MPC for Off-Road Vehicle Trajectory Tracking

by

Steven Tuer

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Mechanical and Mechatronics Engineering

Waterloo, Ontario, Canada, 2024

© Steven Tuer 2024

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

This thesis explores the trajectory tracking control of autonomous off-road vehicles. The research aims to create a suitable trajectory tracking controller for use in industries such as mining, agriculture, and material transport, where enhancing safety and efficiency is paramount. With these applications comes the need to translate across varying off-road conditions. The complexity of off-road environments poses significant challenges, including varying bank and inclination angles, changing traction conditions, vehicle payloads, and complex terrain-tire dynamics. Furthermore, the need for real-time performance mandates that the controller be efficient, as traditional Non-linear Model Predictive Control (NL-MPC) is too computationally intensive for practical and cost efficient use.

To tackle these challenges, a coupled controller for longitudinal and lateral control has been developed using a dual-track vehicle with a linear tire model. This physics-based approach incorporates road angles into the formulation to account for significant bank and inclination angles. Simulation results in off-road scenarios indicate that Road Angle Model Predictive Control (RA-MPC) shows potential for improving trajectory tracking. However, in practical applications, accurately estimating these angles remains difficult due to the varying planes on which the tires operate. This direct modelling approach also limits the generalizability of the controller in off-road conditions as other sources of unmodelled dynamics are ignored.

To enhance the controller's performance in a broader way, a separate method of compensating for off-road modelling complexities through the use learning methods is explored. Gaussian Process Regression (GPR) is employed to improve tracking performance through data-driven modelling of complex off-road dynamics. While this thesis focuses on the integration and inital proof of concept using learning to augment the MPC formulation, the results demonstrate that GPR-MPC can effectively compensate for path inclination and bank angles as well as other sources of unmodelled dynamics. Notably, GPR-MPC excels in low friction (low μ) scenarios where there are significant parameter mismatches in the physics-driven MPC formulation.

Acknowledgments

First and foremost, I would like to thank Professor Amir Khajepour for his leadership, technical expertise, and mentorship during my time with the Mechatronic Vehicle Systems Lab at the University of Waterloo. I would not be where I am today without his guidance over the last two years. It has been a privilege to learn from many in the lab and contribute meaningfully to projects.

Secondly, I would like to thank the members of the MVS Lab that I have had the privilege to learn from as co-workers throughout my time in the lab. Particularly, I would like to thank Ted Ecclestone who mentored me during my first year in the lab. His mentorship in my first year set me on a path to success that I am truly grateful for. I would also like to thank MVS lab members Pouya Panahandeh, Ahmad Alghooneh, Chen Sun, Ruihe Zhang, and Minghao Ning for their mentorship and collaboration on the WATonoBus project. Additionally, I would like to thank MVS technicians Jeff Graansma, Mike Duthie, and Aaron Sherratt for their technical support in particular with the WATonoBus and WATonoTruck projects.

Finally, I would like to thank my family, friends, and partner Emily for their endless support and encouragement throughout my masters program.

Dedication

To my family, friends, and partner Emily, thank you for your endless support and inspiration.

Table of Contents

Author's Declaration ii

Abstract iii

Acknowledgments iv

Dedication v

List of Figures ix

List of Tables xii

1 Introduction 1

1.1 Motivation 1

1.2 Objectives and Contributions 4

1.3 Outline 5

2 Literature Review and Background 7

2.1 Background 7

2.1.1 MPC 7

2.1.2 GPR 9

2.2 Literature Review 11

2.2.1 Trajectory Tracking Control . . . 11
2.2.2 Learning Based MPC . . . 14
2.2.3 Off-Road Vehicle Control . . . 16
2.3 Summary . . . 18

3 Physics-Driven Off-Road Controller Formulation and Results 19
3.1 Off-Road Vehicle Dynamics Prediction Model . . . 19
3.2 Controller State Space Formulation . . . 25
3.2.1 Linearization and Discretization . . . 25
3.2.2 Constraints . . . 30
3.3 Velocity Planner . . . 31
3.4 Simulation Setup and Controller Parameters . . . 34
3.5 CarSim Results and Discussion . . . 36
3.5.1 Constant Bank Angle Scenario . . . 37
3.5.2 Short-Track Off-Road Scenario - High μ . . . 40
3.5.3 Long-Track Off-Road Scenario - High μ . . . 44
3.5.4 Short-Track Off-Road Scenario - Low μ . . . 48
3.5.5 Long-Track Off-Road Scenario - Low μ . . . 50
3.5.6 Computational Performance . . . 52
3.6 Summary and Controller Shortcomings . . . 53

4 Learning-Based Off-Road Controller Formulation and Results 55
4.1 Learning Based MPC Framework . . . 55
4.2 Gaussian Process Regression Based Prediction . . . 57
4.3 GPR Integration and Tuning . . . 59
4.4 CarSim Results and Discussion . . . 60
4.4.1 Short and Long Track Off-Road Scenarios - High μ . . . 60
4.4.2 Short and Long Track Off-Road Scenarios - Low μ . . . 66
4.4.3 Short and Long Track Off-Road Scenarios - Payloads and Low μ . . . 71
4.5 Summary of Controller Tracking Performance . . . 83

5 Conclusions and Future Work **85**

5.1 Conclusions 85

5.2 Future Work 86

References **89**

APPENDICES **94**

A Controller Matlab/Simulink Code **95**

A.1 MATLAB Block Diagrams 95

A.2 MPC Controller Code Block 98

A.3 Adaptive Speed Planner Code Block 111

A.4 GPR Training Code 114

List of Figures

1.1 Software modules within an autonomous vehicle. This figure comes specifically from the WATonoBus project. [4] 2

1.2 WATonoTruck Off-Road Vehicle 4

2.1 MPC Working Principle [9] 8

2.2 1D Gaussian process regression example from Scikit learn. Results show that even with relatively few observations the true function can be estimated accurately.The prediction line represents the mean function of the Gaussian process. [16] 10

2.3 Learning Based MPC [31] 14

3.1 Dual Track Vehicle Model 20

3.2 Road Angles 21

3.3 Tire Slip Angle Dynamics 22

3.4 Path Tracking Terminology 24

3.5 Ackerman steering geometry around central point the vehicle is rotating around. 25

3.6 Simulation Topology 26

3.7 Speed compensation function for dynamic model stability. As the true speed of the vehicle approaches 4 m/s the speed compensation approaches 0. . . . 27

3.8 Adaptive Off-Road Speed Planner Flowchart 33

3.9 Simulation Scenario Landscapes 37

3.10 Road angle estimations for inclination and bank angles. Inclination angle is directly measured by CarSim whereas bank angle is estimated using equation 3.22 38
3.11 Trajectory Tracking Results - Constant Bank Angle with RA-MPC 39
3.12 Controller Outputs - Constant Bank Angle 40
3.13 Road angle estimations for inclination and bank angles. Bank angle is subject to noise and difficult to measure when tires are on different planes. . . 41
3.14 Trajectory Tracking Results - Short Off-road Track with RA-MPC - High μ 43
3.15 Controller Outputs - Short Off-Road Track 44
3.16 Road angle estimations for inclination and bank angles. Both angles are subject to intense noise due to terrain roughness. 45
3.17 Trajectory Tracking Results - Long Off-road Track with RA-MPC - High μ 47
3.18 Controller Outputs - Long Off-Road Track 48
3.19 Trajectory Tracking Results - Short Off-road Track with RA-MPC - Low μ 49
3.20 Controller Outputs - Short Off-Road Track - Low μ 50
3.21 Trajectory Tracking Results - Long Off-road Track with RA-MPC - Low μ 51
3.22 Controller Outputs - Long Off-Road Track - Low μ 52
3.23 Computational time required for RA-MPC controller 53
3.24 Each tire is on a plane that is oriented uniquely. Defining one plane for vehicle bank and inclination angles to be in reference to is difficult. 54

4.1 Learning Based MPC Topology 57
4.2 Trajectory Tracking Results - Short Off-road Track with GPR-MPC - High μ 62
4.3 Controller Outputs - Short Off-Road Track, GPR-MPC vs Baseline 63
4.4 Trajectory Tracking Results - Long Off-road Track with GPR-MPC - High μ 65
4.5 Controller Outputs - Long Off-Road Track, GPR-MPC vs Baseline 66
4.6 Trajectory Tracking Results - Short Off-road Track with GPR-MPC - Low μ 68
4.7 Controller Outputs - Short Off-Road Track, GPR-MPC vs Baseline - Low μ 69
4.8 Trajectory Tracking Results - Long Off-road Track with GPR-MPC - Low μ 70

4.9 Controller Outputs - Long Off-Road Track, GPR-MPC vs Baseline - Low μ 71
4.10 800 kg Payload Centered in Truck Bed 72
4.11 Trajectory Tracking Results - Short Off-road Track with GPR-MPC - 800 kg Centered Payload, Low μ 74
4.12 Controller Outputs - Short Off-Road Track, GPR-MPC - 800 kg Centered Payload, Low μ 75
4.13 Trajectory Tracking Results - Long Off-road Track with GPR-MPC - 800 kg Centered Payload, Low μ 77
4.14 Controller Outputs - Long Off-Road Track, GPR-MPC vs Baseline - 800 kg Centered Payload, Low μ 78
4.15 1000 kg Payload Uncentered in Truck Bed 79
4.16 Trajectory Tracking Results - Long Off-road Track with GPR-MPC - 1000 kg Uncentered Payload, Low μ 81
4.17 Controller Outputs - Long Off-Road Track, GPR-MPC vs Baseline - 1000 kg Uncentered Payload, Low μ 82

A.1 Simulink diagram of MPC controller, Same blocks used for GPR-MPC and RA-MPC 95
A.2 Simulink diagram of CarSim Plant module 96
A.3 Simulink diagram of Adaptive Off-Road Speed Planner 97
A.4 Simulink diagram of GPR Prediction Module 97

List of Tables

3.1 MPC Constraints 31

3.2 Controller Parameters 35

3.3 Vehicle Model Parameters 35

4.1 Comparison of RMS Errors for Heading, Lateral, and Velocity for Short Track Off-Road Scenario - High μ. 83

4.2 Comparison of RMS Errors for Heading, Lateral, and Velocity for Long Track Off-Road Scenario - High μ 83

4.3 Comparison of RMS Errors for Heading, Lateral, and Velocity for Short Track Off-Road Scenario - Low μ 83

4.4 Comparison of RMS Errors for Heading, Lateral, and Velocity for Long Track Off-Road Scenario - Low μ 84

4.5 Comparison of RMS Errors for Heading, Lateral, and Velocity for Short Track Off-Road Scenario - 800 kg Centered Payload and Low μ 84

4.6 Comparison of RMS Errors for Heading, Lateral, and Velocity for Long Track Off-Road Scenario - 800 kg Centered Payload and Low μ 84

4.7 Comparison of RMS Errors for Heading, Lateral, and Velocity for Long Track Off-Road Scenario - 1000 kg Uncentered Payload and Low μ, "DNC" = "Did not complete" 84

Chapter 1

Introduction

1.1 Motivation

Autonomous driving technology continues to advance and be at the forefront of automotive engineering. Studies project that by 2035 37% of vehicles will have advanced automated driving systems [1]. In many real-world driving scenarios the cause of crashes and accidents is human error. The overall adoption of Autonomous Vehicles (AVs) has the potential to remove human error from the loop. This in turn promises safer transportation for all. One study suggests that the adoption of AVs could could lead to accidents being reduced by 15% in Europe by 2030 [2]. Another common issue with traditional transportation is inefficiency. Human error leads to massive traffic congestion, and an overall loss of resources on infrastructure [3]. AVs have the potential to improve this and greatly benefit society.

Within an autonomous vehicle there are various modules that need to work together in unison to provide a safe, reliable, and comfortable driving experience. An overall block diagram of a typical autonomous software stack and its various modules can be seen in Figure 1.1. The first main module is the perception module. This module is the eyes of the vehicle and is responsible for perceiving the world around it. One of the main tasks of this module is detection. Objects such as pedestrians, road signs, and road markings need to be identified in a timely and accurate manner to properly drive the vehicle. Perception is also responsible for predicting the trajectory of dynamic obstacles. Similar to the way a human would predict what a pedestrian may do when approaching a crosswalk, perception needs to identify what the probable trajectory of an object will be. To perform this task, AVs utilize sensors such as cameras, Light Detection And Ranging sensors (LiDARs), and RAdio

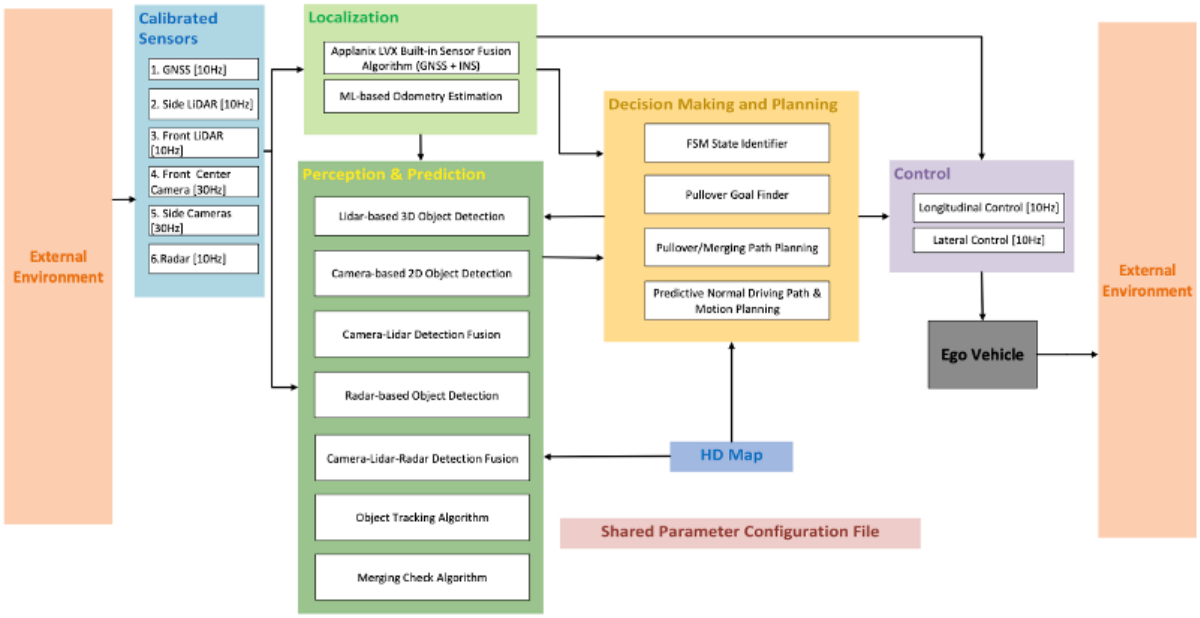


Figure 1.1: Software modules within an autonomous vehicle. This figure comes specifically from the WATonoBus project. [4]

Detection And Ranging sensors (RADARs) to identify object positions and movement. The next main module is the decision making and planning module. This software is responsible for processing the results from the perception module and making informed decisions about the movement of the vehicle. First and foremost, high level decisions such as stopping, or proceeding, need to be made based off of potential obstacles and road signs. From here, the planning portion of the module needs to provide a comfortable and achievable trajectory for the vehicle to follow. The final main module is control. This module needs to request the proper steering and throttle inputs to achieve the desired trajectory of the vehicle. It must achieve this trajectory while considering physical constraints of the vehicle from both an actuator and comfort perspective. Since the nature of the system is safety critical, the controller must be robust and efficient to ensure overall stability of the system. With model based control becoming the standard solution to this problem, the task of an efficient controller is complicated by the inclusion of highly non-linear dynamics in the vehicle model. The combination of the perception, decision making, and planning modules represent the highest computational requirements of the entire AV software stack. Controller modules often have limited resources for implementation and need to be operated in real-time to ensure safe operation of the vehicle.

AVs have great potential to solve problems outside of human transportation. AVs have the potential to improve efficiency in off-road commercial applications such mining, agriculture, manufacturing, and shipping. These applications all contain logistical problems involving the transportation of materials from one location to another. Usually, the transport of these materials would require a specially trained machine operator. By expanding AVs to this problem, the efficiency of commercial operations can be improved. The workplace environment can also be made safer by adopting these vehicles and help prevent heavy machinery injuries. A 2022 census of fatal occupational injuries by the U.S Department of Labour revealed that workers in transportation and material moving experienced the highest rate of fatalities amongst all occupational groups with 1620 deaths [5]. This industry has massive potential for automation that can reduce this high rate of injury.

While the proposition of AVs in off-road scenarios is promising from a cost reduction, safety, and efficiency standpoint, there are difficult environmental factors compared to an on-road scenario that pose as challenges. In an urban on-road scenario, the environment is very structured in nature; the surface is largely smooth and without large bank and inclination angles. Urban roads have been specifically designed to provide a comfortable and achievable trajectory to follow. In off-road applications the terrain becomes less uniform and may contain large bumps and obstacles. The terrain inherently has a lower coefficient of friction leading to traction issues. Surfaces such as thick mud or loose gravel lead to complex dynamical interactions with the tire that only experienced drivers may be able to handle. The combination of these environmental factors all contribute to the difficulty in tracking a desired path.

In recent years, Model Predictive Control (MPC) has been applied to the vehicle trajectory tracking problem on many accounts. With a trajectory-tracking MPC, a vehicle prediction model is used to predict the trajectory of the vehicle over a prediction horizon. Using this information, an optimization can be performed to determine the optimal vehicle control inputs that should be applied to reduce the cost of a predetermined function. Another advantage to MPC is the ability to apply input constraints to the optimization to ensure actuator limits of the vehicle are respected. State constraints can also be considered in the cost function to ensure ride comfort and overall vehicle stability. As previously discussed, to be effective, an off-road MPC formulation needs to include the effects of environmental factors including: non-linear tire dynamics from terrain interaction, large bank and inclination angles, and traction limits.

This work suits an ongoing project within the Mechatronic Vehicle Systems (MVS) Lab at the University of Waterloo referred to as the WATonoTruck (Figure 1.2). This project focuses on creating an autonomous modular vehicle that can be used for material transportation in commercial and off-road settings. The design of the vehicle features four



Figure 1.2: WATonoTruck Off-Road Vehicle

independent corner modules (CMs) that each provide steering, suspension, braking, independent drive system, and a microcontroller [6]. The WATonoTruck CMs all communicate with each other via CAN through a central computer. The vehicle is designed to be fault tolerant and work as a modular system. A previous students' work in the lab focused on creating a distributed and fault tolerant controller to effectively control the vehicles actuators [7]. Given the nature of the tasks the WATonoTruck will be required for, it is important that the vehicle is able to navigate off-road scenarios efficiently and accurately. A specialized MPC controller for off-road navigation should provide the system with the tools to handle difficult terrains.

1.2 Objectives and Contributions

The main objective of this work is to develop a trajectory tracking controller for an off-road autonomous vehicle. Particularly, this work provides insights into applications where a classical trajectory tracking formulation may not provide adequate tracking. As a first step, a first principles, physics oriented controller model that accounts for the added dynamics of banked, and sloped roads will be explored to evaluate its effectiveness. The trajectory tracking model includes the lateral and yaw dynamics of the vehicle for a smooth and accurate steering angle as well as longitudinal dynamics for velocity command tracking through direct motor torque control. The coupled controller formulation enables the use

of advanced vehicle control strategies like torque vectoring and smart torque allocation to improve off-road performance. The controller formulation will allow for the inclusion of actuator constraints to ensure safety and maneuverability. The model is designed to be practical and real-time feasible to ensure that compute resource requirements are kept within reason. This formulation shows that a low to medium fidelity model can be made suitable for difficult terrains.

A secondary objective of this work will be exploring the use of learning based modules to account for unmodelled vehicle dynamics. Since the measurement of road angles can be particularly difficult in uneven road conditions, the learning based approach focuses on augmenting the nominal vehicle model through a linear disturbance term. The approach ensures that no extra compute resources are required for solving for optimal control problem. Additionally, the approach enables the possibility of compensating for tire-terrain interaction. These dynamics are often left unmodelled but can produce dynamics that non-negligibly affect the vehicles motion. Through simulation analysis, the controller is validated against the physics oriented off-road vehicle controller. Scenarios with large hills, bank angles, varying terrain conditions, and vehicle payloads will be tested to show the effectiveness of learning based model augmentation.

The main contribution of this work is the adaptation of a coupled lateral and longitudinal MPC trajectory tracking controller for off-road vehicle use. The work explores the novel use of learning based methods to improve trajectory tracking ability in difficult scenarios. Furthermore, a practical, real-time feasible controller framework is presented that is expandable to cutting edge learning techniques. The findings of this work enables future development of real-time hybrid controllers without the need for expensive and complicated non-linear optimization solvers.

1.3 Outline

Chapter 2 dives into required background information for control and learning techniques utilized in this thesis. Topics in this section include a review of model predictive control basics, and a working introduction to Gaussian Process Regression (GPR). This section will highlight relevant literature on the topics of off-road vehicle control, MPC trajectory tracking control, and learning based control methods; Frameworks for the inclusion of learning modules in an MPC formulation will also be explored.

In Chapter 3 a physics oriented off-road MPC formulation is outlined. This begins with the derivation of vehicle dynamic equations that account for road bank and inclination

angles. The state space representation of this model is then presented along with the necessary linearization and dicretization of the model. This physics oriented controller is then evaluated using off-road simulation scenarios. The chapter highlights the difficulty in measuring road angles reliably through simulation results and analysis.

Chapter 4 will examine the use of learning based modules to compensate for the unmodelled vehicle dynamics instead of relying on complicated estimators. A framework for the inclusion of learning based offsets is presented in this chapter that maintains linearity and convexity in the optimal control problem. The use of GPR for unmodelled vehicle dynamic prediction will then be presented to show the potential effectiveness in off-road applications. Simulation results demonstrate the practicality and effectiveness of the overall hybrid MPC controller.

Chapter 5 concludes this thesis with a summary of its contents and an outline of future research potential.

Chapter 2

Literature Review and Background

2.1 Background

2.1.1 MPC

MPC is a control strategy that has gained significant attention in various fields due to its ability to handle multi-input, multi-output systems [8]. The essence of MPC lies in its ability to optimize the inputs and the predictions of its internal process model [8]. MPC operates in an iterative manner, using a model of the system to predict future behavior and solve an online optimization algorithm to find the optimal control action [9]. The process can be broken down into three basic requirements: a dynamic model for forecasting future states, a cost function that describes the optimal control problem, and feedback mechanism where MPC takes only the first action of the computed control input sequence. The recalculation of the optimal control action at each time step ensures the controller is robust to disturbances and uncertainty in the system model [9]. Figure 2.1 illustrates the working principle of MPC control.

MPC has been investigated for a significant number of potential applications to automotive systems in [10]. It's appealing for such applications due to its ability to handle things like actuator constraints, time delays, and performance optimization [10]. While initially MPC algorithms were too computationally expensive to run in an automotive setting, developments in the early 2000s have enabled the real-time implementation of the algorithm [10]. This opened the door for the use of MPC for vehicle dynamic control in the areas of traction, stability, suspension, and power train control [10]. As computing resources continue to become less expensive, many more applications are possible with

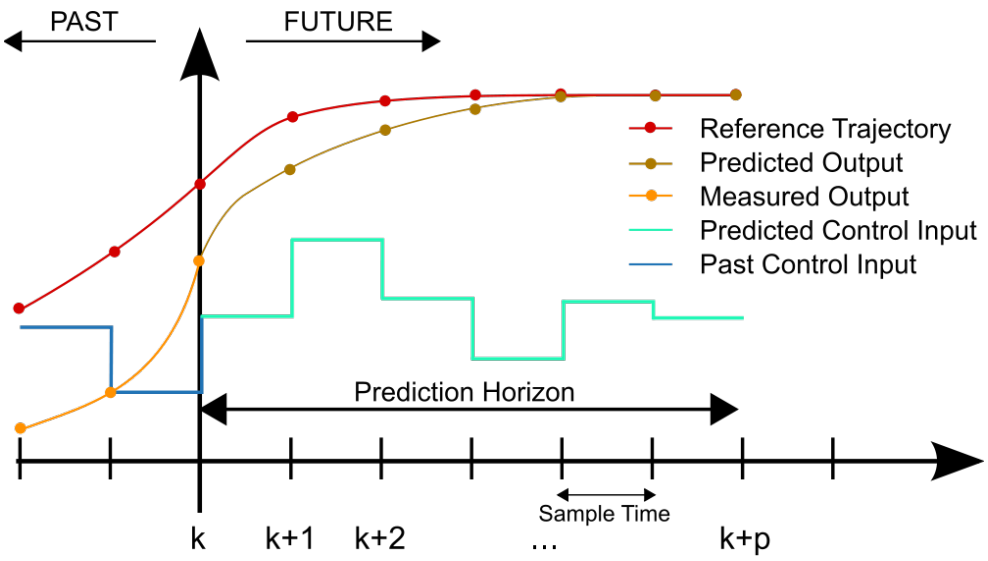


Figure 2.1: MPC Working Principle [9]

MPC; Recent applications include adaptive cruise control where MPC can be used to improve vehicle responsiveness while maintaining passenger comfort and safety, as well as autonomous driving tasks such as lane keeping and obstacle avoidance [10].

Solving the optimization problem in MPC requires the use of optimization solvers. There are several types of solvers available, each with its own computational requirements and characteristics. Quadratic Programming (QP) Solvers are used when the optimization problem in MPC is a quadratic programming problem. To be a quadratic problem the problem must have a quadratic cost function with linear constraints. This means that the process dynamics of the optimization also need to be linear. If the dynamics become non-linear, non-linear programming solvers must be used to solve the optimization. QP solvers are generally computationally cheaper than comparable NLP solvers, making them a good option for embedded hardware implementation. Recent research works focus on improving the speed, scalability, and parallelizability of QP solvers for MPC applications [11, 12, 13, 14].

Various methods can be used to solve QP problems including active-set, interior-point, and the alternating direction method of multipliers (ADMM) algorithms [8]. The active-set solver provides fast and robust performance for small-scale and medium-scale optimization problems [8]. The interior-point solver can provide superior performance for large-scale optimization problems [8]. The ADMM solver is generally computationally cheaper than comparable alternatives, and therefore a good option for embedded hardware implementa-

tion [8]. Matlab's quadprog() library supports the use of these QP solving algorithms as well as support for C/C++ code generation for experimental deployment of algorithms on test hardware.

2.1.2 GPR

GPR is a supervised learning method used to solve regression and probabilistic classification problems [15]. It is a non-parametric approach that makes predictions using prior knowledge while also incorporating an uncertainty measure into its prediction. The inherent uncertainty measure in the predictions make GPR an attractive option for automotive applications; The measure of uncertainty makes it possible to monitor the predictor to ensure safety and accuracy from a quantitative standpoint.

The first main piece of GPR is understanding Gaussian process models. A Gaussian process model defines a probability distribution of functions that fit a given set of data [15]. Since the model is a distribution of functions that fit the data, we can use the mean of the distribution as the most likely function, and the variance of the distribution to represent our confidence in the function [15]. As an example, a one dimensional GPR is shown in figure 2.2. The figure shows how the true function is regressed on using discrete measurements. Areas with observations close to each other have a high confidence whereas unexplored areas tend to have more uncertainty. This figure highlights the importance of various data points in the problems state space in order for GPR to make an high confidence prediction. Furthermore, in the multi-dimensional space, the importance for data grows as the state space becomes much larger.

If two inputs are similar in the feature space, they should produce the same output [15]. In order to ensure this is true, GPR needs a way to evaluate how similar two points in the multi-dimensional feature space are. The function that is used to do this is called the kernel function. If two points are similar in the kernel space, the function values at these points will also be of similar value [15]. A widely used kernel function in GPR is the Squared Exponential kernel. This is the kernel that is used in chapter 4 for GPR results. The kernel is defined as:

$$k(x, x') = \sigma^2 \exp\left(-\frac{\|x - x'\|^2}{2\ell^2}\right) \tag{2.1}$$

where: The parameters in the equation represent the following: σ^2 represents the variance parameter, ℓ represents the length scale parameter, and x and x' are the input

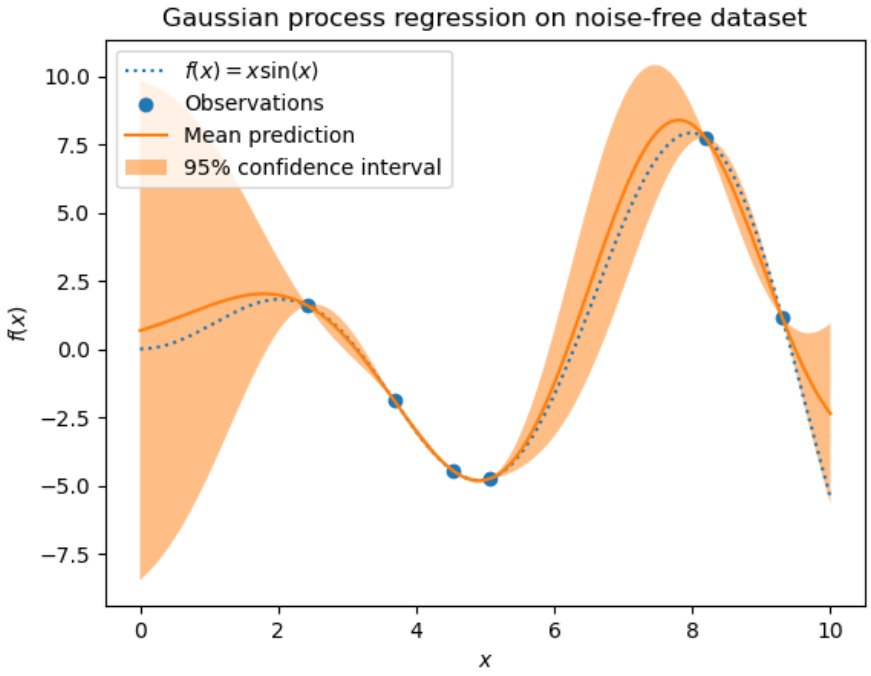


Figure 2.2: 1D Gaussian process regression example from Scikit learn. Results show that even with relatively few observations the true function can be estimated accurately.The prediction line represents the mean function of the Gaussian process. [16]

vectors. These are all hyper-parameters that need to be optimized. Matlab's fitgrp() looks to maximize the marginal log likelihood when optimizing the hyper-parameters of the kernel function [17].

The regression function modeled by a multivariate Gaussian can now be defined as:

$$P(f|X) = \mathcal{N}(f|\mu, K),$$

where $X = [x_1, \ldots, x_n]$ represents the measured data points, $f = [f(x_1), \ldots, f(x_n)]$ the desired function values, $\mu = [m(x_1), \ldots, m(x_n)]$ the mean function, and $K_{ij} = k(x_i, x_j)$ the kernel function. The Gaussian process model is a distribution over functions whose shapes are defined by the covariance/similarity function K [15]. Given a set of observed data points we can predict at new points X_* as $f(X_*)$.

The joint distribution of f and f^* can be written as:

$$\begin{pmatrix} f \\ f^* \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} m(X) \\ m(X^*) \end{pmatrix}, \begin{pmatrix} K & K^* \\ K_*^T & K^{**} \end{pmatrix} \right),$$

where $K = K(X, X)$, $K^* = K(X, X^*)$, and $K^{**} = K(X^*, X^*)$.

From the joint distribution, the conditional distribution of f^* is all that a predictor needs to generate. This can be obtained using the marginal and conditional distributions theorem [15]. The result is written as:

$$f^*|f, X, X^* \sim \mathcal{N}\left(K_*^T[K + \sigma_n^2 I]^{-1} f, K^{**} - K_*^T[K + \sigma_n^2 I]^{-1} K^*\right)$$

In many real systems the measured function values are noisy; This is included in the process model as follows:

$$y = f(x) + \epsilon$$

where $\epsilon \sim \mathcal{N}(0, \sigma_n^2)$ is Gaussian noise with zero mean and variance σ_n^2.

Now, for the given desired predictor point X^*, the prediction of the function y^* is given as:

$$y^*|y, X, X^* \sim \mathcal{N}\left(\text{mean}, \text{cov}\right)$$

where the mean function and the covariance function are expressed as:

$$\text{mean} = K_*^T[K + \sigma_n^2 I]^{-1} y$$

$$\text{cov} = K^{**} - K_*^T[K + \sigma_n^2 I]^{-1} K^*$$

2.2 Literature Review

2.2.1 Trajectory Tracking Control

A popular use of MPC in the application of autonomous vehicles has been for the longitudinal and lateral trajectory tracking of the vehicle. This task involves controlling the motors of the vehicle to track a desired speed request as well as controlling the steering of the vehicle to track the desired path. These control tasks are sometimes decoupled from one another into a speed controller and path tracking controller for simplicity of implementation. Furthermore, other control strategies such as PID and LQR are used in conjunction with an MPC controller for a hybrid approach. In [18] the authors explore the use of a

standard 2 DOF bicycle model for the path tracking of an on-road autonomous vehicle. The authors explore the stability properties of their formulation and conclude that asymptotic stability is guaranteed. In [19] Chen et al. explore a similar approach. The controller architecture presented uses a MPC controller for the lateral dynamics of the vehicle and uses a PID controller for speed request tracking. The MPC used in the paper features a linear prediction model to ensure that computational requirements are kept to a minimum. Results of this paper show that independent control of steering and drive torques works well for on-road vehicles. [20] furthers this idea of an MPC based lateral controller and PID longitudinal controller. This paper presents methods of improving this framework with constraints on the side slip angle in the vehicle prediction model. This approach compensates for the inaccuracies in tire models at large slip angles and overall improves the tracking accuracy of the vehicle. This work also highlights a method of improving the longitudinal MPC control by having a gain scheduled PID controller for various speeds of the vehicle. Furthermore, experimental results highlight the practicality of the decoupled framework. These works collectively give an idea into the large body of work into on-road control schemes featuring separate longitudinal and lateral controllers.

Yuan et al. present a method of coupling the dynamics of longitudinal and lateral dynamics by providing a dual layered approach in [21]. While the lateral and longitudinal dynamics are controlled by separate modules, the dynamics in the longitudinal and lateral controllers consider the dynamics of each other. Furthermore, this paper focuses on the practical application of the control technique with experimental results. The paper utilizes an Extended Kalman Filter to estimate the states of the vehicle required for MPC. Results show that the designed controller can effectively manage the lateral error and track the desired speed. This research is valuable for its practicality in real-vehicle implementation. Another approach at the coupling of the vehicle control tasks is to use a kinematic model of the vehicle. In [22] Pang et al. show that a kinematic derivation can be effective for managing the dynamics in a practical setting. This work shows that for relatively low speeds (6 m/s), the kinematic model is sufficient for on-road trajectory tracking. [23] compares the dynamic and kinematic prediction models for MPC control. This work uses experimental data to examine the prediction error of each model to motivate an MPC controller design. Results show that a kinematic model provides similar results to a dynamic model at low speeds and is less computationally expensive. The kinematic model also remains numerically stable in its discretization at low speed whereas the linear tire model in the dynamic model becomes singular. While a kinematic model would be ideal from a computational perspective, in the practice of off-road control the slip of tires could become non-negligible in the dynamics of the vehicle. This along with other non-linear tire effects make an off-road kinematic derivation less desirable even though the vehicle is

subject to lower speeds.

Something that is not widely covered in current research is the direct coupled control of both lateral and longitudinal dynamics by a single MPC controller. This method has the potential to provide an avenue for enhanced vehicle control. By coupling the two control variables an overall optimization of both control inputs can be done at once. The coupled strategy also allows for advanced vehicle control concepts in the vehicle prediction model including differential braking and torque vectoring. When it comes to off-road trajectory tracking, these advanced features are useful to improve the performance of the coupled controller. In general, a decoupled scheme would not be able to achieve the same level of performance due to ignoring the coupled nature of the dynamics being controlled. The coupled nature of the optimization also has the advantage of being more computationally feasible than two separate optimizations for longitudinal and lateral control. While the coupling of the dynamics provides opportunity for better performance, it also makes the overall MPC controller more difficult to tune. This is due to the multi-objective control task that needs to be represented as one single cost function when dealing with a centralized MPC scheme.

One of the key elements in a dynamic MPC model for vehicle control is the tire model. Many works have explored varying fidelity tire models for vehicle control. These tire models range from heavily non-linear physics based models to empirical data-driven based models. The purely non-linear physics based models are often too computationally expensive to feasibly run in real-time. For this reason, semi-empirical models such as the Magic Tire Model and the Brush tire model are popular in research works where a more accurate tire model is desired. A linear tire model has also been used heavily in research where slip angles of the vehicle can be kept to a minimum; This model is simple and computationally efficient. These tire models have all been designed for on-road use and may not be suitable for extreme off-road tire dynamics. An entire research field called Terramechanics focuses on off-road tire dynamics. This field studies the interaction of tires on loose terrain. Many non-linear effects are introduced in this interaction. For years, various off-road vehicle control research [24, 25, 26, 27] has been based off the popular Bekker Terramechanics model. This model serves as the physics based foundation for all off-road vehicle tire related research. It is important to remember that all tire models contain some level of error as it is extremely difficult to model the tire interaction with high fidelity in control oriented models. This thesis will aim to explore methods of compensating for these errors in Chapter 4.

2.2.2 Learning Based MPC

The inclusion of learning in an MPC formulation can happen in different ways. Figure 2.3 below shows the areas where learning can be included. The areas for learning include prediction model augmentation, prediction model replacement, state estimation, self tuning [28, 29], and an overall controller replacement [30]. The last method is particularly risky due to the possible inexplicable actions of trained networks. Especially relating to complex dynamical systems, a vast amount of data is required to make this approach feasible from a safety perspective.

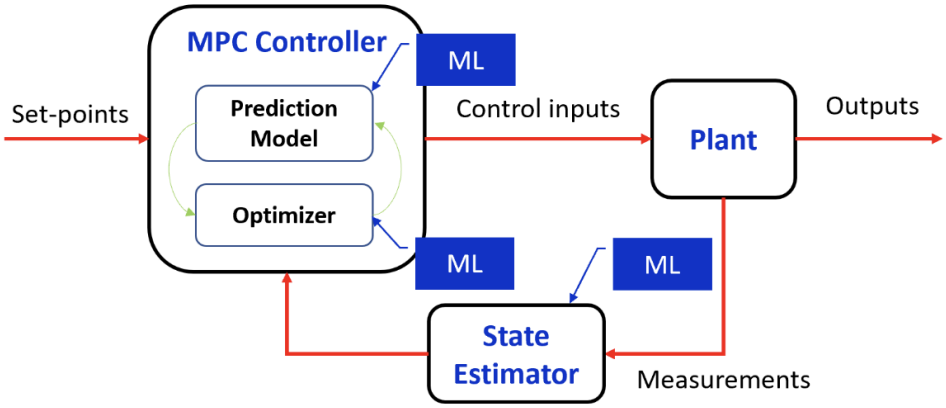


Figure 2.3: Learning Based MPC [31]

Research has shown there exists many areas within an MPC prediction model that are exposed to highly non-linear and difficult dynamics to model. These dynamics are difficult to incorporate in an efficient way when trying to achieve real-time operation. This problem highlights the need for other approaches outside of traditional physics modeling to augment the formulation with these dynamics.

Some researchers have explored an MPC formulation that entirely replaces a physics based prediction model with a learning based model [32, 33]. The approach shown in [32] uses a Recurrent Neural Network as the prediction model for a longitudinal and lateral trajectory tracking controller. The approach highlights the controller effectiveness in simulation by improving prediction accuracy and stability compared to other linear/non-linear formulations. In [33] a feed forward neural network is used as the prediction model of the vehicle. Two models are explored including an offline trained model and online trained model. The online trained model was shown to take time in order to achieve the same

performance as a classical MPC controller for trajectory tracking. The network needed enough data to match the performance of physics based methods. It was also mentioned that the offline trained model needed large amounts of data to be feasible for use. While this approach to the problem is promising when the dynamics of the system can be well represented with empirical data, the MPC formulation has become non-linear and requires an efficient solver to be real-time feasible. This is possible to achieve but the prediction model will also require expensive hardware such as GPUs to run in real-time. This increases the overall cost of the controller system. It is important to note that with pure neural network based prediction models in works such as [32] and [33] it is difficult to ensure safety. If the model does not generalize well and difficult scenarios are encountered the controller could become unstable. For this reason hybrid approaches are preferred.

In order to use a more hybrid approach researchers have looked at ways that learning can augment traditional physics based models to enhance the prediction quality. This ensures the controller still understands the physics involved. Furthermore, learning based augmentations can be monitored for accuracy and safety. In [31] Chao examines the use of GPR to augment the states in a linear MPC scheme. This work focuses on using GPR to learn the unmodelled dynamics from experience to control the vehicle better when that same scenario is encountered. This is done through a linear offset term on the states and acts like a modelled disturbance through the system. This technique is used in the application of vehicle stability to reduce yaw rates in aggressive steering manoeuvres. Experimental results show that the controller is effectively able to manage the stability of the vehicle and operate in real-time.

In a similar manner, some researchers have looked at using learning to only replace certain parts of the prediction model. In some works, learning based tire models are implemented where the network is entirely responsible for determining the forces of the tire. For example, in [34] Boada et al. examine the use of lazy learning to provide the longitudinal and lateral forces of a tire under combined slip conditions. Results show that the learning based model allows for better modeling of asymmetric tire behavior and a strong correlation with empirical data. Other approaches at using learning based methods for tire estimation have focused on parameter estimation within a widely used tire model. Wu et al. in [35] propose an online iterative learning technique to estimate the parameters of the brush tire model. Furthermore, adaptive techniques have been used for a similar purpose to estimate the parameters of the vehicle and tire model such as in [36] where the authors use a least squares online algorithm to estimate the cornering stiffness values of the vehicle model. This small adaptive extension showed improved results in path tracking capability as well as ensuring stability. Similarly, [37] Yang et al. use an amended square root cubature Kalman filter to adjust the cornering stiffness parameter to improve overall

vehicle path tracking.

Overall, a large body of work has focused on the inclusion of learning in vehicle control. Specifically in the application of trajectory tracking, various works examined in [38] have implemented some of these techniques mentioned with success. This survey looks at many papers that have included learning in a linear based MPC formulation. This approach is expandable to off-road scenarios in terms of real-time feasibility and overall practicality.

2.2.3 Off-Road Vehicle Control

When it comes to off-road driving, there are many factors that previous works in autonomous trajectory tracking haven't considered in their formulations. Firstly, it is common in off-road settings for there to be large inclination and bank angles in the path. These angles introduce dynamics due to gravity that are non negligible and will make it more difficult to track the path and desired speed of the vehicle. In [39], Xu et al. include the bank and inclination angles of the road into a MPC formulation for linear time varying lateral path tracking. Included in this study is an estimation of the normal force on each tire for a more accurate tire forces estimation within the model. The study uses simulation with CarSim to compare an MPC controller formulated without the road angles. The results show that the path tracking ability of the vehicle is more than double the ability of the conventional MPC. Simulations included rough terrain to highlight the robustness of the improved controller. Something this work doesn't explore is the coupled control of the vehicle with these angles considered. Furthermore, the estimation strategy of determining the road angles is omitted from the work. Overall, complex off-road scenarios need to be tested to see the potential of this controller formulation. This work highlights an area where this thesis aims to contribute insights by testing this formulation for aggressive off-road driving as well as building off this work to apply the road angles to the longitudinal control problem simultaneously. As will be explored in Chapter 3, the road angles in this formulation become difficult to estimate accurately in off-road settings.

As previously mentioned, a former student in the MVS lab researched the implementation of an agent based MPC (AMPC) framework for the off-road WATonoTruck vehicle. In [7], Ecclestone looks to implement a fault tolerant distributed control system for the WATonoTruck. This framework enables the vehicle to dynamically adapt to actuator faults as well as be configured in various actuator topologies. Results of the thesis show that the vehicle is able to effectively track a reference path under multiple controller topologies as well as in fault injection scenarios. This thesis also looks to build off this work to consider a centralized controller that considers off-road effects the controller will be subject to and

methods to compensate for this. This opens the door for future research into combining the fault tolerant nature of the AMPC controller with the off-road ability of the controller developed in this thesis.

Some works have examined the implementation of robust control methods for the lateral path tracking problem for off-road vehicles. In [40], Tan et al. develop a coupled off-road vehicle MPC controller for trajectory tracking. Using robust methods, the developed controller uses a layered approach to break up the problem into two main pieces. The robust MPC controller first aims to determine the optimal longitudinal, lateral, and rotational forces of the vehicle considering the angles of the terrain the vehicle may be on. An optimal control allocation algorithm then determines the steering angle and wheel forces that will collectively achieve the virtual input value. This control strategy shows strong results in simulation with overall lower tracking errors with respect to a linear MPC formulation. The computational requirements are also kept within reason compared to a linear MPC formulation but are still increased due to the added complexity of control allocation. This two layer approach introduces another optimization that needs to be performed at each control interval along with the optimization for the virtual inputs. Ideally, it would be nice to achieve similar results of this work without the need for multiple optimizations. This would improve the computational efficiency of the controller as well as its overall practicality. Similarly, Fnadi et al. in [41], use a dynamic MPC formulation considering bank and inclination angles on an off-road vehicle subject to slippery terrain. While the LQR controller compared to showed better tracking, the MPC is able to respect actuator constraints at each control interval. This study utilized a simple linear tire model and constrained the slip angle of the optimization to ensure accuracy of this model. Lucet also examines a similar scenario for off-road path tracking in [42]. In this work, the authors use a decoupled approach to control the lateral and longitudinal dynamics of the vehicle. While this work doesn't consider the road angles the vehicle is subject to, it does discuss the desire for a coupled longitudinal and lateral off-road controller. The paper discusses the difficulty in the multi-objective nature of a coupled controller while also considering the benefits of considering the coupled dynamics.

Another major challenge within off-road vehicle control is the added dynamics that loose soil and terrain introduce into the tire model of the vehicle. Vieira et al. in [43], look to build a robust lateral tracking framework that considers the added dynamics of the soil-tire interaction. The resulting controller showed strong results in both simulation and on a small prototype vehicle. This work acknowledges that further research on banked, sloped, and overall more complex off-road terrain is needed to fully validate the approach. Dallas has multiple works [25, 44], that derive a method of estimating the highly non-linear tire-soil interaction. Using neural networks, Dallas shows the ability to effectively

estimate an off-road tire model. This approach is then used in a combined planning and trajectory tracking MPC that is possible to implement in real-time. The inclusion of the neural network makes the overall MPC problem non-linear in nature and therefore requires extra compute resources to be efficient. Results show that between 260-440ms are needed for the NL-MPC controller.

2.3 Summary

The research outlined above highlights some key considerations required when developing an off-road model based controller:

- The angles that a vehicle may be subject to in off-road settings are non-negligible when developing a model based controller.
- Off-road tire models are highly non-linear and complex. The dynamics introduced by loose terrain are also non-negligible in terms of their effect on the vehicle dynamics.
- A computationally efficient implementation is desired that can also effectively manage the added dynamics of off-road scenarios.

With these considerations in mind, a clear gap exists in current research in terms of developing an efficient, single layer, coupled longitudinal and lateral off-road trajectory tracking controller. While many works have addressed various aspects of this problem individually, current research has not examined methods of efficiently compensating for off-road dynamics from both the tire-terrain and road dynamic perspective. Furthermore, an efficient implementation for speed and path tracking is lacking.

Chapter 3

Physics-Driven Off-Road Controller Formulation and Results

3.1 Off-Road Vehicle Dynamics Prediction Model

In order to develop an MPC controller, an accurate model of its dynamics is required to predict the receding horizon. It is important that the model used is both accurate and efficient. This represents a key trade off between model accuracy and real-time ability. In this section, a dual track dynamics model considering the angles of the road will be derived. The model consists of a point mass at the Center of Gravity (CoG) location with four tires generating forces that act upon this mass. The developed controller will control both the wheel torques and the front and rear steering angles of the vehicle; This means that both the lateral and longitudinal dynamics need to be derived. Using Newton's second law, the longitudinal, lateral, and yaw dynamics can be expressed as:

$$\sum F_x = m(\dot{u} - vr) \tag{3.1}$$

$$\sum F_y = m(\dot{v} + ur) \tag{3.2}$$

$$\sum M_z = I_z * \dot{r} \tag{3.3}$$

where m represents the vehicle's mass, u denotes the longitudinal velocity, v signifies the lateral velocity, r stands for the yaw rate, and I_z represents the vehicle's moment of

inertia about the vertical axis passing through its center of mass. Additionally, $\dot{u}$, $\dot{v}$, and $\dot{r}$ respectively denote the longitudinal, lateral, and yaw accelerations. The terms $\sum F_x$, $\sum F_y$, and $\sum M_z$ represent the sum of forces of the vehicle body in the longitudinal, lateral, and yaw directions, respectively.

In order to determine the sum of forces acting on the vehicle body the geometric properties of the model need to be explored. Figure 3.1 shows the layout of the proposed dual track vehicle model. This model has parameters for the wheel base W as well as the track width of the vehicle T. This model assumes that the track width is equal for both front and rear axles. The parameters l_f and l_r represent the distances from the front and rear axles respectively to the CoG location. δ_i represents the steering angle of each wheel.

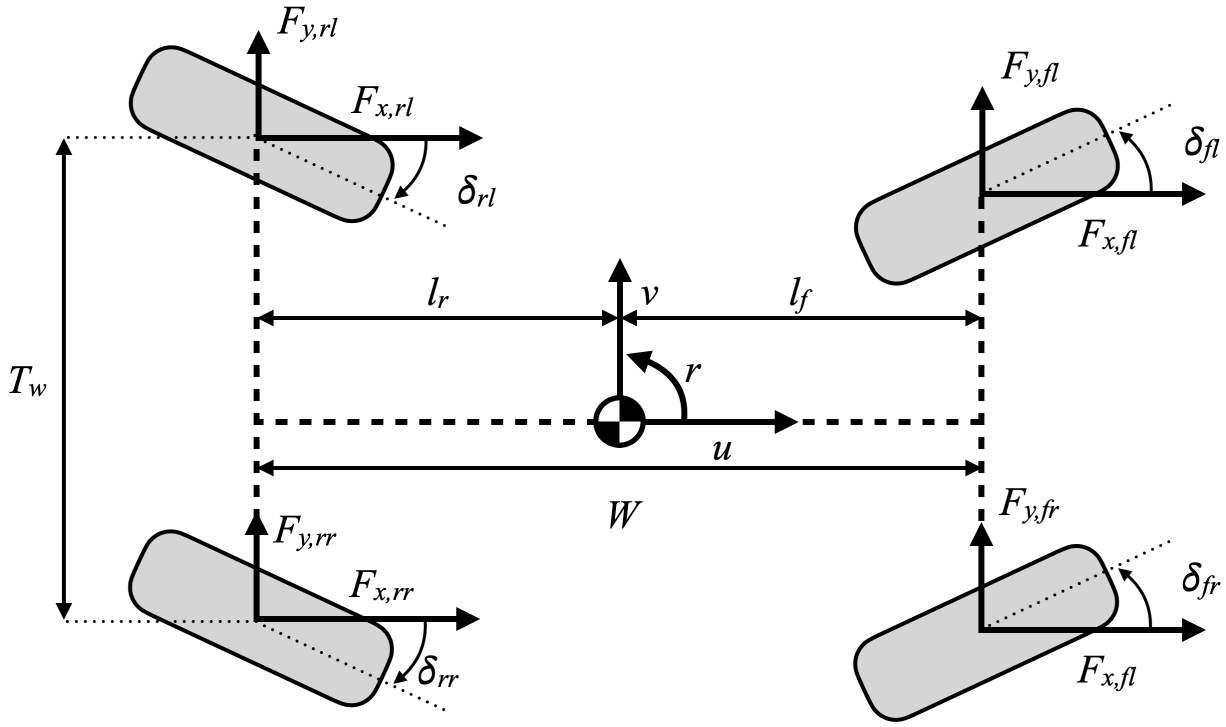


Figure 3.1: Dual Track Vehicle Model

From this figure we can expand the definitions of the vehicle frame forces and moments $\sum F_x$, $\sum F_y$, and $\sum M_z$ as:

$$\sum F_x = \sum f_{xi} cos(\delta_i) - f_{yi} sin(\delta_i) \tag{3.4}$$

$$\sum F_y = \sum f_{yi} cos(\delta_i) + f_{xi} sin(\delta_i) \tag{3.5}$$

$$\sum M_z = \frac{T_w}{2}(F_{xfr} + F_{xrr}) - \frac{T_w}{2}(F_{xfl} + F_{xrl}) + l_f(F_{yfr} + F_{yfl}) - l_r(F_{yrl} + F_{yrr}) \tag{3.6}$$

where $i \in \{fl, fr, rl, rr\}$ and f_i represents the forces and moments generated by each tire in the tire frame of reference within the dual track model.

In order to make this model better suited for off-road vehicles, the model needs to account for the angles of the road that the vehicle may be subject to. Figure 3.2 shows these angles, θ and ϕ, as the inclination and bank angles of the vehicle respectively. In practice these angles need to be estimated to make them available to the controller.

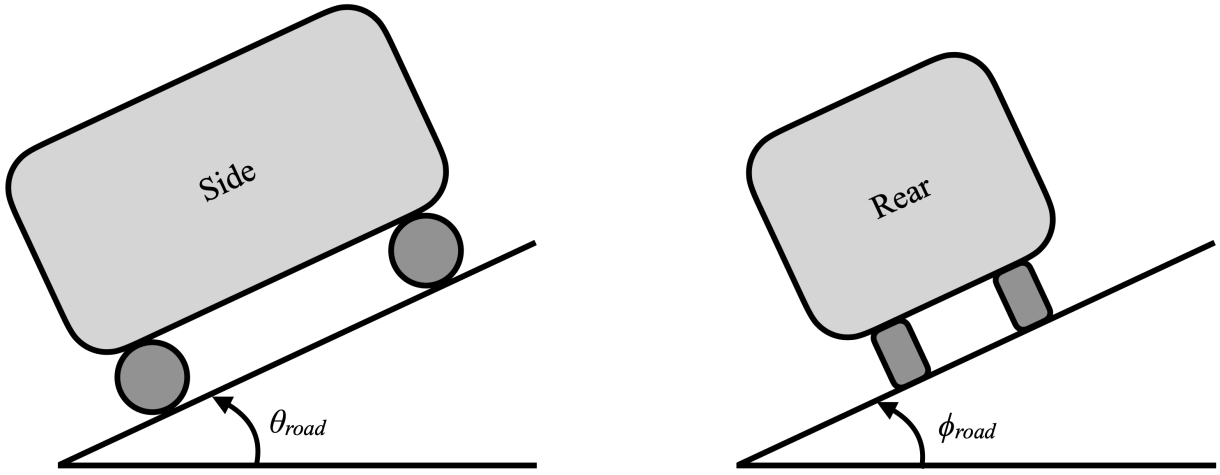


Figure 3.2: Road Angles

The effects of the combined inclination and bank angles can now be added to the longitudinal and lateral dynamics as:

$$\sum F_x = F_{xi} cos(\delta_i) - F_{yi} sin(\delta_i) - g sin(\theta) cos(\phi) \tag{3.7}$$

$$\sum F_y = F_{yi} cos(\delta_i) + F_{xi} sin(\delta_i) + g sin(\phi) cos(\theta) \tag{3.8}$$

Now that the forces of the vehicle have been defined, tire models are required to estimate the lateral and longitudinal forces on the tires generate. Beginning with the lateral tire model, a linear tire model is an accurate estimate of tire forces when side slip angles of the vehicle (α_i) are small. This model is much simpler than other semi-empirical models and will help reduce computational requirements of the controller. Figure 3.3 shows the dynamics of the slip angle of each tire.

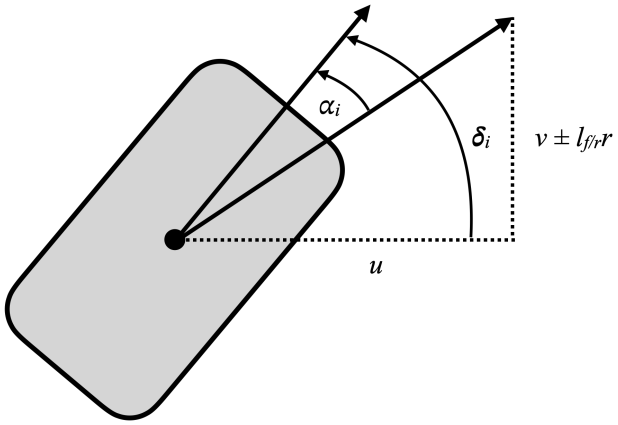


Figure 3.3: Tire Slip Angle Dynamics

Using a linear model, the tire force can be mathematically modelled as:

$$F_{yfi} = C_{\alpha_{fi}} \alpha_{fi} = C_{\alpha_{fi}} (\delta_{fi} - \frac{v + l_f r}{u}) \tag{3.9}$$

$$F_{yri} = C_{\alpha_{ri}} \alpha_{ri} = C_{\alpha_{ri}} (\delta_{ri} - \frac{v - l_r r}{u}) \tag{3.10}$$

where F_{yri} and F_{yfi} represent the lateral tire force at a single tire for the rear and front axles respectively. $C_{\alpha_{fi}}$ and $C_{\alpha_{ri}}$ represent the front and rear tire cornering stiffness' that are assumed to be in the linear unsaturated zone. It is important to note that the force calculated by the linear tire model is a function of the lateral velocity of the vehicle. This means that the bank angle will be implicitly accounted for in the calculation due to the bank of the road directly affecting this state. In [45] the authors use a combined model approach that assumes a linear tire model when the friction coefficient is high and switches to non-linear under low traction conditions. The model is then tested on scenarios with bank angles showcasing the validity of the linear tire model.

Next, a longitudinal tire model is needed to express the longitudinal forces each tire generates. In general, a longitudinal tire model functions similar to the lateral model. The force generated by the tire is a function of the slip ratio of the tire. In practice this estimation can be difficult and prone to noise especially in off-road settings. For this reason, the longitudinal tire model will neglect the wheel dynamics and assume that all torque is converted into longitudinal force. This yields the following tire model:

$$f_{xi} = \frac{Q_i}{R_{wi}} \tag{3.11}$$

where Q_i is the torque generated at each wheel and R_{wi} is the effective radius of the tire. While this introduces dynamics that are neglected, this tire model will ensure real-time performance of the controller. Chapter 4 will discuss methods of augmenting the vehicle model to account for the dynamics neglected by the derived prediction model.

To solve the path-tracking portion of the trajectory tracking problem, two other states, lateral (e_{lat}), and heading deviations (e_ψ), are required. Figure 3.4 shows the intuitive meaning of these states with regards to a desired path. In the context of the derived prediction model, a positive e_{lat} corresponds to being left of desired path and a positive e_ψ corresponds to the vehicle over rotating in the counterclockwise direction. In other words, the convention suggests that to drive a positive error to zero the vehicle must turn its tires to the right.

The equations can be derived as:

$$\dot{e_\psi} = r - r_{desired} = r - uk(s) \tag{3.12}$$

$$\dot{e_{lat}} = v + ue_\psi \tag{3.13}$$

Where $k(s)$ describes the given path's curvature.

To reduce the controller input dimension, an Ackermann steering geometry is enforced for each tire using the following relationships within the prediction model. This relationship can be derived using a common point the vehicle is rotating around as seen in Figure 3.5. The relationships below improves controller performance as well as reduces unnecessary slip at each tire.

$$\delta_{fl} = \arctan\left(\frac{\tan(\delta_f)}{1 - \frac{T}{2w}(\tan(\delta_f) - \tan(\delta_r))}\right) \approx \frac{\delta_f}{1 - \frac{T}{2w}(\delta_f - \delta_r)} \tag{3.14}$$

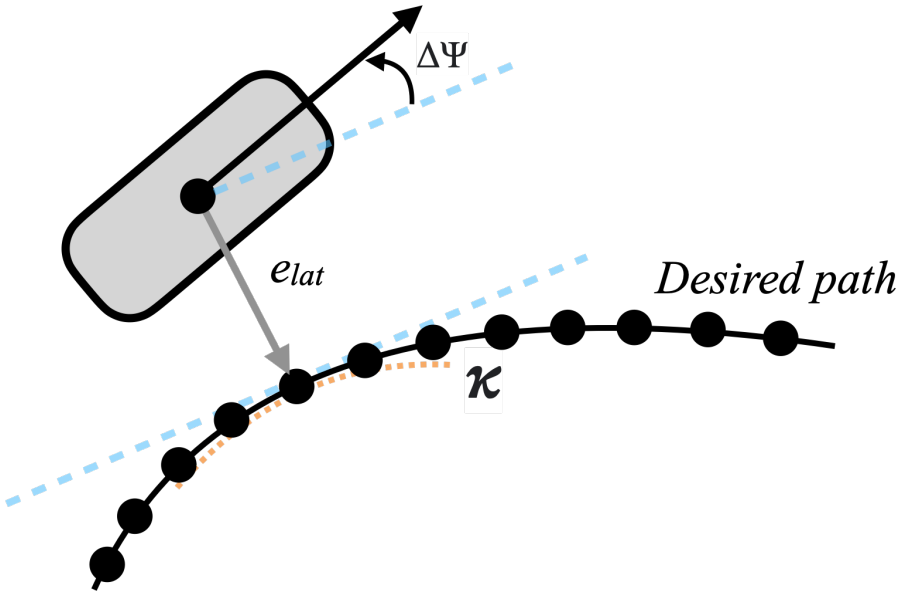


Figure 3.4: Path Tracking Terminology

$$\delta_{fr} = \arctan\left(\frac{\tan(\delta_f)}{1 + \frac{T}{2w}(\tan(\delta_f) - \tan(\delta_r))}\right) \approx \frac{\delta_f}{1 + \frac{T}{2w}(\delta_f - \delta_r)} \tag{3.15}$$

$$\delta_{rl} = \arctan\left(\frac{\tan(\delta_r)}{1 - \frac{T}{2w}(\tan(\delta_f) - \tan(\delta_r))}\right) \approx \frac{\delta_r}{1 - \frac{T}{2w}(\delta_f - \delta_r)} \tag{3.16}$$

$$\delta_{rr} = \arctan\left(\frac{\tan(\delta_r)}{1 + \frac{T}{2w}(\tan(\delta_f) - \tan(\delta_r))}\right) \approx \frac{\delta_r}{1 + \frac{T}{2w}(\delta_f - \delta_r)} \tag{3.17}$$

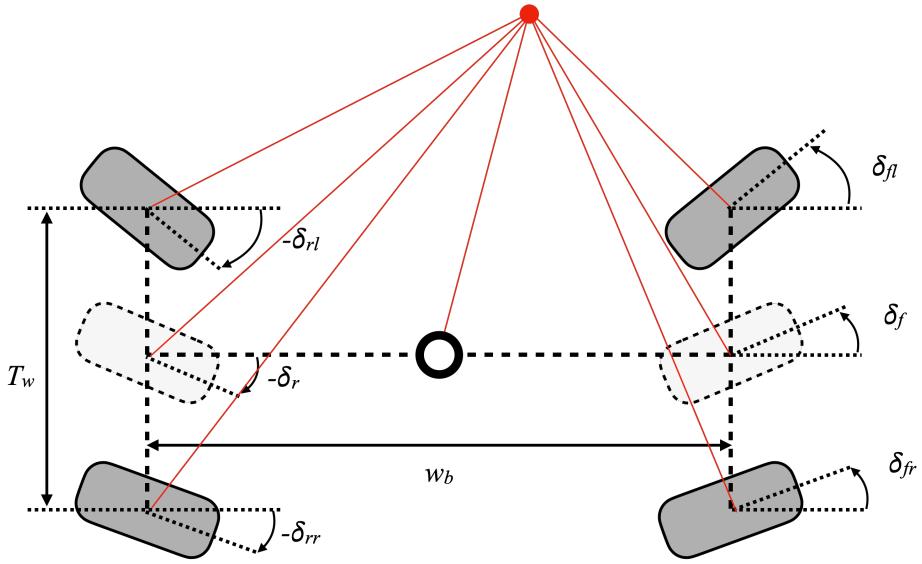


Figure 3.5: Ackerman steering geometry around central point the vehicle is rotating around.

3.2 Controller State Space Formulation

The controller topology shown in Figure 3.6 details the two key modules required for the proposed off-road vehicle controller; These modules include a state-space prediction model, and a vehicle speed planner.

3.2.1 Linearization and Discretization

To derive the prediction model, the dynamics derived in Section 3.1 need to be combined into a single Linear-Time-Varying (LTV) model. In this study, the state vector will be $\mathbf{X} = \begin{bmatrix} u & v & r & e_{lat} & e_\psi \end{bmatrix}$. The input vector will be $\mathbf{U} = \begin{bmatrix} \delta_f & \delta_r & Q_{fl} & Q_{fr} & Q_{rl} & Q_{rr} \end{bmatrix}$. Linearizing around the current operating point $\mathbf{X}_{op}$ and $\mathbf{U}_{op}$ yields the following matrices:

$$\mathbf{A}_{lin} = \begin{bmatrix} \frac{\partial \dot{u}}{\partial u} & \frac{\partial \dot{u}}{\partial v} & \frac{\partial \dot{u}}{\partial r} & \frac{\partial \dot{u}}{\partial e_{lat}} & \frac{\partial \dot{u}}{\partial e_\psi} \\ \frac{\partial \dot{v}}{\partial u} & \frac{\partial \dot{v}}{\partial v} & \frac{\partial \dot{v}}{\partial r} & \frac{\partial \dot{v}}{\partial e_{lat}} & \frac{\partial \dot{v}}{\partial e_\psi} \\ \frac{\partial \dot{r}}{\partial u} & \frac{\partial \dot{r}}{\partial v} & \frac{\partial \dot{r}}{\partial r} & \frac{\partial \dot{r}}{\partial e_{lat}} & \frac{\partial \dot{r}}{\partial e_\psi} \\ \frac{\partial \dot{e}_{lat}}{\partial u} & \frac{\partial \dot{e}_{lat}}{\partial v} & \frac{\partial \dot{e}_{lat}}{\partial r} & \frac{\partial \dot{e}_{lat}}{\partial e_{lat}} & \frac{\partial \dot{e}_{lat}}{\partial e_\psi} \\ \frac{\partial \dot{e}_\psi}{\partial u} & \frac{\partial \dot{e}_\psi}{\partial v} & \frac{\partial \dot{e}_\psi}{\partial r} & \frac{\partial \dot{e}_\psi}{\partial e_{lat}} & \frac{\partial \dot{e}_\psi}{\partial e_\psi} \end{bmatrix} \qquad \mathbf{B}_{lin} = \begin{bmatrix} \frac{\partial \dot{u}}{\partial \mathbf{U}} \\ \frac{\partial \dot{v}}{\partial \mathbf{U}} \\ \frac{\partial \dot{r}}{\partial \mathbf{U}} \\ \frac{\partial \dot{e}_{lat}}{\partial \mathbf{U}} \\ \frac{\partial \dot{e}_\psi}{\partial \mathbf{U}} \end{bmatrix}$$

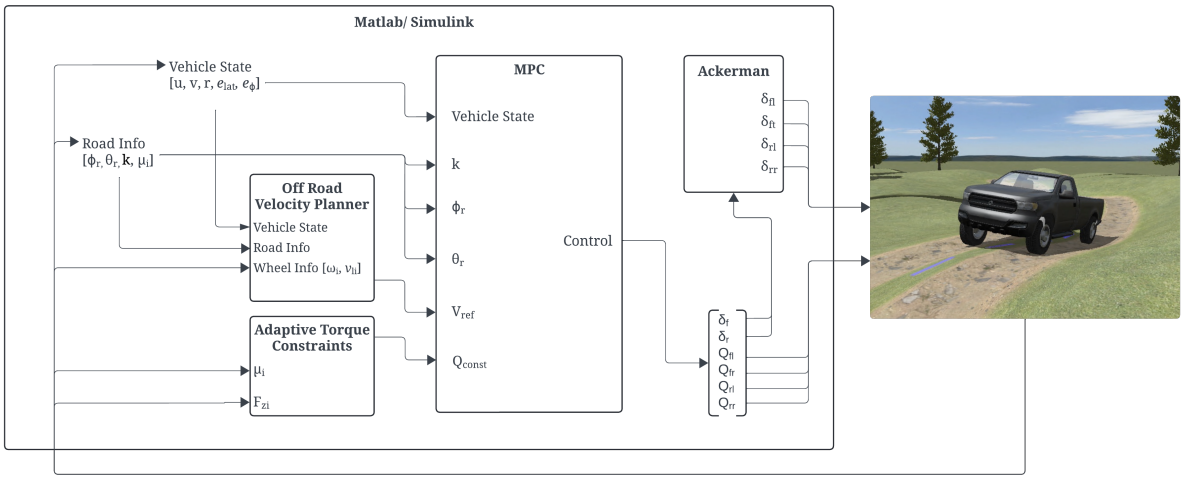


Figure 3.6: Simulation Topology

The LTV system can then be expressed as

$$\begin{cases} \dot{\mathbf{X}} = \mathbf{A}_{lin}(t)\mathbf{X} + \mathbf{B}_{lin}(t)\mathbf{U} + d \\ \mathbf{y} = \mathbf{C}\mathbf{X} \end{cases}$$

Where d is a constant term derived from the Taylor series approximation around the current operating point. This term includes the non-linear effects of the road as well as the curvature of the road as a simple linear offset term.

Since MPC is a discrete control method, the continuous LTV model needs to be discretized using the constant sample time T. In many cases, a dynamic model is used at moderate to high vehicle speeds. At these speeds, the forward Euler discretization works well and is not affected by any numerical instability. However, in this application, the dynamic model is being used for low-speed manoeuvres. This presents a challenge with forward Euler discretization because of inherent numerical instability at low speeds within the dynamic model. The numerical instability at low speeds causes oscillatory behaviour in the steering and torque outputs of the controller. An effective solution to this problem is to include a speed compensation term that artificially adds to the speed of the vehicle when it is below a fixed threshold.

$$u = u_{measured} + speedCompensation \tag{3.18}$$

This compensation term is gradually reduced to zero, the true speed of the vehicle, once numerical stability is maintained. The exact function used for speed compensation is arctan() based and can be see in figure 3.7

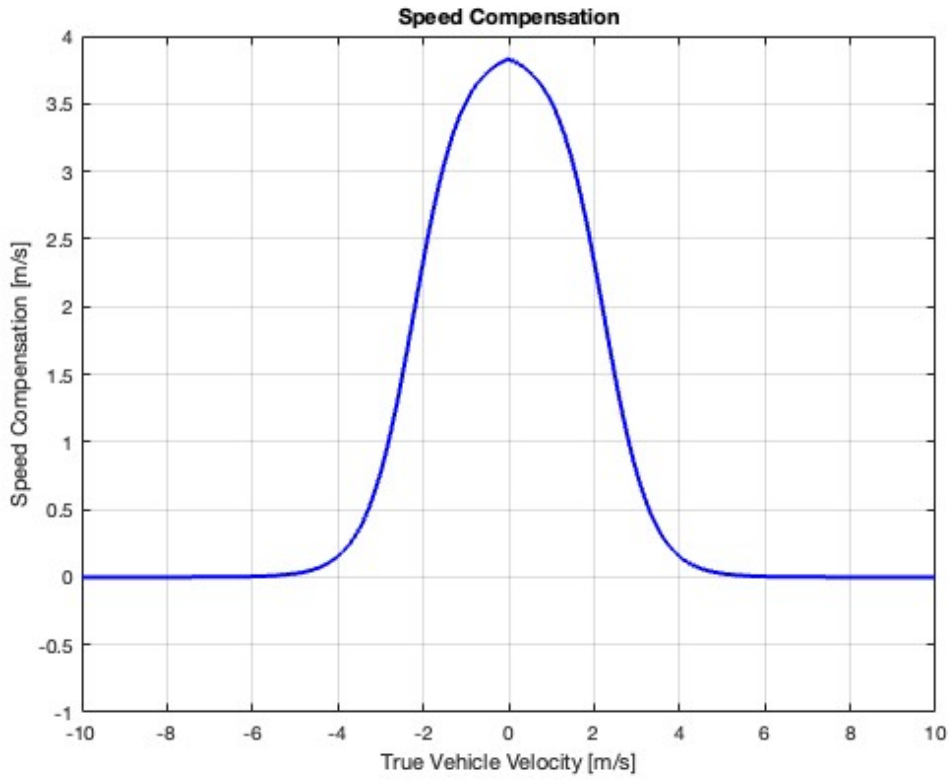


Figure 3.7: Speed compensation function for dynamic model stability. As the true speed of the vehicle approaches 4 m/s the speed compensation approaches 0.

Using a forward Euler discretization the system matrices can be represented as:

$$\begin{cases} \mathbf{X}(k+1) = \mathbf{A}_d(k)\mathbf{X}(k) + \mathbf{B}_d(k)\mathbf{U}(k) + \mathbf{d}_d(k) \\ \mathbf{y} = \mathbf{C}_d\mathbf{X} \end{cases}$$

Where:

$$\mathbf{A}_d = \mathbf{I} + \mathbf{A}_{lin}T, \quad \mathbf{B}_d = \mathbf{B}_{lin}T, \quad \mathbf{C}_d = \mathbf{C}_{lin}$$
$$\mathbf{d}_d = \mathbf{X}(k+1) - \mathbf{A}_d\mathbf{X}(k) - \mathbf{B}_d\mathbf{U}(k)$$

To easily constrain both the input rate and absolute input for the tracking problem, an augmented state vector is defined as $\tilde{\mathbf{X}} = \begin{bmatrix} \mathbf{X}(k) \\ \mathbf{U}(k-1) \end{bmatrix}$. This leads to a new model in terms of $\Delta\mathbf{U}$:

$$\begin{cases} \tilde{\mathbf{X}}(k+1) = \tilde{\mathbf{A}}_d\tilde{\mathbf{X}} + \tilde{\mathbf{B}}_d\Delta\mathbf{U} + \tilde{\mathbf{d}}_d(k) \\ \mathbf{y} = \tilde{\mathbf{C}}_d\tilde{\mathbf{X}} \end{cases} \tag{15}$$

Where:

$$\tilde{\mathbf{A}}_d = \begin{bmatrix} \mathbf{A}_d & \mathbf{B}_d \\ \mathbf{0} & \mathbf{I} \end{bmatrix}, \quad \tilde{\mathbf{B}}_d = \begin{bmatrix} \mathbf{B}_d \\ \mathbf{I} \end{bmatrix}, \quad \tilde{\mathbf{C}}_d = \begin{bmatrix} \mathbf{C}_d & \mathbf{0} \end{bmatrix}$$

$$\tilde{\mathbf{d}}_d = \begin{bmatrix} \mathbf{d}_d(k) \\ 0 \end{bmatrix}$$

Using the augmented model, the lifted system can be defined in terms of a prediction horizon N_p and a control horizon N_c. The predicted output of the system will be governed by:

$$\mathbf{Y}(k) = \mathbf{S}_x(k)\tilde{\mathbf{X}} + \mathbf{S}_u(k)\Delta\mathbf{U} + \mathbf{S}_d(k)\overline{\mathbf{D}} \tag{16}$$

Where:

$$\mathbf{S}_x(k) = \begin{bmatrix} \tilde{\mathbf{A}}_d \\ \tilde{\mathbf{A}}_d^2 \\ \tilde{\mathbf{A}}_d^3 \\ \vdots \\ \tilde{\mathbf{A}}_d^{N_p} \end{bmatrix}, \quad \mathbf{Y}(k) = \begin{bmatrix} \tilde{\mathbf{y}}(k+1|k) \\ \tilde{\mathbf{y}}(k+2|k) \\ \tilde{\mathbf{y}}(k+3|k) \\ \vdots \\ \tilde{\mathbf{y}}(k+N_p|k) \end{bmatrix}$$

$$\mathbf{S}_u(k) = \begin{bmatrix} \mathbf{C}\tilde{\mathbf{B}}_d & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{C}\tilde{\mathbf{A}}_d\mathbf{C}\tilde{\mathbf{B}}_d & \mathbf{C}\tilde{\mathbf{B}}_d & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{C}\tilde{\mathbf{A}}_d^2\mathbf{C}\tilde{\mathbf{B}}_d & \mathbf{C}\tilde{\mathbf{A}}_d\mathbf{C}\tilde{\mathbf{B}}_d & \mathbf{C}\tilde{\mathbf{B}}_d & \cdots & \mathbf{0} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{C}\tilde{\mathbf{A}}_d^{N_p-1}\mathbf{C}\tilde{\mathbf{B}}_d & \mathbf{C}\tilde{\mathbf{A}}_d^{N_p-2}\mathbf{C}\tilde{\mathbf{B}}_d & \cdots & \cdots & \mathbf{C}\tilde{\mathbf{A}}_d^{N_p-N_c}\mathbf{C}\tilde{\mathbf{B}}_d \end{bmatrix}$$

$$\mathbf{S}_d(k) = \begin{bmatrix} \mathbf{C}\mathbf{I} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{C}\tilde{\mathbf{A}}_d & \mathbf{C}\mathbf{I} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{C}\tilde{\mathbf{A}}_d^2 & \mathbf{C}\tilde{\mathbf{A}}_d & \mathbf{C}\mathbf{I} & \cdots & \mathbf{0} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{C}\tilde{\mathbf{A}}_d^{N_p-1} & \mathbf{C}\tilde{\mathbf{A}}_d^{N_p-2} & \cdots & \cdots & \mathbf{C}\tilde{\mathbf{A}}_d^{N_p-N_c} \end{bmatrix}$$

$$\overline{\mathbf{D}}(k) = \begin{bmatrix} \mathbf{d}_d(k) \\ \mathbf{d}_d(k+1) \\ \mathbf{d}_d(k+2) \\ \vdots \\ \mathbf{d}_d(k+N_p-1) \end{bmatrix}, \quad \Delta\mathbf{U}(k) = \begin{bmatrix} \mathbf{U}(k|k) \\ \mathbf{U}(k+1|k) \\ \mathbf{U}(k+2|k) \\ \vdots \\ \mathbf{U}(k+N_c-1|k) \end{bmatrix}$$

In the application of trajectory tracking, a cost function is designed to limit the error between the reference path and the path of the vehicle as well as the reference longitudinal speed and true speed of the vehicle. This cost function includes the steering angles and wheel torques required to achieve this trajectory to ensure that smooth steering and input torque commands are used. The cost function can be constructed as

$$J = [Y(k) - Y_{\text{ref}}(k)]^T Q[Y(k) - Y_{\text{ref}}(k)] + \Delta U(k)^T R \Delta U(k) \tag{3.19}$$

where R and Q are matrices for the weight coefficients for steering angle increments and tracking error, respectively. For the path tracking problem, the reference values are simply 0 within Y_{ref}. The longitudinal velocity reference requires more consideration and will be discussed in Section 3.3.

The cost function now needs to be formulated in a standard way to use open source QP solvers. To do this, the following equation is first defined:

$$E(k) = S_x(k)\tilde{X} + S_d(k)\bar{D} - Y_{\text{ref}}(k) \tag{3.20}$$

Using this, the standard quadratic programming form is defined as

$$J = \frac{1}{2}\Delta U(k)^T H \Delta U(k) + f^T \Delta U(k) \tag{3.21}$$

where:

$$H = 2(S_u(k)^T \bar{Q} S_u(k) + \bar{R}), \quad f = 2E(k)^T \bar{Q} S_u(k)$$

$$\bar{Q} = \begin{bmatrix} Q & 0 & 0 & \cdots & 0 \\ 0 & Q & 0 & \cdots & 0 \\ 0 & 0 & Q & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \cdots & S \end{bmatrix}_{N_p \times N_{\text{States}}}, \quad S = \text{Terminal Cost}$$

$$\bar{R} = \begin{bmatrix} R & 0 & 0 & \cdots & 0 \\ 0 & R & 0 & \cdots & 0 \\ 0 & 0 & R & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \cdots & R \end{bmatrix}_{N_c \times N_{\text{Inputs}}}$$

With a standard quadratic cost function being used, the problem is guaranteed to be a convex optimization problem. This means that the solution to the optimization problem will be guaranteed to be the global optimum. This is a key advantage of an LTV model and quadratic cost formulation versus a non-linear model. A non-linear model requires more computation to find the global optimal solution. Furthermore, it is difficult to guarantee that the solution to the non-linear optimization is the true global minimum of the cost function.

3.2.2 Constraints

With the trajectory tracking controller system, there are various actuator constraints that need to be respected. In terms of the lateral path tracking problem, this includes constraints on both absolute steering angles for the front and rear tires, as well as a constraint on maximum steering rates. To improve the optimization problem definition, a constraint for the front and rear steering angles being the inverse of each other is enforced to ensure solver convergence in a timely manner. In a similar fashion as the steering, constraints need to be included for the both the maximum torque available at each wheel and the maximum torque rate for each wheel. The maximum torque at each wheel is computed on each controller iteration as a function of the normal load at each tire F_{zi} and coefficient of friction at each tire μ_i. This makes the controller adaptive to dynamic torque constraints on split terrains. With the inclusion of a dynamic torque constraint for each wheel, the max torque rate constraint needs to be carefully implemented to ensure the optimization problem remains feasible. If the max torque suddenly goes to zero and was previously a large value, a hard constraint could be problematic. To solve this problem, a hard constraint on the torque rate is used for the entire horizon except for the first step. A rate limiting block outside of the MPC problem will then ensure that the max torque rate of the wheel is not violated for the first time step. By using a dynamic torque constraint, an individual estimation is done for each wheel's coefficient of friction and the centralized controller can better distribute the torque and ensure that tires do not become saturated.

All constraints are selected with the WATonoTruck actuators in mind and are summarized in Table 3.1.

Constraint	Value	Unit	Description
$\delta_{min} \leq \delta_{f/r} \leq \delta_{max}$	± 30	[deg]	Max/Min Steering Angle
$\dot{\delta}_{min} \leq \dot{\delta}_{f/r} \leq \dot{\delta}_{max}$	± 7	[deg/s]	Max/Min Steering Rate
$\delta_f + \delta_r = 0$	$-$	-	Front/Rear Steering Equality
$Q_{min} \leq Q_i \leq Q_{max}$	$\mu \frac{F_{zi}}{R_{wi}}$	[Nm]	Max/Min Wheel Torque
$\dot{Q}_{min} \leq \dot{Q}_i \leq \dot{Q}_{max}$	± 10000	[Nm/s]	Max/Min Wheel Torque Rate

Table 3.1: MPC Constraints

Additionally, constraints for the slip angle of the tire may be desired to limit the uncertainty of the tire model particularly in high slip conditions. Since this constraint is not possible to implement as a hard constraint, slack variables would need to be added to the optimization to enforce it as a soft constraint. This is undesirable due to the added complexity to the optimization. The overall effect is significantly poorer performance and drastically increased compute resources required.

3.3 Velocity Planner

In the MPC controller formulated, there exists a need for a suitable velocity reference for the controller to track. The design of this speed reference should include comfort parameters for the acceleration of the vehicle while also maintaining an optimal speed for path tracking performance. In off-road scenarios, this task can be difficult due to the potentially challenging tight turns and large road angles the vehicle may face. Furthermore, the terrain could make tracking these paths difficult if conditions are poor. For example tracking in loose mud is much more difficult than on solid compact dirt. It is also desired that the velocity planner is not overly complex and requiring sophisticated estimation techniques to generate a velocity profile. The planner should not require advanced compute resources to ensure the optimizer in the controller is the only time consuming task. To achieve this task, an adaptive leaky-bucket approach is used to plan the velocity of the vehicle. The planner will monitor the real-time tracking performance of the vehicle and adjust the velocity reference if the lateral tracking of the vehicle becomes severely impacted. The planner will temporarily decrease the velocity request to ensure the tracking can be

improved before allowing the vehicle to speed up again. This adaptive velocity adjustment can be done with readily available signals the controller already requires. Comfort and safety parameters, including the maximum speed and acceleration limits of the vehicle, can easily be applied to the planner. The overall concept of the planner is outlined in Figure 3.8. Performance parameters e_{lat}, e_{ψ}, $\dot{e_{lat}}$, and λ_i (slip ratio for each tire) are used to evaluate the effectiveness of the current velocity request. The performance parameters are monitored with moving averages to limit the sensitivity of the planner to instantaneous changes. The planner then utilizes arctan() functions to quantify the impact a performance parameter will have on the output velocity request. During the performance adjustment step, the planner keeps track of how long it has been since previous adjustments to ensure the vehicle has a chance to overcome any time delays in the system before further reducing speed. Included in this time keeping ability, if the vehicle has good performance for a configurable period of time, the speed request will reduce the impact of the performance parameters using a leaky-bucket approach. This refers to slowly reducing the performance parameter effect on the velocity request. Finally, after a performance adjusted velocity request is calculated, the profile is generated using speed limit and acceleration parameters.

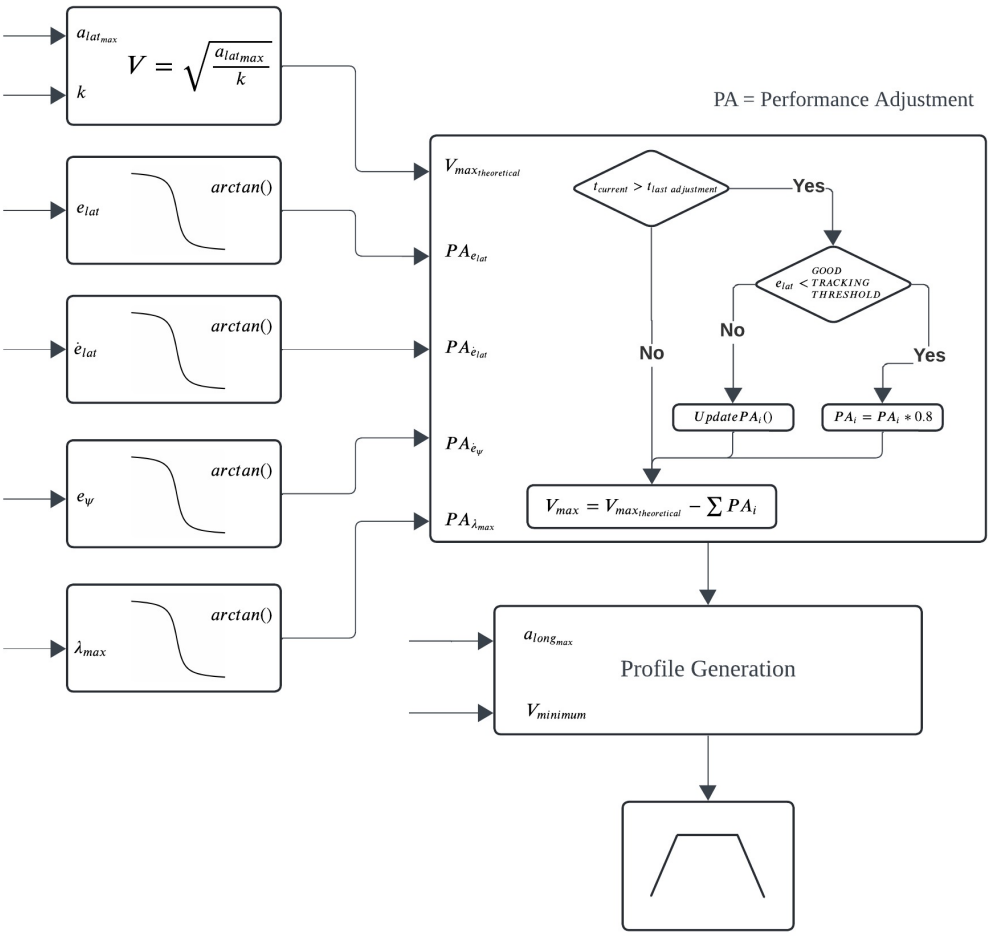


Figure 3.8: Adaptive Off-Road Speed Planner Flowchart

3.4 Simulation Setup and Controller Parameters

CarSim is used as a high-fidelity vehicle plant model to analyze the performance of the formulated controller. The simulations are performed using a co-simulation between CarSim and MATLAB/Simulink on an 11th Gen Intel i7 @ 2.50 GHz processor with 16 GB of Random-Access Memory (RAM). An EV pickup truck model with all wheel steering, independent wheel drives, and the physical parameters outlined in Table 3.3 is selected to best represent the WATonotruck hardware platform within the CarSim environment. As Figure 3.6. illustrates, the state of the vehicle is assumed to be measurable and therefore no state observers are required. The inclination angle, θ_{road}, and bank angle, ϕ_{road}, is also measurable directly from CarSim. The CarSim bank angle measurement is especially subject to noise in off-road scenarios from the roughness of the road and instantaneous roll motion of the vehicle body. This has a negative effect on controller performance and is therefore replaced with a simple estimation. The estimate of the road bank angle can be made using the z-coordinate of the wheel centers for both the front and rear axles of the vehicle. Using the wheel location measurements with the track width of the vehicle an estimate for the bank angle can be made for the front and rear axles using equation 3.22. The front and rear axle estimations can then be averaged to give a more robust estimate.

$$\phi_{road} = \arcsin\left(\frac{\Delta z_{f/r}}{T_w}\right) \tag{3.22}$$

The MPC controller parameters outlined in Table 3.2 are used in all simulations. The implemented controller features a fast sampling time of 0.050s and short horizon of 30 time steps to maintain computational efficiency while also achieving good tracking performance. This short horizon and fast sampling time is also important due to the LTV nature of the system. Since it is a non-linear system being successively linearized, the model is only accurate in the neighborhood of the operating point. It is important not to predict too far from this linearization point for best results. The cost function matrices are tuned to navigate the trajectory tracking problem of lateral path tracking and vehicle speed tracking. This dual objective controller needs to balance the control objectives to ensure a good response for both tasks. The Q and S matrices represent the costs for deviation from the speed, lateral error, heading error, and lateral error derivative references respectively in order by row. The R matrix is used to define the costs on the control inputs in the following order by row; $\delta_f, delta_r, Q_{fl}, Q_{fr}, Q_{rl}, Q_{rr}$.

$$S = Q = \begin{bmatrix} 1200 & 0 & 0 & 0 \\ 0 & 500 & 0 & 0 \\ 0 & 0 & 500 & 0 \\ 0 & 0 & 0 & 50 \end{bmatrix}$$

$$R = \begin{bmatrix} 1000000 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1000000 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2.5 \times 10^{-3} & 0 & 0 & 0 \\ 0 & 0 & 0 & 2.5 \times 10^{-3} & 0 & 0 \\ 0 & 0 & 0 & 0 & 2.5 \times 10^{-3} & 0 \\ 0 & 0 & 0 & 0 & 0 & 2.5 \times 10^{-3} \end{bmatrix}$$

Parameter	**Value**	**Description**
T	0.05 [s]	Controller Sample Time
N_p	30 [time steps]	Prediction Horizon Length
N_c	30 [time steps]	Control Horizon Length

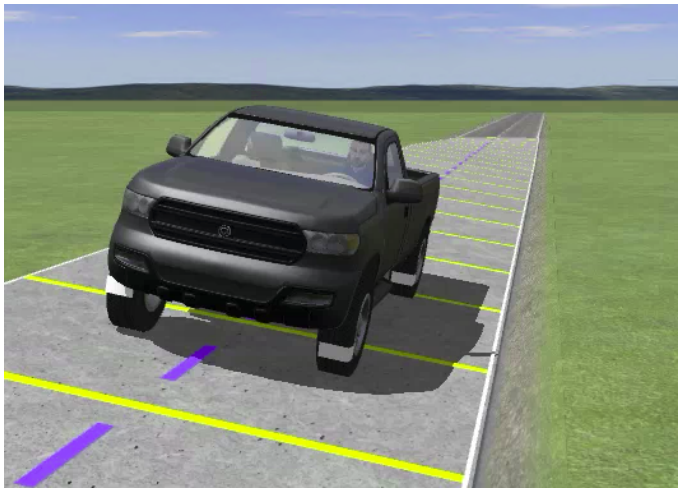
Table 3.2: Controller Parameters

Parameter	**Value**	**Description**
m	1985 [kg]	Total Mass
I_z	5000 [kg.m^2]	Yaw Moment of Inertia
$C_{\alpha f}$	35 000 [N/rad]	Front Cornering Stiffness
$C_{\alpha r}$	35 000 [N/rad]	Rear Cornering Stiffness
l_f	1.885 [m]	Distance from Front Axle to CoG
l_r	1.376 [m]	Distance from Rear Axle to CoG
R_w	0.379 [m]	Effective Radius of Tire
T_w	1.59 [m]	Track width
W_b	3.261 [m]	Wheelbase ($l_f + l_r$)

Table 3.3: Vehicle Model Parameters

3.5 CarSim Results and Discussion

Three CarSim scenarios are used to evaluate the effectiveness of the developed off-road MPC controller. The adaptive speed planner developed in Section 3.3 is used to determine the speed reference for all off-road scenarios tested. In the first scenario, a smooth constant bank angle road is used to evaluate the effectiveness of including the bank angle in the MPC formulation. From here, a short track off-road scenario is tested that features tight turns, relatively smooth bank and inclination angles, and a moderate-high coefficient of friction ($\mu = 0.75$)). The third simulation scenario features a combined on and off-road track that subjects the controller to rough uneven terrain. This terrain features a higher coefficient of friction ($\mu = 0.9$)) as well as aggressive bank and inclination changes. Images of the terrain and scenario can be see in Figure 3.9. Similarly, the two off-road tracks are then tested with lower coefficients of friction to make tracking the reference path more difficult.



(a) Smooth constant bank angle scenario on flat road



(b) Short track off-road scenario featuring tight turns and small hills



(c) Long track off-road scenario featuring rough terrain with large hills

Figure 3.9: Simulation Scenario Landscapes

3.5.1 Constant Bank Angle Scenario

The first scenario looks to gauge the effect of including a smooth bank angle in the MPC formulation. Figure 3.10 shows that the angles in this scenario are quite small and smooth. Figure 3.11 shows that the lateral tracking error is reduced with the Road Angle MPC (RA-MPC) controller when compared with a baseline that assumes a flat road. This highlights the potential for improved lateral performance.

Figure 3.12 shows the control output during this scenario. As expected the RA-MPC controller requests larger steering angles to compensate for the bank of the road.

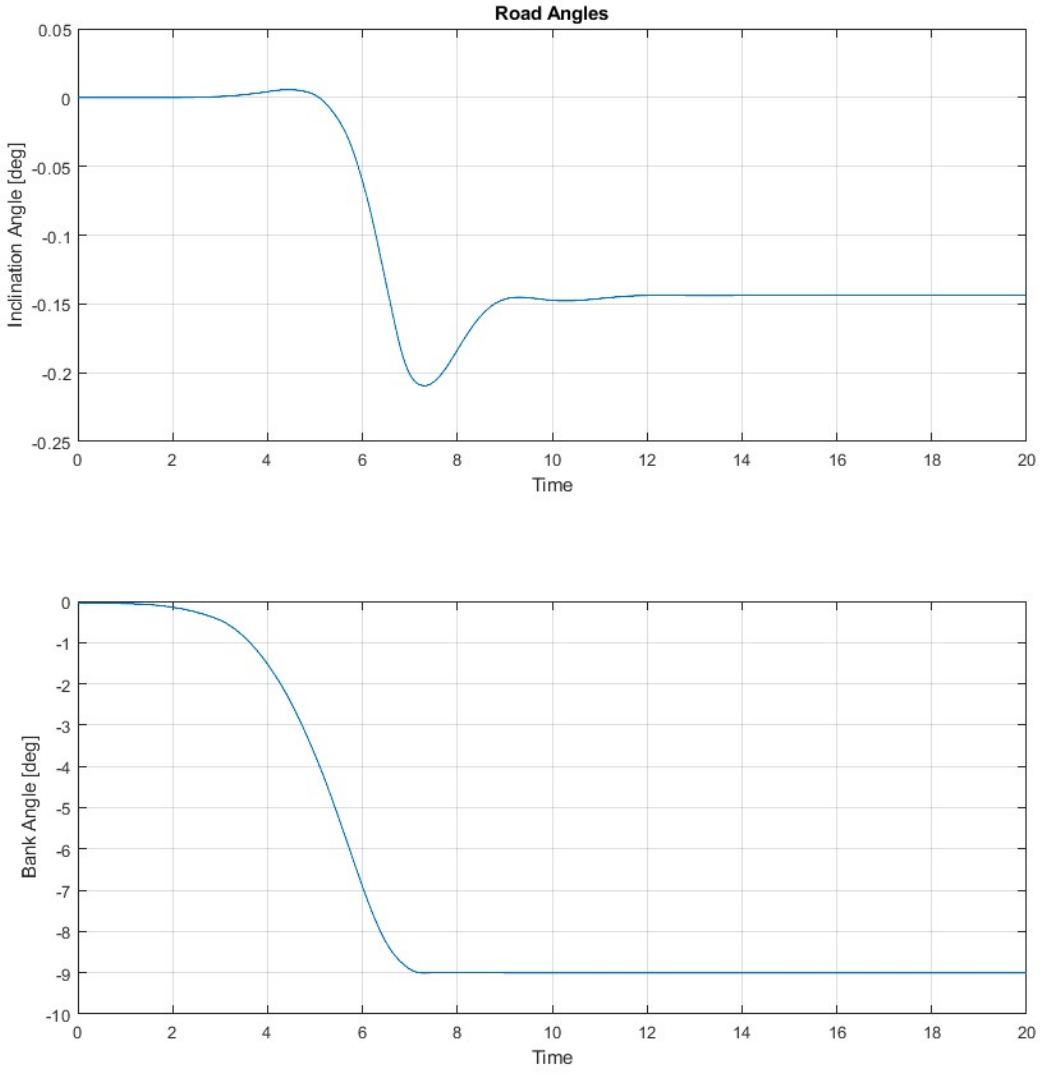


Figure 3.10: Road angle estimations for inclination and bank angles. Inclination angle is directly measured by CarSim whereas bank angle is estimated using equation 3.22

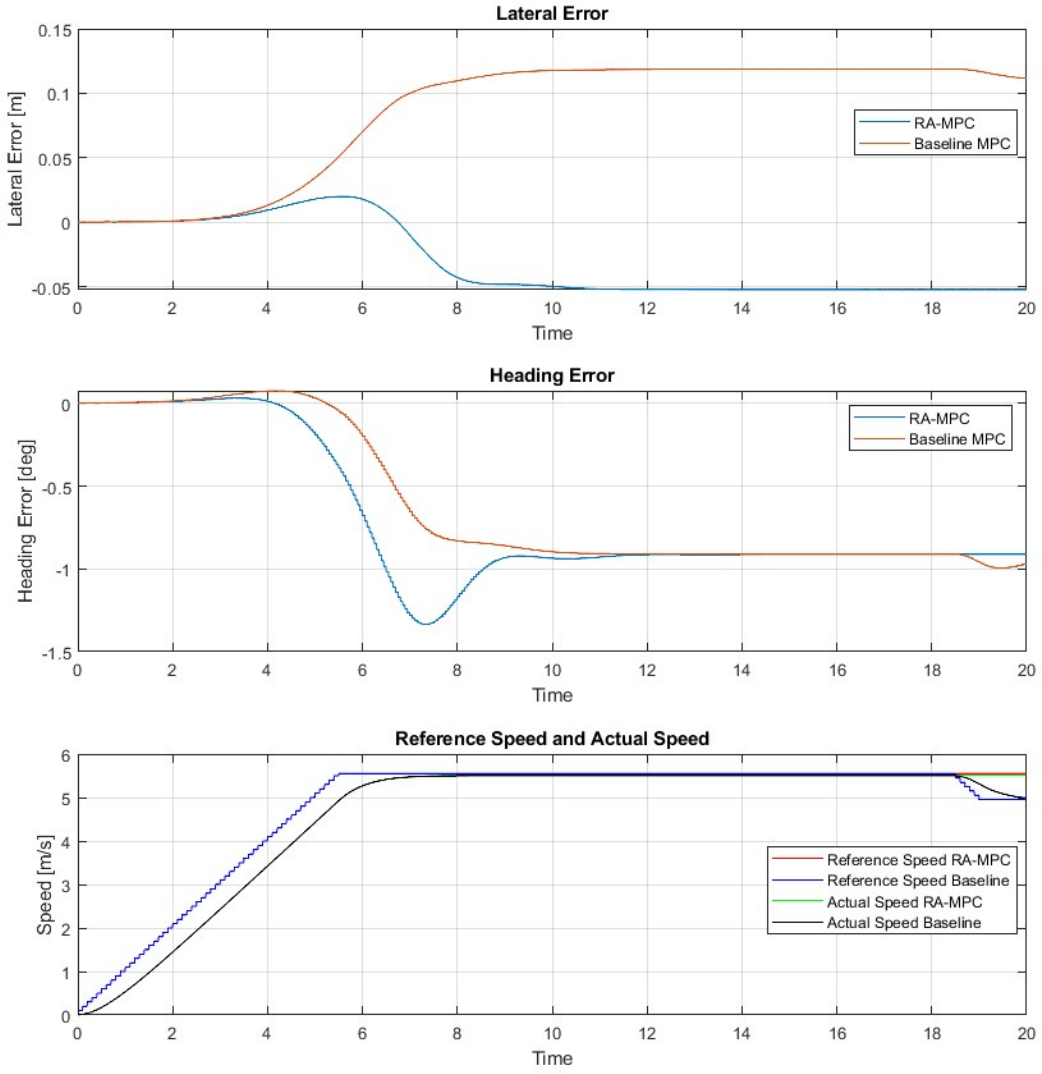
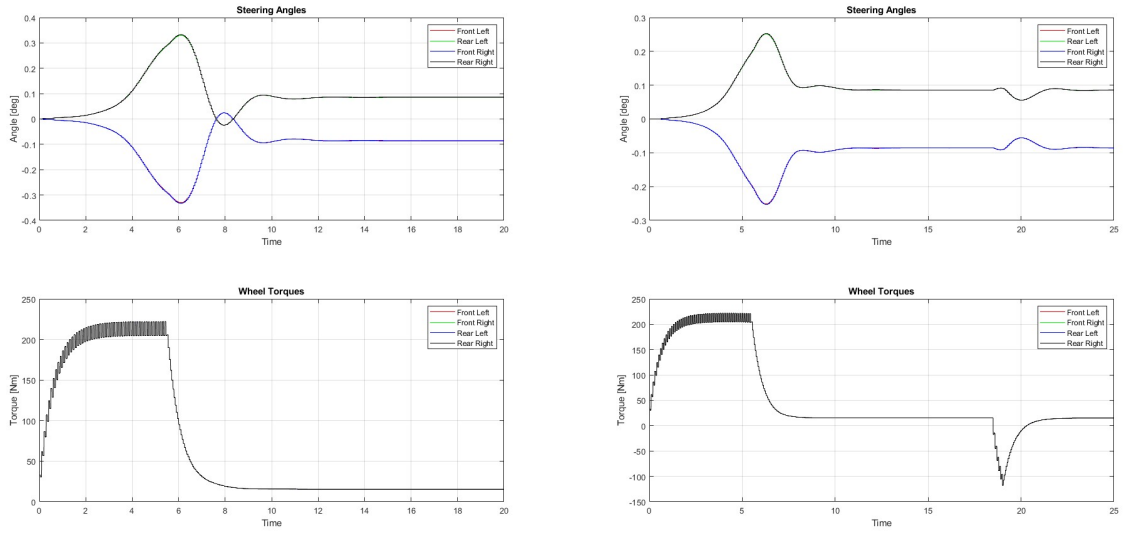


Figure 3.11: Trajectory Tracking Results - Constant Bank Angle with RA-MPC



(a) Torque and steering inputs using RA-MPC (b) Torque and steering inputs using base MPC

Figure 3.12: Controller Outputs - Constant Bank Angle

3.5.2 Short-Track Off-Road Scenario - High μ

The next scenario looks to test the limits of the bank angle estimation and see the effect of the road angles in a true off-road trial. This scenario features a rougher terrain with combined bank and inclination angles. The off-road scenario will also test the effectiveness of the off-road adaptive speed planner. If the controller is struggling to track laterally, the speed should be lowered to improve tracking.

Figure 3.13 shows the inclination and bank angle estimations for the scenario. While overall the angles are relatively smooth, there are rapid changes in the angles with respect to time. This oscillatory nature of the angle makes it difficult to accurately compensate for due to inherent time delays in the system and the rapidly changing dynamics of the vehicle body. Furthermore, the bank angle of the body is difficult to measure due to instantaneous roll motions of the vehicle body as well as the multiple planes that the vehicle tires may be on; This complexity makes a helpful estimation difficult to achieve.

Figure 3.14 makes it clear that the rapidly changing bank angle limits the benefit in modelling the angle in the prediction model. Overall, the peak lateral and heading errors seen with the RA-MPC are slightly larger in some cases compared to the baseline

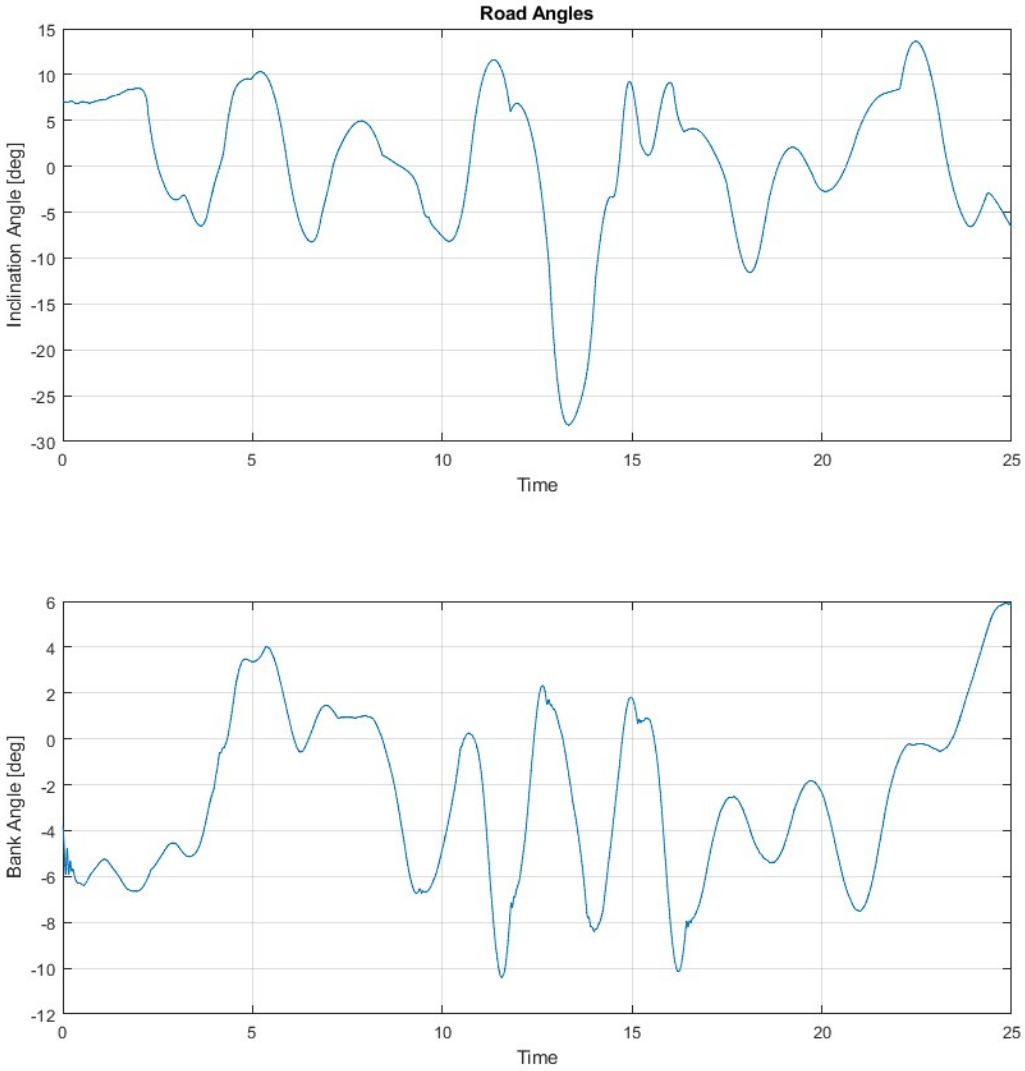


Figure 3.13: Road angle estimations for inclination and bank angles. Bank angle is subject to noise and difficult to measure when tires are on different planes.

formulation. It is worth noting that the Root-Mean-Square (RMS) of the lateral error is reduced from 0.2769 m to 0.2267 m when compared to the baseline controller. This suggests that the RA-MPC may be reaching its limit of improving lateral tracking. If the noise and overall frequency of the bank angle estimation is increased further it may have a negative effect on overall performance. This figure also shows the large effect the inclination angle has on the controllers ability to track the desired speed reference. At approximately 14 s in the scenario the vehicle experiences a large inclination angle of 30°. This corresponds to a hill that the vehicle needs to climb. In the context of a slowly moving off-road vehicle, the disturbance introduced by the hill is the largest unaccounted for factor in the baseline MPC. The velocity tracking results show that the baseline controller experiences a large deviation from the desired speed because of this hill; The RA-MPC is able to compensate for this hill in the prediction model and experience less deviation from the reference point. Overall, the RA-MPC reduces the velocity RMS error from 0.7348 m/s to 0.4140 m/s. This trial confirms that the inclination angle is an important factor to model in off-road scenarios.

Figure 3.15 shows that as expected, the steering angles are larger with the RA-MPC to attempt to reduce the lateral error of the vehicle along the reference path. The torque plot also shows a larger peak torque around the 14 s mark compared to the baseline to compensate for the hill the vehicle is climbing. This plot also highlights the dynamic torque constraints on the vehicle. During the scenario there are multiple points where certain tires lose traction and cannot provide meaningful force to the vehicle body. For example, around the 14 s mark the torque in the front left tire goes to 0 in the Figure. The formulated controller is able to handle this scenario and distribute the required torque to other tires.

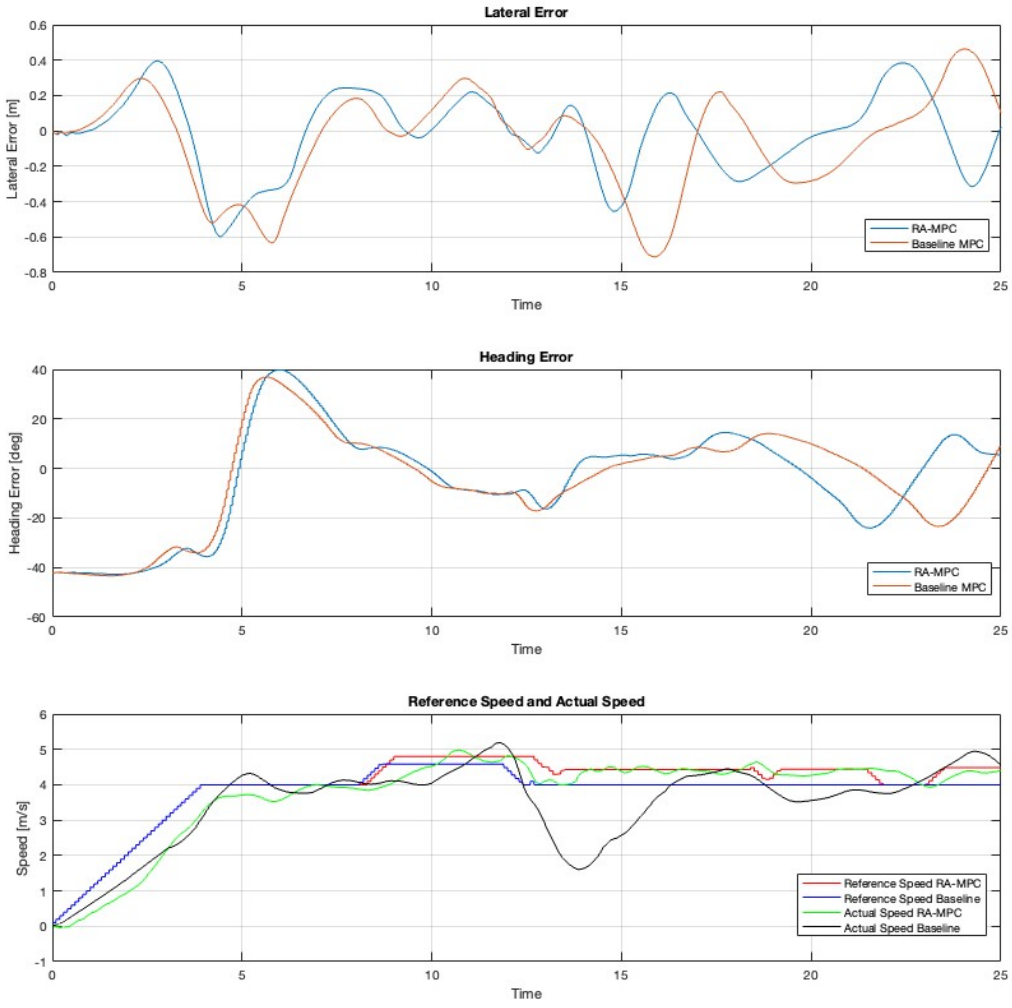
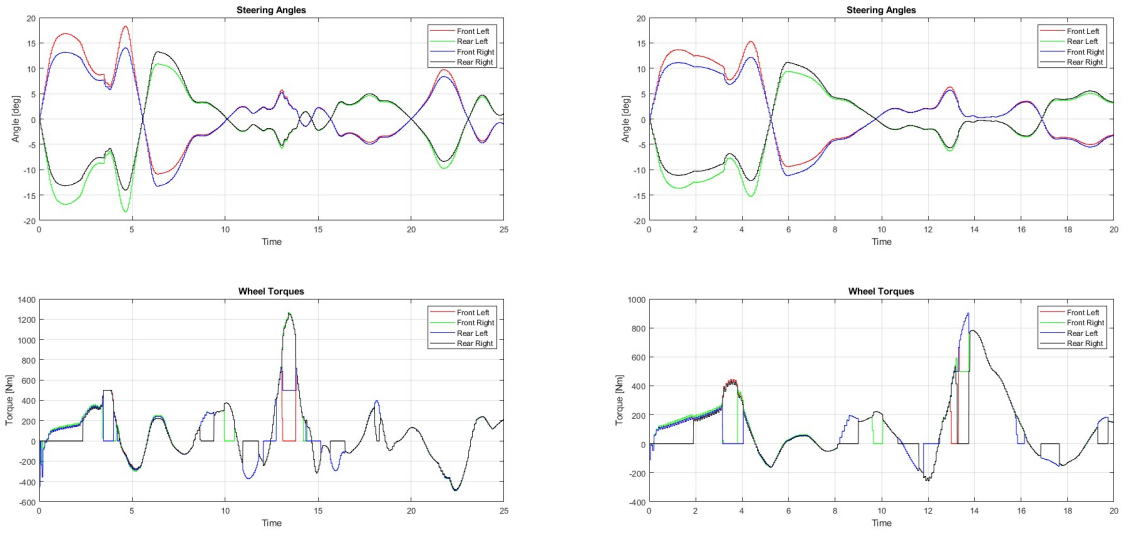


Figure 3.14: Trajectory Tracking Results - Short Off-road Track with RA-MPC - High μ



(a) Torque and steering inputs using RA-MPC (b) Torque and steering inputs using base MPC

Figure 3.15: Controller Outputs - Short Off-Road Track

3.5.3 Long-Track Off-Road Scenario - High μ

In this subsection, a longer off-road track is tested. This scenario has a higher coefficient of friction compared to the previous short track scenario. The scenario also features a much rougher road profile to test the limits of the RA-MPC controller. The hills and bank angles in this scenario are more frequent and rapidly changing.

Figure 3.16 shows that the bank and inclination angles are much more aggressive and subject to noise due to the terrain roughness profile. Both angles have high frequency oscillations that reduce the estimation quality.

Figure 3.17 shows that the RA-MPC has worse lateral tracking results compared to the baseline MPC. At multiple times in the simulation the RA-MPC over-steers and increases the lateral error. This is due to the rapidly changing bank angle that the vehicle prediction model attempts to account for. Since the angle is not changing in a smooth manner the prediction model estimate of the disturbance provides an adverse effect to the controller output. In this scenario, it is better to let the feedback nature of the controller compensate for the disturbance instead of direct modelling. In terms of RMS of the lateral error, the RA-MPC has an error of 0.1270 m whereas the baseline MPC has a error of just 0.1011 m.

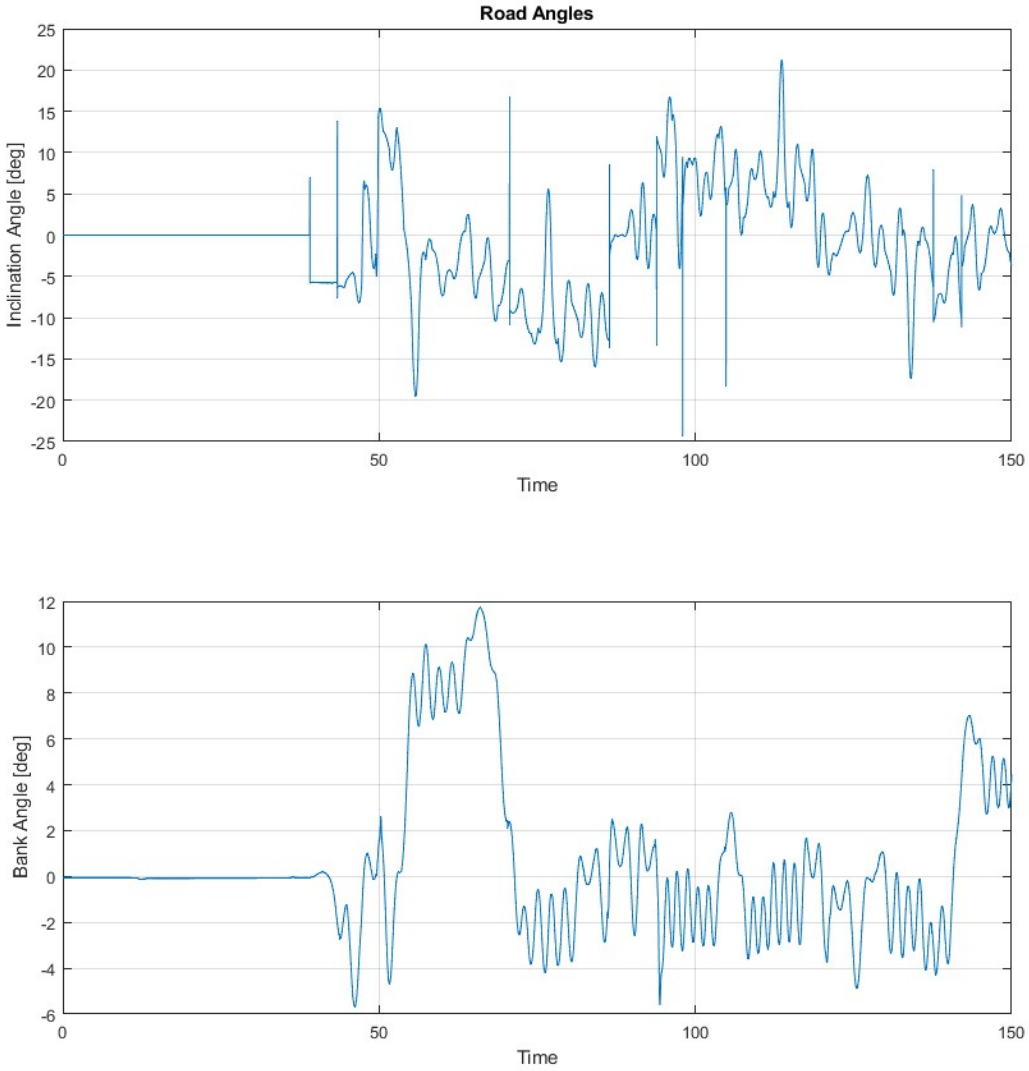


Figure 3.16: Road angle estimations for inclination and bank angles. Both angles are subject to intense noise due to terrain roughness.

The lateral tracking results suggest that the sensitivity of the system to bank angle is not high enough to warrant direct compensation if the angle cannot be estimated accurately. The velocity tracking results, however, show that the velocity tracking of the vehicle is still drastically improved even with the noisy inclination angle estimation. The RMS velocity error is reduced from 0.6321 m/s to 0.2204 m/s with the RA-MPC controller. It should be noted that the improved speed tracking allows for the vehicle to travel at a faster speed due to the adaptive speed planner seeing less slip in the vehicle tire response.

Figure 3.18 shows that the controller outputs have much more noise than previous simulations. This is due to the difficult uneven an rough terrain the vehicle faces in this scenario compared to smoother terrains in other scenarios. Steering angles in particular show oscillation from the rough terrain. The magnitude of these oscillations is quite small and likely not noticeable apart from the road roughness itself. The wheel torques in this scenario show that the vehicle tires all maintain traction throughout the scenario compared to the short track trial. This leads to a relatively even distribution of the torques.

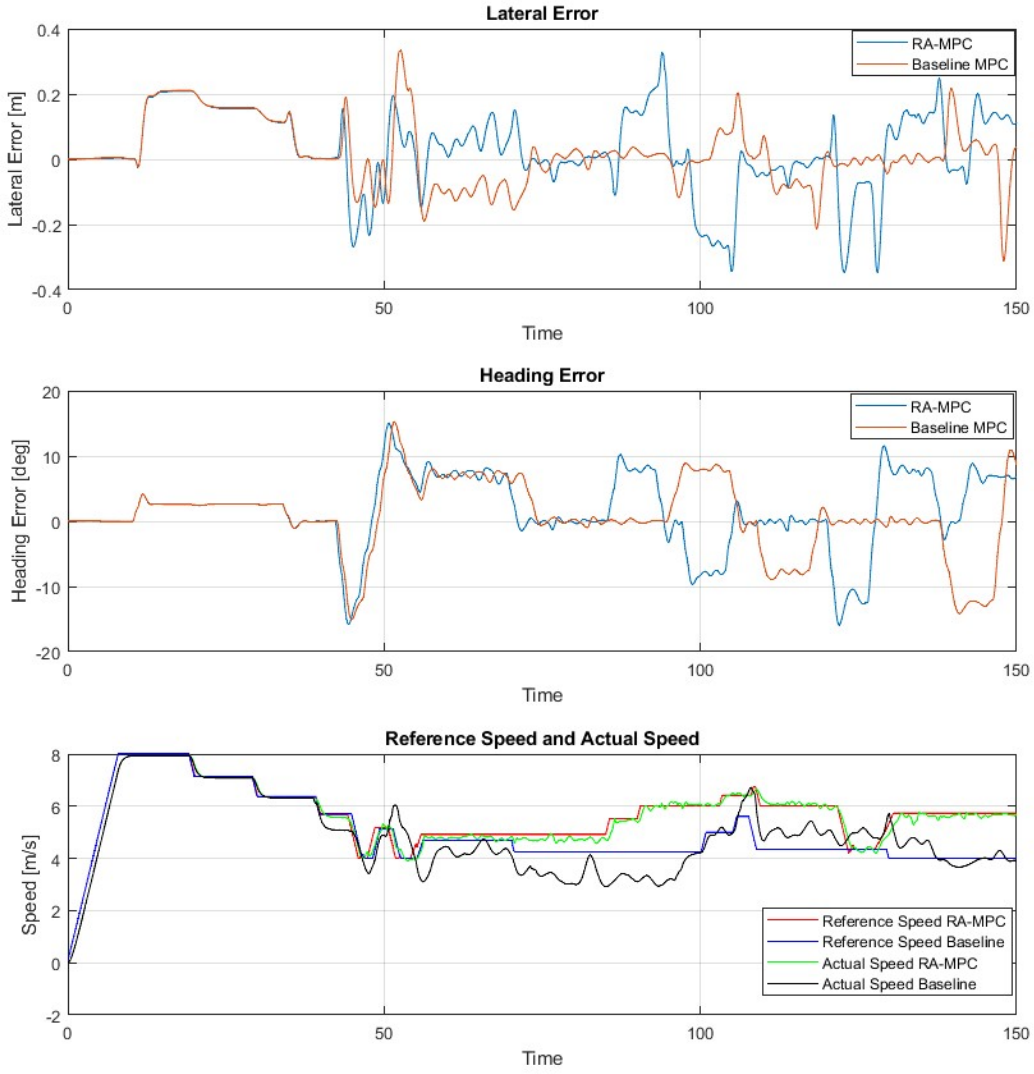
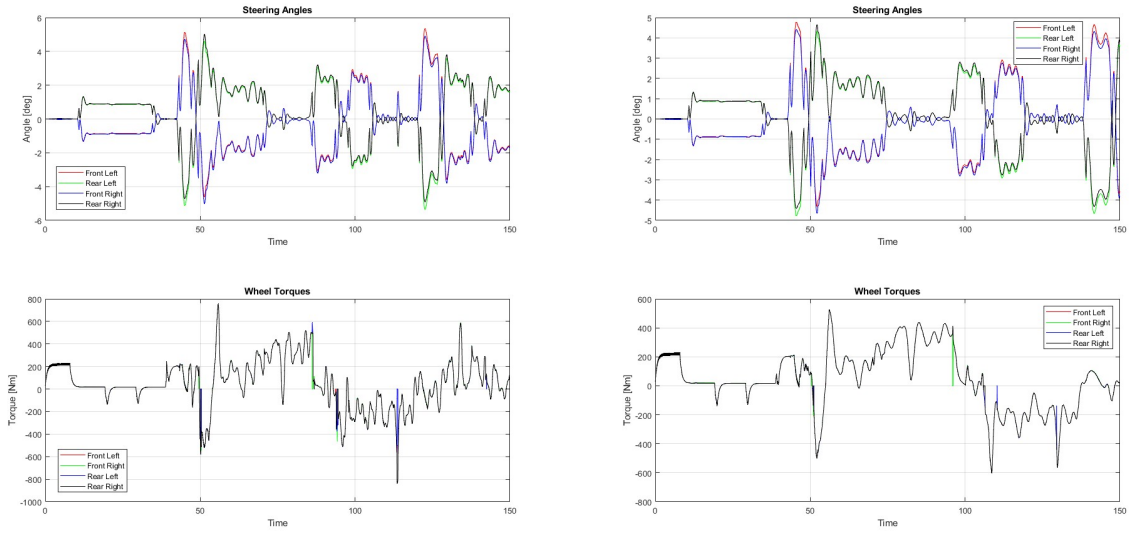


Figure 3.17: Trajectory Tracking Results - Long Off-road Track with RA-MPC - High μ



(a) Torque and steering inputs using RA-MPC (b) Torque and steering inputs using base MPC

Figure 3.18: Controller Outputs - Long Off-Road Track

3.5.4 Short-Track Off-Road Scenario - Low μ

The performance of the RA-MPC formulation should also be tested in low μ conditions to see the effect and robustness of the controller in difficult scenarios. For the short track scenario the coefficient of friction is reduced to 0.55. This value is selected via trial and error to ensure that both baseline and RA-MPC formulations were able to track the path stably. Figure 3.19 shows the tracking performance. The controller is still able to effectively reduce the longitudinal velocity tracking error and reduce the RMS error from 0.7358 m/s to 0.4306 m/s. The lateral tracking error is only minorly improved from 0.2667 m to 0.2556 m. This improvement is smaller compared to the high μ case and suggests that the low μ is causing difficulties for the controller in terms of compensating for lateral performance.

Figure 3.20 shows consistently larger steering angles used by the RA-MPC controller compared to the baseline. This is due to the vehicle compensating for the bank angles of the road. The torque output of the RA-MPC controller shows a large torque around the 14 s mark to compensate for the large inclination angle.

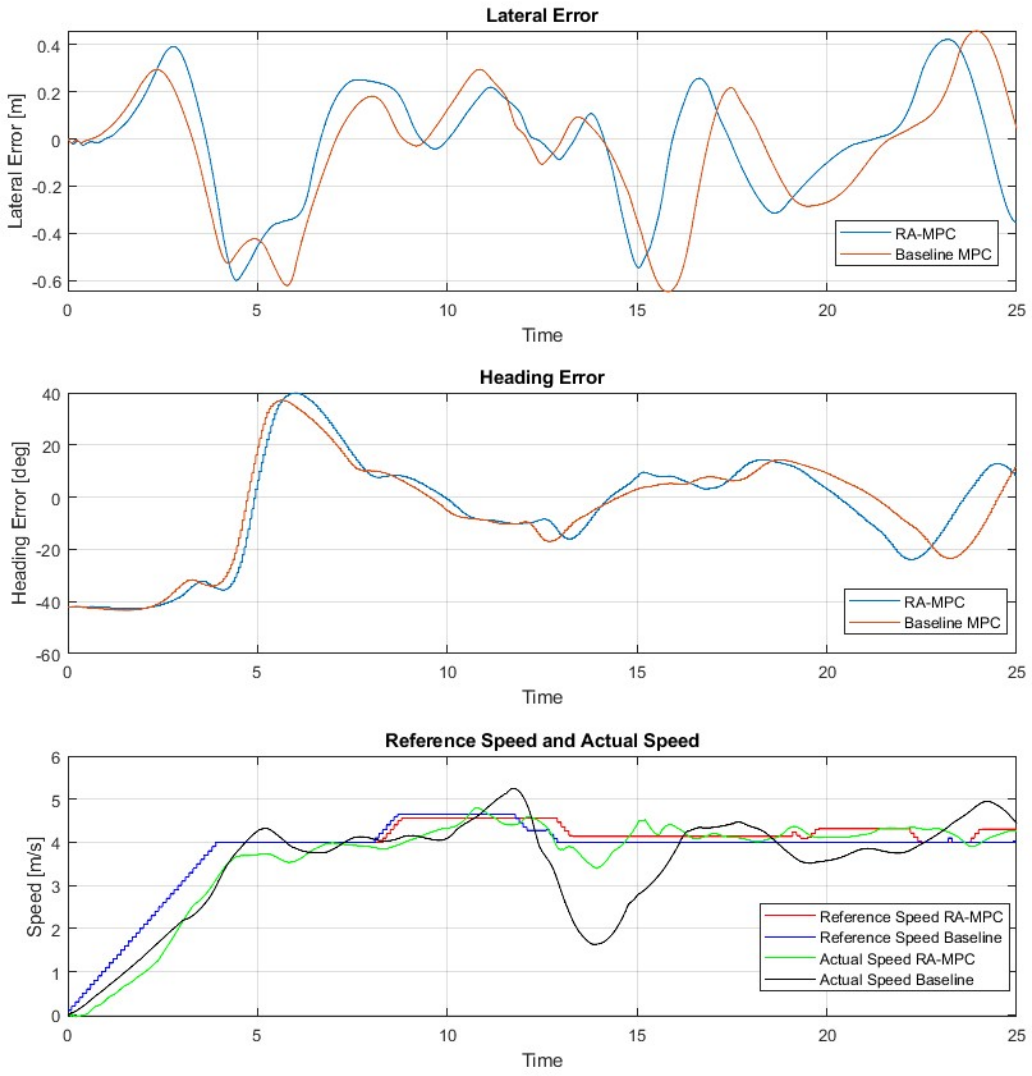
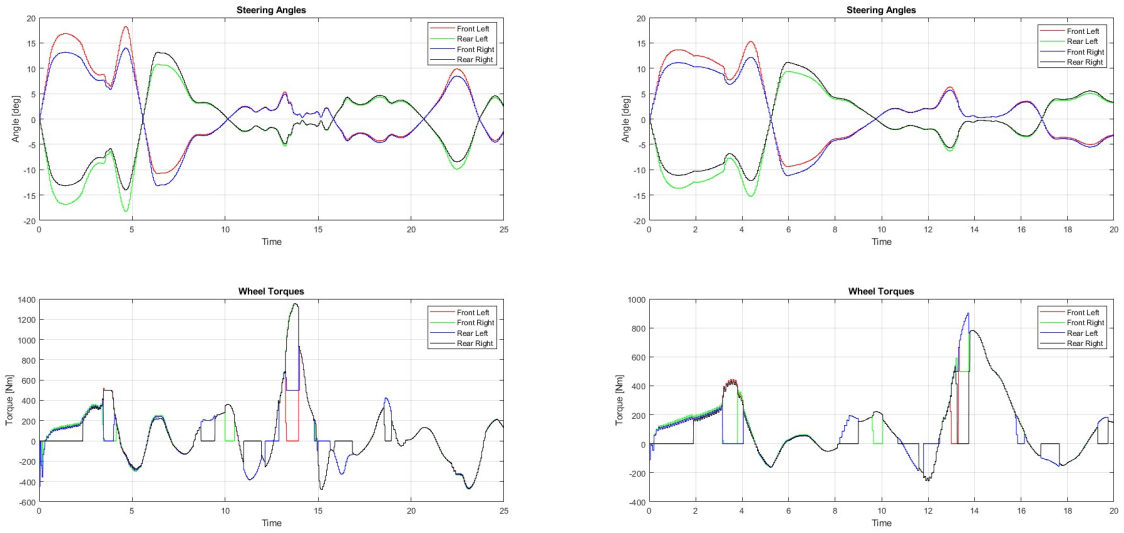


Figure 3.19: Trajectory Tracking Results - Short Off-road Track with RA-MPC - Low μ



(a) Torque and steering inputs using RA-MPC (b) Torque and steering inputs using base MPC

Figure 3.20: Controller Outputs - Short Off-Road Track - Low μ

3.5.5 Long-Track Off-Road Scenario - Low μ

Similarly, the RA-MPC formulation is tested in low μ conditions on the long off-road track. In this scenario the coefficient of friction is reduced to 0.40. This value is lower than the short track off-road scenario due to the smaller bank and inclination angles of the terrain. Figure 3.21 shows the tracking performance. The controller is able to effectively reduce the longitudinal velocity tracking error and reduce the RMS error from 0.6325 m/s to 0.1748 m/s. The lateral tracking error is improved from 0.1128 m to 0.0817 m.

Figure 3.22 shows smaller maximum steering angles used by the RA-MPC controller compared to the baseline. This is due to the vehicle proactively using slightly larger steering angles to compensate for the bank angle of the road instead of waiting for error to build causing a larger single steering correction. Overall, the torque output of the two controllers are similar with RA-MPC requesting larger torques to compensate for the inclination angle of the road.

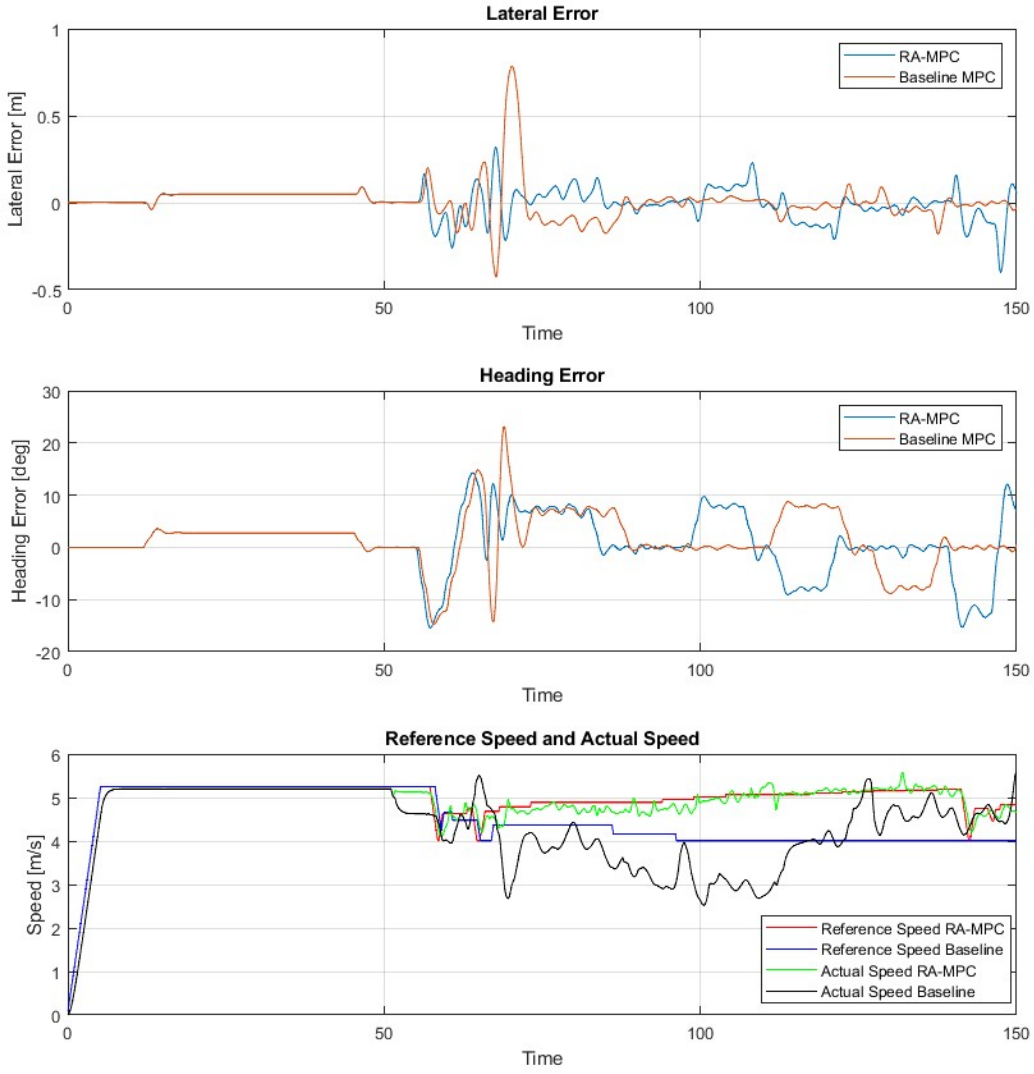
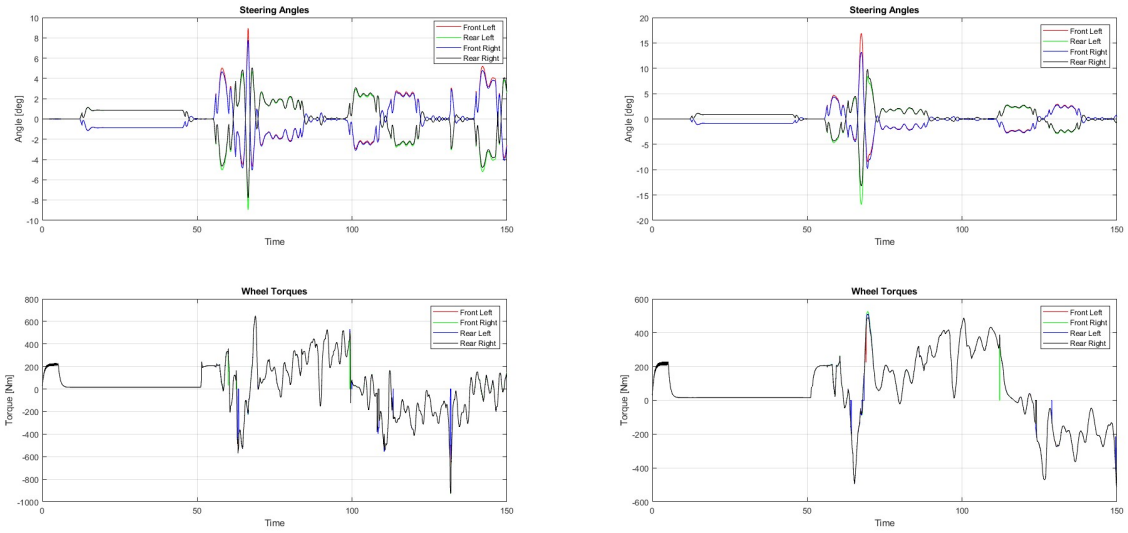


Figure 3.21: Trajectory Tracking Results - Long Off-road Track with RA-MPC - Low μ



(a) Torque and steering inputs using RA-MPC (b) Torque and steering inputs using base MPC

Figure 3.22: Controller Outputs - Long Off-Road Track - Low μ

3.5.6 Computational Performance

Figure 3.23 shows the computational requirements of the RA-MPC controller during the short off-road track simulation. The runtime of the controller is measured using the MATLAB's *tic* and *toc* functions to time the entire controller function. This includes model creation, linearization, discretization, as well as optimal control problem construction and solving. Clearly the controller is able to run fast enough to stay within the sampling time of 0.050 s. This means that the formulation should be suitable for real-time operation on an experimental vehicle. Optimizations could be made on the current code to further improve its computational efficiently. The overall mean runtime of the controller is 0.0113 s per iteration. The adaptive speed planner adds negligible runtime to this with an average runtime of just 0.00000533 s per iteration.

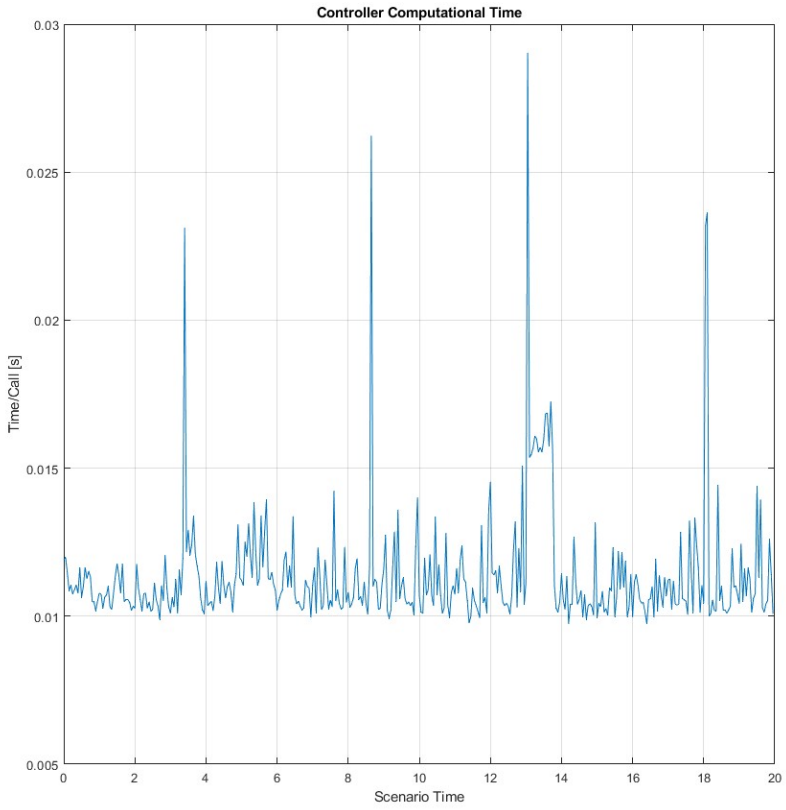


Figure 3.23: Computational time required for RA-MPC controller

3.6 Summary and Controller Shortcomings

Overall, results of the physics-based off-road controller showed that the inclination angle is very important for reliable and robust velocity profile tracking. The controlled system proved to be less sensitive to bank angles in the road; If the bank angle can not be measured accurately, it may be better to not directly model it in the prediction model of the vehicle and ensure the feedback nature of the controller compensates adequately. The controller formulation proposed also showed to be highly efficient and feasible for real-time operation.

Using road angles in the off-road MPC controller is inherently challenging due to several factors. In practical situations, precise angle measurements are not available, necessitating the use of estimation methods. The difficulty is exacerbated by the fact that off-road

conditions typically result in the vehicle's wheels being on different planes, unlike the uniform surface of a flat road. This uneven terrain makes it difficult to determine a reference point for the angle of the road; The uneven terrain also introduces significant instantaneous roll motion, contributing to noise and complicating accurate angle estimation. Although there are academic papers [46, 47] focused on road angle estimation, they predominantly address scenarios involving flat, smooth roads, leaving a gap in the literature for effective methods applicable to the unpredictable and variable nature of off-road environments.



Figure 3.24: Each tire is on a plane that is oriented uniquely. Defining one plane for vehicle bank and inclination angles to be in reference to is difficult.

Furthermore, as discussed in the literature review, significant dynamics are added to the vehicle in off-road scenarios due to the soft ground the tires may be interacting with. This in turn effects the amount of force that a tire can generate to move the vehicle both longitudinally and laterally. In the current formulation these dynamics are entirely neglected. While results show that acceptable performance is achievable, a strategy to compensate for the neglected dynamics may provide even better tracking performance. The strategy presented in chapter 4 will explore both a practical way to account for road angles and unmodelled dynamics simultaneously in a data-driven manner.

Chapter 4

Learning-Based Off-Road Controller Formulation and Results

4.1 Learning Based MPC Framework

Results in Chapter 3 highlighted the difficulty of estimating of the road angles in an off-road scenario; A sophisticated estimation module would be required to effectively utilize the road topography. The approach presented in Chapter 3 also neglected dynamics between the tire and terrain that could become non-negligible in certain circumstances. It is desired that these unmodelled dynamics could all be compensated for using a single data-driven learning based module. This module should be able to predict the behavior of the vehicle at the state level and provide insight into how much error may exist in the LTV model predictions.

The error in the prediction model can be thought of as constant for a small time period of a single prediction step. This makes it straightforward to model the prediction error as some linear offset from the output of the physics-based model. Looking at the $\bar{D}$ vector derived earlier, this vector contains the disturbance the vehicle is subject to over the entire prediction horizon. In the physics based model, this included dynamics for the bank and inclination angles, curvature of the road, as well as the offsets from the the non-linearities within the Taylor series approximation of the prediction model. With the learning based formulation, the bank and inclination angles are removed from the makeup of the $\bar{D}$ vector, with the inclusion of the unmodelled dynamics predictor, $UDP(k)$, the vector now becomes:

$$\bar{\mathbf{D}} = \begin{bmatrix} d(k) + UDP(k) \\ d(k) + UDP(k+1) \\ d(k) + UDP(k+2) \\ \vdots \\ d(k) + UDP(k+N_p) \end{bmatrix} \tag{4.1}$$

It should be noted that the disturbance coming from the linearization of the model is assumed to be constant due to the assumption that the state of the system is near the operating point it is linearized around and the sampling time is fast. The UDP terms however can be time varying depending on the learning technique used and computational resources available. The overall model for a single time step is now expressed as:

$$X(k+1) = A_d(k)X(k) + B_d(k)U(k) + d(k) + UDP(k) \tag{4.2}$$

A learning-based control topology is presented in Figure 4.1. Key to this topology is the flexibility of the data-driven module. In this topology, all unmodelled dynamics will be treated as a modelled disturbance within the MPC controller as discussed above. A learning-based module tells the controller at each time step in the horizon the dynamics that need to be added to the original prediction model to improve its accuracy. This level of abstraction has the advantage of being compatible with many learning techniques. For example, any unmodelled dynamics coming from a learning-based module can be included in the system model developed in Section 3.2 through the $d(k)$ vector within the prediction model. The modelling of the unaccounted for dynamics this way and propagation through the prediction horizon of the controller should provide better prediction results.

Advantages of this framework are the ability to account for various sources of unmodelled dynamics simultaneously, as well as the computational efficiently of the approach. In the off-road scenarios presented, the bank and inclination angles effect as well as the tire model inaccuracy can be accounted for through the states of u, v and r. A learning module can predict these states and compare it to the linear model used in the controller. An offset that accounts for the missing components in the linear model can then be applied to improve the prediction quality. This process includes simple mathematical operations to propagate the dynamics of the internal MPC model for comparison and no added complexity to the optimization problem. Furthermore, this approach looks to utilize readily available signals to reduce the need for complex and noise prone estimations. Signals such as Inertial Measurement Unit (IMU) accelerations, vehicle speed, wheel speeds, and suspension travel sensors can all be used to estimate the forces the vehicle is subject to

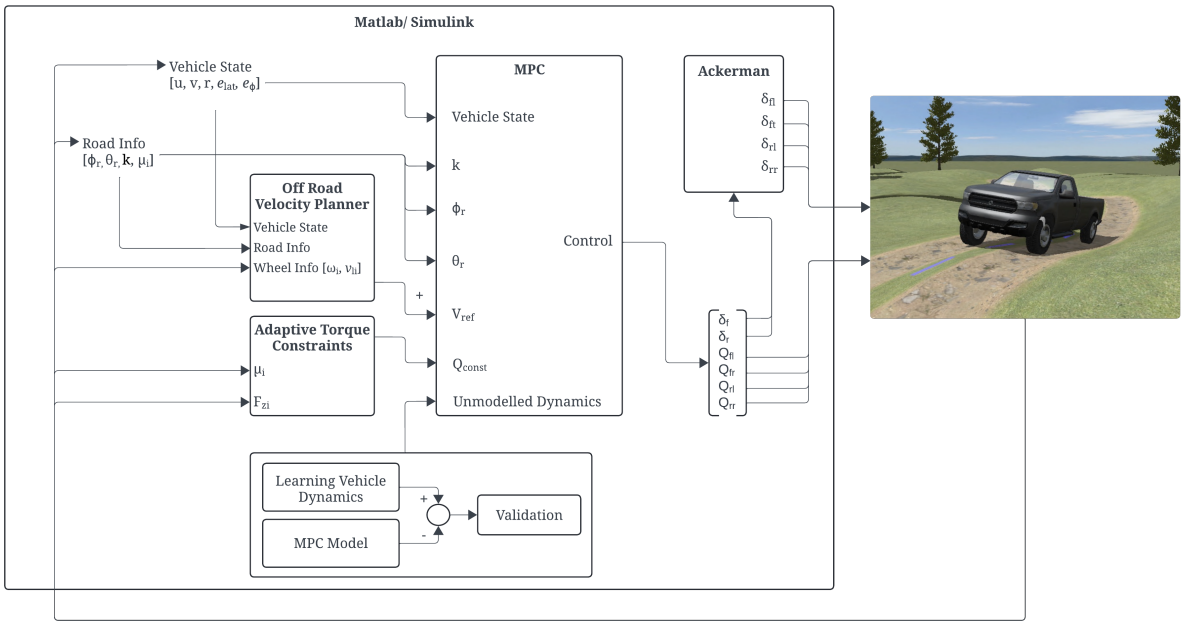


Figure 4.1: Learning Based MPC Topology

without the need for expensive hardware. With recent improvements in available compute power within modern vehicles, it is plausible that optimized learning algorithms like GPR can run online in a cost efficient manner.

The one drawback of using learning based methods is the need to validate the accuracy and reliability of the predictions. With data driven methods there is no guarantee that the predictions will be feasible if the model is not well trained for the current conditions. Depending on the learning technique used, it may be a non-trivial task to evaluate the accuracy of the predictions. This step is necessary to ensure the safety of the system and ensure that unique scenarios do not lead to unpredictable controller outputs.

4.2 Gaussian Process Regression Based Prediction

Gaussian Process Regression (GPR) is a powerful tool in this application due to its inherent ability to quantify the uncertainty in its predictions. In this application it is desired that predictors for the yaw dynamics, r, lateral dynamics, v_{lat}, and the longitudinal dynamics, u, are trained to estimate the true dynamics of the vehicle compared to the internal MPC

prediction model. Using the predictors, the unmodelled dynamics can then be estimated and added to the MPC formulation. It is important to note that the unmodelled dynamics can be implicitly accounted for in the states of the vehicle. Ideally the inputs to the GPR model are able to implicitly map out the vehicle dynamics space to account for unmodelled effects like tire-soil interaction, road angles, and changing friction.

In order to train the vehicle dynamics GPR models, data first needs to be collected. To do this, the features that will be used to predict the behavior of the state variables being estimated should be selected. While many formal methods could be used for this process, a good starting point is examining the dynamic equations derived in Chapter 3. First beginning with the r predictor, equations 3.6, 3.9, and 3.10 show that the evolution of the yaw rate of the vehicle will be a function of the following variables, $\delta_f, \delta_r, u, v_{lat}$ and of course the previous r state. Since it is also desired that the vehicle compensate for road angle dynamics, there needs to be a method of including this information implicitly in the training data. This can be achieved by adding IMU linear accelerations to the training data. In theory, the distribution of gravity to the components of the acceleration of the vehicle should be able to inform the vehicle of any large inclination or bank angles the vehicle faces. Although lateral velocity is not readily available and requires estimation, many works have examined methods of estimating it to a reasonable level of accuracy. In CarSim, the state is readily available to use and a similar analysis can be done to determine the GPR inputs that should be used to predict it. Since the equations say that the lateral velocity is a function of the same variables as the yaw rate, the same inputs will be used to train and predict the state.

In terms of the longitudinal dynamics, u, the variables need to be selected according to the main components of equation 3.4. The main components of this equation will be the combined torques of the front and rear axles, previous vehicle speed u, and acceleration in the x direction according to the IMU to compensate for any inclination angles in the road. The vertical acceleration of the IMU in the z direction is also included since the amount of tire force the vehicle can generate is a function of vertical load at the tire. This IMU measurement should provide an idea of how this may be changing with respect to the terrain. Ideally with these data points, GPR is able to approximate the state transition functions for each variable. In particular, the goal is for the estimator to implicitly compensate for the linear tire model and angles of the road. It is important to not include too many variables in each GPR as if the problem has too large of a dimension the search space becomes too large to give meaningful and accurate predictions in a time efficient manner.

The one drawback of using GPR in this application is the computational complexity of the learning method. As the amount of training data increases, so do the computational requirements for inference procedures. This is due to the number of calculations that need

to be done to determine the current operating points similarity to the data within the GPR model. This creates a delicate balance between storing as much data as possible to enhance the predictors ability and confidence with the computational requirements of the predictor. This is an area where a data management strategy could improve the practicality and effectiveness of using GPR. In order to prototype the system quickly, a dataset of four previous runs of the simulation scenario with the baseline controller is used to create the GPR models. This should provide enough data for the GPR controller to make accurate predictions.

4.3 GPR Integration and Tuning

The final remaining question when using GPR in the MPC controller is how to include the augmentations in the prediction horizon of the controller. As previously mentioned, this could happen in a time varying method if desired. In theory, if there were GPR predictors for each time step in the horizon, the unmodelled dynamics could be included for each respective time step; This method would allow for fast changing dynamics to be accounted for. Unfortunately, this would be a very time consuming and difficult task for GPR. This learning method is not built to model multi point time series data sequences and furthermore it would not be practical from a computational standpoint. The amount of data required to make accurate multi-step predictors would make the learning method infeasible. This learning method can still be useful for a single step approximation of the unmodelled dynamics. Since the control time step is reasonably small (50ms), it can be assumed that the unmodelled dynamics stay constant for a small period of time. This assumption is valid especially for smooth changes in bank and inclination angles in the road. In order to improve the accuracy of this approximation, the unmodelled lateral and yaw dynamics are only added as a constant value to the first ten steps of the 30 point prediction horizon. These dynamics are prone to quicker changes leading to more uncertainty throughout the prediction horizon. For example, the twentieth point in the horizons lateral and yaw unmodelled dynamics may not be similar to the first point in the horizons. The longitudinal velocity offset is however added to the entire length of the prediction horizon. Overall the dynamics of the vehicle longitudinally are slower and less affected by noise. The length of which an unmodelled disturbance is considered constant essentially now becomes a hyper-parameter of the GPR-MPC controller. This can be manually tuned using trial and error methods to ensure the best overall result.

During the integration of the GPR predictions, thresholds for the maximum allowable standard deviation need to be selected. These thresholds are used to qualify if the offset

calculated by the learning module will be used in the horizon of the MPC controller to augment the state predictions. These thresholds are difficult to tune due to the nature of the learning method. The more data that is present in the GPR predictor will lead to lower standard deviations if the data is similar to what the training data was. These thresholds should enable the module to exclude data that may be inaccurate or not adequately represented in the training data. When tuning these thresholds, a trade-off appeared between the lateral tracking and longitudinal tracking performances of the vehicle. If the threshold for u were to be too large, the offsets added to the MPC model made lateral tracking worse while drastically improving the velocity tracking of the vehicle. Conversely, if the bound were to be made more conservatively, it was possible to improve both the lateral tracking performance of the vehicle as well as improve the velocity tracking more modestly. These results suggest that future iterations of this controller formulation may benefit from a decoupled approach to lateral and longitudinal tracking to reduce the trade off effect. Furthermore, a more formulated approach at data management within the GPR predictor could ensure that the thresholds remain consistent and performance predictable by maintaining similar amounts of data in the state space.

4.4 CarSim Results and Discussion

4.4.1 Short and Long Track Off-Road Scenarios - High μ

To align with the previous simulation results, the newly formulated GPR-MPC controller will be tested in the same off-road scenarios as the baseline and RA-MPC controller. The scenarios focus on situations where the coefficient of friction is high (0.7 and 0.9), this means the main unmodelled dynamics result from road angles of the road. Since the vehicle is at low speed, and the terrain is non-deformable, the tire model should be relatively accurate. These tests will essentially look to mimic the results of the RA-MPC controller while also being able to compensate for other small sources of unmodelled dynamics that may have been overlooked.

The first scenario is the short-track off road scenario. Figure 4.2 shows the trajectory tracking for the lateral, heading, and velocity states respectively. The GPR-MPC controller is especially effective in reducing the velocity tracking error from a RMS value of 0.7348 m/s to 0.4063 m/s. This represents a solution that is just as effective as the RA-MPC at reducing the error and is more practical due to the readily available signals required to make the predictions. In particular, around the 14 s mark of the simulation the velocity tracking does not experience a large drop off in the velocity of the vehicle as it faces a large

uphill inclination angle. This showcases the ability of GPR to compensate for road angles through IMU measurements.

The results also show that the lateral tracking of the controller can be modestly improved with both the heading (20.6113° vs 19.7434°) and lateral (0.2769 m vs 0.2213 m) RMS errors reduced compared to the baseline controller. This scenario featured large and smooth bank and inclination angles that made compensating for them in the controller easier to do. In more aggressive scenarios, the compensation for these angles becomes more difficult as seen previously with the RA-MPC results. Due to the low speeds the vehicle is facing, as well as the compact nature of the terrain, the unmodelled dynamics at the tire are most likely small and overall negligible.

Looking at the control inputs of the GPR-MPC controller in Figure 4.3, the steering angles of GPR-MPC compared to the baseline are larger as expected. These larger steering angles help compensate for the lateral error and contribute to the improved results. Looking at the torque plot of the GPR-MPC controller, the torque requested leading up to the large inclination angle at 14 s in the scenario showcase the effect of modelling the external disturbance of the path. This scenario is particularly difficult because some tires lose traction due to the roughness of the hill. With the dynamic torque constraints, the controller is able to effectively divert the torque to the wheels with traction to climb the hill.

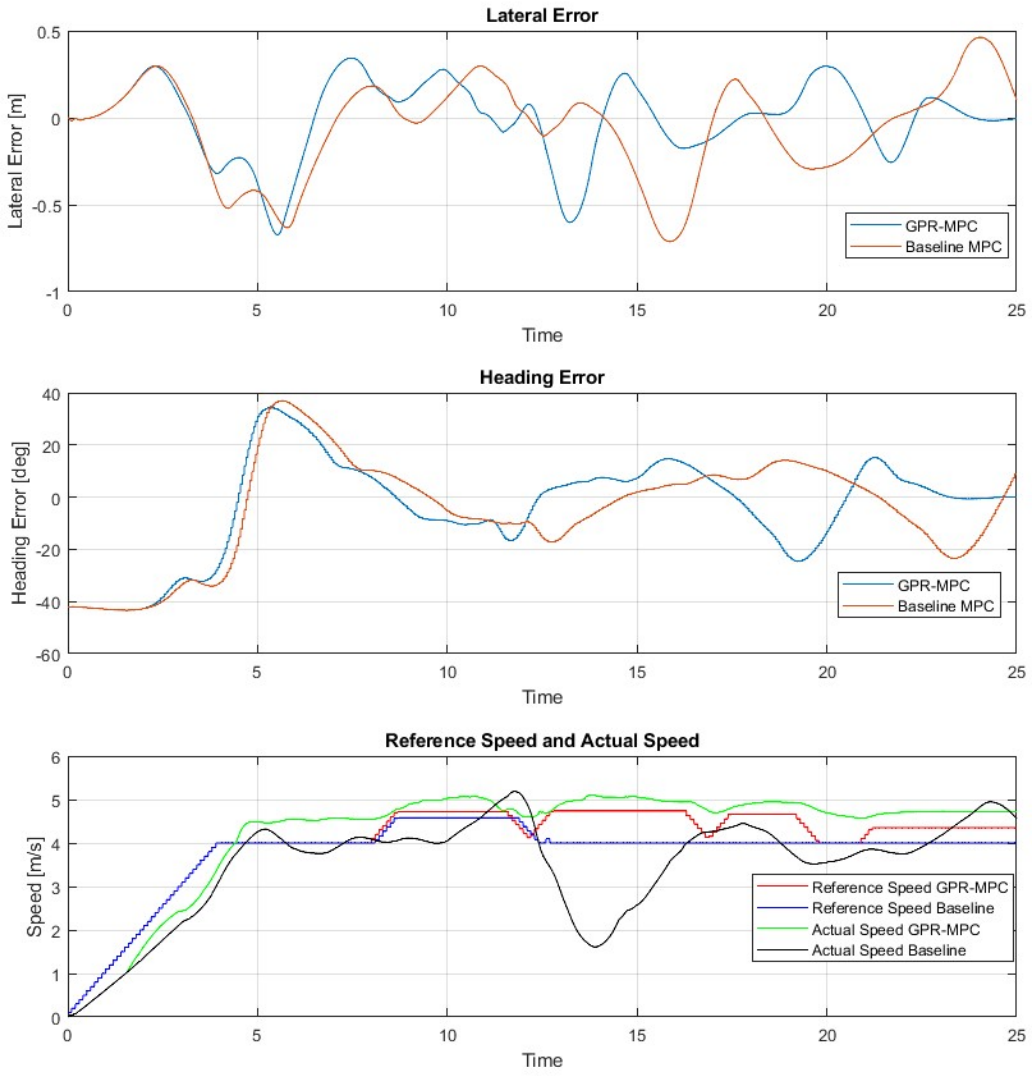
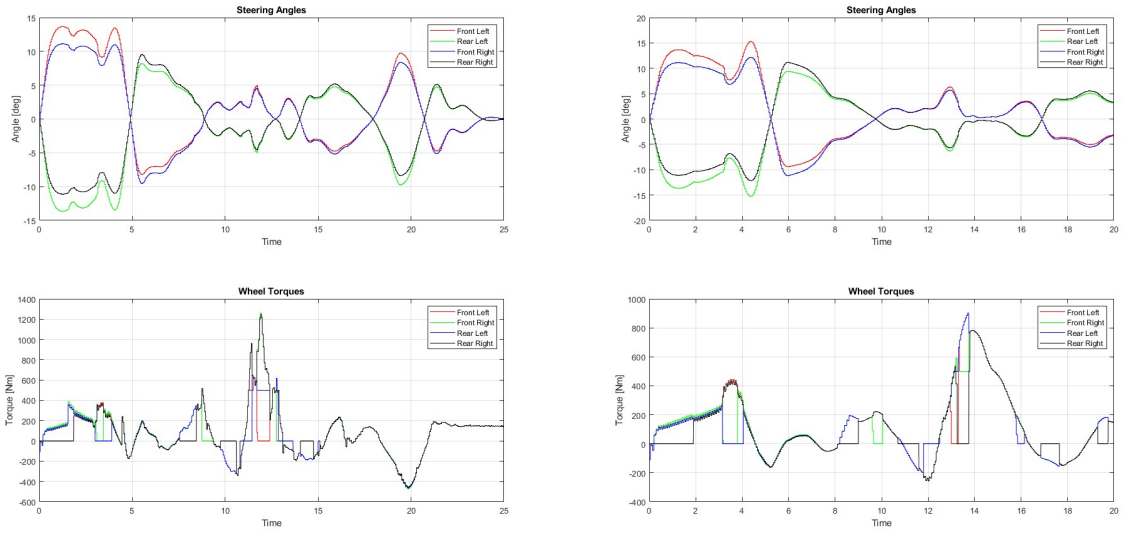


Figure 4.2: Trajectory Tracking Results - Short Off-road Track with GPR-MPC - High μ



(a) Torque and steering inputs using GPR-MPC (b) Torque and steering inputs using base MPC

Figure 4.3: Controller Outputs - Short Off-Road Track, GPR-MPC vs Baseline

The second scenario tested is the long track off-road scenario. As a reminder, this scenario features much rougher terrain with sharper but smaller inclination and bank angles. As seen with the RA-MPC controller, this makes the unmodelled dynamics more difficult to compensate for due to the high frequency noise in the predictions and inherent time delays in the system. This scenario particularly tests the robustness of the GPR-MPC control method.

Figure 4.4 shows that GPR-MPC has a modest effect on the lateral tracking of the vehicle in this difficult scenario. The lateral error RMS is slightly increased from 0.1011 m to 0.1262 m, and the heading error is slightly increased from 5.5948° to 5.7434° when comparing to the baseline formulation. In the figure it can be seen that some lateral error peaks are slightly reduced while others are slightly larger. Overall, the lateral tracking performance of the vehicle is comparable to the baseline. This scenario highlights that since the bank and inclination angles of the vehicle are small it is better for lateral tracking to reject the disturbance via the feedback nature of the vehicle. The aggressive angles of the road make it hard to improve the already solid performance by the baseline controller. When it comes to the velocity tracking of the vehicle, the RMS error is reduced from 0.6321 m/s to 0.3414 m/s. This is a much more notable improvement in the GPR-MPC

controller. This highlights GPRs ability to improve velocity tracking even in difficult scenarios. Overall, the GPR-MPC formulation achieves very similar results to the RA-MPC formulation without the need for complex estimators. A purely data-driven module has shown its ability to compensate for the dynamics coming from the road angles.

Controller outputs of the GPR-MPC controller in Figure 4.5 show torque spikes when the inclination angle of the road is increased to compensate for the extra forces on the vehicle. Controller steering angles are very similar to the baseline formulation with only slight modifications to the magnitude.

Results of the high μ cases showed that the GPR-MPC formulation can achieve similar results to the RA-MPC under similar conditions. The high μ results primarily focused on compensating for the angles of the path implicitly through the unmodelled dynamics estimator. Similarly, the GPR-MPC showed that in rougher scenarios with fast changing smaller bank angles, the dynamics are better off ignored and rejected by the feedback nature of the controller. The direct compensation for these angles in a high μ scenario caused over corrections that ultimately lead to the slightly higher lateral RMS errors. The velocity tracking was shown to be greatly improved under all circumstances even when the angle was subject to high frequency changes.

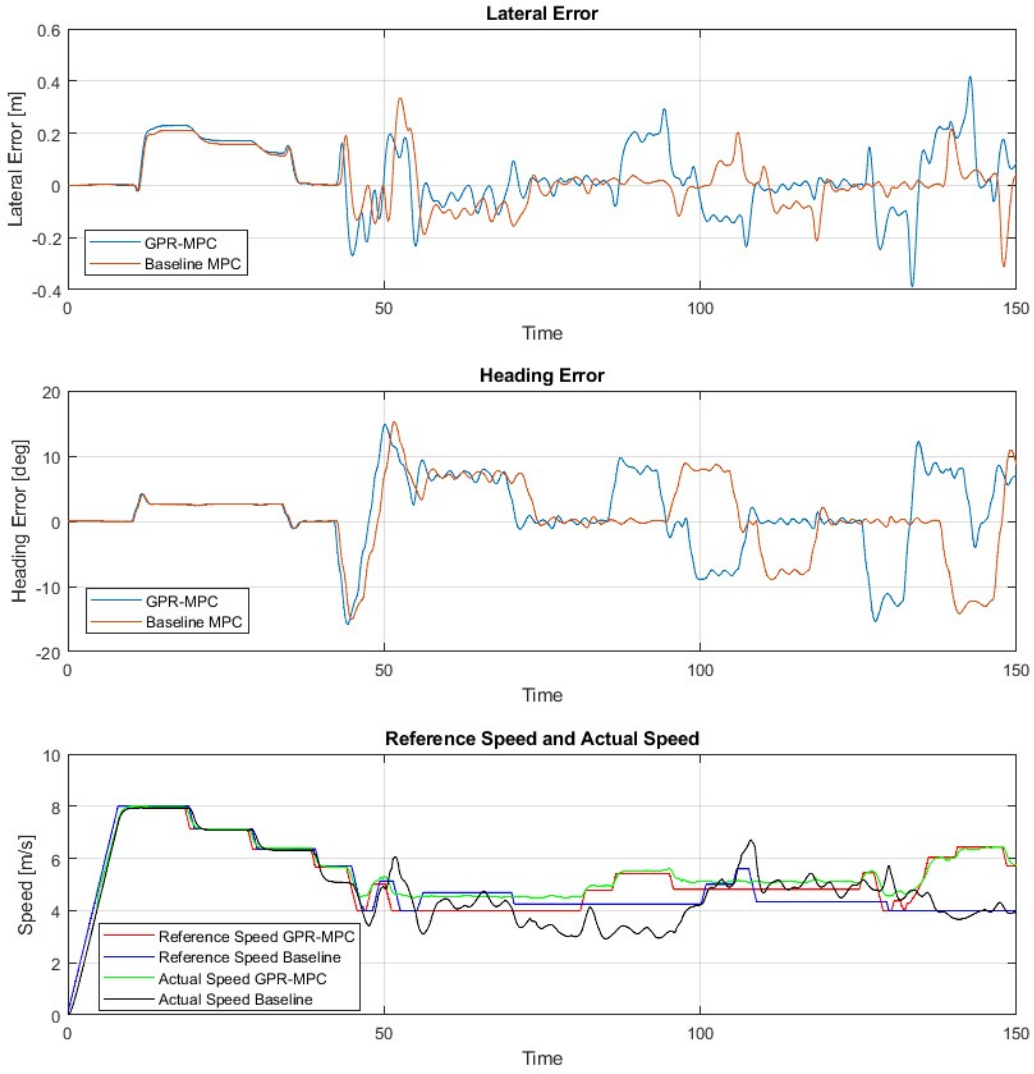
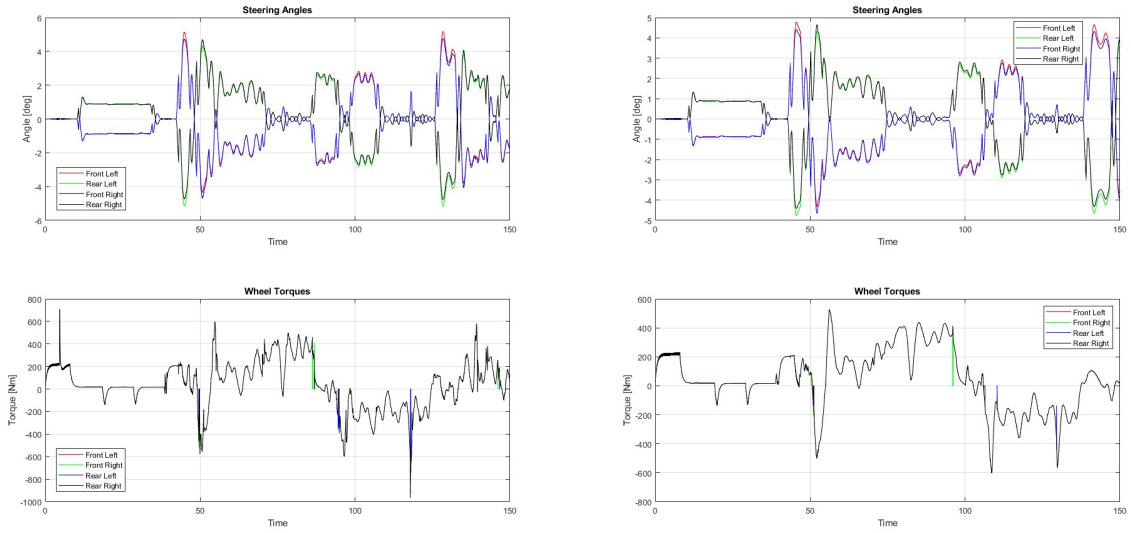


Figure 4.4: Trajectory Tracking Results - Long Off-road Track with GPR-MPC - High μ



(a) Torque and steering inputs using GPR-MPC (b) Torque and steering inputs using base MPC

Figure 4.5: Controller Outputs - Long Off-Road Track, GPR-MPC vs Baseline

4.4.2 Short and Long Track Off-Road Scenarios - Low μ

In order to subject the vehicle to higher amounts of unmodelled dynamics, the coefficient of friction will be reduced for the off-road scenarios. The physics based baseline MPC will be kept the same and therefore will have a large parameter mismatch in terms of the tire dynamics. This will introduce lateral dynamics that are difficult for the baseline to compensate for due to the limited lateral force the vehicle can actually produce in practice. In order for the GPR-MPC formulation to actively compensate for the new dynamics, data needs to be added to the training set. Using the baseline results as extra training data, new GPR predictors can be trained that have experience on low μ surfaces.

The first simulation scenario is the short track off-road scenario. Due to the large inclination and bank angles in this scenario, a reduced μ coefficient of 0.55 is selected. This should provide a more difficult scenario without making the terrain unfeasible to to follow, especially in the case of large inclination angles like the one seen in this scenario. Figure 4.6 shows the meaningful effect that GPR has in this scenario. The peak lateral errors are clearly reduced using GPR as well as the heading error. Overall, the RMS lateral error is reduced from 0.2667 m to 0.2376 m and the heading RMS error is reduced from 20.6127° to 20.1602°. The velocity tracking is also clearly improved and rejects the large

inclination angle disturbance effectively. The better tracking is further represented by a reduced velocity RMS error from 0.7358 m/s to 0.5409 m/s. Figure 4.7 shows the effect GPR has had on the optimized steering angles. The steering angles around some turns are smaller to compensate for the low coefficient of friction and sliding of the vehicle. Clearly the inclusion of GPR has allowed the vehicle to track the motion profile more effectively.

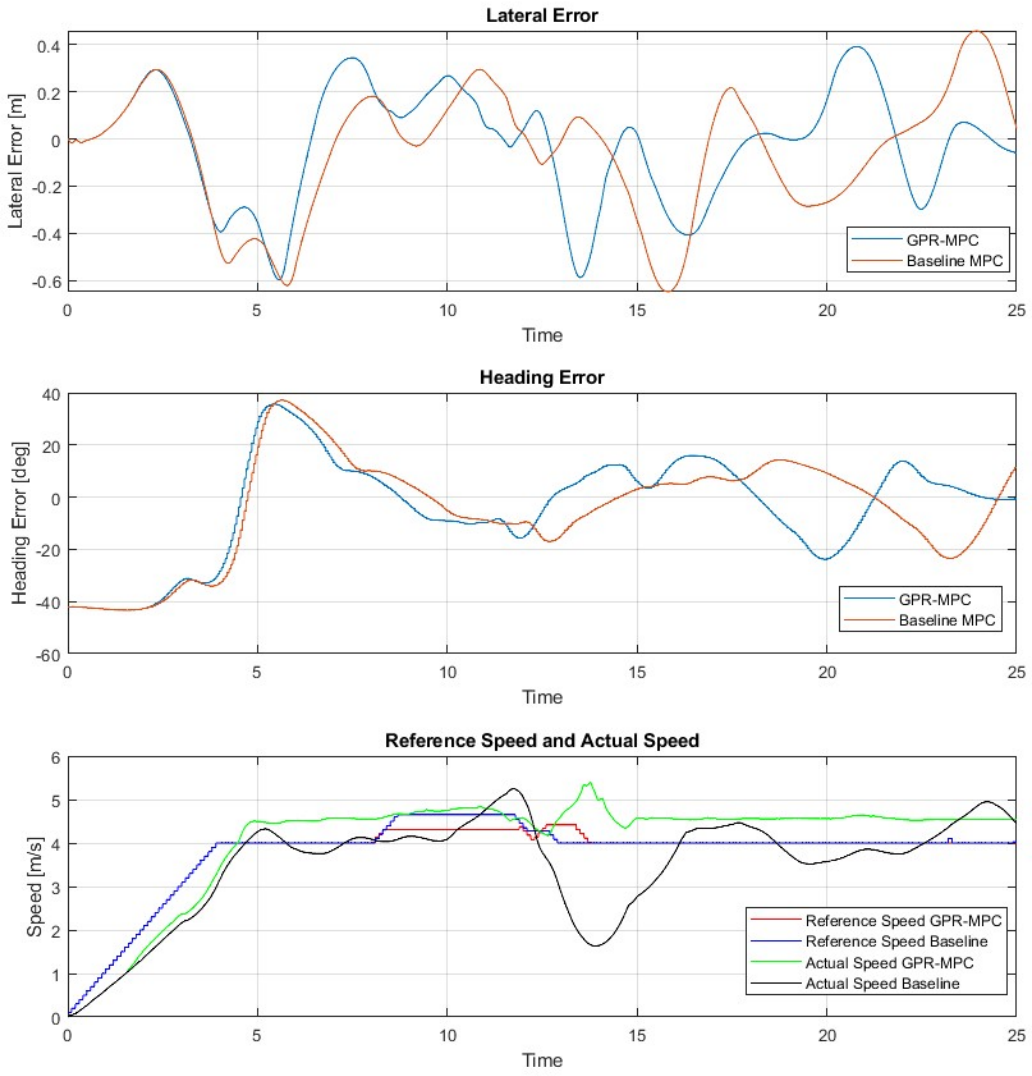
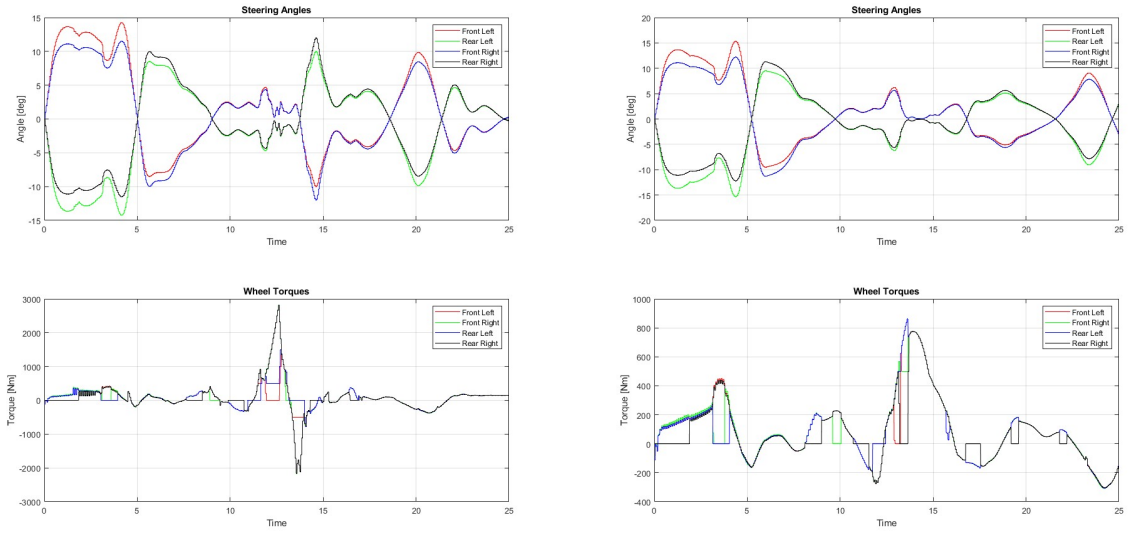


Figure 4.6: Trajectory Tracking Results - Short Off-road Track with GPR-MPC - Low μ



(a) Torque and steering inputs using GPR-MPC (b) Torque and steering inputs using base MPC

Figure 4.7: Controller Outputs - Short Off-Road Track, GPR-MPC vs Baseline - Low μ

The second simulation scenario is the long track off-road scenario. In this scenario, μ is further reduced from 0.9 to 0.4 to showcase the effectiveness of GPR-MPC. This scenario has smaller bank and inclination angles that make a lower coefficient of friction feasible. Figure 4.8 again shows the effectiveness of GPR-MPC. Using the baseline controller, the vehicle has a very large error when encountering a steep hill. The low friction and downhill effect made the vehicle make a steering manoeuvre that was too aggressive and caused the vehicle to momentarily drift. With GPR-MPC, the vehicle is not only able to overcome this part of the scenario but produce a very strong tracking result. The lateral RMS error of GPR-MPC is just 0.0834 m along with a small heading RMS error of 4.8270°. The velocity tracking RMS error of the vehicle is 0.2984 m/s. Figure 4.9 shows that GPR-MPC is effective at reducing the steering angle because of the higher than expected yaw rate and lateral velocity of the vehicle.

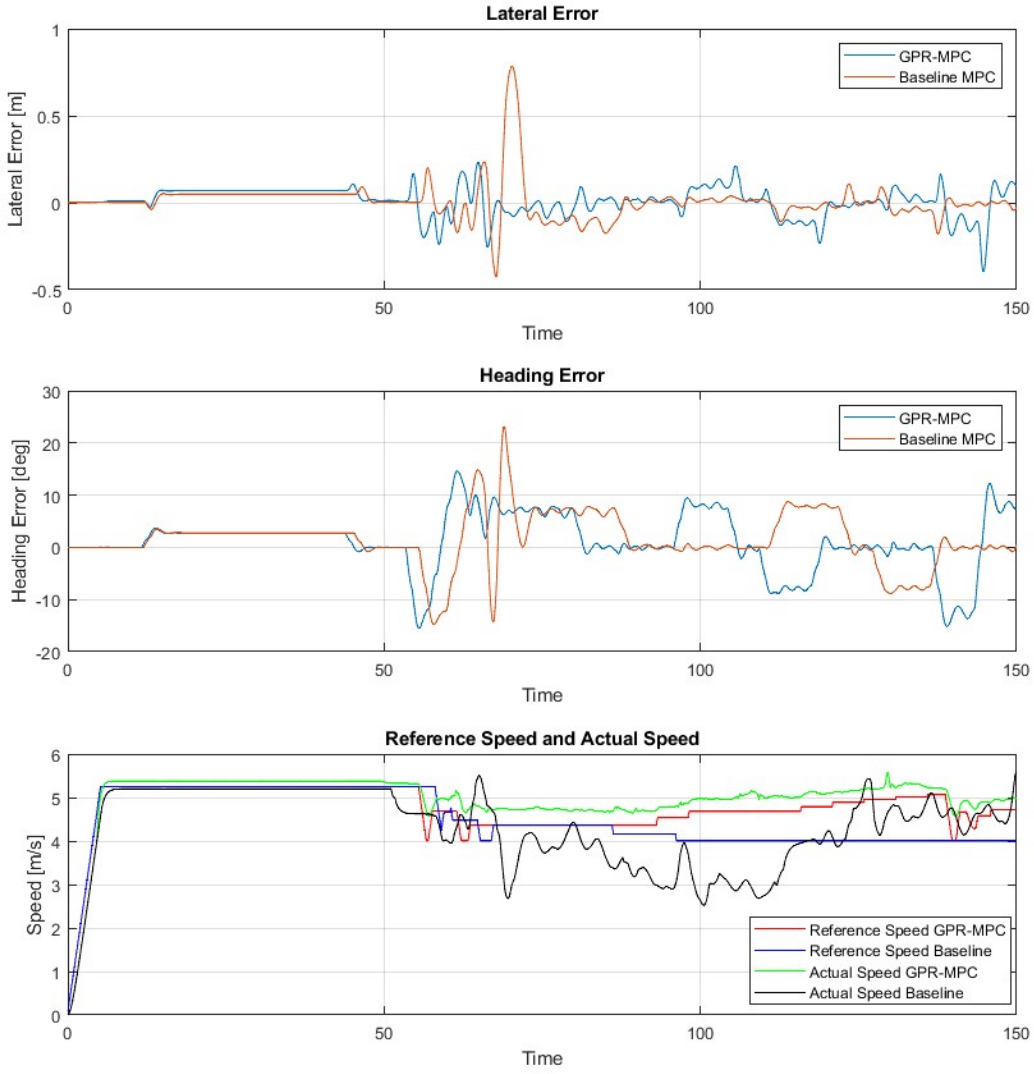
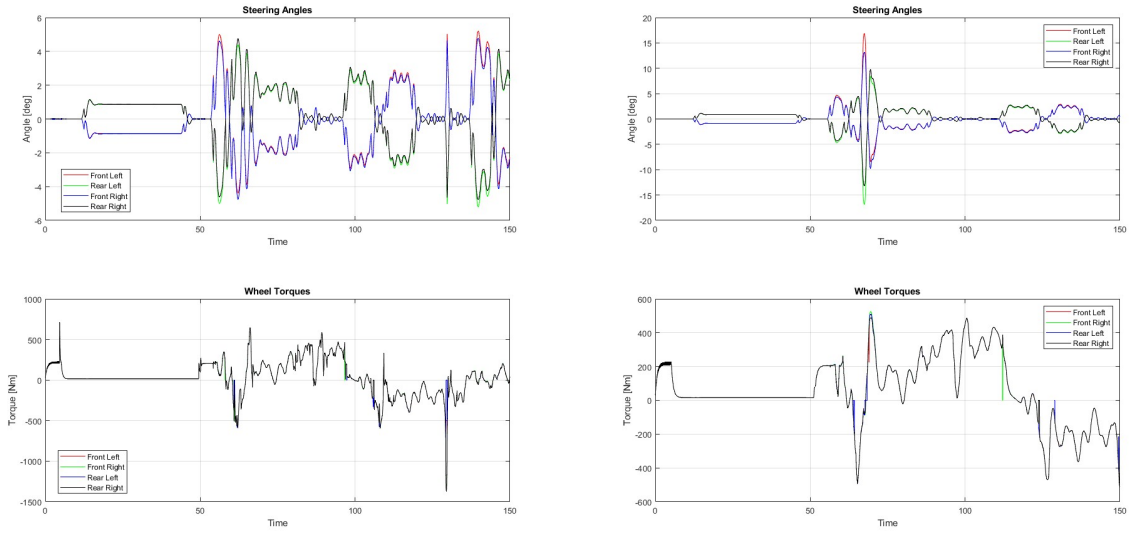


Figure 4.8: Trajectory Tracking Results - Long Off-road Track with GPR-MPC - Low μ



(a) Torque and steering inputs using GPR-MPC (b) Torque and steering inputs using base MPC

Figure 4.9: Controller Outputs - Long Off-Road Track, GPR-MPC vs Baseline - Low μ

4.4.3 Short and Long Track Off-Road Scenarios - Payloads and Low μ

Another interesting simulation scenario is the inclusion of a payload within the test vehicle to simulate off-road material transportation. This has the effect of changing the mass parameter of the vehicle model as well as the lateral and yaw dynamics of the vehicle due to the shifted CoG. In practice, this can not be easily added to the vehicle model each time the controller is used so the control method should be robust to the changes in the dynamics. As a first test, the short track off-road scenario will be modified to include a 800 kg payload that is centered in the truck bed. The centering of the payload in the truck bed should not move the CoG too drastically therefore the main parameter mismatch is the overall mass of the vehicle. Figure 4.10 shows the payload that is carried throughout the scenario. The baseline controller results are added to the GPR dataset to properly train the predictors on the increased mass of the vehicle.

Figure 4.11 shows that GPR is effective at reducing the lateral and longitudinal errors with the heavy payload in the vehicle. In terms of RMS errors the heading error is reduced from 20.4377° to 19.9002°, the lateral error is reduced from 0.3057 m to 0.2549 m and the



Figure 4.10: 800 kg Payload Centered in Truck Bed

velocity error is reduced from 0.9073 m/s to 0.5262 m/s. With the help of GPR-MPC the controller is able to achieve similar results as the unloaded test on low μ surfaces. The 800 kg payload clearly impacts the trajectory tracking difficulty in this scenario where there are large inclination and bank angles in the desired path. Figure 4.12 shows smooth steering and torques requests made by the GPR-MPC controller.

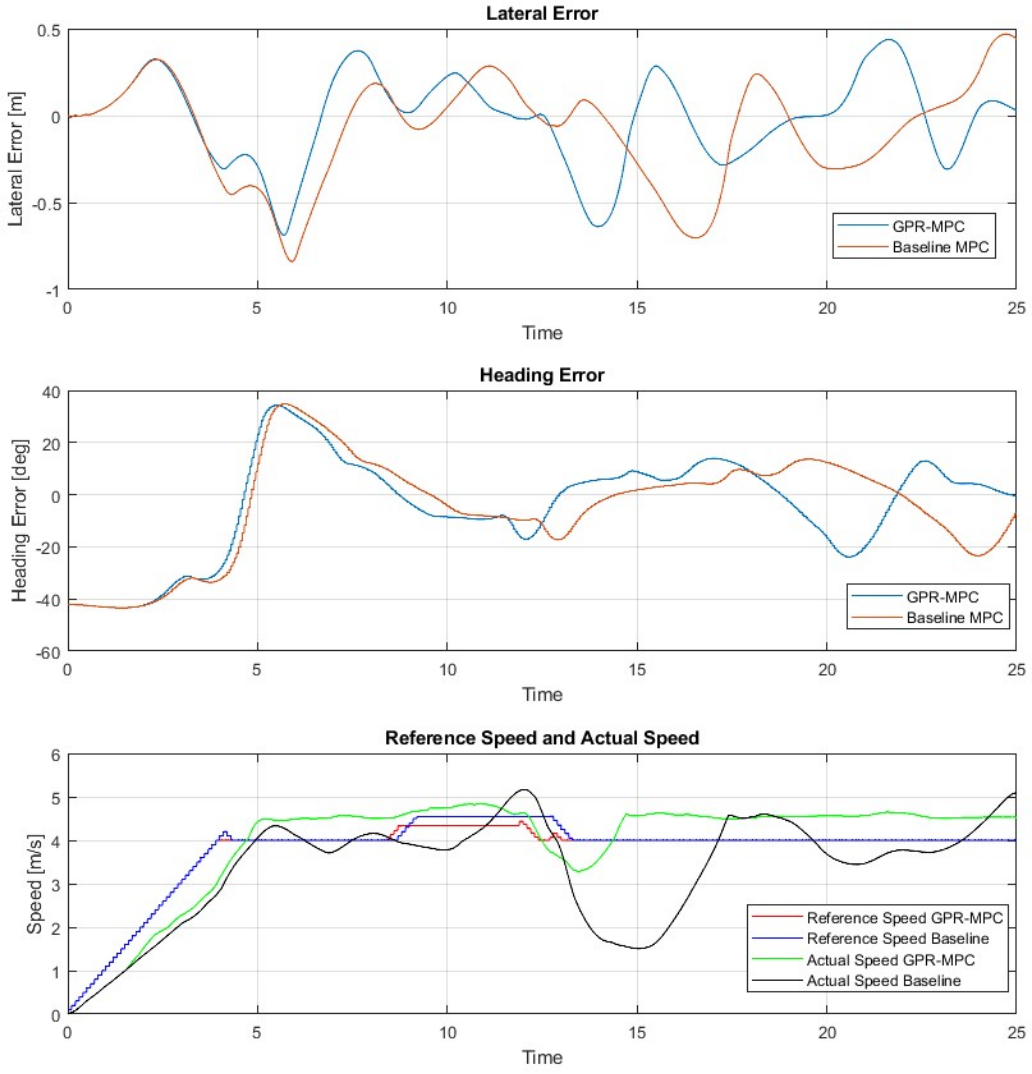


Figure 4.11: Trajectory Tracking Results - Short Off-road Track with GPR-MPC - 800 kg Centered Payload, Low μ

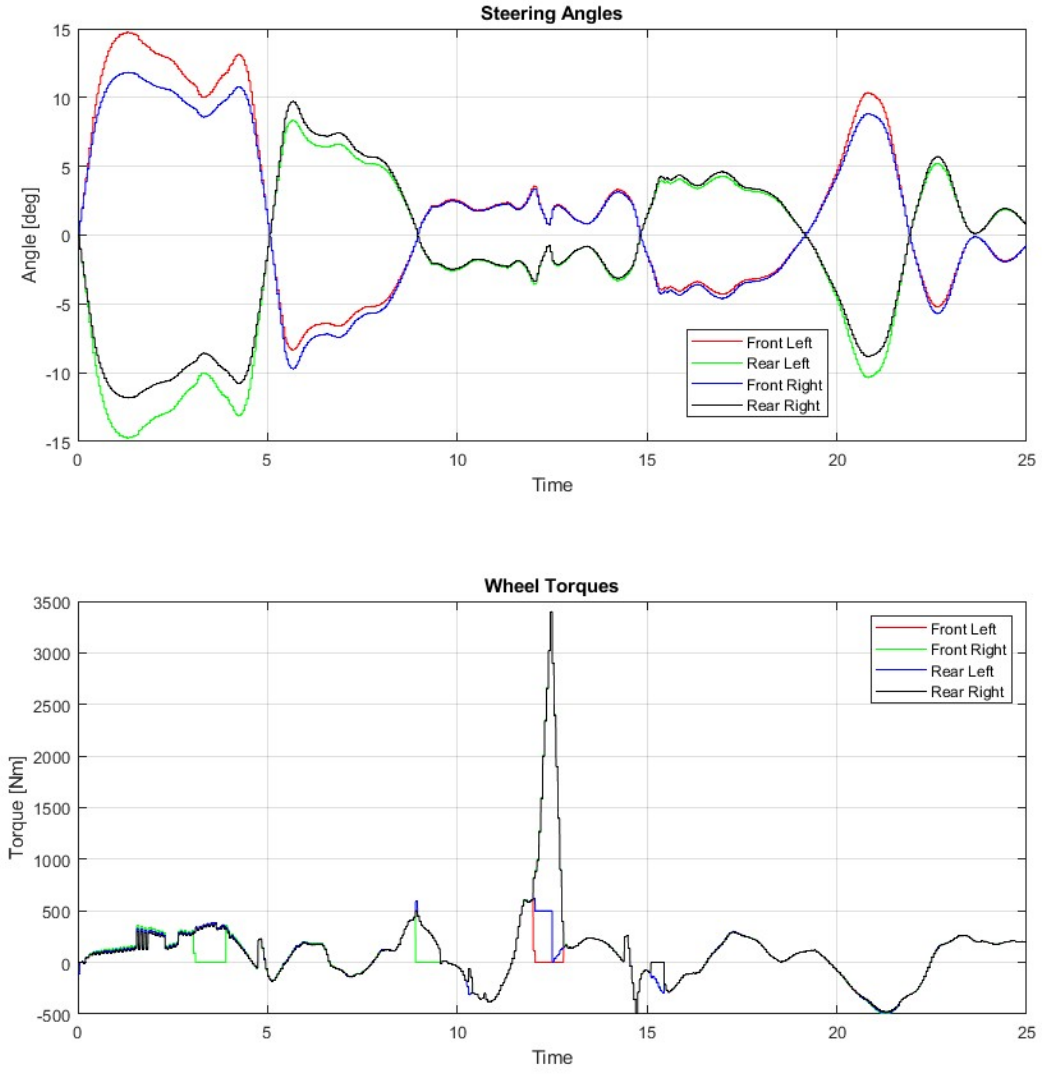


Figure 4.12: Controller Outputs - Short Off-Road Track, GPR-MPC - 800 kg Centered Payload, Low μ

Continuing on with the long track scenario, the 800 kg payload is again added to the low μ scenario. Figure 4.13 shows the tracking results of the scenario. As expected GPR-MPC is able to improve both lateral and velocity tracking. In this scenario it should be noted that the increased mass of the vehicle and smaller bank and inclination angles make the baseline controller quite effective. Since the mass of the vehicle is increased so is the amount of normal force at each tire. This means that the vehicle is able to create larger lateral and longitudinal forces on low μ surfaces leading to better tracking. Figure 4.14 shows a smaller steering angle used by the GPR-MPC controlller in the scenario compared to the high μ, no payload scenario. The figure also shows larger torques being requested by the controller to compensate for the increased mass of the vehicle. Overall, GPR-MPC is ensuring the stable and accurate tracking by ensuring aggressive steering angles are avoided and larger torques are used to track the desired speed.

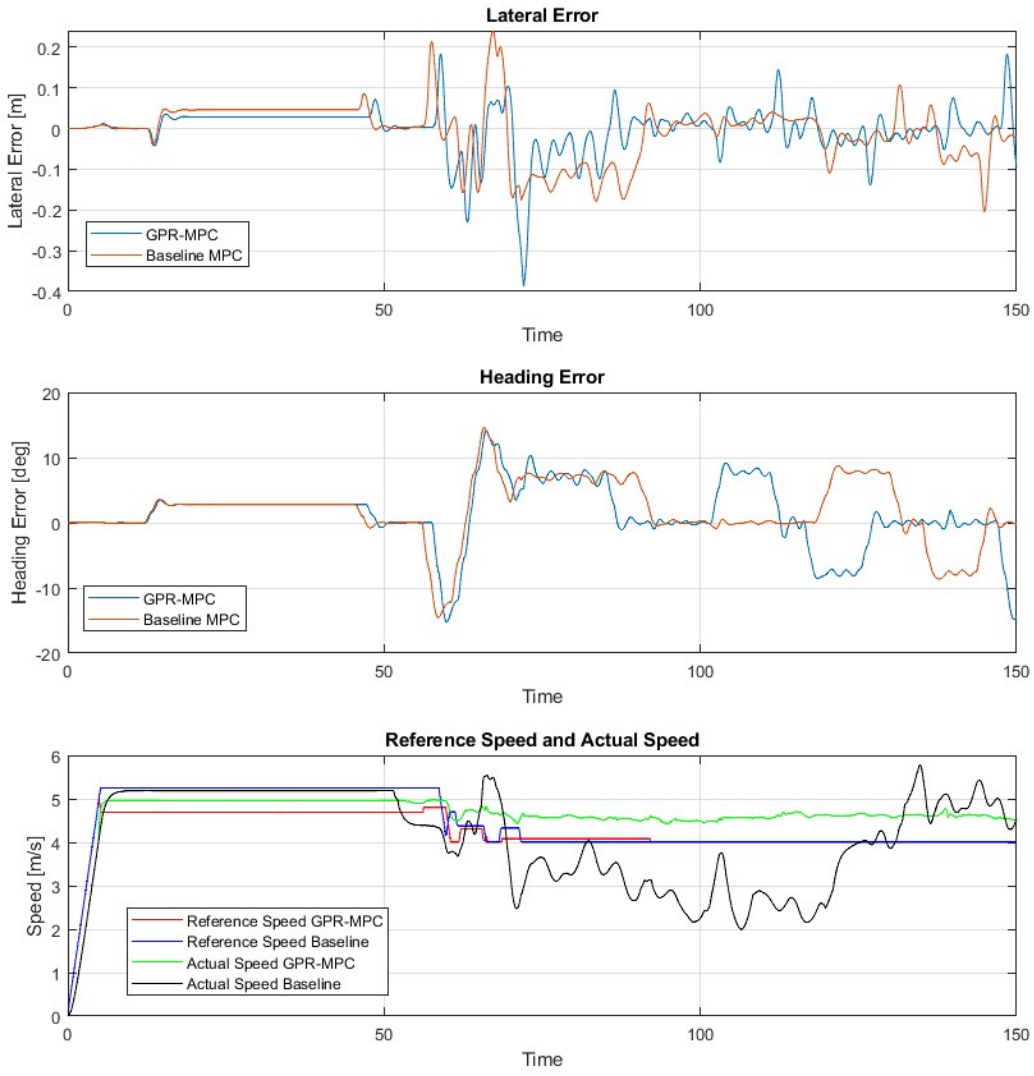


Figure 4.13: Trajectory Tracking Results - Long Off-road Track with GPR-MPC - 800 kg Centered Payload, Low μ

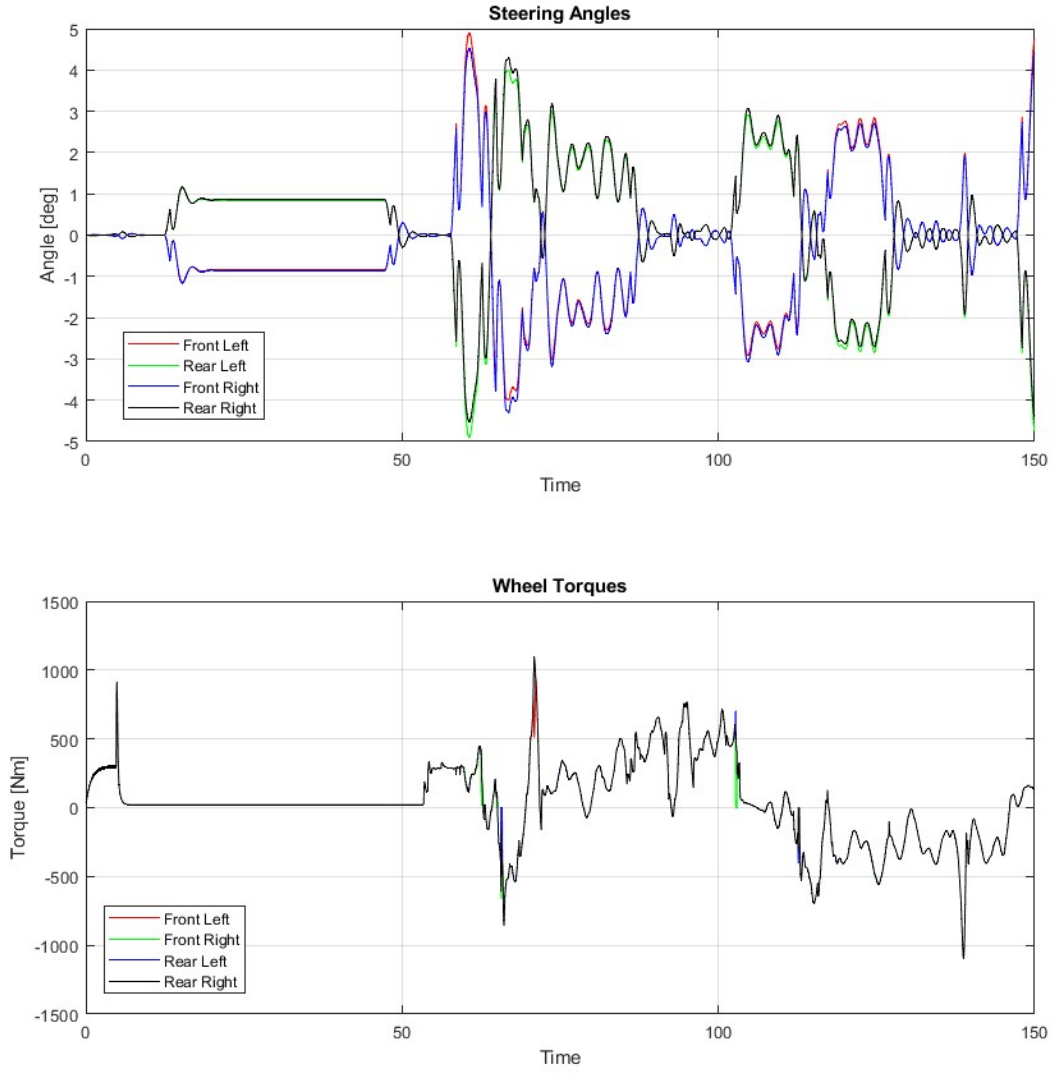


Figure 4.14: Controller Outputs - Long Off-Road Track, GPR-MPC vs Baseline - 800 kg Centered Payload, Low μ

Finally, in order to create an extremely challenging scenario the mass of the payload is increased to 1000 kg and shifted further back to the rear to make the vehicle less laterally stable (Figure 4.15). This should provide dynamics that can't be corrected for by only compensating for the road angles of the vehicle path. In order to focus on the added dynamics of the payload in this difficult scenario, a constant speed target of 4 m/s is selected.



Figure 4.15: 1000 kg Payload Uncentered in Truck Bed

Figure 4.16 confirms GPR-MPC's ability to compensate by stabilizing the system and tracking the given trajectory with small RMS errors. In a similar scenario, the RA-MPC and baseline formulations are unable to track the path given the uncertainty that was added with the uncentered payload. Figure 4.17 shows the steering angles and torques

requested by the GPR-MPC controller. It is important to note that the steering angles are kept within $\pm$ 10° even in the difficult conditions. The ability of the controller to limit the steering has enabled the vehicle to track the desired path without losing lateral stability.

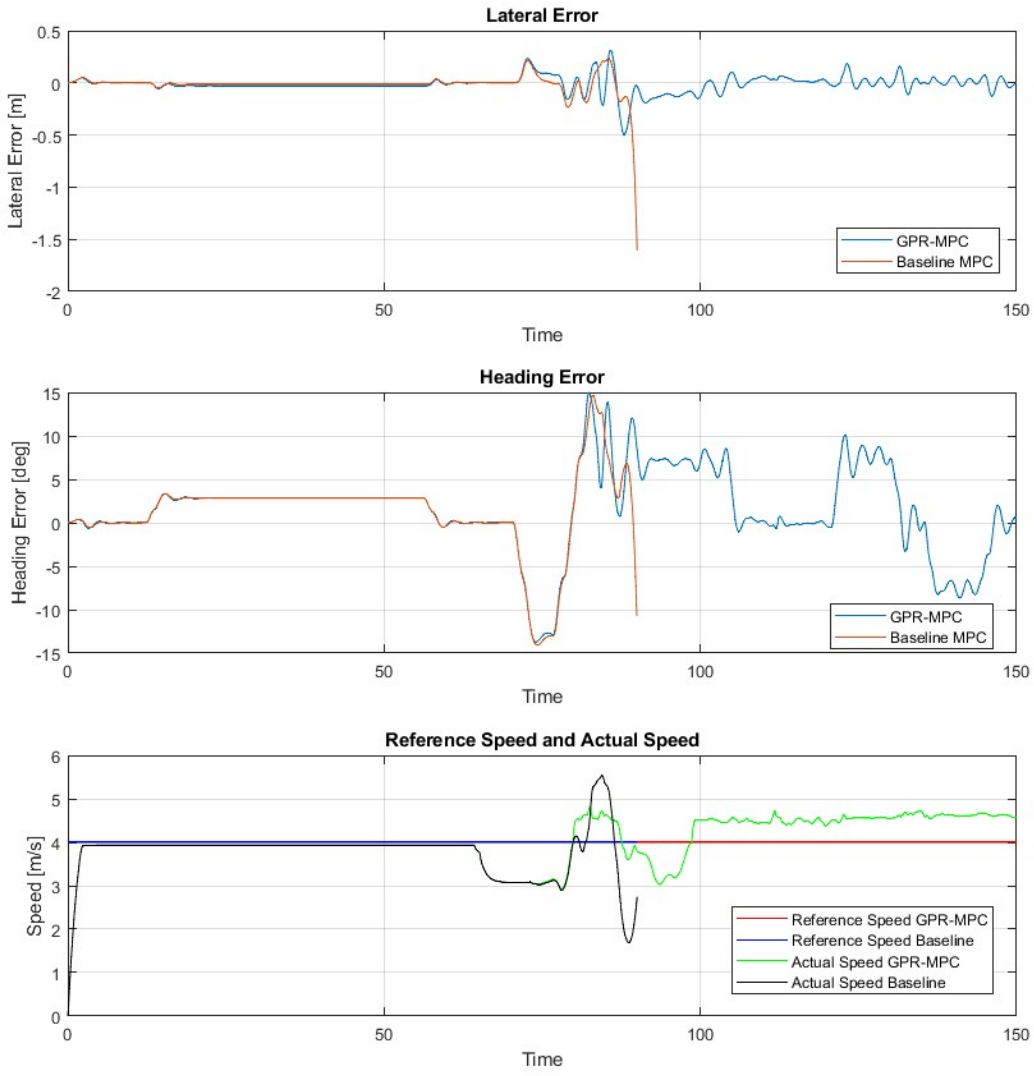


Figure 4.16: Trajectory Tracking Results - Long Off-road Track with GPR-MPC - 1000 kg Uncentered Payload, Low μ

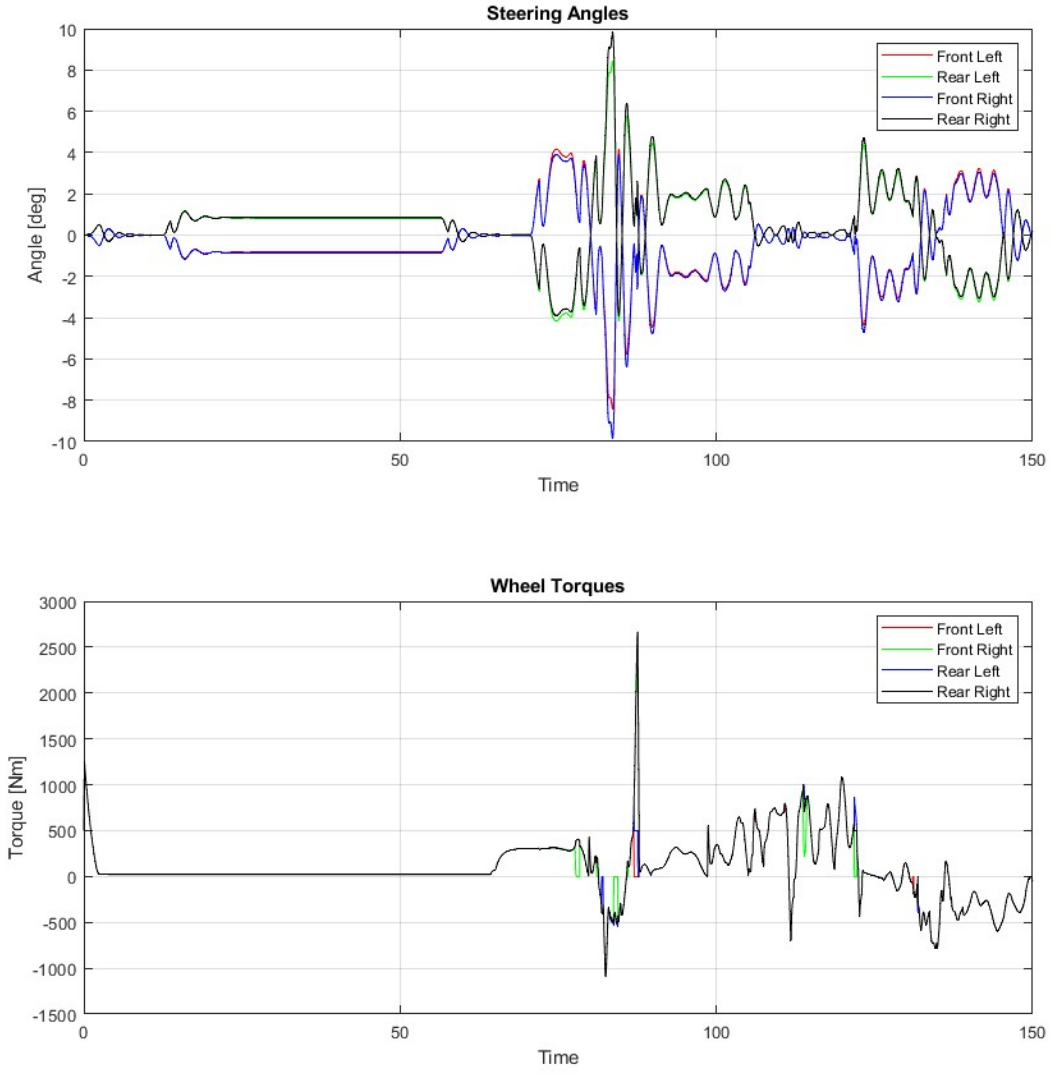


Figure 4.17: Controller Outputs - Long Off-Road Track, GPR-MPC vs Baseline - 1000 kg Uncentered Payload, Low μ

4.5 Summary of Controller Tracking Performance

Tables 4.1, 4.2, 4.3, 4.4, 4.5, 4.6, and 4.7 summarize the RMS errors for the proposed controllers in the various off-road scenarios. All percent improvement metrics are with respect to the baseline controller. RA-MPC results have been included for payload simulation scenarios for comparison. The last scenario with an uncentered payload particularly highlights the advantages of GPR-MPC. In general, both RA-MPC and GPR-MPC formulations show promise in improving the results of the baseline controller in off-road conditions.

Controller	Heading RMS Error (deg)	Lateral RMS Error (m)	Velocity RMS Error (m/s)
BASELINE	20.6113	0.2769	0.7348
RA-MPC	21.3386 ($\downarrow$ 4%)	0.2267 ($\uparrow$ 18%)	0.4140 ($\uparrow$ 44%)
GPR-MPC	19.7434 ($\uparrow$ 4%)	0.2213 ($\uparrow$ 20%)	0.4063 ($\uparrow$ 45%)

Table 4.1: Comparison of RMS Errors for Heading, Lateral, and Velocity for Short Track Off-Road Scenario - High μ.

Controller	Heading RMS Error (deg)	Lateral RMS Error (m)	Velocity RMS Error (m/s)
BASELINE	5.5948	0.1011	0.6321
RA-MPC	5.8487 ($\downarrow$ 5%)	0.1270 ($\downarrow$ 26%)	0.2204 ($\uparrow$ 65%)
GPR-MPC	5.7434 ($\downarrow$ 3%)	0.1262 ($\downarrow$ 25%)	0.3414 ($\uparrow$ 46%)

Table 4.2: Comparison of RMS Errors for Heading, Lateral, and Velocity for Long Track Off-Road Scenario - High μ

Controller	Heading RMS Error (deg)	Lateral RMS Error (m)	Velocity RMS Error (m/s)
BASELINE	20.6127	0.2667	0.7358
RA-MPC	21.3621 ($\downarrow$ 4%)	0.2556 ($\uparrow$ 4%)	0.4306 ($\uparrow$ 41%)
GPR-MPC	20.1602 ($\uparrow$ 2%)	0.2376 ($\uparrow$ 11%)	0.5409 ($\uparrow$ 26%)

Table 4.3: Comparison of RMS Errors for Heading, Lateral, and Velocity for Short Track Off-Road Scenario - Low μ

Controller	Heading RMS Error (deg)	Lateral RMS Error (m)	Velocity RMS Error (m/s)
BASELINE	5.2689	0.1128	0.6325
RA-MPC	5.5376(↓ 5%)	0.0817(↑ 28%)	0.1748(↑ 72%)
GPR-MPC	5.5637(↓ 6%)	0.0834(↑ 26%)	0.2984(↑ 53%)

Table 4.4: Comparison of RMS Errors for Heading, Lateral, and Velocity for Long Track Off-Road Scenario - Low μ

Controller	Heading RMS Error (deg)	Lateral RMS Error (m)	Velocity RMS Error (m/s)
BASELINE	20.4377	0.3057	0.9073
RA-MPC	21.2450(↓ 4%)	0.2629(↑ 14%)	0.4574(↑ 50%)
GPR-MPC	19.9002(↑ 3%)	0.2549(↑ 17%)	0.5262(↑ 42%)

Table 4.5: Comparison of RMS Errors for Heading, Lateral, and Velocity for Short Track Off-Road Scenario - 800 kg Centered Payload and Low μ

Controller	Heading RMS Error (deg)	Lateral RMS Error (m)	Velocity RMS Error (m/s)
BASELINE	5.0130	0.0701	0.8565
RA-MPC	5.0218(↓ 0.2%)	0.0476(↑ 32%)	0.3533(↑ 59%)
GPR-MPC	4.9965(↑ 0.3%)	0.0593(↑ 15%)	0.4823(↑ 44%)

Table 4.6: Comparison of RMS Errors for Heading, Lateral, and Velocity for Long Track Off-Road Scenario - 800 kg Centered Payload and Low μ

Controller	Heading RMS Error (deg)	Lateral RMS Error (m)	Velocity RMS Error (m/s)
BASELINE	DNC	DNC	DNC
RA-MPC	DNC	DNC	DNC
GPR-MPC	5.0896	0.0805	1.0549

Table 4.7: Comparison of RMS Errors for Heading, Lateral, and Velocity for Long Track Off-Road Scenario - 1000 kg Uncentered Payload and Low μ, "DNC" = "Did not complete"

Chapter 5

Conclusions and Future Work

5.1 Conclusions

This thesis aimed to enhance the accuracy of trajectory tracking controllers for off-road scenarios. Particularly, this work looked to augment an efficient MPC formulation to make it better suited for the difficulties faced in off-road trajectory tracking. This included compensating for factors such as path inclination and bank angles, vehicle payloads, as well as being robust to road roughness and difficult terrain conditions. This thesis also looked to keep the coupled nature of the trajectory tracking problem in a single centralized MPC optimization. Many studies focus on breaking this task into two separate controllers for lateral and longitudinal tracking.

As a first step, this work derived a physics-oriented prediction model that accounted for the non-negligible road angles introduced. This model was developed in a manner that was efficient to solve ensuring real-time practicality of the approach. This approach assumed the availability of road angles to the controller and results showed that a knowledge of these angles drastically improves velocity tracking and has the potential to improve lateral tracking under smooth and continuous conditions. This work also explored the difficulty in estimating the road angles in an off-road scenario due to the rough terrain and multiple planes that the vehicle may be on at a given time. The estimation difficulty, and lack of method to account for other unmodelled dynamics in off-road scenarios, motivates a data-driven approach to the off-road adaptation of the MPC controller.

The presentation of a learning-based MPC formulation was then presented to address the unmodelled dynamics problem. In this approach, a separate model is used alongside the

physics-based model to estimate the true non-linear dynamics of the vehicle. The learning-based model is then used to add linear offsets to the MPC prediction model to maintain optimization efficiency while also improving prediction performance. This framework had the advantage of being easily applicable to many learning techniques. The use of GPR was then explored as a learning technique to improve off-road vehicle performance. This learning method had the advantage of learning from prior experiences with small amounts of data and quantifying the confidence of its predictions. Simulation results showed that the learning module can compensate effectively for unmodelled longitudinal dynamics and has the potential to improve lateral performance. The learning based augmentation particularly shines in low μ scenarios where the physics model is no longer an accurate representation of the vehicle behavior. The robustness of the approach was demonstrated on rough terrains where lateral tracking was difficult to compensate for and longitudinal performance was noticeably improved under steep inclination angles. Overall, GPR-MPC was shown to be able to compensate for road angles, terrain friction changes, and payloads without the need for complicated estimators to achieve similar or better performance when compared to the physics-based controller.

5.2 Future Work

There exist a few different avenues for the next steps in off-road autonomous vehicle control. There are many learning techniques that could be used in the presented framework to augment the LTV model. A study of various learning methods used for plant model predictions would be insightful to determine the best paring with this approach. For example, recent advancements in transformer architectures show promise at predicting time series data to very high levels of accuracy. The use of this technique for time series data presents an avenue where a time varying prediction could be used throughout the entire prediction horizon of the MPC controller to better estimate the unmodelled dynamics. This learning technique could be easily integrated into the current learning MPC framework and compared with the GPR solution. Furthermore, different frameworks for integrating the learning with the physics based model could be explored to determine their effectiveness. These frameworks could focus on including learning based modules in other parts of the controller including state estimation or lower level augmentations to the physics model through the tire model. Some research exists in using a learning based method for off-road tire models [44], but in general the field is under represented in current research and more investigation of the feasibility and effectiveness of this strategy would be insightful. One potential framework for including learning based modules in this work is the use of

predictive torque constraints. Ideally, if a learning module is able to predict large slip ratios the MPC controller could proactively constrain the torque to maximize stability and overall controller performance. Another closely related avenue of future research for this work would be the expansion of the GPR module to collect and re-train the predictors in an online fashion. A robust strategy for data management would make this technique practical for real-time implementation on experimental vehicles.

With the inclusion of learning based components, there is a need to consider the safety and monitoring problem of augmenting a physics based model. In many of the learning techniques used, there is no fundamental physical understanding of the model leading to potentially dangerous predictions if the model is not well generalized. Consideration into the monitoring of the learning based module for accurate and safe augmentations needs to be explored and defined further for widespread use of the technology. One interesting option that may solve this problem implicitly are Physics Informed Neural Networks (PINNs). PINNs are able to bridge the gap between physics based models and purely data driven modules [48]. Typically these models operate more robustly on smaller amounts of data compared to purely data driven models due to their underlying understanding of the physics at play [48]. PINNs can be difficult to design due to the combined requirement of domain knowledge and machine learning expertise. Overall, they provide an interesting avenue for future research into learning techniques for MPC.

Another avenue for future research would be the inclusion of traditional adaptive based components in the current vehicle model. The use of parametric models and adaptive methods could provide a robust method of adapting the time varying model to the current conditions the vehicle is facing. In the current vehicle model formulation there are a few parameters in particular that are subject to vary during the operation of the vehicle. Particularly in the case of the mass, m, and the cornering stiffness values, C_α, these parameters are expected to change with the operating conditions of the vehicle and terrain it is traveling on. As mentioned in the literature review, [36] uses a least squares estimation of cornering stiffness to improve vehicle path tracking. The testing of this algorithm in off-road settings would be an interesting study into the algorithms robustness. A more advanced estimation module may be required to make a parametric model feasible in off-road settings. In general an inclusion of a non-linear estimation technique may provide better results in off-road without significantly increasing compute resources required.

Finally, another interesting avenue of research would be the integration of this work with the agent based framework explored in [7]. This work focuses on the same vehicle platform and lays the groundwork for a fault tolerant, distributed control system. The integration of off-road components to the AMPC formulation may provide a robust overall solution for modular vehicles in off-road applications. Specifically for the WATonoTruck,

the combination of these control strategies for trajectory tracking will allow the vehicle to be easily configurable for various commercial applications in agriculture, mining, shipping, and overall material transportation.

References

[1] McKinsey & Company. Autonomous driving's future: Convenient and connected. Accessed: April 19, 2024.

[2] Tom Seymour. Crash repair market to reduce by 17% by 2030 due to advanced driver systems, says icdp. *Automotive Management Online*, April 2018. Accessed: April 19, 2024.

[3] Kyle Edward. Goodbye gridlock: How autonomous vehicles can revolutionize city living, 2023. Accessed: 2024-08-12.

[4] Ruihe Zhang. Watonobus autonomy stack, 2023.

[5] U.S. Bureau of Labor Statistics. Census of Fatal Occupational Injuries Summary, 2023. https://www.bls.gov/news.release/pdf/cfoi.pdf, 2023. Accessed: April 19, 2024.

[6] Allison Waters. A novel universal corner module for urban electric vehicles: Design, prototype, and experiment. Master's thesis, University of Waterloo, 2017.

[7] T. Ecclestone and A. Khajepour. Development of a distributed model predictive controller for over-actuated autonomous vehicle path tracking, 2023.

[8] MathWorks. Understanding model predictive control. https://www.mathworks.com/videos/series/understanding-model-predictive-control.html. Accessed: 2024-05-07.

[9] Control.com. What is model predictive control (mpc)? https://control.com/technical-articles/what-is-model-predictive-control-mpc/. Accessed: 2024-05-07.

[10] Stefano Di Cairano and Ilya V Kolmanovsky. Automotive applications of model predictive control. Technical Report TR2018-213, Mitsubishi Electric Research Laboratories, 2018. Accessed: 2024-05-07.

[11] Arun L. Bishop, John Z. Zhang, Swaminathan Gurumurthy, Kevin Tracy, and Zachary Manchester. Relu-qp: A gpu-accelerated quadratic programming solver for model-predictive control, 2023.

[12] V. T. T. Lam. Qp-mpc solvers for real-time control of fast nonlinear systems using inexpensive microcontrollers. Master's thesis, Eindhoven University of Technology, Jul 2020.

[13] Andrea Favato, Paolo Gherardo Carlet, Francesco Toso, Riccardo Torchio, Ludovico Ortombina, Mattia Bruschetta, Ruggero Carli, Piergiorgio Alotto, Silverio Bolognani, and Jose Rodriguez. Fast solver for implicit continuous set model predictive control of electric drives. *IEEE Access*, 10:17430–17440, 2022.

[14] Riccardo Torchio, Andrea Favato, Paolo Gherardo Carlet, and et al. Fast solver for implicit continuous set model predictive control of electric drives. *TechRxiv*, Oct 2021.

[15] Jie Wang. An intuitive tutorial to gaussian process regression. *Computing in Science & Engineering*, 2024.

[16] scikit-learn contributors. Gaussian processes. https://scikit-learn.org/stable/modules/gaussian_process.html, 2024. [Online; accessed April 2024].

[17] fitrgp - matlab. https://www.mathworks.com/help/stats/fitrgp.html. Accessed: 2024-04-30.

[18] C. Shen, H. Guo, F. Liu, and H. Chen. Mpc-based path tracking controller design for autonomous ground vehicles. In *2017 36th Chinese Control Conference (CCC)*, pages 9584–9589. Technical Committee on Control Theory, CAA, 2017.

[19] S. Chen, G. Xiong, H. Chen, and D. Negrut. Mpc-based path tracking with pid speed control for high-speed autonomous vehicles considering time-optimal travel. *Journal of Central South University*, 27(12):3702–3720, 2020.

[20] S. Peicheng, L. Li, X. Ni, and A. Yang. Intelligent vehicle path tracking control based on improved mpc and hybrid pid. *IEEE Access*, 10:94133–94144, 2022.

[21] T. Yuan and R. Zhao. Lqr-mpc-based trajectory-tracking controller of autonomous vehicle subject to coupling effects and driving state uncertainties. *Sensors (Basel, Switzerland)*, 22(15):5556–, 2022.

[22] H. Pang, N. Liu, C. Hu, and Z. Xu. A practical trajectory tracking control of autonomous vehicles using linear time-varying mpc method. *Proceedings of the Institution of Mechanical Engineers. Part D, Journal of Automobile Engineering*, 236(4):709–723, 2022.

[23] Jason Kong, Martin Pfeiffer, Georg Schildbach, and Francesco Borrelli. Kinematic and dynamic vehicle models for autonomous driving control design. In *2015 IEEE Intelligent Vehicles Symposium (IV)*, pages 1094–1099. IEEE, 2015.

[24] Sean Laughery, Grant Gerhart, and Richard Goetz. Bekker's terramechanics model for off-road vehicle research. Technical Report 31, US Army TARDEC, January 1990.

[25] John Dallas et al. Online terrain estimation for autonomous vehicles on deformable terrains. *Journal of terramechanics*, 91:11–22, 2020.

[26] G. R. Gerhart, S. C. Laughery, and R. C. Goetz. Off-road vehicle locomotion using bekker's model. In *Proceedings of SPIE*, pages 127–136, Bellingham, WA, 2000. SPIE.

[27] R. Majdoubi, L. Masmoudi, and A. Elharif. Coupled nonlinear controller for vehicle trajectory tracking in a deformable soil: Application to a four-wheeled mobile agricultural robot. *Journal of terramechanics*, 110:47–68, 2023.

[28] Y. Liu, N. He, L. He, Y. Zhang, K. Xi, and M. Zhang. Self-tuning of MPC controller for mobile robot path tracking based on machine learning. *Shanghai Jiao Tong Da Xue Xue Bao*, 2022.

[29] Yassine Kebbati, Naima Ait-Oufroukh, Vincent Vigneron, and Dalil Ichalal. Neural network and anfis based auto-adaptive mpc for path tracking in autonomous vehicles. In *2021 IEEE International Conference on Networking, Sensing and Control (ICNSC)*. IEEE, 2021.

[30] L. Su, Y. Mao, F. Zhang, B. Lin, and Y. Zhang. Deviation sequence neural network control for path tracking of autonomous vehicles. *Actuators*, 13(3):101, 2024.

[31] Chao Yu. *Holistic Vehicle Control Using Learning MPC*. PhD thesis, University of Waterloo, 2023.

[32] S. Yang and C. Geng. A longitudinal/lateral coupled neural network model predictive controller for path tracking of self-driving vehicle. *IEEE Access*, 11:117121–117136, 2023.

[33] M. Rokonuzzaman, N. Mohajer, S. Nahavandi, and S. Mohamed. Model predictive control with learned vehicle dynamics for autonomous vehicle path tracking. *IEEE Access*, 9:128233–128249, 2021.

[34] M. J. L. Boada, B. L. Boada, D. Garcia-Pozuelo, and V. Diaz. Neural-empirical tyre model based on recursive lazy learning under combined longitudinal and lateral slip conditions. *International journal of automotive technology*, 12(6):821–829, 2011.

[35] S. Wu, L. Shao, D. Puscul, and C. Lex. Online tire model iterative learning considering tire relaxation behavior. *Proceedings of the Institution of Mechanical Engineers. Part D, Journal of automobile engineering*, 2023.

[36] F. Lin, Y. Chen, Y. Zhao, and S. Wang. Path tracking of autonomous vehicle based on adaptive model predictive control. *International Journal of Advanced Robotic Systems*, 16(5), 2020.

[37] S. Yang, Y. Qian, W. Hu, J. Xu, and H. Sun. Adaptive mpc-based lateral path-tracking control for automatic vehicles. *World Electr. Veh. J.*, 15(3):95, 2024.

[38] K. Zhang et al. A survey on learning-based model predictive control: Toward path tracking control of mobile platforms. *Applied Sciences*, 12(4):1995–, 2022.

[39] J. Xu, B. Zhang, X. Zhu, and Y. Su. Linear time-varying mpc vehicle trajectory tracking controller considering driving road surface factors. In *2021 International Conference on Control Science and Electric Power Systems (CSEPS)*, pages 83–88. IEEE, 2021.

[40] Q. Tan, C. Qiu, J. Huang, Y. Yin, X. Zhang, and H. Liu. Path tracking control strategy for off-road 4ws4wd vehicle based on robust model predictive control. *Robotics and Autonomous Systems*, 158:104267, 2022.

[41] M. Fnadi, W. Du, F. Plumet, and F. Benamar. Constrained model predictive control for dynamic path tracking of a bi-steerable rover on slippery grounds. *Control engineering practice*, 107:104693, 2021.

[42] E. Lucet, R. Lenain, and C. Grand. Dynamic path tracking control of a vehicle on slippery terrain. *Control engineering practice*, 42:60–73, 2015.

[43] D. Vieira, A. Vie, R. Orjuela, M. Spisser, and M. Basset. A robust lateral control architecture for off-road vehicle guidance on deformable soils. *Electronics (Basel)*, 12(11):2395, 2023.

[44] J. Dallas, Y. Weng, and T. Ersal. Combined trajectory planning and tracking for autonomous vehicles on deformable terrains. In *ASME 2020 Dynamic Systems and Control Conference, DSCC 2020*, 2020.

[45] S. Hou, W. Xu, and G. Liu. Design of an interacting multiple model-cubature kalman filter approach for vehicle sideslip angle and tire forces estimation. *Mathematical Problems in Engineering*, 2019:1–13, 2019.

[46] Ehsan Hashemi, Amir Khajepour, Nikolai Moshchuk, and Shih-Ken Chen. Real-time road bank estimation with disturbance observers for vehicle control systems. *IEEE Transactions on Control Systems Technology*, 30(1):443–450, 2022.

[47] E. Hashemi, R. Zarringhalam, A. Khajepour, W. Melek, A. Kasaiezadeh, and S.-K. Chen. Real-time estimation of the road bank and grade angles with unknown input observers. *Vehicle System Dynamics*, 55(5):648–667, 2017.

[48] Raunak Rao. Physics-informed neural networks (pinns): An intuitive guide. https://towardsdatascience.com/physics-informed-neural-networks-pinns-an-intuitive-guide-fff138069563, 2022. Accessed: 2024-05-21.

APPENDICES

Appendix A

Controller Matlab/Simulink Code

A.1 MATLAB Block Diagrams

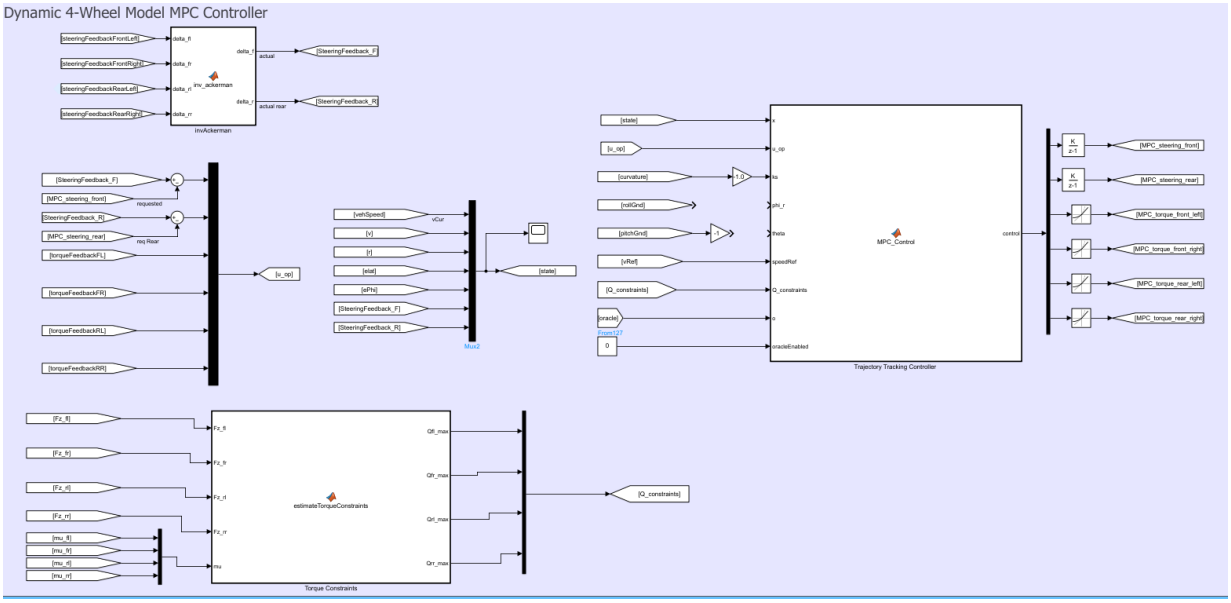


Figure A.1: Simulink diagram of MPC controller, Same blocks used for GPR-MPC and RA-MPC

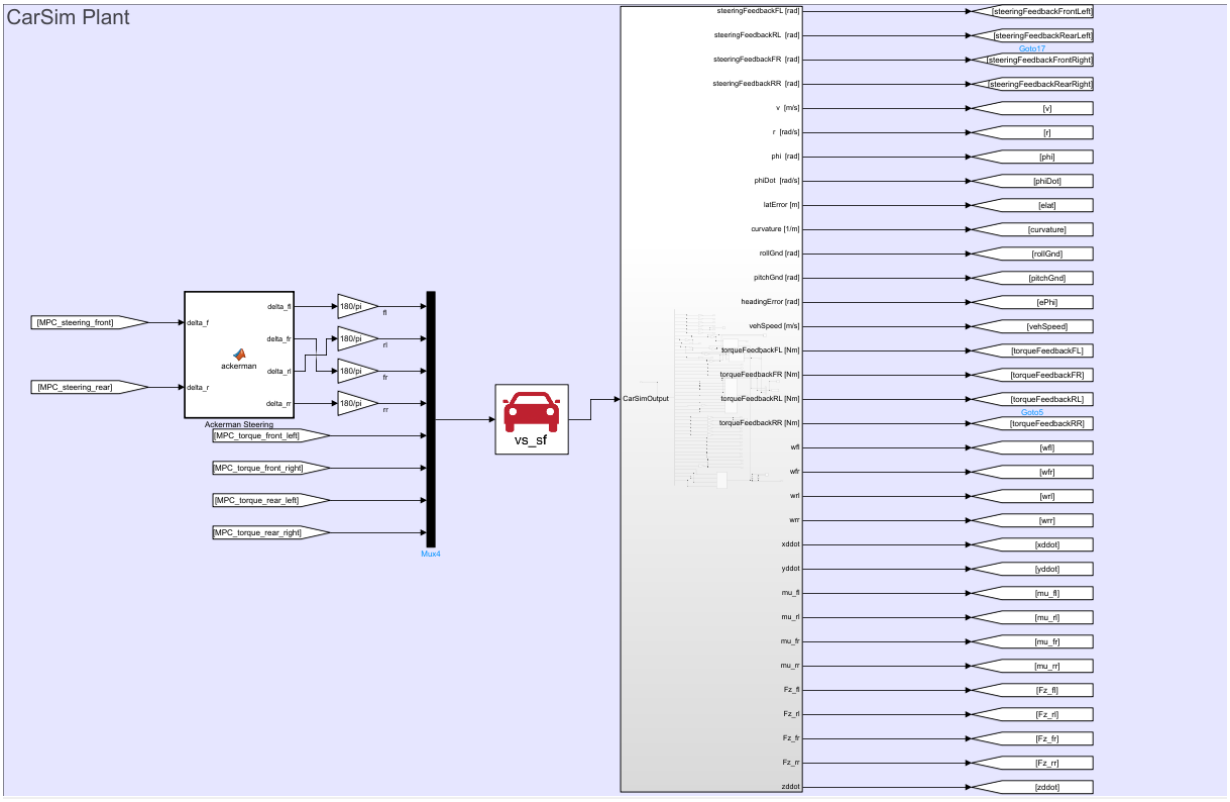


Figure A.2: Simulink diagram of CarSim Plant module

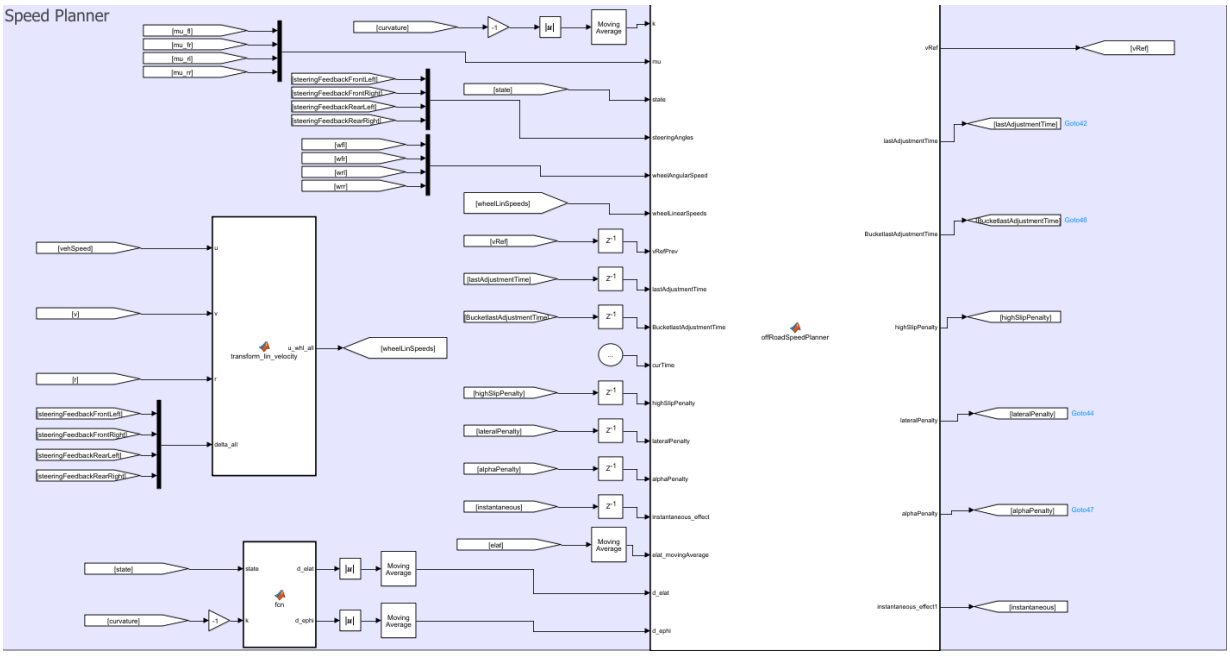


Figure A.3: Simulink diagram of Adaptive Off-Road Speed Planner

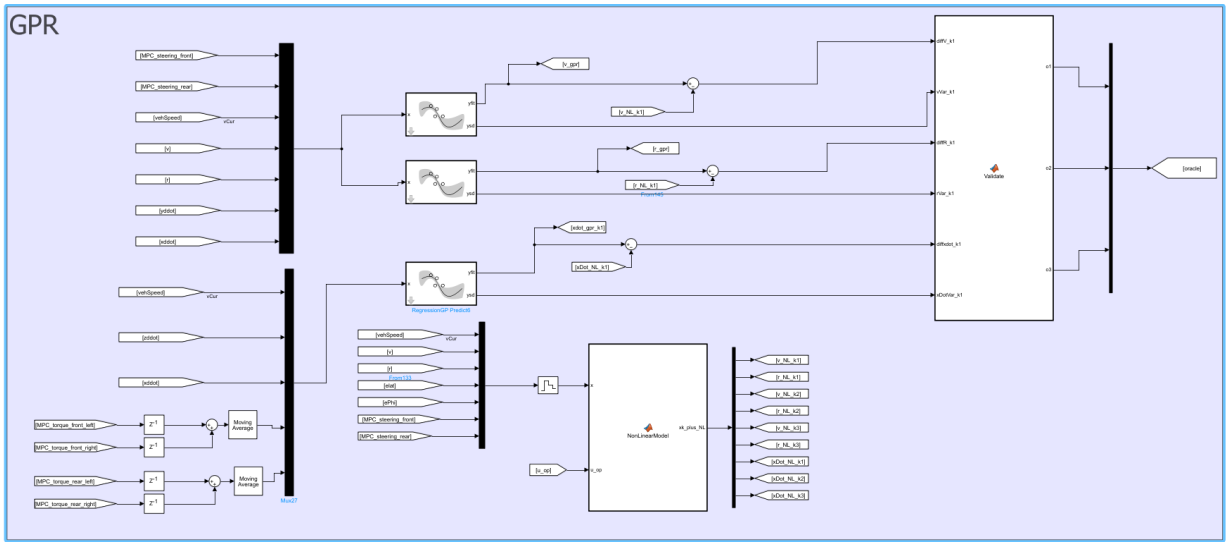


Figure A.4: Simulink diagram of GPR Prediction Module

A.2 MPC Controller Code Block

```
function [control]  = MPC_Control(x, u_op, ks, phi_r, theta, speedRef, Q_constraints, o,
    oracleEnabled, speedOffset)
    % State = [xdot, v, r, elat, ephi, delta_f_k-1, delta_r_k-1]
    % control = [d_delta_f, d_delta_r, Qfl, Qfr, Qrl, Qrr]
    % Steering in rads

    %% MPC Params
    mpcParams = struct;
    mpcParams.Nc = 30;
    mpcParams.Np = 30;
    mpcParams.Ts = 0.050;
    mpcParams.nStates = 7;
    mpcParams.nInputs = 6;
    mpcParams.nOutputs = 4;

    % If speed close to 0 at epsilon to prevent division by 0
    speedComp = -1.95*tanh(0.9*abs(x(1))-2)+1.95;
    x(1) = x(1) + speedComp;

    P =  1.*[1200 0 0 0; % Speed 1000
              0 500 0 0; % Lat Error 500
              0 0 500 0; % Heading Error 500
              0 0 0 50;]; % lat error derivative


    Q = 1.*[1200 0 0 0; % Speed 1000
              0 500 0 0; % Lat Error 500
              0 0 500 0; % Heading Error 500
              0 0 0 50;]; % lat error derivative


    R = [1000000 0 0 0 0 0; %1000000
           0 1000000 0 0 0 0;
           0 0 2.5e-3 0 0 0;
           0 0 0 2.5e-3 0 0;
           0 0 0 0 2.5e-3 0;
           0 0 0 0 0 2.5e-3;];

    mpcParams.P = P;
    mpcParams.Q = Q;
    mpcParams.R = R;

    %% Vehicle Params
    cAlphaR = 35000; %[N/rad]
    cAlphaF = 35000; %[N/rad]
    m = 1985; %[kg]
    lr = 1.885; %[m]
    lf = 1.376; %[m]
    Iz = 5000; %[kgm2]
    Rw = 0.379;
    g = 9.81;
    trackWidth = 1.59;
    wheelBase = 3.261;
```

```
% Current State measurements
delta_t = mpcParams.Ts;
delta_f = x(6);
delta_r = x(7);
Qfl = u_op(3);
Qfr = u_op(4);
Qrl = u_op(5);
Qrr = u_op(6);

xdot = x(1);
v = x(2);
r = x(3);
elat = x(4);
ephi = x(5);

%% System Matricies
% State = [xdot, v, r, elat, ephi, sf_k-1, sr_k-1]
A = [(delta_t*((cAlphaF*sin(delta_f/((trackWidth*(delta_f - delta_r))/(2*wheelBase) -
    1))*(v + lf*r))/xdot^2 - (cAlphaF*sin(delta_f/((trackWidth*(delta_f - delta_r))
    /(2*wheelBase) + 1))*(v + lf*r))/xdot^2 + (cAlphaR*sin(delta_r/((trackWidth*(
    delta_f - delta_r))/(2*wheelBase) - 1))*(v - lr*r))/xdot^2 - (cAlphaR*sin(delta_r
    /((trackWidth*(delta_f - delta_r))/(2*wheelBase) + 1))*(v - lr*r))/xdot^2))/m + 1,
     delta_t*(r - ((cAlphaF*sin(delta_f/((trackWidth*(delta_f - delta_r))/(2*wheelBase
    ) - 1)))/xdot - (cAlphaF*sin(delta_f/((trackWidth*(delta_f - delta_r))/(2*
    wheelBase) + 1)))/xdot + (cAlphaR*sin(delta_r/((trackWidth*(delta_f - delta_r))
    /(2*wheelBase) - 1)))/xdot - (cAlphaR*sin(delta_r/((trackWidth*(delta_f - delta_r)
    )/(2*wheelBase) + 1)))/xdot)/m), delta_t*(v - ((cAlphaF*lf*sin(delta_f/((
    trackWidth*(delta_f - delta_r))/(2*wheelBase) - 1)))/xdot - (cAlphaF*lf*sin(
    delta_f/((trackWidth*(delta_f - delta_r))/(2*wheelBase) + 1)))/xdot - (cAlphaR*lr*
    sin(delta_r/((trackWidth*(delta_f - delta_r))/(2*wheelBase) - 1)))/xdot + (cAlphaR
    *lr*sin(delta_r/((trackWidth*(delta_f - delta_r))/(2*wheelBase) + 1)))/xdot)/m),
    0, 0, -(delta_t*(cAlphaF*sin(delta_f/((trackWidth*(delta_f - delta_r))/(2*
    wheelBase) - 1))*(1/((trackWidth*(delta_f - delta_r))/(2*wheelBase) - 1) - (
    delta_f*trackWidth)/(2*wheelBase*((trackWidth*(delta_f - delta_r))/(2*wheelBase) -
     1)^2)) + cAlphaF*sin(delta_f/((trackWidth*(delta_f - delta_r))/(2*wheelBase) + 1)
    )*(1/((trackWidth*(delta_f - delta_r))/(2*wheelBase) + 1) - (delta_f*trackWidth)
    /(2*wheelBase*((trackWidth*(delta_f - delta_r))/(2*wheelBase) + 1)^2)) + cAlphaF*
    cos(delta_f/((trackWidth*(delta_f - delta_r))/(2*wheelBase) - 1))*(1/((trackWidth
    *(delta_f - delta_r))/(2*wheelBase) - 1) - (delta_f*trackWidth)/(2*wheelBase*((
    trackWidth*(delta_f - delta_r))/(2*wheelBase) - 1)^2))*((v + lf*r)/xdot + delta_f
    /((trackWidth*(delta_f - delta_r))/(2*wheelBase) - 1)) - cAlphaF*cos(delta_f/((
    trackWidth*(delta_f - delta_r))/(2*wheelBase) + 1))*(1/((trackWidth*(delta_f -
    delta_r))/(2*wheelBase) + 1) - (delta_f*trackWidth)/(2*wheelBase*((trackWidth*(
    delta_f - delta_r))/(2*wheelBase) + 1)^2))*((v + lf*r)/xdot - delta_f/((trackWidth
    *(delta_f - delta_r))/(2*wheelBase) + 1)) + (Qfl*sin(delta_f/((trackWidth*(delta_f
     - delta_r))/(2*wheelBase) - 1))*(1/((trackWidth*(delta_f - delta_r))/(2*wheelBase
    ) - 1) - (delta_f*trackWidth)/(2*wheelBase*((trackWidth*(delta_f - delta_r))/(2*
    wheelBase) - 1)^2)))/Rw + (Qfr*sin(delta_f/((trackWidth*(delta_f - delta_r))/(2*
    wheelBase) + 1))*(1/((trackWidth*(delta_f - delta_r))/(2*wheelBase) + 1) - (
    delta_f*trackWidth)/(2*wheelBase*((trackWidth*(delta_f - delta_r))/(2*wheelBase) +
     1)^2)))/Rw -
(cAlphaR*delta_r*trackWidth*sin(delta_r/((trackWidth*(delta_f - delta_r))/(2*wheelBase
    ) - 1)))/(2*wheelBase*((trackWidth*(delta_f - delta_r))/(2*wheelBase) - 1)^2) - (
    cAlphaR*delta_r*trackWidth*sin(delta_r/((trackWidth*(delta_f - delta_r))/(2*
    wheelBase) + 1)))/(2*wheelBase*((trackWidth*(delta_f - delta_r))/(2*wheelBase) +
    1)^2) - (Qrl*delta_r*trackWidth*sin(delta_r/((trackWidth*(delta_f - delta_r))/(2*
    wheelBase) - 1)))/(2*Rw*wheelBase*((trackWidth*(delta_f - delta_r))/(2*wheelBase)
```

```
    - 1)^2) - (Qrr*delta_r*trackWidth*sin(delta_r/((trackWidth*(delta_f - delta_r))
    /(2*wheelBase) + 1)))/(2*Rw*wheelBase*((trackWidth*(delta_f - delta_r))/(2*
    wheelBase) + 1)^2) - (cAlphaR*delta_r*trackWidth*cos(delta_r/((trackWidth*(delta_f
     - delta_r))/(2*wheelBase) - 1))*((v - lr*r)/xdot + delta_r/((trackWidth*(delta_f
    - delta_r))/(2*wheelBase) - 1)))/(2*wheelBase*((trackWidth*(delta_f - delta_r))
    /(2*wheelBase) - 1)^2) + (cAlphaR*delta_r*trackWidth*cos(delta_r/((trackWidth*(
    delta_f - delta_r))/(2*wheelBase) + 1))*((v - lr*r)/xdot - delta_r/((trackWidth*(
    delta_f - delta_r))/(2*wheelBase) + 1)))/(2*wheelBase*((trackWidth*(delta_f -
    delta_r))/(2*wheelBase) + 1)^2)))/m, -(delta_t*(cAlphaR*sin(delta_r/((trackWidth*(
    delta_f - delta_r))/(2*wheelBase) - 1))*(1/((trackWidth*(delta_f - delta_r))/(2*
    wheelBase) - 1) + (delta_r*trackWidth)/(2*wheelBase*((trackWidth*(delta_f -
    delta_r))/(2*wheelBase) - 1)^2)) + cAlphaR*sin(delta_r/((trackWidth*(delta_f -
    delta_r))/(2*wheelBase) + 1))*(1/((trackWidth*(delta_f - delta_r))/(2*wheelBase) +
     1) +
(delta_r*trackWidth)/(2*wheelBase*((trackWidth*(delta_f - delta_r))/(2*wheelBase) + 1)
    ^2)) + cAlphaR*cos(delta_r/((trackWidth*(delta_f - delta_r))/(2*wheelBase) - 1))
    *(1/((trackWidth*(delta_f - delta_r))/(2*wheelBase) - 1) + (delta_r*trackWidth)
    /(2*wheelBase*((trackWidth*(delta_f - delta_r))/(2*wheelBase) - 1)^2))*((v - lr*r)
    /xdot + delta_r/((trackWidth*(delta_f - delta_r))/(2*wheelBase) - 1)) - cAlphaR*
    cos(delta_r/((trackWidth*(delta_f - delta_r))/(2*wheelBase) + 1))*(1/((trackWidth
    *(delta_f - delta_r))/(2*wheelBase) + 1) + (delta_r*trackWidth)/(2*wheelBase*((
    trackWidth*(delta_f - delta_r))/(2*wheelBase) + 1)^2))*((v - lr*r)/xdot - delta_r
    /((trackWidth*(delta_f - delta_r))/(2*wheelBase) + 1)) + (Qrl*sin(delta_r/((
    trackWidth*(delta_f - delta_r))/(2*wheelBase) - 1))*(1/((trackWidth*(delta_f -
    delta_r))/(2*wheelBase) - 1) + (delta_r*trackWidth)/(2*wheelBase*((trackWidth*(
    delta_f - delta_r))/(2*wheelBase) - 1)^2)))/Rw + (Qrr*sin(delta_r/((trackWidth*(
    delta_f - delta_r))/(2*wheelBase) + 1))*(1/((trackWidth*(delta_f - delta_r))/(2*
    wheelBase) + 1) + (delta_r*trackWidth)/(2*wheelBase*((trackWidth*(delta_f -
    delta_r))/(2*wheelBase) + 1)^2)))/Rw + (cAlphaF*delta_f*trackWidth*sin(delta_f/((
    trackWidth*(delta_f - delta_r))/(2*wheelBase) - 1)))/(2*wheelBase*((trackWidth*(
    delta_f - delta_r))/(2*wheelBase) - 1)^2) + (cAlphaF*delta_f*trackWidth*sin(
    delta_f/((trackWidth*(delta_f - delta_r))/(2*wheelBase) + 1)))/(2*wheelBase*((
    trackWidth*(delta_f - delta_r))/(2*wheelBase) + 1)^2) + (Qfl*delta_f*trackWidth*
    sin(delta_f/((trackWidth*(delta_f - delta_r))/(2*wheelBase) - 1)))/(2*Rw*wheelBase
    *((trackWidth*(delta_f - delta_r))/(2*wheelBase) - 1)^2) + (Qfr*delta_f*trackWidth
    *sin(delta_f/((trackWidth*(delta_f - delta_r))/(2*wheelBase) + 1)))/(2*Rw*
    wheelBase*((trackWidth*(delta_f - delta_r))/(2*wheelBase) + 1)^2) + (cAlphaF*
    delta_f*trackWidth*cos(delta_f/((trackWidth*(delta_f - delta_r))/(2*wheelBase) -
    1))*((v + lf*r)/xdot + delta_f/((trackWidth*(delta_f - delta_r))/(2*wheelBase) -
    1)))/(2*wheelBase*((trackWidth*(delta_f - delta_r))/(2*wheelBase) - 1)^2) -
(cAlphaF*delta_f*trackWidth*cos(delta_f/((trackWidth*(delta_f - delta_r))/(2*wheelBase
    ) + 1))*((v + lf*r)/xdot - delta_f/((trackWidth*(delta_f - delta_r))/(2*wheelBase)
     + 1)))/(2*wheelBase*((trackWidth*(delta_f - delta_r))/(2*wheelBase) + 1)^2)))/m;
    -delta_t*(r - ((cAlphaF*cos(delta_f/((trackWidth*(delta_f - delta_r))/(2*wheelBase
    ) - 1))*(v + lf*r))/xdot^2 + (cAlphaF*cos(delta_f/((trackWidth*(delta_f - delta_r)
    )/(2*wheelBase) + 1))*(v + lf*r))/xdot^2 + (cAlphaR*cos(delta_r/((trackWidth*(
    delta_f - delta_r))/(2*wheelBase) - 1))*(v - lr*r))/xdot^2 + (cAlphaR*cos(delta_r
    /((trackWidth*(delta_f - delta_r))/(2*wheelBase) + 1))*(v - lr*r))/xdot^2)/m), 1 -
     (delta_t*((cAlphaF*cos(delta_f/((trackWidth*(delta_f - delta_r))/(2*wheelBase) -
    1)))/xdot + (cAlphaF*cos(delta_f/((trackWidth*(delta_f - delta_r))/(2*wheelBase) +
     1)))/xdot + (cAlphaR*cos(delta_r/((trackWidth*(delta_f - delta_r))/(2*wheelBase)
    - 1)))/xdot + (cAlphaR*cos(delta_r/((trackWidth*(delta_f - delta_r))/(2*wheelBase)
     + 1)))/xdot))/m, -delta_t*(xdot + ((cAlphaF*lf*cos(delta_f/((trackWidth*(delta_f
    - delta_r))/(2*wheelBase) - 1)))/xdot + (cAlphaF*lf*cos(delta_f/((trackWidth*(
    delta_f - delta_r))/(2*wheelBase) + 1)))/xdot - (cAlphaR*lr*cos(delta_r/((
    trackWidth*(delta_f - delta_r))/(2*wheelBase) - 1)))/xdot - (cAlphaR*lr*cos(
    delta_r/((trackWidth*(delta_f - delta_r))/(2*wheelBase) + 1)))/xdot)/m), 0, 0, -(
```

```
    delta_t*(cAlphaF*cos(delta_f/((trackWidth*(delta_f - delta_r))/(2*wheelBase) - 1))
    *(1/((trackWidth*(delta_f - delta_r))/(2*wheelBase) - 1) - (delta_f*trackWidth)
    /(2*wheelBase*((trackWidth*(delta_f - delta_r))/(2*wheelBase) - 1)^2)) - cAlphaF*
    cos(delta_f/((trackWidth*(delta_f - delta_r))/(2*wheelBase) + 1))*(1/((trackWidth
    *(delta_f - delta_r))/(2*wheelBase) + 1) - (delta_f*trackWidth)/(2*wheelBase*((
    trackWidth*(delta_f - delta_r))/(2*wheelBase) + 1)^2)) - cAlphaF*sin(delta_f/((
    trackWidth*(delta_f - delta_r))/(2*wheelBase) - 1))*(1/((trackWidth*(delta_f -
    delta_r))/(2*wheelBase) - 1) - (delta_f*trackWidth)/(2*wheelBase*((trackWidth*(
    delta_f - delta_r))/(2*wheelBase) - 1)^2))*((v + lf*r)/xdot + delta_f/((trackWidth
    *(delta_f - delta_r))/(2*wheelBase) - 1)) -
cAlphaF*sin(delta_f/((trackWidth*(delta_f - delta_r))/(2*wheelBase) + 1))*(1/((
    trackWidth*(delta_f - delta_r))/(2*wheelBase) + 1) - (delta_f*trackWidth)/(2*
    wheelBase*((trackWidth*(delta_f - delta_r))/(2*wheelBase) + 1)^2))*((v + lf*r)/
    xdot - delta_f/((trackWidth*(delta_f - delta_r))/(2*wheelBase) + 1)) + (Qfl*cos(
    delta_f/((trackWidth*(delta_f - delta_r))/(2*wheelBase) - 1))*(1/((trackWidth*(
    delta_f - delta_r))/(2*wheelBase) - 1) - (delta_f*trackWidth)/(2*wheelBase*((
    trackWidth*(delta_f - delta_r))/(2*wheelBase) - 1)^2)))/Rw - (Qfr*cos(delta_f/((
    trackWidth*(delta_f - delta_r))/(2*wheelBase) + 1))*(1/((trackWidth*(delta_f -
    delta_r))/(2*wheelBase) + 1) - (delta_f*trackWidth)/(2*wheelBase*((trackWidth*(
    delta_f - delta_r))/(2*wheelBase) + 1)^2)))/Rw - (cAlphaR*delta_r*trackWidth*cos(
    delta_r/((trackWidth*(delta_f - delta_r))/(2*wheelBase) - 1)))/(2*wheelBase*((
    trackWidth*(delta_f - delta_r))/(2*wheelBase) - 1)^2) + (cAlphaR*delta_r*
    trackWidth*cos(delta_r/((trackWidth*(delta_f - delta_r))/(2*wheelBase) + 1)))/(2*
    wheelBase*((trackWidth*(delta_f - delta_r))/(2*wheelBase) + 1)^2) - (Qrl*delta_r*
    trackWidth*cos(delta_r/((trackWidth*(delta_f - delta_r))/(2*wheelBase) - 1)))/(2*
    Rw*wheelBase*((trackWidth*(delta_f - delta_r))/(2*wheelBase) - 1)^2) + (Qrr*
    delta_r*trackWidth*cos(delta_r/((trackWidth*(delta_f - delta_r))/(2*wheelBase) +
    1)))/(2*Rw*wheelBase*((trackWidth*(delta_f - delta_r))/(2*wheelBase) + 1)^2) + (
    cAlphaR*delta_r*trackWidth*sin(delta_r/((trackWidth*(delta_f - delta_r))/(2*
    wheelBase) - 1))*((v - lr*r)/xdot + delta_r/((trackWidth*(delta_f - delta_r))/(2*
    wheelBase) - 1)))/(2*wheelBase*((trackWidth*(delta_f - delta_r))/(2*wheelBase) -
    1)^2) +
(cAlphaR*delta_r*trackWidth*sin(delta_r/((trackWidth*(delta_f - delta_r))/(2*wheelBase
    ) + 1))*((v - lr*r)/xdot - delta_r/((trackWidth*(delta_f - delta_r))/(2*wheelBase)
     + 1)))/(2*wheelBase*((trackWidth*(delta_f - delta_r))/(2*wheelBase) + 1)^2)))/m,
    (delta_t*(cAlphaR*cos(delta_r/((trackWidth*(delta_f - delta_r))/(2*wheelBase) + 1)
    )*(1/((trackWidth*(delta_f - delta_r))/(2*wheelBase) + 1) + (delta_r*trackWidth)
    /(2*wheelBase*((trackWidth*(delta_f - delta_r))/(2*wheelBase) + 1)^2)) - cAlphaR*
    cos(delta_r/((trackWidth*(delta_f - delta_r))/(2*wheelBase) - 1))*(1/((trackWidth
    *(delta_f - delta_r))/(2*wheelBase) - 1) + (delta_r*trackWidth)/(2*wheelBase*((
    trackWidth*(delta_f - delta_r))/(2*wheelBase) - 1)^2)) + cAlphaR*sin(delta_r/((
    trackWidth*(delta_f - delta_r))/(2*wheelBase) - 1))*(1/((trackWidth*(delta_f -
    delta_r))/(2*wheelBase) - 1) + (delta_r*trackWidth)/(2*wheelBase*((trackWidth*(
    delta_f - delta_r))/(2*wheelBase) - 1)^2))*((v - lr*r)/xdot + delta_r/((trackWidth
    *(delta_f - delta_r))/(2*wheelBase) - 1)) + cAlphaR*sin(delta_r/((trackWidth*(
    delta_f - delta_r))/(2*wheelBase) + 1))*(1/((trackWidth*(delta_f - delta_r))/(2*
    wheelBase) + 1) + (delta_r*trackWidth)/(2*wheelBase*((trackWidth*(delta_f -
    delta_r))/(2*wheelBase) + 1)^2))*((v - lr*r)/xdot - delta_r/((trackWidth*(delta_f
    - delta_r))/(2*wheelBase) + 1)) - (Qrl*cos(delta_r/((trackWidth*(delta_f - delta_r
    ))/(2*wheelBase) - 1))*(1/((trackWidth*(delta_f - delta_r))/(2*wheelBase) - 1) + (
    delta_r*trackWidth)/(2*wheelBase*((trackWidth*(delta_f - delta_r))/(2*wheelBase) -
     1)^2)))/Rw + (Qrr*cos(delta_r/((trackWidth*(delta_f - delta_r))/(2*wheelBase) +
    1))*(1/((trackWidth*(delta_f - delta_r))/(2*wheelBase) + 1) + (delta_r*trackWidth)
    /(2*wheelBase*((trackWidth*(delta_f - delta_r))/(2*wheelBase) + 1)^2)))/Rw - (
    cAlphaF*delta_f*trackWidth*cos(delta_f/((trackWidth*(delta_f - delta_r))/(2*
    wheelBase) - 1)))/(2*wheelBase*((trackWidth*(delta_f - delta_r))/(2*wheelBase) -
    1)^2) + (cAlphaF*delta_f*trackWidth*cos(delta_f/((trackWidth*(delta_f - delta_r))
```

```
    /(2*wheelBase) + 1)))/(2*wheelBase*((trackWidth*(delta_f - delta_r))/(2*wheelBase)
     + 1)^2) - (Qfl*delta_f*trackWidth*cos(delta_f/((trackWidth*(delta_f - delta_r))
    /(2*wheelBase) - 1)))/(2*Rw*wheelBase*((trackWidth*(delta_f - delta_r))/(2*
    wheelBase) - 1)^2) + (Qfr*delta_f*trackWidth*cos(delta_f/((trackWidth*(delta_f -
    delta_r))/(2*wheelBase) + 1)))/(2*Rw*wheelBase*((trackWidth*(delta_f - delta_r))
    /(2*wheelBase) + 1)^2) + (cAlphaF*delta_f*trackWidth*sin(delta_f/((trackWidth*(
    delta_f - delta_r))/(2*wheelBase) - 1))*((v + lf*r)/xdot + delta_f/((trackWidth*(
    delta_f - delta_r))/(2*wheelBase) - 1)))/(2*wheelBase*((trackWidth*(delta_f -
    delta_r))/(2*wheelBase) - 1)^2) +
(cAlphaF*delta_f*trackWidth*sin(delta_f/((trackWidth*(delta_f - delta_r))/(2*wheelBase
    ) + 1))*((v + lf*r)/xdot - delta_f/((trackWidth*(delta_f - delta_r))/(2*wheelBase)
     + 1)))/(2*wheelBase*((trackWidth*(delta_f - delta_r))/(2*wheelBase) + 1)^2)))/m;
    (delta_t*((cAlphaF*lf*cos(delta_f/((trackWidth*(delta_f - delta_r))/(2*wheelBase)
    - 1))*(v + lf*r))/xdot^2 + (cAlphaF*lf*cos(delta_f/((trackWidth*(delta_f - delta_r
    ))/(2*wheelBase) + 1))*(v + lf*r))/xdot^2 - (cAlphaR*lr*cos(delta_r/((trackWidth*(
    delta_f - delta_r))/(2*wheelBase) - 1))*(v - lr*r))/xdot^2 - (cAlphaR*lr*cos(
    delta_r/((trackWidth*(delta_f - delta_r))/(2*wheelBase) + 1))*(v - lr*r))/xdot^2))
    /Iz, -(delta_t*((cAlphaF*lf*cos(delta_f/((trackWidth*(delta_f - delta_r))/(2*
    wheelBase) - 1)))/xdot + (cAlphaF*lf*cos(delta_f/((trackWidth*(delta_f - delta_r))
    /(2*wheelBase) + 1)))/xdot - (cAlphaR*lr*cos(delta_r/((trackWidth*(delta_f -
    delta_r))/(2*wheelBase) - 1)))/xdot - (cAlphaR*lr*cos(delta_r/((trackWidth*(
    delta_f - delta_r))/(2*wheelBase) + 1)))/xdot))/Iz, 1 - (delta_t*((cAlphaF*lf^2*
    cos(delta_f/((trackWidth*(delta_f - delta_r))/(2*wheelBase) - 1)))/xdot + (cAlphaF
    *lf^2*cos(delta_f/((trackWidth*(delta_f - delta_r))/(2*wheelBase) + 1)))/xdot + (
    cAlphaR*lr^2*cos(delta_r/((trackWidth*(delta_f - delta_r))/(2*wheelBase) - 1))))/
    xdot + (cAlphaR*lr^2*cos(delta_r/((trackWidth*(delta_f - delta_r))/(2*wheelBase) +
     1)))/xdot))/Iz, 0, 0, (delta_t*(cAlphaF*lf*cos(delta_f/((trackWidth*(delta_f -
delta_r))/(2*wheelBase) + 1))*(1/((trackWidth*(delta_f - delta_r))/(2*wheelBase) + 1)
    - (delta_f*trackWidth)/(2*wheelBase*((trackWidth*(delta_f - delta_r))/(2*wheelBase
    ) + 1)^2)) - cAlphaF*lf*cos(delta_f/((trackWidth*(delta_f - delta_r))/(2*wheelBase
    ) - 1))*(1/((trackWidth*(delta_f - delta_r))/(2*wheelBase) - 1) - (delta_f*
    trackWidth)/(2*wheelBase*((trackWidth*(delta_f - delta_r))/(2*wheelBase) - 1)^2))
    + cAlphaF*lf*sin(delta_f/((trackWidth*(delta_f - delta_r))/(2*wheelBase) - 1))
    *(1/((trackWidth*(delta_f - delta_r))/(2*wheelBase) - 1) - (delta_f*trackWidth)
    /(2*wheelBase*((trackWidth*(delta_f - delta_r))/(2*wheelBase) - 1)^2))*((v + lf*r)
    /xdot + delta_f/((trackWidth*(delta_f - delta_r))/(2*wheelBase) - 1)) + cAlphaF*lf
    *sin(delta_f/((trackWidth*(delta_f - delta_r))/(2*wheelBase) + 1))*(1/((trackWidth
    *(delta_f - delta_r))/(2*wheelBase) + 1) - (delta_f*trackWidth)/(2*wheelBase*((
    trackWidth*(delta_f - delta_r))/(2*wheelBase) + 1)^2))*((v + lf*r)/xdot - delta_f
    /((trackWidth*(delta_f - delta_r))/(2*wheelBase) + 1)) + (Qfl*trackWidth*cos(
    delta_f/((trackWidth*(delta_f - delta_r))/(2*wheelBase) - 1))*(1/((trackWidth*(
    delta_f - delta_r))/(2*wheelBase) - 1) - (delta_f*trackWidth)/(2*wheelBase*((
    trackWidth*(delta_f - delta_r))/(2*wheelBase) - 1)^2)))/(2*Rw) + (Qfr*trackWidth*
    cos(delta_f/((trackWidth*(delta_f - delta_r))/(2*wheelBase) + 1))*(1/((trackWidth
    *(delta_f - delta_r))/(2*wheelBase) + 1) -
(delta_f*trackWidth)/(2*wheelBase*((trackWidth*(delta_f - delta_r))/(2*wheelBase) + 1)
    ^2)))/(2*Rw) - (Qrl*delta_r*trackWidth^2*cos(delta_r/((trackWidth*(delta_f -
    delta_r))/(2*wheelBase) - 1)))/(4*Rw*wheelBase*((trackWidth*(delta_f - delta_r))
    /(2*wheelBase) - 1)^2) - (Qrr*delta_r*trackWidth^2*cos(delta_r/((trackWidth*(
    delta_f - delta_r))/(2*wheelBase) + 1)))/(4*Rw*wheelBase*((trackWidth*(delta_f -
    delta_r))/(2*wheelBase) + 1)^2) - (cAlphaR*delta_r*lr*trackWidth*cos(delta_r/((
    trackWidth*(delta_f - delta_r))/(2*wheelBase) - 1)))/(2*wheelBase*((trackWidth*(
    delta_f - delta_r))/(2*wheelBase) - 1)^2) + (cAlphaR*delta_r*lr*trackWidth*cos(
    delta_r/((trackWidth*(delta_f - delta_r))/(2*wheelBase) + 1)))/(2*wheelBase*((
    trackWidth*(delta_f - delta_r))/(2*wheelBase) + 1)^2) + (cAlphaR*delta_r*lr*
    trackWidth*sin(delta_r/((trackWidth*(delta_f - delta_r))/(2*wheelBase) - 1))*((v -
     lr*r)/xdot + delta_r/((trackWidth*(delta_f - delta_r))/(2*wheelBase) - 1)))/(2*
```

```
    wheelBase*((trackWidth*(delta_f - delta_r))/(2*wheelBase) - 1)^2) + (cAlphaR*
    delta_r*lr*trackWidth*sin(delta_r/((trackWidth*(delta_f - delta_r))/(2*wheelBase)
    + 1))*((v - lr*r)/xdot - delta_r/((trackWidth*(delta_f - delta_r))/(2*wheelBase) +
     1)))/(2*wheelBase*((trackWidth*(delta_f - delta_r))/(2*wheelBase) + 1)^2)))/Iz, (
    delta_t*(cAlphaR*lr*cos(delta_r/((trackWidth*(delta_f - delta_r))/(2*wheelBase) -
    1))*(1/((trackWidth*(delta_f - delta_r))/(2*wheelBase) - 1) + (delta_r*trackWidth)
    /(2*wheelBase*((trackWidth*(delta_f - delta_r))/(2*wheelBase) - 1)^2)) - cAlphaR*
    lr*cos(delta_r/((trackWidth*(delta_f - delta_r))/(2*wheelBase) + 1))*(1/((
    trackWidth*(delta_f - delta_r))/(2*wheelBase) + 1) + (delta_r*trackWidth)/(2*
    wheelBase*((trackWidth*(delta_f - delta_r))/(2*wheelBase) + 1)^2)) - cAlphaR*lr*
    sin(delta_r/((trackWidth*(delta_f - delta_r))/(2*wheelBase) - 1))*(1/((trackWidth
    *(delta_f - delta_r))/(2*wheelBase) - 1) + (delta_r*trackWidth)/(2*wheelBase*((
    trackWidth*(delta_f - delta_r))/(2*wheelBase) - 1)^2))*((v - lr*r)/xdot + delta_r
    /((trackWidth*(delta_f - delta_r))/(2*wheelBase) - 1)) - cAlphaR*lr*sin(delta_r/((
    trackWidth*(delta_f - delta_r))/(2*wheelBase) + 1))*(1/((trackWidth*(delta_f -
    delta_r))/(2*wheelBase) + 1) + (delta_r*trackWidth)/(2*wheelBase*((trackWidth*(
    delta_f - delta_r))/(2*wheelBase) + 1)^2))*((v - lr*r)/xdot - delta_r/((trackWidth
    *(delta_f - delta_r))/(2*wheelBase) + 1)) + (Qrl*trackWidth*cos(delta_r/((
    trackWidth*(delta_f - delta_r))/(2*wheelBase) - 1))*(1/((trackWidth*(delta_f -
    delta_r))/(2*wheelBase) - 1) + (delta_r*trackWidth)/(2*wheelBase*((trackWidth*(
    delta_f - delta_r))/(2*wheelBase) - 1)^2)))/(2*Rw) + (Qrr*trackWidth*cos(delta_r
    /((trackWidth*(delta_f - delta_r))/(2*wheelBase) + 1))*(1/((trackWidth*(delta_f -
    delta_r))/(2*wheelBase) + 1) +
(delta_r*trackWidth)/(2*wheelBase*((trackWidth*(delta_f - delta_r))/(2*wheelBase) +
1)^2)))/(2*Rw) + (Qfl*delta_f*trackWidth^2*cos(delta_f/((trackWidth*(delta_f - delta_r
    ))/(2*wheelBase) - 1)))/(4*Rw*wheelBase*((trackWidth*(delta_f - delta_r))/(2*
    wheelBase) - 1)^2) + (Qfr*delta_f*trackWidth^2*cos(delta_f/((trackWidth*(delta_f -
     delta_r))/(2*wheelBase) + 1)))/(4*Rw*wheelBase*((trackWidth*(delta_f - delta_r))
    /(2*wheelBase) + 1)^2) - (cAlphaF*delta_f*lf*trackWidth*cos(delta_f/((trackWidth*(
    delta_f - delta_r))/(2*wheelBase) - 1)))/(2*wheelBase*((trackWidth*(delta_f -
    delta_r))/(2*wheelBase) - 1)^2) + (cAlphaF*delta_f*lf*trackWidth*cos(delta_f/((
    trackWidth*(delta_f - delta_r))/(2*wheelBase) + 1)))/(2*wheelBase*((trackWidth*(
    delta_f - delta_r))/(2*wheelBase) + 1)^2) + (cAlphaF*delta_f*lf*trackWidth*sin(
    delta_f/((trackWidth*(delta_f - delta_r))/(2*wheelBase) - 1))*((v + lf*r)/xdot +
    delta_f/((trackWidth*(delta_f - delta_r))/(2*wheelBase) - 1)))/(2*wheelBase*((
    trackWidth*(delta_f - delta_r))/(2*wheelBase) - 1)^2) + (cAlphaF*delta_f*lf*
    trackWidth*sin(delta_f/((trackWidth*(delta_f - delta_r))/(2*wheelBase) + 1))*((v +
     lf*r)/xdot - delta_f/((trackWidth*(delta_f - delta_r))/(2*wheelBase) + 1)))/(2*
    wheelBase*((trackWidth*(delta_f - delta_r))/(2*wheelBase) + 1)^2)))/Iz; delta_t*
    ephi, delta_t, 0, 1, delta_t*xdot, 0, 0; -delta_t*ks, 0, delta_t, 0, 1, 0, 0; 0,
    0, 0, 0, 0, 1, 0; 0, 0, 0, 0, 0, 0, 1];

B = [-(delta_t*(cAlphaF*sin(delta_f/((trackWidth*(delta_f - delta_r))/(2*wheelBase) -
    1))*(1/((trackWidth*(delta_f - delta_r))/(2*wheelBase) - 1) - (delta_f*trackWidth)
    /(2*wheelBase*((trackWidth*(delta_f - delta_r))/(2*wheelBase) - 1)^2)) + cAlphaF*
    sin(delta_f/((trackWidth*(delta_f - delta_r))/(2*wheelBase) + 1))*(1/((trackWidth
    *(delta_f - delta_r))/(2*wheelBase) + 1) - (delta_f*trackWidth)/(2*wheelBase*((
    trackWidth*(delta_f - delta_r))/(2*wheelBase) + 1)^2)) + cAlphaF*cos(delta_f/((
    trackWidth*(delta_f - delta_r))/(2*wheelBase) - 1))*(1/((trackWidth*(delta_f -
    delta_r))/(2*wheelBase) - 1) - (delta_f*trackWidth)/(2*wheelBase*((trackWidth*(
    delta_f - delta_r))/(2*wheelBase) - 1)^2))*((v + lf*r)/xdot + delta_f/((trackWidth
    *(delta_f - delta_r))/(2*wheelBase) - 1)) -
cAlphaF*cos(delta_f/((trackWidth*(delta_f - delta_r))/(2*wheelBase) + 1))*(1/((
    trackWidth*(delta_f - delta_r))/(2*wheelBase) + 1) - (delta_f*trackWidth)/(2*
    wheelBase*((trackWidth*(delta_f - delta_r))/(2*wheelBase) + 1)^2))*((v + lf*r)/
    xdot - delta_f/((trackWidth*(delta_f - delta_r))/(2*wheelBase) + 1)) + (Qfl*sin(
    delta_f/((trackWidth*(delta_f - delta_r))/(2*wheelBase) - 1))*(1/((trackWidth*(
```

```
    delta_f - delta_r))/(2*wheelBase) - 1) - (delta_f*trackWidth)/(2*wheelBase*((
    trackWidth*(delta_f - delta_r))/(2*wheelBase) - 1)^2)))/Rw + (Qfr*sin(delta_f/((
    trackWidth*(delta_f - delta_r))/(2*wheelBase) + 1))*(1/((trackWidth*(delta_f -
    delta_r))/(2*wheelBase) + 1) - (delta_f*trackWidth)/(2*wheelBase*((trackWidth*(
    delta_f - delta_r))/(2*wheelBase) + 1)^2)))/Rw - (cAlphaR*delta_r*trackWidth*sin(
    delta_r/((trackWidth*(delta_f - delta_r))/(2*wheelBase) - 1)))/(2*wheelBase*((
    trackWidth*(delta_f - delta_r))/(2*wheelBase) - 1)^2) - (cAlphaR*delta_r*
    trackWidth*sin(delta_r/((trackWidth*(delta_f - delta_r))/(2*wheelBase) + 1)))/(2*
    wheelBase*((trackWidth*(delta_f - delta_r))/(2*wheelBase) + 1)^2) - (Qrl*delta_r*
    trackWidth*sin(delta_r/((trackWidth*(delta_f - delta_r))/(2*wheelBase) - 1)))/(2*
    Rw*wheelBase*((trackWidth*(delta_f - delta_r))/(2*wheelBase) - 1)^2) - (Qrr*
    delta_r*trackWidth*sin(delta_r/((trackWidth*(delta_f - delta_r))/(2*wheelBase) +
1)))/(2*Rw*wheelBase*((trackWidth*(delta_f - delta_r))/(2*wheelBase) + 1)^2) - (
    cAlphaR*delta_r*trackWidth*cos(delta_r/((trackWidth*(delta_f - delta_r))/(2*
    wheelBase) - 1))*((v - lr*r)/xdot + delta_r/((trackWidth*(delta_f - delta_r))/(2*
    wheelBase) - 1)))/(2*wheelBase*((trackWidth*(delta_f - delta_r))/(2*wheelBase) -
    1)^2) + (cAlphaR*delta_r*trackWidth*cos(delta_r/((trackWidth*(delta_f - delta_r))
    /(2*wheelBase) + 1))*((v - lr*r)/xdot - delta_r/((trackWidth*(delta_f - delta_r))
    /(2*wheelBase) + 1)))/(2*wheelBase*((trackWidth*(delta_f - delta_r))/(2*wheelBase)
     + 1)^2)))/m, -(delta_t*(cAlphaR*sin(delta_r/((trackWidth*(delta_f - delta_r))/(2*
    wheelBase) - 1))*(1/((trackWidth*(delta_f - delta_r))/(2*wheelBase) - 1) + (
    delta_r*trackWidth)/(2*wheelBase*((trackWidth*(delta_f - delta_r))/(2*wheelBase) -
     1)^2)) + cAlphaR*sin(delta_r/((trackWidth*(delta_f - delta_r))/(2*wheelBase) + 1)
    )*(1/((trackWidth*(delta_f - delta_r))/(2*wheelBase) + 1) + (delta_r*trackWidth)
    /(2*wheelBase*((trackWidth*(delta_f - delta_r))/(2*wheelBase) + 1)^2)) + cAlphaR*
    cos(delta_r/((trackWidth*(delta_f - delta_r))/(2*wheelBase) - 1))*(1/((trackWidth
    *(delta_f - delta_r))/(2*wheelBase) - 1) + (delta_r*trackWidth)/(2*wheelBase*((
    trackWidth*(delta_f - delta_r))/(2*wheelBase) - 1)^2))*((v - lr*r)/xdot + delta_r
    /((trackWidth*(delta_f - delta_r))/(2*wheelBase) - 1)) - cAlphaR*cos(delta_r/((
    trackWidth*(delta_f - delta_r))/(2*wheelBase) + 1))*(1/((trackWidth*(delta_f -
    delta_r))/(2*wheelBase) + 1) + (delta_r*trackWidth)/(2*wheelBase*((trackWidth*(
    delta_f - delta_r))/(2*wheelBase) + 1)^2))*((v - lr*r)/xdot - delta_r/((trackWidth
    *(delta_f - delta_r))/(2*wheelBase) + 1)) +
(Qrl*sin(delta_r/((trackWidth*(delta_f - delta_r))/(2*wheelBase) - 1))*(1/((trackWidth
    *(delta_f - delta_r))/(2*wheelBase) - 1) + (delta_r*trackWidth)/(2*wheelBase*((
    trackWidth*(delta_f - delta_r))/(2*wheelBase) - 1)^2)))/Rw + (Qrr*sin(delta_r/((
    trackWidth*(delta_f - delta_r))/(2*wheelBase) + 1))*(1/((trackWidth*(delta_f -
    delta_r))/(2*wheelBase) + 1) + (delta_r*trackWidth)/(2*wheelBase*((trackWidth*(
    delta_f - delta_r))/(2*wheelBase) + 1)^2)))/Rw + (cAlphaF*delta_f*trackWidth*sin(
    delta_f/((trackWidth*(delta_f - delta_r))/(2*wheelBase) - 1)))/(2*wheelBase*((
    trackWidth*(delta_f - delta_r))/(2*wheelBase) - 1)^2) + (cAlphaF*delta_f*
    trackWidth*sin(delta_f/((trackWidth*(delta_f - delta_r))/(2*wheelBase) + 1)))/(2*
    wheelBase*((trackWidth*(delta_f - delta_r))/(2*wheelBase) + 1)^2) + (Qfl*delta_f*
    trackWidth*sin(delta_f/((trackWidth*(delta_f - delta_r))/(2*wheelBase) - 1)))/(2*
    Rw*wheelBase*((trackWidth*(delta_f - delta_r))/(2*wheelBase) - 1)^2) + (Qfr*
    delta_f*trackWidth*sin(delta_f/((trackWidth*(delta_f - delta_r))/(2*wheelBase) +
    1)))/(2*Rw*wheelBase*((trackWidth*(delta_f - delta_r))/(2*wheelBase) + 1)^2) + (
    cAlphaF*delta_f*trackWidth*cos(delta_f/((trackWidth*(delta_f - delta_r))/(2*
    wheelBase) - 1))*((v + lf*r)/xdot + delta_f/((trackWidth*(delta_f - delta_r))/(2*
    wheelBase) - 1)))/(2*wheelBase*((trackWidth*(delta_f - delta_r))/(2*wheelBase) -
    1)^2) - (cAlphaF*delta_f*trackWidth*cos(delta_f/((trackWidth*(delta_f - delta_r))
    /(2*wheelBase) + 1))*((v + lf*r)/xdot - delta_f/((trackWidth*(delta_f - delta_r))
    /(2*wheelBase) + 1)))/(2*wheelBase*((trackWidth*(delta_f - delta_r))/(2*wheelBase)
     + 1)^2)))/m, (delta_t*cos(delta_f/((trackWidth*(delta_f - delta_r))/(2*wheelBase)
     - 1)))/(Rw*m), (delta_t*cos(delta_f/((trackWidth*(delta_f - delta_r))/(2*
    wheelBase) + 1)))/(Rw*m), (delta_t*cos(delta_r/((trackWidth*(delta_f -
delta_r))/(2*wheelBase) - 1)))/(Rw*m), (delta_t*cos(delta_r/((trackWidth*(delta_f -
```

```
    delta_r))/(2*wheelBase) + 1)))/(Rw*m); -(delta_t*(cAlphaF*cos(delta_f/((trackWidth
    *(delta_f - delta_r))/(2*wheelBase) - 1))*(1/((trackWidth*(delta_f - delta_r))/(2*
    wheelBase) - 1) - (delta_f*trackWidth)/(2*wheelBase*((trackWidth*(delta_f -
    delta_r))/(2*wheelBase) - 1)^2)) - cAlphaF*cos(delta_f/((trackWidth*(delta_f -
    delta_r))/(2*wheelBase) + 1))*(1/((trackWidth*(delta_f - delta_r))/(2*wheelBase) +
     1) - (delta_f*trackWidth)/(2*wheelBase*((trackWidth*(delta_f - delta_r))/(2*
    wheelBase) + 1)^2)) - cAlphaF*sin(delta_f/((trackWidth*(delta_f - delta_r))/(2*
    wheelBase) - 1))*(1/((trackWidth*(delta_f - delta_r))/(2*wheelBase) - 1) - (
    delta_f*trackWidth)/(2*wheelBase*((trackWidth*(delta_f - delta_r))/(2*wheelBase) -
     1)^2))*((v + lf*r)/xdot + delta_f/((trackWidth*(delta_f - delta_r))/(2*wheelBase)
     - 1)) - cAlphaF*sin(delta_f/((trackWidth*(delta_f - delta_r))/(2*wheelBase) + 1))
    *(1/((trackWidth*(delta_f - delta_r))/(2*wheelBase) + 1) - (delta_f*trackWidth)
    /(2*wheelBase*((trackWidth*(delta_f - delta_r))/(2*wheelBase) + 1)^2))*((v + lf*r)
    /xdot - delta_f/((trackWidth*(delta_f - delta_r))/(2*wheelBase) + 1)) + (Qfl*cos(
    delta_f/((trackWidth*(delta_f - delta_r))/(2*wheelBase) - 1))*(1/((trackWidth*(
    delta_f - delta_r))/(2*wheelBase) - 1) - (delta_f*trackWidth)/(2*wheelBase*((
    trackWidth*(delta_f - delta_r))/(2*wheelBase) - 1)^2)))/Rw - (Qfr*cos(delta_f/((
    trackWidth*(delta_f - delta_r))/(2*wheelBase) + 1))*(1/((trackWidth*(delta_f -
    delta_r))/(2*wheelBase) + 1) - (delta_f*trackWidth)/(2*wheelBase*((trackWidth*(
    delta_f - delta_r))/(2*wheelBase) + 1)^2)))/Rw -
(cAlphaR*delta_r*trackWidth*cos(delta_r/((trackWidth*(delta_f - delta_r))/(2*wheelBase
    ) - 1)))/(2*wheelBase*((trackWidth*(delta_f - delta_r))/(2*wheelBase) - 1)^2) + (
    cAlphaR*delta_r*trackWidth*cos(delta_r/((trackWidth*(delta_f - delta_r))/(2*
    wheelBase) + 1)))/(2*wheelBase*((trackWidth*(delta_f - delta_r))/(2*wheelBase) +
    1)^2) - (Qrl*delta_r*trackWidth*cos(delta_r/((trackWidth*(delta_f - delta_r))/(2*
    wheelBase) - 1)))/(2*Rw*wheelBase*((trackWidth*(delta_f - delta_r))/(2*wheelBase)
    - 1)^2) + (Qrr*delta_r*trackWidth*cos(delta_r/((trackWidth*(delta_f - delta_r))
    /(2*wheelBase) + 1)))/(2*Rw*wheelBase*((trackWidth*(delta_f - delta_r))/(2*
    wheelBase) + 1)^2) + (cAlphaR*delta_r*trackWidth*sin(delta_r/((trackWidth*(delta_f
     - delta_r))/(2*wheelBase) - 1))*((v - lr*r)/xdot + delta_r/((trackWidth*(delta_f
    - delta_r))/(2*wheelBase) - 1)))/(2*wheelBase*((trackWidth*(delta_f - delta_r))
    /(2*wheelBase) - 1)^2) + (cAlphaR*delta_r*trackWidth*sin(delta_r/((trackWidth*(
    delta_f - delta_r))/(2*wheelBase) + 1))*((v - lr*r)/xdot - delta_r/((trackWidth*(
    delta_f - delta_r))/(2*wheelBase) + 1)))/(2*wheelBase*((trackWidth*(delta_f -
    delta_r))/(2*wheelBase) + 1)^2)))/m, (delta_t*(cAlphaR*cos(delta_r/((trackWidth*(
    delta_f - delta_r))/(2*wheelBase) + 1))*(1/((trackWidth*(delta_f - delta_r))/(2*
    wheelBase) + 1) + (delta_r*trackWidth)/(2*wheelBase*((trackWidth*(delta_f -
    delta_r))/(2*wheelBase) + 1)^2)) - cAlphaR*cos(delta_r/((trackWidth*(delta_f -
    delta_r))/(2*wheelBase) - 1))*(1/((trackWidth*(delta_f - delta_r))/(2*wheelBase) -
     1) +
(delta_r*trackWidth)/(2*wheelBase*((trackWidth*(delta_f - delta_r))/(2*wheelBase) - 1)
    ^2)) + cAlphaR*sin(delta_r/((trackWidth*(delta_f - delta_r))/(2*wheelBase) - 1))
    *(1/((trackWidth*(delta_f - delta_r))/(2*wheelBase) - 1) + (delta_r*trackWidth)
    /(2*wheelBase*((trackWidth*(delta_f - delta_r))/(2*wheelBase) - 1)^2))*((v - lr*r)
    /xdot + delta_r/((trackWidth*(delta_f - delta_r))/(2*wheelBase) - 1)) + cAlphaR*
    sin(delta_r/((trackWidth*(delta_f - delta_r))/(2*wheelBase) + 1))*(1/((trackWidth
    *(delta_f - delta_r))/(2*wheelBase) + 1) + (delta_r*trackWidth)/(2*wheelBase*((
    trackWidth*(delta_f - delta_r))/(2*wheelBase) + 1)^2))*((v - lr*r)/xdot - delta_r
    /((trackWidth*(delta_f - delta_r))/(2*wheelBase) + 1)) - (Qrl*cos(delta_r/((
    trackWidth*(delta_f - delta_r))/(2*wheelBase) - 1))*(1/((trackWidth*(delta_f -
    delta_r))/(2*wheelBase) - 1) + (delta_r*trackWidth)/(2*wheelBase*((trackWidth*(
    delta_f - delta_r))/(2*wheelBase) - 1)^2)))/Rw + (Qrr*cos(delta_r/((trackWidth*(
    delta_f - delta_r))/(2*wheelBase) + 1))*(1/((trackWidth*(delta_f - delta_r))/(2*
    wheelBase) + 1) + (delta_r*trackWidth)/(2*wheelBase*((trackWidth*(delta_f -
    delta_r))/(2*wheelBase) + 1)^2)))/Rw - (cAlphaF*delta_f*trackWidth*cos(delta_f/((
    trackWidth*(delta_f - delta_r))/(2*wheelBase) - 1)))/(2*wheelBase*((trackWidth*(
    delta_f - delta_r))/(2*wheelBase) - 1)^2) + (cAlphaF*delta_f*trackWidth*cos(
```

```
    delta_f/((trackWidth*(delta_f - delta_r))/(2*wheelBase) + 1)))/(2*wheelBase*((
    trackWidth*(delta_f - delta_r))/(2*wheelBase) + 1)^2) - (Qfl*delta_f*trackWidth*
    cos(delta_f/((trackWidth*(delta_f - delta_r))/(2*wheelBase) - 1)))/(2*Rw*wheelBase
    *((trackWidth*(delta_f - delta_r))/(2*wheelBase) - 1)^2) + (Qfr*delta_f*trackWidth
    *cos(delta_f/((trackWidth*(delta_f - delta_r))/(2*wheelBase) + 1)))/(2*Rw*
    wheelBase*((trackWidth*(delta_f - delta_r))/(2*wheelBase) + 1)^2) + (cAlphaF*
    delta_f*trackWidth*sin(delta_f/((trackWidth*(delta_f - delta_r))/(2*wheelBase) -
    1))*((v + lf*r)/xdot + delta_f/((trackWidth*(delta_f - delta_r))/(2*wheelBase) -
    1)))/(2*wheelBase*((trackWidth*(delta_f - delta_r))/(2*wheelBase) - 1)^2) + (
    cAlphaF*delta_f*trackWidth*sin(delta_f/((trackWidth*(delta_f - delta_r))/(2*
    wheelBase) + 1))*((v + lf*r)/xdot - delta_f/((trackWidth*(delta_f - delta_r))/(2*
    wheelBase) + 1)))/(2*wheelBase*((trackWidth*(delta_f - delta_r))/(2*wheelBase) +
    1)^2)))/m, -
(delta_t*sin(delta_f/((trackWidth*(delta_f - delta_r))/(2*wheelBase) - 1)))/(Rw*m), (
    delta_t*sin(delta_f/((trackWidth*(delta_f - delta_r))/(2*wheelBase) + 1)))/(Rw*m),
     -(delta_t*sin(delta_r/((trackWidth*(delta_f - delta_r))/(2*wheelBase) - 1)))/(Rw*
    m), (delta_t*sin(delta_r/((trackWidth*(delta_f - delta_r))/(2*wheelBase) + 1)))/(
    Rw*m); (delta_t*(cAlphaF*lf*cos(delta_f/((trackWidth*(delta_f - delta_r))/(2*
    wheelBase) + 1))*(1/((trackWidth*(delta_f - delta_r))/(2*wheelBase) + 1) - (
    delta_f*trackWidth)/(2*wheelBase*((trackWidth*(delta_f - delta_r))/(2*wheelBase) +
     1)^2)) - cAlphaF*lf*cos(delta_f/((trackWidth*(delta_f - delta_r))/(2*wheelBase) -
     1))*(1/((trackWidth*(delta_f - delta_r))/(2*wheelBase) - 1) - (delta_f*trackWidth
    )/(2*wheelBase*((trackWidth*(delta_f - delta_r))/(2*wheelBase) - 1)^2)) + cAlphaF*
    lf*sin(delta_f/((trackWidth*(delta_f - delta_r))/(2*wheelBase) - 1))*(1/((
    trackWidth*(delta_f - delta_r))/(2*wheelBase) - 1) - (delta_f*trackWidth)/(2*
    wheelBase*((trackWidth*(delta_f - delta_r))/(2*wheelBase) - 1)^2))*((v + lf*r)/
    xdot + delta_f/((trackWidth*(delta_f - delta_r))/(2*wheelBase) - 1)) + cAlphaF*lf*
    sin(delta_f/((trackWidth*(delta_f - delta_r))/(2*wheelBase) + 1))*(1/((trackWidth
    *(delta_f - delta_r))/(2*wheelBase) + 1) - (delta_f*trackWidth)/(2*wheelBase*((
    trackWidth*(delta_f - delta_r))/(2*wheelBase) + 1)^2))*((v + lf*r)/xdot - delta_f
    /((trackWidth*(delta_f - delta_r))/(2*wheelBase) + 1)) + (Qfl*trackWidth*cos(
    delta_f/((trackWidth*(delta_f - delta_r))/(2*wheelBase) - 1))*(1/((trackWidth*(
    delta_f - delta_r))/(2*wheelBase) - 1) - (delta_f*trackWidth)/(2*wheelBase*((
    trackWidth*(delta_f - delta_r))/(2*wheelBase) - 1)^2)))/(2*Rw) +
(Qfr*trackWidth*cos(delta_f/((trackWidth*(delta_f - delta_r))/(2*wheelBase) + 1))
    *(1/((trackWidth*(delta_f - delta_r))/(2*wheelBase) + 1) - (delta_f*trackWidth)
    /(2*wheelBase*((trackWidth*(delta_f - delta_r))/(2*wheelBase) + 1)^2)))/(2*Rw) - (
    Qrl*delta_r*trackWidth^2*cos(delta_r/((trackWidth*(delta_f - delta_r))/(2*
    wheelBase) - 1)))/(4*Rw*wheelBase*((trackWidth*(delta_f - delta_r))/(2*wheelBase)
    - 1)^2) - (Qrr*delta_r*trackWidth^2*cos(delta_r/((trackWidth*(delta_f - delta_r))
    /(2*wheelBase) + 1)))/(4*Rw*wheelBase*((trackWidth*(delta_f - delta_r))/(2*
    wheelBase) + 1)^2) - (cAlphaR*delta_r*lr*trackWidth*cos(delta_r/((trackWidth*(
    delta_f - delta_r))/(2*wheelBase) - 1)))/(2*wheelBase*((trackWidth*(delta_f -
    delta_r))/(2*wheelBase) - 1)^2) + (cAlphaR*delta_r*lr*trackWidth*cos(delta_r/((
    trackWidth*(delta_f - delta_r))/(2*wheelBase) + 1)))/(2*wheelBase*((trackWidth*(
    delta_f - delta_r))/(2*wheelBase) + 1)^2) + (cAlphaR*delta_r*lr*trackWidth*sin(
    delta_r/((trackWidth*(delta_f - delta_r))/(2*wheelBase) - 1))*((v - lr*r)/xdot +
    delta_r/((trackWidth*(delta_f - delta_r))/(2*wheelBase) - 1)))/(2*wheelBase*((
    trackWidth*(delta_f - delta_r))/(2*wheelBase) - 1)^2) + (cAlphaR*delta_r*lr*
    trackWidth*sin(delta_r/((trackWidth*(delta_f - delta_r))/(2*wheelBase) + 1))*((v -
     lr*r)/xdot - delta_r/((trackWidth*(delta_f - delta_r))/(2*wheelBase) + 1)))/(2*
    wheelBase*((trackWidth*(delta_f - delta_r))/(2*wheelBase) + 1)^2)))/Iz, (delta_t*(
    cAlphaR*lr*cos(delta_r/((trackWidth*(delta_f - delta_r))/(2*wheelBase) - 1))*(1/((
    trackWidth*(delta_f - delta_r))/(2*wheelBase) - 1) + (delta_r*trackWidth)/(2*
    wheelBase*((trackWidth*(delta_f - delta_r))/(2*wheelBase) - 1)^2)) - cAlphaR*lr*
    cos(delta_r/((trackWidth*(delta_f - delta_r))/(2*wheelBase) + 1))*(1/((trackWidth
    *(delta_f - delta_r))/(2*wheelBase) + 1) + (delta_r*trackWidth)/(2*wheelBase*((
```

```
    trackWidth*(delta_f - delta_r))/(2*wheelBase) + 1)^2)) - cAlphaR*lr*sin(delta_r/((
    trackWidth*(delta_f - delta_r))/(2*wheelBase) - 1))*(1/((trackWidth*(delta_f -
    delta_r))/(2*wheelBase) - 1) + (delta_r*trackWidth)/(2*wheelBase*((trackWidth*(
    delta_f - delta_r))/(2*wheelBase) - 1)^2))*((v - lr*r)/xdot + delta_r/((trackWidth
    *(delta_f - delta_r))/(2*wheelBase) - 1)) - cAlphaR*lr*sin(delta_r/((trackWidth*(
    delta_f - delta_r))/(2*wheelBase) + 1))*(1/((trackWidth*(delta_f - delta_r))/(2*
    wheelBase) + 1) + (delta_r*trackWidth)/(2*wheelBase*((trackWidth*(delta_f -
    delta_r))/(2*wheelBase) + 1)^2))*((v - lr*r)/xdot - delta_r/((trackWidth*(delta_f
    - delta_r))/(2*wheelBase) + 1)) +
(Qrl*trackWidth*cos(delta_r/((trackWidth*(delta_f - delta_r))/(2*wheelBase) - 1))
    *(1/((trackWidth*(delta_f - delta_r))/(2*wheelBase) - 1) + (delta_r*trackWidth)
    /(2*wheelBase*((trackWidth*(delta_f - delta_r))/(2*wheelBase) - 1)^2)))/(2*Rw) + (
    Qrr*trackWidth*cos(delta_r/((trackWidth*(delta_f - delta_r))/(2*wheelBase) + 1))
    *(1/((trackWidth*(delta_f - delta_r))/(2*wheelBase) + 1) + (delta_r*trackWidth)
    /(2*wheelBase*((trackWidth*(delta_f - delta_r))/(2*wheelBase) + 1)^2)))/(2*Rw) + (
    Qfl*delta_f*trackWidth^2*cos(delta_f/((trackWidth*(delta_f - delta_r))/(2*
    wheelBase) - 1)))/(4*Rw*wheelBase*((trackWidth*(delta_f - delta_r))/(2*wheelBase)
    - 1)^2) + (Qfr*delta_f*trackWidth^2*cos(delta_f/((trackWidth*(delta_f - delta_r))
    /(2*wheelBase) + 1)))/(4*Rw*wheelBase*((trackWidth*(delta_f - delta_r))/(2*
    wheelBase) + 1)^2) - (cAlphaF*delta_f*lf*trackWidth*cos(delta_f/((trackWidth*(
    delta_f - delta_r))/(2*wheelBase) - 1)))/(2*wheelBase*((trackWidth*(delta_f -
    delta_r))/(2*wheelBase) - 1)^2) + (cAlphaF*delta_f*lf*trackWidth*cos(delta_f/((
    trackWidth*(delta_f - delta_r))/(2*wheelBase) + 1)))/(2*wheelBase*((trackWidth*(
    delta_f - delta_r))/(2*wheelBase) + 1)^2) + (cAlphaF*delta_f*lf*trackWidth*sin(
    delta_f/((trackWidth*(delta_f - delta_r))/(2*wheelBase) - 1))*((v + lf*r)/xdot +
    delta_f/((trackWidth*(delta_f - delta_r))/(2*wheelBase) - 1)))/(2*wheelBase*((
    trackWidth*(delta_f - delta_r))/(2*wheelBase) - 1)^2) + (cAlphaF*delta_f*lf*
    trackWidth*sin(delta_f/((trackWidth*(delta_f - delta_r))/(2*wheelBase) + 1))*((v +
     lf*r)/xdot - delta_f/((trackWidth*(delta_f - delta_r))/(2*wheelBase) + 1)))/(2*
    wheelBase*((trackWidth*(delta_f - delta_r))/(2*wheelBase) + 1)^2)))/Iz, (delta_t*
    trackWidth*sin(delta_f/((trackWidth*(delta_f - delta_r))/(2*wheelBase) - 1)))/(2*
    Iz*Rw), (delta_t*trackWidth*sin(delta_f/((trackWidth*(delta_f - delta_r))/(2*
    wheelBase) + 1)))/(2*Iz*Rw), (delta_t*trackWidth*sin(delta_r/((trackWidth*(delta_f
     - delta_r))/(2*wheelBase) - 1)))/(2*Iz*Rw), (delta_t*trackWidth*sin(delta_r/((
    trackWidth*(delta_f - delta_r))/(2*wheelBase) + 1)))/(2*Iz*Rw); 0, 0, 0, 0, 0, 0;
    0, 0, 0, 0, 0, 0; 1, 0, 0, 0, 0, 0; 0, 1, 0, 0, 0, 0];
C = [1 0 0 0 0 0 0; % xdot
     0 0 0 1 0 0 0; % elat
     0 0 0 0 1 0 0; % ephi
     0 1 0 0 xdot 0 0; % d_elat
     ];


%% Prediction Matricies

% Create Sx, Su, Sd
Sx = zeros(mpcParams.Np*mpcParams.nOutputs, mpcParams.nStates);
Su = zeros(mpcParams.Np*mpcParams.nOutputs, mpcParams.Nc*mpcParams.nInputs);
Sd = zeros(mpcParams.Np*mpcParams.nOutputs, mpcParams.Np*mpcParams.nStates);
index = 1;
rowIndex = 1;
for j = 1:mpcParams.Np
    Sx(index:index+mpcParams.nOutputs-1, 1:mpcParams.nStates) =  C*A^j;
    Su = Su + kron(diag(ones(mpcParams.Np-j+1,1), -j+1), C*A^(j-1)*B);
    Sd = Sd + kron(diag(ones(mpcParams.Np-j+1,1), -j+1), C*A^(j-1));
    rowIndex = rowIndex+mpcParams.nStates;
    index = index + +mpcParams.nOutputs;
```

```
end

% Create Dd
% State = [v, r, phi, phiDot, elat, ephi;]
Dd = zeros(mpcParams.Np*mpcParams.nStates, 1);
Yref = zeros(mpcParams.Np*mpcParams.nOutputs, 1);
x_op = [xdot;v;r;elat;ephi;delta_f;delta_r];
f_x0 = x;

delta_fl = delta_f/(1-trackWidth/(2*wheelBase)*(delta_f-delta_r));
delta_fr = delta_f/(1+trackWidth/(2*wheelBase)*(delta_f-delta_r));
delta_rl = delta_r/(1-trackWidth/(2*wheelBase)*(delta_f-delta_r));
delta_rr = delta_r/(1+trackWidth/(2*wheelBase)*(delta_f-delta_r));

alpha_fl = delta_fl - ((v + r*lf)/xdot);
alpha_fr = delta_fr - ((v + r*lf)/xdot);
alpha_rl = delta_rl - ((v - r*lr)/xdot);
alpha_rr = delta_rr - ((v - r*lr)/xdot);

Fyfr = cAlphaF*alpha_fr;
Fyfl = cAlphaF*alpha_fl;
Fyrl = cAlphaR*alpha_rl;
Fyrr = cAlphaR*alpha_rr;

Fxfr = Qfr/Rw;
Fxfl = Qfl/Rw;
Fxrl = Qrl/Rw;
Fxrr = Qrr/Rw;

% Total tire forces
Fy = Fyfl*cos(delta_fl)+Fyfr*cos(delta_fr)+Fyrl*cos(delta_rl)+Fyrr*cos(delta_rr)+Fxfr*
    sin(delta_fr)+Fxfl*sin(delta_fl)+Fxrl*sin(delta_rl)+Fxrr*sin(delta_rr);
Fx = -Fyfl*sin(delta_fl)-Fyfr*sin(delta_fr)-Fyrl*sin(delta_rl)-Fyrr*sin(delta_rr)+Fxfr
    *cos(delta_fr)+Fxfl*cos(delta_fl)+Fxrl*cos(delta_rl)+Fxrr*cos(delta_rr);

f_x0(1) = x_op(1) + (Fx/m + v*r- g*cos(phi_r)*sin(theta))*delta_t;
f_x0(2) = x_op(2) + (-r*(xdot-speedComp) + 1/(m)*Fy - g*sin(phi_r)*cos(theta))*delta_t
    ;
f_x0(3) = x_op(3) + (1/Iz*(lf*Fyfl*cos(delta_fl)+lf*Fyfr*cos(delta_fr)-lr*Fyrl*cos(
    delta_rl)-lr*Fyrr*cos(delta_rr)+trackWidth/2*Fxfr*sin(delta_fr)+trackWidth/2*Fxrr*
    sin(delta_rr)-trackWidth/2*Fxfl*sin(delta_fl)-trackWidth/2*Fxrl*sin(delta_rl)))*
    delta_t;
f_x0(4) = x_op(4) + (v + (xdot)*ephi)*delta_t;
f_x0(5) = x_op(5) + (r - (xdot)*ks)*delta_t;
f_x0(6) = x_op(6) + (u_op(1));
f_x0(7) = x_op(7) + (u_op(2));

d_k = f_x0 - A*x_op - B*u_op;
yref = [speedRef+speedComp;0;0;0];

if xdot - speedComp > 1 && oracleEnabled == 1
    oracle = [o];
    if xdot - speedComp < 4
        oracle(7) = 0;
        oracle(8) = 0;
        oracle(9) = 0;
    end
```

```
else
    oracle = zeros(9,1);
end

rowIndex = 1;
rowIndex2 = 1;
for i = 1:mpcParams.Np
    if i < 10
        o_k = [oracle(7);oracle(1);oracle(2);0;0;0;0;];
    else
       o_k = [oracle(7);0;0;0;0;0;0;];
    end
    Dd(rowIndex:rowIndex+mpcParams.nStates-1, 1) = d_k + o_k;
    Yref(rowIndex2:rowIndex2+mpcParams.nOutputs-1, 1) = yref;
    rowIndex = rowIndex+mpcParams.nStates;
    rowIndex2 = rowIndex2+mpcParams.nOutputs;
end

Qbar = zeros(mpcParams.Np*mpcParams.nOutputs, mpcParams.Np*mpcParams.nOutputs);
Qbar = kron(eye(mpcParams.Np), mpcParams.Q);
Qbar((mpcParams.Np*mpcParams.nOutputs-mpcParams.nOutputs+1):end, (mpcParams.Np*
    mpcParams.nOutputs-mpcParams.nOutputs+1):end) = mpcParams.P;

Rbar = zeros(mpcParams.Nc*mpcParams.nInputs, mpcParams.Nc*mpcParams.nInputs);
Rbar = kron(eye(mpcParams.Nc), mpcParams.R);

%% Input Constraints
numSteeringInputs = 2;
u_min = -30*pi/180*ones(mpcParams.Nc*numSteeringInputs, 1);
u_max = 30*pi/180*ones(mpcParams.Nc*numSteeringInputs, 1);
eyeBmin = zeros(mpcParams.Nc*numSteeringInputs, numSteeringInputs);
eyeBmax = zeros(mpcParams.Nc*numSteeringInputs, numSteeringInputs);
minConstraintB = zeros(mpcParams.Nc*numSteeringInputs, 1);
maxConstraintB = zeros(mpcParams.Nc*numSteeringInputs, 1);

index = 1;
for i = 1:mpcParams.Nc
    eyeBmin(index:index+1, 1:numSteeringInputs) = eye(numSteeringInputs);
    eyeBmax(index:index+1, 1:numSteeringInputs) = eye(numSteeringInputs);
    index = index + numSteeringInputs;
end
xu = [x(6);x(7)];
minConstraintB = eyeBmin*xu-(u_min);
maxConstraintB = u_max - eyeBmax*xu;

minConstraintA = zeros(mpcParams.Nc*numSteeringInputs, mpcParams.Nc*mpcParams.nInputs)
    ;
maxConstraintA = zeros(mpcParams.Nc*numSteeringInputs, mpcParams.Nc*mpcParams.nInputs)
    ;
rowIndex = 1;
for predStep = 1:mpcParams.Nc
    colIndex = mpcParams.nInputs*predStep;
    for l = 0:predStep-1
        minConstraintA(rowIndex:(rowIndex+numSteeringInputs-1), colIndex-mpcParams.
            nInputs+1:colIndex) = -1*[1 0 0 0 0 0;
                                      0 1 0 0 0 0];
        maxConstraintA(rowIndex:(rowIndex+numSteeringInputs-1), colIndex-mpcParams.
```

```
            nInputs+1:colIndex) = [1 0 0 0 0 0;
                                   0 1 0 0 0 0];
        colIndex = colIndex - mpcParams.nInputs;
    end
    rowIndex = rowIndex + numSteeringInputs;
end


minQRateConstraintB = 10000*delta_t*ones(mpcParams.Np*4, 1);
maxQRateConstraintB = 10000*delta_t*ones(mpcParams.Np*4, 1);

qfl = [0 0 1 0 0 0];
qfr = [0 0 0 1 0 0];
qrl = [0 0 0 0 1 0];
qrr = [0 0 0 0 0 1];

mat1 = diag(-1*ones(1,mpcParams.Np));
mat2 = diag(ones(1,mpcParams.Np-1),1);
mat3 = mat2 + mat1;

AQmaxRates = [kron(mat3, qfl);
              kron(mat3, qfr);
              kron(mat3, qrl);
              kron(mat3, qrr)];

AQminsRates = [kron(mat3, -qfl);
               kron(mat3, -qfr);
               kron(mat3, -qrl);
               kron(mat3, -qrr)];

ConstraintA = [minConstraintA;maxConstraintA; AQmaxRates; AQminsRates];
ConstraintB = [minConstraintB;maxConstraintB; maxQRateConstraintB; minQRateConstraintB
    ];

% Steering Rate, Torque Constraints
lb = -20*delta_t*pi/180*ones(mpcParams.nInputs*mpcParams.Nc,1);
ub = 20*delta_t*pi/180*ones(mpcParams.nInputs*mpcParams.Nc,1);
lb(3:4:end) = -Q_constraints(1);
ub(3:4:end) = Q_constraints(1);
lb(4:4:end) =  -Q_constraints(2);
ub(4:4:end) = Q_constraints(2);
lb(5:4:end) = -Q_constraints(3);
ub(5:4:end) = Q_constraints(3);
lb(6:4:end) = -Q_constraints(4);
ub(6:4:end) = Q_constraints(4);

% Equality Constraint for front and rear steering angles
rowIndex = 1;
equalityConstraintB = zeros(mpcParams.Nc*1, 1);
equalityConstraintA = zeros(mpcParams.Nc*1, mpcParams.Nc*mpcParams.nInputs);
for predStep = 1:mpcParams.Nc
    colIndex = mpcParams.nInputs*predStep;
    for l = 0:predStep-1
        equalityConstraintA(rowIndex, colIndex-mpcParams.nInputs+1:colIndex) = [1 1 0
            0 0 0];
        colIndex = colIndex - mpcParams.nInputs;
    end
```

```
        rowIndex = rowIndex + 1;
    end

    %% Reformulate as standard QP
    H = zeros(mpcParams.nInputs*mpcParams.Nc,mpcParams.nInputs*mpcParams.Nc);
    f = zeros(mpcParams.nInputs*mpcParams.Np,1);

    H = 2*(((Su'*Qbar*Su + Rbar)+(Su'*Qbar*Su + Rbar)')/2); % To prevent numerical error
    f = 2*Su'*Qbar*(Sx*x +Sd*Dd - Yref);


    %% Optimize
    options = optimoptions('quadprog','Algorithm','active-set');
    control_horizon = quadprog(H,f,ConstraintA,ConstraintB,equalityConstraintA,
        equalityConstraintB,lb,ub,zeros(mpcParams.Nc*mpcParams.nInputs, 1),options);
    % Only want first time step for MPC
    control = control_horizon((1:mpcParams.nInputs),1);
end
```

A.3 Adaptive Speed Planner Code Block

```
function [vRef,instantaneous_effect, v_max_theoretical, v_max_performance_adjusted,
    lastAdjustmentTime, BucketlastAdjustmentTime, highSlipPenalty, lateralPenalty,
    alphaPenalty, instantaneous_effect1, runtime] = offRoadSpeedPlanner(k, mu, state,
    steeringAngles, wheelAngularSpeed, wheelLinearSpeeds, vRefPrev, lastAdjustmentTime,
    BucketlastAdjustmentTime, curTime, highSlipPenalty,  lateralPenalty, alphaPenalty,
    instantaneous_effect,elat_movingAverage, d_elat, d_ephi)

% Parameters
a_long_max = 1.00; % [m/s2]
vMin = 4;
delta_t = 0.05*2; % [s]
g = 9.81;
lr = 1.885; %[m]
lf = 1.376; %[m]
Rw = 0.379; %[m]

%% Calculate Performance measures
state(1) = state(1) + 0.001;
vCurr = state(1);

alpha_fl = abs(-steeringAngles(1) + ((state(2) + state(3)*lf)/state(1)))*180/pi;
alpha_fr = abs(-steeringAngles(2) + ((state(2) + state(3)*lf)/state(1)))*180/pi;
alpha_rl = abs(-steeringAngles(3) + ((state(2) - state(3)*lr)/state(1)))*180/pi;
alpha_rr = abs(-steeringAngles(4) + ((state(2) - state(3)*lr)/state(1)))*180/pi;

lambda_fl = abs(wheelAngularSpeed(1)*Rw - wheelLinearSpeeds(1))/
            ((wheelAngularSpeed(1)*Rw));
lambda_fr = abs(wheelAngularSpeed(2)*Rw - wheelLinearSpeeds(2))/
            ((wheelAngularSpeed(2)*Rw));
lambda_rl = abs(wheelAngularSpeed(3)*Rw - wheelLinearSpeeds(3))/
            ((wheelAngularSpeed(3)*Rw));
```

```
lambda_rr = abs(wheelAngularSpeed(4)*Rw - wheelLinearSpeeds(4))/
            ((wheelAngularSpeed(4)*Rw));

elat = state(6);
ephi = state(7);
alpha = [alpha_fl, alpha_fr, alpha_rl, alpha_rr];
lambda = [lambda_fl, lambda_fr, lambda_rl, lambda_rr];

alpha_max = max(alpha);
lambda_max = max(lambda);
lambda_max = min([0.5, lambda_max]);

mu_s = min(mu);

%% Calculate Theoretical Max Velocitys given Road info
a_lat_max = 1.75; % [m/s2]

if mu_s < 0.9
    speedLimit = 5.25; % 20 km/hr
else
    speedLimit = 8; % 40 km/hr
end

v_max_lat_a = sqrt(a_lat_max/k);

v_max_theoretical = min([v_max_lat_a,speedLimit]);

%% Adaptive Off-Road real-time gains
max_d_elat = 1.4;
min_d_elat = 0.0;
x0_d_elat = 1;
slope_d_elat = 2;
eta_d_elat = ((max_d_elat-min_d_elat)/2)*tanh(2.25*slope_d_elat*
                (abs(d_elat)-x0_d_elat))+
                ((max_d_elat-min_d_elat)/2 + min_d_elat);

max_d_ephi = 1.0;
min_d_ephi = 0.0;
x0_d_ephi = 0.75;
slope_d_ephi = 2.6;
eta_d_ephi = ((max_d_ephi-min_d_ephi)/2)*tanh(2.25*slope_d_ephi*
            (abs(d_ephi)-x0_d_ephi))+
            ((max_d_ephi-min_d_ephi)/2 + min_d_ephi);

max_elat = 0.4;
min_elat = 0.1;
x0_elat = 0.8;
slope_elat = 1;
eta_elat = ((max_elat-min_elat)/2)*tanh(2.25*slope_elat*
                (abs(elat)-x0_elat))+
                ((max_elat-min_elat)/2 + min_elat);

%% Calculate Reference

lateral_error_adjustment = lateralPenalty;
slip_ratio_adjustment = highSlipPenalty;
alpha_adjustment = alphaPenalty;
```

```
instantaneous_effect_adjustment = instantaneous_effect;

leakyBucketEnough = abs(curTime - lastAdjustmentTime) >= 30 &&
                    abs(curTime - BucketlastAdjustmentTime) >= 5;

if leakyBucketEnough && elat_movingAverage < 0.02
    BucketlastAdjustmentTime = curTime;
    slip_ratio_adjustment = slip_ratio_adjustment - 0.2*slip_ratio_adjustment;
    lateral_error_adjustment = lateral_error_adjustment
                                  - 0.2*lateral_error_adjustment;
    instantaneous_effect_adjustment = instantaneous_effect_adjustment
                                  - 0.2*instantaneous_effect_adjustment;
    alpha_adjustment = alpha_adjustment - 0.2*alpha_adjustment;
    lateral_error_adjustment = max([0, lateral_error_adjustment]);
    slip_ratio_adjustment = max([0, slip_ratio_adjustment]);
    alpha_adjustment = max([0, alpha_adjustment]);
    instantaneous_effect_adjustment = max([0, instantaneous_effect_adjustment]);
    leakyBucketEnough
end

waitedEnoughTime = abs(curTime - lastAdjustmentTime) >= 10|| lastAdjustmentTime == 0;
if abs(elat_movingAverage) > 0.1 && waitedEnoughTime
    lastAdjustmentTime = curTime
    lateral_error_adjustment =  lateral_error_adjustment + vCurr*eta_elat
end

waitedEnoughTime = abs(curTime - lastAdjustmentTime) >= 10|| lastAdjustmentTime == 0;
if lambda_max>0.1 && abs(vCurr-vRefPrev)>0.5 && waitedEnoughTime && vCurr>1
    lastAdjustmentTime = curTime
    slip_ratio_adjustment = slip_ratio_adjustment + vCurr*lambda_max/2
end

instantaneous_effect = eta_d_elat*abs(d_elat) + eta_d_ephi*abs(d_ephi);
if instantaneous_effect > 0.5 && vCurr>3
    instantaneous_effect_adjustment = instantaneous_effect;
end

v_max_performance_adjusted = v_max_theoretical - lateral_error_adjustment -
                               slip_ratio_adjustment - alpha_adjustment
                               - instantaneous_effect_adjustment;
v_max_performance_adjusted = max([v_max_performance_adjusted,vMin]);

if vRefPrev < v_max_performance_adjusted
    a_des = (v_max_performance_adjusted-vRefPrev)/(delta_t);
    a_des_sat = min([a_long_max, a_des]);
    vRef = vRefPrev + a_des_sat*delta_t;
else % vrefPrev > v_max_performance_adjusted
    a_des = (vRefPrev - v_max_performance_adjusted)/(delta_t);
    a_des_sat = min([a_long_max, a_des]);
    vRef = vRefPrev - a_des_sat*delta_t;
end

BucketlastAdjustmentTime = BucketlastAdjustmentTime;
lastAdjustmentTime = lastAdjustmentTime;
lateralPenalty = lateral_error_adjustment;
highSlipPenalty = slip_ratio_adjustment;
alphaPenalty = alpha_adjustment;
```

```
instantaneous_effect1 = instantaneous_effect_adjustment;
```

A.4 GPR Training Code

All GPR predictors were trained using the following code and Matlabs fitrgp() function.

```
% Load Data
data = load("dataset1.mat");
labels = data.out.results.Data;
features_1Step = data.out.features1StepxDot.Data;

data2 = load("dataset2.mat");
labels2 = data2.out.results.Data;
features_1Step2 = data2.out.features1StepxDot.Data;

data3 = load("dataset3.mat");
labels3 = data3.out.results.Data;
features_1Step3 = data3.out.features1StepxDot.Data;

data4 = load("dataset4.mat");
labels4 = data4.out.results.Data;
features_1Step4 = data4.out.features1StepxDot.Data;

labels = [labels;
          labels2;
          labels3;
          labels4;];

features_1Step = [features_1Step;
                  features_1Step2;
                  features_1Step3;
                  features_1Step4;];

% Train Regressors
gpMdlLatVelo_1Step = fitrgp(features_1Step,labels(:,1),'FitMethod','fic','PredictMethod','
    fic','Standardize',1);
gpMdlYawRate_1Step = fitrgp(features_1Step,labels(:,2),'FitMethod','fic','PredictMethod','
    fic','Standardize',1);
```