Tampere University

Aleksi Mäki-Penttilä

UNCERTAINTY-AWARE MPC FOR AUTONOMOUS RACING

Bachelor's thesis
Faculty of Information Technology and Communication Sciences
Examiners: Reza Taheri and Reza Ghabcheloo
December 2023

ABSTRACT

Aleksi Mäki-Penttilä: Uncertainty-aware MPC for autonomous racing
Bachelor's thesis
Tampere University
Bachelor's Programme in Computing and Electrical Engineering
December 2023

Autonomous racing has emerged as a challenging platform for evaluating state-of-the-art control algorithms used in self-driving vehicles. In this context, Model Predictive Control (MPC) is commonly utilized to meet the complex control requirements inherent to the task. However, accurate real-time modeling of nonlinear vehicle dynamics poses a significant challenge and frequently results in model discrepancies. These modeling errors can not only adversely affect control performance, but also compromise the safety of the control system.

Recognizing this limitation, recent works have demonstrated how data-driven dynamics models such as neural networks and Gaussian processes can be utilized to reduce modeling error. Concurrently, robust control techniques such as tube-MPC and stochastic MPC have been shown to mitigate the impact of modeling errors effectively, by taking the imperfect nature of the dynamics model in to consideration during the design phase.

This thesis enriches existing research by analyzing the robustness properties of an uncertainty-aware MPC. The adopted approach combines a first-principles based dynamics model with a sparse Gaussian process (GP), which is trained online to correct errors present in the nominal dynamics model. The model uncertainty is taken in to account by tightening the MPC constraints based on the variance predictions of the GP, which are propagated through the prediction horizon.

While previous research has investigated this combined approach, it has primarily emphasized lap time improvements, rather than offering a detailed analysis of its robustness properties. This thesis aims, through simulation experiments, to investigate whether this integrated approach manifests increased robustness, notably in slippery road conditions and when the tire properties change due to degradation. Unlike lap times, which hold limited relevance for commercial applications, robustness serves as a more pertinent metric for improving commercial autonomous vehicles. Consequently, the findings may offer valuable insights into the broader applicability of these methods beyond autonomous racing.

Keywords: Model Predictive Control, Autonomous driving, Machine learning, Robust control

TIIVISTELMÄ

Aleksi Mäki-Penttilä: Epävarmuustietoinen MPC autonomista kilpa-ajoa varten
Kandidaatintutkielma
Tampereen yliopisto
Faculty of Information Technology and Communication Sciences
Joulukuu 2023

Autonomisesta kilpa-ajosta on kehittynyt haastava evaluointialusta itseohjautuvien ajoneuvojen ohjausalgoritmeille. Malliprediktiivinen säätö (MPC) on usein käytetty menetelmä tässä yhteydessä. Tehtävään liittyvän reaaliaikavaatimuksen vuoksi säätimessä käytettävän prosessimallin on tasapainoteltava mallinnustarkkuutta ja säätimen suoritusaikaa. Tämä kompromissi johtaa usein mallinnusvirheisiin, jotka voivat heikentää säätimen suorituskykyä sekä pahimmassa tapauksessa vaarantaa järjestelmän turvallisuuden.

Mallinnusvirheiden aiheuttamien ongelmien lieventämistä on käsitelty useissa edeltävissä tutkimusartikkeleissa. Etenkin viimeaikaiset tutkimukset ovat kiinnittäneet huomiota neuroverkkojen sekä Gaussin prosessien kaltaisten koneoppimismenetelmien hyödyntämiseen mallinnusvirheiden pienentämiseksi. Toinen vakiintunut lähestymistapa on tube-MPC:n kaltaisten robustien säätömenetelmien käyttö, joissa mallinnusvirheen olemassaolo huomioidaan jo suunnitteluvaiheessa.

Tämä tutkielma laajentaa aikaisempaa ymmärrystä asiaan liittyen tutkimalla stokastista malliprediktiivistä säätöä autonomiseen kilpa-ajoon sovellettuna. Tutkittavassa säätöjärjestelmässä prosessimalli koostuu fysiikan lakeihin perustuvasta matemaattisesta mallista, jonka ennusteita korjataan harvan Gaussin prosessin avulla. Prosessimalliin liittyvä epävarmuus huomioidaan tiukentamalla säätimen rajoitteita Gaussin prosessin varianssiennusteiden perusteella.

Kuvailtua säätöjärjestelmää on käsitelty aikaisemmissa julkaisuissa, mutta kyseisten tutkimusten painopisteenä ovat olleet pelkästään parannukset kilpa-ajon kierrosaikoihin ideaaleissa olosuhteissa. Poiketen tästä lähestymistavasta, tämä työ tutkii säätöjärjestelmän robustisuutta, simuloimalla kilpa-ajoa haastavissa olosuhteissa, kuten sateen liukastamalla radalla. Toisin kuin parannukset kierrosaikoihin ideaaleissa olosuhteissa, järjestelmän kyky sietää haastavia olosuhteita saattaa toimia parempana mittatikkuna säätömenetelmän soveltuvuudesta autonomisen kilpa-ajon ulkopuolelle, kuten kaupallisiin itseajaviin autoihin.

Avainsanat: Malliprediktiivinen säätö, Itseajavat autot, Koneoppiminen, Robusti säätö

CONTENTS

LIST OF SYMBOLS AND ABBREVIATIONS

ACC	Assetto Corsa Competizione
α	Tire lateral slip angle
ARD	Automatic Relevance Determination
$\chi^2(p)$	Quantile function of the chi-squared distribution
CBF	Control Barrier Function
CLF	Control Lyapunov Function
CMA	Covariance Matrix Adaptation
$c(s)$	Track center line Cartesian coordinates at path position s
δ	Vehicle acceleration command
E2E	End To End
E_c	Contouring error
E_l	Lag error
ϵ	Maximum constraint violation probability
$f(x, u)$	Discrete time dynamics function
$f_c(x, u)$	Continuous time dynamics function
FSD	Formula Student Driverless
GP	Gaussian Process
h	Sampling time
HJ	Hamilton-Jacobi
IAC	Indy Autonomous Challenge
LQR	Linear Quadratic Regulator
$l(x, u)$	Cost function of the Model Predictive Control problem
MPC	Model Predictive Control
$\mu^d(z)$	Mean of the Gaussian process dynamics prediction
μ^x	Mean of the vehicle state distribution
N	Prediction horizon of the Model Predictive Control problem
NLP	Nonlinear Program

ν	Arc-length velocity of the vehicle along the reference path
ω	Vehicle angular velocity
ϕ	Vehicle steering angle
$\psi(s)$	The heading of the reference path at path position s
Q	Diagonal cost coefficients matrix of the MPC problem
QP	Quadratic Program
$r(s)$	Reference path Cartesian coordinates at path position s
s	Vehicle arc-length position on the reference path
$\Sigma^d(z)$	Covariance of the Gaussian process dynamics prediction
Σ^x	Covariance of the vehicle state distribution
SQP	Sequential Quadratic Program
θ	Vehicle heading in global coordinates
u	Controls of the vehicle
$\mathcal{U}$	Control constraint set
UCA-MPC	Uncertainty-aware MPC
$v_x^*(s)$	Reference longitudinal velocity at path position s
WSL2	Windows Subsystem for Linux
x	State of the vehicle
$\mathcal{X}$	State constraint set
z	Regression features of the Gaussian process dynamics
ζ	Cost terms vector of the MPC problem

1. INTRODUCTION

Despite the prevalence of safety features in modern vehicles, traffic accidents remain a leading non-age related cause of death globally [1]. Many such accidents are due to human error, underscoring the potential for autonomous vehicles to enhance road safety [2]. Multiple companies have already demonstrated Level 4 autonomous systems, which operate fully autonomously within specific settings, but achieving universal Level 5 automation requires further research [3, 4, 5]. The rise of the commercial autonomous vehicle industry has also stimulated research interest in autonomous racing, with competitions like Formula Student Driverless and the Indy Autonomous Challenge offering platforms for innovation [6]. Insights from these competitions could benefit commercial autonomous vehicles by improving control algorithms for challenging tasks like collision avoidance and driving on low-friction road surfaces.

Model Predictive Control (MPC) is a widely adopted control approach both in commercial autonomous vehicles and autonomous racing [7]. The effectiveness of MPC is critically influenced by the fidelity of the employed dynamics model. In real-time applications such as autonomous driving, a compromise must often be made between computational tractability and model fidelity, resulting in inherent modeling errors. These modelling errors can not only degrade the performance in tasks like lane keeping, but also cause the vehicle to spin out of control, thus having severe safety implications. Recent advancements have explored not only machine learning techniques to improve modelling accuracy, but also robust MPC formulations that mitigate the effects of modelling error [8, 9, 10].

This thesis conducts a comprehensive review of relevant literature, and drawing upon methodologies outlined in [9, 11], examines the application of uncertainty-aware MPC in autonomous racing. Diverging from previous studies [9, 12], this thesis primarily investigates the robustness properties of the chosen method, achieved by simulating racing scenarios under varying tire properties and track conditions. Performance is benchmarked against a nominal system to quantify improvements. The simulations utilize 'Assetto Corsa Competizione', a video game that, despite not being tailored for research, provides a sophisticated vehicle dynamics model well-suited for the study. The thesis is structured as follows: Chapter 2 reviews prior work; Chapter 3 lays out the necessary background information; Chapter 4 details the implementation; Chapter 5 presents the findings; and Chapter 6 discusses the conclusions.

2. RELATED WORK

A broad overview of autonomous racing as a field of research can be obtained from a recent survey published in [7]. The survey findings reveal that among various control approaches, model predictive control (MPC) is the predominant choice in a variety of autonomous racing competitions. It can also be observed, that regardless of the exact control algorithm being used, most participants in these competitions employ a modular control approach, which involves separate planning and control algorithms. The planning algorithm, such as the ones detailed in [13, 14], is tasked with finding a time-optimal trajectory through the race track. This is often accomplished with nonlinear optimization, which needs to be done in a non-real-time manner due to computational constraints. Consequently, this initial trajectory is typically post-processed with a real-time sampling based planner, examples of which can be found in [15, 16]. The purpose of this post-processing step is to compensate for any deviation from the optimal velocity profile, while also taking in to account the presence of other vehicles, which are typically ignored in the inital planning step. The resulting trajectory is then used as a reference for a real-time trajectory-following control algorithm, often based on Linear Quadratic Regulator (LQR) or MPC [17, 18]. The effectiveness of this modular methodology has been demonstrated in the Indy Autonomous Challenge (IAC), where at least two separate teams adopted such an approach to control a real-world race car [19, 20].

Alternatively, some studies have combined both planning and control tasks into a single real-time MPC problem. This has been demonstrated using two methodologies. The first approach, based on contouring MPC [21], employs innovative cost functions to represent the racing task in the original Cartesian coordinate space [22]. The second approach recasts vehicle dynamics in to a curvilinear coordinate system, allowing the application of a standard quadratic cost function [23]. The former approach has been demonstrated on a real race car in the Formula Student Driverless (FSD) competition [24].

Due to being intertwined within the control algorithm itself, the accuracy of the vehicle dynamics model bounds the performance achievable with any MPC scheme. However, the real-time nature of the autonomous racing task places constraints on the complexity of the dynamics model. This limitation becomes evident when comparing the models used in prior work, such as [24], with the more comprehensive models discussed in literature for general purpose vehicle dynamics modeling [25]. Recognizing these limitations,

recent works have investigated various machine learning techniques for dynamics modeling within MPC. For instance, Multilayer Perceptron neural networks have demonstrated superior performance over first-principle-based models for autonomous vehicle control in [8]. Although large neural networks have the capacity to represent complex phenomena, the associated computational cost can be excessive for real-time MPC with small sampling times. This issue has been addressed in [26] by using linearized neural networks, demonstrating improvements against various other modeling approaches in the task of quadcopter control. Beyond neural networks, Gaussian Processes (GPs) have been used not only to correct modeling inaccuracies in nominal vehicle dynamics models [9, 27], but have also been employed as learnable safety constraint functions [28]. Notably, to address the limitations arising from how poorly GPs scale with the size of the training dataset, sparse GP approximations have been used in [9].

Building on the discussion of vehicle dynamics representation, it is beneficial to explore how uncertainties in these models are accounted for in control strategies. Robust and stochastic MPC techniques have been developed, at least in part, to address this exact issue [29, p. 193-203]. In the realm of autonomous racing, robust MPC has been employed in the form of tube-MPC to control a real-world race car both in Roborace and the IAC, demonstrating improved safety by reducing acceleration constraint violations [10, 18]. Separately, another line of research builds on foundational studies [11] to create an uncertainty-aware MPC. This innovative approach explicitly considers model uncertainty by utilizing the variance predictions of a GP dynamics model in conjunction with with approximate uncertainty propagation [9]. When evaluated on a real race car, this method achieved a noteworthy 10 $\%$ reduction in lap times. However, the system's capability to tolerate variable track conditions remains a promising yet unvalidated hypothesis.

Given that the task of autonomous racing is often formulated as a nonlinear MPC problem, it should be noted that the stability and safety analysis of nonlinear MPC remains an open research problem, and practical applicability to real-world systems often presents challenges [29, pp. 163-166]. One could consider the absence of any stability analysis in prior work [18, 22, 24] as anecdotal evidence of this. Recently the explicit integration of Control Lyapunov Functions (CLF) and Control Barrier Functions (CBF) into the nonlinear MPC problem has received research attention, highlighting a promising avenue towards a generally applicable method to address the aforementioned issues [30, 31].

Finally, to provide a full view of the autonomous racing landscape, it is worth mentioning that end-to-end (E2E) approaches such as imitation- and reinforcement learning have also garnered attention in autonomous racing research [7]. While such E2E approaches have the potential to take full advantage of high-dimensional sensor data, produced by e.g. camera and LIDAR, their black box nature can make safety assessment challenging. For this reason, some works have incorporated forms of uncertainty awareness in an attempt to enhance safety [32, 33].

3. THEORY

The objective of this chapter is to provide a brief yet comprehensive introduction to the theoretical frameworks that underpin the implementation of uncertainty-aware MPC. The chapter begins with an introduction to Gaussian Process regression, a robust machine learning technique for system identification. A sparse approximation method more suitable for real-time control applications is also presented. The chapter then explores the fundamentals of MPC, detailing its problem formulation and distinguishing between its linear and nonlinear forms. Important stability and safety properties are discussed, along with different categories of MPC, such as robust and stochastic variants.

3.1 Gaussian process regression

Gaussian processes are non-parametric models widely recognized for their robust performance in regression tasks, even with limited amounts of training data. Similar to linear regression, Gaussian process regression treats the data $\mathbf{D} = \{(x_i, y_i), i = 0, ..., N\}$ as if it was generated by:

$$y = f(x) + \mathbf{w}, \tag{3.1}$$

where the additive noise term $\mathbf{w} \sim \mathcal{N}(0, \sigma_n^2)$ corrupts the observations y of the underlying function $f(x)$. The goal of the regression task is to find $f(x)$, which in the case of Gaussian process regression is a distribution over functions, such that any finite number of function samples are jointly Gaussian:

$$f(x) \sim GP(m(x), k(x, x')). \tag{3.2}$$

The distribution of functions concept is illustrated in Figure 3.1, which also depicts how the mean and standard deviation of the function distribution can be considered as the output of a Gaussian process. The Gaussian process is parameterized by the mean function $m(x)$ and covariance kernel $k(x, x')$ defined as:

$$m(x) = \mathbb{E}[f(x)], \tag{3.3a}$$

$$k(x, x') = \mathbb{E}[(f(x) - m(x))(f(x') - m(x'))], \tag{3.3b}$$

which together describe the the prior distribution of the Gaussian process, representing the initial knowledge about $f(x)$ before any training data is considered. [34, p. 13] While the prior distribution itself is not generally useful for making predictions, it serves as a foundation upon which the posterior distribution is built. Utilizing Bayesian formalism, the posterior distribution is derived by conditioning the prior distribution on the available training data. Effectively, the posterior distribution serves as a trained regression model, enabling predictions f^* for previously unseen data points x^* based on the knowledge acquired from the training data. Figure 3.1 illustrates the process of computing the posterior distribution from the prior, leveraging the training data.

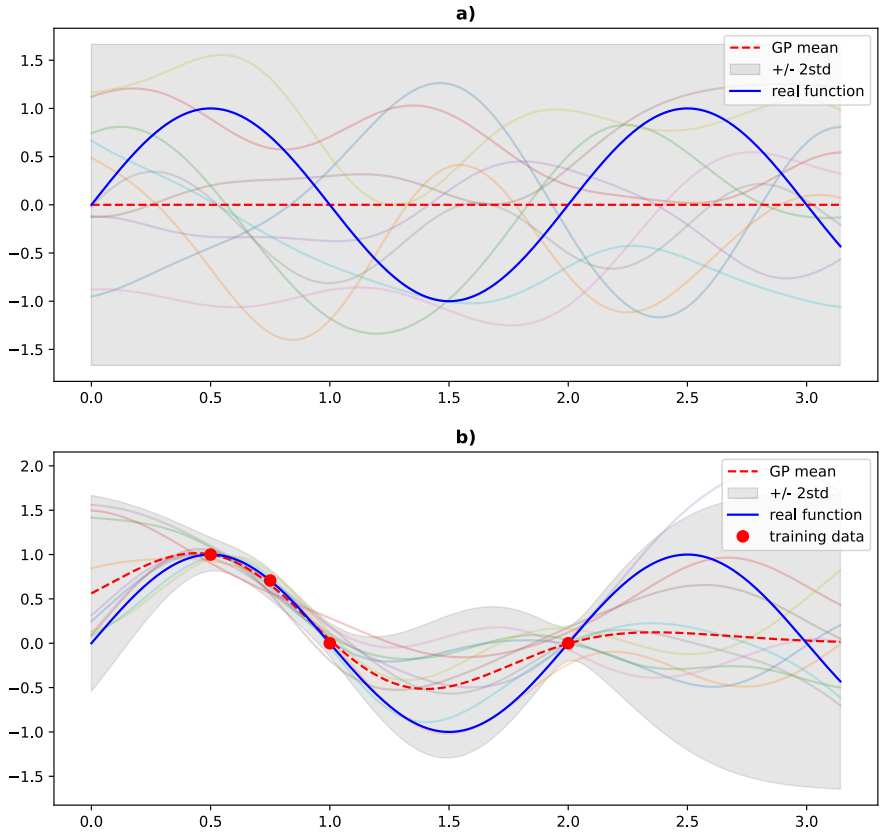

Figure 3.1. *The prior distribution, depicted in part (a) of the figure, is conditioned on four training data points, yielding the posterior distribution shown in part (b). In both plots, ten functions are sampled from their respective distributions and displayed with reduced opacity in the background.*

In practice the mean function is often set as $m(x) = 0$ to simplify both notation and posterior computation [34, p. 27]. However, it's worth noting that choosing a non-zero mean function can incorporate prior knowledge into the model, potentially enhancing predictive accuracy in data-sparse regions. Despite this, the choice of the mean function generally

has a minor impact on the performance of regression compared to the covariance kernel, which plays a more pivotal role. The outputs of a Gaussian process are derived by interpolating training data based on the similarity between input variables x and x'. This similarity is quantified using the covariance kernel $k(x, x')$, underscoring the kernel's importance in the model. As an example, the commonly used squared exponential kernel is defined as follows:

$$k(x, x') = \sigma_f^2 \exp(-\frac{||x - x'||^2}{2\lambda^2}), \tag{3.4}$$

where the variance term σ_f and length scale λ are hyperparameters [34, p. 83-84]. Specifically, the variance term σ_f dictates the magnitude of the variance away from any given training data points, while the length scale λ governs how quickly this maximum variance is attained as one deviates from a training point. Although the hyperparameters of the squared exponential kernel may be intuitively adjusted in a one-dimensional setting, this becomes increasingly impractical in higher-dimensional spaces or when employing more complex covariance kernels. Consequently, optimal parameters are commonly determined through optimization techniques, such as gradient descent.

3.1.1 Gaussian process inference

Having established the foundational definitions, the task of Gaussian process inference can now be detailed. Initially, the matrix $K(X, X')$ is defined to be the gram matrix, which is constructed by evaluating the covariance kernel between all pairs of elements within the input sets X and X':

$$K(X, X') = \begin{bmatrix} k(x_1, x'_1) & \dots & k(x_1, x'_n) \\ \vdots & \ddots & \vdots \\ k(x_n, x'_1) & \dots & k(x_n, x'_n). \end{bmatrix} \tag{3.5}$$

Assuming a zero mean function for the prior, one can then define the joint prior distribution of the training labels Y and the test outputs f^*:

$$\begin{bmatrix} Y \\ f^* \end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} K(X, X) + \sigma_n^2 I & K(X, X^*) \\ K(X^*, X) & K(X^*, X^*) \end{bmatrix}\right), \tag{3.6}$$

where X and $X*$ are the sets of training and test inputs respectively [34, p. 16]. Conditioning this joint prior distribution leads to the posterior distribution, characterized as a multivariate Gaussian with mean $\mu(X^*)$ and covariance $\Sigma(X^*)$. According to [34, p. 16],

the posterior distribution can be formally expressed as:

$$p(f^*|X, Y, X^*) \sim \mathcal{N}(\mu(X^*), \Sigma(X^*)), \tag{3.7a}$$
$$\mu(X^*) = K(X^*, X)[K(X, X) + \sigma_n^2 I]^{-1} Y, \tag{3.7b}$$
$$\Sigma(X^*) = K(X^*, X^*) - K(X^*, X)[K(X, X) + \sigma_n^2 I]^{-1} K(X, X^*). \tag{3.7c}$$

The posterior distribution in Equation (3.7) reveals an important consideration regarding the computational complexity associated with Gaussian process inference. Specifically, the required precomputations for inference involve the inversion of the Gram matrix $K(X, X)$, an operation with a computational complexity of $\mathcal{O}(n^3)$, where n is the number of training data points. Subsequent to this matrix inversion, the computation of predictive means can be executed in $\mathcal{O}(n)$ time complexity, whereas the computation of the predictive covariance requires $\mathcal{O}(n^2)$ time complexity. [35] Due to these computational demands, naive implementations of Gaussian process inference are generally confined to relatively modest-sized datasets.

3.1.2 Sparse Gaussian process inference

The computational complexity outlined in Section 3.1.1 underscores the necessity for a sparse approximation technique that offers improved computational efficiency. While numerous sparse approximations have been proposed, this thesis employs the inducing inputs method introduced by [35]. The fundamental concept is to identify a concise subset of inducing inputs, denoted as $\bar{X}$, to act as a representative for the entire training set X. Consequently, the computational complexity becomes governed by m instead of n, where m is the number of inducing points in $\bar{X}$ and n is the number of training inputs in X. By denoting $\bar{Y}$ as the pseudo target associated with $\bar{X}$, the posterior distribution for the sparse Gaussian process can be written as [35]:

$$p(f^*|\bar{X}, \bar{Y}, X^*) \sim \mathcal{N}(\mu(X^*), \Sigma(X^*)), \tag{3.8a}$$
$$\mu(X^*) = K(X^*, \bar{X}) Q_m^{-1} K(\bar{X}, X)[\Lambda + \sigma^2 I]^{-1} \bar{Y}, \tag{3.8b}$$
$$\Sigma(X^*) = K(X^*, X^*) - K(X^*, \bar{X})[K(\bar{X}, \bar{X})^{-1} - Q_m^{-1}] K(\bar{X}, X^*), \tag{3.8c}$$

where Λ and Q_m are defined as follows:

$$\Lambda = \mathrm{diag}(\lambda), \tag{3.9a}$$
$$\lambda_n = K(x_n, x_n) - K(x_n, \bar{X}) K(\bar{X}, \bar{X})^{-1} K(\bar{X}, x_n), \tag{3.9b}$$
$$Q_m = K(\bar{X}, \bar{X}) + K(\bar{X}, X)[\Lambda + \sigma^2 I]^{-1} K(X, \bar{X}). \tag{3.9c}$$

The time complexity for the required precomputations using the sparse approximation is $\mathcal{O}(m^2 n)$. Subsequently, the time complexity for computing the mean and covariance

predictions are $\mathcal{O}(m)$ and $\mathcal{O}(m^2)$, respectively [35]. Importantly, although the computational complexity is nearly identical to a scenario where $X = \bar{X}$, the full training data X still provides valuable information. This contribution is manifested through the covariance matrix $K(\bar{X}, X)$ allowing for a more robust model that leverages the full dataset. Despite this property, it is important to acknowledge that the sparse approximation is only representative of the full GP in the near vicinity of the inducing points $\bar{X}$.

3.2 Model Predictive Control

Originating in the 1980s as a control strategy tailored for the process control sector, Model Predictive Control (MPC) has since demonstrated its versatility across a broad spectrum of applications, including humanoid robotics and autonomous vehicle technology. The core principle of MPC is to transform the original control problem into a constrained optimization problem, solvable through standard techniques in mathematical optimization. Consistent with seminal text [29, p. 486], the following equations provide the general multiple shooting problem formulation for MPC:

$$\min_{x,u} \quad l_f(x_N) + \sum_{i=0}^{N-1} l(x_i, u_i), \tag{3.10a}$$

$$s.t \quad x_0 = x_{initial}, \tag{3.10b}$$

$$x_{i+1} = f(x_i, u_i), \qquad i = 0, ..., N-1 \tag{3.10c}$$

$$x_i \in \mathcal{X}, \qquad i = 1, ..., N-1 \tag{3.10d}$$

$$u_i \in \mathcal{U}, \qquad i = 0, ..., N-1 \tag{3.10e}$$

$$x_N \in \mathcal{X}_f, \tag{3.10f}$$

where $l(x, u)$ is the cost function, $l_f(x)$ is the terminal cost function and $f(x, u)$ is the discrete-time dynamics function. The states x and controls u are constrained to lie in their respective sets $\mathcal{X}$ and $\mathcal{U}$, and as a special case the terminal state x_N is constrained to lie in the terminal set $\mathcal{X}_f$. The objective of the problem described in Equation (3.10) is to find the optimal x^* and u^* that minimize the cumulative cost over a fixed prediction horizon N, while adhering to all constraints. By designing an adequate cost function which encodes the control task, such that minimal cost corresponds to optimal performance, one can represent arbitrarily complex control tasks as an optimization problem.

The utilization of a prediction horizon facilitates long-term control strategies, which is crucial for solving tasks that require intricate planning. Importantly, although the solution yields a sequence of control inputs u^*, only the first one u_0^* is applied to the system before the optimization problem is resolved, resulting in a closed-loop control strategy. Furthermore, it is assumed that each control u_0^* is held piece-wise constant between MPC iterations.

3.2.1 Linear vs nonlinear MPC

The formulation of the MPC problem, as outlined in Equation (3.10), is highly flexible, making no specific assumptions regarding the cost function, dynamics function, or constraint sets. However, the selection of these elements influence the complexity of the underlying optimization problem, making it advantageous to adhere to certain guidelines in formulating the MPC problem whenever possible. Specifically, it is often beneficial to:

1. Ensure that both the cost function $l(x, u)$ and the terminal cost function $l_f(x)$ are quadratic terms.
2. Represent the discrete dynamics $f(x, u)$ as a linear system.
3. Use linear inequalities to define polyhedral constraint sets $\mathcal{X}, \mathcal{U}$ and $\mathcal{X}_f$.

When these conditions are met, the optimization problem can be formulated as a convex Quadratic Program (QP) [36, p. 152]. Formulating the problem as a convex QP is preferred, since efficient solvers exist for this class of optimization problems [37]. Although the assumption of linear dynamics can be limiting for highly nonlinear systems, various strategies can be effectively employed to circumvent this constraint. For example, Linear Time-Varying (LTV) MPC uses separate linear models for each step in the prediction horizon. These models are typically attained via a first-order Taylor approximation:

$$\begin{aligned} f(x_i, u_i) \approx & \frac{\partial f(x^*_{i+1}, u^*_{i+1})}{\partial x}(x^*_{i+1} - x_i) + \frac{\partial f(x^*_{i+1}, u^*_{i+1})}{\partial u}(u^*_{i+1} - u_i) \\ & + f(x^*_{i+1}, u^*_{i+1}), \end{aligned} \tag{3.11}$$

where (x^*, u^*) is the previous solution to the optimization problem. The approximation method in Equation (3.11) is only accurate in the vicinity of the linearization point. Consequently, significant variations in solutions between iterations can lead to issues stemming from imprecise dynamics representations.

An alternative approach involves the use of the system's actual nonlinear dynamics, which results in a variant known as nonlinear MPC. In nonlinear MPC, the optimization problem becomes a Nonlinear Program (NLP), typically assumed to be non-convex. Even if the loss function and constraints are carefully chosen, achieving convexity with nonlinear dynamics is generally impractical. The issue of convexity is pivotal in optimization, as it assures the existence of a unique global minimum. In non-convex problems, multiple local minima can exist, posing the risk that the solver might converge to a suboptimal solution. [29, pp. 487–488] The computational burden associated with solving a non-convex NLP, even to a local minimum, is significantly greater than that of a convex QP, making linear MPC the preferred choice for time-sensitive applications. Besides convexity, another issue that warrants consideration is the numerical stability of the discrete dynamics $f(x, u)$, which result from integrating the continuous-time dynamics $f_c(x, u)$. This integra-

tion can cause numerical issues, especially for stiff nonlinear systems [29, pp. 500–501]. Nevertheless, owing to advancements in embedded computing and the development of highly optimized Sequential Quadratic Programming (SQP) optimization frameworks [38, 39], which iteratively approximate the nonlinear problem as a convex QP, it has become increasingly feasible to employ nonlinear MPC in real-time applications.

3.2.2 Stability of MPC

The stability of any closed-loop control system is a crucial attribute to analyze, especially for systems like autonomous vehicles, which have the potential to cause fatal accidents. In the realm of linear MPC, stability properties are generally well-established, although not as straightforward as the ones that can be derived for e.g. Linear Quadratic Regulators (LQR). Stability of linear MPC is often demonstrated by designing the cost function $l(x, u)$, terminal loss function $l_f(x)$ and terminal set $\mathcal{X}_f$ in a way that satisfies certain assumptions. These assumptions enable the finite-horizon value function $V_0^N = l_f(x_N) + \sum_{i=0}^{N-1} l(x_i, u_i)$ to be treated as a Control Lyapunov Function (CLF). [29, pp. 112–119, 136–138] Lyapunov functions $V(x)$ are energy-like scalar measures of the state x, which can be used to determine stability and asymptotic behaviour of a system with respect to some equilibrium point x_e. This concept is extended to control systems via CLFs, which incorporate the control input u into the stability analysis [40, pp. 234–238]. Mathematically, to indicate asymptotic stability, a CLF needs to satisfy the following properties:

$$V(x) > 0 \quad \text{for } x \neq x_e, \tag{3.12a}$$

$$V(x) = 0 \quad \text{for } x = x_e, \tag{3.12b}$$

$$\exists\, u \text{ s.t. } \dot{V}(x, u) < 0 \quad \text{for } x \neq x_e, \tag{3.12c}$$

which implies that with specific control inputs, all initial states converge towards x_e as time approaches infinity. The required properties for a CLF can be further tightened to show exponential stability, which bounds the time that it takes for a system to converge to x_e. [40, pp. 57–61, 70–73] Identifying a globally valid CLF can be challenging in the context of MPC due to the presence of constraints, therefore a local CLF is typically used instead [29, p. 136]. Formally, a local CLF states that there exists a constant value c, such that the sub-level set $\Omega_c := \{x \in \mathcal{R}^n : V(x) < c\}$ satisfies the properties in Equation (3.12). The set Ω_c is then referred to as the region of attraction, and can be viewed as an inner approximation of the stabilizable region within the state space. [40, pp. 73–79]

Extending the stability concepts from linear to nonlinear MPC comes with considerable challenges, many of which are detailed in [41]. For example, a seemingly straightforward approach involves utilizing linearizations of the nonlinear system to derive a local CLF,

paralleling methods used with linear systems [29, pp. 139–141]. However, a complication arises because the underlying NLP is typically solved only to a local minimum, resulting in a variant called suboptimal MPC [29, pp. 147–149]. Consequently, additional considerations are required to ensure the validity of such an approach [42]. Arguably many of these challenges stem from trying to design the nonlinear MPC problem in such a way, that finding an optimal solution implies stability, which is not true by default in a finite horizon setting. In contrast, some recent works have incorporated the stability conditions in Equation 3.12 as explicit constraints in the MPC optimization problem [30, 31]. This more direct approach sidesteps many of the common challenges, at the cost of a more complex MPC problem. It is important to recognize, however, that the endeavor to develop a local CLF persists across methodologies, frequently requiring iterative refinement to enhance the region of attraction. In pursuit of efficiency in this process, techniques rooted in optimization, as demonstrated in [43], have shown promise in facilitating the automatic synthesis of local CLFs.

3.2.3 Recursive feasibility and safety of MPC

Equally vital for ensuring reliable operation of the system is the notion of recursive feasibility, which states that if the solution to the optimization problem in Equation (3.10) is applied at iteration k, then the problem needs to remain feasible at iteration $k+1$. Feasibility in this context refers to the existence of a solution that satisfies all constraints. To provide the theoretical framework for guaranteeing recursive feasibility, one first defines $\mathcal{R}$ as the set of states for which all constraints can be satisfied using some control input u. This set $\mathcal{R}$ is assumed to be control invariant, meaning that for any state $x_i \in \mathcal{R}$ there exists a control input u that ensures the subsequent state x_{i+1} also lies within $\mathcal{R}$. With an initial state $x_i \in \mathcal{R}$ and a control invariant terminal set $\mathcal{X}_f$, the recursive feasibility of the optimization problem in Equation 3.10 is guaranteed, provided there are no external disturbances. [29, pp. 111–112] However, while such a theoretical framework can be used to formally express the requirements for recursive feasibility, the exact computation of these control invariant sets is challenging except for some special cases [29, p. 166].

Fortunately, the field of safety critical control has produced practical tools, such as Hamilton-Jacobi (HJ) reachability and Control Barrier Function (CBF), which can be used to compute control invariant sets for a system. By extension, these methods can be used to ensure the forward recursive property for MPC. The HJ reachability method provides a systematic approach to compute a maximal control invariant set $\mathcal{S}$ for a system, however, it is known to be intractable for large problems [44, 45]. The CBF method on the other hand is well suited for large problems, but typically requires a handcrafted component and only produces a subset $\mathcal{C}$ of the maximal control invariant set $\mathcal{S}$ [45, 46]. Formally, a CBF is a Lyapunov like function $h(x)$ which defines the safe, control invariant set $\mathcal{C}$ as follows:

$$\mathcal{C} := \{x \in \mathcal{R}^n : h(x) \geq 0\}, \tag{3.13a}$$

$$s.t. \quad \exists u \quad \dot{h}(x,u) \geq -\beta(h(x)) \quad \text{for all } x \in \mathcal{C}, \tag{3.13b}$$

where $\beta(h(x))$ is a class $\mathcal{K}$ function, i.e. positive definite and strictly increasing. Intuitively, the choice of β controls the rate at which $h(x)$ can descend towards the set boundary, affecting how conservative the resulting controller is. In the context of MPC, $h(x)$ could be interpreted, e.g., as a distance measure to an obstacle that we want to avoid a collision with. The CBF then ensures that when nearing the point of collision, there always exists a control input u that moves us further away from the obstacle. The practical integration of CBFs to a MPC problem formulation has been demonstrated in [31].

3.2.4 Robust and stochastic MPC

The stability and feasibility guarantees presented in Section 3.2.2 rely on the assumption of a perfect dynamics model and the absence of external disturbances. These assumptions are generally unrealistic in real-world applications, which has motivated the development of specialized MPC frameworks that account for these factors. These frameworks are often categorized into two main types: robust MPC and stochastic MPC. According to [29, p. 193], both of these approaches work in the same principal manner. The problem described in Equation (3.10) is adapted such that the discrete dynamics are represented by $f(x,u,w)$ instead, where the disturbances and model uncertainties are encapsulated in the term $w \in \mathcal{W}$. In the case of additive disturbance, the resulting dynamics are of the form $f(x,u,w) = f(x,u) + w$.

The divergence between the two approaches arises when considering the properties of the set $\mathcal{W}$. Robust MPC assumes a bounded $\mathcal{W}$, indicating that disturbances and model uncertainties are limited and well-defined. Under these conditions, it is often possible to design a controller that is robust to disturbances by considering the worst-case scenario. A practical example of robust MPC is tube-MPC, where the system is confined to operate within a tube, that represents the region where the system can tolerate the worst case disturbances. [29, pp. 223–228] In contrast, stochastic MPC operates under the assumption that $\mathcal{W}$ is unbounded. This necessitates a probabilistic approach, focusing on satisfying probabilistic constraints when arbitrarily large disturbances may occur. Probabilistic constraints, as described by [29, pp. 246–248], can be expressed in the form:

$$P(x \in \mathcal{X}) > 1 - \epsilon, \tag{3.14}$$

where $P(x \in \mathcal{X})$ denotes the probability of x being in the constraint set $\mathcal{X}$, and ϵ signifies the maximum acceptable probability of violating the constraint.

4. IMPLEMENTATION

This chapter presents the system utilized to gather the results for this study. Initially, an overview of the simulation environment and the driving software is provided. Subsequently, the chapter introduces the essential components for the implementation of uncertainty-aware MPC, focusing on the vehicle dynamics model and the contouring cost function. The chapter concludes by detailing the uncertainty-aware MPC problem formulation. The code used in this work is fully available at [47].

4.1 Simulation environment and driving software

Assetto Corsa Competizione (ACC) was selected as the simulation environment for this thesis due to its highly detailed vehicle dynamics simulation. While other simulators were considered, including the Formula Student Driverless simulator [48], the Learn To Race simulator [49], and a Simulink model created by the TUM autonomous racing team [50], none seemed to match ACC in simulation detail. The inability of ACC to simulate sensor data, such as from cameras and LIDAR, did not detract from its suitability given the deliberate focus on the aspects of control rather than perception or environment sensing.

The lack of a programming API in ACC necessitated an unconventional approach to facilitate communication between the simulation and the driving software, accomplished by reading the shared memory of ACC, and by utilizing an emulated game controller. The driving software, developed in Python, operates with two main loops: an observation loop and a control loop. The observation loop polls the shared memory at 100Hz to gather vehicle state data such as velocities and accelerations. The control loop runs at 20Hz, using the vehicle state data to compute control inputs via MPC, which are then transmitted back to ACC using the emulated game controller.

The car's state information serves dual purposes: immediate control and as training data for a sparse GP. Notably, the training data goes through a learning buffer, which is essential since ACC's shared memory does not provide angular acceleration, requiring the application of a Savitzky-Golay filter to derive a smooth estimate from the angular velocity data [51]. Despite the predominantly noise-free nature of the data, this step is also beneficial to remove any outliers. A comparison of the car states required for control and learning against the data available from shared memory is shown in Table 4.1.

State name	Used for control	Used for learning	Available
Position and heading	Yes	No	Yes
Linear velocity	Yes	Yes	Yes
Angular velocity	Yes	Yes	Yes
Linear acceleration	No	Yes	Yes
Angular acceleration	No	Yes	No

Table 4.1. *Vehicle states, their usage and availability.*

The MPC controller is implemented using Acados, an open-source optimal control framework [38]. The associated sparse GP model is integrated in to the optimization problem using Casadi symbolics [52], with the necessary precomputations done using PyTorch. Due to compatibility issues with linear algebra libraries that Acados relies on, the MPC controller must operate within a Linux environment. In contrast, the chosen simulator functions exclusively on Windows. To reconcile this, the Windows Subsystem for Linux (WSL2) facilitates the execution of the MPC controller, isolating it from the main driving software. Communication between the driving system and the MPC controller is efficiently managed through a gRPC node, ensuring minimal latency. The overall system architecture, which integrates these components, is illustrated in Figure 4.1.

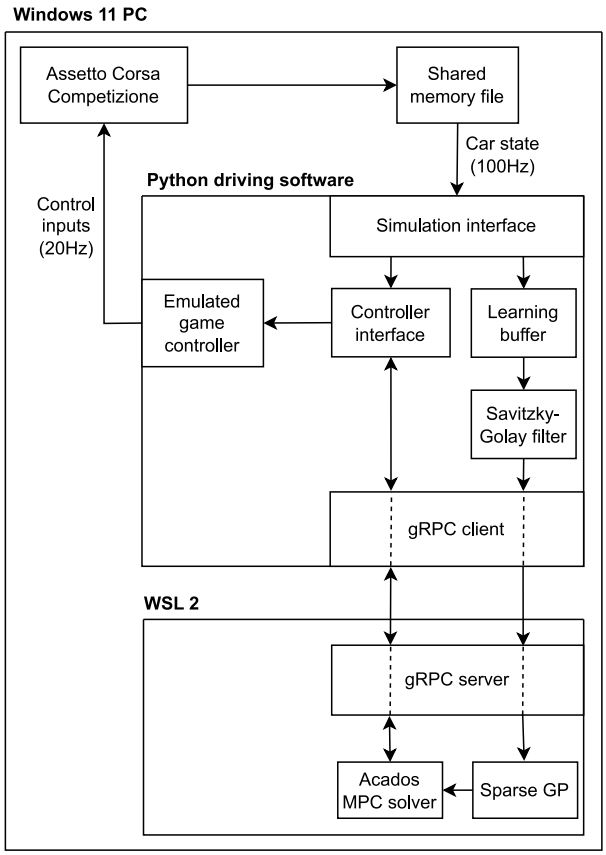

Figure 4.1. *The system architecture.*

4.2 Vehicle dynamics modelling

In alignment with prior work [9], this thesis divides the task of vehicle dynamics modeling into two specific components: nominal and auxiliary dynamics. The nominal dynamics are represented by a first-principles based model, which serves as the baseline framework for capturing the vehicle's behaviour. On the other hand, the auxiliary dynamics are represented by a sparse GP, which attempts to account for phenomena not encompassed by the nominal model.

4.2.1 Nominal vehicle dynamics

The nominal vehicle dynamics in this thesis are represented by a dynamic single track bicycle model, which has seen application in numerous preceding studies [9, 22, 24]. The bicycle model employed here has eight states, which are governed by the following differential equations:

$$f_c(x,u) = \begin{bmatrix} \dot{x}_{car} \\ \dot{y}_{car} \\ \dot{\theta} \\ \dot{v}_x \\ \dot{v}_y \\ \dot{\omega} \\ \dot{\phi} \\ \dot{\delta} \end{bmatrix} = \begin{bmatrix} v_x \cdot \cos(\theta) + v_y \cdot \sin(\theta), \\ v_x \cdot \sin(\theta) + v_y \cdot \cos(\theta), \\ \omega, \\ \frac{F_x}{m} + v_y \omega, \\ \frac{1}{m} \cdot (F_{fy} \cdot \cos(\phi) + F_{ry}) - v_x \omega, \\ \frac{1}{I} \cdot (F_{fy} \cdot l_f \cdot \cos(\phi) - F_{ry} \cdot l_r) \\ \Delta\phi \\ \Delta\delta \end{bmatrix}, \tag{4.1a}$$

where m is the car mass, I is the car inertia, and l_r and l_f are the distances from car center of mass to the rear and front wheel axles respectively. The model inputs are the rate of change for both the steering angle ϕ and acceleration command δ, denoted $\Delta\phi$ and $\Delta\delta$ respectively. The bicycle model and the associated coordinate system is depicted in figure 4.2.

The lateral forces F_{fy} and F_{ry} are produced by a simplified variant of the Pacejka tire model [53], which only considers lateral wheel slip:

$$F_{f/ry} = D_{f/r} \cdot \sin(C_{f/r} \cdot \operatorname{atan}(B_{f/r} \cdot \alpha_{f/r})), \tag{4.2}$$

where $[D_{f/r}, C_{f/r}, B_{f/r}]$ are learnable parameters, and the front (α_f) and rear (α_r) lateral

tire slip angles are computed as:

$$\alpha_f = \text{atan2}(v_y + \omega \cdot l_f,\ v_x) - \phi, \tag{4.3a}$$

$$\alpha_r = \text{atan2}(v_y - \omega \cdot l_r,\ v_x). \tag{4.3b}$$

The longitudinal force F_x is described by the following simplistic drive train model:

$$F_x = (K_{m0} + K_{m1} v_x)\delta - K_{r0}(1 - \tanh(K_{r1} v_x)) - K_{r2} v_x^2, \tag{4.4}$$

where K_{m0}, K_{m1}, K_{r0}, K_{r1} and K_{r2} are learnable parameters.

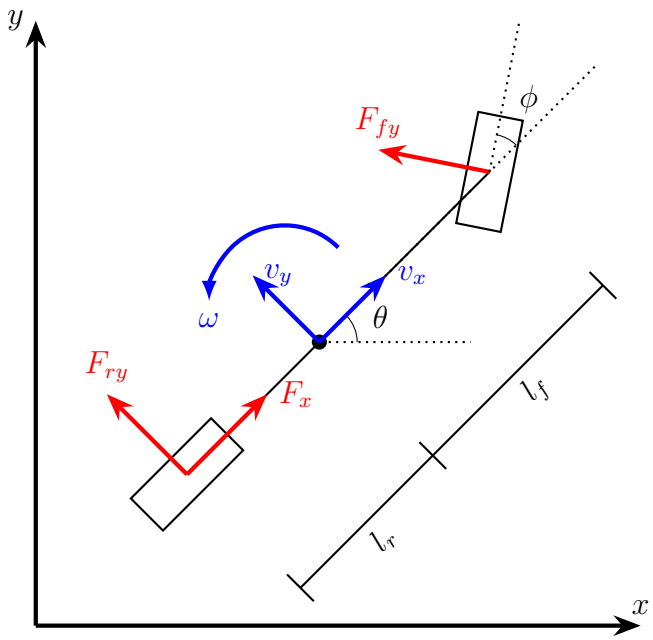

Figure 4.2. *Depiction of the dynamic single track bicycle model.*

For use within MPC, the continuous time dynamics $f_c(x, u)$ described in Equation (4.1) are converted to discrete-time dynamics $f(x, u)$ via the 4th order Excplicit Runge-Kutta method:

$$k_1 = f_c(x, u), \tag{4.5a}$$

$$k_2 = f_c(x + \frac{h}{2} \cdot k_1, u), \tag{4.5b}$$

$$k_3 = f_c(x + \frac{h}{2} \cdot k_2, u), \tag{4.5c}$$

$$k_4 = f_c(x + h \cdot k_3, u), \tag{4.5d}$$

$$f(x, u) = \frac{h}{6} \cdot (k_1 + 2k_2 + 2k_3 + k_4), \tag{4.5e}$$

where h denotes the sampling time [29, pp. 477–479]. The Runge-Kutta method was chosen here over the simpler Euler method to improve numerical stability.

4.2.2 GP vehicle dynamics

It can be observed from Equation (4.1) that the position states, denoted as $[x, y, \theta]$, are governed exclusively by kinematic equations, thereby rendering them negligible sources of modeling error. Consequently, the GP dynamics apply corrections only to the velocity states $[v_x, v_y, \omega]$. To align this approach with prior work [9], a selection matrix $B_{GP} = [\mathbf{0}_{3x3}, \mathbf{I}_{3x3}, \mathbf{0}_{2x3}]$ is introduced. With this matrix, the targets of the GP dynamics can be formulated as the residuals between the actual and predicted states, expressed mathematically as:

$$Y = B_{GP}^{\dagger}(x - f_c(x, u)) = d_{true}(z) + \mathbf{w}, \tag{4.6}$$

where $\dagger$ is the pseudo-inverse operator and $d_{true}(z)$ represents the true residuals corrupted by noise $\mathbf{w}$. The regression features z are defined as $z = C_{GP}x$, which with the selection matrix $C_{GP} = [\mathbf{0}_{3x3}, \mathbf{I}_{5x3}]$, results in $z = [v_x, v_y, \omega, \phi, \delta]$. Notably, unlike prior work, the continuous-time dynamics were chosen as the target via the use of f_c, since it accommodates the use of nonuniform sampling times later on. By adopting this framework, the GP dynamics will yield a distribution defined as:

$$Y_{GP} \sim GP(\mu^d(z), \Sigma^d(z)), \tag{4.7}$$

where $\mu^d(z)$ and $\Sigma^d(z)$ are the mean and covariance of the residual dynamics.

To train the residual dynamics online, data is fed from the simulator to the GP as depicted in Figure 4.1. When the GP training data buffer is full, it is updated based on the posterior variance of all current data points, computed according to Equation (3.7). If a new candidate point has a higher posterior variance than the median of all current points, the point with the lowest posterior variance gets replaced. Additionally, to encourage continuous learning, the posterior variance of each existing data point is scaled by λ, which is computed as:

$$\lambda_i = e^{-\frac{(t-t_i)^2}{h}}, \tag{4.8}$$

where λ_i refers to the scaler for point i, t is the current time, t_i is the time when point i was added and h is a hyperparameter. This scaling procedure, also adopted in [9], ensures that as the vehicle dynamics evolve through time, old data points get replaced with more relevant ones.

4.2.3 Combined dynamics

Using the variance propagation equations presented in [12], the combined discrete-time dynamics, which are now a distribution, can be expressed as:

$$\mu_{i+1}^x = \widetilde{f}(\mu_i^x, u_i), \tag{4.9a}$$

$$\Sigma_{i+1}^x = B_{GP}\left(h^2 \cdot \Sigma^d(C_{GP}\mu_i^x)\right) B_{GP}^T + \nabla_x \widetilde{f}(\mu_i^x, u_i)\Sigma_i^x \left(\nabla_x \widetilde{f}(\mu_i^x, u_i)\right)^T, \tag{4.9b}$$

where i denotes the prediction stage, h is the sampling time and $\widetilde{f}(\mu_i^x, u_i)$ is defined as:

$$\widetilde{f}(\mu_i^x, u_i) = f(\mu_i^x, u_i) + h \cdot B_{GP}\mu^d(C_{GP}\mu_i^x). \tag{4.10}$$

As Equation 4.9 reveals, the Euler-method was used to discretize the GP dynamics. It should be noted that the variance update in 4.9 is merely a linear approximation, since the exact computation would be intractable [54].

4.3 Contouring control

Inspired by previous works [9, 22, 24], the racing problem is formulated as an instance of Model Predictive Contouring Control (MPCC) [21]. In MPCC, the primary objective is to closely follow a two-dimensional reference path as fast as possible. This is realized by a nonlinear, non-convex cost function penalizing deviations perpendicular to the reference path and incentivizing forward progress along it.

The reference path is represented by arc-length parameterized functions $r_x(s)$ and $r_y(s)$ which output the x and y coordinates for any path position $s \in [0, L]$, where L is the total arc length of the path. In the context of autonomous racing, the contouring error E_c is then defined to be the lateral deviation between the car $[x_{car}, y_{car}]$ and the reference path $[r_x(s_{car}), r_y(s_{car})]$, where s_{car} is the projection of car coordinates that minimizes the following optimization problem:

$$s_{car} = \min_s \left((x_{car} - r_x(s))^2 + (y_{car} - r_y(s))^2 \right). \tag{4.11}$$

Continuously solving Equation (4.11) inside the MPC problem would be computationally intractable, which is addressed by using the additional state $\hat{s}$ in the MPC formulation. The state $\hat{s}_i$ represents the approximate path position of the car at prediction step i, and is governed by the simple dynamics:

$$\hat{s}_{i+1} = \hat{s}_i + \nu_i, \tag{4.12}$$

where $\nu \in [0, \nu_{max}]$ is the corresponding control variable. With this formulation the optimization problem in Equation (4.11) can be precomputed once before each MPC iteration to get the initial car path position $\hat{s}_0$, after which the car path position is updated each prediction step using Equation (4.12). To control the approximation quality of $\hat{s}$, the lag error E_l is introduced. It is defined as the arc length distance between s_{car} and $\hat{s}$. Neither the quantities E_c or E_l can be computed exactly without the real path position s_{car}, which leads to the following approximations being used instead [21]:

$$\hat{E}_c = \sin(\psi(\hat{s})) \cdot (x_{car} - r_x(\hat{s})) - \cos(\psi(\hat{s})) \cdot (y_{car} - r_y(\hat{s})), \tag{4.13a}$$

$$\hat{E}_l = -\cos(\psi(\hat{s})) \cdot (x_{car} - r_x(\hat{s})) - \sin(\psi(\hat{s})) \cdot (y_{car} - r_y(\hat{s})), \tag{4.13b}$$

$$\psi(\hat{s}) = \operatorname{atan2}\left(\frac{\nabla r_y(\hat{s})}{\nabla r_x(\hat{s})}\right). \tag{4.13c}$$

Importantly, the approximation $E_c \approx \hat{E}_c$ is accurate when we have a sufficiently small approximate lag error $\hat{E}_l$. The resulting contouring scheme is visualized in Figure 4.3.

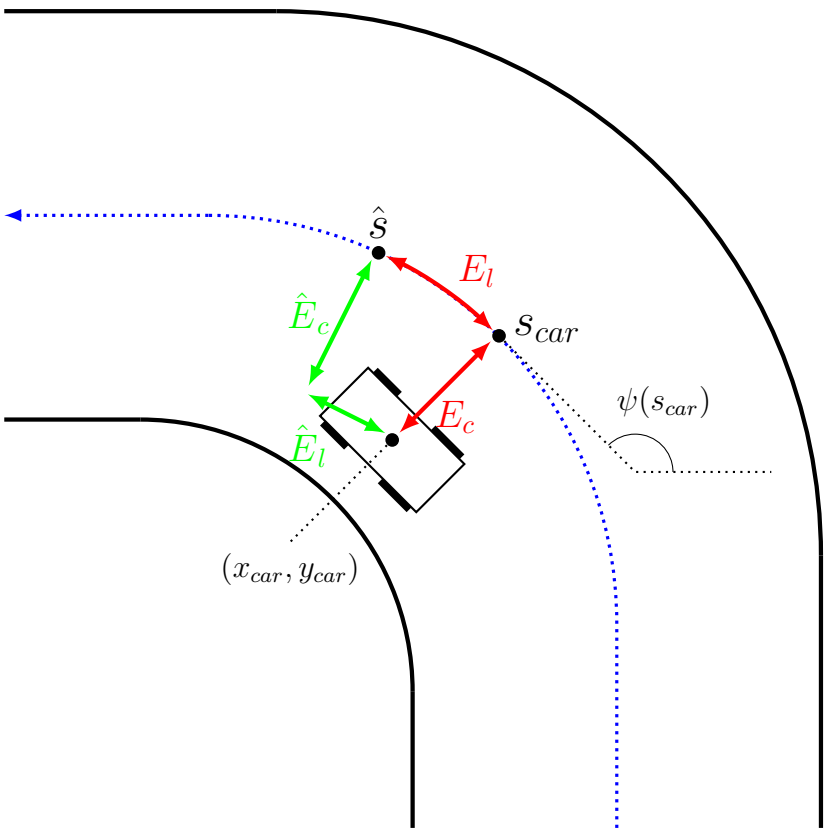

Figure 4.3. *Depiction of the contouring errors, shown here with the race track center line as a reference path.*

The fastest approach through the reference path is encouraged by maximizing the approximate path progress $\hat{s}$ while minimizing the approximate contouring error $\hat{E}_c$. The approximate lag error $\hat{E}_l$ needs to also be minimized to ensure the approximations re-

main accurate. This leads to the contouring cost function:

$$l_c(x, u) = k_c\hat{E}_c - k_s\hat{s} + k_l\hat{E}_l, \tag{4.14}$$

where k_c, k_s and k_l are weights used to tune the behaviour of the method. Importantly, to ensure a small approximation error, the weights should be selected such that $k_l \ggg k_s + k_c$. The ratio of k_s and k_c is application specific, however in a racing scenario it is beneficial to set $k_s \ggg k_c$, since additional constraints can be used to enforce that the car stays within race track boundaries.

4.4 Uncertainty-aware MPC problem formulation

Using the ingredients from Section 4.2 and Section 4.3 the uncertainty-aware MPC can now be formulated. First, the following state constraints are enforced for all stages of the optimization problem:

$$\begin{bmatrix} \phi_{min} \\ \delta_{min} \end{bmatrix} < \begin{bmatrix} \phi \\ \delta \end{bmatrix} < \begin{bmatrix} \phi_{max} \\ \delta_{max} \end{bmatrix}, \tag{4.15a}$$

$$\begin{bmatrix} 0 \\ \Delta\phi_{min} \\ \Delta\delta_{min} \end{bmatrix} < \begin{bmatrix} \nu \\ \Delta\phi \\ \Delta\delta \end{bmatrix} < \begin{bmatrix} \nu_{max} \\ \Delta\phi_{max} \\ \Delta\delta_{max} \end{bmatrix}. \tag{4.15b}$$

Next, a constraint is introduced to limit the combined longitudinal and lateral acceleration of the vehicle. The adoption of this constraint is essential due to the limitations of the pure lateral slip tire model used in the nominal dynamics, which tends to overestimate the tire force during combined slip scenarios. Although recent findings suggest that the actual constraint is more accurately represented by a diamond shape, as discussed in [18], an ellipsoidal constraint is used here instead. This aligns with prior work [9, 24], and minimizes the number of nonlinear constraints. Consequently, the following constraint is used to limit the longitudinal (a_x) and lateral (a_y) accelerations:

$$\left(\frac{a_x + K_{ax}}{\hat{a}_x - \sqrt{\Sigma^d(z)_{ax}}}\right)^2 + \left(\frac{a_y}{\hat{a}_y - \sqrt{\Sigma^d(z)_{ay}}}\right)^2 < 1, \tag{4.16}$$

where $\hat{a}_x$ and $\hat{a}_y$ symbolize the maximum permissible accelerations, which are readily identifiable from a GG-diagram derived from experimental data. In contrast, a_x and a_y are calculated using the nominal vehicle dynamics outlined in Equation (4.1) in conjunction

with the GP dynamics mean predictions $\mu^d(z)$:

$$a_x = \frac{F_x}{m} + \mu^d(z)_{ax}, \tag{4.17a}$$

$$a_y = \frac{2}{m} \cdot (F_{fy} \cdot \cos(\phi) + F_{ry}) + \mu^d(z)_{ay}, \tag{4.17b}$$

where $\mu^d(z)_{ax}$ and $\mu^d(x)_{ay}$ specifically extract longitudinal and lateral accelerations. Unlike prior work, the GP variance predictions $\Sigma^d(z)_{ax}$ and $\Sigma^d(z)_{ax}$ are used to tighten the acceleration constraint in Equation (4.16). Additionally, inspired by [19], the following terminal lateral acceleration constraint is incorporated:

$$a_y = 0, \tag{4.18}$$

which enforces a constant turn radius in the terminal phase. This addition was empirically observed to enhance system stability.

In addition to acceleration constraints, the behaviour of the vehicle is further regulated by imposing a constraint on the rear slip ratio:

$$-\alpha_{r_{max}} \leq \alpha_r \leq \alpha_{r_{max}}. \tag{4.19}$$

This constraint serves as a crucial measure against oversteer, a very unstable operating condition that is not accurately represented by the simplistic nominal dynamics model.

Following the constraints imposed on the vehicle dynamics, the vehicle is further restricted to remain within the race track boundaries. This is accomplished using the following constraint:

$$(c_x(\hat{s}) - x_{car})^2 + (c_y(\hat{s}) - y_{car})^2 < \left(\frac{c_w(\hat{s})}{2}\right)^2 - R_{GP}(\Sigma_i^{xy}), \tag{4.20}$$

where $c_x(\hat{s})$, $c_y(\hat{s})$, and $c_w(\hat{s})$ are arc-length parameterized splines that define the track's center line coordinates (x, y) and its width w. The term $R_{GP}(\Sigma_i^{xy})$ serves to reduce the effective track width in response to the car's positional uncertainty. It is calculated as:

$$R_{GP}(\Sigma_i^{xy}) = \sqrt{\chi_2^2(1-\epsilon)\lambda_{max}(\Sigma_i^{xy})}, \tag{4.21}$$

where $\chi_2(1-\epsilon)$ denotes the chi-squared distribution's quantile function for a given constraint violation probability ϵ, and $\lambda_{max}(\Sigma_i^{xy})$ selects the largest eigenvalue from the covariance matrix Σ_i^{xy}. The matrix Σ_i^{xy} signifies the x and y positional uncertainties at stage i of the prediction horizon, computed using the uncertainty propagation mechanism outlined in Equation (4.9). Constraints of this nature can be viewed as a practical implementation of the chance constraint in Equation (3.14) [11]. Figure 4.4 visually illustrates

this concept, demonstrating the impact of position uncertainty on the track constraint.

Due to the real-time demands of the task, computing $R_{GP}(\Sigma^{xy}i)$ directly within the optimization problem proved unfeasible. To address this, the values were precomputed using the previous MPC solution shifted forwards by one time step. This same approach was applied to the variance terms in Equation (4.16). Additionally, due to the open loop nature of the predictions, Σ_i^{xy} could potentially grow unbounded. Similar to previous work [9], this issue was addressed empirically, by only computing Σ_i^{xy} for the first half of the prediction horizon, beyond which it was kept constant.

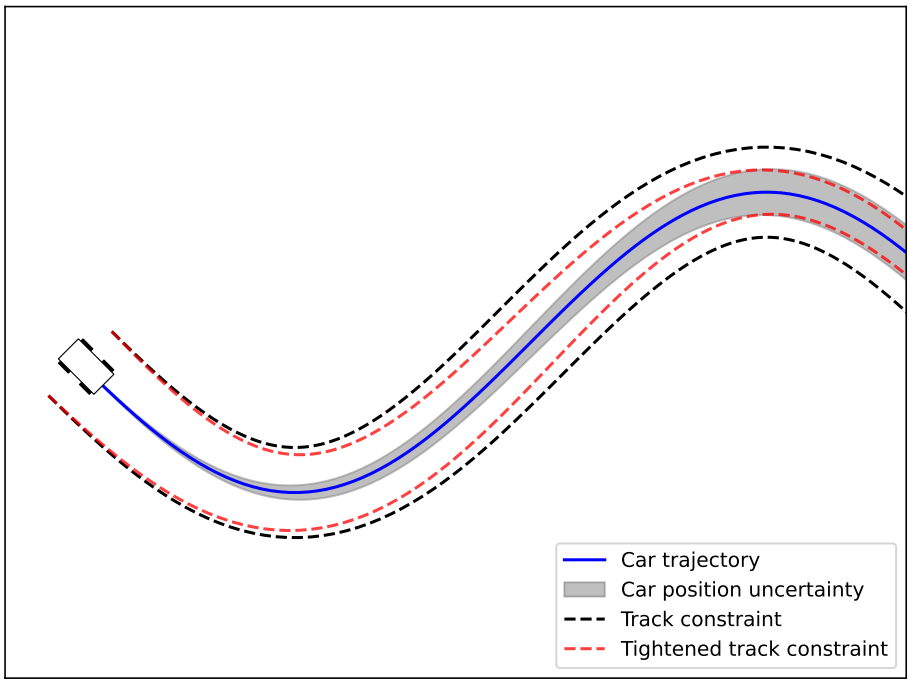

Figure 4.4. *Visualization of the constraint tightening procedure used in uncertainty-aware MPC. As the car position uncertainty grows during the prediction horizon, the effective track width is decreased to guarantee that the car does not exit the track.*

The uncertainty-aware MPC is formulated with a nonlinear least squares objective, using the cost function $l(x, u) = \zeta^T Q \zeta$, and a terminal cost function $l_f(x, u) = \zeta_f^T Q \zeta_f$. Here Q and Q_f are diagonal matrices which contain the cost coefficients, and the vectors ζ and ζ_f are defined as:

$$\zeta = \left[\hat{E}_c, \hat{E}_l, E_{vx}, (\nu + 1), \phi, \Delta\phi, \Delta\delta\right]^T, \tag{4.22a}$$

$$\zeta_f = \left[\hat{E}_c, \hat{E}_l\right]^T. \tag{4.22b}$$

Here $\hat{E}_c$ and $\hat{E}_l$ are the contouring errors, computed as described in Section 4.3, and $E_{vx} = v_x^*(\hat{s}) - v_x$ is the error from a target velocity profile $v_x^*(\hat{s})$. Deviating from previous works, the addition of a velocity profile helps the car find the optimal brake point in corners.

With the above definitions, the resulting uncertainty-aware MPC problem formulation is:

$$\min_{x,u} \quad l_f(x_N) + \sum_{n=0}^{N-1} l(x_n, u_n), \tag{4.23a}$$

$$s.t \quad x_0 = x_{initial}, \tag{4.23b}$$

$$x_{i+1} = f(x_i, u_i) + h \cdot B_{GP}\mu^d(x_i, u_i), \qquad i = 0, ..., N-1 \tag{4.23c}$$

$$(4.15a, 4.16, 4.19, 4.20), \qquad i = 1, ..., N-1 \tag{4.23d}$$

$$(4.15b), \qquad i = 0, ..., N-1 \tag{4.23e}$$

$$(4.15a, 4.18, 4.20). \qquad i = N \tag{4.23f}$$

For clarity in notation, Equation (4.23d) defines the state constraint set $\mathcal{X}$, Equation (4.23e) defines the control constraint set $\mathcal{U}$, and Equation (4.23f) defines the terminal constraint set $\mathcal{X}_f$.

4.5 Implementation details

The MPC described in Equation (4.23) was implemented using the Python interface for Acados, which generated efficient C-code for execution. The HPIPM Sequential Quadratic Programming (SQP) solver was employed for solving the problem [55]. To enhance numerical conditioning, all state variables were scaled to similar magnitudes of around 1. The prediction horizon was set to $N = 50$, which in conjunction with a nonuniform sampling grid, ranging linearly from 50 ms to 100 ms, resulted in a total lookahead time of 3.75 s. During execution 5 SQP real-time iterations were allocated for solving the problem, with 100 iterations allocated to the underlying QP solver. The delay from measurement to control was compensated using the nominal dynamics model.

Since the ingegration of Control Lyapunov Functions (CLFs) and Control Barrier Functions (CBFs) in to the optimization problem proved computationally prohibitive, all constraints were implemented as soft constraints to maintain feasibility. The stability of the system was not studied beyond conclusions drawn empirically from simulations, which seemed to indicate that a sufficiently large terminal contouring error coefficient along with a terminal lateral acceleration constraint helped maintain stability.

The residual dynamics were modeled using independent sparse GPs for each output dimension, employing the Automatic Relevance Detection (ARD) kernel. Each GP was allocated a separate dataset, limited to a maximum of 500 samples to balance detail with computational tractability. For integration within the MPC, 20 inducing points were used, which were strategically sampled from the prior solution, to ensure uniform temporal spacing throughout the prediction horizon. The hyperparameters of the GP kernel, identified offline from experimental data, were maintained constant during execution.

5. RESULTS

The performance of the uncertainty-aware Model Predictive Control (MPC) scheme was evaluated through a comparative analysis against a nominal MPC, which was structurally identical to the uncertainty-aware MPC, except it did not incorporate residual Gaussian Process (GP) dynamics or the associated constraint modifications.

The evaluation of each controller involved two experiments. Initially, both controllers were assessed in ideal track conditions, identical to the scenario used to gather training data for the nominal dynamics. Next, the track conditions were made increasingly slippery, to assess how the controllers handle modelling error caused by changes in the environment. A video showcasing these experiments can be found at: `https://youtu.be/SENTHq9ONTw`.

5.1 Experimental setup

The experiments were conducted on the Ricardo Tormo circuit, which was selected due to its flat elevation profile and variety of turn shapes, making it ideal for testing vehicle dynamics. The track coordinates were manually collected from Google Maps API, and consequently post-processed to be consistent with the in-game coordinates system. The car used in the experiments was the Ferrari 488 GT3 Evo. All of the realism settings of the game were enabled, and the artificial stability assist was disabled.

To produce a reference trajectory for both controllers, a trajectory planning problem was solved offline using the nominal dynamics model and a contouring cost. This offline approach facilitated the use of a much longer planning horizon, and the employment of a more robust NLP solver, specifically IPOPT. The track and the associated reference trajectory is shown in Figure 5.1. The reference trajectory is not time-optimal, resulting in a lap time of 1:35.40, while the record lap time for the track set by a human is 1:29:60. Regardless, it should serve as a good representation of the best lap time attainable with the used nominal dynamics model and the associated constraints.

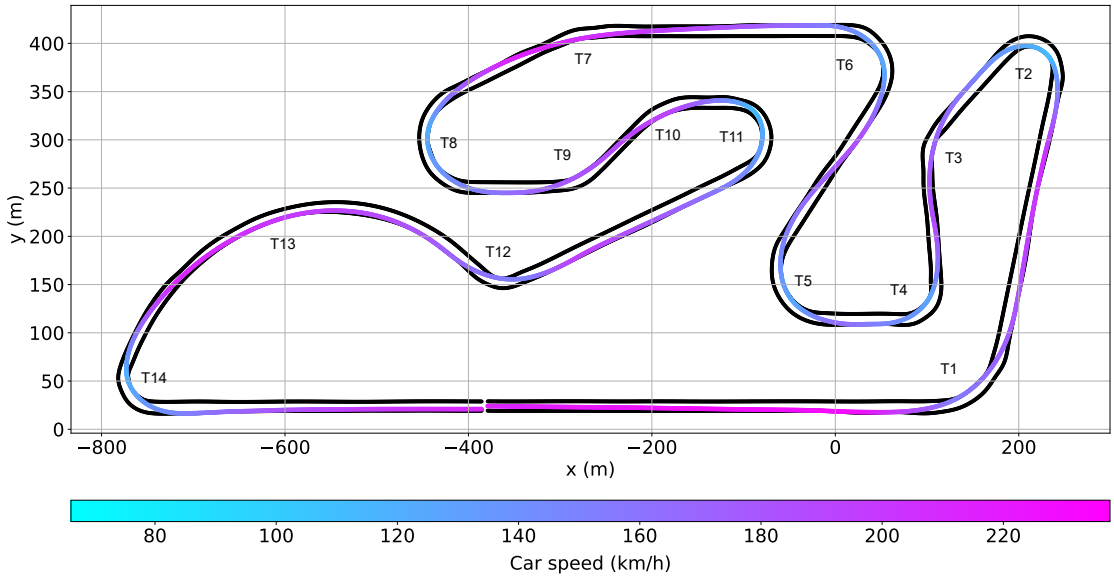

Figure 5.1. *The race track and the reference trajectory.*

The parameters for the nominal dynamics model and the GP kernel were identified from experimental data, collected by manually driving around various tracks in ideal conditions for roughly 2 hours. Covariance Matrix Adaptation (CMA) was used to fit the nominal model parameters [56], while gradient descent was used to find the GP kernel parameters. The MPC problem was identical for both controllers, including the cost coefficients, constraints and solver settings, all tuned in a way that maximized the nominal controller performance.

The identified nominal car parameters can be found in Appendix A1. All results were collected on a desktop PC with an AMD Ryzen 3900x CPU, 64GB RAM and a NVIDIA 3090 GPU.

5.2 Performance in ideal conditions

In this experiment, both controllers drove 10 consequent laps in ideal conditions following the reference trajectory shown in Figure 5.1. The track conditions were identical to those under which data was collected for training the dynamics model. The main objective was to establish a baseline for each controller when the dynamics are well known. Some amount of robustness is expected from each controller even in this experiment, since the track elevation profile, curbs or other track features are in no way considered, and could thus be considered external disturbances.

To get a running start, the controllers drove a single lap before the result collection commenced. The GP dynamics were allowed to train during this initial lap. To avoid the results from being skewed due to individual bad lap times, potentially caused by the non-real-time nature of the evaluation setup, the worst lap time was discarded.

The lap time results are presented in Table 5.1. The worst lap time is greyed out to highlight it was not used when computing the mean and standard deviation.

Table 5.1. *Comparison of lap times in ideal conditions: nominal MPC vs uncertainty-aware MPC*

Lap	**MPC (m:s)**	**UCA-MPC (m:s)**
Lap 1	1:37.87	1:36.53
Lap 2	1:37.73	1:36.14
Lap 3	1:37.69	1:36.09
Lap 4	1:37.68	1:35.71
Lap 5	1:37.65	1:35.71
Lap 6	1:37.79	1:35.73
Lap 7	1:37.91	1:35.75
Lap 8	1:37.79	1:35.74
Lap 9	1:37.75	1:35.71
Lap 10	1:37.78	1:35.74
Mean $\pm$ Std (Best 9)	**1:37.75 $\pm$ 0.18**	**1:35.81 $\pm$ 0.16**

As indicated in Table 5.1, the uncertainty-aware MPC (UCA-MPC) demonstrates a lap time improvement of approximately 1.94 s or 2$\%$. Notably, both controllers effectively tolerated the accumulating variations in tire and brake temperatures, as reflected by the small standard deviations in lap times.

The modeling accuracy of the nominal and GP-augmented models was next compared using one-step-ahead prediction errors. To facilitate a direct comparison of errors, the analysis was conducted using data from the 9 best laps of the uncertainty-aware MPC. Table 5.2 lists the median of the absolute errors for both dynamics model types.

Table 5.2. *Median one-step-ahead prediction errors in ideal conditions*

Dynamics	$\lvert a_x \rvert$ (m/s^2)	$\lvert a_y \rvert$ (m/s^2)	$\lvert \dot{\omega} \rvert$ (deg/s^2)
Nominal	0.52	0.67	7.89
Nominal+GP	0.35	0.39	3.91
Error reduction	32$\%$	41$\%$	50$\%$

The data presented in Table 5.2 clearly illustrates a notable reduction in the error magnitudes for all output dimensions, when using the GP-augmented model. To extend this analysis, Figure 5.2 presents a density plot that graphically represents the distribution of these absolute one-step-ahead prediction errors.

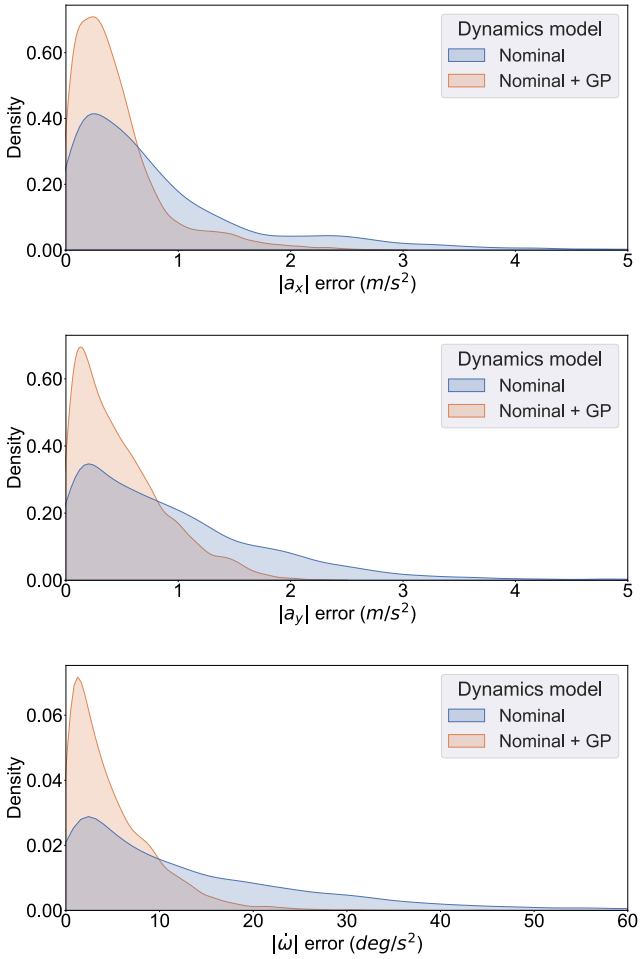

Figure 5.2. *Density plot of one-step-ahead absolute prediction errors in ideal conditions.*

It is clear from visual inspection of the plot in Figure 5.2, that the GP-augmented model demonstrates a significant reduction in the variance of errors, as evidenced by the tighter concentration of the density curve around the lower error values. This indicates that the GP-augmented model consistently predicts with fewer errors.

Regarding the computational performance, the nominal MPC had a mean solution time of (16 ± 5) ms for the optimization problem, compared to (30 ± 6) ms for the uncertainty-aware MPC. Notably, around 5 ms of the solution time for the uncertainty-aware MPC was dedicated to precomputations associated with the sparse GP dynamics.

Finally, Figure 5.3 depicts a GG-diagram drawn with data from the fastest lap of each controller, along with the acceleration constraint that was active. Notably, both controllers utilize a similar amount of the allocated acceleration budget, but slightly larger constraint violations can be observed for the nominal MPC. This observation is significant as it implies that the nominal MPC may lead to increased tire wear due to its propensity for marginally exceeding the acceleration constraints.

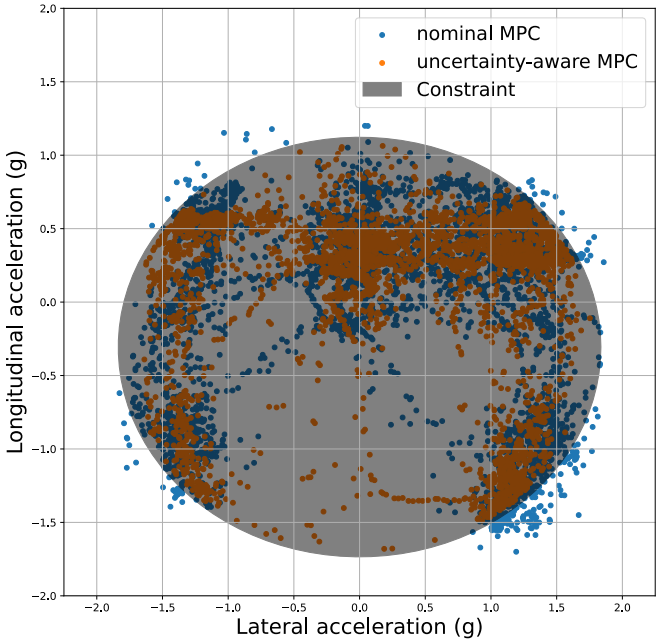

Figure 5.3. *Measured accelerations of the car during the fastest lap of each controller in ideal conditions.*

5.3 Performance in slippery track conditions

For this experiment, 10 individual laps were driven in increasingly slippery track conditions, using the same reference trajectory as in the ideal conditions experiment. The best lap from each controller's 10 attempts was selected as the experimental outcome. The results are shown in Table 5.3. Here Did Not Finish (DNF) indicates that the controller did not manage a single successful lap within the 10 attempts. The weather conditions correspond to presets available in ACC.

Table 5.3. *Best lap times in various slippery conditions: nominal MPC vs uncertainty-aware MPC*

Controller	Light rain	Medium rain	Heavy rain
MPC	DNF	DNF	DNF
UCA-MPC	1:42.07	1:45.32	DNF

The data in Table 5.3 demonstrates the enhanced adaptability of the uncertainty-aware MPC (UCA-MPC) to varying track conditions, achieving successful laps in all 10 attempts under light rain and in 7 out of 10 attempts under medium rain conditions. Most of the failures for both controllers occurred in turn 8, 11 and 14, as labeled in Figure 5.1. .

To assess the accuracy of both dynamics models in slippery track conditions, the median one-step-ahead prediction errors were again computed using the same methodology as in the ideal conditions experiment. The results are presented in Table 5.4.

Table 5.4. *Median one-step-ahead prediction errors in slippery track conditions.*

Weather preset	**Dynamics**	$\lvert a_x \rvert$ (**m/s**^{2})	$\lvert a_y \rvert$ (**m/s**^{2})	$\lvert \dot{\omega} \rvert$ (**deg/s**^{2})
Light Rain	Nominal	0.87	1.99	12.27
	Nominal+GP	0.50	0.65	4.71
Error reduction:		42%	67%	61%
Medium Rain	Nominal	0.94	2.29	13.58
	Nominal+GP	0.59	0.84	5.69
Error reduction:		37%	63%	58%

The data in Table 5.4 highlights significant reductions in the prediction error across all output dimensions. However, even with the GP-augmented dynamics model, there is a clear drop in predictive accuracy when the track conditions degrade. To complete this analysis, a density plot of the errors from the medium rain scenario is shown in Figure 5.4.

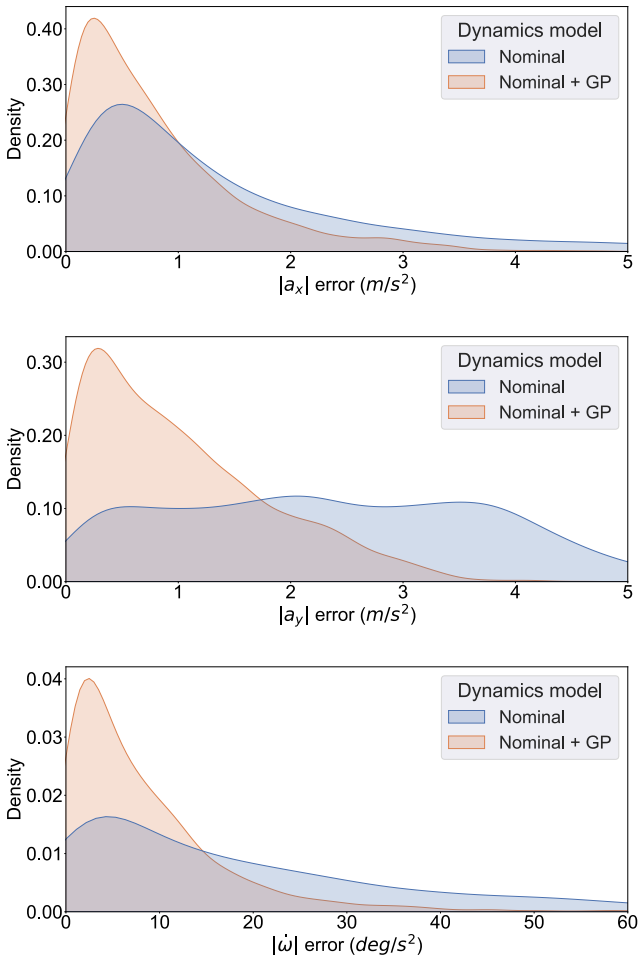

Figure 5.4. *Density plot of one-step-ahead absolute prediction errors in medium rain conditions.*

Upon visual examination of Figure 5.4, it becomes apparent that, when compared to the data under ideal conditions in Figure 5.2, improvements in the a_x error are less pronounced in contrast to the other output dimensions. This observation may be attributed to the slight downward slope of the track during turns 13 and 14. Neglecting the pitch and roll information of the car results in a constant error that cannot be fully rectified. However, it is noteworthy that this factor appears to be considered in the Gaussian Process (GP) variance predictions. Across all output dimensions and experimental conditions, a minimum of 84$\%$ of the prediction error from the GP-augmented model falls within the 95$\%$ confidence interval computed using the predicted variance. This observation suggests that, at least from a theoretical standpoint, the utilization of variance information is justified in the constraint tightening scheme.

Concluding the analysis, an exploration of the distribution of GP training data is conducted. The training data for each output dimension during the fastest lap in medium rain conditions is superimposed on the track in Figure 5.5. Notably, the concentration of training data in regions where failures occurred suggests that the selection scheme is effectively prioritizing improvements in challenging regions.

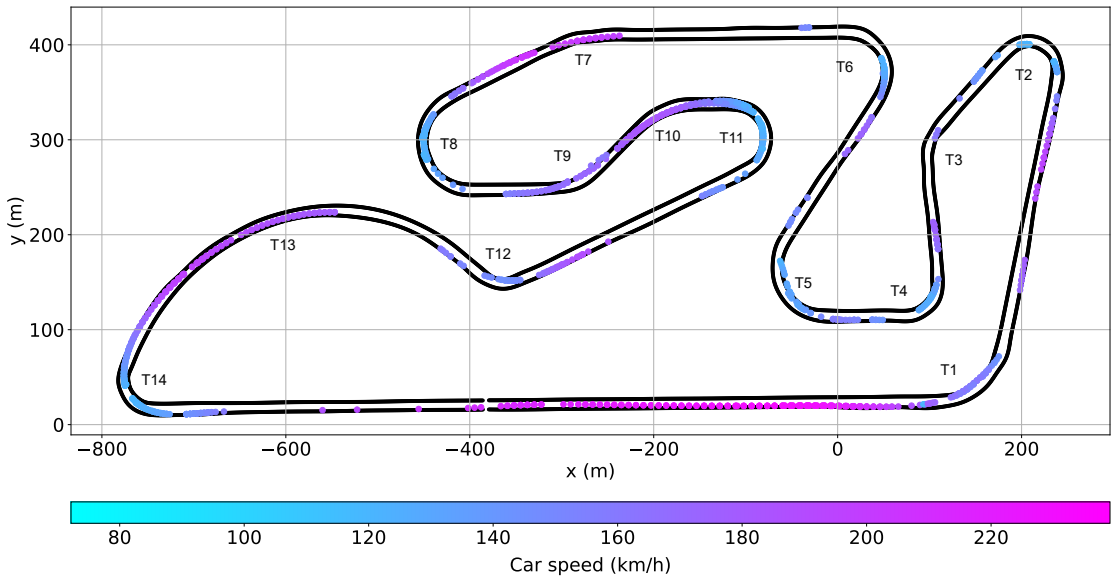

Figure 5.5. *Representation of the training data of the sparse GP overlaid on the track. The data corresponds to the fastest lap in the medium rain conditions, and includes an initial lap, used to get a running start.*

6. CONCLUSION

This thesis investigated the integration of machine learning techniques with a stochastic MPC framework to address the challenges posed by dynamics modeling errors. The comparative analysis, set within the realm of autonomous racing, indicates that the uncertainty-aware MPC approach used in this thesis significantly improves the system's resilience to environmental changes and unexpected disturbances, in comparison to a conventional MPC scheme. The results highlight that a first-principles dynamics model, although adequate under optimal conditions, lacks robustness to environmental changes. In contrast, the uncertainty-aware MPC not only improves performance in ideal conditions but also ensures safe operation in moderately varying environments, as demonstrated in increasingly slippery track conditions.

However, the adopted approach is not devoid of limitations. It imposes a considerable computational burden, doubling the optimization problem's solution time, when compared to a conventional nonlinear MPC scheme. The system's dependence on nonlinear MPC also complicates the stability analysis of the control system, which contradicts the objective of enhancing safety. Moreover, the system is incapable of instantly adapting to significant changes in the environment, as observed in the heavy rain experiments.

Future research should focus on evaluating the role of variance-based constraint tightening compared to relying solely on GP mean predictions for correcting nominal dynamics. The consideration of additional state variables for a more precise representation of environmental influences is also a valuable direction. Specifically, incorporating tire-road friction coefficient estimation into the model, potentially via camera observations, could expedite adaptation to significant environmental shifts.

REFERENCES

[1] *WHO. Road traffic injuries*. Accessed: 20/10/2023. URL: https://who.int/health-topics/road-safety.

[2] Stewart, T. *Overview of Motor Vehicle Traffic Crashes in 2021*. Tech. rep. 2023.

[3] *Nayva self driving cars*. Accessed: 20/10/2023. URL: https://navya.tech.

[4] *Cruise self driving cars*. Accessed: 20/10/2023. URL: https://getcruise.com.

[5] International, S. Taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles. *SAE international* 4970.724 (2018), pp. 1–5.

[6] Wischnewski, A., Geisslinger, M., Betz, J., Betz, T., Fent, F., Heilmeier, A., Hermansdorfer, L., Herrmann, T., Huch, S., Karle, P., Nobis, F., Ögretmen, L., Rowold, M., Sauerbeck, F., Stahl, T., Trauth, R., Lienkamp, M. and Lohmann, B. *Indy Autonomous Challenge – Autonomous Race Cars at the Handling Limits*. 2022. arXiv: 2202.03807 [cs.RO].

[7] Betz, J., Zheng, H., Liniger, A., Rosolia, U., Karle, P., Behl, M., Krovi, V. and Mangharam, R. Autonomous Vehicles on the Edge: A Survey on Autonomous Vehicle Racing. *IEEE Open Journal of Intelligent Transportation Systems* 3 (2022), pp. 458–488. ISSN: 2687-7813. DOI: 10.1109/ojits.2022.3181510.

[8] Spielberg, N. A., Brown, M. and Gerdes, J. C. Neural Network Model Predictive Motion Control Applied to Automated Driving With Unknown Friction. *IEEE Transactions on Control Systems Technology* 30.5 (2022), pp. 1934–1945.

[9] Kabzan, J., Hewing, L., Liniger, A. and Zeilinger, M. N. Learning-Based Model Predictive Control for Autonomous Racing. *IEEE Robotics and Automation Letters* 4.4 (2019), pp. 3363–3370. DOI: 10.1109/LRA.2019.2926677.

[10] A. Wischnewski M. Euler, S. G. and Lohmann, B. Tube model predictive control for an autonomous race car. *Vehicle System Dynamics* 60.9 (2022), pp. 3151–3173. DOI: 10.1080/00423114.2021.1943461.

[11] Hewing, L. Predictive Control of Uncertain Systems. Learning-based and Stochastic Model Predictive Control. en. Doctoral Thesis. Zurich: ETH Zurich, 2020.

[12] Hewing, L., Liniger, A. and Zeilinger, M. N. Cautious NMPC with Gaussian Process Dynamics for Autonomous Miniature Race Cars. *2018 European Control Conference (ECC)*. IEEE, June 2018.

[13] Fabian Christ Alexander Wischnewski, A. H. and Lohmann, B. Time-optimal trajectory planning for a race car considering variable tyre-road friction coefficients. *Vehicle System Dynamics* 59.4 (2021), pp. 588–612. DOI: 10.1080/00423114.2019.1704804.

[14] Lot, R. and Biral, F. A Curvilinear Abscissa Approach for the Lap Time Optimization of Racing Vehicles. *IFAC Proceedings Volumes* 47.3 (2014). 19th IFAC World Congress, pp. 7559–7565. ISSN: 1474-6670. DOI: https://doi.org/10.3182/20140824-6-ZA-1003.00868.

[15] Ögretmen, L., Rowold, M., Ochsenius, M. and Lohmann, B. Smooth Trajectory Planning at the Handling Limits for Oval Racing. *Actuators* 11.11 (2022). ISSN: 2076-0825. DOI: 10.3390/act11110318.

[16] Rowold, M., Ögretmen, L., Kerbl, T. and Lohmann, B. Efficient Spatiotemporal Graph Search for Local Trajectory Planning on Oval Race Tracks. *Actuators* 11.11 (2022). ISSN: 2076-0825. DOI: 10.3390/act11110319.

[17] Heilmeier, A., Wischnewski, A., Hermansdorfer, L., Betz, J., Lienkamp, M. and Lohmann, B. Minimum curvature trajectory planning and control for an autonomous race car. *Vehicle System Dynamics* 58.10 (2020), pp. 1497–1527. DOI: 10.1080/00423114.2019.1631455.

[18] Wischnewski, A., Herrmann, T., Werner, F. and Lohmann, B. A Tube-MPC Approach to Autonomous Multi-Vehicle Racing on High-Speed Ovals. *IEEE Transactions on Intelligent Vehicles* 8.1 (2023), pp. 368–378. DOI: 10.1109/TIV.2022.3169986.

[19] Betz, J., Betz, T., Fent, F., Geisslinger, M., Heilmeier, A., Hermansdorfer, L., Herrmann, T., Huch, S., Karle, P., Lienkamp, M., Lohmann, B., Nobis, F., Ögretmen, L., Rowold, M., Sauerbeck, F., Stahl, T., Trauth, R., Werner, F. and Wischnewski, A. TUM autonomous motorsport: An autonomous racing software for the Indy Autonomous Challenge. *Journal of Field Robotics* 40.4 (Jan. 2023), pp. 783–809. ISSN: 1556-4967. DOI: 10.1002/rob.22153.

[20] Jung, C., Finazzi, A., Seong, H., Lee, D., Lee, S., Kim, B., Gang, G., Han, S. and Shim, D. H. *An Autonomous System for Head-to-Head Race: Design, Implementation and Analysis; Team KAIST at the Indy Autonomous Challenge*. 2023. arXiv: 2303.09463 [cs.RO].

[21] Lam, D., Manzie, C. and Good, M. Model predictive contouring control. *49th IEEE Conference on Decision and Control (CDC)*. IEEE. 2010, pp. 6137–6142.

[22] Liniger, A., Domahidi, A. and Morari, M. Optimization-based autonomous racing of 1:43 scale RC cars. *Optimal Control Applications and Methods* 36.5 (July 2014), pp. 628–647. ISSN: 1099-1514. DOI: 10.1002/oca.2123.

[23] Kloeser, D., Schoels, T., Sartor, T., Zanelli, A., Prison, G. and Diehl, M. NMPC for Racing Using a Singularity-Free Path-Parametric Model with Obstacle Avoidance. *IFAC-PapersOnLine* 53.2 (2020). 21st IFAC World Congress, pp. 14324–14329. ISSN: 2405-8963. DOI: https://doi.org/10.1016/j.ifacol.2020.12.1376.

[24] Kabzan, J., Valls, M. I., Reijgwart, V. J., Hendrikx, H. F., Ehmke, C., Prajapat, M., Bühler, A., Gosala, N., Gupta, M., Sivanesan, R. et al. AMZ driverless: The full autonomous racing system. *Journal of Field Robotics* 37.7 (2020), pp. 1267–1294.

[25] Gillespie, T. *Fundamentals of vehicle dynamics*. SAE international, 2021.

[26] Salzmann, T., Kaufmann, E., Arrizabalaga, J., Pavone, M., Scaramuzza, D. and Ryll, M. Real-time neural MPC: Deep learning model predictive control for quadrotors and agile robotic platforms. *IEEE Robotics and Automation Letters* 8.4 (2023), pp. 2397–2404.

[27] Jain, A., O'Kelly, M., Chaudhari, P. and Morari, M. *BayesRace: Learning to race autonomously using prior experience*. 2020. arXiv: 2005.04755 [cs.RO].

[28] Wischnewski, A., Betz, J. and Lohmann, B. A Model-Free Algorithm to Safely Approach the Handling Limit of an Autonomous Racecar. *2019 IEEE International Conference on Connected Vehicles and Expo (ICCVE)*. 2019, pp. 1–6. DOI: 10.1109/ICCVE45908.2019.8965218.

[29] Rawlings, J. B., Mayne, D. Q. and Diehl, M. *Model predictive control: theory, computation, and design*. Vol. 2. Nob Hill Publishing Madison, WI, 2017.

[30] Grandia, R., Taylor, A., Singletary, A., Hutter, M. and Ames, A. Nonlinear Model Predictive Control of Robotic Systems with Control Lyapunov Functions. *Robotics: Science and Systems XVI*. RSS2020. Robotics: Science and Systems Foundation, July 2020. DOI: 10.15607/rss.2020.xvi.098.

[31] Zeng, J., Zhang, B. and Sreenath, K. Safety-critical model predictive control with discrete-time control barrier function. *2021 American Control Conference (ACC)*. IEEE. 2021, pp. 3882–3889.

[32] Cai, P., Wang, H., Huang, H., Liu, Y. and Liu, M. *Vision-Based Autonomous Car Racing Using Deep Imitative Reinforcement Learning*. 2021. arXiv: 2107.08325 [cs.RO].

[33] Wu, J., Huang, Z. and Lv, C. *Uncertainty-Aware Model-Based Reinforcement Learning with Application to Autonomous Driving*. 2021. arXiv: 2106.12194 [cs.RO].

[34] Williams, C. K. and Rasmussen, C. E. *Gaussian processes for machine learning*. Vol. 2. 3. MIT press Cambridge, MA, 2006.

[35] Snelson, E. and Ghahramani, Z. Sparse Gaussian Processes using Pseudo-inputs. *Advances in Neural Information Processing Systems*. Ed. by Y. Weiss, B. Schölkopf and J. Platt. Vol. 18. MIT Press, 2005.

[36] Boyd, S. P. and Vandenberghe, L. *Convex optimization*. Cambridge university press, 2004.

[37] Stellato, B., Banjac, G., Goulart, P., Bemporad, A. and Boyd, S. OSQP: an operator splitting solver for quadratic programs. *Mathematical Programming Computation* 12.4 (Feb. 2020), pp. 637–672. ISSN: 1867-2957. DOI: 10.1007/s12532-020-00179-2.

[38] Verschueren, R., Frison, G., Kouzoupis, D., Frey, J., Duijkeren, N. van, Zanelli, A., Novoselnik, B., Albin, T., Quirynen, R. and Diehl, M. *acados: a modular open-source framework for fast embedded optimal control*. 2020. arXiv: 1910.13753 [math.OC].

[39] Zanelli, A., Domahidi, A., Jerez, J. and Morari, M. FORCES NLP: an efficient implementation of interior-point... methods for multistage nonlinear nonconvex programs. *International Journal of Control* (2017), pp. 1–17.

[40] Khalil, H. *Nonlinear Control*. Pearson Education, 2014. ISBN: 9780133499599.

[41] Köhler, J., Zeilinger, M. N. and Grüne, L. Stability and Performance Analysis of NMPC: Detectable Stage Costs and General Terminal Costs. *IEEE Transactions on Automatic Control* 68.10 (Oct. 2023), pp. 6114–6129.

[42] Allan, D. A., Bates, C. N., Risbeck, M. J. and Rawlings, J. B. On the inherent robustness of optimal and suboptimal nonlinear MPC. *Systems & Control Letters* 106 (2017), pp. 68–78.

[43] Dai, H. and Permenter, F. *Convex synthesis and verification of control-Lyapunov and barrier functions with input constraints*. 2022. arXiv: 2210.00629 [cs.RO].

[44] Bansal, S., Chen, M., Herbert, S. and Tomlin, C. J. *Hamilton-Jacobi Reachability: A Brief Overview and Recent Advances*. 2017. arXiv: 1709.07523 [cs.SY].

[45] Choi, J. J., Lee, D., Sreenath, K., Tomlin, C. J. and Herbert, S. L. *Robust Control Barrier-Value Functions for Safety-Critical Control*. 2021. arXiv: 2104.02808 [eess.SY].

[46] Ames, A. D., Coogan, S., Egerstedt, M., Notomista, G., Sreenath, K. and Tabuada, P. *Control Barrier Functions: Theory and Applications*. 2019. arXiv: 1903.11199 [cs.SY].

[47] *assetto-mpc: Autonomous racing using MPC in Assetto Corsa Competizione*. Accessed: 15/12/2023). 2023. URL: https://github.com/trouverun/assetto-mpc.

[48] FS-Driverless. *Formula Student Driverless Simulator: An open-source simulator for autonomous driving in a Formula Student competition setting*. Accessed: 7/12/2023. 2023. URL: https://github.com/FS-Driverless/Formula-Student-Driverless-Simulator.

[49] Francis, J., Chen, B., Ganju, S., Kathpal, S., Poonganam, J., Shivani, A., Vyas, V., Genc, S., Zhukov, I., Kumskoy, M., Koul, A., Oh, J. and Nyberg, E. *Learn-to-Race Challenge 2022: Benchmarking Safe Learning and Cross-domain Generalisation in Autonomous Racing*. 2022. arXiv: 2205.02953 [cs.RO].

[50] TUMFTM. *sim_vehicle_dynamics: Vehicle Dynamics Simulation for Autonomous Driving Research*. Accessed: 7/12/2023. 2023. URL: https://github.com/TUMFTM/sim_vehicle_dynamics.

[51] Schafer, R. W. What Is a Savitzky-Golay Filter? [Lecture Notes]. *IEEE Signal Processing Magazine* 28.4 (2011), pp. 111–117.

[52] Andersson, J. A., Gillis, J., Horn, G., Rawlings, J. B. and Diehl, M. CasADi: a software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation* 11 (2019), pp. 1–36.

[53] Bakker, E., Pacejka, H. B. and Lidner, L. A new tire model with an application in vehicle dynamics studies. *SAE transactions* (1989), pp. 101–113.

[54] Quinonero-Candela, J., Girard, A. and Rasmussen, C. E. Prediction at an uncertain input for Gaussian processes and relevance vector machines-application to multiple-step ahead time-series forecasting. (2003).

[55] Frison, G. and Diehl, M. HPIPM: a high-performance quadratic programming framework for model predictive control. *IFAC-PapersOnLine* 53.2 (2020), pp. 6563–6569.

[56] Igel, C., Hansen, N. and Roth, S. Covariance matrix adaptation for multi-objective optimization. *Evolutionary computation* 15.1 (2007), pp. 1–28.

APPENDIX A: PARAMETER VALUES

The nominal vehicle dynamics model parameters used are listed in Table A.1.

Parameter	**Value**
Car mass	1000.24 kg
Car max steer	0.261 rad
Car lf	1.370 m
Car lr	1.359 m
Car inertia	2997.48 kg·m^2
Tire B_f	18.74
Tire C_f	-1.41
Tire D_f	6580.13
Tire B_r	29.97
Tire C_r	-1.06
Tire D_r	7152.61
Drivetrain K_{m0}	6531.97
Drivetrain K_{m1}	68.07
Drivetrain K_{r0}	727.00
Drivetrain K_{r1}	-0.050
Drivetrain K_{r2}	1.91

Table A.1. *Nominal vehicle dynamics model parameters*