix K such that the resulting $n \times n$ square matrix $A + BK$ be either Schur stable or Hurwitz stable, then for the closed loop system with the following control law [144]:

$$\bar{\mathbf{u}}_t = K(\bar{\mathbf{x}}_t - \bar{\mathbf{x}}_{\mathbf{stable}}) + \bar{\mathbf{u}}_{\mathbf{stable}} = K\bar{\mathbf{x}}_t + (\phi - K\psi)\boldsymbol{\theta} \ ,$$

the system's dynamics evolves towards stability, mathematically speaking:

$$\exists \ t_f > 0; \forall t \geq t_f \ \ni \ \|\bar{\mathbf{x}}_t - \bar{\mathbf{x}}_{\mathbf{stable}}\|_2 = 0 \ \& \ \|\bar{\mathbf{u}}_t - \bar{\mathbf{u}}_{\mathbf{stable}}\|_2 = 0 \ .$$

The above condition holds when no disturbance affects system's dynamics ($\mathbf{d}_t \equiv \mathbf{0}_{n\times 1}$). However, as mentioned, we assumed that the nominal model is contaminated by noise to be more realistic. The above results are still useful to come up with a reachable invariant set to be used as the terminal constraint. The motivation is that by constructing such an invariant set, one can ensure that the systems trajectory lies in this set even in the presence of bounded disturbance. There are so many methods to come up with such an invariant reachable set Ω. Given the recommendations of [154, 160, 161], the most useful ones are "*maximal output admissible disturbance invariant set*" and "*minimal robust positively invariant set*" ($\Omega \subseteq \mathcal{X} \times \mathbb{R}^m$). Each element of such a set is a vector $(\bar{\mathbf{x}}; \boldsymbol{\theta})$. The reachable invariant set obtained not only satisfies the mentioned condition, but also ensures the feasibility of control inputs and system states. So, an invariant set

has two properties (a) disturbance invariance, and (b) constraint satisfaction. These conditions can respectively be expressed as:

$$\begin{bmatrix} A+BK & B(\phi-K\psi) \\ 0_{m\times p} & \mathbb{I}_m \end{bmatrix} \Omega \ \oplus \{\mathcal{W}\times \mathbf{0}_{m\times 1}\} \subseteq \Omega \ , \tag{6.5}$$

$$\Omega \subseteq \{(\bar{\mathbf{x}};\boldsymbol{\theta}) \mid \bar{\mathbf{x}} \in \mathcal{X};\ \psi\boldsymbol{\theta} \in \mathcal{X};\ K\bar{\mathbf{x}} + (\phi - K\psi)\boldsymbol{\theta} \in \mathcal{U};\ \phi\boldsymbol{\theta} \in \mathcal{U}\} \ . \tag{6.6}$$

In practice, the exact calculation of invariant set is quite demanding (known to be an NP-hard problem), and thus for most of the applications, it is tried to find a good approximation of Ω. As mentioned, there are a lot of approximation techniques for the calculation of reachable invariant sets. Later, the algorithm used for the approximation of Ω for our case study will be scrutinized [162, 163]. Once such a set is identified, it can be used to constraint the terminal state value, in the optimization problem of LBMPC.

6.2.5 Robustness and stability of LBMPC

Obviously, finding a reachable invariant set and using it as terminal state constraint is very crucial for having stable performance. However, there are some other conditions for making a model-based controller robust, which should be considered as well for LBMPC. As an explanation, other than satisfying the reachability to the invariant set, it is also of eminent importance to make sure the real-time trajectory of plant dynamics remains in a safe region even in the presence of disturbance. As mentioned, the suspension control problem can be viewed as a tracking problem, and there is a desired trajectory that we expect LBMPC to follow in order to reduce the vibrations of body mass, and consequently, to make passengers more convenient on road.

Within the literature of MPC, the most well-known concept for complying with the above criteria is the use of tube-based predictive control [160, 161]. Recall the formulation we provided for true model and nominal model with disturbance. If we drop the effect of unmolded dynamics from true model as well as the effect of disturbance from nominal model, then we could strongly say that two models have an equivalent dynamics. In this context, tube-based MPC works based on the fact that if we ensure the trajectory of nominal model without disturbance ($\mathbf{d}_t \equiv \mathbf{0}_{n\times 1}$) has a certain profile, then, the trajectory of true model (including both linear term and un-modeled dynamics) will always be confined within a finite tube encapsulating the trajectory of nominal model. Also, the formulation of tube-based MPC enables us to enlarge the tube by changing the value of feedback gain matrix K.

Just like any other predictive controller, LBMPC renews the control policy after a certain point of time, using the concept of receding horizon control. In this way, let's say after each 1 *sec*, an optimization problem is solved using N future set points in the prediction horizon. So,

there will be N prediction nodes $\mathcal{L} = \{0, 1, 2, ..., N\}$, and at each set-point $i \in \mathcal{L}$, an optimal control input $\tilde{\mathbf{u}}_i$ and state value $\tilde{\mathbf{x}}_i$ should be determined, while satisfying all of the constraints. Apparently, at each set-point, one needs to construct a tube whose width can be calculated by set $\mathcal{R}_i$. The idea is to shrink the set $\mathcal{X}$ at set-point i by the width of $\mathcal{R}_i$. Having said that at the terminal set-point N in which the constraint on state trajectory $\bar{\mathbf{x}}_N$ is imposed by the invariant set $Proj_{\mathcal{X}}(\Omega)$, (apparently for this case, we are interested in $Proj_{\mathcal{X}}(\Omega)$ since $\Omega \subseteq \mathcal{X} \times \mathbb{R}^m$), the tube $\mathcal{R}_N$ is used to shrink $Proj_{\mathcal{X}}(\Omega)$ to make sure the true system's trajectory will also lie in $Proj_{\mathcal{X}}(\Omega)$ in the presence of un-modeled dynamics, provided that $g(\mathbf{x}_{N-1}, \mathbf{u}_{N-1}) \in \mathcal{W}$. Let's say that discrete system is at set-point t, then the general optimization problem can be formulated as:

$$V_t(\mathbf{x}_t) = \min_{\Theta} J_t\big(\boldsymbol{\theta}, \tilde{\mathbf{x}}_t, ..., \tilde{\mathbf{x}}_{t+N}, \hat{\mathbf{u}}_t, ..., \hat{\mathbf{u}}_{t+N-1}\big) \tag{6.7}$$

$$s.t.$$

$$\begin{cases} \tilde{\mathbf{x}}_t = \mathbf{x}_t, \quad \bar{\mathbf{x}}_t = \mathbf{x}_t \\ \tilde{\mathbf{x}}_{t+i+1} = A\tilde{\mathbf{x}}_{t+i} + B\hat{\mathbf{u}}_{t+i} + \mathcal{O}_t(\tilde{\mathbf{x}}_{t+i}, \hat{\mathbf{u}}_{t+i}) \\ \bar{\mathbf{x}}_{t+i+1} = A\bar{\mathbf{x}}_{t+i} + B\hat{\mathbf{u}}_{t+i} \\ \hat{\mathbf{u}}_{t+i} = K\bar{\mathbf{x}}_{t+i} + (\phi - K\psi)\boldsymbol{\theta}_{t+i} \\ \bar{\mathbf{x}}_{t+i+1} \in \mathcal{X} \ominus \mathcal{R}_i, \quad \hat{\mathbf{u}}_{t+i} \in \mathcal{U} \ominus K\mathcal{R}_i \\ (\bar{\mathbf{x}}_{t+N}; \boldsymbol{\theta}_{t+N}) \in \Omega \ominus \Lambda_{final} \end{cases},$$

where $\Lambda_{final} := \{(\mathbf{h}_N; \mathbf{0}_{m\times 1}) \mid \mathbf{h}_N \in \mathcal{R}_N\}$ and all of its elements live in $\mathbb{R}^{n+m}$, $\Theta = (\boldsymbol{\theta}_t, ..., \boldsymbol{\theta}_{t+N-1})$, $i \in \mathcal{L}$, $\mathcal{R}_0$ has only one element which is $\mathbf{0}_{n\times 1}$, $J_t(\cdot)$ is the cost function, and $V_t(\cdot)$ is the value function. The cost function $J_t(\cdot)$ should be non-negative and continuous. Having said that $Proj_{\mathcal{X}}(\Omega)$ is calculated by $\mathcal{R}_\infty$ (it will be discussed later).

Remark 6.5 As seen, the control command for both learned system and nominal system is $\hat{\mathbf{u}}$. This allows the controller to try to improve the performance by $\hat{\mathbf{u}}$ that minimizes $J_t(\cdot)$, and at the same time, to use the same actuation signal $\hat{\mathbf{u}}$ to satisfy the robustness constraints by means of the nominal model. Such an innovative strategy (i.e. formulating an objective function that improves the performance by learned model and at the same time satisfies the constraints by nominal model) automatically handles a traditional conflictive question among control engineers on how to come up with a trade-off between the efficiency and robustness of controlling commands.

Remark 6.6 The original objective function of LBMPC has two sets of decision parameters [18]. However, due to adding constraints to get a robust feedback-based control input (see Eq. 5 of Ref. [159]), the decision variables for our simulation are only $\boldsymbol{\theta}$.

Remark 6.7 Although LBMPC can take time-varying value function and time-varying cost function, for our case study, they considered to be time-invariant. Thus, in the rest of the formulations, we replace the notations $V_t(\cdot)$ and $J_t(\cdot)$ with $V(\cdot)$ and $J(\cdot)$, respectively. The selection of time-invariant cost function $J(\cdot)$ is also essential for proving an important robustness

property of LBMPC, known as robust asymptotic stability. It is also worth noting that oracle $\mathcal{O}_t(\cdot)$ remains time-varying in our formulation due to the reasons mentioned before.

Remark 6.8 The optimization problem defined for LBMPC fixes the initial state value at each stage of optimization rather than taking it as a decision variable. This is inflicted on both nominal and learnable systems. Such an adjustment is based on the own experiments of the author as well as the recommendations given in [160, 164] to retain the continuity of the system, and avoid a shocking spike-type deviation on states' trajectory.

As can be seen, in our formulation, the controlling output is calculated using a feedback-based strategy. It is important to prove that by using the objective function defined in Eq. 6.7, such a control updating rule retains the feasibility while satisfying the robustness constraints.

Definition 6.8 Let's define $\boldsymbol{\mu}_t$ to be a feasible point at time step t (not necessarily optimal) to the defined optimization problem. Also, let's denote the optimal solution by $\boldsymbol{\mu}_t^*$. Then, the sequence of predicted states and control inputs based on the feasible solution can be denoted by $\bar{\mathbf{x}}_{t+i}[\boldsymbol{\mu}_t]$ and $\bar{\mathbf{u}}_{t+i}[\boldsymbol{\mu}_t]$. So, the feedback-based control input updating rule will be $\bar{\mathbf{u}}_t[\boldsymbol{\mu}_t] = K\bar{\mathbf{x}}_t + (\phi - K\psi)\boldsymbol{\mu}_t$.

Here, two theories are presented regarding the robustness of LBMPC. It is tried to give the proofs of these theories in details so that the reader can find out why the results hold. Also, the author presents some assumption and remarks (when necessary) to facilitate the understanding of the proofs.

Assumption 6.1 For now, let's assume that at each set-point $i \in \mathcal{L}$, the set $\mathcal{R}_i$ defining the width of tube will be determined by $\mathcal{R}_i = \bigoplus_{j=0}^{i-1}(A+BK)^j\mathcal{W}$, where $\mathcal{R}_0$ has one element which is $\mathbf{0}_{n\times 1}$. The reason to this will be discussed later in Section 6.3 in more details.

Remark 6.9 Recall Eq. 20 of Ref. [165]. In their proposition, $\mathbf{x}_{t+k|t}$ represents the estimated state of the nominal system at set-point k with no disturbance, and $\mathbf{x}_{t+k}$ denotes the state of the true model obtained by adding the effect of disturbance to the nominal state value. Note that the term $D\boldsymbol{\omega}_t$ in their true system's state-space model, which is a specific format of linear additive disturbance, is set to be the general form $g(\mathbf{x}_t, \mathbf{u}_t)$ for our case study. Now, let's back to the formulation of our objective function. After obtaining a feasible solution $\boldsymbol{\mu}_t$, the states of nominal system without disturbance can be calculated. Thereafter, we jump to set-point $t+1$, an initially set $\bar{\mathbf{x}}_{t+1}[\boldsymbol{\mu}_t]$ which is our estimation from the solution of previous stage. By repeating the optimization which includes simulating the perturbations imposed by disturbances, an updated solution $\boldsymbol{\mu}_{t+1}$ is obtained, and now the state can be predicted based on the new solution $\bar{\mathbf{x}}_{t+1}[\boldsymbol{\mu}_{t+1}]$. The same process is repeated over the optimization horizon when we jump from set-point i to $i+1$. Based on the mentioned facts, we can say that $\bar{\mathbf{x}}_{t+1}[\boldsymbol{\mu}_t]$ is equivalent to $\mathbf{x}_{t+k|t}$ and $\bar{\mathbf{x}}_{t+1}[\boldsymbol{\mu}_{t+1}]$ is equivalent to $\mathbf{x}_{t+k}$ in Eq. 20 of Ref. [165]. Now, given Eq. 20 of Ref. [165], our argument, and *Assumption 6.1*, we get $\bar{\mathbf{x}}_{t+1+i}[\boldsymbol{\mu}_{t+1}] = \bar{\mathbf{x}}_{t+1+i}[\boldsymbol{\mu}_t] + (A+BK)^i g(\mathbf{x}_t, \mathbf{u}_t)$. Note that the same argument holds for control inputs, provided that the feedback gain K be determined.

Theorem 6.1 [18] *If the set Ω be disturbance invariant and satisfies the constraints on control inputs and states, and also $\boldsymbol{\mu}_t$ be a feasible solution, then the control updating rule $\hat{\mathbf{u}}_t[\boldsymbol{\mu}_t] = K\bar{\mathbf{x}}_t + (\phi - K\psi)\boldsymbol{\mu}_t$ with fixed initial state $\bar{\mathbf{x}}_t$ results in robust feasibility, and there will be a feasible point $\boldsymbol{\mu}_{t+1}$ for $\bar{\mathbf{x}}_{t+1}$.*

Proof. [18, 165] Let's take $\mathbf{d}_{t+1+i}[\boldsymbol{\mu}_t] = (A+BK)^i g(\mathbf{x}_t, \hat{\mathbf{u}}_t)$ where $\mathbf{d}_{t+1+i}[\boldsymbol{\mu}_t]$ belongs to the set $\mathcal{D}_i = \{(A+BK)^i g(\mathbf{x}_t, \hat{\mathbf{u}}_t) \mid g(\mathbf{x}_t, \hat{\mathbf{u}}_t) \in \mathcal{W}\}$. First, lets start by investigating the feasibility of control input based on new solution. Given *Remark 6.9*, $\hat{\mathbf{u}}_{t+1+i}[\boldsymbol{\mu}_{t+1}] = \hat{\mathbf{u}}_{t+1+i}[\boldsymbol{\mu}_t] + K\mathbf{d}_{t+1+i}[\boldsymbol{\mu}_t]$ for $i \in \{0, 1, ..., N-2\}$. With respect to *Definition 6.8*, the feasibility of $\boldsymbol{\mu}_t$ satisfies $\hat{\mathbf{u}}_{t+1+i}[\boldsymbol{\mu}_t] \in \mathcal{U} \ominus K\mathcal{R}_{i+1}$ for $i \in \{0, 1, ..., N-2\}$. Based on *Assumption 6.1*, $\mathcal{R}_{i+1} = \mathcal{R}_i \oplus (A+BK)^{i+1}\mathcal{W}$. Now, let optimization algorithm operate and yield a solution, say $\boldsymbol{\mu}_{t+1}$. Based on the equation given for $\hat{\mathbf{u}}_{t+1+i}[\boldsymbol{\mu}_{t+1}]$ and the equivalent set for $\mathcal{R}_{i+1}$, the new solution will be encapsulated by the set $\hat{\mathbf{u}}_{t+1+i}[\boldsymbol{\mu}_{t+1}] \in \mathcal{U} \ominus K\big(\mathcal{R}_i \oplus (A+BK)^{i+1}\mathcal{W}\big) \oplus K(A+BK)^{i+1}\mathcal{W}$. From the properties presented in *Definition 6.4*, we get $\hat{\mathbf{u}}_{t+1+i}[\boldsymbol{\mu}_{t+1}] \in \mathcal{U} \ominus K\mathcal{R}_i$ for $i \in \{0, 1, ..., N-2\}$. This indicates the feasibility of control input predicted by $\boldsymbol{\mu}_{t+1}$. Now let's check the feasibility of states. Based on *Remark 6.9*, $\bar{\mathbf{x}}_{t+1+i}[\boldsymbol{\mu}_{t+1}] = \bar{\mathbf{x}}_{t+1+i}[\boldsymbol{\mu}_t] + \mathbf{d}_{t+1+i}[\boldsymbol{\mu}_t]$ for $i \in \{0, 1, ..., N-1\}$. Based on *Definition 6.8*, the feasibility of $\boldsymbol{\mu}_t$ implies that the constraint $\bar{\mathbf{x}}_{t+1+i}[\boldsymbol{\mu}_t] \in \mathcal{X} \ominus \mathcal{R}_{i+1}$ is satisfied for $i \in \{0, 1, ..., N-1\}$. Again, assume that after performing the optimization, algorithm converges to solution $\boldsymbol{\mu}_{t+1}$. Given *Remark 6.9* and the equivalent set for $\mathcal{R}_{i+1}$, the new solution's state belongs to $\bar{\mathbf{x}}_{t+1+i}[\boldsymbol{\mu}_{t+1}] \in \mathcal{X} \ominus \big(\mathcal{R}_i \oplus (A+BK)^{i+1}\mathcal{W}\big) \oplus (A+BK)^{i+1}\mathcal{W}$. Based on the properties in *Definition 6.4*, we get $\bar{\mathbf{x}}_{t+1+i}[\boldsymbol{\mu}_{t+1}] \in \mathcal{X} \ominus \mathcal{R}_i$ for $i \in \{0, 1, ..., N-2\}$. This proves the feasibility of state based on the new solution $\boldsymbol{\mu}_{t+1}$. What remains is to check the feasibility of control input for the terminal set-point state $\bar{\mathbf{x}}_{t+N}[\boldsymbol{\mu}_{t+1}]$. Based on the arguments given for the feasibility of the state, we get $\bar{\mathbf{x}}_{t+N}[\boldsymbol{\mu}_{t+1}] \in Proj_{\mathcal{X}}(\Omega) \ominus \mathcal{R}_{N-1} \subset Proj_{\mathcal{X}}(\Omega)$. Given the fact that $\boldsymbol{\mu}_t := \boldsymbol{\theta}_{t+N}[\boldsymbol{\mu}_t]$ is nothing but the first solution of the next stage of optimization, i.e. $\boldsymbol{\mu}_{t+1} := \boldsymbol{\theta}_t[\boldsymbol{\mu}_{t+1}]$, we can say $\big(\bar{\mathbf{x}}_{t+N}[\boldsymbol{\mu}_{t+1}]; \boldsymbol{\theta}_t[\boldsymbol{\mu}_{t+1}]\big) \in \Omega \ominus \Lambda_{final} \subset \Omega$ where $\boldsymbol{\theta}_t[\boldsymbol{\mu}_{t+1}] \in \mathbb{R}^m$. On the other hand, due to the constraint $\hat{\mathbf{u}}_{t+N}[\boldsymbol{\mu}_t] = K\bar{\mathbf{x}}_{t+N}[\boldsymbol{\mu}_t] + (\phi - K\psi)\boldsymbol{\mu}_t$, it can be understood that $\hat{\mathbf{u}}_{t+N}[\boldsymbol{\mu}_t]$ leads to $[K \ (\phi - K\psi)]\big(\bar{\mathbf{x}}_{t+N}[\boldsymbol{\mu}_{t+1}]; \boldsymbol{\theta}_t[\boldsymbol{\mu}_{t+1}]\big)$ which lives in $\mathbb{R}^m$. Based on *Definition 6.4* (in particular $T\mathcal{A} = \{T\mathbf{a} \mid \mathbf{a} \in \mathcal{A}\}$ and the property $T(\mathcal{A} \ominus \mathcal{B}) \subseteq T\mathcal{A} \ominus T\mathcal{B}$), and given the fact that $\hat{\mathbf{u}}_{t+N}[\boldsymbol{\mu}_{t+1}]$ results from linear mapping $T\mathbf{a}$, where $T = [K \ (\phi - K\psi)]$ and $\mathbf{a} = \big(\bar{\mathbf{x}}_{t+N}[\boldsymbol{\mu}_{t+1}]; \boldsymbol{\theta}_t[\boldsymbol{\mu}_{t+1}]\big)$, we get $\hat{\mathbf{u}}_{t+N}[\boldsymbol{\mu}_{t+1}] \in T(\Omega \ominus \Lambda_{final}) \subseteq T\Omega \ominus T\Lambda_{final} = T\Omega \ominus K\mathcal{R}_{N-1}$. On the other hand, the constraint satisfaction property of Eq. 6.5 implies $T\Omega \subseteq \mathcal{U}$, and consequently $\hat{\mathbf{u}}_{t+N}[\boldsymbol{\mu}_{t+1}] \in \mathcal{U} \ominus K\mathcal{R}_{N-1}$. At this point, the proof is complete. □

Corollary 6.1 [18] *If the set Ω be disturbance invariant and satisfies the constraints on control inputs and states, and also $\boldsymbol{\mu}_0$ be a feasible solution with fixed initial state $\bar{\mathbf{x}}_0$, then the considered control updating rule results in robust feasibility for all set-points $t \geq 0$.*

Proof. The proof is given by induction, taking into account the results obtaining from *Theorem 6.1*. The proof comprises three steps: (a) firstly, it should be indicated that if $\boldsymbol{\mu}_0$ is a feasible point for the cost function with initial condition $\bar{\mathbf{x}}_0$, then, $\boldsymbol{\mu}_1$ is feasible for $\bar{\mathbf{x}}_1$, (b) it will be assumed that if $\boldsymbol{\mu}_k$ be a feasible point with initial condition $\bar{\mathbf{x}}_k$, then, $\boldsymbol{\mu}_{k+1}$ is feasible for $\bar{\mathbf{x}}_{k+1}$, and (c)

it should be proven that $\boldsymbol{\mu}_{k+2}$ is feasible for $\bar{\mathbf{x}}_{k+2}$, based on the results of assumption (b). Let's start by proving (a). For this case, we are interested in a specific condition where $t = 0$ and $i \in \mathcal{L}$. Obviously, $\mathbf{d}_{1+i}[\boldsymbol{\mu}_0] \in \mathcal{D}_i$, where $\mathcal{D}_i = \{(A+BK)^i g(\mathbf{x}_0, \hat{\mathbf{u}}_0) \mid g(\mathbf{x}_0, \hat{\mathbf{u}}_0) \in \mathcal{W}\}$. With respect to *Remark 6.9*, $\hat{\mathbf{u}}_{1+i}[\boldsymbol{\mu}_1] = \hat{\mathbf{u}}_{1+i}[\boldsymbol{\mu}_0] + K\mathbf{d}_{1+i}[\boldsymbol{\mu}_0]$ for $i \in \{0, 1, ..., N-2\}$. Based on *Theorem 6.1*, we get $\hat{\mathbf{u}}_{1+i}[\boldsymbol{\mu}_1] \in \mathcal{U} \ominus K\big(\mathcal{R}_i \oplus (A+BK)^{i+1}\mathcal{W}\big) \oplus K(A+BK)^{i+1}\mathcal{W}$, which based on the properties given in *Definition 6.4* implies $\hat{\mathbf{u}}_{1+i}[\boldsymbol{\mu}_1] \in \mathcal{U} \ominus K\mathcal{R}_i$ for $i \in \{0, 1, ..., N-2\}$. Based on the same argument, we get $\mathbf{x}_{1+i}[\boldsymbol{\mu}_1] \in \mathcal{X} \ominus \mathcal{R}_i$, $\mathbf{x}_{1+N}[\boldsymbol{\mu}_1] \in Proj_{\mathcal{X}}(\Omega)$, and $\hat{\mathbf{u}}_{1+N}[\boldsymbol{\mu}_1] \in \mathcal{U} \ominus K\mathcal{R}_{N-1}$. Therefore, the control inputs and states based on $\boldsymbol{\mu}_1$ are feasible, and also the final terminal invariant set constraint is satisfied, and thus, (a) is proven. Now useful results from assumptions made in (b) should be inferred to prove (c), and thus, finalize the proof. From assumption (b), it can be inferred that for $t = k+1$, $\boldsymbol{\mu}_{k+1}$ is a feasible point for initial condition $\bar{\mathbf{x}}_{k+1}$, which satisfies the conditions $\hat{\mathbf{u}}_{k+1+i+1}[\boldsymbol{\mu}_{k+1}] \in \mathcal{U} \ominus K\mathcal{R}_{i+1}$ for $i \in \{0, 1, ..., N-2\}$, and $\bar{\mathbf{x}}_{k+1+i+1}[\boldsymbol{\mu}_{k+1}] \in \mathcal{X} \ominus \mathcal{R}_{i+1}$ for $i \in \{0, 1, ..., N-1\}$. In the light of the stated results, the last step of proof, i.e. (c) the feasibility of $\boldsymbol{\mu}_{k+2}$ for initial condition $\bar{\mathbf{x}}_{k+2}$ is taken. Suppose that we let the optimization to operate and we converge to solution $\boldsymbol{\mu}_{k+2}$ from $\boldsymbol{\mu}_{k+1}$. By the same reasoning done in *Theorem 6.1*, as well as the properties presented in *Definition 6.4*, we get $\hat{\mathbf{u}}_{k+2+i+1}[\boldsymbol{\mu}_{k+2}] \in \mathcal{U} \ominus K\mathcal{R}_i$ for $i \in \{0, 1, ..., N-2\}$, $\bar{\mathbf{x}}_{k+2+i+1}[\boldsymbol{\mu}_{k+2}] \in \mathcal{X} \ominus \mathcal{R}_i$ for $i \in \{0, 1, ..., N-1\}$, $\bar{\mathbf{x}}_{k+2+N}[\boldsymbol{\mu}_{k+2}] \in Proj_{\mathcal{X}}(\Omega)$, and $\hat{\mathbf{u}}_{k+2+N}[\boldsymbol{\mu}_{k+2}] \in \mathcal{U} \ominus K\mathcal{R}_{N-1}$. Thus $\boldsymbol{\mu}_{k+2}$ is feasible and satisfies the terminal invariant set constraint. This completes the proof. □

Theorem 6.2 [18] *If the set Ω be disturbance invariant and satisfies the constraints on control inputs and states, and also $\boldsymbol{\mu}_t$ be a feasible solution, then the control updating rule $\hat{\mathbf{u}}_t[\boldsymbol{\mu}_t] = K\bar{\mathbf{x}}_t + (\phi - K\psi)\boldsymbol{\mu}_t$ with fixed initial state $\bar{\mathbf{x}}_t$ results in robust constraint satisfaction, which means true system state $\mathbf{x}_{t+1} \in \mathcal{X}$.*

Proof. [18] The proof is rather easy. Based on the definition of robust feasibility, we only need to write-down the dynamics of the true system as a function of the dynamics of the nominal system without disturbance plus an additive uncertainty, and then show that such a system satisfies the constraints given the feasibility of $\boldsymbol{\mu}_t$. As mentioned in *Remark 6.9*, once we entered the stage $t+1$, the solution $\boldsymbol{\mu}_t$ from the previous stage has not been exposed to disturbance, and thus $\bar{\mathbf{x}}_{t+1}[\boldsymbol{\mu}_t]$ represents the nominal state vector without the effect of disturbance. Based on Eq. 6.1 and Eq. 6.3, we can easily represent the true system state as $\mathbf{x}_{t+1}[\boldsymbol{\mu}_t] = \bar{\mathbf{x}}_{t+1}[\boldsymbol{\mu}_t] + g(\mathbf{x}_t[\boldsymbol{\mu}_t], \mathbf{u}_t[\boldsymbol{\mu}_t])$ where $g(\mathbf{x}_t[\boldsymbol{\mu}_t], \mathbf{u}_t[\boldsymbol{\mu}_t]) \in \mathcal{W}$. Based on *Theorem 1*, the feasibility of $\boldsymbol{\mu}_t$ implies $\bar{\mathbf{x}}_{t+1}[\boldsymbol{\mu}_t] \in \mathcal{X} \ominus \mathcal{W}$. Based on the formulation given for the true state as the function of nominal state, we get $\mathbf{x}_{t+1}[\boldsymbol{\mu}_t] \in (\mathcal{X} \ominus \mathcal{W}) \oplus \mathcal{W}$. Finally, based on the properties given in *Definition 6.4* (in particular $(\mathcal{A} \ominus \mathcal{B}) \oplus \mathcal{B} \subseteq \mathcal{A}$), we get $(\mathcal{X} \ominus \mathcal{W}) \oplus \mathcal{W} \subseteq \mathcal{X}$, and thus, $\mathbf{x}_{t+1}[\boldsymbol{\mu}_t] \in \mathcal{X}$. At this point, the proof is complete. □

Corollary 6.2 [18] *If the set Ω be disturbance invariant and satisfies the constraints on control inputs and states, and also $\boldsymbol{\mu}_0$ be a feasible solution with fixed initial state $\bar{\mathbf{x}}_0$, then the considered control updating rule satisfies the robust constraint satisfaction of the true state, i.e. $\mathbf{x}_t \in \mathcal{X}$, for all set-points $t \geq 0$.*

Proof. Again the proof should be performed by induction. This time, it is needed to (a) show that if $\boldsymbol{\mu}_0$ be a feasible point for the cost function with initial condition $\bar{\mathbf{x}}_0$, then the true state $\mathbf{x}_1 \in \mathcal{X}$. (b) It will be assumed that if $\boldsymbol{\mu}_k$ be a feasible point with initial condition $\bar{\mathbf{x}}_k$, then the true state $\mathbf{x}_{k+1} \in \mathcal{X}$, and finally (c) based on the assumptions made in (b), we get $\mathbf{x}_{k+2} \in \mathcal{X}$. Let's start by proving (a). In this case, we are interested in $t = 0$. Based on the arithmetic we gave for relating the true state value and nominal value, we get $\mathbf{x}_1[\boldsymbol{\mu}_0] = \bar{\mathbf{x}}_1[\boldsymbol{\mu}_0] + g(\mathbf{x}_0[\boldsymbol{\mu}_0], \mathbf{u}_0[\boldsymbol{\mu}_0])$. Then, in the light of *Theorem 6.2*, it can be easily shown that $\mathbf{x}_1 \in \mathcal{X}$. Based on the assumptions made in (b), as well as the results obtained in *Theorem 6.1*, it can be inferred that when $\boldsymbol{\mu}_k$ is a feasible point with initial condition $\bar{\mathbf{x}}_k$, after conducting the optimization at stage $k+1$ and converging to a solution, say $\boldsymbol{\mu}_{k+1}$, it will also be a feasible solution for $\bar{\mathbf{x}}_{k+1}$. Now, given the feasibility of $\boldsymbol{\mu}_{k+1}$ for $\bar{\mathbf{x}}_{k+1}$ as well as the results obtained in *Theorem 6.2*, we can easily prove that $\mathbf{x}_{k+2} \in \mathcal{X}$. This completes the proof. □

One of the other optional design procedures at the heart of LBMPC is the selection of cost function. In what follows this sub-section, the remaining fundamentals theories regarding the proper selection of a continuous cost function $J(\cdot)$ for satisfying other important aspects of the system robustness will be presented. To do so, given the recommendations of [18, 144, 159, 166], the first step would be to prove a lemma regarding the continuity of the value function $V(\cdot)$ defined in the formulation of the objective function.

Based on what presented so far, it is clear that the set of feasible solutions will create a feasible sub-space in the polytope $\mathcal{X}$. Here, we present a new set of feasible states, which will be used later on.

Definition 6.9 Let's denote the feasible sub-state-space by $\mathcal{X}_{\mathbf{feasible}}$. Then, it can be mathematically expressed as $\mathcal{X}_{\mathbf{feasible}} = \{\mathbf{x}_t[\boldsymbol{\mu}_t] \mid \forall \boldsymbol{\mu}_t\}$. Apparently, $\mathcal{X}_{\mathbf{feasible}} \subseteq \mathcal{X}$.

Also, three classical optimal control theories are required for the proof of the main result in the coming lemma. Consider the general constrained optimal control problem below:

$$\underset{\sim}{V}(\mathbf{x}) = \min_{\mathbf{u}} \underset{\sim}{J}(\mathbf{x}, \mathbf{u}) \ , \tag{6.8}$$

$$\mathbf{u}^*(\mathbf{x}) = \arg\min \underset{\sim}{J}(\mathbf{x}, \mathbf{u})$$

$$s.t. \quad \mathbf{u} \in \Upsilon(\mathbf{x})$$

where the cost function $\underset{\sim}{J} : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$ is continuous, $\underset{\sim}{V} : \mathbb{R}^n \to \mathbb{R}$ is the value function, and $\Upsilon : \mathbb{R}^n \to \mathbb{R}^m$ is possibly a multi-valued (set-valued) function, as the optimal solution to the control problem may not be unique.

Theorem 6.3 [144] *Suppose that the cost function $\underset{\sim}{J} : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$ be continuous. Also, suppose that the polytope $\mathcal{U}$ be compact and the function $\Upsilon : \mathbb{R}^n \to \mathbb{R}^m$ be continuous for all $\mathbf{x} \in \mathcal{X}$. Then, $\underset{\sim}{V}(\cdot)$ is continuous, and $\mathbf{u}^*(\mathbf{x})$ is outer semi-continuous.*

Proof. See the proof of *Theorem C.28* of [144]. □

Definition 6.10 Let's define a polytope $\mathcal{Z}$ which lives in $\mathbb{R}^n \times \mathcal{U}$, and just like before, $\mathcal{U}$ is a polytope in $\mathbb{R}^m$. Let $\mathbf{z} = (\mathbf{x}; \mathbf{u}) \in \mathcal{Z}$, then $\mathcal{X} = Proj_{\mathbf{x}}(\mathcal{Z})$.

Now, based on *Definition 6.10*, Eq. 6.8 can be reformulated as:

$$\underset{\sim}{V}(\mathbf{x}) = \min_{\mathbf{u}} \underset{\sim}{J}(\mathbf{x}, \mathbf{u}) \ ,$$

$$\mathbf{u}^*(\mathbf{x}) = \arg\min \underset{\sim}{J}(\mathbf{x}, \mathbf{u})$$

$$s.t. \quad \mathbf{z} \in \mathcal{Z}$$

Also, the multiple-valued function Υ can be defined as $\Upsilon(\mathbf{x}) := \{\mathbf{u} \in \mathbb{R}^m \mid \mathbf{z} \in \mathcal{Z}\}$.

Theorem 6.4 [144] *Suppose $\mathcal{Z}$ is as defined in Definition 6.10, then the multiple-valued function Υ as defined above is continuous in $\mathcal{X}$.*

Proof. See the proof of *Theorem C.33* of [144]. □

Theorem 6.5 [144] *Suppose $\underset{\sim}{J} : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$ and $\mathcal{Z}$ be as defined in Definition 6.10, then, $\underset{\sim}{V}(\cdot)$ is continuous, and $\mathbf{u}^*(\mathbf{x})$ is outer semi-continuous. Also, if $\mathbf{u}^*(\mathbf{x})$ be the unique optimal solution at each $\mathbf{x} \in \mathcal{X}$, then $\mathbf{u}^* : \mathbb{R}^n \to \mathbb{R}^m$ is continuous in $\mathcal{X}$.*

Proof. See the proof of *Theorem C.34* of [144]. □

Now, we can prove the continuity of the value-function defined in LBMPC in the light of the above theories.

Lemma 6.1 [18] *If $\mathcal{X}_{\textbf{feasible}}$ be the set defined in Definition 6.9, and also $J(\cdot)$ and $\mathcal{O}_t(\cdot)$ be continuous functions, then $V(\cdot)$ is continuous on $int(\mathcal{X}_{\textbf{feasible}})$.*

Proof. [18] Based on the above arguments, we should first reformulate an equivalent optimization problem for LBMPC which is consonant with the preliminaries for using the above theories. In particular, Eq. 6.7 can be reformulated as:

$$V(\mathbf{x}_t) = \min_{\Theta} \underset{\sim}{J}(\boldsymbol{\theta}, \tilde{\mathbf{x}}_t, ..., \tilde{\mathbf{x}}_{t+N}) \ , \tag{6.9}$$

$$\boldsymbol{\theta}^*(\mathbf{x}_t) = \arg\min \underset{\sim}{J}(\boldsymbol{\theta}, \tilde{\mathbf{x}}_t, ..., \tilde{\mathbf{x}}_{t+N})$$

$$s.t. \quad \boldsymbol{\theta} \in \Upsilon(\mathbf{x})$$

where $\underset{\sim}{J} : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$ is the the cost function, $\underset{\sim}{V} : \mathbb{R}^n \to \mathbb{R}$ is the value function, and $\Upsilon : \mathbb{R}^n \to \mathbb{R}^m$ is a multi-valued (set-valued) function. If we can show that $\underset{\sim}{J}$ and Υ are continuous, then according to *Theorem 6.3*, $V(\cdot)$ is continuous and $\mathbf{u}^*(\mathbf{x})$ is outer semi-continuous. The continuity of Υ directly results from *Theorem 6.4*, as all of the constraints of LBMPC in Eq. 6.7

are linear. Now, since, due to the assumption, the original cost function $J(\cdot)$, and the oracle $\mathcal{O}_t(\cdot)$ are continuous, then $V(\cdot)$ which is the composition of these continuous functions will be continuous according to [167]. Finally, by the convex formulation of the optimization problem, we can ensure the solution $\boldsymbol{\theta}^*(\mathbf{x}_t)$ is unique, and according to *Theorem 6.5*, we can manipulate the controller to make sure $\boldsymbol{\theta}^* : \mathbb{R}^n \to \mathbb{R}^m$ is continuous. This completes the proof. □

Remark 6.10 It is worth noting that satisfying the continuity of $\boldsymbol{\theta}^*(\mathbf{x}_t)$ is not of our primary interest. Rather, we just intended to mention that it is possible to have a continuous optimal solution as well. The main goal of *Lemma 6.1* was to prove the continuity of the value function. This allows us to even formulate a multi-modal non-convex optimization problem (neglecting the uniqueness of $\boldsymbol{\theta}^*$, and as a result the continuity condition for $\boldsymbol{\theta}$), and prove further robustness results, in the light of the continuity of the value function $V(\cdot)$. So, we are allowed to converge to sub-optimal solutions by means of a linear optimization problem, even when a multi-modal nonlinear non-convex objective function is formulated, and at the same time satisfy the continuity of $V(\cdot)$, and prove further robustness results. This is obviously very advantageous with respect to computational cost, especially when LBMPC is applied to real-time control of plants with crazy nonlinear dynamics.

Now, we are able to prove one further result concerning the robust asymptotic stability (RAS) of LBMPC, which is another criterion to verify the robustness of controlling commands [166]. RAS states that if the controlling command for the nominal model without disturbance converges to $\bar{\mathbf{x}}_{\mathbf{stable}}$, then the same controlling command can steer the true system within a bounded distance from $\bar{\mathbf{x}}_{\mathbf{stable}}$.

Definition 6.11 A controller is said to be RAS in the vicinity of $\bar{\mathbf{x}}_{\mathbf{stable}}$, if a *type-*$\mathcal{KL}$ function $\beta(\cdot,\cdot)$ can be found such that for all $\mathbf{d}_t$ satisfying $\max_t \|\mathbf{d}_t\|_2 \leq \delta$, where $\delta > 0$, we have $\mathbf{x}_t \in \mathcal{X}$ and there exists $\epsilon > 0$ such that $\|\mathbf{x}_t - \bar{\mathbf{x}}_{\mathbf{stable}}\|_2 \leq \beta(\|\mathbf{x}_0 - \bar{\mathbf{x}}_{\mathbf{stable}}\|_2, t) + \epsilon$, for all $\forall t \geq 0$.

As mentioned, the goal is to prove the RAS for LBMPC. Based on the objective of RAS and what we mentioned above, to prove the RAS of LBMPC, two steps should be taken. Firstly, it should be indicated that LBMPC without disturbance ($\mathcal{O}_t \equiv \mathbf{d}_t \equiv \mathbf{0}_{n\times 1}$) is convergent. Thereafter, it should be proven that the trained oracle $\mathcal{O}_t$ is bounded.

Remark 6.11 LBMPC with $\mathcal{O}_t \equiv \mathbf{d}_t \equiv \mathbf{0}_{n\times 1}$ and $\bar{\mathbf{x}} = \tilde{\mathbf{x}}$ is equivalent to standard linear MPC knowing the true model with no disturbance.

Before proceeding with the statement of the theorem, one definition and one proposition should be presented.

Definition 6.12 A function $f : \mathcal{X} \to \mathcal{Y}$ is said to be continuous in $\mathbf{c} \in int\{\mathcal{X}\}$, if there exists $\exists \epsilon > 0$ and $\exists \boldsymbol{\delta} \in int\{\mathcal{X}\} \ni \|\boldsymbol{\delta}\|_2 > 0$ such that for all elements $\mathbf{x}$ in its domain ($\mathbf{x} \in \mathcal{X}$), satisfying $\|\mathbf{x} - \mathbf{c}\|_2 < \|\boldsymbol{\delta} - \mathbf{c}\|_2$, the inequality $\|f(\mathbf{x}) - f(\mathbf{c})\|_2 < \epsilon$ holds.

Proposition 6.1 [18] *If there exists a Lyapunov function $W(\cdot)$ which is continuous on $int(\mathcal{X}_{\mathbf{feasible}})$, then the controlled system steered by feedback-based control command is RAS on $int(\mathcal{X}_{\mathbf{feasible}})$.*

Proof. See the proof of *Proposition 8* of [166]. □

Theorem 6.6 [18] *Suppose that (a) the set* Ω *be disturbance invariant and satisfies the constraints on control inputs and states, (b)* $\boldsymbol{\mu}_0$ *be a feasible solution with fixed initial state* $\mathbf{x}_0$*, (c) the cost function* $J(\cdot)$ *is time-invariant, strictly-convex, and continuous, and finally (d) we can find a Lyapunov function* $W(\cdot)$ *for nominal system without disturbance when calculating the control commands using linear MPC. Then, under the condition that* $\mathcal{O}_t$ *be continuous and satisfies* $\max_{t,\mathcal{X},\mathcal{U}} \|\mathcal{O}_t\|_2 \leq \delta$*, it can be assured that the control law of LBMPC with bounded disturbance* $\mathbf{d}_t$ *is RAS.*

Proof. [18] Let's show the optimal solution to linear MPC and LBMPC by $\widehat{\boldsymbol{\mu}}_t^*$ and $\boldsymbol{\mu}_t^*$, respectively. It follows that the optimal solution is unique since it is assumed that $J(\cdot)$ is strictly-convex. Also, recall that the initial state is set to be fixed and is the same for both of the systems. Based on these assumptions, the linear MPC and LBMPC models trajectory obtained by optimal solution can be formulated as:

$$\begin{cases} \bar{\mathbf{x}}_{t+1}[\widehat{\boldsymbol{\mu}}_t^*] = A\bar{\mathbf{x}}_t[\widehat{\boldsymbol{\mu}}_t^*] + B\hat{\mathbf{u}}_t[\widehat{\boldsymbol{\mu}}_t^*] + \mathbf{d}_t \\ \bar{\mathbf{x}}_{t+1}[\boldsymbol{\mu}_t^*] = A\bar{\mathbf{x}}_t[\boldsymbol{\mu}_t^*] + B\hat{\mathbf{u}}_t[\boldsymbol{\mu}_t^*] \end{cases} .$$

Now, let's define ϱ_t to be the state-dependent disturbance, as below:

$$\varrho_t := B\big(\hat{\mathbf{u}}_t[\boldsymbol{\mu}_t^*] - \hat{\mathbf{u}}_t[\widehat{\boldsymbol{\mu}}_t^*]\big) + \mathbf{d}_t \ .$$

By construction, we get:

$$\bar{\mathbf{x}}_{t+1}[\boldsymbol{\mu}_t^*] = \bar{\mathbf{x}}_{t+1}[\widehat{\boldsymbol{\mu}}_t^*] + \varrho_t \ .$$

Based on the results of *Corollary 6.1*, *Corollary 6.2*, and *Proposition 6.1*, as well as *Definition 6.11*, it holds that given $\epsilon > 0$, a *type-*$\mathcal{KL}$ function $\beta(\cdot,\cdot)$ can be found such that for all ϱ_t satisfying $\max_t \|\varrho_t\|_2 \leq \delta'$, we have $\mathbf{x}_t \in \mathcal{X}$ and $\|\mathbf{x}_t - \bar{\mathbf{x}}_{\mathbf{stable}}\|_2 \leq \beta(\|\mathbf{x}_0 - \bar{\mathbf{x}}_{\mathbf{stable}}\|_2, t) + \epsilon$, for all $\forall t \geq 0$. To complete the proof, it is needed to indicate that such a δ' exists. Now, based on *Theorem 6.5*, *Lemma 6.1*, and the continuity and the boundedness of $\mathcal{O}_t$ (as assumed), we get $\boldsymbol{\mu}_t^*$ to be continuous.

As mentioned, the first step was to find the solution of linear MPC $\widehat{\boldsymbol{\mu}}_t^*$ under $\mathcal{O}_t \equiv \mathbf{0}_{n\times 1}$ (the solution obtained by enforcing $\mathcal{O}_t(\bar{\mathbf{x}}_t[\widehat{\boldsymbol{\mu}}_t^*], \hat{\mathbf{u}}_t[\widehat{\boldsymbol{\mu}}_t^*]) = \mathbf{0}_{n\times 1}$). Now, because of the continuity of control input, and keeping in mind that (due to the assumed boundedness of oracle) the solution of LBMPC is at most a bounded perturbation from that of linear MPC, i.e. $\|\boldsymbol{\mu}_t^* - \widehat{\boldsymbol{\mu}}_t^*\|_2 < \delta''$, and also because of *Definition 6.12*, it holds that $\|\hat{\mathbf{u}}_t[\boldsymbol{\mu}_t^*] - \hat{\mathbf{u}}_t[\widehat{\boldsymbol{\mu}}_t^*]\|_2 < \epsilon'$. For the ease of representation, let's take $\epsilon' = \delta'''/2\|B\|_{\mathbf{Frobenius}}$. Now, because of the assumed continuity of oracle as a function of control input, and again given *Definition 6.12*, we get:

$$\|\mathcal{O}_t(\bar{\mathbf{x}}_t[\boldsymbol{\mu}_t^*], \hat{\mathbf{u}}_t[\boldsymbol{\mu}_t^*]) - \mathcal{O}_t(\bar{\mathbf{x}}_t[\widehat{\boldsymbol{\mu}}_t^*], \hat{\mathbf{u}}_t[\widehat{\boldsymbol{\mu}}_t^*])\|_2 = \|\mathcal{O}_t(\bar{\mathbf{x}}_t[\boldsymbol{\mu}_t^*], \hat{\mathbf{u}}_t[\boldsymbol{\mu}_t^*])\|_2 < \epsilon'' = \delta \ .$$

Now, based on the construction of ϱ_t, and the boundedness of the term $B\big(\hat{\mathbf{u}}_t[\boldsymbol{\mu}_t^*] - \hat{\mathbf{u}}_t[\widehat{\boldsymbol{\mu}}_t^*]\big)$, it holds that ϱ_t belongs to a bounded polytope, and therefore, δ' exists. So, the proof will be completed by using $\{\mathcal{O}_t \mid \|\mathcal{O}_t\|_2 < \delta\}$ and taking $\delta = \min\{\delta'''/2\ ,\ \delta'\}$. □

Now, the only thing that remains is to select a cost function $J(\cdot)$ which results in Lyapunov function $W(\cdot)$ as required in the statement of proof for *Theorem 6.6*). One of the most applicable cost functions within the realm of optimal control that satisfies this condition is a quadratic cost function with linear model and bounded uncertainty. Therefore, based on recommendations given in seminal books [144, 168] as well as the arguments of [18, 159], the following cost function is selected:

$$J = \|\tilde{\mathbf{x}}_{t+N} - \psi\boldsymbol{\theta}\|_P^2 + \|\bar{\mathbf{x}}_{\mathbf{stable}} - \psi\boldsymbol{\theta}\|_T^2 + \sum_{i=1}^{N-1} \left(\|\tilde{\mathbf{x}}_{t+i} - \psi\boldsymbol{\theta}\|_Q^2 + \|\hat{\mathbf{u}}_{t+i} - \phi\boldsymbol{\theta}\|_R^2\right)\ , \tag{6.10}$$

where P, T, Q, and R are positive definite matrixes, used for tracking $\bar{\mathbf{x}}_{\mathbf{stable}} \in \mathcal{X}_{\mathbf{stable}}$. It is worth noting that T, Q, and R are selected based on the designer preference, and P (known as Lyaponov matrix) is determined by solving the following equation:

$$(A + BK)^T P(A + BK) - P = -(Q + K^T RK)\ . \tag{6.11}$$

By means of the results from converse Lyapunov theory [152, 169], it will proven that selecting the above cost function can afford a Lyapunov function $W(\cdot)$ for the steady-state point $\bar{\mathbf{x}}_{\mathbf{stable}}$ of the nominal model without disturbances. Before stating the proof, a lemma is presented which can be used for the proof.

Lemma 6.2 [159] *For linear MPC defined based on Remark 6.11, if K be a desired control gain, and also the Lyapunov matrix P be determined by Eq. 6.11, and $\bar{\mathbf{x}}_{\mathbf{stable}} \in \mathcal{X}_{\mathbf{stable}}$ kept fixed, if the optimal solution satisfies $\|\mathbf{x} - \psi\boldsymbol{\theta}^*\|_Q^2 = 0$, then $\|\bar{\mathbf{x}}_{\mathbf{stable}} - \psi\boldsymbol{\theta}\|_T^2 = 0$.*

Proof. See the proof of *Lemma 3* of [159]. □

Remark 6.12 The same results hold when Q and T are both identity matrix, and thus, $\|\cdot\|_Q$ and $\|\cdot\|_T$ can be replaced with $\|\cdot\|_2$.

Theorem 6.7 [159] *For the cost function defined in Eq. 6.10, provided that $A + BK$ be either Schur stable or Hurwitz stable, and $\bar{\mathbf{x}}_{\mathbf{stable}} \in \mathcal{X}_{\mathbf{stable}}$ kept fixed, and also the Lyapunov matrix P be determined by Eq. 6.11, there always exits a Lyapunov function for $\bar{\mathbf{x}}_{\mathbf{stable}}$ of nominal system without disturbance when calculating the control commands using linear MPC.*

Proof. [159] Due to *Remark 6.11*, for linear MPC, we have $\bar{\mathbf{x}} = \tilde{\mathbf{x}}$. Also, according to the results of converse Lyapunov theory [169] as well as *Remark 6.12*, for the cost function Eq. 6.10, such a Lyapunov function exists if: (a) it asymptotically holds that $\lim_{t\to\infty} \|\bar{\mathbf{x}}_t - \bar{\mathbf{x}}_{\mathbf{stable}}\|_2 = 0$, for all $\mathbf{x}_0 \in \mathcal{X}_{\mathbf{feasible}}$, and (b) for every $\epsilon > 0$, one can find $\delta > 0$ such that $\|\bar{\mathbf{x}}_0 - \bar{\mathbf{x}}_{\mathbf{stable}}\|_2 < \delta$ implies that $\|\bar{\mathbf{x}}_t - \bar{\mathbf{x}}_{\mathbf{stable}}\|_2 < \delta$, for all $t \geq 0$. First, we start by proving (a). Let's denote the optimal

solution of linear MPC at time t by $\widehat{\boldsymbol{\mu}}_t^*$, and consequently the predicted states and control inputs by $\bar{\mathbf{x}}_t[\widehat{\boldsymbol{\mu}}_t^*]$ and $\hat{\mathbf{u}}_t[\widehat{\boldsymbol{\mu}}_t^*]$. Let's denote the optimal value function at set-point t by $V_t^*(\bar{\mathbf{x}}_t[\widehat{\boldsymbol{\mu}}_t^*])$, and the value function for any other solution by $V_t(\bar{\mathbf{x}}_t[\widehat{\boldsymbol{\mu}}_t])$. Given the feasibility results obtained in *Corollary 6.2*, it comes that we can find the optimal feasible solution $\widehat{\boldsymbol{\mu}}_{t+1}^*$ in the next step. Due to the properties of Lyapunov function presented in *Definition 6.3*, and also the cost function J, it holds that:

$$V^*(\bar{\mathbf{x}}_{t+1}[\widehat{\boldsymbol{\mu}}_{t+1}^*]) \leq V(\bar{\mathbf{x}}_{t+1}[\widehat{\boldsymbol{\mu}}_t]) \leq V^*(\bar{\mathbf{x}}_t[\widehat{\boldsymbol{\mu}}_t^*]) - \|\bar{\mathbf{x}}_t - \bar{\mathbf{x}}_{\mathbf{stable}}\|_Q^2 \ . \tag{6.12}$$

Due to the properties of Lyapunov function ($\lim_{t\to\infty} V^*(\bar{\mathbf{x}}_t[\widehat{\boldsymbol{\mu}}_t^*]) = V(\bar{\mathbf{x}}_{\mathbf{stable}}) = 0$), and the non-negativity of J, it can be concluded that $\lim_{t\to\infty} \|\bar{\mathbf{x}}_t - \bar{\mathbf{x}}_{\mathbf{stable}}\|_Q = 0$. Based on *Remark 6.12*, we get $\lim_{t\to\infty} \|\bar{\mathbf{x}}_t - \bar{\mathbf{x}}_{\mathbf{stable}}\|_2 = 0$. This finishes the proof of (a). Now let's start proving part (b). Since Q and T are positive definite matrixes, we have:

$$\exists S \succ 0 \ \ni \quad Q - S \succ 0 \ \& \ T - S \succ 0 \ .$$

Based on Eq. 6.10, Eq. 6.12, and also the triangular inequality, we get:

$$\begin{aligned}
\|\tilde{\mathbf{x}}_t - \bar{\mathbf{x}}_{\mathbf{stable}}\|_S^2 &= \|\tilde{\mathbf{x}}_t - \psi\boldsymbol{\theta} + \psi\boldsymbol{\theta} - \bar{\mathbf{x}}_{\mathbf{stable}}\|_S^2 \\
&\leq \|\tilde{\mathbf{x}}_t - \psi\boldsymbol{\theta}\|_S^2 + \|\psi\boldsymbol{\theta} - \bar{\mathbf{x}}_{\mathbf{stable}}\|_S^2 \\
&\leq \|\tilde{\mathbf{x}}_t - \psi\boldsymbol{\theta}\|_Q^2 + \|\psi\boldsymbol{\theta} - \bar{\mathbf{x}}_{\mathbf{stable}}\|_T^2 \leq J
\end{aligned}$$

If we perform the optimization on the both sides of the inequality above, we get:

$$\min_{\Theta} \|\tilde{\mathbf{x}}_t - \bar{\mathbf{x}}_{\mathbf{stable}}\|_S^2 \leq \min_{\Theta} J =: V(\mathbf{x}_t)$$

$$s.t.$$

$$\begin{cases}
\tilde{\mathbf{x}}_t = \mathbf{x}_t, \quad \bar{\mathbf{x}}_t = \mathbf{x}_t \\
\tilde{\mathbf{x}}_{t+i+1} = A\tilde{\mathbf{x}}_{t+i} + B\hat{\mathbf{u}}_{t+i} + \mathcal{O}_t(\tilde{\mathbf{x}}_{t+i}, \hat{\mathbf{u}}_{t+i}) \\
\bar{\mathbf{x}}_{t+i+1} = A\bar{\mathbf{x}}_{t+i} + B\hat{\mathbf{u}}_{t+i} \\
\hat{\mathbf{u}}_{t+i} = K\bar{\mathbf{x}}_{t+i} + (\phi - K\psi)\boldsymbol{\theta}_{t+i} \\
\bar{\mathbf{x}}_{t+i+1} \in \mathcal{X} \ominus \mathcal{R}_i, \quad \hat{\mathbf{u}}_{t+i} \in \mathcal{U} \ominus K\mathcal{R}_i \\
(\bar{\mathbf{x}}_{t+N}; \boldsymbol{\theta}_{t+N}) \in \Omega \ominus \Lambda_{final}
\end{cases} .$$

Also, based on *Lemma 6.1*, *Definition 6.12*, and the results of part (a) of the proof (in particular $V(\bar{\mathbf{x}}_{\mathbf{stable}})$), it holds that for every $\epsilon > 0$, one can find $\delta > 0$ such that $V(\bar{\mathbf{x}}_0) < \epsilon$ implies $\|\bar{\mathbf{x}}_0 - \bar{\mathbf{x}}_{\mathbf{stable}}\|_2 < \delta$. Based on the non-negativity of J, and the properties of Lyapunov function,

for all $t \geq 0$, we get $\|\bar{\mathbf{x}}_t - \bar{\mathbf{x}}_{\mathbf{stable}}\|_2 \leq V(\bar{\mathbf{x}}_t) \leq V(\bar{\mathbf{x}}_0) < \epsilon$, which completes the proof of part (b). At this point, the proof is complete. □

By finalizing the proof, we proved that for the cost function given in Eq. 6.10, there exists a Lyapunov function which is required for proving RAS of LBMPC in *Theorem 6.6*.

In this sub-section, the fundamental theories regarding the feasibility, stability and robustness of controlling command at the heart of LBMPC were proven. Now, we can investigate the properties of oracle to come up with fundamental theories to justify the use of learning at the heart of LBMPC.

6.2.6 Adaption and learnability of LBMPC via oracles

In this sub-section, some general description regarding the properties of oracles used at the heart of learnable model is presented. Also, some general fundamental theoretical properties regarding the convergence of oracles whose range are bounded and are continuous are given. As precisely explained in the previous sub-section, it is important that a selected oracle be continuous and bounded to make sure LBMPC is robust and stable, even under mis-learning.

More detailed and specific information on how to select oracles and how to theoretically prove their convergence are given later in this chapter. Because all of the oracles used in this study have statistical foundation, as usual, it should be ensured that the general explanation and properties presented here are valid for both parametric and non-parametric oracles. As mentioned before, oracles are either interpolator or extrapolator black-boxes which use a number of input and output pairs for estimation and forecasting [155]. As stated in the original paper [18], two main questions should be answered regarding the application of learning systems (in particular oracles) to LBMPC design. Firstly, that would be interesting to find out which type of oracles can result the best performance and are best suited to be used at the heart of LBMPC. Secondly, it is very important to choose oracles such that the controlling commands computed by LBMPC with learnable model can converge to that of MPC (linear or nonlinear) that uses the true model.

As mentioned before, the beauty of LBMPC is that all of the nonlinearities of the system can be captured by the oracle which itself appears as an uncertain additive element to a linear model. So, the computation of optimal controlling command of LBMPC is not as computationally extensive as nonlinear MPC (NMPC), and at the same time, it can be ensured that the controlling command is computed without neglecting important nonlinear features of plant.

Let's give the standard definitions and notations for both parametric and non-parametric oracles, and then, proceed with fundamental theories required for selecting an oracle for LBMPC.

Definition 6.13 An oracle is called parametric if it has a well-defined fixed structure, say $\mathcal{O}_t(\tilde{\mathbf{x}}_t, \tilde{\mathbf{u}}_t) = \wp_t(\tilde{\mathbf{x}}_t, \tilde{\mathbf{u}}_t \mid \boldsymbol{\lambda}_t)$, where $\boldsymbol{\lambda}_t = (\lambda_1, \lambda_2, ..., \lambda_p) \in \mathcal{Q}$ includes the parameters of the model, and $\mathcal{Q}$ is a set encapsulating the possible values of coefficients, and lives in $\mathbb{R}^p$ [170].

The parametric model $\wp_t(\tilde{\mathbf{x}}_t, \tilde{\mathbf{u}}_t \mid \boldsymbol{\lambda}_t)$ can take both linear and nonlinear structures, and tries to precisely capture the behavior of un-modeled dynamic $g(\mathbf{x}_t, \mathbf{u}_t)$ in true model. Usually, supervised training techniques with a bunch of labeled data are used to find the optimal values of the parameters, which can be mathematically expressed as:

$$\boldsymbol{\lambda}^* = \sum_{j=0}^{t} \left(\tilde{\mathbf{x}}_{t+1} - (A\tilde{\mathbf{x}}_t + B\tilde{\mathbf{u}}_t) - \wp_t(\tilde{\mathbf{x}}_t, \tilde{\mathbf{u}}_t \mid \boldsymbol{\lambda}_t)\right)^2 .$$

Such type of models have been vigorously used over the decades by control engineers for designing efficient model-based controllers, especially for adaptive control applications [171]. The salient asset of such models refers to their simplicity and easy-tuning; however, they usually have a high-biased, and also cannot be considered as robust models.

Note that, here, we only present the general definition and formulation of parametric oracles and the process of training them. The detailed information on statistical learning for selecting parametric oracles, and remarks on how to use them will be given later.

Definition 6.14 An oracle $\mathcal{O}_t(\tilde{\mathbf{x}}_t, \tilde{\mathbf{u}}_t)$ is called non-parametric if it does not assume any a priori fixed structure for capturing the behavior of $g(\mathbf{x}_t, \mathbf{u}_t)$ in true model. Also, it should be mentioned that the term non-parametric does not imply that such models do not have any parameter. Rather, it means that the number and nature of parameters are flexible and depend on training data, instead of being fixed beforehand.

Fortunately, in recent decades, and extensive research has been conducted by statisticians and the researchers of computational intelligence society to develop efficient non-parametric models for classification and regression [172]. Non-parametric models can often take complicated black-box type structures which are unknown. Thanks to the stability and robustness proofs we made previously, that would be possible to use complex non-parametric tools such as artificial neural network, deep learning machines, and *etc.*, provided that the resulting oracle be bounded and continuous. Non-parametric models are very useful tools, especially because they can be trained in such a way that their performance be robust, and due to this advantage, their combination with controllers has been becoming very popular [18, 173]. Cares should be taken when designing non-parametric models for real-time application, as they may have complicated structure and take a long time for inference. Further information on selecting non-parametric oracles for LBMPC design and their challenges and advantages will be given later.

Remark 6.13 As real-time fast computation and robustness are of highest importance when using LBMPC, it should be ensured that the chosen parametric or non-parametric oracle is bounded and its differential can be calculated easily for the calculation of optimal control command.

Until now, some preliminary explanation regarding the first general concern, i.e. finding that which types of oracles are prone to be used at the heart of LBMPC, has been given. In the next sub-section, the other issue is taken into account.

6.2.7 Convergence properties of oracles

In this sub-section, it is mainly tried to provide a reasonable answer to the second question regarding the use of oracle, i.e. ensuring that the chosen oracle is such that the controlling commands computed by LBMPC with learnable model can converge to that of a MPC with true model.

It should be mentioned that for both LBMPC and MPC, the optimal control policy is calculated based on a sequence of stages of optimization problems. Therefore, by convergence, the intention is to ensure the convergence of the minimizers of a sequence of optimization problems to the minimizer of a limiting optimization problem of the same nature. This will happen if the time-varying oracle $\mathcal{O}_t(\tilde{\mathbf{x}}_t, \tilde{\mathbf{u}}_t)$ stochastically converges to true un-modeled dynamics $g(\mathbf{x}_t, \mathbf{u}_t)$. As the oracle $\mathcal{O}_t(\tilde{\mathbf{x}}_t, \tilde{\mathbf{u}}_t)$ is time-varying, and at each stage of optimization, the time-varying oracle can take different forms, the formulated optimization problem can vary at each stage. Note that by *Remark 6.7*, we fixed the formulation of functional $J(\cdot)$; however, its output still can vary due to the variation of $\mathcal{O}_t(\tilde{\mathbf{x}}_t, \tilde{\mathbf{u}}_t)$ over time. Under such a condition, the only possible option is to investigate the point-wise convergence of $\mathcal{O}_t(\tilde{\mathbf{x}}_t, \tilde{\mathbf{u}}_t)$ to $g(\mathbf{x}_t, \mathbf{u}_t)$, which is insufficient for proving the convergence of the minimizer of a sequence of optimization problems to a limiting optimization problem [174].

Thus, a modified convergence theorem, known as epi-convergence, is applicable for proving the convergence of the control laws of LBMPC and linear MPC with true model.

Definition 6.15 Let's define $\mathbf{Epi}f_t(\cdot, \mathbf{u})$, the epi-graph of a given function $f_t(\cdot, \mathbf{u})$ as the set of all points lying on or above the function $f_t(\cdot, \mathbf{u})$, and denote it by $\mathbf{Epi}f_t(\cdot, \mathbf{u}) := \{(\mathbf{x}, \pi) \mid \pi \geq f_t(\mathbf{x}, \mathbf{u})\}$. So, it is clear that $\mathbf{Epi}f_t(\cdot, \mathbf{u}) \in \mathcal{X} \times \mathbb{R}$.

Definition 6.16 Let's assume that $f_t(\mathbf{x}, \mathbf{u})$ is a cost function. The term $f_t \xrightarrow[\mathcal{X}]{l-prob} f_0$ is defined as epi-convergence, and means that the epi-graph of the cost function $f_t(\mathbf{x}, \mathbf{u})$ (with constraints) of the sequence of optimizations converges in probability to the epi-graph of the cost function $f_0(\mathbf{x}, \mathbf{u})$ (with constraints) in the limiting optimization problem.

Remark 6.14 if $f_t \xrightarrow[\mathcal{X}]{l-prob} f_0$ holds, then the convergence of the sequence of minimizers is proven [18]. Note that for a fixed $\mathbf{u}$, the support of function f is $\mathcal{X}$.

The notions of epi-graph and epi-convergence help us to prove the convergence of the controlling commands of LBMPC and linear MPC with the true model. For clarification, let's denote how the objective function of the true model can be derived.

Remark 6.15 If we replace $\mathcal{O}_t(\tilde{\mathbf{x}}_t, \tilde{\mathbf{u}}_t)$ in the formulation of the constraint of the optimization problem of LBMPC with $g(\mathbf{x}_t, \mathbf{u}_t)$ and calculate the cost function based on the values of $\mathbf{x}$, the resulting controller is linear MPC with the true model.

Keeping the above remark for the standard formulation of objective function presented in Eq. 6.7 in mind, let's consider that the optimization problems of both LBMPC and linear MPC

are formulated in the format given in Eq. 6.9. Also, let's denote the cost function of linear MPC by $\underset{\sim}{J}_0$. If we can show that the cost function of LBMPC, i.e. $\underset{\sim}{J}$, which depends on $\mathcal{O}_t(\tilde{\mathbf{x}}_t, \tilde{\mathbf{u}}_t)$ converges in an appropriate manner to $\underset{\sim}{J}_0$, which depends on $g(\mathbf{x}_t, \mathbf{u}_t)$, then we get the convergence of the controlling commands calculated by LBMPC and linear MPC formulated based on *Remark 6.15*. Obviously, the support of $\underset{\sim}{J}$ at time t is $\Upsilon(\mathbf{x}_t)$ (for interpretation, just imagine we are using a heavy-side penalization, and simply execute the violating solutions).

The above claim is ascertained in the light of theoretical findings of [174] pertaining to epi-convergence. There are a number of tools and concepts within the realm of mathematical statistics that are required for understanding the logic of the proof. So, before stating the proof, it is intended to clarify the definitions and their implications to ease the understanding of the underlying proof.

Let's start by providing a schematic illustration for $\mathbf{Epi} f_t(\cdot, \mathbf{u})$ on $2D$ space. Recall the definition of epi-graph stated in *Definition 6.15*. A typical $\mathbf{Epi} f_t(\cdot, \mathbf{u})$ for a fixed $\mathbf{u}_O$, and $\mathbf{x}_t \in \mathcal{X} \subseteq \mathbb{R}$ (with only one element $n = 1$) is expressed as Figure 6.2.

It is clear that the epi-graph of a function is itself a set. So, as the epi-convergence studies the convergence of $\mathbf{Epi}\ \underset{\sim}{J} \in \underset{\sim}{\mathcal{J}}$ to $\mathbf{Epi}\ \underset{\sim}{J}_0 \in \underset{\sim}{\mathcal{J}}_0$, we need a measure of distance between two compact sets $\underset{\sim}{\mathcal{J}} \in \mathbb{R}^+$ and $\underset{\sim}{\mathcal{J}}_0 \in \mathbb{R}^+$. . In particular, we are interested in a type of distance measure which gives the lowest possible distance of two sets (by identifying the closest points of two sets). The notion of distance between two sets is defined as:

$$\mathbf{Dist}(\underset{\sim}{\mathcal{J}}, \underset{\sim}{\mathcal{J}}_0) = \inf_{\underset{\sim}{J} \in \underset{\sim}{\mathcal{J}}} \Big\{ \inf_{\underset{\sim}{J}_0 \in \underset{\sim}{\mathcal{J}}_0} \{|\underset{\sim}{J}_0 - \underset{\sim}{J}|\} \Big\} \ .$$

A typical schematic illustration is given in Figure 6.3 to visualize the output of the above metric.

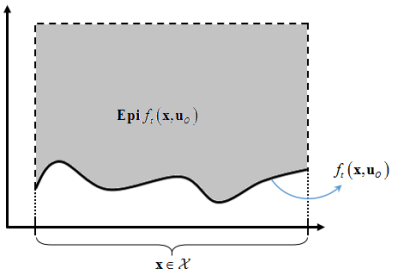

Figure 6.2: Schematic illustration of epi-graph for a given function with fixed $\mathbf{u}_O$.

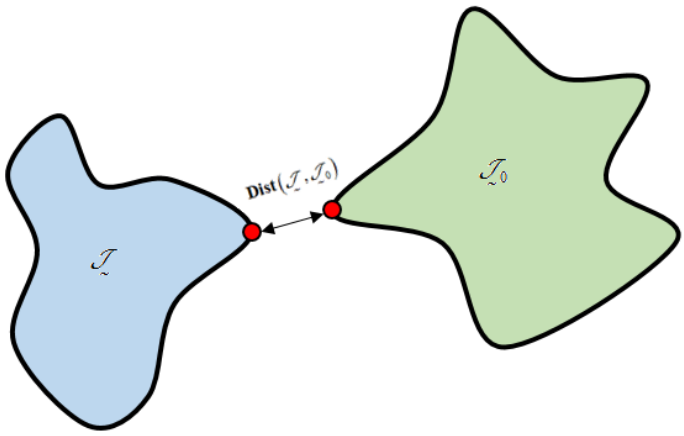

Figure 6.3: Schematic illustration of considered distance between two sets.

Now, it is the time to present two key definitions and discuss what they imply.

Definition 6.17 Let's define the set of all compact subsets of $\mathbb{R}^p$ by $\mathbb{C}^p$.

Definition 6.18 An open ball in $\mathcal{X}$ centered at $\mathbf{p} \in \mathcal{X}$ with radius r is denoted by $\mathbf{B}_r(\mathbf{p})$, and is mathematically expressed as:

$$\mathbf{B}_r(\mathbf{p}) = \{\mathbf{x} \in \mathcal{X} \mid \|\mathbf{x} - \mathbf{p}\|_2 < r\} \ .$$

Definition 6.19 A set $\mathcal{S} \subseteq \mathcal{X}$ is called the r-neighborhood of a set $\mathcal{P} \subseteq \mathcal{X}$ if it contains all points $\mathbf{x} \in \mathcal{X}$ that are located in a distance less than r from $\mathcal{P}$. It can be also interpreted as the union of all open balls $\mathbf{B}_r(\mathbf{p}) \in \mathcal{X}$, for $\forall \mathbf{p} \in \mathcal{P}$. We denote such a set by $\mathcal{S}_r\mathcal{P}$. This can be mathematically expressed as:

$$\mathcal{S}_r\mathcal{P} = \bigcup_{\mathbf{p}\in\mathcal{P}} \mathbf{B}_r(\mathbf{p}) \ .$$

Figure 6.4 presents the schematic illustration of the definition above. Apparently, we have $\mathcal{P} \subseteq \mathcal{S}_r\mathcal{P}$. Note that this set is an open set, as it is the union of unit balls.

Definition 6.20 The closure of $\mathcal{S}_r\mathcal{P}$ is denoted by $clo(\mathcal{S}_r\mathcal{P})$.

Definition 6.21 Recall that $\mathcal{X} \in \mathbb{R}^n$. For two functions $f_t, h_t : \mathbb{R}^n \times \mathcal{U} \to \mathbb{R}^+$, we define:

$$D_{l,r}(f_t, h_t, \mathcal{X}; \mathbf{u}) = \Big\{clo\Big(\mathbf{Epi} f_t(\cdot, \mathbf{u}) \cap \{clo(\mathcal{S}_{1/2}\mathcal{X}) \times \mathbb{R}^+\}\Big)\Big\} \setminus \Big\{\mathcal{S}_r\{\mathbf{Epi}\ h_t(\cdot, \mathbf{u}) \cap \{\mathcal{X} \times \mathbb{R}^+\}\}\Big\} \ .$$

In English, the set $D_{l,r}(f_t, h_t, \mathcal{X}; \mathbf{u})$ includes each point of the epi-graph of f_t in the argument $\mathcal{S}_{1/2}\mathcal{X}$, but with a distance greater than or equal to r from the epi-graph of h_t in the argument $\mathcal{X}$.

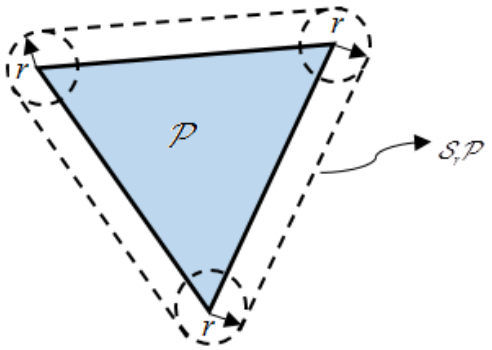

Figure 6.4: Schematic illustration of r-neighborhood of a set.

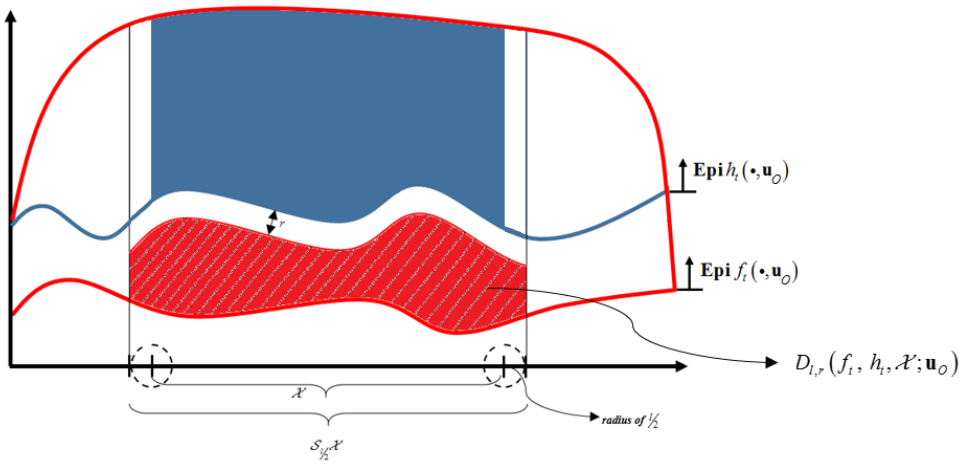

Figure 6.5: Schematic illustration of $D_{l,r}(f_t, h_t, \mathcal{X}; \mathbf{u}_O)$ for a fixed $\mathbf{u}_O$. The blue profile represents $h_t(\cdot, \mathbf{u})$, and the red profile represents $f_t(\cdot, \mathbf{u})$, respectively.

In Figure 6.5, a $2D$ view is presented with fixed $\mathbf{u}_O$, and $\mathbf{x}_t \in \mathcal{X} \subseteq \mathbb{R}$ (with only one element $n = 1$) to make the understanding of $D_{l,r}(f_t, h_t, \mathcal{X}; \mathbf{u})$ easier. In this figure, the set above the blue line represents **Epi** $h_t(\cdot, \mathbf{u_O})$, and the set above the red line represents **Epi** $f_t(\cdot, \mathbf{u_O})$. Note that the x-axis covers the whole $\mathbb{R}$. The region of interest, i.e. $\mathcal{S}_{1/2}\mathcal{X}$, and the set formed by this argument is separated by two vertical lines. The parts of **Epi** $h_t(\cdot, \mathbf{u_O})$ in this region is shown by blue color. Note that **Epi** $f_t(\cdot, \mathbf{u_O})$ in this region includes all of the space above the red profile, i.e. the red color region, the white region and the blue region. The region in red represents the set

$D_{l,r}(f_t, h_t, \mathcal{X}; \mathbf{u}_O)$, with a distance greater than or equal to r from **Epi** $h_t(\cdot, \mathbf{u_O})$. This distance is represented by the white region between the sets $D_{l,r}(f_t, h_t, \mathcal{X}; \mathbf{u}_O)$ and **Epi** $h_t(\cdot, \mathbf{u_O})$, and is selected such that:

$$\mathbf{Dist}\left(\left\{clo\Big(\mathbf{Epi} f_t(\cdot, \mathbf{u}) \cap \{clo(\mathcal{S}_{1/2}\mathcal{X}) \times \mathbb{R}^+\}\Big)\right\}, \left\{\mathbf{Epi}\ h_t(\cdot, \mathbf{u}) \cap \{\mathcal{X} \times \mathbb{R}^+\}\right\}\right) = r\ .$$

So, it is clear that $D_{l,r}(f_t, h_t, \mathcal{X}; \mathbf{u}) \in \mathcal{X} \times R^+$.

Definition 6.21 enables us to derive a probabilistic description for $f_t \xrightarrow[\mathcal{X}]{l-prob} f_0$ given in *Definition 6.16*. In this way, we have $f_t \xrightarrow[\mathcal{X}]{l-prob} f_0$, if:

$$\forall r > 0,\ \forall \mathbf{k} \in \mathbb{C}^{n+1}\ \ni\ \lim_{\substack{t\to\infty \\ l\to\infty}} \mathcal{P}r(\{\mathbf{u} \mid D_{l,r}(f_t, f_0, \mathcal{X}; \mathbf{u}) \cap \mathbf{k} \neq \emptyset\}) = 0\ . \tag{6.13}$$

Based on the mathematical representation given in Eq. 6.13, and also the geometrical scheme presented in Figure 6.5, that would be easy to interpret *Definition 6.16*. In this context, Eq. 6.13 simply implies that the red region in Figure 6.5 gradually contracts so that it asymptotically becomes an empty set, i.e. probabilistic convergence occurs, since we cannot find any $\mathbf{u}$ that results in the red region. Having said that the difference of this approximate type of convergence with deterministic convergence comes from the threshold r. The lesser the value of r, the better we can identify the differences between the two functions f_t and f_0, and the more strict we become regarding the convergence of the two functions.

Note that since $\underset{\sim}{J}, \underset{\sim}{J}_0 : \mathbb{R}^n \times \mathcal{U} \to \mathbb{R}^+$, all of the above probabilistic measures and concepts can be applied to them. The element $\mathbf{u}$ in the general definitions can be interpreted as the control input of LBMPC. In the same fashion, it can be stated that Eq. 6.13 makes it possible to have an interpretation from the control view point. In this context, $\underset{\sim}{J} \xrightarrow[\Upsilon(\mathbf{x}_t)]{l-prob} \underset{\sim}{J}_0$ states that for any given control command $\mathbf{u}$, the two cost functions $\underset{\sim}{J}_0$ and $\underset{\sim}{J}$ will converge to each other probabilistically. Again, the tolerance of approximation depends on the rate r.

Based on the detailed explanation of the probabilistic and geometric implications of the above tools, now, we can proceed with the final fundamental proof.

Theorem 6.8 [174] *If* $\underset{\sim}{J} \xrightarrow[\Upsilon(\mathbf{x}_t)]{l-prob} \underset{\sim}{J}_0$ *for all* $\{\mathbf{x}_t \mid \Upsilon(\mathbf{x}_t) \neq \emptyset\}$ *(which is the subset of state space for which all of the constraints are satisfied), then the set of minimizers converges, i.e.*:

$$\arg\min\left\{\underset{\sim}{J} \mid \boldsymbol{\theta} \in \Upsilon(\mathbf{x}_t)\right\} \xrightarrow{\ p\ } \arg\min\left\{\underset{\sim}{J}_0 \mid \boldsymbol{\theta} \in \Upsilon(\mathbf{x}_t)\right\}\ .$$

Proof. The proof follows from the concept of inclusion. Let's define two events as:

$$\textbf{event 1}: \ \{\underset{\sim}{J} \xrightarrow[\Upsilon(\mathbf{x}_t)]{l-prob} \underset{\sim}{J_0}\} \ .$$

$$\textbf{event 2}: \ \arg\min\{\underset{\sim}{J} \mid \boldsymbol{\theta} \in \Upsilon(\mathbf{x}_t)\} \xrightarrow{\ p\ } \arg\min\{\underset{\sim}{J_0} \mid \boldsymbol{\theta} \in \Upsilon(\mathbf{x}_t)\} \ .$$

The proof of the theorem follows by showing that **event 1** $\cap$ **event 2** = **event 2**. Just note that due to stability concerns, we have used a feedback-based control law calculation, and thus, $\mathbf{u}$ is itself the function of $\boldsymbol{\theta}$, which is the true decision variable vector. So, the probabilistic analysis is carried out using $\boldsymbol{\theta}$ as the control input. Also, recall that the objective function of the controller is subjected to constraints, and we penalize the solutions violating the constraints, i.e. the value function for $\boldsymbol{\theta} \notin \Upsilon(\mathbf{x}_t)$, is penalized. Such a manipulation brings us to the conclusion that:

$$\lim_{t\to\infty} \mathcal{P}r\Big(\{\boldsymbol{\theta} \mid \inf_{\mathbf{x}_t \notin \Upsilon(\mathbf{x}_t)} \underset{\sim}{J}(\mathbf{x}_t, \boldsymbol{\theta}) > \max\{\inf_{\mathbf{x}_t \in \Upsilon(\mathbf{x}_t)} \underset{\sim}{J_0}(\mathbf{x}_t, \boldsymbol{\theta}), \xi_0(\boldsymbol{\theta})\}\}\Big) = 1 \ ,$$

where $\xi_0 : \mathcal{U} \to \mathbb{R}^+$, and $\lim_{t\to\infty} \sup\{\xi_t\} \leq \xi_0$, almost surely. It is also a property of penalization techniques that they try to gradually steer the solutions towards the feasible region, and thus, the rate of penalization decreases gradually so that the value function of penalized solution finally converges to the value function of feasible solutions. Based on this, and also the explanation we gave for probabilistic "soft" convergence, it is true that for some r, we get:

$$\inf_{\mathbf{x}_t \notin \Upsilon(\mathbf{x}_t)} \underset{\sim}{J}(\mathbf{x}_t, \boldsymbol{\theta}) \quad \xrightarrow{\ l-prob\ } \quad \inf_{\mathbf{x}_t \in \Upsilon(\mathbf{x}_t)} \underset{\sim}{J_0}(\mathbf{x}_t, \boldsymbol{\theta}) \ . \tag{6.14}$$

Also, based on the assumption of the theorem, we know that $\underset{\sim}{J} \xrightarrow[\Upsilon(\mathbf{x}_t)]{l-prob} \underset{\sim}{J_0}$. Now, the remaining part of the proof is to show that under the assumptions made in the theorem and the condition derived in Eq. 6.14, it holds that **event 1** $\cap$ **event 2** = **event 2**. It is very important to keep in mind that, according to *Remark 6.15*, the objective function of linear MPC with the true model always remains the same, and is not time-varying. So, we have that $\underset{\sim}{J_0}$ and **Epi** $\underset{\sim}{J_0}(\cdot, \mathbf{u}_O)$ remain constant for fixed $\mathbf{u}_O$. However, the cost function of LBMPC, i.e. $\underset{\sim}{J}$, is time-varying and changes over time because of $\mathcal{O}_t(\tilde{\mathbf{x}}_t, \tilde{\mathbf{u}}_t)$. Thus, **Epi** $\underset{\sim}{J}(\cdot, \mathbf{u}_O)$ varies over time, for a fixed $\mathbf{u}_O$. To have a vision, just refer to Figure 6.5 (the red profile and the space above it constitute **Epi** $\underset{\sim}{J}(\cdot, \mathbf{u}_O)$, and the blue profile and the space above it form **Epi** $\underset{\sim}{J_0}(\cdot, \mathbf{u}_O)$). Also, be noticed that based on the definition, we have that $\mathbf{x}_t \in \Upsilon(\mathbf{x}_t) \iff \mathbf{x}_t \in \underset{\sim}{\mathcal{X}}_{\textbf{feasible}}$. Now, the proof can be easily completed, as for some r, it holds that:

$$\{\boldsymbol{\theta} \mid \inf_{\mathbf{x}_t \in \Upsilon(\mathbf{x}_t)} \underset{\sim}{J}(\mathbf{x}_t, \boldsymbol{\theta}) \leq \Delta \ \ \& \ \inf_{\mathbf{y}_t \in \Upsilon(\mathbf{y}_t)} \underset{\sim}{J_0}(\mathbf{y}_t, \boldsymbol{\theta}) > \Delta + r\}$$

$$\subset \{\boldsymbol{\theta} \mid \exists \mathbf{x}_t \in \Upsilon(\mathbf{x}_t) \ni \underset{\sim}{J}(\mathbf{x}_t, \boldsymbol{\theta}) \leq \Delta \ \ \& \ \inf_{\mathbf{y}_t \in \Upsilon(\mathbf{y}_t)} \underset{\sim}{J}_0(\mathbf{y}_t, \boldsymbol{\theta}) > \Delta + r\}$$

$$\subset \{\boldsymbol{\theta} \mid D_{l,r}(\underset{\sim}{J}, \underset{\sim}{J}_0, \mathcal{X}_{\mathbf{feasible}}; \boldsymbol{\theta}) \ \cap \ \{\mathcal{X}_{\mathbf{feasible}} \times [\Delta, \Delta + r]\} \neq \emptyset\}, \ \forall l \in \mathbb{R}^+ \ .$$

Thus, **event 1** $\cap$ **event 2** = **event 2**, and the proof is complete. $\square$

At this point, all of the fundamental theories and information regarding the general architecture and the formulation of LBMPC are given. In next sections, more details are presented to complete the steps and issues pertinent to the safe implementation of LBMPC.

6.3 Tube-based MPC and Invariant Set Approximation

For the current simulation, the author restricts himself to the concept of *robust positively invariant set* for the calculation of Ω. Here, we give two definitions which precisely clarify the meaning of MRPIS.

Definition 6.21 The *minimal robust positively invariant set* (MRPIS) is a set that is a sub-set of every other robust positively invariant set. Let's denote such a set by $\mathcal{R}_\infty$ (for our case, it can be interpreted as $Proj_{\mathcal{X}}(\Omega)$). We are in particular interested in finding MRPIS, as it reduces the conservativeness of control law. It can be interpreted that $\mathcal{R}_\infty$ is the limit set of all possible trajectories of system's dynamics under bounded disturbance.

As mentioned, it is usually impossible to calculate $\mathcal{R}_\infty$, since it is an NP-Hard problem. Thus, approximated versions are used for real-life applications. It is important for any approximated invariant set to retain the properties given in Eq. 6.5 and Eq. 6.6. Due to the existence of low computational complexity algorithms for approximating MRPIS, as well as the fact that it retains the properties given in Eq. 6.5 and Eq. 6.6, it is vigorously being used by mature control theoreticians [161].

Definition 6.22 The approximated invariant set is denoted by $\hat{\Omega} \subseteq \mathcal{X} \times \mathbb{R}^m$. It is clear that the approximation deals with $Proj_{\mathcal{X}}(\Omega)$, which contains the set of states for which the system's dynamics remains in this set for any control input that lives in $\mathbb{R}^m$, provided that the disturbance be bounded. So when $(\mathbf{x}; \boldsymbol{\theta}) \in \hat{\Omega}$, we mean that $\mathbf{x} \in \mathcal{R}(\varpi, s)$, where the set $\mathcal{R}(\varpi, s)$ is an approximation of $Proj_{\mathcal{X}}(\Omega)$.

In this sub-section, we present an efficient computational tool which, for any disturbance represented by a polytope, not only results in an approximated $\hat{\Omega}$ close to Ω, but also retains the properties given in Eq. 6.5 and Eq. 6.6.

Remark 6.16 Since the considered approximated invariant set still satisfies the constraint satisfaction and disturbance invariant properties, all of the theoretical results presented before for the original Ω are still valid when using $\hat{\Omega}$ to formulate the controlling command of LBMPC.

Remark 6.17 The approximated invariant set is needed since we use it in the terminal state constraint in the formulation of our objective function. As mentioned, in Eq. 6.7, the tube $\mathcal{R}_\infty$ is equivalent to $Proj_{\mathcal{X}}(\Omega)$. By approximation, $Proj_{\mathcal{X}}(\Omega)$ is obtained by the approximated set $\mathcal{R}(\varpi, s)$.

Here, we present an important geometric definition which is useful for approximating $\mathcal{R}_\infty$.

Definition 6.23 The p-norm ball in $\mathbb{R}^n$ is defined as $\mathfrak{B}_p^n(\epsilon) := \{\mathbf{x} \in \mathbb{R}^n \mid \|\mathbf{x}\|_p \leq \epsilon\}$ where $\|\cdot\|_p$ denotes the p-norm.

Based on $\mathfrak{B}_p^n(\epsilon)$, the outer and inner ϵ-approximation of an arbitrary finite set $\mathcal{A}$ can be defined.

Definition 6.24 For $\epsilon > 0$ and a given finite set $\mathcal{A} \in \mathbb{R}^n$, $\mathcal{B} \in \mathbb{R}^n$ is an inner ϵ-approximation of $\mathcal{A}$ if $\mathcal{B} \subseteq \mathcal{A} \subseteq \mathcal{B} \oplus \mathfrak{B}_p^n(\epsilon)$.

Definition 6.25 For $\epsilon > 0$ and a given finite set $\mathcal{A} \in \mathbb{R}^n$, $\mathcal{B} \in \mathbb{R}^n$ is an outer ϵ-approximation of $\mathcal{A}$ if $\mathcal{A} \subseteq \mathcal{B} \subseteq \mathcal{A} \oplus \mathfrak{B}_p^n(\epsilon)$.

For a given strictly positive integer number s, let the compact and convex set $\mathcal{R}_s$ be defined as:

$$\mathcal{R}_s := \bigoplus_{j=0}^{s-1} (A+BK)^j \mathcal{W} \ . \tag{6.15}$$

Recall that $\mathcal{R}_0$ has only one element which is $\mathbf{0}_{n\times 1}$. The set $\mathcal{R}_s$ is itself a sub-set of $\mathcal{R}_\infty$. Since $A+BK$ is stable, $\mathcal{R}_s \to \mathcal{R}_\infty$ as $s \to \infty$. This means that for a given $\epsilon > 0$, one can choose an s such that $\mathcal{R}_s$ becomes an inner ϵ-approximation of $\mathcal{R}_\infty$. Based on Eq. 6.15, the exact $Proj_{\mathcal{X}}(\Omega)$ can be obtained by $\bigoplus_{j=0}^{\infty}(A+BK)^j\mathcal{W}$, which is computationally intractable.

Remark 6.17 It is apparent that $\mathcal{R}_s \subseteq \mathcal{R}_\infty$.

It can be inferred from Eq. 6.15 that if we can find an s, and a scalar $\varpi \in [0, 1)$ such that $(A+BK)^s = \varpi \mathbb{I}_n$, then $\mathcal{R}_\infty = (1-\varpi)^{-1}\mathcal{R}_s$, because:

$$\begin{aligned}
\mathcal{R}_\infty &= \bigoplus_{j=0}^{\infty}(A+BK)^j\mathcal{W} = \left(\bigoplus_{j=0}^{s-1}(A+BK)^j\mathcal{W}\right) \oplus \left(\bigoplus_{j=s}^{\infty}(A+BK)^j\mathcal{W}\right) \\
&= \mathcal{R}_s \ \oplus (A+BK)^s\mathcal{W} \ \oplus (A+BK)^s(A+BK)\mathcal{W} \oplus (A+BK)^s(A+BK)^2\mathcal{W} \oplus ... \\
&= \mathcal{R}_s \ \oplus \varpi\mathbb{I}_n\mathcal{W} \ \oplus \varpi(A+BK)\mathcal{W} \oplus \varpi(A+BK)^2\mathcal{W} \oplus \varpi(A+BK)^3\mathcal{W} \oplus ... \\
&= \mathcal{R}_s \oplus \ \varpi\left(\bigoplus_{j=0}^{\infty}(A+BK)^j\mathcal{W}\right) = \mathcal{R}_s \oplus \varpi\mathcal{R}_\infty \ .
\end{aligned}$$

Also, it can be inferred from Eq. 6.15 that if $(A+BK)$ be nilpotent for s, i.e. $(A+BK)^s = 0$, then $\mathcal{R}_\infty = \mathcal{R}_s$, because:

$$\mathcal{R}_\infty = \bigoplus_{j=0}^{\infty}(A+BK)^j\mathcal{W} = \left(\bigoplus_{j=0}^{s-1}(A+BK)^j\mathcal{W}\right) \oplus \left(\bigoplus_{j=s}^{\infty}(A+BK)^j\mathcal{W}\right)$$

$$= \mathcal{R}_s \ \oplus (A+BK)^s\mathcal{W} \ \oplus (A+BK)^s(A+BK)\mathcal{W} \oplus (A+BK)^s(A+BK)^2\mathcal{W} \oplus ...$$

$$= \mathcal{R}_s \ \oplus (\mathbf{0}_{n\times 1})\mathcal{W} \ \oplus (\mathbf{0}_{n\times 1})(A+BK)\mathcal{W} \oplus (\mathbf{0}_{n\times 1})(A+BK)^2\mathcal{W} \oplus ...$$

$$= \mathcal{R}_s \ .$$

By the same argument, it can be easily concluded that for $(A+BK)$ which is not nilpotent, it is impossible to find a finite s such that $\mathcal{R}_s = \mathcal{R}_\infty$. Given *Remark 6.18*, and *Definition 6.24*, it can be inferred that without having a nilpotent $(A+BK)$, none of the sets in the sequence $\{\mathcal{R}_s \mid s = 0, 1, 2, ...\}$ can be an exact inner approximation of the set $\mathcal{R}_\infty$.

Fortunately, when $(A+BK)$ is not nilpotent, it is possible to have an outer approximation for the set $\mathcal{R}_\infty$. As mentioned before, it is important to make sure the outer approximated set is as close as possible to $\mathcal{R}_\infty$, and thus, the concept defined in *Definition 6.25* can be of great use. As far as the author is concerned, in [163], one of the most efficient strategies for approximating $\mathcal{R}_\infty$ is presented.

In this context, an outer approximation of $\mathcal{R}_\infty$ is conducted by first computing a sufficiently large s, and constructing $\mathcal{R}_s$ accordingly, and then, scaling it suitably by ϖ. The authenticity of the method proposed in [163] lies in its theoretical foundation which indicates that such an approximation is capable of resulting invariant set, which is a very important concern for guaranteeing the robustness and safe performance of the controller.

Theorem 6.9 [175] *If $\mathbf{0}_{n\times 1} \in int(\mathcal{W})$ and there exists a finite integer s and a scalar $\varpi \in [0, 1)$ such that $(A+BK)^s \ \mathcal{W} \subseteq \varpi\mathcal{W}$, the set $\mathcal{R}(\varpi, s) := (1-\varpi)^{-1}\mathcal{R}_s$ is a convex and compact robust positively invariant set for Eq. 6.3, $\mathbf{0}_{n\times 1} \in int(\mathcal{R}(\varpi, s))$, and also $\mathcal{R}(\varpi, s)$ is an outer approximation of MRPIS $\mathcal{R}_\infty$, i.e. $\mathcal{R}_\infty \subseteq \mathcal{R}(\varpi, s)$.*

Proof. [175] The theory comprises of different parts. Firstly, it can be understood from Eq. 6.15 that $\mathcal{R}(\varpi, s)$ is compact and convex since $\mathcal{R}_s$ is the Minkowski sum of a finite set of convex and compact sets. The proof of the main claim of this theorem is straightforward and can be given as follows:

$$(A+K)\mathcal{R}(\varpi, s) \oplus \mathcal{W} = (A+K)\left((1-\varpi)^{-1}\bigoplus_{j=0}^{s-1}(A+BK)^j\mathcal{W}\right) \oplus \mathcal{W}$$

$$= (1-\varpi)^{-1}\bigoplus_{j=1}^{s}(A+BK)^j\mathcal{W} \ \oplus \mathcal{W}$$

$$= (1-\varpi)^{-1}(A+BK)^s\mathcal{W} \oplus \left((1-\varpi)^{-1}\bigoplus_{j=1}^{s-1}(A+BK)^j\mathcal{W}\right) \oplus \mathcal{W} \ .$$

Now, from the properties given in *Definition 6.4* (in particular $\mathcal{A} \subseteq \mathcal{B} \iff \mathcal{A}\oplus\mathcal{C} \subseteq \mathcal{B}\oplus\mathcal{C}$), and the assumption that $(A+BK)^s\mathcal{W} \subseteq \varpi\mathcal{W}$, the proof can be proceed as:

$$\begin{aligned}
&(1-\varpi)^{-1}(A+BK)^s\mathcal{W} \oplus \left((1-\varpi)^{-1}\bigoplus_{j=1}^{s-1}(A+BK)^j\mathcal{W}\right) \oplus \mathcal{W} \\
&\subseteq (1-\varpi)^{-1}\varpi\mathcal{W} \oplus \mathcal{W} \oplus \left((1-\varpi)^{-1}\bigoplus_{j=1}^{s-1}(A+BK)^j\mathcal{W}\right) \\
&= [1+\frac{\varpi}{1-\varpi}]\ \mathcal{W} \oplus \left((1-\varpi)^{-1}\bigoplus_{j=1}^{s-1}(A+BK)^j\mathcal{W}\right) \\
&= (1-\varpi)^{-1}\mathcal{W} \oplus \left((1-\varpi)^{-1}\bigoplus_{j=1}^{s-1}(A+BK)^j\mathcal{W}\right) \\
&= (1-\varpi)^{-1}\bigoplus_{j=0}^{s-1}(A+BK)^j\mathcal{W} = \mathcal{R}(\varpi,s) \ .
\end{aligned}$$

This implies that $(A+BK)\mathcal{R}(\varpi,s) \oplus \mathcal{W} \subseteq \mathcal{R}(\varpi,s)$. Therefore, $\mathcal{R}(\varpi,s)$ is robust positively invariant. Also, since $\mathcal{R}_\infty$ is the minimal robust positively invariant set, based on *Definition 6.21*, it holds that $\mathcal{R}_\infty \subseteq \mathcal{R}(\varpi,s)$. Now, given that $\mathbf{0}_{n\times 1} \in int(\mathcal{W})$, it is inferred that $\mathbf{0}_{n\times 1} \in int(\mathcal{R}_\infty)$, which directly implies that $\mathbf{0}_{n\times 1} \in int(\mathcal{R}(\varpi,s))$. So, the proof is complete. □

Though the above theorem states that $\mathcal{R}_\infty \subseteq \mathcal{R}(\varpi,s)$, it is not enough to make sure $\mathcal{R}(\varpi,s)$ is a good approximation of $\mathcal{R}_\infty$. As mentioned, it is also important to make sure the approximated set is close to the original one. Apparently, the structure of $\mathcal{R}(\varpi,s)$ is related to the selected ϖ and s. Therefore, the limiting behavior and the properness of the error bound of the approximation will be investigated theoretically with respect to selecting a sufficiently large s or equivalently selecting a small ϖ.

Now, the goal is to investigate the limiting behavior of $\mathcal{R}(\varpi,s)$. Prior to proceeding with the main theorem, a mathematical tool is defined which then will be followed by a lemma and a well-known theorem.

Definition 6.26 If $\mathcal{A}$ and $\mathcal{B}$ be two non-empty and compact sets in $\mathbb{R}^n$, then the *Hausdorff metric* is defined as:

$$\mho(\mathcal{A},\mathcal{B}) := \max\Big\{\sup_{\mathbf{a}\in\mathcal{A}}\big\{\inf_{\mathbf{b}\in\mathcal{B}}\|\mathbf{a}-\mathcal{B}\|_p\big\}; \sup_{\mathbf{b}\in\mathcal{B}}\big\{\inf_{\mathbf{a}\in\mathcal{A}}\|\mathbf{b}-\mathcal{A}\|_p\big\}\Big\} \ .$$

Lemma 6.3 [161] *If $\mathcal{A} \in \mathbb{R}^n$ be a convex and compact set containing the origin and $\varpi \in [0,1)$, then it holds that $\mho(\mathcal{A}, (1-\varpi)^{-1}\mathcal{A}) \leq \varpi(1-\varpi)^{-1} \sup_{\mathbf{a}\in\mathcal{A}} \|\mathbf{a}\|_p$.*

Proof. See the results in *Appendix II* of [161]. □

Theorem 6.10 [154] *The sequence $\{\mathcal{R}_s \mid s = 0, 1, 2, ...\}$ is Cauchy, and thus, $\mathcal{N}_\infty := \lim_{s\to\infty} \sup_{\mathbf{x}\in\mathcal{R}_s} \|\mathbf{x}\|_p$ is finite.*

Proof. See the proof of *Theorem 4.1* of [154]. □

Remark 6.19 Since $\mathcal{R}_s \subseteq \mathcal{R}_\infty$, it holds that $\mathcal{N}_s := \sup_{\mathbf{x}\in\mathcal{R}_s} \|\mathbf{x}\|_p \leq \mathcal{N}_\infty < \infty$, for all s.

Theorem 6.11 [161] *Let $\varpi^*(s) := \min\{\varpi \in \mathbb{R}^+ \mid (A+BK)^s\mathcal{W} \subseteq \varpi\mathcal{W}\}$ for a given s (note that $\varpi^*(s) \in [0,1)$ if s be sufficiently large), and $s^*(\varpi) := \min\{s = 0, 1, 2, ... \mid (A+BK)^s\mathcal{W} \subseteq \varpi\mathcal{W}\}$ for a given $\varpi \in [0,1)$. If $\mathbf{0}_{n\times 1} \in int(\mathcal{W})$, then (a) $\mathcal{R}(\varpi^*(s), s) \to \mathcal{R}_\infty$, as $s \to \infty$, and (b) $\mathcal{R}(\varpi, s^*(\varpi)) \to \mathcal{R}_\infty$, as $\varpi \searrow 0$.*

Proof. [161] The proof of each part is given separately. Let's start by proving (a). The proof can be easily derived by taking into account *Lemma 6.3* and *Remark 6.19*. From *Lemma 6.3*, it holds that $\mho(\mathcal{R}_s, (1-\varpi^*(s))^{-1}\mathcal{R}_s) = \mho(\mathcal{R}_s, \mathcal{R}(\varpi^*(s), s)) \leq \varpi^*(s)(1-\varpi^*(s))^{-1}\mathcal{N}_s$. Based on *Remark 6.19*, we know $\mathcal{N}_s \leq \mathcal{N}_\infty < \infty$, for all s. So, since $\varpi^*(s) \searrow 0$, as $s \to \infty$, it holds that $\mho(\mathcal{R}_s, \mathcal{R}(\varpi^*(s), s)) \to 0$, as $s \to \infty$. On the other hand, based on the definition, we have $\mathcal{R}_s \subseteq \mathcal{R}_\infty \subseteq \mathcal{R}(\varpi^*(s), s)$ for all s, and $\mathcal{R}_s \to \mathcal{R}_\infty$, as $s \to \infty$, we get that $\mathcal{R}(\varpi^*(s), s) \to \mathcal{R}_\infty$, as $s \to \infty$. This completes the proof of (a). A relatively similar argument holds for (b). Again, from *Lemma 6.3*, $\mho(\mathcal{R}_{s^*(\varpi)}, (1-\varpi)^{-1}\mathcal{R}_{s^*(\varpi)}) = \mho(\mathcal{R}_{s^*(\varpi)}, \mathcal{R}(\varpi, s^*(\varpi))) \leq \varpi(1-\varpi)^{-1}\mathcal{N}_{s^*(\varpi)}$. Based on *Remark 6.19*, $\mathcal{N}_{s^*(\varpi)} \leq \mathcal{N}_\infty < \infty$, for all $\varpi \in [0,1)$. So, $\mho(\mathcal{R}_{s^*(\varpi)}, \mathcal{R}(\varpi, s^*(\varpi))) \to 0$, as $\varpi \searrow 0$. Also, $\varpi \searrow 0$ implies $s^*(\varpi) \to \infty$. Again, based on the definition, we have $\mathcal{R}_{s^*(\varpi)} \subseteq \mathcal{R}_\infty \subseteq \mathcal{R}(\varpi, s^*(\varpi))$ for all $\varpi \in [0,1)$, and $\mathcal{R}_{s^*(\varpi)} \to \mathcal{R}_\infty$, as $\varpi \searrow 0$. Thus, it can be concluded that $\mathcal{R}(\varpi, s^*(\varpi)) \to \mathcal{R}_\infty$, as $\varpi \searrow 0$. This completes the proof. □

So far, it has been theoretically proven that there exists an outer approximation $(\mathcal{R}(\varpi, s))$ for $\mathcal{R}_\infty$. Also, it has been shown that $\mathcal{R}(\varpi, s)$ asymptotically converges to $\mathcal{R}_\infty$. he only remaining issue is to find out how well this approximation could be. Apparently, the precision of the approximated set depends on the controlling parameters, i.e. ϖ and s. In general, if *Theorem 6.11* holds, and also we have that:

$$\epsilon \geq \varpi(1-\varpi)^{-1} \sup_{\mathbf{x}\in\mathcal{R}_s} \|\mathbf{x}\|_p = \varpi(1-\varpi)^{-1} \min_{\zeta}\{\zeta \mid \mathcal{R}_s \subseteq \mathfrak{B}_p^n(\zeta)\} , \tag{6.16}$$

then, $\mathcal{R}_\infty \subseteq \mathcal{R}(\varpi, s) \subseteq \mathcal{R}_\infty \oplus \mathfrak{B}_p^n(\epsilon)$, which based on *Definition 6.25*, means that $\mathcal{R}(\varpi, s)$ is an outer ϵ-approximation of $\mathcal{R}_\infty$. In the next step, a theoretical result is given, which provides an evaluation of the precision of outer approximation.

Theorem 6.12 [161] *If $\mathbf{0}_{n\times 1} \in int(W)$, then for all $\epsilon > 0$, there exists $\varpi \in [0,1)$ and s such that $(A+BK)^s\mathcal{W} \subseteq \varpi\mathcal{W}$ and $\varpi(1-\varpi)^{-1}\mathcal{R}_s \subseteq \mathfrak{B}_p^n(\epsilon)$ holds, and thus, $\mathcal{R}(\varpi, s)$ is an outer ϵ-approximation of $\mathcal{R}_\infty$.*

Proof. [161] The proof is easy and straightforward. We know $\mathcal{R}_s \subseteq \mathcal{R}_\infty$. Recall from *Theorem 6.9* that if $\mathbf{0}_{n\times 1} \in int(\mathcal{W})$, then $\mathcal{R}_s$ and $\mathcal{R}_\infty$ are convex and contain the origin. Also, recall from *Theorem 6.10* that $\mathcal{N}_\infty = \lim_{s\to\infty} \sup_{\mathbf{x}\in\mathcal{R}_s} \|\mathbf{x}\|_p$, and $0 < \mathcal{N}_\infty < \infty$. Due to the convexity of $\mathcal{R}_s$ and $\mathcal{R}_\infty$, it directly follows that $\varpi(1-\varpi)^{-1}\mathcal{R}_s \subseteq \varpi(1-\varpi)^{-1}\mathcal{R}_\infty$, for any s and $\varpi \in [0,1)$. If $\varpi(1-\varpi)^{-1}\mathcal{N}_\infty \leq \epsilon$, it holds that $\varpi(1-\varpi)^{-1} \lim_{s\to\infty} \sup_{\mathbf{x}\in\mathcal{R}_s} \|\mathbf{x}\|_p \subseteq \{\mathbf{x} \mid \|\mathbf{x}\|_p \leq \epsilon\}$, and thus, $\varpi(1-\varpi)^{-1}\mathcal{R}_\infty \subseteq \mathfrak{B}_p^n(\epsilon)$. So, by selecting $\varpi \in \big[0, \epsilon(\epsilon+\mathcal{N}_\infty)^{-1}\big)$, the validity of $\varpi(1-\varpi)^{-1}\mathcal{N}_\infty \leq \epsilon$ can be assured. Also, a corresponding $s^*(\varpi)$ can be selected to ensure $(A+BK)^s\mathcal{W} \subseteq \varpi\mathcal{W}$ holds as well. On the other hand, it holds that $\mathcal{R}(\varpi, s) = \varpi(1-\varpi)^{-1}\mathcal{R}_s = (1+\varpi(1-\varpi)^{-1})\mathcal{R}_s = \mathcal{R}_s \oplus \varpi(1-\varpi)^{-1}\mathcal{R}_s$. Since, we have $\mathcal{R}_s \subseteq \mathcal{R}_\infty$, and $\varpi(1-\varpi)^{-1}\mathcal{R}_\infty \subseteq \mathfrak{B}_p^n(\epsilon)$, it can be concluded that $\mathcal{R}(\varpi, s) \subseteq \mathfrak{B}_p^n(\epsilon)$. Also, since $\mathcal{R}(\varpi, s) = \mathcal{R}_s \oplus \mathcal{R}(\varpi, s)$, it can be concluded that $\mathcal{R}(\varpi, s) \subseteq \mathcal{R}_s \oplus \mathfrak{B}_p^n(\epsilon)$. This implies $\mathcal{R}(\varpi, s) \subseteq \mathcal{R}_\infty \oplus \mathfrak{B}_p^n(\epsilon)$. Based on *Definition 6.25*, it can be inferred that $\mathcal{R}(\varpi, s)$ is an outer ϵ-approximation of $\mathcal{R}_\infty$. Hence, the proof is complete. □

Remark 6.20 As can be inferred from condition $(A+BK)^s\mathcal{W} \subseteq \varpi\mathcal{W}$, a lower bound can be obtained for ϖ, and based on *Theorem 6.12*, an upper bound, i.e. $\varpi^{max} = \epsilon(\epsilon+\mathcal{N}_\infty)^{-1}$, can be found for ϖ. These bounded span assists us to come up with an algorithm for the calculation of $\mathcal{R}(\varpi, s)$.

Remark 6.21 As scrutinized before, it is arbitrary to whether set the s first and verity $\omega^*(s)$ afterward or to set ϖ first and verify $s^*(\varpi)$ accordingly. However, it is more computationally efficient to find the smallest s and then verify the corresponding $\omega^*(s)$. This is because the larger value of s can increase the computational cost, though ϖ is just a scale, and its magnitude does not increase the computational complexity.

Based on the above theoretical result, an efficient computational frame is presented for the calculation of $\mathcal{R}(\varpi, s)$. Having said that the algorithm is valid for cases that $\mathcal{W}$ is a polytope (which is the case for our study).

Definition 6.27 Let's define the support function of a set $\mathcal{W}$ at point $\mathbf{a}$ as:

$$\Psi_{\mathcal{W}}(\mathbf{a}) := \sup_{\mathbf{w}\in\mathcal{W}} \{\mathbf{a}^T\mathbf{w}\} \ .$$

Definition 6.28 Let's consider the general definition of polytope for representing $\mathcal{W}$, i.e. $\mathcal{W} := \{\mathbf{w} \in \mathbb{R}^n \mid \mathbf{h}_i^T\mathbf{w} \leq g_i \ ; \ i \in \mathcal{I}\}$, where $\mathbf{h}_i \in \mathbb{R}^n$, $g_i \in \mathbb{R}^+$, and $\mathcal{I}$ represents the index set.

If the representation of $\mathcal{W}$ be the one given in *Definition 6.28*, it holds that [154]:

$$(A+BK)^s\mathcal{W} \subseteq \varpi\mathcal{W} \iff \Psi_{\mathcal{W}}\big((A+BK)^s\mathbf{h}_i\big) \leq \varpi g_i, \quad \forall i \in \mathcal{I} \ . \tag{6.17}$$

Since $g_i \in \mathbb{R}^+$, Eq. 6.17 enables us to calculate a lower bound for ϖ, as below:

$$\varpi^*(s) = \max_{i\in\mathcal{I}} \frac{\Psi_{\mathcal{W}}\big((A+BK)^s\mathbf{h}_i\big)}{g_i} \ . \tag{6.18}$$

By the same strategy, it can be verified whether $\mathcal{R}_s$ is contained in the polytope $\mathcal{B}$ that can be represented as $\mathcal{B} := \{\mathbf{b} \in \mathbb{R}^n \mid \mathbf{l}_i^T \mathbf{b} \leq d_i \ ; \ j \in \mathcal{J}\}$, where $\mathbf{l}_i^T \in \mathbb{R}^n$, $d_i \in \mathbb{R}^+$, and $\mathcal{J}$ represents the index set. Based on the same argument given in [154], this holds when:

$$\mathcal{R}_s \subseteq \mathcal{B} \iff \sum_{i=0}^{s-1} \Psi_{\mathcal{W}}\big((A+BK)^i \mathbf{h}_j\big) \leq d_j, \ \forall j \in \mathcal{J} \ .$$

Also, it can be easily inferred that:

$$\mathcal{R}(\varpi, s) \subseteq \mathcal{B} \iff \mathcal{R}_s \subseteq (1-\varpi)\mathcal{B} \iff \sum_{i=0}^{s-1} \Psi_{\mathcal{W}}\big((A+BK)^i \mathbf{h}_j\big) \leq (1-\varpi) d_j, \ \forall j \in \mathcal{J} \ .$$

Given the results of *Theorem 6.12*, i.e. $\varpi(1-\varpi)^{-1}\mathcal{R}_s \subseteq \mathfrak{B}_p^n(\epsilon)$ and $\varpi^{max} = \epsilon(\epsilon + \mathcal{N}_\infty)^{-1}$, and the argument given above as well as the condition given in Eq. 6.16, an upper bound can be obtained for ϖ:

$$\varpi(1-\varpi)^{-1}\mathcal{R}_s \subseteq \mathfrak{B}_\infty^n(\zeta) \iff \varpi \leq \frac{\epsilon}{\epsilon + \mathcal{N}_s} \ . \tag{6.19}$$

Calculation of the above upper bound would be easy when using an ∞-norm, since we have:

$$\mathcal{N}_s := \sup_{\mathbf{x} \in \mathcal{R}_s} \|\mathbf{x}\|_p = \min_{\zeta}\{\zeta \mid \mathcal{R}_s \subseteq \mathfrak{B}_p^n(\zeta)\} \tag{6.20}$$

$$= \max_{j \in \{1,2,\ldots,n\}} \Big\{ \sum_{i=0}^{s-1} \Psi_{\mathcal{W}}\big((A+BK)^i \overrightarrow{\mathbf{e}}_j\big) \ , \ \sum_{i=0}^{s-1} \Psi_{\mathcal{W}}\big(-(A+BK)^i \overrightarrow{\mathbf{e}}_j\big) \Big\} \ ,$$

where $\overrightarrow{\mathbf{e}}$ represents the basis of the space in which the unit ball of the p-norm is defined.

It is apparent that Eq. 6.18 and Eq. 6.19 represent the lower and upper bounds of ϖ, respectively. The above results enable us to come up with an algorithm to compute $\mathcal{R}(\varpi, s)$. The algorithmic flowchart for the computation of $\mathcal{R}(\varpi, s)$ is given in Figure 6.6. Having said that the computation of $\mathcal{R}_s$ and consequently $\mathcal{R}(\varpi, s)$ for a given s is performed using a standard computational geometry software for computing the Minkowski sum of polytopes [176, 177].

6.4 Epi-convergence of Oracle

In this section, the general proofs discussed for the epi-convergence of the control law of LBMPC to linear MPC that knows the un-modeled dynamics will be extended for two types of parametric and non-parametric oracles. The necessary condition for ensuring the validity of the theoretical results presented in this section is *sufficient excitation* (SE). SE states that the control input

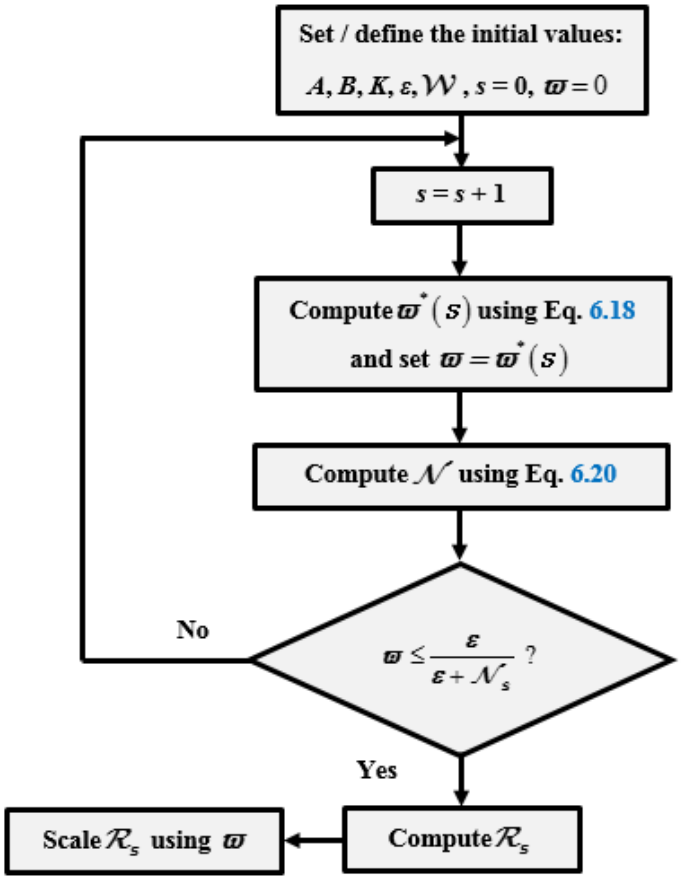

Figure 6.6: The block diagram for computing $\mathcal{R}(\varpi, s)$.

(actuation signal) and the state trajectory of a dynamic system should be chosen such that all of the modes of the system be excited. SE plays an important role for identifying dynamics systems, as it is important to make sure the learned system correctly captures the possible dynamics behavior of the true model, as a function of an actuation signal. There are so many ways to come up with a trajectory to sufficiently explore the state-input space $\mathcal{X} \times \mathcal{U}$. One of the most important ones is the reinforcement learning, which was scrutinized in Chapter 5. So, under SE (or at-least using a technique to ensure SE approximately holds), the theoretical results presented here can be easily extended to any choice of parametric and non-parametric oracles. Having said that ensuring SE affects the efficiency of the trained oracle, and does not have anything to do with the stability results presented before. Therefore, even if SE does not hold, thanks to the nominal model and the objective functioned used at the heart of LBMPC, it

can be still assured that the stability and robustness conditions are valid.

Let's first start by proving the epi-convergence for the class of parametric oracles. Recall that a parametric oracle is represented by $\mathcal{O}_t(\tilde{\mathbf{x}}_t, \tilde{\mathbf{u}}_t) = \wp(\tilde{\mathbf{x}}_t, \tilde{\mathbf{u}}_t \mid \boldsymbol{\lambda}_t)$, where $\boldsymbol{\lambda}_t = (\lambda_1, \lambda_2, ..., \lambda_p) \in \mathcal{Q}$ includes the parameters of the model, and $\mathcal{Q}$ is a set encapsulating the possible values of coefficients, and lives in $\mathbb{R}^p$. Before proceeding with the statement of the main theorem, two definitions, one remark and a well-known theorem should be presented.

Definition 6.29 Suppose that f is a function of $\mathbf{x}$, $\mathbf{u}$ and $g(\mathbf{x}, \mathbf{u})$, then $\hat{f}(\mathbf{x}, \mathbf{u}, h(\mathbf{x}, \mathbf{u} \mid \boldsymbol{\lambda}))$ represents an approximation of f where $h(\mathbf{x}, \mathbf{u} \mid \boldsymbol{\lambda})$ is a shape function for approximating $g(\mathbf{x}, \mathbf{u})$ with parameters $\boldsymbol{\lambda} \in \mathbb{R}^p$.

Definition 6.30 Suppose that $\mathbf{x} \in \mathcal{X} \subseteq \mathbb{R}^n$, $\mathbf{u} \in \mathcal{U} \subseteq \mathbb{R}^m$, and $\boldsymbol{\lambda} \in \mathcal{Q} \subseteq \mathbb{R}^p$, then a function $\hat{f} : \mathbb{R}^{n+m} \times \mathbb{R}^p \to \mathbb{R}$ is said to be lower semi continuous in $\mathcal{X} \times \mathcal{U} \times \mathcal{Q}$ if for each $\mathbf{x}_0 \in \mathcal{X}$, $\mathbf{u}_0 \in \mathcal{U}$, and $\boldsymbol{\lambda}_0 \in \mathcal{Q}$, it holds that $\lim\limits_{\substack{\mathbf{x}\to\mathbf{x}_0 \\ \mathbf{u}\to\mathbf{u}_0 \\ \boldsymbol{\lambda}\to\boldsymbol{\lambda}_0}} \inf\{\hat{f}(\mathbf{x}, \mathbf{u}, h(\mathbf{x}, \mathbf{u} \mid \boldsymbol{\lambda}))\} \geq \hat{f}(\mathbf{x}_0, \mathbf{u}_0, h(\mathbf{x}_0, \mathbf{u}_0 \mid \boldsymbol{\lambda}_0))$.

Remark 6.22 Obviously, the cost function $\underset{\sim}{J}$ of LBMPC can be interpreted as a lower semi continuous function. Also, it is obviously an approximation of $\underset{\sim}{J}_0$, and replaces the true dynamic model $g(\mathbf{x}, \mathbf{u})$ of linear MPC with $\wp(\tilde{\mathbf{x}}_t, \tilde{\mathbf{u}}_t \mid \boldsymbol{\lambda}_t)$. Also, under the condition that $\boldsymbol{\lambda}_0$ be the true vector of parameters, we get $\underset{\sim}{J} = \underset{\sim}{J}_0$. Actually, *Definition 6.30* makes the interpretation of the lower semi continuity of $\underset{\sim}{J}$ easy. As the cost function of LBMPC, there exists an optimal solution $\tilde{\mathbf{x}}_0 \in \mathcal{X}$ amd $\tilde{\mathbf{u}}_0 \in \mathcal{U}$ for $\underset{\sim}{J}$. Also, there exists an optimal parameter set $\boldsymbol{\lambda}_0$ for $\underset{\sim}{J}$. Even a little bit deviation from these optimal values results increases the error of $\underset{\sim}{J}$ compared to $\underset{\sim}{J}_0$. This means that:

$$\lim_{\substack{\mathbf{x}\to\mathbf{x}_0 \\ \mathbf{u}\to\mathbf{u}_0 \\ \boldsymbol{\lambda}\to\boldsymbol{\lambda}_0}} \inf\{\underset{\sim}{J}(\mathbf{x}, \mathbf{u}, h(\mathbf{x}, \mathbf{u} \mid \boldsymbol{\lambda}))\} \geq \underset{\sim}{J}(\mathbf{x}_0, \mathbf{u}_0, h(\mathbf{x}_0, \mathbf{u}_0 \mid \boldsymbol{\lambda}_0)) = \underset{\sim}{J}_0 \ .$$

Theorem 6.13 [174] *For a lower semi continuous function $\hat{f}$ with properties presented in Definition 6.30, it holds that*:

$$h(\mathbf{x}, \mathbf{u} \mid \boldsymbol{\lambda}_t) \xrightarrow{p} h(\mathbf{x}, \mathbf{u} \mid \boldsymbol{\lambda}_0) \Rightarrow \hat{f}_t \xrightarrow[\mathcal{X}\times\mathcal{U}]{l-prob} \hat{f}_0 \ .$$

Proof. [174] Since the function $h(\mathbf{x}, \mathbf{u} \mid \boldsymbol{\lambda}_t)$ depends on the coefficient vector $\boldsymbol{\lambda}_t$, $h(\mathbf{x}, \mathbf{u} \mid \boldsymbol{\lambda}_t) \xrightarrow{p} h(\mathbf{x}, \mathbf{u} \mid \boldsymbol{\lambda}_0)$ implies that $\boldsymbol{\lambda}_t \xrightarrow{p} \boldsymbol{\lambda}_0$. Let's consider sequences $\{\mathbf{u}_t\}_{t\in\mathbb{Z}^+}$ and $\{\mathbf{x}_t\}_{t\in\mathbb{Z}^+}$ such that $\mathbf{u}_t \to \mathbf{u}_0$ and $\mathbf{x}_t \to \mathbf{x}_0$ asymptotically. Recall the *Definition 6.18*. Given the lower semi continuity of $\hat{f}_t$ at $\mathbf{x}_0$, $\mathbf{u}_0$ and $\boldsymbol{\lambda}_0$, there exist $\epsilon > 0$ and $\delta > 0$ such that:

$$\hat{f}_t(\mathbf{x}_t, \mathbf{u}_t, h(\mathbf{x}_t, \mathbf{u}_t \mid \boldsymbol{\lambda}_t)) \geq \hat{f}(\mathbf{x}_0, \mathbf{u}_0, h(\mathbf{x}_0, \mathbf{u}_0 \mid \boldsymbol{\lambda}_0)) + \epsilon \quad \forall \mathbf{u} \in \mathbf{B}_r(\mathbf{u}_0), \forall \mathbf{x} \in \mathbf{B}_r(\mathbf{x}_0), \forall \boldsymbol{\lambda} \in \mathbf{B}_r(\boldsymbol{\lambda}_0) \ .$$

In other words, there exists a t_0 , such that:

$$\hat{f}_t(\mathbf{x}_t, \mathbf{u}_t, h(\mathbf{x}_t, \mathbf{u}_t \mid \boldsymbol{\lambda}_t)) \geq \hat{f}(\mathbf{x}_0, \mathbf{u}_0, h(\mathbf{x}_0, \mathbf{u}_0 \mid \boldsymbol{\lambda}_0)) + \epsilon \quad \forall t \in t_0(\epsilon) \ .$$

which asymptotically gives:

$$\lim_{t\to\infty} \inf\{\hat{f}_t(\mathbf{x}_t, \mathbf{u}_t, h(\mathbf{x}_t, \mathbf{u}_t \mid \boldsymbol{\lambda}_t))\} \geq \hat{f}(\mathbf{x}_0, \mathbf{u}_0, h(\mathbf{x}_0, \mathbf{u}_0 \mid \boldsymbol{\lambda}_0)) \ .$$

which, by definition, is equivalent to $\hat{f}_t \xrightarrow[\mathcal{X}\times\mathcal{U}]{l-prob} \hat{f}_0$. This completes the proof. □

Given the results of the above theory as well as *Remark 6.22*, the epi-convergence of LBMPC with parametric oracles can be proven.

Theorem 6.14 [18] *For* $\underset{\sim}{J}$ *with properties presented in Definition 6.30, it holds that*:

$$\wp(\tilde{\mathbf{x}}_t, \tilde{\mathbf{u}}_t \mid \boldsymbol{\lambda}_t) \xrightarrow{p} \wp(\tilde{\mathbf{x}}_t, \tilde{\mathbf{u}}_t \mid \boldsymbol{\lambda}_0) \Rightarrow \underset{\sim}{J}(\mathbf{x}_t, \mathbf{u}_t, \boldsymbol{\lambda}_t) \xrightarrow[\Upsilon(\mathbf{x}_t)]{l-prob} \underset{\sim}{J}_0(\mathbf{x}, \mathbf{u}, \boldsymbol{\lambda}_0) \ .$$

Proof. [18] From *Remark 6.22*, we know that $\underset{\sim}{J}$ is lower semi continuous, we can use the results of *Theorem 6.13*. It is expected that the solution of a sequence of optimization problems tends to converge to the solution of true optimization problem asymptotically. Also, the selected oracle is time-varying (which makes $\underset{\sim}{J}$ time-varying) and is updated as time passes so that it asymptotically converges to the oracle with true parameters. Therefore, under such conditions, we have $\mathbf{u}_t \to \mathbf{u}_0$, $\mathbf{x}_t \to \mathbf{x}_0$, and $\wp(\tilde{\mathbf{x}}_t, \tilde{\mathbf{u}}_t \mid \boldsymbol{\lambda}_t) \xrightarrow{p} \wp(\tilde{\mathbf{x}}_t, \tilde{\mathbf{u}}_t \mid \boldsymbol{\lambda}_0)$ (or equivalently $\boldsymbol{\lambda}_t \xrightarrow{p} \boldsymbol{\lambda}_0$). Given the lower semi continuity of $\underset{\sim}{J}$ at $\mathbf{u}_0$, $\mathbf{x}_0$ and $\boldsymbol{\lambda}_0$, there exist $\epsilon > 0$ and $\delta > 0$ such that:

$$\underset{\sim}{J}(\mathbf{x}_t, \mathbf{u}_t, \boldsymbol{\lambda}_t) \geq \underset{\sim}{J}_0(\mathbf{x}, \mathbf{u}, \boldsymbol{\lambda}_0) + \epsilon \quad \forall \mathbf{u} \in \mathbf{B}_r(\mathbf{u}_0), \forall \mathbf{x} \in \mathbf{B}_r(\mathbf{x}_0), \forall \boldsymbol{\lambda} \in \mathbf{B}_r(\boldsymbol{\lambda}_0) \ .$$

So, there exists a t_0, such that:

$$\underset{\sim}{J}(\mathbf{x}_t, \mathbf{u}_t, \boldsymbol{\lambda}_t) \geq \underset{\sim}{J}_0(\mathbf{x}, \mathbf{u}, \boldsymbol{\lambda}_0) + \epsilon, \quad \forall t \in t_0(\epsilon) \ ,$$

which asymptotically gives:

$$\lim_{t\to\infty} \inf\{\underset{\sim}{J}(\mathbf{x}_t, \mathbf{u}_t, \boldsymbol{\lambda}_t)\} \geq \underset{\sim}{J}_0(\mathbf{x}, \mathbf{u}, \boldsymbol{\lambda}_0) \iff \underset{\sim}{J}(\mathbf{x}_t, \mathbf{u}_t, \boldsymbol{\lambda}_t) \xrightarrow[\Upsilon(\mathbf{x}_t)]{l-prob} \underset{\sim}{J}_0(\mathbf{x}, \mathbf{u}, \boldsymbol{\lambda}_0) \ .$$

This completes the proof. □

The next step is to prove the epi-convergence for the class of non-parametric oracles. The presented theory is general and can be applied to any variant of non-parametric oracles at the heart of LBMPC. Before presenting the main result, it is required to adopt a theory regarding

the convergence in probability of the composition of functions which individually converge in probability.

Theorem 6.15 [140, 178] *Let $\mathcal{Z}_v \subset \mathbb{R}^a$, $\mathcal{Z}_w \subset \mathbb{R}^b$ and $\mathcal{R} \subset \mathbb{R}^c$ be closed and compact sets. Assume that we have a sequence of functions $V_n(z) : \mathcal{Z}_v \to \mathcal{Z}_w$ and $G_n(z) : \mathcal{Z}_w \to \mathcal{R}$ such that $\sup_{z\in\mathcal{Z}_v} \|V_t(z)-V(z)\|_2 = O_p(r_t)$ and $\sup_{z\in\mathcal{Z}_v} \|G_t(z)-G(z)\|_2 = O_p(s_t)$.* Let G be Lipschitz continuous with constant L_G. Also, assume that the range of the function on the inside of $V_n(\cdot)$ lies within the domain of convergence of the function on the outside of $G_n(\cdot)$, then $\sup_{z\in\mathcal{Z}_v} \|G_t(V_t(z))-G(V(z))\|_2 = O_p(c_t)$ where $c_t = \max\{r_t, s_t\}$.

Proof. [140] The proof of the theory can be done by using the triangular inequality and bounding the resulting probabilistic events. In this context, we have that:

$$\sup_{z\in\mathcal{Z}_v} \|G_t(V_t(z)) - G(V_t(z)) + G(V_t(z)) - G(V(z))\|_2 \ /c_t$$
$$\leq \sup_{z\in\mathcal{Z}_v} \|G_t(V_t(z)) - G(V_t(z))\|_2 \ /c_t + \sup_{z\in\mathcal{Z}_v} \|G(V_t(z)) - G(V(z))\|_2 \ /c_t \ ,$$

which implies that:

$$\mathcal{P}r(\sup_{z\in\mathcal{Z}_v} \|G_t(V_t(z)) - G(V_t(z)) + G(V_t(z)) - G(V(z))\|_2 \ /c_t \geq \epsilon) \tag{6.21}$$
$$\leq \mathcal{P}r(\sup_{z\in\mathcal{Z}_v} \|G_t(V_t(z)) - G(V_t(z))\|_2 \ /c_t \geq \epsilon) + \mathcal{P}r(\sup_{z\in\mathcal{Z}_v} \|G(V_t(z)) - G(V(z))\|_2 \ /c_t \geq \epsilon) \ .$$

Based on the assumption, the first term on the right hand side can be bounded as below:

$$\mathcal{P}r(\sup_{z\in\mathcal{Z}_v} \|G_t(V_t(z)) - G(V_t(z))\|_2 \ /c_t \geq \epsilon) \leq \mathcal{P}r(\sup_{z\in\mathcal{Z}_v} \|G_t(z) - G(z)\|_2 \ /c_t \geq \epsilon) \ .$$

Also, due to $\sup_{z\in\mathcal{Z}_v} \|G_t(z) - G(z)\|_2 = O_p(s_t)$, the limiting behavior will be:

$$\lim_{t\to\infty} \mathcal{P}r(\sup_{z\in\mathcal{Z}_v} \|G_t(V_t(z)) - G(V_t(z))\|_2 \ /c_t \geq \epsilon) \leq \lim_{t\to\infty} \mathcal{P}r(\sup_{z\in\mathcal{Z}_v} \|G_t(z) - G(z)\|_2 \ /c_t \geq \epsilon) = 0.$$

which means:

$$\lim_{t\to\infty} \mathcal{P}r(\sup_{z\in\mathcal{Z}_v} \|G_t(V_t(z)) - G(V_t(z))\|_2 \ /c_t \geq \epsilon) = 0.$$

Based on the Lipschitz continuity of G, the second term in Eq. 6.21 can be bounded as:

$$\mathcal{P}r(\sup_{z\in\mathcal{Z}_v} \|G(V_t(z)) - G(V(z))\|_2 \ /c_t \geq \epsilon) \leq \ \mathcal{P}r(\ L_G.\sup_{z\in\mathcal{Z}_v} \|V_t(z) - V(z)\|_2 \ /c_t \geq \epsilon) \ .$$

On the other hand, due to the assumption $\sup_{z\in\mathcal{Z}_v} \|V_t(z) - V(z)\|_2 = O_p(r_t)$, we have the following limiting property:

$$\lim_{t\to\infty} \mathcal{P}r(\sup_{z\in\mathcal{Z}_v} \|G(V_t(z)) - G(V(z))\|_2 \ /c_t \geq \epsilon) \leq \ \lim_{t\to\infty} \mathcal{P}r(\ L_G.\sup_{z\in\mathcal{Z}_v} \|V_t(z) - V(z)\|_2 \ /c_t \geq \epsilon) = 0 \ ,$$

which implies $\lim_{t\to\infty} \mathcal{P}r(\sup_{z\in\mathcal{Z}_v} \|G(V_t(z)) - G(V(z))\|_2 \ /c_t \geq \epsilon) = 0$. Therefore, the limit of the right hand side of Eq. 6.21 is 0, which implies $\sup_{z\in\mathcal{Z}_v} \|G_t(V_t(z)) - G(V(z))\|_2 = O_p(c_t)$ where $c_t = \max\{r_t, s_t\}$. This completes the proof. □

Now that it has been proven that the convergence in probability is preserved under the composition of functions that converge in probability, it is possible to prove the convergence of LBMPC with non-parametric oracles.

Theorem 6.16 [18] *For LBMPC with non-parametric oracle, if SE holds and* $\sup_{\mathcal{X}\times\mathcal{U}} \|\mathcal{O}_t(\tilde{\mathbf{x}}_t, \tilde{\mathbf{u}}_t) - g(\mathbf{x}_t, \mathbf{u}_t)\|_2 = O_p(r_t)$*, then, the control law converges in probability to the control law of linear MPC which knows the true model.*

Proof. [140] By means of the results of *Theorem 6.15*, and the assumptions made, as well as the results of continuous mapping presented before, it is possible to finalize the proof. Based on the recursive updating rule of the nominal model given in Eq. 6.4, that would be possible to define $\tilde{\mathbf{x}}_{t+i+1}$ as a function of $\tilde{\mathbf{x}}_t = \mathbf{x}_t$ and $\boldsymbol{\theta}_{t+i}$ (see the constraint set in Eq. 6.7). Due to the continuous mapping theorem, and the assumption of the theory, we get $\sup_{\mathbf{x}_t|\Upsilon(\mathbf{x}_t)\neq\emptyset} \|\tilde{\mathbf{x}}_{t+i}(\mathbf{x}_t, \mathcal{O}_t) - \mathbf{x}_{t+i}(\mathbf{x}_t, g)\|_2 = O_p(r_t)$ where r_t is the convergence rate, by assumption. The continuity of the cost function enables us to compose it with $\mathbf{x}_{t+i}$. Therefore, based on the same argument we have had in *Theorem 6.15*, it can be inferred that $\sup_{\mathbf{x}_t|\Upsilon(\mathbf{x}_t)\neq\emptyset} |\underset{\sim}{J} - \underset{\sim}{J_0}| = O_p(r_t)$. Due to the continuity of the cost function, and *Definition 6.12*, for a point $\mathbf{x}_0$ and the sequence of solutions $\boldsymbol{\sigma} = (\boldsymbol{\theta}_t^T, ..., \boldsymbol{\theta}_{t+N-1}^T)^T$, there exists a neighborhood region / set, let's denote it by $\aleph$, such that for all $\boldsymbol{\zeta}$ within this region, we get $|\underset{\sim}{J_0}(\boldsymbol{\zeta}) - \underset{\sim}{J_0}(\mathbf{x}_0, \boldsymbol{\sigma})| < \frac{\epsilon}{2}$. Now, it holds that:

$$\mathcal{P}r(\inf_{\boldsymbol{\zeta}\in\aleph}\{\underset{\sim}{J}(\boldsymbol{\zeta})\} < \underset{\sim}{J_0}(\mathbf{x}_0, \boldsymbol{\sigma}) - \epsilon) \leq \mathcal{P}r(\sup_{\boldsymbol{\zeta}\in\aleph} |\underset{\sim}{J}(\boldsymbol{\zeta}) - \underset{\sim}{J_0}(\mathbf{x}_0, \boldsymbol{\sigma})| > \epsilon) .$$

Since $|\underset{\sim}{J_0}(\boldsymbol{\zeta}) - \underset{\sim}{J_0}(\mathbf{x}_0, \boldsymbol{\sigma})| < \frac{\epsilon}{2}$, the equation above can be further simplified, as below:

$$\mathcal{P}r(\inf_{\boldsymbol{\zeta}\in\aleph}\{\underset{\sim}{J}(\boldsymbol{\zeta})\} < \underset{\sim}{J_0}(\mathbf{x}_0, \boldsymbol{\sigma}) - \epsilon) \leq \mathcal{P}r(\sup_{\boldsymbol{\zeta}\in\aleph} |\underset{\sim}{J}(\boldsymbol{\zeta}) - \underset{\sim}{J_0}(\boldsymbol{\zeta})| > \frac{\epsilon}{2}) .$$

Now, because of $\sup_{\mathbf{x}_t|\Upsilon(\mathbf{x}_t)\neq\emptyset} |\underset{\sim}{J} - \underset{\sim}{J_0}| = O_p(r_t)$, the limit of right hand side goes to 0, and thus, $\mathcal{P}r(\inf_{\boldsymbol{\zeta}\in\aleph}\{\underset{\sim}{J}(\boldsymbol{\zeta})\} < \underset{\sim}{J_0}(\mathbf{x}_0, \boldsymbol{\sigma}) - \epsilon) = 0$. Hence, based on *Definition 6.16*, the proof is complete. □

In above, all necessary theoretical results for the epi-convergence of the oracles of both parametric and non-parametric forms were presented, which will be used later to prove the theoretical convergence of different types of oracles for our problem.

6.5 Optimization Module

Along with the theoretical results presented previously, selection of a proper optimizer plays an important role on the efficiency of LBMPC. As mentioned before, the theoretical results are valid as long as the controlling command is feasible. The role of optimization algorithm at the heart of LBMPC is to guarantee the feasibility of the calculated command, and at the same time, to search for the actuation signals that minimize the cost functions as much as possible.

In a pioneering work on the selection of an optimization module for LBMPC [149], some variants of optimization algorithms were selected, and applied to different control problems. The results indicated that each of the considered optimization algorithms have their own pros and cons. The recommendation was that when using LBMPC for a certain problem, it would be logical to adopt a number of optimization algorithms and apply them to the problem at hand to find out which one works better. Such a consideration is indeed in line with a well-established theoretical result in the field of optimization, known as *No Free Lunch Theorem* (NFL) [179]. NFL states that a certain optimization algorithm cannot beat all of the other optimizers for all class of optimization problems, and thus, it is logical to conduct an experimental study to identify the best optimizer for a certain problem. Also, it was recommended that the best optimization algorithm is the one that not only has a good searching potential, but also can calculate the command in a very short period of time. In this context, the notion of fast MPC has attracted the attention of control engineers in recent years, and is becoming more and more interesting [180].

Given the above stated issues into account, for our case study, a number of optimization scenarios are considered, and compared with respect to the performance and computational time to find out which one works better. In a previous work on selecting an optimization algorithm for MPC to design an active suspension control system, a technique called multiplexed MPC (MMPC) were used [181]. By comparing the results obtained by MMPC and standard MPC, it was observed that MMPC can beat MPC in both computational time and performance. The idea behind the proposition of MMPC is to optimize a sub-set of decision variables at a time rather than optimizing all of the decision variables simultaneously [182]. MMPC has proven its efficient performance especially when the control problem is multivariate, as is the case for suspension control problem. As stated in [57], the philosophy of using such an optimization scenario is that "do something sooner" is more logical than "do the optimal thing later". Let's assume that, the control horizon is N and the optimization algorithm involves m actuation signals, then the complexity of the underlying optimization problem becomes $O\big((N \times m)^3\big)$, if all of the decision variables be updated simultaneously at each sampling time instance. However, for MMPC, the m control inputs are optimized successively in a cyclic manner. This means that only one input is optimized at a certain time, which decreases the computational complexity significantly [183]. Since this optimization scenario works very well for active suspension control, it is adopted and used for our case study as well. To have a preliminary vision on how multiplexed optimization algorithm

works, a schematic illustration of pattern moves for $m = 3$ ($\mathbf{u} \in \mathbb{R}^3, \mathbf{u} = (u(1), u(2), u(3))$) is given in Figure 6.7. Also, the updating of control inputs for a standard predictive control is given for the better understanding of control updating via multiplexed scenario. As can be seen, instead of a simultaneous optimization, the control inputs are updated successively. By doing so, the required time for calculating the optimal command and dispatch it to the plant decreases significantly, and at the same time a near optimal solution is expected.

Here, the general formulation of multiplexed LBMPC is presented, and it is discussed how it is going to be solved. The key point for implementing multiplexed optimization scenario is that the control input vector should be decomposed into scalar elements. In this way, the updating rule can be represented as:

$$\tilde{\mathbf{x}}_{t+1} = A\tilde{\mathbf{x}}_t + \sum_{j=1}^{m} \mathbf{b}(j)\hat{\mathbf{u}}_t(j) + \mathcal{O}_t(\tilde{\mathbf{x}}_t, \hat{\mathbf{u}}_t) \ , \tag{6.22}$$

where $\hat{\mathbf{u}}_t(j)$ is the j^{th} element of the control input $\hat{\mathbf{u}}_t = (\hat{u}_t^1, \hat{u}_t^2, ..., \hat{u}_t^m)$ at set-point t, and $\mathbf{b}(j)$ represents the j^{th} column of matrix $B = [\mathbf{b}(1)\ \mathbf{b}(2)\ \cdots\ \mathbf{b}(m)]$.

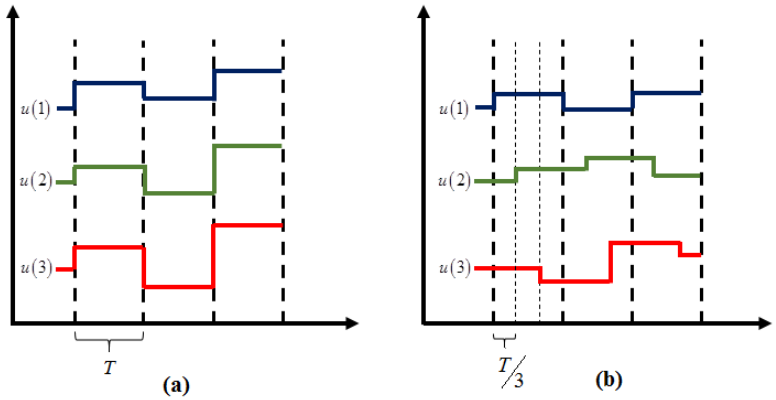

Figure 6.7: The updating rule of (a) standard MPC and (b) multiplexed MPC. As seen, the control commands are updated sooner when using multiplexed optimization scenario. Also, it can be seen that unlike parallel updating of multiple inputs in standard optimization scenario, multiplexed optimizer does the job successively.

Equation 6.22 helps us to study the impact of each control input on the dynamics of the entire system. Let's open Equation 6.22:

$$\tilde{\mathbf{x}}_{t+1} = A\tilde{\mathbf{x}}_t + \mathbf{b}(1)\hat{u}_t^1 + \mathbf{b}(2)\hat{u}_t^2 + \cdots + \mathbf{b}(m)\hat{u}_t^m + \mathcal{O}_t(\tilde{\mathbf{x}}_t, \hat{\mathbf{u}}_t) \ ,$$

then, for example, the effect of the 1^{st} actuation signal on the systems dynamics can be given as:

$$\tilde{\mathbf{x}}_{t+1} = A\tilde{\mathbf{x}}_t + \mathbf{b}(1)\hat{u}_t^1 + \mathcal{O}_t(\tilde{\mathbf{x}}_t, \hat{\mathbf{u}}_t) \ .$$

As can be inferred from Figure 6.7, the actuation signals and consequently the system states are updated successively, following the rule below:

$$\tilde{\mathbf{x}}_{t+1} = A\tilde{\mathbf{x}}_t + \mathbf{b}(\sigma_t)\hat{u}_t^{\sigma_t} + \mathcal{O}_t(\tilde{\mathbf{x}}_t, \hat{\mathbf{u}}_t) \ , \tag{6.23}$$

where $\sigma_t = (t \mod m) + 1$, is an index indicating the control channel that should be moved at time t. The considered indexing function enables us to activate all of the contributing control inputs successively. Figure 6.8 indicates the activation style of σ_t when there are 3 inputs, for 10 successive set-points.

In this way, the standard model can be represented with emphasis on specific control input at each time. Also, based on the results of *Lemma 6.2* and *Theorem 6.7*, the standard predictive objective function can be presented as:

$$J = \|\tilde{\mathbf{x}}_{t+N} - \bar{\mathbf{x}}_{\mathbf{stable}}\|_P^2 + \sum_{i=1}^{N-1} \left(\|\tilde{\mathbf{x}}_{t+i} - \bar{\mathbf{x}}_{\mathbf{stable}}\|_Q^2 + \|\hat{\mathbf{u}}_{t+i} - \bar{\mathbf{u}}_{\mathbf{stable}}\|_R^2 \right) \ .$$

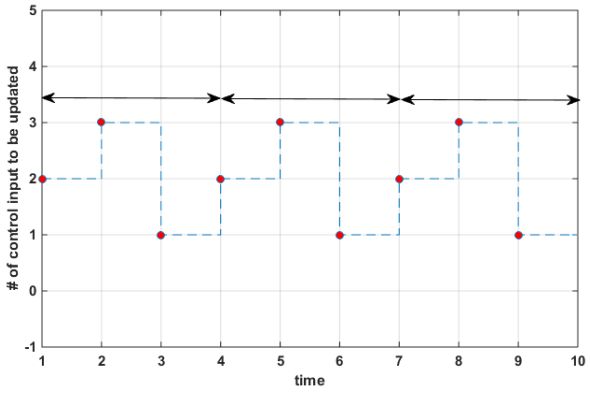

Figure 6.8: The activation style of σ_t for $m = 3$, in 10 set-points.

Obviously, some modification should be done to make the objective function appropriate to be used for multiplexed optimization. In particular, the modification should be made such that at each optimization stage, the optimization be done with respect to a certain controlling input and the other control inputs remain unchanged. Let's define $N_{new} = (N-1)m + 1$, as the new prediction horizon. The objective function can be defined as:

$$
\begin{aligned}
V_t(\mathbf{x}_t) = \min_{\Theta} \quad & \|\tilde{\mathbf{x}}_{t+N} - \bar{\mathbf{x}}_{\mathbf{stable}}\|_P^2 + \sum_{i=1}^{N_{new}-1} \|\tilde{\mathbf{x}}_{t+i} - \bar{\mathbf{x}}_{\mathbf{stable}}\|_Q^2 + \sum_{i=0}^{N_{new}-1} \|\hat{u}_{t+i}^{\sigma_{t+i}} - \bar{\mathbf{u}}_{\mathbf{stable}}(\sigma_{t+i})\|_R^2 \\
& w.r.t. \\
& \hat{u}_{t+i}, \qquad \Big(i \;=\; \sigma_t,\ \sigma_t + m,\ \sigma_t + 2m,\ \ldots\ , N_{new} - \left\lceil \frac{m}{\sigma_t} \right\rceil\Big) \\
& s.t. \\
& \begin{cases}
\tilde{\mathbf{x}}_t = \mathbf{x}_t, \quad \bar{\mathbf{x}}_t = \mathbf{x}_t \\
\tilde{\mathbf{x}}_{t+i+1} = A\tilde{\mathbf{x}}_{t+i} + B\hat{\mathbf{u}}_{t+i} + \mathcal{O}_t(\tilde{\mathbf{x}}_{t+i}, \hat{\mathbf{u}}_{t+i}) \\
\bar{\mathbf{x}}_{t+i+1} = A\bar{\mathbf{x}}_{t+i} + B\hat{\mathbf{u}}_{t+i} \\
\hat{\mathbf{u}}_{t+i} = K\bar{\mathbf{x}}_{t+i} + (\phi - K\psi)_{\sigma_{t+i}} \boldsymbol{\theta}_{t+i}^{new}(\sigma_{t+i}) \\
\bar{\mathbf{x}}_{t+i+1} \in \mathcal{X} \ominus \mathcal{R}_i, \quad \hat{u}_{t+i}^{\sigma_{t+i}} \in \mathcal{U}(\sigma_{t+i}) \ominus K(\sigma_{t+i})\mathcal{R}_i \\
(\bar{\mathbf{x}}_{t+N}; \hat{\mathbf{u}}_{t+N}) \in \Omega \ominus \Lambda_{final}
\end{cases},
\end{aligned}
\tag{6.24}
$$

where $(\phi - K\psi)_{\sigma_{t+i}}$, is the column of the resulting $m \times m$ matrix, $\mathcal{U}(\sigma_{t+i})$ corresponds to the σ_{t+i} th edge of the polytope, and $K(\sigma_{t+i})$ is the σ_{t+i} th row of matrix K. Since the controller is robust, and a feedback-based updating rule is used, it is important to note that at each updating stage $t+i$, all of the elements of control input vector $\hat{\mathbf{u}}_{t+i}$ should be updated with respect to the measured states $K\bar{\mathbf{x}}_{t+i}$, and also the destined control channel $\hat{u}_{t+i}^{\sigma_{t+i}}$ moves further using vector $\boldsymbol{\theta}_{t+i}^{new}(\sigma_{t+i})$, with all of its elements set to be 0 except the σ_{t+i} th element which is $\theta_{t+i}^{\sigma_{t+i}}$, e.g. $\boldsymbol{\theta}_{t+i}^{new}(4) = (0, 0, 0, \theta_{t+i}, ..., 0)^T$. Also, the matrix P can be obtained by:

$$
\begin{aligned}
\Big(A + \mathbf{b}(\sigma_{t+N_{new}})K(\sigma_{t+N_{new}})\Big)^T P \Big(A + \mathbf{b}(\sigma_{t+N_{new}})K(\sigma_{t+N_{new}})\Big) - P \\
= -\Big(Q + K^T(\sigma_{t+N_{new}}) r K(\sigma_{t+N_{new}})\Big)\ ,
\end{aligned}
$$

where r is the $\sigma_{t+N_{new}}$ th diagonal element of matrix R. Note that the updating rule in Equation 6.24 differs from that of Equation 6.23, since it uses the matrix B to update the states. This is because the considered control updating rule is feedback based which uses the measured states of the previous stage to update the control commands. This means that at each updating set-point all of the controlling commands are updated at-least by means of state feedback $K\bar{\mathbf{x}}_{t+i}$.

At this point, the general formulation of the objective function via multiplexed programming is finished. In the rest of this section, the detailed representation of the problem is derived which

turns Equation 6.24 to a quadratic programming (QP) problem. The N_{new}-step prediction model can be given using block-matrix representation, as below:

$$\tilde{\mathbf{X}}_t = \Pi \tilde{\mathbf{x}}_t + \Xi \hat{\mathbf{U}}_t + \mathbf{Pred} \cdot \mathbf{1}_{n\times 1} \ ,$$

where $\mathbf{1}_{n\times 1}$ is a column vector with elements 1, and:

$$\tilde{\mathbf{X}}_t = \begin{bmatrix} \tilde{\mathbf{x}}_{t+1} \\ \tilde{\mathbf{x}}_{t+2} \\ \vdots \\ \tilde{\mathbf{x}}_{t+N_{new}} \end{bmatrix} , \hat{\mathbf{U}}_t = \begin{bmatrix} \hat{\mathbf{u}}_t \\ \hat{\mathbf{u}}_{t+1} \\ \vdots \\ \hat{\mathbf{u}}_{t+N_{new}-1} \end{bmatrix} , \Pi = \begin{bmatrix} A \\ A^2 \\ \vdots \\ A^{N_{new}} \end{bmatrix} , \Xi = \begin{bmatrix} B & 0 & \dots & 0 \\ AB & B & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ A^{N_{new}-1}B & \dots & AB & B \end{bmatrix} ,$$

$$\mathbf{Pred} := \begin{bmatrix} \mathcal{O}_t(\tilde{\mathbf{x}}_t, \hat{\mathbf{u}}_t) & 0 & \dots & 0 \\ A\mathcal{O}_t(\tilde{\mathbf{x}}_t, \hat{\mathbf{u}}_t) & \mathcal{O}_t(\tilde{\mathbf{x}}_t, \hat{\mathbf{u}}_t) & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ A^{N_{new}-1}\mathcal{O}_t(\tilde{\mathbf{x}}_t, \hat{\mathbf{u}}_t) & \dots & A\mathcal{O}_t(\tilde{\mathbf{x}}_t, \hat{\mathbf{u}}_t) & \mathcal{O}_t(\tilde{\mathbf{x}}_t, \hat{\mathbf{u}}_t) \end{bmatrix} .$$

The above block-matrix presentation enables us to derive the algebraic form of the objective function:

$$J = \tilde{\mathbf{X}}_t^T \mathbf{Q} \tilde{\mathbf{X}}_t + \hat{\mathbf{U}}_t^T \mathbf{R} \hat{\mathbf{U}}_t \ ,$$

where $\mathbf{Q} = diag\{Q, Q, ..., P\}$ and $\mathbf{R} = diag\{R, R, ..., R\}$.

The remainder of the QP implementation is the verification of constraints via block-matrix representation. The equality constraints can be easily implemented using the augmented representation, as below:

$$\bar{\mathbf{X}}_t = \Pi \bar{\mathbf{x}}_t + \Xi \hat{\mathbf{U}}_t \ ,$$
$$\hat{\mathbf{U}}_t = \beth \ \bar{\mathbf{X}}_{t-1} + \Psi \mathbf{\Theta}_t^{new} \ ,$$

where

$$\beth = \begin{bmatrix} K & 0 & \dots & 0 \\ 0 & K & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & K \end{bmatrix} , \Psi = \begin{bmatrix} (\phi - K\psi)_{\sigma_t} & 0 & \dots & 0 \\ 0 & (\phi - K\psi)_{\sigma_{t+1}} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & (\phi - K\psi)_{\sigma_{t+N_{new}-1}} \end{bmatrix} ,$$

$$\mathbf{\Theta}_t^{new} = \begin{bmatrix} \boldsymbol{\theta}_t^{new}(\sigma_t) \\ \boldsymbol{\theta}_{t+1}^{new}(\sigma_{t+1}) \\ \vdots \\ \boldsymbol{\theta}_{t+N_{new}-1}^{new}(\sigma_{t+N_{new}-1}) \end{bmatrix}$$

As mentioned before, the polytopes can be represented by a number of half-spaces, as inequality constraints. Keeping this in mind, the remaining constraints (that are inequality constraints)

are implemented using block-matrix representation. So let's assume that the last three constraints are implemented as below:

$$\bar{\mathbf{x}}_{t+i+1} \in \mathcal{X} \ominus \mathcal{R}_i \Rightarrow \mathbf{g}_i^T(j)\bar{\mathbf{x}}_{t+i+1} \leq v_i(j);\ j \in \mathcal{J}\ , \tag{6.25}$$

$$\hat{u}_{t+i}^{\sigma_{t+i}} \in \mathcal{U}(\sigma_{t+i}) \ominus K(\sigma_{t+i})\mathcal{R}_i \Rightarrow \hat{u}_{t+i}^{\sigma_{t+i}} \leq l_i \ \ w.r.t.\Big(i \ = \ \sigma_t,\ \sigma_t + m,\ \sigma_t + 2m,\ \dots\ , N_{new} - \left\lceil \frac{m}{\sigma_t} \right\rceil\Big)\ , \tag{6.26}$$

$$(\bar{\mathbf{x}}_{t+N}; \hat{\mathbf{u}}_{t+N}) \in \Omega \ominus \Lambda_{final} \Rightarrow \mathbf{c}_{t+N}^T(k)(\bar{\mathbf{x}}_{t+N}; \hat{\mathbf{u}}_{t+N}) \leq f_{t+N}(k);\ k \in \mathcal{E}\ , \tag{6.27}$$

where $\mathcal{J}$ and $\mathcal{E}$ are index sets indicating the number of inequality constraints required for representing the polytopes. The next step is to turn the above formulation to the N_{new}-step prediction model using block-matrix representation. Note that the number of edges of each polytope equals the cardinality of the index sets. So, assume that $\#\mathcal{J} \equiv \hbar_{\mathcal{J}}$ and $\#\mathcal{E} \equiv \hbar_{\mathcal{E}}$. Now let's define time-varying matrix G (that is $\hbar_{\mathcal{J}} \times n$) and matrix C (that is $\hbar_{\mathcal{E}} \times n + m$), that are the row-wise concatenation of vectors $\mathbf{g}_i^T(j)$ and $\mathbf{c}_{t+N}^T(k)$, i.e. $G_i = [\mathbf{g}_i^T(1); \mathbf{g}_i^T(2); ...; \mathbf{g}_i^T(\hbar_{\mathcal{J}})]$ and $C = [\mathbf{c}_{t+N}^T(1); \mathbf{c}_{t+N}^T(2); ...; \mathbf{c}_{t+N}^T(\hbar_{\mathcal{E}})]$.

The predictive form of Equation 6.25 can be represented as:

$$\mathfrak{E}\ \bar{\mathbf{X}}_t \leq \varkappa\ ,$$

where

$$\mathfrak{E} = \begin{bmatrix} G_1 & 0 & \dots & 0 & 0 \\ 0 & G_2 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & G_{N_{new}-1} & 0 \\ 0 & 0 & \dots & 0 & 0 \end{bmatrix}\ , \varkappa = \begin{bmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \\ \vdots \\ \mathbf{v}_{N_{new}-1} \\ \mathbf{0}_{\hbar_{\mathcal{J}} \times 1} \end{bmatrix}$$

where $\mathbf{v}_i$ is a $h_{\mathcal{J}} \times 1$ vector. Also, Equation 6.27 can be represented as:

$$C\ \mathbf{z}_{t+N} \leq \mathbf{f}_{t+N}\ ,$$

where $\mathbf{f}_{t+N} = [\ f_{t+N}(1)\ f_{t+N}(2)\ ...\ f_{t+N}(h_{\mathcal{J}})]^T$, and $\mathbf{z} = (\bar{\mathbf{x}}_{t+N}; \hat{\mathbf{u}}_{t+N})$.

For constraint presented in Equation 6.26, it should be noticed that only a limited number of elements of $\hat{\mathbf{U}}_t$ with $i \ = \ \sigma_t,\ \sigma_t + m,\ \sigma_t + 2m,\ \dots\ , N_{new} - \left\lceil \frac{m}{\sigma_t} \right\rceil$ should be considered. Let's define a new vector $\hat{\mathbf{U}}_t^0$ including the elements of interest. Then, Equation 6.26 can be expressed as:

$$\hat{\mathbf{U}}_t^0 \leq \mathbf{l}\ ,$$

where $\mathbf{l} = \begin{bmatrix} l_{\sigma_t} \\ l_{\sigma_t+m} \\ \vdots \\ l_{N_{new}-\left\lceil \frac{m}{\sigma_t} \right\rceil} \end{bmatrix}$.

So, the resulting QP optimization problem can be given as:

$$V(\mathbf{x}_t) = \min_{\Theta} \ \tilde{\mathbf{X}}_t^T \mathbf{Q} \tilde{\mathbf{X}}_t + \hat{\mathbf{U}}_t^T \mathbf{R} \hat{\mathbf{U}}_t$$

$$s.t.$$

$$\begin{cases} \tilde{\mathbf{x}}_t = \mathbf{x}_t, \quad \bar{\mathbf{x}}_t = \mathbf{x}_t \\ \tilde{\mathbf{X}}_t = \Pi \tilde{\mathbf{x}}_t + \Xi \hat{\mathbf{U}}_t + \mathbf{Pred} \cdot \mathbf{1}_{n \times 1} \\ \bar{\mathbf{X}}_t = \Pi \bar{\mathbf{x}}_t + \Xi \hat{\mathbf{U}}_t \\ \hat{\mathbf{U}}_t = \beth \ \bar{\mathbf{X}}_{t-1} + \Psi \boldsymbol{\Theta}_t^{new} \\ \mathfrak{E} \ \bar{\mathbf{X}}_t \leq \varkappa \\ C \ \mathbf{z}_{t+N} \leq \mathbf{f}_{t+N} \\ \hat{\mathbf{U}}_t^0 \leq \mathbf{l} \end{cases} .$$

At this point, the general formulation of the optimization algorithm at the heart of LBMPC is presented. The solution to the above QP can be obtained by Newton's method. Note that to have a fair comparison on the performance of the proposed optimization problem, other variants of optimization scenarios are applied to the same problem. In particular, the standard optimization scenario in which all of the control input channels are optimized at the same time is considered. Also, golden sectioning search (GSS) and simulated annealing (SA) algorithms are used to solve the formulated optimization problem.

6.6 Implementation of Controller for Vehicle Suspension System

The implementation of the controller is exactly based on the descriptions given in previous sections. The remaining step is to specify the states and control inputs as well as the sources of uncertainty by means of bounded polytopes. As mentioned in Chapter 3, other than the uncertainties resulting from state measurement error and *etc.*, some other sources of uncertainties including SU-2 (number of passengers and load) and SU-3 (model-plant mismatch) are taken into account. As can be inferred, the nature of SU-2 and SU-3 are different from other sources of uncertainty, and cannot be covered by the additive polytopes in the state-space. So, the most logical choice is to consider a number of distributions for them based on the belief of the expert and the author's own findings, take sample from distributions representing SU-2 and SU-3, and then simulate the process, and report the statistical results. Having said that the detailed descriptions and quantification of the prior distributions will be given later in Chapter 8, which is dedicated to the results of vehicle suspension control. So, let's start by giving a formal representation for SU-2 and SU-3, and perform the simulation accordingly.

Recall the equivalent linear system presented in Equation 3.2 of Chapter 3. The representation enables us to formulate the time-varying linear term of state-space model. Let the states of the system be defined as:

$$\mathbf{x} = (z_{b_1}\ \dot{z}_{b_1}\ z_{b_2}\ \dot{z}_{b_2}\ z_{u_1}\ \dot{z}_{u_1}\ z_{u_2}\ \dot{z}_{u_2}\ )^T\ .$$

So, $\mathbf{x} \in \mathbb{R}^8$. Also, the half-car vehicle suspension model has 2 actuators $\mathbf{u} = \begin{bmatrix} F_{a_1} \\ F_{a_2} \end{bmatrix}$, i.e. $\mathbf{u} \in \mathbb{R}^2$.

As we know, the model formulated based on vehicles dynamics is continuous ordinary differential equation (ODE), and thus, it should be discretized to be used at the heart of LBMPC which is a discrete controller. There are so many ways for verifying the differenced form of ODE. For the current simulations, the first order difference approximation is considered:

$$\dot{\mathbf{a}}(t) \simeq \frac{\mathbf{a}(t+1) - \mathbf{a}(t)}{\Delta t}\ ,$$

where Δt is the time difference between two sequential set-points, and by considering a small value, the approximation error will be very trivial. For our simulation, $\Delta t = 10^{-4}$ *sec* is considered. So, the rest of the formulation and details are given keeping in mind that the ODE is transformed to the equivalent difference form. For more details on this issue, the interested readers may refer to [58].

Now the linear part of the nominal model can be formulated using Equation 3.2 and vector $\mathbf{x}$ defined above, as below:

$$\bar{\mathbf{x}}_{t+1} = \hat{A}\bar{\mathbf{x}}_t + \hat{B}\bar{\mathbf{u}}_t\ ,$$

where $\hat{A}$ and $\hat{B}$ are uncertain linear maps used in the model. The quantification of the uncertainties of $\hat{A}$ and $\hat{B}$ depends on so many physical and environmental factors, and should be done by means of a prior belief as well as expert's knowledge. For now, let's consider the general representation of distribution for $\hat{A}$ and $\hat{B}$, , i.e. $\hat{A} \overset{d}{\sim} (A_{ave}, \Sigma_A)$ and $\hat{B} \overset{d}{\sim} (B_{ave}, \Sigma_B)$. Apparently, A_{ave} and B_{ave} can be determined from the underlying physics of the model:

$$A_{ave} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -\zeta k^*_{s_1} & -\zeta c^*_{s_1} & -\mu k^*_{s_2} & -\mu c^*_{s_2} & \zeta k^*_{s_1} & \zeta c^*_{s_1} & \mu k^*_{s_2} & \mu c^*_{s_2} \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ -\mu k^*_{s_1} & -\mu c^*_{s_1} & -\gamma k^*_{s_2} & -\gamma c^*_{s_2} & \mu k^*_{s_1} & \mu c^*_{s_1} & \gamma k^*_{s_2} & \gamma c^*_{s_2} \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ k^*_{s_1}/m_{u_1} & c^*_{s_1}/m_{u_1} & 0 & 0 & (-k_{t_1} - k^*_{s_1})/m_{u_1} & (-c_{t_1} - c^*_{s_1})/m_{u_1} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & k^*_{s_2}/m_{u_2} & c^*_{s_2}/m_{u_2} & 0 & 0 & (-k_{t_2} - k^*_{s_2})/m_{u_2} & (-c_{t_2} - c^*_{s_2})/m_{u_2} \end{bmatrix}$$

$$B_{ave} = \begin{bmatrix} 0 & -\zeta & 0 & -\mu & 0 & 1/m_{u_1} & 0 & 1 \\ 0 & -\mu & 0 & -\gamma & 0 & 0 & 0 & 1/m_{u_2} \end{bmatrix}^T .$$

The main terms defining the uncertainties are Σ_A and Σ_B. Later on, these matrixes will be quantified by means of a Bayesian argument using expert's knowledge.

Also, as discussed in Chapter 4, unlike the common belief of automotive engineers, the road roughness does not necessarily follow a white noise distribution. Therefore, we carried out an experimental analysis and estimated this variable using statistical forecasting tools. Actually, this increases the authenticity of our state-space representation, since the road roughness is estimated based on real-data rather than allocating a bounded polytope to simulate its effect on states. Therefore, fixed part of the model is updated as

$$\bar{\mathbf{x}}_{t+1} = \hat{A}\bar{\mathbf{x}}_t + \hat{B}\bar{\mathbf{u}}_t + D\mathbf{r}_t \ ,$$

where $\mathbf{r} = [r_1 \ \dot{r}_1 \ r_2 \ \dot{r}_2]^T$ and

$$D = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & k_{t_1}/m_{u_1} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & c_{t_1}/m_{u_1} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & k_{t_2}/m_{u_2} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & c_{t_2}/m_{u_2} \end{bmatrix}^T .$$

In the above representation, r_1 and r_2 are the deflections of the front and rear tires, and $\dot{r}_1$ and $\dot{r}_2$ are the rate of the deflection of the front and rear tires, respectively.

Indeed, $\mathbf{r}_t$ is the forecasted roughness obtained from statistical model. Keep in mind that, as discussed (see Equation 6.2), *as long as the data-driven statistical model of road-roughness be bounded*, we can include it in a polytope, say $\mathcal{D}$. It is logical to consider the other unknown sources of additive uncertainties and represent them by means of bounded polytope $\mathcal{W}$. According to the argument we have had for formulating Equation 6.2, the new polytope will be $\mathcal{W}^{new} = \mathcal{W} \oplus \mathcal{D}$. This assures that all of the theoretical results presented before are valid. Now let's formulate the final form of the nominal model with road roughness forecasting module and unknown disturbance:

$$\bar{\mathbf{x}}_{t+1} = \hat{A}\bar{\mathbf{x}}_t + \hat{B}\bar{\mathbf{u}}_t + D\mathbf{r}_t + \mathbf{d}_t \ ,$$

Apparently, the learnable state-space can be presented as:

$$\tilde{\mathbf{x}}_{t+1} = \hat{A}\tilde{\mathbf{x}}_t + \hat{B}\tilde{\mathbf{u}}_t + D\mathbf{r}_t + \mathcal{O}_t(\tilde{\mathbf{x}}_t, \tilde{\mathbf{u}}_t) \ ,$$

Note that the quantifying of $\mathcal{W}$ is based on the sources of uncertainty defined in SU-5, i.e. the uncertainties due to the rapture of sensors and internal components of vehicles.

At this point, the detailed description regarding the theoretical and structural properties of LBMPC is completed, and the details on how to use it for suspension control will be given in the the coming chapters.

Chapter 7

Development and Evaluation of Statistical and Soft Oracles

In this chapter, a comparative study is carried out to develop an appropriate oracle for learning-based model predictive control (LBMPC). Also, the polytope $\mathcal{W}$ entailing the unmeasured disturbance (defined in Chapter 6) is quantified with the aid of the sources of uncertainty within the category SU-5, i.e. the uncertainties due to the malfunction of sensors as well as the internal components of vehicles. Note that considering $\mathcal{W}$ as an additive term in the state-space makes sound from physical viewpoint, since the malfunction of sensors usually appears as measurement noise and affects the measured values of the internal states of vehicles.

The rest of the chapter is organized as follows. Firstly, a critical discussion is made regarding the applicability and potential of statistical and soft computing models for designing oracles. Thereafter, the mathematical formulation of the considered models from both statistical computing and soft computing domains are presented. Then, the experiments pertaining to the quantifying of $\mathcal{W}$ is given, and based on that, the considered models are used to develop an efficient oracle for LBMPC. Finally, some recommendations regarding the design of oracles are given.

7.1 Statistical Models vs. Soft Computing Models

Given the applicability of both statistical and soft computing methods for modeling under uncertainty and imprecision, and also given the promising reports on their usage for suspension control (see Chapter 2), here, both of the concepts are considered for designing

oracles to be used at the heart of suspension systems state-space model. Before going into the mathematical details of the considered models, a critical discussion is given and a fair comparison between those two concepts is carried out. In the author's view, both of the methods have their own pros and cons, and each of them could be appropriate for certain classes of problems.

Usually, when using statistical machine learning tools, there is some logic behind each stage of training process, as hypothesis testing and theoretical analysis is made to ensure one follows a meaningful training path. To comply with such an objective, there are a number of pre-requisites which should be provided. Firstly, the available data should be precisely analyzed to get some clues about selecting the proper classes of statistical modeling tools. Also, it is necessary to adopt theoretically well-established model diagnosis and hypothesis testing tools to ensure the trained model is reliable for treating the available dataset. One can claim that without having a descent background of statistical mathematics, it would be very hard for one to use statistical machine learning tools for the proper treating of a given dataset.

If all of the considerations above be satisfied and a fine analysis (at all model selection, parameter estimation and model diagnosis stages) be conducted, one can realize that statistical methods are best suited for inference, and can yield fruitful information regarding the properties of the dataset at hand.

Soft computing methods follow an entirely different philosophy for dealing with datasets. Soft computing has been developed by engineering society fostering the concept that by using complex structural models, it would be possible for anyone to come up with an approximation of any function. As discussed in Chapter 2, the structure of such methods is taken fixed beforehand (which is usually a complicated black-box structure) for any type of datasets. In the case that some of the variants of soft computing methods include model selection or hyper-parameters, those parameters are usually tuned with either a trial and error procedure or a heuristic search, such as genetic algorithm (GA), and *etc.* [184]. Also, practitioners usually give less attention to data analysis to find out which class of soft models better match the available dataset. This is because most of such methods are black-boxes with universal approximation capability, and can be used for any dataset. Moreover, after estimation, the model diagnosis is simply done by reporting the mean error, and *std.* values, and other important aspects are usually left untouched. Therefore, it is usually impossible to justify why a certain soft computing method, e.g. artificial neuro-fuzzy inference system or artificial neural network, is used for a given function approximation problem.

On the other hand, because of the convenience of using such methods, and also their

good point estimation results (thanks to their complex structures and universal approximation capability), they are frequently used by engineers and researchers, especially for engineering problems [185]. It is worth noting that because of such convenient implementation suit, and taking less energy from users to get the results, soft computing methods are also called computational intelligence machines, with the belief that they are smart enough to learn in a self-adaptive fashion and yield a (usually) nonlinear map between a set of inputs and outputs.

All in all, it can be stated that statistical machine learning mostly reclines on the knowledge of the user, and the effort is to make the model simple yet efficient, which can be later used for inference and analysis. On the other hand, intelligent machine learning focuses on using complicated black-box structures with the capability of universal approximation so that anyone can apply them for regression and classification tasks. Such methods are neither proper for inference nor for extracting knowledge from the model, but can be viewed as very good point approximation tools, and have successfully come to the aid of engineers over the past two decades.

7.2 Notations and Preliminaries

To be consistent and avoid possible confusion, some notations should be clarified before formulating the considered methods.

7.2.1 General notations

Throughout the chapter, vectors are shown by lower case bold fonts ($\mathbf{a}, \mathbf{b}$, and *etc.*), sets are shown by calligraphic upper case italic format ($\mathcal{A}$, $\mathcal{B}$, and *etc.*), and matrixes are given by upper case italic format (A, B, and *etc.*). The vectors are presented in columns, i.e. $\mathbf{a} = col(a_1, a_2, ..., a_m)$ where $\mathbf{a} \in \mathbb{R}^m$ (unless something else be mentioned). A^T is the trnspose of A. The Euclidian (L_2) norm is shown by $\|\cdot\|_2$. The inner product between two vectors $\mathbf{a}$ and $\mathbf{b}$ is shown by $\langle \mathbf{a}, \mathbf{b} \rangle$. A vector of one's of appropriate size (e.g. $n \times 1$) is shown by $\mathbf{1}_{n\times 1}$. An identity matrix of appropriate size (e.g. $n \times n$) is shown by $\mathbb{I}_n$. The notation $\big(\mathbf{a}; \mathbf{b}\big)$ concatenates the two vectors $\mathbf{a}$ and $\mathbf{b}$, and $[A\ B]$ concatenates two matrixes A and B with the same number of rows, and possibly different number of columns.

7.2.2 Representation of a dataset

For a given dataset which can be presented by $n \times p + q$ matrix S, n shows the number of available data pairs in the dataset, p is the number of independent variables (inputs), and q is the number of dependent variables (outputs). Mathematically speaking, $S = [\ \mathbf{s}_1^T; \mathbf{s}_2^T; \cdot \cdot \cdot; \mathbf{s}_n^T\ ]$, where $\mathbf{s}_i = \big(\mathbf{z}_i; \mathbf{f}_i(\mathbf{z}_i)\big)$, $\mathbf{z}_i = (z_{1,i}, z_{2,i}, ..., z_{p,i})$, and $\mathbf{f}_i(\mathbf{z}_i) = \big(f_{1,i}(\mathbf{z}_i), f_{2,i}(\mathbf{z}_i), ..., f_{q,i}(\mathbf{z}_i)\big)$. In this presentation, $\mathbf{z}$ is the p-variate input vector and $\mathbf{f}(\mathbf{z})$ is the q-variate output vector. The goal of using statistical and soft tools is to find an approximation $\hat{\mathbf{f}}(\mathbf{z})$ based on the available dataset S and an appropriate distance measure, which can be used later as an approximation of the true output vector for unseen p-variate data $\mathbf{z}_0$, such that:

$$\mathbf{f}(\mathbf{z}_0) = \hat{\mathbf{f}}(\mathbf{z}_0) \ + \ \boldsymbol{\epsilon}_0 \ ,$$

where $\boldsymbol{\epsilon}_0 = (\epsilon_{1,0}, \epsilon_{2,0}, ..., \epsilon_{q,0})$, and ϵ is a Gaussian noise with $E[\epsilon] = 0$. Note that $E[\cdot]$ is the expectation operator.

7.3 Statistical Oracles

In this section, the mathematical formulation of the adopted statistical machine learning methods are given, which will be used later for developing oracles for LBMPC. Based on the discussion we had in Chapter 6, non-parametric statistical methods are chosen, as they give more structural flexibility to oracles, which can be useful especially for real-world applications. Also, recall that to satisfy the theoretical convergence and stability criteria of LBMPC, it is necessary to select continuous and differentiable models.

7.3.1 Kriging

Kriging or Wiener-Kolmogorov predictor is a type of Gaussian process regression which is used for interpolation based on the prior covariance matrix of observed data [186]. Indeed, Kriging interpolates the unseen values based on their distance from observed data in training dataset (by linear combination of data points). Let's say n data are available, and based on that, we would like to estimate $f(\mathbf{z}_0)$ for a data point located in $\mathbf{z}_0$. Then,

the estimated value is given by:

$$\hat{f}(\mathbf{z}_0) = \boldsymbol{\omega}^T . \begin{bmatrix} f(\mathbf{z}_1) \\ f(\mathbf{z}_2) \\ \vdots \\ f(\mathbf{z}_n) \end{bmatrix} ,$$

where $\boldsymbol{\omega} = (\omega_1, \omega_2, ..., \omega_n)^T$ is a $n \times 1$ vector including combination weights.

There are different variants of Kriging methods that can be used for interpolation, from which ordinary Kriging is the most applicable one. Ordinary Kriging assumes a constant unknown mean (μ) only over the search neighborhood of a given point $\mathbf{z}_0$. The structure of Kriging includes an optimization procedure for finding the optimum combination weights $\boldsymbol{\omega}$ while satisfying two important criteria: (1) lack of bias which implies that the mean of estimated value should be equal to the mean of real value, i.e. $E\big[f(\mathbf{z}_0)\big] = E\big[\hat{f}(\mathbf{z}_0)\big] = \mu$, and (2) minimum variance which implies that the mean of squared deviations should be minimal. Obviously, in the case that a group of estimated values are more disperse than the corresponding real observed values, the estimation is not precise. Let's formulate the estimation error as:

$$\epsilon(\mathbf{z}_0) = \hat{f}(\mathbf{z}_0) - f(\mathbf{z}_0) = [\boldsymbol{\omega}^T - 1] \; . \begin{bmatrix} f(\mathbf{z}_1) \\ f(\mathbf{z}_2) \\ \vdots \\ f(\mathbf{z}_n) \end{bmatrix} = \sum_{i=1}^{n} \omega_i f(\mathbf{z}_i) - f(\mathbf{z}_0) \; ,$$

then, the lack of bias criterion, i.e. $E\big[f(\mathbf{z}_0)\big] = E\big[\hat{f}(\mathbf{z}_0)\big] = \mu$, yields:

$$E\big[\epsilon(\mathbf{z}_0)\big] = 0 \iff \sum_{i=1}^{n} \omega_i E\big[f(\mathbf{z}_i)\big] - E\big[f(\mathbf{z}_0)\big] = 0 \iff \mu \sum_{i=1}^{n} \omega_i - \mu = 0 \iff \mathbf{1}_{n\times 1}^T . \boldsymbol{\omega} = 1 \; .$$

The above condition means that the combination weights should sum up to 1. Also, as mentioned, the minimum variance criterion minimizes:

$$\mathrm{Var}\big[\epsilon(\mathbf{z}_0)\big] = \mathrm{Var}\left([\boldsymbol{\omega}^T - 1] \; . \begin{bmatrix} f(\mathbf{z}_1) \\ f(\mathbf{z}_2) \\ \vdots \\ f(\mathbf{z}_n) \end{bmatrix} \right) = [\boldsymbol{\omega}^T - 1] \; . \; \mathrm{Var}\left(\begin{bmatrix} f(\mathbf{z}_1) \\ f(\mathbf{z}_2) \\ \vdots \\ f(\mathbf{z}_n) \end{bmatrix} \right) \; . \; \begin{bmatrix} \boldsymbol{\omega} \\ -1 \end{bmatrix}$$

$$= [\boldsymbol{\omega}^T - 1] \; . \; \begin{bmatrix} \mathrm{Var}_{\mathbf{z}_i} & \mathrm{Cov}_{\mathbf{z}_i,\mathbf{z}_0} \\ \mathrm{Cov}^T_{\mathbf{z}_i,\mathbf{z}_0} & \mathrm{Var}_{\mathbf{z}_0} \end{bmatrix} \; . \; \begin{bmatrix} \boldsymbol{\omega} \\ -1 \end{bmatrix} \; ,$$

where

$$\mathrm{Var}_{\mathbf{z}_i} \equiv \mathrm{Var}\left(\begin{bmatrix} f(\mathbf{z}_1) \\ f(\mathbf{z}_2) \\ \vdots \\ f(\mathbf{z}_n) \end{bmatrix} \right) \; , \; \mathrm{Var}_{\mathbf{z}_0} \equiv \mathrm{Var}\big[f(\mathbf{z}_0)\big] \; , \text{ and } \mathrm{Cov}_{\mathbf{z}_i,\mathbf{z}_0} \equiv \left(\begin{bmatrix} f(\mathbf{z}_1) \\ f(\mathbf{z}_2) \\ \vdots \\ f(\mathbf{z}_n) \end{bmatrix} , f(\mathbf{z}_0) \right) \; .$$

Apparently, $\mathrm{Var}_{\mathbf{z}_i}$ is a $n \times n$ matrix, $\mathrm{Var}_{\mathbf{z}_0}$ is a number, and $\mathrm{Cov}_{\mathbf{z}_i,\mathbf{z}_0}$ is a $n \times 1$ vector. By some algebraic calculations, $\mathrm{Var}\big[\epsilon(\mathbf{z}_0)\big]$ can be represented as:

$$\mathrm{Var}\big[\epsilon(\mathbf{z}_0)\big] = \boldsymbol{\omega}^T \; . \; \mathrm{Var}_{\mathbf{z}_i} \; . \; \boldsymbol{\omega} - \mathrm{Cov}^T_{\mathbf{z}_i,\mathbf{z}_0} . \; \boldsymbol{\omega} - \boldsymbol{\omega}^T . \; \mathrm{Cov}_{\mathbf{z}_i,\mathbf{z}_0} + \mathrm{Var}_{\mathbf{z}_0} \; .$$

Now, the optimization problem can be formulated as:

$$V(\boldsymbol{\omega}) = \min_{\boldsymbol{\omega}} \quad \boldsymbol{\omega}^T \; . \; \mathrm{Var}_{\mathbf{z}_i} \; . \; \boldsymbol{\omega} - \mathrm{Cov}^T_{\mathbf{z}_i,\mathbf{z}_0} . \; \boldsymbol{\omega} - \boldsymbol{\omega}^T . \; \mathrm{Cov}_{\mathbf{z}_i,\mathbf{z}_0} + \mathrm{Var}_{\mathbf{z}_0}$$

$$s.t.$$

$$\mathbf{1}^T_{n\times 1} . \boldsymbol{\omega} = 1 \; ,$$

where $V(\cdot)$ is the value function. By making use of Lagrangian approach, the solution to the optimization above can be given as:

$$\begin{bmatrix} \hat{\boldsymbol{\omega}} \\ \lambda \end{bmatrix} = \begin{bmatrix} \mathrm{Var}_{\mathbf{z}_i} & \mathbf{1}_{n\times 1} \\ \mathbf{1}^T_{n\times 1} & 0 \end{bmatrix} \begin{bmatrix} \mathrm{Cov}_{\mathbf{z}_i,\mathbf{z}_0} \\ 1 \end{bmatrix} = \begin{bmatrix} \gamma\big(f(\mathbf{z}_1), f(\mathbf{z}_1)\big) & \cdots & \gamma\big(f(\mathbf{z}_1), f(\mathbf{z}_n)\big) & 1 \\ \vdots & \ddots & \vdots & \vdots \\ \gamma\big(f(\mathbf{z}_n), f(\mathbf{z}_1)\big) & \cdots & \gamma\big(f(\mathbf{z}_n), f(\mathbf{z}_n)\big) & 1 \\ 1 & \cdots & 1 & 0 \end{bmatrix}^{-1} \begin{bmatrix} \gamma\big(f(\mathbf{z}_1), f(\mathbf{z}_0)\big) \\ \vdots \\ \gamma\big(f(\mathbf{z}_n), f(\mathbf{z}_0)\big) \\ 1 \end{bmatrix} ,$$

where λ is the Lagrange multiplier, and:

$$\gamma\big(f(\mathbf{z}_i), f(\mathbf{z}_j)\big) = E[f(\mathbf{z}_i).f(\mathbf{z}_j)] - E[f(\mathbf{z}_i)] \; . \; E[f(\mathbf{z}_j)] \; .$$

The above procedure can be easily extended to the multi-output version of ordinary Kriging model, i.e. $\mathbf{f}_0\big(\mathbf{z}_0\big) = \big(f_{1,0}(\mathbf{z}_0), f_{2,0}(\mathbf{z}_0), ..., f_{q,0}(\mathbf{z}_0)\big)^T$. In this case, all of the information above remain the same, and we just need to estimate distinct linear combination vectors $\boldsymbol{\omega}_i$ for each of the outputs $i = 1, ..., q$. This means that q optimization problems are formulated to get the $q \times n$ matrix $W = [\hat{\boldsymbol{\omega}}^T_1; \hat{\boldsymbol{\omega}}^T_2; ...; \hat{\boldsymbol{\omega}}^T_q]$, and estimate $\hat{\mathbf{f}}_0(\mathbf{z}_0) = \big(\hat{f}_{1,0}(\mathbf{z}_0), \hat{f}_{2,0}(\mathbf{z}_0), ..., \hat{f}_{q,0}(\mathbf{z}_0)\big)^T$, as:

$$\hat{f}_{j,0}(\mathbf{z}_0) = \hat{\boldsymbol{\omega}}_j^T . \begin{bmatrix} f_{j,1}(\mathbf{z}_1) \\ f_{j,2}(\mathbf{z}_2) \\ \vdots \\ f_{j,n}(\mathbf{z}_n) \end{bmatrix} \quad , \quad j = 1, ..., q.$$

7.3.2 Mixture of experts with expectation maximization

Mixture of experts (MoE) models are powerful statistical tools that can be used for variety of applications such as clustering, classification and regression [187]. Let's denote $f(\mathbf{z})$ by y. As we know, regression studies the relation between y and covariate vector $\mathbf{z}$ by means of a conditional density function, say $g(y \mid \mathbf{z}; \boldsymbol{\Psi})$, where $\boldsymbol{\Psi}$ is the parameter vector of density function. The idea of using MoE is to decompose the conditional density function $g(y \mid \mathbf{z}; \boldsymbol{\Psi})$ into a convex weighted sum of k regression components, as below:

$$g(y_i \mid \mathbf{z}_i; \boldsymbol{\Psi}) = \sum_{h=1}^{k} \pi_h(\mathbf{z}_i; \boldsymbol{\alpha}_h) \; g_h(y_i \mid \mathbf{z}_i; \boldsymbol{\Psi}_h),$$

where $\pi_h(\mathbf{z}_i; \boldsymbol{\alpha}_h)$ is called the gate function which represents the mixing proportion for h^{th} component, and satisfies the following criteria:

$$\begin{cases} \pi_h(\mathbf{z}; \boldsymbol{\alpha}_h) > 0 \; , \; \forall h \\ \sum\limits_{h=1}^{k} \pi_h(\mathbf{z}; \boldsymbol{\alpha}_h) = 1 \end{cases} .$$

One appropriate choice of experts $g_h(y_i \mid \mathbf{z}_i; \boldsymbol{\Psi}_h)$ for regression is the use of Gaussian distribution:

$$g(y_i \mid \mathbf{z}_i; \boldsymbol{\Psi}) = \sum_{h=1}^{k} \pi_h(\mathbf{z}_i; \boldsymbol{\alpha}_h) \; N_h(y_i \mid \mu(\mathbf{z}_i; \boldsymbol{\beta}_h), \sigma_h^2),$$

where $\boldsymbol{\beta}_h$ is a $p \times 1$ vector, $\mu(\mathbf{z}_i; \boldsymbol{\beta}_h)$ is the component mean and σ_h^2 is the component variance. For the current study, the component means $\mu(\mathbf{z}_i; \boldsymbol{\beta}_h)$ are defined as the projection of $\mathbf{z}_i$ on $\boldsymbol{\beta}_h$, i.e. $\mu(\mathbf{z}_i; \boldsymbol{\beta}_h) = \boldsymbol{\beta}_h^T \mathbf{z}_i$.

MoE model is trained using expectation maximization (EM) algorithm. To use EM, it is necessary to define indicator variables v_{hi}, where:

$$v_{hi} = \begin{cases} 1, & \text{if } y_i \text{ belongs to } h^{th}\text{component} \\ 0, & \text{otherwise} \end{cases} .$$

Thus, the missing data $\mathbf{v}$ can be presented as a vector including all of the indicator variables. Now, it would be possible to define the gate function as the probability that v_{hi} is one, which can be mathematically expressed as:

$$\pi_h(\mathbf{z}_i; \boldsymbol{\alpha}_h) = \mathcal{P}r(v_{hi} = 1 \mid \mathbf{z}_i) \ .$$

The log-likelihood of the complete data $(\mathbf{z}; \mathbf{v})$ can be given as:

$$\log L_C(\boldsymbol{\Psi}) \equiv \log L(\boldsymbol{\Psi} \mid \mathbf{z}, \mathbf{v}) = \sum_{i=1}^{n} \sum_{h=1}^{k} v_{hi} \big(\log \pi_h(\mathbf{z}_i; \boldsymbol{\alpha}_h) + \log N_h(y_i \mid \mu(\mathbf{z}_i; \boldsymbol{\beta}_h), \sigma_h^2) \big) \ .$$

The E-step includes determining the conditional expectation of the log-likelihood of complete data given observed data formed by $\boldsymbol{\Psi}^{(t)}$, which is known as Q-function and can be given as:

$$\begin{aligned} Q(\boldsymbol{\Psi}, \boldsymbol{\Psi}^{(t)}) &= E_{\boldsymbol{\Psi}^{(t)}}[\log L_C(\boldsymbol{\Psi}) \mid \mathbf{z}, y] \\ &= \sum_{i=1}^{n} \sum_{h=1}^{k} E_{\boldsymbol{\Psi}^{(t)}}[v_{hi} \mid \mathbf{z}_i, y_i] \big(\log \pi_h(\mathbf{z}_i; \boldsymbol{\alpha}_h) + \log N_h(y_i \mid \mu(\mathbf{z}_i; \boldsymbol{\beta}_h), \sigma_h^2) \big) \ . \end{aligned}$$

Note that $E_{\boldsymbol{\Psi}^{(t)}}[\cdot]$ only operates on v_{hi}, since $\log L_C(\boldsymbol{\Psi})$ is linear in v_{hi}. Let's define:

$$\tau_{hi}^{(t)} = \tau_{hi}(\boldsymbol{\Psi}^{(t)}; \mathbf{z}_i, y_i) := E_{\boldsymbol{\Psi}^{(t)}}[v_{hi} \mid \mathbf{z}_i, y_i] \ .$$

Then the posterior probability $\tau_{hi}(\boldsymbol{\Psi}; \mathbf{z}_i, y_i)$ is given by:

$$\tau_{hi}(\boldsymbol{\Psi}; \mathbf{z}_i, y_i) = \mathcal{P}r(v_{hi} = 1 \mid \mathbf{z}_i, y_i) = \frac{\pi_h(\mathbf{z}_i; \boldsymbol{\alpha}_h) \ N_h(y_i \mid \mu(\mathbf{z}_i; \boldsymbol{\beta}_h), \sigma_h^2)}{\sum_{l=1}^{k-1} \pi_l(\mathbf{z}_i; \boldsymbol{\alpha}_l) \ N_l(y_i \mid \mu(\mathbf{z}_i; \boldsymbol{\beta}_l), \sigma_l^2)} \ ,$$

for $h = 1, 2, ..., k-1$. Now, the M-step updates the parameter vector $\boldsymbol{\Psi}^{(t+1)}$ by measuring the Q-function, as below:

$$\boldsymbol{\Psi}^{(t+1)} = \arg\max_{\boldsymbol{\Psi} \in \Omega} Q(\boldsymbol{\Psi}, \boldsymbol{\Psi}^{(t)}) \ .$$

So, the updated estimate for the parameters $\boldsymbol{\alpha}_h^{(t+1)}$, $\boldsymbol{\beta}_h^{(t+1)}$, and $\sigma_h^{2\ (t+1)}$ can be respectively given as:

$$\sum_{i=1}^{n} \tau_{hi}^{(t)} \frac{\partial \log \pi_h(\mathbf{z}_i; \boldsymbol{\alpha}_h)}{\partial \boldsymbol{\alpha}_h} = 0 \ , \ \ h = 1, ..., k-1, \tag{7.1}$$

$$\sum_{i=1}^{n} \tau_{hi}^{(t)} \frac{\partial \log N_h(y_i \mid \mu(\mathbf{z}_i;\boldsymbol{\beta}_h),\sigma_h^2)}{\partial \boldsymbol{\beta}_h} = 0 \ , \ h = 1, ..., k-1,$$
$$\sum_{i=1}^{n} \tau_{hi}^{(t)} \frac{\partial \log N_h(y_i \mid \mu(\mathbf{z}_i;\boldsymbol{\beta}_h),\sigma_h^2)}{\partial \sigma_h^2} = 0 \ , \ h = 1, ..., k-1 \ .$$

Fortunately, for Gaussian regression components with $\mu(\mathbf{z}_i;\boldsymbol{\beta}_h) = \boldsymbol{\beta}_h^T \mathbf{z}_i$, one can analytically update the parameters $\boldsymbol{\beta}_h^{(t+1)}$, and $\sigma_h^{2\ (t+1)}$, by deriving score and information functions. The closed-form solution can be given as:

$$\boldsymbol{\beta}_h^{(t+1)} = \frac{\sum_{i=1}^{n} \tau_{hi}^{(t)} y_i \mathbf{z}_i}{\sum_{i=1}^{n} \tau_{hi}^{(t)} \mathbf{z}_i^T \mathbf{z}_i} \quad ; \quad \sigma_h^{2\ (t+1)} = \frac{\sum_{i=1}^{n} \tau_{hi}^{(t)} (y_i - \boldsymbol{\beta}_h^{T\ (t+1)} \mathbf{z}_i)^2}{\sum_{i=1}^{n} \tau_{hi}^{(t)}}$$

where $h = 1, ..., k$. For updating $\boldsymbol{\alpha}_h^{(t+1)}$, there is no closed-form solution and it is necessary to use iterative methods. This can be done by using iterative reweighted least square (IRLS) algorithm [188].

The gating function can be modelled by multinomial logit function, as below:

$$\pi_h(\mathbf{z}_i;\boldsymbol{\alpha}_h) = \frac{\exp(\boldsymbol{\alpha}_h^T \mathbf{z}_i)}{1 + \sum_{l=1}^{k-1} \exp(\boldsymbol{\alpha}_l^T \mathbf{z}_i)} \ , \quad h = 1, ..., k-1 \ ,$$

where $\pi_k(\mathbf{z}_i;\boldsymbol{\alpha}_k) = \dfrac{1}{1 + \sum_{l=1}^{k-1} \exp(\boldsymbol{\alpha}_l^T \mathbf{z}_i)}$, which is not a function of $\boldsymbol{\alpha}_k$. This means that $\boldsymbol{\alpha}_k$ is a null-vector. So, Eq. 7.1 can be represented as:

$$\sum_{i=1}^{n} \left(\tau_{hi}^{(t)} - \frac{\exp(\boldsymbol{\alpha}_h^T \mathbf{z}_i)}{1 + \sum_{l=1}^{k-1} \exp(\boldsymbol{\alpha}_l^T \mathbf{z}_i)} \right) \mathbf{z}_i = 0 \ , \ h = 1, ..., k-1.$$

This results in a system of equations with $(k-1)p$ unknown parameters, and is solved using IRLS, as below:

$$\boldsymbol{\alpha}_h^{(t+1)} = \boldsymbol{\alpha}_h^{(t)} + \xi \left(\frac{\partial^2 Q}{\partial^2 \boldsymbol{\alpha}_h^T \ ^{(t)} \boldsymbol{\alpha}_h^{(t)}} \right)^{-1} \frac{\partial Q}{\partial \boldsymbol{\alpha}_h^{(t)}} \ , \quad h = 1, ..., k-1 \ ,$$

where $\xi \leq 1$ is the learning rate. Just like any other numerical optimization algorithm, the above iterative process terminates if the convergence happens or the algorithm reaches the maximum number of iteration numbers.

Obviously, the multi-output version of MoE can be implemented by using multivariate (q-variate) normal components, vector gating functions (with q outputs), and the same training strategy used for single output version. However, due to the assumed independency of the output elements $\mathbf{f}(\mathbf{z}) = \big(f_1(\mathbf{z}), f_2(\mathbf{z}), ..., f_q(\mathbf{z})\big)$, the training process can be replaced with the training of q independent single output MoE model with the same number of components and the same training data, and then, forming the vector gating function and q-variate normal components with diagonal covariance matrix.

7.3.3 Support vector regression

Support vector regression (SVR) is a powerful statistical regression tool which has been proposed with the goal of finding $\hat{f}(\mathbf{z})$ with at-most ϵ deviation from $f(\mathbf{z})$ (let's call it output target y) [189]. Consider the linear regression model below:

$$\hat{f}(\mathbf{z}) = \langle \boldsymbol{\beta}, \mathbf{z} \rangle + \mu,$$

where $\boldsymbol{\beta}$ is the $1 \times p$ parameter vectors, and μ is the intercept. The values of parameter vector can be obtained using the following convex optimization problem:

$$V(\boldsymbol{\beta}) = \min_{\boldsymbol{\beta}} \frac{1}{2} ||\boldsymbol{\beta}||_2^2$$

$$s.t.$$

$$\begin{cases} y_i - \langle \boldsymbol{\beta}, \mathbf{z}_i \rangle - \mu \leq \epsilon \\ \langle \boldsymbol{\beta}, \mathbf{z}_i \rangle + \mu - y_i \leq \epsilon \end{cases},$$

where $i = 1, 2, ..., n$. In order to have a feasible solution, it is necessary that a function $\hat{f}(\mathbf{z})$ with at-most ϵ deviation from $f(\mathbf{z})$ exists. However, this may not be true for all of the regression problems, and a way to get rid of infeasibility is to use slack variables ξ_i and ξ_i^*, and formulate the following objective function:

$$V(\boldsymbol{\beta}) = \min_{\boldsymbol{\beta}} \frac{1}{2} ||\boldsymbol{\beta}||_2^2 + c \sum_{i=1}^{n} (\xi_i + \xi_i^*) \tag{7.2}$$

$$s.t.$$

$$\begin{cases} y_i - \langle \boldsymbol{\beta}, \mathbf{z}_i \rangle - \mu \leq \epsilon + \xi_i \\ \langle \boldsymbol{\beta}, \mathbf{z}_i \rangle + \mu - y_i \leq \epsilon + \xi_i^* \\ \xi_i, \xi_i^* \geq 0 \end{cases} ,$$

where $i = 1, 2, ..., n$, and $c > 0$, which can be viewed as trade-off parameter between regression accuracy (flatness of $f(\mathbf{z})$), and the amount of admissible deviations larger than ϵ, which can be viewed as an ϵ-intensive loss function, defined as:

$$|\xi|_\epsilon := \begin{cases} 0, & \text{if } |\xi| \leq \epsilon \\ |\xi| - \epsilon, & \text{otherwise} \end{cases} .$$

To solve the primal objective function given in Eq. 7.2, one way is to use Lagrangian function, which can be derived by unifying the primal objective function and constraint sets with the aid of dual variables, and considering the fact that the formed Lagrangian function has a minimax point with respect to primal and dual variables at the solution point. This means that the partial derivative of Lagrangian function with respect to primal variables should be equal to 0 for optimality. The Lagrangian function L can be defined as:

$$\begin{aligned} \mathrm{L} := \frac{1}{2}||\boldsymbol{\beta}||_2^2 + c\sum_{i=1}^{n}(\xi_i + \xi_i^*) - \sum_{i=1}^{n}(\eta_i\xi_i + \eta_i^*\xi_i^*) \\ - \sum_{i=1}^{n}\alpha_i(\epsilon + \xi_i - y_i + \langle \boldsymbol{\beta}, \mathbf{z}_i \rangle + \mu) - \sum_{i=1}^{n}\alpha_i^*(\epsilon + \xi_i^* + y_i - \langle \boldsymbol{\beta}, \mathbf{z}_i \rangle - \mu) , \end{aligned}$$

where $\boldsymbol{\beta}$, μ, ξ_i and ξ_i^* are primal variables, and η_i, η_i^*, α_i, and α_i^* are dual variables (Lagrangian multipliers). Apparently, Lagrangian multipliers should be non-negative. Based on the above information, the dual optimization problem can be formulated as:

$$V(\boldsymbol{\alpha}, \boldsymbol{\alpha}^*) = \max_{\boldsymbol{\alpha}, \boldsymbol{\alpha}^*} \; -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}(\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)\langle \mathbf{z}_i, \mathbf{z}_j \rangle - \epsilon\sum_{i=1}^{n}(\alpha_i + \alpha_i^*) + \sum_{i=1}^{n} y_i(\alpha_i - \alpha_i^*)$$

s.t.

$$\begin{cases} \sum_{i=1}^{n}(\alpha_i - \alpha_i^*) = 0 \\ \alpha_i, \alpha_i^* \in [0, c] \end{cases} ,$$

in which the model parameters and estimated function can be given as:

$$\boldsymbol{\beta} = \sum_{i=1}^{n}(\alpha_i - \alpha_i^*)\mathbf{z}_i ,$$

$$\hat{f}(\mathbf{z}_0) = \sum_{i=1}^{n}(\alpha_i - \alpha_i^*)\langle \mathbf{z}_i, \mathbf{z}_0\rangle + \mu \ .$$

For solving the above dual optimization problem, interior point method can be taken into account. Also, μ can be computed using Karush-Kuhn-Tucker (KKT), which states that the product of dual variables and corresponding constraints becomes 0 at the point of optimal solution. Note that the inner product $\langle \mathbf{z}_i, \mathbf{z}_0\rangle$ in the above formulations can be replaced with any kernel, including Gaussian kernel. For the current simulation, SVR with Gaussian kernel is taken into account, which results in:

$$\hat{f}(\mathbf{z}_0) = \sum_{i=1}^{n}(\alpha_i - \alpha_i^*)\langle \mathbf{z}_i, \mathbf{z}_0\rangle + \mu = \mu + \sum_{i=1}^{n}(\alpha_i - \alpha_i^*)\ ker(\mathbf{z}_i, \mathbf{z}_0) = \mu + \sum_{i=1}^{n}(\alpha_i - \alpha_i^*)\exp\left(\frac{-||\mathbf{z}_i - \mathbf{z}_0||_2^2}{2\sigma^2}\right) \ .$$

For extending the above procedure to the multi-output version of ordinary SVR, i.e. estimating $\hat{\mathbf{f}}_0(\mathbf{z}_0) = \left(\hat{f}_{1,0}(\mathbf{z}_0), \hat{f}_{2,0}(\mathbf{z}_0), ..., \hat{f}_{q,0}(\mathbf{z}_0)\right)^T$, one needs to solve q dual optimization problems to get the optimal value of dual parameters for each output, and the kernel remains the same for all of the models.

7.3.4 Projection pursuit regression

Projection pursuit regression (PPR) [77] is a general supervised learning method with the form below:

$$\hat{f}(\mathbf{z}) = \sum_{l=1}^{k} g_l(\boldsymbol{\beta}_l^T\mathbf{z}) \ .$$

The above formulation offers an estimation via the combination of nonlinear maps of the feature spaces $\boldsymbol{\beta}_l^T\mathbf{z}$. Functions g_l (known as ridge functions) are always estimated along the direction of $\boldsymbol{\beta}_l$ using a smoothing method. Also $\boldsymbol{\beta}_l^T\mathbf{z}$ represents the projection of $\mathbf{z}$ on the unit length vector $\boldsymbol{\beta}_l$. The reason behind using the name projection pursuit is that we should also estimate vector $\boldsymbol{\beta}_l$ such that the final model fits well. The following objective function is used to train the PPR model:

$$\sum_{i=1}^{n}\left(f(\mathbf{z}_i) - \sum_{l=1}^{k} g_l(\boldsymbol{\beta}_l^T\mathbf{z}_i)\right)^2 \ .$$

Suppose that an initial guess for vector $\boldsymbol{\beta}_l$ $(l = 1, ..., k)$ be available, then the functions $g_l(\cdot)$ can be estimated by spline smoothing method, as below:

$$\sum_{i=1}^{n}\left(f(\mathbf{z}_i) - \sum_{l=1}^{k} g_l(\boldsymbol{\beta}_l^T\mathbf{z}_i)\right)^2 + \sum_{l=1}^{k}\lambda_l\int_{\mathbf{z}_1}^{\mathbf{z}_n}\left(g_l''(\boldsymbol{\beta}_l^T\mathbf{z}_i)\right)^2.d\mathbf{z} \ .$$

Now, let's define the vector $\mathbf{m} = \Big(g(\boldsymbol{\beta}^T\mathbf{z}_1), ..., g(\boldsymbol{\beta}^T\mathbf{z}_n)\Big)^T$. Apparently, the first term of the above objective function is fixed, and we need to minimize the second term. The minimizer of the above problem yields a cubic spline with knots at $\mathbf{z}_i$ $(i = 1, ..., n)$. Now, given that a spline function can be represented as the linear combination of B-splines (spline basis), we get:

$$g(\boldsymbol{\beta}^T\mathbf{z}) = \sum_{i=1}^{n} g(\boldsymbol{\beta}^T\mathbf{z}_i)\mathrm{B}_i(\boldsymbol{\beta}^T\mathbf{z}) \ ,$$

which means that the second term in the objective function can be presented as:

$$\int_{\mathbf{z}_1}^{\mathbf{z}_n} \Big(g''(\boldsymbol{\beta}^T\mathbf{z})\Big)^2 .d\mathbf{z} = \mathbf{m}^T A\mathbf{m} \ ,$$

where A is a $n \times n$ matrix with elements $\int_{\mathbf{z}_1}^{\mathbf{z}_n} \mathrm{B}_i''(\boldsymbol{\beta}^T\mathbf{z})\mathrm{B}_j''(\boldsymbol{\beta}^T\mathbf{z}).d\mathbf{z}$. The objective function can now be viewed as a penalized least square (Tikhonov):

$$||\mathbf{y} - \mathbf{m}||_2^2 + \lambda(\mathbf{m}^T A\mathbf{m}) \ ,$$

with solution $\hat{\mathbf{m}} = (\mathbb{I}_n + \lambda A)^{-1}\mathbf{y}$. After estimating the function g, the parameter $\boldsymbol{\beta}$ vector should be estimated using Quasi Newton-Raphson algorithm. Let's denote the current parameter vector by $\boldsymbol{\beta}_{old}$, then the estimate of $\boldsymbol{\beta}$ will be:

$$g(\boldsymbol{\beta}^T\mathbf{z}_i) \approx g(\boldsymbol{\beta}_{old}^T\mathbf{z}_i) + g'(\boldsymbol{\beta}_{old}^T\mathbf{z}_i)(\boldsymbol{\beta}^T - \boldsymbol{\beta}_{old}^T)\mathbf{z}_i \ .$$

So, we get:

$$\sum_{i=1}^{n} \Big(f(\mathbf{z}_i) - g(\boldsymbol{\beta}^T\mathbf{z}_i)\Big)^2 \approx \sum_{i=1}^{n} g'(\boldsymbol{\beta}_{old}^T\mathbf{z}_i)^2 \Big(\boldsymbol{\beta}_{old}^T\mathbf{z}_i + \frac{f(\mathbf{z}_i) - g(\boldsymbol{\beta}_{old}^T\mathbf{z}_i)}{g'(\boldsymbol{\beta}_{old}^T\mathbf{z}_i)} - \boldsymbol{\beta}^T\mathbf{z}_i\Big)^2 \ .$$

The minimization of the above objective function can be viewed as a weighted least-square problem with weights $g'(\boldsymbol{\beta}_{old}^T\mathbf{z}_i)^2$ and targets $\boldsymbol{\beta}_{old}^T\mathbf{z}_i + \dfrac{f(\mathbf{z}_i) - g(\boldsymbol{\beta}_{old}^T\mathbf{z}_i)}{g'(\boldsymbol{\beta}_{old}^T\mathbf{z}_i)}$.

Now, if deemed required, one more component can be added to the system and the functions $g_l(\cdot)$ can be estimated by backfitting procedure, and the weights can be obtained using Quasi Newton-Raphson algorithm.

Note that for cases with multiple outputs, multiple PPR models should be trained separately. This will not result in computationally expensive model, since PPR usually require few components to learn the map between input and output spaces.

7.4 Soft Oracles

In this section, the mathematical formulation of the considered soft machine learning methods are derived for developing oracles. All of the considered models are differentiable and continuous.

7.4.1 Multi-layer perceptron

Multi-layer perceptron (MLP) is a type of neural network which uses a deep structure for generating appropriate feature spaces, and then linearly combines them to get the output [190]. The deep structure includes a number of adjacent layers (known as hidden layers) which contain a number of hidden nodes. MLP with one hidden layer is the most applicable variant, as increasing the number of hidden layers remarkably increases the system complexity, and makes the training of the resulting network intricate. MLP with one hidden layer shares some structural similarities with PPR. The main difference between MLP and PPR is that the function g (called activation function) is fixed in MLP, and the training just affects the connection weights. Although this results in a simpler training philosophy, the final model may not be so efficient, since the activation functions may not match the dataset.

For the current study, MLP with one hidden layers and k hidden nodes are used. The mathematical formulation of the model can be given as:

$$\hat{f}(\mathbf{z}) = \sum_{l=1}^{k} \omega_l \cdot g(\boldsymbol{\beta}_l^T \mathbf{z}) \ .$$

where functions $g(\cdot)$ are fixed activation functions, ω_l are scalar weights assigned to each hidden node, k is the number of hidden nodes, and $\boldsymbol{\beta}_l$ is a $p \times 1$ parameter vector which linearly maps the input data to feature space (known as input-hidden weights). As can be inferred, the above formulation offers an estimation via the weighted combination of nonlinear maps of the feature spaces to output space. The activation function used in each hidden node of MLP is the sigmoid function with range [0, 1], defined as:

$$g(v) = \frac{\exp(v)}{1 + \exp(v)} \ ,$$

where $v = \boldsymbol{\beta}^T \mathbf{z}$ is a scalar variable. Training of MLP is performed by back-propagation via gradient-descend algorithm. Here, the optimization parameters are ω_l and $\boldsymbol{\beta}_l$ for $l =$

$1, ..., k$. The objective function used for training MLP is:

$$\sum_{i=1}^{n} \left(f(\mathbf{z}_i) - \sum_{l=1}^{k} \omega_l \cdot g(\boldsymbol{\beta}_l^T \mathbf{z}_i) \right)^2 .$$

Then, the parameter values can be obtained by calculating the derivative of the above function with respect to the model parameters. Note, that no biased (intercept) is considered for the model.

Unlike PPR, the training of MLP weights are done simultaneously, by considering a fixed number of hidden layers (k). This can be done via trial and error, and depends on the properties of data at hand, such as number of input variables, size of collected data, having imbalanced data, and *etc.* Also, MLP can be easily extended to multi-output version with the same back-propagation via gradient-descend algorithm.

7.4.2 Randomized neural network

Randomized neural network (RNN) is a class of multi-layer feed-forward neural networks which are based on random feature transformation of inputs, and analytically calculating the hidden-output weights [191]. Compared to other versions of neural networks, RNN enjoys a very fast training method. The structure of RNN is exactly the same as MLP, but with one hidden layer, and can mathematically be expressed as:

$$\hat{f}(\mathbf{z}) = \sum_{l=1}^{k} \omega_l \cdot g(\boldsymbol{\beta}_l^T \mathbf{z} + b_l) .$$

where $g()$ is sigmoid activation function, b_l is the bias of l^{th} hidden node, and the rest of the parameters are the same as those of MLP. Assume that n data are available. Let's form the so-called hidden layer output matrix H and vector $\mathbf{y}$, as below:

$$H = \begin{bmatrix} g(\boldsymbol{\beta}_1^T \mathbf{z}_1 + b_1) & \dots & g(\boldsymbol{\beta}_p^T \mathbf{z}_1 + b_p) \\ \vdots & \ddots & \vdots \\ g(\boldsymbol{\beta}_1^T \mathbf{z}_n + b_1) & \dots & g(\boldsymbol{\beta}_p^T \mathbf{z}_n + b_p) \end{bmatrix} , \ \mathbf{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} .$$

For training RNN, the vectors $\boldsymbol{\beta}_1$ and bias b_l are assigned randomly, and then, the following penalized objective function is used to determine $\boldsymbol{\omega}$:

$$V(\boldsymbol{\omega}, \lambda) = \min_{\boldsymbol{\omega}, \lambda} \ ||\mathbf{y} - H\boldsymbol{\omega}||_2^2 + \lambda \ ||\boldsymbol{\omega}||_2^2 .$$

The above objective function is called Tikhonov regularization, and has the following analytical solution:

$$\boldsymbol{\omega} = (H^T H + \lambda \mathbb{I}_n)^{-1} H^T \mathbf{y} \ .$$

The value of λ can be verified using methods such as Akaike information criterion (AIC) or trial and error. RNN can be easily extended to be used for multi-output applications. To do so, again the values of $\boldsymbol{\beta}_1$ and b_l are assigned randomly, and the weight vectors $\boldsymbol{\omega}_1, ..., \boldsymbol{\omega}_q$ are analytically determined using Tikhonov regularization method.

7.4.3 Adaptive neuro-fuzzy inference system

Adaptive neuro-fuzzy inference system (ANFIS) is a powerful soft function approximation tool which represents the idea of fuzzy function approximation in the form of neural architecture [192]. Let's call the original input vector the crisp input vector. The first step of forming ANFIS is to transfer the input vectors to feature space, which is known as fuzzification. To do so, each element of $\mathbf{z} = (z_1, z_2, ..., z_p)^T$ can be represented by r fuzzy linguistic expressions as $\{h_1^{(j)}, h_2^{(j)}, ..., h_r^{(j)}\}$, where $j = 1, 2, ..., p$. Each of these fuzzy linguistic expressions are indeed shape functions representing the membership of crisp values. The constraint is that the sum of membership values of each z to linguistic functions should add up to 1. Here, Gaussian shape function is used to represent each of the linguistic expressions h. These expressions form the antecedent part of fuzzy rule-base.

After fuzzification of crisp data, each of the resulting linguistic expressions enters the formed rule base (equivalent to hidden layer of ANNs) for inference. ANFIS uses Takagi-Seugeno-Kang (TSK) technique to form the rule-base with $k = r^p$ components (rules):

$$\mathcal{R} = \left\{ IF \ z_1 \ is \ h_{l_1}^{(1)} \ AND \cdots AND \ z_p \ is \ h_{l_p}^{(p)} \ THEN \ \sigma_l \mid l_1, ..., l_p \in \{1, ..., r\} \right\}$$

where $l = 1, ..., k$. The consequent parameter σ_l is set to be a constant. The firing strength of each rule can be calculated as:

$$\omega_l = \prod_{j=1}^{p} h_{l_j}^{(j)} \ ,$$

where $l = 1, ..., k$. Now, the normalized firing strength of each rule can be determined as:

$$\bar{\omega}_l = \frac{\omega_l}{\sum_{l=1}^{k} \omega_l} \ .$$

The output of each rule can be calculated as:

$$g_l = \bar{\omega}_l \cdot \sigma_l \ .$$

The crisp output of ANFIS is the summation of the outputs of each rule:

$$\hat{f}(\mathbf{z}) = \sum_{l=1}^{k} g_l \ .$$

ANFIS uses a hybrid learning technique to optimize the model parameters. To do so, the antecedent parameters (parameters of Gaussian functions h) are initialized, and based on that the output is calculated. Then, least square method is used to calculate the consequent parameters , and the mean square error between estimate and output data are calculated. Then, the error is back-propagated and the antecedent parameters are tuned using gradient descend algorithm.

For the case with multiple outputs, separate ANFIS models should be used for each of the outputs.

7.4.4 Genetic programming

Genetic programming (GP) is a type of symbolic regression which uses evolutionary computing operators such as crossover and mutation to get the approximated model. In particular, multi gene GP (MGGP) has proven its power for function approximation [193]. In MGGP, each potential model is represented as a chromosome including a number of equal length genes, and the number of genes in each chromosome cannot exceed the maximum number of genes. Each gene represents a tree with a predetermined depth and contains a set of operators, e.g. *sin*, *cos*, log, $*$, $-$, $+$, $/$, tanh, and *etc*, as well as a number of operands which appear at the leaf of the three, that include constants values and decision variables (predictors). Let's say that k genes are used to represent a chromosome. Then, the function approximation by each chromosome can be presented as:

$$\hat{f}(\mathbf{z}) = \sum_{l=1}^{k} \omega_l \cdot g_l(\boldsymbol{\beta}, \mathbf{z}) \ ,$$

where g_l is the l^{th} gene, $\boldsymbol{\beta}_l$ is the parameters in the tree resulting from gene g_l and ω_l is the linear combination weight which sums up the output of each tree (gene) to get the

approximated value. The parameters of each gene are the operators defined above which are obtained using selection, recombination and mutation (by GA). After forming the trees, the matrix G and output vectors can be defined as:

$$H = \begin{bmatrix} g_1(\boldsymbol{\beta}_1, \mathbf{z}_1) & \dots & g_p(\boldsymbol{\beta}_p, \mathbf{z}_1) \\ \vdots & \ddots & \vdots \\ g_1(\boldsymbol{\beta}_1, \mathbf{z}_n) & \dots & g_p(\boldsymbol{\beta}_p, \mathbf{z}_n) \end{bmatrix} \ , \ \mathbf{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} \ .$$

Just like RNN, the combination weights are obtained analytically by Tikhonov regularization, as below:

$$\boldsymbol{\omega} = (G^T G + \lambda \mathbb{I}_n)^{-1} G^T \mathbf{y} \ .$$

Note that to foster the diversity of the obtained models, a chromosome cannot have duplicate genes in its structure. Also, other than the usual combination and mutation operators which evolves the structure of each tree, MGGP uses a hyper-crossover (two-point crossover) which allows replacing the genes (distinct trees) between different chromosomes. Let's say that each chromosome can have up to 5 genes (maximum number of genes equals 5). Then, two typical chromosomes can be:

$$Chrom_1 = (g_1, g_3) \quad \& \quad Chrom_2 = (g_2, g_5, g_6, g_8) \ .$$

As an example, the two point crossover can create the following chromosomes from the parents:

$$\begin{cases} Parent_1 = (g_1, \langle g_3 \rangle) \\ Parent_2 = (g_2, \langle g_5, g_6 \rangle, g_8) \end{cases} \Rightarrow \begin{cases} Child_1 = (g_1, g_5, g_6) \\ Child_2 = (g_2, g_3, g_8) \end{cases}$$

For case that the number of genes in a child exceeds the number of maximum genes, structure pruning is performed by randomly removing genes from the structure until the number of genes reaches the maximum admissible number.

MGGP is a powerful function approximation system. However, the resulting models are usually very complicated and cannot be used for inference. Also, the evolutionary computation can be computationally expensive especially for big datasets.

Just like ANFIS, MGGP is best suited for approximating a single output function, and for multi-output systems, distinct models should be trained.

7.5 Quantifying SU-5

As pointed out, SU-5 represents the sources of uncertainty due to sensors malfunction and rapture of the internal components of vehicle, which has an unknown nature. So, some assumptions should be made to quantify this source of uncertainty in an appropriate fashion. Given the fact that sensors are responsible for measuring system states, the sensor malfunction can be viewed as an additive disturbance / noise in the state space model resulting in state measurement error. Based on the details given in Chapter 6, this source of uncertainty can be tamed by LBMPC using a properly trained oracle.

To extract information from dynamic state space model of vehicle suspension system, some steps should be taken. Firstly, it should be ensured that the system has a stable nature, so that it can be controlled. Thereafter, techniques from random perturbation and dynamic linear programming should be used for sufficient excitation of the system dynamics which give us a view for quantifying the uncertainty. Recall from Chapter 6 that the true model dynamics is presented as:

$$\mathbf{x}_{t+1} = A\mathbf{x}_t + B\mathbf{u}_t + g(\mathbf{x}_t, \mathbf{u}_t) \ , \tag{7.3}$$

where $g(\mathbf{x}_t, \mathbf{u}_t)$ is an unknown term, and A and B are known matrixes. As mentioned, the unknown term should be quantified based on the expert's opinion, statistical tools, and random perturbation tools, which will be then used for determining the polytope $\mathcal{W}$ as well as training oracle $\mathcal{O}_t(\tilde{\mathbf{x}}_t, \tilde{\mathbf{u}}_t)$, for nominal and learned models, respectively:

$$\bar{\mathbf{x}}_{t+1} = A\bar{\mathbf{x}}_t + B\bar{\mathbf{u}}_t + \mathbf{d}_t \ , \tag{7.4}$$

$$\tilde{\mathbf{x}}_{t+1} = A\tilde{\mathbf{x}}_t + B\tilde{\mathbf{u}}_t + \mathcal{O}_t(\tilde{\mathbf{x}}_t, \tilde{\mathbf{u}}_t) \ ,$$

where $\mathbf{d}_t \in \mathcal{W}$.

To check the adopted model's behavior, the standard dynamics of the system (presented in Chapter 3) with stablizable actuation signal (calculated by feedback gain, i.e. $\mathbf{u}_t = K\mathbf{x}_t$) is investigated by enforcing an initial external force of 1000 N for 0.006 *sec* to both rear and front tires (recall that $\mathbf{u}_t = (F_{a_1}, F_{a_2})^T$). The gain matrix K is selected such that the poles of the system states $\mathbf{x} = (z_{b_1}, \dot{z}_{b_1}, z_{b_2}, \dot{z}_{b_2}, z_{u_1}, \dot{z}_{u_1}, z_{u_2}, \dot{z}_{u_2})^T$ be placed at $(-2, -0.5, -1, -5, -3, -1.5, -4, -6)^T$, so that the stability be assured. Also, the value of poles are selected to have distance from each other to avoid numerical issues when calculating the gain matrix K online [194]. The state and control signals for the considered simulation are shown in Figure 7.1 and Figure 7.2, respectively.

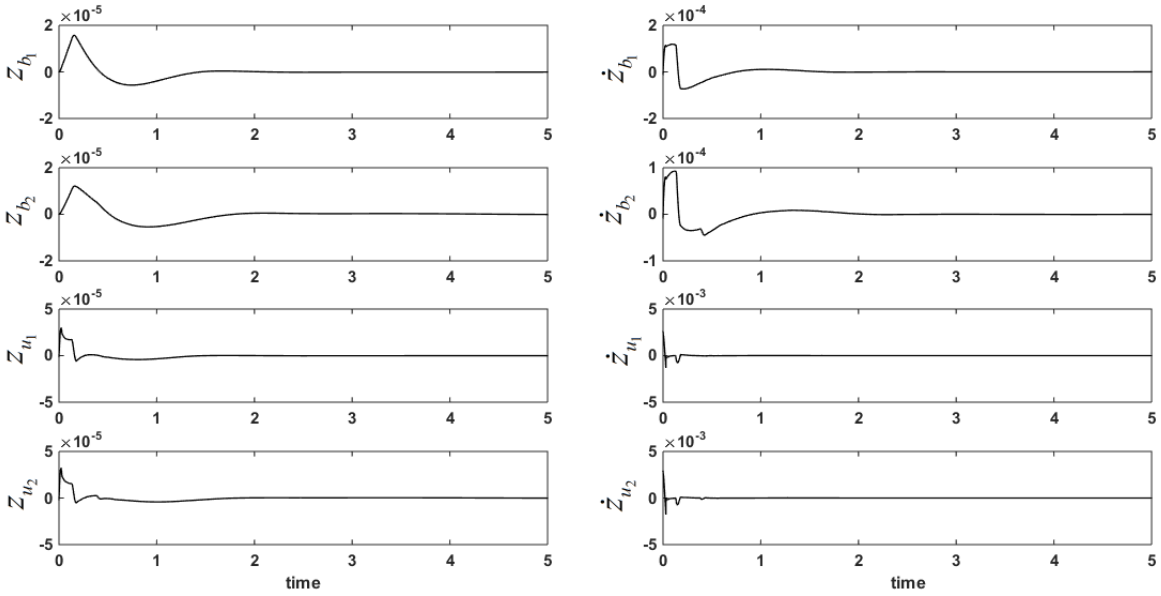

Figure 7.1: The known part ($\mathbf{x}_{t+1} = A\mathbf{x}_t + B\mathbf{u}_t$) of true system's dynamics under stabilizable actuation forces. As seen, system is stable and the base model is proper to be used for further analysis.

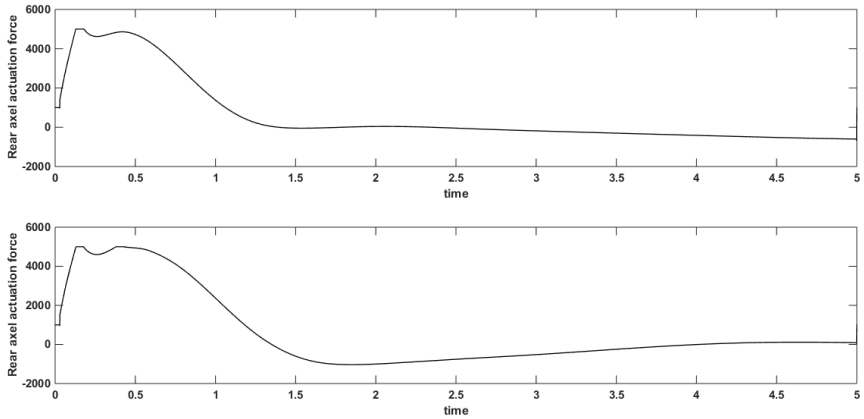

Figure 7.2: Stabilizable actuation forces of front and rear axels.

As can be seen, the control input as a function of feedback gain can make the system states stable (states become 0), and thus, the base model works properly. Also, the actuation forces do not exceed the bounds [-5000 N, 5000 N], which shows that the system is stabilizable by the defined range of actuation force.

Now that the stability potential of the system is verified, a perturbation strategy combined with dynamic linear programming should be used for sufficient excitation of system dynamics to get the required data for quantifying $\mathcal{W}$ as well as training oracle.

If the true state-space model presented in Eq. 7.3 be available, the required data for quantifying $\mathcal{W}$ and training $\mathcal{O}_t(\tilde{\mathbf{x}}_t, \tilde{\mathbf{u}}_t)$ can be obtained by:

$$\mathbf{e}_t = \mathbf{x}_t - (A\mathbf{x}_{t-1} + B\mathbf{u}_{t-1}) \ .$$

Apparently the system state value $\mathbf{x}_t$ and consequently $\mathbf{e}_t$ depends on both unknown term $g(\mathbf{x}_{t-1}, \mathbf{u}_{t-1})$ and actuation signal $\mathbf{u}_{t-1}$.

For the current investigation, the matrixes A and B are known, and are obtained by physical law of motion (see Chapter 6). Also, the range of variation for actuation signal $\mathbf{u}$ is available. However, there is no exact information on how $g(\mathbf{x}_t, \mathbf{u}_t)$ behaves. To come up with an appropriate dataset for training oracle, it is necessary to make sure the system states are excited by means of a wide spectrum of actuation signals, and almost all possible actuation conditions are considered. This can be done by random perturbation techniques to enforce the sufficient excitation / persistent excitation of states [195]. Also, the effect of unknown term $g(\mathbf{x}, \mathbf{u})$ can be studied by means of statistical tools which use the knowledge of expert for calibration of uncertainty. This is the only choice since it is impossible to find an exact (or even approximately exact) expression for $g(\mathbf{x}, \mathbf{u})$. To the best knowledge of the author, in such conditions, Bayesian dynamic models [196] are the most applicable tools for approximating $g(\mathbf{x}, \mathbf{u})$, since the structure of the state-space has a linear dynamic nature plus an unknown uncertain term. In the rest of the section, the details for treating each of the two elements affecting $\mathbf{e}_t$ are scrutinized.

To make sure the actuation signal $\mathbf{u}$ sufficiently excites the states, an additional term called exploratory signal ($\mathbf{u}^{exp}$) is added to the original control command, which can be mathematically expressed as $\mathbf{u}_t = K\mathbf{x}_t + \mathbf{u}_t^{exp}$ [195, 197]. Based on the recommendation given in [197], the best choice for exploratory signal is to consider a rich set of sine and cosine waves of different amplitudes and frequencies, and combine them randomly by weight vector $\boldsymbol{\omega}$, whose elements are drawn from uniform distribution $Unif(0, 1)$, and are normalized to sum up to 1. For the current simulation, 9 sine and 9 cosine waves with amplitudes 100, 1000, and 2000, and frequencies 1, 2 and 4 are taken into account. All possible combinations of the current waves are shown in Figure 7.3 (a).

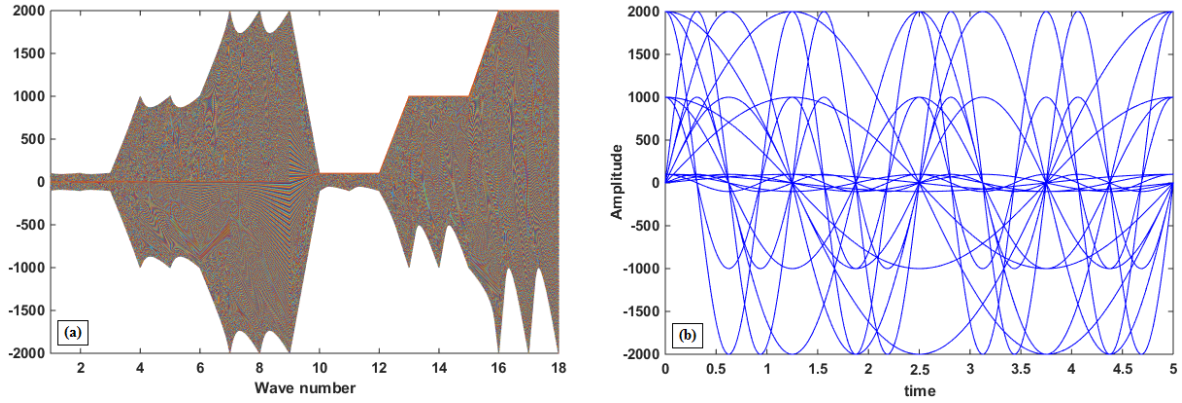

Figure 7.3: All possible combinations of considered sine and cosine waves to form exploratory signals.

Also, the 18 periodic waves in time domain are shown in Figure 7.3 (b). The simulation is conducted for 5 *sec* with resolution 0.001 *sec*. Figure 7.3 (a) shows 5000 possible values of the 18 signals which can be randomly combined over the process. Apparently, the space covered by the considered waves (includes 5000 combinations) is very rich. This can be also verified from the time-domain plot presented in Figure 7.3 (b).

What remains is to use Bayesian dynamic model for approximating the effect of $g(\mathbf{x}, \mathbf{u})$. Recall the true state-space model presented in Eq. 7.3, and note that at this point, we have enough tools and measurements to treat the linear part $A\mathbf{x}_t + B\mathbf{u}_t$. Let's take $\boldsymbol{\mu} = (\mathbf{x}; \mathbf{u})$ and $F = [A\ B]$, then, one way for approximating Eq. 7.3 with unknown uncertainty $g(\mathbf{x}, \mathbf{u})$ is the use of Bayesian dynamic model, as below:

$$
\begin{aligned}
&\mathbf{x}_{t+1} = F\boldsymbol{\mu}_t + \mathbf{w}_t, \qquad \mathbf{w}_t \sim N(\mathbf{0}_{8\times 1}, W)\ , \\
&\boldsymbol{\mu}_t = \boldsymbol{\mu}_t + \mathbf{v}_t, \qquad\quad \mathbf{v}_t \sim N(\mathbf{0}_{10\times 1}, V)\ , \\
&s.t. \\
&\mathbf{u}_t = K\mathbf{x}_t + \mathbf{u}_t^{exp}\ .
\end{aligned}
$$

Here, W and V are covariance matrixes, and based on the prior information $\mathcal{I}_0$, it is known that:

$$\boldsymbol{\mu}_0 \mid \mathcal{I}_0 = (0, 0, 0, 0, 0, 0, 0, 0, 1000, 1000)^T\ .$$

Since $g(\mathbf{x}, \mathbf{u})$ in Eq. 7.3 is uncertain, there is no choice but to find a distribution for the unknown term, and repeat the simulation to get the expected variation of states

over independent simulation. For the current investigation, this is done by means of the considered Bayesian dynamic model. It can be seen that the Bayesian model breaks the total uncertainty into two different uncertain dynamic equations (known as observation and state equations respectively) with additive noises following normal distributions. Indeed, in the case that no exact information for calibrating the uncertain term $g(\mathbf{x}, \mathbf{u})$ be available, using normal distribution seems to be the most logical choice. Now, the remaining step is to find logical W and V matrixes for the model. It is most likely that the measurement error for each of the state signals be independent of each other, and can happen at any time due to sensor malfunction. This offers using diagonal matrix W. Also, for V which stands for the uncertainty of parameter, it is known that the measurement noise of actuation signals and state signals are independent from each other. The variation of state values of passive model (suspension system without any actuation force) when subjected to an initial external force of 1000 N for 0.006 *sec* is shown in Figure 7.4. The obtained results indicate that the range of variation of states in suspension system equipped with an actuator almost surely remains within the following lower and upper bounds:

$$\mathbf{x}_{lb} = (-0.0029, -0.0348, -0.0028, -0.0253, -0.0032, -0.3638, -0.0034, -0.4147)^T \ ,$$

$$\mathbf{x}_{ub} = (0.0009, 0.0300, 0.0006, 0.0209, 0.0073, 0.2966, 0.0077, 0.3210)^T \ .$$

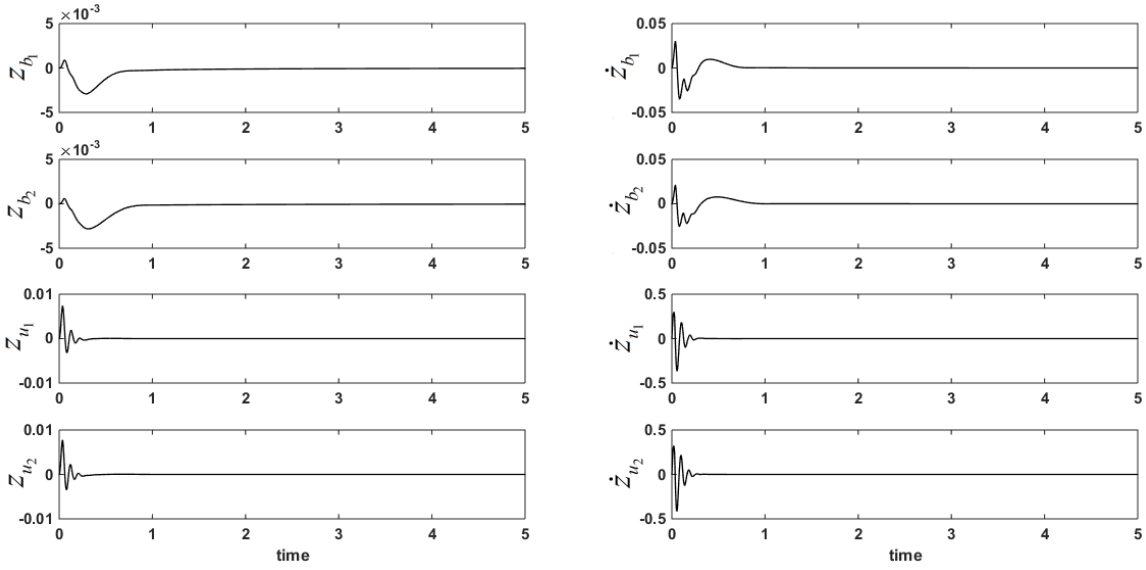

Figure 7.4: Variation of state values of passive model subjected to an initial external force.

So, the mean value of each state based on the above bounds is:

$$\mathbf{x}_{ave} = (-0.0010, -0.0024, -0.0011, -0.0022, 0.0021, -0.0336, 0.0021, -0.0469)^T \ .$$

As mentioned, since the values of suspension system equipped with an actuator almost surely remains within the obtained bounds, the standard deviation (*std.*) of each state values for 99% confidence interval (CI) will be:

$$\mathbf{x}_{std.} = (0.0007, 0.0126, 0.0007, 0.0090, 0.0020, 0.1282, 0.0022, 0.1429)^T \ .$$

Also, *std.* of actuation signals is set to be $\mathbf{u}_{std.} = (10, 10)^T$. So, it is a logical choice to set $diag(V) = (\mathbf{x}^2_{std.}; \mathbf{u}^2_{std.})$. Also, it is apparent that any measurement error pertaining to the displacement of rear and front sprung and unsprung masses affects the velocity of displacement. Based on the author's experimental evaluation, and also the obtained bounds, this dependencies are realized as $\text{Cov}(z_{b_1}, \dot{z}_{b_1}) = 0.000006$, $\text{Cov}(z_{b_2}, \dot{z}_{b_2}) = 0.000003$, $\text{Cov}(z_{u_1}, \dot{z}_{u_1}) = 0.0001$, and $\text{Cov}(z_{u_2}, \dot{z}_{u_2}) = 0.0001$. Also, the variation of actuation signals are independent. To sum up, V can be expressed as:

$$V = \begin{bmatrix} 0.0007^2 & 0.000006 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.000006 & 0.0126^2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.0007^2 & 0.000003 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.000003 & 0.0090^2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.0020^2 & 0.0001 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.0001 & 0.1282^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.0022^2 & 0.0001 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.0001 & 0.1429^2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 10^2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 10^2 \end{bmatrix}$$

Also, the same approach should be followed to find a proper value for W. This can be done by finding a bound for $\mathbf{x}_t - A\mathbf{x}_{t-1}$ of the passive model. Although it cannot be assured that the variation of $\mathbf{x}_t - A\mathbf{x}_{t-1}$ for the passive model encapsulates that of active model, it can be viewed as a good approximation. By following the same procedure (this time considering 90% CI), W is set to be:

$$W = diag(0.0001, 0.0076, 0.0001, 0.0057, 0.0001, 0.0284, 0.0001, 0.0281) \ .$$

Now that the Bayesian dynamic model is verified, Monte-Carlo Markov chain and bootstrap techniques are used to quantify state measurement error $\mathbf{e}$, and based on that

verify $\mathcal{W}$, and also extract dataset for training $\mathcal{O}_t(\tilde{\mathbf{x}}_t, \tilde{\mathbf{u}}_t)$. As mentioned, the simulation is conducted for 5 *sec* with step-time 0.001 *sec* to capture the dynamics of the model under uncertainty. So, at the end of the simulation, a profile with 5000 points are archived. The state measurement under uncertainty can be viewed as an iterative procedure with 5000 iteration. To improve the precision of the obtained results, bootstrap method is taken into account. To do so, each iteration is itself repeated for 100 times, so that the dataset has the following configuration:

$$\begin{aligned}
&\textbf{iter 1}: && \mathbf{e}_1^{(1)}, && \mathbf{e}_1^{(2)}, && ..., \mathbf{e}_1^{(100)} \\
&\textbf{iter 2}: && \mathbf{e}_2^{(1)}, && \mathbf{e}_2^{(2)}, && ..., \mathbf{e}_2^{(100)} \\
&\quad\vdots && && \vdots && \\
&\textbf{iter 5000}: && \mathbf{e}_{5000}^{(1)}, && \mathbf{e}_{5000}^{(2)}, && ..., \mathbf{e}_{5000}^{(100)}
\end{aligned}$$

After the collection of dataset, at each iteration (set-pint), the bootstrap estimate and bootstrap *std.* are calculated using the following equations:

$$\hat{\mathbf{e}}_{boot} = \frac{1}{n_b} \sum_{i=1}^{n_b} \mathbf{e}^{(i)} \ ,$$

$$std._{boot} = \sqrt{\frac{1}{n_b - 1} \sum_{i=1}^{n_b} (\hat{\mathbf{e}}_{boot} - \mathbf{e}^{(i)})^2} \ ,$$

The results of simulation are indicated in the form of bootstrap estimate and 95% bootstrap CI in Figure 7.5. The mean profile is obtained by calculating $\hat{\mathbf{e}}_{boot}$ at each iteration, and also the bounds are obtained by $std._{boot}$ at each iteration. Two datasets are extracted from the simulation results. The first dataset is obtained by $\hat{\mathbf{e}}_{boot}$ which is used for training oracles, and the second dataset is collected by calculating $\hat{\mathbf{e}}_{boot} \pm 1.96\ std._{boot}$ and will be used for determining $\mathcal{W}$ in a nonparametric fashion.

As mentioned in Chapter 6, polytope $\mathcal{W}$ is represented by a number of half-spaces, acting as inequality constraints. This means that $\mathcal{W}$ consists of a set of inequality bounds that are imposed on each of the 8 elements of vector $\mathbf{e}$, and this inequality bounds are derived such that they encapsulate the uncertainty of $g(\mathbf{x}, \mathbf{u})$ by a certain degree of confidence. As mentioned, the inequality bounds forming $\mathcal{W}$ are obtained in a non-parametric fashion based on the results of conducted simulation. To do so, the empirical cumulative density function (CDF) of the obtained distribution for each element of $\hat{\mathbf{e}}_{boot}$ is calculated and the two tail quantiles affording a 95% CI are determined. So, it is expected that the empirically developed $\mathcal{W}$ encapsulates the true uncertainty with 95% confidence.

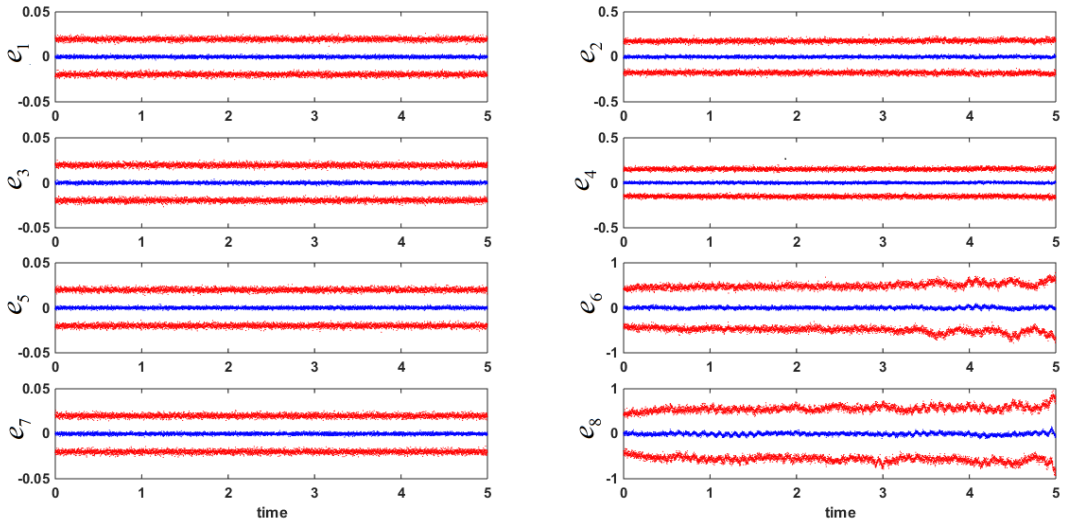

Figure 7.5: Bootstrap estimates (in blue) with their corresponding 95% CI (in red).

The first step is to sort $\hat{\mathbf{e}}_{boot}$ profiles with their corresponding $std._{boot}$. The sorted profiles are shown in Figure 7.6. By having the sorted $\hat{\mathbf{e}}_{boot}$ profiles, the empirical CDF can be obtained using the following equation:

$$\hat{F}_n(a) = \frac{1}{n}\sum_{i=1}^{n}\mathbf{I}(\mathbf{e}_i \leq a) = \frac{\#\{e_i \mid e_i \leq a\}}{n} \quad , a \in \mathbb{R} \ ,$$

where $\mathbf{I}(\mathcal{A})$ is the indicator function of $\mathcal{A}$. It can be easily shown that $E_F[\hat{F}_n(a)] = F(a)$, which means that $\hat{F}_n(a)$ is an unbiased estimator of $F(a)$. Once the empirical CDFs are obtained, the 95% CI bound for each elements of $\mathbf{e}$, i.e. $e \in [d_{min}, d_{max}]$, (recall from Eq. 7.4 that $\mathbf{d}_t \in \mathcal{W}$ is a vector representing the disturbance in state-space model), can be obtained by:

$$\mathcal{P}r(e \leq d_{min}) = \hat{F}_n(d_{min}) = 0.025 \ ,$$
$$\mathcal{P}r(e \leq d_{max}) = \hat{F}_n(d_{max}) = 0.975 \ .$$

Apparently, d_{min} and d_{max} are two data points in the sorted bootstrap estimate profiles shown in Figure 7.6, and each of them also varies between bounds $\hat{\mathbf{e}}_{boot} \pm 1.96\ std._{boot}$. So, even a better choice for finding bounds is to first determine $[d_{min}, d_{max}]$, and then, extract the margin of variation of each bound from the corresponding bootstrap CIs, as below:

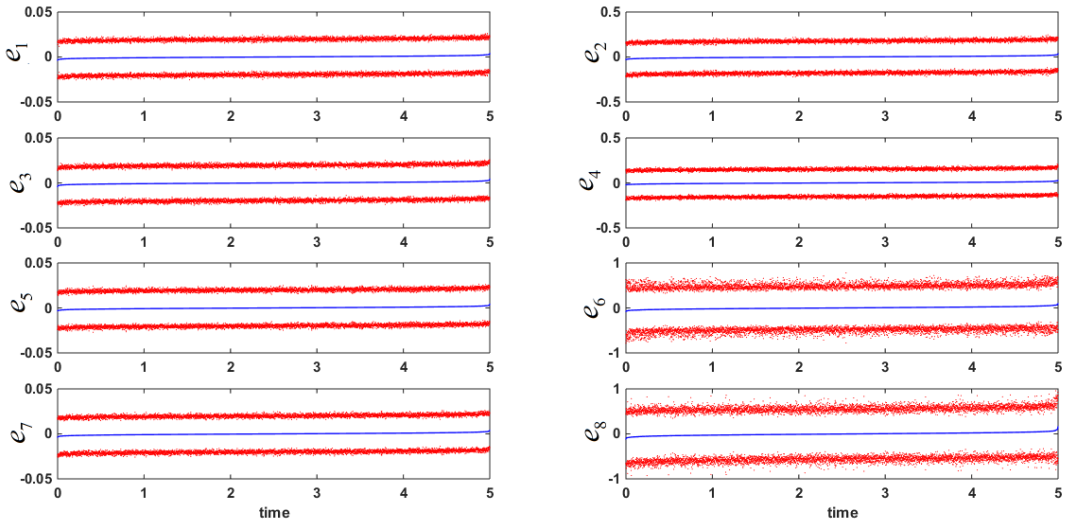

Figure 7.6: Sorted bootstrap estimates (in blue) with their corresponding 95% CI (in red).

$$d_{min} \in [d_{min} - 1.96\ std._{boot}(d_{min})\ ,\ d_{min} + 1.96\ std._{boot}(d_{min})]\ ,$$
$$d_{max} \in [d_{max} - 1.96\ std._{boot}(d_{max})\ ,\ d_{max} + 1.96\ std._{boot}(d_{max})]\ .$$

Empirical CDFs are presented in Figure 7.7.

Based on the obtained CDFs, the following bounds are obtained for the elements of $\mathbf{e}$:

$$-0.0019 \leq e_1 \leq 0.0019\ ,$$
$$-0.0184 \leq e_2 \leq 0.0177\ ,$$
$$-0.0019 \leq e_3 \leq 0.0020\ ,$$
$$-0.0144 \leq e_4 \leq 0.0161\ ,$$
$$-0.0019 \leq e_5 \leq 0.0020\ ,$$
$$-0.0478 \leq e_6 \leq 0.0484\ ,$$
$$-0.0020 \leq e_7 \leq 0.0020\ ,$$
$$-0.0649 \leq e_8 \leq 0.0546\ ,$$

In the next section, the detailed comparative results regarding the use of the considered statistical and soft computing machine learning tools for developing an efficient oracle for LBMPC are presented.

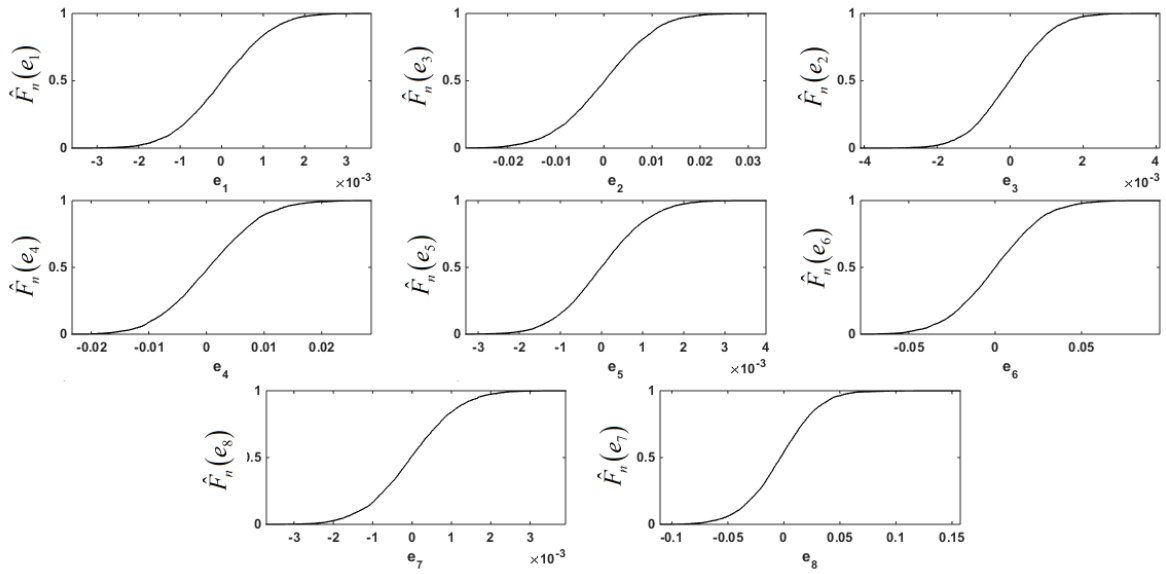

Figure 7.7: Empirical CDFs of each error term e.

7.6 Comparative Study

This section is devoted to the detailed numerical simulation for developing an efficient oracle for LBMPC. In the first sub-section, the simulation setup is presented, and in the second sub-section, the detailed results of the numerical simulation are given.

7.6.1 Simulation setup

As mentioned, the first dataset obtained from bootstrap simulation is used for training oracles, and has $n = 5000$ data points (5 *sec* simulation with step-point 0.001 *sec*), with 10 inputs ($p = 10$) and 8 outputs ($q = 8$). Note that the input dataset is obtained from the known piece-wise linear part of the true state-space model in which the actuation force is determined by stabilizable feedback gain. To excite the suspension system, an external force of 1000 N is imposed on both front and rear tires for 0.006 *sec*. The input vector is $\mathbf{z} = (z_{b_1}, \dot{z}_{b_1}, z_{b_2}, \dot{z}_{b_2}, z_{u_1}, \dot{z}_{u_1}, z_{u_2}, \dot{z}_{u_2}, F_{a_1}, F_{a_2})^T$, and the output vector is $\mathbf{f}(\mathbf{z}) = (f_1(\mathbf{z}) = e_1, f_2(\mathbf{z}) = e_2, ..., f_8(\mathbf{z}) = e_8)^T$. The input profiles used for training the models are shown in Figure 7.1 and Figure 7.2. Also, the output profiles used for training are the bootstrap estimates (blue profiles in Figure 7.5).

The considered models are PPR, mixture of Gaussian experts (MoGE), Kriging, SVR, MLP, RNN, ANFIS, and MGGP. Based on empirical sensitivity analysis and using model selection tools, PPR with 3 components, MoGE with 2 Gaussian regressors and 1 iteration of EM, MLP with 3 hidden layers (respectively with 10, 5 and 2 hidden nodes), RNN with 40 hidden nodes, ANFIS with 2 linguistic expressions and Gaussian membership functions, and MGGP with 4 genes, 100 iteration of evolutionary optimization, operators $*$, $-$, $+$, $/$, and maximum tree depth of 6 are used for simulation.

To have a fair comparison among the adopted machine learning tools, the regression process is repeated 10 times and the statistical results are reported. The considered performance evaluation metrics are mean absolute error (MAE), max absolute error (MaxE) and *std*. Also, graphical tools such as QQ-plot and ACF of residuals, correlation between model estimates and true data, and estimated profile of the chosen model are taken into account for better interpretation and facilitating the analysis.

Note that since the dataset has a time-varying nature and the order of data points are important, based on the argument given in Chapter 4, k-fold forward chaining process is used instead of k-fold cross-validation for simultaneous training and testing / validation. $k = 10$ is found proper for the current simulation.

After training the models, it should be assured that the models are bounded and continuous, which are necessary for theoretical results presented in Chapter 6. Thus, the training model is exposed to different unseen input profiles and the boundedness and continuity of outputs are checked.

Due to the fact that the models will be used at the heart of state-space model in a real-time fashion, to avoid numerical problems which leads to unacceptable simulation, it is better to normalize the input dataset. This is because, for the current database, neglecting this issue can result in the multi-scalability of input variables, which means that some variables excessively affect the fitting, and incur an incorrect performance of trained models. The normalization of input dataset is done in such a way that all data fall within the range [0, 1] to make the dataset compatible with sigmoid activation function used in MLP and RNN.

It should be noticed that the trained model is used as an additive term in the state-space model of vehicle suspension system which has a dynamic nature. So it is important that the range of the trained model be compatible with the range of the variation of state-space model. Since, we are not sure what are the maximum and minimum possible values of additive disturbance, it is not wise to use a normalized output dataset for training (we cannot reliably transform the estimated normalized output to the true disturbance space). So, it is decided to let the output dataset remain in its true form so that the oracle can

directly read the states and actuation signals measured by sensors in real-time, normalize them, and then, estimate the additive disturbance.

Also, due to the linear correlation found between some of the variables and to improve the functioning of the methods, one more pre-processing is done on the input data by using principle component analysis (PCA). In this way, a proper orthogonal transformation is made to get a set of values of linearly uncorrelated variables, which are used for training.

All of the simulation are conducted in MATLAB and R software on a Pentium IV DELL laptop, with Windows 7 operating system, Intel Dual core 2.2 GHz, and 2 GBs RAM.

7.6.2 Numerical results

At the first stage of the numerical analysis, the performance of the methods are compared in terms of the adopted visualization tools and metrics. Figure 7.8 depicts the results obtained by PPR. As seen, in some of the cases, the correlation between the estimated values and data points are weak, and it seems that PPR cannot successfully cover the range of the possible state measurement error values. This is even clearer from the last sub-plot which belongs to the estimated values. With regard to the obtained residuals, QQ-plot and ACF plot indicate that the model residuals follow normal distribution and are not correlated.

Figure 7.9 visualizes the performance of MoGE. It seems that MoGE suffers from the same flaws, and is not able to cover all range of possible outcomes. Also, for e_3, e_4, e_6, and e_8 cases, the ACF plots significantly violate the 95% CI, which augurs the incapability of MoGE to wash out the dependencies among the residual errors. Having said that the estimated values of MoGE are smooth, which can be viewed as a positive point, especially for the current problem that the developed oracle needs to be differentiable.

Figure 7.10 indicates the performance of Kriging method. It is obvious that Kriging has a very good estimation power and in most of the cases, its 95% estimation interval covers the output dataset. Also, QQ-plot has a good shape and the model residual quantiles match the theoretical quantiles. Note that slight violation at some points (lags 1 and 2 of e_1, e_5, and e_6) can be observed for the ACF which are negligible.

Figure 7.11 visualizes the performance of SVR. It can be observed that SVR performs a weak estimation, and for most of the cases, the model output remains constant. This means that the method cannot be used for time dependent estimation application (time-varying estimation). The method obviously fails to capture the underlying dynamics of the datasets. Also, as a result of such a performance, the residual errors have dependency for some of the cases.

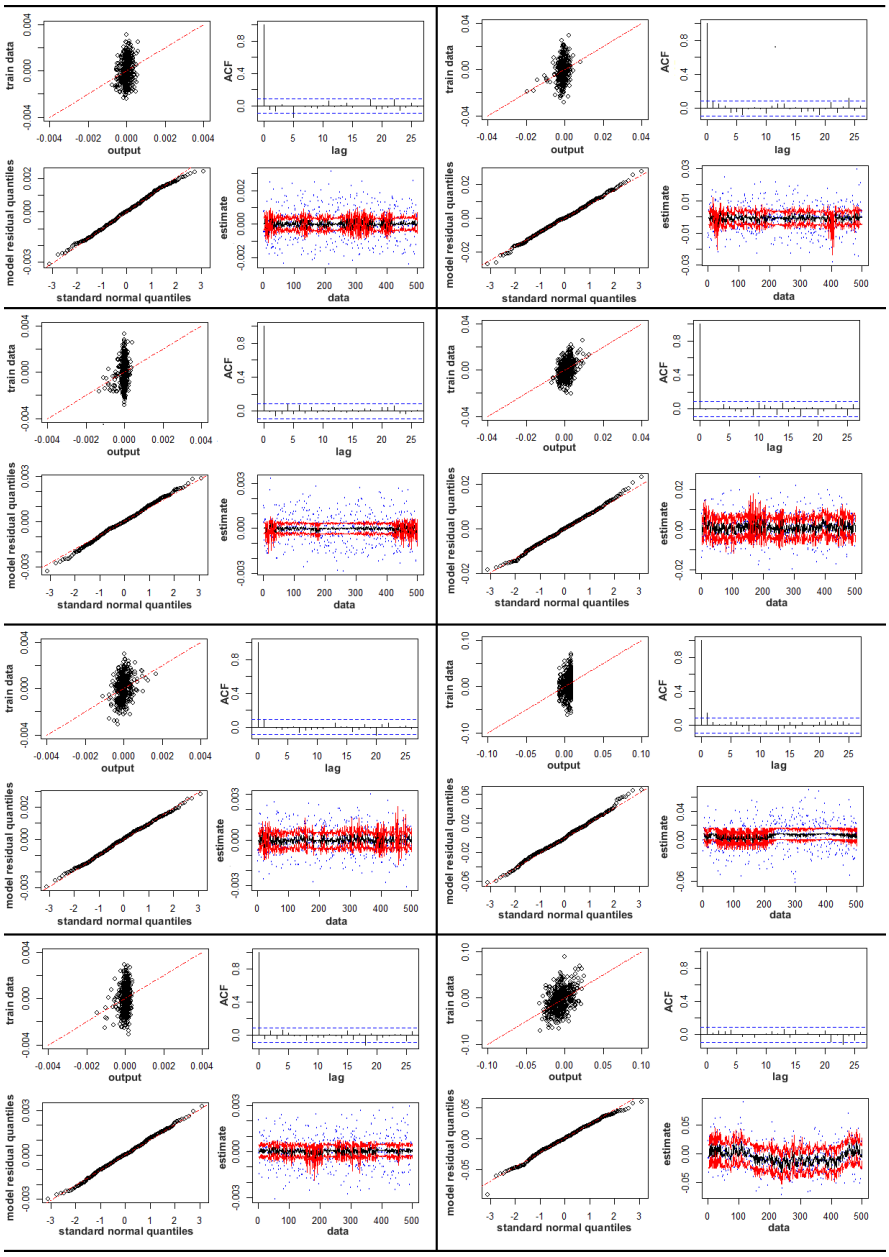

Figure 7.8: Visualization of PPR performance for e_1, e_2, e_3, e_4, e_5, e_6, e_7, and e_8.

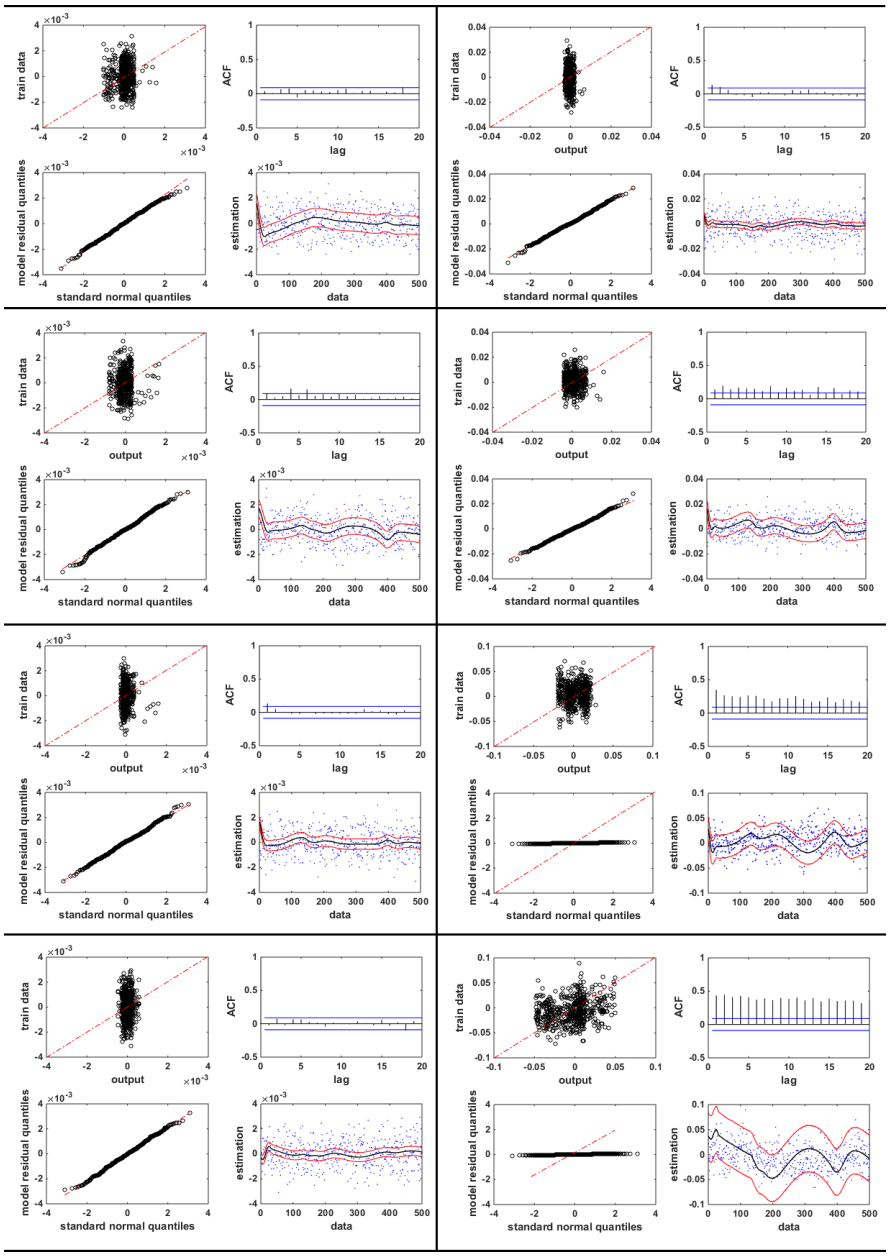

Figure 7.9: Visualization of MoGE performance for e_1, e_2, e_3, e_4, e_5, e_6, e_7, and e_8.

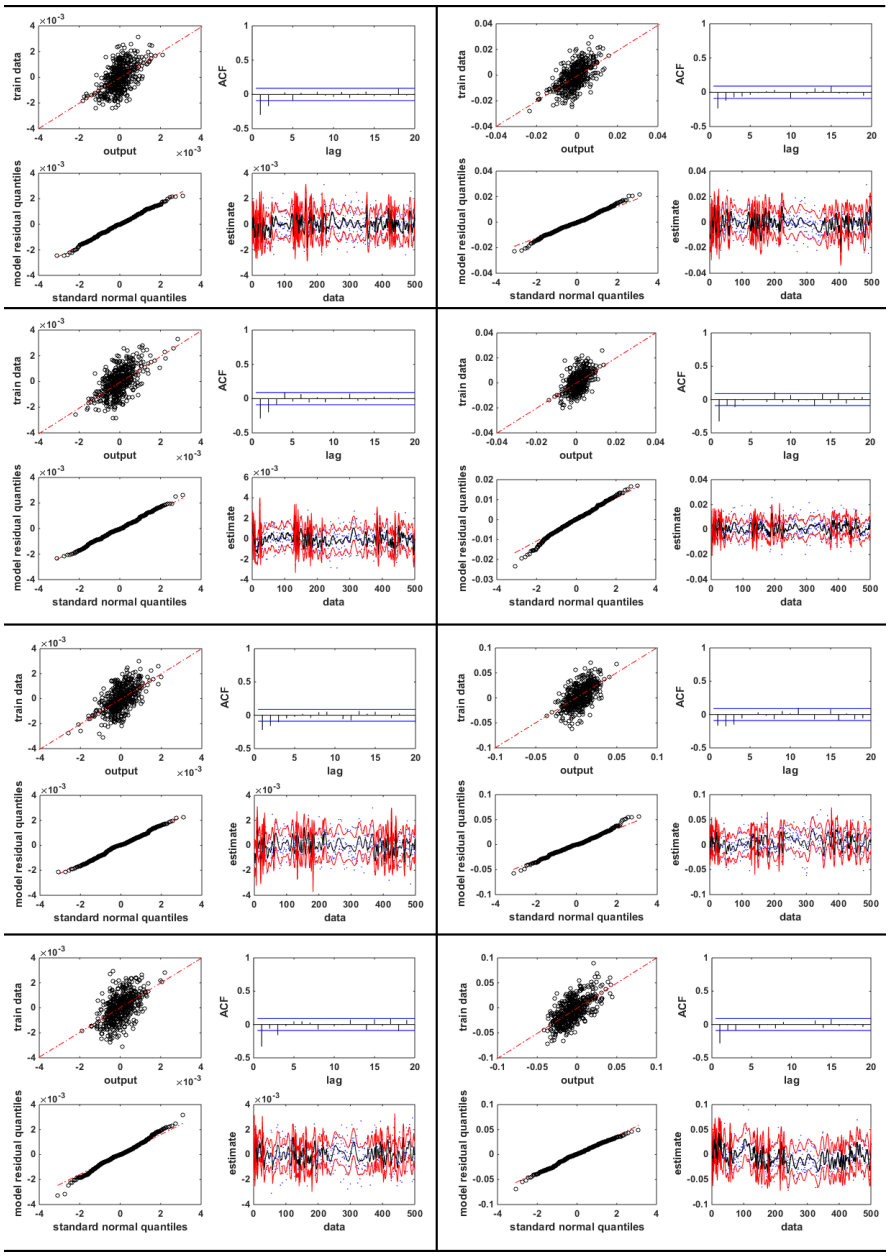

Figure 7.10: Visualization of Kriging performance for e_1, e_2, e_3, e_4, e_5, e_6, e_7, and e_8.

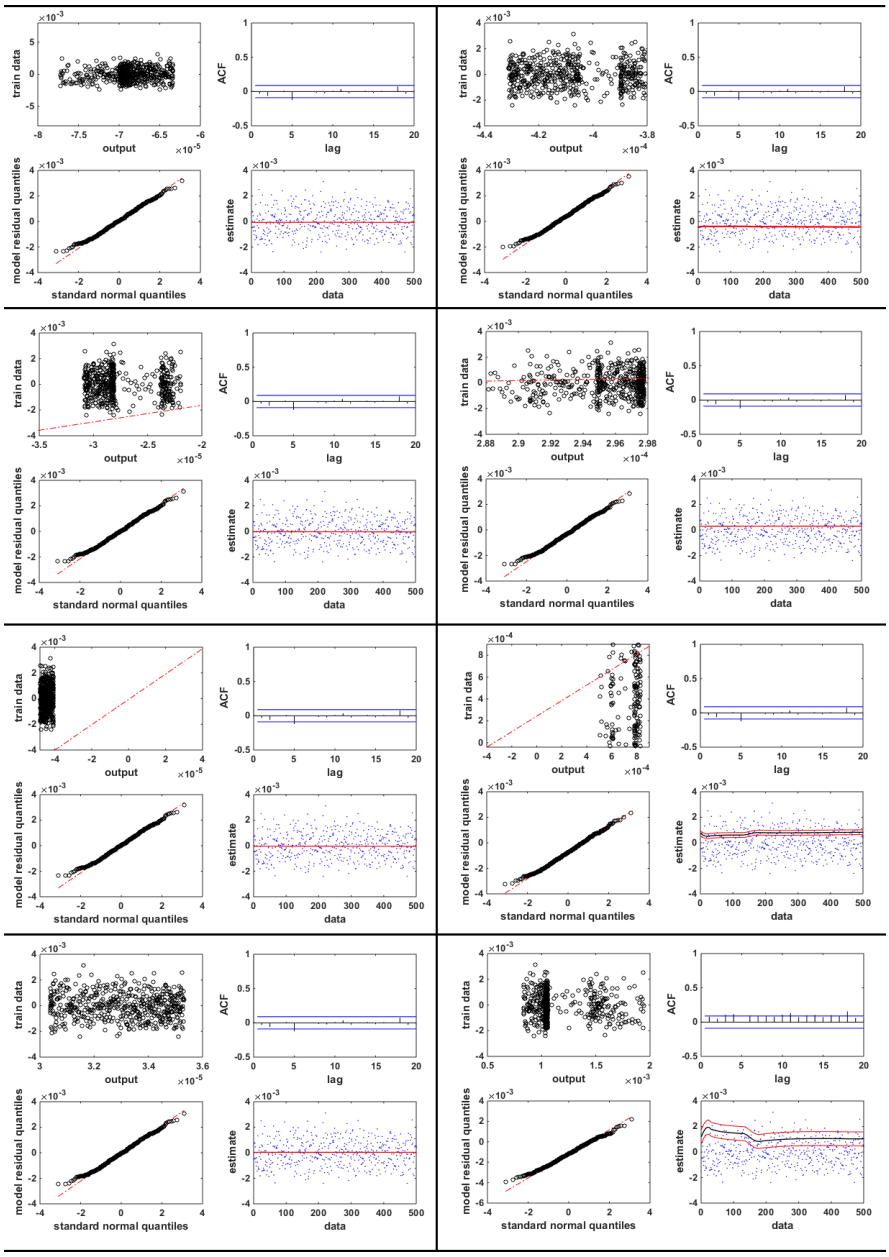

Figure 7.11: Visualization of SVR performance for e_1, e_2, e_3, e_4, e_5, e_6, e_7, and e_8.

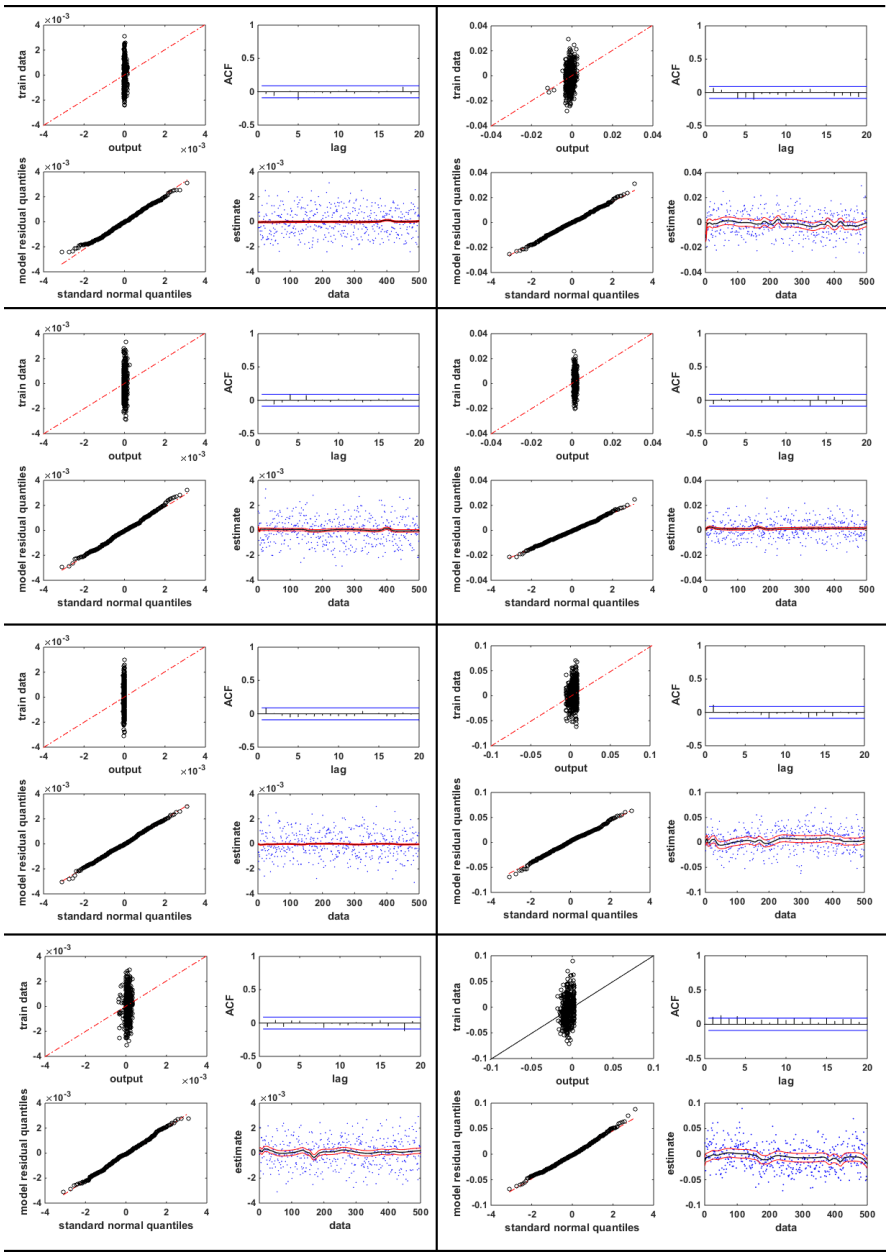

Figure 7.12: Visualization of MLP performance for e_1, e_2, e_3, e_4, e_5, e_6, e_7, and e_8.

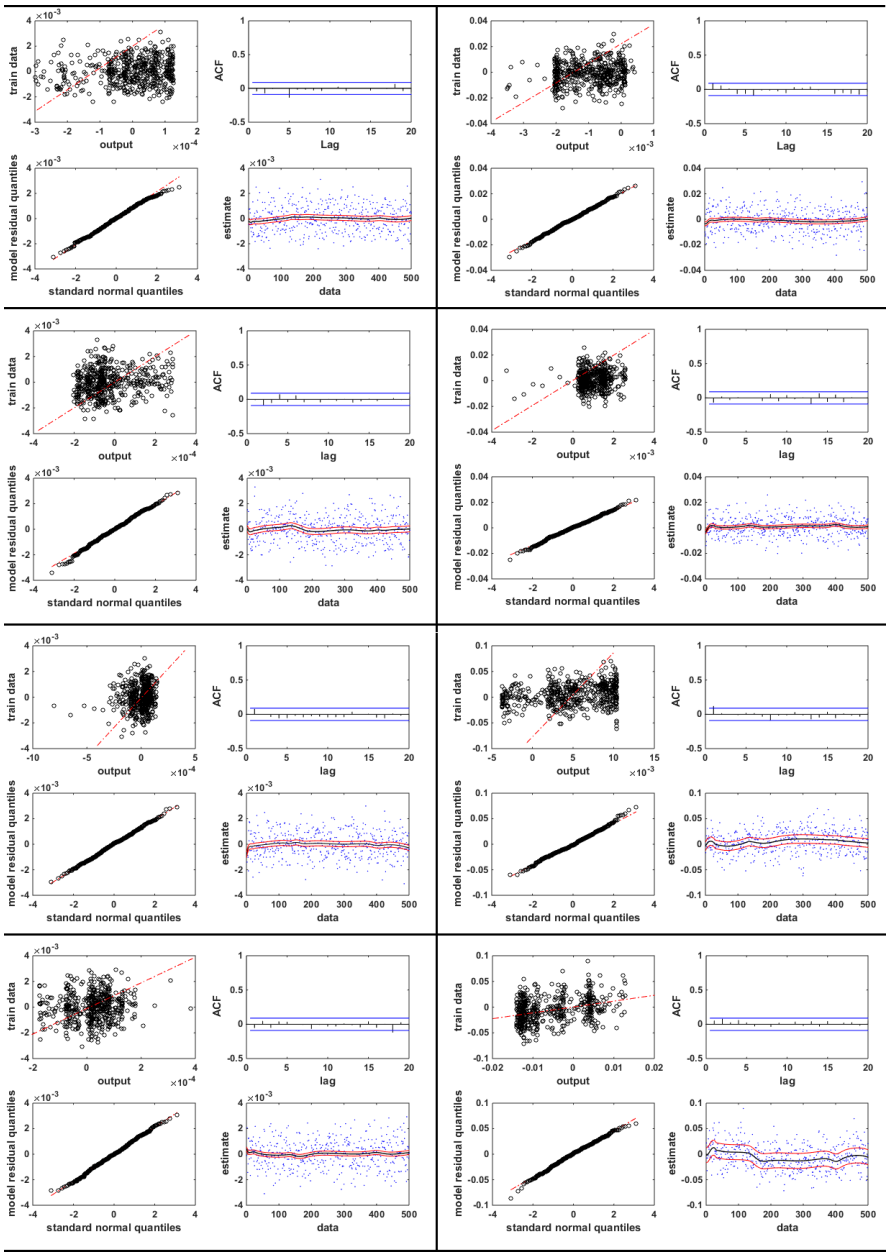

Figure 7.13: Visualization of RNN performance for e_1, e_2, e_3, e_4, e_5, e_6, e_7, and e_8.

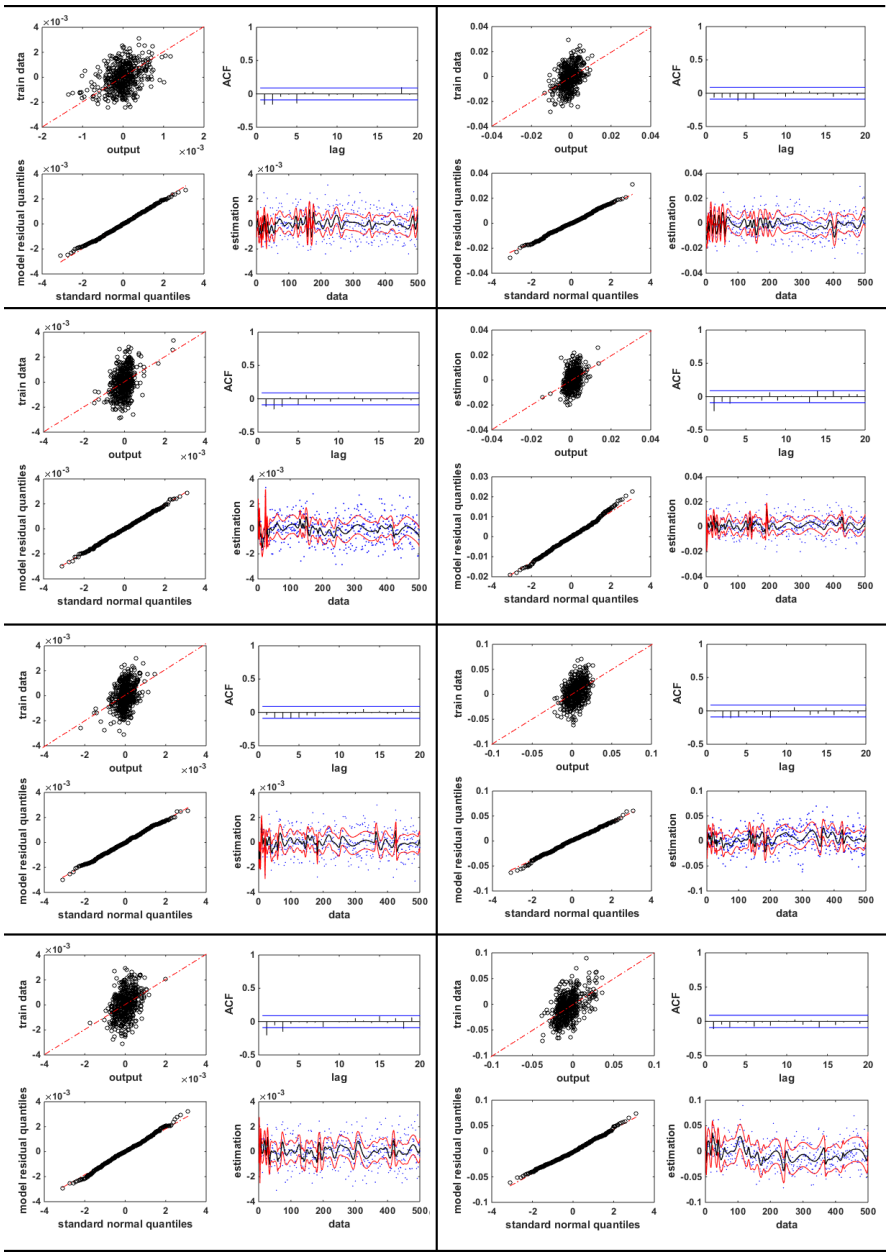

Figure 7.14: Visualization of ANFIS performance for e_1, e_2, e_3, e_4, e_5, e_6, e_7, and e_8.

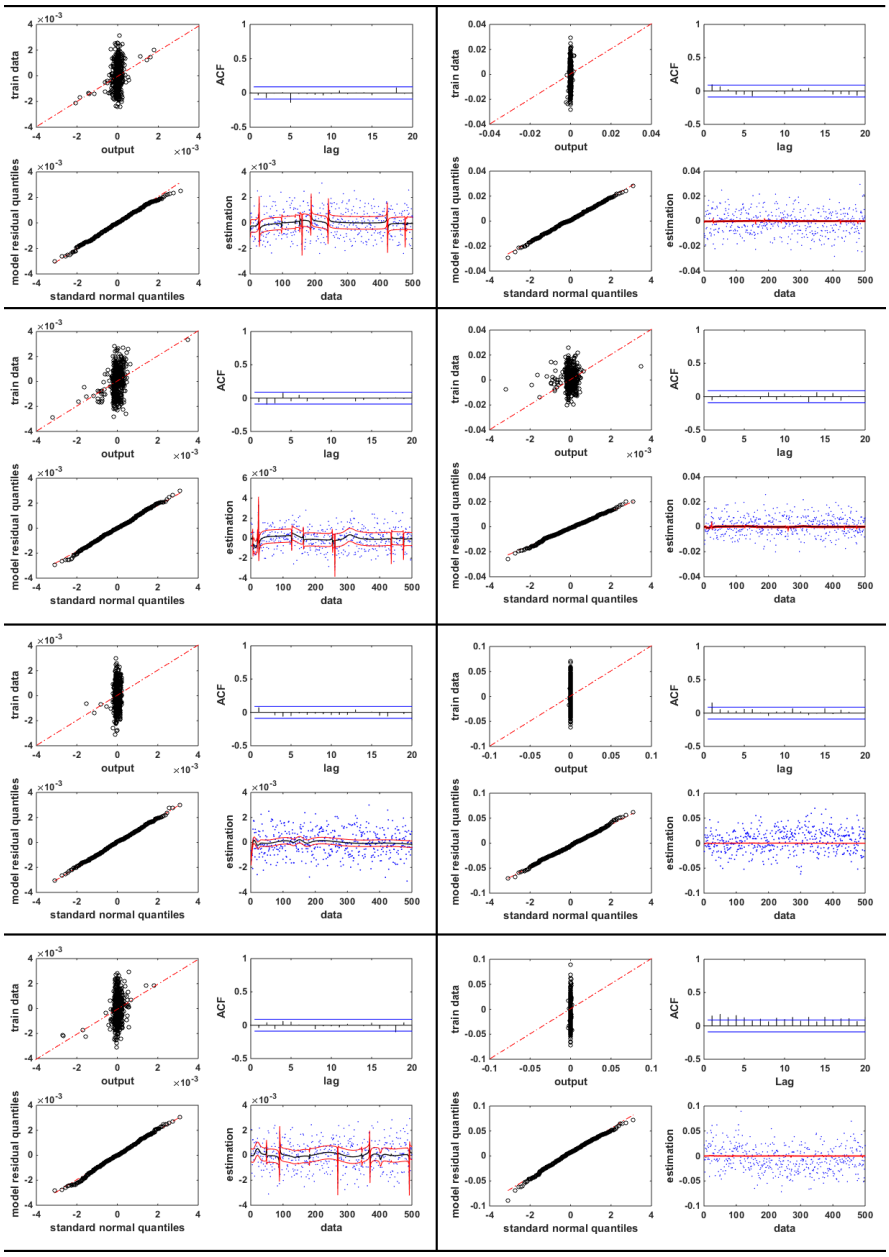

Figure 7.15: Visualization of MGGP performance for e_1, e_2, e_3, e_4, e_5, e_6, e_7, and e_8.

Figure 7.12 depicts the results of MLP. The performance of the method is better than SVR (better estimation for some of the cases and more coverage of the data points). QQ-plots obtained by MLP residuals follow Gaussian distribution and the ACF does not have significant values which is a positive aspect of the method.

Figure 7.13 depicts the estimation results of RNN. It seems that RNN and MLP have very close estimations (MLP is slightly better). Also, based on QQ-plot and ACF plot, it seems that RNN output residuals are independent.

Figure 7.14 belongs to the ANFIS model. As seen, the method has a good approximation, and clearly outperforms the other models (except Kriging). Also, the ACF and QQ-plot results are promising, and the model errors are independent. Compared to Kriging, it seems that the 95% estimation bound of Kriging has a better coverage of data points for most of the cases.

Figure 7.15 indicates the visualization results of MGGP. The performance of the method is not satisfactory, and for some of the cases, the estimated values are constant. Also, significant spikes are observed for some cases, which makes its performance and smoothness questionable. Moreover, for some cases, the model residual errors have dependencies and ACF values violate the 95% CI bounds.

From the visualization plots, it can be concluded that the best performance in terms of estimation power and independency of residual errors belongs to Kriging. In almost all of the cases, the model's estimations follow the trajectory of data profile, meaning that it captures the dynamic trend of data points.

To complete the evaluation regarding the estimation power of the methods, the values of performance metrics are reported in Table 7.1. The best results are shown in bold. It can be seen that Kriging surpasses the other methods for all of the cases in terms of the considered metrics. ANFIS and PPR take the second and third places with respect to the performance evaluation metrics. The only point is that ANFIS has a much complicated structure compared to Kriging and PPR, which can undermine its applicability for our case study in which the oracle used at the heart of LBMPC should be adaptable, and fast. The other methods have a very close performance (except MGGP) with SVR and MoGE being slightly weaker than MLP and RNN. Apparently, MGGP cannot compete with the other models and yielded unsatisfactory results. Also, in terms of MaxE metric, the worst performance belongs to MGGP, meaning that it has the largest deviation among the considered methods. Having said that the largest deviation of MoGE is also considerable.

Finally, the estimated profile of the best model (Kriging) for all 5000 data points (corresponding to 5 *sec* of simulation) is shown in Figure 7.16 for e_8, which is the most complicated simulation scenario.

Table 7.1: Simulation results for the considered models

Methods	Metrics	e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8
PPR	**MAE**	0.0008	0.0067	0.0007	0.0053	0.0007	0.0168	0.0008	0.0178
	MaxE	0.0031	0.0275	0.0033	0.0232	0.0029	0.0653	0.0032	0.0900
	std.	0.0009	0.0086	0.0010	0.0067	0.0009	0.0210	0.0010	0.0224
MoGE	**MAE**	0.0008	0.0073	0.0007	0.0063	0.0007	0.0194	0.0008	0.0246
	MaxE	0.0032	0.0295	0.0034	0.0246	0.0032	0.0752	0.0034	0.1016
	std.	0.0010	0.0091	0.0010	0.0078	0.0009	0.0239	0.0010	0.0286
Kriging	**MAE**	**0.0004**	**0.0041**	**0.0004**	**0.0036**	**0.0004**	**0.0100**	**0.0004**	**0.0102**
	MaxE	**0.0024**	**0.0221**	**0.0024**	**0.0190**	**0.0024**	**0.0490**	**0.0025**	**0.0557**
	std.	**0.0006**	**0.0058**	**0.0006**	**0.0051**	**0.0006**	**0.0139**	**0.0006**	**0.0142**
SVR	**MAE**	0.0007	0.0072	0.0007	0.0062	0.0007	0.0189	0.0008	0.0227
	MaxE	0.0032	0.0289	0.0034	0.0242	0.0032	0.0736	0.0034	0.0966
	std.	0.0009	0.0090	0.0009	0.0077	0.0009	0.0235	0.0010	0.0277
MLP	**MAE**	0.0007	0.0071	0.0007	0.0061	0.0007	0.0167	0.0008	0.0182
	MaxE	0.0032	0.0284	0.0034	0.0242	0.0032	0.0668	0.0034	0.0748
	std.	0.0009	0.0089	0.0009	0.0076	0.0009	0.0209	0.0010	0.0228
RNN	**MAE**	0.0007	0.0071	0.0007	0.0061	0.0007	0.0175	0.0008	0.0205
	MaxE	0.0032	0.0289	0.0034	0.0243	0.0031	0.0690	0.0034	0.0820
	std.	0.0009	0.0089	0.0009	0.0076	0.0009	0.0219	0.0010	0.0257
ANFIS	**MAE**	0.0007	0.0063	0.0007	0.0054	0.0007	0.0149	0.0007	0.0155
	MaxE	0.0029	0.0264	0.0031	0.0220	0.0030	0.0590	0.0032	0.0679
	std.	0.0009	0.0080	0.0008	0.0068	0.0008	0.0187	0.0009	0.0196
MGGP	**MAE**	0.0009	0.0086	0.0009	0.0122	0.0029	0.0286	0.0008	0.0282
	MaxE	0.0092	0.1258	0.0200	1.2819	0.6603	2.2160	0.0035	0.1195
	std.	0.0012	0.0133	0.0016	0.0740	0.0315	0.1367	0.0010	0.0347

Obviously, the 95% estimation bound of Kriging entails most of the data points, which unveils the reliability of the model to be used at the heart of LBMPC for real-time application.

One of the important properties of Kriging lies in its estimation strategy which is interpolation based on the observed data. So, in real-time, by adding new chunks of acquiesced data to the model, one can improve its estimation performance. This means that it is also a very good choice for incremental and adaptive learning, which are of paramount importance for real-time / online applications, such as active suspension control.

All in all, soft and statistical methods show comparable results. In particular, two statistical methods, i.e. Kriging and PPR, and one soft method, i.e. ANFIS, yield promising results. The worst performance belongs to MGGP which is a soft method. By further investigation, it was realized that the complex structure of MGGP and a lot of hyper parameters made it hard for GA to efficiently evolve the MGGP architecture.

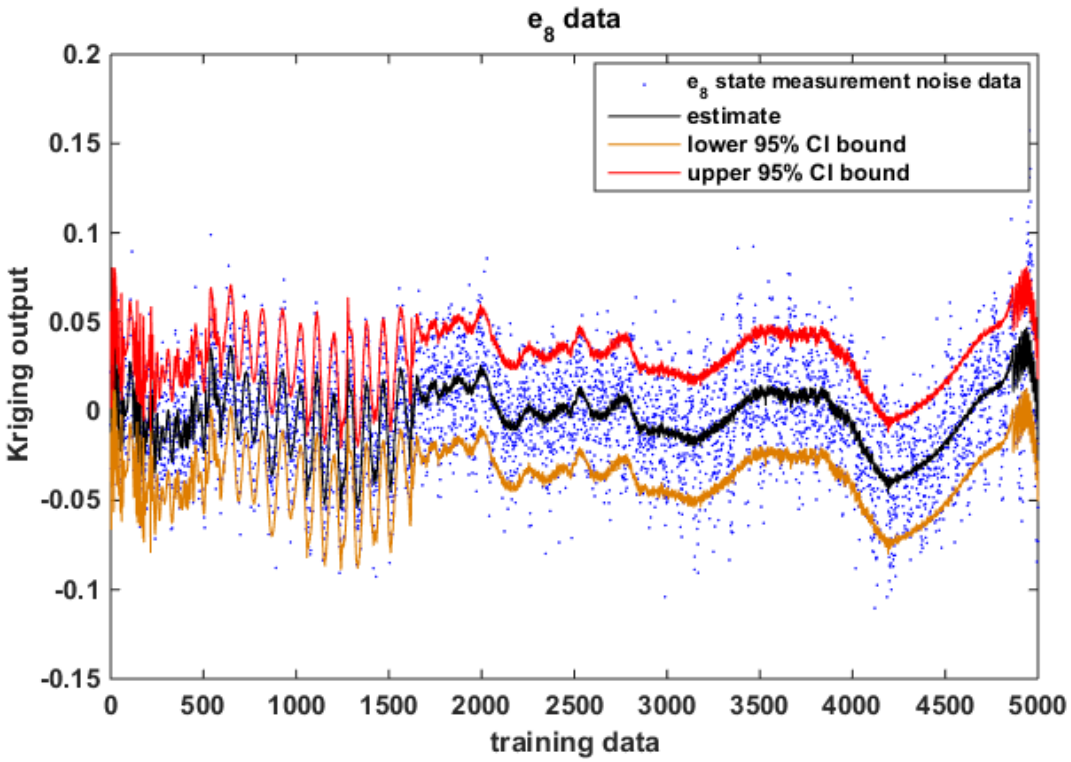

Figure 7.16: Kriging estimated profile for all of the available training data (5 *sec* simulation).

As mentioned, the remarkable structural complexity of soft methods impede an efficient training of the model, especially when too many model parameters are involved.

Now, it is intended to compare the computational efficiency of learning systems used for training each of the models.

Table 7.2 summarizes the average simulation time of each model for all of the estimation cases. As expected, RNN possesses the fastest training algorithm, which is simply the randomized selection of input-hidden connection weights, and calculation of output weights by means of L_2 regularization. PPR, MoGE and Kriging also favor fast learning strategies while MGGP has the slowest learning system, which is not surprising, as it uses GA that is a population-based metaheuristic and involves different searching operators (selection, recombination, mutation, and hyper two-point crossover).

Table 7.2: Average training time of the rival methods (in *sec*)

Methods	e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8	t_{ave}
PPR	0.0217	0.0131	0.0152	0.0315	0.0217	0.0142	0.0162	0.0223	0.0195
MoGE	0.0548	0.0497	0.0459	0.0534	0.0518	0.0466	0.0469	0.0479	0.0496
Kriging	0.0809	0.0766	0.0787	0.0851	0.0797	0.0802	0.0724	0.0725	0.0783
SVR	3.4623	3.3935	3.4996	3.5964	3.4947	3.5824	3.7102	3.5481	3.5359
MLP	0.2600	0.2770	0.2416	0.2932	0.2526	0.3309	0.2479	0.3061	0.2762
RNN	0.0009	0.0010	0.0010	0.0010	0.0012	0.0009	0.0008	0.0008	0.0009
ANFIS	6.9177	6.7323	6.8323	6.7761	6.7861	6.8276	6.8106	6.8522	6.8169
MGGP	17.004	14.357	12.168	14.764	13.392	14.149	17.444	16.355	14.954

To have a better interpretation and a more comprehensive comparison, the computational time of MGGP is considered as a base, and the speed of other methods are determined by determining how many times faster they are, as below:

$$rate(method) := \frac{\tau(MGGP)}{\tau(method)} \ ,$$

where $\tau(\cdot)$ is a function measuring the simulation time of training a model, and can be calculated by *tic / toc* command in MATLAB and by *system.time* command in R. $rate(method)$ shows how many times a model is faster than MGGP.

The log value of the rate function for each model is shown in Figure 7.17. The plot provides a good comparison of the considered methods. As stated, in spite of the good performance of ANFIS, it has a complex hybrid learning structure which makes its training slow, especially when the dataset has more than three input variables (based on the author's assessment).

Also, it can be understood from Figure 7.17 that SVR has a slow learning method. This is rather obvious, since SVR formulates a dual objective function with different types of parameters and constraints, which requires the use of interior point algorithm for tuning the parameters. For example, it was observed that the training speed of RNN is 4435 times faster than that of SVR for e_8. Such observations bring us to the conclusion that SVR is not efficient for regression compared to the other methods (from both computational time and efficiency prospectives), though its classifier counterpart, i.e. support vector machine (SVM), is a very powerful and well-consensuse classification method.

To sum up, it seems that statistical models PPR, MoGE, and Kriging are faster than soft methods such as MLP, ANFIS and MGGP. Also the fastest and lowest models are RNN and MGGP, which both belong to soft methods. It can be concluded that three statistical and one soft models favor a computationally efficient learning systems.

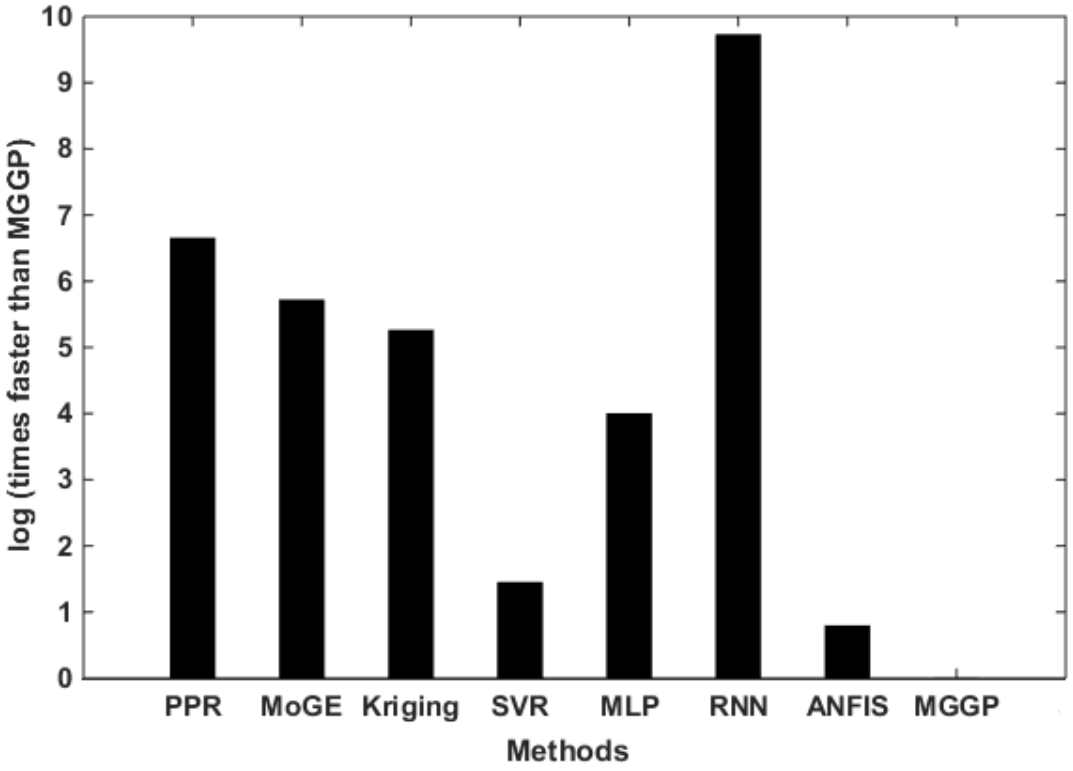

Figure 7.17: Comparison of the computational speed of learning systems used in rival models.

Another important issue pertains to the interpretability of the considered models. As mentioned, one drawback of soft models is their complicated structures, which deters the users from utilizing them for inference. As an example, MGGP suggests the following symbolic model for estimating e_1:

$$e_1 = 7.966 \times 10^{-5} + (1.299 \times 10^{-5})z_2 - \frac{(1.299 \times 10^{-5})z_1}{z_3 - 2z_1 - z_3 z_4} - \frac{(1.299 \times 10^{-5})z_4^2}{z_1 - z_4}$$
$$- \frac{(5.663 \times 10^{-5})z_1(z_2 - z_3)(z_3 - 1)}{z_1 - z_2} + (9.794 \times 10^{-4})z_1 z_4(z_1 - z_2 + z_4)(z_1 - z_2 + z_3 + 1)$$
$$-(1.767 \times 10^{-2})z_1 z_4(z_2 - z_3)(z_3 - 1)(z_3 - z_2 + 1) \ .$$

It seems that the model is sparse, as it removed some variables from the derived symbolic

map. However, the formula is highly nonlinear and too complicated to be used for studying the effect of variables on output or any other type of inference.

Finally, it should be pointed out that to ensure Kriging can be used at the heart of LBMPC, its estimated profile for all of the available data and all of the case studies were checked, and it was observed that the model is continuous and bounded (within 95% CI which never violates the output space specified in the previous section). So, among the adopted models, Kriging is chosen as oracle $\mathcal{O}_t(\tilde{\mathbf{x}}_t, \tilde{\mathbf{u}}_t)$ for LBMPC.

7.7 Some Recommendations

Based on the simulation conducted in this chapter, the following remarks are observed:

1. It was observed that Bayesian uncertain models can be efficiently used for quantifying the uncertainty of un-modelled dynamics in control state-space models. The salient asset of such probabilistic tools was that they could efficiently compensate the lack of knowledge in the true control state-space model by replacing the un-modelled dynamic term (which could be either linear or nonlinear) with additive uncertain noise. Also, several important concepts such as bootstrap could be used for improving the precision of uncertainty quantification.

2. The prominent asset of Bayesian type uncertainty quantification was that the model could be improved over time by updating the knowledge regarding the dynamic model parameters. Indeed, such models not only are found appropriate for inference, but also can be effectively used for incremental learning, as the model can be improved over time by updating the prior information representing the experts opinion regarding the process.

3. It was observed that bootstrap could be used to come up with a rich dataset for efficient quantification of uncertainty influenced suspension system's dynamics. The results obtained by bootstrap simulation was very compatible with physical nature of the suspension system, and it could be concluded that the considered Bayesian dynamic model together with bootstrap estimate were reliable, at-least for the current application. Also, the non-parametric CDF obtained from the conducted simulation was very accurate, thanks to the richness of the obtained dataset, which was used for determining a polytope that encapsulates the uncertainty with 95% CI. To the best knowledge of the author, this is the first time that a precise statistical analysis is

done for the accurate quantification of suspension system's state measurement error, and also there is a reasonable logic behind verifying the polytope $\mathcal{W}$.

4. By performing a thorough comparative study on the considered soft and statistical machine learning methods via different performance evaluation metrics, it was observed that Kriging method is the best choice for developing oracle. In particular, the method was able to robustly estimate the additive unmeasured disturbance, and its training speed was remarkably fast. Also, due to its boundedness and differentiable formulation, it possessed the required features of an efficient oracle which satisfies the theoretically guaranteed robustness and stability of LBMPC.

5. During the experimentations, it became even more transparent that the salient asset of statistical methods is their firm mathematical foundation as well as their interpretability. Due to the fact that there is a logic behind each step of forming a statistical model, it is possible for the one who manipulates them to adapt/tune their structures for certain datasets to get the best results. This is when, in spite of their interesting nature and very acceptable performance, soft computing methods are not interpretable, and usually have black-box like structures with the capability of universal approximation.

Note

All MATLAB and R codes pertaining to the simulation performed in this chapter can be found in Appendix.

Chapter 8

Simulation Results and Comparative Study

In this chapter, the simulation results pertaining to the use of learning-based model predictive controller (LBMPC) for vehicle suspension control problem are presented. The chapter is organized as follows. Firstly, the detailed information regarding the simulation setup are given. Also, the whole simulation and uncertainty quantification performed in the previous chapters are reviewed. Thereafter, the simulation results are presented, and the performance of LBMPC is compared with different well-known controllers. The simulation is then followed by applying the obtained results to the nonlinear suspension system model to validate the performance of LBMPC. Finally, based on the obtained results, the pros and cons of using LBMPC for the current problem are enumerated.

8.1 Simulation Setup

Recall the vehicle suspension model derived in Chapter 3. The model has 8 states $\mathbf{x} = (z_{b_1}, \dot{z}_{b_1}, z_{b_2}, \dot{z}_{b_2}, z_{u_1}, \dot{z}_{u_1}, z_{u_2}, \dot{z}_{u_2})^T$ and 2 control inputs, i.e. $\mathbf{u} = (F_{a_1}, F_{a_2})^T$. Also recall from Chapter 6 that, to ensure the stability of LBMPC, the control command is calculated as $\bar{\mathbf{u}}_t = K\bar{\mathbf{x}}_t + (\phi - K\psi)\boldsymbol{\theta}$, where ϕ is a 2×2 matrix, K is a 2×8 gain matrix, ψ is a 8×2 matrix, and $\boldsymbol{\theta} \in \mathbb{R}^2$ is the solution of the optimization problem formulated in LBMPC. Based on sensitivity analysis and experts' opinion which were in agreement with the physical properties of the vehicle, the constraints on the states are chosen as $-0.05 \leq z_{b_1} \leq 0.05$, $-0.5 \leq \dot{z}_{b_1} \leq 0.5$, $-0.05 \leq z_{b_2} \leq 0.05$, $-0.2 \leq \dot{z}_{b_2} \leq 0.2$, $-0.1 \leq z_{u_1} \leq 0.1$,

$-1 \leq \dot{z}_{u_1} \leq 1$, $-0.1 \leq z_{u_2} \leq 0.1$, and $-1 \leq \dot{z}_{u_2} \leq 1$. Also, the constraints of actuation signals are $-5000 \leq F_{a_1} \leq 5000$ and $-5000 \leq F_{a_2} \leq 5000$. The piece-wise linearization of the nonlinear model is done around time-varying equilibrium / operating points $(\delta_1)_0$, $(\delta_2)_0$, $(\dot{\delta}_1)_0$, and $(\dot{\delta}_2)_0$, defined in Chapter 3. The sampling time for discretization is 10^{-3} *sec*. Apparently, using time varying operating points yields a linear parameter varying (LPV) system from the original nonlinear model, which is a more reliable approximation than using a linear parameter invariant approximation of the original model. The initial condition for the current simulation is $\mathbf{x}_0 = \mathbf{0}_{8\times 1}$ and $\mathbf{u}_0 = \mathbf{0}_{2\times 1}$.

To make the linearized model stable, the feedback gain matrix K is selected such that the poles of the closed-loop system $\mathbf{x}_{t+1} = (A + BK)\mathbf{x}_t$ be placed at $(-2,-0.5,-1,-5, -3,-1.5,-4,-6)^T$. Note that the poles are selected to have distance from each other to avoid numerical difficulties when calculating K in the loop. To calculate the invariant set Ω, the well-known minimal robust positively invariant set (MRPIS) method (see *Definition 6.21*) is taken into account. In this context, at each set-point i, the projection of invariant set Ω on state-space $\mathcal{X} \in \mathbb{R}^8$ is determined using $\mathcal{R}_i = \bigoplus_{j=0}^{i-1}(A+BK)^j\mathcal{W}$, where $\mathcal{R}_0$ only has one element which is the origin $\mathbf{0}_{8\times 1}$. The detailed steps taken for determining the polytope $\mathcal{W}$ which encapsulates the uncertainty due to measurement error were presented in Chapter 7. For the current investigation, geometric computations for determining set $\mathcal{R}_i$ are done in MATLAB. Having said that one can use the computational geometry facilities of the multi-parametric tool-box (MPT) available online at http://people.ee.ethz.ch/ mpt/3/.

The dataset required for training the oracle $\mathcal{O}(\tilde{\mathbf{x}}_t, \tilde{\mathbf{u}}_t)$ were obtained using Bayesian dynamic programming, and bootstrap simulation. Based on a comprehensive numerical investigation, a powerful non-parametric model called ordinary Kriging is selected as the oracle (see Chapter 7 for details). Thanks to the structural flexibility of the learning system used in Kriging, it can be used as an adaptive / incremental / time-varying oracle, if deemed necessary. Also, the model is differentiable and bounded which is required for satisfying *Theorem 6.6*. Also, given that the dataset used for training Kriging was obtained under sufficient excitation (using a rich set of sine and cosine waves with different amplitudes and frequencies), and also given the continuity of the quadratic objective function defined in Eq. 6.10, the epi-convergence of Kriging, as a non-parametric oracle, can be easily proven (it can be shown that *Theorem 6.16* is satisfied), provided that the assumption $\sup_{\mathcal{X}\times\mathcal{U}} ||\mathcal{O}_t(\tilde{\mathbf{x}}_t, \tilde{\mathbf{u}}_t) - g(\mathbf{x}_t, \mathbf{u}_t)||_2 = O_p(r_t)$ holds. It can be shown that this condition can be satisfied when $||\mathcal{O}_t(\tilde{\mathbf{x}}_t, \tilde{\mathbf{u}}_t)||_2 \leq ||g(\mathbf{x}_t, \mathbf{u}_t)||_2$.

Theorem 8.1 *For the trained Kriging model $\mathcal{O}_t(\tilde{\mathbf{x}}_t, \tilde{\mathbf{u}}_t)$ and the true unknown disturbance model $g(\mathbf{x}_t, \mathbf{u}_t)$, with mean 0 and unknown variance, if $||\mathcal{O}_t(\tilde{\mathbf{x}}_t, \tilde{\mathbf{u}}_t)||_2 \leq ||g(\mathbf{x}_t, \mathbf{u}_t)||_2$, it holds that $\sup_{\mathcal{X}\times\mathcal{U}} ||\mathcal{O}_t(\tilde{\mathbf{x}}_t, \tilde{\mathbf{u}}_t) - g(\mathbf{x}_t, \mathbf{u}_t)||_2 = O_p(r_t)$.*

Proof. Recall from Chapter 7 that the learning system used in Kriging satisfies the lack of bias condition, i.e. $E[\mathcal{O}_t(\tilde{\mathbf{x}}_t, \tilde{\mathbf{u}}_t)] = E[g(\mathbf{x}_t, \mathbf{u}_t)] = \boldsymbol{\mu}$ (it is assumed that $\boldsymbol{\mu} = \mathbf{0}_{8\times 1}$). Also, note that Kriging estimation can only vary within a bounded interval. On the other hand, it is impossible to ensure the boundedness of the unknown model $g(\mathbf{x}_t, \mathbf{u}_t)$. So, it is always true that the variances of the outputs of $\mathcal{O}_t(\tilde{\mathbf{x}}_t, \tilde{\mathbf{u}}_t)$, say $\sigma_1^2, \sigma_2^2, ..., \sigma_8^2$, are less than or equal to the variances of the corresponding outputs of $g(\mathbf{x}_t, \mathbf{u}_t)$. Also, the variance of the Kriging $\mathcal{O}_t(\tilde{\mathbf{x}}_t, \tilde{\mathbf{u}}_t)$ estimates are finite, i.e. $0 \leq \sigma_1^2, \sigma_2^2, ..., \sigma_8^2 < \infty$. Indeed, for the trained model, $||\boldsymbol{\sigma}||_2 = 4.55 \times 10^{-4} << 1$ (see Table 7.1), where $\boldsymbol{\sigma} = (\sigma_1^2, \sigma_2^2, ..., \sigma_8^2)^T$. Now, it can be indicated that:

$$
\begin{aligned}
\exists \epsilon > 0, \exists r_t, \quad & \mathcal{P}r(||\mathcal{O}_t(\tilde{\mathbf{x}}_t, \tilde{\mathbf{u}}_t) - g(\mathbf{x}_t, \mathbf{u}_t)||_2 > r_t \cdot \epsilon) \\
& = \mathcal{P}r(||\mathcal{O}_t(\tilde{\mathbf{x}}_t, \tilde{\mathbf{u}}_t) - E[\mathcal{O}_t(\tilde{\mathbf{x}}_t, \tilde{\mathbf{u}}_t)] + E[g(\mathbf{x}_t, \mathbf{u}_t)] - g(\mathbf{x}_t, \mathbf{u}_t)||_2 > r_t \cdot \epsilon) \\
& \leq \mathcal{P}r(||\mathcal{O}_t(\tilde{\mathbf{x}}_t, \tilde{\mathbf{u}}_t) - E[\mathcal{O}_t(\tilde{\mathbf{x}}_t, \tilde{\mathbf{u}}_t)]||_2 + ||E[g(\mathbf{x}_t, \mathbf{u}_t)] - g(\mathbf{x}_t, \mathbf{u}_t)||_2 > r_t \cdot \epsilon) \\
& \leq \mathcal{P}r(||\mathcal{O}_t(\tilde{\mathbf{x}}_t, \tilde{\mathbf{u}}_t) - E[\mathcal{O}_t(\tilde{\mathbf{x}}_t, \tilde{\mathbf{u}}_t)]||_2 > r_t \cdot \frac{\epsilon}{2}) \ .
\end{aligned}
$$

The proof can be completed by using the extended multivariate version of Markov's inequality, known as Ferentinos inequality [198] and taking into account that $||\boldsymbol{\sigma}||_2 < 1$, as below:

$$
\mathcal{P}r(r_t^{-1}||\mathcal{O}_t(\tilde{\mathbf{x}}_t, \tilde{\mathbf{u}}_t) - E[\mathcal{O}_t(\tilde{\mathbf{x}}_t, \tilde{\mathbf{u}}_t)]||_2 > c \cdot ||\boldsymbol{\sigma}||_2) \leq \frac{1}{c^2} \leq \frac{1}{(c \cdot ||\boldsymbol{\sigma}||_2)^2} \ .
$$

This means that for $\epsilon = 2c \cdot ||\boldsymbol{\sigma}||_2$ and $\exists r_t$, it holds that:

$$
\mathcal{P}r(r_t^{-1}||\mathcal{O}_t(\tilde{\mathbf{x}}_t, \tilde{\mathbf{u}}_t) - g(\mathbf{x}_t, \mathbf{u}_t)||_2 > \epsilon|| < \frac{1}{\epsilon^2} \ . \tag{8.1}
$$

Based on *Definition 6.5*, Eq. 8.1 implies that $\sup_{\mathcal{X}\times\mathcal{U}} ||\mathcal{O}_t(\tilde{\mathbf{x}}_t, \tilde{\mathbf{u}}_t) - g(\mathbf{x}_t, \mathbf{u}_t)||_2 = O_p(r_t)$. Hence, the proof is complete. □

To satisfy the robust feasibility and robust constraint satisfaction theorems (*Theorem 6.1* and *Theorem 6.2*), it is necessary to determine matrixes ϕ and ψ, and calculate $\boldsymbol{\theta}$ such that the inequality constraint $\mathbf{u}[\boldsymbol{\theta}_t] = K\mathbf{x}_t + (\phi - K\psi)\boldsymbol{\theta}_t$ be satisfied. For matrixes A and B defined in Section 6.6 of Chapter 6, obtained by values reported in Table 3.1, the null-space of the 8×10 matrix $[(A - \mathbb{I}_n) - B]$ is a set containing the following vectors:

$$v_1 = (-5.08 \times 10^{-6}, -5.08 \times 10^{-6}, -1.33 \times 10^{-8}, -1.33 \times 10^{-8}, -5.24 \times 10^{-6}, -5.24 \times 10^{-6}, -9.57 \times 10^{-10}, -9.57 \times 10^{-10}, 1, 2.41 \times 10^{-11})$$

$$v_2 = (-1.33 \times 10^{-8}, -1.33 \times 10^{-8}, -5.04 \times 10^{-6}, -5.04 \times 10^{-6}, -9.57 \times 10^{-10}, -9.57 \times 10^{-10}, -5.24 \times 10^{-6}, -5.24 \times 10^{-6}, -2.43 \times 10^{-11}, 1)$$

Now, matrixes ϕ and ψ can be calculated such that *range* $([\psi^T\ \phi^T]^T)^T$ includes v_1 and v_2, as follows:

$$\psi = \begin{bmatrix} 5.08 & 1.33 \times 10^{-2} \\ 5.08 & 1.33 \times 10^{-2} \\ 1.33 \times 10^{-2} & 5.04 \\ 1.33 \times 10^{-2} & 5.04 \\ 5.24 & 9.57 \times 10^{-4} \\ 5.24 & 9.57 \times 10^{-4} \\ 9.57 \times 10^{-4} & 5.24 \\ 9.57 \times 10^{-4} & 5.24 \end{bmatrix}, \quad \phi = \begin{bmatrix} -10^{-6} & 2.43 \times 10^{-5} \\ 2.43 \times 10^{-5} & -10^{-6} \end{bmatrix}.$$

The control decision variables can vary within the range $-4 \times 10^{-6} \leq \theta_1 \leq 4 \times 10^{-6}$ and $-4 \times 10^{-6} \leq \theta_2 \leq 4 \times 10^{-6}$ to get feasible actuation signals $\mathbf{u}[\boldsymbol{\theta}_t]$. The horizon length N is chosen to be 10, and the tunable weight matrixes of objective function defined in Eq. 6.10 are selected as: (a) R is a 2×2 diagonal matrix with diagonal elements 10^{-7}, (b) Q is a 8×8 diagonal matrix with diagonal elements 1, and (c) T is a 8×8 diagonal matrix with diagonal elements 10. Also, to ensure the stability of the system through satisfying *Theorem 6.7*, the time-varying 8×8 Lyapunov matrix P is determined by solving the discrete-time Lyapunov equation, defined in Eq. 6.11. The desired trajectory is determined using reinforcement learning (RL) combined with a graph theoretic based approach called maximum likelihood estimation of trajectories in a Markov chain (MLE-TMC). For detailed theories and algorithmic structure of the method, one can refer to Chapter 5.

The importance of the above setup is that the theoretical requirements for (1) robust constraint satisfaction, (2) robust feasibility, (3) Lyapunov stability of system dynamics, (4) epi-convergence of oracle, and (5) robust asymptotic stability of feed-back based control command are satisfied.

The other important issue which should be mentioned refers to the strategies taken to cope with the 5 sources of uncertainty (SU-1 to SU-5) mentioned in Chapter 3. SU-1 which represents the driver's behavior, and appears as the variation of cruise speed on road, is important because after the prediction of future road profile for the front tire, one can determine the time that the rear tire passes the predicted profile as a function of speed (see Eq. 5.1 and description therein). This makes sense because the road profile remains the same along the road, and instead of using two road prediction modules for both tires, all one needs is to do the prediction once (for the front tire) and calculate the time of passing the profile for rear tire as a function of cruise speed. For the estimation of vehicle speed, an absorbing state stochastic process is taken into account. It was theoretically proven in

Chapter 5 that the considered concept is a realistic scenario for the estimation of speed under uncertainty. For our simulation, whenever the deflection of front rear (z_{b_1}) exceeds ± 0.04 (can be viewed as a bump / pothole), a random perturbation of speed is imposed to vehicle, and the considered algorithm is automatically activated to model the driver's behavior under uncertainty. For SU-2 which stands for the number of passengers and load, a logical range of 580 kg to 850 kg with uniform distribution is taken into account. So, for each independent simulation, $\hat{M}_b \overset{d}{\sim} Unif(580, 850)$ is drawn randomly and the simulation is conducted accordingly, and the statistical results are reported at the end. For SU-3 which deals with model-plant mismatch, a relatively same approach is followed. In this way, independent normal distributions are defined for all of the non-zero / non-unit elements of matrixes A_{ave} and B_{ave}. Let's say $\hat{a}$ is such an element in A_{ave}, then it follows the distribution $\hat{a} \overset{d}{\sim} N(a_{ave}, 0.01a_{ave})$. So, for each independent simulation, a value is drawn from this distribution, the simulation is performed, and the statistical results are reported. For SU-4 which represents the randomness of road roughness, a thorough comprehensive study using different statistical forecasting methods was conducted in Chapter 4. The importance of using such a strategy is that unlike conventional controllers which use a predefined unrealistic road profile for simulation, the proposed strategy uses experimental data for training the road roughness forecasting method. Recall from Chapter 6 that $D\mathbf{r}_t$ represents the effect of road roughness in state-space model, i.e. $\tilde{\mathbf{x}}_{t+1} = \hat{A}\tilde{\mathbf{x}}_t + \hat{B}\tilde{\mathbf{u}}_t + D\mathbf{r}_t + \mathcal{O}_t(\tilde{\mathbf{x}}_t, \tilde{\mathbf{u}}_t)$. For SU-5 which represents the uncertainty due to the malfunction of sensors and actuators, a comprehensive statistical analysis was carried out using Bayesian dynamic programming and bootstrap, and an empirical cumulative density function (CDF) was obtained to quantify the uncertainty. This source of uncertainty was used to develop the polytope $\mathcal{W}$ which was then used at the heart of MRPIS algorithm for the calculation of invariant set Ω.

As optimal controller, LBMPC uses an optimization module to calculate the control command. For the current study, the implemented quadratic objective function is solved using Newtons method (NM), golden sectioning search (GSS) [199], and simulated annealing (SA) [200]. GSS can be viewed as a deterministic direct search method which does not use the derivatives for the calculation of optimal solution. Also, it does not have any hyper-parameter, and its implementation is straight-forward. Once the bounds of the solution domain are defined, GSS deterministically converges to a solution (usually a local solution). Thanks to its simple and efficient structure, GSS can be used as a reliable technique for real-time optimization of the control objective function of LBMPC. It is worth mentioning that GSS handles a multivariate optimization problem cooperatively by optimizing decision variables one by one. SA is a single agent metaheuristic optimization algorithm which performs a chain of stochastic jumps in the solution domain until converging to a

solution. Upon fine tuning, SA has the potential of global optimization. To run SA, the initial temperature of 10, final temperature of 0.001, and cooling factor of 0.9996 are used. To jump to a new point, Gaussian proposal distribution with variance 0.1× (upper bound − lower bound) is considered. The acceptance-rejection sampling method [201] is used to decide whether the proposal (new solution) should be accepted.

The block-diagram of the control paradigm implemented in Simulink software is shown in Figure 8.1. The figure provides a schematic view of the parameters and modules involved to get the results. As seen, the control paradigm includes desired trajectory generation module (the orange block), road roughness prediction module (the magenta block), vehicle speed estimation module (the yellow block), LBMPC controlling module (the green block), and a high-fidelity nonlinear model of suspension system (the red block). The controlling commands calculated in LBMPC are fed to the coupled high-fidelity nonlinear model to get the true state values of the suspension system. As mentioned, the states measured by LBMPC and those of nonlinear model are acquiesced in an archive, which is used for validating the veracity of the control-oriented model.

To have a better view regarding the performance of LBMPC, a number of well-known control algorithms are also taken into account, and are applied to the same vehicle suspension system. The methods are extended iterative learning proportional-integral-derivative (EIL-PID) [202], sky-hook controller (SHC) [203], sliding mode controller (SMC) [38], dynamic matrix controller (DMC) [160], dynamic matrix PID (DM-PID) [204], constraint variable structure controller (CVSC) [205], and dynamic matrix CVSC (DM-CVSC) [58]. From the considered controllers, EIL-PID, SHC, SMC, and CVSC are continuous, non-predictive and non-optimal controllers, and DMC, DM-PID, DM-CVSC, and LBMPC are discrete, predictive and optimal controllers, capable of constraint satisfaction. DM-CVSC and DM-PID calculate the optimal commands using sequential quadratic programming (SQP), since the control law results in nonlinear equality constraint when written in the standard quadratic format. DMC calculates the optimal commands using standard quadratic programming (QP). The horizon length N is chosen to be 10 for all predictive controllers. The state and control input constraints for optimal controllers are the same as those of LBMPC, and the initial condition for all rival controllers are $\mathbf{x}_0 = \mathbf{0}_{8\times1}$ and $\mathbf{u}_0 = \mathbf{0}_{2\times1}$. Since the considered rival controllers are not equipped with any strategy / module for the prediction of road roughness, one should use a predefined road roughness to conduct the simulation. It is a deep-seated tradition among automotive control engineers to use sine shaped road profiles for simulation. For the current simulation, a predefined road profile together with 4 different types of disturbances are considered [58]. The considered simulation scenarios have different features, and can be viewed as acceptable approximations of road roughness.

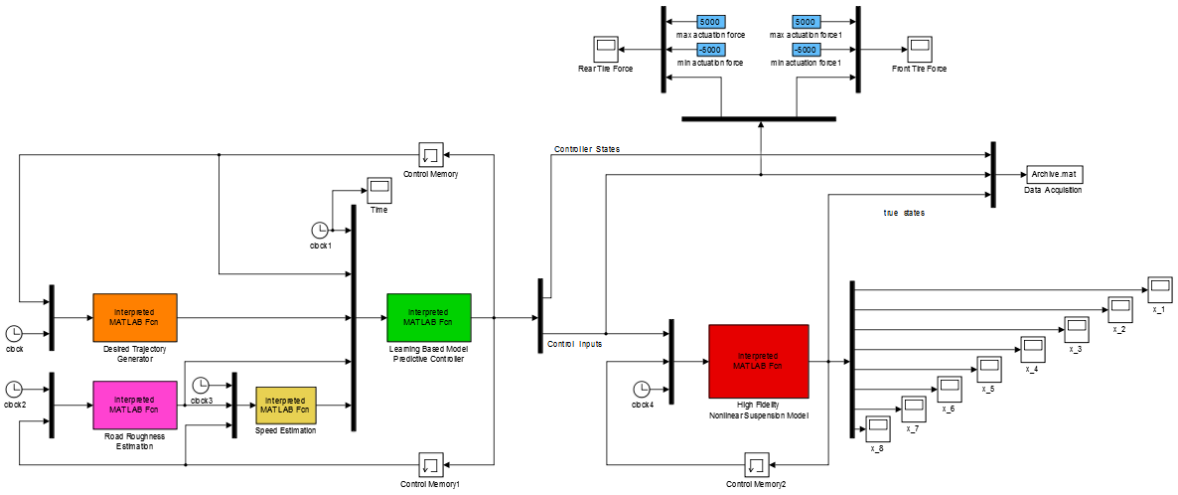

Figure 8.1: Block-diagram of the LBMPC coupled with a nonlinear high fidelity model of suspension system in Simulink software.

To have a better understanding on the performance of the considered methods and to have a fair comparison with LBMPC, at one stage of the numerical experiments, the road roughness forecasting module is replaced with this 4 predefined simulation cases and the results are reported together with those of the conventional controllers. This lets us evaluate the power of learning module at the heart of LBMPC, since the other methods are not learnable. Thereafter, the simulation is carried out using LBMPC equipped with both learning module and road roughness prediction module to show the power of LBMPC in yielding more realistic results, compared to conventional controllers which use predefined road profiles. Also, note that, unlike LBMPC, the considered controllers neither estimate the vehicle speed on road nor use an online desired trajectory building mechanism. So, the simulation is performed, considering the rival controllers as state regulators (to make the body deflection 0 on road).

A number of performance evaluation metrics are considered to compare the performance of the controllers. The first metric which evaluates the potential of controllers in minimizing the body mass deflection is called the sum of the center of gravity of body mass vertical displacement. The second metric is the time required for making the body displacement 0. This is important since a reliable controller should be powerful enough to withstand against external excitations and stabilize the vehicle body to provide comfort for passengers. The third and fourth metrics are called the maximum and average absolute value of actuation

force. These metrics are important as they represent the maximum transient effort of actuators. The less the value of the metrics, the greater the life cycle of actuator. The fifth metric is the mean squared trajectory tracking error, which obviously should be minimized.

The simulation for the 4 artifact cases are conducted for 5 *sec* with sampling time 10^{-3} *sec*, which results in 5000 working points for discrete controllers. For Waterloo road, the simulation is conducted for 10 *sec*. In particular, a segment of road from 70^{th} second to 79^{th} second of Figure 4.5 is chosen.

A spline smooth fit with high precision is fitted to the selected segment to capture 5000 working points. To be consistent with the results of artifact scenarios, the 5000 data points are used with step-size of 0.001 to get 5 seconds of simulation. So, notice that the data are extracted from portion time of $[70, 79]$ *sec* and are mapped to a $[0, 5]$ *sec* interval, without changing the shape of the original profile. For clarification, the performed transformation is shown in Figure 8.2.

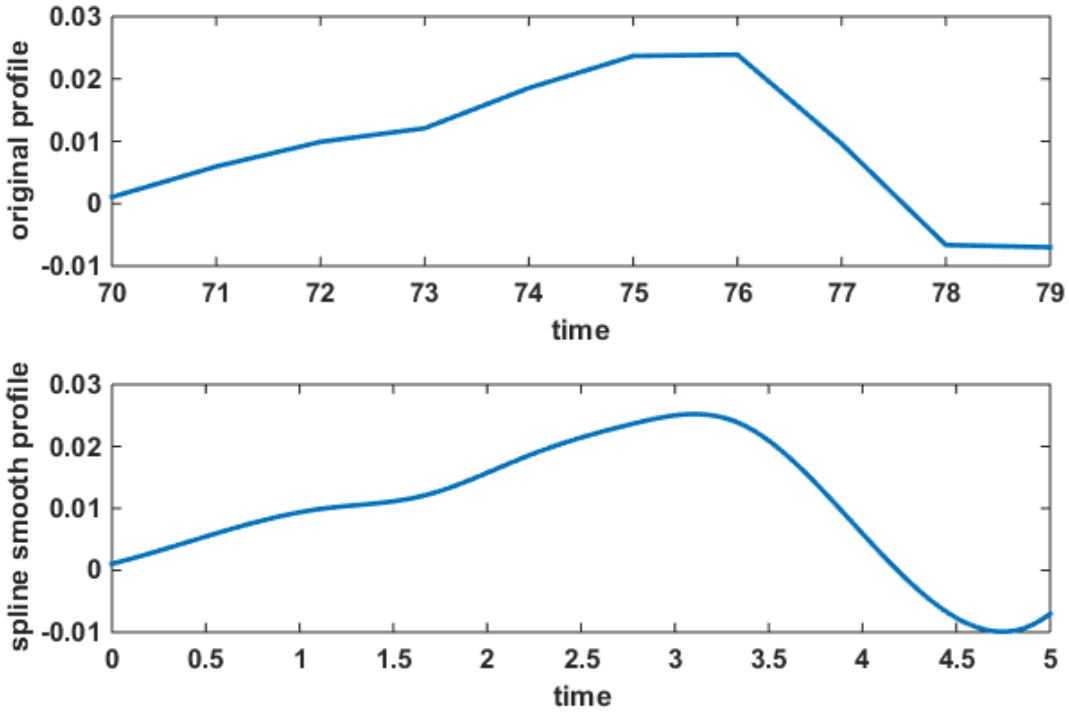

Figure 8.2: Waterloo road roughness data used for simulation: (a) original, (b) spline smooth version.

As seen, unlike the artifact profile which is used by conventional controllers [58], the profile is not a stable profile, which is expectable, since the data comes from the real road. Given the properties of the data, it can be understood that it is not logical to implement the control problem as a regulation problem, since we cannot make the displacement 0. But, the better way is to implement the control problem as a tracking problem with the goal of making the displacement of vehicle body mass stable (making the variation of displacement 0). This is indeed the idea behind the implementation of LBMPC.

All of the simulation are conducted in MATLAB and R software on a Pentium IV DELL laptop, with Windows 7 operating system, Intel Dual core 2.2 GHz, and 2 GBs RAM.

8.2 Simulation Results

At the first stage of the experiment, a comparative study is carried out using the adopted performance evaluation metrics to find out which of the considered optimization methods is more powerful to be used at the heart of LBMPC. Apparently, two conflicting factors play part when comparing the performance of optimization methods for real-time applications. These factors are the calculation speed and the efficiency of an optimization method. Among the considered methods, NM uses a gradient searching strategy, while GSS and SA are deterministic and stochastic direct search methods, respectively. Note that the simulation is conducted on the 4 artifact cases and Waterloo road roughness data, as explained in Chapter 4. The sum of body mass center displacement for the considered cases are listed in Table 8.1. It can be realized that LBMPC-NM outperforms the other LBMPC versions for the 4 artifact cases. This is expected since the NM algorithm uses gradient information to converge to a solution, and can be viewed as a very powerful method, provided that the optimization problem be well-defined. For the Waterloo scenario in which the considered road segment is not stable, it was observed that NM cannot satisfy the constraints and converge to an infeasible solution. This indicates the importance of using a robust gradient-free optimization method which can explore the objective function landscape even when the optimal control problem is demanding. All in all, the results indicate that NM surpasses the direct search methods when the control problem is well-defined and steady; however, for a challenging road profile NM may converge to irrelevant / infeasible actuation signals. Thus, direct search methods could be more useful when using LBMPC for real road profiles. Among the direct search optimizers, the best performance belongs to SA. The simulation details indicated that GSS cannot compete with SA since it coped with the multivariate optimization problem in a cooperative fashion by optimizing the decision variables one by one.

Table 8.1: Sum of body mass center's displacement over the control period

	Case 1	Case 2	Case 3	Case 4	Waterloo
LBMPC-SA	0.0806	0.0822	0.0840	0.0946	**0.1459**
LBMPC-GSS	0.1043	0.0926	0.0979	0.0951	0.1750
LBMPC-NM	**0.0370**	**0.0579**	**0.0411**	**0.0375**	–

Such a feature enforces GSS to converge to premature solution especially when the optimization problem is non-separable. Unlike GSS, SA optimizes all of the decision variables at the same time, and thus, can manage to converge to an acceptable solution even when the objective landscape is complicated (multi-modal) and non-separable. Also, another reason behind the superior performance of SA over GSS is its stochastic direct searching mechanism which gives it a chance to jump from a premature solution to a more qualified solution. The simulation also indicated that NM algorithm requires more computational time to yield the optimum solutions compared to SA and GSS. This is probably because of the need for the calculation of first and second order differentiations which takes more time compared to direct searching methods.

Table 8.2 indicates the required portion of time for the regulation of the vehicle body mass displacement states (z_{b_1} and z_{b_2}). Notice that the reported values are the exact points in which the body mass displacement states fall within a predefined manifold with center 0 and radius ϵ (which is a very little number). This is the same as the concept of trajectory tracking with predefined funnels as defined in [160]. As mentioned before, the Waterloo road data cannot be considered as a regulation problem, since the road profile is unstable. For the 4 artifact cases, it can be seen that LBMPC-NM and LBMPC-SA are capable to make the displacement 0, and LBMPC-GSS fails to stabilize the rear body mass. As expected, the best performance for these 4 cases belongs to LBMPC-NM. Having said that the results of LBMPC-SA are close to those of LBMPC-NM.

Table 8.2: Required portion of control time for the regulation of the body displacement states

	Case 1	Case 2	Case 3	Case 4	Waterloo
			Front actuator		
LBMPC-SA	2.55 / 5	2.77 / 5	2.65 / 5	2.65 / 5	–
LBMPC-GSS	2.55 / 5	4.86 / 5	2.99 / 5	2.71 / 5	–
LBMPC-NM	**1.37 / 5**	**1.41 / 5**	**1.40 / 5**	**1.38 / 5**	–
			Rear actuator		
LBMPC-SA	2.90 / 5	2.92 / 5	2.65 / 5	2.61 / 5	–
LBMPC-GSS	–	–	–	–	–
LBMPC-NM	**1.37 / 5**	**1.38 / 5**	**1.40 / 5**	**1.38 / 5**	–

Table 8.3 and Table 8.4 list the maximum absolute actuation force and mean absolute actuation force of the rival LBMPC variants, respectively. As seen, all of the methods reach the maximum 5000 N actuation force during the control period. It can be discerned that for the 4 artifact cases, the lowest mean absolute actuation force belongs to LBMPC-NM and LMBPC-GSS. Note that the results of LBMPC-SA is comparable to the other two methods with respect to this metric. Most importantly, LBMPC-SA surpasses both LBMPC-NM and LMBPC-GSS for Waterloo simulation. Hence, LBMPC-SA was found to be a reliable method since it not only yields close results to LBMPC-NM, but also could handle the Waterloo simulation scenario.

Table 8.5 lists the results of the LBMPC variants in terms of mean squared trajectory tracking error. It can be seen that for all of the cases, the best performance belongs to LBMPC-SA. Given the importance of this metric, and also the results obtained with respect to the other metrics, it can be concluded that SA is the best optimization method to be used at the heart of LBMPC for the current case study.

Table 8.3: Performance of LBMPC with different optimizers in terms of max $|\mathbf{u}|$ metric

	Case 1	**Case 2**	**Case 3**	**Case 4**	**Waterloo**
			Front actuator		
LBMPC-SA	5000	5000	5000	5000	5000
LBMPC-GSS	5000	5000	5000	5000	5000
LBMPC-NM	5000	5000	5000	5000	–
			Rear actuator		
LBMPC-SA	5000	5000	5000	5000	5000
LBMPC-GSS	5000	5000	5000	5000	5000
LBMPC-NM	5000	5000	5000	5000	–

Table 8.4: Performance of LBMPC with different optimizers in terms of mean$|\mathbf{u}|$ metric

	Case 1	**Case 2**	**Case 3**	**Case 4**	**Waterloo**
			Front actuator		
LBMPC-SA	501.26	482.37	457.48	450.41	**4107.60**
LBMPC-GSS	**349.56**	**350.89**	**350.89**	**350.84**	4744.10
LBMPC-NM	428.85	493.37	841.61	469.02	–
			Rear actuator		
LBMPC-SA	543.65	546.60	547.23	543.50	**4069.10**
LBMPC-GSS	1911.20	1284.70	1083.60	1212.81	4699.80
LBMPC-NM	**363.57**	**449.30**	**461.06**	**391.54**	–

Table 8.5: Performance of LBMPC with different optimizers in terms of tracking error

	Case 1	Case 2	Case 3	Case 4	Waterloo
			Front actuator		
LBMPC-SA	**5.62e-05**	**5.21e-05**	**5.45e-05**	**5.92e-05**	**8.19e-04**
LBMPC-GSS	7.03e-05	7.12e-05	6.98e-05	7.50e-05	1.00e-03
LBMPC-NM	7.15e-05	7.19e-05	6.90e-05	7.58e-05	–
			Rear actuator		
LBMPC-SA	**3.40e-05**	**3.81e-05**	**3.39e-05**	**5.02e-05**	**7.99e-04**
LBMPC-GSS	1.85e-04	9.52e-05	8.50e-05	7.40e-05	9.83e-04
LBMPC-NM	8.22e-05	8.20e-05	7.58e-05	8.15e-05	–

From the computation speed point of view, SA can be considered as an acceptable method since unlike most of the existing stochastic heuristic search methods, it only uses a single agent to search the solution domain and mostly relies on probabilistic jumps to converge to a solution rather than time consuming interaction which takes place among agents of population based / swarm-based metaheuristics.

After realizing the best optimization method to be used at the heart of LBMPC, the simulation is continued by comparing the performance of LBMPC with the conventional control methods on the 4 considered artifact scenarios. Recall that the 4 cases use the same road profile for the simulation. The difference among the cases is the disturbance that contaminates the road profile information, and makes the control problem noisy. This can be viewed as a good mean for evaluating the robustness and stability of the adopted control methods, and comparing them to LBMPC. Since, the goal of the simulation is to compare the performance of the controllers (regardless of the optimization algorithm used at their heart), the results for LBMPC are reported considering both NM and SA methods.

Table 8.6 lists the sum of body mass center displacements for the rival controllers as well as the passive model. Obviously, the successful controllers in terms of this metric are SHC, CVSC, DM-CVSC and LBMPC-NM. By paying a much precise attention to the obtained results, one can easily infer that, except for *Case 4*, the performance of LBMPC-NM is close to the other successful controllers. Given the results, one can infer that LBMPC-NM and DM-CVSC are superior to the other optimal predictive controllers for almost all of the cases. It is worth pointing out that DM-CVSC and DM-PID are nonlinear controllers, and require a considerable computational time as well as a complicated optimization formulation to get the optimal solutions.

Table 8.7 tabulates the required portion of time for the rival controllers to regulate the vehicle body mass displacement. LBMPC-NM remarkably surpasses the rival methods in terms of the considered metric. It can be seen that unlike all of the other rival controllers,

for all of the cases, LBMPC-NM successfully regulates the vehicle displacement. DM-CVSC takes the second place among the other rival controllers. It seems that SHC could regulate the displacement of front body mass, but fails to stabilize the rear part. Also, CVSC could regulate the displacement of the rear part and fails to stabilize the front body mass. By taking a precise look into the results, one can realize that the performance of LBMPC-NM and DM-CVSC are the same for *Case 1* and *Case 4*, but DM-CVSC fails to regulate for *Case 3* and also shows remarkably inferior result for *Case 2*.

Table 8.6: Sum of body mass center's displacement over the control period

	Case 1	**Case 2**	**Case 3**	**Case 4**
Passive	0.0817	0.0817	0.0817	0.0817
SHC	0.0251	**0.0304**	**0.0252**	0.0252
EIL-PID	0.0472	0.0491	0.0502	0.0492
CVSC	0.0643	0.0439	0.0516	**0.0114**
SMC	0.0314	0.0905	0.0718	0.0232
DMC	0.0238	0.4350	0.2718	0.0825
DM-PID	0.0189	0.0985	0.1136	0.0371
DM-CVSC	**0.0166**	0.0476	0.1060	0.0232
LBMPC-NM	0.0370	0.0579	**0.0411**	0.0375

Table 8.7: Required portion of control time for the regulation of the body displacement states

	Case 1	**Case 2**	**Case 3**	**Case 4**
		Front actuator		
SHC	4.79 / 5	4.97 / 5	4.52 / 5	4.59 / 5
EIL-PID	2.73 / 5	–	–	2.55 / 5
CVSC	4.61 / 5	–	–	–
SMC	4.62 / 5	–	–	–
DMC	**1.37 / 5**	–	–	–
DM-PID	**1.37 / 5**	–	–	–
DM-CVSC	**1.37 / 5**	4.92 / 5	–	**1.38 / 5**
LBMPC-NM	**1.37 / 5**	**1.41 / 5**	**1.40 / 5**	**1.38 / 5**
		Rear actuator		
SHC	–	–	–	–
EIL-PID	2.96 / 5	–	–	2.60 / 5
CVSC	4.71 / 5	4.94 / 5	–	–
SMC	4.83 / 5	–	–	–
DMC	1.38 / 5	–	–	–
DM-PID	1.38 / 5	–	–	–
DM-CVSC	**1.37 / 5**	4.91 / 5	–	**1.38 / 5**
LBMPC-NM	**1.37 / 5**	**1.38 / 5**	**1.40 / 5**	**1.38 / 5**

It can be concluded that LBMPC-NM beats the other methods in terms of this metric, which can be enumerated as one of the most important metrics (if not the most important one).

Table 8.8 and Table 8.9 list the maximum absolute actuation force and mean absolute actuation force of the rival LBMPC variants, respectively. It can be inferred from the results that EIL-PID and DM-PID show the superior performance with respect to this metric. This suggests the usefulness of defining a proportional-integral-derivative type control rule for calculating the actuation force, especially when minimizing the control effort / energy is of highest importance. Also, it can be seen that DM-CVSC shows acceptable results in terms of these two metrics, and can compete with DM-PID. Apparently, LBMPC-NM cannot compete with DM-PID and DM-CVSC and only beats SMC and CVSC with respect to this performance metric. Note that although this metric is important for the current case study, but the controllers were developed with the goal of minimizing the vehicle body displacement to provide ride comfort. Indeed, there is a conflicting trade-off between this metric and ride comfort.

Table 8.8: Performance of LBMPC with different optimizers in terms of max $|\mathbf{u}|$ metric

	Case 1	**Case 2**	**Case 3**	**Case 4**
		Front actuator		
Passive	–	–	–	–
SHC	–	–	–	–
EIL-PID	4112	4354	4263	4433
CVSC	4552	4851	3871	4208
SMC	4702	4109	4351	4870
DMC	5000	4883	5000	4978
DM-PID	5000	4452	5000	5000
DM-CVSC	3424	4236	4005	3664
LBMPC	5000	5000	5000	5000
		Rear actuator		
Passive	–	–	–	–
SHC	–	–	–	–
EIL-PID	4837	4657	4012	4211
CVSC	4699	4966	4399	4785
SMC	4650	4709	4553	5534
DMC	4966	5000	5000	4968
DM-PID	5000	5000	4755	5000
DM-CVSC	2975	3730	4856	4949
LBMPC	5000	5000	5000	5000

Table 8.9: Performance of LBMPC with different optimizers in terms of mean|**u**| metric

	Case 1	**Case 2**	**Case 3**	**Case 4**
		Front actuator		
Passive	–	–	–	–
SHC	–	–	–	–
EIL-PID	14.37	**19.77**	**28.33**	**32.98**
CVSC	680.94	730.98	1590.00	1188.2
SMC	409.72	566.11	1198.90	2004.60
DMC	148.13	147.51	293.01	145.46
DM-PID	**7.85**	50.44	271.06	46.24
DM-CVSC	39.51	76.62	205.93	53.91
LBMPC-NM	428.85	493.37	841.61	469.02
		Rear actuator		
Passive	–	–	–	–
SHC	–	–	–	–
EIL-PID	12.16	**16.30**	**12.90**	**14.33**
CVSC	681.73	1669.90	441.69	345.69
SMC	447.64	1140.1	388.59	550.14
DMC	141.23	190.10	222.84	137.92
DM-PID	**8.55**	84.14	57.64	27.67
DM-CVSC	19.97	115.25	148.77	56.24
LBMPC-NM	363.57	449.30	461.06	391.54

Since, LBMPC outperforms the other methods in terms of ride comfort (see Table 8.7), it can still be considered as a successful controller even if it uses greater (yet feasible) actuation force compared to the other predictive controllers for the considered problem.

Table 8.10 lists the obtained results in terms of the mean squared tracking error. The reported results indicate that the performance of all of the considered methods are close to each other in terms of this metric, and most of them yield the best result at-least for one of the cases. All in all, the best performance belongs to DM-CVCS, and CVSC. This is mainly because of the tracking power of the controlling rule implemented at the heart of CVSC and DM-CVSC, which defines a sliding surface near the desired trajectory and calculates the actuation signals such that the controller steers the system towards the desired trajectory. SHC, LBMPC-SA and SMC also show acceptable results for this metric and beat the other methods for 2, 1, and 1 cases, respectively. To summarize the results reported so far, the number of times a given controller beats the other methods (the bold numbers in the above tables) are reported in Table 8.11. It is obvious that LBMPC with 10 times of success and DM-CVSC with 8 times of success outperform the other methods. It is important mentioning that, unlike LBMPC, DM-CVSC is a nonlinear controller, and requires solving a more sophisticated optimization problem to get the results.

Table 8.10: Performance of the rival controllers in terms of mean squared tracking error

	Case 1	Case 2	Case 3	Case 4
		Front actuator		
SHC	1.56e-05	2.58e-05	**1.57e-05**	**1.59e-05**
EIL-PID	1.50e-05	1.86e-05	1.69e-05	1.73e-05
CVSC	4.99e-05	3.09e-05	1.55e-04	4.92e-05
SMC	**1.29e-05**	**1.43e-05**	7.47e-05	1.41e-04
DMC	2.79e-05	5.00e-04	3.57e-04	5.48e-05
DM-PID	1.56e-05	6.42e-05	1.32e-04	3.03e-05
DM-CVSC	1.32e-05	2.95e-05	7.57e-05	2.71e-05
LBMPC-SA	5.62e-05	5.21e-05	5.45e-05	5.92e-05
		Rear actuator		
SHC	7.55e-05	9.28e-05	7.58e-05	7.62e-05
EIL-PID	8.18e-05	7.11e-05	8.08e-05	8.17e-05
CVSC	5.03e-05	1.09e-04	1.84e-05	**2.17e-05**
SMC	1.66e-05	5.04e-05	**1.16e-05**	8.11e-05
DMC	2.51e-05	6.36e-04	2.59e-04	6.71e-05
DM-PID	1.55e-05	9.44e-05	5.19e-05	3.71e-05
DM-CVSC	**1.26e-05**	**3.33e-05**	5.38e-05	**2.17e-05**
LBMPC-SA	3.40e-05	**3.81e-05**	3.39e-05	5.02e-05

Table 8.11: Performance of the rival controllers in terms of mean squared tracking error

SHC	**EIL-PID**	**CVSC**	**SMC**	**DMC**	**DM-PID**	**DM-CVSC**	**LBMPC**
2	6	2	3	1	3	8	10

In line with such a fact, the conducted simulation indicated that LBMPC calculates the control commands quite faster than DM-CVSC, and thus, is more appropriate for real-time applications.

For further evaluation of the performance of LBMPC, and for complementing the comparative study, some graphical tools are used to visualize the performance of the rival controllers. Figure 8.3 depicts the front and rear body mass vertical displacements for all of the considered cases using non-predictive, non-optimal controllers. It can be seen from the figure that for most of the cases, all of the considered controllers can regulate the states. One of the interesting observations pertains to the acceptable performance of SHC for all of the disturbance scenarios. It seems that, as a semi-active controller, SHC can neatly resist against the effect of unmeasured disturbances, and its controlling rules can regulate the considered states for most of the cases. Furthermore, it seems that CVSC and SMC show relatively the same performance. The displacement profiles under EIL-PID control have relatively the same shape as passive scenarios. However, the amplitude of the

resulting sinusoidal-like state profile is less than the passive one, and the considered states reach zero in a shorter period of time.

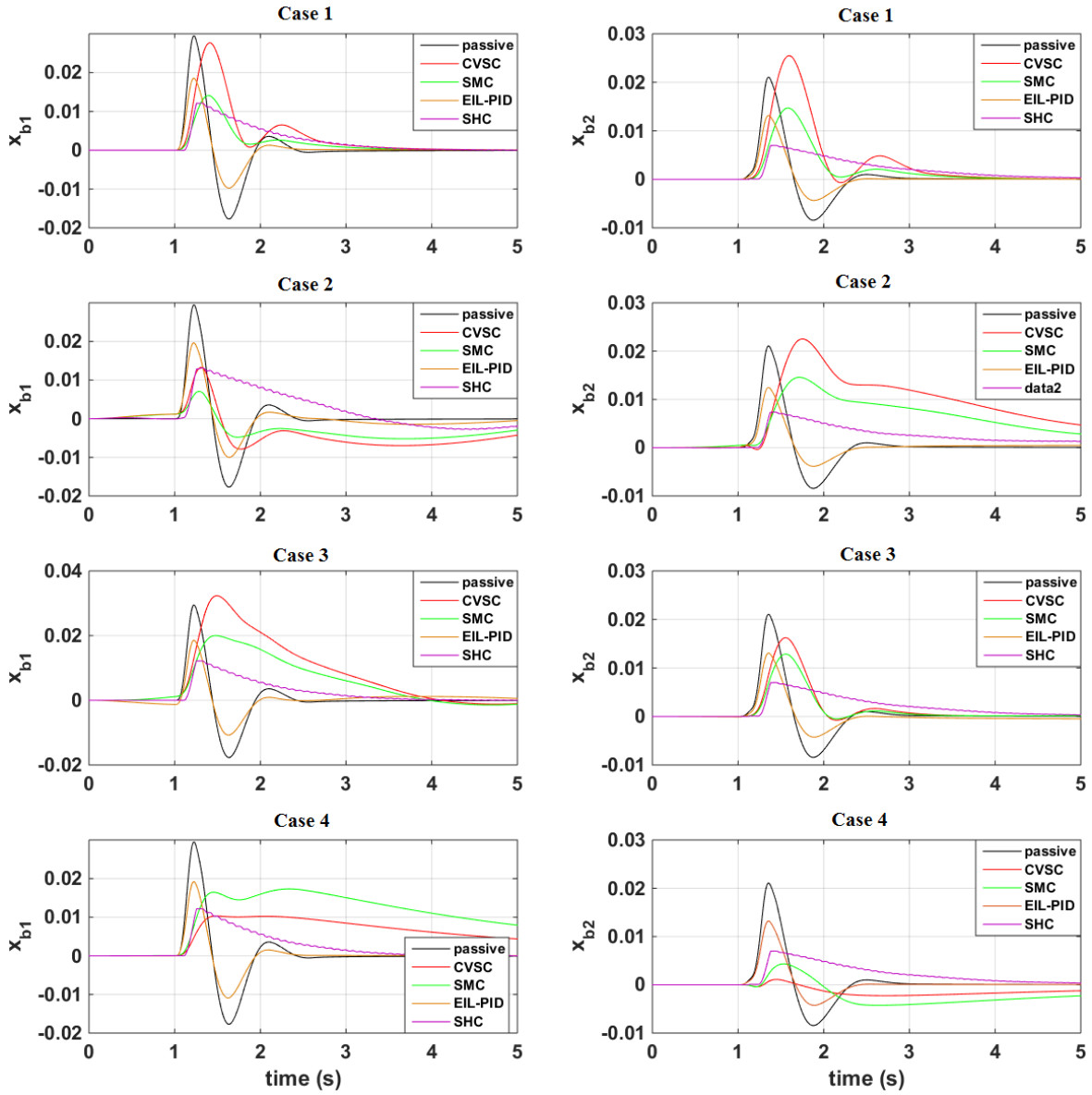

Figure 8.3: Displacements of front and rear body mass for the non-predictive / non-optimal controllers for the four case studies.

The main problem of the above non-optimal controllers is that they cannot guarantee the constraint satisfaction. This is because there is no module in their structure which takes either an optimization problem or a set of constraints. The only method which has an instinct conservativeness to indirectly handle the constraints is CVSC. However, it cannot still be considered as a method that can guarantee the constraint satisfaction. This is a fact that the control problem becomes more complicated when there is a constraint on the actuation force, and a remarkable effort should be made by a controller to yield feasible solutions, especially when a significant rate of disturbance contaminates the external information (for example the road profile plus noise scenario). Figure 8.4 illustrates the actuation force computed by SMC and LBMPC for *Case 4*. It can be seen that the constraint is violated when using SMC, which results in an infeasible solution, since the actuator used in the suspension system has a limited maneuverability and cannot process the control commands when the force is greater than 5000 or less than -5000. This is the main problem with the non-optimal controllers which makes their reliable applicability to the considered problem a bit questionable.

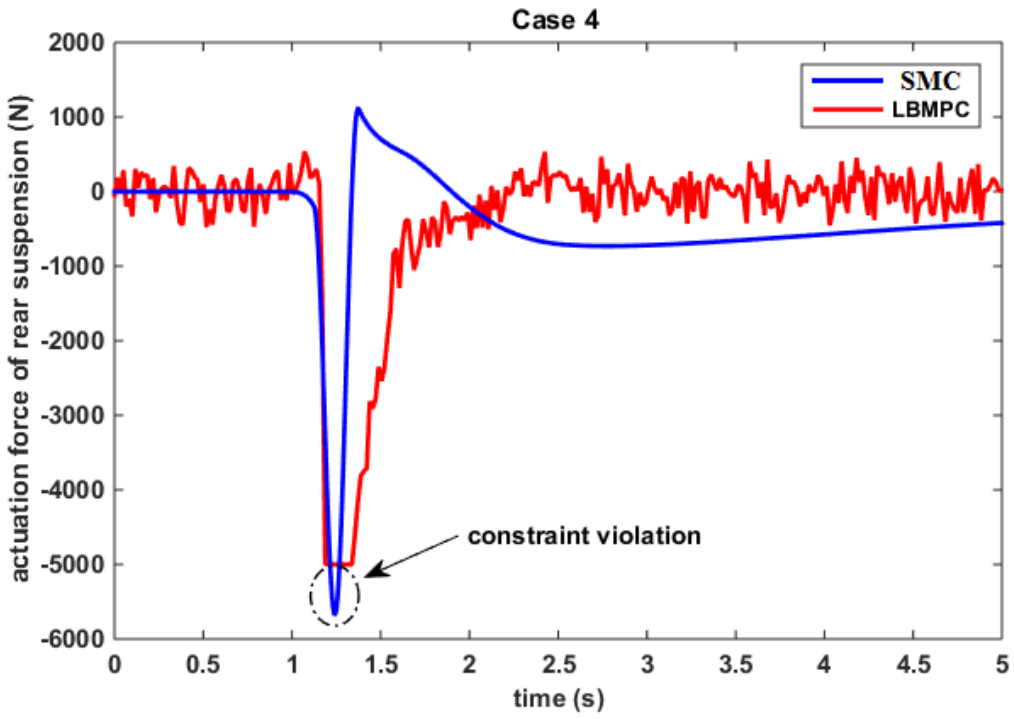

Figure 8.4: Constraint handling for LBMPC and SMC for *Case 4*.

Figure 8.5 indicates the front and rear body mass vertical displacements obtained by predictive controllers. The results indicated that all of the methods successfully satisfied the constraints. It can be seen that LBMPC-NM is the only method which yields a smooth profile, and the other controllers have significant fluctuations, and emphasize on the abrupt regulation of the displacement. It will be indicated that this feature undermines the performance of DMC, DM-PID and DM-CVSC significantly when the profiles are contaminated with noises.

Figure 8.6, Figure 8.7, and Figure 8.8 indicate the obtained results for the rest of the simulation scenarios. It becomes apparent that DMC, DM-PID and DM-CVSC sacrifice the ride comfort in favor of obtaining feasible control solutions in the presence of external disturbance. However, LBMPC retains its quality (because of using stabilizable feedback gain for calculating the control commands) for all of the cases, and regulates the vehicle displacement successfully.

Also, LBMPC is the only optimal controller which yields smooth displacement profiles, and the displacement resulting from the actuation signal of other optimal controllers are non-smooth with abrupt changes during the control period. The findings is in agreement with the theoretical proofs pertaining to the guaranteed stability of LBMPC. This is more apparent in *Case 2* and *Case 3*, where the solutions of the other optimal controllers are not stable at all.

Another observation pertinent to the above figures is the robustness of LBMPC. As mentioned, the 4 scenarios consider the same road profile plus different sources of disturbances. It is apparent that the solutions obtained by LBMPC are very close for the 4 cases, meaning that LBMPC successfully rejects the external disturbances, identified the road profile, and calculate the controlling commands accordingly.

This is when the other optimal controllers are remarkably affected when the road information is contaminated with noises, and cannot calculate the correct control signal, which results in a very non-smooth and improper displacement of vehicle body mass. The resistance of LBMPC against disturbances augurs the veracity of theoretical findings regarding the guaranteed stability of LBMPC. All in all, it can be concluded that LBMPC is the most qualified method among the considered controllers for handling the suspension control problem.

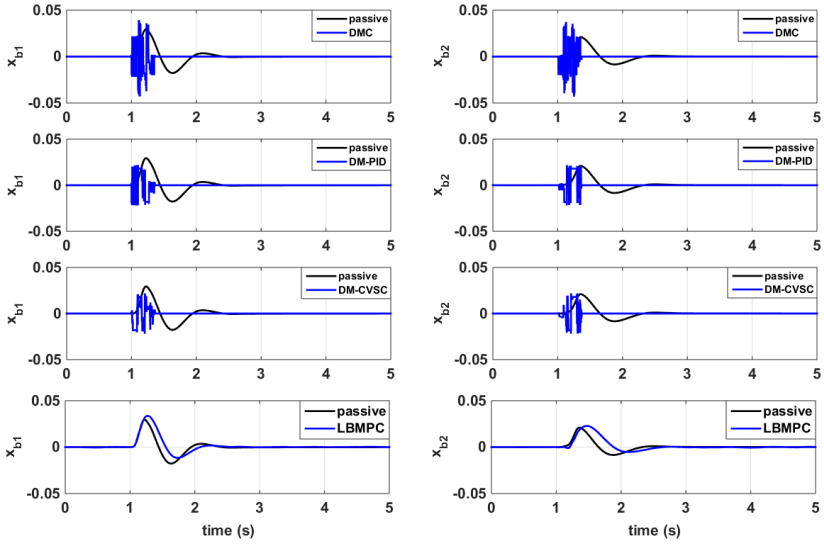

Figure 8.5: Displacements of front and rear body mass for predictive controllers, *Case 1.*

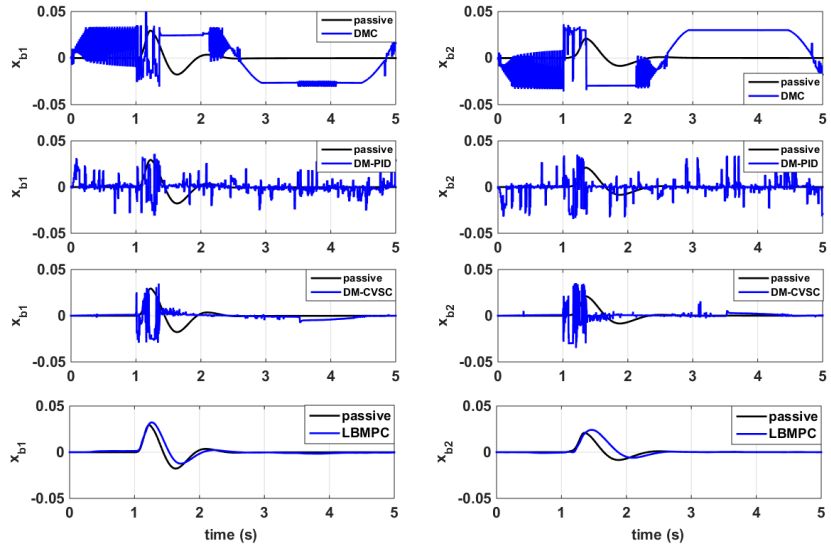

Figure 8.6: Displacements of front and rear body mass for predictive controllers, *Case 2.*

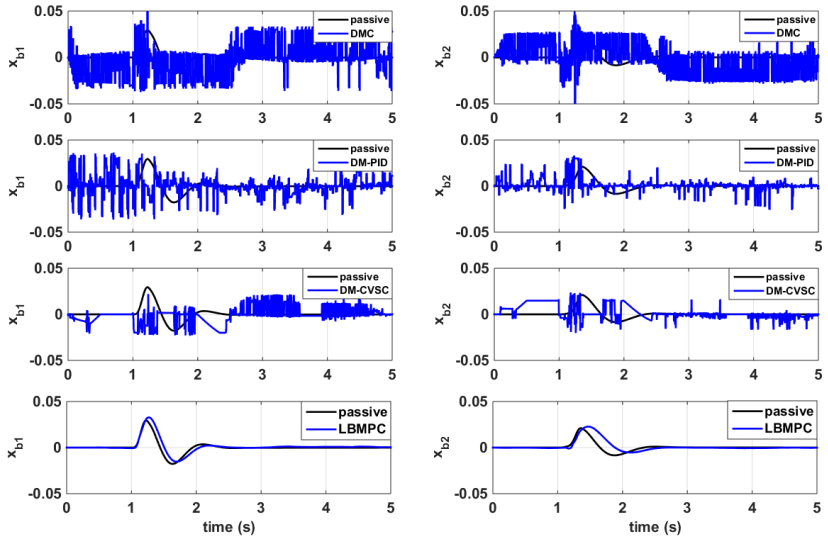

Figure 8.7: Displacements of front and rear body mass for predictive controllers, *Case 3*.

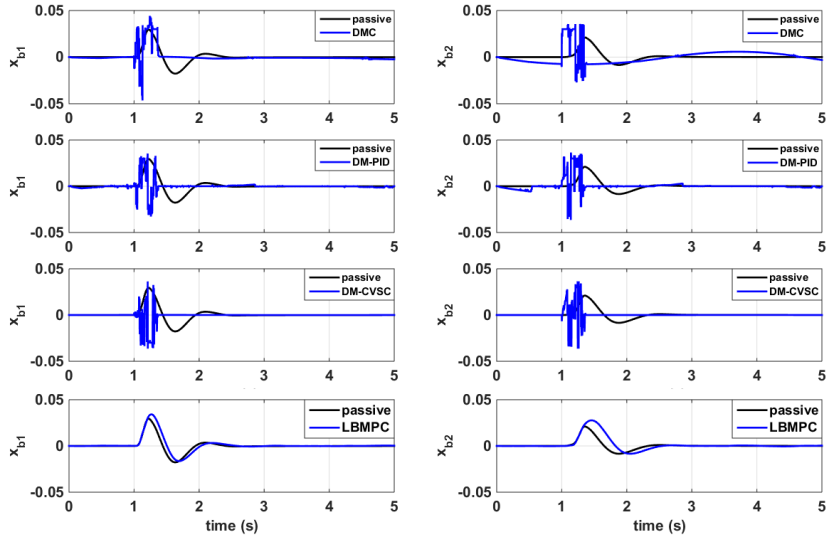

Figure 8.8: Displacements of front and rear body mass for predictive controllers, *Case 4*.

Another importance issue which is the main idea behind the design of LBMPC is the use of statistical learning tools to forecast the road roughness. It is indeed the most prominent aspect of LBMPC compared to the conventional controllers. As mentioned, since the considered optimal and non-optimal controllers are not equipped with a module for forecasting the road roughness, the only choice is to use a predefined road profile and conduct the simulation. For the current simulation, the author used one of the most applicable road profiles (which is considered as a standard base) for calculating the control commands. Figure 8.9 indicates the Waterloo road profile, the predefined road profile, and the road profile forecasted by ARMA(2, 3) at the heart of LBMPC. It is obvious that the solution obtained by LBMPC can be far more realistic, since it uses a very good approximation of the real road profile for calculating the control command. It can be also observed that how much inaccurate is the standard basic predefined road profile compared to the real road profile.

Figure 8.10 indicates the complete state values obtained by LBMPC for Waterloo road. As seen, LBMPC can successfully stabilize all of the states after the first significant upward bump at 0.8 *sec*. Note that at time 4.5 *sec*, the vehicle experiences the second remarkable road disturbance (downward pothole) which is apparently at the end of the considered road segment, and LBMPC does not have any time to stabilize the states at this segment.

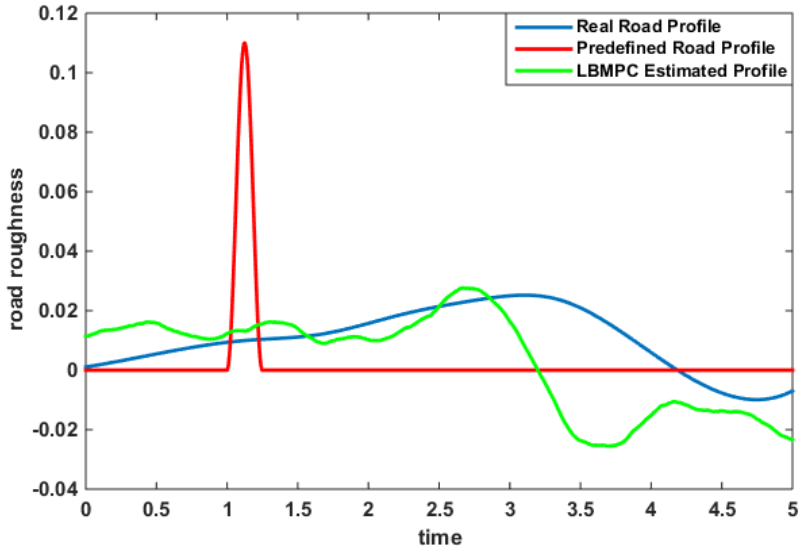

Figure 8.9: Comparison of (a) Waterloo road profile, (b) predefined road profile used by conventional controllers, and (c) LBMPC road profile forecasted by ARMA(2, 3).

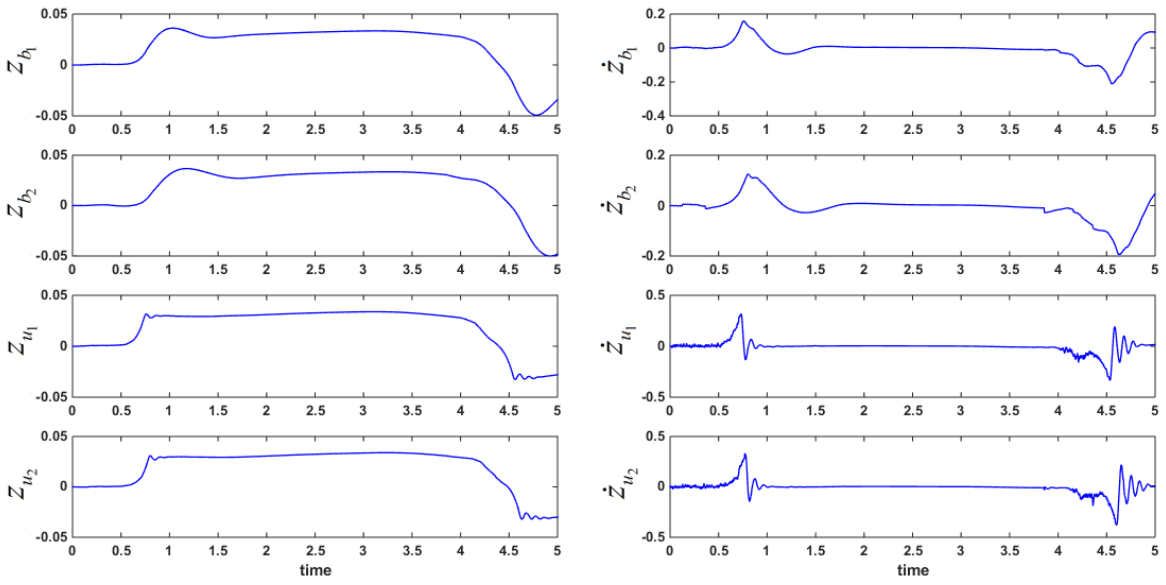

Figure 8.10: Variation of state values of LBMPC for Waterloo road.

It is apparent that the variations of un-sprung and sprung masses (the four right hand side sub-plots) become 0 shortly after encountering the first significant road disturbance, which indicates the stabilization power of LBMPC when used on road.

The last stage of the experiment deals with the power of LBMPC-SA to withstand against the variation of passengers and loads (SU-2) and model-plant mismatch (SU-3). To do so, the uncertain parameters are drawn 10 times from the specified distributions, and the simulation are conducted accordingly. Figure 8.11 indicates the graphical visualization of drawn samples for clarification of their diversity. Since matrix A has randomly drawn elements, its final form is depicted as $2D$ color maps. Note that the 0 and 1 arrays remain the same (which correspond to black and white arrays). It can be seen that the resulting color maps have different gray cells and represent a diverse set of the quantifications of model-plant mismatch. Also, the values for vehicle body mass are diversely drawn from uniform distribution, and cover the considered range. So, this test bed can reliably give us information on how well LBMPC can resist against SU-2 and SU-3.

The statistical results obtained for the considered performance metrics are presented in the form of box-plots in Figure 8.12. By a precise look into the obtained results, one can realize that the range of variation of the metrics is minor, and the box plots are dense. It can be seen that only the mean absolute actuation force of the front body mass has outliers,

and the other metrics always vary within a small range over 10 independent simulation. The findings are in a good agreement with the robustness of LBMPC, and it seems that the variation of SU-2 and SU-3 have negligible impact on the performance of LBMPC.

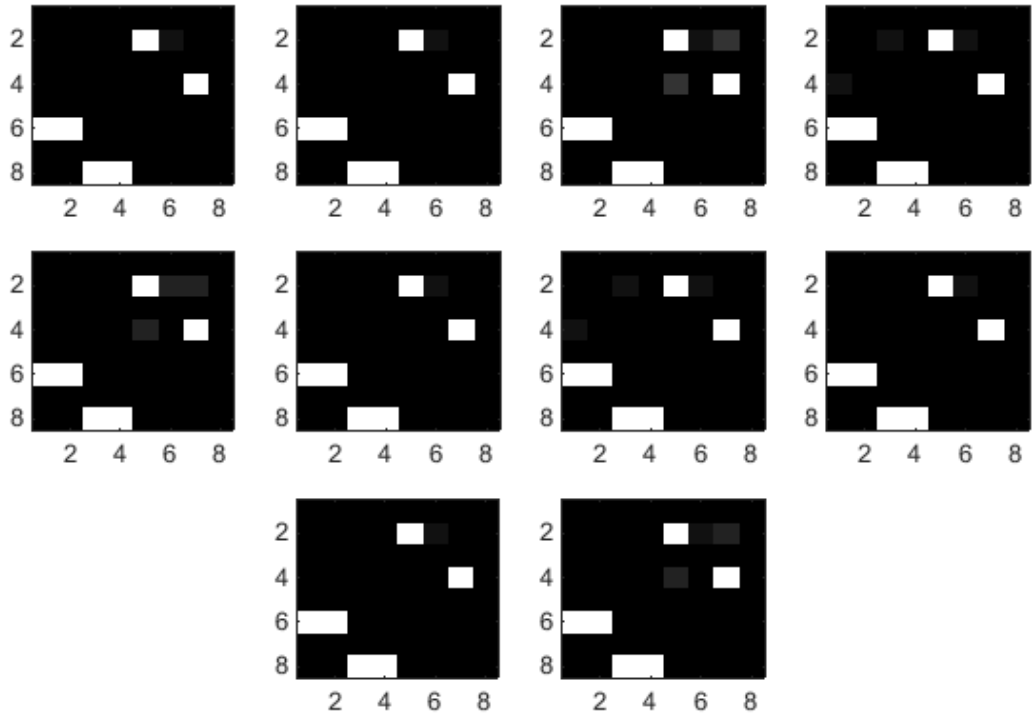

Figure 8.11: Variation the drawn samples (a) the variation of matrix A is shown in the form of color map, and (b) the variation of body mass drawn from a uniformly distribution, shown in a $2D$ field foe better visualization.

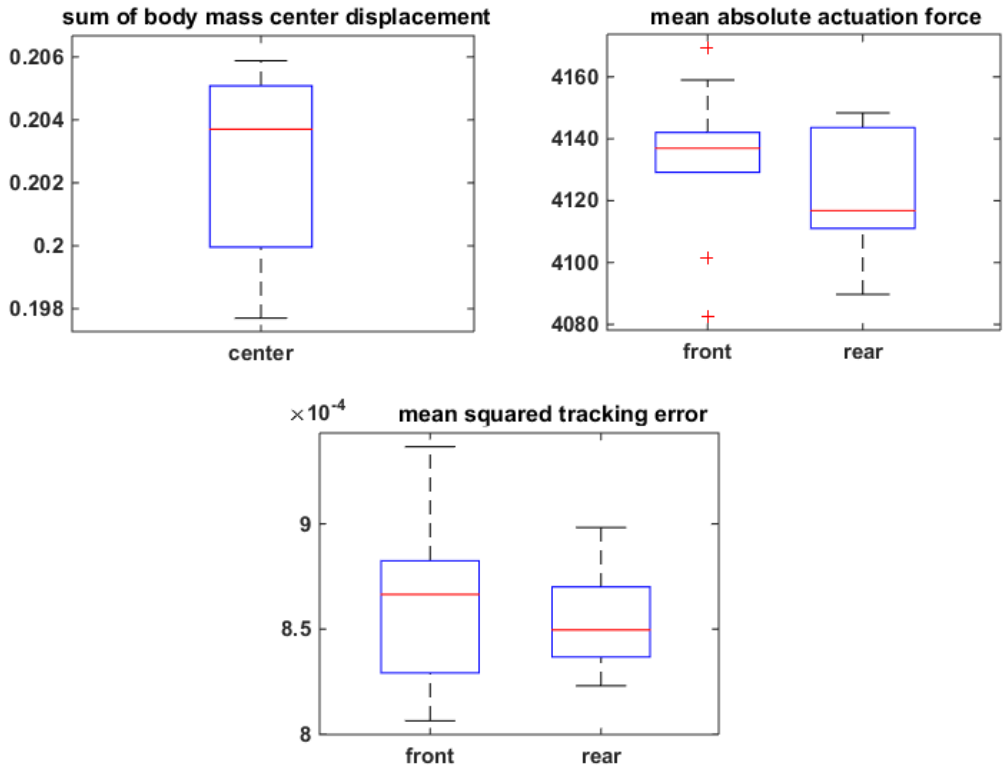

Figure 8.12: Statistical results in terms of the performance metrics obtained over 10 independent simulation.

In the next sub-section, the controlling commands obtained by LBMPC are fed to the true nonlinear model to find out whether the controller can reliably stabilize the real suspension system.

8.3 Validation Using Nonlinear Model

After calculating the control commands using LBMPC, a validation test is conducted to evaluate the precision of the piece-wise linear model (plus disturbance) at the heart of LBMPC with the true nonlinear model of the plant. Figure 8.13 reveals the states of the true nonlinear model as well as those of LBMPC obtained by feeding the calculated actuation signals to the system.

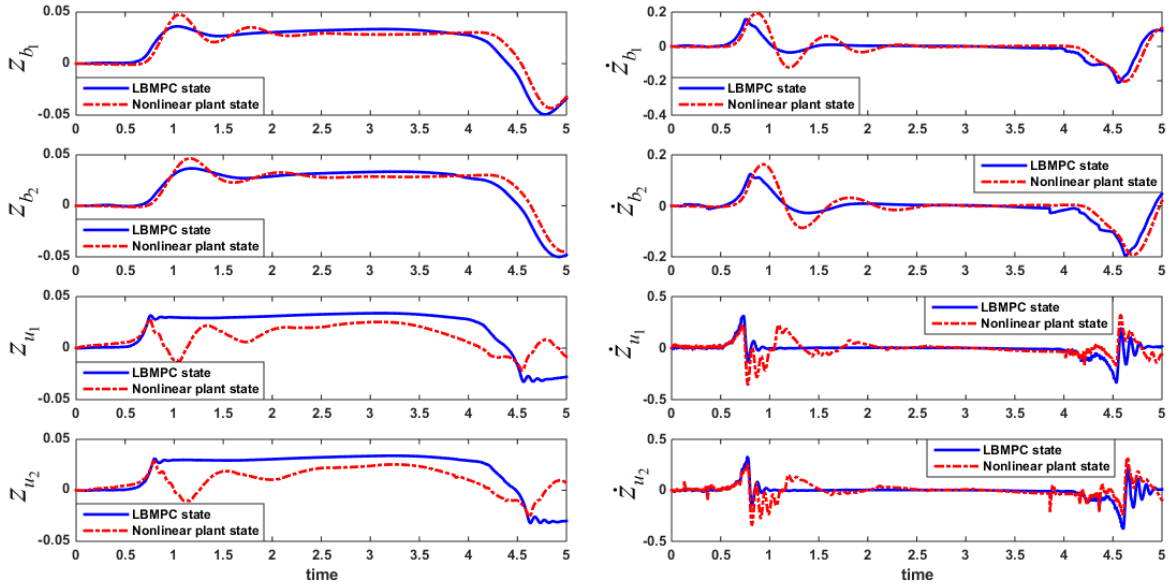

Figure 8.13: Comparison of the states of true nonlinear model (plant model) with the states of piece-wise linear model used in LBMPC.

The obtained results are very promising and indicate the high precision of the state-space model used for calculating the commands of LBMPC. The results also endorse the statistical experiments performed in Chapter 7 to quantify the uncertainty in the linear model due to the un-modelled dynamics. Moreover, the results verify the authenticity of using LBMPC for stabilizing the real-plant. It can be seen that when the states measured by LBMPC become stable, the corresponding states of the true plant model are also stable. All in all, the findings of the model verification are totally in favor of using LBMPC for real-time control of vehicle suspension system.

8.4 Pros and Cons of LBMPC

Based on the conducted simulation, the following remarks are observed:

1. The conducted simulation considering LBMPC with different optimization modules indicated that the controlling commands can be obtained using both gradient-based and direct searching techniques. However, the selection of the proper optimization

algorithm depends on the properties of the control problem at hand. For vehicle suspension control, it was realized that LBMPC with NM can afford efficient control commands, provided that the objective function be convex, unimodal with straightforward constraints. However, by exposing LBMPC with NM optimizer to more tricky suspension control scenarios, the simulation indicated that NM could easily converge to infeasible solutions. Also, the simulation results unveiled that GSS is only proper when the underlying optimal control problem be separable (decision variables be independent). Among the considered optimization scenarios, LBMPC with SA optimizer was found to be the most appropriate choice, since it could successfully satisfy the constraints, and also converge to a near optimal solution (due to its stochastic direct searching strategy), especially when the objective function was multimodal, non-separable with different types of constraints. Also, because of using a single agent, SA was quite fast, and a good fit for real-time applications.

2. It was theoretically and empirically indicated that the oracle at the heart of LBMPC can predict the uncertainties due to the state measurement error, and increase the robustness of the control commands. Also, by applying LBMPC to different types of road roughness scenarios, it was observed that it could successfully reject the external disturbances and calculate the proper controlling commands to stabilize the suspension system. This was not the case for the considered rival controllers, as they fail to manage the suspension system when the road profile information was contaminated with noises. Also, it was observed that, non-optimal controllers could violate the operating constraints, while the optimal ones (including LBMPC) always calculated the control commands such that the actuation signal remain within the feasible bound.

3. By defining distributions for SU-2 (uncertainty of the number of passengers and load) and SU-3 (uncertainty due to model-plant mismatch), drawing samples from them, and repeating the simulation, it was indicated that LBMPC could successfully withstand against these sources of uncertainty. The results were shown in the form of box plots, which were dense and had no outliers (except for one out of five performance measures). The robustness of LBMPC against these two sources of uncertainty guarantees that it has enough potential to be mounted in the vehicle for real-time hardware in the loop application.

4. The conducted simulation demonstrated the usefulness of the proposed forecasting system for the online forecasting of road roughness. It is a deep-seated tradition to use a predefined road profile for conventional controllers and calculate the control commands accordingly. By comparing one of the most applicable standard profiles with

the Waterloo road profile and also with the road roughness forecasted by LBMPC, it was proven that how superior LBMPC is in terms of calculating the commands based on a more realistic profile. In particular, the road roughness forecasted by ARMA (2, 3) at the heart of LBMPC was very close to the Waterloo road profile, which resulted in realistic control commands.

5. The results of LBMPC model validation versus true nonlinear model endorsed the reliability of the calculated control commands for stabilizing the plant (real suspension system). It was observed that when the states measured by LBMPC become stable, the corresponding states of the true nonlinear model also become stable. The results of model validation proved the potential of LBMPC for the safe stabilization of the true suspension system on road.

Note

All MATLAB codes pertaining to the simulation performed in this chapter can be found in Appendix.

Chapter 9

Conclusions and Future Work

In this chapter, the general concluding remarks obtained by simulation are presented. Also, several insights into the future research potentials of LBMPC are pointed out. It is tried to mention different aspects of this powerful and flexible controller, and also to discuss why it can be a good fit for emerging control problems arising in automotive industry, in particular for designing autonomous vehicles. Also, the compatibility of the architecture of LBMPC with one of the most interdisciplinary and sophisticated fields of science, known as artificial intelligence (AI), is analyzed.

9.1 Conclusions

Based on the conducted simulation, the following conclusions were derived regarding the potential of LBMPC for the safe control of vehicle suspension system:

1. The results of simulation conducted in Chapter 4 indicated that statistical forecasting methods and their related model diagnosis tools provided simple yet efficient means for the real-time forecasting of road profiles. It was observed that most of the considered forecasting models possessed at-most 2 to 4 structural parameters, and could be tuned in a short period of time. Also, one can manage to tune such models in a way that their forecasted values fall within a bounded interval with 95% confidence, which is quite desirable when they are used at the heart of a controller where the boundedness of estimation results in a stable and safe control performance. Also, it was observed that Bayesian forecasting methods have a very good potential

to be used for forecasting the road roughness profiles. The salient asset of such techniques is that they can be always improved by collecting more information about the road roughness, and choosing more appropriate prior distributions for the system parameters.

2. The simulation results obtained in Chapter 5 indicated that the probabilistic method proposed for determining the most likely desired trajectory can afford reliable outcomes. In particular, the conducted simulation indicated that whenever the vehicle passes a bump / pothole, where significant oscillations were expected, the proposed method steered the controller to send significant actuation signals to suspension system for damping the vibration. Thereafter, the desired trajectory imposed a safe oscillation around equilibrium point until the vehicle displacement became 0. Also, from computational viewpoint, it was observed that the graph theoretic model used for the calculation of the most probable trajectory could afford the optimal solution in a short period of time with low computational effort, provided that the graph information including anchor walk, mean-weight cycle cover, and free cycles be available. Also, the absorbing state stochastic process proposed for vehicle speed estimation was capable of producing chains which converged to one of the absorbing states within the predetermined control horizon length (N) in a short period of time. Such a feature made the proposed method compatible with the predictive objective function used at the heart of LBMPC for estimating the vehicle speed in an online fashion.

3. The results of simulation obtained in Chapter 7 indicated that the combination of dynamic programming and Bayesian uncertain models could afford a powerful tool for quantifying the uncertainty of un-modelled dynamics in approximated piece-wise linear control state-space model. The salient asset of such probabilistic tools was that they could efficiently compensate the lack of knowledge in the true control state-space model by replacing the un-modelled dynamic term (which could be either linear or nonlinear) with additive uncertain noise. Also, it was realized that bootstrap could be used to come up with a rich dataset for the efficient quantification of uncertainty which influences suspension system's dynamics. Furthermore, the validation tests indicated that the non-parametric cumulative distribution function obtained from bootstrap simulation was accurate due to the richness of the obtained dataset, which was then used for determining a polytope that encapsulates the uncertainty with 95% confidence. The importance of the statistical analysis was that this was the first time that a thorough statistical analysis was done for the accurate quantification of suspension system's state measurement error. The other important simulation conducted in Chapter 7 was the design of efficient oracle for LBMPC. In particu-

lar, comparison was conducted using different statistical and soft machine learning tools considering a wide range of performance evaluation metrics. The results demonstrated the superiority of statistical methods over soft methods, due to their accuracy and interpretable structure. Among the considered methods, ordinary Kriging was selected to be used as the oracle of LBMPC. Also, the boundedness and continuity of Kriging allowed the author to theoretical prove the epi-convergence of Kriging, as a non-parametric oracle (which was an important factor of the oracle used at the heart of LBMPC).

4. The simulation conducted in Chapter 8 indicated that the structure of LBMPC is flexible enough to accept both gradient-based and direct-search heuristics for the calculation of control commands. This was found to be an important feature since, due to the no free lunch theorem, each control problem can be efficiently handled by a certain optimization algorithm. One of the other interesting findings was the ability of LBMPC to reject different types of noises from the upcoming road profile information, and come up with stable and robust control commands. The results of comparative study indicated that traditional controllers were incapable of handling this important task, mainly because of the lack of theoretical foundation to guarantee the robustness of the control commands. Also, the simulation results derived the author to the conclusion that the use of forecasting module at the heart of LBMPC made the controller more efficient than the conventional controllers which calculate the control commands based on a predefined standard road profile. Indeed, it was observed that the road profile estimated by LBMPC (using ARMA(2, 3) for the current study) was very close to the real profile, and accordingly resulted in realistic control commands. Most importantly, the validation test between the piece-wise linear model and the true nonlinear model proved the reliability of using LBMPC for the safe stabilization of vehicle suspension system on road.

All in all, it can be concluded from the results of the current study that LBMPC could afford accurate, stable, and robust control commands even if the model at the heart of LBMPC was uncertain. Also, the comprehensive study conducted in this thesis demonstrated the efficacy of using statistical learning and forecasting tools for both quantification of uncertainty and forecasting the upcoming road profile in a real-time fashion.

Moreover, the findings of the current thesis answered some of the challenging open problems within the realm of vehicle suspension control. In particular, it was demonstrated that by combining the well-established theories for designing optimal, stable and robust controllers with the theoretical achievements within the realm of statistical learning, one can develop an efficient and robust predictive controller which accurately forecasts the

future behavior of the system, as well as the upcoming road roughness to come up with control commands which stabilize the displacement of vehicle body mass on road, and accordingly provide ride comfort.

9.2 Towards Autonomous Vehicle Design

In this sub-section, it is tried to draw the attention of readers to a very attractive research field within the realm of automotive industry, called autonomous systems [206]. Undoubtedly, designing fully automated mechatronic machines have been initiated in the last decades of 20^{th} century, when engineers have exerted a lot of effort to make their optimistic vision of designing smart robots a reality. Providing a detailed review of the chronological advancements and encountered challenges for designing autonomous robots is beyond the scope of this thesis, and thus, the interested readers are referred to seminal review papers [207, 208]. It is sensible for most of the people living in the 21^{st} century that autonomous robots are playing important servicing roles all around the world. A recent report indicates that smart / autonomous robots and machines are even prone to take further steps and handle very complex tasks without any human intervention [209]. At the engineering level, what attained a considerable interest reclines in the much efficient performance of autonomous robots, such as probe robots and mining robots. Indeed, after the invention and successful utilization of such robots, it has been unveiled that not only they are capable of completing the objects with high precision, but also have lots of benefits from economical and life hazard viewpoints. As an example, using autonomous firefighting robots can afford better results, and at the same time, it can be ensured that there is no life-hazard threat. The feedback of initial tests even augurs a possibility to completely replace human with autonomous robots for handling such tasks. Interestingly, automotive and robotic industries share considerable technological similarities, due to the fact that aerial/ground vehicles and robots are sort of mechatronics systems. The astonishing progresses on designing autonomous robots have elevated the expectation and awareness of people around the world, and have indirectly inflicted a meaningful pressure on automotive industrialists to consider designing autonomous vehicles as a grand strategy [210].

It is also quite understandable that designing fully-autonomous vehicles requires more technical efforts and much advanced technological infrastructures, due to the complicated architecture of vehicles compared to most of the existing robotic systems. This is why even the current significant achievements attained by autonomous robotics researchers can be viewed as a starting stage for the successful completion of designing a fully-autonomous vehicle. As a result, autonomous engineers are now trying to design cooperative cruise

system for vehicles hoping to gradually turn this semi-automated vehicle system to a smart fully automated one [211]. There are several key factors which make designing a reliable and practically feasible autonomous vehicle so challenging. It is worth pointing out that the potential challenges are abound. Here, the most important ones are scrutinized.

1. Modern vehicles are complex mechatronics systems comprising different components which should work cooperatively. Any delay in encoding and decoding of controlling commands and other operating signals can make entire system instable. Also, it should be ensured the independent controlling units handling each of the control objectives, such as fuel consumption control, power management, cruise control, suspension control, tire stability control and *etc.*, have enough computational and structural aptitude to have a sense from each other's performance. This is crucial mainly because an autonomous vehicle must have a central controller (something like a core) which evaluates the performance of independent controllers and possibly satisfies a number of control objectives concerning the performance of entire system. So, independent controllers should be aware of any delay in the performance of other controlling units to somehow adapt themselves, and ensure the controlling signals and state information dispatched to the core controller has an acceptable correspondence. One of the most practical ways to comply with this important task is to use model based controllers which make decision based on a unified control objective. This control objective may contain states related to the performance of other controlling units, and by optimizing such an objective function, it can be assured that the independent controllers are in a good compromise. Also, other than considering a model-based optimal controller, it may be also beneficial to define the unified objective function in a predictive fashion. In such a way, if a good predictive model be used, one can increase the reliability of calculated commands, as the final calculated controlling command of each control unit depends on future state values of the other control units. Obviously, the successful implementation of such a sophisticated yet necessary objective function requires so many theoretical and analytical efforts which ensure the system stability, the reachability of states, and the feasibility (in terms of actuation and state constraints) of actuation signals. As a remedy to such a complicated and cumbersome problem, one can think of using learning paradigms to somehow reduce the amount of theoretical considerations and conservatisms from control side [83].

2. It is clear that designing a fully-autonomous vehicle falls within the category of vision-based autonomous navigation system design. This means that autonomous vehicles should be equipped with efficient image and video processing technologies

which make the computer vision feasible [212]. Autonomous vision in dynamic environment is a very challenging task, in particular for automotive applications in which the dynamics of the underlying system is remarkably fast, and also they may be several details at each sampling time which should be correctly captured with the aid of image processing and feature extraction techniques. At the sensing level, the computer vision unit should be capable of denoising and also handling parallel image processing tasks, hopefully by considering a multiobjective optimization problem and multitask inference, as there is a possibility that specific pieces of information captured through autonomous vision be fed to independent control and decision making units with certain obligations. Even if the visioning and inference of autonomous vehicles become highly precise and efficient, the process will not be completed unless it is ensured that control and decision making units receiving the information are accurate and robust enough to not only reject the remaining noises from the information, but also be prepared for abrupt changes. As an example, a controlling / decision processing unit may capture information regarding a sudden accident. Then, it should be flexible enough to instantly adapt its controlling commands / decisions. There are so many other concerns with a relatively same flavor which should be considered and possibly satisfied to make the visioning system efficient, and respectively, make the applicability of an autonomous vehicle feasible [213].

3. Fault diagnosis controlling units should be very accurate since any functioning deficiency can easily get the passengers into serious troubles. In so many cases, the experience of drivers helps them to somehow feel or even predict that the vehicle is going to undergo a deficient functioning. This incurs a proper action and deters possible functioning flaws when vehicles are on roads. However, advanced autonomous vehicles are expectedly operating independently, and there is a possibility that the upcoming faults not be predicted by passengers, probably because of the lack of experience. Also, an autonomous vehicle has a remarkably more complicated architecture which itself introduces so many new concerns, and that would be necessary to devise a new family of fault detection systems responsible for detecting possible faults of components such as computer vision unit, information processing unit and *etc*. The abovementioned facts disclose the necessity of considering two important factors when designing fault detection units for autonomous vehicles. The first one is the use of predictive systems (maybe in the form of predictive models) which can in some sense play the role of an experienced driver and alarm the core controlling unit before the vehicle encounters any problem. The other important issue which deserves consideration is the formulation of fault detection algorithms in the sense that they become robust to measurement noises and unmeasured disturbances [214]. This

will help fault detection units to send alarm commands when it is really necessary, and not react to any simple and possibly irrelevant deviation which could mostly be the result of disturbances. Fortunately, nowadays, designing predictive and robust learning and controlling systems has become a must and this will definitely come to the aid of engineers working on autonomous vehicle design.

4. Autonomous systems should be equipped with highly efficient sensors and adaptable estimation algorithms. This is very essential as such type of vehicles do not use any human inference and should correctly read information from ambient. Also, the decision processing units should be significantly robust and accurate to not only filter the noises which contaminate the main signals, but also make a fast and correct decision to make the functioning of autonomous vehicles optimal. From an economic viewpoint, equipping autonomous vehicles with too many accurate and expectedly expensive sensors can make their applicability questionable. This is because even a basic autonomous vehicle may become too expensive and does not gain popularity in markets. Therefore, automotive industrialists should make a reasonable trade-off between the expense of final products and their safety. In spite of a dramatic confliction between those two key issues, optimal satisfying of both of them becomes a must at-least when the goal is to design autonomous vehicles. So, there are so many philosophical and technical concerns which should be treated somehow. From technical viewpoint which is of interest to engineers and applied mathematicians, the controlling, sensing and state estimation units should be designed such that they can operate based a comprehensive and multi-criteria objective function which takes the mentioned key issues into account.

5. The information processing center of autonomous vehicles should be very powerful since a considerable volume of information should be captured from ambient. Thus, to comply with the requirements of an autonomous system, the information processing center should take advantage from a variety of techniques, such as compressed sensing, optimal filtering and feature selection, to make sure (a) it will not archive remarkably noisy information, (b) the saved data and signals are sparse, and (c) it has potential to organize different sort of information, separate them and fed them to correct computational units for further operations. Obviously, this is not an easy task and algorithm designers have a long way ahead to successfully design such a desirable unit. The issue even becomes more challenging as it is possible that the resulting database becomes tremendously large (in terms of both dimensionality and frequency). Under such circumstances, two important phenomenon may be encountered. Firstly, it is possible that a large portion of captured information be redundant

or yield trivial knowledge, and thus, a lot of optimization type computation should be performed for pruning the database. This may not be practical in dynamic environment, as any unexpected delay can result in the accumulation of information which exponentially increases the computational demand for information pruning [215]. The second scenario is that the captured information be of high importance and, instead of pruning, the processing unit should devise a policy to cope with possibly a remarkably high dimensional database. In such a condition, another important problem can be experienced. Through seminal theoretically supported studies, statistician have proven that in large dimensions, several unpredictable phenomenon may occur which drastically change the properties of databases [216]. Therefore, it is possible that the implemented information processing rules become entirely useless, which itself can directly result in a systematic collapse.

Obviously, from the above remarks, one can realize that learning and inference, optimal design, and controllability are three key factors which take part for improving the functioning of autonomous vehicles. That is to say, the best choice is to design the general architecture of a control unit such that it enjoys from learning, optimization and control. As discussed in details, fortunately, the flexibility of LBMPC makes it possible to consider all of the abovementioned issues. Also, being predictive and robust are intrinsic features of LBMPC which are essential for designing each of the control and fault detection units at the heart of an autonomous vehicle. The other interesting fact is that by a simple modification, the learning module at the heart of LBMPC can turn to a classification, regression and prediction unit which enables LBMPC to be used at the heart of fault-detection units, control units, and decision makers connected to computer vision center. Also, the optimization module makes it possible for LBMPC to be used as independent controller for handling sole control tasks as well as the core controller which requires the formulation of an objective function which takes into account the optimality of controlling commands calculated at each independent control unit. Also, as discussed, it is expected that the objective function of each independent controller be formulated using the states of other independent controllers (preferably in a predictive fashion) to make sure controllers are working cooperatively. This is also feasible by a wise formulation of objective function at the heart of LBMPC. It is also worth mentioning that the distinct architecture of LBMPC which uses both nominal and learning based versions of state-space in tandem can play a remarkable role for complying with robustness concerns, which are crucial for designing fault detection units as well as control units which should be able to properly reject disturbances to neatly calculate the controlling commands.

All of the mentioned issues can be an inspiration to choose LBMPC (of different types) for designing autonomous vehicles. It is worth mentioning that LBMPC has already proven

its high potential for designing a complicated autonomous aerial system, i.e. quadrotor helicopter. This endorses the author's claim regarding the potential of using LBMPC for designing autonomous vehicles.

9.3 LBMPC as Low-level Controller for Network of Connected Vehicles

Before starting the discussion, it should be pointed out that by hyper-level control, we refer to those controllers which try to control a general behavior of the network of interest, and by low-level control we focus on those type of controllers used for handling an internal task in each node of the network. It is apparent that based on the conditions of the defined problems, both hyper-level and low-level controllers can be deterministic or stochastic.

In general, a network of vehicles can be formed to comply with any predefined objective. For example, a network of connected vehicles can be formed to serve as a communication system transmitting information about monitoring the safety of roads, traffic on roads, intense accidents, and *etc.* Such an advanced and interesting paradigm which can be somehow viewed as a generalization of vehicle-to-vehicle (V2V) communication can be very beneficial, as it can serve as a mean for capturing a large amount of information useful for decision making and control units devised in smart modern vehicles [217].

However, there are so many challenges at the network design level since one encounters a mobile network with nodes having stochastic performance. Therefore, initially, it comes to one's mind that different key elements such as the robustness of objective function that the formed network should cope with, stability issues, denoising during information transmission, and *etc.* should be taken into account simultaneously. Also, there are so many challenges related to choosing the proper devices to form the network. Obviously, each node (vehicle) can be activated in so many fashions, ranging from using an advanced communication component devised in vehicle to using smart phones for communication. Up to now, several initial steps have been taken to make the designing of such a complex mobile network pragmatic. Most of the existing proposals rely on the communication of a very limited number of vehicles to improve the safety of cruise controllers [218, 14]. There are also some reports on using V2V connection for improving the performance of active suspension controllers [219, 220]. In all of the above referred researches, emphasis has been placed on using the transmitting information to improve the performance of low-level controllers. This seems to be logical, as the main goal of automotive engineers is to focus on improving the performance of vehicles (which can be viewed as active nodes in a

network of connected vehicles). Therefore, some sophisticated problems that usually arise at the network design level and are related to hyper-level controllers are none of automotive engineers concern. Therefore, by conceding this fact, here, the author will leave the open issues within the realm of vehicular communication network improvement and efficient hyper-level controller design untouched. Rather, all focus is put on discussing the potentials for the optimal utilization of transmitting information to improve the performance of low-level controllers.

In spite of the improvements reported in [218, 14, 219, 220], there exist so many open questions. This is because remarkable assumptions have been made to investigate the potential of V2V for low-level controller design. To name only a few, the following remarks can be pointed out:

1. In most of the conducted researches, the number of connected vehicles remains constant during the control process. This is very optimistic to proceed with such an assumption, especially for designing safety-oriented controllers. Let's give a subjective example for better clarification. Assume that within the context of safety-oriented control design, an optimal controller be formulated as the sum of linear/nonlinear functionals which depend on the sum of the distances of proceeding vehicles. Also, consider that the controller works online, and there is a perpetuate connection between active vehicles. In such a condition, any sudden event such as stirring or stopping of one of the proceeding vehicles will affect the control objective function value (usually appears as a sudden increase / decrease in the magnitude based on the property of the objective function), which in turn changes the calculated controlling command. However, if an adaptive frame be designed which only considers the connection between vehicles in a limited region, it can be ensured that the network has a logical stability and the transmitted information can be reliably used for low-level control. This very simple example reveals why it is meaningless to design V2V based controllers which have constant number of vehicles in the loop. On the other hand, it is expected that efficient design of a mobile V2V-based low-level controller with adaptive number of vehicles be a very complicated issue. This is because, by any change in the number of connected vehicles, the objective function of low-level controller should be adapted which may also require the calculation of new stability criteria. This seems to be necessary as the network of connected vehicles are mobile in essence, and due to so many factors, the number of connected nodes may vary during the process. This very simple argument indicates that a controller without having an adaptive entity or without being capable of the optimal calculation of control command cannot be considered as a practical choice for such applications.

2. A lot of the existing researches have not considered a significant randomness effect in their proposed models to deal with the uncertainties of each connected vehicle. As mentioned before, due to so many reasons (e.g. systematic faults), it is very likely that information coming from one of the connected vehicles be contaminated with a significant amount of noise. Consequently, using such information for calculating control commands can result in serious problems. To the best knowledge of the author, there exist no report in the literature which highlights this negative aspect of the proposed V2V-based controllers. The author's impression is that an independent and thorough investigation has to be conducted to realize the major related sources of uncertainty, and authentic statistical and probabilistic models should be proposed to get a view on their properties. After all, that would be necessary to get stuck in robust control theories to come up with controlling rules working under the outlined sources of uncertainty.

3. It can be strongly claimed that the existing low-level controllers can at-most use the information captured from the other connected vehicles to control an internal property of the corresponding vehicle. By rapid technological advancements, it is not intangible these days to use more computationally expensive controllers in modern vehicles [221]. This allows automotive engineers to design much sophisticated controlling architectures probably having a learning module at their heart. By doing so, low-level controllers can be equipped with techniques which make it possible for them to do statistical inference. This in turn allows low-level controllers of independent nodes to be connected to each other. So, it would be possible to have a vehicular communication network in which adaptive and trainable low-level controllers can incrementally learn from each other. Obviously, having such a controlling scheme can be very beneficial for the better guidance of active suspension and safety-oriented controllers. Although such propositions are a little bit far from the reality (due to the expense of equipping all vehicles with such infrastructures), but they can be a hot practical research topic in very near future. So, at-least at the simulation level and software-in-the-loop implementation, that would be necessary to seriously start working on using learnable low-level controllers which are able to communicate with each other, and to investigate the possible benefits with respect to a wide range of control objectives.

As can be inferred from the abovementioned remarks, LBMPC can be a very good candidate to be used as V2V communication based low-level controller. Apparently, it has a very good adaptive nature, and also calculates the controlling codes in an optimal fashion. This can be a compatible feature to use LBMPC in a mobile network of connected vehi-

cles. In this way, any change in the structural property of mobile network, e.g. exclusion or inclusion of a node, can be sensed and respective adaptation be made to LBMPC. Also, by means of the stability theorems developed for LBMPC, it can be ensured any sudden unwanted change or entropy in macroscopic scale will not result in a microscopic collapse. In other words, even if the performance of low-level controller be affected by the information transmitting among connected vehicles, there is a certain degree of independency at low-level control. The beauty of using LBMPC mainly lies in its ability to maintain a desirable rate of robustness when calculating the controlling commands. Such a feature makes LBMPC a very good solution for practical usage, as that would not be possible to use a low-level controller incapable of withstanding against the noise and disturbances associated with the upcoming information form the other networks. It has theoretically and numerically been proven that LBMPC can use an oracle which is capable of learning a wide range of structured and unstructured uncertainties, and making efficient reactions to reject those noises from original state signals. This trainable learning module at the heart of LBMPC is also a good fit for designing low-level controllers which somehow directly share information with other low-level controllers in connected vehicles. Upon implementation, such a beneficial feature can play an important role in designing novel control objectives with the primary goal of increasing the awareness of low-level controllers. Along with the mentioned features, it is also necessary to be able to implement a low-level controller for both regulation and trajectory tracking applications. For example, in [14], it was indicated that the low-level safety-oriented controller should track a trajectory to retain the vehicles in a safe distance from each other. On the other hand, in [220], the low level V2V-based controller takes information from other vehicles to regulate / stabilize the suspension system of the corresponding vehicle. Fortunately, as a variant of MPC, the control rule of LBMPC can be easily adapted to be used for both regulation and tracking applications.

All in all, the above discussion indicates that LBMPC can be used as low-level controller for a network of connected vehicles since it combines all learning, optimization, and robustness theories in a unified architecture. Without any prejudice, in this section, the author has tried to draw the attention of readers to the great potentials of LBMPC for handling such a control objective. Also, several open questions have been pointed out which deserve lots of research preferably by using LBMPC or any other controller which has features like learnability, optimality, stability, and robustness.

9.4 On Compatibility of LBMPC with AI: Is It a Potential Pragmatic Solution?

Nowadays, there is a compromise among different groups of engineering society that the success and fruitfulness of future research is strongly related to the advancements in AI. That would not be an exaggeration if one claims that AI will soon play a meaningful role in different fields ranging from control to engineering design [222]. To the author's best knowledge, the main reasons to support the above claim are twofold: (a) the astonishing progress of computational and technological facilities which enables the humanbeing to design remarkably powerful and intelligent machines, and (b) the urge for exploring new ideas which are usually beyond the scope of classical mathematics and statistical sciences. The former issue can be easily sensed in real-life.

By a simple comparison of the available facilities and technologies in the late 20^{th} century with those available today, one can easily realize that most of the technological facets have evidenced an unexpected promotion. Such a rapid technological growth can be realized in automotive industry, robotics, medical engineering, chemical engineering and *etc*. It is transparent that the recently emerged engineering problems within the mentioned realm are usually intricate, and in most of the cases, they require a supervisory architecture capable of inference and decision making. Inference and decision making may rely on solving a number of clustering, estimation and classification problems which are quite nonlinear with different types of constraints. Therefore, one logical choice for handling such problems could be the use of AI, which in turn encompasses so many powerful techniques beneficial for inference, estimation and classification [223]. One the other hand, the latter issue which refers to the tendency of researchers to take steps beyond the existing mathematical frames and try to solve complicated problems with the aid of nature-inspired techniques has been attracting a great attention [224]. This sub-field of AI which is known as nature-inspired computing or bio-inspired computing advocates the philosophy that inexact solvers inspired from the existing natural phenomena enable researchers to solve recently spotlighted overwhelming problems. By taking a glance into the exiting literature, one can easily endorse the fact that a great deal of effort is made to propose novel nature-inspired solvers or improve the existing ones with the hope of improving the performance of classical optimization, estimation and clustering techniques [225]. All of the above facts augur the high possibility that AI plays a pivotal role in future, and surpasses the other problem representation paradigms.

Also, during the past decade, AI has found reputation among control engineers. In this way, control engineers have successfully applied AI techniques to design intelligent optimal

controllers, nonlinear controllers, predictive controllers, real-time controllers, and *etc*. For more information on the progresses of AI based controller design, one can refer to [226]. Through theoretical analysis and numerical simulation, it has been proven that techniques from AI have a significant potential to be used in the core of controlling architectures. There are a number of obvious similarities between AI and modules used in control which justify their hybridization for practical applications. In below, we present the similarities between AI and important control paradigms (i.e. optimal control, model-based control, predictive control, and heuristic control):

1. Optimal controllers foster the use of an optimization problem to calculate the most efficient controlling commands [168]. The difficulty of the underlying optimization problem directly depends on the nature of the control problem at hand, and can be convex/non-convex, constrained/unconstrained, multivariate, linear/nonlinear, dynamic, and *etc*. In so many cases, the difficulty of the underlying problem can make it very hard for an optimizer to converge to an optimal solution, especially when the optimization problem should be solved in a real-time fashion. There is a huge amount of archived research from AI research society which explore using stochastic nature-inspired algorithms for global optimization in complicated solution landscapes. It has theoretically and numerically been indicated that AI optimization methods can converge to near global optimal solutions even if the objective function be highly nonlinear, constrained and dynamic [227]. This interesting asset has been identified by control engineers, and the potential of AI optimizers for designing optimal controllers has been evaluated using different control problems. The results are, in general, in favor of using AI based optimal controllers, and such paradigms can improve the quality of actuation signals, especially when the raised control problem is highly nonlinear. Therefore, it is expected that in the near future, controllers having optimal property be further explored and be combined with AI optimization techniques to handle difficult control problems.

2. Model-based controllers mainly rely on a model which represents the behavior of the system to be controlled. Such type of controllers cannot operate unless a proper state-space model be formulated which includes a number of states. Based on a predefined rule, a model-based controller calculates the actuation commands. Since its inception, such a controller has vigorously been applied to different control problems, and the feedbacks of the conducted research indicate its good operational potential [228]. However, a remarkable portion of such controllers use mathematical and physics-based models at their heart. Such models usually cannot fully capture the behavior of real system, especially when it is highly complex. Therefore, due to the

limited accuracy of physics-based models, the performance of model-based controller can be degenerated. In such a circumstance, control practitioners have reached the conclusion to start working on testing the potential of AI based models for designing model-based controllers [229]. In this context, a remarkable progress has been done by adopting and efficient designing of AI based models as control-oriented models. The main positive aspect of such statistical and soft AI models lies in the fact that they heavily rely on the data obtained from real system to develop a model. In this way, by using an efficient design of experiment (DoE) method or sensing technique, and by capturing a comprehensive database, it can be assured that the resulting AI models retain a good trade-off between accuracy and robustness, and can be reliably used as control-oriented models. One of the other promising assets of such AI model-based controllers lies in their capability to produce a fast and computationally efficient model representing the nonlinear and complicated behavior of real-system. Such desirable features of AI models have given them versatility and authenticity to be used for controlling of highly complicated plants. The promising reports have persuaded the researchers to consider model-based controllers, usually with AI models, as one of the most practical choices for handling emerging control problems. Therefore, that would not be far from imagination that those type of controllers which have the potential to be fused with an AI model be a target of comprehensive investigation in a near future.

3. Predictive control can be viewed as one of the most peculiar and powerful variants of controllers which is finding its place among industrialists [160]. The salient asset of such controllers emanates in their operational property which is based on using the predicted behavior of the considered system to make optimal control decision. As can be inferred, such type of controllers are advanced version of model-based controllers which not only are capable of using a model representing the behavior of the considered system, but also are equipped with prediction methodologies to somehow forecast the upcoming values of system states. Due to this specific feature, the controlling rules calculated by such controllers can be very efficient and reliable, compared to those controllers making decision based on the current states of the considered system. Classical prediction and sampling techniques have originally been proposed to come up with such controllers. However, after independent achievements by AI society on designing fast, accurate and robust predictors, it has come to researchers' mind to adopt AI-based prediction methods for predictive control. For detailed information on the compatibility of AI with predictive control, one can refer to [230]. As data-derived models, AI predictors can usually forecast the future behavior of any system with an acceptable accuracy, regardless of their

intrinsic nonlinearity and randomness. Therefore, it is expected that AI-predictors play a key role in future for designing robust and accurate predictive controllers.

4. Heuristic controllers are among the oldest and most applicable variants of controllers used in todays' industry. The main reason is their easy implementation and simple design logic as well as their independency from any system model. Usually, heuristic models work based on a number of rules that are gathered in a rule-base for making inference [231]. The resulting rule-base can be viewed as an experimentally designed map. Due to their versatility, control practitioners have put a great amount of energy to develop automated rule-base designing algorithms which enjoy a good generalization, and can be used for different applications. Fortunately, AI plays a significant role within the realm of rule-base design. Among the existing statistical and soft methods, one can refer to genetic programming, decision trees, neural trees, fuzzy logic, statistical symbolic computing and *etc.* From a control prospective, upon a good module-based architecture design, a model-based controller can be easily turned to a heuristic controller, provided that the state-space model be replaced with a bunch of rules. Researchers of fuzzy control have played around such an idea for a long time and their simulation results indicate that this strategy can afford a promising control outcome. Hence, AI will make a prominent contribution in the coming years to promote the efficacy of heuristic controllers. It is worth mentioning that LBMPC hosts an oracle in its architecture which can be easily turned to a rule-based system, if required.

From the above arguments, it can be concluded that LBMPC shares several similarities with the mentioned properties of a desirable controller, and its architecture is prone to be fused with AI. From an architectural viewpoint, the author believes that, given the control problem at hand, the optimization module, the oracle, the prediction strategy, and the learning module can easily be equipped with methods from AI for performance improvement. Also, the interesting and innovative architecture of LBMPC enables it to be combined with any type of AI techniques as long as the robustness of controlling command, and the convergence of oracle-based model to the nominal state-space model be proven. Therefore, the author believes that LBMPC can be at the core of attention, and rigorous investigations can be conducted to clearly explore the potential of methods form AI for performance improvement.

9.5 Open Computational and Fundamental Challenges

In this thesis, several simulation and analysis were carried out to present a comprehensive result on the potential of LBMPC for active suspension control. Due to the nonlinearity of the problem at hand, the obtained simulation results provided us with interesting and promising findings regarding the power of LBMPC. All in all, the obtained results were in favor of using LBMPC for active suspension control. However, such promising findings cannot put a closure to fundamental questions regarding the generalization and reliability of LBMPC as an efficient controlling algorithm. Actually, a relatively same argument holds for most of the existing control schemes, and it cannot be claimed that a certain type of controlling algorithm can outperform the other existing variants of controllers.

In general, concerns about choosing an appropriate controller can be viewed from computational and theoretical viewpoints. In other words, to have a firm judgement on the performance of a controller, it should be assured that its formulation and control rule calculation strategy have enough theoretical flexibility to prepare it to be used for a wide range of problems within the realms of nonlinear control, robust control, stochastic control, and optimal control. On the other hand, to be practical, the implementation of a controlling rule should not take a lot of computational power from processor. Such a feature is essential to make the hardware-in-the-loop (HIL) implementation of controller feasible. Fortunately, one can place LBMPC among those controllers which have an acceptable mathematical foundation. As clarified in this thesis, LBMPC is the result of several theoretical analysis to guarantee its stability, robustness, trajectory tracking and convergence capabilities. Other than the acceptable theoretical foundation of LBMPC, a number of HIL-tests have been conducted which also augur the real-time implementation capability of this controller. Having the mentioned fact into mind, in what follows this section, the author highlights some of the existing open issues which deserve further investigations and analysis. The provided remarks are divided into two different parts explaining the existing theoretical and computational concerns.

The following remarks can be pointed out concerning the open theoretical questions on using LBMPC:

1. Standard LBMPC has been developed in the light of a strong theoretical analysis proving the convergence and stability properties of the oracle used at the heart of learning module. Under several assumptions, it has been proven that the state-space with learning oracle will be stable and can converge to its nominal uncertain state-space counterpart. However, the structure of learning algorithm used for analysing the oracle is not so complicate. In many of the practical problems, the oracle should

be much more complicated, maybe a deep learner, and under such a condition, it can be very difficult or even impossible to make sure the same stability and convergence conditions hold. This fact has been drawn to the attention in this thesis, and it was indicated that considering a simple learning algorithm enables us to conduct a theoretical analysis. However, the prediction error of such an oracle becomes large when the underlying system is nonlinear. On the other hand, considering complicated and generalized learning algorithm can improve the accuracy of learning module, but that would be very arduous to comment on stability and convergence potentials of such methods. In view of such facts, one way for improving the performance of LBMPC is trying to implement more powerful, and usually complicated learning algorithm, as oracle, and at the same time, providing mathematical proofs on their convergence properties and stability.

2. By precise reading of the existing literature devoted to LBMPC, one can discern that the performed robustness analysis is not comprehensive. In general, it is well-known that the uncertainty can either be structured or unstructured with different properties. However, in the available literature, the conducted robustness analysis is rather general, and limited sources of uncertainty are added to the nominal state-space model for robustness analysis. It sounds that the investigation on this special aspect of LMBPC should be better organized, definitely under the light of well-established probabilistic and stochastic process theories. This enables control engineers to categorize most of the uncertainty scenarios possibly occurring in practice, and to develop corresponding strategies to cope with the uncertainties.

3. Less attention has been given in the literature to the improvement of the optimization module of LBMPC. This is when the conducted investigation on improving/analysing the optimization algorithm at the heart of standard MPC is abound. In this context, due to the particular architecture of LBMPC, an independent study should be carried out, perhaps by considering the results of theoretical optimization analysis on MPC, to elicit the most powerful optimization scenarios for LBMPC. Also, efforts should be made to explore and evaluate the existing constraint handling strategies to improve the performance of LBMPC when the underlying constraint optimization problem is difficult. It is suggested that those optimizers with firm theoretical foundation be selected for analysis. The main reason is that a theoretically well-established optimization method would yield much authentic information about key factors such as convergence speed, multi-modality handling, constraint handling, scalability, and *etc.* As indicated in this thesis, a proper optimizer can be selected from both the deterministic and stochastic optimization paradigms.

4. The potential of LBMPC to be implemented as a component of more complicated control paradigms such as hybrid switching controllers, hierarchical controllers, distributed controllers, and *etc.* should be verified. As we know, each of the mentioned paradigms have their own theoretical foundations which can expand the applicability of LBMPC. This key idea has been given full consideration for most of the powerful controllers such as PID, LQR, and H_2/H_∞. Hence, it is impossible for LBMPC to find its place among control practitioners unless making sure that it has enough flexibility to be implemented as a component of much more advanced control architectures.

Also, the following issues deserve a deep and comprehensive analysis to establish an acceptable foundation addressing the computational concerns about using LBMPC:

1. It is well-known to the control society that a large part of the existing control problems really require computationally efficient controlling algorithms. As a matter of fact, it is a common practice in so many cases to sacrifice the efficiency of a controller in favor of its computational speed. This is actually the main reason that the most applicable controller in nowadays industrial applications is the simplest one, i.e. PID. It sounds logical that to make LBMPC a competitive controlling scheme, at-least for nowadays industrial applications, there is no choice but to put a considerable effort for optimizing the general architecture of LBMPC as well as applying well-known model-reduction tools. The above recommendation is general but can take different forms based on the properties of problems at hand, and also based on the experience of control engineers. For example, when handling a vehicle system with complicated dynamics and different sub-components, the most emphasis should be exerted on applying efficient model reduction techniques to present a simple and approximately precise surrogate dynamical model. As another example, when handling a data-derived control problem, the most emphasis should be exerted on adopting efficient sparse representation and regularization techniques to simplify the representative machine to the highest possible extend. Thus, this general recommendation deserves independent studies for different applications, with the hope of making LBMPC (which unfortunately has a complex architecture) as simple as possible.

2. Other than searching for simple modules to form the architecture of LBMPC, an independent attempt should be made to reduce the algorithmic complexity of the techniques used in each module. By a precise investigation into the literature, one can realize that for most of the applications, there exist several methods with a relatively same efficiency but entirely different computational cost. For example, this

fact can be easily sensed when we try to choose an oracle for the estimation of disturbances or other sort of signals (maybe a same prediction can be performed with a simple statistical model or a complex deep learning system). In application, other than the theoretical concerns, it is also very important to make sure the considered modules have a fast calculation speed and can react in a reasonable portion of time. Thus, empirical investigation for obtaining the most computationally efficient (and theoretically safe) oracles and optimization algorithms is still open, and indeed deserves a thorough attention.

The above computational and fundamental challenges are among the most important elements which deserve independent empirical and theoretical studies on numerical benchmarks and real-world systems. The author believes that due to the very promising and reliable feedbacks on the applicability of LBMPC, control engineers will give attention to this peculiar control architecture and put lots of efforts to improve it from different aspects. So, the above information can play a key role for interested researchers working on the improvement of LBMPC. Having said that the author does not claim that the abovementioned remarks are complete, and several other issues can raise when applying LBMPC to real-world control problems. As a final remark, it is worth to draw the attention of the readers to a relatively recent review paper on MPC [232] in which different theoretical and computational aspects have been considered. As a model predictive control scheme, the same arguments may hold for LBMPC, and thus, a complete investigation on the advantages and downsides of LBMPC would require taking very similar steps.

References

[1] A Reghu and J Vetelino. *Introduction to Sensors*. New York: CRC Press, 2010.

[2] JY Nocedal and J Wright. *Numerical Optimization*. Springer Verlag, 2006.

[3] MK Steven. *Fundamentals of Statistical Processing: Estimation Theory*. Prenctice-Hall, 1993.

[4] VG Kulkarni. *Introduction to Modeling and Analysis of Stochastic Systems*. New York: Springer, 2011.

[5] K Ogata and Y Yang. *Modern Control Engineering*. Prentice-Hall Englewood Cliffs, 1970.

[6] W Liu. *Introduction to Hybrid Vehicle System Modeling and Control*. John Wiley & Sons, 2013.

[7] L Eriksson and L Nielsen. *Modeling and Control of Engines and Drivelines*. John Wiley & Sons, 2014.

[8] Z Nie. Study and application of the smart car control algorithm. In *Communications in Information and Management Engineering*, pages 138–147. Springer, 2011.

[9] H Cheng. Autonomous intelligent vehicles: theory, algorithms, and implementation. In *Advances in Computer Vision and Pattern Recognition*. Springer Science & Business Media, 2011.

[10] AG Ulsoy, H Peng, and M Çakmakci. *Automotive Control Systems*. Cambridge University Press, 2012.

[11] YZ Arslan, A Sezgin, and N Yagiz. Improving the ride comfort of vehicle passenger using fuzzy sliding mode controller. *Journal of Vibration and Control*, 21(9):1667–1679, 2015.

[12] EK Liebemann and DT Fuehrer. More safety with vehicle stability control. Technical report, SAE Technical Paper, doi:10.4271/2007-01-2759, 2007.

[13] I Eski and S Yıldırım. Vibration control of vehicle active suspension system using a new robust neural network control system. *Simulation Modelling Practice and Theory*, 17(5):778–793, 2009.

[14] A Mozaffari, M Vajedi, and NL Azad. A robust safety-oriented autonomous cruise control scheme for electric vehicles based on model predictive control and online sequential extreme learning machine with a hyper-level fault tolerance-based supervisor. *Neurocomputing*, 151:845–856, 2015.

[15] Z Preitl, P Bauer, B Kulcsar, G Rizzo, and J Bokor. Control solutions for hybrid solar vehicle fuel consumption minimization. In *IEEE Intelligent Vehicles Symposium*, pages 767–772. Istanbul, Turkey, 2007.

[16] M Ali. *Decision Making and Control for Automotive Safety*. PhD Thesis, Chalmers University of Technology, 2012.

[17] L Del Re, F Allgöwer, L Glielmo, C Guardiola, and I Kolmanovsky. Automotive model predictive control: models, methods and applications. In *Advances in Computer Vision and Pattern Recognition*. Springer, 2010.

[18] A Aswani, H Gonzalez, SS Sastry, and C Tomlin. Provably safe and robust learning-based model predictive control. *Automatica*, 49(5):1216–1226, 2013.

[19] PK Wong, HC Wong, CM Vong, Z Xie, and S Huang. Model predictive engine air-ratio control using online sequential extreme learning machine. *Neural Computing and Applications*, 27(1):79–92, 2016.

[20] VM Janakiraman, XL Nguyen, and D Assanis. An ELM based predictive control method for HCCI engines. *Engineering Applications of Artificial Intelligence*, 48:106–118, 2016.

[21] P Bouffard, Al Aswani, and C Tomlin. Learning-based model predictive control on a quadrotor: Onboard implementation and experimental results. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 279–284. Minnesota, USA, 2012.

[22] W Lou and X Guo. Adaptive trajectory tracking control using reinforcement learning for quadrotor. *International Journal of Advanced Robotic Systems*, 13(1):38, 2016.

[23] W Sun, H Pan, Y Zhang, and H Gao. Multi-objective control for uncertain nonlinear active suspension systems. *Mechatronics*, 24(4):318–327, 2014.

[24] H Liu, H Gao, and P Li. *Handbook of Vehicle Suspension Control Systems*. Institution of Engineering and Technology, 2013.

[25] P Stalph. *Analysis and Design of Machine Learning Techniques: Evolutionary Solutions for Regression, Prediction, and Control Problems*. Springer Science & Business Media, 2014.

[26] L del Re, P Ortner, and D Alberer. Chances and challenges in automotive predictive control. In *Automotive Model Predictive Control*, pages 1–22. Springer, 2010.

[27] E Guglielmino, T Sireteanu, CW Stammers, G Ghita, and M Giuclea. *Semi-active Suspension Control: Improved Vehicle Ride and Road Friendliness*. Springer Science & Business Media, 2008.

[28] GZ Yao, FF Yap, G Chen, WHo Li, and SH Yeo. Mr damper and its application for semi-active control of vehicle suspension system. *Mechatronics*, 12(7):963–973, 2002.

[29] LVV Gopala Rao and S Narayanan. Sky-hook control of nonlinear quarter car model traversing rough road matching performance of lqr control. *Journal of Sound and Vibration*, 323(3):515–529, 2009.

[30] J Swevers, C Lauwerys, B Vandersmissen, M Maes, K Reybrouck, and P Sas. A model-free control structure for the on-line tuning of the semi-active suspension of a passenger car. *Mechanical Systems and Signal Processing*, 21(3):1422–1436, 2007.

[31] JC Tudón-Martínez, R Morales-Menendez, R A Ramirez-Mendoza, and L Amezquita-Brooks. Comparison of heuristic controllers for an automotive semi-active suspension. *IFAC-PapersOnLine*, 47(3):6307–6312, 2014.

[32] JC Tudón-Martínez, D Hernández-Alcántara, and R Morales-Menendez. Semi-active suspension control with lpv mass adaptation. *IFAC-PapersOnLine*, 48(26):67–72, 2015.

[33] G Pepe and A Carcaterra. VFC-variational feedback controller and its application to semi-active suspensions. *Mechanical Systems and Signal Processing*, 76:72–92, 2016.

[34] LVG Rao and S Narayanan. Preview control of random response of a half-car vehicle model traversing rough road. *Journal of Sound and Vibration*, 310(1):352–365, 2008.

[35] Y Zhao, W Sun, and H Gao. Robust control synthesis for seat suspension systems with actuator saturation and time-varying input delay. *Journal of Sound and Vibration*, 329(21):4335–4353, 2010.

[36] LA Balzer. Optimal control with partial preview of disturbances and rate penalties and its application to vehicle suspension. *International Journal of Control*, 33(2):323–345, 1981.

[37] T Yoshimura and N Ananthanarayana. Stochastic optimal control of vehicle suspension with preview on an irregular surface. *International Journal of Systems Science*, 22(9):1599–1611, 1991.

[38] A Alleyne and JK Hedrick. Nonlinear adaptive control of active suspensions. *IEEE Transactions on Control Systems Technology*, 3(1):94–101, 1995.

[39] D Hrovat. Survey of advanced suspension developments and related optimal control applications. *Automatica*, 33(10):1781–1817, 1997.

[40] YM Sam, MRHA Ghani, and N Ahmad. LQR controller for active car suspension. In *IEEE Proceedings on TENCON*, pages 441–444, 2000.

[41] J Wang and DA Wilson. Mixed GL2/H2/GH2 control with pole placement and its application to vehicle suspension systems. *International Journal of Control*, 74(13):1353–1369, 2001.

[42] FC Wang and MC Smith. Disturbance response decoupling and achievable performance with application to vehicle active suspension. *International Journal of Control*, 75(12):946–953, 2002.

[43] JS Lin and CJ Huang. Nonlinear backstepping active suspension design applied to a half-car model. *Vehicle System Dynamics*, 42(6):373–393, 2004.

[44] D Fischer and R Isermann. Mechatronic semi-active and active vehicle suspensions. *Control Engineering Practice*, 12(11):1353–1367, 2004.

[45] N Yagiz and LE Sakman. Robust sliding mode control of a full vehicle without suspension gap loss. *Modal Analysis*, 11(11):1357–1374, 2005.

[46] A Zin, O Sename, and L Dugard. Switched H-infinity control strategy of automotive active suspensions. *IFAC-PapersOnLine*, 38(1):198–203, 2005.

[47] H Gao, J Lam, and C Wang. Multi-objective control of vehicle active suspension systems via load-dependent controllers. *Journal of Sound and Vibration*, 290(3):654–675, 2006.

[48] MC Readman, M Corless, C Villegas, and R Shorten. Adaptive Williams filters for active vehicle suspensions. *Transactions of the Institute of Measurement and Control*, 32(6):660–676, 2010.

[49] MM Fateh. Robust impedance control of a hydraulic suspension system. *International Journal of Robust and Nonlinear Control*, 20(8):858–872, 2010.

[50] S Rajala. *H-infinity Control Design of an Active Vehicle Suspension System*. MSc Thesis, Tampere University of Technology, 2012.

[51] C Poussot-Vassal, O Sename, L Dugard, P Gaspar, Z Szabo, and J Bokor. Attitude and handling improvements through gain-scheduled suspensions and brakes control. *Control Engineering Practice*, 19(3):252–263, 2011.

[52] Y Shoukry, MW El-Kharashi, and S Hammad. An embedded implementation of the Generalized Predictive Control algorithm applied to automotive active suspension systems. *Computers & Electrical Engineering*, 39(2):512–529, 2013.

[53] TPJ Van der Sande, BLJ Gysen, IJM Besselink, JJH Paulides, EA Lomonova, and H Nijmeijer. Robust control of an electromagnetic active suspension system: Simulations and measurements. *Mechatronics*, 23(2):204–212, 2013.

[54] Y Jin and X Luo. Stochastic optimal active control of a half-car nonlinear suspension under random road excitation. *Nonlinear Dynamics*, 72(1-2):185–195, 2013.

[55] S Fallah, A Sorniotti, and P Gruber. A novel robust optimal active control of vehicle suspension systems. *IFAC-PapersOnLine*, 47(3):11213–11218, 2014.

[56] J Lei, Z Jiang, YL Li, and WX Li. Active vibration control for nonlinear vehicle suspension with actuator delay via I/O feedback linearization. *International Journal of Control*, 87(10):2081–2096, 2014.

[57] Y Hu, MZQ Chen, and Z Hou. Multiplexed model predictive control for active vehicle suspensions. *International Journal of Control*, 88(2):347–363, 2015.

[58] A Mozaffari, A Doosthoseini, and NL Azad. Predictive control of suspension systems through combining dynamic matrix and constrained variable structure

controllers. *Journal of Dynamic Systems, Measurement, and Control*, 138(12), doi:10.1115/1.4034157, 2016.

[59] H Du, J Lam, and KY Sze. Non-fragile output feedback H-inf vehicle suspension control using genetic algorithm. *Engineering Applications of Artificial Intelligence*, 16(7):667–680, 2003.

[60] AB Sharkawy. Fuzzy and adaptive fuzzy control for the automobiles' active suspension system. *Vehicle System Dynamics*, 43(11):795–806, 2005.

[61] M Jonasson and F Roos. Design and evaluation of an active electromechanical wheel suspension system. *Mechatronics*, 18(4):218–230, 2008.

[62] C Tang, L Yue, L Guo, S Zhou, W Zhou, and Z Wang. Fuzzy logic control for vehicle suspension systems. In *International Conference on Intelligent Robotics and Applications*, pages 197–206. Springer, 2008.

[63] OA Dahunsi, JO Pedro, and OT Nyandoro. Neural network-based model predictive control of a servo-hydraulic vehicle suspension system. In *IEEE AFRICON'09.*, pages 1–6, 2009.

[64] MM Fateh and SS Alavi. Impedance control of an active suspension system. *Mechatronics*, 19(1):134–140, 2009.

[65] AL Do, O Sename, L Dugard, and B Soualmi. Multi-objective optimization by genetic algorithms in H-inf/LPV control of semi-active suspension. *IFAC-PapersOnLine*, 44(1):7162–7167, 2011.

[66] MM Fateh and MM Zirkohi. Adaptive impedance control of a hydraulic suspension system using particle swarm optimisation. *Vehicle System Dynamics*, 49(12):1951–1965, 2011.

[67] LH Zong, XL Gong, CY Guo, and SH Xuan. Inverse neuro-fuzzy MR damper model and its application in vibration control of vehicle suspension system. *Vehicle System Dynamics*, 50(7):1025–1041, 2012.

[68] O Demir, I Keskin, and S Cetin. Modeling and control of a nonlinear half-vehicle suspension system: a hybrid fuzzy logic approach. *Nonlinear Dynamics*, 67(3):2139–2151, 2012.

[69] R Kalaivani and P Lakshmi. Adaptive neuro-fuzzy controller for vehicle suspension system. In *IEEE International Conference on Advanced Computing (ICoAC)*, pages 236–240, 2013.

[70] PW Nugroho, H Du, W Li, and G Alici. Implementation of adaptive neuro fuzzy inference system controller on magneto rheological damper suspension. In *IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, pages 1399–1403, 2013.

[71] AA Aly and FA Salem. Vehicle suspension systems control: A review. *International Journal of Control, Automation and Systems*, 2(2):46–54, 2013.

[72] PW Nugroho, W Li, H Du, Gl Alici, and J Yang. An adaptive neuro fuzzy hybrid control strategy for a semiactive suspension with magneto rheological damper. *Advances in Mechanical Engineering*, 2014.

[73] J Meng, Q Chen, and R He. Research on optimal control for the vehicle suspension based on the simulated annealing algorithm. *Journal of Applied Mathematics*, 2014, 2014.

[74] Y Huang, J Na, X Wu, X Liu, and Y Guo. Adaptive control of nonlinear uncertain active suspension systems with prescribed performance. *ISA Transactions*, 54:145–155, 2015.

[75] M Ataei, E Asadi, A Goodarzi, A Khajepour, and MB Khamesee. Multi-objective optimization of a hybrid electromagnetic suspension system for ride comfort, road holding and regenerated power. *Journal of Vibration and Control*, page doi:1077546315585219, 2015.

[76] H Nourisola and B Ahmadi. Robust adaptive H-inf controller based on GA-Wavelet-SVM for nonlinear vehicle suspension with time delay actuator. *Journal of Vibration and Control*, 22(20):4111–4120, 2016.

[77] J Friedman, T Hastie, and R Tibshirani. The elements of statistical learning: data mining, inference, and prediction. In *Springer Series in Statistics*. New York, NY: Springer-Verlag New York, 2009.

[78] M Vidyasagar. Randomized algorithms for robust controller synthesis using statistical learning theory. *Automatica*, 37(10):1515–1528, 2001.

[79] S Vijayakumar, A D'souza, T Shibata, J Conradt, and S Schaal. Statistical learning for humanoid robots. *Autonomous Robots*, 12(1):55–69, 2002.

[80] M Cavalletti, J Piovesan, CT Abdallah, S Longhi, P Dorato, and G Ippoliti. Statistical learning controller for the energy management in a fuel cell electric vehicle. In *IEEE Conference on Decision and Control*, pages 2481–2486, 2008.

[81] P Bodík, R Griffith, CA Sutton, A Fox, MI Jordan, and DA Patterson. Statistical machine learning makes automatic control practical for internet datacenters. *HotCloud*, 9:12–12, 2009.

[82] S Schaal and CG Atkenson. Learning control in reobotics: Trajectory-based optimal control techniques. *IEEE Robotics & Automation Magazine*, 17(2):20–29, 2010.

[83] V Koltchinskii, CT Abdallah, M Ariola, and P Dorato. Statistical learning control of uncertain systems: theory and algorithms. *Applied Mathematics and Computation*, 120(1):31–43, 2001.

[84] J Piovesan, C Abdallah, M Egerstedt, H Tanner, and Y Wardi. Statistical learning for optimal control of hybrid systems. In *American Control Conference*, pages 2775–2780, 2007.

[85] J Nakanishi, JA Farrell, and S Schaal. Composite adaptive control with locally weighted statistical learning. *Neural Networks*, 18(1):71–90, 2005.

[86] D Cao, X Song, and M Ahmadian. Editors' perspectives: road vehicle suspension design, dynamics, and control. *Vehicle System Dynamics*, 49(1-2):3–28, 2011.

[87] RS Sharp. Vehicle dynamics and the judgment of quality. In *Vehicle Performance*, pages 87–96. Pauwelussen, JP, Swets & Zeitlinger Publishers, Lisse, the Netherlands, 1999.

[88] DA Crolla. Vehicle dynamicstheory into practice. *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering*, 210(2):83–94, 1996.

[89] TD Gillespie. Heavy truck ride. Technical report, SAE Technical Paper, doi: 10.4271/850001, 1985.

[90] A Lutz, J Rauh, and W Reinalter. Developments in vehicle dynamics and the tire model performance test. *Vehicle System Dynamics*, 45(S1):7–19, 2007.

[91] MJ Griffin. Discomfort from feeling vehicle vibration. *Vehicle System Dynamics*, 45(7-8):679–698, 2007.

[92] M El-Gindy. An overview of performance measures for heavy commercial vehicles in north america. *International Journal of Vehicle Design*, 16(4-5):441–463, 1995.

[93] RA Williams. Automotive active suspensions Part 1: basic principles. *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering*, 211(6):415–426, 1997.

[94] D Cebon. Interaction between heavy vehicles and roads. *Automotive Engineering International*, 1012:38–44, 1993.

[95] DJ Cole. Fundamental issues in suspension design for heavy road vehicles. *Vehicle System Dynamics*, 35(4-5):319–360, 2001.

[96] G Quaglia and M Sorli. Air suspension dimensionless analysis and design procedure. *Vehicle System Dynamics*, 35(6):443–475, 2001.

[97] M Gobbi, F Levi, and G Mastinu. Multi-objective stochastic optimisation of the suspension system of road vehicles. *Journal of Sound and Vibration*, 298(4):1055–1072, 2006.

[98] L Palkovics and A Fries. Intelligent electronic systems in commercial vehicles for enhanced traffic safety. *Vehicle System Dynamics*, 35(4-5):227–289, 2001.

[99] RS Sharp and DA Crolla. Road vehicle suspension system design-a review. *Vehicle System Dynamics*, 16(3):167–192, 1987.

[100] CV Suciu, T Tobiishi, and R Mouri. Modeling and simulation of a vehicle suspension with variable damping versus the excitation frequency. *Journal of Telecommunications and Information Technology*, pages 83–89, 2012.

[101] WS Aboud, SM Haris, and Y Yaacob. Advances in the control of mechatronic suspension systems. *Journal of Zhejiang University SCIENCE C*, 15(10):848–860, 2014.

[102] MR Jolly and DL Margolis. Regenerative systems for vibration control. *Journal of Vibration and Acoustics*, 119(2):208–215, 1997.

[103] D Karnopp. Active damping in road vehicle suspension systems. *Vehicle System Dynamics*, 12(6):291–311, 1983.

[104] E Alvarez-Sánchez. A quarter-car suspension system: car body mass estimator and sliding mode control. *Procedia Technology*, 7:208–214, 2013.

[105] SK Sharma, V Pare, M Chouksey, and BR Rawal. Numerical studies using full car model for combined primary and cabin suspension. *Procedia Technology*, 23:171–178, 2016.

[106] YM Goh, JD Booker, and CA McMahon. Uncertainty modelling of a suspension unit. *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering*, 219(6):755–771, 2005.

[107] TPJ van der Sande, BLJ Gysen, IJM Besselink, JJH Paulides, E Lomonova, and H Nijmeijer. Robust control of a direct-drive electromagnetic active suspension system. In *Proceedings of the Eighth International Symposium on Linear Drives for Industry Applications (LDIA11)*, pages 1–6, 2011.

[108] Q Pei, J Na, Y Huang, G Gao, and X Wu. Adaptive estimation and control of MR damper for semi-active suspension systems. In *IEEE 35th Chinese Control Conference (CCC)*, pages 3111–3116. China, 2016.

[109] Kun Jiang, Alessandro Corrêa Victorino, and Ali Charara. Adaptive estimation of vehicle dynamics through rls and kalman filter approaches. In *IEEE 18th International Conference on Intelligent Transportation Systems (ITSC)*, pages 1741–1746. Canary Islands, Spain, 2015.

[110] Y Kong, D Zhao, B Yang, C Han, and K Han. Non-fragile multi-objective static output feedback control of vehicle active suspension with time-delay. *Vehicle System Dynamics*, 52(7):948–968, 2014.

[111] H Du and N Zhang. H-infinity control of active vehicle suspensions with actuator time delay. *Journal of Sound and Vibration*, 301(1):236–252, 2007.

[112] YMd Sam, JHS Osman, and MRA Ghani. A class of proportional-integral sliding mode control with application to active suspension system. *Systems & Control Letters*, 51(3):217–223, 2004.

[113] L Li and FY Wang. *Advanced Motion Control and Sensing for Intelligent Vehicles*. Springer Science & Business Media, 2007.

[114] MM Farias, SAD Neto, and RO Souza. Prediction of longitudinal roughness using neural network. In *Maintenance and Rehabilitation of Pavements and Technological Control, Portugal*, 2003.

[115] S Yildirim and I Uzmay. Statistical analysis of vehicles' vibration due to road roughness using radial basis artificial neural network. *Applied Artificial Intelligence*, 15(4):419–427, 2001.

[116] R Guclu. Fuzzy logic control of seat vibrations of a non-linear full vehicle model. *Nonlinear Dynamics*, 40(1):21–34, 2005.

[117] A Giua, M Melas, C Seatzu, and G Usai. Design of a predictive semiactive suspension system. *Vehicle System Dynamics*, 41(4):277–300, 2004.

[118] M Shinozuka and CM Jan. Digital simulation of random processes and its applications. *Journal of Sound and Vibration*, 25(1):111–128, 1972.

[119] MQ Nguyen, M Canale, O Sename, and L Dugard. A model predictive approach for semi active suspension control problem of a full car. In *IEEE Conference on Decision and Control*, 2016.

[120] RH Shumway and DS Stoffer. *Time Series Analysis and its Applications: with R Examples*. Springer Science & Business Media, 2010.

[121] P Marriot. *Lecture Notes for STAT 443: Forecasting*. University of Waterloo, 2017.

[122] M Seyfi, R Rawat, J Weligamage, and R Nayak. A data analytics case study assessing factors affecting pavement deflection values. *International Journal of Business Intelligence and Data Mining*, 8(3):199–226, 2013.

[123] R Van Handel. *Stochastic Calculus, Filtering, and Stochastic Control*. Lecture note in Advanced Topics in Stochastic Analysis, California Institute of Technology, URL http://www. princeton. edu/˜ rvan/acm217/ACM217. pdf, 2007.

[124] D Liu, Q Wei, D Wang, X Yang, and H Li. Adaptive dynamic programming with applications in optimal control, 2017.

[125] V Dragan, T Morozan, and AM Stoica. *Mathematical methods in robust control of discrete-time linear stochastic systems*. Springer, 2010.

[126] A Ray. Mathematical methods in robust control of linear stochastic systems (Mathematical Concepts and Methods in Science and Engineering Series)(by Y. Dragan et al.; 2006)[Book reviews]. *IEEE Transactions on Automatic Control*, 53(3):862–864, 2008.

[127] DP Bertsekas and S Shreve. *Stochastic Optimal Control: The Discrete-Time Case*. Massachusetts Institute of Technology, Athena Scientific, Belmont, Massachusetts, 2004.

[128] J Garrido, W Yu, and X Li. Robot trajectory generation using modified hidden Markov model and Lloyd's algorithm in joint space. *Engineering Applications of Artificial Intelligence*, 53:32–40, 2016.

[129] R Kamalapurkar, L Andrews, P Walters, and WE Dixon. Model-based reinforcement learning for infinite-horizon approximate optimal tracking. *IEEE Transactions on Neural Networks and Learning Systems*, 28(3):753–758, 2017.

[130] F Günter. *Using Reinforcement Learning for Optimizing the Reproduction of Tasks in Robot Programming by Demonstration*. PhD Thesis, Ecole Polytechnique Federal De Laisanne, Switzerland, 2009.

[131] D Silver, A Huang, CJ Maddison, A Guez, L Sifre, et al. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

[132] CJ Watkins. *Learning from Delayed Rewards*. PhD Thesis, University of Cambridge, England, 1989.

[133] L Sun, JD Hedengren, and RW Beard. Optimal trajectory generation using model predictive control for aerially towed cable systems. *Journal of Guidance, Control, and Dynamics*, 37(2):525–539, 2014.

[134] W Feller. *An Introduction to Probability Theory and its Applications*. John Wiley & Sons New York, 1968.

[135] Y Grinberg and TJ Perkins. Fully polynomial-time computation of maximum likelihood trajectories in Markov chains. *Information Processing Letters*, 118:53–57, 2017.

[136] H Schneider and MH Schneider. Towers and cycle covers for max-balanced graphs. *Congressus Numerantium*, 73:159–170, 1990.

[137] CJ Ostafew, AP Schoellig, and TD Barfoot. Learning-based nonlinear model predictive control to improve vision-based mobile robot path-tracking in challenging outdoor environments. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 4029–4036. Hong Kong, China, 2014.

[138] G Chowdhary, M Muhlegg, JP How, and F Holzapfel. A concurrent learning adaptive-optimal control architecture for nonlinear systems. In *IEEE Conference on Decision and Control (CDC)*, pages 868–873. Florence, Italy, 2013.

[139] CJ Ostafew, AP Schoellig, TD Barfoot, and J Collier. Learning-based nonlinear model predictive control to improve vision-based mobile robot path tracking. *Journal of Field Robotics*, 33(1):133–152, 2016.

[140] A Aswani, H Gonzalez, SS Sastry, and C Tomlin. Statistical results on filtering and epi-convergence for learning-based model predictive control. *arXiv preprint arXiv:1208.0864*, 2012.

[141] A Aswani and C Tomlin. Reachability algorithm for biological piecewise-affine hybrid systems. In *International Workshop on Hybrid Systems: Computation and Control*, pages 633–636. Springer, 2007.

[142] IM Mitchell, AM Bayen, and CJ Tomlin. A time-dependent Hamilton-Jacobi formulation of reachable sets for continuous dynamic games. *IEEE Transactions on Automatic Control*, 50(7):947–957, 2005.

[143] F Borrelli. *Constrained Optimal control of Linear and Hybrid systems*. Springer, 2003.

[144] JB Rawlings and DQ Mayne. *Model Predictive Control: Theory and Design*. Nob Hill Publishing, 2009.

[145] P Bouffard. On-board model predictive control of a quadrotor helicopter: design, implementation and experiments. Technical report, University of California, Berkeley, 2012.

[146] A Aswani, N Master, J Taneja, A Krioukov, D Culler, and C Tomlin. Energy-efficient building hvac control using hybrid system LBMPC. *IFAC-PapersOnLine*, 45(17):496–501, 2012.

[147] A Aswani, N Master, J Taneja, D Culler, and C Tomlin. Reducing transient and steady state electricity consumption in HVAC using learning-based model-predictive control. *Proceedings of the IEEE*, 100(1):240–253, 2012.

[148] A Aswani, P Bouffard, and C Tomlin. Extensions of learning-based model predictive control for real-time application to a quadrotor helicopter. In *American Control Conference (ACC)*, pages 4661–4666. Montreal, Canada, 2012.

[149] A Aswani, P Bouffard, X Zhang, and C Tomlin. Practical comparison of optimization algorithms for learning-based MPC with linear models. *arXiv preprint arXiv:1404.2843*, 2014.

[150] SS Sastry. *Nonlinear Systems: Analysis, Stability, and Control*. Springer Science & Business Media, 2013.

[151] C Cai and AR Teel. Input-output-to-state stability for discrete-time systems. *Automatica*, 44(2):326–336, 2008.

[152] ZP Jiang and Y Wang. Input-to-state stability for discrete-time nonlinear systems. *Automatica*, 37(6):857–869, 2001.

[153] JJ E Slotine and W Li. *Applied Nonlinear Control*. Prentice-Hall Englewood Cliffs, 1991.

[154] I Kolmanovsky and EG Gilbert. Theory and computation of disturbance invariant sets for discrete-time linear systems. *Mathematical Problems in Engineering*, 4(4):317–367, 1998.

[155] RI Soare. Turing oracle machines, online computing, and three displacements in computability theory. *Annals of Pure and Applied Logic*, 160(3):368–399, 2009.

[156] S Ozawa, S Pang, and N Kasabov. *Incremental Learning in Intelligent Systems*. Wiley-IEEE Press, 2017.

[157] J Nocedal and S Wright. *Numerical Optimization*. Springer Science & Business Media, 2006.

[158] KR Muske and JB Rawlings. Model predictive control with linear models. *AIChE Journal*, 39(2):262–287, 1993.

[159] D Limón, I Alvarado, T Alamo, and EF Camacho. Mpc for tracking piecewise constant references for constrained linear systems. *Automatica*, 44(9):2382–2387, 2008.

[160] JM Maciejowski. *Predictive Control with Constraints*. Pearson Education, 2002.

[161] S V Rakovic, EC Kerrigan, KI Kouramas, and DQ Mayne. Invariant approximations of the minimal robust positively invariant set. *IEEE Transactions on Automatic Control*, 50(3):406–410, 2005.

[162] DQ Mayne, JB Rawlings, CV Rao, and POM Scokaert. Constrained model predictive control: Stability and optimality. *Automatica*, 36(6):789–814, 2000.

[163] W Langson, I Chryssochoos, SV Raković, and DQ Mayne. Robust model predictive control using tubes. *Automatica*, 40(1):125–133, 2004.

[164] C Kirches, L Wirsching, HG Bock, and JP Schlöder. Efficient direct multiple shooting for nonlinear model predictive control on long horizons. *Journal of Process Control*, 22(3):540–550, 2012.

[165] L Chisci, JA Rossiter, and G Zappa. Systems with persistent disturbances: predictive control with restricted constraints. *Automatica*, 37(7):1019–1028, 2001.

[166] G Grimm, MJ Messina, SE Tuna, and AR Teel. Examples when nonlinear model predictive control is nonrobust. *Automatica*, 40(10):1729–1738, 2004.

[167] W Rudin. *Principles of Mathematical Analysis*. McGraw-Hill New York, 1964.

[168] DS Naidu. *Optimal Control Systems*. CRC Press, 2002.

[169] ZP Jiang and Y Wang. A converse lyapunov theorem for discrete-time systems with disturbances. *Systems & Control Letters*, 45(1):49–58, 2002.

[170] T Hastie, R Tibshirani, and J Friedman. *The Elements of Statistical Learning; Data Mining, Inference and Prediction*. Springer, New York, 2009.

[171] JC Zavala, PR Sanketi, M Wilcutts, T Kaga, and JK Hedrick. Simplified models of engine HC emissions, exhaust temperature and catalyst temperature for automotive coldstart. *IFAC-PapersOnLine*, 40(10):199–205, 2007.

[172] W Härdle, M Müller, S Sperlich, and A Werwatz. *Nonparametric and Semiparametric Models*. Springer Science & Business Media, 2004.

[173] NL Azad, A Mozaffari, and JK Hedrick. A hybrid switching predictive controller based on bi-level kernel-based ELM and online trajectory builder for automotive coldstart emissions reduction. *Neurocomputing*, 173:1124–1141, 2016.

[174] P Lachout and S Vogel. On continuous convergence and epi-convergence of random functions. Part II: Sufficient conditions and applications. *Kybernetika*, 39:99–118, 2003.

[175] KI Kouramas. *Control of Linear Systems with State and Control Constraints*. PhD thesis, Imperial College of Science, Technology and Medicine, University of London, UK, 2002.

[176] SM Veres. Geometric Bounding Toolbox (GBT) for MATLAB. *Official website: http://www. sysbrain. com*, [Online], 2003.

[177] M Kvasnica, P Grieder, M Baotić, and M Morari. Multi-parametric toolbox (MPT). In *International Workshop on Hybrid Systems: Computation and Control*, pages 448–462. Springer, 2004.

[178] S Vogel and P Lachout. On continuous convergence and epi-convergence of random functions. Part I: Theory and relations. *Kybernetika*, 39(1):75–98, 2003.

[179] DH Wolpert and WG Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, 1997.

[180] Y Wang and S Boyd. Fast model predictive control using online optimization. *IEEE Transactions on Control Systems Technology*, 18(2):267–278, 2010.

[181] KV Ling, J Maciejowski, A Richards, and BF Wu. Multiplexed model predictive control. *Automatica*, 48(2):396–401, 2012.

[182] H Richter, AV Singaraju, and JS Litt. Multiplexed predictive control of a large commercial turbofan engine. *Journal of Guidance, Control, and Dynamics*, 31(2):273–281, 2008.

[183] KV Ling, WK Ho, BF Wu, A Lo, and H Yan. Multiplexed MPC for multizone thermal processing in semiconductor manufacturing. *IEEE Transactions on Control Systems Technology*, 18(6):1371–1380, 2010.

[184] O Cord, F Herrera, F Hoffman, and Magdalena L. *Genetic Fuzzy Systems: Evolutionary Tuning and Learning of Fuzzy Knowledge Bases*. World Scientific, 2001.

[185] S Patnaik and B Zhong. *Soft Computing Techniques in Engineering Applications*. Springer, 2014.

[186] ML Stein. *Interpolation of Spatial Data: Some Theory for Kriging*. Springer, 2012.

[187] G McLachlan and T Krishnan. *The EM Algorithm and Extensions*. John Wiley & Sons, 2007.

[188] MI Jordan and RA Jacobs. Hierarchical mixtures of experts and the EM algorithm. *Neural Computation*, 6(2):181–214, 1994.

[189] AJ Smola and B Schölkopf. A tutorial on support vector regression. *Statistics and Computing*, 14(3):199–222, 2004.

[190] S Haykin. *Neural Networks: A Comprehensive Foundation*. Prentice Hall PTR, 1994.

[191] WF Schmidt, MA Kraaijveld, and RPW Duin. Feedforward neural networks with random weights. In *IEEE Proceedings on Pattern Recognition Methodology and Systems*, pages 1–4. IEEE, 1992.

[192] H Allende-Cid, A Veloz, R Salas, S Chabert, and H Allende. Self-organizing neuro-fuzzy inference system. In *Progress in Pattern Recognition, Image Analysis and Applications*, pages 429–436. Springer, 2008.

[193] DP Searson, DE Leahy, and MJ Willis. Gptips: an open source genetic programming toolbox for multigene symbolic regression. In *Proceedings of the International Multiconference of Engineers and Computer Scientists*, pages 77–80, 2010.

[194] RS Sadjad. The eigenvalue multiplicity problem in the pole-placement method of state-variable feedback control design. *available online at http://independent.academia.edu/RhizaSadjad*, 2017.

[195] D Liu, Q Wei, D Wang, X Yang, and H Li. *Adaptive Dynamic Programming with Applications in Optimal Control*. Springer, 2017.

[196] J Harrison and M West. *Bayesian Forecasting & Dynamic Models*. Springer New York City, 1999.

[197] J Skach. Reinforcement learning and approximate dynamic programming for discrete-time systems. *Lecture Notes on Identification and Decision Making, University of West Bohemia, Czech*, 2016.

[198] K Ferentinos. On Tchebycheff's type inequalities. *Trabajos de Estadstica e Investigacin Operativa*, 33(1):125–132, 1982.

[199] A Mozaffari, NL Azad, A Hansen, and JK Hedrick. A receding horizon sliding controller for automotive engine coldstart: design and hardware-in-the-loop testing with an echo state network high-fidelity model. *Asian Journal of Control*, 18(4):1219–1238, 2016.

[200] S Kirkpatrick, CD Gelatt, and MP Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.

[201] G Casella, CP Robert, and MT Wells. Generalized accept-reject sampling schemes. *Institute of Mathematical Statistics Lecture Notes-Monograph Series*, 45:342–347, 2004.

[202] A Madady. An extended PID type iterative learning control. *International Journal of Control, Automation and Systems*, 11(3):470–481, 2013.

[203] D Sammier, O Sename, and L Dugard. Skyhook and Hinf control of semi-active suspensions: some practical aspects. *Vehicle System Dynamics*, 39(4):279–308, 2003.

[204] S Wu, R Zhang, R Lu, and F Gao. Design of dynamic matrix control based PID for residual oil outlet temperature in a coke furnace. *Chemometrics and Intelligent Laboratory Systems*, 134:110–117, 2014.

[205] NL Azad, PR Sanketi, and JK Hedrick. Sliding mode control with bounded inputs and its application to automotive coldstart emissions reduction. In *American Control Conference (ACC)*, pages 2860–2865, Montreal, Canada, 2012.

[206] RK Jurgen. Autonomous vehicles for safer driving. *Training*, 2017:4–20, 2013.

[207] V Nath and SE Levinson. *Autonomous Robotics and Deep Learning*. Springer Science & Business Media, 2014.

[208] NK Dhiman, D Deodhare, and D Khemani. A review of path planning and mapping technologies for autonomous mobile robot systems. In *Proceedings of the 5th ACM COMPUTE Conference: Intelligent & Scalable System Technologies*. New York, USA, 2012.

[209] M Duarte, SM Oliveira, and AL Christensen. Evolution of hybrid robotic controllers for complex tasks. *Journal of Intelligent & Robotic Systems*, 78(3-4):463–484, 2015.

[210] H Fazlollahtabar and M Saidi-Mehrabad. *Autonomous Guided Vehicles: Methods and Models for Optimal Path Planning*. Springer, 2015.

[211] DL Ho, RP Larsen, T Levy, and R Cuenca. The cooperative automotive research for advanced technology program (CARAT): accelerating the commercialization of innovative technology. Technical report, SAE Technical Paper, doi:10.4271/2000-01-1594, 2000.

[212] U Regensburger, R Ott, W Platz, G Wanielik, and N Appenrodt. Computer vision suitable for vehicle applications. In *Advanced Microsystems for Automotive Applications*, pages 183–192. Springer, 1998.

[213] A Broggi and S Berte. Vision-based road detection in automotive systems: A real-time expectation-driven approach. *Journal of Artificial Intelligence Research*, 3:325–348, 1995.

[214] S Varrier, D Koenig, and JJ Martinez. Robust fault detection for vehicle lateral dynamics. In *IEEE Conference on Decision and Control (CDC)*, pages 4366–4371. Hawaii, USA, 2012.

[215] L Rosasco and S Villa. Learning with incremental iterative regularization. In *Advances in Neural Information Processing Systems*, pages 1630–1638, 2015.

[216] P Bühlmann and S Van De Geer. *Statistics for High-Dimensional Data: Methods, Theory and Applications*. Springer Science & Business Media, 2011.

[217] N Lu, N Cheng, N Zhang, X Shen, and JW Mark. Connected vehicles: Solutions and challenges. *IEEE Internet of Things Journal*, 1(4):289–299, 2014.

[218] F Bu and CY Chan. Adaptive and cooperative cruise control. In *Handbook of Intelligent Vehicles*, pages 191–207. Springer, 2012.

[219] Z Li, I Kolmanovsky, E Atkins, J Lu, D Filev, and J Michelini. Cloud aided semi-active suspension control. In *IEEE Symposium on Computational Intelligence in Vehicles and Transportation Systems (CIVTS)*, pages 76–83. Orlando, USA, 2014.

[220] Z Li, I Kolmanovsky, E Atkins, J Lu, and D Filev. H-Inf filtering for cloud-aided semi-active suspension with delayed road information. *IFAC-PapersOnLine*, 48(12):275–280, 2015.

[221] LJ Howell. Innovation in the automobile industry: A new era. *Chemical Innovation*, 30(11):16–21, 2000.

[222] S Russell and P Norvig. *Artificial Intelligence: A Modern Approach*. Prentice-Hall Publisher, 2009.

[223] NG Bourbakis. *Artificial Intelligence Methods and Applications*. World Scientific, 1992.

[224] A Brabazon, M O'Neill, and S McGarraghy. *Natural Computing Algorithms*. Springer, 2015.

[225] LN de Castro. Fundamentals of natural computing: an overview. *Physics of Life Reviews*, 4(1):1–36, 2007.

[226] L Boullart, A Krijgsman, and RA Vingerhoeds. *Application of Artificial Intelligence in Process Control: Lecture Notes Erasmus Intensive Course*. Elsevier, 2013.

[227] XS Yang. *Nature-Inspired Optimization Algorithms*. Elsevier, 2014.

[228] PMJ Van den Hof, CS Scherer, and PSC Heuberger. *Model-Based Control: Bridging Rigorous Theory and Advanced Technology*. Springer, Berlin-Heidelberg, 2009.

[229] GW Irwin. Artificial intelligence approaches to model-based control. *IEEE Colloquium on Update on Developments in Intelligent Control*, doi:10.1049/ic:19981030, 1998.

[230] M Galily, FH Roudsari, M Fardis, and A Yazdian. Intelligent predictive control of micro heat exchanger. In *Neural Nets, Lecture Notes in Computer Science*, pages 83–89. Springer, 2006.

[231] WH Bare, RJ Mulholland, and SS Sofer. Design of a self-tuning rule based controller for a gasoline refinery catalytic reformer. *IEEE Transactions on Automatic Control*, 35(2):156–164, 1990.

[232] DQ Mayne. Model predictive control: Recent developments and future promise. *Automatica*, 50(12):2967–2986, 2014.

APPENDICES

Appendix A

Codes Required for Simulating the Results

A.1 Codes implemented in Chapter 4

The simulation conducted in Chapter 4 includes a MATLAB code for pre-processing of data, and a R code which tests several forecasting methods to get the best one for controller design.

A.1.1 MATLAB code for pre-processing of road roughness data

```
Mfile for reading the times from cell arrays and identify the new
% set-points (with the length of 1 sec for our simulation)

clc; clear;

load('RecordedTimes_2'),
data=RecordedTimes_2;
% convert to strings (each arrays shows the second in which the acceleration is measured)
mcs = cellfun(@(x)(mat2str(x)),data,'uniformoutput',false);

% find unique arrays in the resultung matrix. There are several records in each second
% So, by extracting unique arrays, we find out the number of rec
[uniqueCells,idxOfUnique,idxYouWant] = unique(mcs);

Time_Change=idxOfUnique;

% Extracting the displacement profile
```

```
load('Acceleration_2'),
load('Time_Change_2'),

for i=1:1:size(Time_Change_2,1)
if i==size(Time_Change_2,1)
Acc(i)= mean(Acceleration_2(Time_Change_2(i):end));
else
Acc(i)= mean(Acceleration_2(Time_Change_2(i):Time_Change_2(i+1)));
end
end


V0=-1;        % initial vertical velocity
Acc=Acc-0.25; % taking off the intercept to get the mean-adjusted stationary profile
Acc=Acc';
for i=1:1:size(Acc,1)

if i==1
Velocity(i) = V0 + (Acc(i)) ;
else
Velocity(i) = Velocity(i-1)+ (Acc(i)) ;
end
end

Velocity=Velocity';
X0=0; % initial vertical displacement
for i=1:1:size(Velocity,1)
if i==1
Displacement(i) = X0 + (Velocity(i)) ;
else
Displacement(i) = (Velocity(i-1)+ (Velocity(i))) ;
end
end

% Turning the displacement to cm
Displacement = 0.01*Displacement;

plot(Displacement),
figure,
plot(Velocity)
figure,
plot(Acc, 'r')

% Differencing Experiment

Acc_diff1=diff(Acc); % first order differenced version of mean adjusted acceleration
pp= cumsum ([Acc(1);Acc_diff1]); % check the difference

figure,
plot(Acc,pp,'o')
figure,
histfit(Acc,10,'normal')
figure,
plot(Acc_diff1)
figure,
histfit(Acc_diff1,10,'normal')
```

A.1.2 R code for analysis of forecasting models

```
setwd("C:\\Users\\Desktop\\Statistical Learning-and Stochastic Process for Robust Predictive Control")

X=()
Z=c()
X<-read.csv("Acc.csv",header=TRUE)
Z<- X[1:165,1]
ts.plot(Z)
hist(Z)


###### ************* Step 1 ************* ######
# Original time series
par(mfrow=c(2,2))
plot.ts(Z, xlab="Time", ylab="Acceleration")
hist(Z)

# The first order difference
ZZ=diff(Z)
ts.plot(ZZ,xlab="Time", ylab="1st order difference")
hist(ZZ)

###### ************* Step 2 ************* ######
Dataa=ZZ
par(mfrow=c(3,1))

## the qq norm indicates that the obtained results are trustable for analysis
qqnorm(Dataa,xlim=c(-2,2),ylim=c(-2,2))
qqline(Dataa)
acf(Dataa, lag.max = 15,
plot = TRUE, na.action = na.fail, demean = TRUE)
pacf(Dataa, lag.max = 15,
plot = TRUE, na.action = na.fail, demean = TRUE)

###### ************* Step 3 ************* ######
# Based on analysis, we have some clues to search for reasonable models
# fit for this dataset, i.e. ARMA and MA.

# ------------------------------------------------------------------------------
# ------------------------ Fit MA based on AIC and OLS -------------------------
# ------------------------------------------------------------------------------
Dataa=ZZ[1:160]
d=0
p=0
q=3
fit.ma <- arima(Dataa, order = c(p, d, q),
xreg = NULL, include.mean = FALSE,
transform.pars = TRUE,
fixed = NULL, init = NULL,
method = c("CSS-ML", "ML", "CSS"),
SSinit = c("Gardner1980", "Rossignol2011"),
optim.method = "BFGS",
optim.control = list(), kappa = 1e6)

# analyzing the fitted MA
```

```
tsdiag(fit.ma)

# Use the identified paramaeters for simulation
Sim1=c()
Sim2=c()
Sim3=c()
Sim4=c()
# Sim1 <- arima.sim(list(ma = c(-0.9951)),n=length(Dataa),innov=Dataa)
# Sim2 <- arima.sim(list(ma = c(-1.0290,0.0381)),n=length(Dataa),innov=Dataa)
Sim3 <- arima.sim(list(ma = c(-1.0161,-0.0702,0.1065)),n=length(Dataa),innov=Dataa)
# Sim4 <- arima.sim(list(ma = c(-1.0209,-0.0725,0.1375,-0.0261)),n=length(Dataa),innov=Dataa)
MA_fit_val=Sim3

# calculate the 10-fold training MSE
seg.length= length(Dataa) / 10
err=c()
fff=MA_fit_val-Dataa
for(jj in 1:10)
{
ffff = fff [((jj-1)*seg.length)+1:(jj*seg.length)]
err[jj]= sum((ffff[1:16])^2) / seg.length
ffff = c()
}
mean.error=sum(fff)/length(Dataa)
std= (sqrt(sum(( (fff) - mean.error)^2) )) /length(Dataa)
m.s.error= mean(err)

par(mfrow=c(3,2))
# TRAINING PART
qqnorm((MA_fit_val-Dataa), main="qq-plot for residuals")
qqline((MA_fit_val-Dataa))
ccc=seq(from =-2, to = 2, by = 0.1 )
plot(Dataa,MA_fit_val, xlim=c(-2,2), ylim=c(-2,2), xlab="data", ylab="MA fit")
lines(ccc,ccc, type="l", col="blue")
# TESTING PART
PredLength=5
Newdataa=ZZ[(length(Dataa)+1):(length(Dataa)+PredLength)]
pred <- predict(fit.ma, n.ahead = PredLength)
plot.ts(Dataa,xlim=c(1,length(Dataa)+PredLength), ylab="series and forecast",main= paste("MA of order",q))
lines(pred$pred,col="blue", lwd=3,type="l")
lines(pred$pred+1.96*pred$se,col="red",lwd=3,type="l")
lines(pred$pred-1.96*pred$se,col="red",lwd=3,type="l")

# check the accuracy of the interval
MeanPred = as.numeric(pred$pred)  # ECTRACT THE MEAN VALUES from model
SEPred= as.numeric(pred$se)       # ECTRACT THE STD from model
plot(Newdataa,col="blue", ylim=c(-3,3), xlab = "predicting future", ylab = "predicted value vs. real data")
lines(MeanPred,col="red")
lines(MeanPred+1.96*SEPred,col="red",lwd=3,type="l")
lines(MeanPred-1.96*SEPred,col="red",lwd=3,type="l")

## Best model estimate
ts.plot(Dataa,col="blue",ylab="series values",main= paste("MA of order",q))
lines(MA_fit_val,col="red")
legend("bottomright", inset=.05,c("Data","MA fit"),cex=.8,
col=c("blue","red"),pch=c(9,9))
```

```
# ------------------------------------------------------------------------------
# ----------------------- Fit ARMA based on AIC and OLS ------------------------
# ------------------------------------------------------------------------------
Dataa=ZZ[1:160]
d=0
p=1
q=1
fit.arma <- arima(Dataa, order = c(p, d, q),
xreg = NULL, include.mean = FALSE,
transform.pars = TRUE,
fixed = NULL, init = NULL,
method = c("CSS-ML", "ML", "CSS"),
SSinit = c("Gardner1980", "Rossignol2011"),
optim.method = "BFGS",
optim.control = list(), kappa = 1e6)

# analyzing the fitted ARMA
tsdiag(fit.arma)
# Use the identified paramaeters for simulation
Sim1=c()
Sim2=c()
Sim3=c()
Sim4=c()
Sim1 <- arima.sim(list(ar = c(-0.0299), ma = c(-0.9916)),n=length(Dataa),innov=Dataa)
# Sim2 <- arima.sim(list(ar = c(-0.2816), ma = c(-0.7118,-0.2822)),n=length(Dataa),innov=Dataa)
# Sim3 <- arima.sim(list(ar = c(-0.0012,-0.1124), ma = c(-1.0194,0.0408)),n=length(Dataa),innov=Dataa)
# Sim4 <- arima.sim(list(ar = c(-0.5934, -0.9778), ma = c(-0.4283,0.4432,-0.9905)),n=length(Dataa),innov=Dataa)
ARMA_fit_val=Sim1

# calculate the 10-fold training MSE
seg.length= length(Dataa) / 10
err=c()
fff=ARMA_fit_val-Dataa
for(jj in 1:10)
{
ffff = fff [((jj-1)*seg.length)+1:(jj*seg.length)]
err[jj]= sum((ffff[1:16])^2) / seg.length
ffff = c()
}
mean.error=sum(fff)/length(Dataa)
std= (sqrt(sum(( (fff) - mean.error)^2) )) /length(Dataa)
m.s.error= mean(err)

par(mfrow=c(3,2))
# TRAINING PART
qqnorm((ARMA_fit_val-Dataa), main="qq-plot for residuals")
qqline((ARMA_fit_val-Dataa))
ccc=seq(from =-2, to = 2, by = 0.1 )
plot(Dataa,ARMA_fit_val, xlim=c(-3,3), ylim=c(-3,3), xlab="data", ylab="ARMA fit")
lines(ccc,ccc, type="l", col="blue")
# TESTING PART
PredLength=5
Newdataa=ZZ[(length(Dataa)+1):(length(Dataa)+PredLength)]
pred <- predict(fit.arma, n.ahead = PredLength)
plot.ts(Dataa,xlim=c(1,length(Dataa)+PredLength), ylab="series and forecast")
lines(pred$pred,col="blue", lwd=3,type="l")
lines(pred$pred+1.96*pred$se,col="red",lwd=3,type="l")
```

```
lines(pred$pred-1.96*pred$se,col="red",lwd=3,type="l")

# check the accuracy of the interval
MeanPred = as.numeric(pred$pred)  # EXTRACT THE MEAN VALUES from model
SEPred= as.numeric(pred$se)       # EXTRACT THE STD from model
plot(Newdataa,col="blue", ylim=c(-3,3), xlab = "predicting future", ylab = "predicted value vs. real data")
lines(MeanPred,col="red")
lines(MeanPred+1.96*SEPred,col="red",lwd=3,type="l")
lines(MeanPred-1.96*SEPred,col="red",lwd=3,type="l")

## Best model estimate
ts.plot(Dataa,col="blue",ylab="series values",main= paste("ARMA of order","(",p,",",q,")"))
lines(ARMA_fit_val,col="red")
legend("bottomright", inset=.05,c("Data","ARMA fit"),cex=.8,
col=c("blue","red"),pch=c(9,9)

# ------------------------------------------------------------------------------
# ------------------------------ Fit APARCH model ------------------------------
# ------------------------------------------------------------------------------
Dataa=ZZ[1:160]
d=0
p=1
q=1
fit.arma <- arima(Dataa, order = c(p, d, q),
xreg = NULL, include.mean = FALSE,
transform.pars = TRUE,
fixed = NULL, init = NULL,
method = c("CSS-ML", "ML", "CSS"),
SSinit = c("Gardner1980", "Rossignol2011"),
optim.method = "BFGS",
optim.control = list(), kappa = 1e6)

# analyzing the fitted ARMA
tsdiag(fit.arma)

# Use the identified paramaeters for simulation
ARMA_fit_val=c()
ARMA_fit_val <- arima.sim(list(ar = c(-0.0299), ma = c(-0.9916)),n=length(Dataa),innov=Dataa)
# residual errors
res.error = ARMA_fit_val-Dataa
plot.ts(res.error)
# use t-series package
library("tseries")

res.garch <- garch(res.error, order=c(1,1), trace=F) # fit a GARCH(1,1) model
summary(res.garch)
plot(res.garch$series)
acf((res.garch$residuals[3:160])^2)
qqnorm(res.garch$residuals[3:160])
qqline(res.garch$residuals[3:160])

# Also use "rugarch" package for GARCH (1,1) with t distribution
library("rugarch")
garch11t <- ugarchspec(variance.model=list(model="sGARCH", garchOrder=c(1,1)),
mean.model=list(armaOrder=c(0,0,0)), distribution.model="std")
res.error.garch11t <- ugarchfit(data=res.error, spec=garch11t)
res.error.garch11t@fit$coef # this gives us the coefficients, note 6 df for t
```

```
garch11tres <- res.error/res.error.garch11t@fit$sigma # creating residuals

par(mfcol=c(1,2))
acf(as.numeric(garch11tres)^2)
library("qqtest")
qqtest(garch11tres, dist="student", df=6)

# estimation based on the fitted model
e=c()
w=c()
sigma=c()
sigma.square=c()
a0=8.828e-02
a1=3.884e-01
b1=5.662e-16
w = rnorm(1, mean = 0, sd = 0.2)
e[1] = res.error[1]
sigma[1] = e[1] / w
for(i in 2:160)
{
sigma.square[i] = a0 + (a1 * (e[i-1]^2)) + (b1 * (sigma[i-1]^2))
sigma[i] = sqrt(sigma.square[i])
w = rnorm(1, mean = 0, sd = 0.2)
e[i] = sigma[i] * w
}
plot.ts(e)
ee=res.error - e
plot.ts (ee)
m.s.error=sum((ee)^2)/length(res.error)
m.s.error

# APARCH estimation
APARCH_fit_val = ARMA_fit_val + e
m.s.error=sum((APARCH_fit_val-Dataa)^2)/length(Dataa)
mean.error=sum(APARCH_fit_val-Dataa)/length(Dataa)
std= (sqrt( sum(( (APARCH_fit_val - Dataa) - mean.error)^2) )) /length(Dataa)

par(mfrow=c(3,2))
# TRAINING PART
qqnorm((APARCH_fit_val- Dataa), main="qq-plot for residuals")
qqline((APARCH_fit_val- Dataa))
ccc=seq(from =-2, to = 2, by = 0.1 )
plot(Dataa,APARCH_fit_val, xlim=c(-2,2), ylim=c(-2,2), xlab="data", ylab="APARCH fit")
lines(ccc,ccc, type="l", col="blue")
# TESTING PART
PredLength=5
Newdataa=ZZ[(length(Dataa)+1):(length(Dataa)+PredLength)]
pred <- predict(fit.arma, n.ahead = PredLength)
# Pred Garch (1, 1)
eee=c()
w=c()
sigma=c()
sigma.square=c()
a0=8.828e-02
a1=3.884e-01
b1=5.662e-16
w = rnorm(1, mean = 0, sd = 0.2)
```

```
eee[1] = res.error[160]
sigma[1] = eee[1] / w
for(i in 2:(PredLength+1))
{
sigma.square[i] = a0 + (a1 * (e[i-1]^2)) + (b1 * (sigma[i-1]^2))
sigma[i] = sqrt(sigma.square[i])
w = rnorm(1, mean = 0, sd = 0.2)
eee[i] = sigma[i] * w
}

MeanPred= as.numeric(pred$pred) + eee [2:(PredLength+1)]
mean.error=sum(MeanPred[1:4]-Newdataa[1:4])/(length(Newdataa)-1)
SEPred= (sqrt( sum(( (MeanPred[1:4]-Newdataa[1:4]) - mean.error)^2) ))/(length(Newdataa)-1)
plot.ts(Dataa,xlim=c(1,length(Dataa)+PredLength-1), ylab="series and forecast",main= "APARCH model")
lines(MeanPred,col="blue", lwd=3,type="l")
lines(MeanPred+1.96*SEPred,col="red",lwd=3,type="l")
lines(MeanPred-1.96*SEPred,col="red",lwd=3,type="l")

# check the accuracy of the interval
plot(Newdataa,ylim=c(-3,3),xlab ="predicting future",ylab ="predicted value vs. real data",main="APARCH model")
lines(MeanPred,col="red")
lines(MeanPred+1.96*SEPred,col="red",lwd=3,type="l")
lines(MeanPred-1.96*SEPred,col="red",lwd=3,type="l")

## Best model estimate
plot.ts(Dataa, xlim=c(1,length(Dataa)+PredLength),col="blue")
lines(APARCH_fit_val, col="red")

par(mfrow=c(2,2))
acf((res.garch$residuals[3:160])^2)
acf(APARCH_fit_val- Dataa)

# ------------------------------------------------------------------------
# ---------------------------------- Fit DLM -----------------------------
# ------------------------------------------------------------------------
Dataa=Z[1:160]
m0=mean(Dataa)
var(Dataa)
plot.ts(Dataa)
simpledlm <- dlm(m0=mean(Dataa),C0=var(Dataa),FF=1,V=2.59,GG=1,W=3) # create the dlm model
# parameters V and W are calculated using dlmMLE function as follows:
dlmMLE(Dataa,par=c(1.85, 0.10),build=function(x){dlm(m0=mean(Dataa),C0=var(Dataa),FF=1,V=x[1],GG=1,W= x[1]*x[2])})
# then take the first parameter ($par) value as V and the product of the two parameters as W.

Dataa.simpledlm <- dlmFilter(Dataa, simpledlm) # fit the model to the data
Dataa.simpledlm$f # these are the one-step-ahead forecasts i.e. Y2|Y1, Y3|Y2,Y1, Y4|Y3,Y2,Y1, etc

PredLength=5
plot.ts(Dataa.simpledlm$y, xlim=c(1,length(Dataa)+PredLength))
lines(Dataa.simpledlm$f, col="red")
pred <- dlmForecast(Dataa.simpledlm , nAhead = PredLength)
MeanPred = as.numeric(pred$f)         # EXTRACT THE MEAN VALUES from model
SEPred= sqrt(as.numeric(pred$Q))      # EXTRACT THE STD from model
lines(MeanPred,col="blue",lwd=3,type="l")
lines(MeanPred+1.96*SEPred,col="red",lwd=3,type="l")
lines(MeanPred-1.96*SEPred,col="red",lwd=3,type="l")
m.s.error=sum((Dataa.simpledlm$f-Dataa)^2)/length(Dataa)
```

```
mean.error=sum(Dataa.simpledlm$f-Dataa)/length(Dataa)
std= (sqrt( sum(( (Dataa.simpledlm$f - Dataa) - mean.error)^2) )) /length(Dataa)

par(mfrow=c(3,2))
# TRAINING PART
v=Dataa.simpledlm$f-Dataa
qqnorm(v, main="qq-plot for residuals")
qqline(v)
plot.ts(Dataa.simpledlm$y, col="blue", xlim=c(1,length(Dataa)+PredLength))
lines(Dataa.simpledlm$f, col="red")
ccc=seq(from =-2, to = 2, by = 0.1 )
plot(Dataa,Dataa.simpledlm$f, xlim=c(-2,2), ylim=c(-2,2), xlab="data", ylab="DLM fit")
lines(ccc,ccc,type="l", col="blue")
acf(v,main="ACF of residuals")

#------------- differenced version for comparison with the other techniques
plot.ts(ZZ[1:160],col="blue",lwd=1)
lines(diff(Dataa.simpledlm$f),col="red",lwd=1)
m.s.error=sum((diff(Dataa.simpledlm$f)-ZZ[2:160])^2)/length(Dataa)
v=diff(Dataa.simpledlm$f)-ZZ[2:160]
qqnorm(v, main="qq-plot for residuals")
qqline(v)
acf(v)
```

A.2 Codes implemented in Chapter 5

The simulation conducted in Chapter 5 includes MATLAB codes for desired trajectory generation (extracting mean-weight cycle cover of graph, calculating minimum-weight walk using MLE-TMC), a R code for graphical models analysis, and a MATLAB code for speed estimation using absorbing state stochastic process.

A.2.1 MATLAB code for mean-weight cycle cover extraction

```
% Matlab code for forming the means-weight cycle cover of a directed graph

clc; close all; clear;

%%%%%%% Section 5.4.1: Trajectory Estimation %%%%%%%

% Weight matrix of the graph

W=[Inf 1.2 1.2 1.6 1.6 Inf Inf Inf Inf Inf Inf
Inf Inf 1.2 1.2 1.6 1.6 Inf Inf Inf Inf Inf
Inf Inf Inf 1.2 1.2 1.6 1.6 Inf Inf Inf Inf
Inf Inf Inf Inf 1.2 1.2 1.6 1.6 Inf Inf Inf
Inf Inf Inf 2.3 Inf 0.3 1.6 Inf Inf Inf Inf
Inf Inf Inf Inf 0.7 Inf 0.7 Inf Inf Inf Inf
```

```
Inf Inf Inf Inf 1.6 0.3 Inf 2.3 Inf Inf Inf
Inf Inf Inf 1.6 1.6 1.2 1.2 Inf Inf Inf Inf
Inf Inf Inf Inf 1.6 1.6 1.2 1.2 Inf Inf Inf
Inf Inf Inf Inf Inf 1.6 1.6 1.2 1.2 Inf Inf
Inf Inf Inf Inf Inf Inf 1.6 1.6 1.2 1.2 Inf];

% number of edges in the original graph AND its average wieght

kk=find(W~=Inf); W_G=sum(W(kk))/size(kk,1);
G_edgesNum=size(kk,1); kk=[];

%% Calculation of mean-weight cycle cover

% Identify the direct edges going out from each node (i->j)

E=zeros(size(W,2),2,size(W,2)); % Initially, assume that a full connected graph could be available

for i=1:1:size(W,1)
find(W(i,:)~=Inf);
j=find(W(i,:)~=Inf);
E(1:size(j,2),:,i)=[i*ones(size(j,2),1) j'];
end


% Weight of the extracted simple cycles
w_C1=W(4,5)+ W(5,4);
w_C2=W(4,8)+ W(8,4);
w_C3=W(4,8)+ W(8,5)+ W(5,4);
w_C4=W(4,7)+ W(7,8)+ W(8,4);
w_C5=W(4,7)+ W(7,5)+ W(5,4);
w_C6=W(4,7)+ W(7,6)+ W(6,5)+W(5,4);
w_C7=W(5,6)+W(6,5);
w_C8=W(5,7)+W(7,5);
w_C9=W(5,4)+W(4,6)+W(6,5);
w_C10=W(5,4)+W(4,8)+W(8,5);
w_C11=W(5,7)+W(7,6)+W(6,5);
w_C12=W(5,6)+W(6,7)+W(7,5);
w_C13=W(5,4)+W(4,6)+W(6,7)+W(7,5);
w_C14=W(4,5)+W(5,7)+W(7,8)+W(8,4);
w_C15=W(5,4)+W(4,8)+W(8,6)+W(6,5);
w_C16=W(5,4)+W(4,8)+W(8,7)+W(7,5);
w_C17=W(5,6)+W(6,7)+W(7,8)+W(8,4)+W(4,5);
w_C18=W(6,7)+W(7,6);
w_C19=W(6,7)+W(7,8)+W(8,6);
w_C20=W(5,6)+W(6,7)+W(7,8)+W(8,5);
w_C21=W(4,6)+W(6,7)+W(7,8)+W(8,4);
w_C22=W(6,7)+W(7,8)+W(8,5)+W(5,4)+W(4,6);
w_C23=W(7,8)+W(8,7);

% Edges of the extracted simple cycles
Cycles=zeros(22,10);
Cycles(1,1:4)=[4 5 5 4];
Cycles(2,1:4)=[4 8 8 4];
Cycles(3,1:6)=[4 8 8 5 5 4];
Cycles(4,1:6)=[4 7 7 8 8 4];
Cycles(5,1:6)=[4 7 7 5 5 4];
Cycles(6,1:8)=[4 7 7 6 6 5 5 4];
```

```
Cycles(7,1:4)=[5 6 6 5];
Cycles(8,1:4)=[5 7 7 5];
Cycles(9,1:6)=[5 4 4 6 6 5];
Cycles(10,1:6)=[5 7 7 6 6 5];
Cycles(11,1:6)=[5 6 6 7 7 5];
Cycles(12,1:8)=[5 4 4 6 6 7 7 5];
Cycles(13,1:8)=[4 5 5 7 7 8 8 4];
Cycles(14,1:8)=[5 4 4 8 8 6 6 5];
Cycles(15,1:8)=[5 4 4 8 8 7 7 5];
Cycles(16,1:10)=[5 6 6 7 7 8 8 4 4 5];
Cycles(17,1:4)=[6 7 7 6];
Cycles(18,1:6)=[6 7 7 8 8 6];
Cycles(19,1:8)=[5 6 6 7 7 8 8 5];
Cycles(20,1:8)=[4 6 6 7 7 8 8 4];
Cycles(21,1:10)=[6 7 7 8 8 5 5 4 4 6];
Cycles(22,1:4)=[7 8 8 7];


w_ave=[1.75 1.6 1.83 1.83 1.83 1.22 0.5 1.6 1.4 0.86 0.86 1.45 1.67 1.45 1.67 1.22 0.5 1.4 1.22 1.45 1.62 1.75];

D=[Cycles w_ave'];
[o 1]=sort(D(:,11),'ascend');
D=D(1,:); % sorted cycles based on mean-weight
o=[];

% Forming the mean-weight cycle set
Cycles_container=D;
C=zeros(30,10);
t=1;
while t<=size(D,1) % a large number
if t==1
C(t,:)=D(t,1:10);
else
nodes=zeros(1,size(W,2)); % at-most we can have size(W,1) nodes
% Identifying the nodes of the previous members of C
for i=1:1:t-1
o=find(C(i,:)~=0);
o=o(1,1:2:end);
sss(i)=size(o,2); % identify the number of nodes in each cycle
if i==1
nodes(1:sss(i))=C(i,o);
else
nodes(1+sss(i-1):sss(i-1)+sss(i))=C(i,o);
end
end
% Retain non-zero elements of nodes
iii=find(nodes==0); nodes(iii)=[];
% Remove the duplicated nodes
nodes=unique(nodes);
% Removing nodes and corresponding edges of the previous members of C
lll=[];
for uu=1:1:size(nodes,2)
for ii=1:1:size(Cycles,2) % check all of the simple cycles
ppp=find(Cycles(:,ii)==nodes(uu));
lll=[lll; ppp];
end
end
```

```
lll=unique(lll); % remove duplicate / repeated numbers (nodes) from the vector
Cycles_container(lll,:)=[];
end

% A condition for stopping the search
if  t==1
t=t+1;
elseif isempty(Cycles_container)==1 & t~=1
break
elseif t~=1
% Select the member with lowest weight
C(t,:)=Cycles_container(1,1:10); % since elements in container are already sorted
% Restore the set of simple cycles and continue
Cycles_container=D;
t=t+1;
end
end

% remove duplicated rows from set
C = unique(C,'rows');
```

A.2.2 MATLAB code for minimum-weight walk calculation via MLE-TMC

```
% Matlab Code for determining minimum weight walk using MLE-TMC

clc; close all; clear;


% Load the weigh matrix

W=[Inf 1.2 1.2 1.6 1.6 Inf Inf Inf Inf Inf Inf
Inf Inf 1.2 1.2 1.6 1.6 Inf Inf Inf Inf Inf
Inf Inf Inf 1.2 1.2 1.6 1.6 Inf Inf Inf Inf
Inf Inf Inf Inf 1.2 1.2 1.6 1.6 Inf Inf Inf
Inf Inf Inf 2.3 Inf 0.3 1.6 Inf Inf Inf Inf
Inf Inf Inf Inf 0.7 Inf 0.7 Inf Inf Inf Inf
Inf Inf Inf Inf 1.6 0.3 Inf 2.3 Inf Inf Inf
Inf Inf Inf 1.6 1.6 1.2 1.2 Inf Inf Inf Inf
Inf Inf Inf Inf 1.6 1.6 1.2 1.2 Inf Inf Inf
Inf Inf Inf Inf Inf 1.6 1.6 1.2 1.2 Inf Inf
Inf Inf Inf Inf Inf Inf 1.6 1.6 1.2 1.2 Inf];

W=W(1:8,1:8);

% Load the simple cycles matrix and corresponding mean weights

Cycles=zeros(22,10);
Cycles(1,1:4)=[4 5 5 4];
Cycles(2,1:4)=[4 8 8 4];
Cycles(3,1:6)=[4 8 8 5 5 4];
```

```
Cycles(4,1:6)=[4 7 7 8 8 4];
Cycles(5,1:6)=[4 7 7 5 5 4];
Cycles(6,1:8)=[4 7 7 6 6 5 5 4];
Cycles(7,1:4)=[5 6 6 5];
Cycles(8,1:4)=[5 7 7 5];
Cycles(9,1:6)=[5 4 4 6 6 5];
Cycles(10,1:6)=[5 7 7 6 6 5];
Cycles(11,1:6)=[5 6 6 7 7 5];
Cycles(12,1:8)=[5 4 4 6 6 7 7 5];
Cycles(13,1:8)=[4 5 5 7 7 8 8 4];
Cycles(14,1:8)=[5 4 4 8 8 6 6 5];
Cycles(15,1:8)=[5 4 4 8 8 7 7 5];
Cycles(16,1:10)=[5 6 6 7 7 8 8 4 4 5];
Cycles(17,1:4)=[6 7 7 6];
Cycles(18,1:6)=[6 7 7 8 8 6];
Cycles(19,1:8)=[5 6 6 7 7 8 8 5];
Cycles(20,1:8)=[4 6 6 7 7 8 8 4];
Cycles(21,1:10)=[6 7 7 8 8 5 5 4 4 6];
Cycles(22,1:4)=[7 8 8 7];

w_ave=[1.75 1.6 1.83 1.83 1.83 1.22 0.5 1.6 1.4 0.86 0.86 1.45 1.67 1.45 1.67 1.22 0.5 1.4 1.22 1.45 1.62 1.75];
D=[Cycles w_ave'];

% Load the mean-weight cycle cover
C=[5 6 6 5 0 0 0 0; 6 7 7 6 0 0 0 0; 4 6 6 7 7 8 8 4];
w_C=[1.75 0.5 1.45];
% Determine the length of walk
k=size(W,2);
t=(k^2)+(k*(k+1)/2);

% Set the Anchor node
A=[1 2 2 3 3 4 4 5 5 6 6 7 7 8];
% size (number of edges) of A
pp=unique(A);
jj=find(pp==0); pp(jj)=[]; % check whether there is any 0
A_edge_num=size(pp,2)-1; % "minus 1" BECAUSE it is not a cycle
w_A=1.15;

% calculate the number of edges in the free cycle
for ii=1:1:size(Cycles,1)
% distinct nodes in each member of free cycle
pp=unique(Cycles(ii,:));
jj=find(pp==0); pp(jj)=[]; % check if there is any 0 and remove it
% number of edges
Cycles_edge_num(ii)=size(pp,2);
end

% Add cycle sizes to archive D
D=[D Cycles_edge_num'];

%% Calculation of the walk
S=zeros(size(C,1),40); % this threshold "length of column" may change
m=zeros(1,size(C,1));
r=zeros(1,size(C,1));
for ii=1:1:size(C,1) % loop over all members of mean weight cycle cover

% distinct nodes in each member of C
```

```
pp=unique(C(ii,:));
jj=find(pp==0); pp(jj)=[]; % check if there is any 0 and remove it
% number of edges
C_edge_num(ii)=size(pp,2);
kkk=C_edge_num(ii);
% find the number of repetitions
m=floor((t-A_edge_num-(kkk+1))/kkk); % -2 is used to make sure the remainder is not "1"
M(ii)=m;
% find the remaining edges
r=t-A_edge_num-(m*kkk);
R(ii)=r;
% fill the remaining edges with memebers of free Cycle
qq=find(D(:,end)==r);
LL=D(qq,:);
[i qqq]=sort(LL(:,end-1),'ascend');
LL=LL(qqq,:);
w_LLL=LL(1,end-1);
LLL=LL(1,1:end-2); jjjj=find(LLL==0); LLL(jjjj)=[];
S(ii,1:(size(A,2)+size(LLL,2)))=[A LLL]; % form S by adding anchor walk to selected edge
w_S(ii)=w_A+w_LLL;
% total weight
w_total(ii)=w_S(ii)+(M(ii)*w_C(ii));
end

% select the walk with minimum weight
[lll ooo]=min(w_total);
SS=S(ooo,:); jjjj=find(SS==0); SS(jjjj)=[];
CC=C(ooo,:); jjjj=find(CC==0); CC(jjjj)=[];
% form the final walk based on the obtained results
s0=1;
sf=6;
repeat_CC = repmat(CC', M(ooo), 1);
SSS = [SS(1:10) repeat_CC' SS(11:end)];
S_opt=[s0 SSS(1:2:end) sf];

% plot the desired trajectory
displace=[-0.05 -0.04 -0.03 -0.02 -0.01 0 0.01 0.02 0.03 0.04 0.05];
trajectory=S_opt;
for ii=1:1:size(displace,2)
jjj=find(S_opt==ii);
trajectory(jjj)=displace(ii);
end
plot(trajectory)
```

A.2.3 R code for graphical model analysis

```
# R Code for directed graph in Section 5.4.1: Trajectory Estimation

library(igraph)

g=graph.formula(1-+2,1-+3,1-+4,1-+5,2-+3,2-+4,2-+5,2-+6,3-+4,3-+5,3-+6,3-+7,4-+5,4-+6,4-+7,4-+8,
5-+4,5-+6,5-+7,6-+5,6-+6,6-+7,7-+5,7-+6,7-+8,8-+4,8-+5,8-+6,8-+7,9-+5,9-+6,9-+7,9-+8,10-+6,
10-+7,10-+8,10-+9,11-+7,11-+8,11-+9,11-+10)
```

```
layout=matrix(c(2,12, 3.5,9.5, 9,6, 0,9, 8,14, 12,7, 7,4, 2,5, 14,5, 10,3, 10,0),byrow=TRUE,nrow=14) # location
plot.igraph(g,layout=layout,edge.color="black",vertex.color="white", vertex.label=c("1","2","3","4","5","6",
"7","8","9","10","11"),vertex.size=20,vertex.label.cex=1, vertex.label.color="black",edge.width=1,
edge.arrow.size=0.6,edge.arrow.width=1.2,edge.curved=TRUE)

# Plot the graph without cycles in the original graph
g=graph.formula(1-+2,1-+3,1-+4,1-+5,2-+3,2-+4,2-+5,2-+6,3-+4,3-+5,3-+6,3-+7,9-+5,9-+6,9-+7,9-+8,10-+6,
10-+7,10-+8,10-+9,11-+7,11-+8,11-+9,11-+10)
layout=matrix(c(2,12, 3.5,9.5, 9,6, 0,9, 8,14, 12,8, 7,4, 2,5, 14,5, 10,1.5, 10,-1),byrow=TRUE,nrow=14) # location
plot.igraph(g,layout=layout,edge.color="black",vertex.color="white", vertex.label=c("1","2","3","4","5",
"6","7","8","9","10","11"),vertex.size=20,vertex.label.cex=1, vertex.label.color="black",edge.width=1,
edge.arrow.size=0.6,edge.arrow.width=1.2)

## --------- Plot the cycles

# C1
g=graph.formula(1,2,3,4-+5,5-+4,6,7,8,9,10,11)
layout=matrix(c(2,13, 3.5,8.5, 9,6, 0,9, 8,14, 12,7, 7,4, 2,5, 14,5, 10,3, 10,0),byrow=TRUE,nrow=14) # location
plot.igraph(g,layout=layout,edge.color="black",vertex.color="white", vertex.label=c("1","2","3","4","5",
"6","7","8","9","10","11"),vertex.size=20,vertex.label.cex=1, vertex.label.color="black",edge.width=1,
edge.arrow.size=0.6,edge.arrow.width=1.2,edge.curved=TRUE)

# C2
g=graph.formula(1,2,3,4,5,6,7,8,9,10,11,4-+8,8-+4)
layout=matrix(c(2,13, 3.5,8.5, 9,6, 0,9, 8,14, 12,7, 7,4, 2,5, 14,5, 10,3, 10,0),byrow=TRUE,nrow=11) # location
plot.igraph(g,layout=layout,edge.color="black",vertex.color="white", vertex.label=c("1","2","3","4","5","6",
"7","8","9","10","11"),vertex.size=20,vertex.label.cex=1, vertex.label.color="black",edge.width=1,
edge.arrow.size=0.6,edge.arrow.width=1.2,edge.curved=TRUE)

# C3
g=graph.formula(1,2,3,4,5,6,7,8,9,10,11,4-+8,8-+5,5-+4)
layout=matrix(c(2,13, 3.5,8.5, 9,6, 0,9, 8,14, 12,7, 7,4, 2,5, 14,5, 10,3, 10,0),byrow=TRUE,nrow=11) # location
plot.igraph(g,layout=layout,edge.color="black",vertex.color="white", vertex.label=c("1","2","3","4","5","6",
"7","8","9","10","11"),vertex.size=20,vertex.label.cex=1, vertex.label.color="black",edge.width=1,
edge.arrow.size=0.6,edge.arrow.width=1.2,edge.curved=TRUE)

# C7
g=graph.formula(1,2,3,4,5,6,7,8,9,10,11,5-+7,7-+5)
layout=matrix(c(2,13, 3.5,8.5, 9,6, 0,9, 8,14, 12,7, 7,4, 2,5, 14,5, 10,3, 10,0),byrow=TRUE,nrow=11) # location
plot.igraph(g,layout=layout,edge.color="black",vertex.color="white", vertex.label=c("1","2","3","4","5","6",
"7","8","9","10","11"),vertex.size=20,vertex.label.cex=1, vertex.label.color="black",edge.width=1,
edge.arrow.size=0.6,edge.arrow.width=1.2,edge.curved=TRUE)

##---------- Plot the graph resulting from removing nodes from cycle cover
g=graph.formula(1,2,3,4,5,6,7,8,9,10,11,1-+2,1-+3,2-+3,10-+9,11-+9,11-+10)
layout=matrix(c(2,12, 3.5,9.5, 9,6, 0,9, 8,14, 12,8, 7,4, 2,5, 14,5, 10,1.5, 10,-1),byrow=TRUE,nrow=14) # location
plot.igraph(g,layout=layout,edge.color="black",vertex.color="white", vertex.label=c("1","2","3","4","5","6",
"7","8","9","10","11"),vertex.size=20,vertex.label.cex=1, vertex.label.color="black",edge.width=1,
edge.arrow.size=0.6,edge.arrow.width=1.2)
```

A.2.4 MATLAB code for absorbing state speed estimation

```
% Matlab Code for generating random walk from cannocically decomposed probability
```

```
% matrix of an absorbing state stochastic process

clc; clear; close all;
tic,
%%%%%%% Code for Section 5.4.2: Vehicle Speed Estimation %%%%%%%
Case=2;

%%
if Case==1
%% 1st simulation scenario
N=10; % horizon length of interest
% Transient speeds
V_trans=[35,38,43,45];
% Absorbing speeds
V_absorb=[40,41];
% The information coming from canonically decomposed probability matrix
Q=[0.1 0.2 0.2 0.1
0.1 0.2 0.3 0.1
0.2 0.2 0.2 0.2
0.1 0.3 0.2 0.1];
R=[0.3 0.1
0.1 0.2
0.1 0.1
0.1 0.2];
P=[Q R];
% Generating Random walk until absorbtiomn
V=[V_trans V_absorb];
t=1;
j=2; % the element number in transient set
s(t)=V(j);
while t<=1000  % Note that if the process is absorbed, loop will terminate automatically
t=t+1;
l=P(j,:);
% Draw a random number from UNIF(0,1)
u=rand;
for i=1:1:size(P,2)
if i==1
if 0 <= u & u  <= l(1)
j=i;
end
else
p1=sum(l(1:i-1));
p2=sum(l(1:i));
if p1 < u & u <= p2
j=i;
end
end
end
s(t)=V(j);
if s(t)==V_absorb(1) | s(t)==V_absorb(2)
break
end
end

elseif Case==2

%% 2nd simulation scenario
```

```
N=20; % horizon length of interest
% Transient speeds
V_trans=[35,36,37,38,42,43,44,45];
% Absorbing speeds
V_absorb=[39,40,41];
% The information coming from canonically decomposed probability matrix
Q=[0.1 0.2 0.1 0.1 0.05 0.05 0.05 0.1
0.1 0.1 0.05 0.1 0.1 0.15 0.1 0.15
0.1 0.05 0.05 0.1 0.1 0.05 0.1 0.05
0.1 0.1 0.1 0.1 0.05 0.1 0.05 0.1
0.05 0.05 0.05 0.1 0.1 0.05 0.1 0.1
0.05 0.1 0.05 0.1 0.1 0.05 0.05 0.1
0.1 0.1 0.1 0.1 0.05 0.05 0.05 0.05
0.15 0.05 0.1 0.1 0.05 0.15 0.1 0.05];
R=[0.05 0.1 0.1
0.05 0.05 0.05
0.1 0.2 0.1
0.1 0.1 0.1
0.1 0.2 0.1
0.15 0.15 0.1
0.15 0.1 0.15
0.05 0.15 0.05];
P=[Q R];
% Generating Random walk until absorbtiomn
V=[V_trans V_absorb];
t=1;
j=2; % the element number in transient set
s(t)=V(j);
while t<=1000  % Note that if the process is absorbed, loop will terminate automatically
t=t+1;
l=P(j,:);
% Draw a random number from UNIF(0,1)
u=rand;
for i=1:1:size(P,2)
if i==1
if 0 <= u & u  <= l(1)
j=i;
end
else
p1=sum(l(1:i-1));
p2=sum(l(1:i));
if p1 < u & u <= p2
j=i;
end
end
end
s(t)=V(j);
if s(t)==V_absorb(1) | s(t)==V_absorb(2) | s(t)==V_absorb(3)
break
end
end
end

%%  Make the walk compatible with LBMPC based on horizon length
kk=s(end);
if size(s,2)< N
for i = size(s,2)+1:1:N
```

```
s(i)=kk;
end
elseif size(s,2)> N
s(N+1:end)=[];
end
toc,

%% Visualization
plot(s);
```

A.3 Codes implemented in Chapter 7

The simulation conducted in Chapter 7 includes MATLAB codes for quantifying the uncertainty due to sensor malfunction as well as R and MATLAB codes for designing oracle for learning-based model predictive controller (LBMPC).

A.3.1 MATLAB code for Bayesian dynamic model and booststrapping for uncertainty quantification

```
% Bayesian Dyamic model and booststrap for quatifying "e"

clc; clear; close all;

global A B F
F= [A B];
load('wave')

%% Nonliner model based on equations of motion

%----- System parameters
V=20; % Vehicle Forward Velocity
% Vehicle's Body parameters
Mb=580; Jb=1100; m_u1=40; m_u2=35.5; L1=1.5; L2=1;
Delta=(1/Mb)+((L1^2)/Jb);
Etta=(1/Mb)-((L1*L2)/Jb);
Zetta=(1/Mb)+((L2^2)/Jb);
% Dampers and Springs coefficients
K_l_s1=14000;   K_l_s2=K_l_s1;
K_nl_s1=2.35e4;  K_nl_s2=K_nl_s1;
K_t1=190e3;      K_t2=K_t1;
C_l_s1=800;      C_l_s2=700;
C_nl_s1=400;     C_nl_s2=C_nl_s1;
C_sym_s1=400;    C_sym_s2=C_sym_s1;
C_t1=80;         C_t2=70;
```

```
%----- Initial state values
% Vehicle body states
xb1=0; xb2=0;
xc=0;  teta_c=0;
% Suspension system and sprung mass systems
xu1=0;  xu2=0;
% Disturbances ** {measurement noises, road irregularities}
xr1(1)=0; xr2(1)=0; % road disturbances

%% Initialization of States
X(1,1)=xb1;        %
X(2,1)=0;          % X(2)=xb_dot_1
X(3,1)=xb2;        %
X(4,1)=0;          % X(4)=xb_dot_2
X(5,1)=xu1;        %
X(6,1)=0;          % X(6)=xu_dot_1
X(7,1)=xu2;        %
X(8,1)=0;          % X(8)=xu_dot_2

%% Operating time
t0=0.0;
dt=0.001;
ng=5000;     % Number of working points
ts=ng*dt;     % Active control duration

%% Bayesian Dynamic Model and Bootstrap
for b=1:1:100 % number of bootstrap sampling
% Initial state and initial actuation signal
U=ones(2,5001)*1000;
mu(:,1)=[X(:,1);U(:,1)];
% mu = [1=xb1; 2=xb_dot_1; 3=xb2; 4=xb_dot_2; 5=xu1; x6=u_dot_1; 7=xu2; 8=xu_dot_2 Fa_1 Fa_2]
Xsystem(:,1)=mu(1:8,1);

for n=1:1:ng
% ----- Equal Parameters for liniarization
z1=mu(5,n)-mu(1,n);
z2=mu(7,n)-mu(3,n);
z_dot_1=mu(6,n)-mu(2,n);
z_dot_2=mu(8,n)-mu(4,n);
eps=2.2204e-16;
K_eq_s1=3*K_nl_s1*(z1^2);
K_eq_s2=3*K_nl_s1*(z2^2);
C_eq_s1=(-C_sym_s1*(z_dot_1/abs(z_dot_1+eps)))+(C_nl_s1*(0.5*(z_dot_1/abs(z_dot_1+eps))+...
*((abs(z_dot_1+eps))^-0.5)*sign(z_dot_1)));
C_eq_s2=(-C_sym_s2*(z_dot_2/abs(z_dot_2+eps)))+(C_nl_s2*(0.5*(z_dot_2/abs(z_dot_2+eps))+...
*((abs(z_dot_2+eps))^-0.5)*sign(z_dot_2)));
Kstar_s1=K_l_s1+K_eq_s1;
Kstar_s2=K_l_s2+K_eq_s2;
Cstar_s1=1*(C_l_s1+C_eq_s1);
Cstar_s2=1*(C_l_s2+C_eq_s2);

% f1 and f2 for calculation of body acceleration
f1_System(n)=(Kstar_s1*z1)+(Cstar_s1*z_dot_1);
f2_System(n)=(Kstar_s2*z2)+(Cstar_s2*z_dot_2);

%----- State Space Parameters Updating
A=[0 1 0 0 0 0 0 0
```

```
-(Delta*Kstar_s1) -(Delta*Cstar_s1) -(Etta*Kstar_s2) -(Etta*Cstar_s2) (Delta*Kstar_s1) (Delta*Cstar_s1)...
 (Etta*Kstar_s2) (Etta*Cstar_s2)
0 0 0 1 0 0 0 0
-(Etta*Kstar_s1) -(Etta*Cstar_s1) -(Zetta*Kstar_s2) -(Zetta*Cstar_s2) (Etta*Kstar_s1) (Etta*Cstar_s1)...
 (Zetta*Kstar_s2) (Zetta*Cstar_s2)
0 0 0 0 0 1 0 0
(Kstar_s1/m_u1) (Cstar_s1/m_u1) 0 0 (-K_t1-Kstar_s1)/m_u1 (-C_t1-Cstar_s1)/m_u1 0 0
0 0 0 0 0 0 0 1
0 0 (Kstar_s2/m_u2) (Cstar_s2/m_u2) 0 0 (-K_t2-Kstar_s2)/m_u2 (-C_t2-Cstar_s2)/m_u2];
B=[0 0
-dt*Delta -dt*Etta
0 0
-dt*Etta -dt*Zetta
0 0
(1/m_u1) 0
0 0
0 (1/m_u2)];
% The constraint of the dynamic model
if n > 30
% Calculate stabilization gain matrix K
pole=[-2 -0.5 -1 -5 -3 -1.5 -4 -6];
K_gain = place(A,B,pole);
% Explanatory signal obtained by random combination of considered waves
aa=rand(1,size(wave,1));
bb=aa/sum(aa); % randomly generated numbers are normalized such that they sum up to 1
Ue(n)=bb*wave(:,n);
% Calculate actuation signal
U(:,n)=(-K_gain*mu(1:8,n));%+Ue(n);
% Check for possible bound violation
if U(1,n) > 5000
U(1,n)=5000;
elseif U(1,n)<-5000
U(1,n)=-5000;
end
if U(2,n) > 5000
U(2,n)=5000;
elseif U(2,n)<-5000
U(2,n)=-5000;
end
mu(9:10,n)=U(:,n);
else
mu(9:10,n)=U(:,n);
end

% updating Bayesian dynamic model
V=[0.0007^2 0.000001 0 0 0 0 0 0 0 0
0.000001 0.0126^2 0 0 0 0 0 0 0 0
0 0 0.0007^2 0.000003 0 0 0 0 0 0
0 0 0.000003 0.0090^2 0 0 0 0 0 0
0 0 0 0 0.0020^2 0.00010 0 0 0 0
0 0 0 0 0.00010 0.1282^2 0 0 0 0
0 0 0 0 0 0 0.0022^2 0.00010 0 0
0 0 0 0 0 0 0.00010 0.1429^2 0 0
0 0 0 0 0 0 0       0      100 0
0 0 0 0 0 0 0       0      0 100];
W=blkdiag(0.0001,0.0076,0.0001,0.0057,0.0001,0.0284,0.0001,0.0281);
v=mvnrnd(zeros(1,10),V); v=v';
```

```
w=mvnrnd(zeros(1,8),W); w=w';
mu(:,n)=mu(:,n)+v;
mu(1:8,n+1)=((dt)*(F*mu(:,n)))+mu(1:8,n)+w;
Xsystem(:,n+1)=mu(1:8,n+1);
end

%---- Calculate the error signal
kk=1;
for jj=2:1:size(Xsystem,2)
e(:,kk)= Xsystem(:,jj)-((dt*((A*Xsystem(:,jj-1)) + (B*U(:,jj-1))))+Xsystem(:,jj-1));
kk=kk+1;
end
E_boot(:,:,b)=e;
Xsystem=[]; mu=[]; U=[]; e=[];
end

% Bootstrap Estimate
for n=1:1:ng
e_boot(1,n)=sum(E_boot(1,n,:))/b;
e_boot(2,n)=sum(E_boot(2,n,:))/b;
e_boot(3,n)=sum(E_boot(3,n,:))/b;
e_boot(4,n)=sum(E_boot(4,n,:))/b;
e_boot(5,n)=sum(E_boot(5,n,:))/b;
e_boot(6,n)=sum(E_boot(6,n,:))/b;
e_boot(7,n)=sum(E_boot(7,n,:))/b;
e_boot(8,n)=sum(E_boot(8,n,:))/b;
end

% Bootstrap standard deviation
for n=1:1:ng
std_boot(1,n)= sqrt((1/(b-1))*(sum((e_boot(1,n)-E_boot(1,n,:)).^2)));
std_boot(2,n)= sqrt((1/(b-1))*(sum((e_boot(2,n)-E_boot(2,n,:)).^2)));
std_boot(3,n)= sqrt((1/(b-1))*(sum((e_boot(3,n)-E_boot(3,n,:)).^2)));
std_boot(4,n)= sqrt((1/(b-1))*(sum((e_boot(4,n)-E_boot(4,n,:)).^2)));
std_boot(5,n)= sqrt((1/(b-1))*(sum((e_boot(5,n)-E_boot(5,n,:)).^2)));
std_boot(6,n)= sqrt((1/(b-1))*(sum((e_boot(6,n)-E_boot(6,n,:)).^2)));
std_boot(7,n)= sqrt((1/(b-1))*(sum((e_boot(7,n)-E_boot(7,n,:)).^2)));
std_boot(8,n)= sqrt((1/(b-1))*(sum((e_boot(8,n)-E_boot(8,n,:)).^2)));
end

% Bootstrap 95% CI intervals
for n=1:1:ng
CI_boot(1,n,1:2)=[e_boot(1,n)-(1.96*std_boot(1,n)); e_boot(1,n)+(1.96*std_boot(1,n))];
CI_boot(2,n,1:2)=[e_boot(2,n)-(1.96*std_boot(2,n)); e_boot(2,n)+(1.96*std_boot(2,n))];
CI_boot(3,n,1:2)=[e_boot(3,n)-(1.96*std_boot(3,n)); e_boot(3,n)+(1.96*std_boot(3,n))];
CI_boot(4,n,1:2)=[e_boot(4,n)-(1.96*std_boot(4,n)); e_boot(4,n)+(1.96*std_boot(4,n))];
CI_boot(5,n,1:2)=[e_boot(5,n)-(1.96*std_boot(5,n)); e_boot(5,n)+(1.96*std_boot(5,n))];
CI_boot(6,n,1:2)=[e_boot(6,n)-(1.96*std_boot(6,n)); e_boot(6,n)+(1.96*std_boot(6,n))];
CI_boot(7,n,1:2)=[e_boot(7,n)-(1.96*std_boot(7,n)); e_boot(7,n)+(1.96*std_boot(7,n))];
CI_boot(8,n,1:2)=[e_boot(8,n)-(1.96*std_boot(8,n)); e_boot(8,n)+(1.96*std_boot(8,n))];
end

time=t0:dt:ts; time(end)=[];

figure,
subplot(4,2,1), plot(time,e_boot(1,:)), hold on, plot(time,CI_boot(1,:,1)), hold on, plot(time,CI_boot(1,:,2)),
subplot(4,2,2), plot(time,e_boot(2,:)), hold on, plot(time,CI_boot(2,:,1)), hold on, plot(time,CI_boot(2,:,2)),
```

```
subplot(4,2,3), plot(time,e_boot(3,:)), hold on, plot(time,CI_boot(3,:,1)), hold on, plot(time,CI_boot(3,:,2)),
subplot(4,2,4), plot(time,e_boot(4,:)), hold on, plot(time,CI_boot(4,:,1)), hold on, plot(time,CI_boot(4,:,2)),
subplot(4,2,5), plot(time,e_boot(5,:)), hold on, plot(time,CI_boot(5,:,1)), hold on, plot(time,CI_boot(5,:,2)),
subplot(4,2,6), plot(time,e_boot(6,:)), hold on, plot(time,CI_boot(6,:,1)), hold on, plot(time,CI_boot(6,:,2)),
subplot(4,2,7), plot(time,e_boot(7,:)), hold on, plot(time,CI_boot(7,:,1)), hold on, plot(time,CI_boot(7,:,2)),
subplot(4,2,8), plot(time,e_boot(8,:)), hold on, plot(time,CI_boot(8,:,1)), hold on, plot(time,CI_boot(8,:,2)),

% ------ Non-parametric calculation of 95% interval to form the polytope
[e_sort(1,:), ind1]=sort(e_boot(1,:),'ascend');
[e_sort(2,:), ind2]=sort(e_boot(2,:),'ascend');
[e_sort(3,:), ind3]=sort(e_boot(3,:),'ascend');
[e_sort(4,:), ind4]=sort(e_boot(4,:),'ascend');
[e_sort(5,:), ind5]=sort(e_boot(5,:),'ascend');
[e_sort(6,:), ind6]=sort(e_boot(6,:),'ascend');
[e_sort(7,:), ind7]=sort(e_boot(7,:),'ascend');
[e_sort(8,:), ind8]=sort(e_boot(8,:),'ascend');

% Plot
figure,
subplot(4,2,1), plot(time,e_sort(1,:)), hold on, plot(time,CI_boot(1,ind1,1)), hold on, plot(time,CI_boot(1,ind1,2)),
subplot(4,2,2), plot(time,e_sort(2,:)), hold on, plot(time,CI_boot(2,ind2,1)), hold on, plot(time,CI_boot(2,ind2,2)),
subplot(4,2,3), plot(time,e_sort(3,:)), hold on, plot(time,CI_boot(3,ind3,1)), hold on, plot(time,CI_boot(3,ind3,2)),
subplot(4,2,4), plot(time,e_sort(4,:)), hold on, plot(time,CI_boot(4,ind4,1)), hold on, plot(time,CI_boot(4,ind4,2)),
subplot(4,2,5), plot(time,e_sort(5,:)), hold on, plot(time,CI_boot(5,ind5,1)), hold on, plot(time,CI_boot(5,ind5,2)),
subplot(4,2,6), plot(time,e_sort(6,:)), hold on, plot(time,CI_boot(6,ind6,1)), hold on, plot(time,CI_boot(6,ind6,2)),
subplot(4,2,7), plot(time,e_sort(7,:)), hold on, plot(time,CI_boot(7,ind7,1)), hold on, plot(time,CI_boot(7,ind7,2)),
subplot(4,2,8), plot(time,e_sort(8,:)), hold on, plot(time,CI_boot(8,ind8,1)), hold on, plot(time,CI_boot(8,ind8,2)),

% ------ Empirical CDFs
[F_1,e_1] = ecdf(e_sort(1,:));
[F_2,e_2] = ecdf(e_sort(2,:));
[F_3,e_3] = ecdf(e_sort(3,:));
[F_4,e_4] = ecdf(e_sort(4,:));
[F_5,e_5] = ecdf(e_sort(5,:));
[F_6,e_6] = ecdf(e_sort(6,:));
[F_7,e_7] = ecdf(e_sort(7,:));
[F_8,e_8] = ecdf(e_sort(8,:));

figure,
subplot(4,2,1), plot(e_1,F_1,'b'),
subplot(4,2,2), plot(e_2,F_2,'b'),
subplot(4,2,3), plot(e_3,F_3,'b'),
subplot(4,2,4), plot(e_4,F_4,'b'),
subplot(4,2,5), plot(e_5,F_5,'b'),
subplot(4,2,6), plot(e_6,F_6,'b'),
subplot(4,2,7), plot(e_7,F_7,'b'),
subplot(4,2,8), plot(e_8,F_8,'b'),

% Find the bounds of polytope
[e_1(max(find(F_1<=0.025))) e_1(min(find(F_1>=0.975)))]
[e_2(max(find(F_2<=0.025))) e_2(min(find(F_2>=0.975)))]
[e_3(max(find(F_3<=0.025))) e_3(min(find(F_3>=0.975)))]
[e_4(max(find(F_4<=0.025))) e_4(min(find(F_4>=0.975)))]
[e_5(max(find(F_5<=0.025))) e_5(min(find(F_5>=0.975)))]
[e_6(max(find(F_6<=0.025))) e_6(min(find(F_6>=0.975)))]
[e_7(max(find(F_7<=0.025))) e_7(min(find(F_7>=0.975)))]
[e_8(max(find(F_8<=0.025))) e_8(min(find(F_8>=0.975)))]
```

A.3.2 MATLAB code for adaptive neuro-fuzzy inference system

```
% Adaptive Neuro Fuzzy Inference System (ANFIS) code

clc; clear; close all;

%% Data
load('Inp_dataset'),
load('Outp_dataset'),

% 10 fold-forward chaining
IND= [1:500;501:1000;1001:1500; 1501:2000; 2001:2500; 2501:3000; 3001:3500; 3501:4000; 4001:4500; 4501:5000];

for rr=1:1:10
xdata=Inp_dataset(IND(rr,:),[1 2 5 6 7]);
ydata=Outp_dataset(IND(rr,:),7);
coeff = pca(xdata);
xdata=xdata*coeff;
xtrain=xdata;
ytrain=ydata;
n=size(xtrain,2); % Number of input variables
m=size(ytrain,2); % Number of output variables

% Normalizing the data base within unity [0,1]
for ii=1:1:n
Max_X_train=max(xtrain(:,ii)); Min_X_train=min(xtrain(:,ii));
xtrain(:,ii)=(xtrain(:,ii)-(ones(size(xtrain,1),1)*Min_X_train))/(Max_X_train-Min_X_train);
end

tic,
%% Training
TrainData=[xtrain ytrain];
% Create FIS Rough Structure
nInputMFs=[2 2 2 2 2];
InputMFType=char('gaussmf','gaussmf','gaussmf','gaussmf','gaussmf');
fis=genfis1(TrainData,nInputMFs,InputMFType);
% Train ANFIS
[fis err]=anfis(TrainData,fis,60);

%% Results
Output=evalfis(TrainData(:,1:end-1),fis);
toc,

error=(TrainData(:,end)-Output);
CI_lb=Output-(1.96 * std(Output));
CI_ub=Output+(1.96 * std(Output));
% Calculation of performane metrics
% MAE
MAE(rr)=sum(abs(error))/size(xtrain,1);
% MaxE
MaxE(rr)=max(abs(error));
% std.
```

```
Std(rr)=std(error);
end

% MAE
disp('MEA')
mean(MAE)
% MaxE
disp('MaxE')
mean(MaxE)
% std.
disp('std.')
mean(Std)

% --- plot
figure,
% correlation plot
subplot(2,2,1), plot(Output(:,1),ytrain(:,1),'o')
% ACF plot and QQ-plot
subplot(2,2,2), autocorr(error)
subplot(2,2,3), qqplot(error)
subplot(2,2,4), hold on, plot(ytrain(:,1),'. b'), plot(Output(:,1),'-k'), plot(CI_lb,'-r'),  plot(CI_ub,'-r'),
```

A.3.3 MATLAB code for Kriging

```
% Kriging code
% Download the tool box from "http://www2.imm.dtu.dk/projects/dace/"
% AND run the code below:

clc; close all; clear;

%Load the variables xtrain, ytrain, xtest and ytest and assign to gp structure
load('Inp_dataset'),
load('Outp_dataset'),

% 10 fold-forward chaining
IND= [1:500;501:1000;1001:1500; 1501:2000; 2001:2500; 2501:3000; 3001:3500; 3501:4000; 4001:4500; 4501:5000];

for rr=1:1:10
xdata=Inp_dataset(IND(rr,:),[1 2 5 6 7]);
ydata=Outp_dataset(IND(rr,:),7);
coeff = pca(xdata);
xdata=xdata*coeff;
xtrain=xdata;
ytrain=ydata;
n=size(xtrain,2); % Number of input variables

% Normalizing the data base within unity [0,1]
for ii=1:1:n
Max_X_train=max(xtrain(:,ii)); Min_X_train=min(xtrain(:,ii));
xtrain(:,ii)=(xtrain(:,ii)-(ones(size(xtrain,1),1)*Min_X_train))/(Max_X_train-Min_X_train);
end

%% Training
```

```
tic,
corr='corrgauss';
regr='regpoly0';
theta0=0.2;
[dmodel, perf] = dacefit(xtrain, ytrain, regr, corr, theta0);

%% Results
Output = predictor(xtrain, dmodel);
toc,

error=Output-ytrain;
CI_lb=Output-(1.96 * std(Output));
CI_ub=Output+(1.96 * std(Output));
% Calculation of performane metrics
% MAE
MAE(rr)=sum(abs(error))/size(xtrain,1);
% MaxE
MaxE(rr)=max(abs(error));
% std.
Std(rr)=std(error);
end

% MAE
disp('MEA')
mean(MAE)
% MaxE
disp('MaxE')
mean(MaxE)
% std.
disp('std.')
mean(Std)

% plot
figure,
% correlation plot
subplot(2,2,1), plot(Output(:,1),ytrain(:,1),'o')
% ACF plot and QQ-plot
subplot(2,2,2), autocorr(error)
subplot(2,2,3), qqplot(error)
subplot(2,2,4), hold on, plot(ytrain(:,1),'. b'), plot(Output(:,1),'-k'), plot(CI_lb,'-r'),  plot(CI_ub,'-r'),
```

A.3.4 MATLAB code for multi gene genetic programming

```
% Multi Gene Genetic Programming (MGGP) code

% Download the tool box from "https://sites.google.com/site/gptips4matlab/"
% AND run the code below:

clc; close all; clear;

global xtrain ytrain
```

```
%Load the variables xtrain, ytrain, xtest and ytest and assign to gp structure
load('Inp_dataset'),
load('Outp_dataset'),

% 10 fold-forward chaining
IND= [1:500;501:1000;1001:1500; 1501:2000; 2001:2500; 2501:3000; 3001:3500; 3501:4000; 4001:4500; 4501:5000];

for rr=1:1:10
close all;
xdata=Inp_dataset(IND(rr,:),[1 2 5 6 7]);
ydata=Outp_dataset(IND(rr,:),1);
coeff = pca(xdata);
xdata=xdata*coeff;
xtrain=xdata;
ytrain=ydata;
n=size(xtrain,2); % Number of input variables

% Normalizing the data base within unity [0,1]
for ii=1:1:n
Max_X_train=max(xtrain(:,ii)); Min_X_train=min(xtrain(:,ii));
xtrain(:,ii)=(xtrain(:,ii)-(ones(size(xtrain,1),1)*Min_X_train))/(Max_X_train-Min_X_train);
end

%% Training GP
tic,

gp=rungp('my_config');
runtree(gp,'best')
% renderLatex(gp,'best')
gpmodel2mfile(gp,'best','GPfit');
%% Results
Output = GPfit(xtrain);

toc,

error=Output-ytrain;
CI_lb=Output-(1.96 * std(Output));
CI_ub=Output+(1.96 * std(Output));
% Calculation of performane metrics
% MAE
MAE(rr)=sum(abs(error))/size(xtrain,1);
% MaxE
MaxE(rr)=max(abs(error));
% std.
Std(rr)=std(error);
end

% MAE
disp('MEA')
mean(MAE)
% MaxE
disp('MaxE')
mean(MaxE)
% std.
disp('std.')
mean(Std)
```

```
% plot
figure,
% correlation plot
subplot(2,2,1), plot(Output(:,1),ytrain(:,1),'o')
% ACF plot and QQ-plot
subplot(2,2,2), autocorr(error)
subplot(2,2,3), qqplot(error)
subplot(2,2,4), hold on, plot(ytrain(:,1),'. b'), plot(Output(:,1),'-k'), plot(CI_lb,'-r'),  plot(CI_ub,'-r'),
```

A.3.5 MATLAB code for multi-layer perceptron

```
% Multi Layered Perceptron (MLP) code

clc; clear; close all;

%% Data
load('Inp_dataset'),
load('Outp_dataset'),

% 10 fold-forward chaining
IND= [1:500;501:1000;1001:1500; 1501:2000; 2001:2500; 2501:3000; 3001:3500; 3501:4000; 4001:4500; 4501:5000];

for rr=1:1:10
xdata=Inp_dataset(IND(rr,:),[1 2 5 6 7]);
ydata=Outp_dataset(IND(rr,:),7);
coeff = pca(xdata);
xdata=xdata*coeff;
xtrain=xdata;
ytrain=ydata;
n=size(xtrain,2); % Number of input variables
m=size(ytrain,2); % Number of output variables

% Normalizing the data base within unity [0,1]
for ii=1:1:n
Max_X_train=max(xtrain(:,ii)); Min_X_train=min(xtrain(:,ii));
xtrain(:,ii)=(xtrain(:,ii)-(ones(size(xtrain,1),1)*Min_X_train))/(Max_X_train-Min_X_train);
end

%% Training
tic,
net = feedforwardnet([10 5 2]); % 2 hidden layers
net = train(net,xtrain',ytrain');

%% Results
Output = net(xtrain');
Output=Output';
toc,

error=(ytrain-Output);
CI_lb=Output-(1.96 * std(Output));
CI_ub=Output+(1.96 * std(Output));

% Calculation of performane metrics
```

```
% MAE
MAE(rr)=sum(abs(error))/size(xtrain,1);
% MaxE
MaxE(rr)=max(abs(error));
% std.
Std(rr)=std(error);
end

% MAE
disp('MEA')
mean(MAE)
% MaxE
disp('MaxE')
mean(MaxE)
% std.
disp('std.')
mean(Std)

% --- plot
figure,
% correlation plot
subplot(2,2,1), plot(Output(:,1),ytrain(:,1),'o')
% ACF plot and QQ-plot
subplot(2,2,2), autocorr(error)
subplot(2,2,3), qqplot(error)
subplot(2,2,4), hold on, plot(ytrain(:,1),'. b'), plot(Output(:,1),'-k'), plot(CI_lb,'-r'),  plot(CI_ub,'-r'),
```

A.3.6 MATLAB code for mixture of Gaussian experts

```
% Mixture of Gaussian Experts (MoGE) with Expectation maximization code

clc; close all; clear;

%Load the variables xtrain, ytrain, xtest and ytest and assign to gp structure
load('Inp_dataset'),
load('Outp_dataset'),

% 10 fold-forward chaining
IND= [1:500;501:1000;1001:1500; 1501:2000; 2001:2500; 2501:3000; 3001:3500; 3501:4000; 4001:4500; 4501:5000];

for rr=1:1:10
xdata=Inp_dataset(IND(rr,:),[1 2 5 6 7]);
ydata=Outp_dataset(IND(rr,:),8);
coeff = pca(xdata);
xdata=xdata*coeff;
xtrain=xdata;
ytrain=ydata;
n=size(xtrain,2); % Number of input variables

% Normalizing the data base within unity [0,1]
for ii=1:1:n
Max_X_train=max(xtrain(:,ii)); Min_X_train=min(xtrain(:,ii));
xtrain(:,ii)=(xtrain(:,ii)-(ones(size(xtrain,1),1)*Min_X_train))/(Max_X_train-Min_X_train);
```

```
end

%% Training process
tic,
% Assign label to data (two components)
k=2; % Number of components
Lables=ones(size(ydata,1),1);
ind1=find(ytrain>=0);
ind2=find(ytrain<0);
Lables(ind1)=1;
Lables(ind2)=2;

% Gaussian components
Beta=ones(size(xtrain,2),2);
sigma2= ones(k,1);
% Analytically find the optimized values using Score function

% Beta
xtrain=xtrain';
for h=1:1:k
%Generate Indicator matrix for each expert
Tau_h=diag(Lables==h);
Beta(:,h)=(((xtrain*Tau_h*xtrain')+(0.01*eye(length(xtrain(:,1)))))^-1)*((Tau_h*xdata)')*(ytrain);
end

%sigma2
for h=1:1:k
%Generate Indicator matrix for each expert
Tau_h=diag(Lables==h);
err_h=((Tau_h*ytrain)-((Tau_h*xtrain')*Beta(:,h))).^2;
sigma2(h)=(sum(err_h))/(sum(sum(Tau_h)));
err_h=[];
end

xtrain=xtrain';
Beta=Beta';

% Gate functions (multinomial / softmax) is a function of alpha and input
% Fit a nominal or ordinal multinomial regression model
alpha = mnrfit(xtrain,Lables,'Interaction' ,'on');
%Gating Function values for input input x
% Predict values for a nominal or ordinal multinomial regression model
x=xtrain;

% Calculation of  pi(alhpa,x)
Probability =mnrval(alpha ,x,'Interaction' ,'on');
% Draw numbers from Gaussian components
g=zeros(2,1);
Output=ones(size(x,1),1);
for i=1:1:size(x,1)
for h=1:1:k
mu=10*x(i,:)*(Beta(h,:))';
g(h,1) = normrnd(mu,sigma2(h));
end
Output(i,1)=Probability(i,:)*g;
g=[];
end
```

```
toc,
%% Results
error=Output-ytrain;
CI_lb=Output-(1.96 * std(Output));
CI_ub=Output+(1.96 * std(Output));
% Calculation of performane metrics
% MAE
MAE(rr)=sum(abs(error))/size(xtrain,1);
% MaxE
MaxE(rr)=max(abs(error));
% std.
Std(rr)=std(error);
end

% MAE
disp('MEA')
mean(MAE)
% MaxE
disp('MaxE')
mean(MaxE)
% std.
disp('std.')
mean(Std)

% plot
figure,
% correlation plot
subplot(2,2,1), plot(Output(:,1),ytrain(:,1),'o')
% ACF plot and QQ-plot
subplot(2,2,2), autocorr(error)
subplot(2,2,3), qqplot(error)
subplot(2,2,4), hold on, plot(ytrain(:,1),'. b'), plot(Output(:,1),'-k'), plot(CI_lb,'-r'),  plot(CI_ub,'-r'),
```

A.3.7 R code for projection pursuit regression

```
# Projection Pursuit Regression (PPR) code

setwd("C:\\Users\\amozaffa\\Desktop\\Regression Models\\PPR")

# Load input dataset
Inp.dataset<-read.csv("Inp_dataset.csv",header=FALSE)

# Load output dataset
Outp.dataset<-read.csv("Outp_dataset.csv",header=FALSE)
xdata=Inp.dataset[1:500, c(1,2,5,6,7)]
ydata=Outp.dataset[1:500, 1]
coeff= prcomp(xdata, center = FALSE, scale. = FALSE)
xdata=xdata*coeff$rotation
xtrain=xdata
ytrain=ydata

# normalize data
```

```
xx=c()
for ( i in 1: ncol(xtrain) ){
xx=(xtrain[,i]-min(xtrain[,i]))/(max(xtrain[,i])-min(xtrain[,i]))
xtrain[,i]=xx
xx=c()
}

# train PPR
system.time({
nterms = 3
Output.ppr <-ppr(xdata, ydata, nterms = 3, max.terms = nterms, optlevel = 2,
sm.method = c("gcvspline"),
bass = 0, span = 0, df = 5, gcvpen = 1, trace = FALSE)

# Results
Output = predict(Output.ppr)
})

error=Output-ytrain
CI.lb=Output-(1.96 * sd(Output))
CI.ub=Output+(1.96 * sd(Output))

# MAE
MAE=(mean(abs(Output - ydata), na.rm = TRUE))
# MaxE
MaxE=max(abs(Output - ydata))
# std.
Std=sd(error)

# Plot
par(mfrow=c(2,2))

ccc=seq(from =-0.1, to = 0.1, by = 0.001 )
plot(Output,ydata, xlim=c(-0.1,0.1), ylim=c(-0.1,0.1), xlab="PPR fit", ylab="data")
lines(ccc,ccc, type="l", col="red", lty=4)
acf(error)
qqnorm(error, main="qq-plot for residuals")
qqline((error), col="red", lty=4)
plot(ydata, col='blue', type="p", cex = .4)
lines(Output,col="black", lwd=1,type="l")
lines(CI.ub,col="red",lwd=1,type="l")
lines(CI.lb,col="red",lwd=1,type="l")
```

A.3.8 MATLAB code for randomized neural network

```
% Randomized Neural Network (RNN) code

clc; clear; close all;

%% Data
load('Inp_dataset'),
load('Outp_dataset'),
```

```
% 10 fold-forward chaining
IND= [1:500;501:1000;1001:1500; 1501:2000; 2001:2500; 2501:3000; 3001:3500; 3501:4000; 4001:4500; 4501:5000];

for rr=1:1:10
xdata=Inp_dataset(IND(rr,:),[1 2 5 6 7]);
ydata=Outp_dataset(IND(rr,:),1);
coeff = pca(xdata);
xdata=xdata*coeff;
xtrain=xdata;
ytrain=ydata;

% parameters
N=size(xtrain,1);  % Number of Samples
n=size(xtrain,2);  % Number of input variables
m=size(ytrain,2);  % Number of output variables
N_HN=round(0.08*N); % number of hidden neurons;

if N<N_HN
error('number of samples should be greater than hidden nodes')
end

% Normalizing the data base within unity [0,1]
for ii=1:1:n
Max_X_train=max(xtrain(:,ii)); Min_X_train=min(xtrain(:,ii));
xtrain(:,ii)=(xtrain(:,ii)-(ones(N,1)*Min_X_train))/(Max_X_train-Min_X_train);
end

%% Training
tic,

% Randomly generate the weights of input-hidden connective synaps
W_IH=rand(n,N_HN); % [NumInputs NumHiddenNodes]
bias_IH=rand(1,N_HN);
% Generate the "hidden layer output matrix (H)"
% Activation Function logsig ==> logsig(n) = 1 / (1 + exp(-n))
H=logsig((xtrain*W_IH)+(ones(N,1)*bias_IH)); % [N N_HN]
% Finding the optimum weights of hiddesn-out put using pseudoInverse technique
LAMBDA=0.01;
uu=eye(size(H'*H,1));
H_Pseudoinverse=(((H'*H)+(LAMBDA*uu))^-1)*H';
% Determine the optimum values of hidden-output weights
% depends to number of outputs
W_HO=H_Pseudoinverse*ytrain;

%% Results
Output=H*W_HO;

toc,
error=Output-ytrain;
CI_lb=Output-(1.96 * std(Output));
CI_ub=Output+(1.96 * std(Output));
% Calculation of performane metrics
% MAE
MAE(rr)=sum(abs(error))/size(xtrain,1);
% MaxE
MaxE(rr)=max(abs(error));
% std.
```

```
Std(rr)=std(error);
end

% MAE
disp('MEA')
mean(MAE)
% MaxE
disp('MaxE')
mean(MaxE)
% std.
disp('std.')
mean(Std)

% plot
figure,
% correlation plot
subplot(2,2,1), plot(Output(:,1),ytrain(:,1),'o')
% ACF plot and QQ-plot
subplot(2,2,2), autocorr(error)
subplot(2,2,3), qqplot(error)
subplot(2,2,4), hold on, plot(ytrain(:,1),'. b'), plot(Output(:,1),'-k'), plot(CI_lb,'-r'),  plot(CI_ub,'-r'),
```

A.3.9 MATLAB code for support vector regression

```
% Support Vector Regression (SVR) code

clear; clc; close all;

load('Inp_dataset'),
load('Outp_dataset'),

% 10 fold-forward chaining
IND= [1:500;501:1000;1001:1500; 1501:2000; 2001:2500; 2501:3000; 3001:3500; 3501:4000; 4001:4500; 4501:5000];

for rr=1:1:10
xdata=Inp_dataset(IND(rr,:), [1 2 5 6 7]);
ydata=Outp_dataset(IND(rr,:),1);
xtrain=xdata;
ytrain=ydata;

% Normalizing the data base within unity [0,1]
for ii=1:1:size(xtrain,2)
Max_X_train=max(xtrain(:,ii)); Min_X_train=min(xtrain(:,ii));
xtrain(:,ii)=(xtrain(:,ii)-(ones(size(xtrain,1),1)*Min_X_train))/(Max_X_train-Min_X_train);
end

tic,
%% Training
% model parameters
c=4;
epsilon=0.000000025;
kernel='gaussian';
varargin=0.0005;
```

```
lambda = varargin;
kernel_function = @(x,y) exp(-lambda*norm(x.feature-y.feature,2)^2);

ntrain = size(xtrain,1);
alpha0 = zeros(ntrain,1);

for i=1:ntrain
for j=1:ntrain
xi(i,j).feature = xtrain(i,:);
xj(i,j).feature = xtrain(j,:);
end
end
% Set up the Gram matrix for the training data
M = arrayfun(kernel_function,xi,xj);
M = M + 1/c*eye(ntrain);
% Train the SVR by optimising the dual function ie. find alpha_i's
options = optimoptions('quadprog','Algorithm','interior-point-convex');
H = 0.5*[M zeros(ntrain,3*ntrain); zeros(3*ntrain,4*ntrain)];
lb = [-c*ones(ntrain,1); zeros(ntrain,1); zeros(2*ntrain,1)];
ub = [ c*ones(ntrain,1); 2*c*ones(ntrain,1); c*ones(2*ntrain,1)];
f = [ -ydata; epsilon*ones(ntrain,1);zeros(ntrain,1);zeros(ntrain,1)];
z = quadprog(H,f,[],[],[],[],lb,ub,[],options);
alpha = z(1:ntrain);

% Calculate b
for m=1:ntrain
bmat(m) = ydata(m);
for n = 1:ntrain
bmat(m) = bmat(m) - alpha(n)*M(m,n);
end
bmat(m) = bmat(m) - epsilon - alpha(m)/c;
end
b = mean(bmat);

%% ---- Results
xtest=Inp_dataset(IND(rr,:),[1 2 5 6 7]);
ytest = Outp_dataset(IND(rr,:),1);
y=ytest;
Output = svr_eval(xtest,xtrain,alpha,b,kernel_function);

toc,
error=Output-y;
CI_lb=Output-(1.96 * std(Output));
CI_ub=Output+(1.96 * std(Output));
% Calculation of performane metrics
% MAE
MAE(rr)=sum(abs(error))/size(xtrain,1);
% MaxE
MaxE(rr)=max(abs(error));
% std.
Std(rr)=std(error);
end

% MAE
disp('MEA')
mean(MAE)
```

```
% MaxE
disp('MaxE')
mean(MaxE)
% std.
disp('std.')
mean(Std)

% plot
figure,
% correlation plot
subplot(2,2,1), plot(Output,y,'o')
% ACF plot
subplot(2,2,2), autocorr(y-Output)
% QQ-plot
subplot(2,2,3), qqplot(y-Output)
% Estimation plot
subplot(2,2,4), hold on, plot(y(:,1),'. b'), plot(Output(:,1),'-k'), plot(CI_lb,'-r'),  plot(CI_ub,'-r'),
```

A.4 Codes implemented in Chapter 8

The simulation conducted in Chapter 8 includes MATLAB codes for implementation of learning-based model predictive controller (LBMPC), golden sectioning search (GSS) optimizer and simulated annealing (SA) optmizer, as well as several conventional controllers. Here, the codes for the implementation of the most successfull controller (the proposed LBMPC) are provided.

A.4.1 MATLAB code for learning-based model predictive controller

```
% Learning-Based Model Predictive Controller (LBMPC)

clc; clear; close all;

global A B

%% Nonliner model based on equations of motion
%----- System parameters
V=20; % Vehicle Forward Velocity
% Vehicle‘s Body parameters
Mb=580; Jb=1100; m_u1=40; m_u2=35.5; L1=1.5; L2=1;
Delta=(1/Mb)+((L1^2)/Jb);
Etta=(1/Mb)-((L1*L2)/Jb);
Zetta=(1/Mb)+((L2^2)/Jb);
% Dampers and Springs coefficients
K_l_s1=14000;    K_l_s2=K_l_s1;
```

```
K_nl_s1=2.35e4;  K_nl_s2=K_nl_s1;
K_t1=190e3;      K_t2=K_t1;
C_l_s1=800;      C_l_s2=700;
C_nl_s1=400;     C_nl_s2=C_nl_s1;
C_sym_s1=400;    C_sym_s2=C_sym_s1;
C_t1=80;         C_t2=70;

%% Operating time
t0=0.0;       % Control start time
dt=0.001;     % Sampling time
ng=5000;      % Number of working points
ts=ng*dt;     % Control termination time

%% Initial state and initial actuation signal
%----- Initial state values
% Vehicle body states
xb1=0; xb2=0;
xc=0;  teta_c=0;
% Suspension system and sprung mass system
xu1=0;  xu2=0;

Xsystem(1,1)=xb1;        % xb_1
Xsystem(2,1)=0;          % xb_dot_1
Xsystem(3,1)=xb2;        % xb_2
Xsystem(4,1)=0;          % xb_dot_2
Xsystem(5,1)=xu1;        % xu_1
Xsystem(6,1)=0;          % xu_dot_1
Xsystem(7,1)=xu2;        % xu_2
Xsystem(8,1)=0;          % xu_dot_2

%----- Initial actuation signals
U=zeros(2,ng+1);

%% Control input Stabilization matrixes (U_stable)
Saii=[5.08 1.326e-02
5.08 1.326e-02
1.326e-02 5.04
1.326e-02 5.04
5.24 9.57e-04
5.24 9.57e-04
9.57e-04 5.24
9.57e-04 5.24];

Phaii=[-10e06 2.43e-05
2.41e-05 -10e06];

% Domain of variation of Theta
Theta_min= [-4*10e-6; -4*10e-6];
Theta_max= [ 4*10e-6;  4*10e-6];
Theta=zeros(2,ng+1);

U_stable=zeros(2,ng+1);

%% Road roughness Disturbances

% Four simulated cases and the true road model
Case=1;
```

```
% in design, we can easily adjust the coefficients of D to make sure the signal is feasible
if Case==1

D=[0 0 0 0
0 0 0 0
0 0 0 0
0 0 0 0
0 0 0 0
0.3*K_t1/m_u1 0.3*C_t1/m_u1 0 0
0 0 0 0
0 0 0.3*K_t2/m_u2 0.3*C_t2/m_u2];

elseif Case==2 % road disturbance + body states noise

D=[0 0 0 0 1 0 0 0
0 0 0 0 0 1 0 0
0 0 0 0 0 0 1 0
0 0 0 0 0 0 0 1
0 0 0 0 0 0 0 0
0.3*K_t1/m_u1 0.3*C_t1/m_u1 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0.3*K_t2/m_u2 0.3*C_t2/m_u2 0 0 0 0];

elseif Case==3 % road disturbance + unsprung states noise

D=[0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 1 0 0 0
0.3*K_t1/m_u1 0.3*C_t1/m_u1 0 0 0 1 0 0
0 0 0 0 0 0 1 0
0 0 0.3*K_t2/m_u2 0.3*C_t2/m_u2 0 0 0 1];

elseif Case==4 % % road disturbance + all states noise

D=[0 0 0 0 1 0 0 0 0 0 0 0
0 0 0 0 0 1 0 0 0 0 0 0
0 0 0 0 0 0 1 0 0 0 0 0
0 0 0 0 0 0 0 1 0 0 0 0
0 0 0 0 0 0 0 0 1 0 0 0
0.3*K_t1/m_u1 0.3*C_t1/m_u1 0 0 0 0 0 0 1 0 0
0 0 0 0 0 0 0 0 0 0 1 0
0 0 0.3*K_t2/m_u2 0.3*C_t2/m_u2 0 0 0 0 0 0 0 1];
end

% Road roughness vecor
road=1;
if road==1 % shaped function
% r = [r_1 r_dot_1 r_2 r_dot_2]

r_1=zeros(1,ng+1);
r_2=zeros(1,ng+1);

LambdA=5; % Input disturbance wavelength
Amp=0.11; % Bump amplitute
td=1+((L1+L2)/V); % delay time
```

```
ii=0;
for tt=t0+dt:dt:ts
ii=ii+1;

% Front tyre disturbance
if tt>=1 & tt<=1+(LambdA/V)
r_1(ii)=(Amp/2)*(1-cos(2*pi*V*tt/LambdA));
else
r_1(ii)=0;
end

% Rear tyre disturbance
if tt>=td & tt<=td+(LambdA/V)
r_2(ii)=(Amp/2)*(1-cos(2*pi*V*(tt+((L1+L2)/V))/LambdA));
else
r_2(ii)=0;
end

end

for i=1:ng-1
r_dot_1(i)=(r_1(i+2)-r_1(i))/(2*dt);
r_dot_2(i)=(r_2(i+2)-r_2(i))/(2*dt);
end

r(:,1)=r_1(1:end);
r(:,2)=[r_1(1) r_dot_1 r_1(end)];
r(:,3)=r_2(1:end)';
r(:,4)=[r_2(1) r_dot_2 r_2(end)];
r=r';
elseif road==2 % waterloo data
load('roadroughness')
r=roadroughness(70:79);
xxx = 0:5/9:5;
t0=0.0;
dt=0.001;
ng=5000;      % Number of working points
ts=ng*dt;      % Active control duration
xxxx = t0:dt:ts;
r_1 = spline(xxx,r,xxxx);
r=[];
td=((L1+L2)/20); % delay time

r_2(1:round(td/dt))=0;
r_2(round(td/dt)+1:size(r_1,2))=r_1(1:end-(round(td/dt)));

for i=1:ng-1
r_dot_1(i)=(r_1(i+2)-r_1(i))/(2*dt);
r_dot_2(i)=(r_2(i+2)-r_2(i))/(2*dt);
end

r(:,1)=r_1(1:end);
r(:,2)=[r_1(1) r_dot_1 r_1(end)];
r(:,3)=r_2(1:end)';
r(:,4)=[r_2(1) r_dot_2 r_2(end)];
r=r';
end
```

```
%% Quantiifed uncertainty polytope W
W_lb = [-0.0019; -0.0184; -0.0019; -0.0144; -0.0019; -0.0478; -0.0020; -0.0649];
W_ub = [ 0.0019;  0.0177;  0.0020;  0.0161;  0.0020;  0.0484;  0.0020;  0.0546];

% Form a set with 1000 points taken from the intervals

W=[W_lb(1):(W_ub(1)-W_lb(1))/999:W_ub(1)
W_lb(2):(W_ub(2)-W_lb(2))/999:W_ub(2)
W_lb(3):(W_ub(3)-W_lb(3))/999:W_ub(3)
W_lb(4):(W_ub(4)-W_lb(4))/999:W_ub(4)
W_lb(5):(W_ub(5)-W_lb(5))/999:W_ub(5)
W_lb(6):(W_ub(6)-W_lb(6))/999:W_ub(6)
W_lb(7):(W_ub(7)-W_lb(7))/999:W_ub(7)
W_lb(8):(W_ub(8)-W_lb(8))/999:W_ub(8)];

%% Kriging Oracle

load('KrigingModel1'),
load('KrigingModel2'),
load('KrigingModel3'),
load('KrigingModel4'),
load('KrigingModel5'),
load('KrigingModel6'),
load('KrigingModel7'),
load('KrigingModel8'),

% inp = [x_1 x_2 x_5 x_6 u_1]
min_inp = [-0.0559; -0.0268; 0.0000; -0.0024; -3.1857e-04];
max_inp = [ 0.2937;  0.1565; 0.0581;  0.0061;  1.3724e-05];
inp_size=5; % input data can be more than 1 (up to horizon length N)

load('PCA_coeff'),
% inp=inp*PCA_coeff;

% Normalizing inputs within unity [0,1]
% for ii=1:1:inp_size
%     inp(:,ii)=(inp(:,ii)-(ones(size(inp,1),1)*max_inp(ii)))/(max_inp(ii)-min_inp(ii));
% end

% Output = predictor(xtrain, KrigingModel1);

%% Control Loop
% Horizon length
N = 10;

% Objective function parameters
R=blkdiag(0.0000001,0.0000001);
Q=blkdiag(10,10,10,10,10,10,10,10,10,10);
T=blkdiag(10,10,10,10,10,10,10,10,10,10);

% Bounds of states and control inputs
X_lb = [-0.05; -0.5; -0.05; -0.2; -0.1; -1; -0.1; -1];
X_ub = [ 0.05;  0.5;  0.05;  0.2;  0.1;  1;  0.1;  1];
U_lb= [-5000; -5000];
U_ub= [ 5000;  5000];
```

```
X_set=[X_lb(1):(X_ub(1)-X_lb(1))/999:X_ub(1)
X_lb(2):(X_ub(2)-X_lb(2))/999:X_ub(2)
X_lb(3):(X_ub(3)-X_lb(3))/999:X_ub(3)
X_lb(4):(X_ub(4)-X_lb(4))/999:X_ub(4)
X_lb(5):(X_ub(5)-X_lb(5))/999:X_ub(5)
X_lb(6):(X_ub(6)-X_lb(6))/999:X_ub(6)
X_lb(7):(X_ub(7)-X_lb(7))/999:X_ub(7)
X_lb(8):(X_ub(8)-X_lb(8))/999:X_ub(8)];

U_set=[U_lb(1):(U_ub(1)-U_lb(1))/999:U_ub(1)
U_lb(2):(U_ub(2)-U_lb(2))/999:U_ub(2)];

for n=1:N:ng
xs=Xsystem(:,n);
%%% ------ PIECE-WISE LINEARIZATION ------ %%%
z1=xs(5)-xs(1);
z2=xs(7)-xs(3);
z_dot_1=xs(6)-xs(2);
z_dot_2=xs(8)-xs(4);
eps=2.2204e-16;
K_eq_s1=3*K_nl_s1*(z1^2);
K_eq_s2=3*K_nl_s1*(z2^2);
C_eq_s1=(-C_sym_s1*(z_dot_1/abs(z_dot_1+eps)))+(C_nl_s1*(0.5*(z_dot_1/abs(z_dot_1+eps))...
*((abs(z_dot_1+eps))^-0.5)*sign(z_dot_1)));
C_eq_s2=(-C_sym_s2*(z_dot_2/abs(z_dot_2+eps)))+(C_nl_s2*(0.5*(z_dot_2/abs(z_dot_2+eps))...
*((abs(z_dot_2+eps))^-0.5)*sign(z_dot_2)));
Kstar_s1=K_l_s1+K_eq_s1;
Kstar_s2=K_l_s2+K_eq_s2;
Cstar_s1=1*(C_l_s1+C_eq_s1);
Cstar_s2=1*(C_l_s2+C_eq_s2);

% f1 and f2 for calculation of body acceleration
f1_System(n)=(Kstar_s1*z1)+(Cstar_s1*z_dot_1);
f2_System(n)=(Kstar_s2*z2)+(Cstar_s2*z_dot_2);

%----- State Space Parameters Updating
A=[0 1 0 0 0 0 0 0
-(Delta*Kstar_s1) -(Delta*Cstar_s1) -(Etta*Kstar_s2) -(Etta*Cstar_s2) (Delta*Kstar_s1) (Delta*Cstar_s1)...
 (Etta*Kstar_s2) (Etta*Cstar_s2)
0 0 0 1 0 0 0 0
-(Etta*Kstar_s1) -(Etta*Cstar_s1) -(Zetta*Kstar_s2) -(Zetta*Cstar_s2) (Etta*Kstar_s1) (Etta*Cstar_s1)...
 (Zetta*Kstar_s2) (Zetta*Cstar_s2)
0 0 0 0 0 1 0 0
(Kstar_s1/m_u1) (Cstar_s1/m_u1) 0 0 (-K_t1-Kstar_s1)/m_u1 (-C_t1-Cstar_s1)/m_u1 0 0
0 0 0 0 0 0 0 1
0 0 (Kstar_s2/m_u2) (Cstar_s2/m_u2) 0 0 (-K_t2-Kstar_s2)/m_u2 (-C_t2-Cstar_s2)/m_u2];

B=[0 0
-dt*Delta -dt*Etta
0 0
-dt*Etta -dt*Zetta
0 0
(1/m_u1) 0
0 0
0 (1/m_u2)];

%%% ------ OPTIMIZATION LOOP ------ %%%
```

```
Optimizer=2; % 1: SA; 2: GSS; 3: Newton Method

% Calculate stabilization gain matrix K
pole=[-2 -0.5 -1 -5 -3 -1.5 -4 -6];
K_gain = place(A,B,pole);

% Calculate Lyapunov matrix P using Discrete time Lyapunov Equation
aa = (A + (B*K_gain))';
aa=aa*0.00001;
bb = (Q + ((K_gain')*R*K_gain));
bbb = chol(bb); % Cholesky Decomposition: bb = bbb' * bbb
P = dlyapchol(aa,bbb');

% Calculate the invariant sets
H(:,:,1) = [min(W'); max(W')];
for ii=2:1:N
C = ((A + (B*K_gain))^(ii-1))*W; C=C';
cc = H(:,:,ii-1);
cc_lb = cc(1,:);
cc_ub = cc(2,:);
CC=[cc_lb(1):(cc_ub(1)-cc_lb(1))/999:cc_ub(1)
cc_lb(2):(cc_ub(2)-cc_lb(2))/999:cc_ub(2)
cc_lb(3):(cc_ub(3)-cc_lb(3))/999:cc_ub(3)
cc_lb(4):(cc_ub(4)-cc_lb(4))/999:cc_ub(4)
cc_lb(5):(cc_ub(5)-cc_lb(5))/999:cc_ub(5)
cc_lb(6):(cc_ub(6)-cc_lb(6))/999:cc_ub(6)
cc_lb(7):(cc_ub(7)-cc_lb(7))/999:cc_ub(7)
cc_lb(8):(cc_ub(8)-cc_lb(8))/999:cc_ub(8)];
CC=CC';
[Msum,MsumRange]=MinkSum(C,CC);
H(:,:,ii) =MsumRange;
cc=[]; C=[]; CC=[];
end

% Calculate the bounds for X and U
X_set_horizon=[]; U_set_horizon=[];
for ii=1:1:N
cc = H(:,:,ii);
cc_lb = cc(1,:);
cc_ub = cc(2,:);
CC=[cc_lb(1):(cc_ub(1)-cc_lb(1))/999:cc_ub(1)
cc_lb(2):(cc_ub(2)-cc_lb(2))/999:cc_ub(2)
cc_lb(3):(cc_ub(3)-cc_lb(3))/999:cc_ub(3)
cc_lb(4):(cc_ub(4)-cc_lb(4))/999:cc_ub(4)
cc_lb(5):(cc_ub(5)-cc_lb(5))/999:cc_ub(5)
cc_lb(6):(cc_ub(6)-cc_lb(6))/999:cc_ub(6)
cc_lb(7):(cc_ub(7)-cc_lb(7))/999:cc_ub(7)
cc_lb(8):(cc_ub(8)-cc_lb(8))/999:cc_ub(8)];
CC=CC';
PdiffRange=PontDiff(X_set',CC);
X_set_horizon(:,:,ii)=PdiffRange;
PdiffRange=[];
KCC=K_gain*CC';
PdiffRange=PontDiff(U_set',KCC);
U_set_horizon(:,:,ii)=PdiffRange;
cc=[]; C=[]; CC=[];
```

```
end

% Calculate the road disturbance "R*r"

if Case==1 % just road disturbance
ds(:,n:n+N-1)=r(:,n:n+N-1);
elseif Case==2 % road disturbance + body states noise
for kkk=n:1:n+N-1
ds(:,kkk)=[r(:,kkk); 0.005*sin(2*pi*kkk/ng); 0.01*(pi/ng)*cos(2*pi*kkk/ng); 0.002*sin(0.1*dt*kkk);...
 0.0002*cos(0.1*dt*kkk)];
end
elseif Case==3 % road disturbance + unsprung states noise
for kkk=n:1:n+N-1
ds(:,kkk)=[r(:,kkk); 0.005*sin(2*pi*kkk/ng); 0.01*(pi/ng)*cos(2*pi*kkk/ng); 0.002*sin(0.1*dt*kkk);...
 0.0002*cos(0.1*dt*kkk)];
end
elseif Case==4 % % road disturbance + all states noise
for kkk=n:1:n+N-1
ds(:,kkk)=[r(:,kkk); 0.0001*sin(2*pi*kkk/ng); 0.0002*(pi/ng)*cos(2*pi*kkk/ng); 0.0001*sin(0.1*dt*kkk);...
 0.00001*cos(0.1*dt*kkk);0.0001*sin(2*pi*kkk/ng);...
  0.0002*(pi/ng)*cos(2*pi*kkk/ng); 0.0001*sin(0.1*dt*kkk); 0.00001*cos(0.1*dt*kkk)];
end
end

Dr= D*ds(:,n:n+N-1); % n*N matrix

% Calculate the oracle prediction
xXx=xs;
inp(1:8,1)= xXx;
inp(9:10,1)= [1000;1000];
for ij=1:1:N-1
inp(1:8,ij+1)=( dt*( (A*xXx) + (10*B*ones(2,1)) ) ) + xXx;
inp(9:10,ij+1)= [1000;1000];
xXx=inp(1:8,ij+1);
end
inp([3 4 7 8 10],:)=[];
inp=inp';
inp=inp*PCA_coeff;

for ij=1:1:inp_size
ind=find(inp(:,ij)<min_inp(ij)); inp(ind,ij)=rand(size(ind,1),1)*min_inp(ij);
ind=find(inp(:,ij)>max_inp(ij)); inp(ind,ij)=rand(size(ind,1),1)*max_inp(ij);
end

% Normalizing inputs within unity [0,1]
for ii=1:1:inp_size
inp(:,ii)=(inp(:,ii)-(ones(size(inp,1),1)*min_inp(ii)))/(max_inp(ii)-min_inp(ii));
end

Oracle(:,1) = predictor(inp, KrigingModel1);
Oracle(:,2) = predictor(inp, KrigingModel2);
Oracle(:,3) = predictor(inp, KrigingModel3);
Oracle(:,4) = predictor(inp, KrigingModel4);
Oracle(:,5) = predictor(inp, KrigingModel5);
Oracle(:,6) = predictor(inp, KrigingModel6);
Oracle(:,7) = predictor(inp, KrigingModel7);
Oracle(:,8) = predictor(inp, KrigingModel8);
```

```
Oracle=(10^-4)*Oracle;
Oracle=Oracle';

% Calculate the optimal U_stable

if Optimizer==1 % Simulated Annealing
ttheta = SA(Theta_min,Theta_max,Dr,A,B,K_gain,Phaii,Saii,xs,Oracle,R,Q,T,P);
elseif Optimizer==2 % Golden Sectioning Search
ttheta = GSS(Theta_min,Theta_max,Dr,A,B,K_gain,Phaii,Saii,xs,Oracle,R,Q,T,P);
elseif Optimizer==3 % Newton's Method

end

Oracle=[];
Theta(:,n:n+N-1)=ttheta;
U_stable(:,n:n+N-1)=(Phaii-(-K_gain*Saii))*Theta(:,n:n+N-1);
uu=(-K_gain*xs)+U_stable(:,n);
% Check for possible bound violation
ind1=find((uu(1,:)-U_set_horizon(2,1,:))>0);  uu(1,ind1)=U_set_horizon(2,1,ind1);  ind1=[];
ind2=find((uu(1,:)+U_set_horizon(1,1,:))<0);  uu(1,ind2)=U_set_horizon(1,1,ind1);  ind2=[];
ind1=find((uu(2,:)-U_set_horizon(2,1,:))>0);  uu(2,ind1)=U_set_horizon(2,1,ind1);  ind1=[];
ind2=find((uu(2,:)+U_set_horizon(1,1,:))<0);  uu(2,ind2)=U_set_horizon(1,1,ind1);  ind2=[];
U(:,n)=uu;
%%% ------ STATE UPDATING ------ %%%
for ij=n:1:n+N-1
Xsystem(:,ij+1)=( dt*( (A*Xsystem(:,ij)) + (B*U(:,ij)) + (D*ds(:,ij)) ) ) + Xsystem(:,ij);
% Calculate the actuation signal
uu=(-K_gain*Xsystem(:,ij+1))+U_stable(:,ij+1);
% Check for possible bound violation
ind1=find((uu(1,:)-U_set_horizon(2,1,:))>0);  uu(1,ind1)=U_set_horizon(2,1,ind1);  ind1=[];
ind2=find((uu(1,:)+U_set_horizon(1,1,:))<0);  uu(1,ind2)=U_set_horizon(1,1,ind1);  ind2=[];
ind1=find((uu(2,:)-U_set_horizon(2,1,:))>0);  uu(2,ind1)=U_set_horizon(2,1,ind1);  ind1=[];
ind2=find((uu(2,:)+U_set_horizon(1,1,:))<0);  uu(2,ind2)=U_set_horizon(1,1,ind1);  ind2=[];
U(:,ij+1)=uu;
end

n
end

time=t0:dt:ts;
figure,
subplot(4,2,1), plot(time,Xsystem(1,:),'b'),
subplot(4,2,2), plot(time,Xsystem(2,:),'b'),
subplot(4,2,3), plot(time,Xsystem(3,:),'b'),
subplot(4,2,4), plot(time,Xsystem(4,:),'b'),
subplot(4,2,5), plot(time,Xsystem(5,:),'b'),
subplot(4,2,6), plot(time,Xsystem(6,:),'b'),
subplot(4,2,7), plot(time,Xsystem(7,:),'b'),
subplot(4,2,8), plot(time,Xsystem(8,:),'b'),
%
figure,
subplot(2,2,1), plot(time,U_stable(1,:))
subplot(2,2,2), plot(time,U_stable(2,:))
subplot(2,2,3), plot(time,U(1,:))
subplot(2,2,4), plot(time,U(2,:))
```

A.4.2 MATLAB code for golden sectioning search

```
function Theta = GSS(Theta_min,Theta_max,Dr,A,B,K_gain,Phaii,Saii,xs,Oracle,R,Q,T,P)
N=size(Dr,2); % Horizon Length
dt=0.001;
D=size(Theta_min,1)*N;         % Dimesnion of the problem

% Intial solution
Xsol = zeros(1,D);
Sol1 = zeros(1,D);
Sol2 = zeros(1,D);

% GSS Controlling parameters
iter=1;
phi=1.6181;

for jj=1:1:D
a=Theta_min(1);
b=Theta_max(1);

while iter<=50
Sol1(jj) = b-((b-a)/phi);
Sol2(jj) = a+((b-a)/phi);

% Evaluate the fitness of the two individuals
% Sol1
Theta(1,1:N)=Sol1(1:N);
Theta(2,1:N)=Sol1(N+1:D);
U_stable=(Phaii-(-K_gain*Saii))*Theta;
U(:,1)=(-K_gain*xs)+U_stable(:,1);
Xsystem(:,1)=xs;
for ij=1:1:N-1
Xsystem(:,ij+1)=( dt*( (A*Xsystem(:,ij)) + (B*U(:,ij)) + (Dr(:,ij)) + Oracle(:,ij) ) ) + Xsystem(:,ij);
U(:,ij+1)=(-K_gain*Xsystem(:,ij+1))+U_stable(:,ij+1);
end

SumX=0;
SumU=0;
for ij=2:1:N-1
SumX=SumX+((Xsystem(:,ij)-(Saii*Theta(:,ij)))'*Q*(Xsystem(:,ij)-(Saii*Theta(:,ij))));
SumU=SumU+((U(:,ij)-(Phaii*Theta(:,ij)))'*R*(U(:,ij)-(Phaii*Theta(:,ij))));
end

h_Sol1= ((Xsystem(:,N)-(Saii*Theta(:,N)))'*P*(Xsystem(:,N)-(Saii*Theta(:,N))))...
+((-(Saii*Theta(:,N)))'*T*(-(Saii*Theta(:,N))))+SumX+SumU;

Xsystem=[]; U=[]; Theta=[]; U_stable=[];
% Sol2
Theta(1,1:N)=Sol2(1:N);
Theta(2,1:N)=Sol2(N+1:D);
U_stable=(Phaii-(-K_gain*Saii))*Theta;
U(:,1)=(-K_gain*xs)+U_stable(:,1);
Xsystem(:,1)=xs;
for ij=1:1:N-1
Xsystem(:,ij+1)=( dt*( (A*Xsystem(:,ij)) + (B*U(:,ij)) + (Dr(:,ij)) + Oracle(:,ij) ) ) + Xsystem(:,ij);
U(:,ij+1)=(-K_gain*Xsystem(:,ij+1))+U_stable(:,ij+1);
```

```
end

SumX=0;
SumU=0;
for ij=2:1:N-1
SumX=SumX+((Xsystem(:,ij)-(Saii*Theta(:,ij)))'*Q*(Xsystem(:,ij)-(Saii*Theta(:,ij))));
SumU=SumU+((U(:,ij)-(Phaii*Theta(:,ij)))'*R*(U(:,ij)-(Phaii*Theta(:,ij))));
end
h_Sol2= ((Xsystem(:,N)-(Saii*Theta(:,N)))'*P*(Xsystem(:,N)-(Saii*Theta(:,N))))...
+((-(Saii*Theta(:,N)))'*T*(-(Saii*Theta(:,N))))+SumX+SumU;

%--------------------
if h_Sol1<h_Sol2
b=Sol1(jj);
Xsol(jj)=Sol1(jj);
else
a=Sol2(jj);
Xsol(jj)=Sol2(jj);
end
iter=iter+1;
end
end

end
```

A.4.3 MATLAB code for simulated annealing

```
function Theta = SA(Theta_min,Theta_max,Dr,A,B,K_gain,Phaii,Saii,xs,Oracle,R,Q,T,P)
N=size(Dr,2); % Horizon Length
dt=0.001;
iter=0;
D=size(Theta_min,1)*N;         % Dimesnion of the problem

% SA Controlling parameters
tempi=1;       % Initial Temperature
temps=0.001;     % Final Temperature
cf=0.9996;    % Cooling Factor
temp=tempi;         % Current Temperature
SigmAaa=0.1*(Theta_max(1)-Theta_min(1)); % A parameter in the Gaussian proposal

% Intial solution
Xsol=Theta_min(1)+(rand(1,D)*(Theta_max(1)-Theta_min(1)));

% Point estimate of the target distribution
Theta(1,1:N)=Xsol(1:N);  Theta(2,1:N)=Xsol(N+1:D);
U_stable=(Phaii-(-K_gain*Saii))*Theta;   U(:,1)=(-K_gain*xs)+U_stable(:,1);
Xsystem(:,1)=xs;
for ij=1:1:N-1
Xsystem(:,ij+1)=( dt*( (A*Xsystem(:,ij)) + (B*U(:,ij)) + (Dr(:,ij)) + Oracle(:,ij) ) ) + Xsystem(:,ij);
U(:,ij+1)=(-K_gain*Xsystem(:,ij+1))+U_stable(:,ij+1);
end

SumX=0;  SumU=0;
```

```
for ij=2:1:N-1
SumX=SumX+((Xsystem(:,ij)-(Saii*Theta(:,ij)))'*Q*(Xsystem(:,ij)-(Saii*Theta(:,ij))));
SumU=SumU+((U(:,ij)-(Phaii*Theta(:,ij)))'*R*(U(:,ij)-(Phaii*Theta(:,ij))));
end

h_X= ((Xsystem(:,N)-(Saii*Theta(:,N)))'*P*(Xsystem(:,N)-(Saii*Theta(:,N))))...
+((-(Saii*Theta(:,N)))'*T*(-(Saii*Theta(:,N))))+SumX+SumU;

Pai_X=exp(-h_X/temp);
Fmin=h_X;

while temp > temps
iter = iter+1;  temp=cf*temp;

%-- Proposal distribution "P(X,Y)" (Gaussian pdf)
Ysol = normrnd(Xsol,SigmAaa);
ind=find(Ysol>Theta_max(1)); Ysol(ind)=Theta_max(1); ind=find(Ysol<Theta_min(1)); Ysol(ind)=Theta_min(1);

% Point estimate of the target distribution
Theta(1,1:N)=Ysol(1:N);   Theta(2,1:N)=Ysol(N+1:D);
U_stable=(Phaii-(-K_gain*Saii))*Theta;   U(:,1)=(-K_gain*xs)+U_stable(:,1);
Xsystem(:,1)=xs;
for ij=1:1:N-1
Xsystem(:,ij+1)=( dt*( (A*Xsystem(:,ij)) + (B*U(:,ij)) + (Dr(:,ij)) + Oracle(:,ij) ) ) + Xsystem(:,ij);
U(:,ij+1)=(-K_gain*Xsystem(:,ij+1))+U_stable(:,ij+1);
end

SumX=0;  SumU=0;
for ij=2:1:N-1
SumX=SumX+((Xsystem(:,ij)-(Saii*Theta(:,ij)))'*Q*(Xsystem(:,ij)-(Saii*Theta(:,ij))));
SumU=SumU+((U(:,ij)-(Phaii*Theta(:,ij)))'*R*(U(:,ij)-(Phaii*Theta(:,ij))));
end

h_Y= ((Xsystem(:,N)-(Saii*Theta(:,N)))'*P*(Xsystem(:,N)-(Saii*Theta(:,N))))...
+((-(Saii*Theta(:,N)))'*T*(-(Saii*Theta(:,N))))+SumX+SumU;

Pai_Y=exp(-h_Y/temp);

%-- Acceptance-Rejection determination
% 1: Draw "u" from Uniform Distribution [0,1]
u=rand;
% 2: Verify to accept or not
alphA=min(1,(Pai_Y/Pai_X));
if u<=alphA % Accept the new state
Xsol=Ysol;
h_X=h_Y;
Pai_X=Pai_Y;
else % Reject
end
% Archive the best solution found so-far
Fmin=min(Fmin, h_X);
Gbest(iter)=Fmin;
% Variation of objective value through the movement of the chain
ObjVal(iter)=h_X;
end

end
```