Model Predictive Controller Weight Tuning and Real-Time Learning-Based Weight Selection

by

Ali Shahidi

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Mechanical and Mechatronics Engineering

Waterloo, Ontario, Canada, 2023

Examining Committee Membership

The following served on the Examining Committee for this thesis. The decision of the Examining Committee is by majority vote.

External Examiner:	Mehdi Ahmadian Title: J. Bernad Jones Chair Department: Mechanical Engineering University: Virginia Tech
Supervisor(s):	Amir Khajepour Title: Professor Department: Mechanical and Mechatronics Engineering
Internal Member:	Baris Fidan Title: Professor Department: Mechanical and Mechatronics Engineering
Internal Member:	Hyock Ju Kwon Title: Associate Professor Department: Mechanical and Mechatronics Engineering
Internal-External Member:	Nasser Lashgarian Azad Title: Associate Professor Department: Systems Design Engineering

Author's Declaration

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

A variety of control systems with specific goals are designed and utilized in every vehicle system. Optimal performance of each of these control systems is essential to keep the vehicle in a safe and desirable driving condition. A model predictive controller (MPC) is a type of control system that employs an internal model of the system being controlled to predict its future behavior and determine the optimal control actions to achieve desired outcomes. The controller works by continuously updating its predictions based on the current state of the system and using an optimization algorithm to calculate the best control actions while satisfying any constraints on the system.

In each MPC controller, there is an objective function with a set of weights. These weights can directly affect the response of the system. The appropriate selection of weights results in the generation of an effective control action, which reduces tracking errors to a minimum. In the conventional MPC controllers, the focus is solely on optimizing the control actions, and weight values remain fixed or scheduled for different ranges of system operations. Therefore, the effects of real-time selection of optimum weights in the controller performance are overlooked.

This research aims to improve the performance of MPC control systems by developing a weight tuning and real-time weight selection scheme that considers the dynamic system's state. The proposed approach is applied to the vehicle stability control under a variety of environmental and/or driving conditions. The weight tuning is performed by using the prediction model of the vehicle and the Bayesian optimization (BO) technique. The weight selection is carried out in real-time by learning the adjusted weights through Gaussian process regression (GPR). These are two main modules developed to be used for selecting and tuning the weights of an MPC controller. Hence, in addition to optimizing control actions through the MPC controller's optimization problem, the weights of the MPC controller are also assessed and adjusted to achieve the highest level of optimality in the vehicle control system.

Furthermore, an authentication process is proposed to evaluate the tuned weights after being selected in the tests. This way, unnecessary increases or decreases in the weights stored in the weight selection dataset can be avoided. To further enhance the model

predictions, a blending-based multiple model approach is utilized. In this approach, instead of considering a fixed prediction model with invariant parameters, a combination of finite number of models with different parameters are considered. Based on the prediction error of each model, a weighted sum of matrices of these models are utilized both in the MPC controller and weight tuning modules.

To verify the proposed methodology, MATLAB/Simulink and CarSim co-simulations as well as experimental tests are carried out. Comparing the vehicle responses with and without the proposed weight tuning and real-time weight selection approach strongly corroborates the proposed technique in enhancing the controller performance. The capability of the proposed multiple model technique in improving the weight tuning has been demonstrated in the simulations and experimental results.

Acknowledgements

First and foremost, I extend my deep appreciation to my supervisor, Prof. Amir Khajepour, for his exceptional guidance, support, and encouragement throughout my PhD journey. I feel fortunate to have had the opportunity to work with such a knowledgeable mentor and it is due to his guidance that I have developed my current academic abilities, professional vision, and understanding of engineering ethics.

I also extend my gratitude to everyone at the University of Waterloo and General Motors who supported me throughout my journey. I will always cherish their contributions.

I would like to express my gratitude for the financial support received from Ontario Research Fund (ORF), Natural Sciences and Engineering Research Council of Canada (NSERC), and General Motors in this project.

Appreciation also goes to the technicians at the Waterloo Mechatronics Vehicle Systems laboratory, specifically Jeff Graansma, Aaron Sherratt, and Adrian Neill, for facilitating the experimental tests. I would also like to thank Dr. Ehsan Hashemi, Dr. Mohammad Pirani, and Dr. Reza Valiollahi Mehrizi for their significant technical support and insights during my PhD studies. My colleagues and friends at the University of Waterloo have been a great source of support, help, and camaraderie, so my gratitude goes to Mobin Khamooshi, Mehdi Zabihi, Reza Hajiloo, Amin Habibnejad, Ehsan Mohammadbagher, Mehdi Abroshan, and everyone else who has been part of my experience.

Above all, I extend my gratitude to my family, including my parents and sister, for their unwavering support and encouragement throughout this journey. I am grateful for their sacrifices and acknowledge that I would not have reached this point without them.

Dedication

This dissertation is dedicated to my parents for their unwavering love and encouragement, and to my sister for her invaluable support throughout my academic journey.

Table of Contents

List of Figures

List of Tables

List of Acronyms

ARX Auto-Regressive with eXogenous

AWD All-Wheel Drive

BO Bayesian Optimization

CAN Controlled Area Network

CAV Connected and Automated Vehicle

CG Center of Gravity

DLC Double Lane Change

DMC Dynamic Matrix Control

DYC Direct Yaw-moment Control

EI Expected Improvement

ES Entropy Search

EV Electric Vehicle

FALA Finite Action-set Learning Automation

FOPDT First-order Plus Dead Time

GA Genetic Algorithm

GP Gaussian Process

GPC Generalized Predictive Controller

GPR Gaussian Process Regression

GPS Global Positioning System

HDV Human Driven Vehicle

IAE Integral of the Absolute Error

IDT Interactive Decision Tree

IMU Inertial Measurement Unit

IRL Inverse Reinforcement Learning

LQG Linear Quadratic Gaussian

LQR Linear Quadratic Regulator

LTI Linear Time Invariant

LTV Linear Time Variant

MAC Model Algorithmic Control

MCA Motion Cueing Algorithm

MIMO Multiple-Input and Multiple-Output

MMAC Multiple-Model Adaptive Control

MOFDM Multi Objective Fuzzy Decision Making

MPC Model Predictive Control

NN Neural Network

PSO Particle Swarm Optimization

RL Reinforcement Learning

RMSE Root Mean Square Error

SE Squared Exponential

SISO Single Input and Single Output

SUV Sport Utility Vehicle

WA Weight Authentication

Chapter 1

Introduction

1.1 Motivation

Vehicle control systems play an important role in improving the motion of the vehicle in a variety of conditions. Model predictive controllers (MPCs) utilize a model of the vehicle to obtain optimal control actions based on the current state of the vehicle and the predictions made by the model. This is done by defining an objective function and solving the corresponding constrained optimization problem at each time step. In the MPC's objective function, there are also some weights that directly impact the overall response of the system.

Typically, in MPC controllers, only the control actions are optimized at each time step. The MPC weights, however, are usually tuned off-line and remain unchanged irrespective of the system's state. In vehicle control systems due to the non-linearity in vehicle dynamics, and changes in road conditions and vehicle mass, fixed set of controller weights does not result in a good performance in all driving conditions. Selecting a suitable set of weights that aligns with the current state of the vehicle can assist in preserving the optimal performance of a vehicle control system under different road, driving, and vehicle conditions.

In this study, a weight tuning approach and a real-time learning-based weight selection

method is developed for the vehicle yaw stability control. To evaluate a control system performance, the state errors can be monitored. A high-performance controller can keep state errors in a safe domain even under a variety of vehicle and road conditions. Each set of weights results in a different set of tracking errors corresponding to the current state of the vehicle. Hence, for each vehicle state, an optimal set of weights can be considered such that the controller's overall performance remains optimal. Through the proposed weight tuning technique, the weights are adjusted by using the system's prediction model and Bayesian optimization (BO). Afterwards, the tuned weights as well as the vehicle states are stored in the real-time weight selection dataset. The dataset is trained using the Gaussian process regression (GPR) method to employ the tuned weights in similar vehicle's states. In the next step, the tuned weights that has been selected by the controller in another simulation or experiment become authenticated based on the response of the vehicle.

The proposed weight tuning method is carried out by using the predictions made by the vehicle prediction model. Therefore, improving the accuracy of the model leads to better predictions and hence weight tuning. To enhance the weight tuning module's performance, a multiple-model approach is utilized for the MPC control and weight tuning processes, and the results are compared to those achieved when using a single nominal model.

1.2 Overview of the Proposed Approach

The general architecture of the proposed approach is illustrated in Figure 1.1. The driver, through his/her senses, receives information from the environment and provides the control actions through steering wheel, throttle, and brake pedals. The MPC controller receives sensory data, in addition to the driver inputs to make any additional control actions needed to maintain the the vehicle's stability while following the driver's desired intention. The main contributions of this study are illustrated as two modules in Figure 1.1: The first one is the weight tuning module, and the second one is the real-time learning-based weight selection. Through the weight tuning module, the MPC controller weights are evaluated and tuned. The tuned weights are then stored in the dataset of the weight selection module.

A fixed set of default weights are always defined for the MPC controller. After the

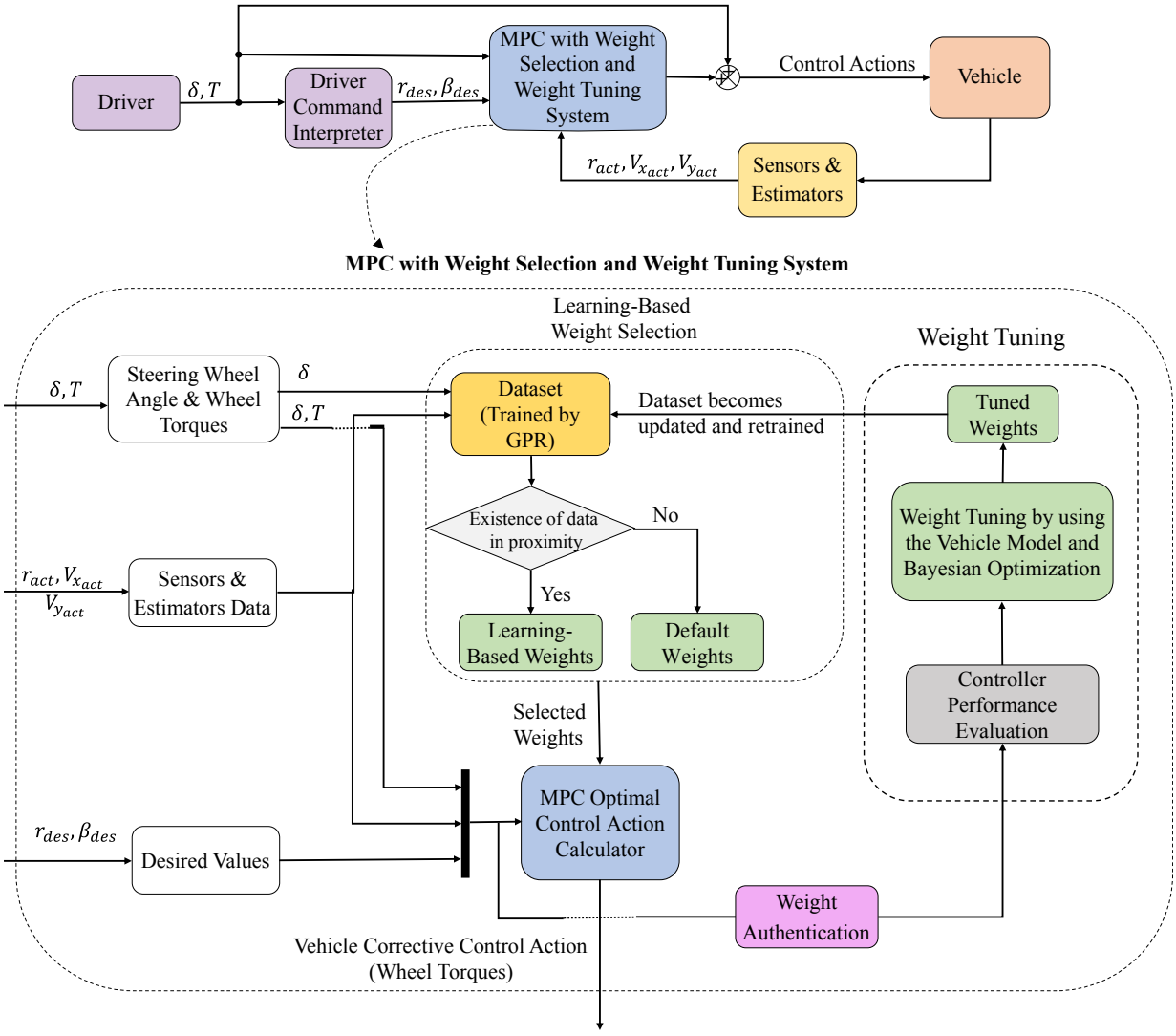

Figure 1.1: General architecture of the proposed approach.

control actions are generated by the controller, applied to the vehicle, and a maneuver is finished, the system's response is assessed based on the tracking errors, and then weights become adjusted if necessary. The weight tuning is performed off-line by using the vehicle prediction model and BO method. Next, the vehicle sensory and driver data as well as the corresponding tuned weights are stored in a dataset that is considered for the real-time

weight selection. Hence, the weight selection dataset becomes enriched and updated by the new data received from the weight tuning module. This dataset becomes trained by using the GPR learning method. As a result, instead of considering a fixed set of weights, the controller's weights are determined with respect to the state of the vehicle in current time.

The weight tuning is conducted according to the model's prediction utilized in the MPC. Thus, the accuracy of the model affects the tuning process. The tuned weights are evaluated after they are selected by the MPC controller in a simulation or a test. The state variable errors resulted by the tuned weights are compared with the previously selected weights for each vehicle state, and then unnecessary weight changes can be avoided through a weight authentication process provided in this study. Additionally, it is possible to enhance the model prediction accuracy by considering a multiple-model approach. Instead of a fixed single model, a blending-based multiple-model is used for the MPC controller and weight tuning calculations. As a result, more accurate predictions can be made and, hence, better tuning can be achieved.

Consequently, in this study, a weight tuning and a real-time learning-based weight selection method for a vehicle stability control system is developed. Its effectiveness in improving the performance of the controller to maintain vehicle stability under a variety of driver and road conditions is investigated using simulations and experimental tests.

1.3 Organization

Chapter 2 comprises a literature review of methods for MPC weight tuning, direct yaw control (DYC), and multiple-model control. Different techniques employed for tuning the weights including the analytical, optimization-based, and learning-based methods are reviewed. In the background section, the vehicle model that is considered for the MPC controller, the MPC controller design, GPR, and BO methods are explained.

In Chapter 3, the developed weight tuning and real-time learning-based weight selection methods are presented for a general MPC controller. Subsequently, the proposed approach is employed for a vehicle yaw stability controller. The studied MPC controller ensures

the vehicle stability by torque vectoring. Weight tuning is performed off-line, and weight selection is carried out in real-time. The details of the proposed weight authentication process are also explained in this chapter. Through co-simulations in MATLAB/Simulink and CarSim, the proposed approach is evaluated.

Chapter 4 describes the multiple-model approaches. A blending-based multiple model method is explained and applied to the MPC controller. A comparison is made between the predictions generated by the single-model and those produced by the multiple-model technique. Then, the blending-based multiple-model is employed as the weight tuning module's prediction model to enhance its tuning performance. The effect of considering a multiple-model method instead of a single model is evaluated and discussed by performing MATLAB/Simulink and CarSim co-simulations.

In Chapter 5, experimental results of the proposed weight tuning and real-time weight selection method are illustrated. The vehicle undergoes various tests under dry and wet road conditions. The tests are conducted using the controller with fixed weights, followed by the controller with the single model tuning approach, and finally the tests are carried out using the controller with the blending-based multiple model approach. To confirm the efficacy of the proposed approaches, a comparison is made between the results obtained for yaw rate, side-slip angle, wheel torques, and tuned weights.

Chapter 6 provides a conclusion to this thesis and suggests potential areas for future research to build upon the findings presented in this thesis.

Chapter 2

Literature Review and Background

In this chapter, first, a comprehensive review of the previous studies in the area of tuning MPC controllers is provided. Then, the DYC control approaches, and multiple model-based control studies in the literature are reviewed. After reviewing the literature, the background information required for developing the MPC controller with weight tuning is provided including: the vehicle model used as the MPC controller's prediction model; the details of the MPC controller's design; the GPR; and BO method that employed for the purpose of this research.

2.1 Model Predictive Control

The efficacy of MPC as a control strategy has been established through numerous studies and real-world applications [1]. Rafal and Stevens [2], Propoi [3], and Zadeh et al. [4], have paved the way for the researches in this field, and it has progressed greatly and been adopted in numerous sectors [5–12]. The MPC control method has particular applicability for multivariate complex systems that are constrained [1]. Stochastic MPC [13–15] and economic MPC [16–18] are study topics that encompass the recent researches in MPC. Since its inception, MPC formulation and execution have undergone significant development. Nonlinear MPC, state-space MPC, dynamic matrix control (DMC) [19], linear quadratic gaussian (LQG) [20], and model algorithmic control (MAC) [21] are among them.

The MPC controller is an optimal control method that control actions are computed according to the predictions made by using a model of the system for a predefined future horizon. An objective function is minimized to obtain the optimal control actions which includes different terms, weights, and parameters to reflect the desired performance and intended control objectives. The performance of any MPC controller is closely dependant on the weight values, and numerous studies in the literature have focused on tuning them. In the following section, different methods and approaches in the area of MPC weight tuning are reviewed.

2.2 Tuning Model Predictive Controller Weights

It has always been challenging to tune the weights of optimal linear quadratic regulator (LQR) and MPC controllers [22, 23]. It takes a long time to choose an appropriate set of weights, and numerous simulation or experimental tests, and a ton of trial-and-error decisions are needed [24,25]. A detailed review of heuristic and analytical tuning approaches for various implementations of MPC control is presented in [20], which includes the studies published up through 2009.

The general linear discretized state-space equation of a dynamic system can be represented as:

$$\begin{aligned} x_p(k+1) &= A_p x(k) + B_p u(k) \\ y_p(k) &= C_p x(k) + D_p u(k) \end{aligned} \tag{2.1}$$

where A_p, B_p, C_p, and D_p denote the plant matrices, $x_p(k)$ indicates the state variables vector, and $u(k)$ represent the manipulated variables vector. The objective function of the MPC controller is written as:

$$J = \big[Y_d(k) - Y(k)\big]^T Q \big[Y_d(k) - Y(k)\big] + U^T(k) R U(k) + \Delta U(k)^T T \Delta U(k) \tag{2.2}$$

subject to:

$$\begin{aligned} EU(k) &\leq e \\ FX(k) &\leq f \end{aligned} \tag{2.3}$$

Eq. (2.3) represents the actuator and state constraints. The elements of Q matrix are the weight values considered for the tracking errors, T matrix includes the weights of proximity of inputs to the previous solution, and R matrix contains the weights for the magnitude of inputs. The relative significance of minimizing the tracking error of each state is reflected by the corresponding relative weight values in Q. They relatively indicate the costs associated with the difference between the state variable values and their corresponding setpoints. The control actions magnitude and rate of variations are significantly influenced by the weight elements of R and T matrices.

The weight selection is mostly based on observed relations that the plant model describes. In some studies, general guidelines are provided to select the weights (e.g. [26]). Nevertheless, when plant parameter uncertainties and model mismatches exist, its performance is negatively impacted. There are difficulties in tuning the MPC controller weights through conventional methods including the necessity of a standard controller, an accurate model, and continuous cost functions. Some of the widely used tuning approaches are provide in the following.

2.2.1 Analytical Weight Tuning

In some studies, analytical approaches have been developed for obtaining and tuning the MPC controller weights. Bagheri and Sedigh [27, 28] constructed closed-form equations for tuning the weight matrices utilizing the pole placement idea for first-order plus dead time (FOPDT) model of the plant. Rapid tracking performance was obtained after evaluating the proposed tuning instructions. Nevertheless, when employed for inaccurate, or time-variant plant models, the performance of the controller was not desirable. This study was expanded in [29] by developing other equations for calculating the weights in closed-form, which was capable of decreasing the controller's tracking error. For multiple-input and multiple-output (MIMO) systems, the approach of [27] was expanded in [30]. In [31], MPC weights are calculated by closed form equations derived for a linear time invariant (LTI) controller and a MPC controller by utilizng a controller-matching technique. Correlations between the weight matrices and covariances of these matrices are investigated in [32], and an on-line MPC tuning method is proposed. This method is applied to a multi-dimensional

robotic system.

In a large number of studies, it is recommended that $Q = I$ and $T = \rho I$ be used. ρ is referred to as a move suppression factor. Even though this approach can make MPC tuning straightforward, it reduces MPC's extensibility and restricts the amount of controllability of the MPC controller. Bagheri and Sedigh [33] used this approach to find an analytical limit for the value of ρ that gives resilience to the uncertainty of the model. Yamashita, et al. [34] suggested having $T = I$ and $Q = R = I$, as well as considering a tiny number for ρ. Burgos, et al. [35] proposed considering $T = \rho I$ and $Q = C^T C$. This approach was employed to address the reference tracking problem of a quadrotor which produced acceptable setpoint tracking results. Other relevant studies and tuning guidelines have been provided for the SISO and MIMO systems in [36–41].

A single-input and single-output (SISO) MPC with no offsets is developed in [42] on the basis of an auto-regressive model with exogenous terms (ARX). Authors proposed obtaining the elements of the weight matrices according to the closed-loop system's input and output variances. The proposed method entails computing the variances at various weight values, ranging from no control to minimal variance control. The weights that correspond to the point of inflection in a log-log graph of the variances are selected.

In [30], an analytical weight tuning procedure is provided when there are inactive constraints. Instead of using numerical methods, MPC weight tuning formulas for control horizon of one are derived to achieve the optimal performance.

2.2.2 Optimization-Based Weight Tuning

Tuning strategies on the basis of optimization methods for several control systems, particularly MPC, are established in the literature. Weight tuning approaches based on optimization were demonstrated to perform better than trial-and-error when the prediction horizon and sampling intervals are fixed [43]. However, it is a challenging procedure to formulate the cost function, which must be minimized for the determination of MPC weights. Several efforts have been documented in the literature in this regard [44–46]. A greater variety of MPC tuning approaches have been introduced for the cost function customization and the

choice of optimization procedures. The formulation of the cost functions are performed utilizing both the economic factors and information about the current plant.

In [47], the values of the weight matrices are obtained by defining a cost function and minimizing it. To adjust weights, Vallerio, et al. [48] used a multi-objective optimization strategy that aimed to minimize the discrepancy between the desired and measured values. Another multi-objective technique is utilized in [46]. Tracking errors' L2-norm is minimized while hard constraints are considered for the weights.

Shah and Engell [49] investigated tuning the weights of an MPC controller for SISO systems without active constraints through closed-loop pole placement. The relationships between the weights, closed-loop zeros, and poles were identified, followed by determining the optimal weights through the solution of an optimization problem. Following that, they offered a methodical procedure for calculating the MIMO MPC controller weights in [50]. This strategy relies on a characterization of the closed-loop system's expected behavior. The system's robustness to model discrepancy is investigated in this study. The tuned weight values were achieved by the following steps, after constructing the requisite transfer functions of the closed-loop system. First, the difference between the actual and the intended transfer function was minimized in order to achieve the gain of the desired transfer function. Then , the weight matrices' values were determined after the solution of an optimization problem which also sought to minimize the desired and actual transfer functions discrepancy.

Two techniques of weight tuning on the basis of lexicographic multi-objective optimization were introduced in [51]. The user orders the objective functions in this approach according to significance. In the first technique, input-output pairing is performed; then, control actions and control outputs are normalized; finally, the sum of the squared errors between the desired and actual responses of the closed-loop system is minimized by solving the corresponding optimization problem. The alternative technique takes into account the same performance index, while finding viable weights that yield the nearest response to the response which is both feasible and desired.

In [52], an optimization strategy was provided that determines the MPC weights by minimizing the variance of a gain factor. In the proposed minimization problem, the

difference between the desired and actual response is constrained. This method was utilized to tune a steering-wheel position controller. A technique for tuning the MPC controller for non-square systems was provided in [53]. Soft state variable constraints are considered in the formulation of the designed MPC controller. They used a three-step approach to tune the controller: 1) The determination of a desired closed-loop response; 2) determining the best scales for the control actions; and 3) Obtaining the optimal weight values through the proposed optimization problem solution. The slack variables that are considered to apply the soft constraints, and the difference between the simulated and desired responses of the closed-loop system are minimized by solving this optimization problem.

For the purpose of tuning the weights of the generalized predictive controller (GPC), Romero, et al. [54] proposed a technique in which the weight optimal values are obtained by minimizing the amount of gain margin under some constraints. The constraints include: phase margin bounds, robustness to changes in the plant gain, rejection of the acceptable output-disturbance, and rejection of the high-frequency noise. Olesen, et al. [55, 56] presented a weight tuning method for a MIMO ARX model-based offset-free MPC. The optimal weight values are achieved by minimization of the integral of the absolute error (IAE) when there are variations in the disturbance and setpoint, and weight values are constrained.

In [57], the weight tuning was performed through solving an optimization problem in which the disturbance and control effort are minimized under a defined constraint. This constraint is defined based on ∞-norm of weights set by the user, and functions of sensitivity between the output response and the disturbance of the load.

On the basis of BO method, an automated weight tuning process is provided in [58] for an MPC controller that is employed to control a vehicle's lateral motion. The approach produces optimal gains based on the determination of the user, aside from making the procedure more convenient by lowering the workload. The proposed approach is evaluated by simulation tests under a driving test case. The vehicle can undertake lane-keeping movements at various speeds.

Kumar and Zavala [59] provided a thorough derivation of BO technique for tuning MPC controller. The proposed approach treats the relationship between the tuning parameters of

the MPC controller and its closed-loop functionality as a black box. The tuning parameters space is aimed to be explored and exploited strategically to rapidly obtain the optimal tuning parameters, in the provided framework.

In order to address the mismatch between the actual dynamic model and the linearized model, BO is employed in [60] to optimize an objective function's weights. The proposed framework automatically enhances a starting set of controller gains in accordance with a pre-specified performance goal assessed by experimental data. The entropy search (ES) is considered as the acquisition function of the utilized BO method.

Dependable performance has been achieved by MPC weight tuning using the constrained least-square optimization method [61]. To be able to find the best combination of MPC weights that ensures the repeatability of the process and satisfies predetermined time-domain criteria, gradient descent technique is examined in [62].

In [63], a goal attainment method is used for tuning MPC controllers weights. In this methodology, a group of desired objectives is defined corresponding to a group of objective functions, such that each objective function is a function of MPC tuning parameters.Several desirable objectives that may be considered by specifying a relative weight indicating the relative importance of each one.

2.2.3 Learning-Based Weight Tuning

Learning algorithms have also been employed for tuning the MPC weight. In the most of the studies in this area, optimal weights are achieved by combining an optimization problem with a learning-based method according to the intended objective.

In [64, 65], the optimum values of MPC weights were learned by using the deep reinforcement learning (RL) method, and these weights could be adapted and changed online. The policy of the proposed RL approach is capable of being learned autonomously. A reward function which is multi-objective has been employed in this study. It has been demonstrated that the performance of the proposed method exceeds those produced by a human expert. A similar concept of using RL to learn the MPC controller tuning parameters is provided in [66]. A quad-copter's MPC controller was tuned by using an RL

method based on Q-Learning which had a piece-wise reward function, and its action space was discrete.

An optimal MPC weight adaptation technique is provided in [67] for a designed game theoretic MPC problem formulated based on the interaction between a human-driven vehicle (HDV) and connected and automated vehicles (CAVs). An inverse reinforcement learning (IRL) method is utilized to learn the HDV objective function's weights, and an approach is proposed on the basis of BO for adapting the weights in the objective function of CAV for minimizing the expected true cost. The proposed optimal approach is evaluated by simulation tests conducted for a vehicle crossing example.

In [68,69], the authors used finite action-set learning automation (FALA) to determine an MPC controller weight elements. Using this method, the weight values converge to an optimal element set by using an iterative training procedure. In [70], a framework is provided in which a neural network (NN) is used for predicting the cost function learned by humans in combination with an MPC weight tuning problem. This problem is posed as an optimization problem. A random search algorithm using "random oracle" object is selected for optimization. Constrained optimization problems are considered in this algorithm and the optimization can be done without requiring derivative information of any order. The focus of the author in this study is to automate the procedure of weight tuning to reduce the cost and time of tuning. For evaluating the performance of the framework, it is used in a simulation for tuning the weights of a MPC controller designed for diesel engine's air path.

2.2.4 Genetic Algorithm-Based Weight Tuning

The genetic algorithm (GA) is a metaheuristic technique that is employed for solving search and optimization problems. Natural selection, the mechanism that propels biological evolution, is the foundation of GA. Some of the researches have proposed using GA for obtaining the optimal MPC weights.

In [71], a multi-objective GA is utilized to optimize the weights of an MPC-based motion cueing algorithm (MCA). It is intended to minimize sensed motion errors, output

displacements, input rates, and motion inputs. A set of predetermined requirements relating to the maximum permitted displacement and maximum permitted error must be met by the modified weights.Through the suggested technique, the best combination of weights is searched, which seeks to minimize the effort of the user associated with weight tuning, and receives feedback regarding the user satisfaction.

In [72], weight values of the MPC that can provide the minimum energy consumption along with decreased tracking error, are determined using GA. Importance-weighted performance indicators are utilized for the formulation of the objective function. Desired response tracking IAE and energy consumption are among the considered metrics. To specify importance weights for each control situation, an interactive decision tree (IDT) is employed.

A GA-based multi-objective optimization method is presented in [73] for tuning prediction horizon and weights of the MPC controller. The defined cost function is minimized for obtaining the optimal values. In comparison to the iterative weighted tuning method, the GA-based algorithms can offer a quicker convergence. A generalized automated MPC weight tuning algorithm is developed in [74]. Multi-objective fuzzy decision making (MOFDM) is combined with GA in the proposed approach. The "optimum" response of the system is determined by the engineer and the algorithm aims at providing the MPC controller with the tuning parameters that result in this optimal response.

Many of the weight tuning methods on the basis of evolutionary algorithms generally employ two main approaches: (i) in a large number of studies, the objective function is reformulated by considering new objectives [24, 25, 75]. In this method, the designer's judgment is used to decide one the weight values in the new objectives, leaving the issue of determining the ideal weights unresolved. Furthermore, a successful tuning of the previous objective functions may not always follow from the new objective functions weight tuning. (ii) A vector of weights is considered as the ideal set of weights. Then, weights become optimized, and then the results get evaluated by comparing them to those obtained by the reference weights. The reference weight values are usually determined by trial and error. Therefore, these weights are biased and are not a reliable measure for assessment. There are instances, though, where the same MPC objective weights are used, or normalizing weights depending on each objective's maximum value, are chosen [76]. It is also possible

to combine the reference weights with additional objectives [77]. Nevertheless, the reference weights are always determined by the designer which may not be the ideal weight values to be considered for the MPC.

2.2.5 Particle Swarm Optimization-Based Weight Tuning

Particle swarm optimization (PSO) is a powerful technique for solving the optimization problems that seeks to solve a problem more effectively by repeatedly attempting to make a candidate solution better in terms of a specified quality metric [78]. PSO has been the focus of various optimization-related studies including the MPC weight tuning.

A technique that integrates PSO and multi-objective optimization was proposed in [79] to adjust a constrained MPC controller weights. The MPC's prediction model is subject to uncertainty. A worst-case scenario for the controller is considered in this method in terms of the resiliency index and condition number [80] in the description of the model uncertainty. The user is able to select the intended performance functions. To optimize a multi-objective cost function in tracking a set point and disturbance rejection test cases, a MPC weight tuning method based on PSO is employed in [81]. PSO algorithm is also used for automatically estimating the nonlinear MPC weights in [82, 83].

The PSO optimization method was utilized in [84] to tune the weights of SISO and MIMO MPC controllers. The settling time, rise time, overshoot, and steady-state error are minimized by solving the provided optimization problem which takes into account some constraints and gives the optimal weight values. Weights are bounded in this approach.

2.2.6 Other Tuning Methods

One of the extensively used methods to obtain the weight values is controller matching. It entails adjusting the MPC weights to make it operate similar to a typical, predetermined controller. Two MPC weight tuning techniques are proposed in [85] on the basis of controller matching. A linear MPC controller without state or input constraints is matched to a desired LTI state feedback controller. An optimization problem is defined based on

this matching problem such that the elements of the weight matrices are bounded, and solution of this optimization problem gives the optimal weights.

Online altering the structure of control is the topic of other relevant research. A multiple-model MPC control strategy is provided in [86]. To address ongoing fluctuations in the operating point and nonlinearities of the model, switching between various MPC controllers is performed. More similar researches have been conducted that involve online change in the structure of the controller, e.g. [61, 87–91]. Shadmand, et al. [92] presented the application of IAE in tracking in performing the MPC weight tuning online. Maran [23] used a weight tuning method that involved manually changing the weights. The weights have to be sufficiently large to prevent impossibility, and without impairing motion perception at the same time.

In almost all of the proposed MPC weight tuning approaches, after the weights are obtained or calculated, they remain fixed although the weights have a crucial effect in the controller's performance. In this thesis, A technique is developed to tune the weights with respect to the state of the system. Therefore, the weights can be changed and selected in real-time according the system's state.

2.3 Direct Yaw Control

The corrective yaw moments produced by torque/brake vectoring are referred to as direct yaw moment control (DYC). The DYC directly affects longitudinal forces to control the vehicle's yaw rate. By providing braking force to the wheels, a corrective yaw moment is generated through the differential braking to maintain the yaw rate close to the desired value. Numerous research have focused into how differential braking affects control of the vehicle.

In [93], differential braking was utilized in the designed MPC controller to control the tractor-trailer vehicle's stability. A friction circle was used to calculate the tire's remaining capacity to limit the maximum value of corrective yaw moment. The performance of the designed controller was evaluated under various trailer and tractor differential braking scenarios.

A control method based on fuzzy logic was developed in [94] to control the stability of vehicle. The deferential braking was performed by brake-by-wire system. A nonlinear model of the vehicle was employed to design the controller. There was an assumption that yaw rate remains within a stable range allowing the driver to react to the fluctuation in the yaw rate which prevents the vehicle to become unstable.

Utilizing the electric powertrain, a practical way is offered in [95] to enhance the stability and handling of the vehicle by managing the amount of torque that is generated and applied to each wheel. In contrast to the differential braking control method, torque vectoring does not decrease the speed of the vehicle which is the reason that torque vectoring is usually preferred. Active differentials and electric motors are employed as the means of distributing wheel torques respectively in conventional and electric vehicles (EVs).

In [96], a modular optimal control approach was provided for vehicle's combined lateral and longitudinal control. In this approach, the applied torque on each wheel is controller. To minimize the tracking errors, the required change in the yaw moment and longitudinal force was determined by using a high-level MPC controller. In accordance with the control actions determined by the high-level controller, the torque applied on each wheel was regulated by using a low-level controller.

In [97], an optimal vehicle control method based on torque vectoring was used to provide a generalized integrated control approach. In order to maintain the vehicle on a desired route, the designed controller adjusted yaw moment and tire forces through applying sufficient amount of torque on each wheel. Lower and upper bounds was considered for the amount of generated wheel torque in the optimization problem that was used for obtaining the torque values.

By controlling the applied wheel torque on each wheel, an MPC controller is designed in [98] to perform the combined control of stability of the vehicle and wheel traction. Yaw rate tracking was the objective of the designed controller subject to the constraint of keeping the lateral velocity of the vehicle and slip ratio of the tire within the predetermined safe regions. The proposed method decreased the computational complexity by reducing the prediction model size. By modifying the reference yaw rate, the controller indirectly accomplished the vehicle's lateral stability.

In [99], to stabilize an EV in real-time and within the handling limits, a nonlinear MPC is developed.Three MPC techniques to maintain the stability of the vehicle were developed. Control actuation was the torque vectoring of the rear wheels.

2.4 Multiple Model Control

Due to changes in the dynamics of the plant, failure or aging of the components, or variations in the operating conditions, parametric uncertainties can grow significantly in experiment. Instability or unsatisfactory transient response may be resulted by these large parametric uncertainties. To deal with these problems across a wider range of system changes and uncertainties, multiple model control approaches have been constructed.

Utilization of multiple models in the design of controllers allows for a more thorough explanation and improved control of systems that are time varying and have various uncertainties. Rather than relying on a single model that may have a significant uncertainty range, this approach suggests employing a collection of models to reflect the system behavior under different operating conditions. This method enables the design of a combination of controllers to create a blended or switching-based control structure in a setting that is noticeably less conservative.

In order to increase the precision of state estimation, multiple-model Kalman filter was first proposed in [100, 101]. A linear SISO system with significant modeling inaccuracies was taken into account in [102]. A group of controllers that were under the supervision of a switching logic in high-level were used. The focus of this study was primarily on the switching logics. Moreover, the flexibility of the switching-based approach compared to the conventional controllers are highlighted in this research. In [103], the benefits of switching-based methods over conventional adaptive control approaches are discussed including the modularity, low computational cost, and ease of parameterization.

To minimize the controller's poor response due to the switch from one fixed controller to another one, an approach of switching and tuning was developed in [104, 105]. The basic concept was to enhance the performance by tuning the gain of the controller after employing switching logic to speed up the adaptation. Additionally, this method may be

applied to uncertain linear time variant (LTV) systems as opposed to the basic approach, which were designed for uncertain LTI systems. This idea is also implemented in [106,107] for parameter identification.

Some influential studies [108,109] have been conducted in the area of traditional multiple-model control that concentrated on the quantity of fixed and adaptive models used for estimation, as well as the information source of each model. These studies suggested blending the gains of predetermined controllers according to their model-specific identification errors. In comparison to earlier multiple-model based strategies, it has demonstrated an enhanced performance and a quicker convergence [110]. A blending-based multiple-model adaptive control (MMAC) scheme is developed in [111,112] for MIMO systems with polytopic parameter uncertainties. LQ based and MPC based designs are provided and applied to the vehicle motion control. In the proposed approach, weighting vectors are generated for a set of fixed linear parametric identification models in real-time to obtain the true model.

In the following sections, the background information required for developing the MPC controller with the weight tuning and the weight selection is provided. First, the vehicle model used as the MPC controller's prediction model is explained. The details of the MPC controller's design are then described. Following this, the GPR and BO techniques that employed for the purpose of this study are discussed.

2.5 Vehicle Prediction Model

In order to design an MPC control system, a prediction model is required. This model includes the yaw motion, and lateral motion. Additionally, this model incorporates the effects of torque vectoring. In Figure 2.1, the dynamics of a vehicle having in-wheel electric motors is illustrated. The equations of the lateral dynamics of the vehicle can be expressed as:

$$\dot{\beta} = -r + \frac{1}{mu}\left(F_{yf}\cos\left(\delta\right) + F_{xf}\sin\left(\delta\right) + F_{yr}\right) \tag{2.4}$$

$$\dot{r} = \frac{1}{I_z}\left(l_f F_{yf}\cos\left(\delta\right) - l_r F_{yr} + l_f F_{xf}\sin\left(\delta\right) + M_{DY}\right) \tag{2.5}$$

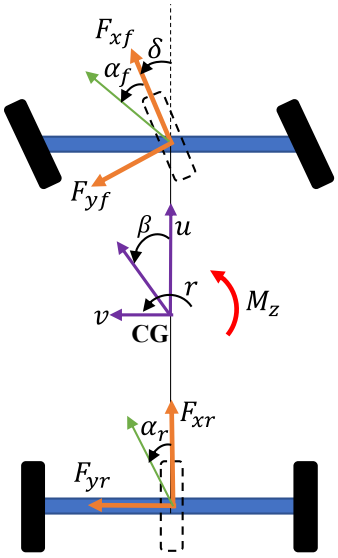

Figure 2.1: Vehicle model diagram

where β is the sideslip angle; r is the yaw rate; m is the total mass of the vehicle; I_z is the vehicle's yaw inertia; l_f and l_r are the distances between the vehicle's center of gravity (CG), and the front and rear axles respectively; M_{DY} is the total corrective yaw moment acted on the vehicle; and g is the gravitational constant.

Torque vectoring is used as the control actuation in this thesis. This actuation generates separate torques for each of the front wheels in order to control the stability of the vehicle. To develop a combined-slip prediction model, it is necessary to incorporate this effect into the tire model. It is important to note, however, that combined slip tire models require an estimation of a large number of parameters. Furthermore, an accurate estimate of slip ratio may not be available. Thus, to overcome these challenges and complexities, a brush tire model is used to consider the effect that the longitudinal forces have on lateral forces [113]:

$$F_y = \begin{cases} -C_\alpha \tan\alpha + \frac{C_\alpha^2}{3\xi\mu F_z} |\tan\alpha| \tan\alpha - \frac{C_\alpha^3}{27\xi^2\mu^2 F_z^2} \tan^3\alpha & |\alpha| < \alpha_{sl} \\ -\xi\mu F_z \text{ sign } \alpha & |\alpha| \geq \alpha_{sl} \end{cases} \tag{2.6}$$

$$\alpha_{sl} = \arctan \frac{3\xi\mu F_z}{C_\alpha},$$

where C_α denotes the cornering stiffness of the tire; α is the slip angle of each of the front and rear tires:

$$\alpha_f = \frac{v + l_f r}{u} - \delta, \qquad \alpha_r = \frac{v - l_r r}{u} \tag{2.7}$$

F_Z represents the normal load on each tire; and, based on the friction limit circle, ξ is a derating factor indicating the remaining lateral force capacity:

$$\xi = \sqrt{1 - \left(\frac{F_x}{\mu F_z}\right)^2} \tag{2.8}$$

A linear relationship between longitudinal forces of the tires and wheel torques is assumed:

$$F_x = \frac{Q}{R_e} \tag{2.9}$$

By linearizing the nonlinear plant dynamics at each timestep, a convex quadratic optimization problem is utilized to implement the MPC controller in real-time. In order to obtain this linear vehicle model, the nonlinear characteristic of the tire can be linearized around its operating point. Calculation of the partial derivatives of the lateral force is required in order to linearize the nonlinear model with respect to slip angle and derating factor. F_{yi} can be expressed as follows using these partial derivatives:

$$F_{yi} = \bar{F}_{yi} + \bar{C}_{\alpha_i} \left(\alpha_i - \bar{\alpha}_i\right) + \bar{C}_{\xi_i} \left(\xi_i - \bar{\xi}_i\right), \tag{2.10}$$

where $\bar{\alpha}_i$ and $\bar{\xi}_i$ are respectively the slip angle and derating factor at the operating point; F_{yi} is the lateral force computed at the operating point by Eq. 2.6; $\bar{C}_{\alpha_i}$ and $\bar{C}_{\xi_i}$ are calculated at the operating point as follows:

$$\bar{C}_{\alpha_i} = \frac{\partial F_{yi}}{\partial \alpha_i}|_{\bar{\alpha}_i, \bar{\xi}_i}, \qquad \bar{C}_{\xi_i} = \frac{\partial F_{yi}}{\partial \xi_i}|_{\bar{\alpha}_i, \bar{\xi}_i}. \tag{2.11}$$

In other words, $\bar{C}_\alpha$ is the slope of the tangent line of the Lateral force, F_y, versus side slip angle, α, at the operating point, as shown in Figure 2.2:

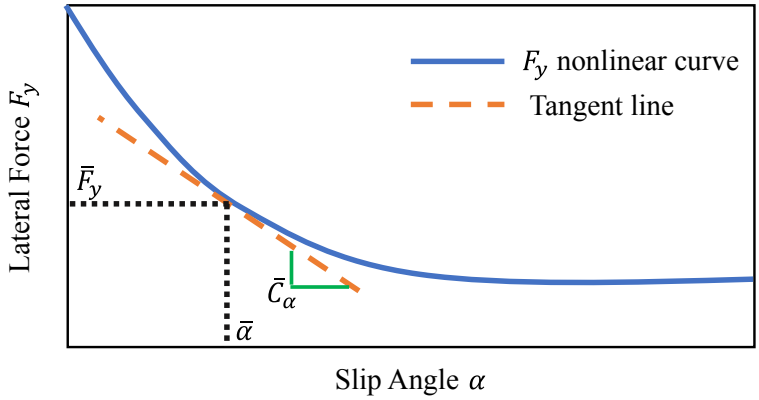

Figure 2.2: Representation of the linearized tire model with respect to side slip angle α.

Similarly, $\bar{C}_\xi$ is the slope of the tangent line of the Lateral force, F_y, and derating factor, ξ, at the operating point, as shown in Figure 2.3.

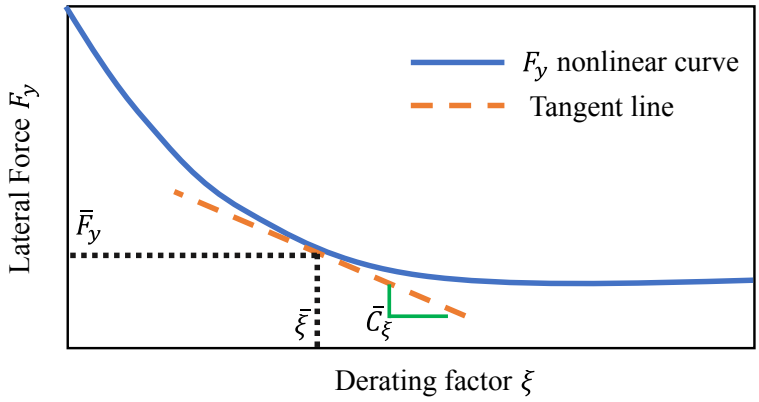

Figure 2.3: Representation of the linearized tire model with respect to derating factor ξ.

It is possible to rewrite the lateral force equation in terms of axle torque by utilizing successive partial derivatives:

$$F_{yi} = \bar{F}_{yi} + \bar{C}_{\alpha_i} \left(\alpha_i - \bar{\alpha}_i\right) + \bar{C}_{Q_i} \left(Q_i - \bar{Q}_i\right), \tag{2.12}$$

where $\bar{C}_{Q_i}$ is calculated at the operating point as:

$$\bar{C}_{Q_i} = \left[\frac{\partial F_{yi}}{\partial \xi_i} \frac{\partial \xi_i}{\partial F_{xi}} \frac{\partial F_{xi}}{\partial Q_i} \right]_{\bar{\alpha}_i, \bar{\xi}_i} \tag{2.13}$$

Eq. (2.12) is substituted into Eq. (2.4) and (2.5) for the lateral forces of the front and rear tires, and by using Eq. (2.7) for slip angles, state-space form of the equations can be rewritten as:

$$\dot{x} = Ax + Bu + D, \tag{2.14}$$

where $x = \begin{bmatrix} \beta \\ r \end{bmatrix}$, $u = \begin{bmatrix} Q_f \\ M_{DY} \end{bmatrix}$, and

$$A = \begin{bmatrix} \frac{\bar{C}_{\alpha_f} \cos\delta + \bar{C}_{\alpha_r}}{mu} & \frac{l_f \bar{C}_{\alpha_f} \cos\delta - l_r \bar{C}_{\alpha_r}}{mu^2} - 1 \\ \frac{l_f \bar{C}_{\alpha_f} \cos\delta - l_r \bar{C}_{\alpha_r}}{I_z} & \frac{l_f^2 \bar{C}_{\alpha_f} \cos\delta + l_r^2 \bar{C}_{\alpha_r}}{I_z u} \end{bmatrix} \tag{2.15}$$

$$B = \begin{bmatrix} \frac{\bar{C}_{Q_f} \cos\delta - \bar{C}_{Q_r}}{mu} + \frac{\sin\delta}{muR_e} & 0 \\ \frac{l_f \bar{C}_{Q_f} \cos\delta + l_r \bar{C}_{Q_r}}{I_z} + \frac{l_f \sin\delta}{I_z R_e} & \frac{1}{I_z} \end{bmatrix}$$

$$D = \begin{bmatrix} \frac{\bar{F}_{yf} \cos\delta + \bar{F}_{yr} - \bar{C}_{\alpha_f} \bar{\alpha}_f \cos\delta - \bar{C}_{\alpha_r} \bar{\alpha}_r - \bar{C}_{Q_f} \bar{Q}_f \cos\delta - \bar{C}_{Q_r} \bar{Q}_r - \bar{C}_{\alpha_f} \delta \cos\delta}{mu} \\ \frac{l_f \bar{F}_{yf} \cos\delta - l_r \bar{F}_{yr} - l_f \bar{C}_{\alpha_f} \bar{\alpha}_f \cos\delta + l_r \bar{C}_{\alpha_r} \bar{\alpha}_r - l_f \bar{C}_{Q_f} \bar{Q}_f \cos\delta + l_r \bar{C}_{Q_r} \bar{Q}_r - l_f \bar{C}_{\alpha_f} \delta \cos\delta}{I_z} \end{bmatrix}$$

The corrective direct yaw moment, M_{DY}, is generated by torque vectoring which is utilized for the simulations and experimental tests this thesis. Eq. (2.14) has to be discretized to be able to be implemented in the MPC controller. The zero-order hold (ZOH)

method [114] is used to discretize the continuous-time model as:

$$x(k+1) = A_{dis}x(k) + B_{dis}u(k) + D_{dis}, \tag{2.16}$$

where $A_{dis} = e^{A_{dis}t_s}, B_{dis} = \int_0^{t_s} e^{A_{dis}\tau} B_{dis}.d\tau$, and $D_{dis} = \int_0^{t_s} e^{A_{dis}\tau} D.d\tau$. This discretized vehicle model is considered as the prediction model of the MPC controller.

2.6 Actuator Constraints

The actuator effort of each MPC controller must be limited to solve the optimization problem. The vehicle stability controllers are generally subject to two main actuator constraints: The first constraint concerns the actuators' maximum capability to generate wheel torque. The maximum friction force between the road and tire is another constraint that must be considered.

The drive torques are generated by the electric motors in the front and rear axles. The maximum torque that can be generated by the electric motor, $Q_{e,max}$, and the maximum friction force, μF_z, constrain the wheel torques. The minimum total torque generated by the front wheels are calculated as:

$$Q_{f,min} = Q_{fl} + Q_{fr} = 0, \tag{2.17}$$

The rear wheel torques are obtained such that the total torque applied to all wheels are equal to the driver requested torque:

$$Q_r = Q_d + Q_f, \tag{2.18}$$

where Q_d is the driver's total requested torque, and Q_r is the rear wheel torques. Both the front and rear torques must be constrained within the allowable ranges. Therefore, the

minimum and maximum front wheel torques are calculated as:

$$\begin{aligned} &Q_{f,min} \leq Q_f \leq Q_{f,max}, \\ &Q_{f,min} = Q_d - Q_{r,max}, \\ &Q_{f,max} = \min\left(Q_{e,max}, \mu F_{z,f}, Q_d\right), \\ &Q_{r,max} = \min\left(Q_{e,max}, \mu F_{z,r}, Q_d\right) \end{aligned} \tag{2.19}$$

2.7 State Constraints

It is necessary to constrain some of the vehicle model's states in order to maintain the vehicle in a stable condition. To define the maximum and minimum allowable vehicle's yaw rate, the steady state condition is assumed in Eq. (2.14):

$$\dot{v} = 0 \qquad \rightarrow \qquad r_{max} = \frac{F_{Y,max}}{mu} \tag{2.20}$$

In the absence of longitudinal forces acting on the tires, the maximum stable yaw rate is:

$$F_x = 0 \qquad \rightarrow \qquad r_{max} = \frac{\mu g}{u} \tag{2.21}$$

Otherwise, the capacity of the tire's lateral force will be decreased on the basis of the derating factor, ξ, as:

$$F_{Y,max} = \mu m g \xi \tag{2.22}$$

where ξ is obtained by Eq. (2.8). Hence, the maximum vehicle's stable yaw rate will be:

$$r_{max} = \frac{\mu g \xi}{u} \tag{2.23}$$

and the minimum stable yaw rate will be:

$$r_{min} = \frac{-\mu g \xi}{u} \tag{2.24}$$

A linear inequality can be used to represent the yaw rate constraint:

$$H_r x \leq G_r, \tag{2.25}$$

where H_r and G_r are expressed as:

$$H_r = \begin{bmatrix} 0 & 1 \\ 0 & 1 \end{bmatrix}, \qquad G_r = \begin{bmatrix} r_{max} \\ r_{min} \end{bmatrix} \tag{2.26}$$

The sideslip angle of the vehicle should also be constrained to ensure the vehicle stability. The bounds of the sideslip angle are obtained by limiting the rear slip angle of the rear tires to:

$$|\alpha_r| \leq \alpha_{r,sat} \tag{2.27}$$

which results in:

$$\frac{l_r r}{u} - \alpha_{r,sat} \leq \beta \leq \frac{l_r r}{u} + \alpha_{r,sat}. \tag{2.28}$$

The Eq. (2.28) can be rewritten in the linear equality matrix from as:

$$H_\beta x \leq G_\beta, \tag{2.29}$$

where H_β and G_β are:

$$H_\beta = \begin{bmatrix} 1 & -\frac{l_r}{u} \\ -1 & \frac{l_r}{u} \end{bmatrix}, \qquad G_\beta = \begin{bmatrix} \alpha_{r,sat} \\ \alpha_{r,sat} \end{bmatrix}. \tag{2.30}$$

Figure 2.4 shows the constraints on the lateral velocity and yaw rate of the vehicle in the $\beta - r$ phase plane. The green region indicates the vehicle's stable area.

2.8 Desired Response of the Vehicle

The desired values of the vehicle's yaw rate and sideslip angle should be determined for the MPC controller. These values are tracked by the controller to improve the handling of the

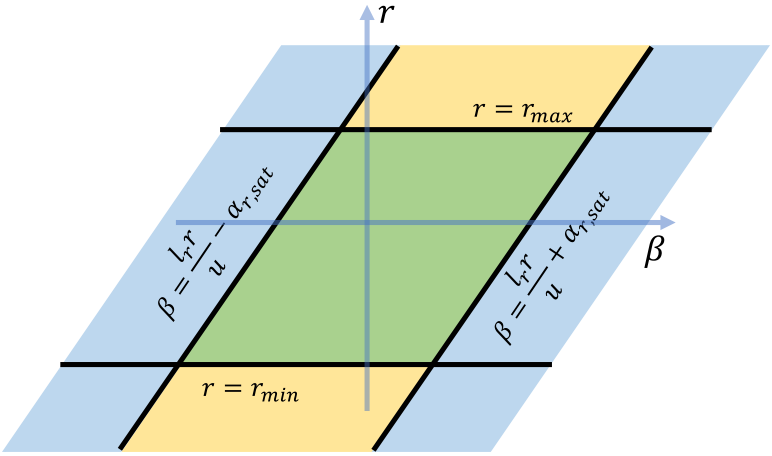

Figure 2.4: The $\beta - r$ phase plane

vehicle. When a vehicle's sideslip angle is small, it means that the lateral velocity is also small and the vehicle is traveling mostly in the direction of its heading. In such a scenario, the sideslip angle is imperceptible, and the vehicle appears to be moving tangentially to its trajectory at all times. This is desirable for stability because it means that the vehicle is not experiencing large lateral forces that could cause it to lose control or tip over. However, if the sideslip angle becomes large, it means that the vehicle is experiencing a significant lateral velocity and is traveling more sideways than forward. This can cause the vehicle to become unstable, particularly at high speeds, and may result in the driver losing control of the vehicle. In extreme cases, a large sideslip angle can cause the vehicle to tip over or spin out of control. Therefore, to minimize the sideslip angle to the greatest extent possible, the desired sideslip angle is determined as:

$$\beta_{des} = 0, \tag{2.31}$$

By considering the steady state yaw rate of the vehicle operating in the linear range, the desired value of the vehicle's yaw rate is determined as:

$$r^*_{des} = \frac{u}{l + k_{us}u^2}\delta, \tag{2.32}$$

where $l = l_f + l_r$, and k_{us} is the understeer coefficient obtained as:

$$k_{us} = \frac{-m\left(aC_{\alpha_f} - bC_{\alpha_r}\right)}{lC_{\alpha_f}C_{\alpha_r}}. \tag{2.33}$$

k_{us} indicates how sensitive the vehicle is to the steering. In the evaluation of the quality of the vehicle's handling, understeer coefficient is an essential parameter. In some circumstances, there is an insufficient friction force to track the desired yaw rate defined in Eq. (2.32) based on the friction coefficient of the road. Hence, the desired yaw rate is changed and limited by the maximum allowable yaw rate defined in Eq. (2.23). The adjusted desired yaw rate can be expressed as:

$$r_{des} = \min\left(r_{max},\ |r^*_{des}|\right)\operatorname{sign}(r^*_{des}) \tag{2.34}$$

2.9 Model Predictive Control for Lateral Stability

This section discusses the MPC controller used to control the vehicle lateral stability. The control actuation is the front wheel torque vectoring that is applied by separate electric motors on each wheel. There are two main objectives defined for this controller: maintaining the vehicle's yaw rate and sideslip angle within the stable region and tracking the desired yaw rate and sideslip angle. The objective function of the MPC controller can therefore be defined based on the aforementioned control actuation and objectives.

This objective function is formed by the weighted summation of the following main terms: (1) system states' tracking error, (2) magnitude of the control actuations or control effort, and (3) proximity of the control actuations to their previous values. Additionally, two types of constraints are considered in this objective function: (1) a hard constraint for the control actuations, and (2) soft constraints for the yaw rate and side slip angle based on limits shown in Eqs. (2.25) to (2.30). The soft constraints are applied by defining slack variables and they are weighted to be penalized for the violation of the corresponding state

constraints. The final objective function of the MPC controller can be expressed as:

$$J = \sum_{k=1}^{N_p} \big(x\,(k) - x_{des}\,(k)\big)^T Q \big(x\,(k) - x_{des}\,(k)\big) + u^T R u\,(k) + \Delta u^T\,(k)\, T \Delta u\,(k) + W_s s\,(k)\,, \tag{2.35}$$

subject to the following constraints:

$$\begin{aligned} Hx(k) &\leq G + s(k), \\ u_{min}(k) &\leq u(k) \leq u_{max}(k), \\ s(k) &\geq 0. \end{aligned} \tag{2.36}$$

and the discretized prediction model of the vehicle (Eq. (2.16)), where Q is the weight matrix of the tracking error, R denotes the weight matrix of the control effort, T is the weight matrix for the control actions' proximity, and W_s is the slack variables' weight matrix. x_{des} is the vector of desired states, N_p denotes the number of time steps in the prediction horizon, and s in the vector of slack variables to enforce the soft constraints on the states.

Using the stability analysis techniques, it can be demonstrated that the MPC controller is stable [115–117], and we do not discuss that in this thesis. At each time step, the convex optimization problem with the objective function of Eq. (2.35) is solved with respect to the linear constraints provided in Eq. (2.36). The qpOASES solver [118] is used to solve this quadratic optimization problem. Upon solving the optimization problem at each time step, a sequence of optimal control inputs in the form of a vector, denoted by u^*, is obtained as:

$$u^* = \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_{N_p} \end{bmatrix} \tag{2.37}$$

The initial sequence of wheel torque inputs denoted by u_1 is the only one that is applied

to the vehicle wheels. The optimization problem is subsequently resolved at the next time step, taking into account the updated measured data.

2.10 Gaussian Process Regression

Supervised learning for classification or regression can involve using Gaussian processes (GPs) as a probabilistic model. These flexible and adaptable models can capture the complexity of non-linear functions. For regression tasks, GPs can offer estimates of output values along with uncertainties for inputs. For classification tasks, GPs can provide probability distributions for classes instead of definitive outputs. GPs offer an effective solution for supervised learning with predictions and uncertainties that can be useful in various applications. GPs are employed when any finite number of random values in the output space could be considered to have a joint gaussian distribution based on the input variables [119].

To specify a GP, it is necessary to define a mean function denoted by $m(x)$ and a covariance function (or kernel) represented by $k(x, x')$. This GP considers a multivariate gaussian distribution based on the previous observations data points, $D = \{x_i, y_i\}_{i=1:N}$ which is assumed to come from an unknown nonlinear function $f : \mathbb{R}^d \to \mathbb{R}$:

$$f(x) \sim \mathcal{GP}\big(m(x), k(x, x')\big) \tag{2.38}$$

$$y_{1:N} \sim \mathcal{N}\big(m_{1:N},\ K(x_{1:N}, x_{1:N})\big) \tag{2.39}$$

$$y_{1:N} = f(x_{1:N}),\quad m_{1:N} = m(x_{1:N}) \tag{2.40}$$

where K is the covariance matrix that is calculated elementwise by the kernel function $k(x, x')$. Using the observed data, a nonlinear function $y^* = f(x^*)$ is trained and the output of this learned function for unobserved or inexperienced data points x^* in the feature space can be predicted as:

$$y^*|\mathcal{D}, x^* \sim \mathcal{N}\big(\mu(x^*|\mathcal{D}),\ \sigma(x^*|\mathcal{D})\big) \tag{2.41}$$

where

$$\mu\left(x^{*}|\mathcal{D}\right)=m^{*}+k\left(x^{*},x_{1:N}\right)K_{1:N}^{-1}\left(y_{1:N}-m_{1:N}\right) \tag{2.42}$$

and

$$\sigma\left(x^{*}|\mathcal{D}\right)=k\left(x^{*},x^{*}\right)+k\left(x^{*},x_{1:N}\right)K_{1:N}^{-1}k\left(x_{1:N},x^{*}\right) \tag{2.43}$$

$$K_{1:N}=K\left(x_{1:N},\ x_{1:N}\right)$$

The selection of the kernel function is very crucial here because the assumptions on the y^{*} are determined by this element. One of the commonly utilized covariance functions is the squared exponential (SE) kernel. In this study, the SE kernel function is selected and is defined as:

$$k\left(x_{i},x_{j}|\theta\right)=\sigma_{f}^{2}\ exp\left[-\frac{1}{2}\frac{\left(x_{i}-x_{j}\right)^{T}\left(x_{i}-x_{j}\right)}{\sigma_{l}^{2}}\right] \tag{2.44}$$

where θ is the general term indicating the kernel function hyperparameters, σ_{f} is the measurement uncertainty, and σ_{l} is the characteristic length scale. Suitable hyperparameters θ are those that give the maximum log marginal likelihood for the observed data. Then, the output of any unobserved data points in the input feature space can be predicted. The GPR steps are illustrated in Figure 2.5. Therefore, based on the observed data, the GPR predicts the most probable output which is the $\mu(x^{*}|\mathcal{D})$ and the probability distribution of other possible outputs as the $\sigma(x^{*}|\mathcal{D})$ function.

2.11 Bayesian Optimization

The GPR regression method, which is categorized under supervised learning, is used to forecast the output of unobserved data points as a function of the observed data. In some cases, using the observed data, it is required to find the data in the input feature space corresponding to the optimum output. In other words, the observed data contains some information about the correlation between inputs and outputs that can help to find data points in the input feature space that maximizes the output of the system. BO is one of

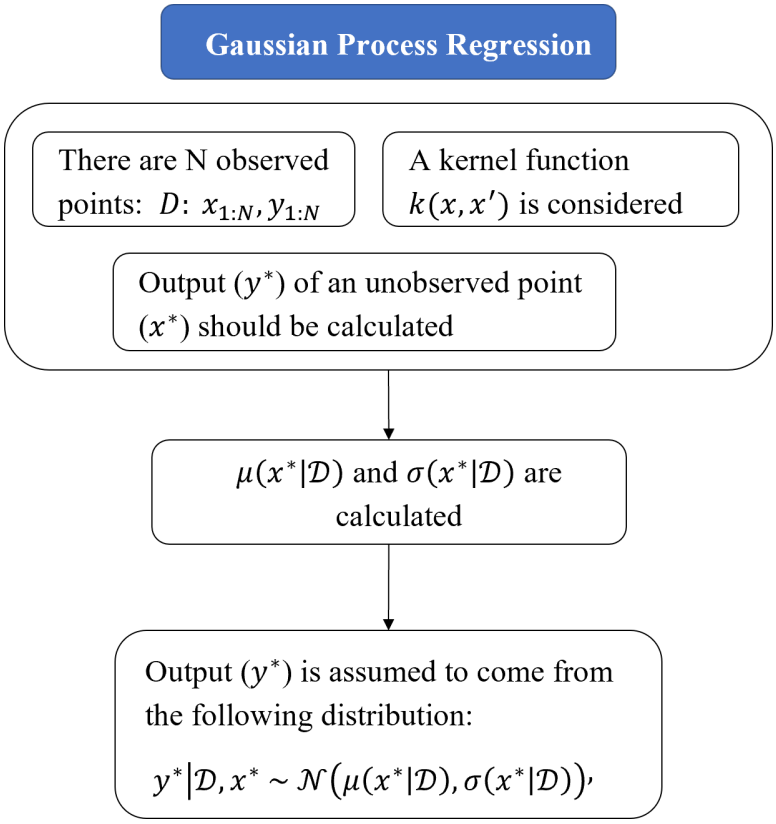

Figure 2.5: The diagram of the GPR method.

the widely used methods for obtaining global optimum of black-box functions. First, a GP must be learned by the initial sample points. Using an auxiliary acquisition function, $\alpha(x)$, BO technique gives the next possible sample point that optimizes the GP mean [120].

In the current study, the next sampling point is determined using the expected improvement (EI) acquisition function, which estimates the potential improvement for any input data point in maximizing the output of the learned function over the current best output value, represented by $y_{\max} = \max(y_{1:N+k})$. The EI function calculates the expectation of the improvement amount based on N initial observations and K additional sampled points, which is defined as:

$$\alpha\left(x\right) = EI\left(x\right) = \sigma\left(x\right)\left[u\Phi\left(u\right) + \phi(u)\right] \tag{2.45}$$

where $u = (y(x) - y_{max}) / \sigma(x)$, $\sigma(x)$ is the predicted variance by the learned GP, $\Phi(.)$ is the cumulative density function, and $\phi(.)$ is the probability density function for the normal distribution. EI function can propose next sample points with high variance and/or high mean.

To better clarify the BO process, the workflow is presented in Figure 2.6, and an example is illustrated in Figure 2.7. The area with the purple color in Figure 2.7 indicates the range of the GP model prediction with 95% confidence. The GP model is learned by 5 initial sample points and the corresponding output values calculated by $f(x)$. The algorithm computes the location of the next potentially optimal point by using the acquisition function $\alpha(x)$. This point is crucial in the iterative process of updating the GP model. The vertical green dashed line denotes the location of this point. During each iteration of the algorithm, the output of the learned function at the next possible optimum point, $f(X_{opt})$, is calculated. This value is then used in conjunction with all the previously evaluated parameters (represented by the red dots) to update the GP model so it can better represent the observed data and increase its predictive accuracy. This iterative process of updating the GP model based on new information is critical to the success of the algorithm.

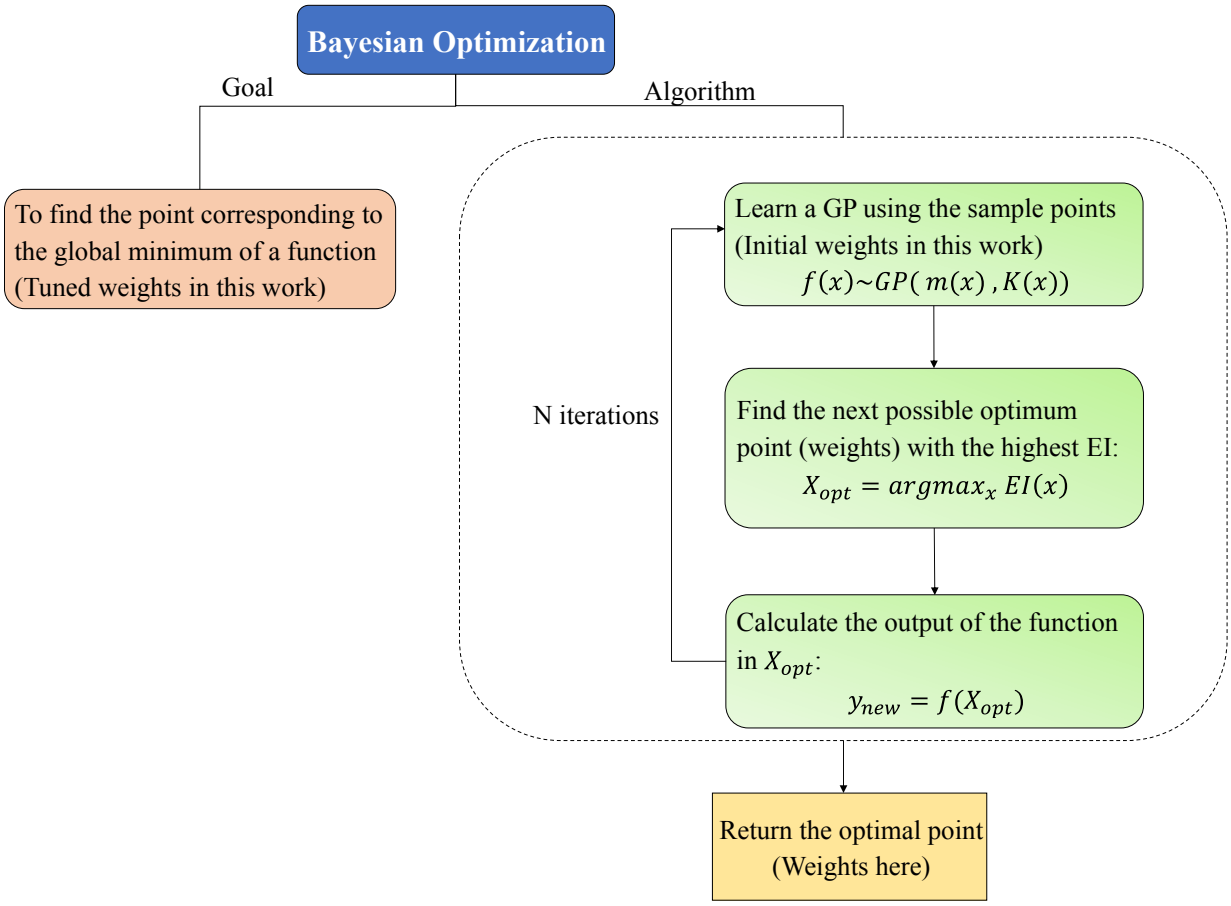

Figure 2.6: Diagram of the BO algorithm.

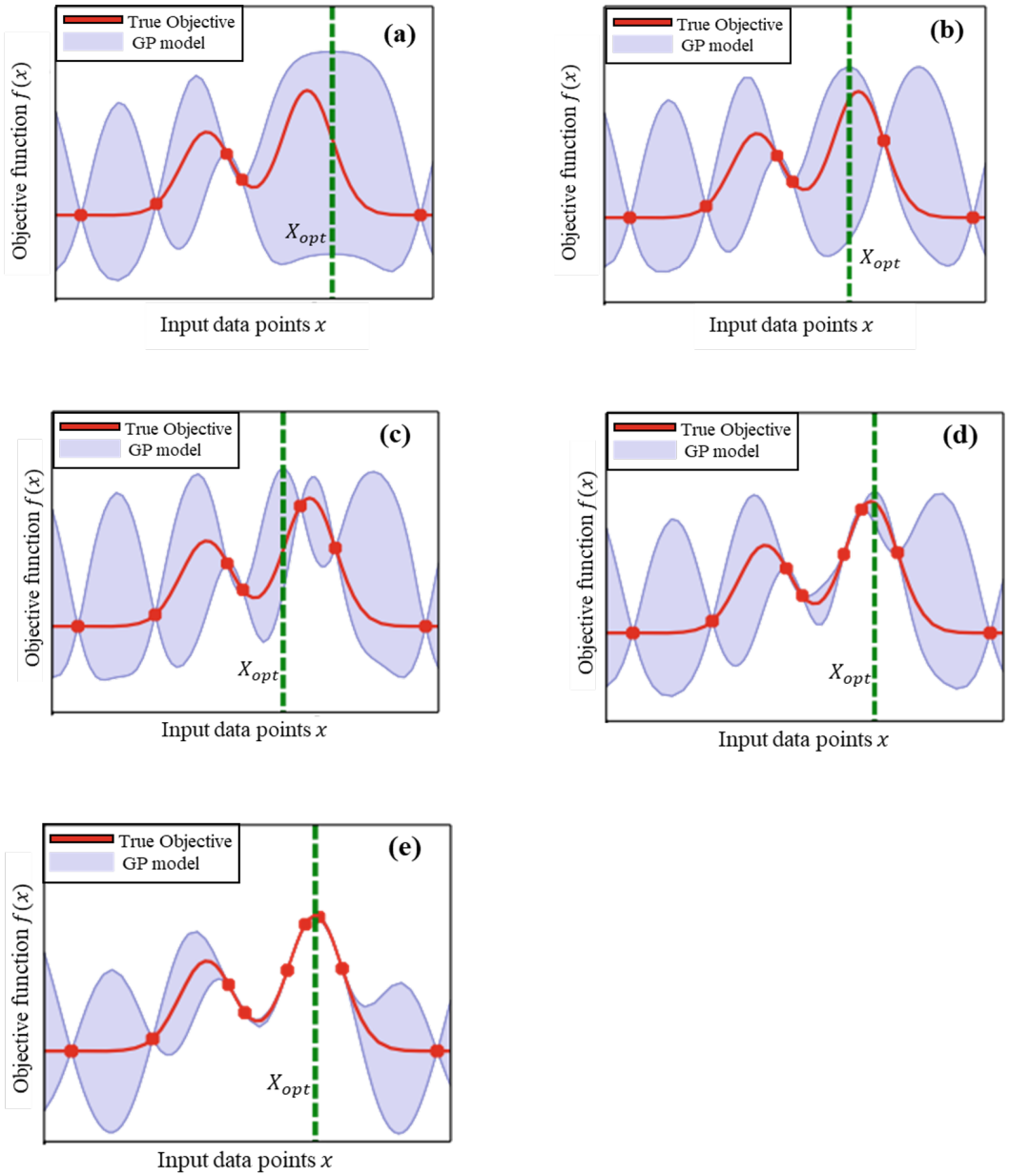

Figure 2.7: An example of the BO process when optimizing an unknown one-dimensional objective function [121].

Chapter 3

MPC Weight Tuning and Weight Authentication

As previously mentioned in Chapter 1, the proposed approach consists of these modules: MPC controller, weight tuning, weight selection, and weight authentication. The MPC controller design has been explained in section 2.9. In this chapter, (as highlighted in Figure 3.1) the details of the proposed method for tuning the MPC controller weights are discussed. This approach is then applied to the MPC controller designed for the vehicle's lateral stability to evaluate its effectiveness. In the next step, the real-time learning-based weight selection based on GPR will be discussed, and finally the weight authentication technique is explained.

3.1 Bayesian Optimization-Based Weight Tuning

In this study, weight tuning is accomplished through a BO based technique. The first step toward defining a BO problem involves learning a GP from the observed or initially available data. It is important to note that GPR is utilized in two distinct modules. In the first instance, the GP is formed to formulate a BO problem for weight tuning. In the second instance, the BO is formed to train a specific dataset that is considered for the

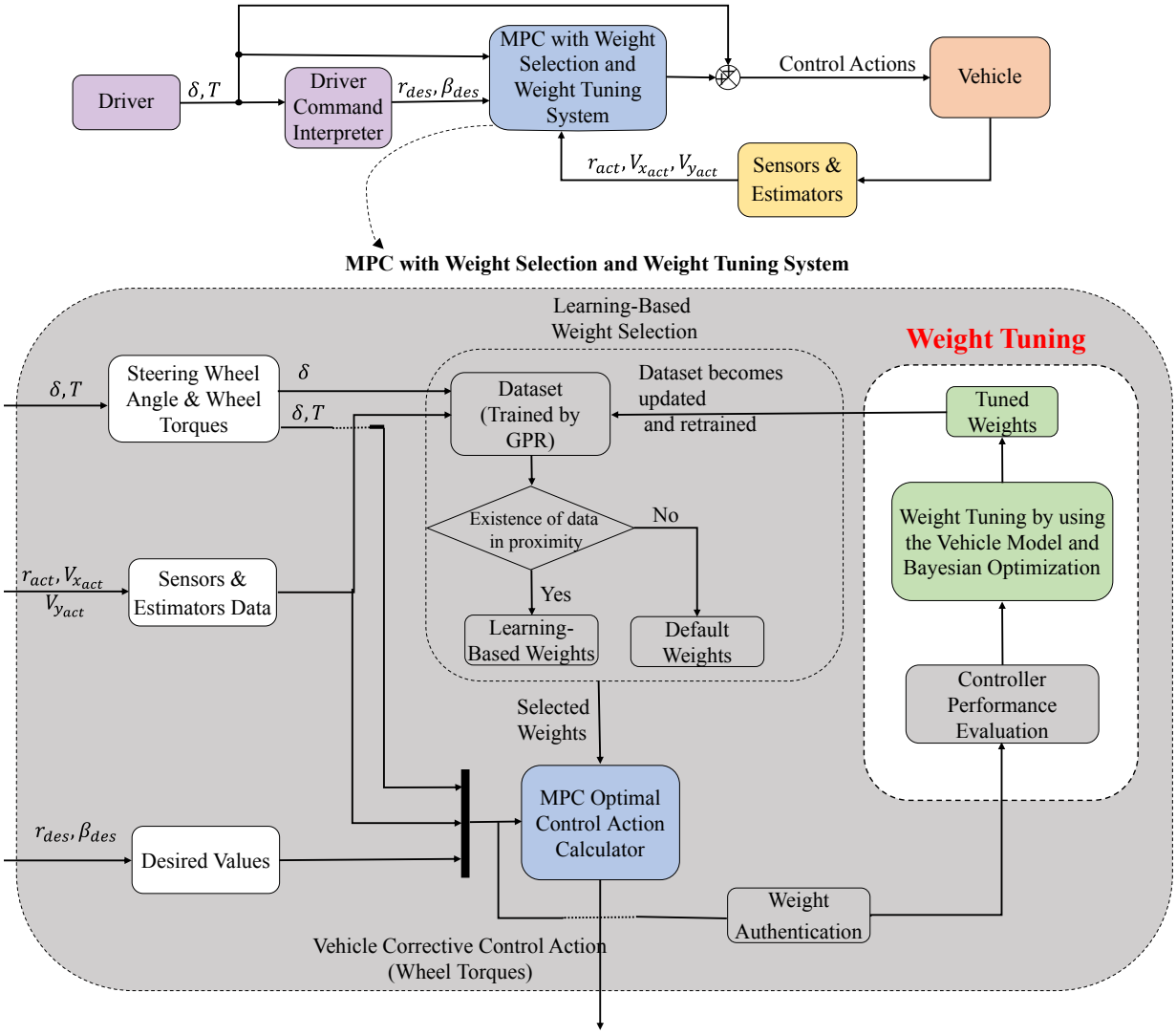

Figure 3.1: The scheme of the weight tuning module

weight selection module. Thus, these two applications of GPR should not be confused.

In this section, details of the weight tuning for a general MPC controller are presented. It is assumed that the MPC controller weights have already been set, and the focus is on tuning them to obtain a better MPC performance. The general state-space representation of a dynamic system can be written as:

$$\begin{aligned} \dot{X} &= F(X,U) \\ Y &= H(X,U) \end{aligned} \tag{3.1}$$

where X represents the vector of states, U is the vector of inputs of the dynamic system or control actions, and Y represents the output vector. An MPC controller is designed to provide the system with suitable corrective control actions, δU, such that the system outputs, Y, get closer to the desired output values, Y_d:

$$\begin{aligned} Y_d &= [y_{1d}, y_{2d}, \ldots, y_{nd}\ ]^T \\ er_Y &= [er_{y1}, er_{y2}, \ldots, er_{yn}\ ]^T = [y_1 - y_{1d},\ y_2 - y_{2d},\ \ldots,\ y_n - y_{nd}] = Y - Y_d \end{aligned} \tag{3.2}$$

where er_Y refers to the vector of tracking errors, and er_{yi} refers to the tracking error associated with the i th output. Desired output values, y_{id}, are obtained as a function of current states, X, and inputs, U, of the system. The MPC controller generates the corrective control actions, represented by the δU vector as:

$$\begin{aligned} \delta U &= [\delta u_1, \delta u_2, \ldots., \delta u_m]^T = \mathcal{MPC}[1](X, U, Y_d, Q, R, T, W_s) \\ Q &= diag(w_1, w_2, \ldots,\ w_n) \end{aligned} \tag{3.3}$$

These control actions are obtained by solving an optimization problem. The optimization problem includes an objective function which, according to Eq. (3.3), is a function of states, inputs, desired values, and the controller weights. In this thesis, the elements of the Q matrix associated with the tracking errors are intended to be tuned. Weights have a significant influence on the performance of the control system. According to Eq. (3.3), the weight values can be specified in accordance with the current condition of the system and

[1]The notation $\mathcal{MPC}(.)$ is used to represent the whole calculations performed for solving the optimization problem of the MPC controller to obtain the corrective control actions, which can be treated as a function of variables in(.).

the control objective for which the MPC controller is designed. In other words, rather than considering a fixed set of weights for all of the conditions that the vehicle will experience, the weights can be adjusted and tuned for each condition.

In what follows, a weight tuning technique is explained to obtain optimal weights by using BO and a prediction model of the dynamic system. A prediction model to be used by the MPC controller can be defined as:

$$\begin{aligned} \dot{\hat{X}} &= \hat{F}(\hat{X}, U) \\ \hat{Y} &= \hat{H}(\hat{X}, U) \end{aligned} \tag{3.4}$$

where $\hat{X}$[2] and $\hat{Y}$ are the predicted states and outputs of the vehicle system, respectively. Depending on the control objective, an error examination criterion, $\bar{E}_Q$, corresponding to the weight matrix, Q, is defined as a function of tracking errors, er_Y, in current and previous time steps:

$$\bar{E}_Q = E(er_Y^t, er_Y^{t-1}, er_Y^{t-2}, \ldots, er_Y^{t-l}) \tag{3.5}$$

where er_Y^t denotes the vector of tracking error in the time step of t. As previously mentioned, tracking errors are impacted by the weights. Therefore, $\bar{E}_Q$ is indirectly obtained by the weights at each time step. In order to determine whether the selected weight set in the MPC controller is appropriate for the current condition of the dynamic system, a threshold can be considered for $\bar{E}_Q$, as $\bar{E}_{Q_{tres}}$. If $\bar{E}_Q$ is less than this threshold, then the selected weight set is acceptable. Otherwise, another weight combination must be chosen, or, in other words, the selected weights should be tuned. The $\bar{E}_Q$ and er_Y are obtained by using the system's actual outputs and the corresponding desired values. For tuning the MPC controller weights in this study, the prediction model of the dynamic system (Eq. (3.4)) is used to predict the tracking errors for other possible weights. As an additional clarification, similar to Eqs. (3.2) and (3.5), predicted tracking errors, $\widehat{er}_Y$, and predicted

[2]Throughout this thesis, the superscript $\hat{}$ denotes the predicted variable obtained through the model, rather than measured or estimated.

error examination criterion, $\widehat{\bar{E}}_Q$, are defined as:

$$\widehat{er}_Y = [\widehat{er}_{y1}, \widehat{er}_{y2}, \ldots, \widehat{er}_{yn}\ ]^T = [\hat{y}_1 - y_{1d},\ \hat{y}_2 - y_{2d},\ \ldots,\ \hat{y}_n - y_{nd}]^T = \hat{Y} - Y_d \tag{3.6}$$

$$\widehat{\bar{E}}_Q = E(\widehat{er}_Y^t, \widehat{er}_Y^{t-1}, \widehat{er}_Y^{t-2}, \ldots, \widehat{er}_Y^{t-l}) \tag{3.7}$$

so that the performance of the control system by using the possible better weights can be predicted. The $\widehat{er}_Y^t$ is obtained by calculating the difference between the predicted outputs and the desired ones at time step t. $\widehat{\bar{E}}_Q$ is calculated as a function of predicted tracking errors in l number of previous time steps if other weight sets were used. In other words, we want to predict what would be the tracking errors in a sequence of previous time steps if the current weights of the MPC controller were substituted by other weights. These predicted values are calculated and used for weight tuning through the following steps:

First, an initial set of new weights and corresponding Q matrices are generated either randomly or by using a predefined algorithm as:

$$\begin{aligned} Q_1^{initial} &= \begin{bmatrix} W_1^1 & 0 & \cdots & 0 \\ 0 & W_2^1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & W_n^1 \end{bmatrix},\ Q_2^{initial} = \begin{bmatrix} W_1^2 & 0 & \cdots & 0 \\ 0 & W_2^2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & W_n^2 \end{bmatrix},\ \ldots, \\ Q_r^{initial} &= \begin{bmatrix} W_1^r & 0 & \cdots & 0 \\ 0 & W_2^r & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & W_n^r \end{bmatrix} \end{aligned} \tag{3.8}$$

The above initial weights as well as the already selected weights (presented by Q) in the MPC controller are considered as the input data points for the BO:

$$\text{BO initial points: } \left\{Q_1^{initial},\ Q_2^{initial}, \ldots,\ Q_r^{initial}, Q\right\} \tag{3.9}$$

Each of the weights in Eq. (3.9) are selected in the MPC controller's objective function,

and, using the prediction model (Eq. (3.4)), the MPC controller's optimal control actions, δU_j, associated with Q_j are obtained for each time step as:

$$\delta U_j^t = \mathcal{MPC}\left(X^{t-1},\ U^{t-1},\ Y_d^{t-1}, Q_j, R, T, W_s\right) \tag{3.10}$$

Therefore, at each time step, t, a vector of optimal control actions, δU_j^t is obtained based on each of the initial weight sets, and the state of the vehicle in the previous time step:

$$\begin{array}{ccc} Q_1^{initial} & \rightarrow [3] & \delta U_1^t \\ Q_2^{initial} & \rightarrow & \delta U_2^t \\ \vdots & \vdots & \vdots \\ Q_r^{initial} & \rightarrow & \delta U_r^t \\ Q & \rightarrow & \delta U^t \end{array} \tag{3.11}$$

By using the prediction model (Eq. (3.4)), the predicted outputs, $\hat{Y}_j^t$, and the corresponding predicted tracking errors, $\widehat{er}_{Y_j}^t$, under corrective control actions, δU_j^t, are calculated as:

$$\hat{Y}_j^t = H\left(X^{t-1},\ U^{t-1} + \delta U_j\right) \rightarrow \widehat{er}_{Y_j}^t = \hat{Y}_j^t - Y_d \tag{3.12}$$

Therefore, the Eq. (3.11) can be rewritten as:

$$\begin{array}{ccccccccc} Q_1^{initial} & \rightarrow & \delta U_1^{t-l:t}\ [4] & \rightarrow & \hat{Y}_1^{t-l:t} & \rightarrow & \widehat{er}_{Y_1}^{t-l:t} & \rightarrow & \hat{\bar{E}}_{Q_1} \\ Q_2^{initial} & \rightarrow & \delta U_2^{t-l:t} & \rightarrow & \hat{Y}_2^{t-l:t} & \rightarrow & \widehat{er}_{Y_2}^{t-l:t} & \rightarrow & \hat{\bar{E}}_{Q_2} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ Q_r^{initial} & \rightarrow & \delta U_r^{t-l:t} & \rightarrow & \hat{Y}_r^{t-l:t} & \rightarrow & \widehat{er}_{Y_r}^{t-l:t} & \rightarrow & \hat{\bar{E}}_{Q_r} \\ Q & \rightarrow & \delta U^{t-l:t} & \rightarrow & \hat{Y}^{t-l:t} & \rightarrow & \widehat{er}_{Y}^{t-l:t} & \rightarrow & \hat{\bar{E}}_{Q} \end{array} \tag{3.13}$$

[3] The notation A→B is used to represent the relationship between A and B, where A is a factor or condition that can lead to or generate B. The arrow (→) indicates that A can result in or lead to the production of B.

[4] $A^{t-l:t} = [A^{t-l}, A^{t-l+1}, \cdots, A^{t-1}, A^t]$

The elements of the first column in Eq. (3.13), $\left\{Q_1^{initial},\ Q_2^{initial},\ldots,\ Q_r^{initial},Q\right\}$, are considered as the BO initial input points, and the last column elements, $\left\{\widehat{\bar{E}}_{Q_1},\widehat{\bar{E}}_{Q_2},\ldots,\widehat{\bar{E}}_{Q_r},\widehat{\bar{E}}_{Q}\right\}$ form the BO problem's output points:

$$\begin{aligned}\text{BO initial inputs}: BO_{inps} &= \left\{Q_1^{initial},\ Q_2^{initial},\ldots,\ Q_r^{initial},Q\right\}\\ \text{BO initial outputs}: BO_{outs} &= \left\{\widehat{\bar{E}}_{Q_1},\widehat{\bar{E}}_{Q_2},\ldots,\widehat{\bar{E}}_{Q_r},\widehat{\bar{E}}_{Q}\right\}\end{aligned} \tag{3.14}$$

According to the details explained in section 2.11 and illustrated in Figure 2.6, the goal of this BO problem is to find the optimum weight set, Q_{opt}, that minimizes the error examination criterion, $\bar{E}_Q$, and, as a result, minimize the MPC controller's tracking errors, er_Y. Based on the information provided in sections 2.10 and 2.11, to formulate the BO problem, a GP must be learned for the BO initial data points presented in Eq. (3.14). This GP considers a multivariate gaussian distribution based on the initial data points, $D=\left\{Q_i,\widehat{\bar{E}}_{Qi}\right\}_{i=1:N}$ which is assumed to come from an unknown nonlinear function $f:\mathbb{R}^d\rightarrow\mathbb{R}$. According to Eqs. (2.38)-(2.44), the GP formula can be written as:

$$\begin{aligned} f\left(Q\right) &\sim \mathcal{GP}\big(m\left(Q\right),k\left(Q,Q'\right)\big)\\ \widehat{\bar{E}}_{Q_{1:N}} &\sim \mathcal{N}\big(m_{1:N},\ K\left(Q_{1:N},Q_{1:N}\right)\big)\\ \widehat{\bar{E}}_{Q_{1:N}} &= f\big(Q_{1:N}\big),\quad m_{1:N}=m\big(Q_{1:N}\big)\end{aligned} \tag{3.15}$$

where K is the covariance matrix that is calculated element-wise by the kernel function $k\left(x,x'\right)$. The kernel function represented in Eq. (2.44) is utilized in this study. Using the initial data (Eq. (3.14)), a nonlinear function $\widehat{\bar{E}}_Q^*=f\left(Q^*\right)$ is trained and the output of this learned function, $\widehat{\bar{E}}_Q^*$, for unobserved weights, Q^*, can be predicted as:

$$\widehat{\bar{E}}_Q^*|\mathcal{D},Q^* \sim \mathcal{N}\left(\mu\left(Q^*|\mathcal{D}\right),\ \sigma\left(Q^*|\mathcal{D}\right)\right) \tag{3.16}$$

where:

$$\mu\left(Q^{*}|\mathcal{D}\right)=m^{*}+k\left(Q^{*},Q_{1:N}\right)K_{1:N}^{-1}\left(\widehat{\bar{E}}_{Q_{1:N}}-m_{1:N}\right) \tag{3.17}$$

and

$$\begin{aligned}\sigma\left(Q^{*}|\mathcal{D}\right)&=k\left(Q^{*},Q^{*}\right)+k\left(Q^{*},Q_{1:N}\right)K_{1:N}^{-1}k\left(Q_{1:N},Q^{*}\right)\\ K_{1:N}&=K\left(Q_{1:N},\ Q_{1:N}\right)\end{aligned} \tag{3.18}$$

After the GP was learned for the BO initial data by means of Eqs. (3.15)-(3.18), using the EI auxiliary acquisition function (Eq. (2.44)), BO technique gives the next possible optimum weights, Q_{opt}^{*}, that minimizes the GP mean, $\mu\left(Q_{opt}^{*}|\mathcal{D}\right)$. As explained previously, EI function calculates the expectation of the amount of improvement for any given input data point (weights in this study) in optimizing the output of the learned GP function ($\widehat{\bar{E}}_{Q}^{*}$) over the current best output value, $\widehat{\bar{E}}_{Q_{min}}$. After the next possible optimum weights, Q_{opt}^{*}, is provided by means of the EI function and the learned GP, the $\widehat{\bar{E}}_{Q_{opt}}$ corresponding to the Q_{opt}^{*} weights will be calculated using the Eqs. (3.10), (3.12), and (3.13).

In the next step, $(Q_{opt}^{*},\widehat{\bar{E}}_{Q_{opt}})$ will be a new datapoint that is added to the set of previous BO datapoints (Eq. (3.14)). These datapoints are updated and the entire process is repeated for a predetermined number of iterations, I. After the final iteration, the updated set of BO datapoints are represented as:

$$\begin{aligned}\text{BO inputs}: BO_{inps}&=\left\{Q_{1}^{initial},\ Q_{2}^{initial},\ldots,\ Q_{r}^{initial},Q_{opt1}^{*},Q_{opt2}^{*},\ldots,Q_{optI}^{*}\ \right\}\\ \text{BO outputs}: BO_{outs}&=\left\{\widehat{\bar{E}}_{Q_{1}},\widehat{\bar{E}}_{Q_{2}},\ldots,\widehat{\bar{E}}_{Q_{r}},\widehat{\bar{E}}_{Q},\widehat{\bar{E}}_{Q_{opt}1},\widehat{\bar{E}}_{Q_{opt}2},\ldots,\widehat{\bar{E}}_{Q_{opt}I}\right\}\end{aligned} \tag{3.19}$$

Finally, the weight set in BO_{inps} corresponding to the minimum value of the error examination criterion, $\widehat{\bar{E}}_{Q_{min}}$, will be considered as the potential optimum weight set, Q_{opt}. The potential weights will be replaced by the weights already used by the MPC controller, and will be added to the weight selection dataset. The proposed weight selection approach is explained in detail in section 3.3. The whole procedure of weight tuning is illustrated in Figure 3.2.

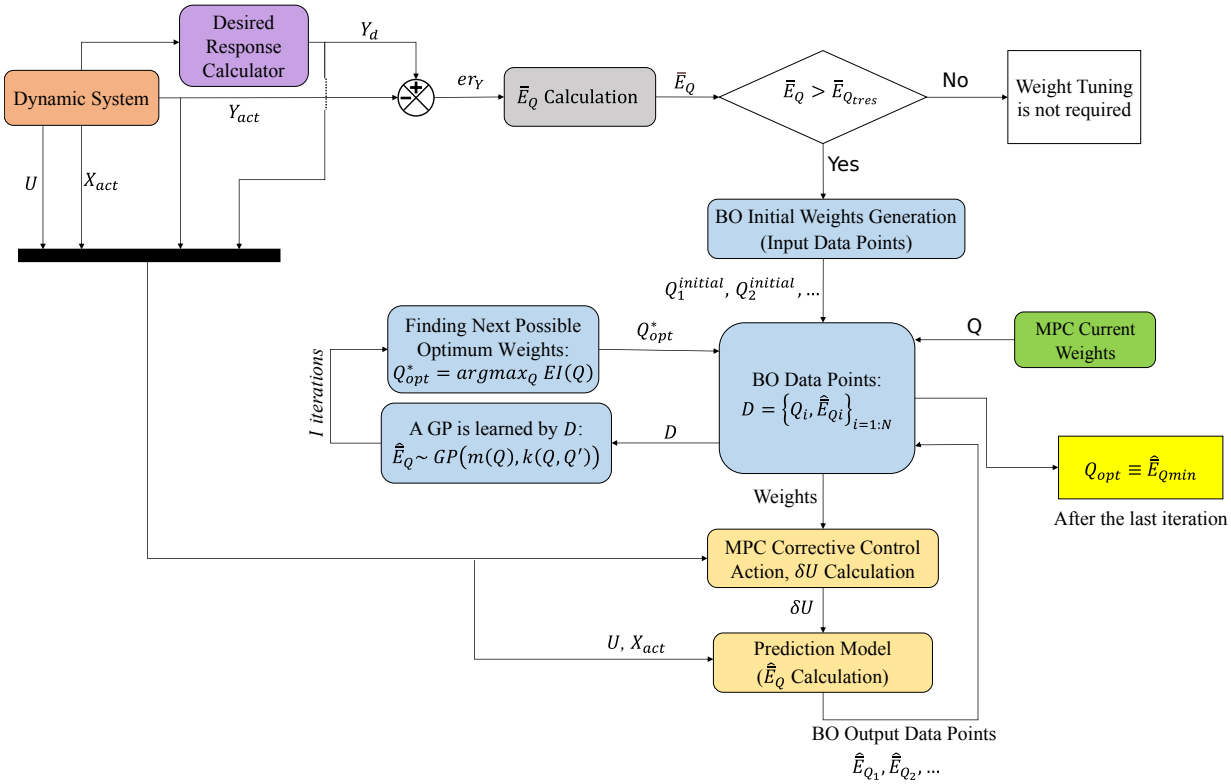

Figure 3.2: Weight tuning procedure for a general dynamic system

3.2 Applying the Weight Tuning Approach to the Vehicle Stability Controller

In this section, the proposed weight tuning approach is applied to the MPC controller designed for vehicle stability control in section 2.9. The main objective of this MPC controller is to maintain the lateral stability of the vehicle by keeping the vehicle's yaw rate and slip angle close to their desired values. Hence, the weight tuning method, can adjust the weights of the MPC controller according to the current state of the vehicle to maintain vehicle stability under a variety of vehicle and/or road conditions. According to Eqs. (2.31) and (2.32), the vector of desired responses and tracking errors can be expressed as:

$$
\begin{aligned}
Y_d &= [r_d, \beta_d\ ]^T \\
er_Y &= [er_r, er_\beta]^T = [r - r_d,\ \beta - 0] = Y - Y_d
\end{aligned}
\tag{3.20}
$$

The vector of driver inputs, U, and corrective control actions, δU, can be written as:

$$
\begin{aligned}
U &= [Q_f, M_{DY}]^T \\
\delta U &= [\delta Q_f \quad \delta M_{DY}]^T = \mathcal{MPC}\left(X, U, Y_d, Q, R, T\right) \\
Q &= diag(W_r, W_\beta)
\end{aligned}
\tag{3.21}
$$

The step time of t_s is considered for the MPC controller. The weight matrix of the tracking errors, Q, is already selected by the MPC controller at the previous time step, $t - ts$, to find the optimal torques (control actions) to achieve the desired vehicle yaw rate and side slip angle. The obtained optimal torques are acted on wheels during t_s. Then, the actual values of yaw rate and side slip angle are measured in the next time step, t, to calculate the tracking errors and to evaluate the control system performance. The actual tracking errors at each time step is calculated as:

$$
\begin{aligned}
&e_r^t = \left|r^t - r_{des}^t\right|, \quad e_r^{t-t_s} = \left|r^{t-t_s} - r_{des}^{t-t_s}\right|, \quad e_r^{t-2t_s} = \left|r^{t-2t_s} - r_{des}^{t-2t_s}\right|, ... \\
&e_\beta^t = \left|\beta^t - 0\right|, \qquad e_\beta^{t-t_s} = \left|\beta^{t-t_s} - 0\right|, \qquad e_\beta^{t-2t_s} = \left|\beta^{t-2t_s} - 0\right|, ...
\end{aligned}
\tag{3.22}
$$

In order to determine whether the selected weights in the controller were appropriate for the driver and/or road condition at l number of previous time steps, the error examination criterion is defined as:

$$
\bar{E}_Q = E\left(er_Y^{t:t-lt_s}\right) = \frac{\gamma_r \sum_{i=0}^{l-1} \left|e_r^{t-it_s}\right| + \gamma_\beta \sum_{i=0}^{l-1} \left|e_\beta^{t-it_s}\right|}{l\left(\gamma_r e_r^{t-lt_s} + \gamma_\beta e_\beta^{t-lts}\right)}
\tag{3.23}
$$

which calculates the average tracking error rate over a fixed time interval window by dividing a weighted sum of tracking errors during the window by the tracking error at

the start of the window. γ_r is the weight of vehicle's yaw rate tracking errors, and γ_β represents the weight of the vehicle's side slip angle tracking error in the defined criterion formulation. $\bar{E}_Q$ is calculated by the actual values of tracking error in Eq. (3.23). It shows how successful the MPC controller was to push actual values of yaw rate and side slip angle to the desired values during previous time steps. High values of $\bar{E}_Q$ indicate weak performance of the controller and vice versa. $\bar{E}_Q$ can be reduced by making some adjustments on the weight values. Therefore, a maximum acceptable value of $\bar{E}_{Qmax}$ is predefined and considered as a threshold for weight tuning (as discussed previously). If $\bar{E}_Q$ is less than this threshold, then the selected weight set is acceptable. Otherwise, the selected yaw rate and side slip angle weights should be tuned.

In order to tune the MPC controller weights, the prediction model of the vehicle (explained in section 2.5) is used to predict the tracking errors of yaw rate and side slip angle if other possible weights were selected. Predicted tracking errors, $\widehat{er}_Y$, and predicted error examination criterion, $\widehat{\bar{E}}_{Q'}$, are defined as:

$$\widehat{er}_Y \triangleq [\widehat{er}_r, \widehat{er}_\beta\ ]^T = \left[\hat{r} - r_d,\ \hat{\beta} - \beta_d\right]^T = \hat{Y} - Y_d \tag{3.24}$$

$$\widehat{\bar{E}}_{Q'} \triangleq E\left(\widehat{er}_Y^{t:t-lt_s}\right) = \frac{\gamma_r \sum_{i=0}^{l-1} \left|\hat{e}_r^{t-it_s}\right| + \gamma_\beta \sum_{i=0}^{l-1} \left|\hat{e}_\beta^{t-it_s}\right|}{l\left(\gamma_r e_r^{t-lt_s} + \gamma_\beta e_\beta^{t-lts}\right)} \tag{3.25}$$

The $\widehat{er}_Y^t$ is obtained by calculating the difference between the predicted outputs in step time t and the desired ones. $\widehat{\bar{E}}_{Q'}$ is calculated as a function of predicted tracking errors during the l number of previous time steps, $\widehat{er}_Y^{t:t-lt_s}$, if other weights, Q', were used. These predicted values are calculated and used for weight tuning through the following steps:

First, 4 initial set of new weights and corresponding Q matrices are generated as:

$$Q_i^{initial} = \begin{bmatrix} w_{ri} & 0 \\ 0 & w_{\beta i} \end{bmatrix}, \qquad i = 1, 2, 3, 4 \tag{3.26}$$

For BO, the initial new weights in Eq. (3.26) as well as the already selected weights by

the MPC controller (presented by Q) are considered as the BO input datapoints:

$$\text{BO initial points} : \left\{Q_1^{initial},\ Q_2^{initial},\ Q_3^{initial},\ Q_4^{initial}, Q\right\} \tag{3.27}$$

An objective function is defined for each weight matrix, Q_i, in the MPC controller, and, by using the vehicle prediction model, the MPC controller's optimal control actions, δM_j, associated with Q_j are obtained as:

$$\delta M_j^t = \mathcal{MPC}\left(X^{t-ts},\ U^{t-ts}, Y^{t-ts},\ Y_d^{t-ts}, Q_j, R, T, W_s\right) \tag{3.28}$$

Therefore, optimal torque control actions, δM_j, corresponding to each of the BO initial points are presented as:

$$Q_j \ \rightarrow \ \delta M_j^t \tag{3.29}$$

By using the vehicle prediction model presented in Eqs. (2.14) and (2.16), the predicted outputs, $\hat{Y}_j^t$, and the corresponding predicted tracking errors, $\widehat{er}_{Y_j}$, under the wheel torques, δM_j^t, are calculated as:

$$\hat{Y}_j^t = H\left(X^{t-ts},\ \delta^{t-ts}, M^{t-ts} + \delta M_j^t\right) \rightarrow \widehat{er}_{Y_j}^t = \hat{Y}_j^t - Y_d = \begin{bmatrix} \hat{r}_j^t - r_d \\ \hat{\beta}_j^t - 0 \end{bmatrix} \tag{3.30}$$

As a result, the Eq. (3.29) can be completed as:

$$Q_j \ \rightarrow \ \delta M_j^{t-lt_s:t} \ \rightarrow \ \hat{Y}_j^{t-lt_s:t} \ \rightarrow \ \widehat{er}_{Y_j}^{t-lt_s:t} \ \rightarrow \ \hat{\bar{E}}_{Q_j} \tag{3.31}$$

The Q_j points are considered as the BO initial input points, and the $\hat{\bar{E}}_{Q_j}$ points form the BO problem's initial output points:

$$\begin{aligned} &\text{BO initial inputs} : BO_{inps} = \left\{Q_1^{initial},\ Q_2^{initial}, Q_3^{initial},\ Q_4^{initial}, Q\right\} \\ &\text{BO initial outputs} : BO_{outs} = \left\{\hat{\bar{E}}_{Q_1}, \hat{\bar{E}}_{Q_2}, \hat{\bar{E}}_{Q_3}, \hat{\bar{E}}_{Q_4}, \hat{\bar{E}}_{Q}\right\} \end{aligned} \tag{3.32}$$

The objective of this BO problem is to determine the optimum yaw rate and side slip

angle weights, $\big(Q_{opt} = \text{diag}\big(W_{r_{opt}}, W_{\beta_{opt}}\big)\big)$, that minimizes the predicted error examination criterion, $\widehat{\bar{E}}_{Q_{min}}$, which minimizes the MPC controller's predicted tracking errors, $\widehat{er}_Y$. To formulate the BO problem, a GP must be learned for the BO initial data presented in Eq. (3.32). This GP considers a multivariate gaussian distribution for the datapoints $D = \Big\{Q_j, \widehat{\bar{E}}_{Qj}\Big\}_{j=1:N}$.

The GP formulation is similar to Eqs. (3.15) - (3.18). After the GP is learned for the BO initial data (Eq. (3.32)), BO technique gives the next possible optimum weights, Q^*_{opt}, by using the EI auxiliary acquisition function (Eq. (2.45)). The EI function calculates the expectation of the amount of improvement for hundreds of input datapoint (weights here) in optimizing the output of the learned GP function ($\widehat{\bar{E}}^*_Q$) over the current best output value, $\widehat{\bar{E}}_{Q_{min}}$. The weight set associated with the highest amount of EI function will be considered as the next possible optimum weights, Q^*_{opt}. Then, the value of $\widehat{\bar{E}}_{Q_{opt}}$ corresponding to the elements of the weight matrix, Q^*_{opt}, will be calculated using the prediction model of the vehicle. Hence, $(Q^*_{opt}, \widehat{\bar{E}}_{Q_{opt}})$ will be a new datapoint that is added to the previous BO dataset (Eq. (3.32)). The BO datapoints are updated and the entire process is repeated for a predetermined number of iterations, I. After the final iteration, the BO datapoints are obtained according to what follows:

$$\begin{aligned} \text{BO inputs} : BO_{inps} &= \big\{Q_1^{initial},\ Q_2^{initial} Q_3^{initial},\ Q_4^{initial}, Q^*_{opt1}, Q^*_{opt2}, \ldots, Q^*_{optI}\ \big\} \\ \text{BO outputs} : BO_{outs} &= \Big\{\widehat{\bar{E}}_{Q_1}, \widehat{\bar{E}}_{Q_2}, \widehat{\bar{E}}_{Q_3}, \widehat{\bar{E}}_{Q_4}, \widehat{\bar{E}}_{Q}, \widehat{\bar{E}}_{Q_{opt}1}, \widehat{\bar{E}}_{Q_{opt}2}, \ldots, \widehat{\bar{E}}_{Q_{opt}I}\Big\} \end{aligned} \tag{3.33}$$

Finally, the weight set in BO_{inps} corresponding to the minimum value of the error reduction criterion, $\widehat{\bar{E}}_{Q_{min}}$, will be considered as the potential optimum weight set, Q_{opt}. The potential weights will be replaced by the weights that have already been selected in the MPC controller, and will be added to the weight selection dataset. Figure 3.3 illustrates the weight tuning steps for the vehicle's MPC controller.

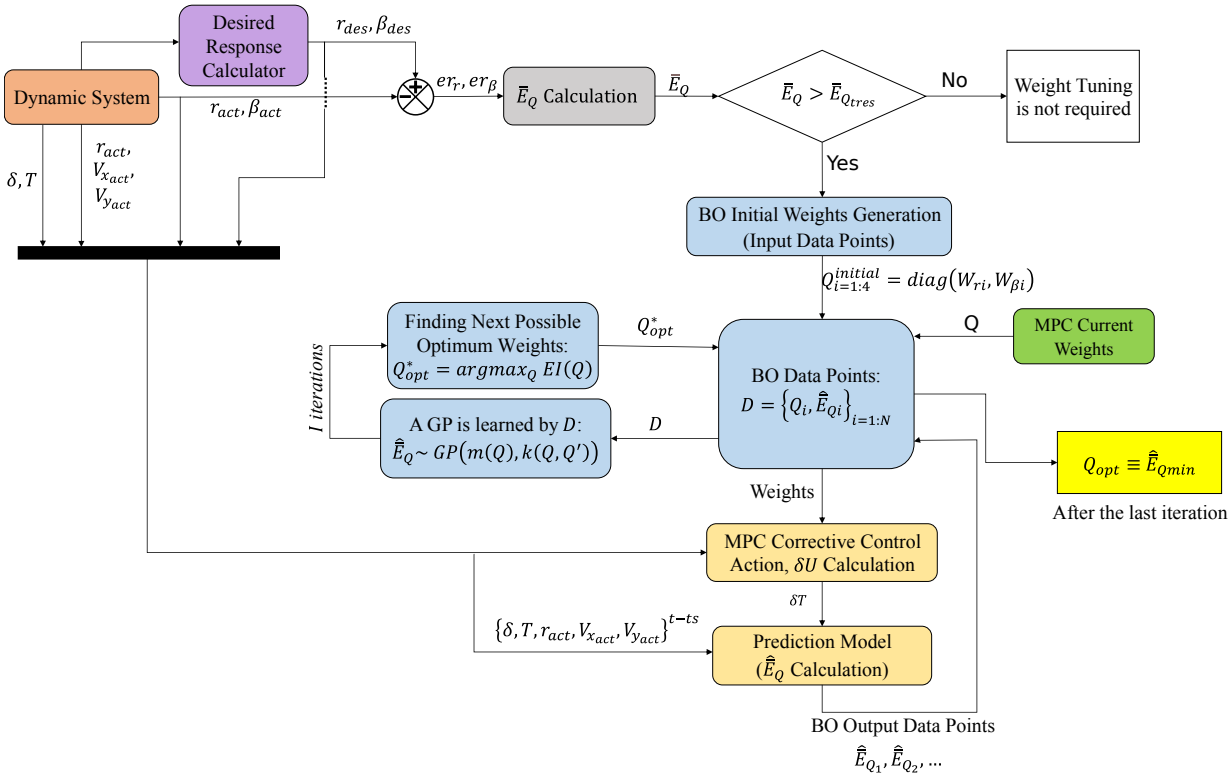

Figure 3.3: Weight tuning procedure for the vehicle's MPC controller

3.3 Learning-Based Weight Selection

The proposed real-time learning-based weight selection technique is presented in this section. Figure 3.4 shows a general representation of the weight selection module. Firstly, it is explained that a dataset is considered to store the optimal weights, which are obtained through weight tuning. A description of the feature space and outputs of this dataset is then provided. Following this, the use of GPR to train the dataset is presented. Finally, selecting the MPC controller weights using the dataset learned by GPR method in a real-time manner is explained.

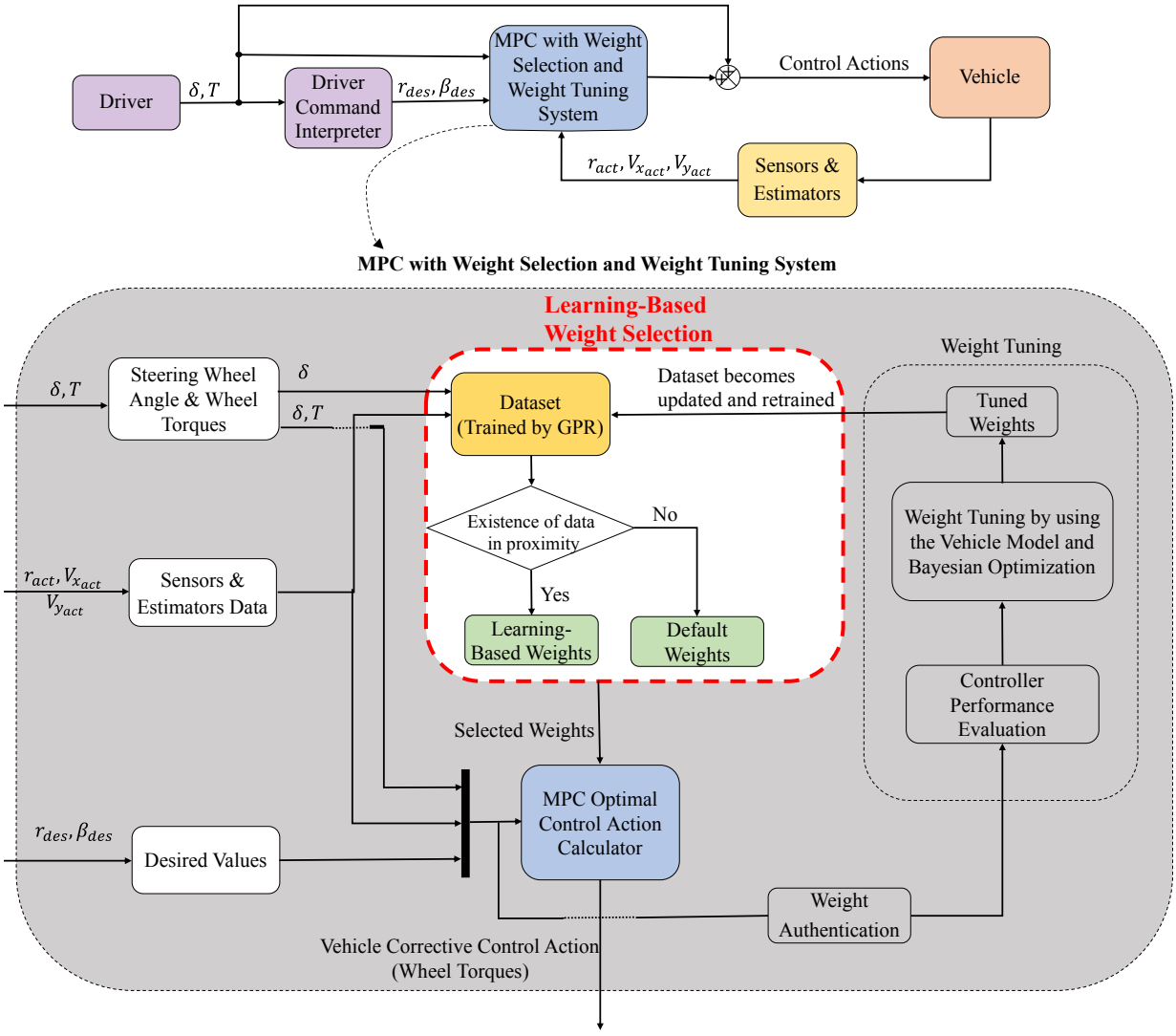

Figure 3.4: Real-time learning-based weight selection module

3.3.1 Weight Selection Dataset

A dataset is considered to store tuned weights along with vehicle and driver data. It provides the MPC control system with access to the optimum weights obtained through weight tuning. This dataset is represented by D_w. D_w is initially empty, however, as the vehicle undergoes a variety of tests, the MPC weights become tuned, and fill the D_w. The

feature space or input of the dataset is denoted by X_D and is defined as:

$$\begin{aligned} X_D &= [x_{D1}, x_{D2}, \ldots, x_{Dn}]^T \\ x_{Di} &= [r_i \;\; V_{xi} \; V_{yi} \; \delta_i]^T \qquad\qquad i = 1:n \end{aligned} \tag{3.34}$$

The dataset inputs include the yaw rate, r, longitudinal velocity, V_x, lateral velocity, V_y, and the steering wheel angle, δ, of the vehicle. The outputs of the dataset, denoted by Y_D, are the yaw rate and the side slip angle tuned weights corresponding to X_D:

$$\begin{aligned} Y_D &= [y_{D1}, y_{D2}, \ldots, y_{Dn}]^T \\ y_{Di} &= [w_{ri}, w_{\beta i}]^T \end{aligned} \tag{3.35}$$

Therefore, after the weight tuning, the data of the X_D matrix as well as the tuned weights become added as a new sample point in the D_w dataset.

3.3.2 Dataset Training Using Gaussian Process Regression

In order to be able to select the appropriate controller weights, two GPs (one GP for obtaining w_r and another one for w_β) are learned by using the vehicle and weight data in the D_w dataset. As previously mentioned, this should not be confused with the other GP considered for the BO during the weight tuning. This GP considers a multivariate gaussian distribution based on the data in the D_w dataset:

$$\begin{aligned} D_{w_r} &= \{x_{Di}, w_{ri}\}_{i=1:n} \\ D_{w_\beta} &= \{x_{Di}, w_{\beta i}\}_{i=1:n} \end{aligned} \tag{3.36}$$

where D_{w_r} and D_{w_β} are two different datasets considered for formulating the GP to obtain yaw rate and side slip angle weights, respectively. According to Eqs. (2.38) - (2.44), the GP formulation can be expressed as:

$$
\begin{aligned}
&f_{wr}\left(x_D\right) \sim \mathcal{GP}\left(m_{wr}\left(x_D\right), k\left(x_D, x'_D\right)\right), \qquad f_{w\beta}\left(x_D\right) \sim \mathcal{GP}\left(m_{w\beta}\left(x_D\right), k\left(x_D, x'_D\right)\right), \\
&w_{r1:n} \sim \mathcal{N}\left(m_{wr1:n}, K\left(x_{D1:n}, x_{D1:n}\right)\right), \qquad w_{\beta 1:n} \sim \mathcal{N}\left(m_{w\beta 1:n}, K\left(x_{D1:n}, x_{D1:n}\right)\right), \\
&w_{r1:n} = f_{wr}\left(x_{D1:n}\right), \qquad w_{\beta 1:n} = f_{w\beta}\left(x_{D1:n}\right).
\end{aligned}
\tag{3.37}
$$

where K is the covariance matrix that is calculated elementwise by the kernel function $k\left(x, x'\right)$. The kernel function represented in Eq. (2.44) is again used here. Using the initial data (Eq. (3.36)) , the nonlinear functions $w_r^* = f\left(x_D^*\right)$ and $w_\beta^* = f\left(x_D^*\right)$ are trained and the output of these learned functions for new input data, x_D^*, can be predicted as:

$$
\begin{aligned}
w_r^* | \mathcal{D}_{wr}, x_D^* &\sim \mathcal{N}\left(\mu\left(x_D^* | \mathcal{D}_{wr}\right), \ \sigma\left(x_D^* | \mathcal{D}_{wr}\right)\right) \\
w_\beta^* | \mathcal{D}_{w\beta}, x_D^* &\sim \mathcal{N}\left(\mu\left(x_D^* | \mathcal{D}_{w\beta}\right), \ \sigma\left(x_D^* | \mathcal{D}_{w\beta}\right)\right)
\end{aligned}
\tag{3.38}
$$

where:

$$
\begin{aligned}
\mu\left(x_D^* | \mathcal{D}_{wr}\right) &= m_{wr}^* + k\left(x_D^*, x_{D1:n}\right) K_{1:n}^{-1}\left(w_{r1:n} - m_{wr1:n}\right) \\
\mu\left(x_D^* | \mathcal{D}_{w\beta}\right) &= m_{w\beta}^* + k\left(x_D^*, x_{D1:n}\right) K_{1:n}^{-1}\left(w_{\beta 1:n} - m_{w\beta 1:n}\right)
\end{aligned}
\tag{3.39}
$$

and

$$
\begin{aligned}
\sigma\left(x_D^* | \mathcal{D}_{wr}\right) &= k\left(x_D^*, x_D^*\right) + k\left(x_D^*, x_{D1:n}\right) K_{1:n}^{-1} k\left(x_{D1:n}, x_D^*\right) \\
\sigma\left(x_D^* | \mathcal{D}_{w\beta}\right) &= k\left(x_D^*, x_D^*\right) + k\left(x_D^*, x_{D1:n}\right) K_{1:n}^{-1} k\left(x_{D1:n}, x_D^*\right) \\
K_{1:n} &= K\left(x_{D1:n}, \ x_{D1:n}\right)
\end{aligned}
\tag{3.40}
$$

Hence, the weights associated with any new point is predicted by using the mean function presented in Eq. (3.39), and Eq. (3.40) calculates the standard deviation of the predicted weights with respect to the training data.

3.3.3 Real-Time Weight Selection

To select the weights for the vehicle's MPC controller, first, yaw rate, longitudinal velocity, lateral velocity, and the steering wheel angle of the vehicle in the current time (x_D^t) is extracted by using the vehicle sensors or estimators. A set of default weights, w_r^{def} and w_β^{def} are predefined in the MPC controller. A neighbourhood variable, ϵ_x, is defined to find all x_D input data in D_w close to the current vehicle's data, x_D^t:

$$\epsilon_X = [\epsilon_r \ \epsilon_{V_x} \ \epsilon_{V_y} \ \epsilon_\delta]^T \tag{3.41}$$

Hence, x_{Di} is assumed to be in proximity to x_D^t if:

$$||x_D^i|| \leq ||x_D^t|| + \epsilon_X \tag{3.42}$$

As illustrated in Algorithm 1 and Figure 3.5, the MPC controller utilizes weights predicted by the GPR if at least one x_{Di} exists in D_w in proximity to x_D^t, otherwise; default weights are chosen.

Algorithm 1 Real-time Weight Selection Algorithm

if $\exists x_{Di} \in \left[x_D^t - \epsilon_X, \ x_D^t + \epsilon_X\right]$ **then**
  $w_r^t \leftarrow \mu\left(x_D^t | \mathcal{D}_{wr}\right)$
  $w_\beta^t \leftarrow \mu\left(x_D^t | \mathcal{D}_{w\beta}\right)$
else
  $w_r^t \leftarrow w_r^{def}$
  $w_\beta^t \leftarrow w_\beta^{def}$
end if

After selecting the weights, the MPC controller calculates the optimal control actions, wheel torques, which are then acted on the vehicle wheels. Clearly, the weight selection is made in real-time. After the vehicle maneuver is finished, the weight tuning is executed according to the explanations of sections 3.1 and 3.2, based on the history of data. Therefore, a MPC controller with real-time weight selection is designed, and the used weights are evaluated after each maneuver and tuned accordingly. As a result, the control system generates the optimal control actions by solving the MPC controller's objective function

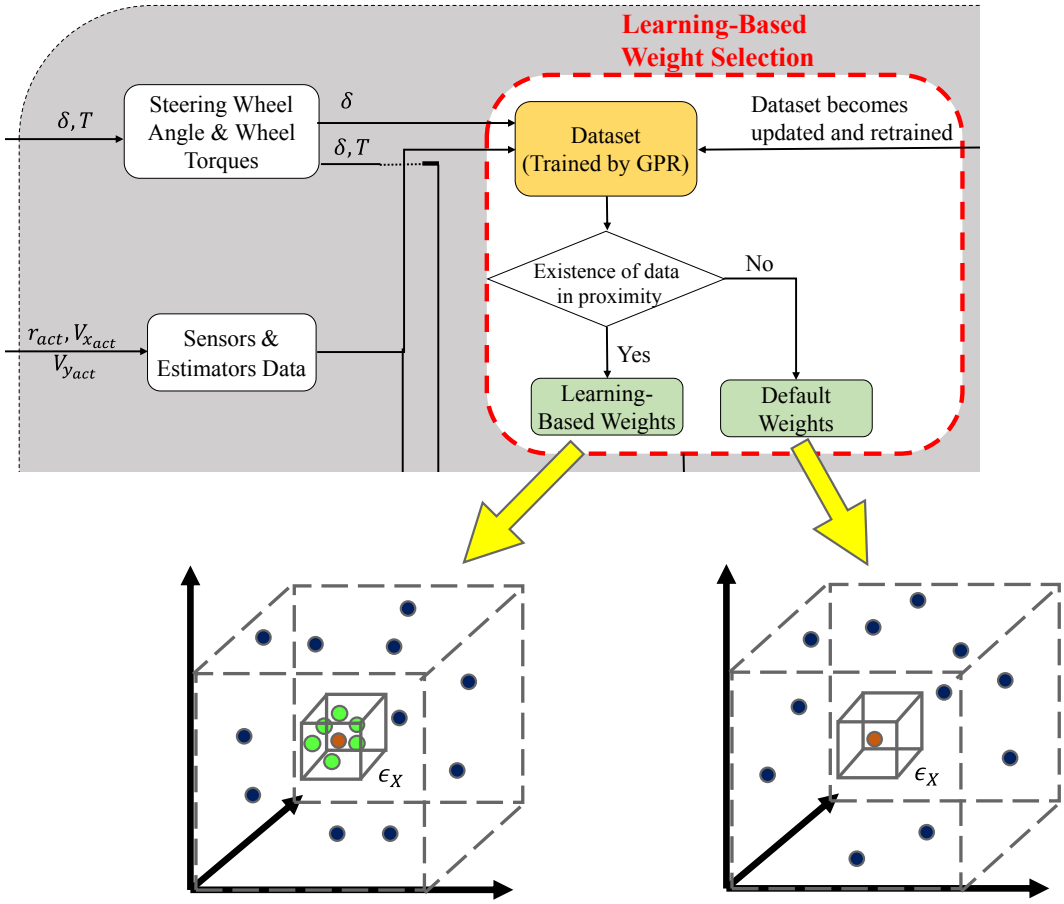

Figure 3.5: Selection of weights based on the points in the neighborhood, ϵ_X.

such that even its weight values are optimized.

3.4 Weight Tuning Simulations

To evaluate the proposed weight tuning and real-time learning-based weight selection method, the designed MPC controller along with the weight tuning and weight selection modules are implemented using MATLAB/Simulink [122] software. A co-simulation with CarSim [123] is used to apply the controller to the vehicle running in CarSim. A CarSim model of an sport utility vehicle (SUV) which accurately represents the vehicle's behavior

with detailed specifications outlined in Table 3.1 is employed for the simulations. The aforementioned specifications have been obtained from the data source within the CarSim software. The overall MATLAB/Simulink and CarSim co-simulations diagram is shown in Figure 3.6.

Table 3.1: Parameters of the vehicle in CarSim.

Parameter	**Unit**	**Value**	**Description**
m	[kg]	2257	Vehicle mass
L_{wb}	[m]	3.14	Wheelbase
l_f	[m]	1.33	CG distance to front axle
l_r	[m]	1.81	CG distance to rear axle
H_{CG}	[m]	0.78	CG hight
R_e	[m]	0.368	Effective radius of the tires
l_s	[m]	1.725	Front and rear track width
I_z	[kg.m^2]	3525	Vehicle yaw moment of inertia
$C_{\alpha f}$	[N/rad]	152343	Front tires cornering stiffness
$C_{\alpha r}$	[N/rad]	121943	Rear tires cornering stiffness

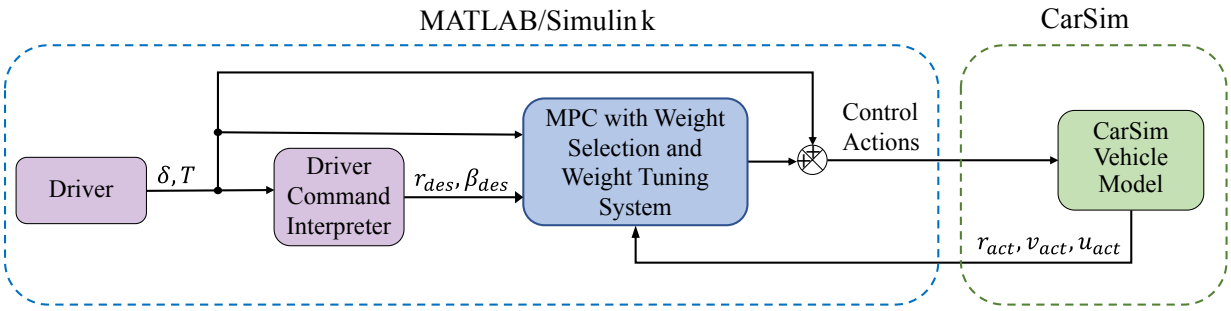

Figure 3.6: Overall MATLAB/Simulink and CarSim co-simulations diagram

According to Figure 3.6, Yaw rate, longitudinal velocity, and lateral velocity received from CarSim are transferred to the MPC controller in MATLAB. During each simulation, the desired values of yaw rate and sideslip angle are determined, and the suitable weights

corresponding to current state of the vehicle are obtained by the weight selection module in real-time (as discussed in the previous sections). Then, the optimal control actions are calculated and applied to the vehicle. The parameters used for the MPC controller, weight selection, and weight tuning are listed in Table 3.2.

Table 3.2: Parameters of the MPC controller, weight tuning, and weight selection modules.

Parameter	**Value**	**Description**
t_s	0.02	Controller sampling time(s)
N_p	6	Prediction horizon size
N_c	6	Control horizon size
T	$5 \times 10^{-5} \times I_{2\times 2}$	MPC weight matrix of proximity to previous solution
R	$3 \times 10^{-6} \times I_{2\times 2}$	MPC weight matrix of control action
ϵ_X	[2 5 0.1 1]	Neighborhood distances = $[\epsilon_r\ \epsilon_u\ \epsilon_v\ \epsilon_\delta]$
w_r^{def}	50	MPC yaw rate tracking error default weight
w_β^{def}	10	MPC sideslip angle tracking error default weight
w_r^t	-	Selected yaw rate weight at time = $t(s)$
w_β^t	-	Selected side slip angle weight at $t(s)$
$[w_{r1}, w_{r2}, w_{r3}, w_{r3}]^t$	$[w_r^t, 0.2w_r^t, w_r^t, 5w_r^t]$	BO initial yaw rate weights at $t(s)$
$[w_{\beta 1}, w_{\beta 2}, w_{\beta 3}, w_{\beta 4}]^t$	$[5w_\beta^t, w_\beta^t, 0.2w_\beta^t, w_\beta^t]$	BO initial sideslip angle weights at $t(s)$
I	5	Number of BO iterations
$\bar{E}_{Q_{th}}$	0.8	Maximum acceptable value of $\bar{E}_Q$
l	5	Number of preceding time steps in $\bar{E}_Q$

Upon completion of each simulation run, the history of the vehicle and controller data is used in the weight tuning process to adjust weights off-line. The weight selection dataset, D_w, is updated by the tuned weights, and then it is retrained to be used for the next simulation run. After a number of simulations, the results are extracted and discussed. The CarSim vehicle model is subjected to double lane change (DLC) maneuvers. Figure

3.7 shows the steering wheel angle input of the driver. High rates of steering wheel angle changes are taken into account to evaluate the efficacy of the proposed weight tuning method in scenarios where the vehicle's stability may be compromised due to improper selection of MPC weights.

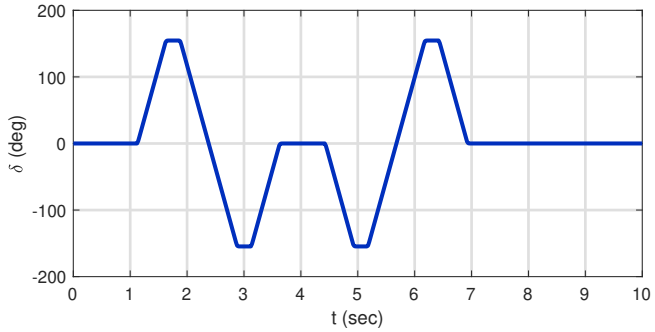

Figure 3.7: Steering wheel angle input for the DLC maneuver.

Two main series of simulations are conducted and discussed in this section. During the first series of simulations, the road condition is dry with $\mu = 0.8$, while in the second series, the road is slightly slippery with $\mu = 0.6$. The initial longitudinal speed of the vehicle is $u_0 = 70$ kph. Three successive simulation runs are performed under each road condition. For the first run, default weights w_r^{def}, w_β^{def} are set in the objective function of the MPC controller.

The results of the vehicle's yaw rate, sideslip angle, and variations in the MPC weights w_r, w_β on dry road are illustrated and compared in Figure 3.8. Yaw rate and sideslip angle tracking errors are shown in Figure 3.9. The yaw rate and sideslip angle responses are improved after the first execution of the MPC weight tuning. The root mean square error (RMSE) of the yaw rate and side slip tracking are respectively decreased about 26% and 16% when fixed default weights are replaced with the tuned weights selected in real-time. Upon repeating the simulation runs, the tracking errors are decreased from the previous repetition, indicating that the proposed method is capable of continuously improving the performance of the controller to maintain the vehicle's lateral stability. Before weight tuning, large peak values are noticed in the tracking error diagrams. These

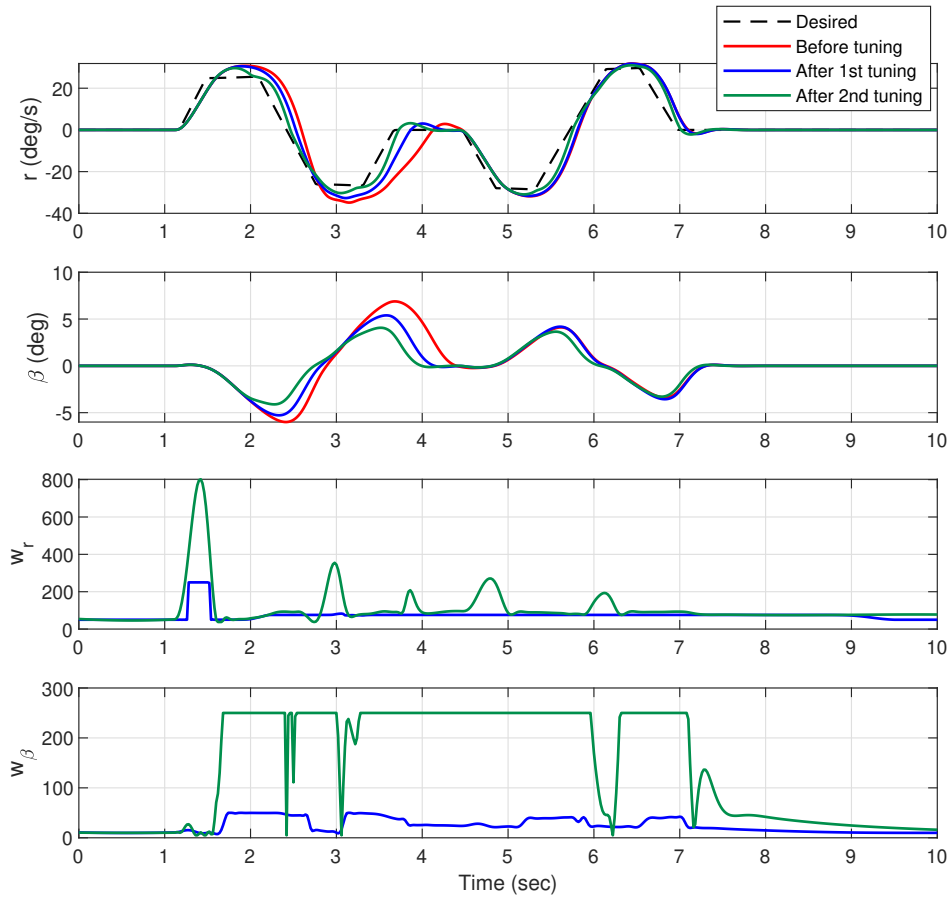

Figure 3.8: Responses of the vehicle in terms of yaw rate and sideslip angle, and the weights w_r, w_β selected in real-time on a dry road.

points correspond to the maximum magnitudes of the steering wheel angle. However, most of these peak values are considerably decreased after weight tuning.

The bottom graphs in Figure 3.8 show the real-time selected MPC weights. The DLC maneuver involves significant changes to the vehicle's steering wheel angle. Therefore, significant increases and/or decreases can be noticed in the weight values to maintain the vehicle's stability. As previously mentioned, the weights are tuned based on the data received from the vehicle and driver, and predictions of the vehicle model. The vehicle model has predicted a decrease in the tracking error based on the value of $\hat{\bar{E}}_{Q\prime}$ (calculated

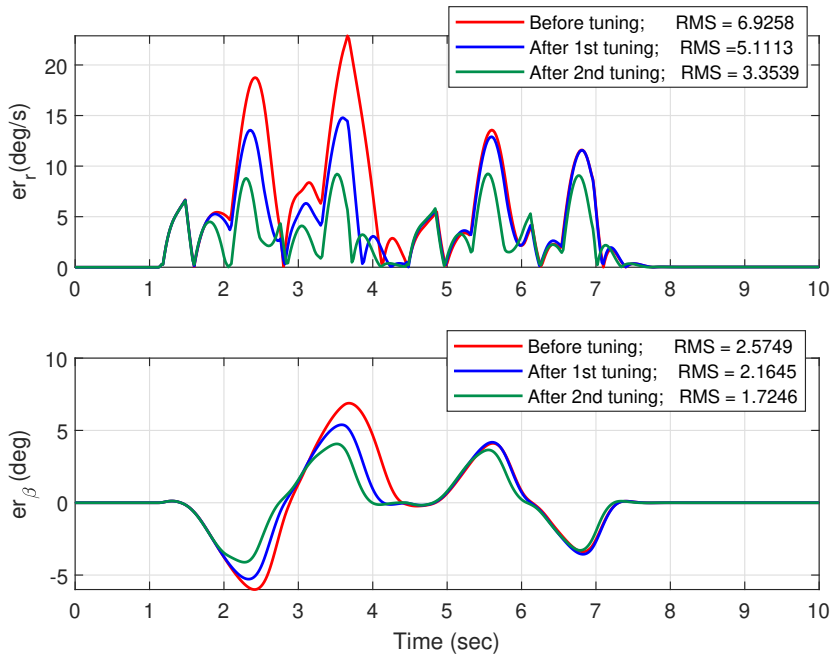

Figure 3.9: Tracking errors of the yaw rate and sideslip angle of the vehicle on a dry road.

by Eq. (3.25)) at each time step that weight values are different from those in the previous simulation. In most of these time steps, the performance of the controller is improved; however, in a few time steps, the change in the weight values has no considerable impact on the yaw rate or sideslip angle response of the vehicle, indicating the predictions made during the weight tuning process were not very accurate at those vehicle states.

Figure 3.10 demonstrates the front wheels torques generated by the MPC controller. As previously mentioned, front wheels torque vectoring is considered as the control action. The vehicle running in the CarSim is drived off-throttle, which means the total requested torque by the driver is equal to zero. Therefore, the generated wheel torques are only intended to control the vehicle lateral stability. The MPC controller's objective function generates small wheel torques when fixed default weights are set, as shown in Figure 3.10. This accounts for the large average sideslip angle and yaw rate tracking error. After tuning the MPC controller weights, wheel torques are increased in some time steps which improves the tracking performance of the controller. Again, in some time steps, the change in the

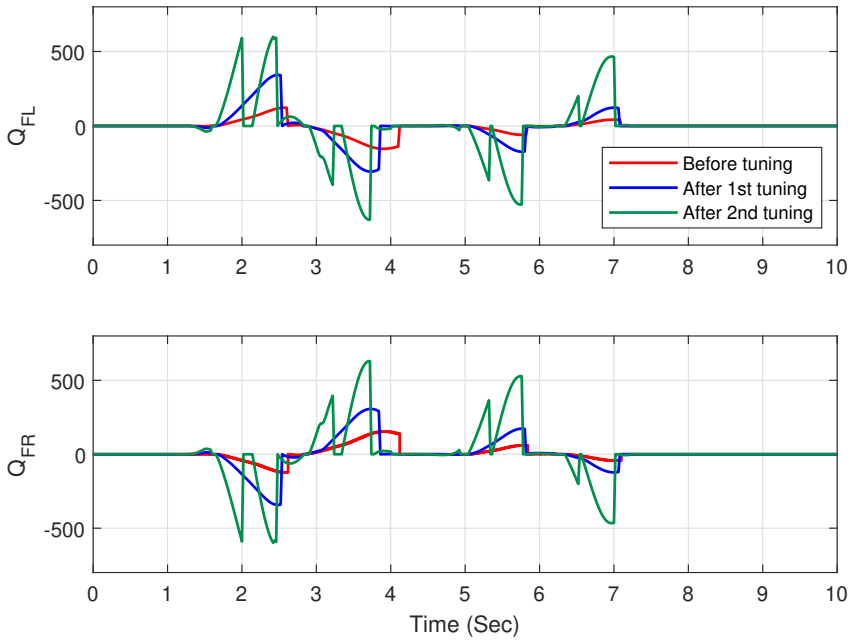

Figure 3.10: Front left and front right wheel torques generated by the MPC controller on a dry road.

wheel torque does not considerably affect the response of the vehicle system. Thus, it is necessary to develop a method to identify these vehicle states, and to avoid storing their modified weights in the weight selection dataset, D_w.

The second series of simulations are conducted for a similar DLC maneuver on a slightly slippery road with road friction of $\mu = 0.6$. Figure 3.11 illustrates the vehicle's yaw rate, sideslip angle, and variations in the MPC weights w_r, w_β, and Figure 3.12 demonstrates the MPC controller's tracking error on a slippery road. The vehicle becomes unstable before weight tuning since the sideslip angle exceeds 10 degrees and the yaw rate tracking error becomes larger than 25 deg/s. Thus, setting the default weights in the MPC controller's objective function cannot maintain the stability of the vehicle under this DLC maneuver. This is due to the fact that the road is slippery with a lower road friction coefficient than in the previous runs, and by default weights, the MPC controller does not generate sufficient torque at the right time.

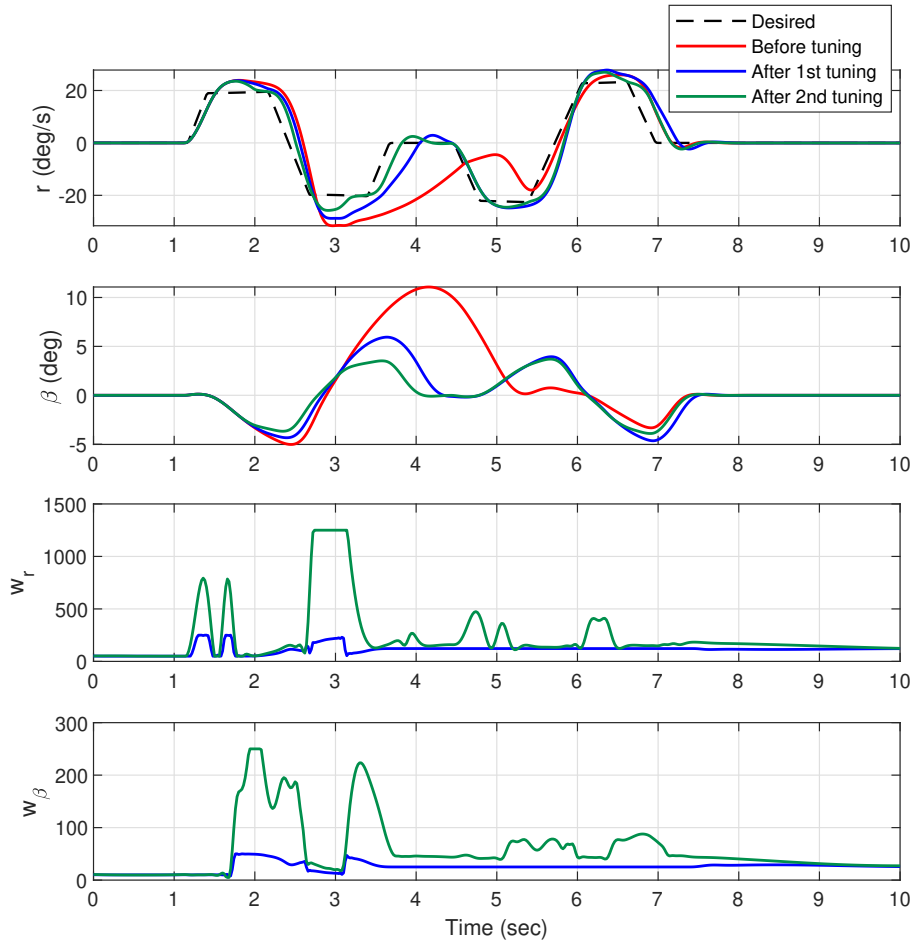

Figure 3.11: Responses of the vehicle in terms of yaw rate and sideslip angle, and the weights w_r, w_β selected in real-time on a slippery road.

After tuning the weights, the DLC maneuver is repeated, and tracking errors are decreased. As depicted in Figure 3.12, after 2nd tuning, the yaw rate and sideslip angle tracking errors are respectively improved about 55% and 54% when fixed default weights are replaced with the tuned weights selected in real-time. The results clearly indicate the effectiveness of the proposed method in improving the performance of the MPC controller to control the vehicle's lateral stability even in slippery road conditions. The selected MPC tracking error weights w_r, w_β are altered in real-time as depicted in the bottom graphs of

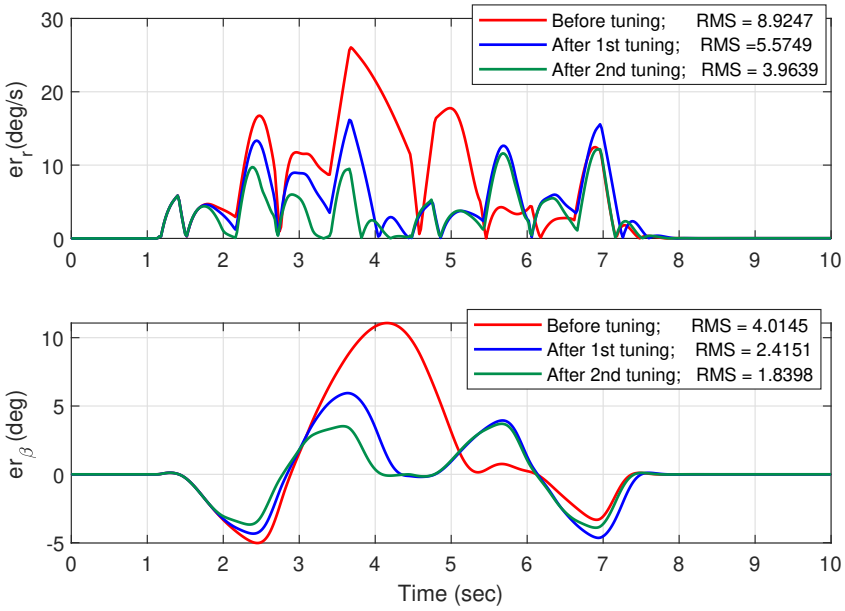

Figure 3.12: Tracking errors of the yaw rate and sideslip angle of the vehicle on a slippery road.

Figure 3.11. At most time steps during the last simulation run (after second tuning), the weights tend to remain the same or very close to those selected during the previous run (after first tuning). The reason for this is that during the weight tuning, the vehicle prediction model did not predicted a better control performance if weights changed. In contrast, at some other time steps the weights are greatly increased to ensure that the controller generates enough torque to maintain stability of the vehicle (Figure 3.13).

Some fluctuations can be noticed in the wheel torque results after final tuning as depicted in Figure 3.13. It can be concluded that the real-time variation of the MPC weights has altered the generated wheel torques such that a sufficient amount of torque is applied to each wheel at the appropriate time. Nevertheless, there are some vehicle states in which even a significant torque does not have an effect on the yaw rate or sideslip angle. This issue is addressed through a proposed weight authentication (WA) technique provided in the following section.

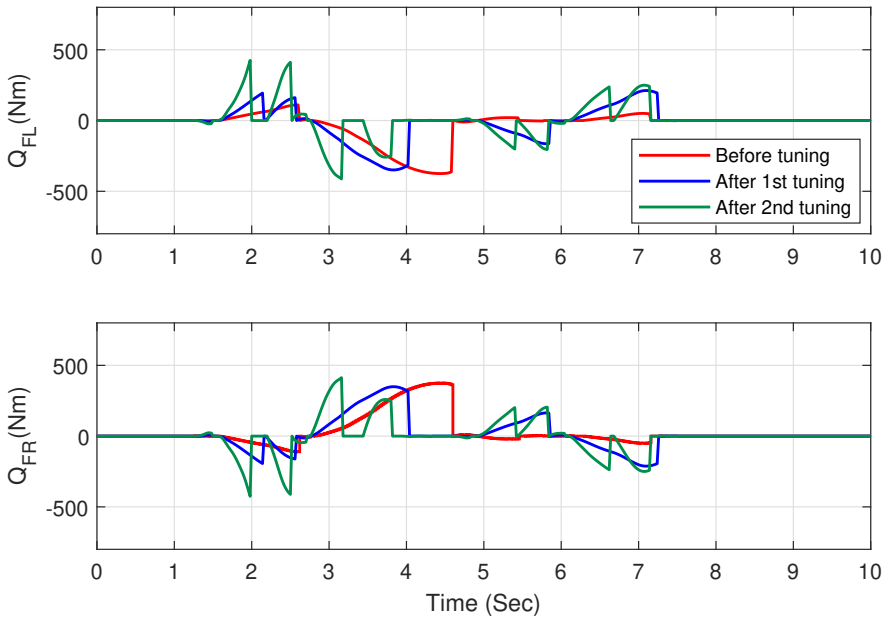

Figure 3.13: Front left and front right wheel torques generated by the MPC controller on a slippery road.

3.5 MPC Weight Authentication

A description of the MPC controller with weight tuning and real-time weight selection was provided in the previous sections. As discussed in section 3.1, weight tuning is performed based on the predictions made by the prediction model of the vehicle. If the prediction model predicts a better response for another set of weights, then, the new weight set will be substituted by the previously selected weights. However, no model is perfect in terms of predicting the future states of the vehicle.

The vehicle's model might predict a better response with lower amount of yaw rate or sideslip angle errors, but in experiment, the response of the vehicle can become worse when the new set of weights are considered for the MPC controller. Moreover, changing the weights may not cause a considerable amount of improvement in the response of the vehicle under some circumstances or vehicle states. It is intended to somehow recognize

the conditions under which no tuning or further tuning of the weights is necessary.

A weight authentication (WA) technique is presented in this section to achieve the aforementioned goal. The general structure of the proposed weight authentication method is shown in Figure 3.14. During each new test (simulation or experimental test), the measurements, either by the sensors or estimators, as well as the driver's inputs (including the steering wheel angle and requested torque), are received by the MPC controller. In some cases, the current vehicle's state values fall within the neighbourhood, ε_x, of other states that has been previously experienced or tested, and their corresponding weights have been tuned. These tuned weight are intended to be selected instead of the previous weights when vehicle undergoes the similar state condition. The MPC controller generates control actions based on the tuned weights which is applied to the vehicle. Through the authentication step, the response of the vehicle will be compared to that when previous weights were selected.

If, according to the vehicle's response, the yaw rate, and lateral velocity error is decreased more than a predefined threshold, then, the tuned weights are authenticated. Otherwise, the tuned weights are neglected, and previously selected weights are accepted. The details of the weight authentication are discussed in the following.

3.5.1 Weight Authentication Criterion

It is essential to consider a criterion to evaluate the effectiveness of tuned weights after they have been used in a test. Similar to the definition of the state variables error, and

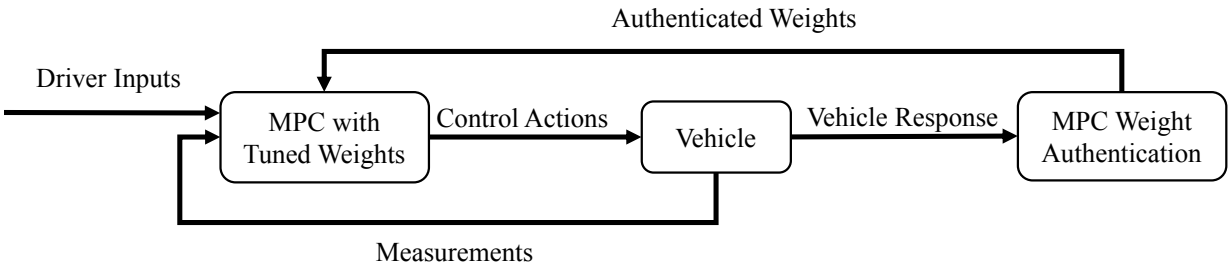

Figure 3.14: General diagram of the weight authentication process

error examination criterion in Eqs. (3.22) and (3.23), weight authentication criterion is defined as:

$$\bar{E}_{W_{pre}} = \frac{\gamma_r \sum_{i=0}^{l-1} \left| e_r^{t-it_s} \right|_{W_{pre}} + \gamma_\beta \sum_{i=0}^{l-1} \left| e_\beta^{t-it_s} \right|_{W_{pre}}}{l \left(\gamma_r e_r^{t-lt_s} + \gamma_\beta e_\beta^{t-lts} \right)} \tag{3.43}$$

$$\bar{E}_{W_{tun}} = \frac{\gamma_r \sum_{i=0}^{l-1} \left| e_r^{t-it_s} \right|_{W_{tun}} + \gamma_\beta \sum_{i=0}^{l-1} \left| e_\beta^{t-it_s} \right|_{W_{tun}}}{l \left(\gamma_r e_r^{t-lt_s} + \gamma_\beta e_\beta^{t-lts} \right)} \tag{3.44}$$

$$C_{W_{tun}} = \frac{\bar{E}_{W_{pre}} - \bar{E}_{W_{tun}}}{\bar{E}_{W_{pre}}} \times 100 \tag{3.45}$$

where $\bar{E}_{W_{pre}}$ and $\bar{E}_{W_{tun}}$ are error examination criteria calculated based on the actual response of the vehicle when previous and tuned MPC weights are used, respectively. Thus, the weight authentication criterion, $C_{W_{tun}}$, denotes the percentage of the error improvement when previous weights are substituted by the tuned weights.

A threshold may be considered for the value of $C_{W_{tun}}$ to decide whether accept or reject the tuned weights. Algorithm 2 presents the general algorithm based on this threshold:

Algorithm 2 Weight Authentication General Algorithm

if $C_{W_{tun}} > C_{W_{trs}}$ **then**
    Tuned weights, W_{tun}, are accepted
else
    Tuned weights, W_{tun}, are rejected
    Previous weights, W_{pre}, are accepted
    No further tuning in the similar vehicle's state
end if

It is noteworthy to mention that $C_{W_{tun}}$ will be negative when $\bar{E}_{W_{tun}} > \bar{E}_{W_{pre}}$. This happens when the response of the system becomes worse when tuned weights are selected by the

MPC controller. Therefore, the weight authentication can avoid unnecessary changes in the weight values as well as the weight changes that deteriorate the response of the system.

3.5.2 Weight Labeling

After the definition of the weight authentication criterion (Eq. (3.45)) and describing the general authentication algorithm, details of the weight authentication process are explained here. A weight authentication dataset is defined as:

$$D_{wa} = \begin{bmatrix} x_{D1} & W_{pre_1} & W_{tun_1} & \bar{E}_{W_{pre,1}} & \bar{E}_{W_{tun,1}} & C_{W_{tun,1}} & L_1 \\ x_{D2} & W_{pre_2} & W_{tun_2} & \bar{E}_{W_{pre,2}} & \bar{E}_{W_{tun,2}} & C_{W_{tun,2}} & L_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_{Dn} & W_{pre_n} & W_{tun_n} & \bar{E}_{W_{pre,n}} & \bar{E}_{W_{tun,n}} & C_{W_{tun,n}} & L_n \end{bmatrix} \tag{3.46}$$

where D_{wa} is the weight authentication dataset, x_{Di} is the array of vehicle's state and steering wheel angle expressed in Eq. (3.34), $\bar{E}_{W_{pre,i}}$, $\bar{E}_{W_{tun,i}}$, and $C_{W_{tun,i}}$ are calculated according to Eqs. (3.43) - (3.45), W_{pre_i} and W_{tun_i} denote the weight values before and after the tuning process, respectively, that can be expressed as:

$$W_{pre,i} = [w_{ri}, w_{\beta i}]_{pre}\,, \qquad W_{tun,i} = [w_{ri}, w_{\beta i}]_{tun} \tag{3.47}$$

Hence, for each of the experienced vehicle's states, there are two sets of weights: one before the weight tuning, and another one after the weights are tuned. The weight authentication criterion, $C_{W_{tun,i}}$ in Eq. (3.45), associated with each state is compared with its threshold, $C_{W_{trs}}$, and a label, L_i, is assigned to the vehicle's state, x_{Di}, according to the following different conditions:

- $L_i = 0$: Weights are changed through the weight tuning process, but not authenticated.

- $L_i = 1$: The tuned weights are tested and authenticated: $C_{W_{tun}} > C_{W_{trs}}$; Therefore, the tuned weights, $W_{tun,i}$, are replaced with the previous weights, $W_{pre,i}$.

- $L_i = 2$: The tuned weights are tested but not authenticated: $C_{W_{tun}} < C_{W_{trs}}$; In this condition, the tuned weights are neglected, and previous weights remain as the most appropriate weight for this vehicle state.

Under the second condition ($L_i = 1$), further weight tuning can be done in case the state of the vehicle becomes in the neighborhood,ϵ_X, of x_{D_i} and error examination criterion is still more than the threshold, $\bar{E}_{Q_{tres}}$. On the other hand, if the third condition happens ($L_i = 2$), no further tuning will be conducted even if the yaw rate and lateral velocity errors make the error examination criterion more than its threshold. Because the authentication process has already shown that the response of the vehicle will not get better than what it is even by changing the weights. This can avoid unnecessary increases or decreases in the weights for such vehicle states. As a means of better explicating the whole weight authentication process, the whole process of weight authentication is described in a step-by-step manner for an exemplary vehicle state as follows:

The vehicle is assumed to be in $x_1 = x(t) = \{r, u, v, \delta\}_t$ state condition in current time step, t. First, the MPC controller's weights are selected according to the real-time weight selection method discussed in 3.3.3. It is assumed that there is no data in the weight selection dataset, D_w, in the ϵ_X neighborhood of the current vehicle state. Thus, the default weights (w_r^{def} and w_β^{def}) are automatically used for the MPC objective function. The MPC controller generates the control actions using these default weights, and then the response of the vehicle in the future time steps are obtained. Based on the discussion provided in section 3.2, if the value of $\bar{E}_Q$ is less than its threshold, the default weights are accepted, the label of this vehicle state will be set to 2 ($L_i = 2$), and the weight tuning will not be carried out. Otherwise, the default weights become tuned according to the steps presented in section 3.2. This is illustrated in Figure 3.15.

In the future event that the vehicle's state falls within the ϵ_X neighborhood of this x_1 state, the tuned weights, W_{tun}, are set in the MPC controller instead of the default weights. Then, the wheel torques (control actuations) are obtained and applied to the vehicle, and the yaw rate and lateral velocity of the vehicle under these torques are captured. The

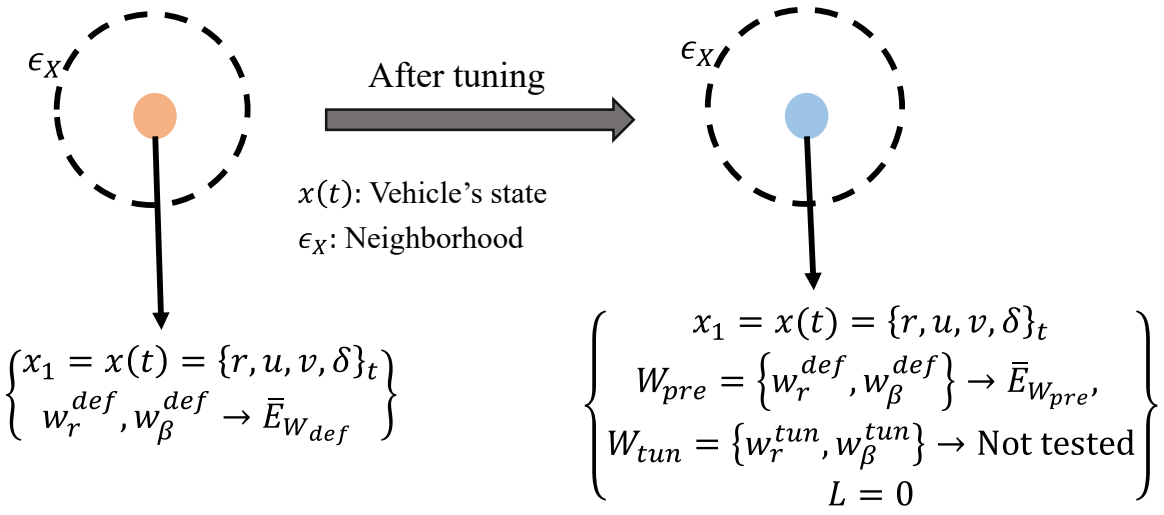

Figure 3.15: Tuning the weights from default values with respect to the vehicle's state datapoint and its neighborhood; the orange point represents the vehicle's state within its hypothetical neighborhood, ϵ_X, before tuning. As there is no point in this neighborhood, the default weights,w_r^{def}, w_β^{def} are selected. The blue point represents the previous vehicle's state after its corresponding weights are tuned. The default weights are considered as the previous weights, the tuned weights are intended to be tested in the future similar condition, and it is labeled by $L = 0$ in the weight authentication dataset, D_{wa}.

$\bar{E}_{W_{tun}}$ is calculated by Eq. (3.44), and is compared with $\bar{E}_{W_{pre}}$ which was obtained in the previous test. Later, the weight authentication criterion is calculated by Eq. (3.45), and a label is assigned to the vehicle's state. This process is illustrated in Figure 3.16

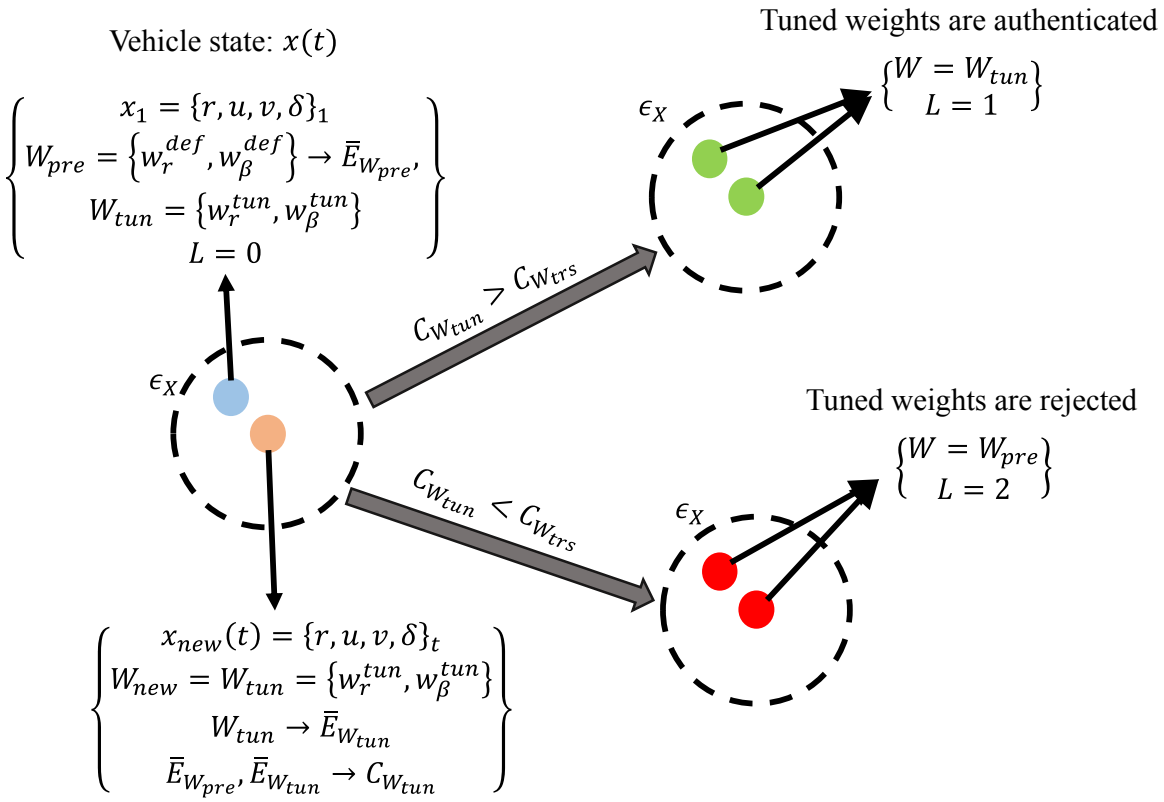

Figure 3.16: The process of authenticating the weights after a new test; The orange point represents the new vehicle's state within its hypothetical neighborhood, ϵ_X, after tuning. The blue point represents the previous vehicle's state in the neighborhood of the new state after its corresponding weight are tuned. The default weights are considered as the previous weights, the tuned weights are tested in this new condition. The labels are assigned according to the value of $C_{W_{tres}}$. Green points represent the authenticated weights, and red points show the rejected or unauthenticated points.

3.6 Weight Authentication Simulations

The effectiveness of the proposed weight authentication technique is evaluated through the following simulations. The first series of simulations include 4 consecutive DLC maneuver with the vehicle and MPC controller parameters provided in Tables 3.1 and 3.2, steering wheel angle input shown in Figure 3.7, and road friction $\mu = 0.8$. During the first maneuver, the MPC default weights are selected which remain fixed until the end of the maneuver. Tracking error weights are tuned three times such that after each tuning, a new simulation run is performed without the weights being authenticated.

Figure 3.17 depicts the yaw rate, sideslip angle, and real-time selected tracking error weights. The tracking error of the vehicle's yaw rate and sideslip angle are shown in Figure 3.18. Similar to the results illustrated in section 3.4, the RMSE of the yaw rate and sideslip angle tracking decreases after each round of weight tuning. However, without weight authentication, the weight values may become too large or small without having a considerable effect in the controller's performance. It can be noticed in Figure 3.17 that weight values become increased up to 500% after the third round of tuning, but yaw rate and sideslip angle results are approximately identical to those in the previous run.

The RMSE of the yaw rate tracking after the second and third weight tunings are very close, as shown in 3.18. This is also true for the sideslip angle tracking errors. Moreover, at some time steps after the third tuning, the amount of generated wheel torque is increased by more than 100% without having a discernible effect on the vehicle response (See Figure 3.19). It is intended to prevent such ineffective changes in the weight values and control actions to make the weight tuning more efficient.

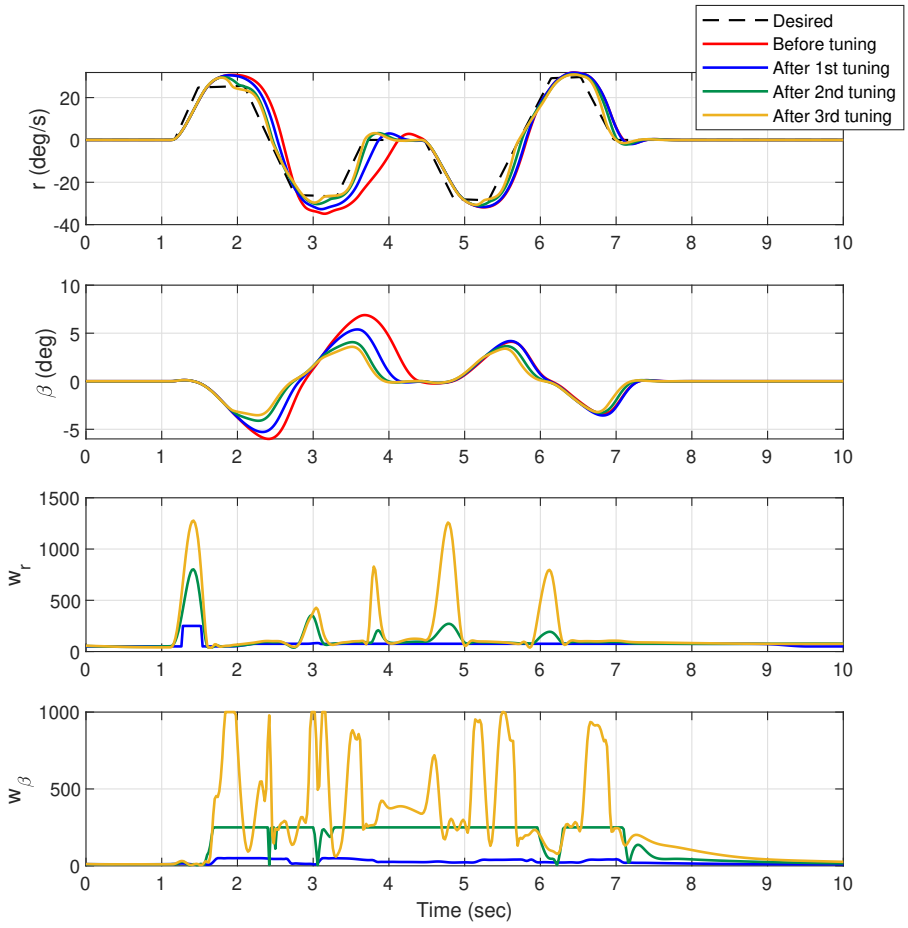

Figure 3.17: Responses of the vehicle in terms of yaw rate and sideslip angle, and the weights w_r, w_β selected in real-time when MPC controller is tuned without weight authentication.

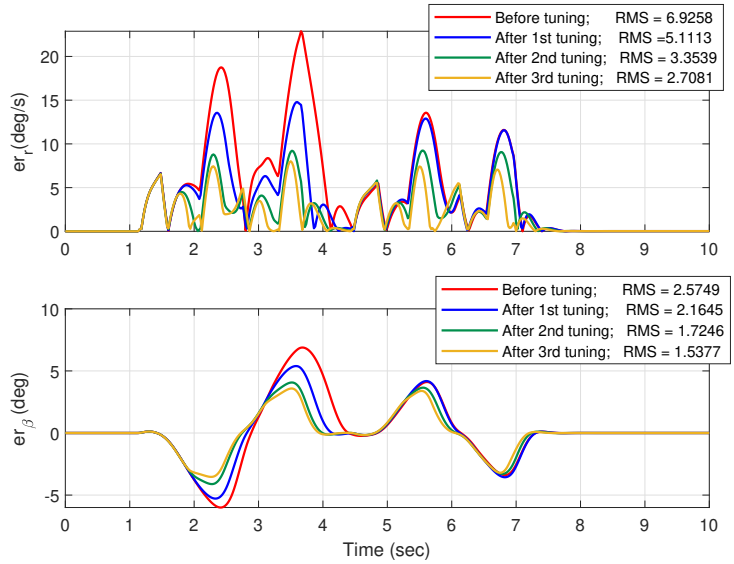

Figure 3.18: Tracking errors of the yaw rate and sideslip angle of the vehicle; MPC controller is tuned without weight authentication.

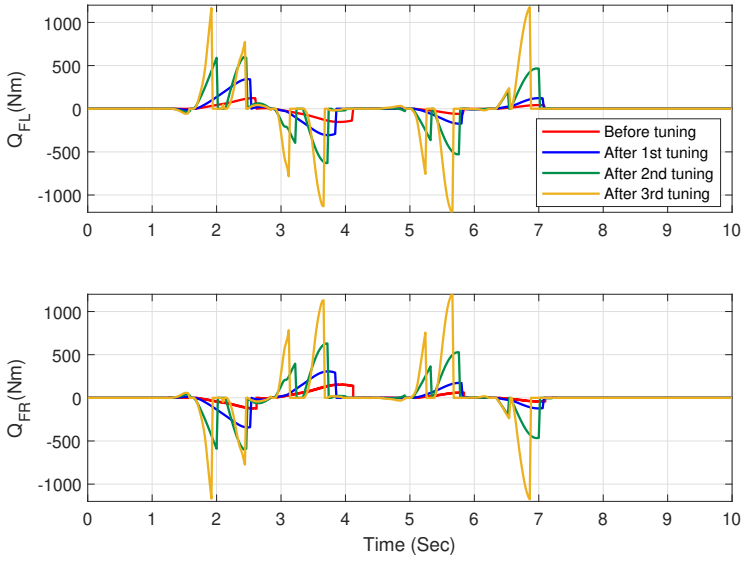

Figure 3.19: Front left and front right wheel torques generated by the MPC controller that becomes tuned without weight authentication.

The next series of simulations are carried out under the same conditions for the vehicle, with the exception that the weights are authenticated at the end of each simulation. As depicted in Figures 3.20 and 3.21, the ineffective sharp changes in the weight values are no longer noticed. It implies that during the weight authentication, the tuned weights are only accepted if they have considerably improved the MPC controller's tracking performance for the corresponding vehicle state. Following the third repetition of weight tuning, it can be observed that the weights remain identical to their previous values, except for a few time steps.

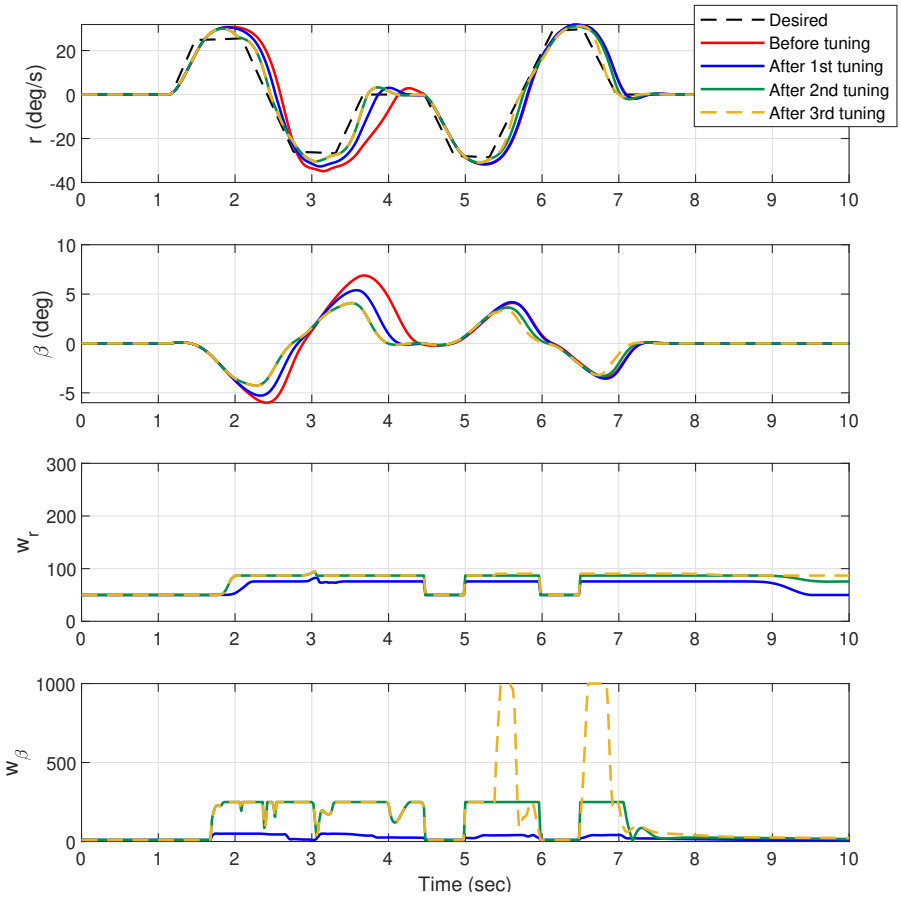

Figure 3.20: Responses of the vehicle in terms of yaw rate and sideslip angle, and the weights w_r, w_β selected in real-time when MPC controller is tuned with weight authentication.

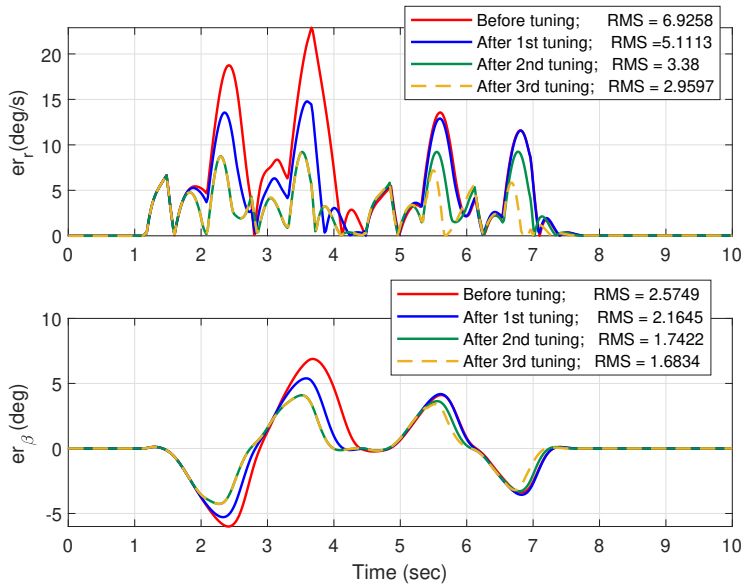

Figure 3.21: Tracking errors of the yaw rate and sideslip angle of the vehicle; MPC controller is tuned with weight authentication.

Similar behavior can be noticed in the generated wheel torques (See Figure 3.22).The majority of the time during the maneuver, the wheel torques do not reach the ineffective peak values observed in the last simulations (Figure 3.19) which makes the controller operate more efficiently. Therefore, the proposed weight authentication method has been effective in avoiding large unnecessary changes in the MPC controller's weights as well as optimizing the generated control action.

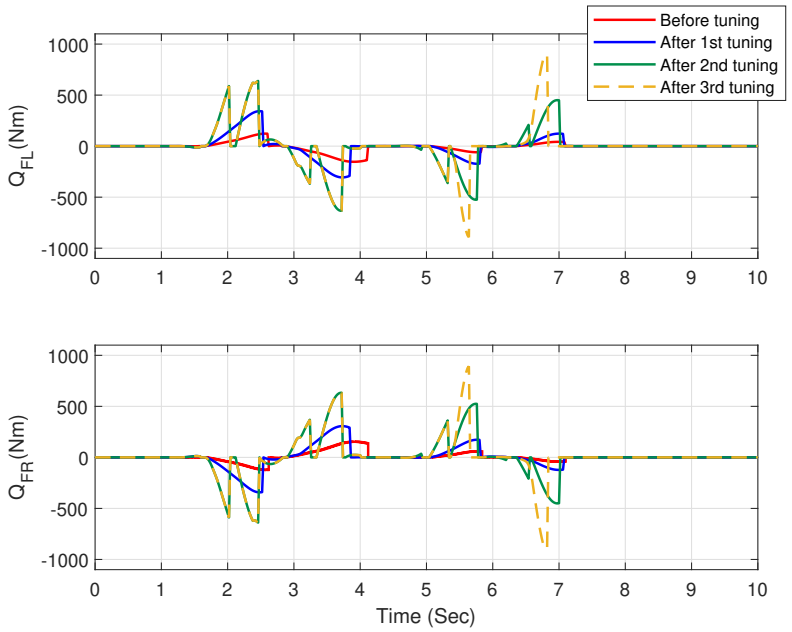

Figure 3.22: Front left and front right wheel torques generated by the MPC controller that becomes tuned with weight authentication.

Chapter 4

Multiple Model MPC Weight Tuning

As discussed in Chapter 3, the weight tuning is performed based on the vehicle model's predictions. Therefore, the accuracy of the predictions can directly impact the tuning process. For instance, if the model predicts that the yaw rate and sideslip angle of the vehicle remain close enough to their desired values, the MPC weights do not change. However, this prediction may not be accurate enough, and the yaw rate error becomes very large through the experiments. In another case, the vehicle model may predict a worsened yaw rate and sideslip angle tracking if the MPC weights remain unchanged, and the weights get increased or decreased in accordance with the prediction. Nevertheless, during the test, the change of weights might not cause any error improvement, or even deteriorate the results. Inaccuracies in predicted values may result in other challenges. Therefore, improving the model prediction improves the weight tuning.

The main reason for inaccuracies in the model prediction is the existence of uncertainties in some of the vehicle model parameters. For instance, the vehicle mass changes by the number of passengers, the road friction depends on the wear status of the tire and the road condition (could be slippery or dry), and all of these variations affect the response of the vehicle. Therefore, the prediction model must capture these parameter variations. Instead of a using a single model with fixed parameters, a multiple model approach [111, 112] can be considered for the MPC control and weight tuning processes. By using the multiple-model approach, the difference between the predicted and actual responses of the system

is monitored. Then, the model or a combination of models with the least prediction error will be employed to improve the predictions which, in turn, will improve the weight tuning. Two distinct approaches to multiple-model control exist: switching-based multiple-model control and blending-based multiple-model control. The following section will delve into the specifics of each approach.

4.1 Switching-Based Multiple-Model Control

The general architecture of a switching-based multiple model approach is illustrated in Figure 4.1. Similar to Eq. (2.14), the linearized real MIMO system is expressed in the form of:

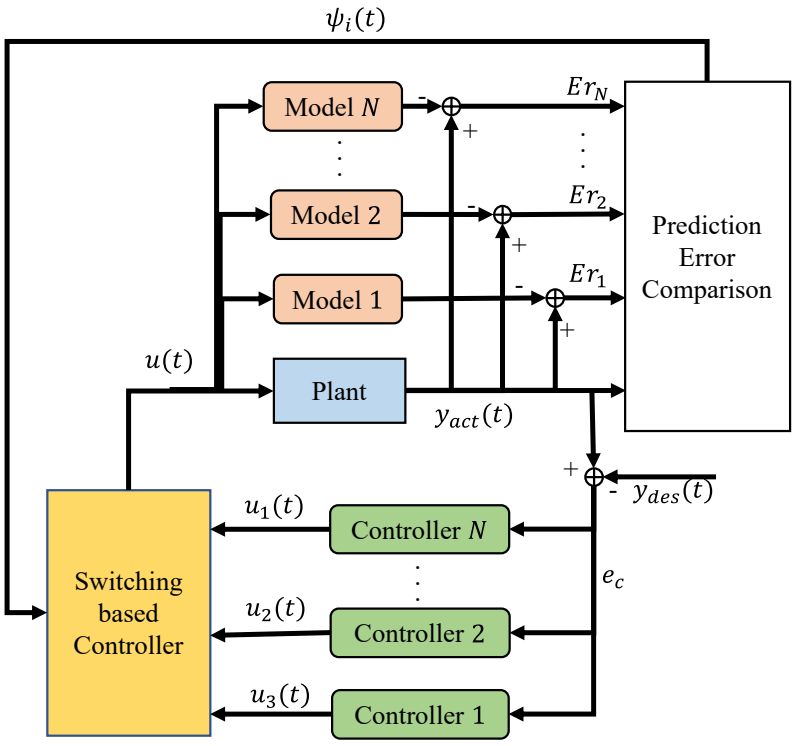

Figure 4.1: The switching based multiple-model diagram. The model with the minimum value of state variable error is selected to be used in the model-based controller.

$$\dot{x}_p(t) = A_p x_p(t) + B_p u(t) + D_p \tag{4.1}$$

where A_p, B_p, and D_p represent the plant matrices at each time step after linearization; $x_p(t)$ is the vector of real states of the system; and $u(t)$ represents the vector of control actuation inputs. In the switching based multiple-model approach, there are N number of fixed models, each with its own system matrices that can be written as:

$$\begin{aligned} \dot{x}_i(t) &= A_i x_i(t) + B_i u(t) + D_i, \\ x_i(t_0) &= x_i(t_0), \quad i = 1,\ 2,\ \ldots,\ N \end{aligned} \tag{4.2}$$

where A_i, B_i, and D_i are the system matrices associated with each model. Each of these matrices are formed based on a set of fixed parameters in the plant model. For instance, if there are 3 uncertain parameters in the model of the plant, and for each of these parameters, 2 different but fixed values are considered, then there are $N = 6$ number of models with their corresponding system matrices. The number, N, of these models are determined according to the level of uncertainties in the system's parameters, the total variation that can be considered for the uncertain parameters, and the amount of accuracy that is intended in the predicted values. The exact same control actuation input is entered to each prediction model, and the state error of each model is obtained as:

$$Er_i(t) = x_p(t) - x_i(t) \tag{4.3}$$

A cost function should be defined to compare the outputs of the models at each time step. In case of discretized prediction models, the cost function $J_i(t)$ can be defined as a function of prediction errors $Er_i(t)$ in a predefined sequence of passed time steps:

$$J_i(t) = \sum_{i=0}^{l} \left\| Er_i(t - i.t_s) \right\| \tag{4.4}$$

where l is the number of preceding time steps from which the prediction error is included in the cost function, and t_s is the sampling time. The cost function $J_i(t)$ in Eq. (4.4) is calculated for each model in real-time. These values are then compared and the model with the minimum $J_i(t)$ is selected as the most accurate model to be utilized in the controller.

The selection of the best model can be done by defining an index signal as:

$$\psi_i(t) = \arg \min_{i \in \{1,2,\ldots,3\}} J_i(t), \qquad \psi_i(t) \in \{1,\ 2, \ldots, N\}. \tag{4.5}$$

At each time step, if the selected best model is close enough to the real plant, then the model-based controller can perform based on the most accurate prediction model. Otherwise, there is still a considerable amount of prediction error. To ensure that there is at least one model with sufficient accuracy (close enough to the actual plant) within the range of operation of the system, it is essential to distribute the N prediction models correctly.

As mentioned earlier, the A_i, B_i, and D_i matrices of each prediction model are formed based on a set of fixed parameters. Each of these uncertain parameters has a maximum and a minimum value, and the prediction model matrices are distributed according to these ranges. If the selected parameters for the fixed models are very far from each other, the predictions may not be accurate enough in some instances. In contrast, if the selected parameters are very close to each other, the number of fixed models will be very large making the switching based multiple model control inefficient in terms of calculation time.

4.2 Blending-Based Multiple-Model Control

The application of a switching-based multiple-model approach may pose some challenges. First, the number of prediction models increases exponentially when the number of uncertain parameters is increased. This undesirable growth in the number of models N is due to the fact that at each time step, there must be at least one model which is close enough to the actual plant [124]. Additionally, only the best model which has the minimum value of the cost function $J_i(t)$ contributes to the prediction of the states. Although the prediction error obtained by other models can give some information to come up with a more accurate model, the switching based approach neglects all of them.

To use the data obtained from all of the models, [111, 112] has developed a blending-based multiple model approach that instead of selecting only one of the prediction models,

a weighted combination of them based on their prediction errors is utilized. In this section, the proposed blending based approach is applied to a linearized MIMO system for the purpose of this thesis.

Similar to Eq. (4.1), the state space equation of an uncertain linearized MIMO system can be written as:

$$\dot{x}_p(t) = A_p(\eta)\, x_p(t) + B_p(\eta)\, u(t) + D_p(\eta) \tag{4.6}$$

where $A_p(\eta)$, $B_p(\eta)$, and $D_p(\eta)$ are the linearized matrices of the plant expressed as a function of the vector of uncertain parameters of the system η, $x_p(t)$ is the vector of real system states, and $u(t)$ denotes the control actuation inputs of the system. The following assumption is considered to be able to utilize the blending based approach:

Assumption 4.1 *There exist N number of linearized matrices of the plant A_i, B_i, and D_i such that for any value of the vector of uncertain parameters η:*

$$\begin{bmatrix} A_p(\eta) & B_p(\eta) & D_p(\eta) \end{bmatrix} \in \mathbf{Co}\left\{ \begin{bmatrix} A_i & B_i & D_i \end{bmatrix} : i = 1, 2, \ldots, N \right\}, \tag{4.7}$$

where $\mathbf{Co}\{.\}$ expresses the convex hull of a set of matrices. We can consider a fixed model for each set of uncertain parameters as:

$$\dot{x}_i(t) = A_i x_i(t) + B_i u(t) + D_i \tag{4.8}$$

According to the convexity property expressed in Assumption 4.1, Eq. (4.7) can be rewritten as:

$$A_p(\eta)\, x_p(t) + B_p(\eta)\, u(t) + D_p(\eta) = \sum_{i=1}^{N} w_i(t) \left[A_i x_p(t) + B_i u(t) + D_i \right], \tag{4.9}$$

where $w_i(t)$ denotes the weights of model i at each time step t, and satisfies the following condition:

$$\sum_{i=1}^{N} w_i(t) = 1, \qquad w_i(t) \geq 1 \tag{4.10}$$

There are N number of weights w_i corresponding to N system matrices A_i, B_i, and D_i at each time step, and it is intended to obtain the weights based on the actual response of the system. Therefore, the plant matrices $A_p(\eta)$, $B_p(\eta)$, and $D_p(\eta)$ are a weighted summation of the N models' matrices.

4.2.1 Obtaining the Vector of Weights

For the purpose of this thesis, the discretized form of Eq. (4.6) is considered:

$$x_p(k+1) = A_p^d(\eta)x_p(k) + B_p^d(\eta)u(k) + D_p^d(\eta), \tag{4.11}$$

where $A_p^d(\eta)$, $B_p^d(\eta)$, and $D_p^d(\eta)$ denote the plant matrices of the discretized state space equation of the system as a function of the vector of uncertain parameters η. Eq. (4.11) can be rewritten in the parametric form as:

$$x_p(k+1) = \Theta_p(\eta)\,\Phi(k) \tag{4.12}$$

where

$$\Theta_p(\eta) = \begin{bmatrix} A_p^d(\eta) & B_p^d(\eta) & D_p^d(\eta) \end{bmatrix}$$

$$\Phi(k) = \begin{bmatrix} x_p(k) \\ u(k) \\ 1 \end{bmatrix} \tag{4.13}$$

The N number of discretized fixed models at each time step are represented in the parametric form as:

$$\begin{aligned} x_i(k+1) &= \Theta_i\Phi(k), \qquad i = 1, 2, \ldots, N, \\ \Theta_i &= \begin{bmatrix} A_i^d & B_i^d & D_i^d \end{bmatrix} \end{aligned} \tag{4.14}$$

The state prediction errors for the N prediction models can be defined as:

$$\varepsilon_i(k) = x_p(k+1) - \Theta_i\Phi(k), \qquad i = 1, 2, \ldots, N, \tag{4.15}$$

The following equations can be derived from Eq. (4.9) based on the error definition in Eq. (4.15):

$$\begin{aligned} \Theta_p(k) &= \sum_{i=1}^{N} w_i(k)\Theta_i, \\ \sum_{i=1}^{N} w_i(k)\varepsilon_i(k) &= 0, \end{aligned} \tag{4.16}$$

These relations can be rewritten as:

$$\begin{aligned} &E(k)W(k) = 0 \\ &E(k) = \begin{bmatrix} \varepsilon_1(k) & \varepsilon_2(k) & \dots & \varepsilon_N(k) \end{bmatrix} \\ &W(k) = \begin{bmatrix} w_1(k) & w_2(k) & \dots & w_N(k) \end{bmatrix}^T \end{aligned} \tag{4.17}$$

Using Eq. (4.10), the weight of the last model is calculated as:

$$w_N(k) = 1 - \sum_{i=1}^{N-1} w_i(k). \tag{4.18}$$

Therefore, by substituting Eq. (4.18) in Eq. (4.17), the following relations are derived:

$$\begin{aligned} &E'(k) = \begin{bmatrix} \varepsilon_1(k) - \varepsilon_N(k) & \varepsilon_2(k) - \varepsilon_N(k) & \dots & \varepsilon_{N-1}(k) - \varepsilon_N(k) \end{bmatrix} \\ &W'(k) = \begin{bmatrix} w_1(k) & w_2(k) & \dots & w_{N-1}(k) \end{bmatrix}^T \\ &E'(k)W'(k) = -\varepsilon_N(k) \end{aligned} \tag{4.19}$$

It is intended to calculate the elements of $W(k)$ at each time step k to obtain an estimation of $\Theta_p(k)$ using Eq. (4.16). According to the relations in Eq. (4.19), the elements of $W'(k)$ vector can be computed as a function of prediction errors ε_i. In the case

of $E'(k)$ in Eq. (4.19) being full-rank, $W'(k)$ is computed as:

$$W'(k) = \begin{cases} \left[E'^T(t)E'(t)\right]^{-1} E'^T(t)\varepsilon_N(t) & n \geq N-1 \\ -E'^T(t)\left[E'(t)E'^T(t)\right]^{-1}\varepsilon_N(t) & n < N-1 \end{cases} \tag{4.20}$$

and satisfies:

$$E'^T E'(k)W'(k) + E'^T + \varepsilon_N(k) = 0 \tag{4.21}$$

The convexity property described in Assumption 4.1, and the blending model based control approach are illustrated respectively in Figures 4.2 and 4.3.

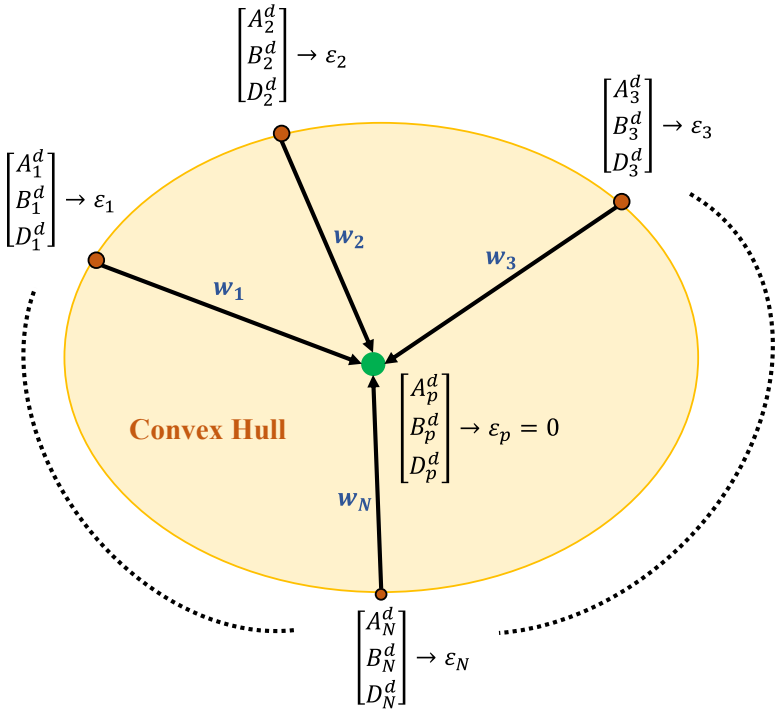

Figure 4.2: Graphical illustration of the convex hull formed by the N models, and contribution of each model to the estimation of the plant model.

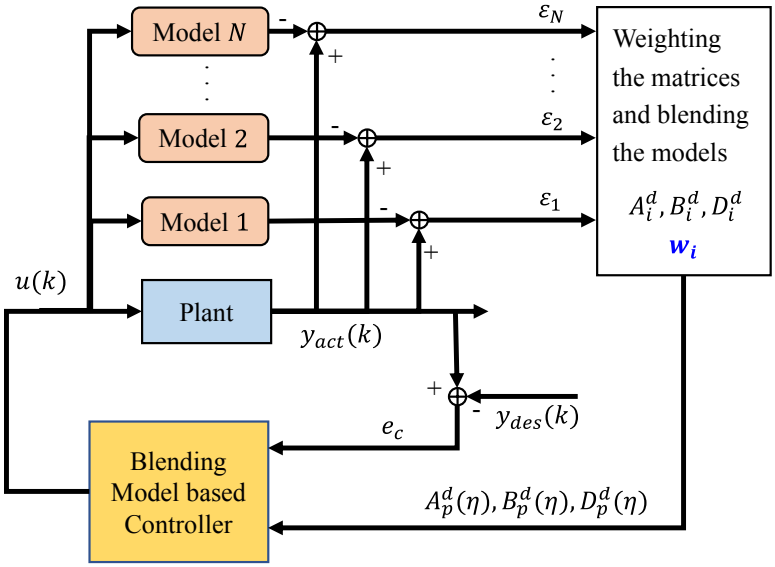

Figure 4.3: The Blending Based multiple-model diagram. The models' matrices are weighted based on their prediction errors and blended.

4.3 Blending-Based Multiple-Model MPC for Vehicle Lateral Stability

In this section, the blending based multiple model approach presented in section 4.2 is applied to the vehicle prediction model described in section 2.5 to be used in the MPC controller presented in section 2.9 to control the vehicle lateral stability. The equations of the lateral dynamics of the vehicle (Eqs. 2.4 and 2.5) can be rewritten as:

$$\dot{\beta} = -r + \frac{1}{\eta_m m u}\left(F_{yf}\cos(\delta) + F_{xf}\sin(\delta) + F_{yr}\right) \tag{4.22}$$

$$\dot{r} = \frac{1}{I_z}\left(l_f F_{yf}\cos(\delta) - l_r F_{yr} + l_f F_{xf}\sin(\delta) + M_{DY}\right) \tag{4.23}$$

where η_m is the uncertainty variable associated with the vehicle's mass that captures the changes in the mass; F_{yf} and F_{yr} are respectively the uncertain front and rear lateral forces

caluculated by reforming the Eq. (2.6) as:

$$F_y(\eta) = \begin{cases} -\eta_c C_\alpha \tan\alpha + \frac{(\eta_c C_\alpha)^2}{3\xi\eta_\mu\mu F_z} |\tan\alpha| \tan\alpha - \frac{(\eta_c C_\alpha)^3}{27\xi^2(\eta_\mu\mu)^2 F_z^2} \tan^3\alpha & |\alpha| < \alpha_{sl} \\ -\xi\eta_\mu\mu F_z \text{ sign } \alpha & |\alpha| \geq \alpha_{sl} \end{cases} \tag{4.24}$$

$$\alpha_{sl} = \arctan \frac{3\xi\eta_\mu\mu F_z}{\eta_c C_\alpha},$$

where η_c denotes the uncertainty in the tire's cornering stiffness, and η_μ specifies the road friction coefficient uncertainty.

After obtaining each tire's lateral force $F_{yi}(\eta)$ by using Eq. (4.24), $\bar{C}_{\alpha_i}$ and $\bar{C}_{Q_i}$ are computed by Eqs. (2.11) and (2.13), respectively as:

$$\begin{aligned} \bar{C}_{\alpha_i}(\eta) &= \frac{\partial F_{yi}(\eta)}{\partial \alpha_i}|_{\bar{\alpha}_i, \bar{\xi}_i}, \\ \bar{C}_{Q_i}(\eta) &= \left[\frac{\partial F_{yi}(\eta)}{\partial \xi_i} \frac{\partial \xi_i}{\partial F_{xi}} \frac{\partial F_{xi}}{\partial Q_i}\right]_{\bar{\alpha}_i, \bar{\xi}_i} \end{aligned} \tag{4.25}$$

Then, the linearized state-space form of Eqs. (4.22) and (4.23) is expressed as:

$$\dot{x}_p(t) = A_p(\eta)\, x_p(t) + B_p(\eta)\, u(t) + D_p(\eta) \tag{4.26}$$

where $\eta = \begin{bmatrix} \eta_m \\ \eta_c \\ \eta_\mu \end{bmatrix}$ denotes the vector of uncertain parameters; $x_p(t) = \begin{bmatrix} \beta \\ r \end{bmatrix}$ is the vector of vehicle's system states; $u(t) = \begin{bmatrix} Q_f \\ M_{DY} \end{bmatrix}$ represents the system inputs including the total front torque, Q_f and corrective direct yaw moment, M_{DY} generated by the torque vectoring; $A_p(\eta)$, $B_p(\eta)$, and $D_p(\eta)$ are the uncertain matrices of the plant that can be computed as:

$$A_p(\eta) = \begin{bmatrix} \frac{\bar{C}_{\alpha_f}(\eta)\cos\delta + \bar{C}_{\alpha_r}(\eta)}{\eta_m m u} & \frac{l_f \bar{C}_{\alpha_f}(\eta)\cos\delta - l_r \bar{C}_{\alpha_r}(\eta)}{\eta_m m u^2} - 1 \\ \frac{l_f \bar{C}_{\alpha_f}(\eta)\cos\delta - l_r \bar{C}_{\alpha_r}(\eta)}{I_z} & \frac{l_f^2 \bar{C}_{\alpha_f}(\eta)\cos\delta + l_r^2 \bar{C}_{\alpha_r}(\eta)}{I_z u} \end{bmatrix} \tag{4.27}$$

$$B_p(\eta) = \begin{bmatrix} \frac{\bar{C}_{Q_f}(\eta)\cos\delta - \bar{C}_{Q_r}(\eta)}{\eta_m m u} + \frac{\sin\delta}{\eta_m m u R_e} & 0 \\ \frac{l_f \bar{C}_{Q_f}(\eta)\cos\delta + l_r \bar{C}_{Q_r}(\eta)}{I_z} + \frac{l_f \sin\delta}{I_z R_e} & \frac{1}{I_z} \end{bmatrix}$$

$$D_p(\eta) = \begin{bmatrix} \frac{\bar{F}_{yf}(\eta)\cos\delta + \bar{F}_{yr}(\eta) - \bar{C}_{\alpha_f}(\eta)\bar{\alpha}_f\cos\delta - \bar{C}_{\alpha r}(\eta)\bar{\alpha}_r - \bar{C}_{Q_f}(\eta)\bar{Q}_f\cos\delta - \bar{C}_{Q_r}(\eta)\bar{Q}_r - \bar{C}_{\alpha_f}(\eta)\delta\cos\delta}{\eta_m m u} \\ \frac{l_f \bar{F}_{yf}(\eta)\cos\delta - l_r \bar{F}_{yr}(\eta) - l_f \bar{C}_{\alpha_f}(\eta)\bar{\alpha}_f\cos\delta + l_r \bar{C}_{\alpha r}(\eta)\bar{\alpha}_r - l_f \bar{C}_{Q_f}(\eta)\bar{Q}_f\cos\delta + l_r \bar{C}_{Q_r}(\eta)\bar{Q}_r - l_f \bar{C}_{\alpha_f}(\eta)\delta\cos\delta}{I_z} \end{bmatrix}$$

The elements of uncertain parameters in η are assumed to lie within the following ranges:

$$\eta_{m,\min} \le \eta_m \le \eta_{m,\max}, \quad \eta_{c,\min} \le \eta_c \le \eta_{c,\max}, \quad \eta_{\mu,\min} \le \eta_\mu \le \eta_{\mu,\max} \tag{4.28}$$

The zero-order hold (ZOH) method is used to discretize the continuous-time model of Eqs. (4.26) and (4.27) as:

$$\begin{aligned} x_p(k+1) &= A_p^d(\eta)x_p(k) + B_p^d(\eta)u(k) + D_p^d(\eta), \\ A_p^d(\eta) &= e^{A_p^d(\eta)t_s}, \quad B_p^d(\eta) = e^{B_p^d(\eta)t_s}, \quad D_p^d(\eta) = e^{D_p^d(\eta)t_s} \end{aligned} \tag{4.29}$$

where $A_p^d(\eta)$, $B_p^d(\eta)$, and $D_p^d(\eta)$ indicate the plant matrices of the discretized state space equation of the system with the sampling time, t_s, and as a function of the vector of uncertain parameters η. According to the Assumption 4.1, it is essential to construct N number of fixed models (A_i^d, B_i^d, and D_i^d) using Eqs. (4.27) and (4.29) by considering

uncertain parameters limits (Eq. (4.28)) such that:

$$\left[A_p^d(\eta) \quad B_p^d(\eta) \quad D_p^d(\eta)\right] \in \mathbf{Co}\left\{\left[A_i^d \quad B_i^d \quad D_i^d\right] : i = 1, 2, \ldots, N\right\} \tag{4.30}$$

Using the parametric form of Eq. (4.15), the vehicle's state prediction errors are expressed as:

$$\varepsilon_i(k) = x_p(k+1) - \Theta_i \Phi(k), \qquad i = 1, 2, \ldots, N, \tag{4.31}$$

where

$$\Theta_i = \left[A_i^d \quad B_i^d \quad D_i^d\right]$$

$$\Phi(k) = \begin{bmatrix} \beta \\ r \\ u(k) \\ 1 \end{bmatrix} \tag{4.32}$$

Thus, weights $w_i(k)$ associated with each model can be obtained using Eqs. (4.17)-(4.20).

In section (2.9), the MPC controller used to control the vehicle lateral stability was discussed. It was assumed that discretized plant matrices (A_p, B_p, and D_p) are formed based on fixed parameters of m, μ, and C_α. Now, instead of fixed plant matrices, the weighted combination of the matrices of N models (A_i^d, B_i^d, and D_i^d) in Eq. (4.30) are utilized for the MPC controller with the same control objective and constraints. The discretized uncertain plant model (Eq. 4.29) is considered with respect to the actuator and state constraints provided in Eq. (2.36).

Based on the bounds of uncertain parameters in (4.28), matrices of the N models $\Theta_i = \left[A_i^d \quad B_i^d \quad D_i^d\right]$ are determined at each time step k. Then, the weighting vector $W(k)$ is obtained by using the vehicle's state prediction errors calculated for each model in Eq. (4.31) and the relations provided in Eqs. (4.17)-(4.20). After that, the estimated

discretized plant matrices are obtained as:

$$\hat{A}_p^d = \sum_{i=1}^{N} w_i A_i^d \hat{B}_p^d \quad = \sum_{i=1}^{N} w_i B_i^d \hat{D}_p^d = \sum_{i=1}^{N} w_i D_i^d \tag{4.33}$$

At each time step, $\hat{A}_p^d$, $\hat{B}_p^d$, and $\hat{B}_p^d$ are used as the matrices of the vehicle prediction model for the MPC controller. Figure (4.4) illustrates this blending based multiple-model predictive controller.

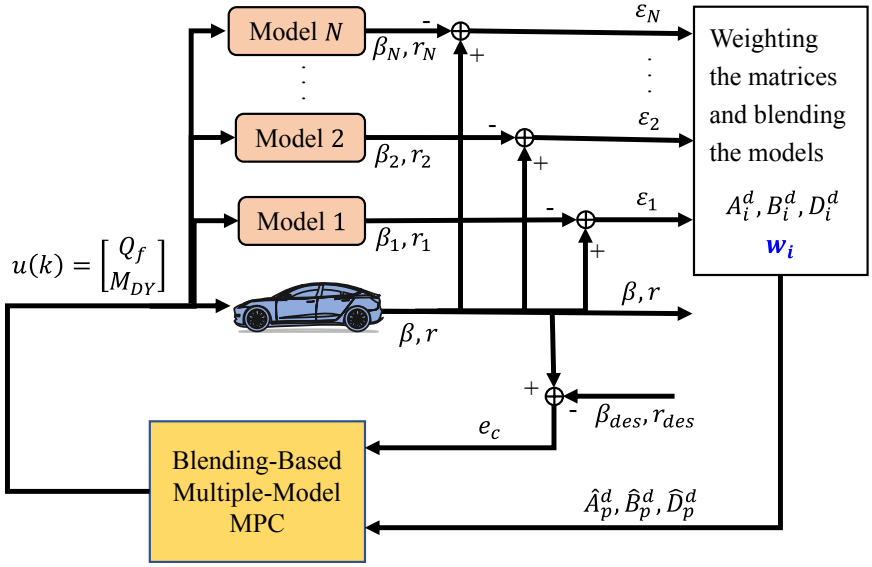

Figure 4.4: The blending based multiple-model predictive control for vehicle lateral stability.

4.4 Multiple-Model-Based Weight Tuning

The developed MPC controller weight tuning and weight selection method has been discussed and used to control the vehicle's lateral stability in Chapter 3. As previously mentioned, the weight tuning is performed based on the actual response of the system and state predictions made by the vehicle's predictions model. The single prediction model

utilized in the weight tuning module can be substituted by the estimated multiple model plant matrices $\hat{A}_p^d$, $\hat{B}_p^d$, and $\hat{B}_p^d$ in Eq. (4.33).

4.5 Simulations

In this section, the proposed multiple model approach is evaluated through MATLAB/Simulink and CarSim co-simulations. The parameters of the vehicle, and MPC controller are respectively provided in Tables 3.1 and 3.2. The number of models, N, as well as the bounds of uncertain parameters in η are provided in Table 4.1. The simulations in this section consider DLC maneuvers with the steering wheel angle input shown in Figure 4.5.

Table 4.1: Parameters of the multiple prediction model.

Parameter	**Value**	**Description**
N	8	Number of models
$\eta_{m,min}$	0.8	Mass uncertainty lower bound
$\eta_{m,max}$	1.2	Mass uncertainty upper bound
$\eta_{c,min}$	0.5	Cornering stiffness uncertainty lower bound
$\eta_{c,max}$	1.3	Cornering stiffness uncertainty upper bound
$\eta_{\mu,min}$	0.2	Road friction coefficient uncertainty lower bound
$\eta_{\mu,max}$	1.2	Road friction coefficient uncertainty upper bound

Two initial simulation runs are performed on a dry road with $\mu = 0.8$ and initial longitudinal speed of $u_0 = 70$ kph to evaluate the prediction accuracy of the MPC controller vehicle model. Figure 4.6 demonstrates the differences between the predictions made by the vehicle's single (nominal) model and multiple model, and compares them with the actual response of the vehicle. The predicted yaw rate and sideslip angle by the multiple model are more precise than those predicted by the single nominal model. It is because the multiple model method uses the prediction error obtained from a variety of models with different parameters and weights them according to their accuracy. In this case,

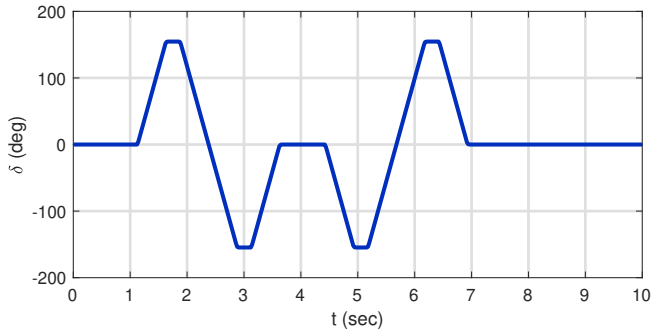

Figure 4.5: Steering wheel angle input.

the multiple model approach has improved the RMSE of the yaw rate and sideslip angle predictions by about 40% and 42%, respectively.

In the next step, the simulation cases discussed in sections 3.6 and 3.4 are repeated with the designed multiple model MPC controller weight tuning technique. Figure 4.7 compares the vehicle responses along with the selected MPC weight w_r, w_β when single model and multiple model methods are applied, and vehicle is on a dry road. The yaw rate becomes closer to the desired value, and sideslip angle decreases when the nominal prediction model is replaced by the multiple model. The RMSE of the tracking is improved by 20% and 13%, respectively for the yaw rate and sideslip angle (Fig. 4.8).

As illustrated in the two bottom graphs in Figure 4.7, the variation of MPC weights tuned by the multiple model is smoother than that by the single model. Furthermore, multiple model tuned weights are slightly larger than those tuned by the single model, which is resulted by the fact that multiple model can better predict the future state of the vehicle, hence adjust the controller weights quicker. This has also affected the generated front wheel torques as depicted in Figure 4.9. At some time steps, the wheel torque values are slightly larger when controller is tuned with the multiple model, resulted by a more precise state prediction and better adjusted MPC weights w_r, w_β.

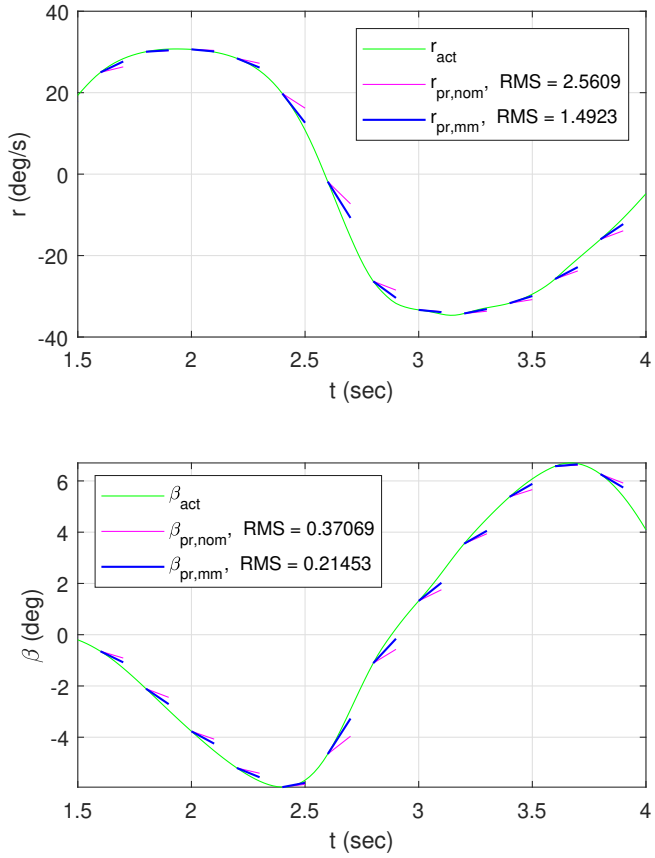

Figure 4.6: Comparison of the yaw rate and sideslip angle predictions made by the nominal model and multiple model. Green curve shows the actual response of the vehicle; Pink and blue lines indicate the predicted response by the nominal and multiple model, respectively.

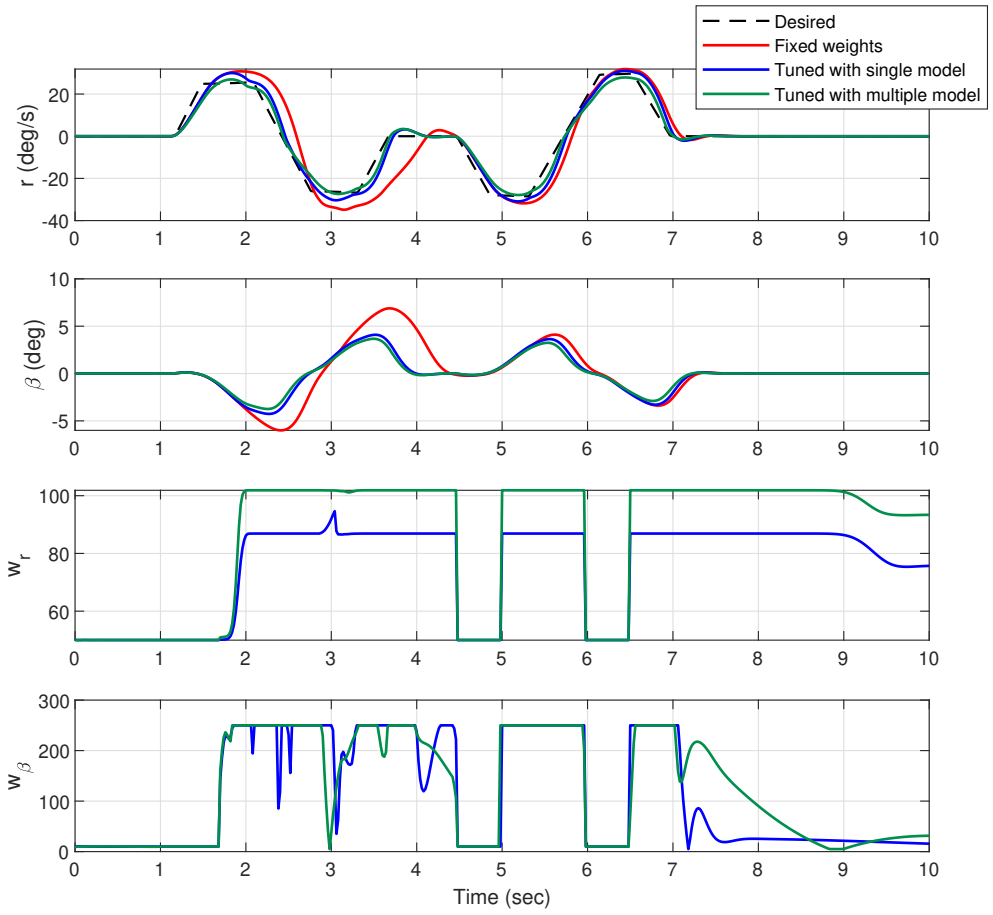

Figure 4.7: Responses of the vehicle in terms of yaw rate and sideslip angle, and the weights w_r, w_β selected in real-time after the MPC is tuned with single and multiple model approaches; Vehicle is on a dry road.

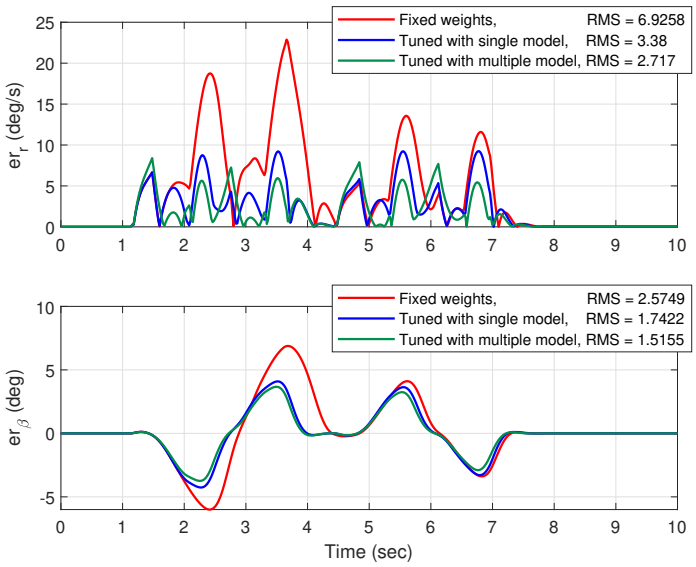

Figure 4.8: Tracking errors of yaw rate and sideslip angle after the MPC is tuned with single and multiple model approaches; Vehicle is on a dry road.

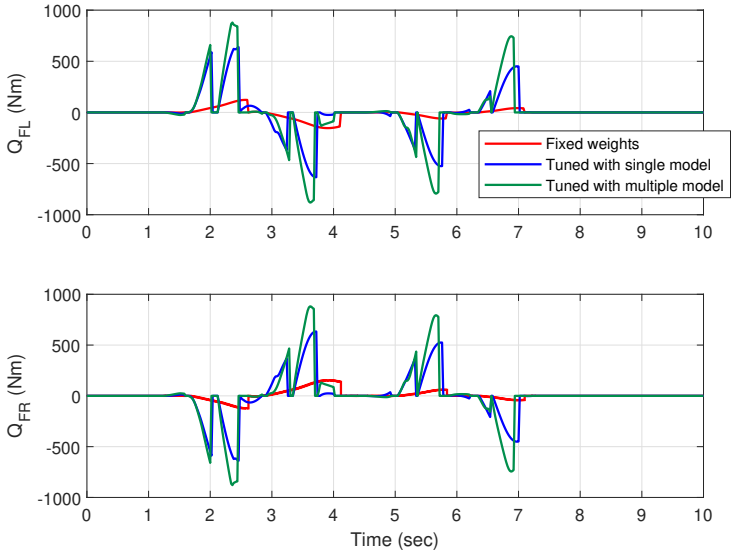

Figure 4.9: Comparison of the generated front wheel torques when MPC is tuned with the single and multiple model approaches; Vehicle is on a dry road.

The road condition changes from dry to slippery with $\mu = 0.6$, and the following set of simulations are performed. To have a better insight in the capability of the proposed multiple model method, one simulation run of weight tuning is carried out when single (nominal) model is used; then, one run of multiple model weight tuning is preformed, and final results are compared (Fig. 4.10). As the road is slightly slippery, the deviation of the yaw rate from its desired value in the middle of the DLC maneuver is considerable when fixed weights are set in the objective function of the MPC controller. Additionally, the sideslip angle reaches a peak value of more than 10 degrees. Therefore, the default weights cannot maintain the stability of the vehicle.

The RMSE of the yaw rate and sideslip angle tracking are respectively improved by about 26% and 20%, when nominal model is replaced with the multiple model(Fig. 4.11). Hence, even one run of multiple model weight tuning performs better than the MPC tuning by means of the fixed nominal model. The fluctuations in the weights are less when tuned by the multiple model than those tuned by the single model. As a result of the aforementioned improvement, the generated wheel torques have been adjusted such that sufficient amount of torque is acted on each wheel to satisfy the control objective (Fig. (4.12)).

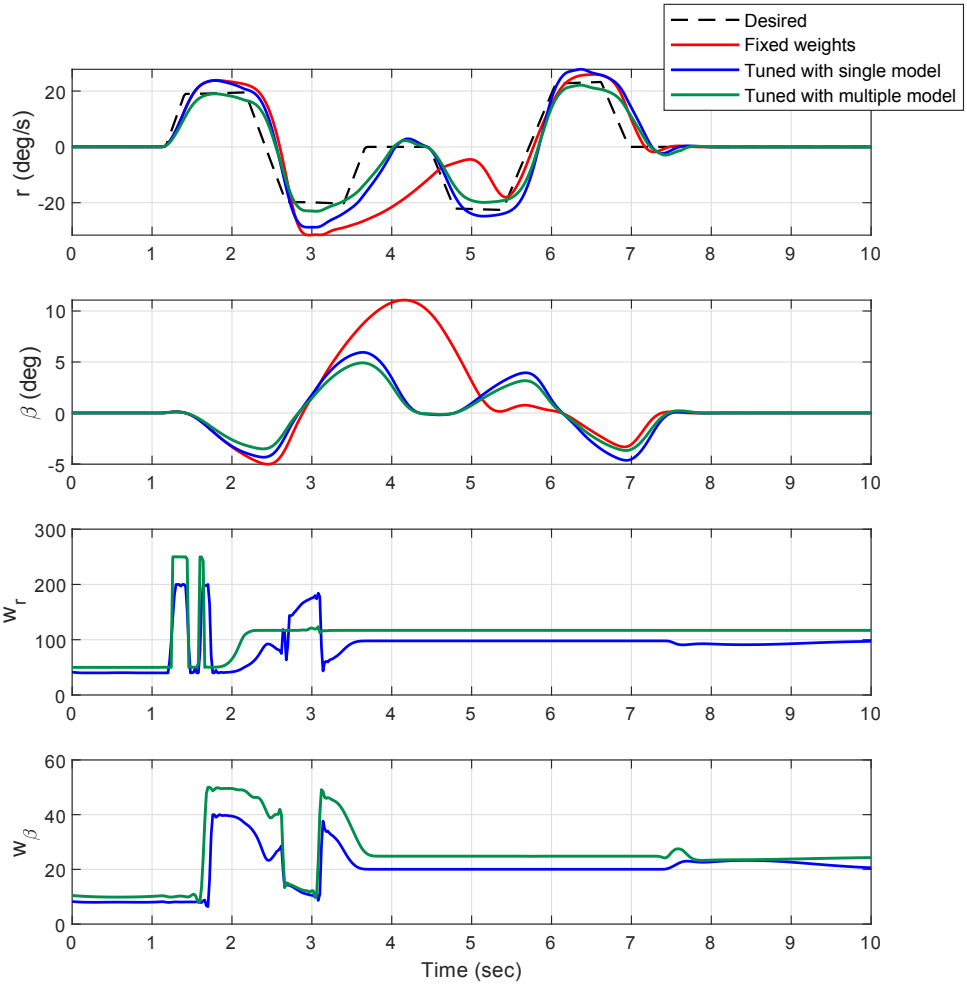

Figure 4.10: Responses of the vehicle in terms of yaw rate and sideslip angle, and the weights w_r, w_β selected in real-time after the MPC is tuned with single and multiple model approaches; Vehicle is on a slippery road.

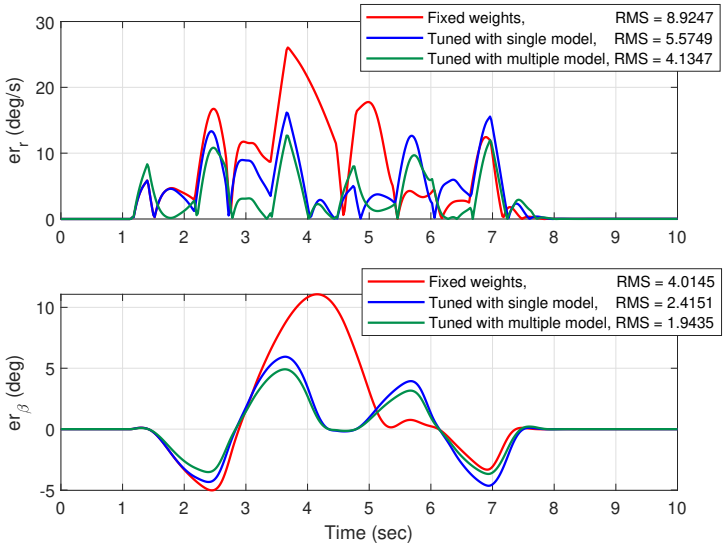

Figure 4.11: Tracking errors of yaw rate and sideslip angle after the MPC is tuned with single and multiple model approaches; Vehicle is on a slippery road.

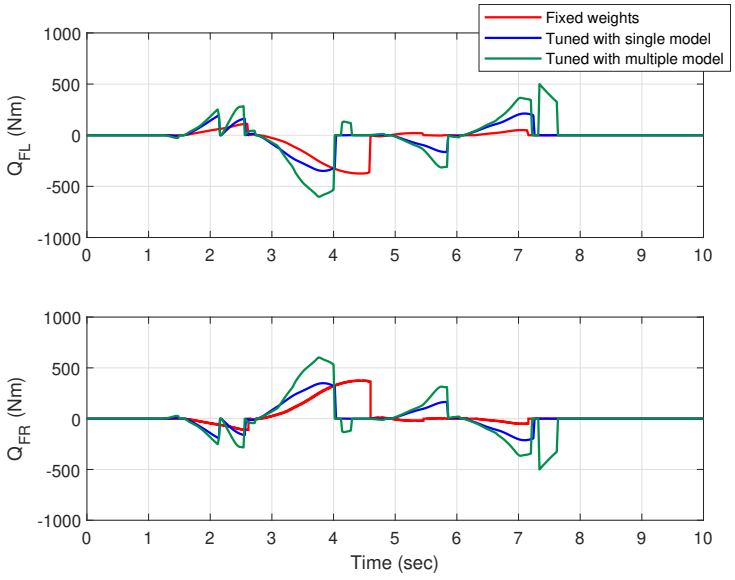

Figure 4.12: Comparison of the generated front wheel torques when MPC is tuned with the single and multiple model approaches; Vehicle is on a slippery road.

Chapter 5

Experimental Studies

In this chapter, the proposed MPC controller weight tuning is evaluated through two main series of experimental tests. An all-wheel drive (AWD) Chevrolet Equinox EV (Figure 5.1) with the parameters provided in Table 5.1 is employed for the experiments. The maintenance, customization, and operation of this test EV are carried out by the highly skilled technicians at the "Mechatronic Vehicle Systems Laboratory" [125] from which the vehicle's specifications are sourced. The vehicle is equipped with four electric motors, with one of these motors installed on each corner of the vehicle. Figure 5.2 illustrates the

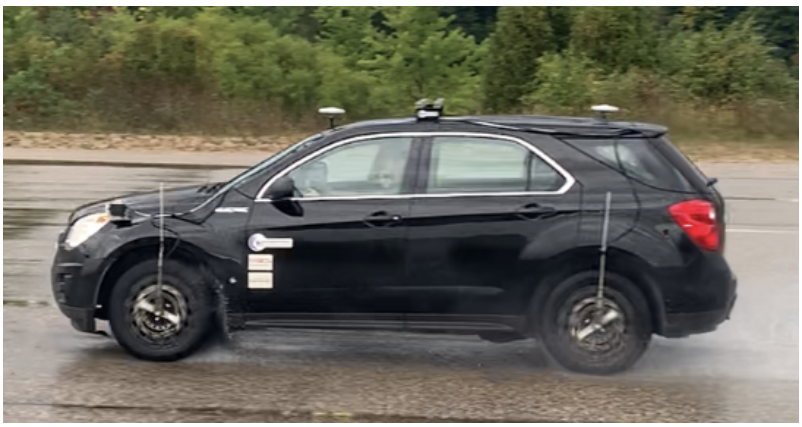

Figure 5.1: The Chevrolet Equinox EV utilized in the experimental tests.

Table 5.1: Parameters of the Chevrolet Equinox EV.

Parameter	Unit	Value	Description
m	[kg]	2270	Vehicle mass
L_{wb}	[m]	2085	Wheelbase
l_f	[m]	1.41	CG distance to front axle
l_r	[m]	1.44	CG distance to rear axle
H_{CG}	[m]	0.72	CG hight
R_e	[m]	0.35	Effective radius of the tires
$C_{\alpha f}$	[N/rad]	130000	Front tires cornering stiffness
$C_{\alpha r}$	[N/rad]	130000	Rear tires cornering stiffness
l_s	[m]	1.63	Front and rear track width
I_z	[kg.m^2]	4600	Vehicle yaw moment of inertia

schematic diagram of the main components utilized in the experiments. With the help of a CAN bus network, the Micro-Autobox is able to interact with sensors and actuators on the vehicle. The Micro-Autobox compiles the designed MPC controller with the weight tuning module. As described in the previous chapters, the MATLAB/Simulink has been

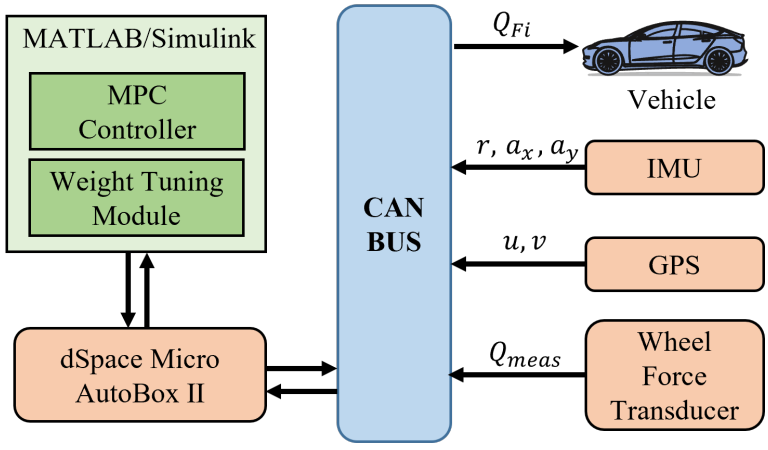

Figure 5.2: The diagram of the setup for experimental tests.

used for the implementation of this MPC controller with the parameters specified in Table 3.2. During the testing process, the Chevrolet Equinox utilizes a six-axis GPS system to measure lateral and longitudinal velocities. The yaw rate of the vehicle is measured by the inertial measurement unit (IMU) sensor. The requested drive torque is delivered to the electric motors via the CAN bus. The actual torques applied to the wheels are measures by using the wheel force transducer.

Two series of tests have been conducted on the test vehicle. In the first set of tests, the vehicle undergoes DLC maneuvers on a dry road, while in the second set, the road is wet. For the initial test under each road condition, the tracking error weights in the objective function of the MPC controller are the default weights that remain constant until the end of the maneuver. Then, these weights are tuned by using the single model and multiple model weight tuning methods and set in the MPC controller in real-time. Finally, the results are compared and discussed.

5.1 MPC Weight Tuning on a Dry road

The first series of tests are conducted on a dry road. The steering wheel angle is under the control of a manual driver which tries to generate the same DLC steering inputs. Nevertheless, there are always some differences in the steering inputs in the trials. Therefore, the graph of steering wheel angle inputs of the driver for three different tests are separately shown in Figure 5.3 to avoid confusion. The top, middle, and bottom graphs present the driver inputs respectively for the test with constant MPC weights, MPC weights tuned with the nominal model, and MPC weights tuned with the multiple model. During the Maneuver, the vehicle is driven off-throttle, and the average longitudinal speed is about $u_0 = 70$kph. In all of the experiments, the weight tuning (either by the nominal model or multiple model approach) is repeated twice. At the end of the final test, the yaw rate, sideslip angle, front wheels torques, and real-time selected MPC controller weights are demonstrated and discussed.

Figure 5.4 compares the vehicle's yaw rate results with the corresponding desired values. When fixed default MPC weights are used, the deviation of the yaw rate from the desired

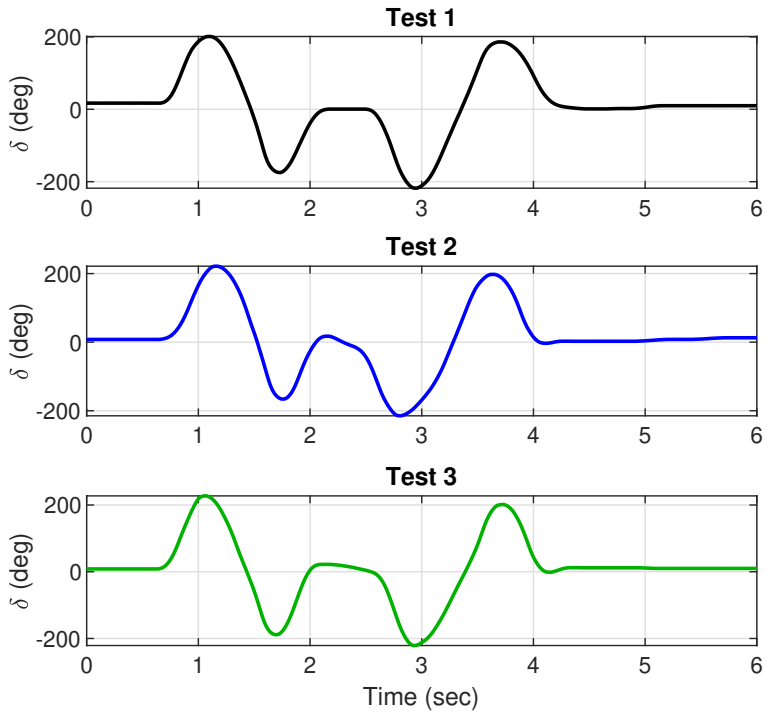

Figure 5.3: Steering wheel angle input of the manual driver on a dry road; top figure: default weights are selected; middle figure: weights are tuned by the nominal prediction model; bottom figure: weights are tuned by the multiple model method.

values is considerable; particularly after $t = 1.5$ sec when the first lane change is about to finish until the end of second lane change. By tuning the MPC controller weights by the nominal model, the RMSE of yaw rate tracking is decreased by about 17%, while by using multiple model, it is reduced by 42%. As depicted in Figure 5.4, the desired yaw rate is better followed by the MPC controller tuned with the multiple model method. Similar improvement can be noticed in the sideslip angle results (Figure 5.5). The vehicle's sideslip angle exceeds 8 degrees during the second lane change when default weights are used. Single model and multiple model weight tuning can respectively improve the RMSE of the sideslip angle tracking by about 18% and 32%.

Figure 5.6 demonstrates the changes in the torques generated by the MPCs with three

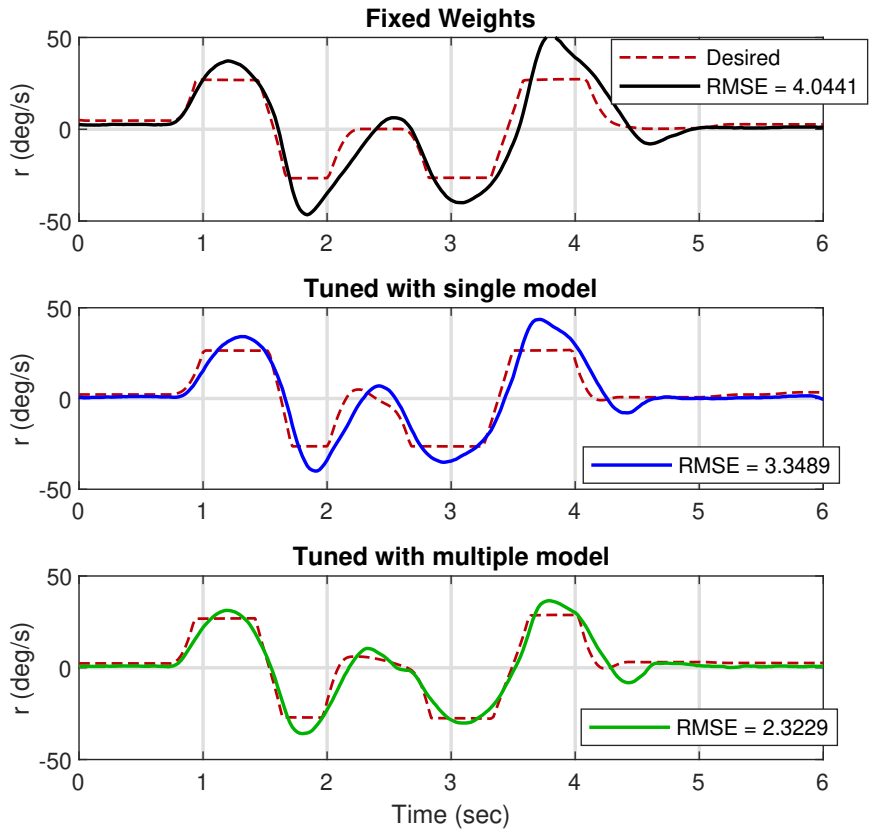

Figure 5.4: Yaw rate responses of the vehicle during the experimental tests on a dry road; the yaw rate results are compared with the desired values when MPC weights are fixed, tuned with the nominal model, and tuned with multiple model.

different weight tuning strategies. The weights selected in real-time under each approach are displayed in Figure 5.7. By examining yaw rate, side slip angle, and weight results (Figures 5.4-5.7), it is apparent that weight changes have resulted in decreasing yaw rates and side slip angle tracking errors. Moreover, the weights that have been tuned by the multiple model approach resulted in better controller performance. Since the multiple model method yields a higher prediction accuracy, the tuned weights make the vehicle more controllable. The proposed weight tuning approach has improved the performance of the MPC controller significantly and the multiple model approach has further enhanced

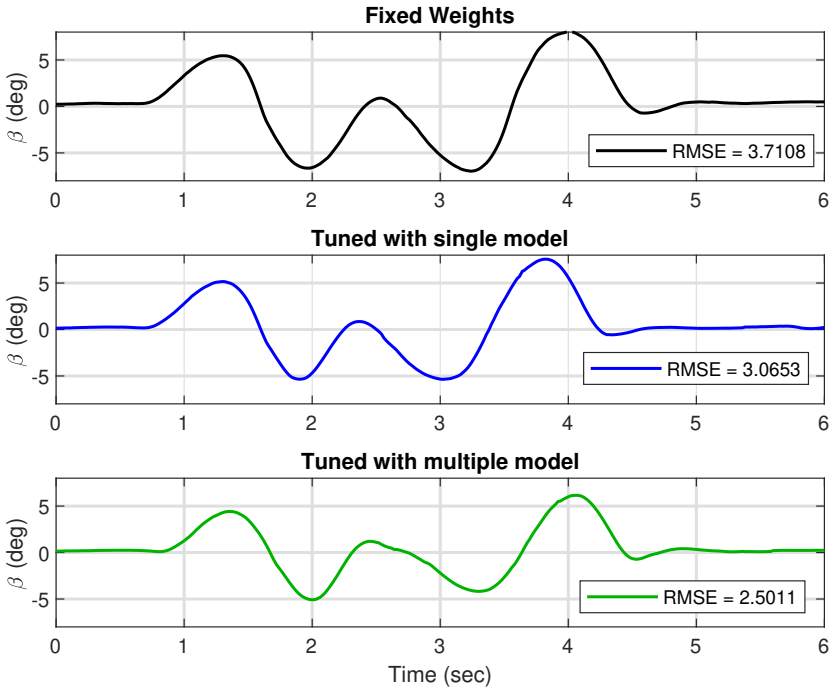

Figure 5.5: Sideslip angle responses of the vehicle during the experimental tests on a dry road; the results are compared when MPC weights are fixed, tuned with the nominal model, and tuned with multiple model.

the controller effectiveness.

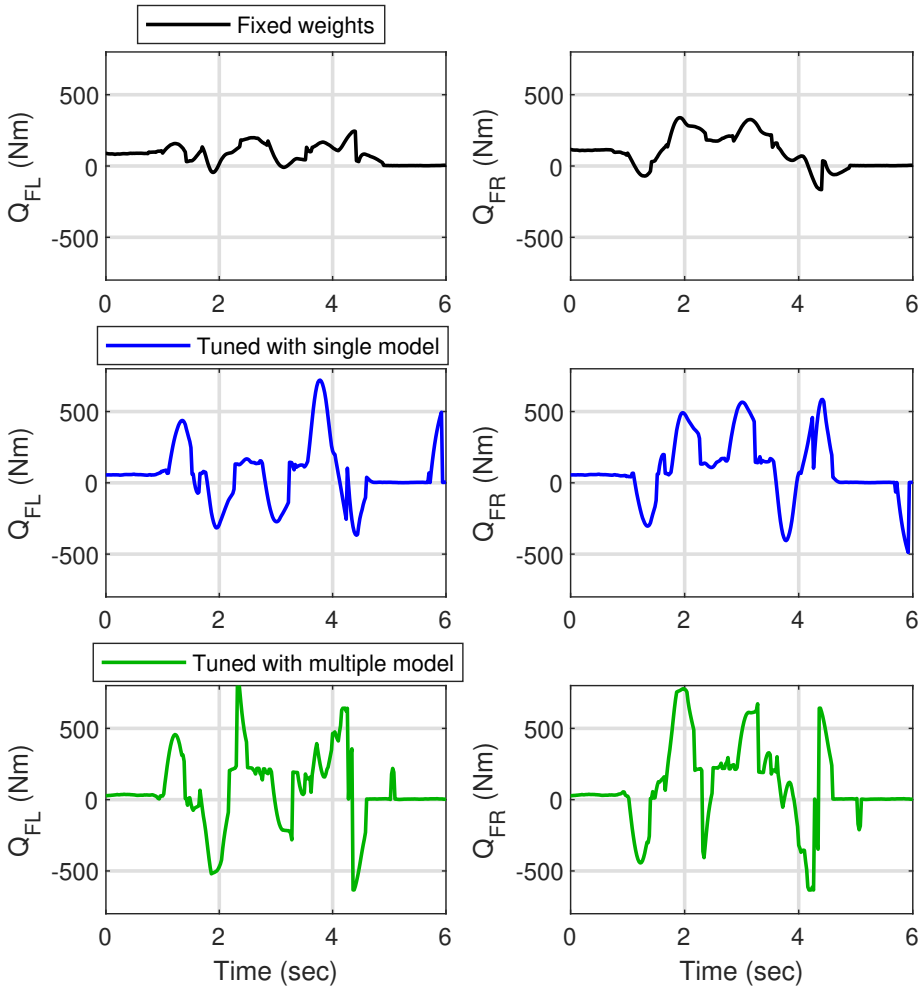

Figure 5.6: Generated front wheel torques during the experimental tests on a dry road; the results are compared when MPC weights are fixed, tuned with the nominal model, and tuned with multiple model.

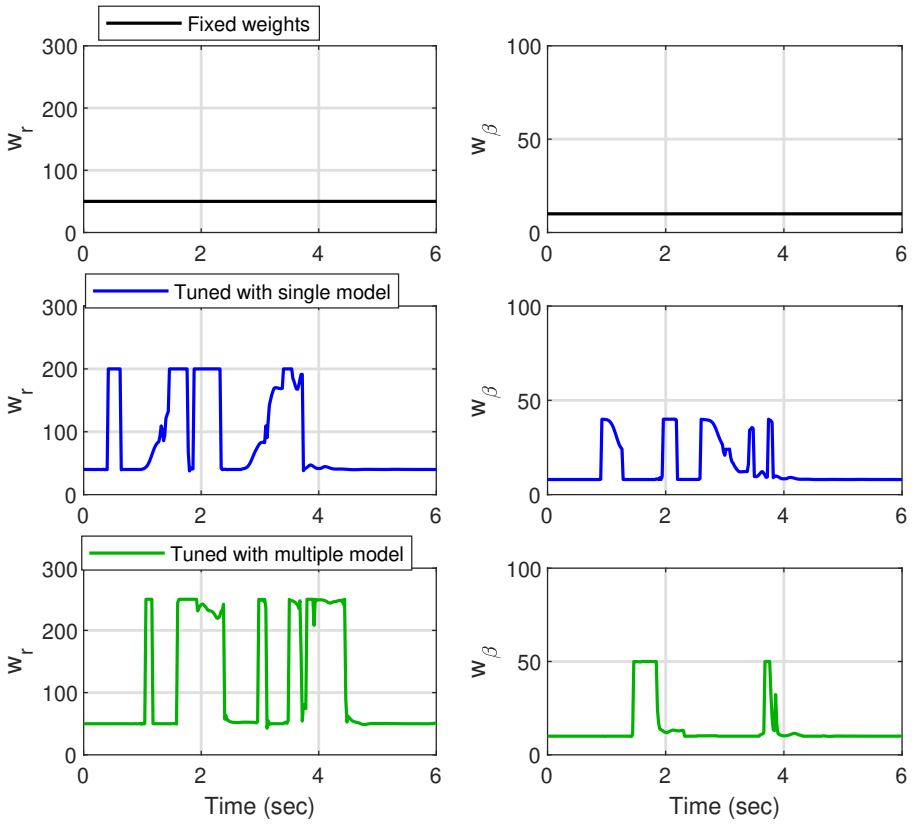

Figure 5.7: Real-time selected weights w_r, w_β during the experimental tests on a dry road; the results are compared when MPC weights are fixed, tuned with the nominal model, and tuned with multiple model.

5.2 MPC Weight Tuning on a Wet road

For the next set of experiments, a wet road is used to evaluate the functionality of the proposed weight tuning approach on a slippery road condition. The road friction coefficient is approximately $\mu = 0.5$. Prior to commencing the primary experiments on the wet road, preliminary DLC tests were conducted at a vehicle speed of $u_0 = 70$kph. It was observed that the tires became completely saturated at an early stage, and that weight tuning had a minimal impact on enhancing the performance of the controller. Hence, it was determined

that lowering the vehicle's average longitudinal speed to $u_0 = 60$kph would facilitate a more thorough assessment of the efficacy of the proposed weight tuning method under wet road conditions. The steering wheel angle is manually controlled by the driver as shown in Figure 5.8. Due to the slippery condition of the road, the inputs from the driver may differ on wet and dry roads (compare with Fig. 5.3).

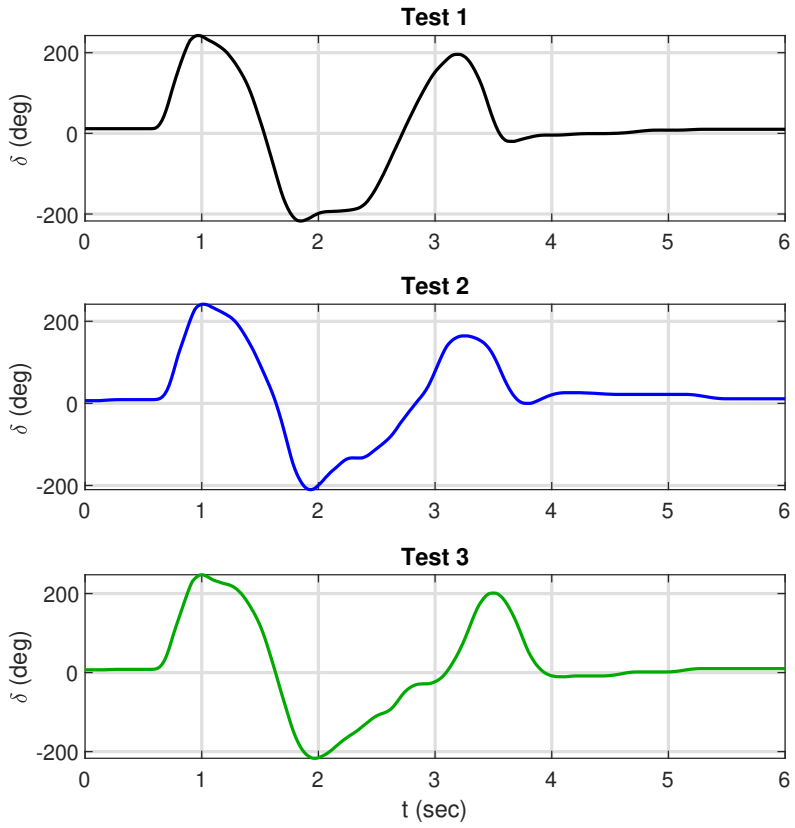

Figure 5.8: Steering wheel angle input of the manual driver on a wet road; top figure: default weights are selected; middle figure: weights are tuned by the nominal prediction model; bottom figure: weights are tuned by the multiple model method.

Similar to the previous tests described in section 5.1, the initial test is performed by the MPC controller with fixed default weights which is followed by weight tuning. As depicted in Figure 5.9, before weight tuning, the yaw rate of the vehicle deviates considerably from

the desired value during the second turning. The sideslip angle becomes larger than $\beta = 9$ degrees (Figure 5.10), which indicates an unstable condition of the vehicle. The single and multiple model weight tuning methods improve the RMSE of yaw rate tracking by 29% and 42%, respectively. Regarding the sideslip angle results, the single model weight tuning reduces the RMSE by 12%, while the multiple model weight tuning is capable of decreasing the RMSE by 40%. It is evident from the sideslip angle results that the multiple model approach has dropped the peak value of $\beta \simeq 9$ degrees at $t = 2.7$ sec to $\beta \simeq 5$ degrees.

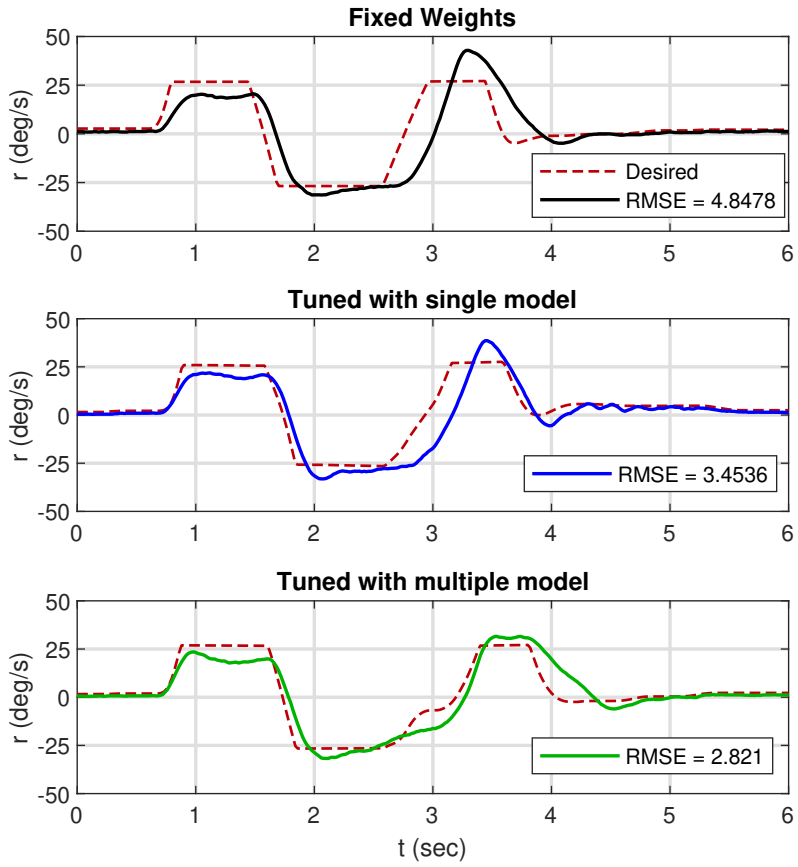

Figure 5.9: Yaw rate responses of the vehicle during the experimental tests on a wet road; the yaw rate results are compared with the desired values when MPC weights are fixed, tuned with the nominal model, and tuned with multiple model.

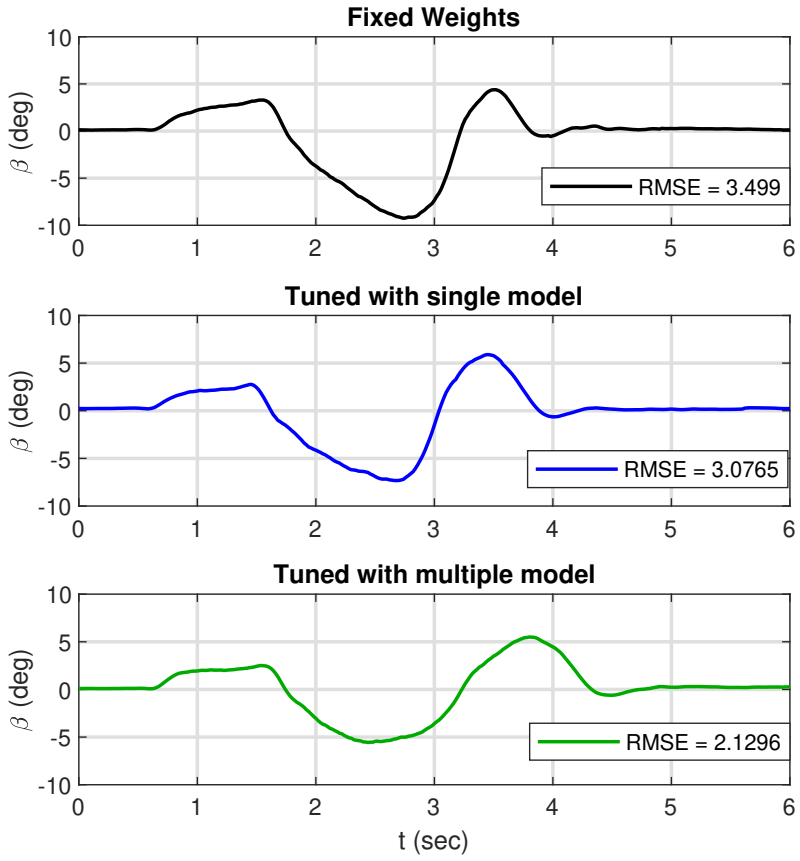

Figure 5.10: Sideslip angle responses of the vehicle during the experimental tests on a wet road; the results are compared when MPC weights are fixed, tuned with the nominal model, and tuned with multiple model.

Due to the change in weights and their corresponding wheel torques, the tracking performance of the MPC controller is improved (Figures 5.11 and 5.12). It has been noted previously that MPC weights directly affect torque generation and, as a result, controller performance. In the initial tests where fixed set of default weights are used in MPC, large tracking errors are noticed which indicate weak performance of MPC. The performance of the controller has been improved slightly after single model weight tuning, but this change has been less significant than the change observed after multiple model weight

tuning. Moreover, the torque result of the former exhibits more fluctuations. Therefore, multiple model weight tuning could better enhance the performance of MPC and decrease the torque fluctuation.

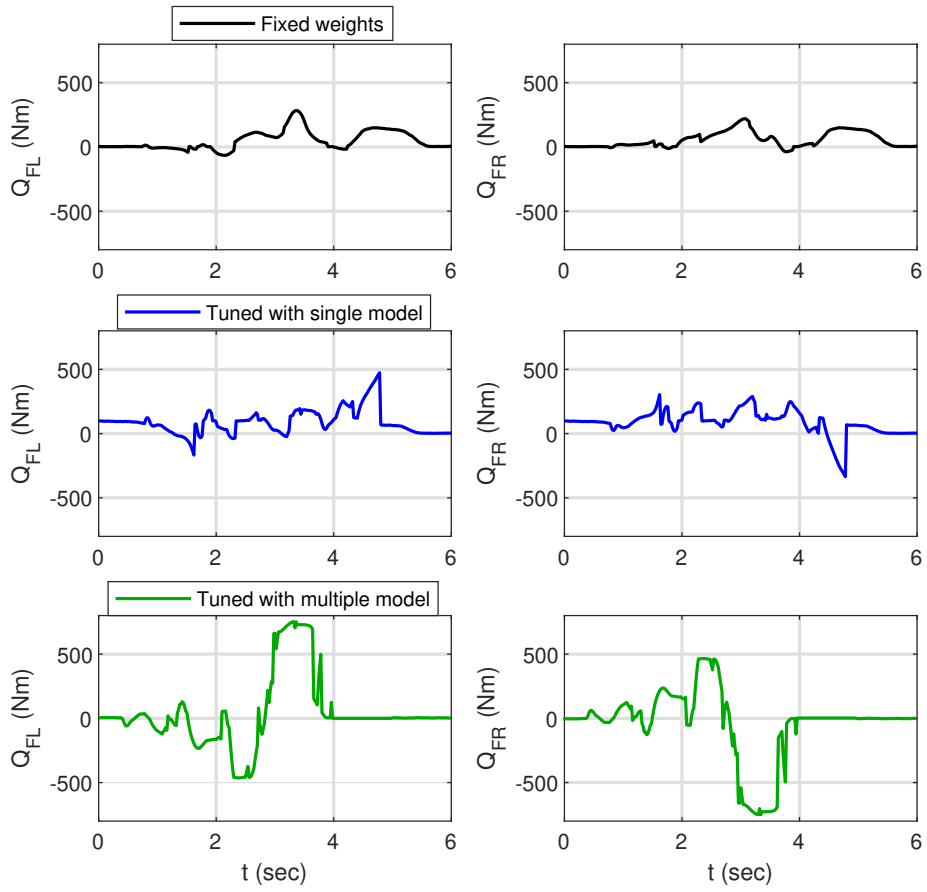

Figure 5.11: Generated front wheel torques during the experimental tests on a wet road; the results are compared when MPC weights are fixed, tuned with the nominal model, and tuned with multiple model.

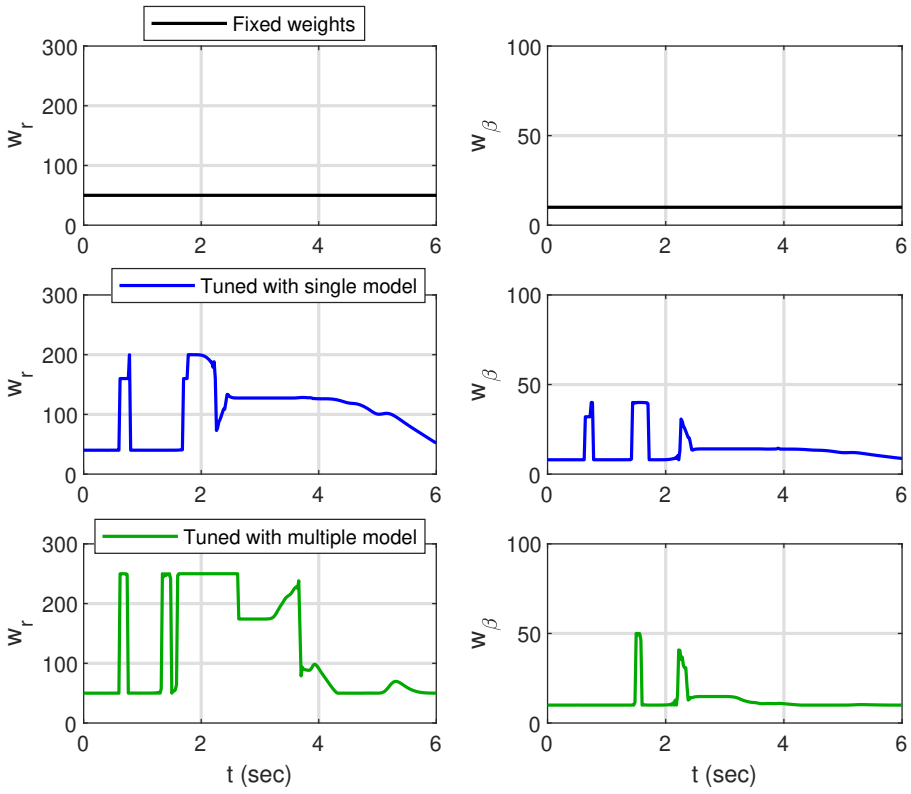

Figure 5.12: Real-time selected weights w_r, w_β during the experimental tests on a wet road; the results are compared when MPC weights are fixed, tuned with the nominal model, and tuned with multiple model.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

In this thesis, an MPC controller weight tuning as well as a real-time learning-based weight selection approach was proposed to improve the tracking performance of MPC controllers. The proposed approach was applied to an MPC controller designed to control the vehicle lateral stability. A weight authentication technique was provided to evaluate the efficacy of the tuned weight in controller's performance. A multiple model MPC controller weight tuning method was employed to address the inaccuracies of the nominal prediction model. Following is a list of this thesis's significant contributions and findings.

The prediction model of the vehicle contains lateral and yaw motions which utilized for designing the MPC controller to control the lateral stability of the vehicle. It was discussed that using fixed set of weight do not necessarily give the optimal controller performance. Therefore, a weight tuning approach was developed for tuning the tracking error weights of the MPC controller's objective function. An error examination criterion was defined as a function of tracking errors during a predefined time interval to determine whether the weights currently used in the controller should be adjusted. The BO method was used to obtain the optimum weights based on the predictions and error examination criterion. It was explained that real-time selection of weights can improve the tracking performance of the controller. Therefore, a learning-based weight selection using GPR was provided.

The proposed weight tuning approach was evaluated through MATLAB/Simulink and CarSim co-simulations. The vehicle underwent DLC maneuvers under different road conditions. The tracking error results indicated the capability of the developed weight tuning technique in improving the tracking performance of the MPC controller. After two rounds of tuning, the RMSE of yaw rate and sideslip angle tracking were respectively decreased by 51% and 33% on a dry road. For a slippery road, the aforementioned errors were decreased by 55% and 54%. It was clearly seen that tuning the weights and real-time selection of them can considerably improve response of the vehicle system.

It was observed that in some vehicle states, adjusting the weights would not considerably affect the response of the vehicle, even if the generated control action significantly changed. It was understood that a weight authentication technique was required, and was proposed to address this issue. A weight authentication criterion was defined to evaluate the adjusted weights as a function of tracking errors in a predefined time interval. The simulations demonstrated that ineffective sharp changes in the adjusted weights as well as the torque spikes were eliminated with the weight authentication process.

From the simulation results, it was noticed that predictions made by the nominal model were not always accurate enough to avoid ineffective weight changes, or sometimes effective weight changes were missed during the weight tuning. It was discussed that instead of a single prediction model, a multiple model prediction approach could address this issue. Therefore, a multiple model MPC controller weight tuning method was proposed to minimize the inaccuracies observed in the state predictions of the nominal prediction model. A weighted summation of different models formed by different parameter values was employed for the prediction purposes of the MPC control and weight tuning. The simulation results demonstrated that the multiple model technique could reduce the yaw rate and sideslip angle prediction errors by about 40% compared to the nominal model.

After performing multiple model weight tuning, it was shown that the MPC controller's tracking errors were decreased when nominal prediction model was replaced by the multiple model both on the dry and slippery road conditions. The results indicated that the variations of MPC weights tuned by the multiple model was smoother than those tuned by the single model. The simulations carried out under slippery road condition showed that even one run of multiple model weight tuning performed better than one run of single

model weight tuning.

The effectiveness of the proposed weight tuning approach was verified by conducting two series of experiments. A Chevrolet Equinox EV was employed for DLC maneuvers under dry and wet road conditions. The deviation of the yaw rate and sideslip angle from the desired values were considerable when fixed default MPC weights were used on the dry road. This deviation of the yaw rate was reduced by 17% and 42% after tuning the weights respectively by the nominal and multiple model methods. Similar improvement was noticed in the sideslip angle results due to the wheel torque adjustment after the weight tuning process. Under harsh DLC maneuvers on a wet road, the multiple model weight tuning could prevent the vehicle from becoming unstable. It can be concluded that the proposed MPC controller weight tuning approach was capable of improving the controller's performance under various road conditions. The best performance was achieved when multiple model technique was utilized due to its higher prediction accuracy.

6.2 Future Work

The objective of this section is to provide some suggestions for further developing the study conducted in this thesis, with the aim of improving the efficiency of the suggested method for adjusting controller weights.

Other than tracking error weights, there are other weights in the objective function of MPC controllers. The elements of weight matrices of control effort R, control action proximity T, and slack variable W_s are considered constant in the proposed weight tuning method. Additionally, the MPC controller has some parameters including sampling time t_s and prediction horizon length N_p that are all predetermined based on some initial trial and errors. The performance of the MPC controller can be improved further by including the above weights and parameters in the tuning process based on the predicted or experienced responses of the system.

References

[1] M. L. Darby and M. Nikolaou, "Mpc: Current practice and challenges," *Control Engineering Practice*, vol. 20, no. 4, pp. 328–342, 2012.

[2] M. D. Rafal and W. F. Stevens, "Discrete dynamic optimization applied to on-line optimal control," *AIChE Journal*, vol. 14, no. 1, pp. 85–91, 1968.

[3] A. Propoi, "Use of linear programming methods for synthesizing sampled-data automatic systems," *Automn. Remote Control*, vol. 24, no. 7, pp. 837–844, 1963.

[4] L. Zadeh and B. Whalen, "On optimal control and linear programming," *IRE Transactions on Automatic Control*, vol. 7, no. 4, pp. 45–46, 1962.

[5] M. Morari and J. H. Lee, "Model predictive control: past, present and future," *Computers & Chemical Engineering*, vol. 23, no. 4-5, pp. 667–682, 1999.

[6] S. J. Qin and T. A. Badgwell, "A survey of industrial model predictive control technology," *Control engineering practice*, vol. 11, no. 7, pp. 733–764, 2003.

[7] B. W. Bequette, "Non-linear model predictive control: a personal retrospective," *The canadian journal of chemical engineering*, vol. 85, no. 4, pp. 408–415, 2007.

[8] C. E. Garcia and M. Morari, "Internal model control. a unifying review and some new results," *Industrial & Engineering Chemistry Process Design and Development*, vol. 21, no. 2, pp. 308–323, 1982.

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

[10] R. Findeisen, F. Allgöwer, and L. T. Biegler, *Assessment and future directions of nonlinear model predictive control*. Springer, 2007, vol. 358, no. 7.

[11] B. Kouvaritakis and M. Cannon, *Non-linear Predictive Control: theory and practice*. Iet, 2001, no. 61.

[12] C. E. Garcia, D. M. Prett, and M. Morari, "Model predictive control: Theory and practice—a survey," *Automatica*, vol. 25, no. 3, pp. 335–348, 1989.

[13] A. Mesbah, "Stochastic model predictive control: An overview and perspectives for future research," *IEEE Control Systems Magazine*, vol. 36, no. 6, pp. 30–44, 2016.

[14] S. Di Cairano, D. Bernardini, A. Bemporad, and I. V. Kolmanovsky, "Stochastic mpc with learning for driver-predictive vehicle control and its application to hev energy management," *IEEE Transactions on Control Systems Technology*, vol. 22, no. 3, pp. 1018–1031, 2013.

[15] A. Mesbah, S. Streif, R. Findeisen, and R. D. Braatz, "Stochastic nonlinear model predictive control with probabilistic constraints," in *2014 American control conference*. IEEE, 2014, pp. 2413–2419.

[16] A. Ferramosca, J. B. Rawlings, D. Limón, and E. F. Camacho, "Economic mpc for a changing economic criterion," in *49th IEEE Conference on Decision and Control (CDC)*. IEEE, 2010, pp. 6131–6136.

[17] C. R. Touretzky and M. Baldea, "Integrating scheduling and control for economic mpc of buildings with energy storage," *Journal of Process Control*, vol. 24, no. 8, pp. 1292–1300, 2014.

[18] M. Heidarinejad, J. Liu, and P. D. Christofides, "State-estimation-based economic model predictive control of nonlinear systems," *Systems & Control Letters*, vol. 61, no. 9, pp. 926–935, 2012.

[19] C. R. Cutler and B. L. Ramaker, "Dynamic matrix control: A computer control algorithm," in *joint automatic control conference*, no. 17, 1980, p. 72.

[20] J. L. Garriga and M. Soroush, "Model predictive control tuning methods: A review," *Industrial & Engineering Chemistry Research*, vol. 49, no. 8, pp. 3505–3515, 2010.

[21] J. Richalet, A. Rault, J. Testud, and J. Papon, "Model predictive heuristic control: Applications to industrial processes," *Automatica*, vol. 14, no. 5, pp. 413–428, 1978.

[22] N. J. Garrett and M. C. Best, "Model predictive driving simulator motion cueing algorithm with actuator-based constraints," *Vehicle System Dynamics*, vol. 51, no. 8, pp. 1151–1172, 2013.

[23] F. Maran, "Model-based control techniques," Ph.D. dissertation, University of Padova, 2013.

[24] M. B. Poodeh, S. Eshtehardiha, A. Kiyoumarsi, and M. Ataei, "Optimizing lqr and pole placement to control buck converter by genetic algorithm," in *2007 International Conference on Control, Automation and Systems*. IEEE, 2007, pp. 2195–2200.

[25] S. T. Branch, "Optimal design of lqr weighting matrices based on intelligent optimization methods," *International Journal of Intelligent Information Processing*, vol. 2, no. 1, pp. 63–74, 2011.

[26] M. Mamdouh, M. Abido, and Z. Hamouz, "Weighting factor selection techniques for predictive torque control of induction motor drives: A comparison study," *Arabian Journal for Science and Engineering*, vol. 43, no. 2, pp. 433–445, 2018.

[27] P. Bagheri and A. K. Sedigh, "Analytical approach to tuning of model predictive control for first-order plus dead time models," *IET Control Theory & Applications*, vol. 7, no. 14, pp. 1806–1817, 2013.

[28] P. Bagheri and A. Khaki-Sedigh, "Closed form tuning equations for model predictive control of first-order plus fractional dead time models," *International Journal of Control, Automation and Systems*, vol. 13, no. 1, pp. 73–80, 2015.

[29] T. Gholaminejad, A. Khaki-Sedigh, and P. Bagheri, "Adaptive tuning of model predictive control based on analytical results," in *2016 4th International Conference on Control, Instrumentation, and Automation (ICCIA)*. IEEE, 2016, pp. 226–232.

[30] P. Bagheri and A. Khaki-Sedigh, "An analytical tuning approach to multivariable model predictive controllers," *Journal of Process Control*, vol. 24, no. 12, pp. 41–54, 2014.

[31] Q. N. Tran, L. Özkan, and A. Backx, "Generalized predictive control tuning by controller matching," *Journal of Process Control*, vol. 25, pp. 1–18, 2015.

[32] K. Belda, "On-line parameter tuning of model predictive control," *IFAC Proceedings Volumes*, vol. 44, no. 1, pp. 5489–5494, 2011.

[33] P. Bagheri and A. K. Sedigh, "Robust tuning of dynamic matrix controllers for first order plus dead time models," *Applied Mathematical Modelling*, vol. 39, no. 22, pp. 7017–7031, 2015.

[34] A. S. Yamashita, P. M. Alexandre, A. C. Zanin, and D. Odloak, "Reference trajectory tuning of model predictive control," *Control Engineering Practice*, vol. 50, pp. 1–11, 2016.

[35] J. G. Burgos, C. A. L. Martínez, R. van de Molengraft, and M. Steinbuch, "Frequency domain tuning method for unconstrained linear output feedback model predictive control," *IFAC Proceedings Volumes*, vol. 47, no. 3, pp. 7455–7460, 2014, 19th IFAC World Congress.

[36] D. W. Clarke, C. Mohtadi, and P. S. Tuffs, "Generalized predictive control—part i. the basic algorithm," *Automatica*, vol. 23, no. 2, pp. 137–148, 1987.

[37] R. Shridhar and D. J. Cooper, "A tuning strategy for unconstrained siso model predictive control," *Industrial & Engineering Chemistry Research*, vol. 36, no. 3, pp. 729–746, 1997.

[38] ——, "A tuning strategy for unconstrained multivariable model predictive control," *Industrial & engineering chemistry research*, vol. 37, no. 10, pp. 4003–4016, 1998.

[39] A. R. Neshasteriz, A. Khaki-Sedigh, and H. Sadjadian, "An analysis of variance approach to tuning of generalized predictive controllers for second order plus dead time models," in *IEEE ICCA 2010*. IEEE, 2010, pp. 1059–1064.

[40] P. Bagheri and A. Khaki-Sedigh, "Tuning of dynamic matrix controller for fopdt models using analysis of variance," *IFAC Proceedings Volumes*, vol. 44, no. 1, pp. 12 319–12 324, 2011.

[41] Z. Ebrahimi, P. Bagheri, and A. Khaki-Sedigh, "Tuning of generalized predictive controllers for first order plus dead time models based on anova," in *2015 23rd Iranian Conference on Electrical Engineering*. IEEE, 2015, pp. 928–933.

[42] J. K. Huusom, N. K. Poulsen, S. B. Jørgensen, and J. B. Jørgensen, "Tuning siso offset-free model predictive control based on arx models," *Journal of Process Control*, vol. 22, no. 10, pp. 1997–2007, 2012.

[43] F. Lozano Santamaria and J. M. Gomez, "An algorithm for tuning nmpc controllers with application to chemical processes," *Industrial & Engineering Chemistry Research*, vol. 55, no. 34, pp. 9215–9228, 2016.

[44] M. Luzi, M. Vaccarini, and M. Lemma, "A tuning methodology of model predictive control design for energy efficient building thermal control," *Journal of Building Engineering*, vol. 21, pp. 28–36, 2019.

[45] S. Lee, J. Joe, P. Karava, I. Bilionis, and A. Tzempelikos, "Implementation of a self-tuned hvac controller to satisfy occupant thermal preferences and optimize energy use," *Energy and Buildings*, vol. 194, pp. 301–316, 2019.

[46] A. S. Yamashita, A. C. Zanin, and D. Odloak, "Tuning the model predictive control of a crude distillation unit," *ISA transactions*, vol. 60, pp. 178–190, 2016.

[47] M. Turki, N. Langlois, and A. Yassine, "A practical tuning approach for multivariable model predictive control," in *2017 13th IEEE International Conference on Control & Automation (ICCA)*. IEEE, 2017, pp. 1107–1112.

[48] M. Vallerio, J. Van Impe, and F. Logist, "Tuning of nmpc controllers via multi-objective optimisation," *Computers & Chemical Engineering*, vol. 61, pp. 38–50, 2014.

[49] G. Shah and S. Engell, "Tuning mpc for desired closed-loop performance for siso systems," *18th Mediterranean Conference on Control and Automation, MED'10*, pp. 628–633, 2010.

[50] ——, "Tuning mpc for desired closed-loop performance for mimo systems," in *Proceedings of the 2011 American Control Conference*. IEEE, 2011, pp. 4404–4409.

[51] A. Yamashita, A. Zanin, and D. Odloak, "Tuning of model predictive control with multi-objective optimization," *Brazilian Journal of Chemical Engineering*, vol. 33, pp. 333–346, 2016.

[52] S. Abrashov, T. B. Airimitoaie, P. Lanusse, F. Aioun, R. Malti, X. Moreau, and F. Guillemard, "Model predictive control tuning: Methods and issues. application to steering wheel position control," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 11 331–11 336, 2017.

[53] J. E. W. Santos, J. O. Trierweiler, and M. Farenzena, "Model predictive control tuning strategy for non-square systems and range controlled variables based on multi-scenarios approach," *Industrial & Engineering Chemistry Research*, vol. 56, no. 40, pp. 11 496–11 506, 2017.

[54] M. Romero, I. Tejado, A. De Madrid, and B. Vinagre, "Tuning predictive controllers with optimization: Application to gpc and fgpc," *IFAC Proceedings Volumes*, vol. 44, no. 1, pp. 10 824–10 829, 2011.

[55] D. H. Olesen, J. K. Huusom, and J. B. Jørgensen, "A tuning procedure for arx-based mpc of multivariate processes," in *2013 American Control Conference*. IEEE, 2013, pp. 1721–1726.

[56] ——, "Optimization based tuning approach for offset free mpc," in *10th European Workshop on Advanced Control and Diagnosis*. Technical University of Denmark (DTU), 2012.

[57] M. Francisco, P. Vega, and S. Revollar, "Model predictive control of bsm1 benchmark of wastewater treatment process: a tuning procedure," in *2011 50th IEEE Conference*

on Decision and Control and European Control Conference. IEEE, 2011, pp. 7057–7062.

[58] W. Stróżecki, N. A. Oufroukh, Y. Kebbati, D. Ichalal, and S. Mammar, "Automatic tuning of mpc for autonomous vehicle using Bayesian optimization," in *2021 IEEE International Conference on Networking, Sensing and Control (ICNSC)*, vol. 1. IEEE, 2021, pp. 1–6.

[59] Q. Lu, R. Kumar, and V. M. Zavala, "Mpc controller tuning using Bayesian optimization techniques," *arXiv preprint arXiv:2009.14175*, 2020.

[60] A. Marco, P. Hennig, J. Bohg, S. Schaal, and S. Trimpe, "Automatic lqr tuning based on Gaussian process global optimization," in *2016 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2016, pp. 270–277.

[61] A. Al-Ghazzawi, E. Ali, A. Nouh, and E. Zafiriou, "On-line tuning strategy for model predictive controllers," *Journal of Process Control*, vol. 11, no. 3, pp. 265–284, 2001.

[62] G. A. Bunin, F. F. Tirado, G. François, and D. Bonvin, "Run-to-run mpc tuning via gradient descent," in *Computer aided chemical engineering*. Elsevier, 2012, vol. 30, pp. 927–931.

[63] V. Exadaktylos and C. J. Taylor, "Multi-objective performance optimisation for model predictive control by goal attainment," *International Journal of Control*, vol. 83, no. 7, pp. 1374–1386, 2010.

[64] B. Zarrouki, V. Klös, N. Heppner, S. Schwan, R. Ritschel, and R. Voßwinkel, "Weights-varying mpc for autonomous vehicle guidance: a deep reinforcement learning approach," in *2021 European Control Conference (ECC)*. IEEE, 2021, pp. 119–125.

[65] B. Zarrouki, "Reinforcement learning of model predictive control parameters for autonomous vehicle guidance," *Master's thesis*, 2020.

[66] M. Mehndiratta, E. Camci, and E. Kayacan, "Automated tuning of nonlinear model predictive controller by reinforcement learning," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 3016–3021.

[67] V.-A. Le and A. A. Malikopoulos, "Optimal weight adaptation of model predictive control for connected and automated vehicles in mixed traffic with Bayesian optimization," *arXiv preprint arXiv:2210.00700*, 2022.

[68] P. T. Jardine, S. Givigi, and S. Yousefi, "Parameter tuning for prediction-based quadcopter trajectory planning using learning automata," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 2341–2346, 2017.

[69] P. T. Jardine, S. N. Givigi, and S. Yousefi, "Experimental results for autonomous model-predictive trajectory planning tuned with machine learning," in *2017 Annual IEEE International Systems Conference (SysCon)*. IEEE, 2017, pp. 1–7.

[70] A. S. Ira, I. Shames, C. Manzie, R. Chin, D. Nešić, H. Nakada, and T. Sano, "A machine learning approach for tuning model predictive controllers," in *2018 15th International Conference on Control, Automation, Robotics and Vision (ICARCV)*. IEEE, 2018, pp. 2003–2008.

[71] A. Mohammadi, H. Asadi, S. Mohamed, K. Nelson, and S. Nahavandi, "Multiobjective and interactive genetic algorithms for weight tuning of a model predictive control-based motion cueing algorithm," *IEEE transactions on cybernetics*, vol. 49, no. 9, pp. 3471–3481, 2018.

[72] V. Ramasamy, R. K. Sidharthan, R. Kannan, and G. Muralidharan, "Optimal tuning of model predictive controller weights using genetic algorithm with interactive decision tree for industrial cement kiln process," *Processes*, vol. 7, no. 12, p. 938, 2019.

[73] D. Gorni and A. Visioli, "Genetic algorithms based reference signal determination for temperature control of residential buildings," *Applied Sciences*, vol. 8, no. 11, p. 2129, 2018.

[74] J. Van der Lee, W. Svrcek, and B. Young, "A tuning algorithm for model predictive controllers based on genetic algorithms and fuzzy decision making," *ISA transactions*, vol. 47, no. 1, pp. 53–59, 2008.

[75] C. Wongsathan and C. Sirima, "Application of ga to design lqr controller for an inverted pendulum system," in *2008 IEEE International Conference on Robotics and Biomimetics*. IEEE, 2009, pp. 951–954.

[76] M. Johnson and M. Grimble, "Recent trends in linear optimal quadratic multivariable control system design," in *IEE proceedings D (control theory and applications)*, vol. 134, no. 1. IET, 1987, pp. 53–71.

[77] Y. Li, J. Liu, and Y. Wang, "Design approach of weighting matrices for lqr based on multi-objective evolution algorithm," in *2008 international conference on information and automation*. IEEE, 2008, pp. 1188–1192.

[78] M. R. Bonyadi and Z. Michalewicz, "Particle swarm optimization for single objective continuous space problems: a review," *Evolutionary computation*, vol. 25, no. 1, pp. 1–54, 2017.

[79] G. A. N. Júnior, M. A. Martins, and R. Kalid, "A pso-based optimal tuning strategy for constrained multivariable predictive controllers with model uncertainty," *ISA transactions*, vol. 53, no. 2, pp. 560–567, 2014.

[80] M. Morari, "Design of resilient processing plants—iii: A general framework for the assessment of dynamic resilience," *Chemical Engineering Science*, vol. 38, no. 11, pp. 1881–1891, 1983.

[81] A. Thamallah, A. Sakly, and F. M'sahli, "Constrained multiobjective pso and ts fuzzy models for predictive control," *Turkish Journal of Electrical Engineering and Computer Sciences*, vol. 26, no. 6, pp. 3239–3257, 2018.

[82] L. Feng, F. Yang, W. Zhang, and H. Tian, "Model predictive control of duplex inlet and outlet ball mill system based on parameter adaptive particle swarm optimization," *Mathematical Problems in Engineering*, vol. 2019, 2019.

[83] P. Mc Namara, R. R. Negenborn, B. De Schutter, and G. Lightbody, "Weight optimisation for iterative distributed model predictive control applied to power networks," *Engineering Applications of Artificial Intelligence*, vol. 26, no. 1, pp. 532–543, 2013.

[84] R. Suzuki, F. Kawai, C. Nakazawa, T. Matsui, and E. Aiyoshi, "Parameter optimization of model predictive control by pso," *Electrical Engineering in Japan*, vol. 178, no. 1, pp. 40–49, 2012.

[85] S. Di Cairano and A. Bemporad, "Model predictive control tuning by controller matching," *IEEE Transactions on Automatic Control*, vol. 55, no. 1, pp. 185–190, 2009.

[86] M. Soliman, O. Malik, and D. T. Westwick, "Multiple model predictive control for wind turbines with doubly fed induction generators," *IEEE Transactions on Sustainable Energy*, vol. 2, no. 3, pp. 215–225, 2011.

[87] P. Falcone, M. Tufo, F. Borrelli, J. Asgari, and H. E. Tseng, "A linear time varying model predictive control approach to the integrated vehicle dynamics control problem in autonomous systems," in *2007 46th IEEE Conference on Decision and Control*. IEEE, 2007, pp. 2980–2985.

[88] K. Alexis, G. Nikolakopoulos, and A. Tzes, "Switching model predictive attitude control for a quadrotor helicopter subject to atmospheric disturbances," *Control Engineering Practice*, vol. 19, no. 10, pp. 1195–1207, 2011.

[89] I. Masar and E. Stöhr, "Gain-scheduled lqr-control for an autonomous airship," in *18th International Conference on Process Control*, 2011, pp. 14–17.

[90] M. X. Huang and I. Kolmanovsky, "Switch gain scheduled explicit model predictive control of diesel engines," Sep. 19 2017, uS Patent 9,765,621.

[91] R. G. Patel and J. J. Trivedi, "Sagd real-time production optimization using adaptive and gain-scheduled model-predictive-control: A field case study," in *SPE Western Regional Meeting*. OnePetro, 2017.

[92] M. B. Shadmand, R. S. Balog, and H. A. Rub, "Auto-tuning the cost function weight factors in a model predictive controller for a matrix converter var compensator," in *2015 IEEE Energy Conversion Congress and Exposition (ECCE)*. IEEE, 2015, pp. 3807–3814.

[93] M. Abroshan, R. Hajiloo, E. Hashemi, and A. Khajepour, "Model predictive-based tractor-trailer stabilisation using differential braking with experimental verification," *Vehicle system dynamics*, vol. 59, no. 8, pp. 1190–1213, 2021.

[94] C. Zhao, W. Xiang, and P. Richardson, "Vehicle lateral control and yaw stability control through differential braking," in *2006 IEEE international symposium on industrial electronics*, vol. 1. IEEE, 2006, pp. 384–389.

[95] S. A. Nahidi, "Reconfigurable integrated vehicle stability control using optimal control techniques," 2017.

[96] A. Nahidi, A. Kasaiezadeh, S. Khosravani, A. Khajepour, S.-K. Chen, and B. Litkouhi, "Modular integrated longitudinal and lateral vehicle stability control for electric vehicles," *Mechatronics*, vol. 44, pp. 60–70, 2017.

[97] D. Kasinathan, A. Kasaiezadeh, A. Wong, A. Khajepour, S.-K. Chen, and B. Litkouhi, "An optimal torque vectoring control for vehicle applications via real-time constraints," *IEEE Transactions on Vehicular Technology*, vol. 65, no. 6, pp. 4368–4378, 2015.

[98] M. Jalali, E. Hashemi, A. Khajepour, S.-k. Chen, and B. Litkouhi, "Integrated model predictive control and velocity estimation of electric vehicles," *Mechatronics*, vol. 46, pp. 84–100, 2017.

[99] E. Siampis, E. Velenis, S. Gariuolo, and S. Longo, "A real-time nonlinear model predictive control strategy for stabilization of an electric vehicle at the limits of handling," *IEEE Transactions on Control Systems Technology*, vol. 26, no. 6, pp. 1982–1994, 2017.

[100] M. Athans, D. Castanon, K.-P. Dunn, C. Greene, W. Lee, N. Sandell, and A. Willsky, "The stochastic control of the f-8c aircraft using a multiple model adaptive control (mmac) method–part i: Equilibrium flight," *IEEE Transactions on Automatic Control*, vol. 22, no. 5, pp. 768–780, 1977.

[101] D. Lainiotis, "A unifying framework for adaptive systems, i: Estimation, ii," *Proceedings of the IEEE*, vol. 64, no. 8, pp. 1126–1134, 1976.

[102] J. Hespanha, D. Liberzon, A. Stephen Morse, B. D. Anderson, T. S. Brinsmead, and F. De Bruyne, "Multiple model adaptive control. part 2: switching," *International Journal of Robust and Nonlinear Control: IFAC-Affiliated Journal*, vol. 11, no. 5, pp. 479–496, 2001.

[103] J. P. Hespanha, D. Liberzon, and A. S. Morse, "Overcoming the limitations of adaptive control by means of logic-based switching," *Systems & control letters*, vol. 49, no. 1, pp. 49–65, 2003.

[104] K. S. Narendra and J. Balakrishnan, "Adaptive control using multiple models," *IEEE transactions on automatic control*, vol. 42, no. 2, pp. 171–187, 1997.

[105] ——, "Improving transient response of adaptive control systems using multiple models and switching," *IEEE Transactions on automatic control*, vol. 39, no. 9, pp. 1861–1866, 1994.

[106] S. Solmaz, M. Akar, and R. Shorten, "Online center of gravity estimation in automotive vehicles using multiple models and switching," in *2006 9th International Conference on Control, Automation, Robotics and Vision*. IEEE, 2006, pp. 1–7.

[107] S. Solmaz, M. Akar, R. Shorten, and J. Kalkkuhl, "Real-time multiple-model estimation of centre of gravity position in automotive vehicles," *Vehicle System Dynamics*, vol. 46, no. 9, pp. 763–788, 2008.

[108] K. Narendra and Z. Han, "A new approach to adaptive control using multiple models," *International Journal of adaptive Control and signal processing*, vol. 26, no. 8, pp. 778–799, 2012.

[109] Z. Han and K. S. Narendra, "New concepts in adaptive control using multiple models," *IEEE Transactions on Automatic Control*, vol. 57, no. 1, pp. 78–89, 2011.

[110] K. Narendra, "Hierarchical adaptive control of rapidly time-varying systems using multiple models," in *Control of Complex Systems*. Elsevier, 2016, pp. 33–66.

[111] H. Zengin, "Multiple-model robust adaptive vehicle motion control," Ph.D. dissertation, University of Waterloo, 2019.

[112] H. Zengin, N. Zengin, B. Fidan, and A. Khajepour, "Blending based multiple-model adaptive control of multivariable systems with application to lateral vehicle motion control," *European Journal of Control*, vol. 58, pp. 1–10, 2021.

[113] H. B. Pacejka, "Chapter 3 - theory of steady-state slip force and moment generation," in *Tire and Vehicle Dynamics*, 3rd ed. Oxford: Butterworth-Heinemann, 2012, pp. 87–147.

[114] L. C. Westphal, "Conversions of continuous time to discrete time models," in *Handbook of Control Systems Engineering*, 2nd ed. Boston, MA: Springer US, 2001, pp. 285–303.

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

[116] W.-H. Chen, "Stability analysis of classic finite horizon model predictive control," *International Journal of Control, Automation and Systems*, vol. 8, no. 2, pp. 187–197, 2010.

[117] S. Valluri and V. Kapila, "Stability analysis for linear/nonlinear model predictive control of constrained processes," in *Proceedings of the 1998 American Control Conference. ACC (IEEE Cat. No. 98CH36207)*, vol. 3. IEEE, 1998, pp. 1679–1683.

[118] H. J. Ferreau, C. Kirches, A. Potschka, H. G. Bock, and M. Diehl, "qpOASES: A parametric active-set algorithm for quadratic programming," *Mathematical Programming Computation*, vol. 6, no. 4, pp. 327–363, 2014.

[119] C. E. Rasmussen, "Gaussian processes in machine learning," in *Summer school on machine learning*. Springer, 2003, pp. 63–71.

[120] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. De Freitas, "Taking the human out of the loop: A review of Bayesian optimization," *Proceedings of the IEEE*, vol. 104, no. 1, pp. 148–175, 2015.

[121] R. Calandra, A. Seyfarth, J. Peters, and M. P. Deisenroth, "Bayesian optimization for learning gaits under uncertainty," *Annals of Mathematics and Artificial Intelligence*, vol. 76, no. 1, pp. 5–23, 2016.

[122] The MathWorks, Inc., "MATLAB/Simulink," Natick, MA 01760, USA, 2022, *Version R2022a (9.12.0.1975300)*.

[123] Mechanical Simulation Corporation, "CarSim," Ann Arbor, MI 48108, USA, 2019, *Version 2019.1*.

[124] K. S. Narendra and Z. Han, "The changing face of adaptive control: the use of multiple models," *Annual reviews in control*, vol. 35, no. 1, pp. 1–12, 2011.

[125] Mechatronic Vehicle Systems Laboratory. (2023) https://uwaterloo.ca/mechatronic-vehicle-systems-lab/. Accessed: April 4, 2023.